

SpiderBoost и Natasha 2 – продвинутые алгоритмы решения задачи оптимизации

25 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 локального минимума произвольной гладкой невыпуклой функции (в смысле нахождения такого вектора $x : \nabla f(x) \leq \epsilon$) наилучшую скорость сходимости $O(\epsilon^{-4})$ предоставляет SGD.

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

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

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. $\forall i \in 1 \dots n, \forall x, y \in \mathbb{R}^d, \|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$

Требуется найти такой вектор $\hat{\mathbf{x}}$, что

$$\hat{\mathbf{x}} \in \mathbb{R}^d : \|\nabla f(\hat{\mathbf{x}})\| \leq \epsilon$$

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

$$\mathbb{E}[B_i(\mathbf{x}_k) - B_i(\mathbf{x}_{k-1}) \mid \mathbf{x}_{0:k}] = \nu_k - \nu_{k-1}$$

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

3. Пусть также выполнено

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

для $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \leq \epsilon_1, \forall k \in 1 \dots K$

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 ν^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, получаемому 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 :

$$1) \nu_0 = \nabla f(x_0)$$

$$2) \nu_k = \frac{1}{|S|} \sum_{i \in S} [\nabla f_i(x_k) - \nabla f_i(x_{k-1}) + \nu_{k-1}]$$

3. Опишем псевдокод для SpiderBoost:

– **input** : $\eta = \frac{1}{2L}$; $q, K, |S| \in \mathbb{N}$

– **for** $k \in 1 \dots K$:

– – **if** $\text{mod}(k, q) = 0 \Rightarrow \nu_k = \nabla f(x_k)$

– – **else** \Rightarrow **draw** $|S|$ samples with replacement, **compute** ν_k

– **calculate** $x_{k+1} = x_k - \eta \nu_k$

– **return** $x_\xi, \xi \stackrel{Unif}{\sim} \{0 \dots K-1\}$

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 $\|\nabla f(\hat{x})\| \leq \epsilon$ 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: $\|\nabla f(x) - \nabla f(y)\| \leq L \cdot \|x - y\|$
3. L_2 -Липшицевость гессиана: $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_2 \cdot \|x - y\|$

3.3 Описание алгоритма

3.3.1 Основная идея:

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

1. Если $\exists v : v^T \nabla^2 f(y_k) v \leq -\frac{\delta}{2}$, то пытаемся отдалиться от седла: $y_{k+1} = y_k \pm \frac{\delta}{L_2} v$ (+ или - выбираем случайно). Для поиска v пользуемся алгоритмом Ожа 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 $\hat{x} : \|\nabla F^k(\hat{x})\| \leq \epsilon$ и переходим туда: $y_{k+1} = \hat{x}$. Как видно, F^k штрафует за выход из безопасной зоны $\{x : \|x - y_k\| \leq \frac{\delta}{L_2}\}$. 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

3.3.2 Псевдокод:

Input: функция $f(x)$, удовлетворяющая 3.1 и 3.2, начальный вектор y_0 , целевое приближение $\epsilon > 0$, $\delta > 0$

// Также предполагаем $\nu \geq \Omega(\epsilon^2)$ и $\delta^4 \leq O(\nu \epsilon L_2^3 / L^2)$

if $L_2 \geq \frac{L\delta}{\nu^{1/3}\epsilon^{1/3}}$ **then**
 $\tilde{L} = \tilde{\sigma} \leftarrow \Theta\left(\frac{L_2 \nu^{1/3} \epsilon^{1/3}}{\delta}\right) \in [L, \infty)$;

else
 $\tilde{L} \leftarrow L$ и $\tilde{\sigma} \leftarrow \max\{\frac{\nu \epsilon L_2^3}{L^2 \delta^3}, \frac{\epsilon L}{\nu^{1/2}}\} \in [\delta, L]$;

end

$B \leftarrow \Theta(\nu/\epsilon^2)$; $p \leftarrow \Theta(\frac{\tilde{\sigma}^2}{L^2})$; $\alpha \leftarrow \Theta(\frac{\tilde{\sigma}}{p^2 \tilde{L}^2})$;

$X \leftarrow \square$; $N_1 \leftarrow \Theta(\frac{\tilde{\sigma} \nabla f}{p \epsilon^2})$, где ∇f ограничивает сверху $f(y_0) - \min_y \{f(y)\}$;

for $k \leftarrow 0$ **to** ∞ **do**

 Применить алгоритм Ожа для нахождения $\min EV$ для $\nabla^2 f(y_k)$;

if найдена $v \in \mathbb{R}^d$ при $v^T \nabla^2 f(y_k) v \leq -\frac{\delta}{2}$ **then**

$y_{k+1} \leftarrow y_k \pm \frac{\delta}{L_2} v$, где знак случайный ;

else

$F(x) = F^k(x) := f(x) + L(\max\{0, \|x - y_k\| - \frac{\delta}{L_2}\})^2$;

$(\hat{y}_k, y_{k+1}) \leftarrow \text{Natasha1.5}(F, y_k, B, p, 1, \alpha)$;

$X \leftarrow [X, (y_k, \hat{x})]$;

 прекратить цикл по прошествии N_1 шагов первого порядка

end

end

$(\hat{y}_k, y_{k+1}) \leftarrow$ случайная пара из X ;

Algorithm 1: Natasha2 (f, y_0, ϵ, δ)

3.4 Составляющие

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

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

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

Псевдокод :

Input: функция $F(x)$, удовл. 3.1, начальный вектор x^\varnothing , длина эпохи $B \in [n]$,
 количество суб-эпох $p \in [B]$, количество эпох $T' \geq 1$, learning rate $\alpha > 0$
 $\hat{x} \leftarrow x^\varnothing; m \leftarrow B/p; X \leftarrow \square$;
for $k \leftarrow 1$ **to** T' **do**
 $\tilde{x} \leftarrow \hat{x}; \mu \leftarrow \frac{1}{B} \sum_{i \in S} \nabla f_i(\tilde{x})$
 где S распределение случайного подмножества $[n]$ при $|S| = B$;
 for $s \leftarrow 0$ **to** $p - 1$ **do**
 $x_0 \leftarrow \tilde{x}; X \leftarrow [X, \tilde{x}]$;
 for $t \leftarrow 0$ **to** $m - 1$ **do**
 $i \leftarrow$ случайный индекс из $[n]$;
 $\tilde{\nabla} \leftarrow \nabla f_i(x_t) - \nabla f_i(\tilde{x}) + \mu + 2\sigma(x_t - \tilde{x})$;
 $x_{t+1} = \arg \min_{y \in \mathbb{R}^d} \{\psi(y) + \frac{1}{2\alpha} \|y - x_t\|^2 + \langle \tilde{\nabla}, y \rangle\}$
 // в нашем случае $x_{t+1} = x_t - \alpha \tilde{\nabla}$ так как $\psi \equiv 0$
 end
 $\hat{x} \leftarrow$ случайно выбран из $\{x_0, \dots, x_{m-1}\}$
 end
end
 $\hat{y} \leftarrow$ случайный вектор из X и $y^+ \leftarrow \hat{x}$
Algorithm 2: Natasha1.5 ($F, x^\varnothing, B, p, T', \alpha$)

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 $\mathbf{A}_1, \dots, \mathbf{A}_T \sim \mathcal{D}$. Алгоритм начинает со случайного гауссовского вектора $w_1 \in \mathbb{R}^d$ единичной нормы. Затем на каждой итерации $k = 2, \dots, T$ 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 $\{w_1, \dots, w_T\}$

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 избегании седловых точек за счет информации гессиана, идея SpiderBoost – построение вспомогательной оценки требующей на пересчет меньше операций чем градиент.

Подробнее с техническими отличиями алгоритмов можно ознакомиться в таблице 1:

Таблица 1: сравнение характеристик Natasha 2 и SpiderBoost		
Характеристика	SpiderBoost	Natasha 2
Порядок метода	I	II
Искомый вектор	$x \in \mathbb{R}^d : \hat{x} \in \mathbb{R}^d : \ \nabla f(\hat{x})\ \leq \epsilon$	$x \in \mathbb{R}^d : \ \nabla f(\hat{x})\ \leq \epsilon; \nabla^2 f(\hat{x}) \succeq -\delta \mathbf{I}$
Вид целевой функции	<ul style="list-style-type: none"> $\exists \inf_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) > -\infty$ $\ \nabla f_i(x) - \nabla f_i(y)\ \leq L\ x - y\$ $\forall i \in 1 \dots n, \forall x, y \in \mathbb{R}^d$ 	<ul style="list-style-type: none"> $\exists \nu : \mathbb{E}_i[\ \nabla f_i(x) - \nabla f(x)\ ^2] < \nu$ $\ \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:

SpiderBoost:

https://github.com/zhestyatsky/mipt-opt-project/blob/master/src/spider_boost.py

Natasha 2:

<https://github.com/zhestyatsky/mipt-opt-project/blob/master/src/natasha.py>

5.2 Результаты экспериментов

5.2.1 SpiderBoost

SpiderBoost на практике показал себя хорошо: при оптимизации функции потерь для логистической регрессии по скорости сходимости он значительно обошел градиентный спуск.

Также использование импульса в SpiderBoost привело к улучшению результата на некотором классе функций.

По ссылке ниже можно ознакомиться с поведением SpiderBoost на логистической регрессии в сравнении с другими алгоритмами:

SpiderBoost: результаты

<https://github.com/zhestyatsky/mipt-opt-project/blob/master/src/racing.ipynb>

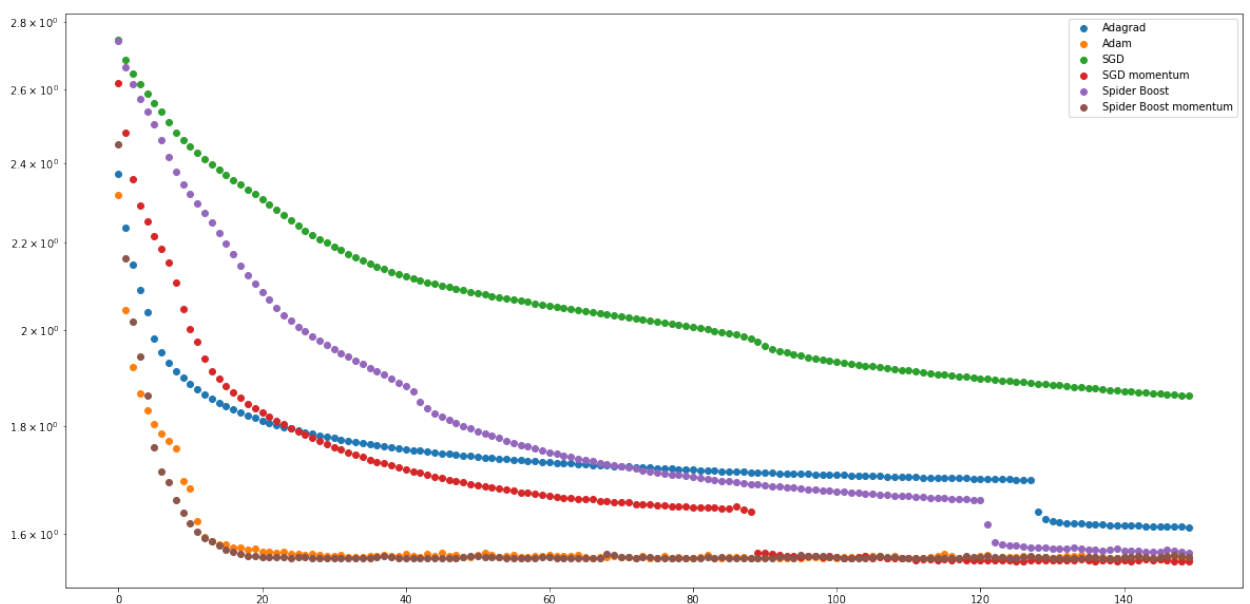
5.2.2 Natasha 2

По результатам эксперимента мы видим, что алгоритм Natasha плохо реализуем на практике из-за наличия следующих сложностей:

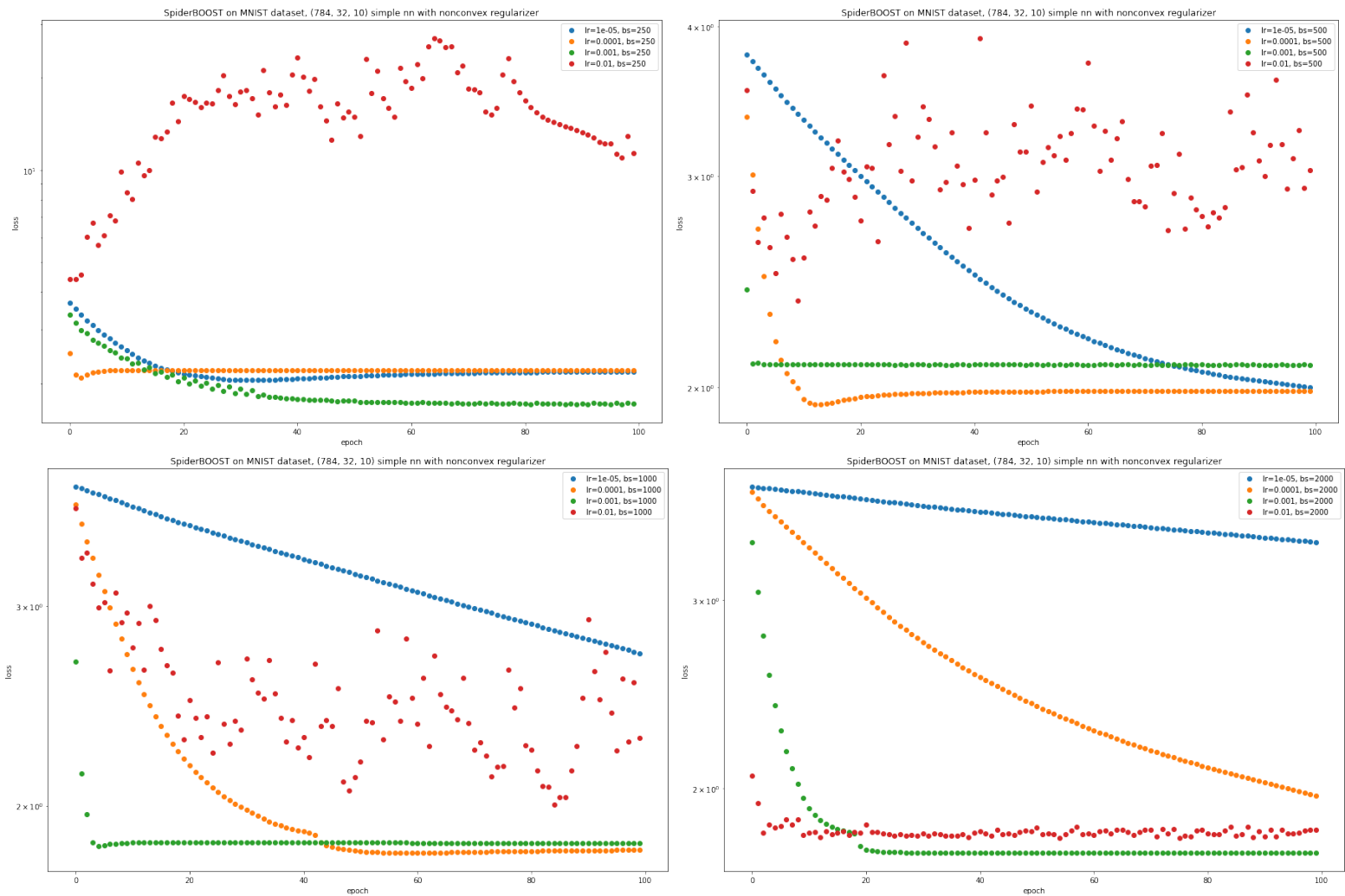
- Сложность в подборе оптимальных гиперпараметров, требуемых для алгоритма и сильно влияющих на его поведение
- Присутствие коэффициента Липшица для гессиана в качестве одного из гиперпараметров
- Также реализация осложнена работой с гессианом в PyTorch.

5.2.3 Сравнение результатов

Ниже приведен график сравнения работы алгоритма SpiderBoost с конкурирующими алгоритмами на логистической регрессии



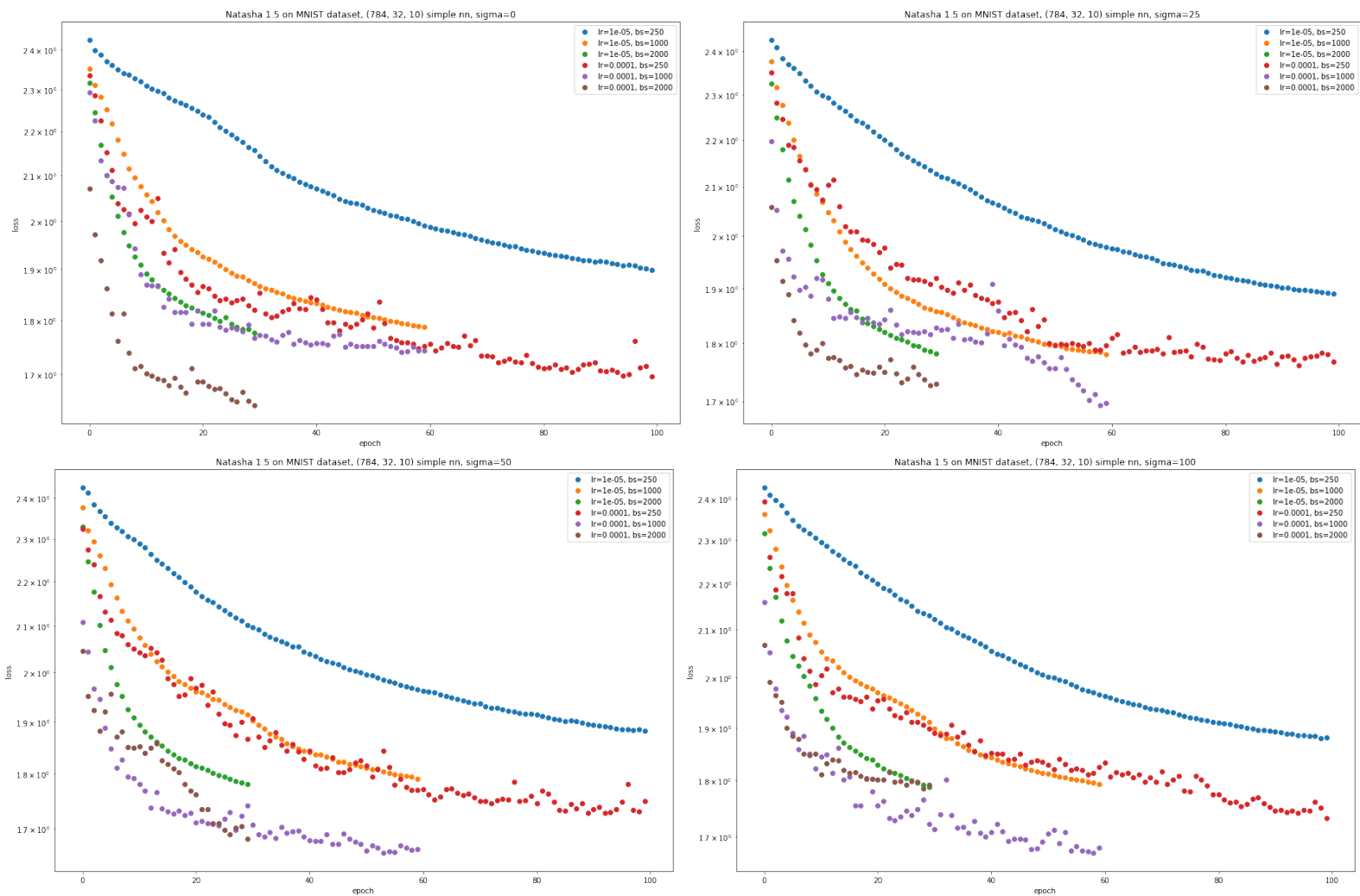
Работа SpiderBoost на нейросети в зависимости от входных параметров



Посмотреть на GitHub

https://github.com/zhestyatsky/mipt-opt-project/blob/master/src/params_dependency.ipynb

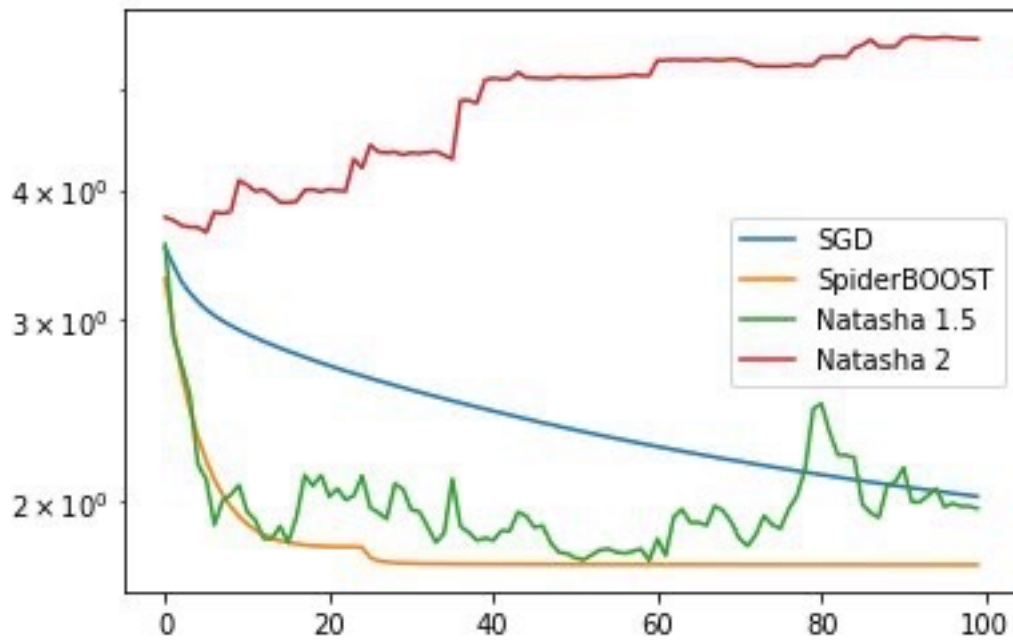
Работа Natasha 1.5 на нейросети в зависимости от входных параметров



Продолжение в источнике...

https://github.com/zhestyatsky/mipt-opt-project/blob/master/src/params_dependency.ipynb

Сравнение работы SpiderBoost и Natasha 1.5



В целом, результаты сравнения алгоритмов SpiderBoost и Natasha2 не в пользу последней

5.2.4 Выводы

По результатам опытов мы видим, что SpiderBoost значительно лучше реализуем и применим на практике чем Natasha 2. Так, скорость сходимости SpiderBoost на логистической регрессии превзошла это значение у SGD. Версия SpiderBoost, ускоренная посредством momentum также превосходит по скорости сходимости Adam, Adagrad и SGD momentum

Такое различие в проблематичности использования алгоритмов SpiderBoost и Natasha 2 во многом объясняется множеством требований последнего к гиперпараметрам.

6 Заключение

Мы рассмотрели стохастические алгоритмы решения оптимизационных задач Natasha 2 и SpiderBoost. Были также проведены теоретические и практические сравнения алгоритмов, по результатам которых мы видим, что, хотя Natasha 2 и SpiderBoost несколько разные задачи с разными запросами, тем не менее, SpiderBoost значительно лучше проявляет себя в применении на практике.