ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 519.86

Отчет об исследовательском проекте на тему:

Распределенные алгоритмы в условиях схожести слагаемых

Выполнил студент:

группы #БПМИ212, 3 курса Панкевич Сергей Александрович

Принял руководитель проекта:

Двинских Дарина Михайловна
Старший научный сотрудник
Факультет компьютерных наук НИУ ВШЭ

Москва 2024

Содержание

Аннотация

Алгоритмы распределенной оптимизации становятся все более актуальными, так как современные модели машинного обучения зачастую тренируются на массивах данных, которые невозможно разместить на одном вычислительном устройстве в силу их большого размера или приватности. В данной работе рассматривается случай централизованной оптимизации, где один сервер хранит параметры модели, данные разделены между несколькими устройствами и сервером, управляющим процессом оптимизации. Целью работы является получение лучшего на практике алгоритма для случая сильно выпуклых функций и в условиях схожести слагаемых.

Ключевые слова

Методы оптимизации, распределенная оптимизация, централизованная оптимизация, сильная выпуклость, схожесть слагаемых.

1 Введение

1.1 Описание предметной области

Распределенная оптимизация возникает в случае, когда функционал зависит от данных, размещенных на различных носителях. Например, в случае поиска параметров некой модели машинного обучения оптимизируется эмпирическая функция потерь. В централизованном случае распределенной оптимизации данные размещены на сервере и других устройствах. Происходит итерационный процесс коммуникаций, в ходе которого сервер поручает другим устройствам выполнять некоторые вычисления, они направляют результаты обратно серверу и сервер выполняет шаги оптимизационного процесса, используя данные, размещенные на нем и полученные значения от других устройств.

1.2 Постановка задачи

В данной работе работе рассматривается централизованная оптимизация в случае, когда слагаемые, из которых состоит оптимизируемый функционал, в какой-то мере схожи между собой, а сам функционал является сильно выпуклым. Целью работы является исследование возможностей улучшения алгоритмов, работающих при упомянутых ограничениях, для лучшего применения на практике. Кроме практической части в данной работе так же приводятся теоретическое исследование полученного метода.

2 Формальная постановка задачи

Скажем, что всего есть n устройств и m элементов обучающей выборки(примеров) на каждом устройстве, $N = nm$ – суммарное количество примеров. Обозначим за $z_i^{(j)}$ – элемент обучающей выборки под номером i на устройстве номер j. Будем обозначать за $x \in \mathbb{R}^d$ параметры. Обозначим за $l(x, z)$ – функцию потерь на примере z, при векторе параметров x. Кроме того введем следующие обозначения:

$$f_i(x) = \frac{1}{m} \sum_{j=1}^{m} l(x, z_i^j) \tag{1}$$

$$r(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{2}$$

Если вектор параметров равен x, то $r(x)$ представляет из себя эмпирический риск на

всем массиве данных, $f_i(x)$ – на устройстве i.

Нашей целью будет являться решение следующей задачи оптимизации

$$\min_{x \in \mathbb{R}^d} \quad r(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{3}$$

В [] предлагается представить функцию в следующем виде:

$$r(x) = p(x) + q(x), \tag{4}$$

$$q(x) = f_1(x) \tag{5}$$

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \left[f_i(x) - f_1(x) \right]. \tag{6}$$

В дальнейшем мы увидим, что это способствует уменьшению числа коммуникаций, если f_i похожи между собой.

Предположения

Функция $g : \mathbb{R}^d \to \mathbb{R}$ называется $L-$гладкой на множестве $\mathbb{R}^d$, если ее градиент $L-$липшицев, то есть верно

$$\|\nabla g(y) - \nabla g(x)\|_2 \leq L \|x - y\|_2, \quad \forall\, x, y \in \mathbb{R}^d. \tag{7}$$

Функция $g : \mathbb{R}^d \to \mathbb{R}$ называется $\mu-$сильно выпуклой на множестве $\mathbb{R}^d$, если для положительного μ выполняется

$$g(y) \geq g(x) + \nabla g(x)^\top (y - x) + \frac{\mu}{2} \|x - y\|_2^2, \quad \forall\, x, y \in \mathbb{R}^d. \tag{8}$$

Теоретический анализ и все методы построены в следующих предположениях:

- $p(x)$ является L_p-гладкой на $\mathbb{R}^d$,
- $q(x)$ является L_q-гладкой и выпуклой на $\mathbb{R}^d$,
- $r(x)$ является $L-$гладкой и μ-сильно выпуклой на $\mathbb{R}^d$.

Отметим, что последнее предположение часто выполняется в задачах машинного обучения из-за добавления L_2-регуляризации к выпуклым функциям потерь, и чем больше

коэффициент регуляризации, тем выше константа сильной выпуклости. На практике оптимальный коэффициент регуляризации обычно довольно мал, из-за чего классические методы с оценкой числа итераций до достижения определенной точности пропорциональной k(алгоритм [??]) или $\sqrt{k}$(алгоритм [??]) (где $k = \frac{L}{\mu}$ – число обусловленности), сходятся заметно медленее методов, разработанных для случаев, когда имеет место схожесть слагаемых(алгоритмы [??], [??]).

3 Методы решения задачи

3.1 Примитивы

Для начала определим примитив, которым будут пользоваться последующие алгоритмы

Algorithm 1 Подсчет $\nabla p(x)$

1: **Input**: $x \in \mathbb{R}^d$
2: **send** x to all computers
3: **for** $i = 2, 3 \ldots, n$ **do**
4: **receive** x
5: **compute** $\nabla f_i(x)$
6: **send** $\nabla f_i(x)$
7: **end for**
8: **receive** $\nabla f_2(x), \nabla f_2(x), \ldots, \nabla f_n(x)$
9: **compute** $\nabla f_1(x)$
10: **compute** $\nabla p(x) = \frac{1}{n} \sum_{i=1}^{n} [\nabla f_i(x) - \nabla f_1(x)]$
11: **return** $\nabla p(x)$

Приведем подробное описание схемы взаимодействия сервера и других устройств, Сервер получает на вход значение x. В строчке 2 он отсылает его всем устройствам. После чего все устройства параллельно исполняют строчки 4, 5 и 6. В строчке 8 сервер получает резульаты вычислений всех устройств. В конце он вычисляет $\nabla r(x)$, используя полученные значения градиентов.

Данное действие требует 1 раунд коммуникаций и производит один вызов оракула градиента на сервере.

Вычисление $\nabla q(x)$ не требует коммуникаций и производит один вызов оракула градиента на сервере.

Algorithm 2 Подсчет $\nabla q(x)$

Input: $x \in \mathbb{R}^d$
compute $\nabla f_1(x)$
return $\nabla f_1(x)$

Algorithm 3 Подсчет $\nabla r(x)$

Input: $x \in \mathbb{R}^d$
compute $\nabla q(x)$
compute $\nabla p(x)$
return $\nabla q(x) + \nabla p(x)$

3.2 Централизованная версия градиентного спуска

Приведем пример работы градиентного спуска в условиях централизованной оптимизации.

Algorithm 4 Централизованный градиентный спуск

Input: $x^0 \in \mathbb{R}^d$
Parameters: η, K
for $t = 0, 1 \ldots, K-1$ **do**
    $x_{t+1} = x_t - \eta \nabla r(x_t)$
end for
return x_K

Где

$$\eta = \tfrac{1}{L}.$$

3.3 Централизованная версия ускоренного градиентного спуска

Где

$$\eta = \tfrac{1}{\eta}, \quad \kappa = \tfrac{L}{\mu}, \quad \gamma = \tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}.$$

Данные алгоритмы эквивалентны градиентному и ускоренному градиентному спуску в классическом нераспределенном случае по производимым вычислениям. Мы будем использовать их в качестве базовых решений для сравнения эффективности по числу коммуникаций и числу вызовов оракула градиента.

3.4 Accelerated Extragradient

В статье [???] приведен алгоритм, который является асимптотически оптимальным как по числу локальных вызовов оракула градиента, так и по числу коммуникаций.

Algorithm 5 Централизованный ускоренный градиентный спуск

Input: $x^0 \in \mathbb{R}^d$
Parameters: $\eta, \kappa, \mu, L, \gamma$
$y_0 = x_0$
for $t = 0, 1 \ldots, K-1$ **do**
  $y_{t+1} = x_t - \eta \nabla r(x_t)$
  $x_{t+1} = (1+\gamma)y_{t+1} - \gamma y_t$
end for
return x_K

Algorithm 6 Accelerated Extragradient

Input: $x^0 \in \mathbb{R}^d$
Parameters: K, η, θ, α
$x_f^0 = x^0$
for $t = 0, 1 \ldots, K-1$ **do**
  $x_g^k = \tau x^k + (1-\tau)x_f^k$
  $x_f^{k+1} \approx \operatorname{argmin}_{x \in \mathbb{R}^d} \left[A_\theta^k(x) \coloneqq p(x_g^k) + \langle \nabla p(x_g^k), x - x_g^k \rangle + \frac{1}{2\theta}\|x - x_g^k\|_2^2 + q(x) \right]$
  $x^{t+1} = x^t + \eta\alpha(x_f^{t+1} - x^t) - \eta \nabla r(x_f^{t+1})$
end for
return x^K

Где

$$\tau = \min\left\{1, \frac{\sqrt{\mu}}{2\sqrt{L_p}}\right\}, \quad \theta = \frac{1}{2L_p}, \quad \eta = \min\left\{\frac{1}{2\mu}, \frac{1}{2\sqrt{\mu L_p}}\right\}, \quad \alpha = \mu;$$

В данном случае в строчке 6 имеется в виду приближенное решение задачи, для которого соблюдается условие

$$\|\nabla A_\theta^k(x_f^{k+1})\|_2^2 \leq \frac{L_p^2}{3}\|x_g^k - \operatorname*{argmin}_{x \in \mathbb{R}^d} A_\theta^k(x)\|_2^2$$

Данный алгоритм на каждой итерации совершает 2 раунда коммуникации. Один для вычисления $\nabla r(x_f^{t+1})$ и один во время вычисления $\nabla p(x_g^k)$ в начале решения подзадачи. Кроме того при решении подзадачи алгоритм T раз вычисляет ∇q. В итоге алгоритм совершает $2K$ вызовов ∇p и $(T+1)K$ вызовов ∇q, если вычислять $\nabla r(x)$ как $\nabla p(x) + \nabla q(x)$.

В статье [] показано, что при

$$K \geq 2\max\left\{1, \sqrt{\frac{L_p}{\mu}}\right\} \log \frac{\|x^0 - x^*\|_2^2 + \frac{2\eta}{\tau}\left[R(x^0) - R(x^*)\right]}{\varepsilon}, \tag{9}$$

Расстояние до решения задачи x^*:

$$\|x^K - x^*\|_2^2 \leq \varepsilon. \tag{10}$$

И условие [] гарантированно будет выполнено следующем числе итераций решения

подзадачи

$$T = \sqrt[4]{3} D \max\left\{1, \sqrt{\frac{L_q}{L_p}}\right\}, \tag{11}$$

где D является констаной, не зависящей от констант выпуклости и гладкости.

Что дает следуюшие оценки при условии $\mu \leq L_p \leq L_q$:

$$\mathcal{O}\left(\sqrt{\frac{L_q}{\mu}} \log \frac{1}{\epsilon}\right) \text{ вызовов } \nabla q(x) \;\; ; \;\; \mathcal{O}\left(\sqrt{\frac{L_p}{\mu}} \log \frac{1}{\epsilon}\right) \text{ вызовов } \nabla p(x).$$

И это является асимптотически оптимальным результатом.

3.4.1 Алгоритм решения

3.5 Practical Accelerated Extragradient

Заметим, что условие [???] непрактично, так как невозможно применять итерационный метод для поиска решения и прекращать итерации, как только оно стало выполнено. Это из-за того что в выражении справа необходимо знать $\operatorname{argmin}_{x \in \mathbb{R}^d} A_\theta^k(x)$. Поэтому в представленном в статье [???] алгоритме необходимо делать фиксированное число итераций T оптмизационного процесса такое, что условие [] гарантированно выполнено.

В нашем методе будем искать решение такое, что выполняется другое условие:

$$\|\nabla A_\theta^k(x_f^{k+1})\|_2^2 \leq L_p^2 \|x_g^k - x_f^{k+1}\|_2^2$$

Искать эту точку будем, минимизируя A_θ^k, с использованием ускоренного градиентного спуска, начиная из точки x_g.

Algorithm 7 Subproblem solver 2

Input: $x_g^k \in \mathbb{R}^d$
Parameters: $\eta, \kappa, \mu, L_p, \gamma$
$y_0 = x_0 = x_g^k$
compute $\nabla p(x_g^k)$
for $t = 0, 1 \ldots$ **do**
    compute $\nabla q(x_t)$
    $\nabla A_\theta^k(x_t) = \nabla p(x_g^k) + \frac{1}{\theta}(x_t - x_g^k) + \nabla q(x_t)$
    if $\nabla A_\theta^k(x_t) \leq L_p^2 \|x_g^k - x_f^{k+1}\|_2^2$
    return x_t
    $y_{t+1} = x_t - \eta \nabla r(x_t)$
    $x_{t+1} = (1 + \gamma) y_{t+1} - \gamma y_t$
end for

Данный алгоритм в отличие от предыдущего позволяет завершать процесс раньше,

чем будет произведено количество итераций, соответсвующее верхней оценке, приведенной в последующем анализе. В части с экспериментами на реальных и синтетических данных показано, что это позволяет добиться заметно меньшего числа вызовов оракула градиента.

4 Теоретический анализ

Отметим несколько свойств функции $A_\theta^k(x)$, которые важны для приближенного решения задачи минимизации:

- $\nabla A_\theta^k(x)$ имеет константу липшицевости $2L_p + L_q$

  Доказательство:

  $\nabla A_\theta^k(x) = \nabla p(x_g^k) + \frac{1}{\theta}(x - x_g^k) + \nabla q(x)$

  Первое слагаемое константно, а следующие два дают, что градиент липшицев с константой $\frac{1}{\theta} + L_q = 2L_p + L_q$

- $A_\theta^k(x)$ является $\frac{1}{\theta}$-сильно выпуклой

  Доказательство:

  $p(x_g^k)$ константна, $\langle \nabla p(x_g^k), x - x_g^k \rangle$ линейна, $q(x)$ выпукла, поэтому слагаемое $\frac{1}{2\theta}\|x - x_g^k\|_2^2$ обеспечивает, что $A_\theta^k(x)$ является $\frac{1}{\theta} = 2L_p$-сильно выпуклой

4.1 Число итераций для решения поздадачи

Необходимо найти такое x_f^{k+1}, для которого верно $\|\nabla A_\theta^k(x_f^{k+1})\|_2^2 \leq L_p^2\|x_g^k - x_f^{k+1}\|_2^2$, что эквивалентно $\|\nabla A_\theta^k(x_f^{k+1})\|_2 \leq L_p\|x_g^k - x_f^{k+1}\|_2$

Из липшицевости градиента следует

$$\|\nabla A_\theta^k(x)\|_2 \leq (2L_p + L_q)\|x - x_*\|_2$$

Значит достаточно, чтобы выполнялось

$$\|x_f^{k+1} - x_*\|_2 \leq \beta\|x_g^k - x_f^{k+1}\|_2$$

Где $\beta = \frac{L_p}{2L_p + L_q}$

Воспользуемся неравенством треугольника и получим

$$\|x_f - x_g\|_2 \geq \|x_g - x_*\|_2 - \|x_f - x_*\|_2$$

Тогда достаточно

$$\|x_f^{k+1} - x_*\|_2 \leq \beta(\|x_g - x_*\|_2 - \|x_f - x_*\|_2)$$

$$\|x_f^{k+1} - x_*\|_2 \leq \frac{\beta}{1+\beta}\|x_g^k - x_*\|_2$$

Что эквивалентно

$$\frac{\|x_f^{k+1} - x_*\|_2}{\|x_g^k - x_*\|_2} \leq \frac{\beta}{1+\beta}$$

То есть ошибка по аргументу минимизируемой функции должна составлять $\frac{\beta}{1+\beta}$ от изначальной. Для удобства анализа привведем во сколько раз должна уменьшиться невязка, чтобы данное условие гарантированно выполнилось.

Для этого необходимо оценить сверху отношение невязок, если ошибки по аргументу отличаются в $\frac{\beta}{1+\beta}$ раз.

Нам понадобятся два неравенства следующих неравенства:

$$A_\theta^k(x) - A_\theta^k(x_*) \geq \nabla A_\theta^k(x_*)^T(x - x_*) + \frac{1}{2\theta}\|x - x_*\|_2^2 = \frac{1}{2\theta}\|x - x_*\|_2^2;$$

$$A_\theta^k(x) - A_\theta^k(x_*) \leq \int_0^{\|x-x_*\|_2} (2L_p + L_q)t dt = (2L_p + L_q)\frac{\|x - x_*\|_2^2}{2};$$

первое следует из сильной выпуклости, а второе из липшицевости градиента.

Таким образом $A_\theta^k(x_f^{k+1}) - A_\theta^k(x_*) \geq \frac{1}{2\theta}\|x_f^{k+1} - x_*\|_2^2$ и $A_\theta^k(x_g) - A_\theta^k(x_*) \leq (2L_p + L_q)\frac{\|x_g - x_*\|_2^2}{2}$, значит

$$\frac{\|x_f^{k+1} - x_*\|_2^2}{\|x_g - x_*\|_2^2} \leq \theta(2L_p + L_q)\frac{A_\theta^k(x_f^{k+1}) - A_\theta^k(x_*)}{A_\theta^k(x_g) - A_\theta^k(x_*)}$$

Тогда для [???] достаточно

$$\frac{A_\theta^k(x_f^{k+1}) - A_\theta^k(x_*)}{A_\theta^k(x_g) - A_\theta^k(x_*)} \leq \frac{1}{\theta(2L_p + L_q)}\frac{\beta^2}{(1+\beta)^2} = \frac{2\beta^3}{(1+\beta)^2}$$

Так как $\beta \leq 1$, неравенство выше выполняется и когда

$$\frac{A_\theta^k(x_f^{k+1}) - A_\theta^k(x_*)}{A_\theta^k(x_g) - A_\theta^k(x_*)} \leq \frac{\beta^3}{2}$$

Найдем число обусловленности A_θ^k:

$$\kappa_A = \frac{2L_p + L_q}{\frac{1}{\theta}} = \frac{2L_p + L_q}{2L_p} = \frac{1}{2\beta}$$

Тогда для достижения [???] ускоренному градиентному спуску необходимо

$$O\left(\sqrt{\kappa_A}\log\frac{2}{\beta^3}\right) = O\left(\sqrt{\kappa_A}\log\frac{1}{\beta}\right) = O\left(\sqrt{\kappa_A}\log\kappa_A\right)$$

итераций

Заметим, что в предположении $\mu \leq L_q \leq L_p$

5 Условие схожести слагаемых

Скажем, что функции $f(x)$ и $g(x)$ являются δ-схожими на некотором множестве S, если выполнено следующее.

$$\left\|\nabla^2 f(x) - \nabla^2 g(x)\right\| \leq \delta, \quad \forall\, x \in S, \tag{12}$$

Рассматриваемая нами функция r имеет вид суммы нескольких слагаемых. Заметим, что если данные на каждом устройстве получены независимо из одного и того же распределения, то f_i весьма вероятно похожи между собой и хорошо приближают функцию r. А именно, в (Troop J., 2015)[1] с помощью матричного неравенства Хеффдинга показано, что с вероятностью не меньше $1 - p$ выполнено:

$$\left\|\nabla^2 f_i(x) - \nabla^2 r(x)\right\| \leq \sqrt{\frac{32A_l^2\log(d/p)}{m}}, \tag{13}$$

Для некоторого числа A_l такого, что $A_l \geq \|\nabla^2\ell(x, z_i)\|\ \forall\, x \in S$.

Упрощенно это можно записать так: с высокой вероятностью $\delta = \tilde{O}(1/\sqrt{m})$, где $\tilde{O}$ показывает асимпотическую зависимость без учета размерности и логарифмических множителей.

Пусть выполнена δ-схожесть слагаемых:

$$\|\nabla^2 f_i(x) - \nabla^2 f_1(x)\|_2 \leq \delta,\ \forall i \in \{1 \dots n\},\ x \in R^d. \tag{14}$$

Тогда

$$\nabla^2 p(x) = \frac{1}{n}\sum_{i=1}^{n}\left[\nabla^2 f_i(x) - \nabla^2 f_1(x)\right]$$

$$L_p \leq \|\nabla^2 p(x)\|_2 \leq \frac{1}{n}\sum_{i=1}^{n}\delta = \delta.$$

Во многих исследованиях δ-схожесть используется для уменьшения числа коммуникаций [??] [??Hendrikx]. В алгоритмах [] и [] для ее использования функция $r(x)$ разделялась на $q(x)$, вся информация о которой содержится на сервере и не требует коммуникаций, и на $p(x)$, для которой выполнено $L_p \leq \delta = \tilde{O}(1/\sqrt{m})$.

6 Экспериментальное исследование метода

Будем рассматривать следующую задачу минимизации эмпирического риска (Ridge-регрессию)

$$F(w) = \frac{\lambda}{2}||w||_2^2 + \frac{1}{2N}\sum_{i=1}^{n}(w^t x_i - y_i)^2 \to_w \min$$

Где x_i – вектор параметров объекта i, w – параметры модели, λ – коэффициент регуляризации. Эквивалентно ее можно переписать в матричном виде следующим образом:

$$\frac{1}{2N}||Xw - y||_2^2 + \frac{\lambda}{2}||w||_2^2 \to_w \min$$

Где X – матрица высоты N и ширины d, в строке i которой находится x_i^t. Будем ее называть матрицей дизайна.

Будем делать два типа экспериментов: на синтетических данных и реальных. В обоих случаях распределенность будет симулироваться. Мы будем производить все вычисления на одном устройстве, но при соверешении действий, эквивалентных коммуникации сервера и устройств, будем инкрементировать счетчик числа коммуникаций. Аналогично при подсчете градиента функции, относящихся к слагаемым на одной из устройств, будем инкрементировать счетчик числа локальных вызовов.

6.1 Эксперимент на синтетических данных

Будем рассматривать сеть из n устройств, первая из которых является сервером. На каждом устройстве будут располагаться m примеров.

- Сгенерируем матрицу дизайна для сервера X_1, где $X_{1ij} \sim^{i.i.d} \mathcal{N}(0, L_{mult})$ с использова-

нием **numpy**. Константа L_{mult} позволяет регулировать константу липшица градиента функции потерь по параметрам. Чем больше L_{mult}, тем в среднем выше константа липшицевости.

- После чего сгенерируем вектор w, $w_i \sim^{i.i.d} \mathcal{N}(0, 1)$.
- Теперь положим $y = X_1 w + \varepsilon$, где $\varepsilon_i \sim^{i.i.d} \mathcal{N}(0, 1)$

  Это соответствует классическим статистическим предположениям в задаче линейной регрессии.

- Сгенерируем матрицы дизайна для выборки других устройств ($k \geq 2$) следующим образом $X_k = X_1 + \delta_{mult} P_k$, где $P_{kij} \sim^{i.i.d} \mathcal{N}(0, 1)$. Константа δ_{mult} позволяет регулировать схожесть данных на сервере и других устройствах. Чем она меньше, тем в среднем ближе будут гессианы функций на разных устройствах.

Частично для приближения к условиям статьи [] были выбраны следующие константы:

$$\lambda = 0.1, \quad n = 25, \quad m = 100, \quad d = 50, \quad \delta_{mult} = 0.1, \quad L_{mult} = 2.$$

6.2 Эксперимент на реальных данных

В качестве реальных данных для эксперимента использовался набор данных, в котором собрана информация о погоде.

В качестве признаков выбирались столбцы

- “Wind Speed (km/h)’
- “Humidity’
- “Wind Bearing (degrees)’
- “Visibility (km)’
- “Loud Cover’
- “Pressure (millibars)’
- “Temperature (C)’

В качестве предсказываемой величины “Apparent Temperature (C)’.

Кроме того к признакам добавлен единичный свободный член. Признаки отнормированы так, что имеют нулевое среднее и единичную выборочную дисперсию. Объекты распределены между 25 устройствами в случайном порядке.

6.3 Оценка констант δ, L_p, L_q, μ

Для работы некоторых методов необходимо знание констант гладкости и сильной выпуклости. Кроме того их знание полезно для отслеживания зависимостей качества различных методов от свойств данных.

$$f_i(w) = \frac{1}{m}||X_i w - y||_2^2 + \frac{\lambda}{2}||w||_2^2$$

$$\nabla_w f_i = \frac{1}{m} X_i^T (X_i w - y) + \lambda w$$

$$\nabla_w^2 f_i = \frac{1}{m} X_i^T X_i + \lambda I_d$$

6.3.1 Оценка L_q

$$q(w) = f_1(w)$$

$$\nabla_w^2 q = \frac{1}{m} X_i^T X_i + \lambda I_d$$

$$L_q = \frac{1}{m} \lambda_{\max}(X_i^T X_i) + \lambda$$

6.3.2 Оценка L_p

$$p(w) = \frac{1}{n} \sum_{i=1}^{n} [f_i(w) - f_1(w)]$$

$$\nabla_w^2 p = \frac{1}{n} \sum_{i=1}^{n} [\nabla_w^2 f_i - \nabla_w^2 f_1] =$$

$$= \frac{1}{n} \sum_{i=1}^{n} [\frac{1}{m} X_i^T X_i + \lambda I_d - \frac{1}{m} X_1^T X_1 - \lambda I_d] =$$

$$= \frac{1}{n} \sum_{i=1}^{n} [\frac{1}{m} X_i^T X_i] - \frac{1}{m} X_1^T X_1 =$$

$$L_p = \lambda_{\max}(\frac{1}{n}\sum_{i=1}^{n}[\frac{1}{m}X_i^T X_i] - \frac{1}{m}X_1^T X_1)$$

Где $\lambda_{\max}$ – наибольшее собственное значение матрицы, которое можно найти с помощью функции **eigvalsh** из пакета **scipy**, которая подходит для работы с симметричными вещественными матрицами.

6.3.3 Оценка δ

Отметим, что в нашем случае у всех функций гессиан является константным, поэтому

$$\delta = \max_i ||\nabla_w^2 f_1 - \nabla_w^2 f_i||_2 = \frac{1}{m}\max_i ||X_1^T X_1 - X_i^T X_i||$$

6.3.4 Оценка μ

Каждая функция $\frac{1}{m}||X_i w - y||_2^2$ является выпуклой, $\frac{\lambda}{2}||w||_2^2$ является λ-сильно выпуклой, поэтому каждая функция f_i и $r = \sum \frac{1}{n} f_i$ являются μ-сильно выпуклой, где $\mu = \lambda$.

6.4 Оценка сходимости методов

Для оценки сходимости каждого метода на итерации T будем вычислять во сколько раз квадрат расстояния до оптимума меньше, то есть

$$\frac{||w^T - w_*||_2^2}{||w^0 - w_*||},$$

где

$$w_* = \operatorname{argmin} \frac{1}{2N}||Xw - y||_2^2 + \frac{\lambda}{2}||w||_2^2$$

Во всех методах будем полагать $w_0 = 0$. Для задачи (todo) сущетсвует точное аналитическое решение:

$$w_* = (X^T X + N\lambda I_d)^{-1} X^T y,$$

которое мы будем использовать в экспериментах.

Отметим, что зачастую аналитическое решение непрактично из-за высокой временной сложности обращения матриц и вычислительной неустойчивости, поэтому на практике зачастую используются именно итерационные методы, подобные методам, описываемым в данной работе. Кроме того для широкого класса задач, не существует явного аналитического выражения решения, поэтому такие методы

7 Результаты экспериментов

8 Выводы

Список литературы

[1] Troop J. "User-friendly tail bounds for sums of random matrices". B: *Foundations of computational mathematic* (2012).