ЭВРИСТИЧЕСКИЕ ВЕРОЯТНОСТНЫЕ АЛГОРИТМЫ ОРТОГОНАЛЬНОГО ПРОЕКТИРОВАНИЯ ТОЧКИ НА МНОЖЕСТВО*

М. Э. Аббасов
abbasov.majid@gmail.com

19 ноября 2015 г.

Аннотация. Задача ортогонального проектирования точки на множество возникает во многих отраслях математики и ее приложениях. Существуют эффективные методы проектирования (см. например [1–5]), позволяющие находить решение с любой наперед заданной точностью. Однако, как правило, для запуска этих алгоритмов необходимо начальное приближение. Его поиск является самостоятельной, подчас достаточно сложной, задачей. Трудность заключается еще и в том, что при неудачном выборе начального приближения применяемые методы могут приводить к локальным минимумам. Поэтому на практике возникает необходимость в вычислительных процедурах, позволяющих, пусть и грубо, получать начальные точки, находящиеся в некоторой окрестности глобального решения.

Отметим, что в некоторых задачах даже такое грубое приближение может быть принято в качестве решения. Так, при реализации многих методов оптимизации негладких функций на каждой итерации решается подзадача по поиску проекции начала координат на некоторый выпуклый компакт (см. [6–10]). Эта проекция определяет направление наискорейшего спуска/подъема, вдоль которого делается некий положительный шаг, в результате чего значение функции на очередной итерации уменьшается/увеличивается. В таких методах можно двигаться и вдоль направлений, близких к указанному – это также обеспечивают убывание/рост функции.

В данной работе предлагается несколько простых, эффективных вычислительных процедур, использующих случайный поиск для нахождения начального приближения в задаче ортогонального проектирования точки на множество. Они, как и другие представители класса вероятных методов [11], нала-

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

гают незначительные ограничения на исследуемую задачу: множество может иметь достаточно сложную структуру, важно лишь чтобы мы могли проверять принадлежность произвольной точки этому множеству.

1°. Постановка задачи. Рассмотрим задачу

$$\begin{cases} \|x - \widetilde{x}\| \longrightarrow \inf \\ x \in X \end{cases} \tag{1}$$

где $X \subset \mathbb{R}^n$ — некоторое замкнутое множество. Без уменьшения общности можно считать $\widetilde{x} = 0_n$. С учетом сказанного, (1) перепишется в виде

$$\begin{cases} \|x\| \longrightarrow \min \\ x \in X \end{cases}$$

Эту задачу и будем рассматривать в дальнейшем.

2°. Базовый алгоритм. Выберем некоторое малое $\varepsilon > 0$, натуральное m, произвольную точку $x_0 \in \mathbb{R}^n$, такую что $\mathbb{U}_{\|x_0\|} \bigcap X \neq \emptyset$. Здесь $\mathbb{U}_r = \left\{x \in \mathbb{R}^n \mid \|x\| < r\right\}$ — открытый шар радиуса r с центром в начале координат. Если размерность множества X (или какой-то его части) меньше размерности пространства, рассматриваем множество X с некоторой его ε окрестностью. Такое множество обозначим X_ε. Опишем алгоритм.

Предположим имеется $x_k \in X_\varepsilon$. Принимаем $j = 0$.

1) Генерируем с помощью многомерного равномерного распределения на $\mathbb{U}_{\|x_k\|}$ случайный вектор y_j, проверяем условие $y_j \in X_\varepsilon$.

2) Если условие выполнено, принимаем $x_{k+1} = y_j$, $j = 0$, увеличиваем k на единицу и переходим к следующей итерации.

3) Если условие не выполнено, и

   (a) $j < m$, то увеличиваем j на единицу и переходим к шагу 1;

   (b) $j = m$, то принимаем $x_* = x_k$ и завершаем процесс.

Использование ε-окрестности множества вместо самого множества обусловлено тем, что при размерности X меньшей размерности пространства, вероятность попадания случайной точки в X будет равна нулю. Очевидно, если размерность множества X равна n, можно принять $\varepsilon = 0$. Если X задано с помощью равенств и неравенств, то для определения X_ε можно неравенства $g(x) \leqslant 0$ заменить на $g(x) \leqslant \varepsilon$, а равенства $g(x) = 0$ на $|g(x)| \leqslant \varepsilon$. Идея самого алгоритма достаточно проста для программной реализации. Однако в большинстве математических пакетов и языков программирования встроенными

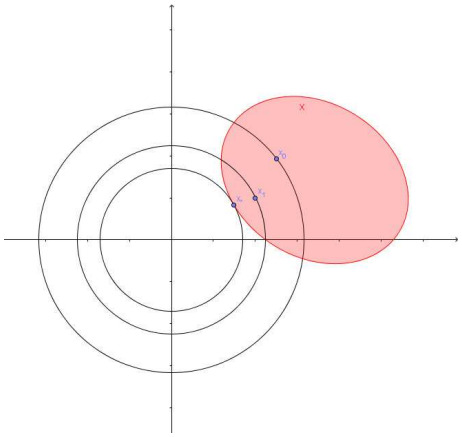


Иллюстрация работы базового алгоритма

средствами удается генерировать случайные вектора равномерно распределенные лишь на параллелепипеде. Это препятствие можно обойти, установив взаимно однозначное соответствие между открытым кубом с центром в нуле и ребрами длины 2, параллельными осям координат, и открытым шаром единичного радиуса с центром в нуле.

Возьмем точку $z = (z^{(1)}, \ldots, z^{(n)})$ из куба $(-1, 1) \times \cdots \times (-1, 1)$, и пусть $|z^{(i)}| = \max\{|z^{(1)}|, \ldots, |z^{(n)}|\} > 0$. Прямая, проходящая через начало координат O и точку z, очевидно, пересечет ту грань куба, в которой i-я координата равна $\operatorname{sign} z^{(i)}$. Точку пересечения обозначим A.

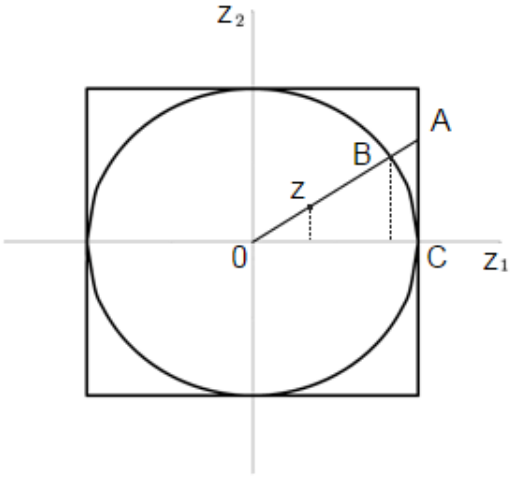


Рис. 1. Взаимно однозначное соответствие в двумерном случае

Имеем

$$\frac{OA}{1} = \frac{\|z\|}{|z^{(i)}|}, \tag{2}$$

то есть $OA = \frac{\|z\|}{\max\{|z^{(1)}|,\ldots,|z^{(n)}|\}}$. В справедливости (2) при $n = 2$ можно убедиться на рис. 1. Уменьшая каждую из координат вектора z в OA раз, получим соответствующую точку из открытого единичного шара с центром в нуле. Итак, окончательно,

$$z \Leftrightarrow y = \frac{\max\{|z^{(1)}|, \ldots, |z^{(n)}|\}}{\|z\|} z.$$

Ясно, что построенное таким образом соответствие является взаимно однозначным.

Замечание. Использование сферических координат для построения указанного соответствия является более дорогостоящим с вычислительной точки зрения. Так, предложенный метод требует порядка $O(n)$ простейших арифметических операций для перехода от точки в кубе к точке в шаре, в то время, как подход, основанный на использовании сферических координат, для аналогичного перехода потребовал бы $O(n^2)$ таких операций, и такого же количества вычислений тригонометрических функций.

3°. Алгоритм с половинным делением. Можно усилить алгоритм используя идею половинного деления. Обозначим

$$\mathbb{U}_{r,R} = \left\{x \in \mathbb{R}^n \middle|\ r < \|x\| < R\right\}$$

открытый шар радиуса R с вырезанной центральной частью радиуса r.

Выберем произвольную точку $x_0 \in X$, $R_0 = \|x_0\|$, $r_0 = 0$, некое натуральное m, и малое $\varepsilon > 0$, $\widetilde{\varepsilon} > 0$. Пусть имеется x_k, r_k, R_k. Принимаем $j = 0$.

1) Генерируем с помощью многомерного равномерного распределения на $\mathbb{U}_{r_k, \frac{R_k + r_k}{2}}$ случайный вектор y_j, проверяем условие $y_j \in X_\varepsilon$.

2) Если условие выполнено, принимаем $R_{k+1} = \|y_j\|$, $r_{k+1} = r_k$, $x_{k+1} = y_j$, $j = 0$, переходим к пункту 4.

3) Если условие не выполнено и

   (a) $j < m$, то увеличиваем j на единицу и переходим к шагу 1;

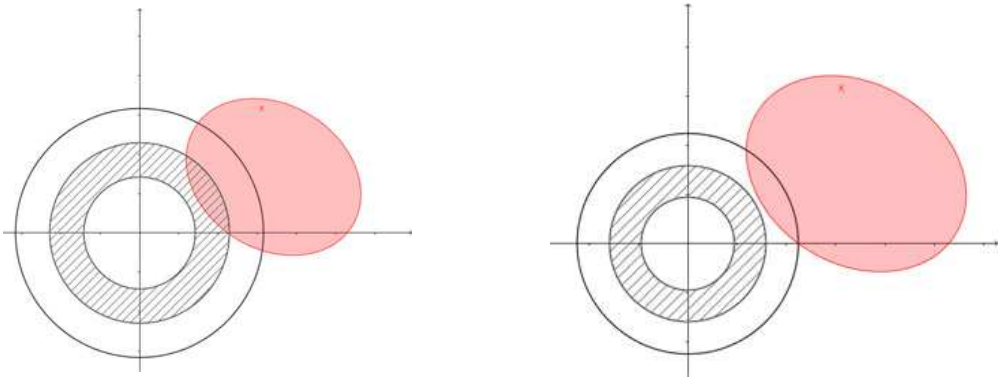
   (b) $j = m$, принимаем $r_{k+1} = \frac{R_k + r_k}{2}$, $R_{k+1} = R_k$, $x_{k+1} = x_k$, переходим к пункту 4.

4) Проверяем критерий остановки

$$\|R_{k+1} - r_{k+1}\| \leqslant \widetilde{\varepsilon}, \tag{3}$$

(a) если он выполнен, то принимаем $x_* = x_{k+1}$ и заканчиваем вычисления,

(b) если он не выполнен, то увеличиваем k на единицу и переходим к следующей итерации.



а) б)

Штриховкой указана область, в которой генерируются случайные точки. Случаи: а) $r_{k+1} = r_k$, $R_{k+1} = \|y\|$. б) $r_{k+1} = \frac{R_k + r_k}{2}$, $R_{k+1} = R_k$.

Для генерации равномерно распределенных на $\mathbb{U}_{r,R}$ векторов нужно воспользоваться идеей, аналогичной использованной в предыдущем пункте. Пусть точка

$$z = (z^{(1)}, \ldots, z^{(n)}) \in (-1, 1) \times \cdots \times (-1, 1).$$

Тогда с помощью взаимно однозначного соответствия

$$z \Leftrightarrow y = r\frac{u}{\|u\|} + (R - r)u, \;\; u = \frac{z}{\|z\|} \max_{i \in 1:m} |z^{(i)}|$$

получаем требуемое.

4°. Алгоритм с обучением. В двух описанных выше алгоритмах при генерации случайных точек все направления равнозначны, мы движемся «вслепую», никак не используя информацию с предыдущих итераций. Ясно однако, что точки множества x, полученные на предыдущих шагах определяют некое перспективное направления поиска. Логичнее было бы построить алгоритм так, чтобы в процессе вычислений он перестраивал закон распределения, делая перспективные направления более вероятными и одновременно корректируя само это перспективное направление на основании вновь полученной информации. Алгоритмы, обладающие такими свойствами, называю алгоритмами с обучением [12].

Выберем произвольную точку $x_0 \in X$, $\omega_0 = 0_n$, $\beta \in \mathbb{R}$, $\delta \in \mathbb{R}$, натуральное m, и малое ε. Пусть имеется x_k, ω_k. Принимаем $j = 0$.

1) Получаем с помощью генератора случайных векторов, равномерно распределенных на открытом единичном шаре $\mathbb{U}_1$ с центром в нуле, элемент η; вычисляем
$$y_j = \frac{\omega_k + \eta}{\|\omega_k + \eta\|}\frac{\|x\|}{2} + \frac{x}{2}; \tag{4}$$
проверяем условие $y_j \in X_\varepsilon$.

2) Если условие выполнено, принимаем $x_{k+1} = y_j$, $\omega_{k+1} = \beta\omega_k + \delta x_{k+1}$, $j = 0$, увеличиваем k на единицу и переходим к следующей итерации.

3) Если условие не выполнено и

   (a) $j < m$, то увеличиваем j на единицу и переходим к шагу 1;

   (b) $j = m$, то принимаем $x_* = x_k$ и завершаем процесс.

Параметр β называют коэффициентом забывания, а δ – коэффициентом интенсивности обучения. Таким образом происходит накопление успешного опыта, полученного на предыдущих итерациях, и ω_k все точнее указывает направление, в котором лежит решение задачи.

Из (4) следует, что при больших значениях нормы вектора ω_k, именно это направление является главным направлением случайного варьирования x_k. Однако из того же выражения очевидно, что при очень больших значениях нормы вектора ω, метод становится излишне детерминированным (роль случайного вектора η становится незначительной) и алгоритм теряет способность к быстрому обучению. Для преодоления этой проблемы приходится ограничивать возможность чрезмерного доминирования ω.

С этой целью во второй пункт изложенного выше алгоритма можно ввести дополнительную проверку, сравнивая норму полученного ω_{k+1} с некоторым заранее выбранным числом c, принимая
$$\omega_{k+1} = c\frac{\omega_{k+1}}{\|\omega_{k+1}\|}, \text{ если } \|\omega_{k+1}\| > c.$$

5°. Численные эксперименты. Покажем, как указанные подходы работает на примерах[1]. Во всех примерах, если не оговорено иное, принимается $\varepsilon = 0.001$, $\widetilde{\varepsilon} = 0.001$, $m = 5000$, $\beta = 1$, $\delta = 1$, $c = 25$, задача каждый раз решается 100 раз, выводятся средние значения отклонения от точного решения и среднее время работы предложенных алгоритмов в секундах.

[1]Вычисления велись в математическом пакете MATLAB R2013, на компьютере с ЦП Core i5-3450, 3.1 GHz; ОЗУ 8,00 ГБ DDR3 1833MHz, ОС Windows 10

ПРИМЕР 1. Пусть в $\mathbb{R}^3$ нужно найти расстояние от начала координат до множества

$$\begin{cases} x^{(1)} + x^{(2)} + x^{(3)} = 1 \\ x^{(i)} \geqslant 0,\ i = 1, 2, 3 \end{cases}$$

Приняв $x_0 = (1/2, 1/2, 1/4)^T$, получим

Алгоритм	$\|x - x_*\|$	$time$
Исходный	0.1531	0.2403
С дихотомией	0.0191	0.9059
С обучением	0.0006	0.8675

ПРИМЕР 2. Найдем в $\mathbb{R}^2$ расстояние от начала координат до эллипсоида

$$X = \{x \in \mathbb{R}^2 |\ 4{x^{(1)}}^2 + 2{x^{(2)}}^2 + 5x^{(1)}x^{(2)} - 30x^{(1)} - 21x^{(2)} + 47 \leqslant 0\}.$$

Здесь размерность множества совпадает с размерностью пространства, поэтому можно принять $\varepsilon = 0$. Начальную точку возьмем $x_0 = (2, 2)^T$. Имеем

Алгоритм	$\|x - x_*\|$	$time$
Исходный	0.057874	0.0738
С дихотомией	0.0015	0.7616
С обучением	0.000006	0.2610

ПРИМЕР 3. Найдем минимальное расстояние от нуля до надграфика функции

$$x^{(2)} = \min\left\{|x^{(1)} + 1| + 1; |x^{(1)} - 1| + 1\right\}.$$

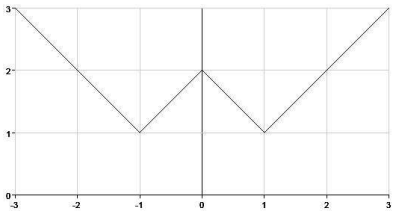


График функции из примера 3

Принимаем $\varepsilon = 0$. Очевидно, здесь два возможных решения $x_* = (1,1)^T$ и $x_{**} = (-1,1)^T$. Выберем $x_0 = (0,2)^T$, равноудаленную от решений. Имеем

Алгоритм	$\min\{\|x - x_*\|; \|x - x_{**}\|\}$	$time$	N_1	N_2
Исходный	0.045703	0.0697	46	54
С дихотомией	0.00088	0.7753	49	51
С обучением	0.000006	0.3129	55	45

N_1 – количество решений из окрестности x_*,

N_2 – количество решений из окрестности x_{**}.

ПРИМЕР 4. Найдем минимальное расстояние от нуля до надграфика функции

$$x^{(2)} = \min\left\{|x^{(1)} + 2| + 2; |x^{(1)} - 1| + 1\right\}$$

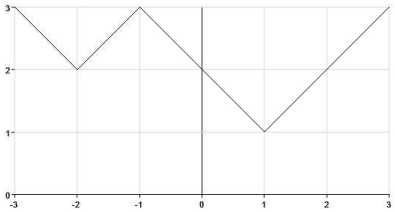


График функции из примера 4

Возьмем $\varepsilon = 0$, в качестве начальной выберем точку $x_0 = (-2,3)^T$, расположенную ближе к локальному решению $x_{**} = (-2,2)^T$, чем к глобальному $x_* = (1,1)^T$. Получим

Алгоритм	$\|x - x_*\|$	$time$	N_1	N_2
Исходный	0.051789	0.0731	100	0
С дихотомией	0.0027	0.7950	100	0
С обучением[1]	1.1384	0.3619	64	36
С обучением[2]	0.000144	0.2697	100	0

N_1 – количество решений из окрестности x_*,

N_2 – количество решений из окрестности x_{**},

[1] при $\beta = 1$, $\delta = 1$, $c = 25$,

[2] при $\beta = 1$, $\delta = 1$, $c = 1$.

ПРИМЕР 5. Пусть $x \in \mathbb{R}^n$. Найдем минимальное расстояние от нуля до надграфика функции от n переменных

$$x^{(2)} = \min\left\{|x^{(1)} + 2| + 2; |x^{(1)} - 1| + 1\right\}$$

при различных n. В качестве начальной точки возьмем случайный n-мерный вектор с элементами из $(0,1)$ умноженный на 5, примем $\varepsilon = 0$.

Исходный алгоритм	$n = 5^I$	$n = 7^{II}$	$n = 10^{III}$	$n = 12^{IV}$
$\|x - x_*\|$	0.4340	0.4872	0.6877	0.8069
$time$	0.0680	0.7646	8.2627	15.8842

$^{\text{I}}$ $m = 5 \cdot 10^3$, $^{\text{II}}$ $m = 5 \cdot 10^4$, $^{\text{III}}$ $m = 5 \cdot 10^5$, $^{\text{IV}}$ $m = 10^6$.

Алгоритм с дихотомией	$n = 5^I$	$n = 7^{II}$	$n = 10^{III}$	$n = 12^{IV}$
$\|x - x_*\|$	0.2683	0.3805	0.5769	0.7167
$time$	1.1362	13.5747	141.1904	298.2748

$^{\text{I}}$ $m = 5 \cdot 10^3$, $^{\text{II}}$ $m = 5 \cdot 10^4$, $^{\text{III}}$ $m = 5 \cdot 10^5$, $^{\text{IV}}$ $m = 10^6$.

Алгоритм с обучением	$n = 5^I$	$n = 10^{II}$	$n = 100^{III}$	$n = 1000^{IV}$
$\|x - x_*\|$	0.2291	0.2428	0.2538	0.1787
$time$	0.0793	0.0643	0.2970	11.913

$^{\text{I}}$ $\beta = 0.05$, $\delta = 1$, $c = 2$, $m = 5 \cdot 10^3$
$^{\text{II}}$ $\beta = 0.05$, $\delta = 3$, $c = 4$, $m = 5 \cdot 10^3$
$^{\text{III}}$ $\beta = 0.05$, $\delta = 17$, $c = 20$, $m = 5 \cdot 10^3$
$^{\text{IV}}$ $\beta = 0.05$, $\delta = 50$, $c = 75$, $m = 5 \cdot 10^3$

6°. Выводы. Оборотной стороной эффективности алгоритма с обучением является возможное его «застревание» в локальном решении в случае невыпуклого множества X. Для преодоления этой проблемы нужно, чтобы алгоритм имел способность быстро обучаться, а для этого следует выбирать малые значения параметра c, которые, однако, приходится увеличивать при увеличении размерности пространства (см. пример 5).

В то же время для первых двух алгоритмов, в силу использования ими «слепого» поиска, не важно выпукло множество X или нет. Но, как показывает практика, они плохо работают в задачах больших размерностей — для получения приемлемой точности приходится увеличивать m, что приводит к значительному росту времени расчетов. Отметим, что алгоритм с дихотомией, как правило, работает дольше исходного, но выдает более точное решение.

Предложенные вычислительные процедуры могут быть использованы для поиска начального приближения, находящегося в окрестности глобального решения, откуда можно запускать более совершенные методы. Такая комбинация методов позволяет находить глобальное решение. К примеру, предлагаемая идея тандемного использования алгоритмов делает возможным применение метода заряженных шариков и к невыпуклым задачам.

7°. Благодарности. Работа выполнена при поддержке Санкт-Петербургского Государственного Университета в рамках гранта 9.38.205.2014.

ЛИТЕРАТУРА

1. Аббасов М. Э. *Метод заряженных шариков* // Семинар «DHA & CAGD». Избранные доклады. 21 мая 2015 г. (`http://www.apmath.spbu.ru/cnsa/reps15.shtml#0521`)

2. A. Lin, S. P. Han, *On the distance between two ellipsoids* // SIAM Journal on Optimization, 2002. Vol. 13. P. 298–308.

3. S.-M. Hu, J. Wallner, *A second order algorithm for orthogonal projection onto curves and surfaces* // Computer Aided Geometric Design, 2005. Vol. 22. P. 251–260.

4. Лебедев Д. М., Полякова Л. Н. *Задача проектирования нулевой точки на квадрику* // Вестник СПбГУ. Сер. 10, 2013. Вып. 1. С. 11–17.

5. Тамасян Г. Ш., Чумаков А. А. *Нахождение расстояния между эллипсоидами* // Дискретн. анализ и исслед. опер., 2014. Т. 21, № 3, С. 87–102.

6. V. F. Demyanov, A. M. Rubinov, *Constructive Nonsmooth Analysis, Approximation & Optimization* // Vol. 7. Peter Lang, Frankfurt am Main, 1995. iv+416 pp.

7. V. F. Demyanov, *Exhausters and Convexificators – New Tools in Nonsmooth Analysis* // In: V. Demyanov and A. Rubinov (Eds.) Quasidifferentiability and related topics. Dordrecht: Kluwer Academic Publishers, 2000. P. 85–137.

8. V. F. Demyanov, V. A. Roschina, *Optimality conditions in terms of upper and lower exhausters* // Optimization. Vol. 55, Issue 5-6, 2006. P. 525–540.

9. M. E. Abbasov, V. F. Demyanov, *Proper and adjoint exhausters in nonsmooth analysis: optimality conditions* // J. Glob. Optim., 2013. Vol. 56, Issue 2, P. 569–585.

10. M. E. Abbasov, V. F. Demyanov, *Adjoint Coexhausters in Nonsmooth Analysis and Extremality Conditions* // Journal of Optimization Theory and Applications. Vol. 156, Issue 3, 2013. P. 535–553.

11. Жиглявский А. А. *Математическая теория глобального случайного поиска* // Л.: Изд-во ЛГУ, 1985. 296 с.

12. Васильев Ф. П. *Методы оптимизации* // М.: Издательство «Факториал Пресс», 2002. 824 с.