Курс: Методы оптимизации в машинном обучении, 2012

Продвинутые методы многомерной оптимизации

Рассмотрим задачу безусловной оптимизации в многомерном пространстве:

$$f(\boldsymbol{x}) \to \min_{\boldsymbol{x} \in \mathbb{R}^N}.$$

Ранее для решения этой задачи были рассмотрены методы покоординатного и градиентного спуска, метод Ньютона, а также комбинированные методы. Во всех этих подходах очередное направление оптимизации $\boldsymbol{d}_k$ вычисляется только с использованием информации от оракула в текущей точке $\boldsymbol{x}_k$. Таким образом, траектория оптимизации никак не учитывается[1]. В методах оптимизации, рассматриваемых ниже, очередное направление оптимизации $\boldsymbol{d}_k$ существенно зависит от траектории оптимизации и, в частности, зависит от предыдущих направлений $\boldsymbol{d}_{k-1}, \ldots, \boldsymbol{d}_0$.

Решение СЛАУ с помощью метода сопряжённых градиентов

Рассмотрим задачу минимизации квадратичной функции с положительно-определённым гессианом:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b} \to \min_{\boldsymbol{x}}, \quad A = A^T \succ 0. \tag{1}$$

Приравнивая к нулю градиент f, получаем:

$$\nabla f(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0} \;\Rightarrow\; A\boldsymbol{x} = \boldsymbol{b}.$$

Таким образом, задача минимизации f эквивалентна решению СЛАУ $A\boldsymbol{x} = \boldsymbol{b}$.

Назовём набор векторов $\{\boldsymbol{d}_i\}_{i=0}^k$ сопряжённым относительно положительно-определённой матрицы A, если выполнено условие:

$$\boldsymbol{d}_i^T A\boldsymbol{d}_j = 0 \;\; \forall i \neq j.$$

Примером сопряжённых направлений является набор собственных векторов матрицы A. Действительно, в этом случае $\boldsymbol{d}_i^T A\boldsymbol{d}_j = \lambda_i \boldsymbol{d}_i^T\boldsymbol{d}_j = 0$. Последнее условие выполнено, т.к. собственные вектора, отвечающие различным собственным значениям, ортогональны, а среди собственных векторов с одинаковым собственным значением всегда можно выбрать ортогональный набор.

Набор сопряжённых векторов является линейно-независимым[2]. Рассмотрим нетривиальную линейную комбинацию сопряжённых векторов:

$$\sum_{i=0}^{k} \alpha_i \boldsymbol{d}_i = \boldsymbol{0}, \;\; \exists \alpha_i \neq 0.$$

Домножим слева это уравнение на $\boldsymbol{d}_j^T A$ для некоторого j. В результате получим:

$$\sum_{i=0}^{k} \alpha_i \boldsymbol{d}_j^T A\boldsymbol{d}_i = \alpha_j \boldsymbol{d}_j^T A\boldsymbol{d}_j = 0 \;\Rightarrow\; \alpha_j = 0.$$

Последнее условие следует из того, что матрица A является строго положительно-определённой. Из линейной независимости вытекает, что сопряжённый набор $\{\boldsymbol{d}_i\}_{i=0}^{N-1}$ является базисом всего пространства $\mathbb{R}^N$. В частности, решение СЛАУ $\boldsymbol{x}_*$ можно разложить по этому базису с некоторыми коэффициентами α_i:

$$\boldsymbol{x}_* = \sum_{i=0}^{N-1} \alpha_i \boldsymbol{d}_i. \tag{2}$$

Домножая это уравнение слева на $\boldsymbol{d}_j^T A$, получаем:

$$\boldsymbol{d}_j^T A\boldsymbol{x}_* = \sum_{i=0}^{N-1} \alpha_i \boldsymbol{d}_j^T A\boldsymbol{d}_i = \alpha_j \boldsymbol{d}_j^T A\boldsymbol{d}_j \;\Rightarrow\; \alpha_j = \frac{\boldsymbol{d}_j^T A\boldsymbol{x}_*}{\boldsymbol{d}_j^T A\boldsymbol{d}_j} = \frac{\boldsymbol{d}_j^T\boldsymbol{b}}{\boldsymbol{d}_j^T A\boldsymbol{d}_j}. \tag{3}$$

[1] это не совсем так при использовании адаптивной коррекции длины шага

[2] На самом деле набор сопряжённых векторов можно рассматривать как ортогональный базис относительно скалярного произведения $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T A\boldsymbol{y}$. Отсюда становится очевидным линейная независимость векторов, а также то, что матрица A должна быть положительно-определённой (иначе не будет скалярного произведения).

Последнее равенство вытекает из того, что $\boldsymbol{x}_*$ – решение СЛАУ, т.е. $A\boldsymbol{x}_* = \boldsymbol{b}$. Здесь становится очевидным преимущество сопряжённого набора векторов относительно других базисов для поиска $\boldsymbol{x}_*$: для сопряжённого набора коэффициенты разложения α_j вычисляются аналитически. Поиск $\boldsymbol{x}_*$ по формуле (2) с коэффициентами (3) можно представить в виде итерационного процесса:

$$\boldsymbol{x}_0 = \boldsymbol{0}, \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \quad \alpha_k = \frac{\boldsymbol{d}_k^T \boldsymbol{b}}{\boldsymbol{d}_k^T A \boldsymbol{d}_k}, \ k = 0, \ldots, N.$$

В результате $\boldsymbol{x}_{N+1} = \boldsymbol{x}_*$.

Теперь получим аналогичный итерационный процесс для поиска $\boldsymbol{x}_*$, который начинается с произвольного ненулевого вектора $\boldsymbol{x}_0$. Для этого разложим вектор $\boldsymbol{x}_* - \boldsymbol{x}_0$ по сопряжённому базису:

$$\boldsymbol{x}_* - \boldsymbol{x}_0 = \sum_{i=0}^{N-1} \alpha_i \boldsymbol{d}_i.$$

Домножая это равенство на $\boldsymbol{d}_j^T A$ слева, получаем:

$$\boldsymbol{d}_j^T A(\boldsymbol{x}_* - \boldsymbol{x}_0) = \alpha_j \boldsymbol{d}_j^T A \boldsymbol{d}_j \ \Rightarrow \ \alpha_j = \frac{\boldsymbol{d}_j^T A(\boldsymbol{x}_* - \boldsymbol{x}_0)}{\boldsymbol{d}_j^T A \boldsymbol{d}_j} = \frac{\boldsymbol{d}_j^T (\boldsymbol{b} - A\boldsymbol{x}_0)}{\boldsymbol{d}_j^T A \boldsymbol{d}_j} = -\frac{\boldsymbol{d}_j^T \boldsymbol{g}_0}{\boldsymbol{d}_j^T A \boldsymbol{d}_j} = -\frac{\boldsymbol{d}_j^T \boldsymbol{g}_j}{\boldsymbol{d}_j^T A \boldsymbol{d}_j}. \tag{4}$$

Здесь и далее через $\boldsymbol{g}_j$ обозначается градиент функции f в точке $\boldsymbol{x}_j$. Для квадратичной функции $\boldsymbol{g}_j = A\boldsymbol{x}_j - \boldsymbol{b}$. Докажем последний переход в формуле (4):

$$\boldsymbol{d}_j^T (\boldsymbol{g}_j - \boldsymbol{g}_0) = \boldsymbol{d}_j^T (A\boldsymbol{x}_j - \boldsymbol{b} - A\boldsymbol{x}_0 + \boldsymbol{b}) = \boldsymbol{d}_j^T A \left(\sum_{i=0}^{j-1} \alpha_i \boldsymbol{d}_i \right) = \sum_{i=0}^{j-1} \alpha_i \boldsymbol{d}_j^T A \boldsymbol{d}_i = 0 \ \Rightarrow \ \boldsymbol{d}_j^T \boldsymbol{g}_j = \boldsymbol{d}_j^T \boldsymbol{g}_0. \tag{5}$$

Таким образом, получаем следующий итерационный процесс для поиска $\boldsymbol{x}_*$ из произвольного начального приближения $\boldsymbol{x}_0$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \quad \alpha_k = -\frac{\boldsymbol{d}_k^T \boldsymbol{g}_k}{\boldsymbol{d}_k^T A \boldsymbol{d}_k}, \ k = 0, \ldots, N. \tag{6}$$

Можно показать, что данный итерационный процесс является процессом наискорейшего спуска для функции f вдоль сопряжённых направлений $\{\boldsymbol{d}_i\}_{i=0}^{N-1}$. Для этого достаточно убедиться в том, что коэффициент α_j, вычисляемый по формуле (4), доставляет минимум по α функции $f(\boldsymbol{x}_j + \alpha \boldsymbol{d}_j)$. Действительно,

$$\frac{\partial}{\partial \alpha} f(\boldsymbol{x}_j + \alpha \boldsymbol{d}_j)\Big|_{\alpha=\alpha_j} = \nabla f(\boldsymbol{x}_j + \alpha_j \boldsymbol{d}_j)^T \boldsymbol{d}_j = \boldsymbol{g}_{j+1}^T \boldsymbol{d}_j = (A\boldsymbol{x}_j + \alpha_j A\boldsymbol{d}_j - \boldsymbol{b})^T \boldsymbol{d}_j =$$

$$= (A\boldsymbol{x}_j - \boldsymbol{b})^T \boldsymbol{d}_j + \alpha_j \boldsymbol{d}_j^T A \boldsymbol{d}_j = \boldsymbol{g}_j^T \boldsymbol{d}_j - \boldsymbol{d}_j^T \boldsymbol{g}_j = 0. \tag{7}$$

Одновременно здесь было доказано, что $\boldsymbol{g}_{j+1}^T \boldsymbol{d}_j = 0$. Покажем, что $\boldsymbol{g}_{j+1}^T \boldsymbol{d}_i = 0 \ \forall i \le j$. Действительно,

$$\boldsymbol{g}_{j+1}^T \boldsymbol{d}_i = (A\boldsymbol{x}_{j+1} - \boldsymbol{b})^T \boldsymbol{d}_i = (A(\boldsymbol{x}_0 + \sum_{i=0}^{j} \alpha_i \boldsymbol{d}_i) - \boldsymbol{b})^T \boldsymbol{d}_i = (A\boldsymbol{x}_0 - \boldsymbol{b})^T \boldsymbol{d}_i + \alpha_i \boldsymbol{d}_i^T A \boldsymbol{d}_i = \boldsymbol{g}_0^T \boldsymbol{d}_i - \boldsymbol{g}_i^T \boldsymbol{d}_i = 0.$$

Последнее равенство следует из (5). Условие ортогональности градиента $\boldsymbol{g}_{j+1}$ всем предыдущим направлениям оптимизации означает, что $\boldsymbol{x}_{j+1}$ доставляет минимум функции f в линейной оболочке $\mathcal{L}(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_j)$. Более того, можно показать, что точка $\boldsymbol{x}_j + \alpha \boldsymbol{d}_j$ минимизирует функцию f в линейной оболочке $\mathcal{L}(\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{j-1})$ для любого α. Действительно,

$$\nabla f(\boldsymbol{x}_j + \alpha \boldsymbol{d}_j)^T \boldsymbol{d}_i = (A(\boldsymbol{x}_j + \alpha \boldsymbol{d}_j) - \boldsymbol{b})^T \boldsymbol{d}_i = (A\boldsymbol{x}_j - \boldsymbol{b})^T \boldsymbol{d}_i + \alpha \boldsymbol{d}_j^T A \boldsymbol{d}_i = \boldsymbol{g}_j^T \boldsymbol{d}_i = 0. \ \forall \ i < j.$$

Таким образом, при движении вдоль очередного сопряжённого направления оптимизацию по предыдущим направлениям проводить не нужно. Это одна из отличительных особенностей метода сопряжённых направлений.

Для проведения итерационного процесса (6) необходимо указать полный набор сопряжённых направлений для матрицы A. Рассмотрим следующую схему генерации $\boldsymbol{d}_k$:

$$\boldsymbol{d}_0 = -\boldsymbol{g}_0, \quad \boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k, \quad \beta_k = \frac{\boldsymbol{g}_{k+1}^T A \boldsymbol{d}_k}{\boldsymbol{d}_k^T A \boldsymbol{d}_k}, \ k = 0, \ldots, N-1. \tag{8}$$

Верна следующая последовательность рассуждений:

$$\begin{aligned}
&\boldsymbol{d}_0 = -\boldsymbol{g}_0 \in \mathcal{L}(\boldsymbol{g}_0), \\
&\boldsymbol{g}_1 = \boldsymbol{g}_0 + \alpha_0 A\boldsymbol{d}_0 = \boldsymbol{g}_0 - \alpha_0 A\boldsymbol{g}_0 \in \mathcal{L}(\boldsymbol{g}_0, A\boldsymbol{g}_0), \\
&\boldsymbol{d}_1 = -\boldsymbol{g}_1 + \beta_0 \boldsymbol{d}_0 \in \mathcal{L}(\boldsymbol{g}_0, A\boldsymbol{g}_0), \\
&\boldsymbol{g}_2 = \boldsymbol{g}_1 + \alpha_1 A\boldsymbol{d}_1 \in \mathcal{L}(\boldsymbol{g}_0, A\boldsymbol{g}_0, A^2\boldsymbol{g}_0), \\
&\boldsymbol{d}_2 = -\boldsymbol{g}_1 + \beta_1 \boldsymbol{d}_1 \in \mathcal{L}(\boldsymbol{g}_0, A\boldsymbol{g}_0, A^2\boldsymbol{g}_0), \\
&\ldots\ldots
\end{aligned}$$

В результате набор $\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_i\}$, генерируемый по схеме (8), и набор $\{\boldsymbol{g}_0, \ldots, \boldsymbol{g}_i\}$ принадлежат одному и тому же линейному пространству $\mathcal{L}(\boldsymbol{g}_0, A\boldsymbol{g}_0, \ldots, A^i\boldsymbol{g}_0)$[3]. Следовательно, вектор $A\boldsymbol{d}_i$ можно представить в базисе $\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_k\}$, $k > i$, а вектор $\boldsymbol{g}_i$ можно представить в базисе $\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_i\}$:

$$A\boldsymbol{d}_i = \sum_{j=0}^{k} a_j \boldsymbol{d}_j, \tag{9}$$

$$\boldsymbol{g}_i = \sum_{j=0}^{i} b_j \boldsymbol{d}_j. \tag{10}$$

Покажем теперь, что схема (8) позволяет построить набор сопряжённых направлений для матрицы A. Кроме того, покажем, что при её использовании $\boldsymbol{g}_k^T \boldsymbol{d}_i = 0 \ \forall i < k$ и $\boldsymbol{g}_k^T \boldsymbol{g}_i = 0 \ \forall i < k$. Проведём доказательство по индукции. Пусть известно, что

$$\begin{aligned}
&\boldsymbol{g}_k^T \boldsymbol{d}_i = 0 \ \forall i < k, \\
&\boldsymbol{d}_k^T A\boldsymbol{d}_i = 0 \ \forall i < k, \\
&\boldsymbol{g}_k^T \boldsymbol{g}_i = 0 \ \forall i < k.
\end{aligned}$$

Докажем, что аналогичные утверждения верны для $k+1$. Из рассуждений (7) следует, что $\boldsymbol{g}_{k+1}^T \boldsymbol{d}_k = 0 \ \forall k$. Далее

$$\boldsymbol{g}_{k+1}^T \boldsymbol{d}_i = (\boldsymbol{g}_k + \alpha_k A\boldsymbol{d}_k)^T \boldsymbol{d}_i = \boldsymbol{g}_k^T \boldsymbol{d}_i + \alpha_k \boldsymbol{d}_k^T A\boldsymbol{d}_i = 0.$$

Последнее утверждение выполняется, исходя из предположения индукции. Покажем теперь, что $\boldsymbol{d}_{k+1}^T A\boldsymbol{d}_i = 0 \ \forall i \leq k$. Используя (8), получим, что

$$\boldsymbol{d}_{k+1}^T A\boldsymbol{d}_k = (-\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k)^T A\boldsymbol{d}_k = -\boldsymbol{g}_{k+1}^T A\boldsymbol{d}_k + \boldsymbol{g}_{k+1}^T A\boldsymbol{d}_k = 0 \ \forall k.$$

Далее с помощью (9) заключаем, что для $i < k$

$$\boldsymbol{d}_{k+1}^T A\boldsymbol{d}_i = (-\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k)^T A\boldsymbol{d}_i = -\boldsymbol{g}_{k+1} A\boldsymbol{d}_i = -\boldsymbol{g}_{k+1} \left(\sum_{j=0}^{k} a_j \boldsymbol{d}_j \right) = -\sum_{j=0}^{k} a_j \boldsymbol{g}_{k+1}^T \boldsymbol{d}_j = 0.$$

Осталось показать, что $\boldsymbol{g}_{k+1}^T \boldsymbol{g}_i = 0 \ \forall i < k$. Используя (10), получаем

$$\boldsymbol{g}_{k+1}^T \boldsymbol{g}_i = \boldsymbol{g}_{k+1}^T \left(\sum_{j=0}^{i} b_j \boldsymbol{d}_j \right) = \sum_{j=0}^{i} b_j \boldsymbol{g}_{k+1}^T \boldsymbol{d}_j = 0.$$

В результате получаем, что набор $\{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_{N-1}\}$ действительно является сопряжённым относительно матрицы A. Заметим, что в доказательстве сопряжённости существенно используется тот факт, что первое направление выбирается в соответствии с антиградиентом. При другом выборе $\boldsymbol{d}_0$ итерационный процесс (8) не приводит к набору сопряжённых направлений.

С помощью доказанных выше свойств $\boldsymbol{g}_k^T \boldsymbol{d}_i = 0 \ \forall i < k$ и $\boldsymbol{g}_k^T \boldsymbol{g}_i = 0 \ \forall i < k$ можно несколько упростить выражения для α_k и β_k в итерационных процессах (6), (8):

$$\begin{aligned}
\alpha_k &= -\frac{\boldsymbol{g}_k \boldsymbol{d}_k}{\boldsymbol{d}_k^T A\boldsymbol{d}_k} = -\frac{\boldsymbol{g}_k(-\boldsymbol{g}_k + \beta_{k-1}\boldsymbol{d}_{k-1})}{\boldsymbol{d}_k^T A\boldsymbol{d}_k} = \frac{\boldsymbol{g}_k^T \boldsymbol{g}_k}{\boldsymbol{d}_k^T A\boldsymbol{d}_k}, \\
\beta_k &= \frac{\boldsymbol{g}_{k+1}^T A\boldsymbol{d}_k}{\boldsymbol{d}_k^T A\boldsymbol{d}_k} = \frac{\boldsymbol{g}_{k+1}^T(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)}{\boldsymbol{d}_k^T A\boldsymbol{d}_k \alpha_k} = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}.
\end{aligned} \tag{11}$$

В результате получаем итоговый алгоритм решения СЛАУ $A\boldsymbol{x} = \boldsymbol{b}$, который получил название метода сопряжённых градиентов:

1. Задаём начальное приближение $\boldsymbol{x}_0$ и требуемую точность ε;
2. Инициализация $\boldsymbol{d}_0 = -\boldsymbol{g}_0$, $k = 0$;
3. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, где $\alpha_k = \frac{\boldsymbol{g}_k^T \boldsymbol{g}_k}{\boldsymbol{d}_k^T A\boldsymbol{d}_k}$;
4. Если $\alpha_k \|\boldsymbol{d}_k\| < \varepsilon$, то стоп;

[3] Пространства такого вида известны в линейной алгебре как пространства Крылова

5. $\boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k$, где $\beta_k = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}$;

6. $k = k + 1$ и переход к шагу 3.

Данный алгоритм гарантированно сходится к решению за N шагов, где N – размерность пространства решения. На каждом шаге итерационного процесса требуется проводить только одно умножение матрицы A на вектор $\boldsymbol{d}_k$ (вектор $A\boldsymbol{x}_{k+1}$ при этом вычисляется как $A\boldsymbol{x}_k + \alpha_k A\boldsymbol{d}_k$). Остальные операции в алгоритме являются векторными: скалярное произведение двух векторов и сумма двух векторов. На рис. 1 показан пример применения этого алгоритма. Для сравнения показана также траектория метода наискорейшего спуска.

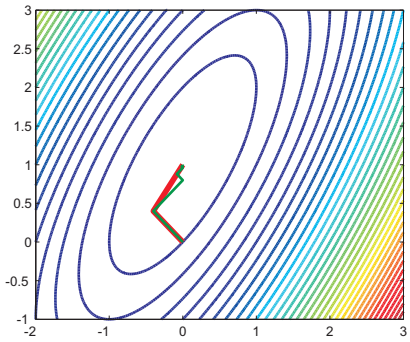


Рис. 1: Пример работы метода наискорейшего спуска (зеленая траектория) и метода сопряжённых градиентов (красная траектория) для оптимизации квадратичной функции.

Основные преимущества метода сопряжённых градиентов связаны с пространствами большой размерности. При $N \gg 1$ методы решения СЛАУ, основанные на матричных разложениях, например, разложении Холецкого или QR-разложении, перестают быть применимыми в силу необходимости итерационного пересчета матриц, размер которых совпадает с размером матрицы A. Напротив, в методе сопряжённых градиентов все операции производятся только для векторов размерности N, а самая сложная операция в алгоритме – это умножение матрицы A на очередной вектор $\boldsymbol{d}_k$. Для ряда матриц специального вида, например, разреженных матриц или матриц, представляющих базис Фурье, такое умножение может быть проведено эффективнее общего случая.

Метод сопряжённых градиентов для минимизации произвольной функции

Рассмотрим теперь задачу оптимизации произвольной гладкой функции f. Тогда в схеме метода сопряжённых градиентов матрица A заменяется на гессиан H_k в текущей точке $\boldsymbol{x}_k$. Практически данный подход превращается в метод Ньютона, в котором квадратичная аппроксимация функции минимизируется с помощью сопряжённых градиентов. Поэтому метод сопряжённых градиентов имеет квадратичную скорость сходимости в малой окрестности оптимального решения.

Для того, чтобы застраховаться от возможной неадекватности квадратичного приближения функции f в текущей точке, в методе сопряжённых градиентов предлагается решать одномерную задачу оптимизации при движении вдоль очередного направления $\boldsymbol{d}_k$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \;\; \alpha_k = \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k).$$

Заметим, что в этом случае отпадает необходимость вычисления гессиана, и метод превращается в метод оптимизации первого порядка. При использовании формулы (11) для β_k соответствующий метод сопряжённых градиентов получил название Флетчера-Ривса (Fletcher-Reeves). В методе Полака-Рибье (Polak-Ribiere) предлагается использовать другую формулу для β_k:

$$\beta_k = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1} - \boldsymbol{g}_{k+1}^T \boldsymbol{g}_k}{\boldsymbol{g}_k^T \boldsymbol{g}_k}.$$

Для случая квадратичной функции последняя формула переходит в формулу (11), т.к. $\boldsymbol{g}_{k+1}^T \boldsymbol{g}_k = 0$. Однако, для произвольной функции она позволяет добиться более устойчивой и качественной работы метода.

В методе сопряжённых градиентов направление $\boldsymbol{d}_{k+1}$ зависит от $\boldsymbol{d}_k$, а, значит, неявно зависит от всех предыдущих направлений $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_k$. Для повышения устойчивости работы метода для неквадратичной функции предлагается устанавливать $\beta_k = 0$ после каждой N-ой итерации, т.е. периодически «очищать» предысторию, в которой могут накапливаться неудачные направления. Обнуление β_k соответствует выбору направления оптимизации по антиградиенту. На рис. 2,а показан пример применения метода сопряжённых градиентов без использования обнуления. В результате в ходе итераций метод «проскочил» оптимальную точку $[1, 1]^T$ за счет

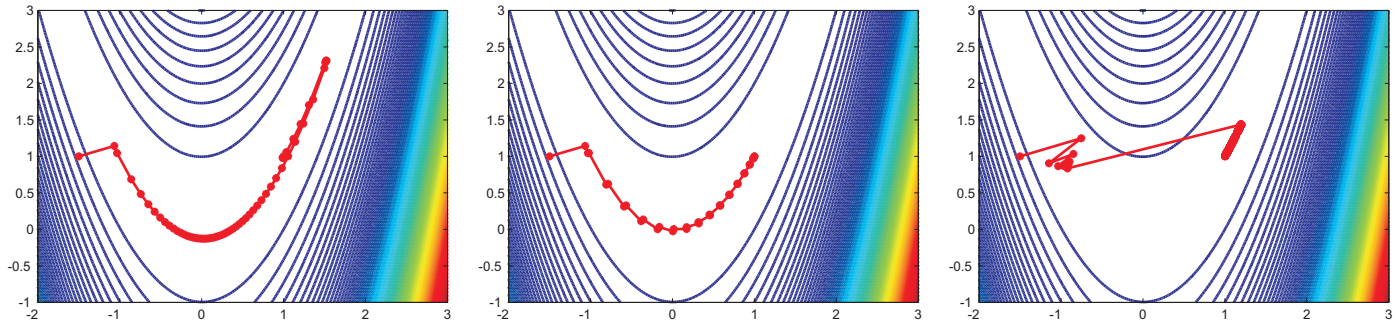


Рис. 2: Примеры работы метода сопряжённых градиентов для функции Розенблока. Случай (a): без использования обнуления β_k, всего 64 итерации, случай (b): с использованием обнуления, всего 36 итераций, случай (c): использование backtracking, всего 1260 итераций.

сильной зависимости от предыдущих направлений. Напротив, использование обнуления (см. рис. 2,b) позволило успешно обнаружить минимум за меньшее число итераций.

Как было отмечено выше, в методе сопряжённых градиентов очередное направление оптимизации выбирается таким образом, чтобы при движении вдоль него сохранить минимум по всем предыдущим направлениям. За счёт этого удаётся избежать ступенчатого поведения, характерного для покоординатного и градиентного спуска. Однако, такой подход неявно предполагает, что вдоль очередного направления проводится точная оптимизация, т.к. в дальнейшем возврат к этому направлению не запланирован. На рис. 2,c показан пример работы метода, в котором на этапе одномерной оптимизации используется backtracking. В результате метод начинает работать крайне неустойчиво, а сходимость достигается только за 1260 итераций. В реальности метод сопряжённых градиентов может устойчиво сходиться и при использовании неточной одномерной оптимизации. Однако, её использование связано с определёнными рисками.

Квази-ньютоновские методы

В квази-ньютоновских методах оптимизации пересчёт осуществляется по формуле

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k S_k \boldsymbol{g}_k. \tag{12}$$

Здесь S_k – некоторая положительно-определённая матрица. Условие положительной определённости S_k гарантирует возможность уменьшения функции f вдоль направления $\boldsymbol{d}_k = -S_k \boldsymbol{g}_k$. Действительно,

$$\frac{\partial}{\partial \alpha} f(\boldsymbol{x}_k - \alpha S_k \boldsymbol{g}_k)\Big|_{\alpha=0} = -\boldsymbol{g}_k^T S_k \boldsymbol{g}_k < 0.$$

Если $S_k = I$, то (12) переходит в градиентный спуск. Если $S_k = H_k^{-1}$, то (12) переходит в метод Ньютона.

По аналогии с методом сопряжённых градиентов, рассмотрим сначала квази-ньютоновские методы для случая минимизации квадратичной функции (1). Введем обозначения: $\boldsymbol{\delta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = -\alpha_k S_k \boldsymbol{g}_k$, $\boldsymbol{\gamma}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$. Для квадратичной функции $\boldsymbol{\gamma}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k = A\boldsymbol{\delta}_k \ \forall k$. В результате

$$[\boldsymbol{\gamma}_0 \ \boldsymbol{\gamma}_1 \ \dots \ \boldsymbol{\gamma}_{N-1}] = A[\boldsymbol{\delta}_0 \ \boldsymbol{\delta}_1 \ \dots \ \boldsymbol{\delta}_{N-1}].$$

Пусть в первый момент времени задано некоторое начальное приближение $\boldsymbol{x}_0$. Положим $S_0 = I$[4] и будем пересчитывать S_k по правилу $S_{k+1} = S_k + C_k$, где C_k – некоторая матрица. Проведем N шагов вида (12) и получим набор $\{\boldsymbol{\delta}_0, \dots, \boldsymbol{\delta}_{N-1}\}$, набор $\{\boldsymbol{\gamma}_0, \dots, \boldsymbol{\gamma}_{N-1}\}$ и набор матриц $S_1, S_2, \dots, S_N$. Если оказывается, что

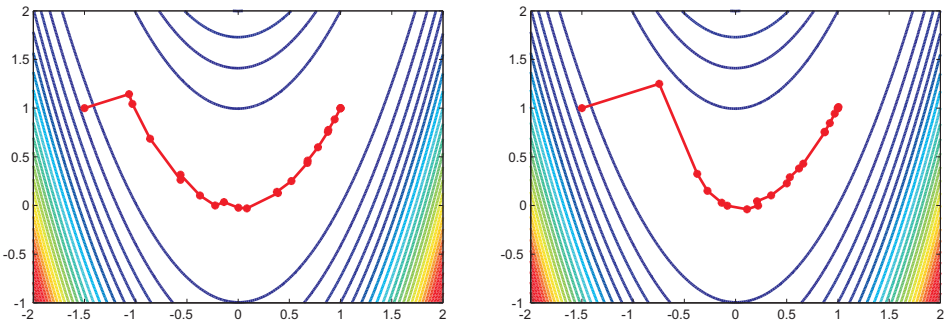
$$S_N[\boldsymbol{\gamma}_0 \ \boldsymbol{\gamma}_1 \ \dots \ \boldsymbol{\gamma}_{N-1}] = [\boldsymbol{\delta}_0 \ \boldsymbol{\delta}_1 \ \dots \ \boldsymbol{\delta}_{N-1}],$$

то матрица $S_N = A^{-1}$ и следующий шаг по схеме (12) соответствует шагу Ньютона, т.е. $\boldsymbol{x}_{N+1} = \boldsymbol{x}_*$.

По аналогии с методом сопряжённых градиентов, в котором существуют разные формулы для β_k и, соответственно, разные итерационные схемы пересчета $\boldsymbol{d}_k$, для квази-ньютоновских методов также предложено несколько способов выбора C_k. Однако, все эти способы гарантируют выполнение следующих свойств:

1. Метод сходится за N шагов для квадратичной функции;

[4] это соответствует использованию направления антиградиента



(a) (b)

Рис. 3: Примеры работы метода L-BFGS для функции Розенблока. Случай (a): точная одномерная оптимизация, случай (b): использование метода Флетчера.

2. Для квадратичной функции набор векторов $\{\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{N-1}\}$ образует сопряжённый базис относительно матрицы A;

3. Матрица S_k всегда остаётся положительно-определённой.

Последнее свойство важно для принципиальной возможности уменьшения f на каждой итерации.

Рассмотрим несколько схем выбора C_k:

- Схема DFP (Davidon-Fletcher-Powell).

$$S_{k+1} = S_k + \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T \boldsymbol{\gamma}_k} - \frac{S_k \boldsymbol{\gamma}_k \boldsymbol{\gamma}_k^T S_k}{\boldsymbol{\gamma}_k^T S_k \boldsymbol{\gamma}_k}.$$

- Схема BFGS (Broyden-Fletcher-Goldfarb-Shanno).

$$S_{k+1} = S_k + \left(1 + \frac{\boldsymbol{\gamma}_k^T S_k \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}\right) \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} - \frac{\boldsymbol{\delta}_k \boldsymbol{\gamma}_k^T S_k + S_k \boldsymbol{\gamma}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}.$$

- Схема L-BGFS (Limited Memory BFGS). Формула пересчета аналогична BFGS, но для $S_k = I$:

$$S_{k+1} = I + \left(1 + \frac{\boldsymbol{\gamma}_k^T \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}\right) \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} - \frac{\boldsymbol{\delta}_k \boldsymbol{\gamma}_k^T + \boldsymbol{\gamma}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}.$$

  Подставляя эту формулу в (12), получаем:

$$\begin{aligned}
\boldsymbol{d}_{k+1} &= -S_{k+1}\boldsymbol{g}_{k+1} = -\boldsymbol{g}_{k+1} + A_k \boldsymbol{\delta}_k + B_k \boldsymbol{\gamma}_k, \\
A_k &= -\frac{\boldsymbol{g}_{k+1}^T \boldsymbol{\delta}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}\left(1 + \frac{\boldsymbol{\gamma}_k^T \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}\right) + \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}, \\
B_k &= -\frac{\boldsymbol{g}_{k+1}^T \boldsymbol{\delta}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}.
\end{aligned}$$

Заметим, что все квази-ньютоновские методы являются методами первого порядка. В случае квадратичной функции направления $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{N-1}$ являются сопряжёнными относительно A. Поэтому в этом случае все квази-ньютоновские методы эквивалентны методу сопряжённых градиентов. При этом существенным моментом является использование направления антиградиента в первый момент времени ($S_0 = I$). При использовании другой матрицы S_0 генерируемые направления оптимизации $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{N-1}$ теряют свойства сопряжённости, а сам метод – свойство гарантированной сходимости за N итераций.

На рис. 3,a показан пример работы метода L-BFGS для функции Розенблока. Заметим, что в отличие от метода сопряжённых градиентов, в квази-ньютоновских методах нет необходимости «очищать» предысторию и периодически приравнивать $S_k = I$. Кроме того, квази-ньютоновские методы являются более толерантными к использованию неточной одномерной оптимизации (см. рис. 3,b).