Стохастическая кубическая регуляризация для быстрых невыпуклых оптимизация

Нилеш Трипуранени ~ Митчелл Стерн ~ Чи Джин Джеффри Реджир Майкл И. Джордан
{Nilesh_tripuraneni, Mitchell, chijin, Regier} @ berkeley.edu
jordan@cs.berkeley.edu

Калифорнийский университет, Беркли

Аннотация

В данной работе предлагается стохастический вариант классического алгоритма - кубически-регуляризованный метод Ньютона [Нестеров, Поляк, 2006]. Предложенный алгоритм эффективно избегает седловых точек и находит **приближенные локальные минимумы для общих гладких невыпуклых функций только в ~**

***O (✏-3,5)* стохастический градиент и стохастик**

Гессиано-векторные оценки произведений. Последнее может быть вычислено так же эффективно, как и **стохастические градиенты. Это улучшает ~**

***O (✏-4)* скорость стохастического градиента**

спуск. Наша скорость соответствует наиболее известному результату для поиска локальных минимумов, не требуя деликатных методов ускорения или уменьшения дисперсии.

1. Введение

Рассмотрим проблему невыпуклой оптимизации в рамках стохастической аппроксимации [Роббинс и Монро, 1951]:

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{\xi \sim D}[f(x; \xi)]. \tag{1}$$

В этой настройке мы имеем доступ только к стохастической функции *e (Икс; ←-),* где случайная величина ← выбирается из основного распределения *D.* Задача состоит в том, чтобы оптимизировать ожидаемую функцию *e (Икс),* который в целом может быть невыпуклым. Эта структура охватывает широкий спектр проблем, в том числе настройку, в которой мы минимизируем эмпирические потери при фиксированном объеме данных, и настройку в режиме онлайн, когда данные поступают последовательно. Одним из наиболее важных приложений стохастической оптимизации является крупномасштабная статистика и проблемы машинного обучения, такие как оптимизация глубоких нейронных сетей.

Классический анализ в невыпуклой оптимизации гарантирует только сходимость к стационарной точке первого порядка (т.е. точке Икс удовлетворяющих *кр e (Икс) к = 0), который может быть локальным минимумом, локальным максимумом или седловой точкой. Эта статья идет дальше, предлагая алгоритм, который избегает седловых точек и сходится к локальному минимуму. Локальный минимум определяется как точка Икс удовлетворяющих *кр e (Икс) к = 0*
а также *р2 e (Икс) ⌫ 0. Поиск такой точки представляет особый интерес для большого класса статистических задач обучения, где локальные минимумы являются глобальными или почти глобальными решениями (например, Choromanska и др. [2015], Sun и др. [2016a, b], Ge et al. [2017]).

Среди алгоритмов стохастической оптимизации первого порядка стохастический градиентный спуск (SGD) является, пожалуй, самым **простым и наиболее универсальным. В то время как SGD вычислительно недорог, лучшая текущая гарантия для нахождения ✏- приблизительный локальный минимум (см. определение 1) требует *O (✏-4* поли(*г)) итерации [Ge et al., 2015], что неэффективно в многомерном режиме.**

˜ Равный вклад

32-я конференция по нейронным системам обработки информации (NeurIPS 2018), Монреаль, Канада.

В отличие от методов второго порядка, которые имеют доступ к гессиану *e* может использовать кривизну для более эффективного выхода из седел и достижения локальных минимумов. Однако построение полного гессиана может быть чрезмерно дорогим в больших размерах. Таким образом, в недавней работе было исследовано использование векторов Гессиана. $p_2\, e\,(\text{Икс})\cdot \text{v}$, которые могут быть вычислены так же эффективно, как градиенты во многих случаях, включая нейронные сети [Pearlmutter, 1994].

Среди алгоритмов второго порядка одним из наиболее естественных расширений алгоритма градиентного спуска является кубически-регуляризованный метод Ньютона Нестерова и Поляка [2006]. Принимая во внимание, что градиентный спуск находит минимизатор локального разложения Тейлора второго порядка на каждом этапе,

$$\text{Икс}_{t+1}^{\text{GD}} = \operatorname{argmin}_{\text{Икс}} \; e\,(\text{Икс}_t) + p\, e\,(\text{Икс}_t)^{\top}(\text{xx}_t) + \frac{}{2\,K\text{xx}_t\,K_2},$$

кубический регуляризованный метод Ньютона находит минимизатор локального разложения Тейлора третьего порядка,

час

$$\text{Икс}_{t+1}^{\text{кубический}} = \operatorname{argmin}_{\text{Икс}} \; e\,(\text{Икс}_t) + p\, e\,(\text{Икс}_t)^{\top}(\text{xx}_t) + 1 \quad \frac{}{2\,(\text{xx}_t)^{\top} p_2\, e\,(\text{Икс}_t)\,(\text{xx}_t) + \rightarrow} \quad \frac{}{6\,K\text{xx}_t\,K_3\,\text{я}},$$

Большинство предыдущих работ по кубически-регуляризованному методу Ньютона было сосредоточено на нестохастических или частично стохастических параметрах. Это заставляет нас задать центральные вопросы этой статьи: Можем ли мы разработать полностью стохастический вариант кубически-регуляризованного метода Ньютона? Если так, такой алгоритм быстрее, чем SGD?

В этой работе мы дадим положительный ответ на оба вопроса, сократив разрыв между его использованием в нестохастических и стохастических условиях.

В частности, мы предлагаем стохастический вариант кубически-регуляризованного метода Ньютона. Мы предоставляем неасимптотический анализ его сложности, показывающий, что предложенный алгоритм находит

$\tilde{\epsilon}$-стационарная точка второго порядка, использующая только $\tilde{O}\,(\epsilon^{-3.5})$ стохастический градиент и стохастический гессиан-вектор оценки, где $\tilde{O}\,(\cdot)$ скрывает полилогарифмические факторы.[1] Наш курс улучшается на $\tilde{O}\,(\epsilon^{-4})$ скорость стохастический градиентный спуск и соответствует наиболее известному результату для поиска локальных минимумов без необходимости каких-либо деликатных методов ускорения или уменьшения дисперсии (подробности см. в разделе 1.1). Мы также эмпирически показываем, что стохастический кубически-регуляризованный метод Ньютона, предложенный в этой статье, выгодно работает как на синтетических, так и на реальных невыпуклых задачах относительно современных методов оптимизации.

1.1 Связанная работа

В последнее время наблюдается всплеск интереса к методам оптимизации, которые могут избежать седловых точек и найти $\tilde{\epsilon}$-приблизительные локальные минимумы (см. определение 1) в различных условиях. Мы предоставляем краткое резюме этих результатов. Все сложности итераций в этом разделе изложены в терминах поиска приблизительных локальных минимумов и только подчеркивают зависимость от ϵ а также d.

1.1.1 Синглтон-функция

Эта линия работы оптимизируется по общей функции *e* без каких-либо особых структурных допущений. В этой настройке алгоритм оптимизации имеет прямой доступ к градиентным или гессианским оракулам на каждой итерации. В работе Нестерова и Поляка [2006] впервые предложен кубически-регуляризованный метод Ньютона, который требует $O\,(\epsilon^{-1.5})$ градиент и гессенский оракул призывает весь *e,* найти $\tilde{\epsilon}$-стационарная точка второго порядка. Позже алгоритм ARC [Cartis et al., 2011] и методы области доверия [Curtis et al., 2017] также продемонстрировали, что достигают той же гарантии при аналогичном доступе к гессианскому оракулу. Однако эти алгоритмы полагаются на доступ к полному гессиану на каждой итерации, что непомерно в больших измерениях.

Недавно, вместо использования полного Гессиана, Кармон и Дучи [2016] показали, что использование решателя градиентного спуска для подзадачи кубической регуляризации позволяет их алгоритму найти $\tilde{\epsilon}$-стационарные точки второго порядка в $\tilde{O}\,(\epsilon^{-2})$ Гессиано-векторные оценки произведений. С методами ускорения, число произведений вектора Гессиана может быть уменьшено до $\tilde{O}\,(\epsilon^{-1.75})$ [Carmon et al., 2016, Agarwal et al., 2017, Ройер и Райт, 2017].

[1] Один запрос для одной реализации $\leftarrow \leftarrow D$, в $p\, e\,(\text{Икс}, \leftarrow)$ или $p_2\, e\,(\text{Икс}, \leftarrow)\cdot \text{v}$ (для предварительно заданного v) упоминается как стохастический градиент или стохастическая оценка оракула по гессиану-вектору.

метод	время выполнения	Уменьшение дисперсии
Стохастический градиентный спуск [Ge et al., 2015] $O(\epsilon^{-4}$ поли$(d))$		не нужно
Наташа 2 [Аллен-Чжу, 2017]	$O(\epsilon^{-3.5})$ [2]	необходимый
Стохастическая кубическая регуляризация (эта статья)	$O(\epsilon^{-3.5})$	не нужно

Таблица 1: Сравнение наших результатов с существующими результатами для стохастической невыпуклой оптимизации с доказуемой сходимостью к приближенным локальным минимумам.

Между тем, в области результатов, совершенно не связанных с гессианом, Jin et al. [2017] показали, что простой вариант градиентного спуска может **найти** ϵ**- вторые стационарные точки в ~** $O(\epsilon^{-2})$ **оценки градиента.**

Обратите внимание, что это направление работы не учитывает стохастические градиенты или стохастические гессианы. Ограничение доступа только к запросам стохастических функций делает проблему оптимизации более сложной.

1.1.2 Настройка конечной суммы

В настройке конечной суммы (также известной как установка бесконечной суммы) где $f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x)$**, один предполагает, что алгоритмы имеют доступ к отдельным функциям** f_i**. В этом случае могут использоваться методы уменьшения дисперсии [Johnson and Zhang, 2013]. Агарвал и соавт. [2017] дать алгоритм, требующий ~** $O(\sqrt{n}\epsilon^{-3/2} + N\epsilon^{-3/4})$ **Гессенский вектор оракула называет (каждый к** $f_i(x)$**) найти** ϵ**- приближенный** местный минимум. Аналогичный результат достигается алгоритмом, предложенным Reddi et al. [2017].

1.1.3 Стохастическая аппроксимация

Каркас стохастической аппроксимации где $f(x) := \mathbb{E}_{\xi\sim D}[f(x;\xi)]$ **предполагает только доступ к стохастическому градиенту и гессиану через** $f(x;\xi)$**. В этом случае Ge et al. [2015] показали, что общая сложность итераций градиента для SGD, чтобы найти** ϵ**- стационарная точка второго порядка была** $O(\epsilon^{-4}$ поли$(d))$**.**
Совсем недавно, Kohler и Lucchi [2017] рассмотрели субсэмплированную версию алгоритма кубической регуляризации, но не предоставляют неасимптотический анализ для их алгоритма, чтобы найти приблизительный локальный минимум; они также предполагают доступ к точным (ожидаемым) значениям функций на каждой итерации, которые недоступны в полностью стохастическом параметре. Сюй и соавт. [2017] рассматривают случай стохастических гессианов, но также требуют доступа к точным градиентам и значениям функций на каждой итерации. Недавно Аллен-Чжу [2017] предложил алгоритм с механизмом, использующим уменьшение дисперсии, который находит стационарную точку **второго порядка с ~** $O(\epsilon^{-3.5})$ [2] **Гессиано-векторные оценки произведений.**

После этой работы Аллен-Чжу и Ли [2017] и Сюй и Янг [2017] показывают, как использовать градиентные оценки для эффективной аппроксимации векторно-гессианских произведений. Используя эту технику вместе с уменьшением дисперсии, они оба обеспечивают **алгоритмы достижения ~** $O(\epsilon^{-3.5})$ **Оценить с использованием градиентных оценок.**

Мы отмечаем, что наш результат соответствует лучшим на сегодняшний день результатам, используя более простой подход без каких-либо деликатных методов ускорения или уменьшения дисперсии. См. Таблицу 1 для краткого сравнения.

2 предварительных

Нас интересуют задачи стохастической оптимизации вида мин $_{x\in\mathbb{R}^d} f(x) := \mathbb{E}_{\xi\sim D}[f(x;\xi)]$**,**
где ξ **случайная величина с распределением** D**. В общем, функция** $f(x)$ **может быть невыпуклым Эта формулировка охватывает как стандартную установку, где целевая функция может быть выражена в виде конечной суммы** N **отдельные функции** $f(x,$ $\xi_i)$**, а также** онлайн-настройки, когда данные поступают последовательно.

Наша цель - минимизировать функцию $f(x)$ **используя только стохастические градиенты** $\nabla f(x;\xi)$ **и стохастические гессиано-векторные произведения** $\nabla^2 f(x;\xi)\cdot v$**, где v это вектор нашего выбора. Хотя на практике формирование целого гессиана является дорогостоящим и** зачастую трудоемким, вычисление векторного продукта Гессиана столь же дешево, как вычисление градиента, когда наша функция представляется в виде арифметической схемы [Pearlmutter, 1994], как в случае нейронных сетей ,

Обозначения: Мы используем жирные заглавные буквы A, Б для обозначения матриц и жирных строчных букв x, y для обозначения векторов. Для векторов мы используем $\|\cdot\|$ **обозначать** $\|\cdot\|_2$**- норма, а для матриц мы используем** $\|\cdot\|$ **обозначать**

[2] Оригинальный документ сообщает о скорости ~ $O(\epsilon^{-3.25})$ **из-за другого определения** ϵ**- стационарные второго порядка точка,** мин$(\nabla^2 F(X)) \geq -\sqrt{\epsilon}$ $O(\epsilon^{-1.4})$**, который слабее стандартного определения, как в определении 1.**

спектральная норма и мин (·) обозначить минимальное собственное значение. Если не указано иное, мы используем обозначение *O (·)* скрыть только абсолютные константы, которые не зависят ни от какого параметра задачи, и обозначения ~

O (·) скрыть только абсолютные константы и логарифмические факторы.

2.1 Допущения

На протяжении всей статьи мы предполагаем, что функция *е (*Икс) ограничен снизу некоторым оптимальным значением *е*.

Мы также делаем следующие предположения о гладкости функции:

Предположение 1. *Функция е (*Икс) *имеет*

- *Градиенты Липшица и Липшиц гессиан: для всех* Икс 1 *а также* Икс 2,

$$кр\ е\ (Икс_1)\ р\ е\ (Икс_2)\| к \cdot Икс_1\ Икс_2\ К;\ р_2\ е\ (Икс_1)\ р_2\ е\ (Икс_2) \cdot \to К Икс_1\ Икс_2\ к.$$

Приведенные выше предположения утверждают, что градиент и гессиан не могут резко измениться в небольшой локальной области, и являются стандартными в предыдущей работе по обходу седловых точек и нахождению локальных минимумов. Далее мы сделаем следующие предположения о дисперсии стохастических градиентов и стохастических гессианов:

Предположение 2. *Функция е (*Икс, *·) имеет*

- *для всех* Икс, Е час $кр\ е\ (Икс, \cdot) р\ е\ (Икс)\ К_2$ я $\cdot_2$ $_1$ *а также* $кр\ е\ (Икс, \cdot) р\ е\ (Икс)\ к \cdot М_1$ *в виде;*
- *для всех* Икс, Е час $р_2\ е\ (Икс, \cdot) р_2\ е\ (Икс)_2$ я $\cdot_2$ $_2$ *а также* $р_2\ е\ (Икс, \cdot) р_2\ е\ (Икс) \cdot М_2$ *в виде*

Мы отмечаем, что приведенные выше допущения не являются существенными для нашего результата и могут быть заменены любыми условиями, которые вызывают концентрацию. Более того, ограниченное гессианское предположение может быть удалено, если допустить, что *е (*Икс, *·)* имеет ` *о*- Градиенты Липшица для всех ·, что гарантирует концентрацию Гессе без дальнейшего предположения о дисперсии $р_2\ е\ (Икс, \cdot)$ [Tropp et al., 2015].

2.2 Кубическая регуляризация

Наша цель в этой статье - найти ✏- стационарная точка второго порядка, которую мы определяем следующим образом:

Определение 1. Для →- Функция Гессиана Липшица *е,* мы говорим, что Икс является ✏- *стационарная точка второго порядка (*или ✏- *приблизительный локальный минимум)* если

$$кр\ е\ (Икс) к \cdot ✏ \text{ а также } мин\ (р_2\ е\ (Икс)) \quad \overset{п \to ✏}{} \tag{2}$$

✏- Стационарная точка второго порядка не только имеет небольшой градиент, но также имеет гессиан, близкий к положительному полуопределенному. Таким образом, это часто также упоминается как ✏- приблизительный локальный минимум. В детерминированной среде кубическая регуляризация [Нестеров, Поляк, 2006] является классическим алгоритмом поиска стационарной точки второго порядка →- Функция Гессиана-Липшица *е (*Икс). В каждой итерации сначала формируется локальная верхняя граница функции с использованием разложения Тейлора третьего порядка вокруг текущей итерации. Икс т:

$$м_т(х) = е\ (Икс_т) + р\ е\ (Икс_т)>(хх_т) + 1 \quad \overline{2\ (хх_т)> р_2\ е\ (Икс_т)(хх_т) + \to} \quad \overline{6\ К хх_т\ К_3}.$$

Это называется *кубическая подмодель.* Затем кубическая регуляризация минимизирует эту кубическую подмодель для получения следующей итерации: $Икс_{т+1} = argmin_{Икс} м_т(Икс)$. Когда кубическая подмодель может быть решена точно, кубическая регуляризация требует $O \downarrow_{п \to (е(Икс_0) - е_*)/✏^{1,5}}$ итерации, чтобы найти ✏- стационарная точка второго порядка.

Чтобы применить этот алгоритм в стохастической установке, необходимо решить три проблемы: (1) у нас есть доступ только к стохастическим градиентам и гессианам, а не к истинному градиенту и гессиану; (2) наше единственное средство взаимодействия с гессианом - через произведения гессен-векторов; (3) кубическая подмодель не может быть решена точно на практике, только до некоторого допуска. Мы обсуждаем, как преодолеть каждое из этих препятствий в нашей статье.

3 Основные результаты

Мы начнем с мета-алгоритма стохастической кубической регуляризации общего назначения в алгоритме 1, который использует подпрограмму черного ящика для решения стохастических кубических подзадач. На высоком уровне, в

Алгоритм 1 Стохастическая кубическая регуляризация (мета-алгоритм)

Входные данные: размеры мини-партии N_1, N_2, инициализация Икс$_0$, количество итераций $T_{\text{макс}}$, и окончательная терпимость ✒.

1: за $\tau = 0, , , , , T_{\text{макс}}$ **делать**

2: Образец S_1 { ←$_{B_1}$ $_{я} = 1$, S_2 { ←$_{B_2}$ $_{я} = 1$.

3: г τ $\frac{1}{|S_1|}$ п ←$_{я \in S_1}$ p *e* (Икс $_\tau$, ←$_{я}$)

4: В $_\tau$ [·] $\frac{1}{|S_2|}$ п ←$_{я \in S_2}$ p_2 *e* (Икс $_\tau$, ←$_{я}$)(·)

5: , $_{м}$ Cubic-Subsolver (г $_\tau$, В $_\tau$ [·], ✒)

6: Икс $_{\tau+1}$ Икс $_{\tau+}$ Q ✒$_3$

7: если $_{м}$ $\frac{1}{100}$ → тогда

8: Cubic-Finalsolver (г $_\tau$, В $_\tau$ [·], ✒)

9: Икс ← Икс $_{\tau+}$

10: перемена

11: конец, если

12: конец для вывода: Икс ← если условие досрочного завершения было достигнуто, в противном случае финальная итерация *Икс* $_{T_{\text{макс}}+1}$.

Чтобы разобраться со случайными градиентами и гессианами, мы выберем две независимые миниатчи S_1 **а также** S_2 **на каждой итерации. Обозначая средний градиент и средний гессиан**

$$г_\tau = 1 \quad \frac{}{|S_1|} \text{Икс}_{\leftarrow_{я} S_1} \quad p\, e\,(\text{Икс}_\tau, \leftarrow_{я}), \quad В_\tau = 1 \quad \frac{}{|S_2|} \text{Икс}_{\leftarrow_{я} S_2} \quad p_2\, e\,(\text{Икс}_\tau, \leftarrow_{я}), \tag{3}$$

это подразумевает *стохастическая кубическая подмодель:*

$$м_\tau(\text{х}) = e\,(\text{Икс}_{\tau)} + (\text{хх}_\tau) > г_{\tau+1} \quad \frac{}{2\,(\text{хх}_\tau) > В_\tau(\text{хх}_\tau) + \rightarrow} \quad \frac{}{6\,K\,\text{хх}_\tau\,K_3}. \tag{4}$$

Хотя подзадача зависит от $В_\tau$, **отметим, что наш мета-алгоритм никогда явно не формулирует эту матрицу, предоставляя только доступ к подсольверу** $В_\tau$ **через гессиано-векторные произведения, которые мы обозначим** $В_\tau[\cdot]$: $р_{\text{г}}$ $р_д$. **Следовательно, мы предполагаем, что субсольвер выполняет градиентную оптимизацию для решения подзадачи, так как** $р\, м_\tau$ (Икс) **зависит от** $В_\tau$ **только через** $В_\tau[\text{хх}_\tau]$.

После выборки мини-пакетов для градиента и гессиана алгоритм 1 вызывает черный кубический субсольвер для оптимизации **стохастической подмодели** $м_\tau$ (Икс). **Подсольвер возвращает изменение параметра**

, примерный минимизатор подмодели вместе с соответствующим изменением

в значении подмодели, $м_{\text{с}} = м_\tau(\text{Икс}_{\tau+}) - м_\tau(\text{Икс}_\tau)$. **Затем алгоритм обновляет параметры, добавляя**

к текущей итерации и проверяет, $_{м}$ удовлетворяет условию остановки. Более подробно, подпрограмма Cubic-Subsolver **принимает вектор** г **и функция для вычисления гессиано-векторных произведений** $В[\cdot]$, **затем оптимизирует полином третьего порядка ~**

$$м\,(\text{знак равно} > г + {}_{12} > В[\quad - \qquad] +$$

$\frac{}{6\,кк_3}$. **Позволять** $^{?\,\text{знак равно}}$ аргмин ~ $м\,(\,)$ обозначим минимизатор этого многочлена. В общем,

subolver не может вернуть точное решение ?. Следовательно, мы терпим определенную степень субоптимальности:

Состояние 1. Для любой фиксированной малой константы c, Cubic-Subsolver (г, $В[\cdot]$, ✒) **заканчивается в** T (✒) **градиентные итерации (которые могут зависеть от** c), **и возвращает** удовлетворяющий по крайней мере одному из следующего:

1. Максимум $м\,(\,), f\,(\text{Икс}_{\tau+})\, e\,(\text{Икс}_\tau)) \cdot ⊠ > (\text{п} ✒_{3/} \rightarrow)$. (Дело 1)

2. $kk \cdot k? k \cdot c$ Q ✒ $\frac{}{\rightarrow}$ и если $к? к$ $\frac{12}{}$ п ✒$_{\rightarrow}$, тогда ~ $м\,(\,) \cdot \sim$ $м\,(?) + {}_{c}$ $\frac{}{12 \cdot \rightarrow к? к_3}$.

(Дело 2)

Первое условие выполняется при изменении параметра приводит к уменьшению подмодели и функции **оба достаточно большие (Дело 1). Если это не выполняется, второе условие гарантирует, что оно не слишком велико по отношению к истинному решению. и что кубическая подмодель решается с точностью**

$c \cdot \rightarrow к? к_3$ **когда** $к? к$ **большой (Случай 2).**

Как упомянуто выше, мы предполагаем, что субсольвер использует градиентную оптимизацию для решения подзадачи так, чтобы он получал доступ к гессиану только через произведения вектора гессиана. Соответственно, это может быть любой стандартный алгоритм первого порядка, такой как градиентный спуск, ускоренный градиент Нестерова.

Алгоритм 2 Куб-субсольвер с градиентным спуском

Входные данные: г, В [:], толерантность ✒.

1: если *K*г *K* тогда

2: *p c*

3: *p c*г

4: **еще**

5: 0, *c*0*p*

6: **гг + *i* за *i* ← Unif (S *d* 1)**

7: **за *т* = 1, , , , , *T* (✒) делать**

8: ⌘ (Ғ + б [ ] + → *KK*)

9: конец для

10: **конец, если**

11: *м* г> + 12 > В [ ] + → *KK*3

Вывод: , *м*

Алгоритм 3 Cubic-Finalsolver с градиентным спуском

Входные данные: г, В [:], толерантность ✒.

1: **0, г *м* г, ⌘**

2: **пока *K* г *м* *K*>✒** делать

3: ⌘ г *м*

4: **г *м* г + б [**] + → *KK*

5: конец при

выходе:

спуск и т. д. Градиентный спуск представляет особый интерес, поскольку может быть показано, что он удовлетворяет условию 1 (см. лемму 1).

После обзора нашего мета-алгоритма и проверки наличия подходящего субсольвера мы готовы представить нашу основную теорему:

Теорема 1. ***Существует абсолютная константа с такой, что если е (*Икс*) удовлетворяет предположениям 1, 2, CubicSubsolver*** ⌘, *а также* N_2

удовлетворяет условию 1 с с, $н_1$ **Максимум(** журнал

Максимум(журнал ⌘, *тогда для всех >* 0 *а также е*

*е (*Икс 0) *е←, Алгоритм 1 превзойдет*

***положить ✒- стационарная точка второго порядка е с вероятностью не менее* 1** *в пределах*

$$O\checkmark \quad \text{Максимум} \quad + \text{макс} \qquad (5)$$

оценки суммарного стохастического градиента и вектора Гессиана.

В пределе где ✒ мал, результат упрощает:

N 2 о, тогда в условиях теоремы 1 можно сделать вывод, что

Замечание 1 Если ✒ • мин

Алгоритм 1 выведет ✒- стационарная точка второго порядка *е* с вероятностью не менее 1 в пределах

$$O\checkmark \qquad (6)$$

оценки суммарного стохастического градиента и вектора Гессиана.

Теорема 1 утверждает, что после *T*out = ~ *O* ( ) итерации, стохастическая кубическая регуляризация **(Алгоритм 1) найдет ✒- стационарная точка второго порядка whp В каждой итерации нам требуется *N*1 = ~**

O ( ) **образцы для градиентного усреднения и *N*2 = ~** *O* ( ) образцы для гессиана-вектора

усреднение продукта. Напомним, что усредненный градиент г *т* фиксируется для данной кубической подмодели, в то время как усредненное произведение вектора Гессе В *т* *[* *v]* необходимо пересчитывать каждый раз, когда кубический субсольвер

запрашивает градиент $\rho \sim$ $м(\cdot)$. В большинстве $T(\epsilon)$ такие запросы будут сделаны по определению. Таким образом, каждый итерация занимает $\sim O(2^{\frac{1}{\epsilon^{2}+2}} \cdot T(\epsilon))$ стохастический градиент / оценка гессиано-векторного произведения при ϵ является маленький (см. замечание 1).

Наконец, отметим, что строки 8-11 Алгоритма 1 дают условие завершения нашего мета-алгоритма. При уменьшении значения подмодели $м$ слишком мала, наша теория гарантирует Икс $т+?$ является ϵ-стационарная точка второго порядка, где? является оптимальным решением кубической подмодели. Тем не менее, Cubic-Subsolver может дать только неточное решение удовлетворяющее Условию 1, которое не достаточно для Икс $т+$ быть ϵ- стационарная точка второго порядка. Поэтому мы называем Cubic-Finalsolver (который просто использует градиентный спуск и подробно описан в алгоритме 3) для решения подзадачи с более высокой точностью.

3.1 Градиентный спуск как кубический субсольвер

Одним конкретным примером кубического субсольвера является простой вариант градиентного спуска (Алгоритм 2), изученный в работе Кармона и Дучи [2016]. Два основных различия относительно стандартного градиентного спуска: (1) строки 1–3: когда г большой, подмодель $\sim$ $м()$ может быть плохо обусловлен, поэтому вместо делая градиентный спуск, итерация перемещается только на один шаг в г направление, которое уже гарантирует достаточный спуск; (2) строка 6: алгоритм добавляет небольшое возмущение к г чтобы избежать некоторого «трудного» случая для кубической подмодели. Мы отсылаем читателей к Carmon и Duchi [2016] для получения более подробной информации об алгоритме 2.

Адаптируя их результат для нашей установки, мы получаем следующую лемму, которая утверждает, что субсольвер градиентного спуска удовлетворяет нашему условию 1.

Лемма 1. *Существует абсолютная константа* c_0, *такой, что при одних и тех же предположениях* e (Икс) *и такой же выбор параметров* N_1, N_2 *как и в теореме 1, алгоритм 2 удовлетворяет условию 1 с вероятностью не менее* $1 - 0$ $c\,T(\epsilon) \cdot \sim O(\cdot)$.

В нашем следующем следствии применяется градиентный спуск (алгоритм 2) в качестве приближенного кубического субсольвера в нашем мета-алгоритме (алгоритм 1), который сразу же дает общее количество оценок градиента и вектора Гессиана для полного алгоритма.

Следствие 1. *При тех же настройках, что и теорема 1, если* $\epsilon \cdot$ мин $\left\{\frac{1}{c_1 M_1}, \frac{1}{c_{22} M_2}, \frac{4}{2}\right\}$ о, *и если мы будем создавать подпрограмма Cubic-Subsolver с алгоритмом 2, то с вероятностью, большей чем* $1 - 0$ *Алгоритм 1 выведет* ϵ- *стационарная точка второго порядка* e (Икс) *в пределах*

$$O\left(\frac{\sqrt{2}\,(f(x_1) - f^*)}{\epsilon^{1,5}} \left(\frac{1}{\epsilon^{2+2}} + \frac{2}{\rightarrow \epsilon} \cdot \log d\right)\right) \quad (7)$$

оценки суммарного стохастического градиента и вектора Гессиана.

Из следствия 1 мы видим, что доминирующий член в решении подмодели $\frac{1}{\epsilon^{2}}$ когда ϵ является достаточно мал, что дает общую сложность итерации $\sim O(\epsilon^{3,5})$ когда другие проблемно-зависимые параметры постоянны. Это улучшает $\sim O(\epsilon^{4}$ поли$(\log d))$ сложность достигается SGD.

Эскиз 4 доказательства

В этом разделе описаны основные шаги, необходимые для понимания и доказательства нашей основной теоремы (теорема 1). Мы описываем наш высокоуровневый подход и приводим эскизный пример в Приложении А в стохастической настройке, предполагающий доступ оракула к точному субсольверу. В случае неточного субсольвера и подробных доказательств мы отложим добавление Б. Напомним, что при итерации T алгоритма 1, стохастическая кубическая подмодель $м_т$ построен вокруг текущей итерации Икс $т$ с формой, приведенной в следующем уравнении, $м_т(x) = e(\text{Икс}_т) + (x - \text{Икс}_т)\nabla г_т +$

$\frac{1}{2}(x - \text{Икс}_т)\nabla^2 г_т(x - \text{Икс}_т) + \frac{\rho}{6}\|x - \text{Икс}_т\|^3$, где г $т$ а также В $т$ усредненные стохастические градиенты и Гессенцы. На высоком уровне мы покажем, что для каждой итерации выполняются следующие два утверждения:

Претензия 1 Если Икс $т+1$ это не ϵ- стационарная точка второго порядка e (Икс), кубическая субмодель имеет большой спуск $м_т(\text{Икс}_{т+1}) - м_т(\text{Икс}_т)$.

Претензия 2 Если кубическая субмодель имеет большой спуск $м_т(\text{Икс}_{т+1}) - м_т(\text{Икс}_т)$, тогда истинная функция также имеет большой спуск $e(\text{Икс}_{т+1}) - e(\text{Икс}_т)$.

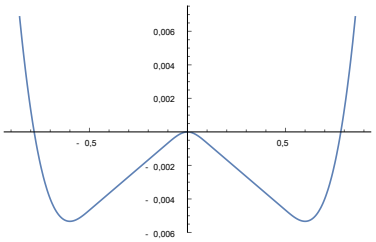

Рисунок 1: кусочно-кубическая функция *и (x)* используется вдоль одного из измерений в синтетическом эксперименте. Другое измерение использует масштабированный квадратик.

Учитывая эти претензии, очень просто поспорить за правильность нашего подхода. Мы знаем, что если мы наблюдаем большое уменьшение **значения кубической субмодели *м т* (Икс *т* +1) *м т* (Икс *т*) в течение алгоритма 1, затем по пункту 2 функция также будет иметь большой спуск. Но с тех пор *е* ограничен снизу, это не может происходить бесконечно, поэтому в конечном итоге мы должны встретить итерацию с малым кубическим подмодельным спуском. Когда это произойдет, мы можем сделать вывод по пункту 1, что Икс *т* +1 является ε- стационарная точка второго** порядка. Отметим, что утверждение 2 особенно важно в стохастической установке, поскольку у нас больше нет доступа к истинной функции, а **есть только подмодель. Утверждение 2 гарантирует, что прогресс в *м т* по-прежнему указывает на прогресс в *е,* позволяя алгоритму завершиться** в нужное время. Более подробный эскиз и полные доказательства можно найти в Приложении.

5 экспериментов

В этом разделе мы представляем эмпирические результаты на синтетических и реальных наборах данных, чтобы продемонстрировать эффективность нашего подхода. Все эксперименты выполняются с использованием TensorFlow [Abadi et al., 2016], который позволяет эффективно вычислять произведения вектора Гессиана с использованием метода, описанного Pearlmutter [1994].

5.1 Синтетическая невыпуклая проблема

Мы начнем с построения невыпуклой задачи с седловой точкой, чтобы сравнить предложенный нами подход со стохастическим градиентным спуском. **Позволять *и (x)* быть W-образной скалярной функцией, изображенной на рисунке 1, с локальным максимумом в начале координат и двумя локальными** минимумами с каждой стороны. Мы откладываем точную форму
***и (x)* к Приложению D. Мы стремимся**

решить проблему

$$\min_{\text{Икс } 2 \in \mathbb{R}^2} W(X_1) + 10\, \text{Икс}_{22},$$

с независимым шумом, взятым из *N* (0, 1) добавляется отдельно к каждому компоненту каждого градиента и оценки вектора Гессиана. По построению целевая функция имеет седловую точку в начале координат с собственными значениями Гессе -0,2 и 20, что обеспечивает простой, но сложный тестовый пример. Мы применили наш метод и SGD к этой проблеме, нанося на график объективное значение в зависимости от количества вызовов оракула. Размеры партий и скорости обучения для каждого метода настраиваются отдельно для обеспечения справедливого сравнения; см. Приложение D для деталей. Мы видим, что наш метод способен избежать седловой точки в начале координат и сходиться к одному из глобальных минимумов быстрее, чем SGD.

5.2 Глубокий автоэнкодер

В дополнение к описанной выше синтетической проблеме мы также исследуем эффективность нашего подхода к более практической проблеме глубокого обучения, а именно к обучению глубокому автоэнкодеру на MNIST [LeCun and Cortes, 2010]. Наша архитектура состоит из полностью подключенного кодера с размерами
(28 ·» 28) 512! 256! 128! 32 вместе с симметричным декодером. Мы используем нелинейность softplus (определенную как Softplus (*x)* = log (1 + exp (*Икс*))) для каждого скрытого слоя - поэлементная сигмоида для конечного слоя и попиксельно 2 потеря между входом и восстановленным выходом как наша целевая функция. Результаты этой задачи автоматического кодирования представлены на рисунке 3. Помимо обучения модели с помощью нашего метода и SGD, мы также включаем результаты с использованием AdaGrad, популярного адаптивного метода первого порядка с высокими эмпирическими характеристиками [Duchi et al., 2011], вместе с результатами для метода, объединяющего уменьшение дисперсии, и информацией второго порядка, предложенной Reddi et al. [2017]. Более подробную информацию о нашей экспериментальной установке можно найти в Приложении D.

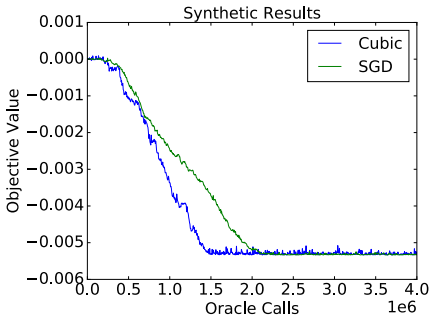

Рисунок 2: Результаты нашей синтетической невыпуклой задачи оптимизации. Стохастическая кубическая регуляризация избегает седловой точки в начале координат и сходится к глобальному оптимуму быстрее, чем SGD.

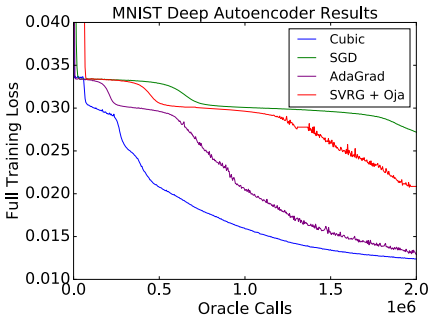

Рисунок 3: Результаты выполнения задачи глубокого автокодирования MNIST. В задаче оптимизации присутствуют несколько седловых точек. Стохастическая кубическая регуляризация способна ускользнуть от них быстрее, что позволяет достичь локального минимума быстрее, чем другие методы.

Мы наблюдаем, что стохастическая кубическая регуляризация быстро выходит за пределы двух седловых точек и опускается к локальному оптимуму, в то время как AdaGrad требуется в два-три раза дольше, чтобы избежать каждой седловой точки, а SGD еще медленнее. Это демонстрирует, что включение информации о кривизне с помощью векторно-гессианских продуктов может помочь на практике избежать седловых точек. Гибридный метод, состоящий из SVRG, чередующийся с алгоритмом Oja для гессенского спуска, также улучшает SGD, но не соответствует производительности нашего метода или AdaGrad.

6. Заключение

В этой статье мы представили стохастический алгоритм, основанный на классическом кубически-регуляризованном методе Ньютона для невыпуклой оптимизации. Наш алгоритм доказуемо находит ε-приблизительные локальные минимумы в $O(\epsilon^{-3.5})$ оценки общего градиента и вектора Гессе, улучшая ~ $O(\epsilon^{-4}$ поли$(\log d))$ курс SGD. Наши эксперименты показывают благоприятную эффективность нашего метода относительно SGD как в синтетической, так и в реальной проблеме.

Ссылки

Мартин Абади, Пол Бархам и др. Тензорный поток: система для крупномасштабного машинного обучения. В *12 USENIX Symposium по разработке и внедрению операционных систем,* стр. 265–283, 2016. Наман Агарвал, Зейуан Аллен-Чжу,

Брайан Буллинс, Эльад Хазан и Тэньюма. Нахождение приблизительного локальные минимумы быстрее градиентного спуска. В *Материалы 49-го ежегодного симпозиума ACM SIGACT по теории вычислений,* 2017.

Зеюань Аллен-Чжу. Наташа 2: Быстрее невыпуклая оптимизация, чем SGD. *Препринт arXiv Arxiv: 1708,08694,* 2017.

Зеюань Аллен-Чжу и Юаньчжи Ли. Neon2: поиск локальных минимумов с помощью оракулов первого порядка. *Arxiv Препринт arXiv: 1711.06673,* 2017.

Яир Кармон и Джон Дючи. Градиентный спуск эффективно находит кубически-регуляризованные невыпуклые Шаг Ньютона. *Препринт arXiv arXiv: 1612.00547,* 2016.

Яир Кармон, Джон Дючи, Оливер Хиндер и Аарон Сидфорд. Ускоренные методы для невыпуклых оптимизация. *Препринт arXiv arXiv: 1611.00756,* 2016.

Коралия Картис, Николас Гулд и Филипп Тоинт. Адаптивные методы кубической регуляризации для оптимизация без ограничений. Часть II: сложность оценки функций и производных в худшем случае. *Математическое программирование,* 130 (2): 295–319, 2011.

Анна Чороманска, Микаэль Хенафф, Майкл Матье, Жерар Бен Арус и Янн ЛеКун.
Поверхности потерь многослойных сетей. В *Материалы восемнадцатой Международной конференции по искусственному интеллекту и статистике,* 2015.

Фрэнк Э Кертис, Даниэль П Робинсон и Мохаммадреза Самади. Алгоритм области доверия с
сложность итерации в худшем случае $O(\epsilon^{-3/2})$ для невыпуклой оптимизации. *Математическое программирование,* 162 (1-2): 1–32, 2017.

Джон К. Дучи, Элад Хазан и Йорам Сингер. Адаптивные субградиентные методы для онлайн-обучения
и стохастическая оптимизация. *Журнал исследований машинного обучения,* 12: 2121–2159, 2011. Ронг Ге, Фуронг Хуан,

Чи Джин и Ян Юань. Спасаясь от седловых точек - онлайн стохастик
градиент для тензорного разложения. В *Материалы 28-й конференции по теории обучения,*
2015.

Ронг Гэ, Чи Джин и Йи Чжэн. Никаких ложных локальных минимумов в невыпуклых задачах низкого ранга: А
единый геометрический анализ. В *Материалы 34-й Международной конференции по машинному обучению,* 2017.

Пратик Джайн, Чи Джин, Шам М. Какаде, Пранит Нетрапалли и Аарон Сидфорд. Потоковая PCA:
Матрица соответствия Бернштейна и почти оптимальная конечная выборка гарантируют алгоритм Оя. В
Конференция по теории обучения, страницы 1147–1164, 2016.

Чи Джин, Ронг Ге, Пранит Нетрапалли, Шам М. Какаде и Майкл И. Джордан. Как убежать
Седловые точки эффективно. В *Материалы 34-й Международной конференции по машинному обучению,*
2017.

Ри Джонсон и Тонг Чжан. Ускорение стохастического градиентного спуска с использованием прогнозирующей дисперсии
снижение. В *Достижения в нейронных системах обработки информации,* 2013.

Йонас Мориц Колер и Аурелиен Луччи. Подвыборка кубической регуляризации для невыпуклых опти-
mization. В *Материалы 34-й Международной конференции по машинному обучению,* 2017.

Ян ЛеКун и Коринна Кортес. MNIST База рукописных цифр, 2010. URL Http: // Янн.
lecun.com/exdb/mnist/.

Юрий Нестеров. *Вводные лекции по выпуклой оптимизации: базовый курс,* Том 87. Спрингер
Наука и Бизнес Медиа, 2013.

Юрий Нестеров и Борис Т Поляк. Кубическая регуляризация метода Ньютона и его глобальные характеристики.
Математическое программирование, 108 (1): 177–205, 2006. Barak A Pearlmutter. Быстрое точное умножение на гессиан. *Нейронные*

вычисления, 6 (1): 147-160,
1994.

Sashank J Reddi, Manzil Zaheer, et al. Общий подход к выходу из седловых точек. *Препринт arXiv*
Arxiv: 1709,01434, 2017.

Герберт Роббинс и Саттон Монро. Метод стохастической аппроксимации. *Анналы математики*
Статистика, страницы 400–407, 1951.

Клемент В. Ройер и Стивен Дж. Райт. Анализ сложности алгоритмов поиска строк второго порядка
для плавной невыпуклой оптимизации. *Препринт arXiv arXiv: 1706.03131,* 2017.

Джу Сун, Цин Цюй и Джон Райт. Полное восстановление словаря по сфере I: обзор и
геометрическая картина. *IEEE Труды по теории информации,* 2016a. Джу Сун, Цин Цюй и Джон Райт.

Геометрический анализ фазы поиска. В *IEEE International*
Симпозиум по теории информации. IEEE, 2016b.

Джоэл А. Тропп и соавт. Введение в матричные неравенства концентрации. *Основы и тенденции* в
в машинном обучении, 8 (1-2): 1–230, 2015.

Пэн Сюй, Фарбод Руста-Хорасани и Майкл В.М. Методы ньютоновского типа для невыпуклых
оптимизация под неточной гессенской информацией. *Препринт arXiv arXiv: 1708.07164,* 2017.

И Сюй и Тяньбао Ян. Стохастические алгоритмы первого порядка для выхода из седловых точек почти
линейное время *Препринт arXiv arXiv: 1711.01944,* 2017.