线性递推关系与矩阵乘法

致远学院 2011 级计算机科学班
郭晓旭　杨宽

March 2, 2013

Abstract

对于一般的具有常系数线性递推关系的递推数列，若需要很快算出某一项的精确值，一般的方法是求出特征方程的解然后解出这个递推关系的通项公式。可是随着递推关系阶数的升高，解特征方程的难度也逐渐增大，甚至在递推关系阶数大于 5 之后，特征方程的次数随之超过 5，根本没有代数解法。本文利用矩阵乘法，提出了一个在 $O(k^3\lceil\log n\rceil)$ 的时间复杂度内算出 k 阶常系数线性递推数列第 n 项的精确值的算法，并利用转移矩阵和特征方程的联系，把这个算法的时间复杂度优化到了 $O(k^2\lceil\log n\rceil)$.

1　常系数线性递推关系的特征方程解法

1.1　常系数线性递推关系

对于一个数列 $\{a_n\}$，如果 $\{a_n\}$ 满足一个递推关系:

$$a_n = c_k a_{n-1} + c_{k-1}a_{n-2} + \cdots + c_1 a_{n-k} + c_0 \qquad \forall n \in \mathbb{N}, n > k$$

其中 $\forall i \in \mathbb{N}, i \le k, c_i$ 是常数.
则称数列 $\{a_n\}$ 满足 k 阶常系数线性递推关系。
特别地，若 $c_0 = 0$，则称 $\{a_n\}$ 满足 k 阶常系数线性齐次递推关系。

1.2　齐次线性递推关系的特征方程

设数列 $\{h_n\}(n \ge 0)$, 且存在 $a_1, a_2, \ldots, a_k$ 满足：

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + \cdots + a_k h_{n-k}$$

定理 **1.1**：令 q 为非零常数，则 $h_n = q^n$ 是常系数线性齐次递推关系

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + \cdots + a_k h_{n-k}$$

的解的充分必要条件是：q 是多项式方程 $x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k = 0$ 的一个根。该多项式方程称为对应的递推关系的特征方程，q 称为特征方程的一个特征根。
若该多项式方程有 k 个不同的特征根 $q_1, q_2, q_3, \cdots, q_k$, 则：

$$h_n = c_1 q_1^n + c_2 q_2^n + c_3 q_3^n + \cdots + c_k q_k^n$$

是下述意义下的一般解:
无论给定 $h_0, h_1, h_2, \ldots, h_{k-1}$ 什么初始值，都存在常数 $c_1, c_2, c_3, \ldots, c_k$ 使得该通项公式是满足其递推关系和初始条件的唯一序列。

然而当某个递推关系的特征方程有重根时，我们会发现无法应用以上定理求得该数列的通项公式，这时候我们需要将定理加强为有重根的形式：

定理 **1.2**：令 q 为非零常数，则 $h_n = q^n$ 是常系数线性齐次递推关系

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + \cdots + a_k h_{n-k}$$

的特征方程中的 s 个等根，设其余部分特征根的一般解为 T_n，则：

$$c_1 q^n + c_2 n q^n + c_3 n^2 q^n + \cdots + c_s n^{s-1} q^n + T_n$$

是原递推关系的一般解。

1.3 非齐次线性递推关系的特解和通项公式

在常系数线性递推关系：

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + a_3 h_{n-3} + \cdots + a_k h_{n-k} + b_n$$

中，如果 $b_n \neq 0$ 为常数，那么这个递推关系称为常系数线性非齐次递推关系。
事实上，若或 b_n 是与 n 有关的函数这个递推关系也是可以解出通项公式的。

定理 **1.3**：常系数线性非齐次递推关系：

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + a_3 h_{n-3} + \cdots + a_k h_{n-k} + b_n$$

的一般解可以写成如下形式：

$$h_n = T_n + p_n$$

其中 T_n 是递推关系

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + a_3 h_{n-3} + \cdots + a_k h_{n-k}$$

的一般解。
而 p_n 是一个常数 (或关于 n 的函数)，满足

$$p_n = a_1 p_{n-1} + a_2 p_{n-2} + a_3 p_{n-3} + \cdots + a_k p_{n-k} + b_n$$

称为原递推关系的一个特解。
若 b_n 为常数，则 p_n 为常数或 n 的一次多项式，这取决于方程

$$p_n = \sum_{i=1}^{k} a_i \cdot p_n + b_n$$

是否有解。

一般地，p_n 可以由待定系数法求得。常见的 b_n 和特解 p_n 的对应关系如下：

1. b_n 是 n 的 k 次多项式，那么特解 p_n 也应当是 n 的 k 次多项式；
2. b_n 是 n 的指数形式，那么 p_n 是 n 的多项式与指数形式的乘积，例如 $b_n = d^n$，那么 p_n 应当具有 $r \cdot d^n$ 或 $r \cdot n \cdot d^n$ 的形式。

2 利用矩阵乘法计算递推数列的某一项

2.1 构造递推矩阵

设数列 $\{h_n\}$ 满足 k 阶常系数线性递推关系：

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + a_3 h_{n-3} + \cdots + a_k h_{n-k}$$

构造矩阵

$$\mathbf{M} = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_{k-2} & a_{k-1} & a_k \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}_{k \times k}$$

与初始值向量

$$\mathbf{X} = \begin{pmatrix} h_{k-1} \\ h_{k-2} \\ h_{k-3} \\ \vdots \\ h_2 \\ h_1 \\ h_0 \end{pmatrix}_{k \times 1}$$

易见

$$\mathbf{Y}=\mathbf{MX}=\begin{pmatrix}a_1&a_2&a_3&\cdots&a_{k-2}&a_{k-1}&a_k\\1&0&0&\cdots&0&0&0\\0&1&0&\cdots&0&0&0\\0&0&1&\cdots&0&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&\cdots&1&0&0\\0&0&0&\cdots&0&1&0\end{pmatrix}\begin{pmatrix}h_{k-1}\\h_{k-2}\\h_{k-3}\\\vdots\\h_2\\h_1\\h_0\end{pmatrix}=\begin{pmatrix}h_k\\h_{k-1}\\h_{k-2}\\\vdots\\h_3\\h_2\\h_1\end{pmatrix}$$

对于非齐次的线性递推关系

$$h_n=a_1h_{n-1}+a_2h_{n-2}+a_3h_{n-3}+\cdots+a_kh_{n-k}+b_n$$

若 b_n 为常数，则构造转移矩阵：

$$\mathbf{M}=\begin{pmatrix}a_1&a_2&a_3&\cdots&a_{k-1}&a_k&b_n\\1&0&0&\cdots&0&0&0\\0&1&0&\cdots&0&0&0\\0&0&1&\cdots&0&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&\cdots&1&0&0\\0&0&0&\cdots&0&0&1\end{pmatrix}_{(k+1)\times(k+1)}$$

与初始向量

$$\mathbf{X}=\begin{pmatrix}h_{k-1}\\h_{k-2}\\h_{k-3}\\\vdots\\h_2\\h_1\\h_0\\1\end{pmatrix}_{(k+1)\times1}$$

易见

$$\mathbf{Y}=\mathbf{MX}=\begin{pmatrix}a_1&a_2&a_3&\cdots&a_{k-1}&a_k&b_n\\1&0&0&\cdots&0&0&0\\0&1&0&\cdots&0&0&0\\0&0&1&\cdots&0&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&\cdots&1&0&0\\0&0&0&\cdots&0&0&1\end{pmatrix}\begin{pmatrix}h_{k-1}\\h_{k-2}\\h_{k-3}\\\vdots\\h_2\\h_1\\h_0\\1\end{pmatrix}=\begin{pmatrix}h_k\\h_{k-1}\\h_{k-2}\\\vdots\\h_3\\h_2\\h_1\\1\end{pmatrix}$$

事实上我们可以发现，对于任意的 k 阶常系数线性递推关系，我们总可以构造一个 $k\times k$ 或 $(k+1)\times(k+1)$ 的转移矩阵 M, 对于初始值向量

$$\mathbf{X}=\begin{pmatrix}h_{k-1}\\h_{k-2}\\h_{k-3}\\\vdots\\h_2\\h_1\\h_0\end{pmatrix}_{k*1}\quad or\begin{pmatrix}h_{k-1}\\h_{k-2}\\h_{k-3}\\\vdots\\h_1\\h_0\\b_n\end{pmatrix}_{(k+1)*1}$$

使得 $\mathbf{Y}=\mathbf{M}^{n-k+1}\mathbf{X}$，$\mathbf{Y}$ 第一行第一列的元素恰好为 h_n。

2.2 矩阵乘法的快速幂

根据以上分析，我们可以知道如果已知 $h_0, h_1, h_2, \ldots, h_k$ 和递推关系，对于任意 $n \geq k$，我们都可以很快计算出 h_n 的值。
我们计算一下这个算法的时间复杂度，对于 $k \times k$ 的矩阵，一次矩阵乘法的时间复杂度为 $O(k^3)$，我们一共需要做 $n-k$ 次矩阵乘法，于是整个算法的时间复杂度为 $O((n-k)k^3)$.
可以看到，整个算法的时间瓶颈在于矩阵乘法的次数，当 n 很大时，这个算法就完全失去了优势，那么对算法的优化势必先从矩阵乘法的次数入手。

定理 **2.1**：矩阵乘法满足结合律.

定理 **2.2**：设 $\mathbf{A}$ 为一 $k \times k$ 的矩阵，$n \in \mathbb{N}$，则有如下公式成立：

$$\mathbf{A}^n = \begin{cases} (\mathbf{A}^{\frac{n}{2}})^2 & n \text{ 为偶数} \\ (\mathbf{A}^{\frac{n-1}{2}})^2\mathbf{A} & n \text{ 为奇数} \end{cases}$$

显然定理 2.2 是定理 2.1 的直接推论。
于是我们可以利用 $\mathbf{A}^{\frac{n}{2}}$ 的结果做一次自乘得到 $\mathbf{A}^n$ 的结果。
这个算法称为快速幂。

2.3 时间复杂度计算

现在我们可以考虑这个算法的时间复杂度了，只需要计算快速幂算法一共做了多少次矩阵乘法即可。

定理 **2.3**：设 $\mathbf{A}$ 为任意矩阵，$n \in \mathbb{N}$，则快速幂算法计算 $\mathbf{A}^n$ 至多需要 $(\lfloor \log n \rfloor + 1) \times 2$ 次矩阵乘法。

定理的证明非常简单，直接利用数学归纳法即可。
现在我们已经从计算次数的角度上对算法进行了优化，那么是否可以考虑从矩阵乘法本身的复杂度上对算法进行优化呢？因为事实上 $O(k^3)$ 并不是矩阵乘法时间复杂度的下限，而且这类问题中转移矩阵的形式又很特殊，于是答案是显然存在优化方法。

3 矩阵乘法的算法优化

考虑矩阵 $\mathbf{M}$ 的特征多项式，$f(\lambda) = |\lambda\mathbf{E} - \mathbf{M}| = \begin{pmatrix} \lambda - a_1 & -a_2 & a_3 & \cdots & -a_{k-2} & -a_{k-1} & -a_k \\ -1 & \lambda & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & \lambda & \cdots & 0 & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & \lambda & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & \lambda \end{pmatrix}_{k \times k}$，
按第一行展开，得 $f(\lambda) = \lambda^k - a_1\lambda^{k-1} - a_2\lambda^{k-2} - \cdots - a_k$.

根据 Cayley-hamilton 定理，$f(\lambda)$ 是化零多项式，即有 $f(\mathbf{M}) = \mathbf{0}$.

下面我们断言，对于 $\forall i$，$\mathbf{M}^i$ 都可以表示成 $\mathbf{E}, \mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^{k-1}$ 的线性组合。

当 $0 \leq i \leq k-1$ 时，结论显然。

用数学归纳法，当 $i = k$ 时，因为 $f(\mathbf{M}) = \mathbf{0}$，即有 $\mathbf{M}^k = a_1\mathbf{M}^{k-1} + a_2\mathbf{M}^{k-2} + \cdots + a_k\mathbf{E}$，所以结论成立。

现在假设当 $i < k_0$ 时结论成立，于是当 $i = k_0$ 时，我们取 $1 \leq j \leq i-1$，从而 $\mathbf{M}^i = \mathbf{M}^j\mathbf{M}^{i-j}$, 因为 $\mathbf{M}^j$ 和 $\mathbf{M}^{i-j}$ 都可以由 $\mathbf{E}, \mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^{k-1}$ 的线性组合得到，所以 $\mathbf{M}^i$ 是 $\mathbf{E}, \mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^{2k-2}$ 的线性组合。

注意到 $f(\mathbf{M}) = 0$ 可以得到 $\mathbf{M}^i f(\mathbf{M}) = 0$，展开来写即是

$$\mathbf{M}^{i+k} = \sum_{j=1}^{k} a_j \mathbf{M}^{i+k-j}$$

，也就是说，$\mathbf{M}^{i+k}$ 可以表示为 $\mathbf{M}^i, \mathbf{M}^{i+1}, \ldots, \mathbf{M}^{i+k-1}$ 的线性组合。反复利用这个式子，结论就证到了。

容易观察到，上面证明的过程也给出了求线性组合的方法，而这个过程实质上是一个多项式的乘法，因为多项式乘法也满足结合律，不妨利用和上面类似的技术，取 $j = \lceil i/2 \rceil$，于是只需要 $O(\log n)$ 次乘法即可，每次乘法的代价是 $O(k^2)$，总的复杂度是 $O(k^2 \log n)$.

此时离最终结果尚有一步之遥，我们知道 $\mathbf{M}^n = b_1\mathbf{M}^{k-1} + b_2\mathbf{M}^{k-2} + \cdots + b_k\mathbf{E}$，两端右乘 $\mathbf{X}$，于是有 $\mathbf{M}^n\mathbf{X} = b_1\mathbf{M}^{k-1}\mathbf{X} + b_2\mathbf{M}^{k-2}\mathbf{X} + \cdots + b_k\mathbf{X}$，而我们又有 $\mathbf{M}^i\mathbf{X} = \begin{pmatrix} h_{i+k-1} \\ h_{i+k-2} \\ \vdots \\ h_i \end{pmatrix}$，直接求得 $h_0, h_1, \ldots, h_{2k-2}$ 的值代入即可。

至此，问题完整解决。