МЕТОД ЭЛЛИПСОИДОВ ДЛЯ НАХОЖДЕНИЯ L_p-РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

Стецюк П.И., Била Г.Д., Стовба В.А.
stetsyukp@gmail.com

Институт кибернетики имени В.М. Глушкова, Киев

VIII ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
"ІНФОРМАТИКА ТА СИСТЕМНІ НАУКИ"
16–18 березня 2016 року, м. Полтава

Содержание

Содержание

Метод эллипсоидов предложили

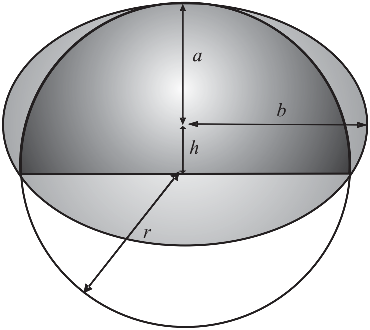
1976 **Юдин Д.Б. и Немировский А.С.** как метод последовательных отсечений [1].

1977 **Шор Н.З.** как вариант метода с растяжением пространства в направлении субградиента [2].

1. Юдин Д.Б., Немировский А.С. *Информационная сложность и эффективные методы решения выпуклых экстремальных задач* // Экономика и математические методы. – 1976. – Вып. 2. – С. 357–369.

2. Шор Н.З. *Метод отсечения с растяжением пространства для решения задач выпуклого программирования* // Кибернетика. – 1977. – № 1. – С. 94–95.

Идея: минимальный по объему эллипсоид $\mathcal{E}_n$



Эллипсоид $\mathcal{E}_n$, содержащий полушар в E^n, имеет параметры

$$a = \frac{n}{n+1},\ b = \frac{n}{\sqrt{n^2-1}}r,\ h = \frac{1}{n+1}r,$$

где $\alpha = \frac{b}{a}$ и r – радиус шара S_n.

Если пространство „растянуть“ в направлении a с коэффициентом $\alpha = \sqrt{\frac{n+1}{n-1}}$, то $\mathcal{E}_n$ станет шаром в преобразованном пространстве.

Отношение объема эллипсоида $\mathcal{E}_n$ к объему шара S_n равно

$$q(n) = \frac{\mathrm{vol}(\mathcal{E}_n)}{\mathrm{vol}(S_n)} = \frac{1}{\alpha}\left(\frac{b}{r}\right)^n = \sqrt{\frac{n-1}{n+1}}\left(\frac{n}{\sqrt{n^2-1}}\right)^n < e^{-1/2n} < 1.$$

Содержание

В чем состоит задача?

Пусть имеется система линейных уравнений:

$$Ax \approx b, \tag{1.a}$$

при условиях

$$l \le x \le u. \tag{1.b}$$

Здесь A – $m \times n$-матрица; $b \in R^m$ – m-мерный вектор;
$l \in R^n$, $u \in R^n$ – n-мерные векторы, такие, что $l < u$;
$x \in R^n$ – n-мерный вектор неизвестных.

Требуется найти такой вектор неизвестных x_p^*, который "наилучшим образом" выполняет соотношение (1.a).

Что означает "наилучшим образом"?

x_p^* – "наилучшее" решение системы (1) в L_p-норме, т. е. когда норма вектора невязок $y = Ax - b = (y_1, \ldots, y_m)^T$ определена следующим образом:

$$\|y\|_p = \left(\sum_{i=1}^{m} |y_i|^p \right)^{1/p}, \quad \text{где} \quad p \geq 1.$$

Три частных случая:

Случай $p = \infty$ определяется как $\|y\|_\infty = \max_{i=1,\ldots,m} |y_i|$.

$p = 2$ соответствует $\|y\|$ – стандартной евклидовой норме.

$p = 1$ соответствует "манхеттеновской" норме $\|y\|_1 = \sum_{i=1}^{m} |y_i|$.

"Наилучшее" решение системы (1)

соответствует задаче выпуклого программирования:

найти

$$f_p^* = f_p(x_p^*) = \min_{x \in R^n} \left\{ f_p(x) = \|Ax - b\|_p \right\} \quad (2)$$

при ограничениях

$$l \le x \le u, \quad (3)$$

где $p \in R$ – скалярный параметр, такой что $p \ge 1$.

Здесь x_p^* – решение задачи (2)–(3).

Для удобства будем считать, что оно единственное.

О приложениях задачи (2)–(3)

Подобные задачи часто встречаются в самых разных областях прикладной математики, например: при обработке результатов наблюдений, построении и анализе различного рода моделей (физических, биологических, экономических, социальных и других), при поиске компромиссных решений в моделях с противоречивыми данными, и т.д.

Для линейной регрессии задача (2)–(3) соответствует

методу наименьших квадратов ($p = 2$),
методу наименьших модулей ($p = 1$),
минимаксному (чебышевскому) методу ($p = \infty$).

Содержание

Для решения задачи (2)–(3)

методом эллипсоидов требуется:

1. определить градиентное поле $g_p(x)$ (способ построения в точке $x \in R^n$ такой гиперплоскости, которая локализует точку x_p^* в одном из полупространств пространства R^n);
2. выбрать начальный радиус шара, в котором локализовано оптимальное решение x_p^*.

Удовлетворить эти требования можно с помощью

[3]. Стецюк П.И., Колесник Ю.С., Березовский О.А. *Об одном методе нахождения L_p-решения системы линейных уравнений* // Теория оптимальных решений. – Киев: Ин-т кибернетики им. В.М.Глушкова НАН Украины, 2003. – С. 83–90.

Градиентное поле $g_p(x)$

Лемма 1 (Стецюк, Колесник, Березовский, 2003)

Пусть $\partial f_p(x)$ – субградиент функции $f_p(x)$ в точке x;

$t^* = \max\{t_{i^*}, t_{j^*}\}$, $t_{i^*} = \max\limits_{i=1,\dots,n} \{x_i - u_i\}$, $t_{j^*} = \max\limits_{j=1,\dots,n} \{l_j - x_j\}$;

i^*, j^* – значения i, j $(1 \le i, j \le n)$, на которых достигаются t_{i^*}, t_{j^*};

e_k – k-й орт в R^n, $1 \le k \le n$. Тогда, вектор

$$g_p(x) = \begin{cases} \partial f_p(x), & \text{если} \quad t^* \le 0, \\ e_{i^*}, & \text{если} \quad t^* > 0 \quad \text{и} \quad t^* > t_{i^*}, \\ -e_{j^*}, & \text{если} \quad t^* > 0 \quad \text{и} \quad t^* \le t_{j^*}, \end{cases}$$

удовлетворяет свойству

$$\left(g_p(x),\ x - x_p^*\right) \ge 0 \quad \text{для всех} \quad x \in R^n. \tag{4}$$

Что означает лемма 1?

Если точка x находится

- в допустимой области, заданной ограничениями (3), то в качестве $g_p(x)$ выбирается субградиент функции $f_p(x)$, вычисленный в точке x;
- вне допустимой области, то выбирается субградиент к максимально нарушенному ограничению вида (3).

Выпуклость функции $f_p(x)$ и ограничений (3) гарантирует выполнение свойства (4) для векторного поля $g_p(x)$.

Локализация x_p^* в шаре $S(x_0, r_0)$

Лемма 2 (Стецюк, Колесник, Березовский, 2003)

Если $x_0 = \frac{1}{2}(u+l)$ и $r_0 = \frac{1}{2}\|u-l\|$, то параллелепипед $P(x) = \{x : l \le x \le u\}$ содержится в n-мерном шаре $S(x_0, r_0) = \{x : \ \|x - x_0\| \le r_0\}$.

Что означает лемма 2?

За центр шара выбираем центр параллелепипеда, а радиус шара выбираем так, чтобы шар имел минимальный объем.

Содержание

Параметры и критерий останова

Входные параметры метода:

параметр p ($p \geq 1$) и параметр $\varepsilon_f > 0$ – точность, с которой требуется найти значение $f_p^* = f_p(x_p^*)$.

Останов использует формулу для вычисления

"обобщенного" значения функции в задаче (2)–(3):

$$F_p(x) = \begin{cases} +\infty, & \text{если} \quad t^* > 0; \\ f_p(x), & \text{если} \quad t^* \leq 0. \end{cases}$$

Ицицилизация в методе

Инициализация:

- Положим стартовую точку $x_0 = (u + l)/2$ и начальный радиус $r_0 = \|u - l\|/2$. Вычислим $\beta = \sqrt{\frac{n-1}{n+1}}$.
- Введем в рассмотрение $n{\times}n$-матрицу B и положим $B_0 := I_n$, где I_n – единичная $n{\times}n$-матрица. Перейдем к первой итерации со значениями x_0, r_0 и B_0.

Пусть на k-й итерации найдены значения $x_k \in R^n$, r_k, B_k. Переход к $(k+1)$-й итерации состоит в выполнении такой последовательности действий.

Итерация метода

Шаг 1. Вычислим $F_p(x_k)$. Если $F_p(x_k) = 0$, то "Останов: $k^* = k$ и $x_p^* = x_k$". Иначе вычислим $g_p(x_k)$. Если $F_p(x_k) < +\infty$ и $\left\|B_k^T g_p(x_k)\right\| r_k \le \varepsilon_f$, то "Останов: $k^* = k$ и $x_p^* = x_k$". Иначе переходим к шагу 2.

Шаг 2. Положим $\xi_k := \dfrac{B_k^T g_p(x_k)}{\left\|B_k^T g_p(x_k)\right\|}$.

Шаг 3. Вычислим очередную точку

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{где} \quad h_k = \frac{1}{n+1} r_k.$$

Шаг 4. Вычислим

$$B_{k+1} := B_k + (\beta - 1)\left(B_k \xi_k\right) \xi_k^T \quad \text{и} \quad r_{k+1} := r_k \frac{n}{\sqrt{n^2 - 1}}.$$

Шаг 5. Переходим к итерации $(k+1)$ со значениями $x_{k+1}, r_{k+1}, B_{k+1}$.

Сходимость метода

Теорема.

Пусть $A_k = B_k^{-1}$. Последовательность точек $\{x_k\}_{k=0}^{k^*}$ удовлетворяет неравенству

$$\left\|A_k(x_k - x_p^*)\right\| \leq r_k, \quad k = 0, 1, 2, \ldots, k^*$$

На каждой итерации $k>0$ уменьшение объема эллипсоида $\mathcal{E}_k = \{x \in R^n : \|A_k(x_k - x)\| \leq r_k\}$, локализующего точку x_p^*, есть величина постоянна и равна

$$q = \frac{vol(\mathcal{E}_k)}{vol(\mathcal{E}_{k-1})} = \sqrt{\frac{n-1}{n+1}}\left(\frac{n}{\sqrt{n^2-1}}\right)^n < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

Что следует из теоремы?

Для уменьшения в 10 раз объема эллипсоида, локализующего x_p^*, требуется K итераций, где

$$K = -\frac{\ln 10}{\ln q} \approx (2\ln 10)n \approx 4.6n,$$

т.е. чтобы на порядок улучшить отклонение найденного рекордного значения функции $f_p(x)$ от ее оптимального значения f_p^*, потребуется сделать $4.6n^2$ итераций.

Величина m не влияет на скорость сходимости,

от нее зависит трудоемкость вычисления значения функции $f_p(x)$ и ее субградиента $\partial f_p(x)$. При $m{\sim}1000$ это вносит в трудоемкость метода более весомый вклад, чем алгоритмические операции (шаги 2–4).

Содержание

Заключение (Применение 1)

Метод эллипсоидов можно успешно применять для нахождения x_p^*, если количество переменных $n = 2 \div 10$.

Чтобы найти f_p^* с относительной точностью равной 10^{-10}, максимальное количество итераций следует из таблицы

n	itn	n	itn	n	itn
2	177	5	1144	8	2940
3	407	6	1651	9	3723
4	730	7	2249	10	4598

Заключение (Применение 2)

Метод эллипсоидов может быть использован для линейной регрессии с общими ограничениями на параметры θ:

$$\left(\sum_{i=1}^{m} |y_i - x_i^T \theta|^p\right)^{1/n} \to \min_{\theta \in R^n} \qquad (5.a)$$

при условиях

$$b^{low} \le A\theta \le b^{up}, \qquad (5.b)$$

Задача (5) для $p = 2$ исследована в книге

[4]. Knopov P.S., Korkhin A.S. *Regression Analysis Under a Priori Parameter Restrictions.* – Springer, 2012. – 234 p.

Благодарности

Работа выполнена при поддержке

Volkswagen Foundation, грант No 90 306
(первый и второй авторы)

и НАН Украины, проект № 0116U004558
(первый и третий авторы).

Запитання?

ДЯКУЮ ЗА УВАГУ!

BACK UP SLIDES: octave-функция emshor

```
# octave-code for emshor (version 1.0, 13.09.2013)      #row00
# находит точку минимума выпуклой функции f(x)          #row00a
% Входные параметры:                                    #row00b
% calcfg - имя функции calcfg(x): вычисляет f и g(1:n) #row00c
% x  - начальная точка x0(1:n) (на выходе портится)     #row00d
% rad - радиус шара с центром в точке x0                #row00e
% epsf - точность останова по значению функции          #row00f
% maxitn - максимальное количество итераций             #row00g
% Выходные параметры:                                   #row00h
% xr -- точка минимума (рекордная)                      #row00i
% fr -- значение функции в точке xr                     #row00j
% ist -- код останова (1 = epsf, 4 = maxitn)            #row00k
function [xr,fr,ist]=emshor(calcfg,x,rad,epsf,maxitn); #row01
dn=float(length(x)); beta=sqrt((dn-1.d0)/(dn+1.d0));   #row02
radn=rad; B=eye(length(x)); fr=inf; ist=4;             #row03
for (itn = 0:maxitn)                                    #row04
   [f, g1] = calcfg(x); g=B'*g1; dg=norm(g);           #row05
   if (f < fr)  fr = f; xr = x; itr=itn; endif         #row06
   if(radn*dg < epsf) ist = 1;  return; endif          #row07
   xi=(1.d0/dg)*g; dx = B * xi; hs=radn/(dn+1.d0);     #row08
   x -= hs * dx;  B += (beta - 1) * B * xi * xi';      #row09
   radn=radn/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);    #row10
   printf("itn %4d itr %4d f %16.8e fr %16.8e\n",      #row11
          itn,itr,f,fr);
endfor                                                  #row12
endfunction                                             #row13
```

Что находит программа emshor?

Пусть $f(x)$ – выпуклая функция, x^* – точка минимума, $f^* = f(x^*)$, x_0 – начальное приближение.

Теорема (о программе emshor)

Если x_0 такое, что $\|x_0 - x^*\| \leq rad$, то программа **emshor** заканчивает работу выполнением одного из условий:

1. найдена точка x_r – такая, что $f(x_r) - f^* \leq \varepsilon_f$ (ist=1),
2. **maxitn** итераций оказалось недостаточно (ist=4).

octave-функция emshor (комментарии)

```
# octave-code for emshor (version 1.0, 13.09.2013)      #row00
# находит точку минимума выпуклой функции f(x)          #row00a
% Входные параметры:                                    #row00b
% calcfg - имя функции calcfg(x): вычисляет f и g(1:n) #row00c
% x   - начальная точка x0(1:n) (на выходе портится)    #row00d
% rad - радиус шара с центром в точке x0                #row00e
% epsf - точность останова по значению функции          #row00f
% maxitn - максимальное количество итераций             #row00g
% Выходные параметры:                                   #row00h
% xr -- точка минимума (рекордная)                      #row00i
% fr -- значение функции в точке xr                     #row00j
% ist -- код останова (1 = epsf, 4 = maxitn)            #row00k
```

octave-функция emshor (код программы)

```
function [xr,fr,ist]=emshor(calcfg,x,rad,epsf,maxitn); #row01
dn=float(length(x)); beta=sqrt((dn-1.d0)/(dn+1.d0));   #row02
radn=rad; B=eye(length(x)); fr=inf; ist=4;             #row03
for (itn = 0:maxitn)                                   #row04
   [f, g1] = calcfg(x); g=B'*g1; dg=norm(g);           #row05
   if (f < fr)  fr = f; xr = x; itr=itn; endif         #row06
   if(radn*dg < epsf) ist = 1;  return; endif          #row07
   xi=(1.d0/dg)*g; dx = B * xi; hs=radn/(dn+1.d0);     #row08
   x -= hs * dx;  B += (beta - 1) * B * xi * xi';      #row09
   radn=radn/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);    #row10
   printf("itn %4d itr %4d f %16.8e fr %16.8e\n",      #row11
          itn,itr,f,fr);
endfor                                                 #row12
endfunction                                            #row13
```