Методы оптимизации в машинном обучении, ШАД, весна 2018

Практическое задание 3: Градиентные методы для композитной оптимизации и метод барьеров.

Срок сдачи (мягкий и жесткий): 10 мая 2018 (четверг), 23:59.
Срок сдачи (жесткий): 17 мая 2018 (четверг), 23:59.
Язык программирования: Python 3.

1 Алгоритмы

1.1 Композитная оптимизация

Задача композитной минимизации представляет собой задачу минимизации специального вида:

$$\min_{x \in Q} f(x) + h(x),$$

где $h : Q \to \mathbb{R}$ — *простая* выпуклая замкнутая функция, определенная на множестве Q в $\mathbb{R}^n$, $f : E \to \mathbb{R}$ — дифференцируемая функция с липшицевым градиентом, определенная на открытом множестве E в $\mathbb{R}^n$, содержащем Q.

Под простотой функции h подразумевается то, что для любого $\lambda > 0$ возможно эффективно вычислить проксимальное отображение

$$\mathrm{Prox}_{\lambda h}(x) := \operatorname*{argmin}_{y \in Q} \left\{ \lambda h(y) + \frac{1}{2}\|y - x\|^2 \right\}$$

для любого $x \in \mathbb{R}^n$. Например, для функции $\|\|_1 : \mathbb{R}^n \to \mathbb{R}$ соответствующее проксимальное отображение может быть вычислено по формуле

$$\mathrm{Prox}_{\lambda \|\|_1}(x) = (\mathrm{sign}(x_i)[|x_i| - \lambda]_+)_{1 \le i \le n}$$

для $x \in \mathbb{R}^n$, $\lambda > 0$, где $[t]_+ := \max\{t, 0\}$ — положительная срезка для $t \in \mathbb{R}$.

1.1.1 Градиентный метод

Одним из самых простых методов решения задачи композитной минимизации является градиентный метод. Приведем схему этого метода, в которой используется одномерный поиск для динамической регулировки оценки константы Липшица градиента:

Градиентный метод для композитной минимизации

Вход: выпуклая замкнутая функция $h : Q \to \mathbb{R}$, определенная на множестве Q в $\mathbb{R}^n$, дифференцируемая функция $f : E \to \mathbb{R}$, определенная на открытом множестве E в $\mathbb{R}^n$, содержащем Q, начальная точка $x_0 \in Q$, начальная оценка константа Липшица $L_0 > 0$ для $\nabla f|_Q$.

1 Итерация $k \ge 0$:

(a) Положить $\bar{L}_k := L_k$.

(b) Положить $x_{k+1} := \mathrm{argmin}_{x \in Q}\{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{\bar{L}_k}{2}\|x - x_k\|^2 + h(x)\}$

(c) Если $f(x_{k+1}) > f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{\bar{L}_k}{2}\|x_{k+1} - x_k\|^2$, положить $\bar{L}_k := 2\bar{L}_k$ и вернуться к шагу 1(b).

(d) Положить $L_{k+1} := \bar{L}_k / 2$.

Несмотря на то, что на отдельных шагах этой схемы может выполняться достаточно много итераций одномерного поиска по подбору параметра $\bar{L}_k$, можно показать, что *среднее* число итераций одномерного поиска за итерацию метода примерно равно *двум*. Параметр L_0, по сути дела, влияет

лишь на число одномерных поисков на самых первых итерациях метода; его всегда можно выбрать равным единице ($L_0 = 1$) без принципиального ущерба для скорости сходимости метода, поскольку в итоговую оценку суммарной трудоемкости метода этот параметр входит под логарифмом.

Заметим, что формула для x_{k+1} может быть переписана в эквивалентной форме с помощью проксимального отображения:

$$x_{k+1} = \mathrm{Prox}_{h/\bar{L}_k}\left(x_k - \frac{1}{\bar{L}_k}\nabla f(x_k)\right).$$

1.1.2 Быстрый градиентный метод

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

Быстрый градиентный метод для композитной минимизации

Вход: выпуклая замкнутая функция $h : Q \to \mathbb{R}$, определенная на множестве Q в $\mathbb{R}^n$, дифференцируемая выпуклая функция $f : E \to \mathbb{R}$, определенная на открытом множестве E в $\mathbb{R}^n$, содержащем Q, начальная точка $x_0 \in Q$, начальная оценка константа Липшица $L_0 > 0$ для $\nabla f|_Q$.

1 Положить $A_0 := 0$ и $v_0 := x_0$.

2 Итерация $k \geq 0$:

(a) Положить $\bar{L}_k := L_k$.

(b) Положить $a_k := \frac{1+\sqrt{1+4\bar{L}_k A_k}}{2\bar{L}_k}$ и $A_{k+1} := A_k + a_k$.

(c) Положить $y_k := \frac{A_k x_k + a_k v_k}{A_{k+1}}$.

(d) Вычислить $v_{k+1} := \mathrm{argmin}_{x\in Q}\{\frac{1}{2}\|x - x_0\|^2 + \sum_{i=0}^{k} a_i[f(y_i) + \langle \nabla f(y_i), x - y_i\rangle + h(x)]\}$.

(e) Положить $x_{k+1} := \frac{A_k x_k + a_k v_{k+1}}{A_{k+1}}$.

(f) Если $f(x_{k+1}) > f(y_k) + \langle \nabla f(y_k), x_{k+1} - y_k\rangle + \frac{\bar{L}_k}{2}\|x_{k+1} - y_k\|^2$, положить $\bar{L}_k := 2\bar{L}_k$ и вернуться к шагу 2(b).

(g) Положить $L_{k+1} := \bar{L}_k/2$.

Обратите внимание, что для вычисления точки v_{k+1} не нужно каждый раз суммировать по всем $0 \leq i \leq k$; вместо этого достаточно обновлять в итерациях взвешенную сумму градиентов $\sum_{i=0}^{k} a_i \nabla f(y_i)$.

В отличие от обычного градиентного метода, быстрый градиентный метод работает уже с несколькими последовательностями точек. При этом вдоль каждой из этих последовательностей целевая функция может уменьшаться не монотонно. Поэтому в качестве ответа $\bar{x}_k$ метода следует выдавать ту точку, в которой наблюдалось наименьшее (так называемое *рекордное*) значение целевой функции среди всех точек x_k, y_k, в которых выполнялись вычисления функции.

2 Метод барьеров

Метод барьеров (также называемый *методом внутренней точки* или *методом внутренних штрафов*) является методом решения условных задач оптимизации с ограничениями вида неравенств:

$$\min_{x\in\mathbb{R}^n} f(x) \qquad \text{s. t.} \quad g_i(x) \leq 0, \qquad i = 1, \ldots, m, \tag{2.1}$$

где $f, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ — дважды непрерывно дифференцируемые функции, $x \in \mathbb{R}^n$ — целевая переменная.

2.1 Общая идея

Основная идея метода барьеров заключается в сведении исходной условной задачи (2.1) к последовательности специальным образом построенных безусловных задач, решения которых сходятся к решению исходной условной задачи. Это делается с помощью барьерной функции.

Пусть Q — множество, задаваемое функциональными ограничениями $g_1, \ldots, g_m$:

$$Q := \{x \in \mathbb{R}^n : g_i(x) \le 0 \text{ для всех } 1 \le i \le m\}.$$

Барьерной функцией (или *функцией внутренних штрафов*) для множества Q называется любая функция $F : \operatorname{int}(Q) \to \mathbb{R}$, определенная на внутренности множества Q и являющаяся достаточно гладкой, которая обладает *барьерным свойством*: $F(x) \to +\infty$ при приближении x к границе множества Q изнутри.

Имея в распоряжении барьерную функцию, определим для каждого $t \ge 0$ вспомогательную функцию $f_t : \operatorname{int}(Q) \to \mathbb{R}$ по формуле

$$f_t(x) := tf(x) + F(x).$$

Оказывается, что при $t \to +\infty$ точка $x^*(t)$ сходится к множеству решений исходной условной задачи (2.1). Обозначим $x^*(t) := \operatorname{argmin}_{x \in \operatorname{int}(Q)} f_t(x)$. Множество точек $\{x^*(t) : t \ge 0\}$ называется *центральным путем*. Таким образом, для решения исходной условной задачи (2.1), можно использовать следующую *схему отслеживания пути* (она же и есть метод барьеров).

Общая схема метода барьеров

1 Выбрать начальное приближение $x_0 \in \operatorname{int}(Q)$ и начальный параметр $t_0 > 0$.

2 На каждой итерации $k \ge 0$:

(a) Вычислить точку

$$x_{k+1} := \operatorname*{argmin}_{y \in \operatorname{int}(Q)} f_{t_k}(y), \tag{2.2}$$

используя некоторый метод безусловной оптимизации с начальной точкой x_k.

(b) Увеличить параметр t: выбрать $t_{k+1} > t_k$ (чтобы в итоге $t_k \to +\infty$).

2.2 Решение вспомогательной задачи

Заметим, что задача оптимизации, возникающая на каждой итерации метода барьеров в реальности является «безусловной», несмотря на присутствие в этой задаче условия $x \in \operatorname{int}(Q)$. Действительно, за счет барьерного свойства целевая функция f_t стремится к $+\infty$ при приближении к границе множества Q, поэтому никакой «разумный» метод оптимизации достаточно близко к границе множества Q никогда не подойдет, и, с точки зрения самого метода, целевая функция как бы задана всюду. Тем не менее, здесь необходима определенная аккуратность.

Например, пусть для решения вспомогательной задачи (2.2) используется метод спуска: сначала $y_0 := x_k$, далее на каждой итерации $\ell \ge 0$ строится направление спуска $d_\ell \in \mathbb{R}^n$ для функции f_{t_k} в точке y_ℓ, и выполняется шаг

$$y_{\ell+1} := y_\ell + \alpha_\ell d_\ell,$$

где $\alpha_\ell \ge 0$ — длина шага. Поскольку метод барьеров по построению всегда работает с внутренними точками множества Q (отсюда и альтернативное название — метод внутренней точки), то для всех *достаточно маленьких* длин шагов α_ℓ новая точка $y_{\ell+1}$ будет принадлежать множеству Q, а значение функции f_{t_k} в этой точке уменьшится по сравнению с текущей точкой y_ℓ. Однако, если длина шага α_ℓ выбрана *слишком большой*, то точка $y_{\ell+1}$ может вообще оказаться за пределами множества Q — такое вполне может произойти при подборе шага в стандартном алгоритме линейного поиска. Чтобы избежать этой проблемы, стандартную процедуру линейного поиска необходимо модифицировать — запретить ей вычислять функцию в точках за пределами множества Q.

Согласно определению множества Q, чтобы $y_\ell + \alpha d_\ell \in \operatorname{int}(Q)$, длина шага $\alpha \geq 0$ должна удовлетворять следующей системе неравенств:

$$g_i(y_\ell + \alpha d_\ell) < 0, \qquad 1 \leq i \leq m.$$

Наиболее просто эта система решается в том случае, когда множество Q является полиэдром, т. е. когда функции $g_1, \ldots, g_m$ являются аффинными: $g_i(x) = \langle q_i, x\rangle - s_i$. В этом случае соответствующая система неравенств выглядит следующим образом:

$$\langle q_i, y_\ell + \alpha d_\ell\rangle < s_i, \qquad 1 \leq i \leq m.$$

Раскрывая скобки, получаем

$$\langle q_i, d_\ell\rangle\alpha < s_i - \langle q_i, y_\ell\rangle, \qquad 1 \leq i \leq m. \tag{2.3}$$

Поскольку по построению точка y_ℓ — внутренняя, то все правые части в этих неравенствах строго положительные. Значит, если для некоторого $1 \leq i \leq m$ коэффициент $\langle q_i, d_\ell\rangle \leq 0$, то соответствующее неравенство бессодержательно (поскольку $\alpha \geq 0$). Введем обозначение

$$I_\ell := \{1 \leq i \leq m : \langle q_i, d_\ell\rangle > 0\}.$$

Тогда система (2.3) эквивалентна следующей системе:

$$\langle q_i, d_\ell\rangle\alpha < s_i - \langle q_i, y_\ell\rangle, \qquad i \in I_\ell.$$

Заметим, что теперь все коэффициенты при α строго положительные, поэтому законно разделить:

$$\alpha < \frac{s_i - \langle q_i, y_\ell\rangle}{\langle q_i, d_\ell\rangle}, \qquad i \in I_\ell.$$

Наконец, последнюю систему можно переписать в виде одного скалярного неравенства:

$$\alpha < \alpha_\ell^{\max}, \qquad \text{где } \alpha_\ell^{\max} := \min_{i \in I_\ell} \frac{s_i - \langle q_i, y_\ell\rangle}{\langle q_i, d_\ell\rangle}.$$

Итак, согласно проведенным выкладкам, все допустимые длины шагов α лежат в интервале $[0, \alpha_\ell^{\max})$, т. е. в процедуре линейного поиска никогда нельзя брать шаг $\alpha \geq \alpha_\ell^{\max}$. Например, если для подбора длины шага используется процедура бэктрекинга, то в этом случае единственная модификация, которую нужно сделать по сравнению со стандартным случаем, — это изменить начальную пробную длину шага: вместо прежде используемого значения α_ℓ^0 теперь нужно использовать значение

$$\tilde{\alpha}_\ell^0 := \min\{\alpha_\ell^0, \theta\alpha_\ell^{\max}\},$$

где $0 < \theta < 1$ — некоторая константа (например, 0.99).

2.3 Полная спецификация метода

К настоящему моменту схема метода барьеров все еще довольно абстрактная, и имеется несколько вопросов, на которые нужно ответить:

1 Как именно выбирать барьерную функцию F?

2 Какой метод безусловной оптимизации нужно использовать для решения внутренней задачи, и какова должна быть точность ее решения?

3 Какая конкретно должна быть стратегия увеличения параметра t_k в схеме отслеживания пути?

Сперва ответим на первые два вопроса. Несмотря на то, что гипотетически в методе барьеров можно использовать любую комбинацию барьерной функции и метода безусловной минимизации, наиболее эффективной (как с точки зрения теории, так и практики) оказывается связка *логарифмического* барьера и *метода Ньютона*. Логарифмический барьер в общем виде имеет вид

$$F(x) := -\sum_{i=1}^{m} \ln(-g_i(x)).$$

В качестве *критерия остановки* в методе Ньютона в этом задании Вам нужно будет использовать стандартный критерий по норме градиента целевой функции (который использовался в предыдущих заданиях):

$$\|\nabla f_{t_k}(y_\ell)\|_2^2 \le \epsilon_{\text{inner}} \|\nabla f_{t_k}(x_k)\|_2^2,$$

где $\epsilon_{\text{inner}} > 0$ — некоторый параметр, который обычно выбирается довольно маленьким (например, 10^{-7}), что соответствует практически точному решению внутренней задачи (в выпуклом случае). Заметим, что поскольку в качестве алгоритма минимизации используется метод Ньютона, а стартовая точка выбирается достаточно «хорошей», то маленькое значение ϵ_{inner} совсем не представляет никакой проблемы для метода (напомним, что метод Ньютона имеет локальную квадратичную сходимость).

Что касается последнего вопроса о стратегии увеличения параметров t_k, то здесь имеется несколько эффективных стратегий. Самая простая из них, которую Вам нужно будет использовать в этом задании, состоит в постоянном *линейном увеличении*:

$$t_{k+1} := \gamma t_k,$$

где $\gamma > 1$ — постоянная константа (обычно порядка 10–100).

3 Модели

3.1 Модель 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:

$$\phi(x) := \frac{1}{2}\sum_{i=1}^{m}(\langle a_i, x\rangle - b_i)^2 + \lambda \sum_{j=1}^{n} |x_j| =: \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1 \to \min_{x \in \mathbb{R}^n}. \tag{3.1}$$

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 $\lambda \ge \|A^T b\|_\infty$ все компоненты будут нулевыми). Нулевые веса соответствуют исключению соответствующих признаков из модели (признание их неинформативными).

3.2 Двойственная задача и критерий остановки

Двойственной задачей к задаче (3.1) является

$$\max_{\mu \in \mathbb{R}^m} \left\{ -\frac{1}{2}\|\mu\|_2^2 - \langle b, \mu \rangle : \|A^T \mu\|_\infty \leq \lambda \right\}. \tag{3.2}$$

Таким образом, имея в распоряжении допустимую двойственную точку $\mu \in \mathbb{R}^m$, т. е. такую что $\|A^T \mu\|_\infty \leq \lambda$, можно вычислить следующую оценку для невязки в задаче (3.1):

$$\phi(x) - \phi^* \leq \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1 + \frac{1}{2}\|\mu\|_2^2 + \langle b, \mu \rangle =: \eta(x, \mu). \tag{3.3}$$

Величина $\eta(x, \mu)$ называется *зазором двойственности* и обращается в ноль в оптимальных решениях x^* и μ^* задач (3.1) и (3.2). Заметим, что решения x^* и μ^* связаны между собой следующим соотношением: $Ax^* - b = \mu^*$. Поэтому для фиксированного x естественным выбором соответствующего μ будет

$$\mu(x) := \min \left\{ 1, \frac{\lambda}{\|A^T(Ax - b)\|_\infty} \right\} (Ax - b).$$

Такой выбор обеспечивает стремление зазора двойственности $\eta(x, \mu(x))$ к нулю при $x \to x^*$, что позволяет использовать условие $\eta(x, \mu(x)) < \epsilon$ в качестве критерия остановки в любом итерационном методе решения задачи (3.1).

В этом задании необходимо использовать критерий остановки по зазору двойственности $\eta(x, \mu(x))$.

3.3 Сведение к гладкой условной задаче оптимизации

Выполнив эквивалентное преобразование задачи через надграфик, задачу (3.1) можно переписать в виде гладкой условной задачи:

$$\min_{x,u} \left\{ \frac{1}{2}\|Ax - b\|_2^2 + \lambda \langle 1_n, u \rangle : -u \preceq x \preceq u \right\}. \tag{3.4}$$

где $x, u \in \mathbb{R}^n$ и $1_n := (1, \ldots, 1) \in \mathbb{R}^n$. Эта задача является задачей квадратичного программирования, и ее можно решать методом барьеров.

4 Формулировка задания

1 Скачайте коды, прилагаемые к заданию:

`https://github.com/arodomanov/shad18-optml-course/tree/master/task3`

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

2 Реализуйте процедуру вычисления зазора двойственности (3.3) (функция `lasso_duality_gap` в модуле `oracles`).

3 Реализуйте композитный оракул для функции (3.1) (классы `L1RegOracle` и `LassoProxOracle` в модуле `oracles`).

4 Реализуйте градиентный метод для композитной минимизации (функция `proximal_gradient_method` в модуле `optimization`).

5 Реализуйте быстрый градиентный метод для композитной минимизации (функция `proximal_fast_gradient_method` в модуле `optimization`).

6 Выпишите для задачи (3.4) вспомогательную функцию $\phi_t(x, u)$, минимизируемую в методе барьеров. Выпишите систему линейных уравнений, задающую ньютоновоское направление $d_k = (d_k^x, d_k^u)$. Предложите свой способ решения этой системы и прокомментируйте его достоинства и недостатки. Для текущей точки (x, u) и направления d_k определите максимальную допустимую длину шага α. Какую начальную точку (x_0, u_0) можно выбрать для метода барьеров?

7 Реализуйте метод барьеров для задачи (3.4) (функция `barrier_method_lasso` в модуле `optimization`). Для подбора длины шага на внутренних итерациях используйте метод дробления шага с условием Армихо. (Не забудьте выбрать начальное значение длины шага равным единице, если оно допустимо.)

8 Проведите эксперименты, описанные ниже. Напишите отчет.

4.1 Эксперимент: Среднее число итераций одномерного поиска в градиентных методах

Для градиентного метода и быстрого градиентного метода постройте график зависимости суммарного числа итераций одномерного поиска от номера итерации метода. Действительно ли среднее число итераций линейного поиска примерно равно двум в обоих методах?

4.2 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.

5 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.

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.

6 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:

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. 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.