Методы оптимизации, ФКН ВШЭ, зима 2017

Практическое задание 2: Продвинутые методы безусловной оптимизации.

Срок сдачи: 9 марта 2017 (23:59).
Язык программирования: Python 3.

1 Алгоритмы

В этом задании все рассматриваемые алгоритмы будут решать общую задачу нелинейной безусловной оптимизации:

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

где функция $f : \mathbb{R}^n \to \mathbb{R}$ является достаточно гладкой.

1.1 Метод сопряженных градиентов

Метод сопряженных градиентов является итерационным методом (неточного) решения системы линейных уравнений

$$Ax = b,$$

где $A \in \mathbb{S}^n_{++}$ и $b, x \in \mathbb{R}^n$.

Обозначим невязку на k-м шаге через $g_k := Ax_k - b_k$.

Итерация метода:

$$x_{k+1} = x_k + \frac{g_k^T g_k}{d_k^T A d_k} d_k.$$

Направления пересчитываются по правилу:

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k,$$

где $d_0 := -g_0$.

Заметим, что методу сама матрица A не нужна, а нужна процедура умножения этой матрицы на произвольный вектор. Это является одним из основных преимуществ метода над точными процедурами типа метода Гаусса или разложения Холецкого.

Важно: В эффективной реализации метода должно быть лишь *одно* матрично-векторное произведение за итерацию.

В качестве критерия остановки обычно используют тест

$$\|Ax_k - b\|_2 \leq \varepsilon \|b\|_2.$$

1.2 Усеченный метод Ньютона

Напомним, что обычный (демпфированный) метод Ньютона итеративно строит последовательность точек $(x_k)_{k=0}^{\infty}$ по следующему правилу:

$$x_{k+1} = x_k + \alpha_k d_k,$$

где направление поиска d_k находится из следующей системы линейных уравнений:

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k). \tag{1}$$

Здесь $\nabla^2 f(x_k) \in \mathbb{S}^n_{++}$ — гессиан[1] функции f в точке x_k. Длина шага α_k выбирается с помощью линейного поиска.

В усеченном методе Ньютона[2] система (1) решается с помощью метода сопряженных градиентов. В этом случае сама матрица $\nabla^2 f(x_k)$ методу не нужна, а нужна только процедура умножения матрицы $\nabla^2 f(x_k)$ на произвольный вектор $v \in \mathbb{R}^n$. Кроме того, не имеет большого смысла решать систему (1) очень точно, если текущая точка x_k находится далеко от оптимума. Поэтому обычно метод сопряженных градиентов останавливают, как только невязка $r_k(d) := \nabla^2 f(x_k)d + \nabla f(x_k)$ удовлетворяет условию

$$\|r_k(d)\|_2 \leq \eta_k \|\nabla f(x_k)\|_2. \tag{2}$$

Последовательность $\eta_k \in (0, 1)$, называется *форсирующей последовательностью* и обычно выбирается следующим образом:

$$\eta_k := \min\{0.5, \sqrt{\|\nabla f(x_k)\|_2}\}.$$

Такой выбор гарантирует локальную сверхлинейную скорость сходимости метода в невырожденном случае.

Следует отметить, что ранний выход из метода сопряженных градиентов по условию (2), вообще говоря, не гарантирует того, что найденное направление d_k будет направлением спуска функции f в точке x_k (в отличие от точного решения $-[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$ системы (1)). Поэтому после выхода из метода сопряженных градиентов нужно дополнительно проверять, удовлетворяет ли d_k условию $d_k^T g_k > 0$, и если нет, то снова запустить метод сопряженных градиентов, но теперь уже из начальной точки d_k и с меньшим значением η_k (например, уменьшить η_k в 10 раз). Такую процедуру нужно повторять до тех пор, пока d_k не станет направлением спуска (что гарантированно рано или поздно произойдет).

Заметим, что в усеченном методе Ньютона, так же, как и в обычном методе Ньютона, важно начинать линейный поиск с $\alpha_k^{(0)} = 1$. В противном случае никакой локальной сверхлинейной сходимости может и не быть.

1.3 Метод L-BFGS

Метод BFGS принадлежит классу *квазиньютоновских* методов, которые на каждом шаге аппроксимируют настоящий гессиан $\nabla^2 f(x_k)$ с помощью некоторой матрицы B_k и выбирают направление спуска d_k как решение следующей системы (аналогичной ньютоновской):

$$B_k d_k = -\nabla f(x_k) \qquad \Leftrightarrow \qquad d_k = -H_k \nabla f(x_k), \quad \text{где } H_k := B_k^{-1}.$$

Дальше, из текущей точки x_k, как обычно, выполняется шаг в этом направлении:

$$x_{k+1} = x_k + \alpha_k d_k,$$

где $\alpha_k > 0$ — длина шага, настраиваемая с помощью линейного поиска.

Основная работа на каждой итерации квазиньютоновского метода затрачивается на построение аппроксимации гессиана и вычислении направления поиска.

Начиная с $S_0 = I$, алгоритм пересчитывает аппроксимацию гессиана по правилу $B_{k+1} = B_k + U_k$, где U_k — некоторое низкоранговое обновление. Маленький ранг U_k необходим для построения эффективной процедуры вычисления обратной матрицы $H_{k+1} = B_{k+1}^{-1} = (B_k + U_k)^{-1}$.

Конкретный вид обновления U_k следует из выполнения нескольких требований. Основное из них — *уравнение секущей*: $B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$, справедливое для всех квазиньютоновских методов.

[1] Здесь предполагается, что гессиан $\nabla^2 f(x_k)$ является положительно определенным. Одно из достаточных условий для этого — строгая выпуклость функции f. Если гессиан $\nabla^2 f(x_k)$ оказался не положительно определенным, то обычно выполняют его модификацию. В этом задании мы не будем рассматривать модификации гессиана.

[2] Также известном как *Безгессианный метод Ньютона* или *Неточный метод Ньютона*.

Одного этого уравнения недостаточно, чтобы однозначно определить B_{k+1}. Конкретную квазиньютоновскую схему получают с помощью наложения на аппроксимацию гессиана дополнительных требований.

Наиболее популярным и устойчивым на практике является правило *BFGS* (*Бройдена-Флетчера-Гольдфарба-Шанно*).

Обозначим:

$$s_k := x_{k+1} - x_k,$$
$$y_k := \nabla f(x_{k+1}) - \nabla f(x_k).$$

Тогда, схема пересчёта BFGS:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$
$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

где $\rho_k := 1/(y_k^T s_k)$.

Метод L-BFGS является модификацией метода BFGS для случаев, когда не удаётся поместить матрицу H_k в память. Для этого, в методе сохраняется история $\mathcal{H}_k := ((s_{k-i}, y_{k-i}))_{i=1}^{\ell}$ из последних ℓ векторов[3] s_k и y_k.

По имеющейся истории происходит неточное восстановление матрицы H_k из схемы BFGS и вычисление нового направления $d_k = -H_k \nabla f(x_k)$.

В эффективном виде эта процедура записывается следующим образом:

Алгоритм 1 Вычисление направления поиска d_k в методе L-BFGS

$d \leftarrow -\nabla f(x_k)$
for $i = k-1, \ldots, k-\ell$ **do**
    $\mu_i \leftarrow (s_i^T d)/(s_i^T y_i)$
    $d \leftarrow d - \mu_i y_i$
end for
$d \leftarrow ((s_{k-1}^T y_{k-1})/(y_{k-1}^T y_{k-1}))d$
for $i = k-\ell, \ldots, k-1$ **do**
    $\beta \leftarrow (y_i^T d)/(s_i^T y_i)$
    $d \leftarrow d + (\mu_i - \beta)s_i$
end for
return d

1.4 Разностная проверка произведения гессиана на вектор

Проверить правильность подсчета произведения гессиана $\nabla^2 f(x) \in \mathbb{S}^n$ на заданный вектор $v \in \mathbb{R}^n$ можно с помощью конечных разностей:

$$[\nabla^2 f(x)v]_i \approx \frac{f(x + \varepsilon_2 v + \varepsilon_2 e_i) - f(x + \varepsilon_2 v) - f(x + \varepsilon_2 e_i) + f(x)}{\varepsilon_2^2},$$

где $e_i := (0, \ldots, 0, 1, 0, \ldots, 0)$ — i-й базисный орт, а $\varepsilon_2 \sim \sqrt[3]{\varepsilon_{\text{mach}}}$, где $\varepsilon_{\text{mach}}$ — машинная точность ($\approx 10^{-16}$ для типа `double`).

[3] При $k < \ell$ история $\mathcal{H}_k$ состоит из k пар. Типичное значение $\ell = 10$.

2 Формулировка задания

1 Скачайте коды, прилагаемые к заданию:

`https://github.com/mihaha/hse-optimization-course/tree/master/task2`

Эти файлы содержат прототипы функций, которые Вам нужно будет реализовать. Некоторые процедуры уже частично или полностью реализованы. Обратите внимание, что реализацию классов линейного поиска и оракула логистической регрессии Вы можете взять из Вашей реализации первого практического задания.

2 Реализуйте метод сопряженных градиентов (функция `conjugate_gradients` в модуле `optimization`).

3 Для оракула логистической регрессии (класс `LogRegL2Oracle` в модуле `oracles`) реализуйте метод `hess_vec`, выполняющий умножение гессиана на заданный вектор.

Замечание: Ваш код должен поддерживать как плотные матрицы A типа `np.array`, так и разреженные типа `scipy.sparse.csr_matrix`.

Замечание: Если при выполнении первого практического задания Вы дополнительно реализовывали оптимизированный оракул (класс `LogRegL2OptimizedOracle`), то добавьте реализацию метода `hess_vec` в этот класс тоже.

4 Реализуйте подсчет разностной аппроксимации произведения гессиана на заданный вектор (функция `hess_vec_finite_diff` в модуле `oracles`). С помощью реализованной процедуры проверьте правильность реализации метода `hess_vec` логистического оракула. Для этого сгенерируйте небольшую модельную выборку и сравните значения, выдаваемые методом `hess_vec`, с соответствующими разностными аппроксимациями в нескольких пробных точках.

5 Реализуйте усеченный метод Ньютона (функция `hessian_free_newton` в модуле `optimization`). При реализации этого метода используйте свою реализацию метода сопряженных градиентов (функция `conjugate_gradients`).

6 Реализуйте метод L-BFGS (функция `lbfgs` в модуле `optimization`).

7 Проведите эксперименты, описанные ниже. Напишите отчет.

8 (Бонусная часть для эксперимента 2.4) Реализуйте процедуру, вычисляющую аналитический минимум многомерной квадратичной функции по заданному направлению (метод `minimize_directional` в классе `QuadraticOracle` в модуле `oracles`). Эта процедура далее будет использоваться в линейном поиске (класс `LineSearchTool` в модуле `utils`) для вычисления оптимальной длины шага на квадратичной функции.

2.1 Эксперимент: Зависимость числа итераций метода сопряженных градиентов от числа обусловленности и размерности пространства

Проведите эксперимент, аналогичный соответствующему эксперименту с градиентным спуском из первого практического задания. Сравните результат метода сопряженных градиентов с результатом градиентного спуска. Какие выводы можно сделать?

2.2 Эксперимент: Выбор размера истории в методе L-BFGS

Исследуйте, как влияет размер истории в методе L-BFGS на поведение метода.

Прежде всего, оцените размер требуемой памяти и сложность итерации метода L-BFGS в зависимости от размера истории ℓ и размерности пространства n. (Здесь не нужно учитывать сложность оракула.)

Рассмотрите несколько вариантов вариантов выбора размера истории (например, $\ell = 0$, $\ell = 1$, $\ell = 5$, $\ell = 10$, $\ell = 50$, $\ell = 100$) и постройте следующие графики:

1. Зависимость относительного квадрата нормы градиента $\|\nabla f(x_k)\|_2^2 / \|\nabla f(x_0)\|_2^2$ (в логарифмической шкале) против номера итерации.

2. Зависимость относительного квадрата нормы градиента $\|\nabla f(x_k)\|_2^2 / \|\nabla f(x_0)\|_2^2$ (в логарифмической шкале) против реального времени работы.

При этом разные варианты выбора размера истории нужно рисовать на одном и том же графике.

В качестве тестовой функции возьмите логистическую регрессию с ℓ_2-регуляризатором на данных *gisette* или *news20.binary* с сайта LIBSVM[4]. Коэффициент регуляризации и начальную точку выберите стандартным образом: $\lambda = 1/m$, $x_0 = 0$.

Какие выводы можно сделать?

2.3 Эксперимент: Сравнение методов на реальной задаче логистической регрессии

Сравните усеченный метод Ньютона, метод L-BFGS и градиентный спуск на реальной задаче логистической регрессии. В качестве реальных данных используйте следующие наборы данных с сайта LIBSVM: *w8a*, *gisette*, *real-sim*, *news20.binary*, *rcv1.binary*. Коэффициент регуляризации возьмите стандартным: $\lambda = 1/m$. Параметры всех методов возьмите равным параметрам по умолчанию. Начальную точку выберите $x_0 = 0$.

Постройте следующие графики:

1. Зависимость значения функции против номера итерации метода.

2. Зависимость значения функции против реального времени работы.

3. Зависимость относительного квадрата нормы градиента $\|\nabla f(x_k)\|_2^2 / \|\nabla f(x_0)\|_2^2$ (в логарифмической шкале) против реального времени работы.

При этом все три метода нужно рисовать на одном и том же графике.

Какие выводы можно сделать по результатам этого эксперимента? Какой из методов лучше и в каких ситуациях?

2.4 (Бонусная часть) Эксперимент: Сравнение метода сопряженных градиентов и L-BFGS на квадратичной функции

Сравните метод сопряженных градиентов и метод L-BFGS на квадратичной строго выпуклой функции:

$$f(x) := \frac{1}{2} x^T A x - b^T x, \qquad x \in \mathbb{R}^n,$$

где $A \in \mathbb{S}^n_{++}$ и $b \in \mathbb{R}^n$.

Для этого сгенерируйте случайным образом пару квадратичных функций и запустите на каждой из них оба метода (из одной и той же начальной точки). Постройте графики сходимости в терминах евклидовой нормы невязки $r_k := Ax_k - b$ (в логарифмической шкале) против номера итерации. При этом оба метода нарисуйте на одном и том же графике, но разными линиями: один — сплошным, другой — пунктиром.

Важно: Поскольку функция квадратичная, то в этом случае для произвольного метода спуска $x_{k+1} = x_k + \alpha_k d_k$ можно *аналитически* вычислить наилучшую длину шага $\alpha_k := \operatorname{argmin}_{\alpha \geq 0} f(x_k + \alpha d_k)$. Выпишите в отчет соответствующее выражение для α_k. Заметим, что в методе сопряженных

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

градиентов автоматически используется оптимальная длина шага. Поэтому, чтобы сравнение было честным, в методе L-BFGS также используйте точный линейный поиск.

Как отличаются графики этих методов? Попробуйте выбрать другие значения размера истории в методе L-BFGS (в частности, попробуйте крайние случаи $\ell = 1$ и $\ell = 0$). Меняется ли при этом что-нибудь? Как Вы можете объяснить полученные результаты?

2.5 (Бонусная часть) Эксперимент: Какая точность оптимизации нужна в реальных задачах?

В реальных задачах целевая функция, которая оптимизируется методами оптимизации, как правило, не является конечным критерием качества решения задачи. Например, рассмотрим задачу классификации и модель логистической регрессии. В этом случае оптимизируемой функцией является логистическая функция потерь. Однако само значение логистической функции потерь, с точки зрения решения задачи классификации, представляет мало интереса. А что, действительно, представляет интерес — это, например, процент ошибок при классификации на тестовой выборке. Возникает естественный вопрос: как влияет точность оптимизации целевой функции на итоговое качество решения самой задачи?

В этом эксперименте Вам предлагается исследовать этот вопрос для задачи бинарной классификации и модели логистической регрессии с ℓ_2-регуляризатором. Для этого выберите несколько реальных наборов данных и выполните следующий эксперимент.

Возьмите любой метод оптимизации (например, L-BFGS) и запустите его (на обучающей выборке) с разными параметрами требуемой точности[5] ε. Для каждого ε возьмите итоговую точку $\hat{x}$, которую вернул метод, и сравните качество прогноза логистической регрессии $\hat{b}_{\text{test}} := \text{sign}(A_{\text{test}}\hat{x}_{\text{test}})$ с истинными значениями меток b_{test}. При сравнении используйте процент ошибок — среднее число позиций, в которых векторы b_{test} и $\hat{b}_{\text{test}}$ отличаются. Нарисуйте график зависимости процента ошибок против точности оптимизации ε (в логарифмической шкале). Коэффициент регуляризации и начальную точку возьмите стандартными: $\lambda = 1/m$ и $x_0 = 0$.

В чем разница между маленькой точностью оптимизации и большой? Какие выводы можно сделать?

Рекомендация. Параметр ε имеет смысл перебрать по логарифмической сетке: от самой маленькой точности $\varepsilon = 1$ (никакой оптимизации, вернуть начальную точку x_0) до самой большой точности $\varepsilon = 10^{-6}$ или $\varepsilon = 10^{-8}$ (оптимизировать функцию до «машинной» точности).

3 Оформление задания

Результатом выполнения задания являются

1. Файлы `optimization.py` и `oracles.py` с реализованными методами и оракулами.
2. Полные исходные коды для проведения экспериментов и рисования всех графиков. Все результаты должны быть воспроизводимыми. Если вы используете случайность — зафиксируйте `seed`.
3. Отчет в формате PDF о проведенных исследованиях.

Выполненное задание следует отправить письмом на почту своей группы[6] с заголовком

Практическое задание 2, Фамилия Имя.

[5] Напомним, что во всех наших методах оптимизации $\varepsilon \in (0, 1)$ задает относительную точность по квадрату нормы градиента: $\|\nabla f(\hat{x})\|_2^2 \leq \varepsilon \|\nabla f(x_0)\|_2^2$.

[6] Для 141 группы: opt.homework+141@gmail.com. Для 142 группы: opt.homework+142@gmail.com. Для 145 группы: opt.homework+145@gmail.com.

Задание следует присылать только один раз с окончательным вариантом.

Каждый проведенный эксперимент следует оформить как отдельный раздел в PDF документе (название раздела — название соответствующего эксперимента). Для каждого эксперимента необходимо сначала написать его описание: какие функции оптимизируются, каким образом генерируются данные, какие методы и с какими параметрами используются. Далее должны быть представлены результаты соответствующего эксперимента — графики, таблицы и т.д. Наконец, после результатов эксперимента должны быть написаны Ваши выводы — какая зависимость наблюдается и почему.

Важно: Отчет не должен содержать никакого кода. Каждый график должен быть прокомментирован — что на нем изображено, какие выводы можно сделать из этого эксперимента. Обязательно должны быть подписаны оси. Если на графике нарисовано несколько кривых, то должна быть легенда. Сами линии следует рисовать достаточно толстыми, чтобы они были хорошо видимыми.

4 Проверка задания

Перед отправкой задания обязательно убедитесь, что Ваша реализация проходит автоматические *предварительные* тесты `presubmit_tests.py`, выданные вместе с заданием. Для этого запустите команду

```
python presubmit_tests.py
```

Важно: Файл с тестами может измениться. Перед отправкой обязательно убедитесь, что ваша версия `presubmit_tests.py` — последняя.

Важно: Решения, которые не будут проходить тесты `presubmit_tests.py`, будут автоматически оценены в 0 баллов. Проверяющий не будет разбираться, почему Ваш код не работает и читать Ваш отчет.

Оценка за задание (из 10 баллов) будет складываться из двух частей:

1. Правильность и эффективность реализованного кода.
2. Качество отчета.

Правильность и эффективность реализованного кода будет оцениваться автоматически с помощью независимых тестов (отличных от предварительных тестов). Качество отчета будет оцениваться проверяющим. При этом оценка может быть субъективной и аппеляции не подлежит.

За реализацию модификаций алгоритмов и хорошие дополнительные эксперименты могут быть начислены дополнительные баллы (до +4 баллов). Начисление этих баллов является субъективным и безапелляционным.

Важно: Практическое задание выполняется самостоятельно. Если вы получили ценные советы (по реализации или проведению экспериментов) от другого студента, то об этом должно быть явно написано в отчёте. В противном случае «похожие» решения считаются плагиатом и все задействованные студенты (в том числе те, у кого списали) будут сурово наказаны.