Глава 5

Численные методы оптимизации

Задачи оптимизации, т.е. задачи поиска минимума или максимума некоторой функции при определённых условиях, играют важную роль в обработке и анализе данных. Действительно, в главе 3 мы уже столкнулись с необходимость поиска минимума квадрата нормы невязки в методе наименьших квадратов, в главе 4 оказалось, что главные компоненты образуют базис, минимизирующий ошибку проекции данных на этот базис и одновременно максимизирующий дисперсию проекции. В главе 6 будет рассмотрен метод максимального правдоподобия, из самого названия которого понятно, что речь идёт о максимизации некоторой функции. Методы машинного обучения, например, нейронные сети (см. главу 10) или ансамбли решающих деревьев (см. раздел 9.2) основаны на задачах оптимизации.

К сожалению, далеко не всегда удаётся решить задачу оптимизации аналитически и свести её к вычислениям по готовой формуле, как это происходит в линейном методе наименьших квадратов (см. главу 3) или анализе главных компонент (см. главу 4). В таком случае необходимо использовать численные методы оптимизации. На первый взгляд может показаться не очевидным, но численные методы оптимизации могут оказаться предпочтительными даже в тех случаях, когда задача оптимизации решается аналитически. Например, для решения задачи линейных наименьших квадратов с матрицами задачи специального вида часто применяется метод сопряженных градиентов (см. раздел 5.2.2).

Основная сложность состоит в том, что универсального эффективного алгоритма численной оптимизации не существует. Как правило, библиотеки методов оптимизации предоставляют целый набор разных реализованных алгоритмов на выбор. Интересно, что типичные наборы алгоритмов оптимизации из библиотеки общего назначения и из библиотеки для работы с нейронными сетями обычно имеют мало общих методов. Такое разнообразие обусловлено тем, что задачи оптимизации бывают слишком разными.

Процедуру выбора конкретного алгоритма оптимизации можно разделить на два этапа: во-первых, выбор алгоритма, который теоретически способен решить конкретную задачу; во-вторых, выбор алгоритма, который справляется с решением этой задачи наиболее эффективным образом, где под эффективностью может пониматься скорость работы, точность результата, и т.п. Классификации задач оптимизации посвящен раздел 5.1, а алгоритмы оптимизации рассматриваются в разделе 5.2.

5.1 Задачи оптимизации

Задачей минимизации будем называть следующую задачу:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in X} f(\mathbf{x}), \tag{5.1}$$

где $f(\mathbf{x})$ называют *целевой функцией*[1], а X — называют *допустимым множеством*[2]. Сразу заметим, что *задача максимизации* сводится к задаче минимизации с помощью приписывания знака минус перед целевой функцией, поэтому принято говорить сразу о *задачах оптимизации*. Кроме того, в задачах оптимизации, возникающих при обработке и анализе данных, положение минимума $\mathbf{x}^*$ чаще всего имеет больший физический смысл, чем значение целевой функции в этой точке $f(\mathbf{x}^*)$. Действительно, мы видели в линейном методе наименьших квадратов, что положение минимума целевой функции использовалось нами как оценка неизвестных параметров модели, а значение целевой функции в точке минимума имело смысл некоторой точности. Понятно, что чаще всего нас интересуют именно оценки значений параметров той или иной модели. По этой же причине аддитивные константы в целевой функции часто игнорируются.

Напомним, что $\mathbf{x}^*$ называется точкой *глобального минимума* если

$$f(\mathbf{x}^*) \le f(\mathbf{x}), \quad \forall \mathbf{x} \in X, \tag{5.2}$$

если же условие выполняется в строгой форме, то говорят о *строгом глобальном минимуме*. Если условие выполняется лишь для некоторой окрестности точки $\mathbf{x}^*$, то говорят, что имеет место *локальный минимум*. Неограниченная снизу на множестве X функция $f(\mathbf{x})$ не имеет глобального минимума, но может иметь один или несколько локальных минимумов; ограниченная снизу функция $f(\mathbf{x})$ имеет глобальный минимум (это утверждение иногда известно под именем теоремы Вейерштрасса), но может иметь и несколько локальных минимумов.

Если допустимое множество X совпадает со всем пространством $\mathbf{R}^n$, как часто и случается, то в таком случае задача оптимизации называется *безусловной задачей оптимизации* или *неограниченной задачей оптимизации*[3]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x}), \tag{5.3}$$

[1] *cost function*
[2] *feasible set*
[3] *unconstrained optimization problem*

иначе — задача называется *условной задачей оптимизации* или *ограниченной задачей оптимизации*[4]. В этом случае допустимое множество X чаще всего задаётся с помощью ограничений в виде равенств и неравенств, тогда такая задача называется *задачей математического программирования*:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x}), \tag{5.4}$$

$$g_i(\mathbf{x}) = 0 \quad i \in \mathcal{E}, \tag{5.5}$$

$$g_i(\mathbf{x}) \geq 0 \quad i \in \mathcal{I}. \tag{5.6}$$

Функции $g_i(\mathbf{x})$ называются ограничениями. Они и задают допустимое множество X. Понятно, что множества $\mathcal{E}$ и $\mathcal{I}$ могут быть пустыми, таким образом можно сформулировать задачи с ограничениями равенствами и ограничениями неравенствами. Более-менее очевидно, что для задачи ограниченной оптимизации в виде задачи математического программирования проще предложить какие-то алгоритмы поиска минимума, чем если бы допустимое множество X было бы задано некоторым более замысловатым образом.

Если допустимое множество X является дискретным множеством, либо $f(\mathbf{x})$ является дискретной функцией, то такая задача оптимизации называется *задачей дискретной оптимизации*. Задачи дискретной оптимизации чаще всего являются NP-полными задачами (см. раздел 2.3.1). Примером таких задач является задача о рюкзаке (2.8)–(2.9). К счастью, в физике такие задачи встречаются скорее в виде исключений.

Важно различать точки локального минимума функции $f(\mathbf{x})$ и *критические точки* (или *стационарные точки*) функции $f(\mathbf{x})$, потому-что большинство алгоритмов численной оптимизации находят именно критические точки первого или второго порядка, в то время как нас интересует вообще говоря глобальный минимум. Определения в виде (5.2) вообще плохо алгоритмизируются, хотя бы потому, что проверка выполнения условия для бесконечного набора точек окрестности $\mathbf{x}^*$ занимает бесконечное количество вычислительного времени.

Итак, *критической точкой первого порядка* неограниченной задачи оптимизации (5.3) называется точка $\mathbf{x}^*$ такая, что градиент целевой функции в этой точке равен нулю:

$$\nabla f(\mathbf{x}^*) = 0. \tag{5.7}$$

Необходимое условие оптимальности первого рода можно сформулировать следующим образом: если $\mathbf{x}^*$ локальный минимум $f(\mathbf{x})$, то $\mathbf{x}^*$ критическая точка первого порядка.

Критической точкой второго порядка неограниченной задачи оптимизации (5.3) называется критическая точка первого порядка $\mathbf{x}^*$ такая, что вычисленная в данной точке матрица вторых производных целевой функции

$$(H)_{ij} \equiv \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\mathbf{x}^*}, \tag{5.8}$$

[4] *constrained optimization problem*

известная как *матрица Гессе* или *гессиан*, неотрицательно определена:

$$(\mathbf{h}, H\mathbf{h}) \geq 0, \quad \forall \mathbf{h} \in \mathbf{R}^n. \tag{5.9}$$

Необходимое условие оптимальности второго рода может быть сформулировано аналогично: если $\mathbf{x}^*$ локальный минимум $f(\mathbf{x})$, то $\mathbf{x}^*$ критическая точка второго порядка. Иными словами, все точки минимума являются критическими точками, но не все критические точки являются точками локального минимума. Например, для функции $f(x) = x^3$, неограниченной снизу, точка $x^* = 0$ очевидно критическая точка, но не положение локального минимума.

Понятно, что для задач ограниченной оптимизации требуются свои определения. *Критической точкой первого порядка* для задачи математического программирования (5.4)–(5.6) называется точка $\mathbf{x}^*$, если существует такой вектор $\mathbf{y}^*$, при котором выполняются следующие условия известные под названием *условий Каруша-Куна-Такера* или просто *условий Куна-Такера*:

$$\nabla f(\mathbf{x}^*) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} y_i^* \nabla g_i(\mathbf{x}^*) = 0, \tag{5.10}$$

$$g_i(\mathbf{x}^*) = 0 \quad i \in \mathcal{E}, \tag{5.11}$$

$$g_i(\mathbf{x}^*) \geq 0 \quad \text{и} \quad y_i^* \geq 0 \quad i \in \mathcal{I}, \tag{5.12}$$

$$g_i(\mathbf{x}^*) y_i^* = 0 \quad i \in \mathcal{I}. \tag{5.13}$$

Условия (5.10)–(5.13) представляют собой необходимое условие оптимальности первого рода для задачи (5.4)–(5.6).[5]

Условия критической точки первого рода могут иметь следующую интересную геометрическую интерпретацию. Назовём *активными ограничениями* ограничения $g_i(\mathbf{x}) = 0$ для $i \in \mathcal{E} \cup \mathcal{I}$, понятно, что органичения равенства всегда активные. Иначе, ограничения называются *пассивными ограничениями*. Тогда из условий (5.10) и (5.13) видно, что $\nabla f(\mathbf{x}^*)$ является линейной комбинацией векторов $\nabla g_i(\mathbf{x}^*)$ для всех активных ограничений. Пусть матрица $N \in \mathbf{R}^{n \times m}$ состоит из колонок, образующих базис

[5]Вообще говоря, формально условия Каруша-Куна-Такера (5.10)–(5.13) сформулированы только для так называемых *регулярных задач* ограниченной оптимизации. Приведём пример нерегулярной задачи математического программирования (Сухарев, Тимохов и Федоров 2008):

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbf{R}^2} x_1,$$
$$x_1^3 - x_2 \geq 0,$$
$$x_1^3 + x_2 \geq 0,$$
$$1 - x_1^2 - x_2^2 \geq 0.$$

Не составит большого труда убедиться в том, что $\mathbf{x}^* = 0$ — решение такой задачи, однако условия Каруша-Куна-Такера для $\mathbf{x}^*$ очевидно не выполняются. Для нас важно то, что большинство известных алгоритмов численного решения задачи оптимизации (5.4)–(5.6) сходятся в точки, где выполняются эти условия. Поэтому с нашей точки зрения проблема именно в сложности нерегулярных задач, а вовсе не в том, что условия Каруша-Куна-Такера недостаточно общо сформулированы.

линейного подпространства, ортогонального ко всем градиентам активных ограничений, тогда проекция вектора градиента целевой функции на это подпространство, называемое *нуль-пространством*:

$$N^T \nabla f(\mathbf{x}^*) = 0, \tag{5.14}$$

т.е. градиент целевой функции равен нулю в том подпространстве, где ограничения «не действуют».

Критической точкой второго порядка в слабом слысле[6] для математического программирования (5.4)–(5.6) называется критическая точка первого порядка $\mathbf{x}^*$, такая, что проекция матрицы вторых производных на нуль-пространство неотрицательно определена:

$$(\mathbf{h}, N^T H N \mathbf{h}) \geq 0, \quad \forall \mathbf{h} \in \mathbf{R}^m. \tag{5.15}$$

Кроме того, допустимое множество X может быть выпуклым множеством. Напомним, что выпуклым множеством называется такое множество, что

$$\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in X, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in X, \quad \lambda \in [0,1]. \tag{5.16}$$

Некоторые полезные примеры выпуклых множеств:

- $\mathbf{R}^n$;
- $\|\mathbf{x} - \mathbf{x}_0\| \leq \Delta$;
- $A\mathbf{x} \leq \mathbf{b}$, где под сравнением векторов понимается система поэлементных неравенств.

Целевая функция $f(\mathbf{x})$ в свою очередь тоже может быть выпуклой. Напомним, что выпуклой называется функция такая, что

$$f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2), \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in X, \lambda \in [0,1]. \tag{5.17}$$

Примерами выпуклых функций являются:

- x^2,
- $\|\mathbf{x}\|^2$,
- $\|A\mathbf{x} - \mathbf{b}\|^2$.

Задача оптимизации называется *выпуклой задачей оптимизации*, если $f(\mathbf{x})$ и X выпуклые. Напомним, что $\mathbf{R}^n$ выпуклое, значит задача безусловной оптимизации — выпуклая задача, когда $f(\mathbf{x})$ выпуклая. Можно доказать важную теорему о том, что выпуклая задача оптимизации имеет всего одну критическую точку, которая является положением глобального минимума целевой функции. Иными словами выпуклая задача оптимизации

[6] Определения критической точки второго порядка в сильном смысле мы не приводим ради краткости.

всегда имеет единственное решение, а достаточным условием оптимальности выпуклой функции в точке $\mathbf{x}^*$ является $\nabla f(\mathbf{x}^*) = 0$.

Некоторые задачи с конкретным видом целевой функции имеют свои собственные названия и специализированные алгоритмы решения.

Например, *задача линейного программирования* — это задача о поиске условного максимума линейной функции:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbf{R}^n}(\mathbf{c},\mathbf{x}), \tag{5.18}$$

$$A\mathbf{x} \leq \mathbf{b}, \tag{5.19}$$

где сравнение двух векторов $A\mathbf{x}$ и $\mathbf{b}$ понимается как поэлементное сравнение. Понятно, что если бы не ограничение (5.19), то задача поиска максимума неограниченной сверху функции $(\mathbf{c},\mathbf{x})$ не имела бы смысла.

Задача квадратичного программирования — это задача о поиске минимума квадратичной формы:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{R}^n}\left((\mathbf{c},\mathbf{x}) + \frac{1}{2}(\mathbf{x}, H\mathbf{x})\right). \tag{5.20}$$

Она имеет как самостоятельный интерес, например в случае метода наименьших квадратов (3.8), так и является вспомогательной задачей для некоторых численных методов оптимизации. Например, алгоритмы Ньютона и Левенберга–Марквандa (см. раздел. 5.2.1) предполагают, что задача (5.20) решается на каждом шаге.

Несмотря на то, что выражения для точки минимума этой функции $\mathbf{x}^*$ можно выписать аналитически:

$$\mathbf{x}^* = -H^{-1}\mathbf{c}, \tag{5.21}$$

существуют альтернативные численные методы решения этой задачи, такие, например, как метод сопряженных градиентов (см. раздел. 5.2.2), одним из преимуществ которого является отсутсвие необходимости обращать матрицу H.

Кстати, к задаче квадратичного программирования могут быть добавлены ограничения, следующие из физического смысла поставленной задачи. Например, неотрицательная задача наименьших квадратов предполагает, что искомые параметры физической модели не могут иметь отрицательные значения по своему физическому смыслу:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{R}^n}\|A\mathbf{x} - \hat{\mathbf{b}}\|^2, \tag{5.22}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{5.23}$$

где сравнение вектора $\mathbf{x}$ с нулём снова понимается поэлементно. К сожалению, оценка параметров линейной модели, полученная как решение неотрицательной задачи наименьших квадратов, не обладает такими полезными свойствами как, например, несмещённость, однако обладает другим серьёзным свойством — физичностью получаемых значений параметров.

5.1.1 Задача нелинейных наименьших квадратов

Другим интересным частным случаем является *задача нелинейных наименьших квадратов*, попытка обобщения метода наименьших квадратов (см. главу 3) на случай нелинейных моделей:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{R}^n} \frac{1}{2}\|\phi(\mathbf{x}) - \hat{\mathbf{b}}\|^2. \tag{5.24}$$

Отметим сразу, что в отличии от линейной задачи наименьших квадратов, оценки параметров нелинейных наименьших квадратов не обладают всеми теми свойствами, которыми обладает их линейный аналог. Кроме того, адекватная оценка ошибок оценок параметров может оказаться затруднена, особенно в случае сильно нелинейных задач. Тем не менее, подход нелинейных наименьших квадратов применяется на практике как для моделей имеющих физическое происхождение, так и для эмпирических моделей: например, решение задачи регрессии с помощью нейронных сетей (см. главу 10) по-сути сводится к задаче (5.24).

Для оценки ошибок оценки параметров нелинейной модели методом наименьших квадратов воспользуемся приближением малых ошибок. Введём матрицу Якоби (или якобиан):

$$(J)_{ij}(\mathbf{x}) \equiv \left.\frac{\phi_i}{x_j}\right|_{\mathbf{x}}, \tag{5.25}$$

это матрица первых производных вектор-функции $\phi(\mathbf{x})$. Заметим, что в случае линейных наименьших квадратов матрица задачи A совпадает с якобианом J. Тогда по аналогии с формулой (3.18) получим следующее выражение:

$$\operatorname{cov}\mathbf{x}^* = \frac{\|\hat{\epsilon}\|^2}{n-m}\left(J(\mathbf{x}^*)^T J(\mathbf{x}^*)\right)^{-1}, \tag{5.26}$$

где

$$\hat{\epsilon} = \hat{\mathbf{b}} - \phi(\hat{\mathbf{x}}^*). \tag{5.27}$$

Это выражение справедливо лишь в случае маленьких ошибок, поэтому на практике следует соблюдать осторожность.

По аналогии с линейной задачей наименьших квадратов, число обусловленности матрицы Якоби J отвечает за обусловленность задачи, и формально обусловленность задачи может быть разной при разных значениях параметров $\mathbf{x}$. В случае плохой обусловленности задачи, численный поиск минимума становится затруднён, потому-что произведение $J^T J$ имеет смысл аналогичный матрице Гессе H в выражении (5.9) и становится трудно отличать критические точки второго рода от седловых точек.

5.2 Алгоритмы оптимизации

Когда говорят о численных методах оптимизации, в большинстве случаев подразумеваются итеративные алгоритмы оптимизации, т.е. алгоритмы,

составленные из потенциально бесконечного числа одинаковых последовательных шагов, где на каждом шаге вычисляется улучшенный кандидат в решение задачи $\mathbf{x}^{(k+1)} = \mathbf{S}_k\left[\mathbf{x}^{(k)}\right]$ на основе известного кандидата в решение задачи $\mathbf{x}^{(k)}$. На следующем шаге операция повторяется, и вычисляется следующий улучшенный кандидат $\mathbf{S}_{k+1}\left[\mathbf{x}^{(k+1)}\right]$.

Вычисления, производимые на каждом шаге могут быть строго детерминированными, тогда алгоритм называют *детерминированным*, либо включать в себя элемент случайности, тогда алгоритм будет называться *стохастическим алгоритмом оптимизации* (см. раздел 5.4).

В качестве простого примера итеративного алгоритма оптимизации назовём *метод градиентного спуска*, в котором вычисления на каждом шаге производятся по следующему правилу:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}), \tag{5.28}$$

где $\nabla f(\mathbf{x}^{(k)})$ обозначает вычисленный вектор градиента целевой функции в точке $\mathbf{x}^{(k)}$, а $\alpha_k > 0$ некоторый параметр алгоритма, обычно убывающий с ростом k.

Алгоритм шага $\mathbf{S}_k\left[\mathbf{x}^{(k)}\right]$ выбирается таким образом, чтобы можно было доказать сходимость получаемой последовательности через конечное или бесконечное число шагов. А в случае стохастических алгоритмов оптимизации — сходимость по вероятности или сходимость по распределению. В большинстве случаев, рассматриваемая последовательность сходится (или, как говорят, алгоритм оптимизации сходится) к критической точке первого рода или критической точке второго рода той или иной задачи оптимизации.

Таким образом, можно выделить несколько признаков, по которым классифицируются алгоритмы оптимизации. Во-первых, род критической точки, в которую сходится алгоритм. Например, если целевая функция выпуклая, то достаточно использовать алгоритм, сходящийся к критической точке первого рода.

Во-вторых, вид целевой функции, с которой алгоритм умеет работать. Более специализированные алгоритмы как правило лучше справляются с частными задачами оптимизации, такими, например, как задача квадратичного программирования (5.20), или, например, задача нелинейных наименьших квадратов (5.24). Специализированные алгоритмы, которые справлялись хуже алгоритмов оптимизации общего назначения, естественным образом оказались преданы забвению.

В-третьих, количество задействованной информации на каждом шаге. Если алгоритм оптимизации требует для работы возможность вычисления целевой функции $f(\mathbf{x})$ в любой точке, то он называется *алгоритмом нулевого порядка*, если дополнительно требуется возможность вычисления градиента $\nabla f(\mathbf{x})$, то такой алгоритм называется *алгоритмом первого порядка*, если дополнительно требуется умение вычислять ещё и матрицу Гессе вторых производных H, то такой алгоритм — *второго порядка*. С одной стороны, интуитивно ожидается, что чем больше информации задействовано

в алгоритме, тем меньше потребуется итераций для достижения заданной точности. Либо, например, если известно, что целевая функция обладает седловыми точками, то потребуется использовать алгоритм, сходящийся к критической точке второго рода, который очевидно окажется алгоритмом второго порядка. С другой стороны, вовлечение матрицы вторых производных может быть практически затруднено при решении задач оптимизации очень большой размерности, например, при обучении нейронных сетей, где размерность вектора $\mathbf{x}$ может достигать десятков миллионов, а значит матрица вторых производных состояла бы из порядка 10^{14} элементов. Либо, например, вычисление градиента или матрицы производных затруднено, например, по причине громоздких формул. В последнем случае могут использоваться приёмы, аппроксимирующие первые и вторые производные с использованием конечных разностей.

В-четвертых, алгоритм может учитывать или не учитывать ограничения. Если требуется решить задачу условной оптимизации и известно, что потенциальное решение может лежать на границе допустимого множества, то необходимо применять алгоритм, который учитывает ограничения. При применении проекционных методов условной оптимизации (см. раздел 5.3.1) необходимо учитывать, что каждый проекционный метод работает только с допустимыми множествами определённой формы, например:

- положительный ортант $\mathbf{x} \geq \mathbf{0}$,
- координатный параллелепипед[7] $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$,
- полиэдр $A\mathbf{x} \geq \mathbf{b}$,
- внутренность шара $\|\mathbf{x} - \mathbf{x}_0\| \leq R$.

Сравнения векторов, как и ранее в этой главе, производятся поэлементно и обозначают систему неравенств.

Скорость работы итеративного алгоритма оптимизации может быть теоретически охарактеризована с двух точек зрения. Во-первых, можно охарактеризовать скорость работы одного шага в привычных терминах O-нотации в зависимости от размерности пространства n в котором решается задача.

Во-вторых, можно охарактеризовать количество требуемых шагов для достижения результата. Обычно различают следующие классы алгоритмов оптимизации по скорости сходимости производимых ими последовательностей. Последовательность $\mathbf{x}^{(k)}$ сходится с *линейной скоростью* в точку $\mathbf{x}^*$, если

$$\exists q \in (0,1), k_0 : \quad \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq q\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \quad \forall k \geq k_0. \tag{5.29}$$

Последовательность $\mathbf{x}^{(k)}$ сходится с *сверхлинейной скоростью* в точку $\mathbf{x}^*$, если

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq q_k\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \quad q_k \to 0 \quad k \to \infty. \tag{5.30}$$

[7] box

Последовательность $\mathbf{x}^{(k)}$ сходится с *квадратичной скоростью* в точку $\mathbf{x}^*$, если

$$\exists q > 0, k_0 : \quad \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq q\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 \quad \forall k \geq k_0. \tag{5.31}$$

Такой подход удобен для случаев, когда формально алгоритму требуется бесконечное число шагов. Понятно, что на практике алгоритм оптимизации, формально сходящийся за бесконечное число шагов, принудительно завершается при выполнении того или иного критерия, например:

- скорость убывания целевой функции $f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)})$ ниже некоторого порога,
- норма вектора изменения параметров $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\|$ ниже некоторого порога,
- норма градиента функции $\|\nabla f(\mathbf{x}^{(k+1)})\|$ ниже некоторого порога.

Сравнение в терминах скорости сходимости позволяет исключить из рассмотрения конкретный выбранный критерий остановки.

На практике, скорость работы алгоритма оптимизации это баланс между двумя крайними случаями: малым числом шагов, вычисление каждого из которых занимает значительное время, и большим числом шагов, вычисление каждого из которых не требует значительных затрат. Поэтому при необходимости решать большое число однотипных задач оптимизации, имеет смысл произвести замер скорости работы реализации алгоритма, выбранного для решения задачи. В качестве альтернативы измерению времени по секундомеру часто используется подсчёт числа точек, в которых алгоритму потребовалось вычислять значение целевой функции $f(\mathbf{x})$. Такой подход наиболее полезен в ситуациях, когда вычисление целевой функции $f(\mathbf{x})$ само по себе является операцией, требующей значительных вычислительных затрат: возможность вычислить функцию $f(\mathbf{x})$ не подразумевает, что она должна быть задана в виде формулы. Например, вычисление вектор-функции нелинейной модели $\phi(\mathbf{x})$ из нелинейной задачи наименьших квадратов (5.24) может потребовать численного решения системы дифференциальных уравнений, если идёт речь о модели движения некоторой системы, параметры $\mathbf{x}$ которой не известны, а координаты которой в разные моменты времени представлены вектором измерений $\hat{\mathbf{b}}$.

Итеративная природа алгоритмов подразумевает вопрос не только о точке куда сходится последовательность, но и о *начальном приближении* $\mathbf{x}^{(0)}$, откуда начинается процесс вычисления последовательности векторов $\mathbf{x}^{(k)}$. Кстати говоря, в ряде случаев удаётся формально доказать сходимость последовательности (т.е. сходимость алгоритма), только если начальное приближение находится в некоторой конечной окрестности предельной точки. Очевидно, что конкретное числовое значение предельной точки нам неизвестно, его то мы и ищем численно.

Когда начальное приближение $\mathbf{x}^{(0)}$ является параметром алгоритма, то на практике выбирается тем или иным подходящим способом. Если параметры $\mathbf{x}$ допускают физическую интерпретацию, то начальное приближение

может быть выбрано, например, как оценка этих же параметров выполненная для более грубой модели, допускающей оценку параметров более простым способом, не требующим знание начального приближения. Например, линейная модель позволяет воспользоваться линейным методом наименьших квадратов (см. главу 3). Когда параметры $\mathbf{x}$ не допускают физической интерпретации, например, как в случае машинного обучения, то зачастую начальное приближение $\mathbf{x}^{(0)}$ выбирается случайным образом.

5.2.1 Алгоритмы безусловной оптимизации

Метод градиентного спуска

Методом градиентного спуска называется численный алгоритм оптимизации, в котором следующее приближение вычисляется по формуле

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}), \tag{5.32}$$

где $\alpha_k > 0$ некоторый параметр, закон эволюции которого выбирается заранее. На практике часто выбирают $\alpha_k = \alpha_0$, или, например, $\alpha_k = \alpha_0/k$, где α_0 некоторая постоянная.

Идею метода градиентного спуска можно достаточно просто объяснить: вектор антиградиента $-\nabla f(\mathbf{x}^{(k)})$ показывает направление, в котором значение функции убывает. Это просто показать, разложив функцию $f(\mathbf{x})$ в ряд Тейлора в окрестности точки $\mathbf{x}^{(k)}$ следующим образом:

$$f(\mathbf{x}) = f(\mathbf{x}^{(k)}) + (\nabla f(\mathbf{x}^{(k)}), \mathbf{x} - \mathbf{x}^{(k)}) + o\left(\|\mathbf{x} - \mathbf{x}^{(k)}\|\right). \tag{5.33}$$

Для гладкой функции $f(\mathbf{x})$ можно выбрать шар $\|\mathbf{x}-\mathbf{x}^{(k)}\| \leq \rho$ настолько малым, что в этом шаре разложение (5.33) будет отличаться от функции $f(\mathbf{x})$ не более чем на наперёд заданную точность. Тогда минимум разложения (5.33) при условии $\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \rho$ можно выразить в следующем виде:

$$\mathbf{x}^* = \mathbf{x}^{(k)} - \frac{\rho}{\|\nabla f(\mathbf{x}^{(k)})\|} \nabla f(\mathbf{x}^{(k)}). \tag{5.34}$$

Значит, если предположить, что коэффициент α_k выбран достаточно малым, то можно ожидать, что

$$f(\mathbf{x}^{(k+1)}) = f\left(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})\right) < f(\mathbf{x}^{(k)}). \tag{5.35}$$

Конечно, это только наши ожидания, на практике это утверждение выполняется далеко не всегда из-за того, что нельзя сказать заранее достаточно ли мал α_k. Однако, для каждого конкретного случая справедливость (5.35) можно проверить численно, и, если неравенство не выполняется, то уменьшить шаг $\alpha'_k = \beta\alpha_k$, где $0 < \beta < 1$. Такой подход часто называют методом дробления шага.

Удаётся доказать (см., например, Сухарев, Тимохов и Федоров 2008, или другие учебники по методам оптимизации), что если шаг α_k выбирается методом дробления так, что

$$f\left(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})\right) \leq f(\mathbf{x}^{(k)}) - \epsilon\alpha_k \|\nabla f(\mathbf{x}^{(k)})\|^2, \tag{5.36}$$

где $\epsilon \in (0,1)$, а градиент функции — Липшиц-непрерывен:

$$\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\| \leq M\|\mathbf{x}_1 - \mathbf{x}_2\| \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbf{R^n}, \tag{5.37}$$

тогда последовательность векторов $\mathbf{x}^{(k)}$, построенных по схеме 5.32, независимо от выбора начального приближения $\mathbf{x}^{(0)}$ сходится к критической точке первого рода безусловной задачи оптимизации (5.3).

Алгоритм Ньютона

Алгоритмом (методом) Ньютона называется численный метод оптимизации с шагом:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - H(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)}), \tag{5.38}$$

где $H(\mathbf{x}^{(k)})$ — матрица Гессе, вычисленная в точке $\mathbf{x}^{(k)}$. В отличии от метода градиентного спуска, вектор $\nabla f(\mathbf{x}^{(k)})$ умножается не на константу, а на обратную матрицу вторых производных.

Идея метода аналогична методу градиентного спуска, но вместо разложения (5.33) используется разложение до второго порядка малости:

$$f(\mathbf{x}) = f(\mathbf{x}^{(k)}) + (\nabla f(\mathbf{x}^{(k)}), \mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)}, H(\mathbf{x} - \mathbf{x}^{(k)})) + o\left(\|\mathbf{x} - \mathbf{x}^{(k)}\|^2\right). \tag{5.39}$$

Разложение (5.39) есть квадратичная форма, минимум которой может быть найден по известной формуле аналогично главе 3:

$$\mathbf{x}^* = \mathbf{x}^{(k)} - H(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)}), \tag{5.40}$$

однако требуется, чтобы матрица вторых производных H была положительно определённой, иначе разложение (5.39) не ограничено снизу.

При ряде условий удаётся доказать сходимость метода к минимуму выпуклых функций с квадратичной скоростью (см., например, Сухарев, Тимохов и Федоров 2008). К недостаткам метода относят необходимость вычислять и обращать матрицу вторых производных на каждом шаге. Кроме того, $\mathbf{x}^{(k+1)}$, вычисленный по схеме (5.38), может оказаться настолько далеко от точки $\mathbf{x}^{(k)}$, что само разложение (5.39) в этой точке неадекватно аппроксимирует целевую функцию $f(\mathbf{x}^{(k+1)})$, а значит такой шаг может привести не к уменьшению, а к росту целевой функции за счёт неправильной аппроксимации. Одной из модификаций алгоритма является алгоритм Ньютона с регулировкой шага:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k H(\mathbf{x}^{(k)})^{-1}\nabla f(\mathbf{x}^{(k)}), \tag{5.41}$$

где α_k — параметр, имеющий смысл, аналогичный одноименному параметру в методе градиентного спуска.

Алгоритм Гаусса–Ньютона

Алгоритм Гаусса–Ньютона — это адаптация алгоритма Ньютона для решения задачи нелинейных наименьших квадратов (5.24):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (J^T J)^{-1} J^T \left(\phi(\mathbf{x}^{(k)}) - \hat{\mathbf{b}} \right), \tag{5.42}$$

где J обозначает матрицу Якоби вектор-функции $\phi(\mathbf{x}^{(k)})$, вычисленную в точке $\mathbf{x}^{(k)}$. Вычисления вторых производных функции $\phi(\mathbf{x}^{(k)})$ в методе не требуется, а форма $J^T J$ гарантирует неотрицательную определённость матрицы $J^T J$.

Метод основан на разложении в ряд Тейлора вектор-функции $\phi(\mathbf{x})$ в окрестности точки $\mathbf{x}^{(k)}$:

$$\phi(\mathbf{x}) = \phi(\mathbf{x}^{(k)}) + J(\mathbf{x} - \mathbf{x}^{(k)}) + o\left(\|\mathbf{x} - \mathbf{x}^{(k)}\| \right). \tag{5.43}$$

Подставляя разложение (5.43) в целевую функцию задачи нелинейных наименьших квадратов (5.24) получим следующую аппроксимацию целевой функции $f(\mathbf{x})$:

$$f_k(\mathbf{x}) = \frac{1}{2}\|\phi(\mathbf{x}^{(k)}) - \mathbf{b}\|^2 + (J^T(\phi(\mathbf{x}^{(k)}) - \mathbf{b}), \mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)}, J^T J(\mathbf{x} - \mathbf{x}^{(k)})). \tag{5.44}$$

Аппроксимация (5.44) не является разложением в ряд Тейлора, так как нужно либо проигнорировать третье слагаемое, которое имеет второй порядок малости по отношению к разложению (5.43), либо добавить ещё одно слагаемое такого же порядка малости, включающее вторые производные вектор-функции $\phi(\mathbf{x})$. Тем не менее, аналогично рассуждениям метода Ньютона находится минимум этой квадратичной формы, что и приводит к выражению (5.42).

Алгоритм Левенберга–Марквардта

Общей проблемой методов градиентного спуска (5.32), Ньютона (5.38) и Гаусса–Ньютона (5.42) является потенциально большие смещения, которые могут возникнуть на одном или нескольких шагах. Большим смещением $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ будем считать такое смещение, при котором

$$f(\mathbf{x}^{(k+1)}) > f(\mathbf{x}^{(k)}), \tag{5.45}$$

что является следствием неадекватности используемой апроксимации целевой функции на данном смещении относительно точки разложения $\mathbf{x}^{(k)}$. Понятно, что апроксимации (5.33), (5.39) и (5.43) идеально точно совпадают с целевой функцией $f(\mathbf{x})$ только в точке разложенгия $\mathbf{x}^{(k)}$, и, чем дальше мы отступаем от точки разложения, тем больше шансов столкнуться с ситуацией (5.45).

Американские исследователи Кеннет Левенберг (Levenberg 1944) и Дональд Марквардт (Marquardt 1963) независимо предложили способ разрешения этой проблемы, альтернативный подходу дробления шага или методу Ньютона с регулировкой шага (5.41). Предложенный ими метод решения задачи нелинейных наименьших квадратов (5.24) впоследствии был назван алгоритмом Левенберга–Марквардта.

Интересно, что авторы рассуждали по разному, но пришли в конечном итоге к одному и тому же методу. Левенберг предложил следующую модификацию схемы метода Гаусса–Ньютона (5.42):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \arg\min_{\mathbf{\Delta x}\in\mathbf{R^n}} \left(\left\| J\mathbf{\Delta x} - \left(\phi(\mathbf{x}^{(k)}) - \mathbf{b}\right)\right\|^2 + \frac{1}{w}\|\mathbf{\Delta x}\|^2 \right), \quad (5.46)$$

или в явном виде для сравнения с (5.42):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J^T J + \frac{1}{w} I \right)^{-1} J^T \left(\phi(\mathbf{x}^{(k)}) - \hat{\mathbf{b}} \right), \quad (5.47)$$

где I — единичная матрица, якобиан J вычисляется в точке $\mathbf{x}^{(k)}$, а параметр w подбирается таким образом, чтобы $f(\mathbf{x}^{(k+1)}(w)) < f(\mathbf{x}^{(k)})$. Самый простой способ подбора w состоит в последовательном вычислении кандидатов (5.47) на некоторой сетке. Заметим, что эти рассуждения перекликаются с подходом регуляризации Тихонова для линейного метода наименьших квадратов, фактически выражения (5.46) и (5.47) с точностью до обозначений совпадают с соответствующими выражениями (3.42) и (3.43).

Марквардт предложил выбирать наилучшее смещение $\mathbf{\Delta x}$ среди всех векторов с фиксированной нормой $\|\mathbf{\Delta x}\| = \Delta$, т.е. на каждом решать искать минимум квадратичной формы на сфере радиуса Δ:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{\Delta x}^*, \quad (5.48)$$

$$\mathbf{\Delta x}^* = \arg\min_{\mathbf{\Delta x}\in\mathbf{R}^n} \left\| J\mathbf{\Delta x} - \left(\phi(\mathbf{x}^{(k)}) - \mathbf{b}\right)\right\|^2, \quad (5.49)$$

$$\|\mathbf{\Delta x}\|^2 = \Delta^2, \quad (5.50)$$

что в обозначениях множетелей Лагранжа приводит к такой же схеме метода, как и (5.47):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J^T J + \lambda I \right)^{-1} J^T \left(\phi(\mathbf{x}^{(k)}) - \hat{\mathbf{b}} \right). \quad (5.51)$$

Неявный параметр радиуса сферы Δ монотонно убывает с ростом множителя Лагранжа λ, который предлагается подбирать на каждом шаге солгасно следующей процедуре. Пусть $\nu > 1$ — некоторый постоянный параметр, а $\lambda^{(k)}$ обозначает величину множителя Лагранжа λ, которая была выбрана на предыдущем шаге. Начальное значение $\lambda^{(0)} > 0$ может быть выбрано произвольно, обычно берётся значение меньше единицы. Тогда рассчитаем два пробных кандидата по формуле (5.51) для двух разных значений λ: для

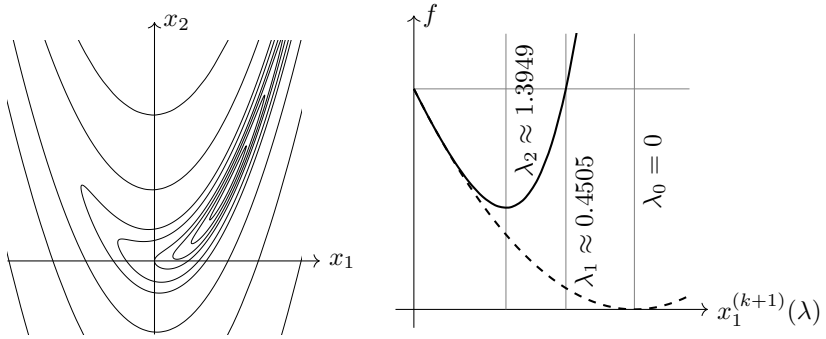

Рис. 5.1: Применение метода Левенберга-Марквардта для поиска минимума функции $f(\mathbf{x}) = (1 - x_1)^2 + 4(x_2 - x_1^2)^2$. Слева показан контурный график целевой функции. Справа показан пример поведения целевой функции и её квадратичной аппроксимации методом Левенберга-Марквардта в точке $\mathbf{x}^{(k+1)}(\lambda)$ для $\mathbf{x}^{(k)} = \mathbf{0}$ и различных значений параметра λ. Сплошая линия показывает поведение целевой функции вдоль $\mathbf{x}^{(k+1)}(\lambda)$, штриховая линия показывает поведение квадратичной апроксимации целевой функции. Случай $\lambda_0 = 0$ соответствует применению шага Гаусса–Ньютона (5.42).

шага предыдущего размера $\lambda^{(k)}$ и для увеличенного шага $\lambda^{(k)}/\nu$, а затем сравним значение целевой функции в точках $\mathbf{x}^{(k+1)}(\lambda^{(k)})$, $\mathbf{x}^{(k+1)}(\lambda^{(k)}/\nu)$ и $\mathbf{x}^{(k)}$:

- если
$$f(\mathbf{x}^{(k+1)}(\lambda^{(k)}/\nu)) \leq f(\mathbf{x}^{(k)}),$$
значит бóльший шаг улучшил приближение, поэтому выбирается более длинный шаг: $\lambda^{(k+1)} = \lambda^{(k)}/\nu$, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)}(\lambda^{(k)}/\nu)$;

- если
$$f(\mathbf{x}^{(k+1)}(\lambda^{(k)})) \leq f(\mathbf{x}^{(k)}),$$
$$f(\mathbf{x}^{(k+1)}(\lambda^{(k)}/\nu)) > f(\mathbf{x}^{(k)}),$$
значит шаг текущего размера улучшает приближение, а бóльший шаг ухудшает приближение, поэтому выбирается шаг с консервативной длиной: $\lambda^{(k+1)} = \lambda^{(k)}$, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)}(\lambda^{(k)})$;

- если
$$f(\mathbf{x}^{(k+1)}(\lambda^{(k)}/\nu)) > f(\mathbf{x}^{(k)}),$$
$$f(\mathbf{x}^{(k+1)}(\lambda^{(k)})) > f(\mathbf{x}^{(k)}),$$

значит что оба шага слишком большие, поэтому необходимо выбрать уменьшенный шаг: $\lambda^{(k+1)} = \lambda^{(k)}\nu^w$, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)}(\lambda^{(k)}\nu^w)$, где w некоторая целая степень, такая, что $f(\mathbf{x}^{(k+1)}(\lambda^{(k)}\nu^w)) \leq f(\mathbf{x}^{(k)})$.

Пример поведения $f(\mathbf{x}^{(k+1)}(\lambda))$ приводится на Рис. 5.1, где видно, что слишком маленькие значения λ соответствуют области, в которой квадратичная апроксимация описывает целевую функцию неадекватно, а слишком большие значения λ, напротив, приводят к маленьким шагам и медленному продвижению алгоритма.

Предложенный способ одновременного выбора $\mathbf{x}^{(k+1)}$ и $\lambda^{(k+1)}$ позволяет динамически изменять размер шага в зависимости от локальных особенностей целевой функции $f(\mathbf{x})$. Иными словами, поддерживается оптимальный размер шага: с одной стороны он достаточно мал, чтобы апроксимация целевой функции квадратичной формой работала корректно на каждом этапе, с другой стороны шаг достаточно велик, чтобы увеличивать скорость сходимости за счёт уменьшения числа шагов.

Методы доверительных областей

Существует целый класс методов, объединенных под общим названием *методы доверительных областей*[8], пожалуй, наиболее полное описание которых дано в монографии Conn, Gould и Toint 2000. Можно усмотреть аналогии между этих подходом и методом Левенберга–Марквардa, хотя методы доверительных областей гораздо богаче и позволяют решать не только нелинейную задачу наименьших квадратов (5.24), но и общую задачу безусловной оптимизации (5.3), задачи условной оптимизации (5.4), и т.д. Во-первых, как и раньше целевая функция $f(\mathbf{x})$ в каждой точке $\mathbf{x}^{(k)}$ заменяется апроксимацией $f_k(\mathbf{x})$, которая, например, в зависимости от решаемой задачи и выбранного метода может быть линейной (5.33), квадратичной вида (5.39) или квадратичной вида (5.43), если решается нелинейная задача наименьших квадратов, или любой другой удобной апроксимацией.

Во-вторых, считается, что выбранная апроксимация действительна (адекватна) лишь в окрестности точки разложения, которую называют *доверительной областью*, и размер Δ_k которой явно отслеживается и динамически изменяется. Например, доверительная область чаще всего задаётся в виде многомерного шара: $\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \Delta_k$, но может быть задана и в виде координатного параллелепипеда $\|\mathbf{x} - \mathbf{x}^{(k)}\|_\infty \leq \Delta_k$. Следующее приближение $\mathbf{x}^{(k+1)}$ может быть выбрано любым способом, но только внутри доверительной области.

Например, линейная и квадратичные апроксимации допускают решение следующих условных задач оптимизации:

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}\in\mathbf{R}^n} f_k(\mathbf{x}), \tag{5.52}$$

$$\|\mathbf{x} - \mathbf{x}^{(k)}\|^2 \leq \Delta_k^2. \tag{5.53}$$

[8] Trust-region method

В случае линейной апроксимации $\mathbf{x}^{(k+1)}$ находится по формуле (5.34).

В случае квадратичной апроксимации задача сводится к поиску собственных значений матрицы H апроксимирующей квадратичной формы и численному решению нелинейного уравнения относительно множителя Лагранжа λ, а затем $\mathbf{x}^{(k+1)}$ вычисляется по знакомой формуле:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (H + \lambda I)^{-1} \nabla f(\mathbf{x}^{(k)}), \tag{5.54}$$

где I — единичная матрица, а множитель Лагранжа $\lambda \geq 0$ найден ранее. Напомним, что если безусловный минимум квадратичной формы лежит внутри доверительной области, то множитель Лагранжа $\lambda = 0$.

Если решается нелинейная задача наименьших квадратов, то

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(J^T J + \lambda I\right)^{-1} J^T \left(\phi(\mathbf{x}^{(k)}) - \hat{\mathbf{b}}\right), \tag{5.55}$$

но в отличии от метода Левенберга–Марквардта (5.51), $\lambda \geq 0$ вычисляется заранее исходя из условия $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|^2 \leq \Delta_k^2$. Такие вычисления сопряжены с необходимостью отыскания собственных значений матрицы H. Кроме того, благодаря ограничению (5.53) матрица H апроксимирующей квадратичной формы может быть отрицательно определена, чего нельзя было допустить в методе Ньютона. Квадратичная форма с отрицательно определённой матрицей H имеет условный минимум на поверхности сферы $\|\mathbf{x} - \mathbf{x}^{(k)}\| = \Delta_k$, и этот минимум может быть найден с помощью собственных векторов матрицы H.

Более того, вспоминая, что мы имеем дело с апроксимацией $f_k(\mathbf{x})$ вместо истинной целевой функции $f(\mathbf{x})$, можно предложить способ вычисления $\mathbf{x}^{(k+1)}$ на основе приближенного решения задачи (5.52)-(5.53). Один из эффективных способов приближенного поиска $\mathbf{x}^{(k+1)}$ основан на методе сопряженных градиентов (см. раздел 5.2.2).

В-третьих, размер доверительной области Δ_k на каждом шаге контролируется на основе количественной меры:

$$\rho_k = \frac{f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})}{f_k(\mathbf{x}^{(k+1)}) - f_k(\mathbf{x}^{(k)})}, \tag{5.56}$$

где в числителе стоит разность значений целевой функции, а в знаменателе — разность значений апроксимации. В идеальном случае значение ρ_k должно быть близко к 1, однако в общем случае ρ_k может быть как положительной так и отрицательной величиной. Количественная мера ρ_k используется для адаптации размера доверительной области Δ_k, например следующим образом:

- если $\rho_k \in [1 - \alpha_1, 1 + \alpha_1]$, где α_1 некоторый постоянный порог, значит апроксимация работает хорошо, и чтобы ускорить сходимость, для следующей итерации Δ_{k+1} выбирается больше, чем Δ_k;
- если $\rho_k \in [1 - \alpha_2, 1 + \alpha_2]$, где $\alpha_2 > \alpha_1$ некоторый постоянный порог, значит апроксимация работает удовлетворительно, и размер доверительной области сохраняется $\Delta_{k+1} = \Delta_k$;

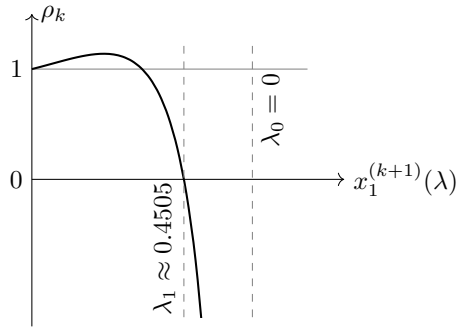

Рис. 5.2: Поведение функции ρ_k для случая Рис. 5.1: рассматривается квадратичная аппроксимация функции $f(\mathbf{x}) = (1 - x_1)^2 + 4(x_2 - x_1^2)^2$ в точке $\mathbf{x}^{(k)} = \mathbf{0}$.

- если $|\rho_k - 1| > \alpha_2$, значит был выбран слишком большой размер доверительной области, и для следующего шага Δ_{k+1} выбирается меньше, чем Δ_k, причём, если $\rho_k < 0$, то дополнительно $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$.

На Рис. 5.2 приводится пример помедения функции ρ_k.

5.2.2 Метод сопряженных градиентов

Вернёмся к задаче квадратичного программирования (5.20) и зададимся следующим вопросом. Пусть у нас есть линейное подпространство, в котором столбцы матрицы $S_k \in \mathbf{R^{n \times k}}$ задают базис, такой, что $S_k^T H S_k$ невырожденная матрица. Рассмотрим афинное преобразование $\mathbf{x} = \mathbf{x}^{(0)} + S_k \mathbf{s}$, где $\mathbf{s} \in \mathbf{R^k}$, и найдём точку минимума квадратичной формы среди всех $\mathbf{x}$ удовлетворяющих этому условию. Можно доказать[9], что искомая точка условного минимума выражается формулой:

$$\mathbf{x}^* = \mathbf{x}^{(0)} - S_k \left(S_k^T H S_k\right)^{-1} S_k^T \nabla f(\mathbf{x}^{(0)}), \tag{5.57}$$

и удовлетворяет следующему условию

$$S_k^T \nabla f(\mathbf{x}^*) = 0, \tag{5.58}$$

которое можно прочитать, как условие ортогональности вектора градиента $\nabla f(\mathbf{x}^*)$ выбранному линейному подпространству. Условие (5.58) полностью соответствует геометрической интерпретации условий Каруша-Куна-Такера (5.14).

Теперь рассмотрим вложенную последовательность линейных подпространств возрастающей размерности, т.е. базис каждого нового подпространства S_{k+1} получается с помощью добавления нового базисного вектора $\mathbf{s}^{(k)}$ к существующему базису S_k. Тогда соотношение (5.57) с учётом

[9] Достаточно рассмотреть $\nabla_s f(\mathbf{x}^{(0)} + S\mathbf{s}) = 0$, а затем заменить все вхождения $\mathbf{c} + H\mathbf{x}^{(0)}$ назад на $\nabla f(\mathbf{x}^{(0)})$.

свойства (5.58) перепишется в виде:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - S_{k+1}\left(S_{k+1}^T H S_{k+1}\right)^{-1} S_{k+1}^T \nabla f(\mathbf{x}^{(k)}) \\ = \mathbf{x}^{(k)} - (\mathbf{s}^{(k)}, \nabla f(\mathbf{x}^{(k)}))S_{k+1}\left(S_{k+1}^T H S_{k+1}\right)^{-1}\mathbf{e}_k, \quad (5.59)$$

где $\mathbf{x}^{(k+1)}$ — точка минимума для пространства размерности $k+1$, $\mathbf{x}^{(k)}$ — точка минимума для пространства размерности k, $\mathbf{s}^{(k)}$ новый дополнительный вектор базиса, а $\mathbf{e}_k$ обозначает орт с ненулевым последним компонентом.

Пусть S_k состоит из H-*сопряжённых*[10] колонок. H-сопряженными, или просто сопряженными, будем называть такие вектора $\mathbf{p}^{(i)}$, для которых $(\mathbf{p}^{(i)}, H\mathbf{p}^{(j)}) = 0$ для $\forall i, j : i \neq j$. Тогда соотношение (5.59) перепишется в следующем виде:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{(\mathbf{p}^{(k)}, \nabla f(\mathbf{x}^{(k)}))}{(\mathbf{p}^{(k)}, H\mathbf{p}^{(k)})}\mathbf{p}^{(k)}, \quad 0 \leq k < n, \quad (5.60)$$

а выражение для градиента

$$\nabla f(\mathbf{x}^{(k+1)}) = \nabla f(\mathbf{x}^{(k)}) - \frac{(\mathbf{p}^{(k)}, \nabla f(\mathbf{x}^{(k)}))}{(\mathbf{p}^{(k)}, H\mathbf{p}^{(k)})}H\mathbf{p}^{(k)}, \quad 0 \leq k < n. \quad (5.61)$$

Такая рекуррентная схема обладает целым рядом полезных свойств. Во-первых, схема за n шагов сходится к точке безусловного минимума квадратичной формы. Заметим, что алгоритм устойчив в том смысле, что при продолжении рассчёта для $k \geq n$ получаемая последовательность $\mathbf{x}^{(k)}$ сходится к точке безусловного минимума квадратичной формы. Рассчёт для $k \geq n$ используют для устранения ошибок округления, накопленных по мере рассчёта.

Во-вторых, схема не требует нахождения обратной матрицы H^{-1}, алгоритм предполагает умение умножать матрицу H на любой вектор, что позволяет оптимизировать хранение матрицы H в памяти, если структура матрицы имеет какую-то особую форму.

В-третьих, на каждом шаге алгоритма значение целевой функции монотонно убывает, значит алгоритм можно оборвать на любом шаге, когда задача квадратичного программирования используется в составе итеративного алгоритма численной оптимизации.

В-четвертых, на каждом шаге итеративного алгоритма требуется дополнительно хранение только нескольких векторов размера n, т.е. требование по дополнительной памяти $O(n)$.

Однако, пока остаётся непонятным как эффективно построить H-сопряжённый вложенный базис $\mathbf{p}^{(k)}$. К счастью, H-сопряжённый базис можно легко построить прямо по ходу рассчёта, например, по следующей рекуррентной схеме:

$$\mathbf{p}^{(k+1)} = -\nabla f(\mathbf{x}^{(k+1)}) + \frac{(\nabla f(\mathbf{x}^{(k+1)}), H\mathbf{p}^{(k)})}{(\mathbf{p}^{(k)}, H\mathbf{p}^{(k)})}\mathbf{p}^{(k)}, \quad 0 \leq k < n-1, \quad (5.62)$$

[10] *conjugated*

а $\mathbf{p}^{(0)} = -\nabla f(\mathbf{x}^{(0)})$, где $\mathbf{x}^{(0)}$ некоторая начальная точка, например, $\mathbf{x}^{(0)} = 0$, тогда $\mathbf{p}^{(0)} = -\mathbf{c}$.

Формула (5.62) нуждается в пояснении. Предположим, что существует $n' \leq n$ такое, что

$$(\mathbf{p}^{(i)}, H\mathbf{p}^{(j)}) = 0 \quad \forall i, j : 0 \leq i, j < n', i \neq j, \tag{5.63}$$

т.е. формулы (5.60), (5.61) и (5.62) справедливы для $n = n'$, тогда выполняется следующее соотношение:

$$(\nabla f(\mathbf{x}^{(k+1)}), \nabla f(\mathbf{x}^{(j)})) = 0 \quad \forall j, k : 0 \leq j \leq k < n', \tag{5.64}$$

иначе говоря, градиенты квадратичной формы, взятые в первых n' точках взаимоортогональны.

- Для $j = 0$ это утверждение следует из $\nabla f(\mathbf{x}^{(0)}) = -\mathbf{p}^{(0)}$ и формулы (5.58).
- Для $j > 0$ следует скалярно домножить формулу (5.62), записанную для $k+1 = j$, на $\nabla f(\mathbf{x}^{(k+1)})$: тогда слагаемое слева и второе слагаемое справа окажутся равными нулю в силу формулы (5.58), следовательно, равно нулю и оставшееся слагаемое, равное $(\nabla f(\mathbf{x}^{(k+1)}), \nabla f(\mathbf{x}^{(j)}))$, где $1 \leq j < n'$.

Теперь вспомним, что вектор $-\nabla f(\mathbf{x}^{(n')})$ и вектора $\mathbf{p}^{(k)}$ линейно независимы, значит можно построить новый H-сопряженный базисный вектор $\mathbf{p}^{(n')}$ с помощью процедуры ортоганализации Грама-Шмидта:

$$\mathbf{p}^{(n')} = -\nabla f(\mathbf{x}^{(n')}) - \sum_{k=0}^{n'-1} \frac{(\nabla f(\mathbf{x}^{(n')}), H\mathbf{p}^{(k)})}{(\mathbf{p}^{(k)}, H\mathbf{p}^{(k)})} \mathbf{p}^{(k)}. \tag{5.65}$$

Заметим, однако, что $H\mathbf{p}^{(k)}$ можно выразить из формулы (5.61), затем воспользоваться свойством (5.64) и показать, что в сумме правой части (5.65) ненулевым является только слагаемое с $k = n' - 1$. Значит, $\mathbf{p}^{(n')}$ вычсиленный по формуле (5.62) — H-сопряженный ко всем остальным базисным векторам $\mathbf{p}^{(k)}$.

В случае $n' = 2$, скалярно домножая (5.62) при $k = 0$ на $H\mathbf{p}^{(0)}$, можно доказать, что $(\mathbf{p}^{(1)}, H\mathbf{p}^{(0)}) = 0$, откуда по индукции следует, что рассчитаный по формулам (5.60), (5.61) и (5.62) базис $\mathbf{p}^{(k)}$ — H-сопряжённый для всего рассматриваемого линейного пространства размерности n.

Из формул (5.61) и (5.62) следует, что некоторый базис (не H-сопряженный и не ортогональный) исследуемых вложенных линейных подпространств может быть представлен набором векторов:

$$\mathbf{p}^{(0)}, H\mathbf{p}^{(0)}, \ldots, H^{k-1}\mathbf{p}^{(0)}, \tag{5.66}$$

т.е. фактически базис задаётся начальным вектором $\mathbf{p}^{(0)}$ и квадратной матрицей H, которая последовательно возводится в разную степень. Постро-

енные по такому принципу линейные подпространства называются *подпространствами Крылова*[11], в честь академика А.Н. Крылова, русского математика и корабельного инженера.

Использование формул (5.61) и (5.62) — не единственный способ построить удобный базис в подпространствах Крылова. Известно несколько методов, объединенных названием методов подпространств Крылова. Например, Корнелий Ланцош, математик венгерского происхождения, нашёл изящный итеративный метод пригодный для решения задачи (5.52)–(5.53) с $f_k(\mathbf{x})$ в виде квадратичной формы (Lanczos 1950), причём матрица H не обязательно должна быть положительно определённой.

Метод Ланцоша во многом аналогичен методу споряженных градиентов и обладает всеми преимуществами последнего: требуется только операция уножения матрицы H на произвольный вектор, требуется дополнительно только $O(n)$ памяти, а вычисления сходятся за n итераций. Рекуррентные соотношения для построения базиса включают в себя три последовательных базисных вектора, а не два как в (5.62). Предложенный алгоритм позволяет отыскивать собственные вектора, отвечающие нескольким минимальным или максимальным собственным значениями матрицы H.

5.3 Алгоритмы решения ограниченных задач оптимизации

5.3.1 Проекционные методы

Семейство проекционных методов численного решения ограниченных задач оптимизации применяется к выпуклым целевым множествам X, не требует вычисления целевой функции $f(\mathbf{x})$ и её градиента $\nabla f(\mathbf{x})$ в точках $\mathbf{x} \notin X$, и основано на следующей геометрической интерпретации условий критической точки первого порядка. Если множество X выпукло и выполняется условие:

$$(\nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in X, \tag{5.67}$$

тогда $\mathbf{x}^*$ критическая точка первого порядка. Следовательно, необходимое условия оптимальности первого рода можно переформулировать следующим образом: если $\mathbf{x}^*$ — локальное решение органиченной задачи оптимизации, и X — выпуклое множество, тогда выполняется (5.67). Предположим, что $\mathbf{x}^*$ — локальное решение, но условие (5.67) не выполняется, т.е. $\exists \mathbf{x}_0$ такой, что $(\nabla f(\mathbf{x}^*), \mathbf{x}_0 - \mathbf{x}^*) < 0$. Тогда, во-первых, в силу выпуклости множества X весь отрезок $\mathbf{x}_\lambda = \lambda \mathbf{x}_0 + (1 - \lambda)\mathbf{x}^* \in X$, где $\lambda \in [0, 1]$. Во-вторых, подставляя $\mathbf{x}_\lambda$ в (5.67), нетрудно убедиться, что условие не выполняется для всего отрезка, кроме самой точки $\mathbf{x}^*$. Раскладывая целевую функцию в ряд Тейлора вдоль этого отрезка в окрестности $\lambda = 0$, получаем:

$$.f(\lambda \mathbf{x}_0 + (1 - \lambda)\mathbf{x}^*) - f(\mathbf{x}^*) = \lambda(\nabla f(\mathbf{x}^*), \mathbf{x}_0 - \mathbf{x}^*) + o(\lambda), \tag{5.68}$$

[11] Krylov subspace

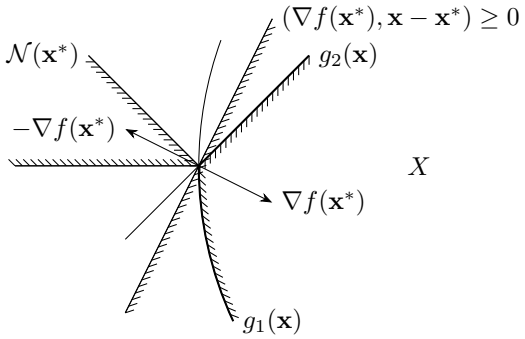

Рис. 5.3: Пример множества, заданного условием (5.67) в точке $\mathbf{x}^* = \mathbf{0}$, для задачи минимизации функции $f(\mathbf{x}) = (4x_1 - 2x_2 + 10)^2 + (x_1 + 2x_2)^2$ с ограничениями $g_1(\mathbf{x}) = -(x_1 - 1)^2 - x_2^2 + 1 \geq 0$, $g_2(\mathbf{x}) = x_1 - x_2 \geq 0$. Дополнительно показан *нормальный конус* $\mathcal{N}(\mathbf{x}^*)$, построенный на векторах $-\nabla g_1(\mathbf{x}^*)$ и $-\nabla g_2(\mathbf{x}^*)$. В соответствии с условием Каруша-Куна-Такера (5.10), вектор $-\nabla f(\mathbf{x}^*)$ лежит внутри нормального конуса.

откуда видно, что всегда можно подобрать такое достаточно малое λ_0, что $f(\lambda \mathbf{x}_0 + (1 - \lambda)\mathbf{x}^*) < f(\mathbf{x}^*)$ для $\lambda \in (0, \lambda_0)$, что в свою очередь противоречит исходному предположению о том, что $\mathbf{x}^*$ — локальное решение органиченной задачи оптимизации. Геометрический смысл (5.67) поясняется на Рис. 5.3.

Условие (5.67) выглядит не слишком практичным с точки зрения реальных алгоритмов, поэтому рассмотрим вспомогательную задачу. Проекцией точки $\mathbf{a} \in \mathrm{R}^n$ на множество X называется точка $\pi_X(\mathbf{a})$, ближайшая к $\mathbf{a}$ из всех точек X:

$$\pi_X(\mathbf{a}) \equiv \arg\min_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{a}\|^2. \tag{5.69}$$

Для ряда простых множеств X можно предложить аналитическое выражение для $\pi_X(\mathbf{a})$. Далее нас будут интересовать только такие множества, для которых проекция вычисляется просто.

Например, если X — шар радиус R с центром в $\mathbf{x}_0$, тогда

$$\pi_X(\mathbf{a}) = \begin{cases} \mathbf{a} & \mathbf{a} \in X \\ \mathbf{x_0} + \frac{\mathbf{a} - \mathbf{x}_0}{\|\mathbf{a} - \mathbf{x}_0\|} R & \mathbf{a} \notin X. \end{cases} \tag{5.70}$$

Если X — неотрицательный ортант (т.е. $x_i \geq 0 \quad \forall i$), тогда

$$(\pi_X(\mathbf{a}))_i = \max\{0, a_i\}. \tag{5.71}$$

Более общий случай — координатный параллелепипед ($l_i \leq x_i \leq u_i \quad \forall i$, причем некоторые из l_i могут принимать значение $-\infty$, а некоторые из u_i могут принимать значение $+\infty$), проекция на который вычисляется следующим образом:

$$(\pi_X(\mathbf{a}))_i = \min\{\max\{l_i, a_i\}, u_i\} = \max\{l_i, \min\{a_i, u_i\}\}. \tag{5.72}$$

Пусть X — афинное множество ($A\mathbf{x} = \mathbf{b} \quad \forall \mathbf{x} \in X$), тогда проекция запишется следующим образом:

$$\pi_X(\mathbf{a}) = \mathbf{a} - A^T(AA^T)^{-1}(A\mathbf{a} - \mathbf{b}). \tag{5.73}$$

Кроме того, для разнообразия отметим полезный случай, когда вместо аналитического выражения для проекции можно предложить простой алгоритм. Так, например, если искомый вектор $\mathbf{x}$ представляет некоторую монотонную зависимость, заданную на некоторой дискретной сетке (т.е. $x_1 \leq x_2 \leq ... \leq x_n$), тогда проекцию $\pi_X(\mathbf{a})$ можно вычислить за $O(n)$ операций (см. Best и Chakravarti 1990).

Проекция $\pi_X(\mathbf{a})$ обладает рядом следующих общих свойств, при условии, что X — выпуклое. Во-первых, проекция $\pi_X(\mathbf{a})$ существует для всех $\mathbf{a}$ и единственна, это следствие выпуклости самой функции $\|\mathbf{x} - \mathbf{a}\|^2$. Во-вторых, выполняется условие

$$(\pi_X(\mathbf{a}) - \mathbf{a}, \mathbf{x} - \pi_X(\mathbf{a})) \geq 0 \quad \forall \mathbf{x} \in X, \tag{5.74}$$

что является следствием (5.67). В-третьих, при проецировании, расстояние между любыми двумя точками $\mathbf{a}_1$ и $\mathbf{a}_2$ не возрастает:

$$\|\pi_X(\mathbf{a}_1) - \pi_X(\mathbf{a}_2)\| \leq \|\mathbf{a}_1 - \mathbf{a}_2\|, \tag{5.75}$$

что является следствием утверждения (5.74) примененного к $\pi_X(\mathbf{a}_1)$ и $\pi_X(\mathbf{a}_2)$, и неравенства Коши-Буняковского.

Теперь можно сформулировать окончательное условие критической точки первого порядка для выпуклого допустимого множества X в терминах проекции:

$$\mathbf{x}^* = \pi_X\left(\mathbf{x}^* - \lambda \nabla f(\mathbf{x}^*)\right) \quad \forall \lambda > 0, \tag{5.76}$$

где точка $\mathbf{x}^*$ — критическая точка первого порядка. Действительно, с одной стороны, при рассмотрении точки $\mathbf{a} = \mathbf{x}^* - \lambda \nabla f(\mathbf{x}^*)$ из (5.74) и (5.76) следует геометрическое определения критической точки (5.67).

С другой стороны, рассмотрим вектор $\pi_X(\mathbf{a}) - \mathbf{x}^*$, тогда, во-первых, из (5.67) следует $(\nabla f(\mathbf{x}^*), \pi_X(\mathbf{a}) - \mathbf{x}^*) \geq 0$, и, во-вторых, из (5.74) следует $\|\pi_X(\mathbf{a}) - \mathbf{x}^*\|^2 \leq -\lambda(\nabla f(\mathbf{x}^*), \pi_X(\mathbf{a}) - \mathbf{x}^*)$, следовательно, выполняется (5.76).

Метод проекции градиента

Условие (5.76) — основа проекционных методов численного решения ограниченных задач оптимизации. Например, можно предложить метод проекции градиента, адаптировав метод градиентного спуска (5.32) следующим образом:

$$\mathbf{x}^{(k+1)} = \pi_X\left(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})\right). \tag{5.77}$$

Сходимость такого метода объясняется тем, что при определённых условиях (5.77) является сжимающим отображением, для которого должна существовать неподвижная точка $\mathbf{x}^*$, удовлетворяющая условию (5.76).

Метод наименьших квадратов с ограничениями неотрицательности параметров

В качестве другого примера приведём метод, предложенный для численного решения неотрицательной задачи наименьших квадратов[12] (5.22)-(5.23), см. Лоусон и Хенсон 1986. Заметим, что ограничения вида (5.23) разбивают R^n на 2^n подмножеств. Мы могли бы в каждом из таких подмножеств решить задачу наименьших квадратов одним из известных способов, а затем сравнить значения целевых функций и выбрать окончатльное решение задачи. Такой подход на практике не реализуем, так как число таких подзадач растет как степень двойки размерности пространства параметров.

Вместо этого, заметим, что целевая функция (5.22) — выпуклая, допустимое множество X, заданное условиями (5.23), — выпуклое, следовательно, условие (5.76) является необходимым и достаточным условием оптимальности ограниченной задачи. Перепишем (5.76) для конкретного вида операции проекции и градиента целевой функции:

$$(\mathbf{x}^*)_i = \max\left\{0, \left(\mathbf{x}^* + \lambda A^T(\hat{\mathbf{b}} - A\mathbf{x}^*)\right)_i\right\}, \tag{5.78}$$

откуда видно, что либо одновременно $(\mathbf{x}^*)_i = 0$ и соответствующий компонент вектора антиградиента $\left(A^T(\hat{\mathbf{b}} - A\mathbf{x}^*)\right)_i \leq 0$, либо $(\mathbf{x}^*)_i > 0$ и соответствующий компонент вектора градиента равен нулю.

Предлагаемый алгоритм строит последовательность $\mathbf{x}^{(k)}$, начинающуюся с $\mathbf{x}^{(0)} = 0$ и сходящуюся к решению задачи $\mathbf{x}^*$, а также дополнительно хранит $\mathcal{Z}^{(k)}$ — множество индексов таких, что $\left(\mathbf{x}^{(k)}\right)_i = 0$ для всех $i \in \mathcal{Z}^{(k)}$, и дополнительное к этому множеству $\mathcal{P}^{(k)}$. Обозначение $A_{\mathcal{P}}$ используется для редуцированной матрицы задачи, построенной только из тех колонок исходной матрицы A, индексы которых принадлежат множеству $\mathcal{P}$.

Алгоритм состоит из основого и вспомогательного циклов:

- Каждый шаг начинается с вычисления вектора антиградиента $\mathbf{w}^{(k)} \equiv A^T(\hat{\mathbf{b}} - A\mathbf{x}^k)$ и проверки условий (5.78) для $i \in \mathcal{Z}^{(k)}$, условия для $i \in \mathcal{P}^{(k)}$ выполняются автоматически по построению каждого приближения.

- Если оказалось, что необходимые и достаточные условия (5.78) не выполнены, т.е. существует такой индекс t, при котором одновременно $x_t^{(k)} = 0$ и $w_t^{(k)} > 0$, тогда необходимо выполнить процедуру поиска лучшего приближения $\mathbf{x}^{(k+1)}$, в котором, очевидно, $x_t^{(k+1)} > 0$ и его значение подлежит определению. В случае, когда условия нарушены сразу для нескольких индексов, разумно выбрать и работать с индексом соответствующим максимальному значению антиградиента: $t = \arg\max_{i \in \mathcal{Z}^{(k)}} w_i^{(k)}$.

[12] *Non-negative least squares (NNLS)*

- Итак, теперь $\mathcal{Z}^{(k+1)}$ состоит из всех индексов $\mathcal{Z}^{(k)}$, кроме t, а в $\mathcal{P}^{(k+1)}$ индекс t, напротив, добавлен. Чтобы вычислить $x_i^{(k+1)}$ для $i \in \mathcal{P}^{(k+1)}$ требуется решить редуцированную задачу наименьших квадратов с матрицей $A_{\mathcal{P}^{(k+1)}}$ одним из уже известных способов, в то время как $x_i^{(k+1)} = 0$ для $i \in \mathcal{Z}^{(k+1)}$.

- Если $x_i^{(k+1)} \geq 0$ для всех индексов i, тогда следует начать новый шаг с проверки условий (5.78), иначе потребуется вспомогательный восстановительный цикл:

  – Рассмотрим отрезок $\mathbf{x} = \lambda \mathbf{x}^{(k+1)} + (1-\lambda)\mathbf{x}^{(k)}$, для $\lambda \in [0, 1]$, и найдём $\lambda^{'} > 0$ соответвующую пересечению с границей допустимого множества X. Это удобно сделать перебором:

  $$\lambda^{'} = \min_{\substack{i \in \mathcal{P}^{(k+1)} \\ x_i^{(k+1)} \leq 0}} \frac{x_i^{(k)}}{x_i^{(k)} - x_i^{(k+1)}}.$$

  В силу выпуклости целевой функции, точка $\mathbf{x}^{(k+2)} = \lambda' \mathbf{x}^{(k+1)} + (1 - \lambda')\mathbf{x}^{(k)}$ обеспечивает приближение с меньшим значение целевой функции, чем в исходной точке $\mathbf{x}^{(k)}$. Множества $\mathcal{Z}^{(k+2)}$ и $\mathcal{P}^{(k+2)}$ определены согласованным с $\mathbf{x}^{(k+2)}$ образом.

  – Заметим теперь, что после процедуры выбора $\lambda^{'}$ компоненты градиента целевой функции в точке $\mathbf{x}^{(k+2)}$ могут быть отличны от нуля для индексов $i \in \mathcal{P}^{(k+2)}$, что противоречит условиям (5.78). Необходмо вновь решить редуцированную задачу наименьших квадратов с матрицей $A_{\mathcal{P}^{(k+2)}}$, что приведёт к новому улучшенному приближению, некоторые из компонентов которого в свою очередь опять могут оказаться отрицательными, и в таком случае вспомогательный цикл следует вновь повторить.

Видно, что на каждом шаге основного и вспомогательного циклов значение целевой функции убывает. Значит, алгоритм сойдётся к решению через конечное число шагов, среднее число которых оценивается авторами в $O(n)$.

5.3.2 Методы штрафных функций и барьерные методы

К сожалению, численное решение задачи математического программирования (5.4)–(5.6) в самом общем её виде затруднительно. *Методы штрафных функций*[13] — одна из идей, которую можно попробовать применить для решения такой задачи. Принцип состоит в том, чтобы заменить ограниченную задачу оптимизации (5.4)–(5.6) на последовательность неограниченных задач оптимизации, решения которых сходятся к решению исходной задачи.

[13] *Penalty-function methods*

В разделе 3.4, кроме всего прочего, обсуждался метод регуляризации Тихонова, который состоял в модификации целевой функции метода линейных наименьших квадратов путём доавления аддитивного члена с некоторым весом α. Было наглядно показано, что по мере $\alpha \to \infty$ искомая оценка $\hat{\theta}_\delta \to \mathbf{0}$ независимо от задачи. Идейно, предлагается поступить похожим образом, т.е. заменить исходную целевую функцию на новую:

$$\Phi(\mathbf{x}, C) \equiv f(\mathbf{x}) + \phi(\mathbf{x}, C), \tag{5.79}$$

где $\phi(\mathbf{x}, C)$ — штрафная функция, а C — некоторый параметр. Штрафная функция $\phi(\mathbf{x}, C) \to 0$ при $C \to \infty$ внутри допустимого множества X, возможно, за исключением границы множества, вместе с этим $\phi(\mathbf{x}, C) \to \infty$ при $C \to \infty$ вне допустимого множества X.

Приведём несколько примеров популярных штрафных функций:

- Степенная штрафная функция:

$$\phi(\mathbf{x}, C) = C \sum_{i \in \mathcal{I}} |\min\{0, g_i(\mathbf{x})\}|^q, \quad q > 0. \tag{5.80}$$

- Квадратичная штрафная функция:

$$\phi(\mathbf{x}, C) = C \sum_{i \in \mathcal{E}} g_i^2(\mathbf{x}). \tag{5.81}$$

- Экспоненциальная штрафная функция:

$$\phi(\mathbf{x}, C) = C \sum_{i \in \mathcal{I}} \exp\left(-C g_i(\mathbf{x})\right). \tag{5.82}$$

- Обратная функция:

$$\phi(\mathbf{x}, C) = \begin{cases} \frac{1}{C} \sum\limits_{i \in \mathcal{I}} g_i^{-1}(\mathbf{x}), & \text{если } g_i(\mathbf{x}) > 0 \quad \forall i \in \mathcal{I} \\ +\infty & \text{иначе.} \end{cases} \tag{5.83}$$

- Логарифмическая функция:

$$\phi(\mathbf{x}, C) = -\frac{1}{C} \sum_{i \in \mathcal{I}} \ln g_i(\mathbf{x}). \tag{5.84}$$

Функции вида (5.83) или (5.84) часто называют *барьерными*[14] функциями, так как такие функции, в отличии например от (5.80), отличны от нуля внутри допустимой области и начинают сильно расти по мере приближения к границе.

[14] *Barrier*

Может возникнуть наивное желание, состоящее в том, чтобы выбрать C достаточно большим числом и применить один из известных алгоритмов численного решения неограниченной задачи оптимизации для минимизации функции (5.79). Отметим, что на практике так сделать невозможно, так как это непрактично с численной точки зрения из-за возникающей овражности функции $\Phi(\mathbf{x}, C)$, больших значений градиента, и резких перепадов функции. Вместо этого, следует рассматривать последовательность задач оптимизации (5.79) для $C_k \to \infty$, находя на каждом шаге $\mathbf{x}^{(k)}$ — минимум задачи соответствующей C_k. Сделать это можно каким-то методом безусловной численной оптимизации. Если повторять этот шаг, одновременно увеличивая C_k и уменьшая ошибку ϵ_k, то последовательность $\mathbf{x}^{(k)}$ будет стремиться к локальному решению исходной ограниченной задачи.

5.4 Стохастические алгоритмы оптимизации

Термин *стохастическая оптимизация* зачастую понимается двояко. Во-первых, речь может идти об оптимизации случайной целевой функции, т.е. о задаче поиска минимума случайной функции в среднем или наиболее вероятном смысле при условии, что известен набор реализаций этой функции при разных значениях переменной. Можно было бы подумать, что из-за ошибок округления о любой функции можно подумать как о стохастической, но на самом деле имеются ввиду задачи близкие области, которая называется теория управления.

Во-вторых, речь может идти о том, что в алгоритм вычисления улучшенного приближения $\mathbf{x}^{(k+1)} = \mathbf{S}_k\left[\mathbf{x}^{(k)}\right]$ вносится элемент случайности. Мы будем использовать термин стохастическая оптимизация только в этом смысле и говорить о *стохастических алгоритмах оптимизации*. В отличии от детерминированных алгоритмов численной оптимизации, теперь идет речь о случайных последовательностях $\mathbf{x}^{(k)}$, которые, как ожидается, сходятся в среднем или с некоторой вероятностью к критической точке целевой функции. В качестве меры эффективности алгоритма, с одной стороны, аналогично детерминированному случаю мы можем рассматривать асимптотическое поведение среднего уклонения точки $\mathbf{x}^{(k)}$ от критической точки $\mathbf{x}^{(*)}$ с ростом номера шага. С другой стороны, сходимость стохастических алгоритмов оптимизации часто гарантируется с некоторой конечной вероятностью p, которая некоторым образом может зависеть как от настоечных параметров, так и от ожидаемой точности решения или числа итераций.

Приведем пример простого стохастического алгоритма оптимизации, известного как *метод случайного спуска*. Метод состоит из двух шагов:

1. Пусть $\mathbf{x}^{(k)}$ текущая точка, выберем случайную точку $\mathbf{x}'$ в окрестности $\|\mathbf{x}^{(k)} - \mathbf{x}'\|^2 \leq \Delta$.

2. Если $f(\mathbf{x}') < f(\mathbf{x}^{(k)})$, то $\mathbf{x}^{(k+1)} = \mathbf{x}'$, иначе выберем другую реализацию $\mathbf{x}'$ и повторим сравнение.

В эффективности такого подхода возникают естественные сомнения, так как при неудачном выборе распределения $\mathbf{x}'$ вероятность события $f(\mathbf{x}') < f(\mathbf{x}^{(k)})$ может оказаться малой.

5.4.1 Стохастический градиентный спуск

Стохастический градиентный спуск[15] — одни из важных представителей стохастических алгоритмов оптимизации, завоевавший популярность в последнее время благодаря росту популярности методов машинного обучения. Считается, что впервые идея метода была опубликована в работе Robbins и Monro 1951.

Метод применим для оптимизации целевых функций, представимых в виде суммы:

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}), \tag{5.85}$$

и, следовательно,

$$\nabla f(\mathbf{x}) = \nabla \sum_{i=1}^{m} f_i(\mathbf{x}). \tag{5.86}$$

Например, таким свойством очевидно обладает целевая функция задачи нелинейных наименьших квадратов (5.24).

Ранее обсуждалась дилемма состоящая в том, что на каждом шаге алгоритма требуется вычислить аппроксимацию целевой функции и баланс между качеством этой аппроксимации и требуемыми вычислительными ресурсами не всегда очевиден. Так, например, можно затратить вычислительные ресурсы для расчета матрицы вторых производных целевой функции, и достичь более высокой скорости сходимости и меньшего числа требуемых шагов, либо применить алгоритм требующий расчета только первых производных, который потребует большего числа шагов, зато каждый шаг будет рассчитываться очень быстро. В задачах типа (5.24) число m может оказаться относительно велико, например, когда речь идет о применении метода нейронных сетей к задаче регрессии (5.24), то размер так называемой обучающей выборки варьируется от тысяч до миллионов записей, каждая из которых является отдельным элементом наблюдаемого вектора $\hat{\mathbf{b}}$. В таком случае, вычисление даже самой целевой функции $f(\mathbf{x})$ и ее градиента $\nabla f(\mathbf{x})$ требует заметных вычислительных ресурсов.

Идея метода стохастического градиентного спуска состоит в следующем. Заметим, что задача минимизации (5.85) эквивалентна задаче

$$f(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} f_i'(\mathbf{x}), \tag{5.87}$$

где правая часть выражения имеет вид выборочного среднего, составленного из m элементов. При этом, если рассмотреть такую же сумму по любому

[15] *stochastic gradient descent*

подмножеству индексов i, то получится тоже выборочное среднее, обладающее меньшей точностью в силу меньшего размера выборки. Понятно, что для метода градиентного спуска вместо $\nabla f(\mathbf{x}^{(k)})$ можно взять

$$\mathbf{g}^{(k)} \equiv \sum_{i \in I^{(k)}} \nabla f_i(\mathbf{x}), \tag{5.88}$$

где $I^{(k)}$ некоторое случайное подмножество индексов маленького размера, новое на каждом шаге, часто его называют *пакетом*[16]:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}. \tag{5.89}$$

При таком подходе скорость вычисления каждого шага может быть увеличена относительно детерменированного метода градиентного спуска.

5.4.2 Метод адаптивной оценки моментов

Понятно, что идея стохастического градиентного спуска имеет множество дополнений и улучшений. Например, предложенный в работе Kingma и Ba 2017, метод адаптивной оценки моментов[17] улучшает точность с которой оценивается вектор градиента $\nabla f_i(\mathbf{x})$. Понятно, что среднее различие между $\nabla f(\mathbf{x}^{(k)})$ и $\mathbf{g}^{(k)}$ увеличивается по мере уменьшения размера выборки $I^{(k)}$, а увеличивать эту выборку мы не хотим по причинам вычислительных затрат. Вместо этого, можно в некотором смысле заменить среднее по ансамблю средним по времени, предполагая, что на шагах k и $k+1$ вектора $\nabla f(\mathbf{x}^{(k)})$ и $\nabla f(\mathbf{x}^{(k+1)})$ отличаются не сильно, т.е. много меньше, чем неопределенность вносимая за счет стохастического метода оценивания.

Предлагается наряду с вектором $\mathbf{g}^{(k)}$ вычислять и хранить его сглаженную версию $\mathbf{m}^{(k)}$:

$$\mathbf{m}^{(k)} = \beta_1 \mathbf{m}^{(k-1)} + (1 - \beta_1)\mathbf{g}^{(k)}, \tag{5.90}$$

где β_1 — константа сглаживания, например $\beta_1 = 0.9$, и вектор оценки поэлементных нецентральных вторых моментов $\mathbf{v}^{(k)}$:

$$v_i^{(k)} = \beta_2 v_i^{(k-1)} + (1 - \beta_2) \cdot \left(g_i^{(k)}\right)^2, \tag{5.91}$$

где β_2 — константа сглаживания для вектора $\mathbf{v}^{(k)}$, например, равная $\beta_2 = 0.999$. Вектор $\mathbf{v}^{(k)}$ характеризует степень разброса каждой компоненты $\mathbf{g}^{(k)}$. Следующее приближение вычисляется аналогично методу стохастического градиентного спуска:

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{1 - \beta_1^k} \left(\epsilon + \sqrt{\frac{v_i^{(k)}}{1 - \beta_2^k}} \right)^{-1} m_i^{(k)}, \tag{5.92}$$

где рекомендуемые значения констант $\alpha = 0.001$ и $\epsilon = 10^{-8}$.

[16] *batch*

[17] Adaptive moment estimator (ADAM)

5.4.3 Методы Монте-Карло по схеме марковской цепи

Алгоритм имитации отжига[18] (имеется ввиду металлургический термин) — еще один популярный стохастический алгоритм численной оптимизации, позволяющий находить глобальный минимум функции и обладающий красивой физической интерпретацией. Однако, перед рассмотрением непосредственно задачи оптимизации, изучим вспомогательную задачу о генерации реализаций случайного вектора с требуемым распределением.

Алгоритм Метрополиса

Николас Метрополис вместе с коллегами по Лос-Аламосской национальной лаборатории предложили (см. Metropolis и др. 1953) численный метод взятия многомерных интегралов из статистической физики:

$$\langle X \rangle = \int \mathbf{dpdq} X(\mathbf{p}, \mathbf{q}) \exp\left(-\frac{E(\mathbf{p}, \mathbf{q})}{\theta}\right), \tag{5.93}$$

где X некоторая физическая величина, которую требуется усреднить по ансамблю большого числа частиц, а вектора $\mathbf{p} \in R^{3n}$ и $\mathbf{q} \in R^{3n}$ обозначают состояние системы в фазовом пространстве, θ — некоторый параметр, называемый температурой, и $E(\mathbf{p}, \mathbf{q})$ — обобщенная энергия системы. Понятно, что проблема размерности помешает взять такой интеграл классическими методами на равномерной сетке, однако и наивный подход Монте-Карло с набрасыванием точек будет не очень результативен, так как экспонента стоящая в подынтегральном выражении принимает значения близкие к нулю почти во всем фазовом пространстве. Выход состоит в том, чтобы генерировать случайные состояния системы ($\mathbf{p}$, $\mathbf{q}$) таким образом, чтобы плотность вероятности значений функции $E(\mathbf{p}, \mathbf{q})$ была пропорционально $\exp\left(-\frac{E(\mathbf{p},\mathbf{q})}{\theta}\right)$.

Пусть известно текущее состояние $\mathbf{x}^{(k)} \equiv (\mathbf{p}^{(k)}, \mathbf{q}^{(k)})$, сгенерируем новое пробное состояние $\mathbf{x}'$ используя вспомогательное условное распределение $p_G(\mathbf{x}'|\mathbf{x}^{(k)})$. Вспомогательное распределение $p_G(\mathbf{x}'|\mathbf{x}^{(k)})$ предполагается таким, для которого известен эффективный алгоритм генерирования реализаций. Например, в оригинальной работе предлагается следующая схема:

$$\mathbf{x}' = \mathbf{x}^{(k)} + \alpha\xi, \tag{5.94}$$

где α параметр, а ξ вектор случайных величин равномерно распределенных от -1 до 1. В качестве альтернативы мы могли бы предложить использовать многомерное нормальное распределение с средним в точке $\mathbf{x}^{(k)}$ и некоторой дисперсией σ_0^2 одинаковой для всех компонент. Сравним значения функции $E(\mathbf{x}')$ и $E(\mathbf{x}^{(k)})$:

- Если $E(\mathbf{x}') < E(\mathbf{x}^{(k)})$, то $\mathbf{x}^{(k+1)} = \mathbf{x}'$

[18] *simulated annealing*

- Если $E(\mathbf{x}') \geq E(\mathbf{x}^{(k)})$, тогда
  - $\mathbf{x}^{(k+1)} = \mathbf{x}'$ с вероятностью $\exp\left(-\frac{\Delta E}{\theta}\right)$,
  - $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ с вероятностью $1 - \exp\left(-\frac{\Delta E}{\theta}\right)$.

Во втором случае разыгрывается дополнительная случайная величина, равномерно распределенная на отрезке $[0, 1]$, и на основании её значения и значения $\exp\left(-\frac{\Delta E}{\theta}\right)$ происходит окончательное решение.

Видно, что представленный алгоритм фактически задает способ построения марковской цепи, причем вероятность перехода из состояния i в состояние $j \neq i$ определяется как

$$p_{ij} \equiv p(\mathbf{x}^{(k+1)} = \mathbf{x}_j | \mathbf{x}^{(k)} = \mathbf{x}_i) = p_G(\mathbf{x}^{(k+1)} | \mathbf{x}^{(k)}) \cdot \min\left\{1, \exp\left(-\frac{\Delta E}{\theta}\right)\right\}, \tag{5.95}$$

при этом выполняется условие нормировки вероятности

$$\sum_j p_{ij} = 1, \tag{5.96}$$

откуда при необходимости можно определить вероятность остаться в текущем состоянии $p_{ii} = p(\mathbf{x}^{(k+1)} = \mathbf{x}_i | \mathbf{x}^{(k)} = \mathbf{x}_i)$. Для простоты, мы считаем, что число возможных состояний дискретно, но для континуума состояний рассуждения тоже остаются верными, а соответствующие суммы заменяются на интегралы.

Видно, что вероятность перехода из состояния i в состояние j не зависит от номера шага k, значит марковская цепь является однородной по времени. Одним из важных свойством алгоритма должна быть возможность достигнуть любое состояние из любого другого за конечное число шагов с ненулевой вероятностью, это достигается правильным выбором $p_G(\mathbf{x}^{(k+1)} | \mathbf{x}^{(k)})$. Известно, что хорошая цепь Маркова будет иметь предельное распределение вероятностей состояний для $k \to \infty$, а в случае эргодической цепи можно сказать, что частота разных состояний во времени соответствует предельному распределению вероятностей:

$$p(\mathbf{x}^{(k)}) = p(\mathbf{x}^{(\infty)}). \tag{5.97}$$

Используя принцип детального равновесия можно показать, что в случае алгоритма Метрополиса вероятность выпадения каждого состояния i действительно пропорциональна $\exp\left(-\frac{\Delta E}{\theta}\right)$. Пусть N_i — число точек в состоянии i, которое нам встретилось за некоторое большое число шагов $N \to \infty$, пусть N_j — число точек в состоянии j, и пусть для определенности $E(\mathbf{x}_i) < E(\mathbf{x}_j)$. Согласно принципу детального равновесия, в эргодической цепи Маркова количество переходов «туда» (из i и j) должно соответствовать числу переходов «оттуда» (из j в i). Количество переходов из i в j равно $N_i \exp\left(-\frac{E(\mathbf{x}_j) - E(\mathbf{x}_i)}{\theta}\right)$, количество обратных переходов — строго N_j, так как в состоянии i значение энергии меньше, чем в состоянии j. Приравнивая

число переходов между состояниями в обе стороны, получим следующую пропорцию:

$$\frac{N_i}{N_j} = \frac{\exp\left(-\frac{E(\mathbf{x}_i)}{\theta}\right)}{\exp\left(-\frac{E(\mathbf{x}_j)}{\theta}\right)}, \tag{5.98}$$

откуда понятно, что $p_i = \frac{N_i}{N} \sim \exp\left(-\frac{E(\mathbf{x}_i)}{\theta}\right)$.

Алгоритм имитации отжига

Из курса статистической физики известна теорема, именуемая иногда третьим началом термодинамики. Распределение $p(\mathbf{x}) \sim \exp\left(-\frac{E(\mathbf{x})}{\theta}\right)$ при $\theta \to 0$ устремится к дельта-функции в точке $\mathbf{x}^*$, соответствующей глобальному минимуму функции $E(\mathbf{x})$. Иными словами, при температуре абсолютного нуля система многих частиц может занимать только одно состояние — соответствующее минимально возможной энергии.

Используя целевую функцию $f(\mathbf{x})$ вместо $E(\mathbf{x})$ можно попытаться построить распределение $p(\mathbf{x})$ при очень маленьком значении параметра θ. Однако, чем меньше θ тем больше шагов требуется для достижения равновесного состояния, поэтому на практике применяется подход постепенного охлаждения $\theta_k \to 0$, который и называется алгоритмом имитации отжига. Подход в чём-то похож на подход из раздела 5.3.2 о методах штрафных функций.

При больших температурах θ (начальную температуру следует выбрать такой, чтобы разрешить системе занимать все возможные состояния с большой вероятностью) система быстро приходит в равновесное состояние, распределение в котором $p(\mathbf{x}) \sim \exp\left(-\frac{E(\mathbf{x})}{\theta}\right)$. При постепенном и правильном уменьшении параметра θ распределение будет квазиравновестно сходиться к ожидаемому распределению в окрестности решения $\mathbf{x}^*$.

К сожалению, снизить температуру не испортив равновесности состояния бывает достаточно затруднительно. Предлагаются различные схемы охлаждения, например

$$\theta^{(k+1)} = \alpha\theta^{(k)}, \tag{5.99}$$

где α параметр, который чуть меньше единицы. Рекомендуется делать понижение температуры следует делать раз в некоторое число итераций основного алгоритма Метрополиса. Ориентиром может являться число принятых переходов, свидетельствующее о том, что система пришла в равновесие. Завершать алгоритм предлагается в тот момент, когда отношение принятых и предложенных переходов становится меньше некоторой границы.

Следует однако сказать, что при слишком консервативном (медленном) охлаждении алгоритм сходится чрезвычайно медленно, а при быстром охлаждении алгоритм попадает в локальные минимумы (аналог метастабильных состояний в кристаллических решетках) и застревает там.