Методы оптимизации в машинном обучении, ВМК+Физтех, осень 2017

Практическое задание 4: Композитная оптимизация.

Срок сдачи: 7 декабря 2017 (четверг), 23:59 для ВМК
9 декабря 2017 (суббота), 23:59 для Физтеха

1 Алгоритмы

1.1 Субградиентный метод

Субградиентный метод — это метод решения безусловной негладкой задачи оптимизации

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

где $\phi : \mathbb{R}^n \to \mathbb{R}$ — выпуклая недифференцируемая функция.

Итерация субградиентного метода заключается в шаге из текущей точки x_k в направлении (произвольного) анти-субградиента $\phi'(x_k)$. При этом, поскольку для негладких задач норма субградиента $\|\phi'(x_k)\|$ не является информативной, субградиентный метод использует в качестве направления нормированный вектор $\phi'(x_k)/\|\phi'(x_k)\|$:

$$x_{k+1} = x_k - \alpha_k \frac{\phi'(x_k)}{\|\phi'(x_k)\|}.$$

Для сходимости метода необходимо, чтобы длины шагов α_k убывали к нулю, но не слишком быстро:

$$\alpha_k > 0, \quad \alpha_k \to 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty.$$

Обычно длины шагов выбирают по правилу $\alpha_k = \alpha/\sqrt{k+1}$, где $\alpha > 0$ — некоторая константа.

Нужно отметить, что последовательность $(x_k)_{k=0}^{\infty}$, построенная субградиентным методом, может не быть (и, как правило, зачастую не является) релаксационной последовательностью для функции ϕ, т. е. неравенство $\phi(x_{k+1}) \leq \phi(x_k)$ может не выполняться. Поэтому в субградиентном методе в качестве результата работы после N итераций метода вместо точки x_N возвращается точка $y_N := \operatorname{argmin}\{\phi(x) : x \in \{x_0, \ldots, x_N\}\}$ (т. е. из всех пробных точек x_k, построенных методом, выбирается та, в которой значение функции оказалось наименьшим).[1]

1.2 Проксимальный градиентный метод

Проксимальный градиентный метод используется для минимизации *композитных функций*:

$$\phi(x) := f(x) + h(x),$$

где функция $f : \mathbb{R}^n \to \mathbb{R}$ непрерывно дифференцируемая, а функция $h : \mathbb{R}^n \to \mathbb{R}$ выпуклая (не обязательно дифференцируемая) и достаточно простая.

Под *простотой* функции h подразумевается то, что для этой функции возможно эффективно вычислить проксимальное отображение

$$\operatorname{Prox}_{\alpha h}(x) := \operatorname*{argmin}_{y \in \mathbb{R}^n} \left\{ \alpha h(y) + \frac{1}{2}\|y - x\|^2 \right\},$$

где $x \in \mathbb{R}^n$, $\alpha > 0$. Например, для $h(x) = \|x\|_1$ проксимальное отображение может быть вычислено аналитически независимо для каждой из координат $1 \leq i \leq n$:

$$[\operatorname{Prox}_{\alpha\|\cdot\|_1}(x)]_i = \begin{cases} x_i + \alpha, & x_i < -\alpha, \\ 0, & |x_i| \leq \alpha, \\ x_i - \alpha, & x_i > \alpha. \end{cases}$$

[1] Заметим, что в практической реализации метода для вычисления результата y_N сами точки $x_0, \ldots, x_N$ хранить в памяти не нужно.

Итерация проксимального градиентного метода имеет следующий вид:

$$x_{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^n} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2\alpha_k} \|x - x_k\|^2 + h(x) \right\} = \operatorname{Prox}_{\alpha_k h}(x_k - \alpha_k \nabla f(x_k)),$$

где $\nabla f(x_k)$ — градиент функции f точке x_k, а $\alpha_k > 0$ — (должным образом выбранная) длина шага.

1.2.1 Адаптивный подбор шага по схеме Нестерова

Итерация проксимального градиентного метода имеет следующий геометрический смысл. Если градиент функции f удовлетворяет условию Липшица с константой $L_f > 0$, то для любых $x, y \in \mathbb{R}^n$ и $L \geq L_f$ справедлива оценка

$$\phi(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 + h(y),$$

которая является точной в точке $y = x$. Выбрав $x = x_k$, $L = L_k := 1/\alpha_k$, нетрудно убедиться, что итерация проксимального градиентного метода в точности соответствует минимизации этой верхней оценки, построенной в точке x_k:

$$x_{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^n} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L_k}{2} \|x - x_k\|^2 + h(x) \right\}.$$

Если $L_k \geq L_f$, то из липшицевости градиента получаем:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L_k}{2} \|x_{k+1} - x_k\|^2. \tag{1.1}$$

Оказывается, что именно это неравенство, а не условие $L_k \geq L_f$ отвечает за сходимость проксимального градиентного метода. Поэтому для подбора константы L_k (или, эквивалентно, длины шага α_k) на текущей итерации k можно использовать следующую простую процедуру линейного поиска: начать с некоторого значения L и увеличивать его вдвое, пока не выполнится неравенство (1.1). Заметим, что эта процедура определена корректно: как только значение L превысит L_f, неравенство (1.1) обязательно будет выполнено в силу липшицевости градиента функции f. О константе L_k удобно думать как о «локальной» константе Липшица градиента функции f.

Поскольку каждое пробное значение константы L требует полного вычисления функции ϕ, то число итераций линейного поиска необходимо, по возможности, сократить. Для этого имеет смысл инициализировать значение L_{k+1} с помощью уже найденного значения L_k. Естественным вариантом является инициализация $L_{k+1} = L_k$. Однако в этом случае константы L_k всегда будут только увеличиваться и никогда не уменьшаться; это плохо, поскольку при прочих равных лучше выбрать значение L как можно меньше, что будет соответствовать большему шагу и большему прогрессу в оптимизации (в некоторых областях пространства локальная константа Липшица может быть меньше, чем в области, содержащей начальную точку метода). Таким образом, чтобы константы Липшица на соседних итерациях были близки, но также со временем могли уменьшаться, в схеме Нестерова выполняется инициализация $L_{k+1} = L_k/2$.

Итоговый алгоритм проксимального градиентного метода с подбором длины шага по схеме Нестерова представлен в алгоритме 1. Здесь $L_0 > 0$ — параметр метода (нижняя оценка на глобальную константу Липшица L_f); его всегда можно выбрать равным единице ($L_0 = 1$) без принципиального ущерба для сходимости метода.

Можно показать, что, несмотря на то, что на отдельных итерациях проксимального градиентного метода может выполняться много шагов линейного поиска, среднее число вычислений функции ϕ за итерацию примерно равно двум (задание 3.2).

Алгоритм 1 Проксимальный градиентный метод с адаптивным подбором длины шага

Вход: Начальная точка $x_0 \in \mathbb{R}^n$, начальная оценка константы Липшица $L_0 > 0$.

$L \leftarrow L_0$
for $k = 0, 1, \ldots$ **do**
  while True do
    $y \leftarrow \operatorname{argmin}_{x \in \mathbb{R}^n} \{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2}\|x - x_k\|^2 + h(x)\}$
    if $f(y) \leq f(x_k) + \langle \nabla f(x_k), y - x_k \rangle + \frac{L}{2}\|y - x_k\|^2$ **then**
      break
    end if
    $L \leftarrow 2L$
  end while
  $x_{k+1} \leftarrow y$
  $L \leftarrow \max\{L_0, L/2\}$
end for

Выход: Последняя вычисленная точка x_k

1.3 Ускоренный проксимальный градиентный метод

Используя *технику ускорения Нестерова*, проксимальный градиентный метод можно ускорить:

Алгоритм 2 Ускоренный проксимальный градиентный метод с адаптивным подбором длины шага

Вход: Начальная точка $x_0 \in \mathbb{R}^n$, начальная оценка константы Липшица $L_0 > 0$.

$L \leftarrow L_0$, $A_{-1} \leftarrow 0$, $y_{-1} \leftarrow x_0$, $v_{-1} \leftarrow x_0$.
for $k = 0, 1, \ldots$ **do**
  while True do
    Найти $a > 0$ из уравнения $La^2 = a + A_{k-1}$.
    $z \leftarrow \tau v_{k-1} + (1 - \tau) y_{k-1}$, где $\tau := \frac{a}{a + A_{k-1}}$.
    $y \leftarrow \operatorname{argmin}_{x \in \mathbb{R}^n} \{f(z) + \langle \nabla f(z), x - z \rangle + \frac{L}{2}\|x - z\|^2 + h(x)\}$.
    if $f(y) \leq f(z) + \langle \nabla f(z), y - z \rangle + \frac{L}{2}\|y - z\|^2$ **then**
      break
    end if
    $L \leftarrow 2L$
  end while
  $a_k \leftarrow a$, $A_k \leftarrow A_{k-1} + a_k$, $y_k \leftarrow y$, $z_k \leftarrow z$.
  $v_k \leftarrow \operatorname{argmin}_{x \in \mathbb{R}^n} \{\frac{1}{2}\|x - x_0\|^2 + \sum_{i=0}^{k} a_i [f(z_i) + \langle \nabla f(z_i), x - z_i \rangle + h(x)]\}$.
  $L \leftarrow \max\{L_0, L/2\}$
end for

Выход: Точка (одна из всех просмотренных y или z) с наименьшим значением целевой функции ϕ

Обратите внимание, что для вычисления точки минимума v_k модели не нужно каждый раз суммировать по всем $0 \leq i \leq k$; вместо этого достаточно обновлять в итерациях взвешенную сумму градиентов $\sum_{i=0}^{k} a_i \nabla f(z_i)$.

Заметим, что, в отличие от обычного градиентного метода, ускоренный метод работает уже не с одной, а с тремя последовательностями точек. При этом любая из этих последовательностей может не быть релаксационной. Таким образом, в качестве ответа метода разумно выдавать ту точку, в которой наблюдалось наименьшее значение целевой функции.

2 Задача LASSO

Модель LASSO является одной из стандартных моделей линейной регрессии. Имеется обучающая выборка $((a_i, b_i))_{i=1}^m$, где $a_i \in \mathbb{R}^n$ — вектор признаков i-го объекта, а $b_i \in \mathbb{R}$ — его регрессионное значение. Задача заключается в прогнозировании регрессионного значения b_{new} для нового объекта, представленного своим вектором признаков a_{new}.

В модели LASSO, как и в любой модели линейной регрессии, прогнозирование выполняется с помощью линейной комбинации компонент вектора a с некоторыми фиксированными коэффициентами $x \in \mathbb{R}^n$:

$$b(a) := \langle a, x \rangle.$$

Коэффициенты x являются параметрами модели и настраиваются с помощью решения следующей оптимизационной задачи:

$$\phi(x) := \frac{1}{2}\sum_{i=1}^{m}(\langle a_i, x\rangle - b_i)^2 + \lambda\sum_{j=1}^{n}|x_j| =: \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1 \to \min_{x\in\mathbb{R}^n}. \tag{2.1}$$

Здесь $\lambda > 0$ — коэффициент регуляризации (параметр модели). Особенностью LASSO является использование именно l^1-регуляризации (а не, например, l^2-регуляризации). Такая регуляризация позволяет получить разреженное решение. В разреженном решении x^* часть компонент равна нулю. (Можно показать, что при $\lambda \geq \|A^T b\|_\infty$ все компоненты будут нулевыми). Нулевые веса соответствуют исключению соответствующих признаков из модели (признание их неинформативными).

В этом задании все рассматриваемые методы должны использовать критерий остановки по зазору двойственности (который Вы уже реализовывали в предыдущем домашнем задании).

3 Формулировка задания

1 Скачайте коды, прилагаемые к заданию:

`https://github.com/arodomanov/cmc-mipt17-opt-course/tree/master/task4`

Эти файлы содержат прототипы функций, которые Вам нужно будет реализовать. Некоторые процедуры уже частично или полностью реализованы.

2 Реализуйте негладкий оракул для функции (2.1) (классы `LeastSquaresOracle` и `LassoNonsmoothOracle` в модуле `oracles`).

3 Реализуйте субградиентный метод (функция `subgradient_method` в модуле `optimization`).

4 Реализуйте композитный оракул для функции (2.1) (классы `L1RegOracle` и `LassoProxOracle` в модуле `oracles`).

5 Реализуйте проксимальный градиентный метод (функция `proximal_gradient_descent` в модуле `optimization`).

6 Реализуйте ускоренный проксимальный градиентный метод (функция `accelerated_proximal_gradient_descent` в модуле `optimization`).

7 Проведите эксперименты, описанные ниже. Напишите отчет.

3.1 Эксперимент: Выбор длины шага в субградиентном методе

Исследуйте работу субградиентного метода в зависимости от выбора константы α_0 в формуле для длины шага. При этом для одной и той же задачи рассмотрите различные начальные точки x_0. Есть ли связь между «наилучшим» коэффициентом α_0 и начальной точкой x_0?

3.2 Эксперимент: Среднее число итераций линейного поиска в схеме Нестерова

Для проксимального градиентного метода постройте график суммарного (куммулятивного) числа итераций линейного поиска в зависимости от номера итерации. Действительно ли среднее число итераций линейного поиска примерно равно двум? Что насчет ускоренного метода?

3.3 Эксперимент: Сравнение методов

Сравните три реализованных метода на задаче LASSO. При этом рассмотрите различные значения размерности пространства n, размера выборки m и коэффициента регуляризации λ.

Для сравнения методов постройте графики 1) гарантируемая точность по зазору двойственности против числа итераций и 2) гарантированная точность по зазору двойственности против реального времени работы. Для гарантированной точности по зазору двойственности используйте логарифмическую шкалу.

Данные (матрицу A и вектор b) для задачи LASSO можно сгенерировать случайно, либо взять реальные данные с сайта LIBSVM.

4 Оформление задания

Результатом выполнения задания являются

(a) Файлы `optimization.py` и `oracles.py` с реализованными методами и оракулами.

(b) Полные исходные коды для проведения экспериментов и рисования всех графиков. Все результаты должны быть воспроизводимыми. Если вы используете случайность — зафиксируйте `seed`.

(c) Отчет в формате PDF о проведенных исследованиях.

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.

5 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

```
nosetests3 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 будет складываться из двух частей:

(a) Правильность и эффективность реализованного кода.

(b) 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 апелляции не подлежит.

За реализацию модификаций алгоритмов и хорошие дополнительные эксперименты могут быть начислены дополнительные баллы. 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.