

# Метод сопряжённых градиентов

МЕТОДЫ ВЫПУКЛОЙ ОПТИМИЗАЦИИ

НЕДЕЛЯ 9

Даня Меркулов



# **Метод сопряженных градиентов**

## **Семинар**

**Оптимизация для всех! ЦУ**



# Повторение лекции

# Сильно выпуклые квадратичные функции

Рассмотрим следующую квадратичную задачу оптимизации:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ где } A \in \mathbb{S}_{++}^d.$$

Условия оптимальности:

$$\nabla f(x^*) = Ax^* - b = 0 \iff Ax^* = b$$

# СИЛЬНО ВЫПУКЛЫЕ КВАДРАТИЧНЫЕ ФУНКЦИИ

Рассмотрим следующую квадратичную задачу оптимизации:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ где } A \in \mathbb{S}_{++}^d.$$

Условия оптимальности:

$$\nabla f(x^*) = Ax^* - b = 0 \iff Ax^* = b$$

Steepest Descent



Conjugate Gradient



# Обзор метода сопряжённых градиентов для квадратичной задачи

1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .

# Обзор метода сопряжённых градиентов для квадратичной задачи

1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_0 = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

- 3) **Итерация алгоритма.** Обновляем положение  $x_k$ , двигаясь в направлении  $d_k$  с длиной шага  $\alpha_k$ :

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

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_0 = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

- 3) **Итерация алгоритма.** Обновляем положение  $x_k$ , двигаясь в направлении  $d_k$  с длиной шага  $\alpha_k$ :

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

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

- 3) **Итерация алгоритма.** Обновляем положение  $x_k$ , двигаясь в направлении  $d_k$  с длиной шага  $\alpha_k$ :

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

- 4) **Обновление направления.** Обновляем  $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$ , где  $\beta_k$  вычисляется по формуле:

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k}.$$

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

- 3) **Итерация алгоритма.** Обновляем положение  $x_k$ , двигаясь в направлении  $d_k$  с длиной шага  $\alpha_k$ :

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

- 4) **Обновление направления.** Обновляем  $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$ , где  $\beta_k$  вычисляется по формуле:

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k}.$$

# Обзор метода сопряжённых градиентов для квадратичной задачи

- 1) **Инициализация.**  $k = 0$  и  $x_k = x_0$ ,  $d_k = d_0 = -\nabla f(x_0)$ .
- 2) **Оптимальная длина шага.** С помощью процедуры одномерного поиска (*line search*) находим оптимальную длину шага. Это означает вычислить  $\alpha_k$ , минимизирующий  $f(x_k + \alpha_k d_k)$ :

$$\alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top Ad_k}$$

- 3) **Итерация алгоритма.** Обновляем положение  $x_k$ , двигаясь в направлении  $d_k$  с длиной шага  $\alpha_k$ :

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

- 4) **Обновление направления.** Обновляем  $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$ , где  $\beta_k$  вычисляется по формуле:

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k}.$$

- 5) **Цикл до сходимости.** Повторяем шаги 2–4, пока не построено  $n$  направлений, где  $n$  — размерность пространства (размерность  $x$ ).

# Оптимальная длина шага

Точный одномерный поиск:

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k + \alpha d_k)$$

# Оптимальная длина шага

Точный одномерный поиск:

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k + \alpha d_k)$$

Найдём аналитическое выражение для шага  $\alpha_k$ :

$$\begin{aligned} f(x_k + \alpha d_k) &= \frac{1}{2} (x_k + \alpha d_k)^\top A (x_k + \alpha d_k) - b^\top (x_k + \alpha d_k) + c \\ &= \frac{1}{2} \alpha^2 d_k^\top A d_k + d_k^\top (A x_k - b) \alpha + \left( \frac{1}{2} x_k^\top A x_k + x_k^\top d_k + c \right) \end{aligned}$$

# Оптимальная длина шага

Точный одномерный поиск:

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k + \alpha d_k)$$

Найдём аналитическое выражение для шага  $\alpha_k$ :

$$\begin{aligned} f(x_k + \alpha d_k) &= \frac{1}{2} (x_k + \alpha d_k)^\top A (x_k + \alpha d_k) - b^\top (x_k + \alpha d_k) + c \\ &= \frac{1}{2} \alpha^2 d_k^\top A d_k + d_k^\top (Ax_k - b) \alpha + \left( \frac{1}{2} x_k^\top A x_k + x_k^\top d_k + c \right) \end{aligned}$$

Поскольку  $A \in \mathbb{S}_{++}^d$ , точка с нулевой производной на этой параболе является минимумом:

$$(d_k^\top A d_k) \alpha_k + d_k^\top (Ax_k - b) = 0 \iff \alpha_k = -\frac{d_k^\top (Ax_k - b)}{d_k^\top A d_k}$$

# Обновление направления

Обновляем направление так, чтобы следующее направление было  $A$ -ортогонально предыдущему:

$$d_{k+1} \perp_A d_k \iff d_{k+1}^\top A d_k = 0$$

# Обновление направления

Обновляем направление так, чтобы следующее направление было  $A$ -ортогонально предыдущему:

$$d_{k+1} \perp_A d_k \iff d_{k+1}^\top A d_k = 0$$

Поскольку  $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$ , выбираем  $\beta_k$  так, чтобы выполнялась  $A$ -ортогональность:

$$d_{k+1}^\top A d_k = -\nabla f(x_{k+1})^\top A d_k + \beta_k d_k^\top A d_k = 0 \iff \beta_k = \frac{\nabla f(x_{k+1})^\top A d_k}{d_k^\top A d_k}$$

# Обновление направления

Обновляем направление так, чтобы следующее направление было  $A$ -ортогонально предыдущему:

$$d_{k+1} \perp_A d_k \iff d_{k+1}^\top A d_k = 0$$

Поскольку  $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$ , выбираем  $\beta_k$  так, чтобы выполнялась  $A$ -ортогональность:

$$d_{k+1}^\top A d_k = -\nabla f(x_{k+1})^\top A d_k + \beta_k d_k^\top A d_k = 0 \iff \beta_k = \frac{\nabla f(x_{k+1})^\top A d_k}{d_k^\top A d_k}$$

## 💡 Лемма 1

Все направления, строящиеся по описанной выше процедуре,  $A$ -ортогональны друг другу:

$$d_i^\top A d_j = 0, \text{ if } i \neq j$$

$$d_i^\top A d_j > 0, \text{ if } i = j$$

# $A$ -ортогональность



$v_1$  and  $v_2$  are orthogonal

$$v_1^T v_2 = 0.00$$

$$v_1^T A v_2 = 1.19$$



$\hat{v}_1$  and  $\hat{v}_2$  are  $A$ -orthogonal

$$\hat{v}_1^T \hat{v}_2 = -0.80$$

$$\hat{v}_1^T A \hat{v}_2 = -0.00$$



# Сходимость метода сопряжённых градиентов

## Лемма 2

Пусть решается  $n$ -мерная квадратичная выпуклая задача оптимизации. Метод сопряжённых направлений:

$$x_{k+1} = x_0 + \sum_{i=0}^k \alpha_i d_i,$$

где  $\alpha_i = -\frac{d_i^\top (Ax_i - b)}{d_i^\top Ad_i}$ , взятые из одномерного поиска, обеспечивают сходимость не более чем за  $n$  шагов алгоритма.

# Метод сопряжённых градиентов на практике

На практике для шага  $\alpha_k$  и коэффициента  $\beta_k$  обычно используют следующие формулы:

$$\alpha_k = \frac{r_k^\top r_k}{d_k^\top A d_k} \quad \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k},$$

где  $r_k = b - Ax_k$ , так как  $x_{k+1} = x_k + \alpha_k d_k$ , то  $r_{k+1} = r_k - \alpha_k Ad_k$ . Также,  $r_i^\top r_k = 0, \forall i \neq k$  (**Лемма 5** из лекции).

# Метод сопряжённых градиентов на практике

На практике для шага  $\alpha_k$  и коэффициента  $\beta_k$  обычно используют следующие формулы:

$$\alpha_k = \frac{r_k^\top r_k}{d_k^\top A d_k} \quad \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k},$$

где  $r_k = b - Ax_k$ , так как  $x_{k+1} = x_k + \alpha_k d_k$ , то  $r_{k+1} = r_k - \alpha_k Ad_k$ . Также,  $r_i^\top r_k = 0, \forall i \neq k$  (**Лемма 5** из лекции).

Выведем выражение для  $\beta_k$ :

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k} = -\frac{r_{k+1}^\top Ad_k}{d_k^\top Ad_k}$$

# Метод сопряжённых градиентов на практике

На практике для шага  $\alpha_k$  и коэффициента  $\beta_k$  обычно используют следующие формулы:

$$\alpha_k = \frac{r_k^\top r_k}{d_k^\top A d_k} \quad \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k},$$

где  $r_k = b - Ax_k$ , так как  $x_{k+1} = x_k + \alpha_k d_k$ , то  $r_{k+1} = r_k - \alpha_k Ad_k$ . Также,  $r_i^\top r_k = 0, \forall i \neq k$  (**Лемма 5** из лекции).

Выведем выражение для  $\beta_k$ :

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k} = -\frac{r_{k+1}^\top Ad_k}{d_k^\top Ad_k}$$

Числитель:  $r_{k+1}^\top Ad_k = \frac{1}{\alpha_k} r_{k+1}^\top (r_k - r_{k+1}) = [r_{k+1}^\top r_k = 0] = -\frac{1}{\alpha_k} r_{k+1}^\top r_{k+1}$

Знаменатель:  $d_k^\top Ad_k = (r_k + \beta_{k-1} d_{k-1})^\top Ad_k = \frac{1}{\alpha_k} r_k^\top (r_k - r_{k+1}) = \frac{1}{\alpha_k} r_k^\top r_k$

# Метод сопряжённых градиентов на практике

На практике для шага  $\alpha_k$  и коэффициента  $\beta_k$  обычно используют следующие формулы:

$$\alpha_k = \frac{r_k^\top r_k}{d_k^\top A d_k} \quad \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k},$$

где  $r_k = b - Ax_k$ , так как  $x_{k+1} = x_k + \alpha_k d_k$ , то  $r_{k+1} = r_k - \alpha_k Ad_k$ . Также,  $r_i^\top r_k = 0, \forall i \neq k$  (**Лемма 5** из лекции).

Выведем выражение для  $\beta_k$ :

$$\beta_k = \frac{\nabla f(x_{k+1})^\top Ad_k}{d_k^\top Ad_k} = -\frac{r_{k+1}^\top Ad_k}{d_k^\top Ad_k}$$

Числитель:  $r_{k+1}^\top Ad_k = \frac{1}{\alpha_k} r_{k+1}^\top (r_k - r_{k+1}) = [r_{k+1}^\top r_k = 0] = -\frac{1}{\alpha_k} r_{k+1}^\top r_{k+1}$

Знаменатель:  $d_k^\top Ad_k = (r_k + \beta_{k-1} d_{k-1})^\top Ad_k = \frac{1}{\alpha_k} r_k^\top (r_k - r_{k+1}) = \frac{1}{\alpha_k} r_k^\top r_k$

## Question

Почему эта модификация лучше стандартной версии?

# Метод сопряжённых градиентов на практике.

## Псевдокод

$$r_0 := b - Ax_0$$

if  $r_0$  is sufficiently small, then return  $x_0$  as the result

$$d_0 := r_0$$

$$k := 0$$

repeat

$$\alpha_k := \frac{r_k^\top r_k}{d_k^\top A d_k}$$

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

$$r_{k+1} := r_k - \alpha_k A d_k$$

if  $r_{k+1}$  is sufficiently small, then exit loop

$$\beta_k := \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$$

$$d_{k+1} := r_{k+1} + \beta_k d_k$$

$$k := k + 1$$

end repeat

return  $x_{k+1}$  as the result

# Упражнение 1

Реализуйте итерации метода сопряжённых градиентов для квадратичной задачи

$$f(x) = \frac{1}{2}x^T Ax - b^T x \longrightarrow \min_{x \in \mathbb{R}^n}$$

и запустите эксперименты для нескольких матриц  $A$ . Смотрите код здесь [于此](#).

# Нелинейный метод сопряжённых градиентов

Если у нас нет аналитического выражения для функции или её градиента, мы, скорее всего, не сможем аналитически решить одномерную задачу минимизации. Поэтому  $\alpha_k$  подбирается обычной процедурой одномерного поиска. Но для выбора  $\beta_k$  есть следующий математический трюк:

Для двух последовательных итераций верно:

$$x_{k+1} - x_k = cd_k,$$

где  $c$  — некоторая константа. Тогда для квадратичного случая имеем:

$$\nabla f(x_{k+1}) - \nabla f(x_k) = (Ax_{k+1} - b) - (Ax_k - b) = A(x_{k+1} - x_k) = cAd_k$$

Выражая из этого равенства  $Ad_k = \frac{1}{c} (\nabla f(x_{k+1}) - \nabla f(x_k))$ , избавляемся от «знания» функции в определении шага  $\beta_k$ , тогда пункт 4 переписывается так:

$$\beta_k = \frac{\nabla f(x_{k+1})^\top (\nabla f(x_{k+1}) - \nabla f(x_k))}{d_k^\top (\nabla f(x_{k+1}) - \nabla f(x_k))}.$$

Этот метод называется методом Полака—Рибьера.

## Упражнение 2

Реализуйте итерации метода Полака—Рибьера и запустите эксперименты для нескольких  $\mu$  в бинарной логистической регрессии:

$$f(x) = \frac{\mu}{2} \|x\|_2^2 + \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i \langle a_i, x \rangle)) \longrightarrow \min_{x \in \mathbb{R}^n}$$

Смотрите код здесь .

# Численные эксперименты

# Патологический пример

Пусть  $t \in (0, 1)$  и

$$W = \begin{bmatrix} t & \sqrt{t} & & \\ \sqrt{t} & 1+t & \sqrt{t} & \\ & \sqrt{t} & 1+t & \sqrt{t} \\ & & \ddots & \ddots & \ddots \\ & & & \sqrt{t} & 1+t \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Так как  $W$  невырождена, существует единственное решение  $Wx = b$ . Решение методом сопряжённых градиентов даёт довольно плохую сходимость. Во время работы CG ошибка растёт экспоненциально (!), пока внезапно не становится нулевой, когда находится единственное решение. Невязка  $\|Wx_k - b\|^2$  растёт экспоненциально как  $(1/t)^k$  до  $n$ -й итерации, после чего резко падает к нулю. См. эксперимент здесь [🔗](#). ## Другие численные эксперименты Посмотрим другие примеры здесь [🔗](#). Код взят из [🔗](#).