Mailman矩阵向量相乘算法

朱志翔

2019/5/3

注：本文的向量符号形式 $\mathbf{v}$ 均表示列向量，$\mathbf{v}^T$ 均表示行向量。

1 Mailman算法适用的场景

Mailman算法解决的是矩阵 $\mathbf{A}$ 与向量 $\mathbf{x}$ 相乘的问题。其中 $\mathbf{A}$ 为 m 行 n 列，且其元素必须来自于某个有限整数集 Σ。$\mathbf{x}$ 的长度为 n，其元素可以为任意实数。

若是采用普通的矩阵乘法，$\mathbf{Ax}$ 包含 mn 个乘法与 $mn-m$ 个加法，因此时间复杂度是 $O(mn)$。

Mailman 可以将复杂度优化至 $O(mn/\log(\max\{m,n\}))$。

不过这个优化的复杂度也是有前提的。这是因为Mailman算法要对 $\mathbf{A}$ 做预处理，将其分解为两个矩阵的乘积 $\mathbf{UP}$，然后再求 $\mathbf{U}(\mathbf{Px})$。Mailman可以将 $\mathbf{U}(\mathbf{Px})$ 的时间复杂度优化至 $O(mn/\log(\max\{m,n\}))$，但是 $\mathbf{A}=\mathbf{UP}$ 预处理的时间复杂度仍然是 $O(mn)$。因此只有在 $\mathbf{A}$ 已经预先被分解为 $\mathbf{U}$ 和 $\mathbf{P}$，或是我们要将 $\mathbf{A}$ 与许多向量相乘的时候，Mailman的优势才能体现出来。

2 从一个简单的例子开始说起

为了便于说明，我们先构造一个简单的例子，并且在这个例子中 $n=|\Sigma|^m$。假设 $\Sigma=\{0,1\}$，$S=|\Sigma|=2$。又假设矩阵 $\mathbf{A}$ 为

$$\mathbf{A}=\begin{pmatrix}1&1&1&1&0&1&1&0\\0&1&1&0&1&0&1&0\\0&0&1&1&1&0&1&1\end{pmatrix}。$$

可见 $m=3$，$n=S^m=8$。

接着假设向量 $\mathbf{x}$ 为

$$\mathbf{x}=\begin{pmatrix}0.24\\0.50\\-1.72\\-0.08\\0.71\\-2.81\\-1.50\\0.31\end{pmatrix}。$$

我们可以很容易用普通的矩阵乘法算出

$$\mathbf{Ax}=\begin{pmatrix}-5.37\\-2.01\\-2.28\end{pmatrix}。$$

接下来讲述如何用Mailman算法算出这一结果。

2.1 预处理

首先要对 $\mathbf{A}$ 做预处理，将其分解为两个矩阵 $\mathbf{U}_n$ 和 $\mathbf{P}$ 的乘积。

$$\mathbf{A}=\mathbf{U}_n\mathbf{P}=\begin{pmatrix}0&0&0&0&1&1&1&1\\0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\end{pmatrix}\begin{pmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\1&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&1&0\end{pmatrix}$$

其中 $\mathbf{U}_n$ 陈列了长度为 m 且元素取自 Σ 的列向量的所有可能性。 显然其列数为 $n=|\Sigma|^m=S^m$。 $\mathbf{P}$ 里面为1的元素只有n个，按列排序分别是$p_{5,1}$、$p_{7,2}$、$p_{8,3}$、$p_{6,4}$、$p_{4,5}$、$p_{5,6}$、$p_{8,7}$和$p_{2,8}$。 可以看出每列有且只有一个元素为1，其余元素均为0。 另外还可以看出，当且仅当$\mathbf{U}^{(i)}=\mathbf{A}^{(j)}$，即$\mathbf{U}$的第$i$列与$\mathbf{A}$的第$j$列相同时，$p_{i,j}=1$。

2.2 计算 $\mathbf{Ax}$

因为 $\mathbf{P}$ 有且只有 n 个值为1的元素，其余元素都为0，所以我们可以用不超过 n 个加法算得

$$\mathbf{Px}=\begin{pmatrix}0\\p_{2,8}x_8\\0\\p_{4,5}x_5\\p_{5,1}x_1+p_{5,6}x_6\\p_{6,4}x_4\\p_{7,2}x_2\\p_{8,3}x_3+p_{8,7}x_7\end{pmatrix}=\begin{pmatrix}0\\0.31\\0\\0.71\\0.24-2.81\\-0.08\\0.5\\-1.72-1.5\end{pmatrix}=\begin{pmatrix}0\\0.31\\0\\0.71\\-2.57\\-0.08\\0.5\\-3.22\end{pmatrix}$$

注意观察 $\mathbf{U}_n$ 的结构。 我们将 $\mathbf{U}_n$ 按纵向拆分为 S 个子矩阵。 接着将每个子矩阵横向拆分为第1行与第2至m行分别组成的上下两个子矩阵。

$$\left(\begin{array}{cccc:cccc}0&0&0&0&1&1&1&1\\\hdashline 0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\end{array}\right)$$

可以看出 $\mathbf{U}_n$ 的第1行由 S 个子向量组成的，其中第 i 个子向量的元素全都来自 Σ 的第 i 个元素，长度为 n/S。 $\mathbf{U}_n$ 的第2至 m 行组成的 S 个子矩阵都是一样的。 于是 $\mathbf{U}_n$ 的结构可以表示为

$$\mathbf{U}_n=\left(\begin{array}{c:c:c}\sigma_1\mathbf{1}^T_{n/S}&\cdots&\sigma_S\mathbf{1}^T_{n/S}\\\hdashline \mathbf{U}_{n/S}&\cdots&\mathbf{U}_{n/S}\end{array}\right)$$

我们将 $\mathbf{Px}$ 记为 $\mathbf{z}^n$，又将 $\mathbf{z}^n$ 拆分为 S 个长度为 n/S 的子向量 $\mathbf{z}_1^{n/S}$、$\mathbf{z}_2^{n/S}$、……、$\mathbf{z}_S^{n/S}$。 于是 $\mathbf{U}_n\cdot\mathbf{Px}=\mathbf{U}_n\mathbf{z}^n$ 的计算过程可以化为

$$\mathbf{U}_n = \left(\begin{array}{c:c:c} \sigma_1 \mathbf{1}_{n/S}^T & \cdots & \sigma_S \mathbf{1}_{n/S}^T \\ \hdashline \mathbf{U}_{n/S} & \cdots & \mathbf{U}_{n/S} \end{array}\right) \cdot \begin{pmatrix} \mathbf{z}_1^{n/S} \\ \mathbf{z}_2^{n/S} \\ \vdots \\ \mathbf{z}_S^{n/S} \end{pmatrix} = \left(\begin{array}{c} \sigma_1 \mathbf{1}_{n/S}^T \mathbf{z}_1^{n/S} + \sigma_2 \mathbf{1}_{n/S}^T \mathbf{z}_2^{n/S} + \cdots + \sigma_S \mathbf{1}_{n/S}^T \mathbf{z}_S^{n/S} \\ \hdashline \mathbf{U}_{n/S}(\mathbf{z}_1^{n/S} + \mathbf{z}_2^{n/S} + \cdots + \mathbf{z}_S^{n/S}) \end{array}\right)$$

代入本节简单例子里的具体数值，即为

$$\left(\begin{array}{c} 0 \cdot \mathbf{1}^T \begin{pmatrix} 0 \\ 0.31 \\ 0 \\ 0.71 \end{pmatrix} + 1 \cdot \mathbf{1}^T \begin{pmatrix} -2.57 \\ -0.08 \\ 0.5 \\ -3.22 \end{pmatrix} \\ \hdashline \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \cdot \left(\begin{pmatrix} 0 \\ 0.31 \\ 0 \\ 0.71 \end{pmatrix} + \begin{pmatrix} -2.57 \\ -0.08 \\ 0.5 \\ -3.22 \end{pmatrix} \right) \end{array}\right)$$

$\sigma_1 \mathbf{1}_{n/S}^T \mathbf{z}_1^{n/S} + \sigma_2 \mathbf{1}_{n/S}^T \mathbf{z}_2^{n/S} + \cdots + \sigma_S \mathbf{1}_{n/S}^T \mathbf{z}_S^{n/S}$ 的计算包含不超过 n 个加法与 n 个乘法，即 $2n$ 个四则运算。 代入本节$\Sigma = \{0, 1\}$，$S = |\Sigma| = 2$的例子则是

$$\begin{aligned} & \sigma_1 \mathbf{1}_{n/2}^T \mathbf{z}_1 + \sigma_2 \mathbf{1}_{n/2}^T \mathbf{z}_2 \\ =& 0 \cdot \mathbf{1}_{n/2}^T \mathbf{z}_1 + 1 \cdot \mathbf{1}_{n/2}^T \mathbf{z}_2 \\ =& 0 \times (0 + 0.31 + 0 + 0.71) + 1 \times (-2.57 - 0.08 + 0.5 - 3.22) \\ =& -5.37。 \end{aligned}$$

可以看出 $\mathbf{Ax}$ 的第1个元素已经计算完成。 为了计算该元素，我们使用了不超过 n 个加法与 n 个乘法，即 $2n$ 个四则运算。

$\mathbf{z}_1^{n/S} + \mathbf{z}_2^{n/S} + \cdots + \mathbf{z}_S^{n/S}$ 的计算可以在不超过 n 个加法内完成。 代入本节 $n = 8$，$S = 2$ 的例子则是

$$\mathbf{z}_1^{8/2} + \mathbf{z}_2^{8/2} = \begin{pmatrix} 0 - 2.57 \\ 0.31 - 0.08 \\ 0 + 0.5 \\ 0.71 - 3.22 \end{pmatrix} = \begin{pmatrix} -2.57 \\ 0.23 \\ 0.5 \\ -2.51 \end{pmatrix}$$

将 $\mathbf{z}_1^{n/S} + \mathbf{z}_2^{n/S} + \cdots + \mathbf{z}_S^{n/S}$ 的结果记为 $\mathbf{z}^{n/S}$。 接下来要计算 $\mathbf{U}_{n/S}\mathbf{z}^{n/S}$ 的结果。

继续观察 $\mathbf{U}_{n/S}$ 的结构，会发现其结构和 $\mathbf{U}_n$ 类似。 可见 $\mathbf{U}_n$ 是递归结构。 于是仿照 $\mathbf{U}_n$ 将 $\mathbf{U}_{n/S}$ 按纵向拆分为 S 个子矩阵，再将每个子矩阵横向拆分为第1行与第2至 $m-1$ 行组成的上下两个子矩阵：

$$\mathbf{U}_{n/S} = \mathbf{U}_{8/2} = \mathbf{U}_4 = \left(\begin{array}{cc:cc} 0 & 0 & 1 & 1 \\ \hdashline 0 & 1 & 0 & 1 \end{array}\right) = \left(\begin{array}{c:c:c} \sigma_1 \mathbf{1}_{n/S^2}^T & \cdots & \sigma_S \mathbf{1}_{n/S^2}^T \\ \hdashline \mathbf{U}_{n/S^2} & \cdots & \mathbf{U}_{n/S^2} \end{array}\right)$$

$\mathbf{z}^{n/S}$ 也可以拆分为 S 个长度为 n/S^2 的子向量 $\mathbf{z}_1^{n/S^2}$、$\mathbf{z}_2^{n/S^2}$、……、$\mathbf{z}_S^{n/S^2}$。 于是

$$\mathbf{U}_{n/S}\mathbf{z}^{n/S} = \left(\begin{array}{c:c:c} \sigma_1 \mathbf{1}_{n/S^2}^T & \cdots & \sigma_S \mathbf{1}_{n/S^2}^T \\ \hdashline \mathbf{U}_{n/S^2} & \cdots & \mathbf{U}_{n/S^2} \end{array}\right) \begin{pmatrix} \mathbf{z}_1^{n/S^2} \\ \mathbf{z}_2^{n/S^2} \\ \vdots \\ \mathbf{z}_S^{n/S^2} \end{pmatrix}。$$

代入本节的例子则有

$$\mathbf{U}_4\mathbf{z}^4 = \left(\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 \end{array}\right)\left(\begin{array}{c} -2.57 \\ 0.23 \\ \hline 0.5 \\ -2.51 \end{array}\right)。$$

与上文类似，$\mathbf{U}_{n/S}$ 第1行与 $\mathbf{z}^{n/S}$ 的乘积就是 $\mathbf{Ax}$ 第2个元素的值：

$$\left(\begin{array}{cc|cc} 0 & 0 & 1 & 1 \end{array}\right)\left(\begin{array}{c} -2.57 \\ 0.23 \\ \hline 0.5 \\ -2.51 \end{array}\right) = -2.01$$

这一步所需的运算不超过 n/S 个加法与 n/S 个乘法，即 $2n/S$ 个四则运算。

接着计算 $\mathbf{U}_{n/S^2} \cdot (\mathbf{z}_1^{n/S^2} + \mathbf{z}_2^{n/S^2} + \cdots + \mathbf{z}_S^{n/S^2}) = \mathbf{U}_{n/S^2} \cdot \mathbf{z}^{n/S^2}$ 。 $\mathbf{z}^{n/S^2}$ 的计算可以在不超过 n/S 个加法内完成。在本节的例子中，

$$\mathbf{z}^{n/S^2} = \mathbf{z}^{8/2^2} = \mathbf{z}^2 = \mathbf{z}_1^2 + \mathbf{z}_2^2 = \begin{pmatrix} -2.57 \\ 0.23 \end{pmatrix} + \begin{pmatrix} 0.5 \\ -2.51 \end{pmatrix} = \begin{pmatrix} -2.07 \\ -2.28 \end{pmatrix}。$$

$\mathbf{U}_{n/S^2} = \mathbf{U}_{8/2^2} = \mathbf{U}_2$ 已经只剩一行，且正好就是 Σ。 到了这一步已无需再继续递归，直接用普通的矩阵乘法得出

$$\mathbf{U}_{n/S^2} \cdot \mathbf{z}^{n/S^2} = \mathbf{U}_2 \cdot \mathbf{z}^2 = \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} -2.07 \\ -2.28 \end{pmatrix} = -2.28。$$

至此，$\mathbf{Ax}$ 的最后一个元素也已计算完毕。

2.3 时间复杂度

从上面的例子可以看出，计算 $\mathbf{U}_n\mathbf{z}^n$ 的方法如下：

1. 计算向量乘积 $\mathbf{U}_n(1, 1 \ldots n) \cdot \mathbf{z}^n$ 。
2. 将 $\mathbf{z}^n$ 拆分为 $\mathbf{z}_1^{n/S}$ 、$\mathbf{z}_2^{n/S}$ 、……、$\mathbf{z}_S^{n/S}$ 。
3. 计算 $\mathbf{z}^{n/S} = \mathbf{z}_1^{n/S} + \mathbf{z}_2^{n/S} + \cdots + \mathbf{z}_S^{n/S}$ 。
4. 从 $\mathbf{U}_n$ 拆分出 $\mathbf{U}_{n/S}$。
5. 以相同的方法计算 $\mathbf{U}_{n/S}\mathbf{z}^{n/S}$，直至拆分出 $\mathbf{U}_S$。

我们将按照上述方法计算 $\mathbf{U}_n\mathbf{z}^n$ 所需的四则运算数量上限为 $T(n)$，则有

$$\begin{aligned} T(n) =& T(n/S) + 3n \ , \\ T(n/S) =& T(n/S^2) + 3(n/S) \ , \\ & \vdots \\ T(S^2) =& T(S) + 3S \ , \\ T(S) =& 2S。 \end{aligned}$$

于是

$$
\begin{aligned}
T(n) =& T(n/S) + 3n \\
=& T(n/S^2) + 3(n/S) + 3n \\
& \vdots \\
=& T(S) + 3S + 3S^2 + \cdots + 3(n/S) + 3n \\
=& 2S + 3S + 3S^2 + \cdots + 3S^m \\
=& 2S + \frac{3S(S^m - 1)}{S - 1} \\
=& 2S + \frac{3S(n - 1)}{S - 1}。
\end{aligned}
$$

因为我们将 S 视为常数，所以 $T(n) = O(n)$。 再加上 $\mathbf{z}^n = \mathbf{Px}$ 只需不超过 n 个加法，所以 $\mathbf{U}_n(\mathbf{Px})$ 的时间复杂度是 $O(n)$。

3 如果 $n \neq S^m$

3.1 如果 $m \leq n$

3.1.1 如果 $m < \lceil \log_S(n) \rceil$

首先要保证 Σ 含有0。 如果 Σ 不含0，则将0添加至 Σ，S 的值也相应加1。 然后給 $\mathbf{A}$ 添加 $\lceil \log_S(n) \rceil - m$ 个元素全部为0的行向量与 $S^{\lceil \log_S(n) \rceil} - n$ 个元素全部为0的列向量，将其补齐为 $\lceil \log_S(n) \rceil$ 行 $S^{\lceil \log_S(n) \rceil}$ 列。 给 $\mathbf{x}$ 添加 $S^{\lceil \log_S(n) \rceil} - n$ 个0元素，将其长度补齐为 $S^{\lceil \log_S(n) \rceil}$。 最后按照第2节阐述的方法计算即可。

3.1.2 如果 $\lceil \log_S(n) \rceil < m \leq n$

首先要保证 Σ 含有0。 如果 Σ 不含0，则将0添加至 Σ，S 的值也相应加1。 然后給 $\mathbf{A}$ 添加 $S^{\lceil \log_S(n) \rceil} - n$ 个元素全部为0的列向量，将其补齐为 $S^{\lceil \log_S(n) \rceil}$ 列。

接着将 $\mathbf{A}$ 横向拆分为 $\lceil m/\log_S(n) \rceil$ 个子矩阵，使得每个子矩阵有 $\lceil \log_S(n) \rceil$ 行。 最后一个子矩阵的行数用 $\mathbf{0}^T$ 补齐。

最后对每个子矩阵采用第2节的方法计算。

因为有 $\lceil m/\log_S(n) \rceil$ 个子矩阵，每个子矩阵的计算时间复杂度是 $O(n)$，所以 $\mathbf{U}(\mathbf{Px})$ 的计算时间复杂度是 $O(mn/\log(n))$。

3.2 如果 $m > n$

此时我们有

$$
\begin{aligned}
& \mathbf{A}^T = \mathbf{UP} \\
\Rightarrow & \mathbf{A} = \mathbf{P}^T\mathbf{U}^T \\
\Rightarrow & \mathbf{Ax} = \mathbf{P}^T(\mathbf{U}^T\mathbf{x})。
\end{aligned}
$$

我们可以类似地证得 $\mathbf{U}^T\mathbf{x}$ 的计算时间复杂度是 $O(mn/\log(m))$。 所不同的是，由于 $\mathbf{U}$ 被转置，其拆分的方向也要相应地转置。 相应地，$\mathbf{x}$ 也不再是平均拆分成 S 个子向量，而是拆分成第1个元素与剩余元素分别组成的两个子向量。

$$\mathbf{U}_m^T\mathbf{x} = \left(\begin{array}{c:c} \sigma_1\,\mathbf{1}_{m/S} & \mathbf{U}_{m/S}^T \\ \hdashline \vdots & \vdots \\ \hdashline \sigma_S\,\mathbf{1}_{m/S} & \mathbf{U}_{m/S}^T \end{array}\right)\left(\begin{array}{c} x_1 \\ \hdashline \mathbf{x}_{2:n} \end{array}\right)。$$

主要思路与第2节类似，即递归拆分 $\mathbf{U}^T$，直至其只剩一列。