SpiderBoost и Natasha 2 – продвинутые алгоритмы решения задачи оптимизации

15 мая 2020 г.

1 Введение

1.1 Постановка задачи

SpiderBoost и Natasha 2 - недавно разработанные стохастические алгоритмы решения задач оптимизации.

Оба алгоритма предусматривают рассмотрение задачи:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) = \mathbb{E}_i[f_i(x)] \quad (1)$$

где $\{f_i(x)\}_{i=1}^n$ - гладкие, в общем случае невыпуклые функции

1.2 Мотивация

Известно, что для задачи поиска ϵ-приближенного локального минимума произвольной гладкой невыпуклой функции, наилучшую скорость сходимости $O(\epsilon^{-4})$ предоставляет SGD.

Также известно, что при нахождении ϵ-приближенной стационарной точки SGD требует $O(\min(n\epsilon^{-2}, \epsilon^{-4}))$ операций вычисления градиента (gradient-cost).

Для конечной суммы $\sum_{i=1}^n f_i(x)$ известен алгоритм SCSG, повышающий эффективность до $\tilde{O}(\min(n^{2/3}\epsilon^{-2}, \epsilon^{-10/3}))$.

Natasha 2 предполагает скорость сходимости $O(\epsilon^{-3.25})$ для нахождения ϵ-приближенного локального минимума произвольной гладкой невыпуклой функции, пользуясь информацией о стохастических градентах в качестве оракула.

SpiderBoost предполагает скорость сходимости $O(\min(n^{1/2}\epsilon^{-2}, \epsilon^{-3}))$, оптимальный результат для режима $n \leq O(\epsilon^{-4})$

Далее более детально рассмотрены алгоритмы SpiderBoost и Natasha 2, их реализация, а также сравнение между собой и с другими известными алгоритмами оптимизации.

2 SpiderBoost

2.1 Общие сведения

SpiderBoost – один из стохастических алгоритмов оптимизации основанных на методе Стохастической Интегрированной Дифференциальной Оценки (Stochastic Path-Integrated Differential Estimator – SPIDER). Подробнее о SPIDER в разделе **2.3 SPIDER - принцип работы**

2.2 Постановка задачи

Рассматривается функция

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{n}\Sigma_{i=1}^{n} f_i(\mathbf{x}) \quad (1.1)$$

где функции $f_i(\mathbf{x})$ в общем случае невыпуклые.

Также пусть для функции выполнены ограничения:
1. $\exists \inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) > -\infty$
2. $\forall i \in 1\dots n, \forall x, y \in \mathbb{R}^d, \|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x-y\|$

Требуется найти такой вектор x, что

$$\hat{x} \in \mathbb{R}^d : \hat{x} \in \mathbb{R}^d : \|\nabla f(\hat{x})\| \le \varepsilon$$

Т.е. ищется ϵ-приближение стационарной точки липшецевой ограниченной снизу гладкой функции.

2.3 SPIDER - принцип работы

SpiderBoost – одна из реализаций метода SPIDER вспомогательной функции оценки:

SPIDER – метод первого порядка, в свою очередь основанный на аппарате SGD, за исключением следующих особенностей:

1. На некоторых итерациях будем просчитывать градиент полностью, как в алгоритме GD
2. На всех остальных шагах (таких будет большинство) мы станем отслеживать предварительно введенную функцию оценки ν, зависящую от градиента мини-батча (вместо самого градиента, как предполагает SGD)

Наша задача добиться таким образом, с одной стороны, хорошей точности результата посредством (1), и, с другой стороны, высокой производительности посредством (2).

2.4 Оценка точности SPIDER

1. Пусть $B(\mathbf{x})$ отображает $\mathbf{x} \in \mathbb{R}^d$ на случайную функцию $B_i(\mathbf{x})$: в алгоритмах под $B(x)$ будет подразумеваться градиент

2. Пусть функция ν такова, что

$$\mathbb{E}[B(\mathbf{x}^k) - B(\mathbf{x}^{k-1}) \mid \mathbf{x}_{0:k}] = \nu^k - \nu^{k-1}$$

где $\mathbf{x}_{0:k} = \mathbf{x}_1 \dots \mathbf{x}_k$

3. Пусть также выполнено

$$\mathbb{E}\|B_i(x) - B_i(y)\|^2 \le L_B^2\|x-y\|^2$$

для $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| \le \epsilon_1$, $\forall k \in 1\dots K$

4. Наконец, пусть

$$\nu : \nu^k = B_{S_*}(\mathbf{x}^k) - B_{S_*}(\mathbf{x^{k-1}}) + \nu^{k-1}$$

здесь S_* – мощность батча
$\Rightarrow$тогда имеет место **оценка для ошибки $\nu^{\mathbf{k}}$ в терминах дисперсии**

$$\mathbb{E}\|\nu^k - B(\mathbf{x}^k)\|^2 \leq \frac{kL_B^2\epsilon_1^2}{S_*} + \mathbb{E}\|\nu^0 - B(\mathbf{x}^0)\|^2$$

т.е. при удачном выборе функции ν, а также шага в зависимости от L_B значение построенной функции ν^k с хорошей точностью приближается к значению, получаемом при GD: $B(\mathbf{x}^k)$; и есть предпосылка сэкономить на скорости вычисления ν вместо вычисления полного градиента. Скорости сходимости метода зависят от выбора функции ν, в следующей секции рассмотрим алгоритм, в котором ν хорошо подобрана:

2.5 SpiderBoost - описание алгоритма

1. Рассматриваем задачу (1.1) с соблюдением всех дополнительных ограничений

2. Пользуемся методом Spider, выбрав вспомогательную функцию ν:
   1) $\nu_0 = \nabla f(x_0)$
   2) $\nu_k = \frac{1}{|S|}\Sigma_{i\in S}[\nabla f_i(x_k) - \nabla f_i(x_{k-1}) + \nu_{k-1}]$

3. Опишем псевдокод для SpiderBoost:
– **input** : $\eta = \frac{1}{2L}; q, K, \mid S \mid \in \mathbb{N}$
– **for** $k \in 1 \dots K$:
– – **if** $\mod (k,q) = 0 \Rightarrow \nu_k = \nabla f(x_k)$
– – **else** $\Rightarrow$ **draw** $\mid S \mid$ samples with replacement, **compute** ν_k
– **calculate** $x_{k+1} = x_k - \eta\nu_k$
– **return** $x_\xi, \xi \overset{Unif}{\sim} \{0 \dots K-1\}$

3 Natasha 2

3.1 Постановка задачи

Исследуется задача

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) = \mathbb{E}_i[f_i(x)],$$

где $\{f_i(x)\}_{i=1}^n$ - в общем случае гладкие невыпуклые функции

Необходимо найти точку $\hat{x} \in \mathbb{R}^d$, такую что $\|\nabla f(\hat{x})\| \le \varepsilon$ и $\nabla^2 f(\hat{x}) \succeq -\delta \mathbf{I}$

3.2 Предположения о виде функции

1. f имеет ограниченную дисперсию, откуда $\mathbb{E}_i[\|\nabla f_i(x) - \nabla f(x)\|^2] < \nu$
2. L-Липшицевость градиента: $\|\nabla f(x) - \nabla f(y)\| \le L \cdot \|x - y\|$
3. L_2-Липшицевость гессиана: $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L_2 \cdot \|x - y\|$

3.3 Описание алгоритма

Основная идея: избегать седловых точек, используя для этого информацию из гессиана

На каждой итерации в точке y_k сравниваем минимальное собственное значение $\lambda_{min}(\nabla^2 f(y_k))$ и $-\delta$.

1. Если $\exists v : v^T \nabla^2 f(y_k) v \le -\frac{\delta}{2}$, то пытаемся отдалиться от седла: $y_{k+1} = y_k \pm \frac{\delta}{L_2} v$ ($+$ или $-$ выбираем случайно). Для поиска v пользуемся алгоримом Oja с $\Theta(\frac{L^2}{\delta^2}\log(dk))$ итерациями

2. Иначе берём $F^k(x) = f(x) + L(\max\{0, \|x - y_k\| - \frac{\delta}{L_2}\})^2$, которая является $5L$-Липшицевой по градиенту и 3δ-невыпуклой ($\nabla^2 F^k(x) \succeq -3\delta\mathbf{I}$); ищем для неё точку $\widehat{x} : \|\nabla F^k(\hat{x})\| \le \varepsilon$ и переходим туда: $y_{k+1} = \hat{x}$. Как видно, F^k штрафует за выход из безопасной зоны $\{x : \|x - y_k\| \le \frac{\delta}{L_2}\}$. Для поиска $\hat{x}$ запускаем Natasha 1.5 на одну эпоху

Цикл обрывается, когда было совершено N_1 шагов первого порядка, и алгоритм выдает последнюю рассмотренную точку y_k

Псевдокод:

Algorithm 5 `Natasha2`$^{\mathsf{full}}(f, y_0, \varepsilon, \delta)$

Input: function $f(x)$ satisfying Problem (7.1), starting vector y_0, target accuracy $\varepsilon > 0$ and $\delta > 0$.
◇ *assume $\mathcal{V} \ge \Omega(\varepsilon^2)$ and $\delta^4 \le O(\mathcal{V}\varepsilon L_2^3/L^2)$*

if $L_2 \ge \frac{L\delta}{\mathcal{V}^{1/3}\varepsilon^{1/3}}$ **then** ◇ *the boundary case for large L_2*
  $\widetilde{L} = \widetilde{\sigma} \leftarrow \Theta(\frac{L_2 \mathcal{V}^{1/3}\varepsilon^{1/3}}{\delta}) \in [L, \infty)$.
else ◇ *the most interesting case*
  $\widetilde{L} \leftarrow L$ and $\widetilde{\sigma} \leftarrow \Theta\big(\max\{\frac{\mathcal{V}\varepsilon L_2^3}{L^2\delta^3}, \frac{\varepsilon L}{\mathcal{V}^{1/2}}\}\big) \in [\delta, L]$.
end if
$B \leftarrow \Theta(\mathcal{V}/\varepsilon^2)$; $p \leftarrow \Theta\big((\frac{\widetilde{\sigma}^2}{\widetilde{L}^2}B)^{1/3}\big)$ $\alpha \leftarrow \Theta(\frac{\widetilde{\sigma}}{p^2\widetilde{L}^2})$. ◇ *same p and α as* `Natasha1.5`$^{\mathsf{prune}}$
$X \leftarrow []$; $N_1 \leftarrow \Theta\big(\frac{\widetilde{\sigma}\Delta_f}{p\varepsilon^2}\big)$, where Δ_f is an upper bound on $f(y_0) - \min_y\{f(y)\}$.
for $k \leftarrow 0$ **to** ∞ **do**
  Apply Oja's algorithm to find minEV of $\nabla^2 f(y_k)$. ◇ *use Lemma 5.3 with $T_{\mathrm{oja}} = \Theta(\frac{L^2}{\delta^2}\log(dk))$*
  2nd-order step:
  if $v \in \mathbb{R}^d$ is found s.t. $v^\top \nabla^2 f(y_k) v \le -\frac{\delta}{2}$ **then**
    $y_{k+1} \leftarrow y_k \pm \frac{\delta}{L_2} v$ where the sign is random.
  else ◇ *it satisfies $\nabla^2 f(y_k) \succeq -\delta\mathbf{I}$ w.p. $\ge 1 - \frac{1}{20(k+1)^2}$, see Lemma 5.3*
    1st-order step:
    $F(x) = F^k(x) \stackrel{\text{def}}{=} f(x) + L(\max\{0, \|x - y_k\| - \frac{\delta}{L_2}\})^2$.
    ◇ *$F(\cdot)$ is $O(\widetilde{L})$-smooth and $O(\widetilde{\sigma})$-nonconvex*
    $(\widehat{y}_k, y_{k+1}) \leftarrow$ `Natasha1.5`$^{\mathsf{full}}(F, y_k, B, p, 1, \alpha)$
    $X \leftarrow [X, (y_k, \widehat{y}_k)]$.
    break the for loop if have performed N_1 first-order steps.
  end if
end for
$(y, \widehat{y}) \leftarrow$ a random pair in X. ◇ *in practice, letting x^{out} = the last $\widehat{y}_k$ should be good enough*

3.4 Составляющие

3.4.1 Natasha 1.5 : Поиск стационарных точек для σ-невыпуклых функций

Каждая эпоха делится на p суб-эпох со стартовым вектором $\widehat{x}$.

При подсчёте градиентов, вместо $f(x)$ фактически используется $f(x)+\sigma\|x-\widehat{x}\|^2$, что должно стабилизировать алгоритм, немного замедляя его. Вместе с этим, используется пересчёт градиента из алгоритма SVRG. Алгоритм выдает последнюю точку $\hat{x}$

Псевдокод :

Algorithm 3 Natasha1.5$^{\text{full}}(F, x^{\varnothing}, B, p, T', \alpha)$

Input: function $F(\cdot)$ satisfying Problem (6.1), starting vector $x^{\varnothing}$, epoch length $B \in [n]$, sub-epoch count $p \in [B]$, epoch count $T' \geq 1$, learning rate $\alpha > 0$. ◇ *p should be $\Theta((\sigma^2 B/L^2)^{1/3})$*

Output: two vectors $\widehat{\mathrm{y}}$ and y^+.

1: $\widehat{\mathrm{x}} \leftarrow x^{\varnothing}$; $m \leftarrow B/p$; $X \leftarrow []$;
2: **for** $k \leftarrow 1$ **to** T' **do** ◇ *T' epochs each of length B*
3: $\quad\widetilde{\mathbf{x}} \leftarrow \widehat{\mathrm{x}}$; $\mu \leftarrow \frac{1}{B}\sum_{i\in S}\nabla f_i(\widetilde{\mathbf{x}})$ where S is a uniform random subset of $[n]$ with $|S| = B$;
4: $\quad$**for** $s \leftarrow 0$ **to** $p-1$ **do** ◇ *p sub-epochs each of length m*
5: $\quad\quad x_0 \leftarrow \widehat{\mathrm{x}}$; $X \leftarrow [X, \widehat{\mathrm{x}}]$;
6: $\quad\quad$**for** $t \leftarrow 0$ **to** $m-1$ **do** ◇ *m iterations in each sub-epoch*
7: $\quad\quad\quad i \leftarrow$ a random index from $[n]$.
8: $\quad\quad\quad \widetilde{\nabla} \leftarrow \nabla f_i(x_t) - \nabla f_i(\widetilde{\mathbf{x}}) + \mu + 2\sigma(x_t - \widehat{\mathrm{x}})$
9: $\quad\quad\quad x_{t+1} = \arg\min_{y\in\mathbb{R}^d}\left\{\psi(y) + \frac{1}{2\alpha}\|y - x_t\|^2 + \langle\widetilde{\nabla}, y\rangle\right\}$
10: $\quad\quad$**end for**
11: $\quad\quad \widehat{\mathrm{x}} \leftarrow$ a random choice from $\{x_0, x_1, \ldots, x_{m-1}\}$; ◇ *in practice, choose the average*
12: $\quad$**end for**
13: **end for**
14: $\widehat{\mathrm{y}} \leftarrow$ a random vector in X and $y^+ \leftarrow \widehat{\mathrm{x}}$. ◇ *in practice, choose the last*

(в нашем случае $\psi(x) \equiv 0$, поэтому строчка 9 будет выглядеть как $x_{t+1} = x_t - \alpha\tilde{\nabla}$)

3.4.2 Oja's algorithm : быстрый поиск максимального собственного вектора

Пусть имеется некоторое распределение $\mathcal{D}$ симметричных матриц, собственные числа которых лежат на $[0,1]$ и $\mathbf{B} = \mathbb{E}_{\mathbf{A}\sim\mathcal{D}}[\mathbf{A}]$ - среднее. Пусть $\mathbf{A}_1, \ldots \mathbf{A}_T \sim \mathcal{D}$. Алгоритм начинает со случайного гауссовского вектора $w_1 \in \mathbb{R}^d$ единичной нормы. Затем на каждой итерации $k = 2, \ldots T$ подсчитывается $w_k = \frac{1}{C}(\mathbf{I} + \eta\mathbf{A}_{k-1})w_{k-1}$, так чтобы $\|w_k\| = 1$. Параметр η выбирается, так чтобы $\eta = \Theta(\sqrt{T})$ Алгоритм выдаёт случайный вектор из равномерного распределения $\{w_1, \ldots w_T\}$

В нашем случае $\mathcal{D} = \{-\frac{1}{L}\nabla^2 f_i(x)\}_i$, а итерации выглядят как

$$w_k = \frac{1}{CL}(Lw_{k-1} - \eta\nabla^2 f_{k-1}(x)w_{k-1})$$

4 Natasha 2 и SpiderBoost: сравнение

4.1 Принципиальные отличия

Natasha 2 и SpiderBoost - недавно разработанные стох- алгоритмы решения задачи (1)

Алгоритмы отличаются друг от друга принципом работы: идея алгоритма Natasha 2 заключается в избежании седловых точек засчет информации гессиана, идея SpiderBoost – построение вспомогательной оценки требующей на пересчет меньше операций чем градиент.

Подробнее с техническими отличиями алгоритмов моджно ознакомиться в таблице 1:

Таблица 1: сравнение характеристик Natasha 2 и SpiderBoost

Характеристика	**SpiderBoost**	**Natasha 2**
Порядок метода	I	II
Искомый вектор	$x \in \mathbb{R}^d : \hat{x} \in \mathbb{R}^d : \|\nabla f(\hat{x})\| \leq \varepsilon$	$x \in \mathbb{R}^d : \|\nabla f(\hat{x})\| \leq \varepsilon; \nabla^2 f(\hat{x}) \succeq -\delta \mathbf{I}$
Вид целевой функции	• $\exists \inf_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) > -\infty$ • $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ $\forall i \in 1 \dots n, \forall x, y \in \mathbb{R}^d$	• $\exists \nu : \mathbb{E}_i[\|\nabla f_i(x) - \nabla f(x)\|^2] < \nu$ • $\|\nabla f(x) - \nabla f(y)\| \leq L \cdot \|x - y\|$ • $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_2 \cdot \|x - y\|$
Основная идея	Стохастический спуск по данным собственной функции ν вместо градиента	Избегать седловых точек пользуясь информацией из Гессиана
Порядок сходимости	$O(\min(n^{\frac{1}{2}}\epsilon^{-2}, \epsilon^{-3}))$	$\tilde{O}(\frac{1}{\delta^5} + \frac{1}{\delta\epsilon^3} + \frac{1}{\epsilon^{3.25}})$
Шаг алгоритма	$O(\epsilon^{\frac{2}{3}} L^{-\frac{2}{3}})$	$O(L^{-1})$

4.2 Предположения о приоритете методов

Исходя из данных таблицы, можно предположить, что использование Natasha 2 выгодно для функций с наличием большого числа седловых точек, а также в случаях с пониженным δ – параметром, задающим норму гессиана. SpiderBoost, с другой стороны, предположительно имеет приоритет при стремлении к $n \leq O(\epsilon^{-4})$ когда число функций относительно невелико.

В следующем разделе мы приведем реализацию алгоритмов и проверку гипотез.

5 Natasha 2 и SpiderBoost: реализация алгоритмов

5.1 Проведение экспериментов

По ссылкам ниже можно ознакомиться с реализацией алгоритмов Natasha 2 и SpiderBoost:
-Реализация SpiderBoost
-Реализация Natasha 2

5.2 Результаты экспериментов

По результатам опытов мы видим, что
Это связано с тем, что
Данные результаты согласуются с теоретическими предположениями в некоторой степени:

6 Заключение

6.1 Natasha 2 и SpiderBoost среди других алгоритмов оптимизации

Мы ознакомились с результатами работы алгоритмов Natasha 2 и SpiderBoost. Приведем ниже сравнение сложностей исследуемых алгоритмов с другими известными алгоритмами решения оптимизационной задачи.

Таблица 2: Сравнение сложности различных алгоритмов оптимизации

Алгоритм	**Сложность**
Natasha 2	$\tilde{O}(\frac{1}{\delta^5} + \frac{1}{\delta\epsilon^3} + \frac{1}{\epsilon^{3.25}})$
SpiderBoost	$O(\min(n^{\frac{1}{2}}\epsilon^{-2}, \epsilon^{-3}))$
Mirror Descent	$f(x_k) - f(x^*) = O(L_f \frac{\log k}{k})$
Nesterov Momentum	$f(x_k) - f(x^*) = O\big(L_f \|x_k - x_0\|_2^2 (1 - \frac{1}{\sqrt{k}})^k\big)$
Newton	• Сходимость: $$\|x_{k+1} - x^*\|_2 \leq \frac{L_{Hess}\|x_k - x^*\|_2^2}{2(\delta - L_{Hess}\|x_k - x^*\|_2)}$$ • Сложность: $O(n^3)$

6.2 Вывод

Мы рассмотрели стохастические алгоритмы решения оптимизационных задач Natasha 2 и SpiderBoost. Были также проведены теоретические и практические сравнения алгоритмов, по результатом которых мы видим, что: