# ПРЯМОЙ МЕТОД ПОСТРОЕНИЯ МИНИМАЛЬНОГО ЭЛЛИПСОИДА*

М. А. Кольцов
kolmax94@gmail.com

26 ноября 2015 г.

**1°. Постановка задачи.** В докладе [1] рассматривалась задача построения минимального эллипсоида. Напомним её постановку.

Дано множество точек $c_j \in \mathbb{R}^n$, $j \in 1 : m$. Требуется построить эллипсоид $\mathcal{E} \subset \mathbb{R}^n$ минимального объёма, который содержит все точки $c_j$. Эллипсоид $\mathcal{E}$ задаётся в виде

$$\mathcal{E} = \{x \in \mathbb{R}^n \mid \langle M(x-c), x-c\rangle \leqslant 1\} \ ,$$

где $c \in \mathbb{R}^n$ — его центр, а $M$ — симметричная положительно определённая матрица порядка $n$.

Задача формализуется так:

$$\begin{aligned} &F(M) := -\log\det M \longrightarrow \min \\ &\langle M(c_j - c), c_j - c\rangle \leqslant 1, \ j \in 1:m \\ &M = M^\intercal \\ &M \succ 0, \end{aligned} \tag{1}$$

где $M \succ 0$ означает, что матрица $M$ положительно определена.

В предыдущем докладе приведён алгоритм решения задачи, предложенный Н.З. Шором, в котором строятся приближения к ответу, основываясь на некоторых геометрических идеях. Теперь же рассмотрим подход, основанный на непосредственном решении задачи (1).

**2°. Представление в виде задачи полуопределённого программирования.** В статье [2] дан обзор по экстремальным эллипсоидам, в котором задача (1) сформулирована как задача полуопределённого программирования: переменной выступает симметричная полуопределённая (неотрицательно определённая) матрица, а в ограничениях требуется полуопределённость некоторых матриц (так называемые линейные матричные неравенства).

Для того чтобы свести задачу (1) к такому виду, докажем вспомогательную лемму (см. [5]).

**ЛЕММА 1** (Шур). *Пусть $M$ — симметричная блочная матрица вида*

$$M = \begin{pmatrix} A & B \\ B^\intercal & C \end{pmatrix},$$

*где $A$ положительно определена. Тогда $M$ неотрицательно определена тогда и только тогда, когда неотрицательно определена матрица $S := C - B^\intercal A^{-1} B$.*

*Семинар по конструктивному негладкому анализу и недифференцируемой оптимизации «CNSA & NDO»: `http://www.apmath.spbu.ru/cnsa/`

Доказательство. Рассмотрим вектор $x$ подходящей размерности и разобьём его на блоки $x = \begin{pmatrix} u \\ v \end{pmatrix}$ в соответствии с блочной структурой $M$. Распишем произведение $\langle Mx, x\rangle$:

$$\langle Mx, x\rangle = x^\intercal Mx = \begin{pmatrix} u^\intercal & v^\intercal \end{pmatrix} \left[ \begin{pmatrix} A & B \\ B^\intercal & C \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \right] = \begin{pmatrix} u^\intercal & v^\intercal \end{pmatrix} \begin{pmatrix} Au + Bv \\ B^\intercal u + Cv \end{pmatrix} =$$
$$= u^\intercal Au + u^\intercal Bv + v^\intercal B^\intercal u + v^\intercal Cv =$$
$$= \langle Au, u\rangle + \langle u, Bv\rangle + \langle v, B^\intercal u\rangle + \langle Cv, v\rangle =$$
$$= \langle Au, u\rangle + 2\langle u, Bv\rangle + \langle Cv, v\rangle$$

Теперь можно доказать саму лемму.
Достаточность. Пусть $S$ неотрицательно определена. Зафиксируем произвольное $v$ и рассмотрим квадратичную функцию

$$Q_v(u) = \frac{1}{2}\langle 2Au, u\rangle + \langle u, 2Bv\rangle + \langle Cv, v\rangle$$

Эта функция строго выпукла, а значит она достигает глобального минимума при $Q_v^{'}(u^*) = = 2Au + 2Bv = 0$, то есть в точке $u^* = -A^{-1}Bv$. Вычислим её значение в ней:

$$Q_v(u^*) = \left\langle AA^{-1}Bv, A^{-1}Bv\right\rangle - 2\left\langle A^{-1}Bv, Bv\right\rangle + \langle Cv, v\rangle =$$
$$= \langle Cv, v\rangle - \left\langle B^\intercal A^{-1}Bv, v\right\rangle =$$
$$= \left\langle (C - B^\intercal A^{-1}B)v, v\right\rangle = \langle Sv, v\rangle$$

Но по условию леммы $\langle Sv, v\rangle \geqslant 0$, а значит

$$\langle Mx, x\rangle = Q_v(u) \geqslant Q_v(u^*) = \langle Sv, v\rangle \geqslant 0,$$

что и требовалось доказать.
Необходимость. Теперь предположим, что $M$ неотрицательно определена. Рассмотрим произвольный вектор $v$ и обозначим $u^* := -A^{-1}Bv$, $x := \begin{pmatrix} u^* \\ v \end{pmatrix}$. Тогда

$$0 \leqslant \langle Mx, x\rangle = \langle Sv, v\rangle$$

Лемма доказана. □

В дальнейшем положительную определённость матрицы $A$ будем записывать $A \succ 0$, а полуопределённость — $A \succeq 0$.

Вернёмся теперь к задаче (1). Так как $M \succ 0$, то существует симметричная положительно определённая матрица $M^{1/2}$ такая, что $M = M^{1/2}M^{1/2}$. Перепишем первое ограничение задачи:

$$\langle M(c_j - c), c_j - c\rangle \leqslant 1$$
$$1 - \left\langle M^{1/2}(c_j - c), M^{1/2}(c_j - c)\right\rangle \geqslant 0$$
$$1 - \langle Qc_j - b, Qc_j - b\rangle \geqslant 0,$$

где $Q = M^{1/2}$, $b = M^{1/2}c$. Теперь последнее неравенство с помощью леммы Шура можно записать в виде линейного матричного неравенства:

$$\begin{pmatrix} E & Qc_j - b \\ (Qc_j - b)^\intercal & 1 \end{pmatrix} \succeq 0$$

В этом случае $A = E$ — положительно определённая матрица и

$$S = 1 - (Qc_j - b)^\intercal E^{-1}(Qc_j - b) = 1 - \langle Qc_j - b, Qc_j - b\rangle$$

Так как $S$ — число, то $S \succeq 0 \Leftrightarrow S \geqslant 0$, что и есть требуемое условие.

В итоге получаем задачу полуопределённого программирования

$$\begin{aligned}
&f(Q) := -\log\det Q \longrightarrow \min \\
&Q = Q^\intercal \\
&Q \succ 0 \\
&\begin{pmatrix} E & Qc_j - b \\ (Qc_j - b)^\intercal & 1 \end{pmatrix} \succeq 0\,,\ j \in 1:m
\end{aligned} \tag{2}$$

После того, как матрица $Q$ и вектор $b$ найдены, требуется лишь вычислить искомую матрицу $M := Q^2$ и центр эллипсоида $c := Q^{-1}b$.

**ЛЕММА 2.** *Рассмотрим функцию*

$$f(Q) = -\ln\det Q$$

*где* $Q = \{q_{ij}\}_{i,j=1}^n$ *— положительно определённая матрица. Тогда*

$$f'(Q) = \left\{\frac{\partial f}{\partial q_{ij}}\right\}_{i,j=1}^n = -Q^{-1}$$

Доказательство. Во-первых,

$$\frac{\partial f}{\partial q_{ij}} = -\frac{1}{\det Q}\left[\frac{\partial}{\partial q_{ij}}\det Q\right]$$

Сосредоточимся на частной производной определителя. Запишем его разложение по $i$-й строке:

$$\det Q = \sum_{k=1}^n q_{ik}\Delta_{ik}$$

где $\Delta_{ik}$ — алгебраическое дополнение элемента $q_{ik}$, то есть определитель матрицы, полученной из $Q$ удалением $i$-й строки и $j$-го столбца, умноженный на $(-1)^{i+j}$.

Теперь очевидно, что

$$\frac{\partial}{\partial q_{ij}}\det Q = \Delta_{ik}$$

Таким образом, матрица производных $\det Q$ составлена из алгебраических дополнений соответствующих элементов матрицы $Q$. Обозначим её через $C$. Получили

$$f'(Q) = -\frac{1}{\det Q}C$$

Из линейной алгебры известно, что $\dfrac{1}{\det Q}C^\intercal = Q^{-1}$, следовательно, с учётом симметричности $Q$

$$f'(Q) = -\left(\frac{1}{\det Q}C^\intercal\right)^\intercal = -Q^{-1}$$

$\square$

**3°. Реализация метода и примеры работы.** Данный метод был реализован с помощью пакета CVX для MATLAB ([3]), который предназначен для решения различных задач выпуклого программирования. Стоит остановиться на этом пакете подробнее.

Пакет CVX позволяет решать широкий круг оптимизационных задач, в которых целевые функции, ограничения-равенства и ограничения-неравенства представлены выпуклыми функциями от векторов и матриц. При использовании этого пакета появляется возможность ставить задачу, конструируя выпуклые функции и выпуклые множества с помощью привычных алгебраических выражений MATLAB. Такая простота достигается за счёт того, что CVX определяет жёсткий набор правил, по которым из известных выпуклых функций можно получать новые с сохранением выпуклости. Авторы пакета называют эти правила «дисциплинированное выпуклое программирование» (DCP).

Отметим также ограничение пакета CVX: для матрицы $Q$ может быть потребована только полуопределённость, а не положительная определённость, поэтому на самом деле решается расширенная задача (2), в которой второе ограничение выглядит как $Q \succeq 0$.

Для начала запустим алгоритм для 100 случайных точек, лежащих внутри эллипсоида с центром точке $c = [1, 2]$ и полуосями длиной 1 и 2 (см. рис. 1). Такие точки были получены из случайных точек внутри единичного шара умножением на матрицу $A = M^{-1/2}$, где $M$ — матрица эллипсоида. Для нахождения ответа MATLAB делает 16 шагов, после которых объёмы истинного и построенного эллипсоидов совпадают с точностью до 9 знака после запятой. Решение задачи занимает около 5 секунд, а построенный эллипсоид невозможно отличить на рисунке от истинного.

Для случая $n = 5$ и $n = 10$ CVX делает 17 шагов, достигая теперь точности уже до седьмого и второго знака после запятой соответственно, с временем работы около 4 секунд.

Если же подать на вход заведомо вырожденный эллипсоид (точки на одной прямой на плоскости или в одной плоскости в пространстве), CVX вообще не может решить задачу и завершает работу с ошибкой:

```
stop: progress is too slow
...
termination code       = -5
...
Status: Failed
Optimal value (cvx_optval): NaN
```

Документация CVX приводит такое объяснения кода завершения $-5$: «Schur complement matrix becomes too ill-conditioned for further progress».

Кроме того, документация несколько размыто описывает параметры точности. В частности, говорится, что каждый реализованный там алгоритм сам оценивает погрешность своего решения и останавливается, когда она становится ниже заданного $\varepsilon$, по умолчанию равного $(2.22 \times 10^{-16})^{1/2}$. Также с помощью команды «`cvx_precision`» можно задавать точность вычислений на одном из предустановленных уровней или по желанию пользователя. Например, на уровне «`high`» CVX достигает точности до пятого знака после запятой при $n = 10$ за время 5 секунд.

Документация же пакета SDPT3, который CVX вызывает для решения задачи полуопределённого программирования, говорит, что вычисления останавливаются, когда и относительный зазор двойственности (duality gap), и мера несовместности (infeasibility measure) становятся меньше $\varepsilon$.

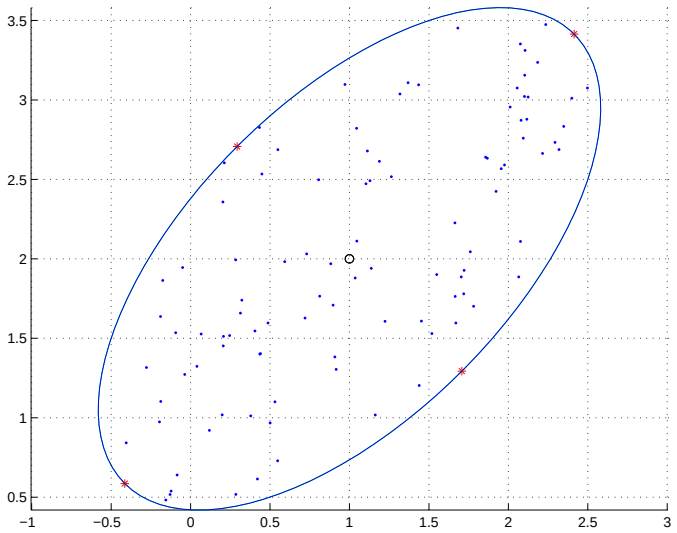


Рис. 1. Тестовое множество точек. Синим отмечен построенный эллипсоид, который не отличим от истинного минимального эллипсоида

**4°. Сравнение с алгоритмом Шора.** Для того, чтобы провести сравнение, реализация алгоритма Шора была улучшена в соответствии с оригинальной статьей [4] — был добавлен контроль сходимости и условие останова. Теперь алгоритм выглядит так:

1) Перевести все точки в пространство на единицу большей размерности
2) Выполнять обычные шаги алгоритма Шора с постоянным коэффициентом сжатия $\alpha$
3) Пока максимальные нормы ($\tau_k$) векторов меняются сильнее, чем на $\varepsilon_1$, повторять шаг 2, иначе перейти на шаг 4
4) Выполнять шаги алгоритма Шора с возрастающими коэффициентами сжатия $\alpha_k$
5) Пока $\tau_k$ меняются сильнее, чем на $\varepsilon_2$, повторять шаг 4, иначе закончить вычисления

Отметим, что параметры $\varepsilon_1$ и $\varepsilon_2$ не являются независимыми и их требуется подбирать для конкретной задачи, при этом должно выполняться неравенство $\varepsilon_2 < \varepsilon_1$.

Для численных экспериментов было взято то же множество точек с рис. 1 при $\alpha = 1 - 0.2$, $\alpha_k = 1 - \frac{1}{k}$, $\varepsilon_1 = 10^{-5}$, $\varepsilon_2 = 10^{-9}$. С такими параметрами удалось достичь точности объёма эллипсоида до третьего знака после запятой за время 0.4 секунды, при этом было сделано 3914 шага алгоритма Шора, результат визуально неотличим от рис. 1. Наверняка, с помощью подбора параметров можно добиться и более точного результата. Приведём также результаты, получающиеся при другом задании параметров:

- $\varepsilon_1 = 10^{-5}$, $\varepsilon_2 = 10^{-10}$: точность до третьего знака, 21990 итераций, время 2.4 секунды
- $\varepsilon_1 = 10^{-4}$, $\varepsilon_2 = 10^{-9}$: точность до третьего знака, 20591 итераций, время 2.3 секунды
- $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-10}$: точность до третьего знака, 4121 итераций, время всего 0.4 секунды
- $\varepsilon_1 = 10^{-4}$, $\varepsilon_2 = 10^{-10}$: сходимости нет, сделано 100000 итераций, точность текущего приближения до четвертого знака, время 21.2 секунды

Для $n = 5$, $\varepsilon_1 = 10^{-5}$, $\varepsilon_2 = 10^{-10}$ результаты такие: точность до второго знака, число итераций 91518 и время работы 16.3 секунды. Для $n = 10$ не удалось получить решение, разница объемов эллипсоидов достигает нескольких десятков тысяч.

Таким образом, можно сделать вывод, что алгоритм Шора в текущей реализации проигрывает прямому методу решения задачи.

## ЛИТЕРАТУРА


1. Кольцов М. А. *Построение минимального эллипсоида* // Семинар «CNSA & NDO». Избранные доклады. 14 мая 2015 г. (`http://www.apmath.spbu.ru/cnsa/pdf/2015/ellipsoid.pdf`)

2. Бедринцев А. А., Чепыжов В. В. *Представление данных с помощью экстремальных эллипсоидов* // Материалы конференции «Информационные технологии и системы - 2013» 37-я конференция-школа молодых ученых и специалистов 1 — 6 сентября, Калининград, Россия. С. 55-60.

3. CVX: Matlab Software for Disciplined Convex Programming, `http://cvxr.com/cvx`

4. Шор Н. З., Стеценко С. И. *Алгоритм последовательного сжатия пространства для построения описанного эллипсоида минимального объёма* // Исследование методов решения экстремальных задач, 1990.

5. Boyd S., Vandenberghe L. *Convex optimization.* Cambridge University Press, 2004.