Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Разработка метода стохастической оптимизации для задач машинного обучения с большими данными

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Выполнил:
студент 4 курса 417 группы
Родоманов Антон Олегович

Научный руководитель:
к.ф-м.н., доцент
Ветров Дмитрий Петрович,
научный сотрудник
Кропотов Дмитрий Александрович

Москва, 2015

Аннотация

Рассматривается решение задачи оптимизации, возникающей при обучении моделей машинного обучения для случая большого объёма обучающей выборки. Классические подходы для этой задачи в классе стохастической оптимизации первого порядка обладают лишь линейной скоростью сходимости. В работе предлагается метод стохастической оптимизации второго порядка типа Ньютона. Идея метода заключается в рассмотрении общей квадратичной модели для оптимизируемой функции с дальнейшим обновлением в итерациях её отдельных компонент. В результате получается метод оптимизации, обладающий суперлинейной скоростью сходимости. Данный результат подтверждается как теоретически, так и экспериментально.

Содержание

1. Введение

Многие задачи, возникающие в машинном обучении, являются задачами оптимизации вида

$$\min_{\mathbf{w}\in\mathbb{R}^D} \quad \left[F(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w}) + \Omega(\mathbf{w}) \right] \tag{1}$$

где $f_i : \mathbb{R}^D \to \mathbb{R}$, $i = 1, \ldots, N$ и Ω — всюду определенные функции, а $\mathbf{w} \in \mathbb{R}^D$ — оптимизируемые переменные. Типичным примером может служить задача настройки параметрической модели по данным: в этом случае переменные $\mathbf{w}$ соответствуют параметрам модели, количество параметров равно D, значение $f_i(\mathbf{w})$ измеряет ошибку модели на i-м объекте обучающей выборки, число N обозначает общее число объектов обучения, а величина $\Omega(\mathbf{w})$ является регуляризатором (штрафным слагаемым, помогающим избежать переобучения модели)[1].

Всюду в дальнейшем будем считать, что все функции f_i являются *дважды непрерывно дифференцируемыми*. Также, несмотря на то, что метод можно применять и в более общих предположениях, для простоты дальнейшего изложения дополнительно будем считать, что все f_i являются *выпуклыми*, а в качестве регуляризатора Ω используется ℓ_2*-регуляризатор*:

$$\Omega(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

где $\lambda > 0$ — заданный коэффициент регуляризации. В таких предположениях оптимизируемая функция F является всюду определенной, дважды непрерывно дифференцируемой и *сильно выпуклой*. Как следствие, задача (1) имеет единственное решение.

В данной работе рассматривается случай, когда число слагаемых N в задаче (1) очень большое (например, несколько миллионов). В такой ситуации для минимизации функции F удобно использовать т. н. *инкрементальные* методы оптимизации, стоимость итерации которых не зависит от N. В отличие от стандартных методов, которые на каждой итерации вычисляют все N функций, инкрементальные методы на каждой итерации вычисляют лишь одну отдельную компоненту f_i. Если каждый инкрементальный шаг приводит к достаточному прогрессу в некотором «среднем» смысле, то, в зависимости от числа N, инкрементальный метод может сильно опережать свой неинкрементальный аналог [Bertsekas, 2011].

Одним из наиболее известных инкрементальных методов оптимизации является метод *стохастического градиента* (Stochastic Gradient Descent, SGD) [Robbins and Monro, 1951], который использует итерации вида

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \left(\nabla f_{i_k}(\mathbf{w}_k) + \lambda \mathbf{w}_k \right),$$

где $i_k \in \{1, \ldots, N\}$ — случайно выбранный индекс функции, а α_k — положительная длина шага[2]. Преимуществами этого метода являются минимальные требования к функциям f_i (например, не требуется выпуклости), использование $O(1)$ дополнительной памяти, низкая сложность итерации $O(D)$ и простота реализации. Ценой всех этих преимуществ является медленная скорость сходимости: можно показать, что если функции f_i удовлетворяют некоторым стандартным предположениям [см. Bertsekas, 2011], а последовательность длин шагов $\{\alpha_k\}$ выбрана подходящим образом, то метод SGD имеет лишь *сублинейную* скорость сходимости.

Более эффективным инкрементальным методом является метод *стохастического среднего градиента* (Stochastic Average Gradient, SAG) [Schmidt et al., 2013]. Итерации этого метода имеют вид

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \left(\mathbf{g}_k + \lambda \mathbf{w}_k \right),$$

где α_k — положительная длина шага, а $\mathbf{g}_k$ — вектор «среднего» градиента, обновляемый в итерациях по принципу «прибавить-вычесть»:

$$\mathbf{g}_k = \mathbf{g}_{k-1} + \frac{1}{N}\left(\nabla f_{i_k}(\mathbf{w}_k) - \mathbf{y}_{i_k}^{k-1} \right), \qquad \mathbf{y}_{i_k}^{k-1} := \nabla f_{i_k}(\mathbf{v}_{i_k}^{k-1}).$$

[1] Отметим, что на практике для ускорения оптимизации объекты обучающей выборки обычно группируют в т. н. *мини-батчи* (небольшие наборы некоторого размера, например, 10 или 100). В этом случае число N может соответствовать общему числу мини-батчей, а отдельная функция f_i — сумме функций потерь модели на объектах из соответствующего мини-батча.

[2] Строго говоря, метод SGD относится к категории *стохастических* методов оптимизации, а не инкрементальных, поскольку в общем виде не предполагает конечности числа слагаемых в сумме (1). Тем не менее, в применении конкретно к задаче (1) метод SGD можно рассматривать как инкрементальный градиентный метод со случайным порядком выбора компонент. См. также обсуждение в [Bertsekas, 2011, с. 7–8].

Здесь $i_k \in \{1, \dots, N\}$ — случайно выбранный индекс функции, а $\mathbf{v}_{i_k}^{k-1}$ — точка, в которой последний раз вычислялся градиент выбранной функции (т. е. после обновления эта точка заменится на текущую, $\mathbf{v}_{i_k}^k = \mathbf{w}_k$, а все остальные не изменятся, $\mathbf{v}_i^k = \mathbf{v}_i^{k-1}$, $i \neq i_k$). Таким образом, на каждом шаге метода SAG для аппроксимации градиента $\nabla F(\mathbf{w}_k)$ полной функции F используется некоторый «средний» градиент $\mathbf{g}_k = (1/N) \sum_{i=1}^N \nabla f_i(\mathbf{v}_i^k)$, в котором участвуют *все* функции f_i (пусть и в разных точках), а не только *одна* единственная f_{i_k}, как в методе SGD. Также, в отличие от метода SGD, метод SAG использует *дополнительную память* для хранения градиентов $\mathbf{y}_i$ отдельных функций. При отсутствии каких-либо предположений о структуре функций f_i для такого хранения требуется $O(ND)$ памяти. За счет использования дополнительной памяти, итерация метода SAG имеет такую же сложность, как и итерация метода SGD, т. е. $O(D)$. Однако скорость сходимости метода SAG уже на порядок выше, чем у метода SGD: можно показать, что если градиенты функций f_i удовлетворяют условию Липшица, то метод SAG имеет *линейную* скорость сходимости [Schmidt et al., 2013].

В данной работе предлагается новый инкрементальный метод для оптимизации функций вида (1), называемый *инкрементальным методом Ньютона* (Incremental Newton, IN). В отличие от методов SGD и SAG, использующих о функциях f_i информацию только *первого* порядка (о градиентах), метод IN дополнительно использует информацию *второго* порядка (о гессианах). За счет этого метод IN имеет более высокие требования по памяти и более высокую стоимость итерации, чем SGD и SAG. Тем не менее, сложность итерации метода IN по-прежнему *не зависит от числа функций N*. Также, в достаточно общих предположениях, скорость сходимости метода IN *суперлинейная*, что на порядок выше, чем у метода SAG и на два порядка выше, чем у SGD. В дополнение к этому, интересной особенностью метода является то, что в терминах сходимости по т. н. *эпохам* (т. е. каждые N шагов, $\mathbf{w}_0, \mathbf{w}_N, \mathbf{w}_{2N}, \dots$) метод имеет *квадратичную* скорость сходимости. Подобный эффект отсутствует у методов SGD и SAG; повышения порядка скорости сходимости в терминах эпох у этих методов не происходит.

2. Инкрементальный метод Ньютона

2.1. Квадратичная модель

Основным объектом в методе IN является *квадратичная модель* Q^k полной функции F. Эта модель зависит от номера текущей итерации k. Опишем, как она строится.

Сначала построим квадратичную модель q_i^k каждой отдельной функции f_i. Для этого разложим f_i в ряд Тейлора до второго порядка с центром в точке $\mathbf{v}_i^k \in \mathbb{R}^D$:

$$f_i(\mathbf{w}) \approx q_i^k(\mathbf{w}) := f_i(\mathbf{v}_i^k) + \nabla f_i(\mathbf{v}_i^k)^\top (\mathbf{w} - \mathbf{v}_i^k) + \frac{1}{2}(\mathbf{w} - \mathbf{v}_i^k)^\top \nabla^2 f_i(\mathbf{v}_i^k)(\mathbf{w} - \mathbf{v}_i^k).$$

Точка $\mathbf{v}_i^k$, центр разложения, является параметром модели q_i^k, и обновляется в итерациях.

Имея квадратичные модели всех функций f_i, легко получить квадратичную модель полной функции F:

$$F(\mathbf{w}) \approx Q^k(\mathbf{w}) := \frac{1}{N} \sum_{i=1}^N q_i^k(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 . \tag{2}$$

Параметрами этой модели являются N центров разложения $\mathbf{v}_i^k$, $i = 1, \dots, N$. Если все центры разложения совпадают, $\mathbf{v}_1^k = \dots = \mathbf{v}_N^k =: \mathbf{v}^k$, то Q^k совпадает с аппроксимаций Тейлора второго порядка полной функции F с центром в точке $\mathbf{v}^k$. Однако, в общем случае, когда центры разложения $\mathbf{v}_i^k$ различные, это не так.

Заметим, что поскольку все функции f_i являются выпуклыми, то и их квадратичные модели q_i^k тоже являются выпуклыми. Кроме этого, за счет квадратичного регуляризатора полная модель Q^k является строго выпуклой и имеет единственный минимум.

2.2. Итерация метода

Итерация метода IN осуществляется в два этапа: сначала происходит обновление модели Q^k, затем выполняется шаг метода.

1. **Обновление модели**. На этом этапе старая модель Q^{k-1} обновляется до новой Q^k. При этом обновление выполняется *инкрементально*, т. е. вместо обновления *всех* компонент q_i^{k-1} одновременно, обновляется

лишь *одна* из них, с номером $i_k \in \{1, \ldots, N\}$. Обновление i_k-ой модели заключается в смещении ее старого центра $\mathbf{v}_{i_k}^{k-1}$ в текущую точку $\mathbf{w}_k$. Это можно записать в следующем виде:

$$\mathbf{v}_i^k := \begin{cases} \mathbf{w}_k, & i = i_k, \\ \mathbf{v}_i^{k-1}, & i \neq i_k. \end{cases} \tag{3}$$

В результате, итогом обновления модели является ее уточнение в текущей точке $\mathbf{w}_k$.

2. **Шаг метода**. После обновления модели осуществляется поиск ее минимума:

$$\bar{\mathbf{w}}_k := \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} Q^k(\mathbf{w}).$$

Далее выполняется шаг метода в направлении найденного минимума модели:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k(\bar{\mathbf{w}}_k - \mathbf{w}_k), \tag{4}$$

где α_k — положительная длина шага.

2.3. Минимизация модели

Приравнивая градиент модели Q^k к нулю, можно получить следующее выражение для точки минимума $\bar{\mathbf{w}}_k$:

$$\bar{\mathbf{w}}_k = (\mathbf{H}_k + \lambda \mathbf{I})^{-1} (\mathbf{p}_k - \mathbf{g}_k), \tag{5}$$

где введены обозначения

$$\mathbf{H}_k := \frac{1}{N} \sum_{i=1}^{N} \nabla^2 f_i(\mathbf{v}_i^k), \qquad \mathbf{p}_k := \frac{1}{N} \sum_{i=1}^{N} \nabla^2 f_i(\mathbf{v}_i^k) \mathbf{v}_i^k, \qquad \mathbf{g}_k := \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{v}_i^k). \tag{6}$$

Согласно этим формулам, вектор $\mathbf{g}_k$ — это в точности «средний» градиент, который используется в методе SAG. По аналогии, величины $\mathbf{H}_k$ и $\mathbf{p}_k$ можно назвать «средним» гессианом и «средним» шкалированным центром.

Заметим, что для нахождения точки $\bar{\mathbf{w}}_k$ необязательно на каждой итерации совершать полный проход по всем N функциям. Поскольку метод инкрементальный, и на каждом шаге обновляется лишь один центр $\mathbf{v}_i^k$, то во всех суммах в (6) на каждой итерации меняется лишь одно слагаемое. Поэтому, аналогично тому, как это делается в методе SAG, величины $\mathbf{H}_k$, $\mathbf{p}_k$ и $\mathbf{g}_k$ можно обновлять в итерациях по принципу «прибавить-вычесть»:

$$\begin{aligned} \mathbf{H}_k &= \mathbf{H}_{k-1} + \frac{1}{N} \left(\nabla^2 f_{i_k}(\mathbf{w}_k) - \nabla^2 f_{i_k}(\mathbf{v}_{i_k}^{k-1}) \right), \\ \mathbf{p}_k &= \mathbf{p}_{k-1} + \frac{1}{N} \left(\nabla^2 f_{i_k}(\mathbf{w}_k) \mathbf{w}_k - \nabla^2 f_{i_k}(\mathbf{v}_{i_k}^{k-1}) \mathbf{v}_{i_k}^{k-1} \right), \\ \mathbf{g}_k &= \mathbf{g}_{k-1} + \frac{1}{N} \left(\nabla f_{i_k}(\mathbf{w}_k) - \nabla f_{i_k}(\mathbf{v}_{i_k}^{k-1}) \right), \end{aligned} \tag{7}$$

где $i_k \in \{1, \ldots, N\}$ — номер обновляемой компоненты на текущей итерации.

Отметим, что для осуществления обновлений по формулам (7) методу необходимо использовать память. При этом имеется два варианта:

1. **Хранить все компоненты**. Эта схема является прямым обобщением схемы хранения, которую использует SAG. В этом случае в память сохраняются все компоненты, участвующие в обновлениях в формулах (7), конкретно:
   - отдельные гессианы $\nabla^2 f_i(\mathbf{v}_i^k),\ i = 1, \ldots, N$;
   - отдельные шкалированные центры $\nabla^2 f_i(\mathbf{v}_i^k)\mathbf{v}_i^k,\ i = 1, \ldots, N$;
   - отдельные градиенты $\nabla f_i(\mathbf{v}_i^k),\ i = 1, \ldots, N$;
   - сами обновляемые величины: $\mathbf{H}_k$, $\mathbf{p}_k$ и $\mathbf{g}_k$.

В условиях отсутствия каких-либо предположений о структуре функций f_i хранение отдельных гессианов требует $O(ND^2)$ памяти, отдельных шкалированных центров и градиентов — $O(ND)$ памяти, «среднего» гессина $\mathbf{H}_k$ — $O(D^2)$ памяти, «среднего» шкалированного центра $\mathbf{p}_k$ и «среднего» градиента $\mathbf{g}_k$ — $O(D)$ памяти. Итого суммарный объем требуемой памяти для этой схемы составляет $O(ND^2)$.

2. **Хранить только центры**. В этой схеме помимо самих обновляемых величин $\mathbf{H}_k$, $\mathbf{p}_k$ и $\mathbf{g}_k$ предлагается хранить в памяти только центры $\mathbf{v}_i^k$, $i = 1, \ldots, N$, а при обновлениях по формулам (7) считать вычитаемые величины *заново*. Недостатком этого подхода по сравнению с предыдущим является то, что в итоге все величины в формулах (7) будут вычислены по два раза: первый раз при прибавлении и второй раз при вычитании, когда та же компонента будет выбрана снова. Тем не менее, в этом случае объем требуемой памяти сильно сокращается и составляет $O(ND + D^2)$.

Пренебрегая небольшими накладными расходами по вычислению некоторых величин дважды во второй схеме, получаем, что, в отсутствии каких-либо предположений о структуре функций f_i вторая схема является более предпочтительной, чем первая[3]. Поэтому в общем случае рекомендуется использовать именно схему «хранить только центры».

2.4. Порядок обновления компонент

Существует как минимум два стандартных способа выбора номеров i_k обновляемых компонент в итерациях инкрементального метода: *циклический* и *случайный* [Bertsekas, 2011]. В первом случае номера i_k выбираются по циклу от 1 до N, т. е. $(i_0, i_1, \ldots, i_{N-1}; i_N, i_{N+1}, \ldots, i_{2N-1}; \ldots) = (1, 2, \ldots, N; 1, 2, \ldots, N; \ldots)$. Во втором случае i_k выбираются случайно, обычно из равномерного распределения на множестве $\{1, \ldots, N\}$, независимо от номера итерации.

Оба метода SGD и SAG используют *случайный* порядок обновления компонент. Использование в методе SGD циклического порядка вместо случайного приводит к более медленной скорости сходимости [Bertsekas, 2011]. Если в методе SAG заменить случайный порядок на циклический, то получится метод *инкрементального агрегированного градиента* (incremental aggregated gradient, IAG) [Blatt et al., 2007], который, аналогично ситуации с SGD, имеет более медленную сходимость, чем SAG [Schmidt et al., 2013].

Как показывают численные эксперименты, в инкрементальных методах, строящих модель оптимизируемой функции, ситуация иная. В подобных методах крайне важно, чтобы модель оптимизируемой функции в окрестности текущей точки $\mathbf{w}_k$ была как можно точнее. Случайный порядок обновления компонент (независимо из равномерного распределения) приводит к тому, что за цикл из последовательных N итераций некоторые компоненты модели обновляются несколько раз, в то время как другие компоненты могут ни разу так и не обновиться. В результате при неудачном выборе обновляемых компонент новая модель может быть не точной в окрестности текущей точки $\mathbf{w}_k$ за счет того, что «важные» компоненты могли быть так и не обновлены. Поскольку заранее неизвестно, какие компоненты модели являются более «важными», чем другие, то более надежной стратегией в таком случае является циклический порядок обновления, обеспечивающий за цикл из последовательных N итераций *ровно одно* обновление, но зато *каждой* компоненты модели.

С инженерной точки зрения для всех методов более эффективной стратегией является циклическая схема. Например, пусть функция f_i измеряет величину потери модели на i-м объекте обучающей выборки. Если сама обучающая выборка хранится в памяти, то метод, использующий случайный порядок обновления компонент, на каждой итерации будет выполнять запросы в различные области памяти и, как следствие, страдать от регулярных промахов кэша. Аналогично, в ситуации, когда объем данных слишком большой для их полного сохранения в память, и обучающая выборка хранится на жестком диске (hard disk drive, HDD), то при использовании случайного порядка придется выполнять многократные физические перемещения головки диска; время этих перемещений может доминировать время самих итераций метода[4]. Циклическая схема обновления компонент лишена подобного рода недостатков.

[3] В разделе 4 будет рассмотрен особый класс функций f_i, часто используемый в задачах машинного обучения, — класс *линейных моделей*. За счет учета специальной структуры функций f_i для этого класса оказывается возможным хранить отдельные компоненты модели *неявным образом*. В этом случае не нужно вычислять дважды одни и те же величины, а суммарный объем используемой памяти составляет $O(N + D^2)$.

[4] Отметим, что при использовании твердотельных накопителей (solid-state drive, SSD) ситуация совсем иная. В этих устройствах случайные считывания являются довольно эффективными. Тем не менее, все это справедливо лишь в том случае, когда заранее известны байтовые позиции объектов обучающей выборки (например, если обучающая выборка хранится на диске в бинарном формате).

В результате, в методе IN рекомендуется использовать именно *циклический* порядок обновления компонент.

2.5. Инициализация

Согласно разделу 2.2, каждая итерация метода IN заключается в обновлении предыдущей модели Q^{k-1} до новой версии Q^k и дальнейшей минимизации Q^k. Тем не менее, пока что ничего не было сказано о том, чем инициализировать модель на самой первой итерации, когда $k = 0$.

Для инициализации модели предлагаются следующие две стратегии:

1. **Полный Ньютон**. В этом случае все первоначальные центры $\mathbf{v}_i^{-1}$ инициализируются начальной точкой $\mathbf{w}_0$: $\mathbf{v}_i^{-1} := \mathbf{w}_0,\ i = 1, \ldots, N$. Согласно формулам (4), (5) и (6), в результате такой инициализации первый шаг метода будет полностью эквивалентен шагу стандартного метода Ньютона для полной функции F из точки $\mathbf{w}_0$. Минусом этой стратегии является высокая стоимость самой первой итерации: для нахождения минимума $\bar{\mathbf{w}}_0$ модели необходимо будет вычислить градиенты и гессианы всех функций $f_i,\ i = 1, \ldots, N$ в точке $\mathbf{w}_0$.

2. **Самоинициализация**. В этой стратегии первоначальная модель Q^{-1} выбирается пустой (не содержащей ни одной компоненты), и затем наращивается в итерациях метода. При этом под отсутствием компоненты с номером i в модели Q^k понимается то, что в соответствующей сумме (2) отсутствует слагаемое с номером i. Наращивание модели осуществляется по следующему правилу: если компонента с номером i_k отсутствует в модели Q^{k-1}, то происходит ее добавление, т. е. в сумму (2) добавляется i_k-ая модель с центром в точке $\mathbf{w}_k$; в противном случае, если компонента с номером i_k присутствует в модели Q^{k-1}, то происходит стандартное обновление центра i_k-ой модели по формуле (3). По-другому, в терминах величин из раздела 2.3, данный процесс наращивания можно понимать как обновление компонент $\mathbf{H}_k$, $\mathbf{p}_k$ и $\mathbf{g}_k$ по формулам (7) с оговоркой, что если предыдущего центра $\mathbf{v}_i^{k-1}$ не существует, то соответствующую компоненту вычитать не нужно. Таким образом, при такой схеме инициализации модель Q^0 будет состоять лишь из одной компоненты с номером i_0 и центром в точке $\mathbf{w}_0$. На следующей итерации произойдет выбор нового индекса i_1; если $i_1 \neq i_0$, то модель Q^1 будет состоять уже из двух компонент: компоненты с номером i_0 и центром в точке $\mathbf{w}_0$ и компоненты с номером i_1 и центром в точке $\mathbf{w}_1$; если же $i_1 = i_0$[5], то произойдет обновление центра i_0-й модели, и Q^1 будет по-прежнему состоять из одной компоненты с номером i_0, но с новым центром $\mathbf{w}_1$; и т. д. Можно сказать, что при данной схеме инициализация модели происходит автоматически. При этом, в отличие от предыдущей стратегии, такая инициализация не повышает стоимость начальных итераций метода.

Поскольку в стратегии «полный Ньютон» стоимость инициализации очень высокая, а ее результатом является лишь *один* шаг метода, то по-умолчанию рекомендуется использовать вторую стратегию, т. е. «самоинициализацию».

2.6. Критерий остановки

Одним из недостатков некоторых инкрементальных методов (например, SGD) является отсутствие эффективного *критерия остановки*. В частности, поскольку инкрементальные методы не вычисляют полный градиент $\nabla F(\mathbf{w}_k)$ функции F в каждой точке $\mathbf{w}_k$, то использование такого популярного критерия остановки, как сравнение нормы градиента $\|\nabla F(\mathbf{w}_k)\|$ с некоторой заданной точностью оптимизации ε становится невозможным. При этом естественная идея оценить полный градиент $\nabla F(\mathbf{w}_k)$, используя только градиент одной отдельной функции $\nabla f_i(\mathbf{w}_k)$, как это делается в методе SGD, и сравнивать норму $\|\nabla f_i(\mathbf{w}_k) + \lambda \mathbf{w}_k\|$ с требуемой точностью ε к успеху не приводит, например, из-за того, что если последовательность полных градиентов $\{\nabla F(\mathbf{w}_k)\}$ сходится к нулю, то последовательность их оценок $\{\nabla f_i(\mathbf{w}_k) + \lambda \mathbf{w}_k\}$, как правило, к нулю сходиться не будет.

В методе SAG проблемы с критерием остановки не возникает, поскольку в методе присутствует «средний» градиент

$$\mathbf{g}_k := \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{v}_i^k),$$

[5]Такое возможно, если, например, используется случайный порядок обновления компонент.

который по построению является асимптотически точной оценкой истинного градиента суммы функций в точке $\mathbf{w}_k$. Действительно, считая, что все центры метода в итоге сходятся к оптимуму $\mathbf{w}_*$, т. е. $\mathbf{v}_i^k \to \mathbf{w}_*,\ i = 1, \ldots, N$, получаем, что $\mathbf{w}_k \to \mathbf{w}_*$ (т. к. по построению $\mathbf{w}_k$ совпадает с одним из центров). Отсюда $\mathbf{v}_i^k - \mathbf{w}_k \to 0,\ i = 1, \ldots, N$. Предполагая дополнительно, что градиенты функций f_i удовлетворяют условию Липшица с константой $L > 0$, получаем, что

$$\left\| \mathbf{g}_k - \frac{1}{N} \nabla f_i(\mathbf{w}_k) \right\| \leq \frac{1}{N} \sum_{i=1}^{N} \left\| \nabla f_i(\mathbf{v}_i^k) - \nabla f_i(\mathbf{w}_k) \right\| \leq \frac{L}{N} \sum_{i=1}^{N} \left\| \mathbf{v}_i^k - \mathbf{w}_k \right\| \longrightarrow 0,$$

т. е. $\mathbf{g}_k$ есть асимптотически точная оценка $(1/N) \sum_{i=1}^{N} \nabla f_i(\mathbf{w}_k)$. Итак, в качестве критерия остановки в методе SAG можно использовать следующее условие [Schmidt et al., 2013]:

$$\left\| \mathbf{g}_k + \lambda \mathbf{w}_k \right\| \leq \varepsilon, \tag{8}$$

где $\|\cdot\|$ — некоторая векторная норма, например, ℓ_∞ или ℓ_2.

Заметим, что в методе IN тоже присутствует «средний» градиент $\mathbf{g}_k$ (см. раздел 2.3). Это обстоятельство позволяет использовать в методе IN критерий остановки (8).

2.7. Длина шага

Поскольку метод IN использует модель оптимизируемой функции, то, аналогично ситуации с методом Ньютона, в методе IN автоматически возникает понятие «естественной» длины шага $\alpha_k = 1$. В этом случае шаг метода выполняется в точности в точку минимума модели, $\mathbf{w}_{k+1} = \bar{\mathbf{w}}_k$. Единичный шаг является оптимальным выбором, когда текущая точка $\mathbf{w}_k$ находится достаточно близко к оптимуму $\mathbf{w}_*$, поскольку только при такой длине шага метод сходится суперлинейно (см. раздел 3). Однако, в общем случае для обеспечения глобальной сходимости метода может понадобится использовать длину шага, отличную от единицы.

2.8. Итоговый алгоритм

Реализация метода IN с учетом всех предыдущих замечаний представлена в алгоритме 1.

2.9. Используемая память и сложность итерации

Метод IN хранит в памяти и обновляет в итерациях следующие величины:

1. все центры $\mathbf{v}_i^k,\ i = 1, \ldots, N$ (требуется $O(ND)$ памяти);
2. «средний» гессиан $\mathbf{H}_k$ (требуется $O(D^2)$ памяти);
3. «средний» шкалированный центр $\mathbf{p}_k$ (требуется $O(D)$ памяти);
4. «средний» градиент $\mathbf{g}_k$ (требуется $O(D)$ памяти).

Общий объем используемой памяти составляет $O(ND + D^2)$.

Стоимость итерации метода IN складывается из следующих двух компонент:

1. **Обновление модели**. Согласно формулам (7), для обновления модели требуется вычислить градиент и гессиан i_k-ой функции, а затем выполнить несколько матричных и векторных операций над матрицами и векторами размера D. Обозначив стоимость вычисления градиента и гессиана отдельной функции за C_2, получаем, что сложность обновления модели составляет $O(C_2 + D^2)$.
2. **Минимизация модели**. Поиск минимума модели осуществляется по формуле (5). При этом матрица $\mathbf{H}_k$ и векторы $\mathbf{p}_k$ и $\mathbf{g}_k$ хранятся в памяти. Таким образом, минимизация модели сводится к решению системы линейных уравнений размером D. Будем считать, что сложность этой операции составляет $O(D^3)$.

Итого сложность итерации составляет $O(C_2 + D^3)$.

Краткий итог относительно методов SGD, SAG и IN подведен в таблице 1.

Вход : 1) $\mathbf{w}_0 \in \mathbb{R}^D$: начальная точка;
2) N: общее число функций; 3) $\lambda > 0$: коэффициент регуляризации;
4) $K \in \mathbb{N}$: макс. число итераций; 5) $\varepsilon > 0$: точность оптимизации;

Выход: $\mathbf{w} \in \mathbb{R}^D$: решение задачи (1) в пределах заданной точности ε.

/* Инициализация модели (схема <<самоинициализация>>) */

$\mathbf{H} := 0 \in \mathbb{R}^{D \times D}$; $\mathbf{p} := 0 \in \mathbb{R}^D$; $\mathbf{g} := 0 \in \mathbb{R}^D$;

$\mathbf{v}_i :=$ undefined, $i = 1, \ldots, N$;

/* Основной цикл */

$\mathbf{w} := \mathbf{w}_0$; $i := -1$;

for $k = 0, 1, \ldots, K-1$ **do**

  $i := (i+1) \bmod N$; // Выбор обновляемой компоненты: циклическая схема

  {Вычислить $\nabla f_i(\mathbf{w})$ и $\nabla^2 f_i(\mathbf{w})$} ; // Вычисление градиента и гессиана в текущей точке

  $\mathbf{H} := \mathbf{H} + (1/N)\nabla^2 f_i(\mathbf{w})$; // Обновление <<среднего>> гессиана: прибавление

  $\mathbf{p} := \mathbf{p} + (1/N)\nabla^2 f_i(\mathbf{w})\mathbf{w}$; // Обновление <<среднего>> шкал. центра: прибавление

  $\mathbf{g} := \mathbf{g} + (1/N)\nabla f_i(\mathbf{w})$; // Обновление <<среднего>> градиента: прибавление

  if $\mathbf{v}_i \neq$ undefined **then** // Если i-ая компонента входит в текущую модель

    {Вычислить $\nabla f_i(\mathbf{v}_i)$ и $\nabla^2 f_i(\mathbf{v}_i)$} ; // Вычисление градиента и гессиана в прошлой точке

    $\mathbf{H} := \mathbf{H} - (1/N)\nabla^2 f_i(\mathbf{v}_i)$; // Обновление <<среднего>> гессиана: вычитание

    $\mathbf{p} := \mathbf{p} - (1/N)\nabla^2 f_i(\mathbf{v}_i)\mathbf{v}_i$; // Обновление <<среднего>> шкал. центра: вычитание

    $\mathbf{g} := \mathbf{g} - (1/N)\nabla f_i(\mathbf{v}_i)$; // Обновление <<среднего>> градиента: вычитание

  end

  $\mathbf{v}_i := \mathbf{w}$; // Сдвиг центра i-й модели в текущую точку

  if $\|\mathbf{g} + \lambda\mathbf{w}\|_\infty < \varepsilon$ **then break**; // Критерий остановки

  Решить систему уравнений $(\mathbf{H} + \lambda\mathbf{I})\bar{\mathbf{w}} = \mathbf{p} - \mathbf{g}$; найти точку $\bar{\mathbf{w}}$; // Поиск минимума модели

  {Выбрать длину шага $\alpha > 0$};

  $\mathbf{w} := \mathbf{w} + \alpha(\bar{\mathbf{w}} - \mathbf{w})$; // Шаг метода

end

return $\mathbf{w}$; // Возврат найденной точки

Алгоритм 1: Метод IN для оптимизации функций вида (1) в общих предположениях о функциях f_i.

Метод	Сложность итерации	Используемая память	Скорость сходимости	
			По итерациям	По эпохам
SGD	$O(C_1 + D)$	$O(D)$	Сублинейная	Сублинейная
SAG	$O(C_1 + D)$	$O(ND)$	Линейная	Линейная
IN	$O(C_2 + D^3)$	$O(ND + D^2)$	Суперлинейная	Квадратичная

Таблица 1: Сравнение методов SGD, SAG и IN для оптимизации функций вида (1) при отсутствии каких-либо предположений о структуре функций f_i. Обозначения: C_1 — стоимость вычисления градиента отдельной функции f_i; C_2 — стоимость вычисления градиента и гессиана отдельной функции f_i; N — общее число функций; D — число оптимизируемых переменных. «По эпохам» означает каждую N-ую итерацию.

3. Локальная скорость сходимости

В этом разделено приведено доказательство теоремы о том, что метод IN имеет суперлинейную локальную скорость сходимости по итерациям и квадратичную по эпохам.

Доказательство построено по следующей схеме:

1. Пусть $\tilde{r}_k := \|\mathbf{w}_k - \mathbf{w}_*\|_2$ — невязки метода IN, где $\mathbf{w}_*$ — точка оптимума функции F.
2. Показывается, что $\{\tilde{r}_k\}$ ограничена сверху следующей рекуррентной последовательностью (лемма 1):

$$\begin{aligned} r_k &:= \frac{R}{2\lambda N}(r_{k-1}^2 + r_{k-2}^2 + \cdots + r_{k-N}^2), && k = N, N+1, \ldots, \\ r_k &:= \tilde{r}_k, && k = 0, \ldots, N-1, \end{aligned}$$

   где $R > 0$ — некоторая константа.
3. Доказывается, что, $\{r_k\}$ является монотонно убывающей, начиная с номера $k = 2N$ (лемма 5):

$$r_{k+1} \leq r_k, \qquad k = 2N, 2N+1, \ldots.$$

4. Отсюда вытекает N-шаговая (т. е. по эпохам) квадратичная скорость сходимости (лемма 6):

$$r_k \leq \frac{R}{2\lambda} r_{k-N}^2, \qquad k = 3N, 3N+1, \ldots.$$

5. Далее с помощью лемм 7 и 8 строится мажорирующая последовательность $\{c_k\}$ для отношения r_{k+1}/r_k:

$$\frac{r_{k+1}}{r_k} \leq c_k, \quad k = 3N, 3N+1, \ldots,$$

   где $c_k \to 0$ при $k \to \infty$. Из этой оценки следует суперлинейная скорость сходимости по итерациям:

$$\lim_{k\to\infty} \frac{r_{k+1}}{r_k} = 0.$$

Лемма 2 является вспомогательной леммой, которая используется при доказательстве всех остальных лемм. Лемма 3 нужна для доказательства леммы 4, которая, в свою очередь, нужна для доказательства леммы 5. Финальный результат доказан в теореме 1.

3.1. Основная оценка

Приведенная ниже лемма дает рекуррентную оценку для последовательности невязок $\tilde{r}_k := \|\mathbf{w}_k - \mathbf{w}_*\|_2$. Доказательство этой леммы почти полностью повторяет доказательство теоремы о квадратичной скорости сходимости стандартного метода Ньютона [см., например, Nocedal and Wright, 2006, теорема 3.5].

***Лемма* 1** (основная оценка). *Пусть функции f_i, $i = 1, \ldots, N$ являются дважды непрерывно дифференцируемыми и выпуклыми, а их гессианы удовлетворяют условию Липшица:*

$$\left\|\nabla^2 f_i(\mathbf{w}) - \nabla^2 f_i(\mathbf{u})\right\|_2 \leq R_i \|\mathbf{w} - \mathbf{u}\|_2, \qquad \forall \mathbf{w}, \mathbf{u} \in \mathbb{R}^D,$$

где $R_i > 0$ — константы Липшица. Обозначим за $\{\mathbf{w}_k\}$ последовательность точек, построенную методом IN с единичным шагом и циклическим порядком обновления компонент, а за $\mathbf{w}_*$ точку оптимума функции (1). *Тогда последовательность невязок $\tilde{r}_k := \|\mathbf{w}_k - \mathbf{w}_*\|_2$ удовлетворяет следующему рекуррентному неравенству:*

$$\tilde{r}_k \leq \frac{R}{2\lambda N}(\tilde{r}_{k-1}^2 + \tilde{r}_{k-2}^2 + \cdots + \tilde{r}_{k-N}^2), \qquad k = N, N+1, \ldots, \tag{9}$$

где $R := \max\{R_i \mid i = 1, \ldots, N\}$.

Доказательство. Пусть $k \geq N-1$ — номер текущей итерации.

Поскольку шаг единичный, $\alpha_k = 1$, то, согласно формуле (4), $\mathbf{w}_{k+1} = \bar{\mathbf{w}}_k$, где $\bar{\mathbf{w}}_k$ — точка минимума модели Q^k. Применяя формулы (5) и (6), получаем следующую формулу для вычисления точки $\mathbf{w}_{k+1}$:

$$\mathbf{w}_{k+1} = \left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k)\mathbf{v}_i^k - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{v}_i^k)\right).$$

Используя оптимальное условие первого порядка, $0 = \nabla F(\mathbf{w}_*) = (1/N)\sum_{i=1}^{N}\nabla f_i(\mathbf{w}_*) + \lambda\mathbf{w}_*$, можно написать

$$\begin{aligned}\mathbf{w}_{k+1} - \mathbf{w}_* &= \left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k)\mathbf{v}_i^k - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{v}_i^k) - \frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k)\mathbf{w}_* - \lambda\mathbf{w}_*\right)\\ &= \left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k)(\mathbf{v}_i^k - \mathbf{w}_*) - \frac{1}{N}\sum_{i=1}^{N}\left[\nabla f_i(\mathbf{v}_i^k) - \nabla f_i(\mathbf{w}_*)\right]\right).\end{aligned}$$

Для разности градиентов применим формулу Тейлора:

$$\nabla f_i(\mathbf{v}_i^k) - \nabla f_i(\mathbf{w}_*) = \int_0^1 \nabla^2 f_i(\mathbf{w}_* + t(\mathbf{v}_i^k - \mathbf{w}_*))(\mathbf{v}_i^k - \mathbf{w}_*)\mathrm{d}t.$$

Значит,

$$\mathbf{w}_{k+1} - \mathbf{w}_* = \left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\frac{1}{N}\sum_{i=1}^{N}\int_0^1\left[\nabla^2 f_i(\mathbf{v}_i^k) - \nabla^2 f_i(\mathbf{w}_* + t(\mathbf{v}_i^k - \mathbf{w}_*))\right](\mathbf{v}_i^k - \mathbf{w}_*)\mathrm{d}t.$$

Перейдем к нормам и воспользуемся условием Липшица:

$$\tilde{r}_{k+1} \leq \left\|\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\right\|_2 \frac{R}{2N}\sum_{i=1}^{N}\left\|\mathbf{v}_i^k - \mathbf{w}_*\right\|_2^2. \tag{10}$$

Так как функции f_i выпуклые, то гессианы $\nabla^2 f_i(\mathbf{v}_i^k)$ являются неотрицательно определенными. Поэтому

$$\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right\|_2 \geq \lambda \qquad \text{и} \qquad \left\|\left(\frac{1}{N}\sum_{i=1}^{N}\nabla^2 f_i(\mathbf{v}_i^k) + \lambda\mathbf{I}\right)^{-1}\right\|_2 \leq \frac{1}{\lambda}. \tag{11}$$

Поскольку для обновления центров $\mathbf{v}_i^k$ используется циклический порядок, то

$$\sum_{i=1}^{N}\left\|\mathbf{v}_i^k - \mathbf{w}_*\right\|_2^2 = \tilde{r}_k^2 + \tilde{r}_{k-1}^2 + \cdots + \tilde{r}_{k-N+1}^2. \tag{12}$$

В итоге, подставляя (11) и (12) в неравенство (10), получаем оценку

$$\tilde{r}_{k+1} \leq \frac{R}{2\lambda N}(\tilde{r}_k^2 + \tilde{r}_{k-1}^2 + \cdots + \tilde{r}_{k-N+1}^2),$$

которая совпадает с (9) с точностью до сдвига индексов. □

3.2. Вспомогательные леммы

Дальнейшее доказательство основано на анализе следующей рекуррентной последовательности[6]:

$$r_k := \frac{C}{N}\left(r_{k-1}^2 + r_{k-2}^2 + \cdots + r_{k-N}^2\right), \qquad k = N, N+1, \ldots, \tag{13}$$

где $N \in \mathbb{N}$ и $C > 0$ — некоторые константы. Согласно (9), при $r_k := \tilde{r}_k$, $k = 0, \ldots, N-1$ и $C = R/(2\lambda)$ последовательность $\{r_k\}$ является верхней оценкой для последовательности $\{\tilde{r}_k\}$ невязок метода IN:

$$\tilde{r}_k \leq r_k, \qquad k = 0, 1, 2, \ldots.$$

[6]Предполагается, что начальные члены $r_0, r_1, \ldots, r_{N-1}$ этой последовательности являются неотрицательными.

Каждая из приведенных в этом разделе лемм доказывает один небольшой факт о последовательности $\{r_k\}$. Леммы упорядочены таким образом, что каждая следующая лемма опирается на предыдущие. Почти во всех леммах предполагается, что начальные члены $r_0, r_1, \ldots, r_{N-1}$ последовательности $\{r_k\}$ выбраны достаточно малыми. Это соответствует предположению о локальности (запуске метода IN из достаточно малой окрестности точки оптимума).

***Лемма* 2** (базовая оценка). *Для последовательности $\{r_k\}$ справедливы следующие две рекуррентные оценки:*

$$\frac{C}{N}\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2 \le r_k \le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2, \qquad k = N, N+1, \ldots.$$

Доказательство. Следует из определения (13) и того факта, что

$$\max\left\{r_{k-1}^2, r_{k-2}^2, \ldots, r_{k-N}^2\right\} = \max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2. \qquad \square$$

***Лемма* 3** (ограниченность). *Если начальные члены последовательности $\{r_k\}$ ограничены,*

$$\max\{r_0, r_1, \ldots, r_{N-1}\} \le \frac{1}{C\sqrt{N}},$$

то и все остальные члены этой последовательности тоже ограничены:

$$r_k \le \frac{1}{C\sqrt{N}}, \qquad k = 0, 1, 2, \ldots. \tag{14}$$

Доказательство. По индукции.

Пусть оценка (14) верна для всех номеров от 0 до k включительно. Докажем, что тогда она верна и для номера $k+1$.

Используя лемму 2 о базовой оценке и предположение индукции, получаем, что

$$r_{k+1} \le C\max\{r_k, r_{k-1}, \ldots, r_{k-N+1}\}^2 \le C\frac{1}{C^2N} = \frac{1}{CN} \le \frac{1}{C\sqrt{N}}. \qquad \square$$

***Лемма* 4** (блочная квадратичная сходимость). *Пусть начальные члены последовательности $\{r_k\}$ ограничены:*

$$\max\{r_0, r_1, \ldots, r_{N-1}\} \le \frac{1}{C\sqrt{N}}.$$

Тогда

$$\max\{r_{k+N-1}, r_{k+N-2}, \ldots, r_k\} \le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2, \qquad k = N, N+1, \ldots. \tag{15}$$

Доказательство. Фиксируем произвольный номер $k \ge N$.

Из леммы 3 об ограниченности следует, что

$$\max\left\{Cr_{k-1}^2, Cr_{k-2}^2, \ldots, Cr_{k-N}^2\right\} \le C\frac{1}{C\sqrt{N}}\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\} \le \max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}. \tag{16}$$

Согласно лемме 2 о базовой оценке,

$$r_k \le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2. \tag{17}$$

Используя неравенства (17) и (16), получаем:

$$\begin{aligned} r_{k+1} &\le C\max\{r_k, r_{k-1}, \ldots, r_{k-N+1}\}^2 \\ &\le C\max\left\{\max\left\{Cr_{k-1}^2, Cr_{k-2}^2, \ldots, Cr_{k-N}^2\right\}, r_{k-1}, \ldots, r_{k-N+1}\right\}^2 \\ &\le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2. \end{aligned} \tag{18}$$

Далее, из неравенств (18), (17) и (16):

$$\begin{aligned} r_{k+2} &\le C\max\{r_{k+1}, r_k, r_{k-1} \ldots, r_{k-N+2}\}^2 \\ &\le C\max\left\{\max\left\{Cr_{k-1}^2, Cr_{k-2}^2, \ldots, Cr_{k-N}^2\right\}, \max\left\{Cr_{k-1}^2, Cr_{k-2}^2, \ldots, Cr_{k-N}^2\right\}, r_{k-1}, \ldots, r_{k-N+2}\right\}^2 \\ &\le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2. \end{aligned}$$

Последовательно применяя аналогичные рассуждения для $r_{k+3}, \ldots, r_{k+N-1}$, получаем (15). $\square$

***Лемма* 5** (монотонность)**.** *Пусть начальные члены последовательности* $\{r_k\}$ *ограничены:*

$$\max\{r_0, r_1, \ldots, r_{N-1}\} \le \frac{1}{C\sqrt{N}}.$$

Тогда, начиная с номера $k = 2N$*, последовательность* $\{r_k\}$ *является монотонно убывающей:*

$$r_{k+1} \le r_k, \qquad k = 2N, 2N+1, \ldots.$$

Доказательство. Фиксируем произвольный номер $k \ge 2N$.

Заметим, что, согласно определению (13) последовательности $\{r_k\}$, неравенство $r_{k+1} \le r_k$ выполнено тогда и только тогда, когда выполнено неравенство $r_k \le r_{k-N}$. Поэтому достаточно доказать неравенство $r_k \le r_{k-N}$.

Применяя лемму 2 о базовой оценке и лемму 4 о блочной квадратичной сходимости, получаем:

$$\begin{aligned} r_k &\le C\max\{r_{k-1}, r_{k-2}, \ldots, r_{k-N}\}^2 \\ &\le C\left(C\max\{r_{k-N-1}, r_{k-N-2}, \ldots, r_{k-2N}\}^2\right)^2 \\ &= C^3\max\{r_{k-N-1}, r_{k-N-2}, \ldots, r_{k-2N}\}^4. \end{aligned}$$

Из леммы 2 о базовой оценке,

$$r_{k-N} \ge \frac{C}{N}\max\{r_{k-N-1}, r_{k-N-2}, \ldots, r_{k-2N}\}^2.$$

Сравнивая правые части последних двух неравенств и используя лемму 3 об ограниченности, получаем, что $r_k \le r_{k-N}$. Согласно сделанному в начале доказательства замечанию, отсюда следует монотонность. □

***Лемма* 6** (N-шаговая квадратичная скорость сходимости)**.** *Пусть начальные члены последовательности* $\{r_k\}$ *ограничены:*

$$\max\{r_0, r_1, \ldots, r_{N-1}\} \le \frac{1}{C\sqrt{N}}.$$

Тогда эта последовательность имеет N*-шаговую квадратичную скорость сходимости:*

$$r_k \le Cr_{k-N}^2, \qquad k = 3N, 3N+1, \ldots. \tag{19}$$

Доказательство. Следует из леммы 2 о базовой оценке и леммы 5 о монотонности. □

***Лемма* 7** (линейная скорость сходимости)**.** *Пусть начальные члены последовательности* $\{r_k\}$ *ограничены:*

$$\max\{r_0, r_1, \ldots, r_{N-1}\} \le \frac{1}{C\sqrt{N}}.$$

Тогда скорость сходимости этой последовательности как минимум линейная[7]*:*

$$r_{k+1} \le \left(1 - \frac{N-1}{N^2}\right) r_k, \qquad k = 3N, 3N+1, \ldots,$$

Доказательство. Фиксируем произвольный номер $k \ge 3N$.

Из леммы 6 об N-шаговой квадратичной скорости сходимости и леммы 3 об ограниченности

$$r_k \le (Cr_{k-N})\, r_{k-N} \le \frac{r_{k-N}}{\sqrt{N}}.$$

Отсюда

$$\begin{aligned} r_{k+1} &= \frac{C}{N}(r_k^2 + r_{k-1}^2 + \cdots + r_{k-N+1}^2) \\ &\le \frac{C}{N}\left(\frac{r_{k-N}^2}{N} + r_{k-1}^2 + \cdots + r_{k-N+1}^2\right) \\ &= \frac{C}{N}\left(\frac{r_{k-N}^2}{N} + r_{k-1}^2 + \cdots + r_{k-N+1}^2 + r_{k-N}^2 - r_{k-N}^2\right) \\ &= r_k - \frac{N-1}{N^2}Cr_{k-N}^2. \end{aligned}$$

[7] *Здесь подразумевается, что* $N \ge 2$*. В случае* $N = 1$ *утверждение тривиально и сразу же следует из определения* (13).

Комбинируя полученную оценку с оценкой (19), получаем требуемое неравенство:

$$r_{k+1} \le r_k - \frac{N-1}{N^2} r_k = \left(1 - \frac{N-1}{N^2}\right) r_k. \qquad \square$$

***Лемма* 8** (улучшение константы)**.** *Пусть последовательность $\{r_k\}$, начиная с номера k_0, имеет линейную скорость сходимости с константой $c_0 \in (0,1)$:*

$$r_{k+1} \le c_0 r_k, \qquad k = k_0, k_0+1, \ldots, \tag{20}$$

Тогда, начиная с номера $k_0 + N$, константу c_0 можно улучшить:

$$r_{k+1} \le c_0^2 r_k, \qquad k = k_0 + N, k_0 + N + 1, \ldots.$$

Доказательство. Фиксируем произвольный номер $k \ge k_0 + N$.

Воспользовавшись определением (13) последовательности $\{r_k\}$ и оценками (20), получаем, что

$$\begin{aligned} r_{k+1} &= \frac{C}{N}(r_k^2 + r_{k-1}^2 + \cdots + r_{k-N+1}^2) \\ &\le \frac{C}{N}(c_0^2 r_{k-1}^2 + c_0^2 r_{k-2}^2 + \cdots + c_0^2 r_{k-N}^2) \\ &= c_0^2 \frac{C}{N}(r_{k-1}^2 + r_{k-2}^2 + \cdots + r_{k-N}^2) \\ &= c_0^2 r_k. \end{aligned}$$

$\square$

3.3. Теорема о локальной скорости сходимости

Финальный результат относительно локальной скорости сходимости метода IN приведен ниже в теореме 1. Предположения теоремы полностью такие же, как и в лемме 1 об основной оценке. Единственное дополнительное предположение заключается в локальности (запуске метода из достаточно малой окрестности точки оптимума).

***Теорема* 1.** *Пусть функции $f_i,\ i = 1, \ldots, N$ являются дважды непрерывно дифференцируемыми и выпуклыми, а их гессианы удовлетворяют условию Липшица:*

$$\left\| \nabla^2 f_i(\mathbf{w}) - \nabla^2 f_i(\mathbf{u}) \right\|_2 \le R_i \left\| \mathbf{w} - \mathbf{u} \right\|_2, \qquad \forall \mathbf{w}, \mathbf{u} \in \mathbb{R}^D,$$

где $R_i > 0$ — константы Липшица. Обозначим за $\{\mathbf{w}_k\}$ последовательность точек, построенную методом IN с единичным шагом и циклическим порядком обновления компонент, а за $\mathbf{w}_*$ точку оптимума функции (1)*. Пусть также все центры модели инициализированы достаточно близко к оптимуму:*

$$\left\| \mathbf{v}_i^{-1} - \mathbf{w}_* \right\|_2 \le \frac{2\lambda}{R\sqrt{N}}, \qquad i = 1, \ldots, N,$$

где $R := \max\{R_i \mid i = 1, \ldots, N\}$. Тогда последовательность невязок $\tilde{r}_k := \left\| \mathbf{w}_k - \mathbf{w}_* \right\|_2$ ограничена сверху последовательностью $\{r_k\}$, имеющей суперлинейную скорость сходимости:

$$\lim_{k \to \infty} \frac{r_{k+1}}{r_k} = 0.$$

Более того, последовательность $\{r_k\}$ имеет N-шаговую квадратичную скорость сходимости:

$$r_{k+N} \le \frac{R}{2\lambda} r_k^2, \qquad k = 2N, 2N+1, \ldots.$$

Доказательство. Согласно лемме 1 об основной оценке, последовательность невязок $\{\tilde{r}_k\}$ метода IN ограничена сверху последовательностью $\{r_k\}$, заданной формулой (13), где $r_k := \tilde{r}_k,\ k = 0, \ldots, N-1$ и $C = R/(2\lambda)$.

Утверждение об N-шаговой квадратичной скорости сходимости доказано в лемме 6.

Докажем утверждение о суперлинейной скорости сходимости.[8] Используя лемму 7 о линейной скорости сходимости и лемму 8 об улучшении константы, можно выписать следующий набор оценок:

$$\begin{aligned}
\frac{r_{k+1}}{r_k} &\le \left(1 - \frac{N-1}{N^2}\right), && k = 3N, 3N+1, \ldots, \\
\frac{r_{k+1}}{r_k} &\le \left(1 - \frac{N-1}{N^2}\right)^2, && k = 4N, 4N+1, \ldots, \\
\frac{r_{k+1}}{r_k} &\le \left(1 - \frac{N-1}{N^2}\right)^4, && k = 5N, 5N+1, \ldots, \\
\frac{r_{k+1}}{r_k} &\le \left(1 - \frac{N-1}{N^2}\right)^8, && k = 6N, 6N+1, \ldots, \\
&\ldots
\end{aligned}$$

Объединяя все эти оценку в одну, получаем:

$$\frac{r_{k+1}}{r_k} \le \left(1 - \frac{N-1}{N^2}\right)^{2^{\lfloor k/N \rfloor - 3}}, \qquad k = 3N, 3N+1, \ldots.$$

Поскольку правая часть стремится к нулю при $k \to \infty$, то и $r_{k+1}/r_k \to 0$ при $k \to \infty$. □

4. Инкрементальный метод Ньютона для линейных моделей

Оказывается, что в случае рассмотрения особого класса функций — класса линейных моделей — в методе IN можно сократить объем дополнительно используемой памяти, а также понизить сложность итерации.

4.1. Линейные модели

Будем называть функцию F, определенную в (1), (регуляризованной) *линейной моделью*, если каждая из функций f_i имеет вид

$$f_i(\mathbf{w}) := \phi_i(\mathbf{x}_i^\top \mathbf{w}), \tag{21}$$

где $\phi_i : \mathbb{R} \to \mathbb{R}$ — одномерная функция, измеряющая величину ошибки ответа модели на i-м объекте обучающей выборки, а $\mathbf{x}_i \in \mathbb{R}^D$ — вектор признаков i-го объекта.

Линейные модели являются популярными в машинном обучении. В качестве примеров можно привести, как минимум, следующие модели:

1. **Линейная регрессия**: $\phi_i(t) := (t - y_i)^2$, где $y_i \in \mathbb{R}$ — вещественный отклик на i-м объекте.
2. **Логистическая регрессия**: $\phi_i(t) := \ln(1 + \exp(-y_i t))$, где $y_i \in \{-1, +1\}$ — бинарная метка класса i-го объекта.
3. **Машина опорных векторов (SVM)**: $\phi_i(t) := \max\{0, 1 - y_i t\}$, где $y_i \in \{-1, +1\}$ — бинарная метка класса i-го объекта.

Отметим, что, в отличие от первых двух функций, функция потерь SVM не является дифференцируемой. Далее, аналогично тому, как это было в предыдущих разделах, будем рассматривать только те модели, в которых все функции $\phi_i,\ i = 1, \ldots, N$ являются *дважды непрерывно дифференцируемыми* и *выпуклыми*.

4.2. Обновление модели

Напомним, что метод IN использует дополнительную память для хранения всех центров $\mathbf{v}_i^k,\ i = 1, \ldots, N$, квадратичной модели (2). Эти центры, в свою очередь, нужны для восстановления вычитаемых величин при обновлениях по формулам (7). Особая структура градиентов и гессианов функций f_i для линейных моделей позволяет провести эти обновления другим, более эффективным, способом.

[8] Будем считать, что $N \ge 2$. В противном случае утверждение теряет смысл в силу доказанной квадратичной скорости сходимости.

В случае линейных моделей градиенты и гессианы функций f_i имеют следующий вид:

$$\begin{aligned}\nabla f_i(\mathbf{w}) &= \phi_i'(\mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i,\\ \nabla^2 f_i(\mathbf{w}) &= \phi_i''(\mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i\mathbf{x}_i^\top.\end{aligned} \tag{22}$$

Выпишем формулы обновления (7) с учетом формул (22):

$$\begin{aligned}\mathbf{H}_k &= \mathbf{H}_{k-1} + \frac{1}{N}\left(\phi_{i_k}''(\mathbf{x}_{i_k}^\top \mathbf{w}_k)\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top - \phi_{i_k}''(\mathbf{x}_{i_k}^\top \mathbf{v}_{i_k}^{k-1})\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top\right),\\ \mathbf{p}_k &= \mathbf{p}_{k-1} + \frac{1}{N}\left(\phi_{i_k}''(\mathbf{x}_{i_k}^\top \mathbf{w}_k)\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top \mathbf{w}_k - \phi_{i_k}''(\mathbf{x}_{i_k}^\top \mathbf{v}_{i_k}^{k-1})\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top \mathbf{v}_{i_k}^{k-1}\right)\\ \mathbf{g}_k &= \mathbf{g}_{k-1} + \frac{1}{N}\left(\phi_{i_k}'(\mathbf{x}_{i_k}^\top \mathbf{w}_k)\mathbf{x}_{i_k} - \phi_{i_k}'(\mathbf{x}_{i_k}^\top \mathbf{v}_{i_k})\mathbf{x}_{i_k}\right).\end{aligned} \tag{23}$$

Поскольку векторы $\mathbf{x}_i,\ i = 1, \ldots, N$, фиксированы (хранятся в памяти или на диске), то величины, участвующие в этих формулах обновления, зависят от центров $\mathbf{v}_i^k,\ i = 1, \ldots, N$, *только через скалярные произведения*

$$\mu_i^k := \mathbf{x}_i^\top \mathbf{v}_i^k, \qquad i = 1, \ldots, N. \tag{24}$$

Действительно, используя введенные обозначения, формулы (23) можно переписать в следующем, более компактном, виде[9]:

$$\begin{aligned}\mathbf{H}_k &= \mathbf{H}_{k-1} + \frac{1}{N}\left(\phi_{i_k}''(\mu_{i_k}^k) - \phi_{i_k}''(\mu_{i_k}^{k-1})\right)\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top,\\ \mathbf{p}_k &= \mathbf{p}_{k-1} + \frac{1}{N}\left(\phi_{i_k}''(\mu_{i_k}^k)\mu_{i_k}^k - \phi_{i_k}''(\mu_{i_k}^{k-1})\mu_{i_k}^{k-1}\right)\mathbf{x}_{i_k},\\ \mathbf{g}_k &= \mathbf{g}_{k-1} + \frac{1}{N}\left(\phi_{i_k}'(\mu_{i_k}^k) - \phi_{i_k}'(\mu_{i_k}^{k-1})\right)\mathbf{x}_{i_k}.\end{aligned} \tag{25}$$

Таким образом, для линейных моделей вместо самих центров $\mathbf{v}_i^k,\ i = 1, \ldots, N$ можно хранить только соответствующие скалярные произведения $\mu_i^k,\ i = 1, \ldots, N$. Объем необходимой для этого памяти составляет $O(N)$ вместо $O(ND)$, как при хранении самих центров.[10]

Помимо самих скалярных произведений μ_i^k, можно дополнительно сохранять в память значения производных $\phi_i'(\mu_i^k)$ и $\phi_i''(\mu_i^k)$. Для этого по-прежнему понадобится лишь $O(N)$ памяти, однако это позволит не вычислять каждый раз заново величины $\phi_{i_k}'(\mu_{i_k}^{k-1})$ и $\phi_{i_k}''(\mu_{i_k}^{k-1})$ при обновлениях по формулам (25) (см. соответствующие комментарии к схеме «хранить все компоненты» в разделе 2.3).

4.3. Обновление обратной матрицы

Самой дорогой операцией в итерации метода IN является решение системы линейных уравнений для нахождения минимума модели $\bar{\mathbf{w}}_k$ (формула (5)). Согласно формулам (25), для линейных моделей матрицы системы на соседних итерациях различаются лишь на *одноранговую* матрицу. Этот факт позволяет полностью избавиться от этапа решения системы линейных уравнений на каждой итерации за счет эффективного пересчета обратной матрицы системы, аналогично тому, как это делается в квазиньютоновских методах [см., например, Nocedal and Wright, 2006, глава 6].

Введем обозначение для обратной матрицы системы:

$$\mathbf{B}_k := \left(\mathbf{H}_k + \lambda \mathbf{I}\right)^{-1}. \tag{26}$$

Зная матрицу $\mathbf{B}_k$, точку минимума модели $\bar{\mathbf{w}}_k$ можно найти с помощью матрично-векторного умножения:

$$\bar{\mathbf{w}}_k = \mathbf{B}_k(\mathbf{p}_k - \mathbf{g}_k). \tag{27}$$

[9] Здесь $\mu_{i_k}^k = \mathbf{x}_{i_k}^\top \mathbf{w}_k$, что согласуется с определением (24) в силу формулы обновления центров (3).

[10] Здесь имеется в виду память, используемая только для хранения скалярных произведений / центров. Для быстрого обновления по формулам (25) по-прежнему необходимо хранить сами обновляемые величины $\mathbf{H}_k$, $\mathbf{g}_k$ и $\mathbf{p}_k$, для чего дополнительно требуется $O(D^2)$ памяти.

Для получения формулы обновления матрицы $\mathbf{B}_k$, применим формулу Шермана–Моррисона [см., например, Golub and Van Loan, 2012, формула (2.1.5)] к выражению для пересчета $\mathbf{H}_k$ в (25):

$$\mathbf{B}_k = \mathbf{B}_{k-1} - \frac{\delta_k \mathbf{B}_{k-1}\mathbf{x}_{i_k}\mathbf{x}_{i_k}^\top \mathbf{B}_{k-1}}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{B}_{k-1}\mathbf{x}_{i_k}}, \tag{28}$$

где введено обозначение

$$\delta_k := \phi_{i_k}''(\mu_{i_k}^k) - \phi_{i_k}''(\mu_{i_k}^{k-1}). \tag{29}$$

Согласно формуле (28), обновление матрицы $\mathbf{B}_k$ заключается в ее одноранговой коррекции. Стоимость такой коррекции составляет $O(D^2)$. В итоге, за счет устранения этапа решения системы уравнений сложность итерации метода понижается с $O(D^3)$ до $O(D^2)$.

Выбор начальной матрицы $\mathbf{B}_{-1}$ зависит от используемого способа инициализации модели (см. раздел 2.5). Для схемы «полный Ньютон» $\mathbf{B}_{-1} := \nabla^2 F(\mathbf{w}_0)^{-1}$; для схемы «самоинициализация» $\mathbf{B}_{-1} := (1/\lambda)\mathbf{I}$. Как и раньше, по-умолчанию рекомендуется использовать схему «самоинициализация».

4.4. Обновление минимума модели

В текущей версии метода IN для линейных моделей основные усилия на итерации затрачиваются на операции, вовлекающие в себя обратную матрицу системы: одноранговая коррекция матрицы $\mathbf{B}_{k-1}$ по формуле (28) и вычисление минимума модели $\bar{\mathbf{w}}_k$ по формуле (27). При этом для осуществления одноранговой коррекции матрицы $\mathbf{B}_{k-1}$ необходимо вычислить вектор

$$\mathbf{z}_k := \mathbf{B}_{k-1}\mathbf{x}_{i_k}. \tag{30}$$

В результате, на каждой итерации метода нужно выполнить *два* матрично-векторных произведения: одно для вычисления $\mathbf{z}_k$ по формуле (30) и одно для вычисления минимума модели $\bar{\mathbf{w}}_k$ по формуле (27). Покажем как понизить стоимость итерации метода, сократив число матрично-векторных произведений до *одного*.[11]

Будем с помощью доступного одного матрично-векторного произведения вычислять вектор $\mathbf{z}_k$ по формуле (30), а минимум модели $\bar{\mathbf{w}}_k$ будем пересчитывать через вектор $\mathbf{z}_k$.

Выведем формулу для пересчета минимума модели $\bar{\mathbf{w}}_k$. Для упрощения выкладок введем следующее обозначение для разности коэффициентов перед $\mathbf{x}_{i_k}$ в формулах (25):

$$s_k := \left[\phi_{i_k}''(\mu_{i_k}^k)\mu_{i_k}^k - \phi_{i_k}'(\mu_{i_k}^k)\right] - \left[\phi_{i_k}''(\mu_{i_k}^{k-1})\mu_{i_k}^{k-1} - \phi_{i_k}'(\mu_{i_k}^{k-1})\right]. \tag{31}$$

Используя формулы (25) и (30), можно выписать следующую цепочку равенств:

$$\begin{aligned}
\bar{\mathbf{w}}_k &= \mathbf{B}_k(\mathbf{p}_k - \mathbf{g}_k) \\
&= \mathbf{B}_k\left([\mathbf{p}_{k-1} - \mathbf{g}_{k-1}] + \frac{s_k}{N}\mathbf{x}_{i_k}\right) \\
&= \mathbf{B}_k(\mathbf{p}_{k-1} - \mathbf{g}_{k-1}) + \frac{s_k}{N}\mathbf{B}_k\mathbf{x}_{i_k} \\
&= \left(\mathbf{B}_{k-1} - \frac{\delta_k \mathbf{z}_k\mathbf{x}_{i_k}^\top \mathbf{B}_{k-1}}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\right)(\mathbf{p}_{k-1} - \mathbf{g}_{k-1}) + \frac{s_k}{N}\left(\mathbf{B}_{k-1} - \frac{\delta_k \mathbf{z}_k\mathbf{z}_k^\top}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\right)\mathbf{x}_{i_k} \\
&= \bar{\mathbf{w}}_{k-1} - \frac{\delta_k \mathbf{x}_{i_k}^\top \bar{\mathbf{w}}_{k-1}}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\mathbf{z}_k + \frac{s_k}{N}\left(\mathbf{z}_k - \frac{\delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\mathbf{z}_k\right) \\
&= \bar{\mathbf{w}}_{k-1} + \frac{s_k - \delta_k \mathbf{x}_{i_k}^\top \bar{\mathbf{w}}_{k-1}}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\mathbf{z}_k.
\end{aligned}$$

Таким образом, мы получили следующую формулу для обновления минимума модели $\bar{\mathbf{w}}_k$ через вектор $\mathbf{z}_k$:

$$\bar{\mathbf{w}}_k = \bar{\mathbf{w}}_{k-1} + \frac{s_k - \delta_k \mathbf{x}_{i_k}^\top \bar{\mathbf{w}}_{k-1}}{N + \delta_k \mathbf{x}_{i_k}^\top \mathbf{z}_k}\mathbf{z}_k. \tag{32}$$

[11] Несмотря на то, что асимптотически сложность итерации метода остается прежней, $O(D^2)$, на практике такая модификация дает ускорение.

Заметим, что матричные операции в этой формуле отсутствуют.

Согласно определению, инициализацию нужно выполнять по следующей формуле: $\bar{\mathbf{w}}_{-1} := \mathbf{B}_{-1}(\mathbf{p}_{-1} - \mathbf{g}_{-1})$. Для схемы «самоинициализация» это эквивалентно тому, что $\bar{\mathbf{w}}_{-1} = 0$.

4.5. Итоговый алгоритм

Описанная адаптация метода IN на случай линейных моделей представлена в алгоритме 2.

Вход : 1) $\mathbf{w}_0 \in \mathbb{R}^D$: начальная точка;
2) N: общее число объектов обучения; 3) $\lambda > 0$: коэффициент регуляризации;
4) $K \in \mathbb{N}$: макс. число итераций; 5) $\varepsilon > 0$: точность оптимизации;

Выход: $\mathbf{w} \in \mathbb{R}^D$: решение задачи (1) в пределах заданной точности ε.

/* Инициализация модели (схема <<самоинициализация>>) */

$\mathbf{B} := (1/\lambda)\mathbf{I} \in \mathbb{R}^{D\times D}$; $\mathbf{p} := \mathbf{g} := \bar{\mathbf{w}} := 0 \in \mathbb{R}^D$;

$\mu_i^{\text{old}} := (\phi_i')^{\text{old}} := (\phi_i'')^{\text{old}} := 0, \quad i = 1, \ldots, N$; // Компоненты модели вместо центров

/* Основной цикл */

$\mathbf{w} := \mathbf{w}_0$; $i := -1$;

for $k = 0, 1, \ldots, K-1$ **do**

- $i := (i+1) \bmod N$; // Выбор обновляемой компоненты: циклическая схема
- {Загрузить вектор признаков $\mathbf{x}_i$} ; // Взять из памяти или считать с диска
- $\mu_i^{\text{new}} := \mathbf{x}_i^\top \mathbf{w}_k$; // Скалярное произведение (формула (24))
- {Вычислить $\phi_i'(\mu_i^{\text{new}})$ и $\phi_i''(\mu_i^{\text{new}})$};
- $\delta := (\phi_i'')^{\text{new}} - (\phi_i'')^{\text{old}}$; // Вспомог. коэффициент (формула (29))
- $\mathbf{z} := \mathbf{B}\mathbf{x}_i$; // Матрично-векторное умножение (формула (30))
- $\mathbf{B} := \mathbf{B} - \delta/(N + \delta\mathbf{x}_i^\top\mathbf{z})\mathbf{z}\mathbf{z}^\top$; // Одноранг. коррекция обратной матрицы (формула (28))
- $\mathbf{p} := \mathbf{p} + (1/N)[(\phi_i'')^{\text{new}}\mu_i^{\text{new}} - (\phi_i'')^{\text{old}}\mu_i^{\text{old}}]\mathbf{x}_i$; // Обновление шкал. центра (формула (25))
- $\mathbf{g} := \mathbf{g} + (1/N)[(\phi_i')^{\text{new}} - (\phi_i')^{\text{old}}]\mathbf{x}_i$; // Обновление градиента (формула (25))
- **if** $\|\mathbf{g} + \lambda\mathbf{w}\|_\infty < \varepsilon$ **then break**; // Критерий остановки
- $s := [(\phi_i'')^{\text{new}}\mu_i^{\text{new}} - (\phi_i')^{\text{new}}] - [(\phi_i'')^{\text{old}}\mu_i^{\text{old}} - (\phi_i')^{\text{old}}]$; // Вспомог. коэффициент (формула (31))
- $\bar{\mathbf{w}} := \bar{\mathbf{w}} + (s - \delta\mathbf{x}_i^\top\bar{\mathbf{w}})/(N + \delta\mathbf{x}_i^\top\mathbf{z})\mathbf{z}$; // Обновление минимума модели (формула (32))
- {Выбрать длину шага $\alpha > 0$};
- $\mathbf{w} := \mathbf{w} + \alpha(\bar{\mathbf{w}} - \mathbf{w})$; // Шаг метода
- $\mu_i^{\text{old}} := \mu_i^{\text{new}}$; $(\phi_i')^{\text{old}} := (\phi_i')^{\text{new}}$; $(\phi_i'')^{\text{old}} := (\phi_i'')^{\text{new}}$; // Подготовка к следующей итерации

end

return $\mathbf{w}$; // Возврат найденной точки

Алгоритм 2: Метод IN для оптимизации линейных моделей (21).

4.6. Используемая память и сложность итерации

Основные отличия схемы для линейных моделей от общей схемы, описанной в разделе 2, следующие:

1. Вместо «среднего» гессиана $\mathbf{H}_k$ хранится и обновляется в итерациях обратная матрица системы $\mathbf{B}_k$, определенная формулой (26). Это позволяет не решать на каждой итерации метода систему линейных уравнений. В результате, стоимость итерации понижается с $O(D^3)$ до $O(D^2)$.

2. Вместо центров $\mathbf{v}_i^k,\ i = 1, \ldots, N$ в памяти хранятся скалярные произведения $\mu_i^k,\ i = 1, \ldots, N$, определенные формулой (24), и значения производных $\phi_i'(\mu_i^k)$ и $\phi_i''(\mu_i^k)$. За счет этого объем используемой памяти сокращается с $O(ND + D^2)$ до $O(N + D^2)$.

Краткое резюме новой схемы представлено в табл. 2.

Метод	Сложность итерации	Используемая память	Скорость сходимости	
			По итерациям	По эпохам
SGD	$O(C+D)$	$O(D)$	Сублинейная	Сублинейная
SAG	$O(C+D)$	$O(N+D)$	Линейная	Линейная
IN	$O(C+D^2)$	$O(N+D^2)$	Суперлинейная	Квадратичная

Таблица 2: Сравнение методов SGD, SAG и IN для оптимизации линейных моделей (21); аналогично табл. 1. Обозначения: C — стоимость вычисления функции ϕ_i; N — общее число объектов обучения; D — число оптимизируемых переменных. «По эпохам» означает каждую N-ую итерацию.

5. Эксперименты

Эксперименты проводятся на задаче обучения двухклассовой логистической регрессии с ℓ_2-регуляризатором:

$$\min_{\mathbf{w}\in\mathbb{R}^D} \quad \frac{1}{N}\sum_{i=1}^{N}\ln(1+\exp(-y_i\mathbf{x}_i^\top\mathbf{w})) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2, \tag{33}$$

где N — общее число объектов обучения, D — общее число признаков, $\mathbf{x}_i \in \mathbb{R}^D$ — вектор признаков i-го объекта, $y_i \in \{-1,+1\}$ — метка класса i-го объекта, $\lambda > 0$ — коэффициент регуляризации.

В терминах используемых в (1) обозначений:

$$f_i(\mathbf{w}) := \phi_i(\mathbf{x}_i^\top\mathbf{w}), \qquad i = 1,\ldots,N,$$

где ϕ_i есть логистическая функция потерь для i-го объекта:

$$\phi_i(t) := \ln(1+\exp(-y_i t)), \qquad i = 1,\ldots,N.$$

В качестве данных для обучения используются следующие наборы:

Название	Объектов N	Признаков D	Источник
mushrooms	8 124	112	[Schlimmer, 1981]
a9a	32 561	123	[Platt et al., 1999]
w8a	49 749	300	[Platt et al., 1999]
quantum	50 000	65	[Caruana et al., 2004]
protein	145 751	74	[Caruana et al., 2004]
covtype	581 012	54	Blackard, Jock, and Dean [Frank et al., 2010]
alpha	500 000	500	[Sonnenburg et al., 2008]
SUSY	5 000 000	18	[Baldi et al., 2014]
epsilon, *zeta*	500 000	2 000	[Sonnenburg et al., 2008]
ocr	3 500 000	1 156	[Sonnenburg et al., 2008]
fd	5 469 800	900	[Sonnenburg et al., 2008]
mnist8m	8 100 000	784	[Loosli et al., 2007]
dna18m	18 000 000	800	[Sonnenburg et al., 2008]

Наборы данных *quantum* и *protein* взяты с сайта KDD Cup 2004[12]; *covtype*, *mushrooms*, *a9a*, *w8a*, *SUSY* и *mnist8m* — с сайта LIBSVM[13]; *alpha*, *epsilon*, *zeta*, *ocr*, *fd* и *dna18m* — с сайта Pascal Large Scale Learning Challenge 2006[14]. В качестве предобработки данных использовалось независимое шкалирование каждого признака в интервал $[-1,1]$ с предварительным удалением всех константных признаков; если данные итак были отшкалированы, то никакой дополнительной предобработки не выполнялось. Бинаризации набора данных *mnist8m* осуществлялась следующим образом: классы 0, 1, 2, 3, 4 были сгруппированы в один (с меткой −1), а классы 5, 6, 7, 8, 9 — в другой (с меткой +1). Набор данных *dna18m* был получен оставлением только первых 18 млн. из оригинального набора данных *dna*.

[12] `http://osmot.cs.cornell.edu/kddcup/datasets.html`.

[13] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[14] `http://largescale.ml.tu-berlin.de/about/`.

В качестве начального приближения используется ноль: $\mathbf{w}_0 := 0$. Коэффициент регуляризации λ выбирается равным $1/N$ (что соответствует довольно слабой регуляризации).

В сравнении участвуют следующие методы:

1. **IN**: предлагаемый инкрементальный метод Ньютона для линейных моделей (алгоритм 2).
2. **SGD**: метод стохастического градиентного спуска [Robbins and Monro, 1951].
3. **SAG**: метод стохастического среднего градиента [Schmidt et al., 2013].
4. **L-BFGS**: метод BFGS с ограниченной памятью [Liu and Nocedal, 1989].
5. **HFN**: безгессианный/неточный/усеченный метод Ньютона [Dembo et al., 1982].
6. **Newton**: стандартный метод Ньютона.

В ньютоновских методах (IN, HFN и Newton) на всех итерациях используется единичный шаг. В методе L-BFGS длина шага выбирается на каждой итерации автоматически, исходя из условия Армихо, с помощью стратегии бактракинга (уменьшение шага вдвое, начиная с единицы, пока не выполнится условие Армихо); если ℓ_∞-норма градиента меньше 10^{-6} (близость к оптимуму), длина шага устанавливается равной единице. В методе SAG используется константный шаг, равный $1/L$, где L — константа Липшица для градиента функции (33); константа L определяется по формуле $L = \lambda + 0.25 \max\{\|\mathbf{x}_i\|_2^2 \mid i = 1, \ldots, N\}$ [Schmidt et al., 2013]. В методе SGD длина шага выбирается убывающей и равной $0.01/t$, где t — номер текущей эпохи (начиная, с единицы). Размер истории, используемой в методе L-BFGS, равен 10.

Все методы реализованы на языке C++ и скомпилированы компилятором g++ версии 4.8.2 с использованием оптимизации третьего уровня (флаг `-O3`). Для работы с матрицами используется библиотека Eigen версии 3.2.4 [Guennebaud et al., 2014].

Эксперименты разбиты на четыре группы:

1. **Малое число переменных**: наборы данных *quantum*, *protein*, *covtype* и *SUSY*, в которых число признаков меньше 100; обучающая выборка хранится в оперативной памяти.
2. **Среднее число переменных**: наборы данных *mushrooms*, *a9a*, *w8a* и *alpha*, в которых число признаков находится в диапазоне от 100 до 1 000; обучающая выборка хранится в оперативной памяти.
3. **Большое число переменных**: наборы данных *epsilon* и *zeta*, в которых число признаков превосходит 1 000; обучающая выборка хранится в оперативной памяти.
4. **Большие данные**: наборы данных *ocr*, *fd*, *mnist8m* и *dna18m*, объем которых более 20 GB и которые не помещаются в оперативную память обычного компьютера.

Предполагается, что методы L-BFGS и HFN используются только в первых трех случаях, когда обучающая выборка полностью хранится в оперативной памяти; в противном случае, поскольку методы делают много итераций (HFN — внутри метода сопряженных градиентов), стоимость регулярного считывания полной выборки с диска будет сильно доминировать стоимость итерации самого метода, и общее время работы будет недопустимо высоким.

Запуск всех методов выполняется в операционной системе Ubuntu Linux на одном ядре процессора (без распараллеливания). Эксперименты для малого и среднего числа переменных запускаются на ноутбуке с процессором Intel Core i7-3630QM и 8 GB оперативной памяти; эксперименты для большого числа переменных — на компьютере с процессором Intel Xeon E5-2670 v2 и 15 GB оперативной памяти. Эксперименты на больших данных тоже проводятся на процессоре Intel Xeon E5-2670 v2; для наборов данных *ocr*, *fd* и *mnist8m* используется компьютер с 61 GB оперативной памяти; для *dna18m* — со 122 GB.[15]

[15]Так много оперативной памяти используется, чтобы полностью сохранить обучающую выборку в память. Это нужно только для удобства проведения экспериментов. В реальности, саму обучающую выборку можно хранить на диске и считывать на каждой итерации по одному вектору признаков.

Итоговые результаты представлены на рис. 1 (малое число переменных), рис. 2 (среднее число переменных), рис. 3 (большое число переменных) и рис. 4 (большие данные).[16][17] Из приведенных графиков можно сделать следующие наблюдения:

1. **Скорость сходимости**. Из графиков сходимости по эпохам (нормированное число итераций) хорошо видно, что кривая метода IN в пределе убывает быстрее, чем любая прямая линия. Это подтверждает доказанную теорему 1 о суперлинейной скорости сходимости метода IN.

2. **Зависимость от числа переменных**. Метод IN работает наиболее быстро при *небольшом* числе переменных (скажем, до 500 или до 1000). В этом плане он аналогичен обычному методу Ньютона. Причина медленной работы в случае большого числа переменных заключается в сложности итерации метода: согласно таблице 2, сложность итерации метода IN зависит от D *квадратично*. Сложность итерации остальных методов (кроме Newton) зависит от D лишь *линейно*.

3. **Скорость работы по эпохам**. Интересно отметить, что в терминах эпох метод IN работает *равномерно* быстрее всех остальных сравниваемых методов. Этот факт свидетельствует о том, что метод особенно эффективен в ситуации, когда объем данных очень большой, сами данные хранятся на диске, а их считывание занимает много времени (чем меньше считываний, тем лучше). Кроме того, на тестируемой задаче во всех случаях методу требуется всего 3–5 проходов по всей выборке для нахождения решения с очень большой точностью. Поскольку каждая итерация занимает одно и то же время, то появляется возможность довольно точно спрогнозировать время работы метода: достаточно замерить время работы одной итерации и умножить его на $3N$ или $5N$.

4. **Сравнение с обычным методом Ньютона**. Во всех проведенных экспериментах метод IN сходится быстрее, чем метод Newton, как в терминах эпох, так и в терминах реального времени работы. При небольшом числе функций N разница несущественная, однако она растет с увеличением N. Наиболее явно преимущество метода IN над методом Newton видно на наборе данных *dna18m*. Также нужно принимать во внимание тот факт, что одна итерация метода Newton требует *полного* прохода по всей выборке, и поэтому метод Newton нельзя остановить до завершения итерации (ступенчатый график), в отличие от метода IN, итерации которого являются более гранулированными, поскольку используют лишь *одним* объект из всей выборки.

5. **Сравнение с другими методами**. По сравнению с классическими методами HFN и L-BFGS метод IN работает сравнимо в случае малого и среднего числа переменных, и сильно медленнее в случае большого числа переменных. Тем не менее, методы HFN и L-BFGS можно эффективно применять лишь в том случае, когда выборка полностью хранится в оперативной памяти, в отличие от метода IN, который, как и большинство других инкрементальных методов, можно применять и в случае, когда выборка хранится на внешнем диске. Сравнивая методы IN и SAG, можно сделать вывод, что при малом и среднем числе переменных метод IN, в целом, работает быстрее, чем метод SAG (это особенно заметно на рис. 1, а также на наборе данных *mnist8m*).

[16] Классические методы (L-BFGS, HFN и Newton) изображены на графиках кусочно постоянными, поскольку имеют трудоемкие итерации (происходит вычисление *полной* функции), и поэтому не могут быть прерваны в середине.

[17] Для инкрементальных методов (SGD, SAG и IN) одной эпохой считается N итераций метода, т. е. полный проход по выборке. Для классических методов (L-BFGS, HFN и Newton) одной эпохой считается одна итерация метода, поскольку она требует вычисления *полной* функции.

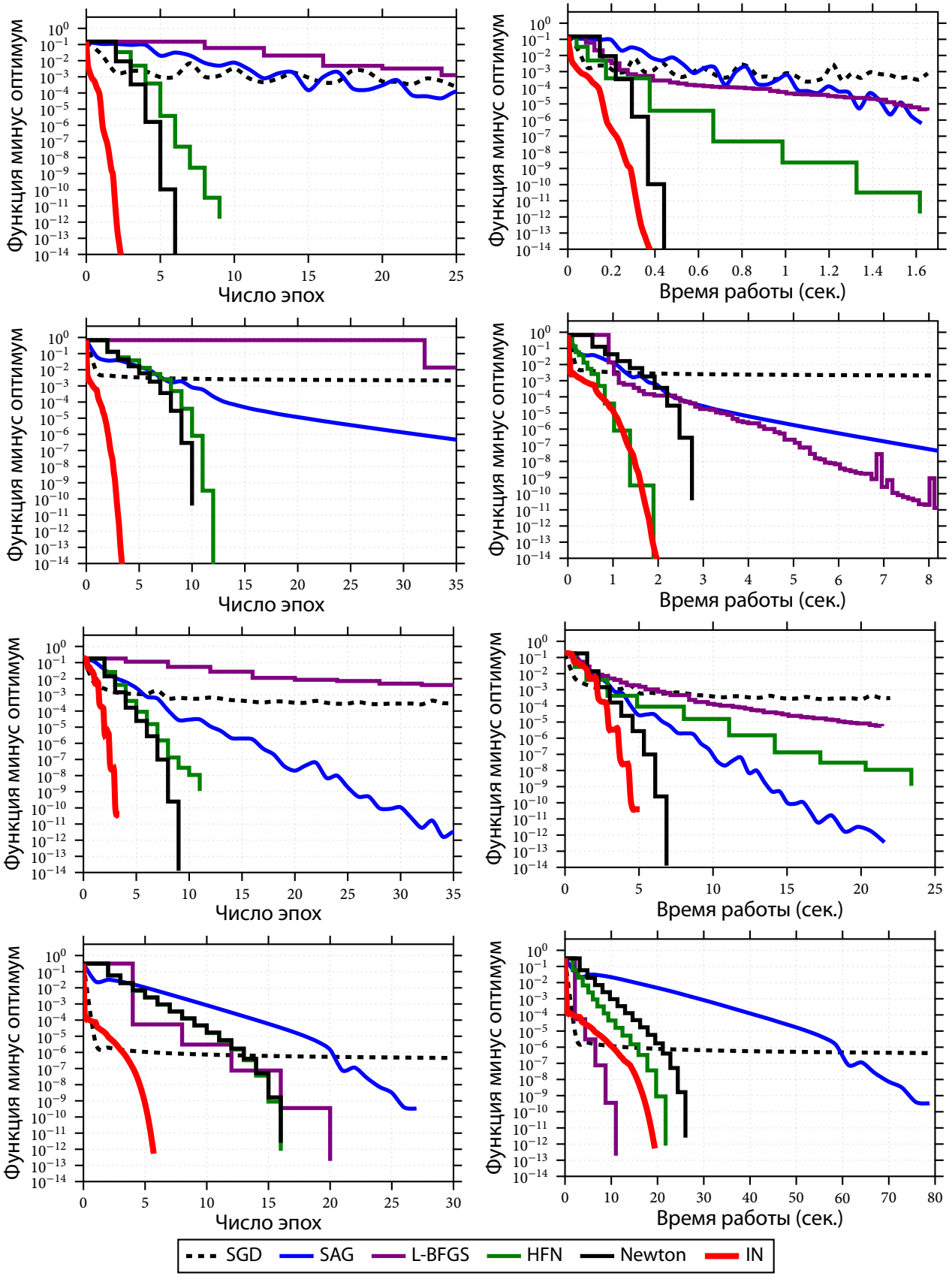

Рис. 1: Сравнение предложенного метода IN с методами SGD, SAG, L-BFGS, HFN и Newton на задаче обучения логистической регрессии с ℓ_2-регуляризатором. Случай *малого* числа признаков. По горизонтальной оси отложено время работы методов (в эпохах или реальное), по вертикальной — невязка по функции $F(\mathbf{w}_k) - F(\mathbf{w}_*)$ (где значение $F(\mathbf{w}_*)$ вычислено приближенно: в точке с нормой градиента $< 10^{-9}$). Строки соответствуют разным наборам данных (сверху вниз): 1) *quantum*, 2) *protein*, 3) *covtype*, 4) *SUSY*; столбцы соответствуют разным единицам времени работы (слева направо): 1) эпохи (доля суммарно обработанных объектов), 2) реальное время работы.

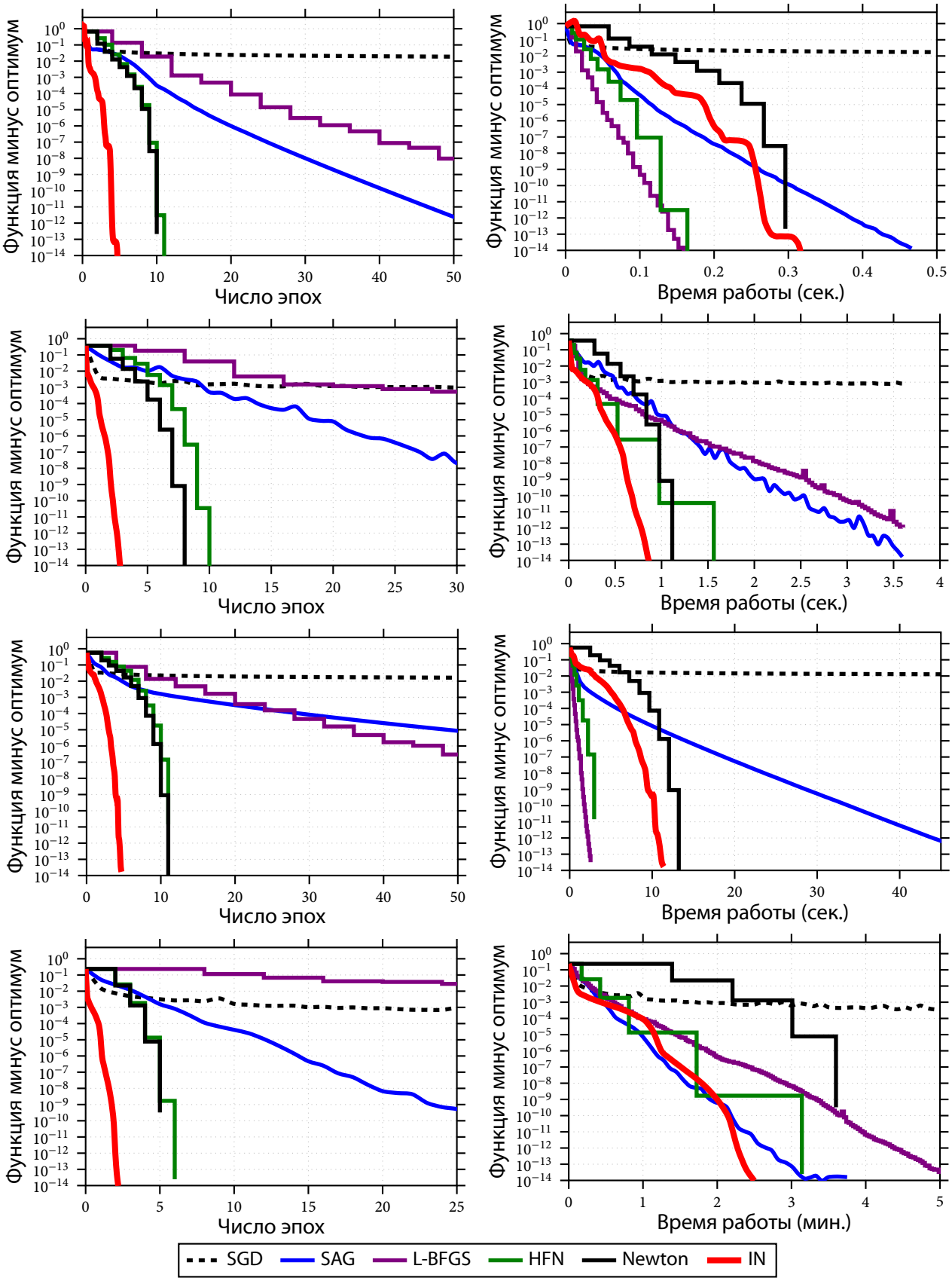

Рис. 2: Аналогично рис. 1. Случай *среднего* числа признаков. Наборы данных (сверху вниз): 1) *mushrooms*, 2) *a9a*, 3) *w8a*, 4) *alpha*.

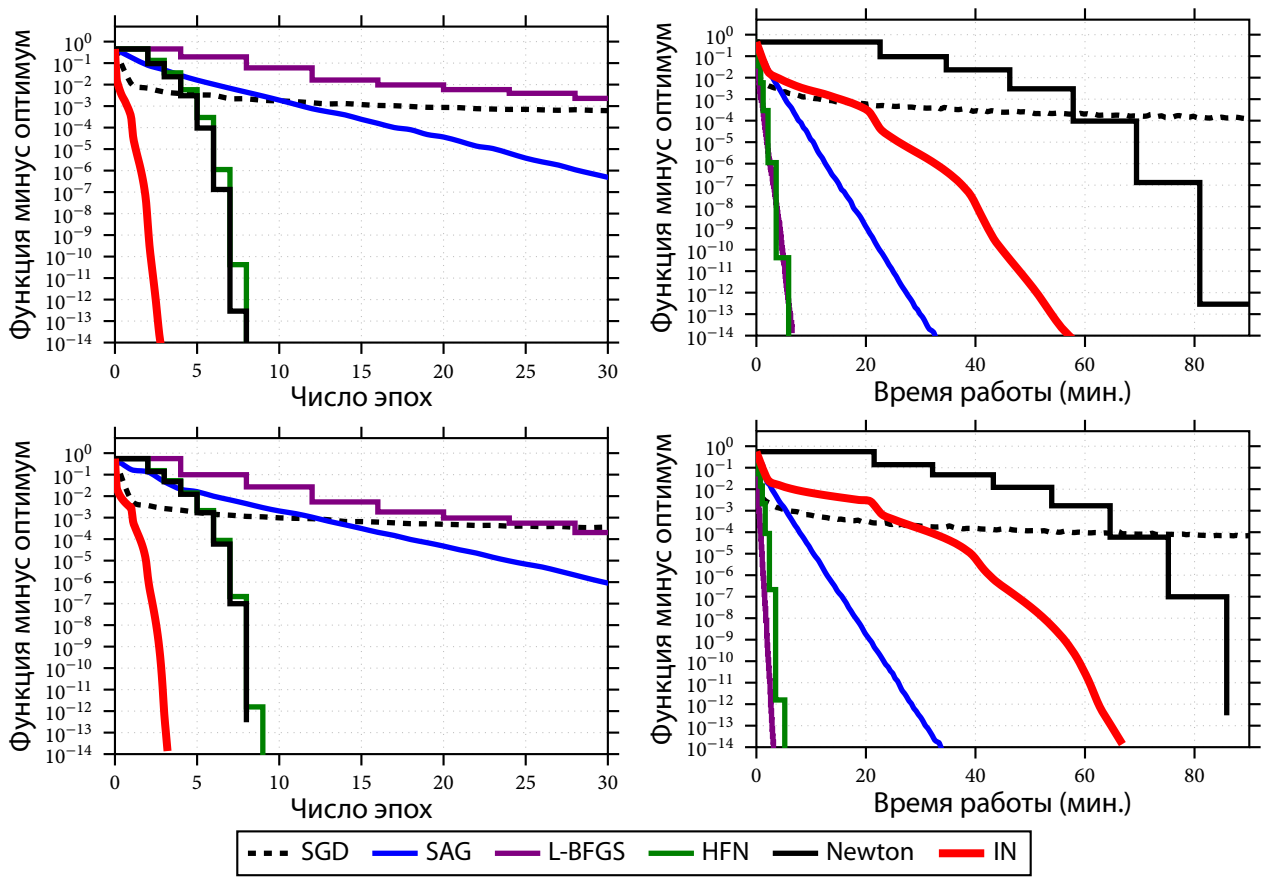

Рис. 3: Аналогично рис. 1. Случай *большого* числа признаков. Наборы данных (сверху вниз): 1) *epsilon*, 2) *zeta*.

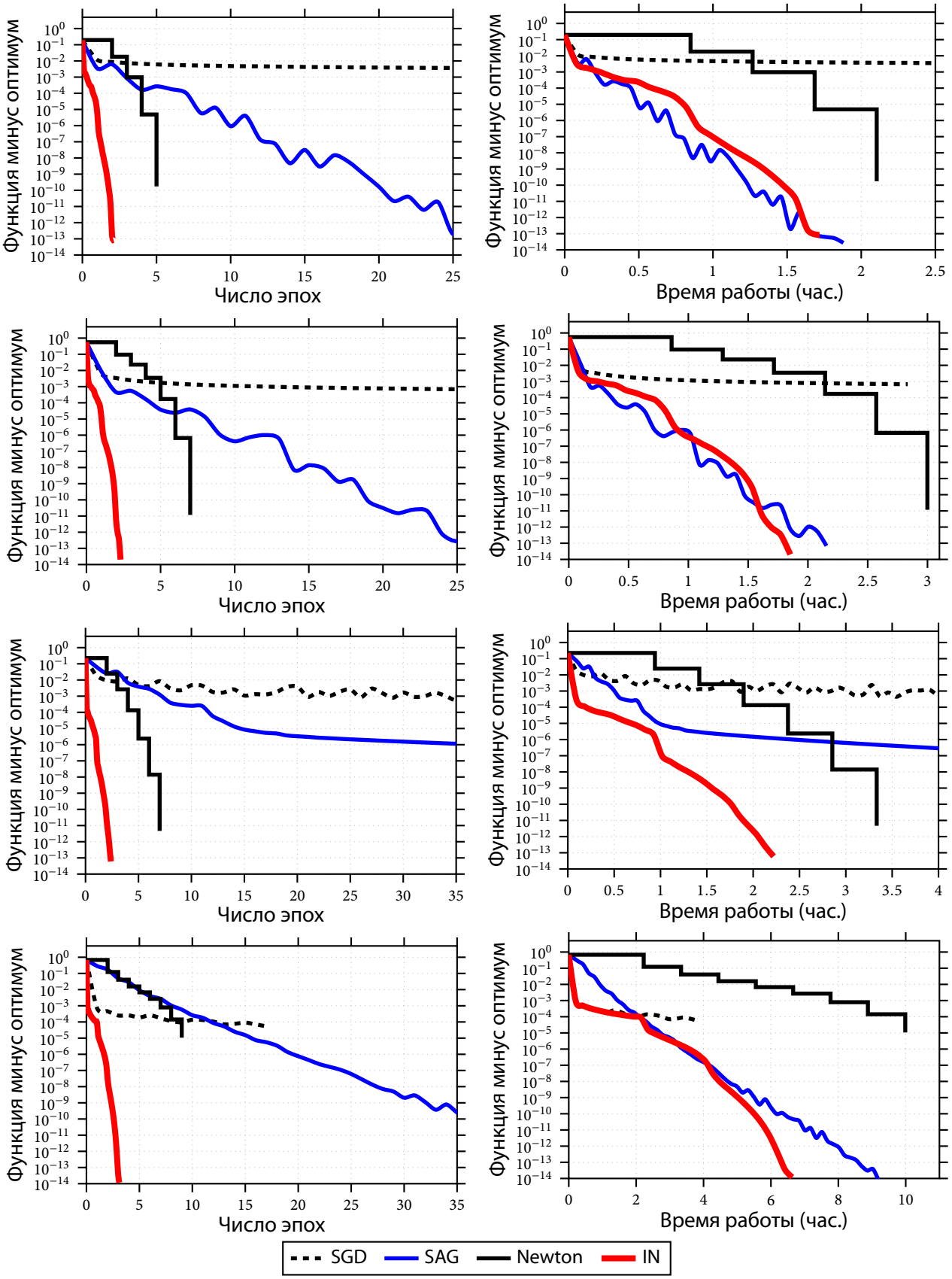

Рис. 4: Аналогично рис. 1, но для больших данных. Предполагается, что объема оперативной памяти *не хватает* для полного хранения обучающей выборки, и поэтому методы L-BFGS и HFN использовать не имеет смысла. Наборы данных (сверху вниз): 1) *ocr*, 2) *fd*, 3) *mnist8m*, 4) *dna18m*.

6. Заключение

По итогам работы:

1. Предложен инкрементальный метод Ньютона (IN) для оптимизации функций общего вида, являющихся суммой большого числа компонент.
2. Предложена эффективная модификация общей схемы метода IN на специальный случай линейных моделей в машинном обучении.
3. Доказана теорема о локальной скорости сходимости предложенного метода.

Список литературы

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

D. Blatt, A. O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

R. Caruana, T. Joachims, and L. Backstrom. KDD-Cup 2004: results and analysis. *ACM SIGKDD Explorations Newsletter*, 6(2):95–108, 2004.

R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical analysis*, 19(2): 400–408, 1982.

A. Frank, A. Asuncion, et al. UCI machine learning repository. 2010. URL `http://archive.ics.uci.edu/ml/`.

G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

G. Guennebaud, B. Jacob, et al. Eigen: A C++ linear algebra library, 2014. URL `http://eigen.tuxfamily.org/`.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

J. Nocedal and S. Wright. Numerical optimization, series in operations research and financial engineering. *Springer, New York, USA*, 2006.

J. Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods—support vector learning*, 3, 1999.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

J. Schlimmer. Mushroom records drawn from The Audubon Society field guide to north American mushrooms. *GH Lincoff (Pres), New York*, 1981.

M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge. In *25th International Conference on Machine Learning (ICML2008) Workshop*, volume 10, pages 1937–1953, 2008. URL `http://largescale.ml.tu-berlin.de`.