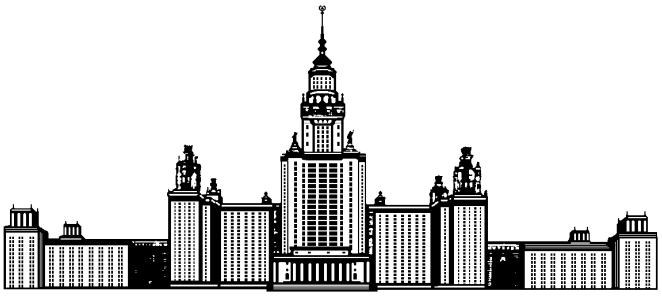
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МАТЕМАТИЧЕСКИХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ



АНАЛИЗ БЫСТРОГО ГРАДИЕНТНОГО МЕТОДА НЕСТЕРОВА ДЛЯ ЗАДАЧ МАШИННОГО ОБУЧЕНИЯ С L_1-РЕГУЛЯРИЗАЦИЕЙ

КУРСОВАЯ РАБОТА

Выполнил:
студент 3 курса 317 группы
Родоманов Антон Олегович

Куратор:
м.н.с. ВЦ РАН
Кропотов Дмитрий Александрович

Научный руководитель:
к.ф-м.н., доцент
Ветров Дмитрий Петрович

Москва, 2014

Аннотация

В данной работе описывается быстрый градиентный метод (метод Нестерова), а также его наиболее успешные модификации: переменная длина шага, адаптивный рестарт и схема для функций с L_1-регуляризатором. Производится экспериментальное сравнение описанных методов с другими методами оптимизации 1-го порядка на задачах, возникающих в области машинного обучения.

Содержание

1 Введение

Быстрый градиентный метод был впервые предложен Ю. Е. Нестеровым в 1983 году [12] как альтернатива стандартному методу градиентного спуска для выпуклых функций. Ю. Е. Нестеров показал, что небольшой модификацией стандартного метода можно добиться существенного ускорения в скорости сходимости: вместо классического порядка $O\left(\frac{1}{k}\right)$ получается порядок $O\left(\frac{1}{k^2}\right)$. При этом арифметическая сложность одной итерации практически не меняется.

За последние годы интерес к быстрому градиентному методу сильно возрос [9, 5, 19]. Появилось множество различных модификаций метода [4, 6, 10, 15, 16]. При этом наиболее успешные из них (например, переменная длина шага [15] и адаптивный рестарт [16]) были описаны в различных работах и так и не были объединены в один алгоритм. Что же касается экспериментального сравнения быстрого градиентного метода с другими методами оптимизации 1-го порядка, то такое сравнение (насколько нам известно) проводилось лишь со стандартным методом градиентного спуска (работы [6, 15, 16, 8]) и с градиентным методом Барзилая-Борвейна (работа [7]).

Данная работа имеет перед собой следующие две цели:

1. *Выделить из общего числа наиболее успешные модификации быстрого градиентного метода и объединить их в один алгоритм.*

2. *Провести экспериментальное сравнение быстрого градиентного метода с другими методами оптимизации 1-го порядка.*

Структура данной работы следующая. В разделе 2 мы подробно выводим основной алгоритм быстрого градиентного метода. В разделах 3, 4 и 5 описываются наиболее успешные модификации основного алгоритма: схема с переменной длиной шага, рестарт и схема для L_1-регуляризованных функций. Наконец, в разделе 6 происходит экспериментальное сравнение описанных алгоритмов как между собой, так и с другими методами оптимизации 1-го порядка.

1.1 Определения и обозначения

Мы будем рассматривать гладкую функцию $f(\mathbf{x})$, заданную в n-мерном евклидовом пространстве $\mathbb{R}^n$, в котором скалярное произведение $\langle \cdot, \cdot \rangle$ и норма $\|\cdot\|$ определены стандартным образом:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=1}^{n} x^{(j)} y^{(j)}, \quad \|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

Нас будут интересовать следующие две безусловные задачи оптимизации:

1. **Минимизация гладкой функции**:

$$f(\mathbf{x}) \to \min_{\mathbf{x} \in \mathbb{R}^n}$$

2. **Минимизация гладкой функции с L_1-регуляризатором**:

$$f(\mathbf{x}) + \tau \|\mathbf{x}\|_1 \to \min_{\mathbf{x} \in \mathbb{R}^n}$$

Напомним основные определения и неравенства, которые нам понадобятся.

Определение 1. *Говорят, что гладкая функция $f(\mathbf{x})$ обладает липшицевым градиентным с константой Липшица $L > 0$, если для любых $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ верно следующее неравенство:*

$$f(\mathbf{x}) \leqslant f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2. \tag{1}$$

Определение 2. *Гладкая функция $f(\mathbf{x})$ называется выпуклой, если для любых $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ выполнено следующее неравенство:*

$$f(\mathbf{x}) \geqslant f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle. \tag{2}$$

Определение 3. *Гладкая функция $f(\mathbf{x})$ называется строго выпуклой с параметром выпуклости $\mu > 0$, если для любых $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ выполнено следующее неравенство:*

$$f(\mathbf{x}) \geqslant f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{3}$$

Всюду в дальнейшем мы будем считать, что функция $f(\mathbf{x})$ является выпуклой и обладает липшицевым градиентом с константой L. В разделе 4 мы потребуем, чтобы функция $f(\mathbf{x})$ также была строго выпуклой с параметром выпуклости μ.

2 Основной алгоритм

В этом разделе мы подробно покажем, как выводится основной алгоритм быстрого градиентного метода.

2.1 Оценочная последовательность

Ключевым объектом, на основе которого строится быстрый градиентный метод, является т. н. оценочная последовательность.

Определение 4. *Оценочной последовательностью для функции $f(\mathbf{x})$ будем называть тройку последовательностей, состоящую из*

- *последовательности точек $\{\mathbf{x}_k\}_{k=0}^{\infty}$,*
- *последовательности коэффициентов $\{A_k\}_{k=0}^{\infty}$,*
- *последовательности функций $\{\psi_k(\mathbf{x})\}_{k=0}^{\infty}$,*

обеспечивающих выполнение следующих двух отношений для всех $k \geqslant 0$:

$$\begin{aligned} &\mathcal{R}_k^1: \qquad A_k f(\mathbf{x}_k) \leqslant \psi_k^* \equiv \min_{\mathbf{x} \in \mathbb{R}^n} \psi_k(\mathbf{x}), \\ &\mathcal{R}_k^2: \qquad \psi_k(\mathbf{x}) \leqslant A_k f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2, \quad \forall\, \mathbf{x} \in \mathbb{R}^n. \end{aligned} \tag{4}$$

Смысл оценочной последовательности заключается в следующем. Мы строим итерационный процесс, который «шагает» по точкам $\mathbf{x}_k$. От этих точек $\mathbf{x}_k$ мы требуем, чтобы они были «достаточно хорошими», т. е. удовлетворяли неравенству $\mathcal{R}_k^1$ для некоторых $\psi_k(\mathbf{x})$ и A_k. Эти $\psi_k(\mathbf{x})$ и A_k, в свою очередь, тоже должны быть «достаточно хорошими» в том смысле, что они должны удовлетворять неравенству $\mathcal{R}_k^2$. Если наш итерационный процесс удовлетворяет этим требованиям, то для него справедлива следующая оценка скорости сходимости.

Лемма 1. *Если для всех $k \geqslant 0$ выполняются отношения (4), то для всех $k \geqslant 1$ справедлива следующая оценка:*

$$f(\mathbf{x}_k) - f^* \leqslant \frac{1}{2A_k}\|\mathbf{x}^* - \mathbf{x}_0\|^2.$$

Доказательство. Следует из цепочки неравенств:

$$A_k f(\mathbf{x}_k) \overset{\mathcal{R}_k^1}{\leqslant} \psi_k^* \leqslant \psi_k(\mathbf{x}^*) \overset{\mathcal{R}_k^2}{\leqslant} A_k f(\mathbf{x}^*) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^*\|^2.$$

□

Таким образом, скорость сходимости нашего итерационного процесса полностью определяется скоростью роста коэффициентов A_k. В следующем разделе мы рассмотрим, как построить оценочную последовательность, у которой коэффициенты A_k будут расти как $O(k^2)$. Таким образом, мы получим метод оптимизации, скорость сходимости которого будет $O\left(\frac{1}{k^2}\right)$.

2.2 Схема с постоянной длиной шага

Итак, нам нужно построить последовательности $\{\mathbf{x}_k\}_{k=0}$, $\{A_k\}_{k=0}^{\infty}$ и $\{\psi_k(\mathbf{x})\}_{k=0}^{\infty}$, удовлетворяющие отношениям $\mathcal{R}_k^1$ и $\mathcal{R}_k^2$.

Начнем с отношения $\mathcal{R}_k^2$. Следующая лемма показывает, как нужно выбрать функции $\psi_k(\mathbf{x})$, чтобы выполнялось отношение $\mathcal{R}_k^2$.

Лемма 2. *Для любых последовательностей $\{\mathbf{x}_k\}_{k=1}^{\infty}$ и $\{a_k\}_{k=1}^{\infty}$, таких что $a_k > 0$, функции $\psi_k(\mathbf{x})$ и коэффициенты A_k, определенные равенствами*

$$\begin{aligned} \psi_k(\mathbf{x}) &= \psi_{k-1}(\mathbf{x}) + a_k\left[f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k\rangle\right], \qquad k \geqslant 1, \\ A_k &= A_{k-1} + a_k, \qquad k \geqslant 1, \\ \psi_0(\mathbf{x}) &= \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2, \quad A_0 = 0. \end{aligned} \tag{5}$$

для всех $k \geqslant 0$ удовлетворяют отношению $\mathcal{R}_k^2$.

Доказательство. По индукции.

Отношение $\mathcal{R}_0^2$ выполнено, т. к. $A_0 = 0$.

Пусть отношение $\mathcal{R}_k^2$ выполнено для некоторого $k \geqslant 0$. Тогда

$$\psi_{k+1}(\mathbf{x}) \overset{(5)}{=} \psi_k(\mathbf{x}) + a_{k+1}[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle] \overset{\mathcal{R}_k^2,(2)}{\leqslant}$$
$$\leqslant \left(A_k f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2\right) + a_{k+1}f(\mathbf{x}) = A_{k+1}f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2,$$

т. е. отношение $\mathcal{R}_{k+1}^2$ тоже выполнено. □

Для завершения построения оценочной последовательности осталось выбрать $\{\mathbf{x}_k\}_{k=1}^{\infty}$ и $\{a_k\}_{k=1}^{\infty}$, которые будут гарантировать выполнение оставшегося отношения $\mathcal{R}_k^1$ (для $\psi_k(\mathbf{x})$ и A_k, определенных равенствами (5)).

Последовательности $\{\mathbf{x}_k\}_{k=0}^{\infty}$ и $\{a_k\}_{k=1}^{\infty}$, удовлетворяющие $\mathcal{R}_k^1$, будем строить итеративно. Пусть текущий номер итерации k, и отношение $\mathcal{R}_k^1$ выполняется. Как выбрать x_{k+1} и a_{k+1} так, чтобы выполнялось отношение $\mathcal{R}_{k+1}^1$? Чтобы ответить на этот вопрос, сначала оценим ψ_{k+1}^*, стоящий в правой части отношения $\mathcal{R}_{k+1}^1$.

Лемма 3. *Пусть выполняется отношение $\mathcal{R}_k^1$. Тогда справедлива оценка:*

$$\psi_{k+1}^* \geqslant A_{k+1}f(\mathbf{x}_{k+1}) + \xi_k^*,$$

где

$$\xi_k^* \equiv A_{k+1}\left[\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{y}_k - \mathbf{x}_{k+1}\rangle - \frac{a_{k+1}^2}{2(A_k + a_{k+1})}\|\nabla f(\mathbf{x}_{k+1})\|^2\right], \tag{6}$$

$$\mathbf{y}_k = \frac{A_k\mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}},$$

$$\mathbf{v}_k = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \psi_k(\mathbf{x}). \tag{7}$$

Доказательство. Заметим, что, согласно (5), функцию $\psi_k(\mathbf{x})$ можно выразить следующим образом:

$$\psi_k(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2 + \sum_{i=1}^{k} a_i[f(\mathbf{x}_i) + \langle \nabla f(\mathbf{x}_i), \mathbf{x} - \mathbf{x}_i\rangle]. \tag{8}$$

Таким образом, функция $\psi_k(\mathbf{x})$ является строго выпуклой с параметром выпуклости 1 (как сумма строго выпуклой функции с параметром выпуклости 1 и выпуклых функций). Поэтому, согласно (3), для $\psi_k(\mathbf{x})$ справедлива следующая нижняя оценка

$$\psi_k(\mathbf{x}) \geqslant \psi_k^* + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_k\|^2, \tag{9}$$

где $\mathbf{v}_k = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \psi_k(\mathbf{x})$, $\psi_k^* = \psi(\mathbf{v}_k)$.

Далее:

$$\begin{aligned}
\psi_{k+1}(\mathbf{x}) &\overset{(5)}{=} \psi_k(\mathbf{x}) + a_{k+1}[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle] \overset{(9)}{\geqslant} \\
&\geqslant \left(\psi_k^* + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_k\|^2\right) + a_{k+1}[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle] \overset{\mathcal{R}_k^1}{\geqslant} \\
&\geqslant \left(A_k f(\mathbf{x}_k) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_k\|^2\right) + a_{k+1}[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle] \overset{(2)}{\geqslant} \\
&\geqslant \left(A_k[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k\rangle] + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_k\|^2\right) + \\
&\qquad + a_{k+1}[f(\mathbf{x}_{k+1}) + \langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle] = \\
&= A_{k+1} f(\mathbf{x}_{k+1}) + \xi_k(\mathbf{x}),
\end{aligned}$$

где

$$\xi_k(\mathbf{x}) \equiv A_k\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k\rangle + a_{k+1}\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x} - \mathbf{x}_{k+1}\rangle + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_k\|^2.$$

Минимум функции $\xi_k(\mathbf{x})$ достигается в точке $\mathbf{z}_k = \mathbf{v}_k - a_{k+1}\nabla f(\mathbf{x}_{k+1})$ и равен

$$\xi_k^* = \xi_k(\mathbf{z}_k) = A_k\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k\rangle + a_{k+1}\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{v}_k - \mathbf{x}_{k+1}\rangle - \frac{a_{k+1}^2}{2}\|\nabla f(\mathbf{x}_{k+1})\|^2.$$

Для завершения доказательства осталось осталось объединить первые два слагаемых в одно и вынести множитель A_{k+1} за скобки. □

Заметим, что неравенство 3 с точностью до слагаемого ξ_k^* является отношением $\mathcal{R}_{k+1}^1$. Поэтому, для того чтобы гарантировать выполнение отношения $\mathcal{R}_{k+1}^1$, достаточно выбрать $\mathbf{x}_{k+1}$ и a_{k+1} таким образом, чтобы ξ_k^*, определенное формулой (6), было неотрицательным. Следующая лемма показывает, что такой выбор можно сделать если:

- выбрать $\mathbf{x}_{k+1}$ с помощью градиентного шага из $\mathbf{y}_k$: $\mathbf{x}_{k+1} = \mathbf{y}_k - \frac{1}{L}\nabla f(\mathbf{y}_k)$;
- выбрать a_{k+1} как положительное решение уравнения: $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \frac{1}{L}$.

Лемма 4. *Для любого $\mathbf{y} \in \mathbb{R}^n$, любого $0 < \alpha \leqslant \frac{1}{L}$ и $\mathbf{x} = \mathbf{y} - \alpha\nabla f(\mathbf{y})$ справедливо следующее неравенство:*

$$\langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle \geqslant \alpha\|\nabla f(\mathbf{x})\|^2. \tag{10}$$

Доказательство. Заметим, что

$$\mathbf{y} - \mathbf{x} = \alpha\nabla f(\mathbf{y}). \tag{11}$$

Проведем цепочку преобразований:

$$\begin{aligned}
\langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle - \alpha\|\nabla f(\mathbf{x})\|^2 &\overset{(11)}{=} \alpha\langle \nabla f(\mathbf{x}), \nabla f(\mathbf{y})\rangle - \alpha\|\nabla f(\mathbf{x})\|^2 = \\
&= \alpha\langle \nabla f(\mathbf{x}), \nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\rangle = \\
&= \alpha\langle \nabla f(\mathbf{y}), \nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\rangle - \alpha\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|^2 \overset{(11)}{=} \\
&= \langle \nabla f(\mathbf{y}) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle - \alpha\|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|^2.
\end{aligned}$$

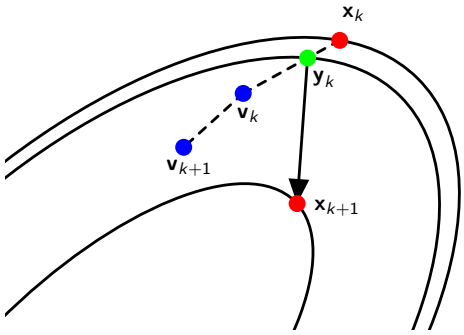


Рис. 1: Иллюстрация одной итерации алгоритма 1. Сначала выбирается некоторая вспомогательная точка $\mathbf{y}_k = \frac{A_k\mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}}$ на отрезке между $\mathbf{x}_k$ и $\mathbf{v}_k$. Далее из этой точки осуществляется градиентный шаг: $\mathbf{x}_{k+1} = \mathbf{y}_k - \frac{1}{L}\nabla f(\mathbf{y}_k)$.

Для завершения доказательства осталось воспользоваться следующим неравенством, справедливым для функций, обладающих липшицевым градиентом с константой L (см. [13, теорема 2.1.5]):

$$\langle \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geqslant \frac{1}{L}\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|, \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

□

Прежде, чем выписать итоговый алгоритм, получим формулы для нахождения $\mathbf{v}_k$, определенных в (7).

Лемма 5. *Для последовательности $\{\mathbf{v}_k\}_{k=0}^{\infty}$, определенной в (7), справедливы следующие рекуррентные формулы:*

$$\begin{aligned} &\mathbf{v}_k = \mathbf{v}_{k-1} - a_k \nabla f(\mathbf{x}_k), \qquad k \geqslant 1, \\ &\mathbf{v}_0 = \mathbf{x}_0. \end{aligned} \tag{12}$$

Доказательство. Точку $\mathbf{v}_k$ можно найти в явном виде из формулы (8):

$$\mathbf{v}_k = \mathbf{x}_0 - \sum_{i=1}^{k} a_i \nabla f(\mathbf{x}_i).$$

Таким образом, для последовательности $\{\mathbf{v}_k\}_{k=0}^{\infty}$ справедливы формулы (12). □

Итоговая схема быстрого градиентного метода приведена в алгоритме 1. Будем называть эту схему *схемой с постоянной длиной шага*. Иллюстрация одной итерации алгоритма 1 представлена на рис. 1.

Алгоритм 1 Схема с постоянной длиной шага

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, $L > 0$ — константа Липшица для $\nabla f(\mathbf{x})$;

$\mathbf{v}_0 := \mathbf{x}_0$; $A_0 := 0$;
for $k := 0, 1, 2, \ldots$ **do**
  {найти $a_{k+1} > 0$ из уравнения $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \frac{1}{L}$};
  $\mathbf{y}_k := \frac{A_k\mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}}$;
  $\mathbf{x}_{k+1} := \mathbf{y}_k - \frac{1}{L}\nabla f(\mathbf{y}_k)$;
  $\mathbf{v}_{k+1} := \mathbf{v}_k - a_{k+1}\nabla f(\mathbf{x}_{k+1})$;
  $A_{k+1} := A_k + a_{k+1}$;
end for

Следующая теорема устанавливает скорость сходимости описанного метода.

Теорема 1. *Быстрый градиентный метод с постоянной длиной шага имеет следующую скорость сходимости:*

$$f(\mathbf{x}_k) - f^* \leqslant \frac{L}{k^2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2. \tag{13}$$

Доказательство. Согласно лемме 1, для доказательства теоремы достаточно показать, что для коэффициентов A_k будет справедлива следующая оценка:

$$A_k \geqslant \frac{k^2}{2L}.$$

Доказательство этой оценки приведено в [15, лемма 7]. □

3 Схема с переменной длиной шага

Алгоритм 1 делает градиентные шаги с постоянной длиной шага $\alpha_k \equiv \frac{1}{L}$ и предполагает, что константа Липшица L для $\nabla f(\mathbf{x})$ известна. На практике, эта константа бывает известна редко. Более того, даже в тех случаях, когда константа L известна, постоянная длина шага является слишком «консервативной». В этом разделе будет описана схема быстрого градиентного метода с *переменной* длиной шага α_k, в которой α_k будет выбираться на каждой итерации автоматически.

3.1 Основная схема

Напомним, что в качестве длины шага α_k можно взять любое число, которое гарантирует выполнение неравенства (10) для $\mathbf{y} = \mathbf{y}_k$, $\alpha = \alpha_k$ и $\mathbf{x} = \mathbf{x}_{k+1}$. Для автоматического подбора нужной длины шага α_k предлагается использовать бэктрекинг. Модифицированная схема представлена в алгоритме 2.

Алгоритм 2 Схема с переменной длиной шага

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\rho > 1$, $\theta \geqslant 1$;
1: $\mathbf{v}_0 := \mathbf{x}_0$; $A_0 := 0$;
2: **for** $k := 0, 1, 2, \ldots$ **do**
3: **loop**
4: {найти a_{k+1} из уравнения $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \alpha_k$};
5: $\mathbf{y}_k := \frac{A_k \mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}}$;
6: $\mathbf{x}_{k+1} := \mathbf{y}_k - \alpha_k \nabla f(\mathbf{y}_k)$;
7: **if** $\langle \nabla f(\mathbf{x}_{k+1}), \mathbf{y}_k - \mathbf{x}_{k+1} \rangle \geqslant \alpha_k \|\nabla f(\mathbf{x}_{k+1})\|^2$ **then break**;
8: $\alpha_k := \alpha_k / \rho$;
9: **end loop**
10: $\mathbf{v}_{k+1} := \mathbf{v}_k - a_{k+1}\nabla f(\mathbf{x}_{k+1})$;
11: $A_{k+1} := A_k + a_{k+1}$;
12: $\alpha_{k+1} := \theta \alpha_k$;
13: **end for**

Подбор нужной длины шага в алгоритме 2 осуществляется с помощью внутреннего цикла (строки 3–9). Длина шага α_k постепенно уменьшается (строка 8) до тех пор, пока не будет выполнено условие в строке 7. Заметим, что, согласно лемме 4, внутренний цикл является конечным. Таким образом, алгоритм 2 определен корректно.

Алгоритм 2 имеет три параметра: α_0, ρ и θ. Параметр α_0 определяет начальную длину шага на самой первой ($k = 0$) итерации. Параметр ρ определяет скорость уменьшения длины шага при бэктрекинге (строка 8). Параметр θ влияет на начальную длину шага для следующей итерации (строка 12). На практике рекомендуется использовать следующие значения: $\alpha_0 = 1$, $\rho = 2$, $\theta = 1.1$.

В отличие от алгоритма 1, алгоритм 2 за одну итерацию может совершать более двух вызовов оракула. Тем не менее, оказывается, что *в среднем* на каждой итерации количество вызовов оракула в алгоритме 2 невелико.

Теорема 2. *Пусть N_k — общее количество вызовов оракула в алгоритме 2 после k итераций. Справедлива следующая оценка:*

$$N_k \leqslant 2\left[1 + \frac{\ln\theta}{\ln\rho}\right](k+1) + \frac{2}{\ln\rho}\ln\frac{\rho\alpha_0 L}{\theta}.$$

Доказательство. Доказательство приведено в [15, лемма 5]. □

Например, если $\alpha_0 = 1$, $\rho = 2$ и $\theta = 1.1$, то

$$N_k \leqslant 2.3(k+1) + 2\log_2(2L).$$

Заметим, что на практике число $\log_2(2L)$ является небольшим. Таким образом, при данном выборе параметров α_0, ρ и θ среднее количество вызовов оракула за итерацию в алгоритме 2 составляет 2.3.

В заключение данного раздела отметим, что для алгоритма 2 справедлива теорема, аналогичная теореме 1. Более того, на практике алгоритм 2 работает существенно быстрее алгоритма 1 (см. раздел 6.1).

3.2 Модифицированное условие на длину шага

В этом разделе мы предлагаем следующую модификацию алгоритма 2: потребуем, чтобы длина шага $\alpha_k > 0$ удовлетворяла следующему условию:

$$f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{y}_k) + \langle\nabla f(\mathbf{y}_k), \mathbf{x}_{k+1} - \mathbf{y}_k\rangle + \frac{1}{2\alpha_k}\|\mathbf{x}_{k+1} - \mathbf{y}_k\|^2.$$

В случае постоянной длины шага $\alpha_k = \frac{1}{L}$ данное условие соответствует условию липшицевости градиента функции $f(\mathbf{x})$ (см. (1)) и поэтому всегда выполняется.

Алгоритм 2 с модифицированным условием на длину шага представлен как алгоритм 3. Алгоритмы отличаются лишь условием в строке 7. Отметим, что теоретические гарантии на скорость сходимости у алгоритма 3 не хуже, чем у алгоритма 2.

Алгоритм 3 Схема с модифицированным условием на длину шага

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\rho > 1$, $\theta \geqslant 1$;
1: $\mathbf{v}_0 := \mathbf{x}_0$; $A_0 := 0$;
2: **for** $k := 0, 1, 2, \ldots$ **do**
3: **loop**
4: {найти a_{k+1} из уравнения $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \alpha_k$};
5: $\mathbf{y}_k := \frac{A_k\mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}}$;
6: $\mathbf{x}_{k+1} := \mathbf{y}_k - \alpha_k\nabla f(\mathbf{y}_k)$;
7: **if** $f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{y}_k) + \langle\nabla f(\mathbf{y}_k), \mathbf{x}_{k+1} - \mathbf{y}_k\rangle + \frac{1}{2\alpha_k}\|\mathbf{x}_{k+1} - \mathbf{y}_k\|^2$ **then break**;
8: $\alpha_k := \alpha_k/\rho$;
9: **end loop**
10: $\mathbf{v}_{k+1} := \mathbf{v}_k - a_{k+1}\nabla f(\mathbf{x}_{k+1})$;
11: $A_{k+1} := A_k + a_{k+1}$;
12: $\alpha_{k+1} := \theta\alpha_k$;
13: **end for**

4 Рестарт

В этом разделе мы покажем, как можно ускорить быстрый градиентный метод для класса строго выпуклых функций с помощью т. н. *рестарта*.

4.1 Рестарт с фиксированной частотой

Пусть функция $f(\mathbf{x})$ является строго выпуклой с параметром выпуклости $\mu > 0$. Тогда справедливо следующее неравенство (см. (2)):

$$f(\mathbf{x}_0) - f^* \geqslant \frac{\mu}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

Применяя это неравенство к оценке (13), получаем следующую скорость сходимости для алгоритма 1:

$$f(\mathbf{x}_k) - f^* \leqslant \frac{2L}{k^2\mu}(f(\mathbf{x}_0) - f^*).$$

Таким образом, через

$$N = \left\lceil \sqrt{\frac{4L}{\mu}} \right\rceil \tag{14}$$

итераций алгоритм 1 гарантирует уменьшение невязки по значению функции как минимум вдвое:

$$f(\mathbf{x}_N) - f^* \leqslant \frac{1}{2}(f(\mathbf{x}_0) - f^*). \tag{15}$$

Аналогичное уменьшение гарантирует и алгоритм 3.

Будем обозначать $\mathcal{A}_N(\mathbf{u})$ точку, полученную алгоритмом 3 за N итераций из начального приближения $\mathbf{u}$. Рассмотрим метод, представленный в алгоритме 4. Этот метод совершает N итераций алгоритма 3, затем «перезапускает» алгоритм 3 из новой начальной точки $\mathbf{u}_1 = \mathcal{A}_N(\mathbf{x}_0)$, снова совершает N итераций, затем опять «перезапускает» алгоритм 3 из новой начальной точки $\mathbf{u}_2 = \mathcal{A}_N(\mathbf{u}_1)$ и т. д.

Алгоритм 4 Общая схема рестарта

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, N — частота рестартов;

1: $\mathbf{u}_0 = \mathbf{x}_0$;
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{u}_{t+1} := \mathcal{A}_N(\mathbf{u}_t)$; $\quad$ // *совершить N итераций алгоритма 3*
4: **end for**

Согласно неравенству (15), получаем, что за t итераций алгоритм 4 уменьшит невязку по значению функции как минимум в 2^t раз:

$$f(\mathbf{u}_t) - f^* \leqslant \frac{1}{2^t}(f(\mathbf{x}_0) - f^*).$$

Это означает, что для нахождения ε-решения по значению функции, алгоритму 4 достаточно суммарно совершить $O\left(\sqrt{\frac{L}{\mu}}\ln\frac{1}{\varepsilon}\right)$ итераций алгоритма 3. Заметим, что гарантия алгоритма 3 составляет лишь $O\left(\sqrt{\frac{L}{\varepsilon}}\right)$ итераций (см. (13)), что значительно хуже.

4.2 Адаптивный рестарт

У вышеописанного метода есть один большой недостаток: требуется знание констант μ и L, которые на практике бывают известны редко. Данный недостаток можно попытаться устранить следующим образом: вместо «оптимальной» частоты рестартов, определяемой формулой (14), выбрать какую-то конкретную частоту, например, $N = 100$ или $N = 500$. Однако есть и другой вариант.

Алгоритм 4 осуществляет рестарт через каждые N итераций алгоритма 3. Это утверждение можно переформулировать немного по-другому. Можно сказать, что алгоритм 4 осуществляет рестарт лишь тогда, когда выполняется следующее *условие рестарта*: количество итераций, совершенное

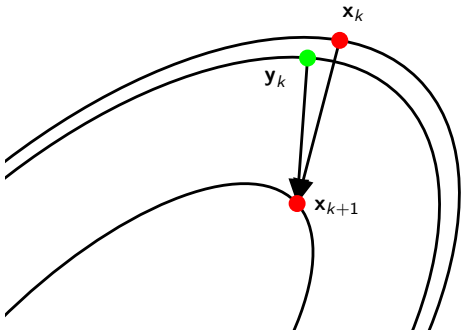


Рис. 2: Иллюстрация градиентного условия рестарта. Если переход от точки $\mathbf{x}_k$ к точке $\mathbf{x}_{k+1}$ произошел *не* по направлению убывания функции в точке $\mathbf{y}_k$, то осуществляется рестарт. В данном случае рестарт *не* произойдет.

алгоритмом 3, равно N. При такой формулировке сразу же возникает идея рассматривать и другие, более сложные, условия рестарта.

В работе [16] предлагается использовать следующее т. н. *градиентное условие рестарта*: $\langle \mathbf{y}_k - \mathbf{x}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k \rangle > 0$ (см. рис. 2). Данное условие можно интегрировать прямо внутрь алгоритма 3 (т. е. необязательно использовать обертку типа алгоритма 4). Рестарт будет заключаться в «сбрасывании» памяти метода, т. е. в «сбрасывании» оценочной последовательности. Итоговая схема, которую мы будем называть *схемой с адаптивным рестартом*, представлена как алгоритм 5. По сути, алгоритм 5 отличается от алгоритма 3 лишь добавлением инструкций в строках 13–15.

Алгоритм 5 Схема с адаптивным рестартом

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\rho > 1$, $\theta \geqslant 1$;

1: $\mathbf{v}_0 := \mathbf{x}_0$; $A_0 := 0$;
2: **for** $k := 0, 1, 2, \ldots$ **do**
3: **loop**
4: {найти a_{k+1} из уравнения $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \alpha_k$};
5: $\mathbf{y}_k := \frac{A_k \mathbf{x}_k + a_{k+1} \mathbf{v}_k}{A_k + a_{k+1}}$;
6: $\mathbf{x}_{k+1} := \mathbf{y}_k - \alpha_k \nabla f(\mathbf{y}_k)$;
7: **if** $f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{y}_k) + \langle \nabla f(\mathbf{y}_k), \mathbf{x}_{k+1} - \mathbf{y}_k \rangle + \frac{1}{2\alpha_k} \|\mathbf{x}_{k+1} - \mathbf{y}_k\|^2$ **then break**;
8: $\alpha_k := \alpha_k / \rho$;
9: **end loop**
10: $\mathbf{v}_{k+1} := \mathbf{v}_k - a_{k+1} \nabla f(\mathbf{x}_{k+1})$;
11: $A_{k+1} := A_k + a_{k+1}$;
12: $\alpha_{k+1} := \theta \alpha_k$;
13: **if** $\langle \mathbf{y}_k - \mathbf{x}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k \rangle > 0$ **then**
14: $\mathbf{x}_{k+1} := \mathbf{x}_k$; $\mathbf{v}_k := \mathbf{x}_k$; $A_{k+1} := 0$; // *рестарт*
15: **end if**
16: **end for**

Отметим, что вышеописанная схема с градиентным условием рестарта в общем случае не имеет гарантий на скорость сходимости. Гарантии имеются только в случае квадратичной функции: в этом случае для нахождения ε-решения по значению функции, алгоритму 5 достаточно будет совершить $O\left(\sqrt{\frac{L}{\mu}} \ln \frac{1}{\varepsilon}\right)$ итераций (см. [16]), т. е. столько же, сколько и алгоритму 4. Однако, несмотря на отсутствие теоретических гарантий, на практике алгоритм 5 всегда работает хорошо и сходится по крайней мере не медленнее алгоритма 3 (см. разделы 6.2 и 6.3).

5 Схема для L_1-регуляризованных функций

В этом разделе мы обобщим алгоритм 5 на случай т. н *L_1-регуляризованных функций*, т. е. функций следующего вида:

$$F(\mathbf{x}) = f(\mathbf{x}) + \tau\|\mathbf{x}\|_1, \tag{16}$$

где функция $f(\mathbf{x})$ является выпуклой и гладкой. Параметр $\tau \geqslant 0$ называется *параметром регуляризации*. Заметим, что в случае $\tau > 0$ функция $F(\mathbf{x})$ уже *не* является гладкой, поэтому напрямую применять алгоритм 5 для оптимизации $F(\mathbf{x})$ нельзя.

Для получения схемы быстрого градиентного метода для функций вида (16) нужно обобщить оценочные функции и основные леммы, полученные в разделе 2.2, на случай L_1-регуляризованных функций. Такое обобщение сделано в работе [15]. Мы не будем повторять необходимые выкладки, а лишь ограничимся итоговым алгоритмом.

Поскольку оптимизируемая функция $F(\mathbf{x})$ уже не является гладкой, то в модифицированном алгоритме нам понадобится осуществлять проекцию. Более конкретно, нам понадобится следующий оператор проектирования, действующий покомпонентно:

$$\mathcal{P}_\delta^{(j)}(\mathbf{x}) = \max\left(|x^{(j)}| - \delta,\, 0\right) \operatorname{sgn}\left(x^{(j)}\right), \quad j = 1, \ldots, n.$$

Поясним смысл этого оператора. Если $|\mathbf{x}^{(j)}| < \delta$, то $\mathcal{P}_\delta^{(j)}(\mathbf{x}) = 0$, т. е. если какая-то компонента вектора $\mathbf{x}$ близка к нулю, то оператор $\mathcal{P}$ обнулит эту компоненту. Таким образом, оператор $\mathcal{P}$ производит «разреживание» вектора $\mathbf{x}$.

Схема для случая L_1-регуляризованных функций приведена в алгоритме 6. Единственное отличие алгоритма 6 от алгоритма 5 заключается в том, что теперь для точек $\mathbf{v}_k$ и $\mathbf{x}_{k+1}$ дополнительно применяется оператор проектирования $\mathcal{P}$ (строки 3 и 7).

Алгоритм 6 Схема для L_1-регуляризованных функций

Вход: $\mathbf{x}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\rho > 1$, $\theta \geqslant 1$, $\tau \geqslant 0$ — коэффициент регуляризации;

1: $\tilde{\mathbf{v}}_0 := \mathbf{x}_0$; $A_0 := 0$;
2: **for** $k := 0, 1, 2, \ldots$ **do**
3: $\quad \mathbf{v}_k := \mathcal{P}_{\tau A_k}(\tilde{\mathbf{v}}_k)$;
4: $\quad$ **loop**
5: $\quad\quad$ {найти a_{k+1} из уравнения $\frac{a_{k+1}^2}{2(A_k + a_{k+1})} = \alpha_k$};
6: $\quad\quad \mathbf{y}_k := \frac{A_k \mathbf{x}_k + a_{k+1}\mathbf{v}_k}{A_k + a_{k+1}}$;
7: $\quad\quad \mathbf{x}_{k+1} := \mathcal{P}_{\tau\alpha_k}\left(\mathbf{y}_k - \alpha_k \nabla f(\mathbf{y}_k)\right)$;
8: $\quad\quad$ **if** $f(\mathbf{x}_{k+1}) \leqslant f(\mathbf{y}_k) + \langle \nabla f(\mathbf{y}_k), \mathbf{x}_{k+1} - \mathbf{y}_k\rangle + \frac{1}{2\alpha_k}\|\mathbf{x}_{k+1} - \mathbf{y}_k\|^2$ **then break**;
9: $\quad\quad \alpha_k := \alpha_k / \rho$;
10: $\quad$ **end loop**
11: $\quad \tilde{\mathbf{v}}_{k+1} := \tilde{\mathbf{v}}_k - a_{k+1}\nabla f(\mathbf{x}_{k+1})$;
12: $\quad A_{k+1} := A_k + a_{k+1}$;
13: $\quad \alpha_{k+1} := \theta\alpha_k$;
14: $\quad$ **if** $\langle \mathbf{y}_k - \mathbf{x}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}_k\rangle > 0$ **then**
15: $\quad\quad \mathbf{x}_{k+1} := \mathbf{x}_k$; $\tilde{\mathbf{v}}_k := \mathbf{x}_k$; $A_{k+1} := 0$; $\quad$ // *рестарт*
16: $\quad$ **end if**
17: **end for**

6 Эксперименты

В этом разделе мы осуществляем эмпирическое сравнение описанных модификаций быстрого градиентного метода как между собой, так и с другими методами оптимизации 1-го порядка.

Мы проводим наши эксперименты на следующих трех задачах:

- **Логистическая регрессия с L_2-регуляризатором**:

$$Q(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\ln(1+\exp(-y_i\langle \mathbf{x}_i, \mathbf{w}\rangle)) + \frac{\tau}{2}\|\mathbf{w}\|^2 \to \min_{\mathbf{w}\in\mathbb{R}^n} \tag{17}$$

- **Логистическая регрессия с L_1-регуляризатором**:

$$Q(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\ln(1+\exp(-y_i\langle \mathbf{x}_i, \mathbf{w}\rangle)) + \tau\|\mathbf{w}\|_1 \to \min_{\mathbf{w}\in\mathbb{R}^n} \tag{18}$$

- **Квадратичная функция**:

$$f(\mathbf{x}) = \frac{1}{2}\langle A\mathbf{x}, \mathbf{x}\rangle + \langle \mathbf{b}, \mathbf{x}\rangle, \quad A = A^T \succeq 0. \tag{19}$$

В качестве данных для логистической регрессии мы используем следующие три набора данных:

- *colon-cancer* [1] с числом признаков $n = 2\,000$ и числом объектов $m = 62$;
- *mnist3vs5* с числом признаков $n = 400$ и числом объектов $m = 11\,552$;
- *sido0* [3] с числом признаков $n = 4\,932$ и числом объектов $m = 12\,678$.

Набор данных *mnist3vs5* получен из базы данных MNIST [2] путем удаления из общей выборки тех объектов, чьи метки отличны от «3» и «5».

Всюду в этом разделе мы будем использовать обозначение «FGM-k» ($k = 1, \ldots, 6$) для соответствующих алгоритмов, описанных в данной работе. Например, FGM-6 соответствует алгоритму 6. Отметим, что для всех алгоритмов быстрого градиентного метода, кроме схемы с постоянной длиной шага, мы используем следующие значения параметров, рекомендуемые в разделе 3.1: $\alpha_0 = 1$, $\rho = 2$, $\theta = 1.1$.

Во всех экспериментах мы проводим сравнение различных методов оптимизации 1-го порядка по скорости сходимости. В качестве результата эксперимента мы приводим график, на котором по горизонтальной оси отложено количество вызовов оракула, а по вертикальной оси — значение оптимизируемой функции (в логарифмической шкале). Мы всегда запускаем все методы оптимизации из нулевой начальной точки: $\mathbf{x}_0 = \mathbf{0}$.

Перейдем к самим экспериментам.

6.1 Стратегии выбора длины шага

В этом эксперименте мы проводим сравнение алгоритмов 1 и 2 на задаче L_2-регуляризованной логистической регрессии (17). Константу Липшица L для алгоритма 1 мы вычисляем по следующей формуле: $L = \frac{1}{m}\lambda_{max}(X^TX) + \tau$, в которой символом $\lambda_{max}(X^TX)$ обозначено максимальное собственное значение матрицы X^TX. Параметр регуляризации τ полагается равным 0.001.

Результаты экспериментов представлены на рис. 3. Мы видим, что схема с переменной длиной шага (алгоритм 2) во всех трех случаях работает намного быстрее, чем аналогичная схема с постоянной длиной шага (алгоритм 1). Отметим, что при этом алгоритму 2, в отличие от алгоритма 1, не требуется предварительного вычисления константы Липшица L.

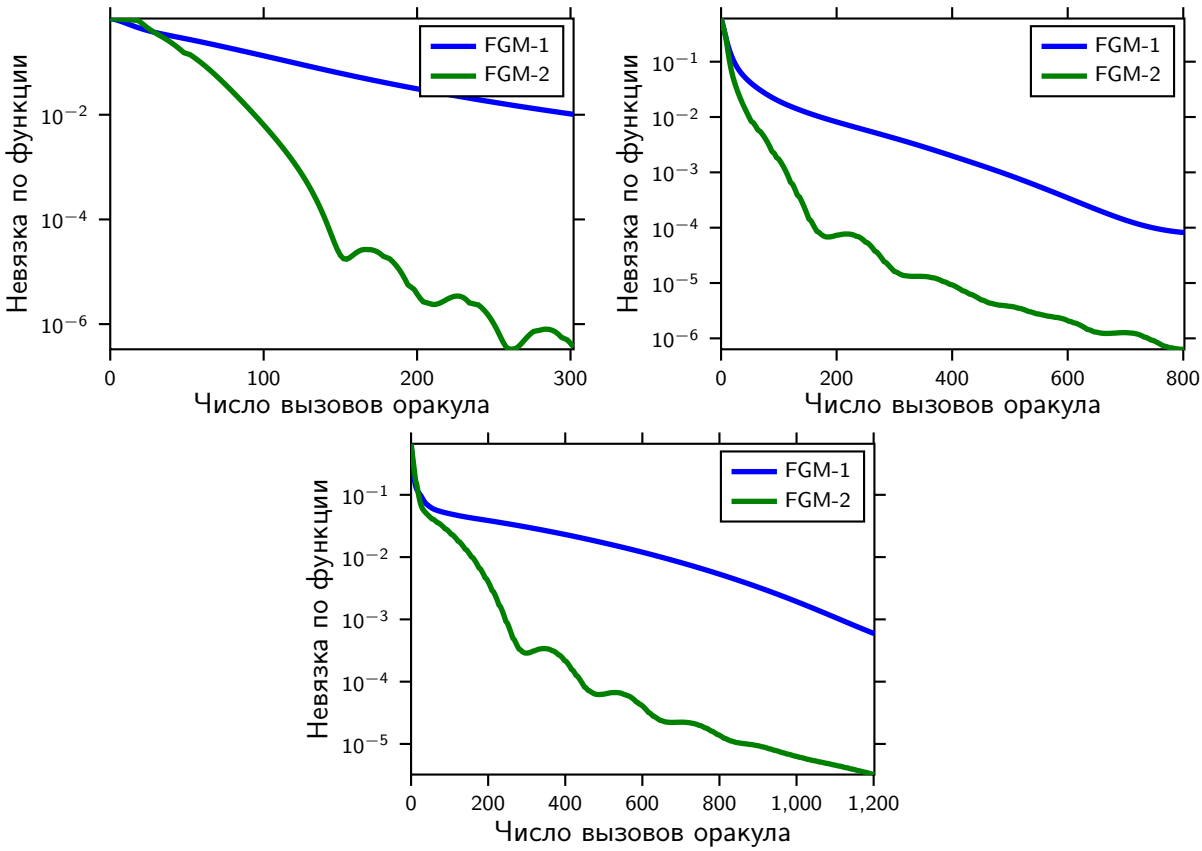


Рис. 3: Сравнение стратегий шага постоянной и переменной длины. Наборы данных (слева направо сверху вниз): *colon-cancer*, *mnist3vs5*, *sido0*.

6.2 Стратегии рестарта

В этом эксперименте мы проводим эмпирическое сравнение двух стратегий рестарта, описанных в разделе 4: схемы рестарта с фиксированной частотой (алгоритм 4) и схемы адаптивного рестарта (алгоритм 5). Мы проводим сравнение на 1000-мерной квадратичной функции (19) с параметром выпуклости $\mu = 0.01$ и константой Липшица для градиента $L = 10$. Параметры квадратичной функции (A и $\mathbf{b}$) генерируются случайно. В данном случае, «оптимальная» частота рестартов, определяемая формулой (14), равна 64.

Результаты эксперимента приведены на рис. 4. Мы видим, что стратегии редкого рестарта (каждые 200 или 400 итераций) ускоряют метод, причем это ускорение тем больше, чем частота рестартов ближе к «оптимальной» ($N = 64$). Если же рестарт осуществляется слишком часто (каждые 10 или 30 итераций), то скорость сходимости сильно замедляется. Таким образом, если указать «неправильное» значение частоты рестартов N, то алгоритм может замедлиться. Напомним, что схема адаптивного рестарта (алгоритм 5), в отличие от схемы с фиксированной частотой рестарта (алгоритм 4) не требует никакого дополнительного параметра N и, как показывает этот эксперимент, работает гораздо лучше схемы с фиксированной частотой рестартов (даже в случае «оптимальной» частоты рестартов).

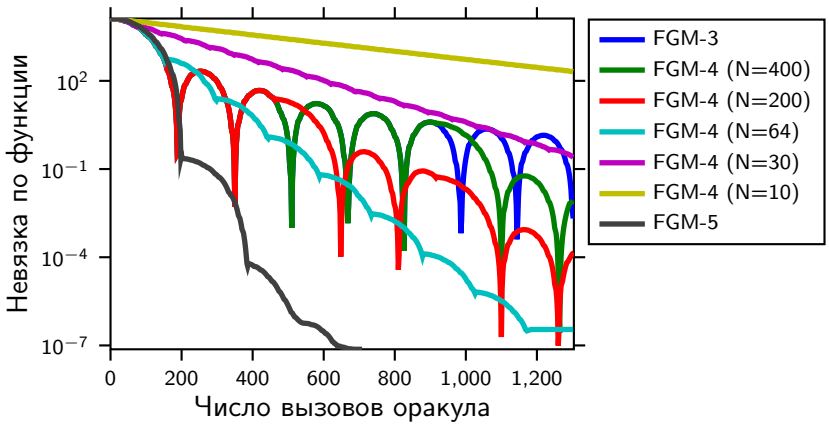


Рис. 4: Сравнение различных стратегий рестарта.

6.3 Адаптивный рестарт

Этим экспериментом мы подтверждаем заявление, сделанное в разделе 4.2, о том, что на практике схема с адаптивным рестартом (алгоритм 5) работает не медленнее, чем аналогичная схема без рестарта (алгоритм 3). Более того, мы демонстрируем, что схема с адаптивным рестартом зачастую приводит к ускорению метода.

Сравнение алгоритмов 3 и 5 проводится на задаче L_2-регуляризованной логистической регрессии (17). Значение параметра регуляризации τ мы берем равным 0.001. Результаты эксперимента приведены на рис. 5.

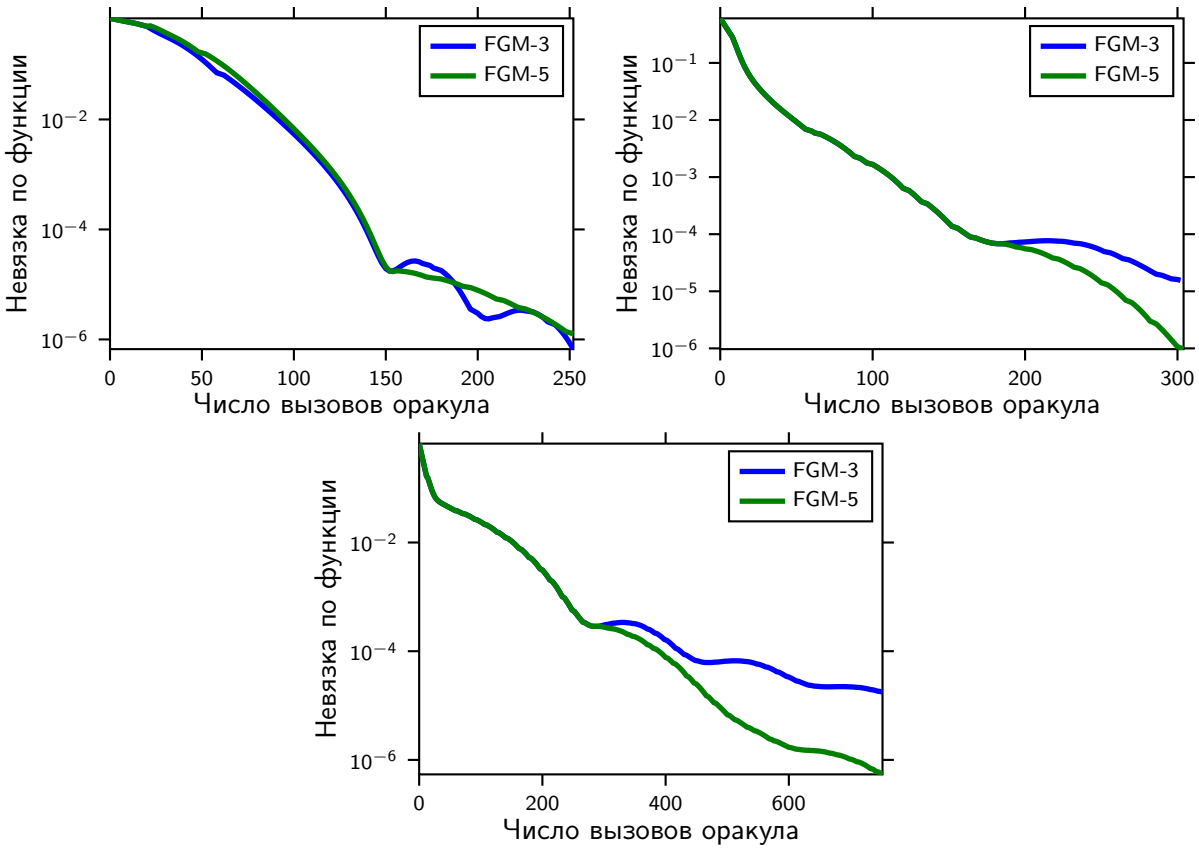


Рис. 5: Сравнение схемы адаптивного рестарта с аналогичной схемой без рестарта. Наборы данных выстроены в том же порядке, что и на рис. 3.

6.4 Сравнение с другими методами гладкой оптимизации

В этом эксперименте мы проводим сравнение быстрого градиентного метода с другими методами гладкой оптимизации на задаче L_2-регуляризованной логистической регрессии (17). Более конкретно, мы проводим сравнение алгоритма 5 со следующими методами, реализованными в библиотеке minFunc [18]:

- **CSD**: Cyclic steepest descent;
- **BB**: Barzilai and Borwein gradient;
- **CG**: Conjugate gradient;
- **LBFGS**: Quasi-Newton with limited-memory BFGS updating.

Значение параметра регуляризации τ мы устанавливаем равным 0.0001.

Результаты сравнения представлены на рис. 6. Мы видим, что среди сравниваемых методов абсолютным лидером по скорости работы является метод LBFGS. Быстрый градиентный метод (алгоритм 5) работает медленнее всех остальных методов.

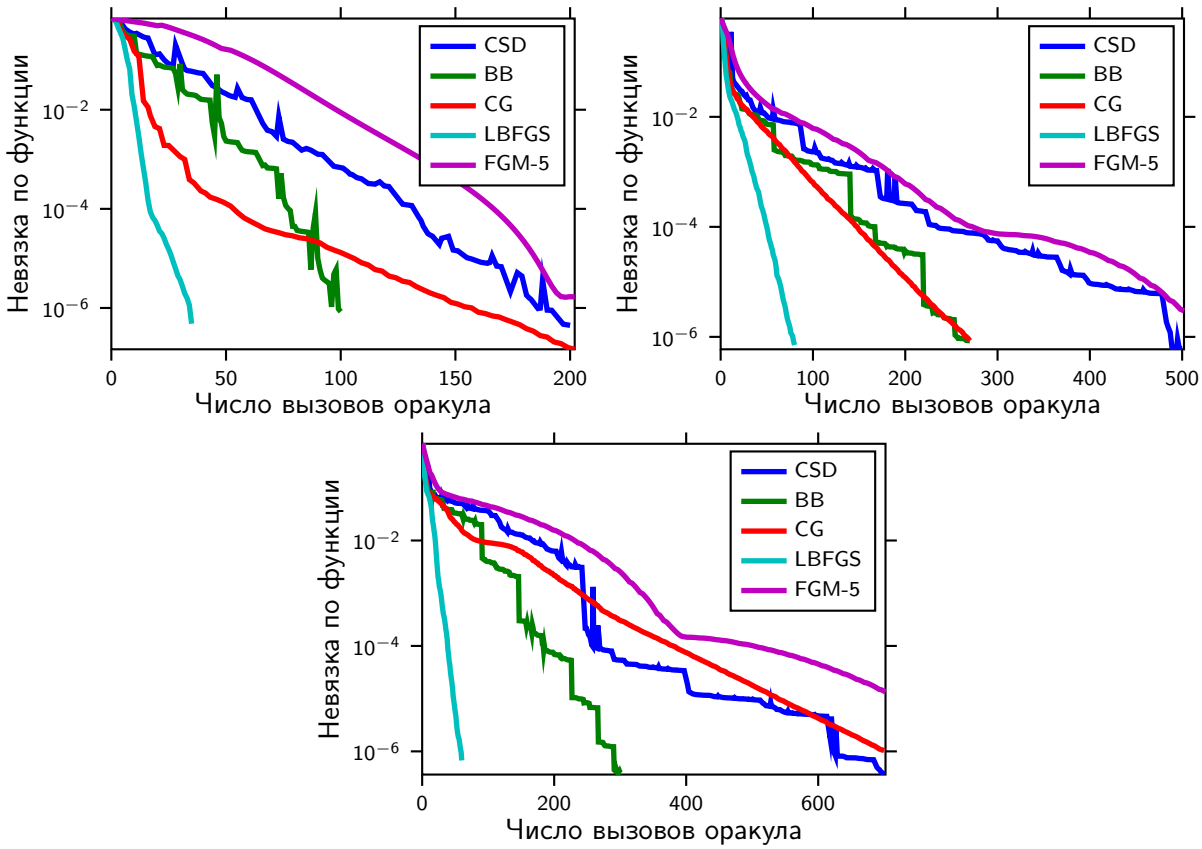


Рис. 6: Сравнение быстрого градиентного метода с другими методами гладкой оптимизации. Наборы данных выстроены в том же порядке, что и на рис. 3.

6.5 Сравнение для L_1-регуляризованных функций

В этом эксперименте мы проводим сравнение алгоритма 6 с другими методами, разработанными специально для оптимизации L_1-регуляризованных функций. Сравнение производится на задаче логистической регрессии с L_1-регуляризатором (18). Значение параметра регуляризации τ мы устанавливаем равным 0.0001.

В качестве «конкурентов» для алгоритма 6 мы выбираем следующие методы, реализованные в библиотеке L1General [17]:

- **BBSG**: Barzilai-Borwein sub-gradient;
- **OWL**: Orthant-wise learning;
- **SPG**: Spectral projected gradient;
- **TMP**: Two-metric projection;
- **PSSas**: Projected scaled sub-gradient active set.

Результаты эксперимента приведены на рис. 7. Как видно из рисунка, поведение методов сильно зависит от набора данных. Однако точно можно сказать, что метод PSSas является фаворитом. Быстрый градиентный метод (алгоритм 6) среди сравниваемых методов занимает «среднее» положение.

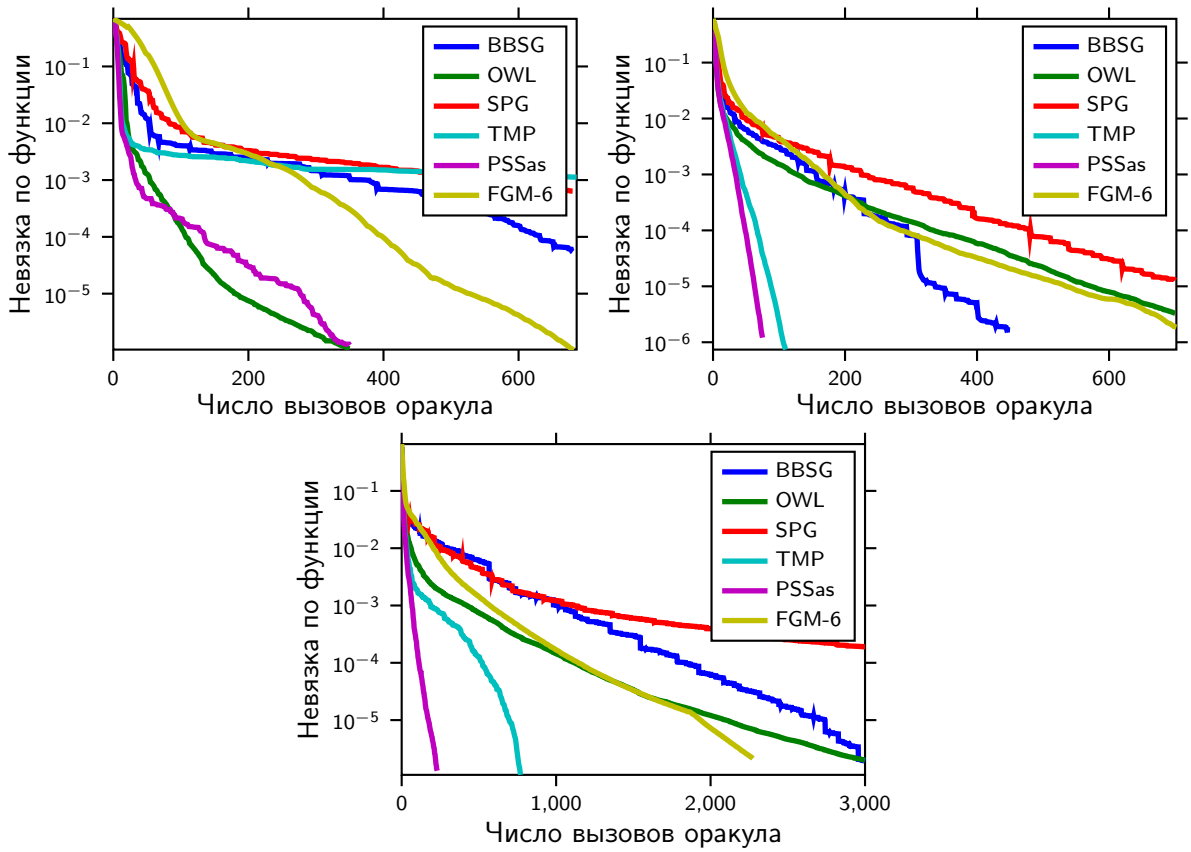


Рис. 7: Сравнение быстрого градиентного метода с другими методами для оптимизации функций с L_1-регуляризатором. Наборы данных выстроены в том же порядке, что и на рис. 3.

7 Заключение

В данной работе был описан быстрый градиентный метод и его наиболее успешные модификации: переменная длина шага, адаптивный рестарт и схема для функций с L_1-регуляризатором. Было проведено экспериментальное сравнение описанных методов с другими методами оптимизации первого порядка на задачах, возникающих в области машинного обучения.

Несмотря на то, что мы рассматривали быстрый градиентный метод применительно к задачам безусловной оптимизации, метод легко модифицируется (см., например, [11]) на случай задач оптимизации по некоторому простому множеству (например, симплексу).

В заключение отметим, что недавно Ю. Е. Нестеров предложил обобщение быстрого градиентного метода на класс менее гладких функций, чем функции с липшицевым градиентом: класс функций, градиент которых удовлетворяет условию Гельдера [14].

Список литературы

[1] Data pertaining to the article 'Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays'. `http://genomics-pubs.princeton.edu/oncology/affydata/index.html`.

[2] The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[3] SIDO phamacology dataset. `http://www.causality.inf.ethz.ch/data/SIDO.html`.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[5] Stephen R Becker, Emmanuel J Candès, and Michael C Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.

[6] ClóvisC. Gonzaga and ElizabethW. Karas. Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming. *Mathematical Programming*, 138(1-2):141–166, 2013.

[7] Tobias Lindstrøm Jensen, Jakob Heide Jørgensen, Per Christian Hansen, and Søren Holdt Jensen. Implementation of an optimal first-order method for strongly convex total variation regularization. *BIT Numerical Mathematics*, 52(2):329–356, 2012.

[8] Qihang Lin and Lin Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. In *Proceedings of The 31st International Conference on Machine Learning*, pages 73–81, 2014.

[9] Jun Liu, Jianhui Chen, and Jieping Ye. Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–556. ACM, 2009.

[10] Xiangrui Meng and Hao Chen. Accelerating Nesterov's method for strongly convex functions with lipschitz gradient. *arXiv preprint arXiv:1109.6058*, 2011.

[11] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[12] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

[14] Yurii Nesterov. Universal gradient methods for convex optimization problems. *Optimization online eprints, Université Catholique de Louvain, CORE*, 2013.

[15] Yurii Nesterov et al. Gradient methods for minimizing composite objective function, 2007.

[16] Brendan O'Donoghue and Emmanuel Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, pages 1–18, 2013.

[17] Mark Schmidt. L1General: a set of Matlab routines implementing several of the available strategies for solving L1-regularization problems. `http://www.di.ens.fr/~mschmidt/Software/L1General.html`.

[18] Mark Schmidt. minFunc: unconstrained differentiable multivariate optimization in Matlab. `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`.

[19] Xinhua Zhang, Ankan Saha, and SVN Vishwanathan. Regularized risk minimization by Nesterov's accelerated gradient methods: Algorithmic extensions and empirical studies. *arXiv preprint arXiv:1011.0472*, 2010.