Численные методы

УДК 519.853.62

ПРОЦЕДУРЫ ОБНОВЛЕНИЯ В МЕТОДЕ СОПРЯЖЕННЫХ ГРАДИЕНТОВ

Г.И.Забиняко

Рассмотрим применение метода сопряженных градиентов для определения минимума выпуклой непрерывно дифференцируемой функции $f(x)$, $x \in R^n$. Если f в окрестности точки минимума имеет матрицу вторых производных, удовлетворяющую условию Липшица, то для метода установлена сверхлинейная скорость сходимости (см., например, [1]) при дополнительном условии, что в итерационном процессе используется обновление. Обновление состоит в том, что после выполнения по методу сопряженных градиентов цикла из n итераций смещение из текущей точки производится в направлении антиградиента.

Из опыта численных расчетов известно, что результативность цикла из n итераций может существенно зависеть от того, сделано ли обновление. Использование антиградиента в качестве начального направления в цикле зачастую снижает эффективность итерации, на которой производится обновление. В связи с этим в [2] на шагах обновления предлагается использовать не направление антиградиента, а некоторое другое направление. Для обоснования этого рассмотрен вариант метода сопряженных градиентов минимизации квадратичной функции, в котором процесс можно начинать с любого направления ρ такого, что $(\rho, \nabla f(x^0)) < 0$ ($\nabla f(x^0)$ – градиент в начальной точке x^0). Ниже предложена модификация этого метода, в котором за счет использования операции контроля за нарушением ортогональности повышается численная устойчивость. Для разъяснения избранного подхода приведем алгоритм из [2].

Итак, рассматривается задача минимизации $f(x) = 1/2(Gx, x) + (b, x)$. Предположим, что G - положительно определена. Пусть дана система линейно-независимых векторов $z^0, \ldots, z^{n-1}$. Используя процесс Грама - Шмидта, из них можно образовать систему сопряженных относительно матрицы G векторов $s^0, \ldots, s^{n-1}$. В качестве z^0 примем вектор ρ такой, что $(\rho, \nabla f(x^0)) < 0$, а остальные $z^i = -\nabla f(x^i)$ (для краткости обозначим $\nabla f^i = \nabla f(x^i)$). Процесс ортогонализации в этом случае дает следующую систему векторов:

$$s^0 = \rho,$$
$$s^1 = -\nabla f^1 + \frac{(\nabla f^0, \nabla f^1 - \nabla f^0)}{(\rho, \nabla f^1 - \nabla f^0)}\rho, \qquad (I)$$
$$s^i = -\nabla f^i + \frac{(\nabla f^i, \nabla f^1 - \nabla f^0)}{(\rho, \nabla f^1 - \nabla f^0)}\rho + \frac{\|\nabla f^i\|^2}{\|\nabla f^{i-1}\|^2} s^{i-1}, \; i > 1.$$

Этот метод отличается от стандартного, в котором, как известно, используется система направлений:

$$s^0 = -\nabla f^0,$$
$$s^i = -\nabla f^i + \beta_i s^{i+1}, \quad \left(\beta_i = \frac{\|\nabla f^i\|^2}{\|\nabla f^{i-1}\|^2}\right),$$

добавкой, учитывающей начальное направление $\rho \neq -\nabla f^0$.

В работах [2,3] предлагается следующая процедура обновления в методе сопряженных градиентов для минимизации функции общего вида. Пусть выполнено n итераций по стандартному алгоритму и получена точка x^n. В качестве вектора ρ предлагается принять $\rho = -\nabla f^n + \beta_n s^{n-1}$, т.е. продолжить вычисления с $\beta_i \neq 0$ и на итерации обновления. Далее,

$$s^{n+1} = -\nabla f^{n+1} + \beta_{n+1} s^n$$

и

$$s^{n+i} = -\nabla f^{n+i} + \beta_{n+i} s^{n+i-1} + \frac{(\nabla f^{n+i}, \nabla f^{n+1} - \nabla f^n)}{(s^n, \nabla f^{n+1} - \nabla f^n)} s^n \quad \text{для } i > 1.$$

В приведенном выше алгоритме из [2] не учитывается влияние распределения собственных значений матрицы G на процесс метода сопряженных градиентов. Известно, что если матрица G имеет m кратных собственных значений, то для минимизации квадратичной функции методом сопряженных градиентов требуется не более чем $n-m+1$ итераций (для того, чтобы это реализовалось на практике, нужна хорошая обусловленность G). Таким образом, в этом случае метод сопряженных градиентов не использует полную систему сопряженных направлений и приведенный выше алгоритм некорректен. Хотя, конечно, процессом ортогонализации Грама – Шмидта можно получить n сопряженных векторов для любой положительно определенной матрицы G . Алгоритм из [2] может не обеспечить сходимость к решению в случаях, когда матрица G имеет $m>1$ кратных (или близких) собственных значений, а обусловленность G такова, что для решения задачи на ЭВМ требуется выполнить более чем $n-m+1$ итераций.

Пусть матрица G имеет m близких собственных значений, а минимизация квадратичной функции производится методом сопряженных градиентов с использованием системы направлений (I). Для вычислений величины смещения α_i вдоль направления S^i определим два варианта:

$$1)\ \alpha_i = \frac{\|\nabla f^i\|^2}{(GS^i, S^i)}, \qquad 2)\ \alpha_i = -\frac{(\nabla f^i, S^i)}{(GS^i, S^i)}.$$

Тогда на практике при применении рассматриваемого алгоритма могут реализоваться два особых случая в зависимости от используемого способа вычисления длины шага вдоль направления. В первом случае после выполнения некоторого числа итераций значение целевого функционала начинает возрастать и алгоритм, как правило, расходится. Во втором случае алгоритм генерирует последовательность точек, которая сходится к неоптимальной точке.

Предположим, что при применении нелинейного метода сопряженных градиентов мы попали в окрестность, в которой функция хорошо описывается квадратичной моделью, а матрица вторых производных имеет группу близких собственных значений. Тогда второй случай дает иллюстрацию особенностей, к которым может привести применение метода сопряженных градиентов с использованием направлений (I). Эти особенности обусловлены тем, что в процессе решения нарушается ортогональность между текущим значением

градиента и вектором ρ, который использовался на итерации обновления.

Алгоритм можно подправить, введя операцию контроля ортогональности векторов ∇f^i и ρ. Если на некоторой итерации выявляется нарушение ортогональности этих векторов, то предлагается на этой же итерации произвести обновление и в качестве ρ принять вектор $-\nabla f^i + \beta_i s^{i-1}$.

Нам неизвестны данные о применении алгоритма из [2] в линейном случае. Для решения систем линейных уравнений с симметричными положительно определенными матрицами разработаны эффективные алгоритмы на основе метода сопряженных градиентов без обновления [4]. Далее проведем сопоставление эффективности стандартного метода с обновлением через n итераций и модифицированного алгоритма [2] на линейных задачах.

Линейный метод сопряженных градиентов состоит в последовательном выполнении следующих вычислений:

$$s^k = -r^k + \beta_k s^{k-1},$$

$$\alpha_k = \frac{\|r^k\|^2}{(Gs^k, s^k)},$$

$$x^{k+1} = x^k + \alpha_k s^k,$$

$$r^{k+1} = r^k + \alpha_k G s^k,$$

$$\beta_{k+1} = \frac{\|r^{k+1}\|^2}{\|r^k\|^2},$$

где $r^k = \nabla f^k = Gx^k + b$. Для реализации стандартного линейного метода сопряженных градиентов на ЭВМ требуется 4 массива размера n. Трудоемкость выполнения одной итерации примерно равна трудоемкости вычисления $n+5$ скалярных произведений.

Реализация алгоритма по схеме (I) требует дополнительно 2 массива памяти размера n. Трудоемкость выполнения одной итерации, после выполнения обновления по схеме (I), примерно равна трудоемкости вычисления $n+8$ скалярных произведений (при этом учтена и операция контроля ортогональности векторов r^k и ρ).

Сопоставление процедур обновления в линейной случае

Задача	Распределение λ_i	Число итераций	$\|x\|_\infty$	$\|r\|_\infty$
I	$\lambda_i = i^2$, $i = \overline{1, 100}$.	700 I75	3xI0^{-5} 4xI0^{-6}	2xI0^{-4} 9xI0^{-8}
2	$\lambda_i = 0.1i^3$, $i = \overline{1, 10}$; $\lambda_i = i^2$, $i = \overline{11, 100}$.	700 284	7xI0^{-1} 5xI0^{-5}	4 4xI0^{-8}
3	$\lambda_i = 0.01i^3$, $i = \overline{1, 10}$; $\lambda_i = i^2$, $i = \overline{11, 100}$.	700 700	I5 2xI0^{-3}	I7 8xI0^{-4}
4	$\lambda_i = 10^3/i^3$, $i = \overline{1, 100}$.	502 458	3xI0^{-5} 2xI0^{-5}	5xI0^{-8} 7xI0^{-8}
5	$\lambda_i = 10^3/i^3$, $i = \overline{1, 50}$; $\lambda_i = 10^{-3}$, $i = \overline{51, 100}$.	295 2I3	2xI0^{-5} IxI0^{-5}	2xI0^{-8} 7xI0^{-8}
6	$\lambda_i = 10^3/i^3$, $i = \overline{1, 50}$; $\lambda_i = 10^{-4}$, $i = \overline{51, 100}$.	422 286	IxI0^{-4} IxI0^{-4}	9xI0^{-8} IxI0^{-7}
7	$\lambda_i = 10^3/i$, $i = \overline{1, 50}$; $\lambda_i = 10^{-3}$ $(i-50)$, $i=\overline{51,100}$	700 6I0	IxI0^{-4} 7xI0^{-5}	2xI0^{-6} 9xI0^{-8}
8	$\lambda_i = 10^3/i$, $i = \overline{1, 50}$; $\lambda_i = 10^{-3}(i-50)^3$, $i=\overline{51, 100}$.	700 4I7	3 5xI0^{-6}	6xI0^{-2} 9xI0^{-8}
9	$\lambda_i = (i-1)+0,1i^2$, $i=\overline{1,100}$	40I I29	5xI0^{-6} 5xI0^{-6}	9xI0^{-8} 6xI0^{-8}
I0	$\lambda_i = 10^3 i$, $i = \overline{1, 30}$; $\lambda_i = 0.01(1 + 0,03(i-31))$, $i = \overline{31, 100}$.	200 I9I *700	2xI0^{-4} 2xI0^{-4} 54	2xI0^{-8} IxI0^{-7} 397

Примечание. Для каждой задачи первая строка результатов решения соответствует алгоритму *GG*, вторая – *GG1*; * – решение задачи алгоритмом *GG1* без контроля ортогональности векторов z^k и ρ.

В таблице приведены данные по решению задач размерности $n = 100$ на ЭВМ БЭСМ-6. Стандартному алгоритму в таблице дано наименование CG, а алгоритму, соответствующему схеме (I), - $CG1$.

Формирование задач основывалось на использовании матрицы W, столбцы которой представляют ортонормированную систему векторов. Матрица G, фигурирующая в задачах, строилась из определенного распределения собственных значений λ_i и $G = W\lambda W^T$, где λ - диагональная матрица с диагональными элементами λ_i. Таким образом, задачи состояли в минимизации квадратичных форм (Gx, x) или, что одно и то же, в решении однородных систем $Gx = 0$. Вычисления во всех задачах начинались с точки $x_i^0 = 10, i = 1, \ldots, 100$. В качестве критерия окончания решения использовалось условие $\|r^k\|_\infty < 10^{-7}$. Решение задачи также заканчивалось после выполнения 700 итераций.

Для контроля за нарушением ортогональности векторов r^i и p в алгоритме $CG1$ вычислялись значения $\varepsilon_i = |\cos\angle(r^i, p)|$ и сопоставлялись с заранее заданной величиной ε. Значение ε принималось равным $\sqrt{n \cdot \varepsilon m}$, где n - размерность задачи, а εm - машинный эпсилон (для ЭВМ БЭСМ-6 $\sqrt{\varepsilon m} \approx 4{,}5 \times 10^{-13}$). По-видимому, при определении значения параметра ε размерность задачи необязательно учитывать, пока n не слишком велико. Наши численные расчеты для разных задач размерности $n \leq 100$ подтверждают это предположение. (Данные таблицы практически не изменятся, если принять $\varepsilon = \sqrt{\varepsilon m}$.).

Если в алгоритме $CG1$ на некотором шаге оказывается, что $\varepsilon_i > \varepsilon$, то на этой итерации производится обновление. Кроме того, в $CG1$ контролируется число итераций ℓ, которые выполняются между очередными итерациями обновления. Пусть на некоторой итерации алгоритма $CG1$ оказалось, что $\varepsilon_i > \varepsilon$, но после последнего обновления произведено ℓ итераций и ℓ меньше некоторого критического значения ℓ_0, то в $CG1$ обновление производится не по схеме (I) а по стандартному методу (в программе $\ell_0 = 3$).

Проведем анализ некоторых данных из таблицы. В задаче I число обусловленности $C = 10^4$, и матрицу G можно считать хорошо обусловленной для применения метода сопряженных градиентов, но G имеет распределение λ_i, неудобное для стандартного метода: все значения λ_i существенно различаются

друг от друга. Для целей анализа вычислительного процесса на итерациях алгоритма CG вычислялись величины $\delta_k = |\cos < r^k, r^{k-1} >|$ для $k > 1$. При решении задачи I по стандартному алгоритму максимальное $\delta_k \approx 10^{-10}$. Преимущества алгоритма $CG1$ в этой задаче сохраняются при произвольном выборе параметра $\varepsilon > 10^{-10}$. Заметим, что максимальное δ_k и для задачи 3 примерно равно 10^{-10}.)

Распределение λ_i в задачах 2 и 3 отличается от распределения собственных значений матрицы G в задаче I тем, что произведены изменения значений для небольшой группы λ_i в сторону их уменьшения. Аналогичным образом можно построить другие примеры задач, исходя из других распределений λ_i, которые не решаются по алгоритму CG, а алгоритм $CG1$ позволяет получать решения с приемлемой точностью.

В задаче 4 ($\lambda_i = 10^3/i^3, i = 1, \ldots, 100$) более половины $\lambda_i \approx 10^{-3}$, причем величины $\delta\lambda_{ij} = |\lambda_i - \lambda_j|$ при $i \neq j$ для многих из них имеют значения, примерно равные 10^{-5}. Видимо, здесь более уместно рассматривать величины $\delta\lambda_{ij}/c$, которые в нашем случае достигают значений порядка 10^{-11} и близки к уровню ошибок округления на ЭВМ БЭСМ-6. При решении задачи по стандартному методу сопряженных градиентов параметр δ_k изменяется в пределах от 10^{-11} до 10^{-5}. Число итераций, необходимое для решения этой задачи по алгоритму $CG1$, существенно зависит от выбора значения параметра ε. При $\varepsilon = 10^{-4}$ для получения приближения к решению с заданной точностью алгоритм требует выполнения 592 итераций, а при $\varepsilon = 10^{-6}$ – 434 итераций. Без операций контроля ортогональности векторов для получения решения с заданной точностью в $CG1$ требуется выполнить 648 итераций.

Решение задач 5,6 и 8 по алгоритму $CG1$ без использования контроля за нарушением ортогональности получается за несколько меньшее количество итераций, чем указано в таблице. В задаче 7, наоборот, исключение операций контроля ведет к небольшому возрастанию числа итераций.

Задача 10 служит контрпримером для алгоритма из работы [2]. В таблице для этой задачи приведены три строки результатов. Первая и вторая строки соответствуют алгоритмам $CG1$ и $CG1$, а третья – использованию системы направлений (I) без контроля ортогональности векторов r^k и p. В последнем слу-

чае имеет место расходимость вычислительного процесса.

Данные таблицы наглядно показывают полезность применения метода сопряженных градиентов на основе (I) в линейном случае. Использование антиградиента в качестве направления спуска на итерациях обновления при решении плохо обусловленных задач может приводить к тому, что в разных циклах из n итераций формируются близкие системы векторов направления. Схема (I) позволяет получать большее разнообразие направлений и нам представляется, что корректное использование этого алгоритма позволяет несколько продвинуться вперёд в части решения плохо обусловленных задач.

В заключение вернемся к вопросу применения схемы (I) в нелинейном методе сопряженных градиентов. В работе [3] достаточно детально рассматривался этот вопрос. В частности, там предлагалось на итерациях контролировать ортогональность ∇f^k и ∇f^{k+1} Если условие ортогональности не выполняется с заданной точностью, то предлагалось производить обновление, однако возможность нарушения ортогональности векторов объяснялась только за счет того, что функция не является квадратичной.

Наши рассмотрения показывают, что в нелинейный метод сопряженных градиентов необходимо дополнительно включить контроль точности выполнения условия $(\nabla f^k, \rho) = 0$ внутри самой схемы (I). Следовательно, на итерациях должно контролироваться выполнение двух соотношений: $|\cos < \nabla f^k, \nabla f^{k+1} >| \le \delta$ и $|\cos < \nabla f^k, \rho >| \le \delta_1$. Обновление производится, если нарушается одно из этих условий или после последнего обновления выполнено n итераций. Значения величин δ и δ_1 должны существенно различаться. Параметр δ нельзя выбирать слишком малым, так как это может привести к частным обновлениям по стандартному методу и падению скорости сходимости. Значение δ_1 должно быть достаточно малым, потому что, как видно из анализа численных расчетов в линейном случае, необходимо строго контролировать ортогональность векторов f^k и ρ. Кроме того, нужно следить за количеством итераций между очередными моментами обновления, которые выполняются по схеме (I). Если такая проверка не предусмотрена, то возможно вырождение алгоритма, которое состоит в том, что используются направления (I), а обновление производится на каждой итерации. Последнее эквивалентно применению стандартного метода сопряженных градиентов без обновления. Таким образом, в нели-

нейном алгоритме необходимо предусмотреть обновление по стандартному методу, если по схеме (I) выполняется ℓ итераций между очередными моментами обновления и ℓ меньше критического значения ℓ_0 .

При численной проверке нелинейного метода на тестовых задачах нами использовались значения $\delta = 0{,}5$ и $\delta_1 = 0{,}01$. Численные эксперименты показали высокую эффективность рассматриваемого алгоритма. Для некоторых тестовых задач наблюдалось сокращение числа итераций по сравнению со стандартным методом в 2–3 раза.

Анализ расчетов показывает, что типичным для применения нелинейного метода является случай, когда вначале вычислительного процесса обновления проводятся стандартным образом. И только после того, как точки оптимизирующей последовательности попадают в область, в которой целевая функция хорошо приближается квадратичной моделью, используются направления (I).

Автор выражает благодарность А.А.Каплану за полезные обсуждения.

Литература

I. Поляк Б.Т. Введение в оптимизацию. – М.: Наука, I983.

2. Beale E.M.L. A derivation of conjugate gradients in numerical methods for non-linear optimization. – London – New York: Academic Press, I972. – P.39–43.

3. Powell M.J.D. Restart procedures for the conjugate-gradient method// Math. Prog.– I977. N I2. – P.24I–254.

4. Уилкинсон Дж., Райнш К. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. – М.: Машиностроение, I976.

Поступила в ред.-изд. отдел
0I.07.I989 г.