Теорія оптимальних рішень

УДК 519.8

Н.Г. ЖУРБЕНКО, А.П. ЛИХОВИД

РЕГУЛЯРИЗАЦИЯ МАТРИЦЫ ПРЕОБРАЗОВАНИЯ *R*-АЛГОРИТМА

Предлагается модификация r-алгоритма, предназначенная для предотвращения вырождения матрицы, обратной к матрице преобразования пространства. Этот вариант r-алгоритма целесообразно использовать для задач минимизации существенно овражных функций сравнительно небольшой размерности.

Введение. Более 40 лет назад был разработан субградиентный алгоритм минимизации с растяжением пространства в направлении разности двух последовательных градиентов – *r*-алгоритм [1]. Практика использования *r*-алгоритма показывает, что до настоящего времени он является одним из наиболее эффективных алгоритмов негладкой оптимизации. *R*-алгоритм используется с большими значениями коэффициентов растяжения пространства (≈ 2). Поэтому при решении задач небольшой размерности с большой точностью это может привести к вырождению матрицы преобразования. В предлагаемой модификации *r*-алгоритма предложена процедура регулиризации этой матрицы.

Численная схема *r*-алгоритма. Рассматривается задача безусловной минимизации субдифференцируемой функции $f(x)$ в R^n. Обозначим $\partial f(x)$ множество субградиентов функции $f(x)$ в точке x. В *r*-алгоритме используется оператор растяжения пространства [2] $R(\eta) = (\alpha - 1)\eta\eta^T + I$, где $\eta \in R^n$, α – направление и коэффициент растяжения про-странства, $|\eta| = 1$, $\alpha \geq 0$. Вычислительная схе-ма *r*-алгоритма применительно к задаче отыскания безусловного минимума функции $f(x)$ состоит в следующем.

0-ой шаг алгоритма (инициализация). Выбираем начальное приближение x_0. Вычисляем: $g(x_0) \in \partial f(x_0)$; $g_0^* = B_0^* g(x_0)$ – субградиент в преобразованном пространстве $Y_0 = B_0^{-1} X \equiv A_0 X$, где X – исходное пространство.

Пусть на шаге k алгоритма $(k = 0,1,2,\ldots)$ получены определенные значения векторов x_k, g_k^* (субградиент в преобразованном пространстве) и матрицы B_k ($A_k = B_k^{-1}$ – матрица преобразования пространства).

$(k+1)$-вый шаг алгоритма $(k = 0,1,2,\ldots)$.

Вычисляем:

1) h_{k+1} – шаговый множитель, $h_{k+1} \geq 0$;

2) $x_{k+1} = x_k - h_{k+1} B_k g_k^* / |g_k^*|$;

3) $g(x_{k+1}) \in \partial f(x_k)$ (субградиент в точке x_{k+1});

4) $\tilde{g}_{k+1}^* = B_k^* g(x_{k+1})$ (субградиент в преобразованном пространстве $Y_k = A_k X$);

5) $\eta_{k+1} = (\tilde{g}_{k+1}^* - g_k^*)/|\tilde{g}_{k+1}^* - g_k^*|$ (направление растяжения пространства Y_k);

6) $\alpha_{k+1} \geq 1, \beta_{k+1} = 1/\alpha_{k+1}$ (α_k коэффициент растяжения пространства Y_k);

7) $B_{k+1} = B_k R_{\beta_{k+1}}(\eta_k)$, (обратный оператор преобразования пространства $Y_{k+1} = A_{k+1} X = B_{k+1}^{-1} X$);

8) $g_{k+1}^* = R_{\beta_{k+1}}(\eta_{k+1}) \tilde{g}_{k+1}$ $(g_{k+1}^* = B_{k+1}^* g(x_k)$ (субградиент в преобразованном пространстве $Y_{k+1} = A_{k+1} X = B_{k+1}^{-1} X$).

Переходим к $(k+2)$-му шагу алгоритма, или прекращаем работу при выполнении критерия останова.

В известных вариантах *r*-алгоритма выбор коэффициентов растяжения пространства и шаговых множителей определяется следующим образом. Значения коэффициентов растяжения пространства α_k (параметр *r*-алгоритма) выбираются одинаковыми на всех итерациях: $\alpha_k = \alpha > 1$. На практике рекомендуется это значение выбирать порядка 2. Величина шагового множителя определяется выбранной процедурой минимизации (процедурой «спуска») по направлению $p_k = -B_k g_k^* / |g_k^*|$. Применяемые процедуры являются достаточно грубой реализацией алгоритма локализации минимума по направлению p_k. Основным требованием при этом является выполнение условия $(p_k, g(x_{k+1})) \geq 0$ (это условие обеспечивает, что $|\tilde{g}_{k+1}^* - g_k^*| > 0$).

Наиболее часто используется следующая процедура регулировки шаговых множителей. Пусть на $(k+1)-$ой итерации помимо точки x_k, вектора спуска p_k определено значение «пробного шага» $\tilde{h}_k > 0$.

Пусть заданы числа $0 < q_1 \leq 1$, $q_2 \geq 1$ и целое число $L \geq 2$. Эти величины являются параметрами (константами) алгоритма. Они будут определять регулировку величин пробного шага. Параметр q_1 используется для уменьшения пробного шага, а q_2 и L для его увеличения.

На итерации k алгоритма определена величина пробного шага $\tilde{h}_k$. Величина пробного шага $\tilde{h}_0$ является входным параметром *r*-алгоритма.

Положим $z_0 = x_k$. Вычисляем: $z_i = z_{i-1} + \tilde{h}_k p_k$; $g_i = g(z_i) \in \partial f(z_i)$, $i = 1,2,\ldots$, до тех пор, пока при некотором $i = l$ выполнится неравенство $(g_i, p_k) \geq 0$. Тогда $x_{k+1} = z_l$. Если $i = 1$, то $\tilde{h}_{k+1} := q_1 \tilde{h}_k$. Если $i > L$, то $\tilde{h}_k := q_2 \tilde{h}_k$.

Наиболее часто используются следующие значения параметров регулировки пробного шага $q_1 = 0.9$, $q_2 = 1.2$, $L = 3$.

Как видно из описания, на итерации *r*-алгоритма (п. 2) используется операция деления на $|g_k^*|$ – норму субградиента в преобразованном пространстве. Поэтому при программной реализации *r*-алгоритма необходимо принять меры по устранению возможной ошибки «деление на нуль». Это связано с тем, что $g_k^* = B_k^* g(x_{k-1})$, а матрица B_k равна произведению матриц операторов «сжатия» ($B_k = B_{k-1} R_{\beta_k}(\eta_{k-1})$) при достаточно малых значениях коэффициентов β_k (при $\alpha_k = 2$, $\beta_k = 1/\alpha_k = 1/2$). Отсюда следует, что с увеличением числа итераций величина $|g_k^*|$ может быть существенно меньше $|g_{k-1}|$. Таким образом, возможна ситуация, когда задача с заданной точностью еще не решена, а величина $|g_k^*|$ принимает недопустимо малое значение.

Для учета такой ситуации в некоторых программных реализациях *r*-алгоритма, например, AmplRalg [3] (размещена на сайте http://www.icyb.kiev.ua/file/NonDiffOpt_Nurm/indexk.htm), используется процедура «восстановления» матрицы B_k. По существу она состоит в рестарте *r*-алгоритма с начальной точки, соответствующей полученному приближению x_{k-1} и коррекции пробного шага. Использование такой процедуры может привести к увеличению трудоемкости алгоритма при решении задач небольшой размерности с высокой точностью.

В данной работе предлагаются дополнительные меры предотвращения описанной ситуации вырождения матрицы B_k. Эти меры состоят в следующем.

При пересчете матрицы B_k (п. 7 схемы алгоритма) вместо оператора $R_{\beta_{k+1}}(\eta_k)$ использовать оператор $\tilde{R}_{\beta_{k+1}}(\eta_k) = (1/\beta_{k+1}) R_{\beta_{k+1}}(\eta_k)$ (соответственно $\tilde{R}_{\alpha_{k+1}}(\eta_k) = (1/\alpha_{k+1}) R_{\alpha_{k+1}}(\eta_k)$). Основное отличие оператора $\tilde{R}_{\beta_{k+1}}(\eta_k)$ от оператора $R_{\beta_{k+1}}(\eta_k)$ состоит в следующем: $\det(\tilde{R}_{\beta_{k+1}}(\eta_k)) = 1$, $\det(R_{\beta_{k+1}}(\eta_k)) = \beta_{k+1} < 1$. Таким образом, соответствующее оператору $\tilde{R}_{\beta_{k+1}}(\eta_k)$ преобразование пространства происходит с сохранением объемов. Оператор $\tilde{R}_{\alpha_{k+1}}(\eta_k)$ фактически можно интерпретировать как «растяжение» пространства оператором $R_{\alpha_{k+1}}(\eta_k)$ и равномерным сжатием по всем направлениям с коэффициентом $\sqrt[n]{\beta_{k+1}} = 1/\sqrt[n]{\alpha_{k+1}}$.

Заметим, что это дополнительное сжатие не изменяет структуру поверхностей уровня функции – изменяется лишь их масштаб.

На эвристическом уровне эффект использования оператора $\tilde{R}_{\beta_{k+1}}(\eta_k)$ состоит в следующем. Рассмотрим минимизацию овражной квадратичной функции – функции с вытянутыми поверхностями уровня. При использовании оператора $R_{\alpha_{k+1}}(\eta_k)$ степень вытянутости эллипсоидов поверхностей уменьшается, приближаясь к близким сферическим (это – основное свойство *r*-алгоритма). Однако при этом уменьшается объем эллипсоидов. Вообще говоря, это полезное свойство используемого в *r*-алгоритме оператора $R_{\alpha_{k+1}}(\eta_k)$, которое косвенно влияет на управление шаговыми множителями алгоритма. Однако возникает указанная проблема вырожденности матрицы B_k. Отметим, что эта проблема может возникнуть при минимизации существенно овражных функций – функций, для которых направление антисубградиента с направлением на точку минимума может составлять угол сколь угодно близкий к $\pi/2$.

Предлагаемая модификация *r*-алгоритма состоит в использовании операторов $\tilde{R}_{\beta_{k+1}}(\eta_k)$ взамен оператора $R_{\beta_{k+1}}(\eta_k)$ (пп. 7) и 8)) вычислительной схемы *r*-алгоритма. При этом описанная выше процедура корректировки величины пробного шага $\tilde{h}_k$ дополняется в п. 7) оператором $\tilde{h}_k := \sqrt[n]{\alpha_k}$. Таким образом модифицированный алгоритм обозначается в дальнейшем как *Ř*-алгоритм.

Для выяснения целесообразности использования *Ř*-алгоритма проведены исследования его численной эффективности.

Численная эффективность *Ř*-алгоритма. Приведем результаты численных исследований эффективности *Ř*-алгоритма в сравнении с *r* -алгоритмом.

1. Вычислительные эксперименты для примера Лемарешаля [6]. Задача состоит в минимизации существенно овражной выпуклой негладкой функции от 10 переменных:

$$f(x) = \max_{1 \le k \le 5} f_k(x) \to \min_x, \qquad (1)$$

где $f_k(x) = x^T A_k(x) - b_k^T(x)$, A_k – симметричные 10×10-матрицы, такие, что $A_{kij} = e^{i/j}\cos(ij)\sin k$, если $i < j$ и $A_{kii} = i|\sin k|/10 + \sum_{j \ne i}|A_{kij}|$, а компоненты векторов b_k определяются $b_{ki} = e^{i/k}\sin(ik)$. Начальным приближением является точка $x_0 = (1,\ldots,1)^T \in R^{10}$, в которой реализуется значение функции $f(x_0) = 5337.06643$. Функция имеет единственную точку минимума, минимальное значение функции f^* определяется его достаточно точным приближением $f^*_{\min} = -0.841408334596$ (с точностью 10^{-12}, 12 цифр после точки).

В табл. 1 приведены результаты (точность по аргументу ε_x = 1.00000000E-010, точность по градиенту ε_g = 1.00000000E-010). Здесь приняты следующие обозначения: k – номер итерации, на которой алгоритм прекратил работу; k_g – количество вычислений субградиента; $k_{restart}$ – количество процедур восстановления матрицы.

ТАБЛИЦА 1. Решение задачи для примера Лемарешаля

Параметры / Алгоритм	q_1	q_2	α	k	k_g	$k_{restart}$
Ř	0.9	1.2	10	128	452	0
r	0.9	1.2	10	154	520	2
Ř	0.9	1.2	4	137	243	0
r	0.9	1.2	4	166	316	1
Ř	0.9	1.2	6	117	270	0
r	0.9	1.2	6	149	348	1

2. Вычислительные эксперименты для примера с квадратичной функцией. В качестве тестовой задачи рассматривалась задача минимизации следующей функции: $f(x)=\sum_{i=1}^{n}\rho_n^{i-1}x_i^2$, где параметр ρ_n выбирался в зависимости от размерности задачи n по формуле $\rho_n=10^{6/(n-1)}$. Таким образом, степень вытянутости линий уровня (степень «овражности») функций определяется значением параметра $\rho_n^{n-1}=10^6$, она одинакова для всех функций независимо от числа переменных. Начальная точка $x_i=1.0, i=1,2,...n.$ Критерий останова выбирался, как и для предыдущей задачи.

Результаты решения тестовых задач минимизации функции $f(x)$ приведены в табл. 2.

ТАБЛИЦА 2 Решение задачи для примера с квадратичной функцией

Параметры / Алгоритм	n	q_1	q_2	α	k	k_g	$k_{restart}$
$\check{R}$	10	0.9	1.2	4	135	213	0
r	10	0.9	1.2	4	144	231	2
$\check{R}$	10	0.9	1.2	10	116	308	0
r	10	0.9	1.2	10	151	404	3
$\check{R}$	10	0.9	1.2	20	110	398	0
r	10	0.9	1.2	20	172	642	5

3. Вычислительные эксперименты для примера Розенброка [7]. Задача состоит в минимизации овражной невыпуклой функции.

Функция Розенброка для двух переменных определяется как:

$$f(x,y)=(1-x)^2+100(y-x^2)^2.$$

Она имеет глобальный минимум в точке $(x,y)=(1,1)$, где $f(x,y)=0$. Начальное приближение – точка $x_0=(-1.2,1.0)^T\in R^2$. Критерий останова выбирался, как и для предыдущей задачи. В табл. 3 приведены результаты.

ТАБЛИЦА 3. Решение задачи для примера Розенброка

Параметры / Алгоритм	q_1	q_2	α	k	k_g	$k_{restart}$
Ř	0.9	1.2	2	66	224	0
r	0.9	1.2	2	72	231	3
Ř	0.9	1.2	3	48	265	0
r	0.9	1.2	3	61	289	4

Выводы. *Ř*-алгоритм – это модификация *r*-алгоритма, предназначенная для предотвращения вырождения матрицы, обратной к матрице преобразования пространства. Численные эксперименты показали, что эффективность *Ř*-алгоритма и *r*-алгоритма примерно одинакова. Однако для задач минимизации существенно овражных функций и сравнительно небольшой размерности целесообразно использовать *Ř*-алгоритм.

Работа выполнена при частичной поддержке Volkswagen Foundation (грант No 90 306 – Н. Г. Журбенко).

М.Г. Журбенко, О.П. Лиховид

РЕГУЛЯРИЗАЦІЯ МАТРИЦІ ПЕРЕТВОРЕННЯ *R*-АЛГОРИТМУ

Пропонується модифікація *r*-алгоритму, призначена для запобігання виродження матриці, оберненої до матриці перетворення простору. Цей варіант *r*-алгоритму доцільно використовувати для задач мінімізації істотно яружних функцій порівняно невеликої розмірності.

M.G. Zhurbenko, O.P. Lykhovyd

REGULARIZATION OF THE TRANSFORMATION MATRIX OF *R*-ALGORITHM

A modification of *r*-algorithm is proposed, developed to prevent degeneration of the matrix inverse to the space transformation matrix. This variant of *r*-algorithm should be used to minimize essentially ravine functions of relatively small dimension.

Список литературы

1. Шор Н.З., Журбенко Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. *Кибернетика*. 1971. № 3. С. 51 – 59.
2. Шор Н.З. Методы минимизации недифференцируемых функций и их применение. Киев: Наук. думка, 1979. 200 с.
3. Лиховид А.П. Об одной реализации *r*-алгоритма. *Теорія оптимальних рішень.* Київ: Ін-т кібернетики імені В.М. Глушкова НАН України, 2011. С. 91 – 95.
4. Журбенко Н.Г. Об одной модификации *r*-алгоритма. Материалы III-й Международной конференции *„Математическое моделирование, оптимизация и информационные технологии”*. Кишинеу: Эврика, 2012. С. 355 – 361.
5. Журбенко Н.Г., Чумаков Б.М. Программное управление коэффициентами растяжения *r*-алгоритма. *Теорія оптимальних рішень*. Київ: Ін-т кібернетики імені В.М. Глушкова НАН України, 2012. С. 113 – 118.
6. Nonsmooth optimization. Eds. C.Lemareshal, R.Mifflin. Oxford: Pergamon Press, 1978. 186 p.
7. Rosenbrock H.H. An automatic method for finding the greatest or least value of a function. *The Computer Journal*. 1960. 3 (3). P. 175 – 184.

Получено 15.03.2018