lec16.3 The Apriori Algorithm

Apriori 算法

主要思想:
忽略那些不满足向下闭包性质的候选 $(k+1)$-项目集，因为这些候选项目集不可能是频繁的

Denotes:

- $\mathscr{C}_k$：候选 k-项目集的集合
- $\mathscr{F}$：频繁 k-项目集的集合

Apriori算法步骤:

Algorithm 1: Apriori

Input: Universe of items U, dataset $\mathscr{D}$, frequency threshold f
Output: Set of all frequent itemsets

Compute $\mathscr{F}_1$, the set of all frequent 1-itemsets;
Initialize $\mathscr{F}_i = \varnothing$ for $i = 2, 3, \ldots, d$;
for $k = 2, 3, \ldots, d$ **do**
  if $\mathscr{F}_{k-1}$ *is empty* **then**
    break;
  $\mathscr{C}_k = \text{generate-candidates}(\mathscr{F}_{k-1}, k)$;
  for *each* $I \in \mathscr{C}_k$ **do**
    Compute the support of I;
    if $sup(I) \geq f$ **then**
      Add I to $\mathscr{F}_k$;
return $\bigcup_{i=1}^{d} \mathscr{F}_i$;

1. **初始化**:
   - **输入**
2. **计算频繁的1-项集**:
   - 首先，计算频繁的1-项集 $\mathscr{F}_1$，即找到在 $\mathscr{D}$ 中至少出现 f 次的所有单个项目
   - 初始化 $\mathscr{F}_i = \emptyset$，$i = 2, 3, \ldots, d$
3. **迭代 $k = 2, 3, \ldots, d$**:
4. **检查前一个频繁项集是否为空**:
   - **如果 $\mathscr{F}_{k-1}$ 为空**: 如果频繁的 $(k-1)$-项集 $\mathscr{F}_{k-1}$ 为空，则算法终止，因为无法找到更大的频繁项集
5. **生成候选的 k-项集**:
   - **$\mathscr{C}_k = \text{generate-candidates}(\mathscr{F}_{k-1}, k)$**: 算法从频繁的 $(k-1)$-项集生成候选的 k-项集。通过连接共享 $(k-2)$-前缀的 $(k-1)$-项集来完成
6. **评估每个候选项集**:
   - **对于每个 $I \in \mathscr{C}_k$**: 对于每个候选的 k-项集 I，算法检查其支持度
7. **检查支持度**:

- 如果 $\sup(I) \geq f$: 如果候选的 k-项集 I 的支持度（包含 I 的交易数）大于或等于阈值 f，则将其添加到频繁的 k-项集

8. 添加到频繁的 k-项集:
   - 将 I 添加到 $\mathscr{F}_k$: 如果 I 满足支持度阈值，则将其添加到频繁的 k-项集 $\mathscr{F}_k$ 中
9. 返回所有频繁项集:
   - 返回 $\bigcup_{i=1}^{d} \mathscr{F}_i$: 最后，算法返回所有频繁项集的并集

假设:
假设 U 是一个包含 $1, 2, \ldots, d$ 的集合，每个项集是 U 的一个有序子集，这意味着项集中的元素按照一定顺序排列
数据集中的交易按字典序排列，字典序（lexicographically）是一种排序方法，类似于字典中的单词排序方式

例子

假设 $U = \{1, 2, 3, 4, 5\}$
数据集按字典序排序的例子：

```
{1,2,4}
{1,3,4}
{1,3,5}
{2,1,5}
{2,2,3}
```

Join Representation of Candidates

Apriori 算法中的候选项集生成（Join）和修剪（Prune）过程

候选项集生成表示（Join Representation of Candidates）

假设：$I = \{j_1, \ldots, j_k\}$ 是一个频繁的 k-项集，即 $I \in \mathscr{F}_k$

1. 对于 I 中的每个元素 j，项集 $I - \{j\}$ 是一个频繁的 $k-1$-项集，即 $I - \{j\} \in \mathscr{F}_{k-1}$
2. 具体来说，项集 I 可以表示为以下两个 $(k-1)$-项集的并集（也称为连接）：
   1. $\{j_1, \ldots, j_{k-1}\}$
   2. $\{j_1, \ldots, j_k\}$

因此，项集 I 只有在可以表示为两个 $(k-1)$-项集的并集时，才属于 $\mathscr{F}_k$

生成候选项集算法（Generate-candidates）

输入: 频繁项集 $\mathscr{F}$，项集大小 k

1. 假设 $\mathscr{F}$ 中的项集按字典序排列
2. 初始化候选项集集合 $\mathscr{C} = \emptyset$
3. 对于 $\mathscr{F}$ 中的每个项集 I:
   1. 令 $I = \{j_1, \ldots, j_{k-1}\}$，满足$j_1 < \ldots, j_{k-1}$
   2. 对于 j 从 $j_{k-1} + 1$ 到 d:
      1. 构造 $I' = \{j_1, \ldots, j_{k-2}, j\}$

        2. 如果 $I' \in \mathscr{F}$：
            1. 将 $\{j_1, \ldots, j_{k-1}, j\}$ 添加到 $\mathscr{C}$ 中
4. 对于 $\mathscr{C}$ 中的每个项集 I：
    1. 对于 I 中的每个元素 j：
        1. 如果 $I - \{j\} \notin \mathscr{F}$：
            1. 从 $\mathscr{C}$ 中删除 I；并中断循环
5. 返回候选项集 $\mathscr{C}$

例子

假设我们有以下频繁的2-项集（频繁项集集合 $\mathcal{F}_2$）：

$$\mathcal{F}_2 = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\}\}$$

现在，我们希望生成候选的3-项集（$\mathcal{C}_3$）并通过修剪来筛选出频繁的3-项集。

步骤1: 连接阶段（Join Phase）

根据连接步骤，我们通过将频繁的2-项集连接生成候选的3-项集。连接两个2-项集的条件是，它们共享一个相同的前缀。例如：

1. $\{1,2\}$和$\{1,3\}$可以连接成$\{1,2,3\}$
2. $\{1,2\}$和$\{1,4\}$可以连接成$\{1,2,4\}$
3. $\{1,3\}$和$\{1,4\}$可以连接成$\{1,3,4\}$
4. $\{2,3\}$和$\{2,4\}$可以连接成$\{2,3,4\}$

所以候选3-项集$\mathcal{C}_3$为：

$$\mathcal{C}_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\}$$

步骤2: 修剪阶段（Prune Phase）

在修剪阶段，我们检查每个候选3-项集的所有2-项子集是否都是频繁的。如果一个候选项集的任意一个2-项子集不是频繁的，那么这个候选项集就不是频繁的，需要被移除。

让我们逐个检查：

1. **候选项集** $\{1, 2, 3\}$：
   - 2-项子集：$\{1, 2\}, \{1, 3\}, \{2, 3\}$
   - 都在 $\mathcal{F}_2$ 中，所以 $\{1, 2, 3\}$ 保留。
2. **候选项集** $\{1, 2, 4\}$：
   - 2-项子集：$\{1, 2\}, \{1, 4\}, \{2, 4\}$
   - 都在 $\mathcal{F}_2$ 中，所以 $\{1, 2, 4\}$ 保留。
3. **候选项集** $\{1, 3, 4\}$：
   - 2-项子集：$\{1, 3\}, \{1, 4\}, \{3, 4\}$
   - $\{3, 4\}$ 不在 $\mathcal{F}_2$ 中，所以 $\{1, 3, 4\}$ 被移除。
4. **候选项集** $\{2, 3, 4\}$：
   - 2-项子集：$\{2, 3\}, \{2, 4\}, \{3, 4\}$
   - $\{3, 4\}$ 不在 $\mathcal{F}_2$ 中，所以 $\{2, 3, 4\}$ 被移除。

经过修剪后的频繁3-项集 $\mathcal{F}_3$ 为：

$$\mathcal{F}_3 = \{\{1, 2, 3\}, \{1, 2, 4\}\}$$