Улучшение оценки распределенного стохастического градиентного спуска через аппроксимацию функции потерь[1]

А. А. Сенов, аспирант
Санкт-Петербургский государственный университет
alexander.senov@gmail.com

Проблемы обучения и оптимизации в контексте больших объемов данных в настоящее время играют важную роль. К сожалению, даже методы параллельной и онлайн оптимизации не справляются с обработкой большие объемов данных из-за временных и технических ограничений. В этом случае распределенные методы оптимизации могут быть единственным решением проблемы. В статье рассмотрены некоторые современные методы распределенной оптимизации и исследован частный случай задачи оптимизации с разделимой стохастической квадратичной функцией потерь, предложены два новых алгоритма распределенной оптимизации, развивающие работы других авторов. Преимущества алгоритмов продемонстрированы экспериментально.

Ключевые слова: стохастическая оптимизация, оценка параметров, параллельные вычисления, распределенные вычисления, распределенная оптимизация, взвешенный метод наименьших квадратов.

1. Введение

Такие математические дисциплины как машинное обучение и оптимизация становятся популярными и широко используемыми в современных приложениях теории управления [1]. Хорошо известно, что большая часть задач машинного обучения может быть сведена к задаче оптимизации вещественнозначной выпуклой параметризованной функции $\mathcal{L}(\omega)$, называемой *функцией потерь*, *штрафной функцией*, или *эмпирическим функционалом среднего риска* (см., например, [2, 3], главу 4 в [4]). Более того в большинстве случаев функция $\mathcal{L}$ является *разделимой* [5]:

$$\mathcal{L}(\omega) = \sum_{i=1}^{N} \ell_i(\omega).$$

Т. е., оптимизация функций такого вида играет важную роль в машинном обучении.

[1] ©А. А. Сенов, 2015

Задача оптимизации выпуклых функций хорошо изучена (например, см. [6,7,8] для полного обзора). Предполагая выпуклость и дифференцируемость функции можно использовать широкий спектр итеративных методов оптимизации первого порядка. Один из наиболее простых и распространенных примеров: *градиентный спуск*, предложенный в [9]. Основная идея итеративных методов первого порядка заключается в последовательном улучшении оценки argmin функции $\mathcal{L}$ за счет движения в направлении противоположном направлению градиента. Такие алгоритмы вычисляют градиент функции на каждой итерации. Общее число вычислений градиента функций ℓ_i составляет как минимум $\mathcal{O}(NT)$, где T — число итераций. Учитывая то, что T может исчисляться сотнями и тысячами, подобные методы непозволительно дороги в случае больших значений N. Использование *рандомизированных* алгоритмов оптимизации [3,4] может уменьшить трудоемкость вычислений градиентов функций ℓ_i, но не уменьшает асимптотику оценки $\mathcal{O}(NT)$.

Так называемые *онлайн* методы оптимизации завоевали большую популярность в задачах, где значение N велико (например, 10^9). Основная идея онлайн алгоритмов заключается в использовании лишь вычисления фиксированного числа B ($<< N$) функций ℓ_i на каждой итерации, уменьшая общее число вычислений градиента до $\mathcal{O}(BT)$ (стоит отметить, что в этом случае для достижения той же точности онлайн методам обычно требуется больше итераций T, чем не-онлайн методам оптимизации). Один из наиболее популярных онлайн методов оптимизации — *стохастический градиентный спуск*, предложенный в [10], и в дальнейшем развитый в [11] в особенности для задач машинного обучения.

Однако, даже онлайн алгоритмы оптимизации могут потребовать недопустимо много времени для достижения заданной точности. Для ускорения можно использовать *параллельные* вычисления и соответствующие методы *параллельной оптимизации*. Подобные алгоритмы эффективно используют общую память, связность и быструю передачу сообщений между процессами для интенсивной коммуникации между ними, позволяющей значительно ускорить процесс оптимизации [5,12].

Несмотря на то, что использование параллельных алгоритмов позволяет значительно ускорить процесс оптимизации на одном *вычислительном узле* (ЭВМ), даже они могут не справится с ситуаци-

ями, когда N велико и/или вычисление функций ℓ_i требует значительных временных затрат (в особенности, когда все функции $\{\ell_i\}_1^N$ не помещаются на жестком диске одного узла). Так как технические характеристики ЭВМ ограничены финансовыми и конструктивными соображениями, появляется необходимость в качественно ином подходе. Таким подходом являются *распределенные* вычисления, оперирующие на множестве вычислительных узлов, соединенных в общую сеть, образуя *вычислительный кластер*. Вычислительные характеристики кластера, в отличие от одного вычислительного узла, могут наращиваться практически неограниченно за счет увеличения числа узлов. Основным отличием распределенных и параллельных вычислений является то, что в случае распределенных вычислений взаимодействие между узлами значительно ограничено пропускной способностью сети. Таким образом, параллельные методы оптимизации не могут быть напрямую перенесены на модель распределенных вычислений из-за жестких ограничений на объем и скорость коммуникаций между узлами, и появляется потребность в новых *распределенных* методах оптимизации. Так, в работе [13] авторы описывают распределенные реализации нескольких алгоритмов машинного обучения (и, неявно, методов оптимизации первого-порядка) в модели распределенных вычислений MapReduce [15]. В [16] авторы решают задачу распределенной оптимизации посредством простой, но эффективной стратегии: независимом выполнении оптимизационных процессов на каждом из узлов и последующем усреднении полученных оценок $\operatorname{argmin} \mathcal{L}$. Этот подход был развит в работе [14], где авторы использовали стохастический градиентный спуск на каждом узле, используя лишь подмножество функций $\{\ell_i\}_1^N$, и провели тщательный анализ сходимости как классического стохастического градиентного спуска так и предложенной распределенной версии с помощью техники основанной на *сжимающих отображениях на распределениях*.

Работы [13, 14, 16] используют схожую идею распределенной оптимизации: выполнении независимых оптимизационных процессов на каждом из узлов и последующем усреднении полученных оценок. В этой работе шаг усреднения заменяется на более сложный и эффективный шаг *восстановления* функции потерь $\mathcal{L}$ и использования использования argmin восстановленной функции как окончательной оценки. Стоит отметить, что в ограничиваемся выпуклыми функциями ℓ_i.

Статья организована следующим образом. В разделе 2 приводятся необходимые сведения из теории оптимизации и оценивания, а так же важные предшествующие результаты. Затем, в разделе 3 формулируется постановка частной задачи распределенной оптимизации. В 4 предлагается алгоритм решения поставленной проблемы и в 5 он иллюстрируется примерами и проводится сравнительный анализ с методом, предложенным в [14].

2. Необходимые сведения из оптимизации и оценивания

В этом разделе будут рассмотрены методы и понятия, необходимые для изложения последующего материала.

2.1. Обозначения

Прописные символы x используются для обозначения скаляров и индексов, прописные жирные символы $\mathbf{x}$ для обозначения векторов, заглавные символы X для обозначения констант и множеств, заглавные жирные символы X для обозначения матриц. В частности, K используются для обозначения числа узлов, T — числа итераций, λ — множителя шага оптимизации, а B — размера пакета в пакетном градиентном спуске.

2.2. Разделимая стохастическая выпуклая функция

В работе рассматривается задача оптимизации лишь функций определенного вида: разделимых стохастичных выпуклых вещественнозначных функции (обозначается $\mathcal{L}$) параметризованной $\omega \in \mathbb{R}^d$. Называя функцию $\mathcal{L}$ *разделимой* имеется в виду, что она представима в виде

$$\mathcal{L}(\omega) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(\omega), \tag{1}$$

где функции ℓ_i значительно проще в вычислении, чем $\mathcal{L}$. Под *стохастичностью* понимается то, что каждая ℓ_i имеют следующий вид

$$\ell_i(\omega) = \ell(\omega) + \varepsilon_i^{\ell}(\omega), \tag{2}$$

где $\ell(\omega)$ некоторая детерминированная функция, а $\varepsilon_i^\ell(\omega)$ — случайная величина со средним ноль и дисперсией, зависящей от ω.

Выбор подобного вида функции кажется не очевидным, но может быть легко объяснен. Рассмотрим задачу линейной регрессии (смотри раздел 2.5) с векторами входа $\mathbf{x}_i$, скалярами выхода y_i и параметром ω. Тогда функция потерь на каждом наблюдении будет выглядеть следующим образом

$$\ell_i^{lr} = (y_i - \mathbf{x}_i^T\omega)^2 = \omega^T\left(\mathbf{x}_i\mathbf{x}_i^T\right)\omega - 2y_i\mathbf{x}_i^T\omega + y_i^2 = \dot{\omega}^T\mathbf{Z}_i\dot{\omega},$$

— частный вид функции ℓ_i. В действительности, функции потерь для многих задач из области восстановления образов (машинного обучения с учителем) удовлетворяют предположениям, наложенным на функцию $\mathcal{L}$.

2.3. Стохастический градиентный спуск

В последние годы метода стохастического градиентного спуска стал очень популярным методом оптимизации, в особенности в машинном обучении и управлении по причине его простоты и эффективности. Он был предложен в работе [10] для решения задачи линейной регрессии и в последовательности развит, в частности для задач онлайн обучения [2]. Метод стохастического градиентного спуска нацелен на решения задачи оптимизации стохастических разделимых функций следующим образом. Предполагая выпуклость ℓ_i и верность $\min\limits_\omega \ell = \mathrm{E}\min\limits_\omega \ell_i$, на каждой итерации ℓ_j выбирается из $\{\ell_i\}_1^N$ случайным образом и текущая оценка параметра ($\arg\min\mathcal{L}$) сдвигается против направления градиента $\partial_\omega\ell_j$. Псевдокод метода приведен в 1.

Алгоритм 1 $\mathrm{SGD}(\{\ell_1,\ldots,\ell_N\}, T, \lambda, \omega_0)$.

for all $t \leftarrow 1$ to T **do**
  Draw $j \in \{1,\ldots,N\}$ uniformly at random
  $\omega_t \leftarrow \omega_{t-1} - \lambda\,\partial_\omega\ell_j(\omega_{t-1})$

return ω_T

Метод *пакетного стохастического градиентного спуска* использует аналогичную стратегию, но на каждой итерации выбирается

не одна а подмножество функций $\{\ell_i\}_1^N$ размером B и вычисляется усредненное по ним направление градиента. Псевдокод этого метода приведен в 2.

Алгоритм 2 mbSGD($\{\ell_1, \ldots, \ell_N\}, T, \lambda, \omega_0, B$).

for all $t \leftarrow 1$ to T **do**
  Draw $J \subset \{1, \ldots, N\}$ of size $|J| = B$ uniformly at random
  $\omega_t \leftarrow \omega_{t-1} - \lambda \frac{1}{B} \sum_{j \in J} \partial_\omega \ell_j(\omega_{t-1})$

return ω_T

2.4. Распределенный стохастический градиентный спуск

Приведем измененную версию SimuParallel mbSGD алгоритма SimuParallel SGD (с заменой используемого в оригинальной версии SGD на mbSGD), предложенного в [17], на котором в значительной степени основывается эта работа. Псевдокод алгоритма приведен в 3. Стоит отметить, что в [14] авторы предполагают, что размер выборки настолько велик, что N может быть выбрано практически любым, исходя из необходимого числа итераций T, размера пакета B и числа машин K. Это значит, что N может быть задано функцией от B, T и K из соотношения $T = \lfloor N/B/K \rfloor$. То есть, в качестве параметров алгоритма можно исключить N, задавая его через параметры B, K, и T. Подобное предположение действует здесь и в дальнейшем.

2.5. Взвешенный метод наименьших квадратов

Метод наименьших квадратов (МНК) используется для оценки параметров в следующей линейной модели входа-выхода, называемой моделью *линейной регрессии*

$$y_i = \omega_i^T \theta + \varepsilon_i, \qquad i = 1..N \tag{3}$$

где $\omega_i \in \mathbb{R}^d$ — вход модели, $y_i \in \mathbb{R}$ — выход модели, $\varepsilon_i \in \mathbb{R}$ — неизвестная помеха, а $\theta \in \mathbb{R}^d$ — неизвестный и искомый параметр.

Алгоритм 3 SimuParallel mbSGD($\{\ell_1, \ldots, \ell_N\}, \lambda, B, K, T$).

Ensure that $T = \lfloor N/K/B \rfloor$
Randomly partition the examples on K samples I_k
of size $|I_k| = T$
for all $k \leftarrow 1$ to K **in parallel do**
  Randomly shuffle sample I_k
  Randomly pick $\omega_{k,0}$ from ℓ domain
  for all $t \leftarrow 1$ to T **do**
    Draw $J \subset I_k$ of size $|J| = B$ uniformly at random
    $\overline{\ell}(\omega_{k,t-1}) \leftarrow \frac{1}{B} \sum_{j \in J} \ell_j(\omega_{k,t-1})$
    $\omega_{k,t} \leftarrow \omega_{k,t-1} - \lambda \partial_\omega \overline{\ell}(\omega_{k,t-1})$
$\hat{\omega} \leftarrow \frac{1}{K} \sum\limits_{k=1}^{K} \omega_{k,T}$

return $\hat{\omega}$

Здесь и далее свободный член опускается в предположении, что он учтен в ω_i: $\omega_i = (1, \omega_i^{(2)}, \ldots, \omega_i^{(d)})^T$.

Обозначим $\Omega = [\omega_1, \ldots, \omega_N]^T$ and $\mathbf{y} = (y_1, \ldots, y_N)^T$. Тогда, в предположении $\mathrm{E}\,\varepsilon_i = 0$, $\mathrm{Var}\,\varepsilon_i = \sigma^2$, $\mathrm{E}\,\varepsilon_i\varepsilon_j = 0\ \forall i \neq j$ оценка параметра θ методом наименьших квадратов равна

$$\hat{\theta}_{LS} = (\Omega^T \Omega)^{-1} \Omega \mathbf{y} \tag{4}$$

и она имеет наименьшую возможную дисперсию среди все линейных несмещенных оценок. Как бы то ни было, если предположение $\mathrm{Var}\,\varepsilon_i = \sigma^2$ (т.н. *гомоскедастичность*) не выполняется, то оценка МНК теряет это свойство. В частности, это происходит в случае $\mathrm{Var}\,\varepsilon_i = \sigma_i^2$ (*гетероскедастичность*) — ошибки имеют различную дисперсию. Для улучшения МНК оценки для случая гетероскедастичности используется *взвешенный метод наименьших квадратов*. Взвешенный метод наименьших квадратов (ВМНК) исправляет гетероскидастичность [18] используя специально подобранную матрицу весов $\mathbf{W} = diag(\mathrm{w}_1, \ldots, \mathrm{w}_N)$ где $\mathrm{w}_i \sim 1/\sigma_i^2$, что приводит к следующей оценке

$$\hat{\theta}_{WLS} = (\Omega^T \mathbf{W} \Omega)^{-1} \Omega \mathbf{y} \tag{5}$$

Легко увидеть, что МНК эквивалентен ВМНК с той лишь разницей, что в последнем наблюдения y_i, ω_i умножаются на соответствую-

щий вес w_i. За счет этого умножения оригинальная модель (3) изменяется на

$$\mathrm{w}_i y_i = \mathrm{w}_i \omega_i^T \theta + \mathrm{w}_i \varepsilon_i, \qquad i = 1..N \tag{6}$$

и так как $\operatorname{Var} \mathrm{w}_i \varepsilon_i = const \ \forall i$, то новая модель является гомоскедастичной.

2.6. МНК для квадратичной модели

В разделе 2.5описан (взвешенный) метод наименьших квадратов в приложении к линейной модели. Рассмотрим квадратичную модель

$$y_i = \omega_i^T \Theta \, \omega_i + \varepsilon_i, \qquad i = 1..N \tag{7}$$

где $\Theta = (\theta_{i,j})_{i,j} \in \mathbb{R}^{d \times d}$ — симметричная матрица параметров. *Стоит отметить, что линейные члены не включены в уравнения явным образом — они неявно учитываются за счет того, что первый элемент вектора ω_i равен 1.* Несмотря на то, что уравнение (7) является полиномом второй степени по вектору входов ω_i оно все так же линейно по параметрам $\theta_{i,j}$. Следовательно, ВМНК может быть использован для оценки матрицы параметров Θ после некоторых переобозначений

2.7. Область компетенции статьи

Работа фокусируется на проблеме улучшения оценки $\hat{\omega}$ на основе оценок полученных на каждом из узлов $\{\hat{\omega}_k\}_{k=1}^K$ и некоторой дополнительной информации в метода распределенного (пакетного) градиентного спуска. Все прочие аспекты задачи оптимизации упускаются из рассмотрения (например, выбор оптимального размера шага λ, размера пакета B, числа узлов K и т.п.). Несмотря на то, что некоторые из аспектов неявно затрагиваются в ходе экспериментов в разделе 5, за информацией по этим и другим аспектам оптимизации читателю стоит обратится к другим работам.

3. Постановка задачи

Постановка задачи состоит из предположений и непосредственно оптимизационной задачи. Следующие теоретические предполо-

жения описывают функции $\mathcal{L}$ и ℓ_i.

Предположение 1.

- *Функция потерь $\mathcal{L}$ разделима* (1)*:* $\mathcal{L} = \sum_{i=1}^{N} \ell_i$
- *Функции ℓ_i — стохастические, т.е.*

$$\ell_i(\omega) = \ell(\omega) + \varepsilon_i(\omega),$$

  где $\varepsilon_i(\omega)$ — независимо одинаково распределенные помехи, зависящие от ω со средним равным нулю ($\mathrm{E}\,\varepsilon_i(\omega) = 0\ \forall i = 1..N$), и ограниченным вторым моментом, зависящем от ω.

- *Функция ℓ выпукла по ω и может быть приближена полиномом второй степени в окрестности своего минимума*

$$\ell(\omega) = \dot{\omega}^T \Theta \tag{8}$$

  здесь $\dot{\omega}_i = (1, \omega_i^T)^T$ — вектор полученный добавлением 1 к ω_i

Исходя из объемов данных, временных, программных и аппаратных ограничений к теоретическим предположениям добавляются следующие вычислительные ограничения

Предположение 2.

- *Алгоритм должен быть распределенным: параллельно выполнятся на K вычислительных узлах*
- *Алгоритм должен быть онлайн по отношению к функциям ℓ_i: на каждой итерации алгоритма должно использоваться $\mathcal{O}(1)$ (относительно N) функций ℓ_i*
- *Алгоритм должен содержать лишь один шаг коммуникации между K вычислительными узлами в конце вычислений*

В дальнейшем рассматриваться следующая оптимизационная задача задача.

Постановка задачи.

Построить алгоритм, определяющий минимум функции $\mathcal{L}$ (1) *в теоретических 1 и вычислительных 2 предположениях.*

4. Описание алгоритмов LSDSGD и WLSDSGD

В этом разделе будут предложены два алгоритма, нацеленные на решение поставленной задачи 3.

Приведенный выше алгоритм SimuParallel mbSGD решает поставленную задачу и удовлетворяет сделанным ограничениям. Однако, последний его шаг — усреднение оценок параметра с каждого вычислительного узла может быть значительно улучшен в смысле полученной точности. В последующих разделах 4.2и 4.3будут предложены два алгоритма: LSDSGD и WLSDSGD, основанные на восстановлении функции потерь ℓ.

4.1. Основная идея алгоритмов LSDSGD и WLSDSGD

Алгоритм SimuParallel mbSGD вкратце можно описать следующим образом

1. Разбиение исходных данных на K частей и их распределение на K узлов
2. Применение mbSGD на каждом из K узлов к соответствующей части исходных данных
3. Передача полученной оценки точки минимума функции с каждого из K узлов на терминальный узел
4. Усреднение оценок с каждого и K узлов для получения финальной оценки

Детально описываемые далее алгоритмы LSDSGD и WLSDSGD направлены на замену шагов 3–4 на более сложные, но эффективные:

- К шагу 3 добавляется
  - дополнительная сбор и передача оценок значений функции (и возможно, ошибки ее измерения — оценки ее дисперсии в точке) в каждой из точек ω_k, t, полученных на каждой итерации mbSGD каждого узла на терминальный узел.

- Шаг 4 заменяется на
  - аппроксимацию ℓ квадратичной по ω функцией $\widehat{\ell}$ на основе дополнительно полученной с каждого узла информацией: значений ω_k, t и оценок значения и дисперсии ℓ в этих точках;
  - выбор точки минимума $\widehat{\ell}$ как итоговой оценки минимума ℓ .

Основную сложность здесь составляет задача построения аппроксимирующей функции $\widehat{\ell}$ по полученным данным. Из-за того, что дисперсия $\varepsilon_i^{\ell}(\omega)$ — отклонения $\ell(\omega)$ от $\ell_i(\omega)$ — зависит от ω стандартные методы решения регрессионных задач (к которым относится задача восстановления квадратичного полинома $\widehat{\ell}$ по собранным данным), например МНК, не подходят. Для этого необходимо применять методы, направленные на борьбу с гетероскедастичностью (неоднородностью дисперсии), например ВМНК. Именно подобный подход (борьбу с гетероскедастичностью за счет применения ВМНК) и использует метод WLSDSGD, в то время как LSDSGD использует стандартный МНК в предположении, что гетероскедастичность не окажет критического влияния на точность аппроксимации функции ℓ.

Для наглядности, общая структура работы алгоритма WLSDSGD изображена на схеме 1.

Замечание **1.** *Из-за того, что функция ℓ в общем случае не является квадратичной, то дисперсия разницы между $\ell(\omega_0)$ и $\widehat{\ell}(\omega_0)$ в произвольной точке ω_0 становится непостоянной, увеличиваясь по мере отдаления ω_0 от ω^*. Более, того разница между ними в общем случае не равна нулю, т.е. является смещенной. Если проблему непостоянства дисперсии — гетероскедастичность — решается применением соответствующих методов (например, ВМНК, как в WLSDSGD), то проблема смещенности является более сложной.*

Один из вариантов является решения проблемы смещенности $\widehat{\ell}$ относительно ℓ — усложнение модели $\widehat{loss}$: например, использования полинома не второй, а более высокой степени. К сожалению подобный подход значительно увеличивает вычислительную сложность и потребность в данных.

Альтернативный вариант заключается в том, чтобы "перекинуть" смещенность — разницу между $\ell(\omega_0)$ и $\widehat{\ell}(\omega_0)$ — в стохастическую функцию $\varepsilon_i^\ell(\omega_0)$, тем самым сделав ее в среднем не равной нулю: $\varepsilon_i^\ell(\omega 0) \neq 0$. Методы, направленные на решения подобных задач оценки параметров при произвольных внешних помехах, активно развиваются в последние годы: смотри главу 4 в [3], а также [19] и [20]. Однако, подобная проблема в этой работе не рассматривается, являясь направлением будущих исследований.

4.2. Алгоритм LSDSGD

Псевдокод первого алгоритма, для удобства обозначенного как *LSDSGD* (L_{east} $S_{quares\ loss\ reconstruction\ based}$ $D_{istributed\ online\ mini-batch}$ $S_{tochastic}$ $G_{radient}$ D_{escent}), приведен в 4.

Основная идея алгоритма и в тот же момент основное его отличие от алгоритма SimuParallel mbSGD заключается в следующем

- в дополнение к оценке параметра на последней итерации $\omega_{k,T}$ на каждому узле k вычисляется оценка среднего функции потерь $\overline{\ell}(\omega_{k,t-1})$ на каждой итерации t в точке $\omega_{k,t-1}$ (оценка параметра на итерации t в узле k)
- на последнем шаге вместо простого усреднения оценки параметра $\omega_{k,T}$, $k = 1$ используется следующая процедура:
  1. аппроксимация функции потерь ℓ квадратичным полиномом на основе собранных с каждого узла данных ($S_{\overline{\ell}}^{K,T}$ и $S_{\omega}^{K,T}$) посредством метода наименьших квадратов
  2. выбор минимума восстановленной функции (полинома второй степени) в качестве итоговой оценки параметра

Распишем эти пункты более подробно. После параллельного и независимого выполнения процессов оптимизации на K вычислительных узлах оценки параметра и среднего функции потерь с каждого узла $k = 1..K$ и каждой итерации $t = 1..T$ собираются на последнем узле в множества $S_{\omega}^{K,T}$ и $S_{\overline{\ell}}^{K,T}$ соответственно. Отметим, что

$$\overline{\ell}(\omega) = \dot{\omega}^T \overline{\Theta}\, \dot{\omega},$$

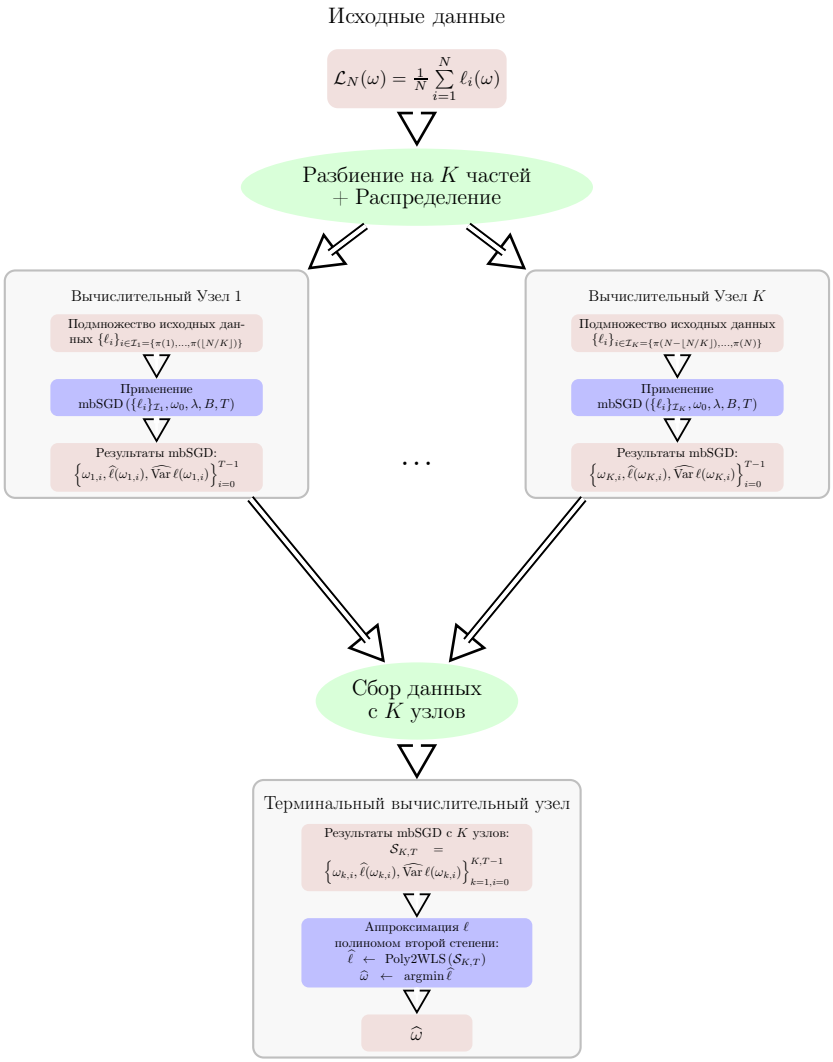

Рис. 1: Схема алгоритма WLSDSGD.

где $\overline{\Theta} = \frac{1}{B}\sum_{i\in J}\Theta_i$. $\overline{\ell}$ квадратичная по ω и линейная по $\overline{\Theta}$, а следовательно метод наименьших квадратов 4 может быть использован для получения $\widetilde{\Theta}_{LS}$ — оценки матрицы параметров $\overline{\Theta}$ (именно это и делает функция $Poly2LS$ на строке 14 алгоритма 4). Получив оценку $\overline{\Theta}$, можно вычислить argmin соответствующей $\widetilde{\ell}_{LS}$

$$\hat{\omega} = \operatorname*{argmin}_{\omega} \widetilde{\ell}_{LS} = \operatorname*{argmin}_{\omega} \dot{\omega}^T \widetilde{\Theta}_{LS}\, \dot{\omega} \tag{9}$$

(строка 15 алгоритма 4 соответственно).

4.3. Алгоритм WLSDSGD

Согласно предположениям 1, функции ℓ_i могут иметь различную дисперсию. А значит, и дисперсии различных $\overline{\ell}(\omega)$ могут отличаться — следовательно, имеет место гетероскедастичность. Как отмечено а разделе 2.5, оценка МНК в случается гетероскедастичности остается несмещенной, но теряет свою эффективность. В задаче оценки параметров линейной модели для нивелирования эффекта гетероскидастичности используется взвешенный метод наименьших квадратов (ВМНК). Аналогичным образом, для нивелирования гетероскедастичности $\overline{\ell}(\omega)$ в задаче 3 предлагается алгоритм WLSDSGD (Weighted LSDSGD), псевдокод которого приведен в листинге 5). Алгоритм WLSDSGD является модификацией LSDSGD со следующим основным отличием: в последнем шаге оценки параметра для аппроксимация функции ℓ квадратичным полиномом происходит посредством ВМНК, а не МНК. Как следует из описания ВМНК 4.2, для его использования необходимо обладать оценками дисперсий ошибок (весами). В нашем случае, веса — это оценки дисперсии $\operatorname{Var}\ell(\omega_{k,t})$, которые дополнительно вычисляются на каждой итерации и на каждом узле. Эти оценки собираются в набор $S^{K,T}_{\operatorname{Var}\overline{\ell}}$, который вместе с $S^{K,T}_{\omega}$ и $S^{K,T}_{\overline{\ell}}$ используется для получения $\widetilde{\Theta}_{WLS}$ — оценки $\overline{\Theta}$ взвешенным МНК (строка 16 алгоритма 5).

5. Моделирование

В данном разделе исследуется зависимость точность определения параметра ω^* алгоритмами LSDSGD, WLSDSGD и алгоритмом

Алгоритм 4 LSDSGD($\{\ell_1, \ldots, \ell_N\}, \lambda, B, K, T$).

Ensure that $T = \lfloor N/K/B \rfloor$
Randomly partition the examples on K samples I_k
of size $|I_k| = TB$
for all $k \leftarrow 1$ to K **in parallel do**
  Randomly shuffle sample I_k
  Randomly pick $\omega_{k,0}$ from ℓ domain
  for all $t \leftarrow 1$ to T **do**
    Draw $J \subset I_k$ of size $|J| = B$ uniformly at random
    $\overline{\ell}(\omega_{k,t-1}) \leftarrow \frac{1}{B}\sum_{j \in J} \ell_j(\omega_{k,t-1})$
    $\omega_{k,t} \leftarrow \omega_{k,t-1} - \lambda \partial_\omega \overline{\ell}(\omega_{k,t-1})$

$S_{\overline{\ell}}^{K,T} \leftarrow \{\overline{\ell}(\omega_{k,t})\}_{k=1,t=0}^{K,T-1}$
$S_{\omega}^{K,T} \leftarrow \{\omega_{k,t}\}_{k=1,t=0}^{K,T-1}$
$\hat{\ell} \leftarrow \text{Poly2LS}\left(S_{\ell}^{K,T}, S_{\omega}^{K,T}\right)$
$\hat{\omega} \leftarrow \underset{\omega}{\operatorname{argmin}}\, \hat{\ell}$

return $\hat{\omega}$

SimuParallel mbSGD. Далее рассмотрим стратегию моделирования и способ вычисления точности оценки ω^*.

Стратегия моделирования

1. Фиксируется определенный вид функции ℓ_i
2. Задаются значения параметров алгоритмов LSDSGD, WLSDSGD и SimuParallel mbSGD по умолчанию: значения B, K, T и стратегия выбора $\omega_{k,0}$
3. По очереди варьируется один из параметров выше, вычисляя точность оценки ω^* каждым из трех алгоритмов. Таким образом, демонстрируется зависимость точность оценки ω^* от значений параметра для каждого из трех алгоритмов.
4. Для каждого значения варьируемого параметра (и стандартными значениями для остальных параметров) каждый алгоритм запускается 1000 раз, каждый раз вычисляя ошибку оценки ω^* (расстояние Евклида: $\hat{\omega} - \omega^*$, где $\hat{\omega}$ — оценка алгоритма). Далее, вычисляется 0.25, 0.5 и 0.75 квантили по-

Алгоритм 5 WLSDSGD($\{\ell_1, \ldots, \ell_N\}, \lambda, B, K, T$).

Ensure that $T = \lfloor N/K/B \rfloor$
Randomly partition the examples on K samples I_k
of size $|I_k| = TB$
for all $k \leftarrow 1$ to K **in parallel do**
  Randomly shuffle sample I_k
  Randomly pick $\omega_{k,0}$ from ℓ domain
  for all $t \leftarrow 1$ to T **do**
    Draw $J \subset I_k$ of size $|J| = B$ uniformly at random
    $\overline{\ell}(\omega_{k,t-1}) \leftarrow \frac{1}{B} \sum_{j \in J} \ell_j(\omega_{k,t-1})$
    $\widehat{\mathrm{Var}}\left[\ell(\omega_{k,t-1})\right] \leftarrow \frac{1}{B} \sum_{j \in J} \left(\ell(\omega_{k,t-1} - \overline{\ell}(\omega_{k,t-1}\right)^2$
    $\omega_{k,t} \leftarrow \omega_{k,t-1} - \lambda \partial_\omega \overline{\ell}(\omega_{k,t-1})$

$S_{\overline{\ell}}^{K,T} \leftarrow \left\{\overline{\ell}(\omega_{k,t})\right\}_{k=1,t=0}^{K,T-1}$
$S_{\omega}^{K,T} \leftarrow \left\{\omega_{k,t}\right\}_{k=1,t=0}^{K,T-1}$
$S_{\mathrm{Var}[\ell]}^{K,T} \left\{\widehat{\mathrm{Var}}\left[\ell(\omega_{k,t-1})\right]\right\}_{k=1,t=0}^{K,T-1}$
$\hat{\ell} \leftarrow \mathrm{Poly2WLS}\left(S_{\ell}^{K,T}, S_{\omega}^{K,T}, S_{\mathrm{Var}[\ell]}^{K,T}\right)$
$\hat{\omega} \leftarrow \underset{\omega}{\operatorname{argmin}}\, \hat{\ell}$

return $\hat{\omega}$

лученных ошибок и строится график зависимости квантилей каждого из трех алгоритмов от варьируемого параметра.

Используя подобную стратегию можно продемонстрировать как индивидуальные зависимости каждого из алгоритмов от значения параметров, так и сравнить их друг с другом. Далее, рассмотрим каждый из пунктов стратегии в подробностях.

5.1. Моделирование функции ℓ_i

Для моделирования используются функции ℓ_i следующего виды (можно заметить, что эти функции соответствуют задаче двумерной линейной регрессии):

$$\begin{aligned} \ell_i(\omega) &= (y_i - \mathbf{x}_i^T\omega)^2 = \omega^T\mathbf{x}_i\mathbf{x}_i^T\omega - 2y_i\mathbf{x}_i^T\omega, \\ \mathbf{x}_i &\sim N(m_{\mathbf{x}}, \Sigma_{\mathbf{x}}), \\ y_i &= \mathbf{x}_i\omega^* + \varepsilon_i, \qquad \mathrm{E}\,\varepsilon_i = 0, \mathrm{Var}\,\varepsilon_i = \sigma_\varepsilon^2, \end{aligned} \tag{10}$$

где $\mathbf{x}_i$ — независимые переменный, y_i — зависимые переменные и ε_i — помехи, т.ч. $\mathrm{E}\,\varepsilon_i\varepsilon_j = \sigma_\varepsilon^2\delta_{ij}$ and $\mathrm{E}\,\mathbf{x}_i\varepsilon_j = 0\ \forall\, i, j$. В качестве $m_{\mathbf{x}}$, $\Sigma_{\mathbf{x}}$ и σ_ε^2 используются следующие значения

$$m_{\mathbf{x}} = (2, -1)^T, \quad \Sigma_{\mathbf{x}} = \left[(2, 0.5)^T, (0.5, 1)^T\right], \\ \omega^* = (1, 0.5)^T,$$

Очевидно, что подобные функции ℓ_i удовлетворяют предположениям 1. Более того, $\omega^* = \underset{\omega}{\operatorname{argmin}}\,\ell$ — Искомое значение параметра, которое нам необходимо найти.

5.2. Значения параметров по умолчанию

Используются следующие значения параметров λ, B, K and T по умолчанию.

$$\lambda = 0.01, \quad B = 10, \quad K = 4, \quad T = 50 \tag{11}$$

Стратегия выбора $\omega_{k,0}$ следующая. Напомним, что истинное значения параметра $\omega^* = (1, -2)^T$. Для выбора $\omega_{k,0}$ используется равномерное распределение в квадрате 5×5 с центром в $\omega_{sq} = (1, -2)^T$.

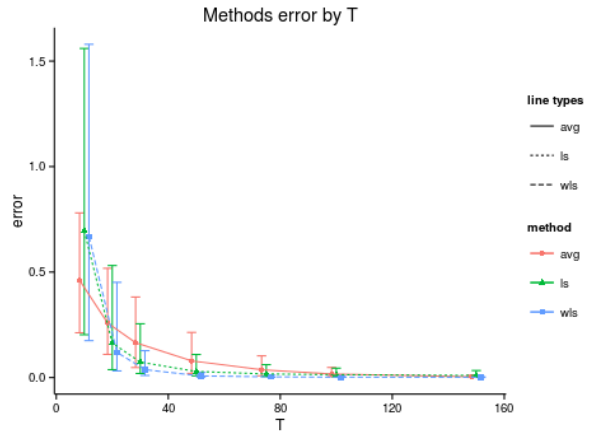

Рис. 2: Квантили ошибок оценки точки минимума функции в зависимости от количества итераций T.

Другими словами, $\omega_{k,0}^{(j)}$ выбирается из отрезка $[\omega_{sq}^{(j)} - 2.5, \omega_{sq}^{(j)} + 2.5]$ с равной вероятностью $\forall\, k = 1..K$, $j = 1..2$.

5.3. Варьирование параметров

Значения параметров B, K, T и стратегия выбора $\omega_{k,0}$ варьируются независимо по следующим начениям

$$\begin{aligned} B &= \{2, 4, 8, 16, 32, 64, 128\}, \\ K &= \{2, 3, 4, 5, 8, 16, 32\}, \\ T &= \{10, 20, 30, 50, 75, 100, 150\}, \\ \omega_{sq} &= \{\omega^*, \omega^* + 0.5, \omega^* + 1, \omega^* + 2, \\ &\qquad \omega^* + 4, \omega^* + 8, \omega^* + 16\}. \end{aligned} \tag{12}$$

5.4. Результаты моделирования

Согласно стратегии моделирования, для каждого алгоритма и каждого набора значений параметров 1000 раз вычисляется ошибку

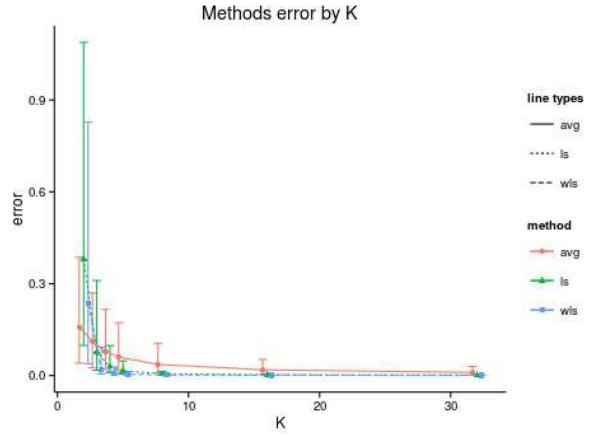

Рис. 3: Квантили ошибок оценки точки минимума функции в зависимости от количества узлов K.

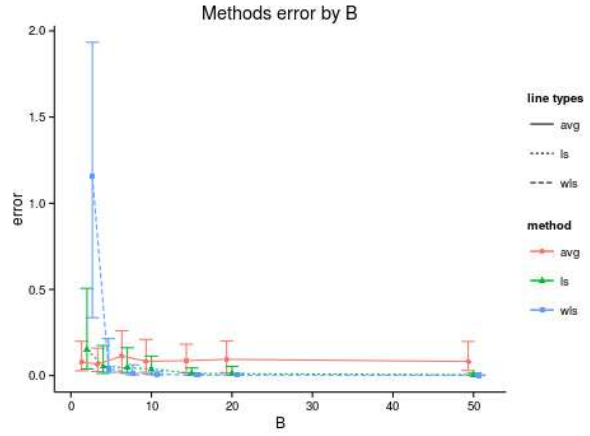

Рис. 4: Квантили ошибок оценки точки минимума функции в зависимости от размера пакета B.

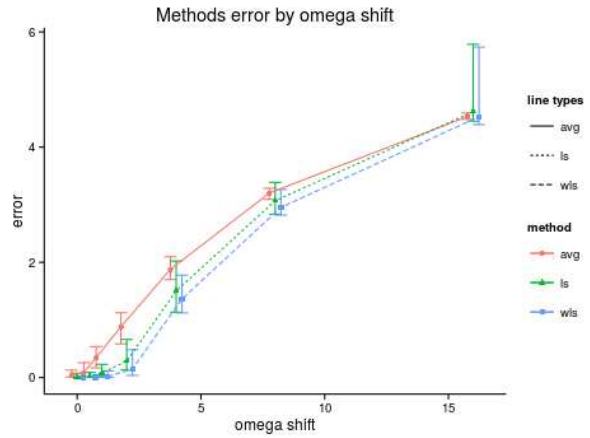

Рис. 5: Квантили ошибок оценки точки минимума функции в зависимости от количества от сдвига центра генерации начального приближения $||\omega_{sq} - \omega^*||_2$.

и считаются 0.25, 0.5 и 0.75 квантили этих ошибок. Затем для каждого варьируемого параметра график полученных квантили каждого алгоритма строится следующим образом. Для каждого алгоритма (с использованием различных типов линий и цветов)

- значения варьируемого параметра откладываются по оси абсцисс;
- значения квантилей откладываются по оси ординат и соответствующие точки отмечаются на графики;
- точки, соответствующие 0.5 квантилям соединяются.

На каждом графике присутствует легенда, где указан тип линий и цвет для каждого алгоритма, где алгоритмы SimuParallel mbSGD, LSDSGD, WLSDSGD обозначаются соответственно как *avg*, *ls* и *wls*.

Сначала рассмотрим варьируемый параметр T. Ошибки алгоритмов в зависимости от параметра T продемонстрированы на ри-

сунке 2. Видно, что LSDSGD и в особенности WLSDSGD превосходят алгоритм SimuParallel mbSGD начиная с 20 итераций.

Далее рассмотрим рисунок 3, где варьируется параметр K. Здесь ситуация схожа: ошибка WLSDSGD меньше, чем у LSDSGD, в свою очередь меньшая, чем ошибка алгоритма SimuParallel mbSGD начиная с количества узлов равного 4.

На рисунке 4 изображена зависимость ошибок алгоритмов от B. Можно заметить, что результат WLSDSGD значительно хуже результата других алгоритмов при малых B. Это можно объяснить тем, что дисперсия оценки весов пропорционально B^{-1} и относительно велика при малых B. Как бы то ни было, начиная с разумных значений B (~ 5) и WLSDSGD и LSDSGD вновь превосходят алгоритм SimuParallel mbSGD

Наконец, рассмотрим рисунок 5. D Несмотря на то, что ошибка всех трех алгоритмов увеличивается при росте сдвига ω, оба алгоритма LSDSGD и WLSDSGD превосходят алгоритм SimuParallel mbSGD, и WLSDSGD, в свою очередь, превосходит LSDSGD.

6. Заключение

В работе рассмотрена частную задачу распределенной оптимизации разделимой стохастической квадратичной функции и предложена два алгоритма для ее решения: LSDSGD и WLSDSGD, основанные на восстановлении функции потерь на основе ее измерений. Эти алгоритмы в значительной степени базируются на алгоритме SimuParallelSGD, предложенном в работе [14] и также отлично подходят для реализации в парадигме распределенных вычислений MapReduce. Экспериментально исследована точность оценки точки минимума функции алгоритмами SimuParallelSGD, LSDSGD и WLSDSGD на модельных данных при различных значениях параметров алгоритмов, показав, что LSDSGD и WLSDSGD показывают лучшую точность оценки точки минимума функции чем алгоритм SimuParallel mbSGD при большинстве значений параметров.

В последующих работах на эту тематику возможно изучение влияния размерности области определения функции потерь (параметра ω) на точность оценки минимума функции, а так же апро-

бация на реальных задачах. Кроме того, как отмечено в замечании 1, предложенные алгоритмы не учитывают проблему смещенности оценки функции потерь, что может пагубно сказаться на точности их оценок. Одним из возможных решений данной проблемы заключается в применении одного из методов оценки параметров при произвольных внешних возмущениях (смотри работы [19, 20]).

Список литературы

[1] *Boyd S.* Global optimization in control system analysis and design // Control and Dynamic Systems V53: High Performance Systems Techniques and Applications: Advances in Theory and Applications. 2012. Vol. **53**.

[2] *Bottou L.* Online learning and stochastic approximations On-line learning in neural networks. 1998. Vol. **17**. No. 9. 142 p.

[3] *Granichin O., Volkovich V., Toledano-Kitai D.* Randomized Algorithms In Automatic Control And Data Mining. — Springer-Verlag: Heidelberg, New York, Dordrecht, London. 2015. 251p.

[4] *Граничин О.Н., Поляк Б.Т.* Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. — М.: Наука. 2003. 293 с.

[5] *Boyd S., Parikh N., Chu E., Peleato B., Eckstein J.* Distributed optimization and statistical learning via the alternating direction method of multipliers // Foundations and Trends in Machine Learning. 2011. Vol. **3**. No. 1. pp. 1–122.

[6] *Поляк Б.Т.* Введение в оптимизацию. — М.: Наука. 1983. 384 с.

[7] *Boyd S., Vandenberghe L.* Convex Optimization. — Cambridge university press. 2004.

[8] *Нестеров Ю.Е.* Введение в выпуклую оптимизацию. — М.: МЦНМО. 2010. 274 с.

[9] *Cauchy A.* Méthode générale pour la résolution des systemes d'équations simultanées // Comp. Rend. Sci. Paris. 1847. Vol. **25**. pp. 536–538.

[10] Davidon W.C. New least-square algorithms // Journal of Optimization Theory and Applications. 1976. Vol. **18**. No. 2. pp. 187–197.

[11] *Yann L.C., Bottou L.* Large scale online learning // Advances in neural information processing systems. 2004. Vol. **16**. 217. p.

[12] *Bertsekas D. P.* Parallel and Distributed Computation: Numerical Methods. — Athena Scientific. 1997.

[13] *Chu C., Kim S.K., Lin Y.A., Yu Y., Bradski G., Ng A.Y., Olukotun K.* Map-reduce for machine learning on multicore // Advances in neural information processing systems. 2007. Vol. **19**.

[14] *Zinkevich M., Weimer M., Li L., Smola A.J.* Parallelized stochastic gradient descent // In: Advances in Neural Information Processing Systems. 2010. pp. 2595–2603.

[15] *Dean J., Sanjay G.* Mapreduce: simplified data processing on large clusters // In: OSDI'04, Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation. 2004. USENIX Association. pp. 10–10.

[16] *Mann G., McDonald R., Mohri M., Silberman N., Walker D.* Efficient large-scale distributed training of conditional maximum entropy models // In: Advances in Neural Information Processing Systems. 2009. Vol. **22**. pp. 1231–1239.

[17] *Dekel O., Gilad-Bachrach R., Shamir R., Xiao L.* Optimal distributed online prediction using mini-batches // The Journal of Machine Learning Research. 2002. Vol. **13**. No. 1. pp. 165–202.

[18] *White H.* A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity // Econometrica: Journal of the Econometric Society. 1980. pp. 817–838.

[19] *Senov A., Amelin K., Amelina N., Granichin O.* Exact confidence regions for linear regression parameter under external arbitrary noise // In: Proc. of the 2014 American Control Conference (ACC). 2014. Portland, USA, 4-6 June. pp. 5097–5102.

[20] *Сенов А.А., Граничин О.Н.* Идентификация параметров линейной регрессии при произвольных внешних помехах в наблюдениях // В сб. трудов XII Всероссийское совещание по проблемам управления (ВСПУ-2014). 2014. С. 2708–2719.