1.3.4 数值算法推导

接下来我们推导 Hulu 论文中求解 (1.13) 的数值算法，对数学不感兴趣的读者可以跳过。给定矩阵 $\boldsymbol{A}$，算法总共花费 $\mathcal{O}(nk^2)$ 时间选出 k 个物品。因为 $\boldsymbol{A}_{\mathcal{S}}$ 是对称正定矩阵，所以它存在 Cholesky 分解 $\boldsymbol{A}_{\mathcal{S}} = \boldsymbol{L}\boldsymbol{L}^\top$，这里的 $\boldsymbol{L}$ 是大小为 $|\mathcal{S}| \times |\mathcal{S}|$ 的下三角矩阵。下三角矩阵的意思是对角线以上的元素都为零。矩阵 $\boldsymbol{A}_{\mathcal{S}\cup\{i\}}$ 比 $\boldsymbol{A}_{\mathcal{S}}$ 多了一行和一列，记作：

$$\boldsymbol{A}_{\mathcal{S}\cup\{i\}} \; = \; \begin{bmatrix} \boldsymbol{A}_{\mathcal{S}} & \boldsymbol{a}_i \\ \boldsymbol{a}_i^\top & a_{ii} \end{bmatrix}. \tag{1.14}$$

上式中的 $\boldsymbol{a}_i$ 的元素是 $\boldsymbol{v}_i^\top \boldsymbol{v}_j$，$\forall j \in \mathcal{S}$，而 $a_{ii} = \boldsymbol{v}_i^\top \boldsymbol{v}_i = 1$。矩阵 $\boldsymbol{A}_{\mathcal{S}\cup\{i\}}$ 的 Cholesky 分解可以写作：

$$\boldsymbol{A}_{\mathcal{S}\cup\{i\}} \; = \; \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}_i^\top & d_i \end{bmatrix} \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}_i^\top & d_i \end{bmatrix}^\top, \tag{1.15}$$

其中 $\boldsymbol{c}_i$ 和 d_i 是未知的。由公式 (1.16) 和 (1.15) 可得：

$$\boldsymbol{A}_{\mathcal{S}\cup\{i\}} \; = \; \begin{bmatrix} \boldsymbol{A}_{\mathcal{S}} & \boldsymbol{a}_i \\ \boldsymbol{a}_i^\top & 1 \end{bmatrix} \; = \; \begin{bmatrix} \boldsymbol{L}\boldsymbol{L}^\top & \boldsymbol{L}\boldsymbol{c}_i \\ \boldsymbol{c}_i^\top \boldsymbol{L}^\top & \boldsymbol{c}_i^\top \boldsymbol{c}_i + d_i^2 \end{bmatrix}.$$

我们得到两个公式：

$$\boldsymbol{a}_i = \boldsymbol{L}\boldsymbol{c}_i \qquad \text{和} \qquad 1 = \boldsymbol{c}_i^T \boldsymbol{c}_i + d_i^2.$$

$\boldsymbol{L}$ 和 $\boldsymbol{a}_i$ 是已知的，$\boldsymbol{L}$ 是上一轮算出的 Cholesky 分解，$\boldsymbol{a}_i$ 包含矩阵 $\boldsymbol{A}$ 的元素。我们需要求出未知的 $\boldsymbol{c}_i$ 和 d_i。由于 $\boldsymbol{L}$ 是下三角矩阵，只需要 $\mathcal{O}(|\mathcal{S}|^2)$ 的浮点数运算即可求出 $\boldsymbol{c}_i = \boldsymbol{L}^{-1}\boldsymbol{a}_i$。然后就可以算出 $d_i^2 = 1 - \boldsymbol{c}_i^\top \boldsymbol{c}_i$。有了 d_i，我们就能快速求出 $\det(\boldsymbol{A}_{\mathcal{S}\cup\{i\}})$。由下三角矩阵和行列式的定义可知：

$$\det\left(\begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}_i^\top & d_i \end{bmatrix} \right) \; = \; \det(\boldsymbol{L}) \times d_i.$$

由于 $\det(\boldsymbol{X}\boldsymbol{Y}) = \det(\boldsymbol{X})\det(\boldsymbol{Y})$，我们得到

$$\det\left(\boldsymbol{A}_{\mathcal{S}\cup\{i\}}\right) \; = \; \det\left(\begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}^{Ti} & d_i \end{bmatrix} \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}_i^\top & d_i \end{bmatrix}^\top \right) \; = \; \det(\boldsymbol{L})^2 \times d_i^2.$$

贪心算法的公式 (1.13) 可以等价写作：

$$\underset{i \in \mathcal{R}}{\operatorname{argmax}} \; \theta \cdot \text{reward}_i \; + \; (1-\theta) \cdot \Big(\log\det(\boldsymbol{L})^2 \; + \; \log d_i^2 \Big).$$

由于 $\boldsymbol{L}$ 与 i 无关，上面的公式可以等价写作

$$\underset{i \in \mathcal{R}}{\operatorname{argmax}} \; \theta \cdot \text{reward}_i \; + \; (1-\theta) \cdot \log d_i^2. \tag{1.16}$$

这样我们就推导出了求解 k-DPP 的贪心算法：

1. 输入：n 个物品的向量表征 $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n \in \mathbb{R}^d$ 和分数 $\text{reward}_1, \cdots, \text{reward}_n$。
2. 计算 $n \times n$ 的相似度矩阵 $\boldsymbol{A}$，它的第 (i, j) 个元素等于 $a_{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$。时间复杂度为 $\mathcal{O}(n^2 d)$。
3. 选中 reward 分数最高的物品，记作 i。初始化集合 $\mathcal{S} = \{i\}$ 和 1×1 的矩阵 $\boldsymbol{L} = [1]$。（由于 $a_{ii} = \boldsymbol{v}_i^\top \boldsymbol{v}_i = 1$，此时 $\boldsymbol{A}_{\mathcal{S}} = [a_{ii}] = \boldsymbol{L}\boldsymbol{L}^\top$。）
4. 做循环，从 $t = 1$ 到 $k - 1$：
   (a). 对于每一个 $i \in \mathcal{R}$：
      I. 行向量 $[\boldsymbol{a}_i^T, 1]$ 是矩阵 $\boldsymbol{A}_{\mathcal{S} \cup \{i\}}$ 的最后一行。
      II. 求解线性方程组 $\boldsymbol{a}_i = \boldsymbol{L}\boldsymbol{c}_i$，得到 $\boldsymbol{c}_i$。时间复杂度为 $\mathcal{O}(|\mathcal{S}|^2)$。
      III. 计算 $d_i^2 = 1 - \boldsymbol{c}_i^T \boldsymbol{c}_i$。
   (b). 求解 (1.16)：$i^\star \; = \; \text{argmax}_{i \in \mathcal{R}} \; \theta \cdot \text{reward}_i \; + \; (1 - \theta) \cdot \log d_i^2$.
   (c). 更新集合 $\mathcal{S} \leftarrow \mathcal{S} \cup \{i^\star\}$。
   (d). 更新下三角矩阵：
$$\boldsymbol{L} \overset{22}{\leftarrow} \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{c}^{T i^\star} & d_{i^\star} \end{bmatrix}.$$
5. 返回集合 $\mathcal{S}$，其中包含 k 个物品。

该算法总时间复杂度为 $\mathcal{O}(n^2 d + nk^3)$。如果进一步优化线性方程组 $\boldsymbol{a}_i = \boldsymbol{L}\boldsymbol{c}_i$ 的求解，那么总时间复杂度可以降低到 $\mathcal{O}(n^2 d + nk^2)$。原理是在第 t 轮循环中，利用第 $t - 1$ 轮对$\boldsymbol{a}_i = \boldsymbol{L}\boldsymbol{c}_i$ 的求解。这里的数学有点复杂，就不展开介绍了。