这一课时，我们进入第四种数据挖掘算法——关联分析的学习。关联分析是一种无监督学习，它的目标就是从大数据中找出那些经常一起出现的东西，不管是商品还是其他什么 item，然后靠这些结果总结出关联规则以用于后续的商业目的或者其他项目需求。

一个例子

不管你在哪一个数据挖掘课堂上，几乎都会听到这样一个 “都市传说”：在一个大型超市中，数据分析人员整理了一整年的购物篮数据，来分析大家都买过什么样的东西。就在对购物篮的数据进行分析的时候，分析人员惊奇地发现，与“尿不湿” 出现在一个购物小票上最频繁的商品竟然是啤酒。

这个结果背后的原因，是女人嘱托丈夫去超市给孩子买尿不湿，而丈夫通常会顺便买上一些自己喜欢的啤酒。超市发现了这个神奇的组合，于是毫不犹豫地把尿布和啤酒摆在了一起进行销售，以起到互相促进的作用。

关于这个故事是否真实发生过，我们保持怀疑的态度，但是这个故事确实反映了数据挖掘在商业运作中的价值。同时，购物篮分析也早已经是大型商超必备的技术手段。那么，如何发现这些潜在的价值，就用到了我们这一课时要讲的关联分析算法。

算法原理

要了解算法原理，首先我们需要对关联分析中的一些概念进行一下解释。

为了方便说明，我在这里编造了十条购物小票数据（如有雷同，纯属巧合），如下表所示。

表 1 购物小票数据

序号	购物小票
1	尿布，啤酒，奶粉，洋葱
2	尿布，啤酒，奶粉，洋葱
3	尿布，啤酒，苹果，洋葱
4	尿布，啤酒，苹果
5	尿布，啤酒，奶粉
6	尿布，啤酒，奶粉
7	尿布，啤酒，苹果
8	尿布，啤酒，苹果
9	尿布，奶粉，洋葱
10	奶粉，洋葱

项集（Item Set）： 第一个要介绍的概念叫项集，中文名称有点拗口，还是看英文比较容易理解。项集可以是单个的项，也可以是一系列项目的合集。在我们的例子中，项目就是啤酒、尿布等商品，一个小票上的内容就可以看作一个项集，通过关联分析得到的经常一起出现的啤酒和尿布可以称为一个 “频繁项集”。

关联规则： 根据频繁项集挖掘出的结果，例如 {尿布}→{啤酒}，规则的左侧称为先导，右侧称为后继。

支持度： 支持度就是一个项集在数据中出现的比例。在我们的 10 条数据中，{尿布} 出现了 9 次，那它的支持度就是 0.9；{啤酒} 出现了 8 次，啤酒的支持度就是 0.8。{尿布，啤酒} 的支持度是 8/10=0.8，以此类推。支持度还可以用来判定一条规则是否还需要继续进行挖掘，如果支持度已经很低，再加入新的项肯定会更低，挖掘的意义不大。

$$支持度（尿布）=\frac{尿布出现的次数}{购物小票的数量}=0.9$$

置信度： 置信度指的是在一条规则中，出现先导也出现后继的比例。我们可以用公式来看一下，

置信度表示的是一条规则的可靠程度。

$$置信度（尿布\rightarrow啤酒）=\frac{支持度（尿布u啤酒）}{支持度（尿布）}=0.89$$

提升度： 在置信度中，只考虑了规则中的先导与后继同时发生的情况，而对于后继单独发生的情况没有加以考虑。所以又有人提出了一个提升度，用来衡量先导和后继的独立性。比如在前面我们算出 “尿布→啤酒” 的置信度为 0.89，这说明买了 “尿布” 的人里有 89% 会买 “啤酒”，这看起来已经很高了。但是如果在没有买“尿布” 的购物小票中，购买 “啤酒” 的概率仍然为 0.89，那其实购买 “尿布” 和购买 “啤酒” 并没有什么关系。所以，提升度的计算公式如下：

$$提升度（尿布\rightarrow啤酒）=\frac{支持度（尿布u啤酒）}{支持度（尿布）\cdot 支持度（啤酒）}=1.1$$

如果提升度大于 1，说明是有提升的。

确信度： 最后一个指标是确信度，确信度指的是对于一条规则，不发生先导而发生后继的概率与这条规则错误的概率比值。公式如下：

$$确信度（尿布\rightarrow啤酒）=\frac{1-支持度（啤酒）}{1-置信度（尿布\rightarrow啤酒）}=1.82$$

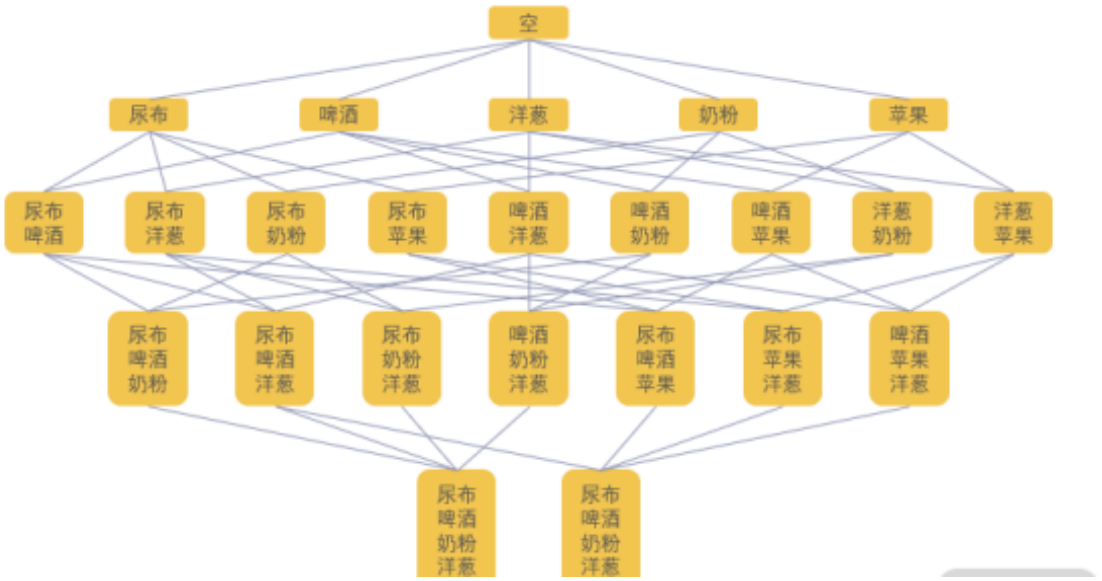
上面的结果为 1.82，这说明该条规则是真的概率比它只是偶然发生的概率高 82%。

一口气看了这么多的概念和指标，下面终于进入我们的正题环节，看看在关联挖掘中的算法原理。

Apriori

关联挖掘的目标已经很明确了，而关联挖掘的步骤也就只有两个：第一步是找出频繁项集，第二步是从频繁项集中提取规则。Apriori 算法的核心就是：如果某个项集是频繁项集，那么它的全部子集也都是频繁项集。

为了方便了解算法的效果，我预先画出了数据集中所有可能存在的项集关系，就如下图显示：

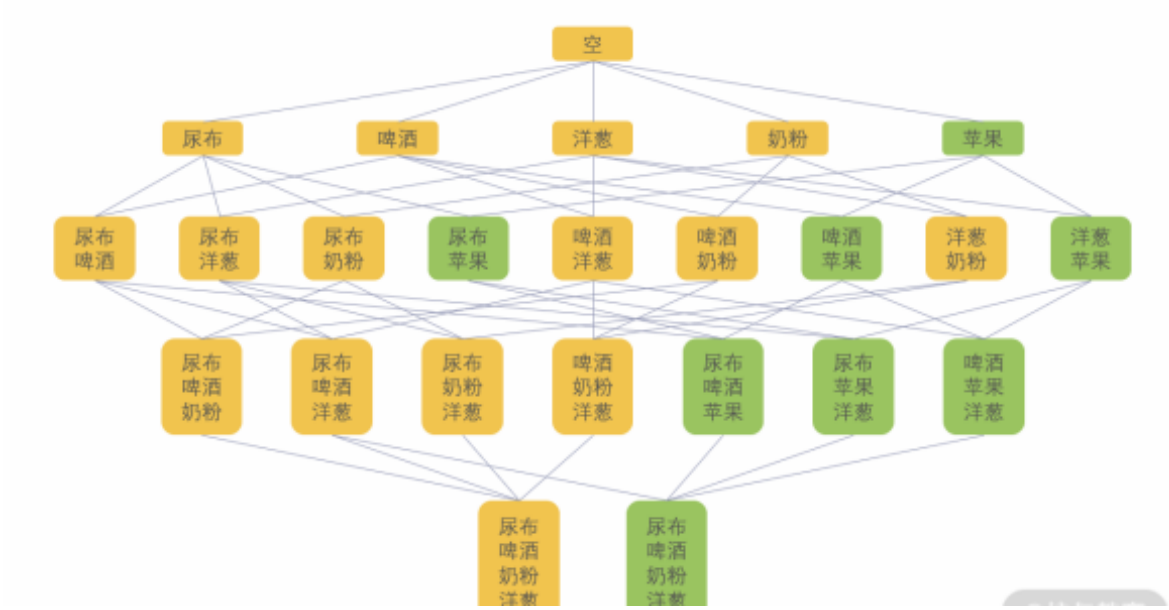


首先，需要设定一个最小支持度阈值，假设我们设定为 0.5，那么高于 0.5 的就认为是频繁项集。然后，我们计算出所有单个商品的支持度，如下表所示：

表 2 一阶项集支持度

项集	支持度
尿布	0.9
啤酒	0.8
奶粉	0.6
洋葱	0.5
苹果	0.4

从这里可以看出，“苹果”的支持度达不到阈值，于是把它删掉。因此，所有跟 “苹果” 相关的父集也都是低频的，在后续的计算也不会涉及了，就是下图标绿色的这些。



接下来我们再计算二阶的项集支持度。

表 3 二阶项集支持度

项集	支持度
尿布，啤酒	0.8
尿布，奶粉	0.5
尿布，洋葱	0.4
啤酒，洋葱	0.3
啤酒，奶粉	0.4
洋葱，奶粉	0.4

到了这一步，后四个项集的支持度已经达不到我们的阈值 0.5，于是也删掉。

接下来再计算三阶的项集支持度，这个时候发现已经没有可用的三阶项集了，所以算法计算结束。这时候我们得到了三组频繁项集 {尿布，啤酒}、{尿布，洋葱}、{尿布，奶粉}。接下来从这三个频繁项集，我们可以**得到三个关联关系：尿布→啤酒，尿布→洋葱，尿布→奶粉。**根据前面的公式，分别计算这三个关系的置信度、提升度和确信度。

表 4 三个关联关系的置信度、提升度和确信度

关系	支持度	置信度	提升度	确信度
尿布	0.9	|||		
啤酒	0.8	|||		
洋葱	0.5	|||		
奶粉	0.6	|||		
尿布→啤酒	0.8	0.89	1.1	1.82
尿布→奶粉	0.5	0.55	0.926	0.89

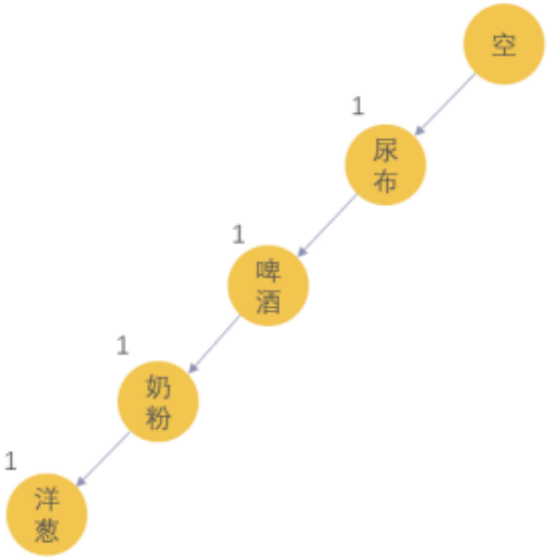
到了这里，再根据我们的需求设定阈值来筛选最终需要留下来的规则。

FP-Growth（Frequent Pattern Growth）

根据上面的 Apriori 计算过程，我们可以知道 Apriori 计算的过程中，会使用排列组合的方式列举出所有可能的项集，每一次计算都需要重新读取整个数据集，从而计算本轮次的项集支持度。所以 Apriori 会耗费大量的计算资源，这时候就有了一个更高效的算法——FP-Growth 算法。

Apriori 算法一开始需要对所有的规则进行枚举，然后再进行计算，而 FP-Growth 则是首先使用数据生成一棵 FP-Growth 树，然后再根据这棵树来生成频繁项集。下面我们来看一下如何构建 FP 树。

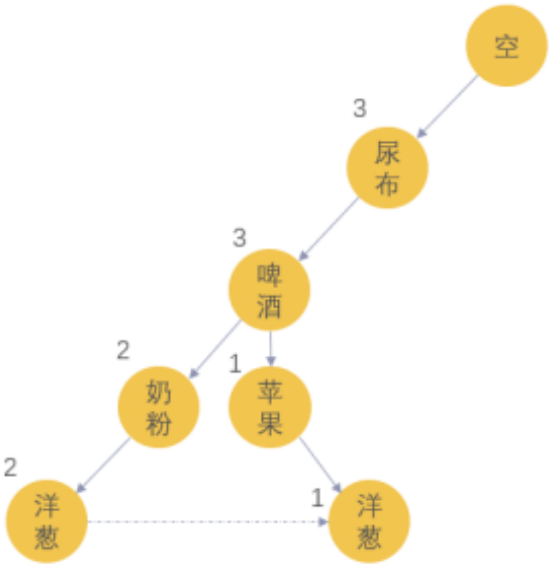
仍然使用上面编的购物小票数据，并对每一个小票里的项按统一的顺序进行排序，设置一个空集作为根节点。我们把第一条数据录入这个树中，每一个单项作为上一个节点的叶子节点，旁边的数据代表该路径访问的次数。下图就是我们录入第一条数据后的结果：



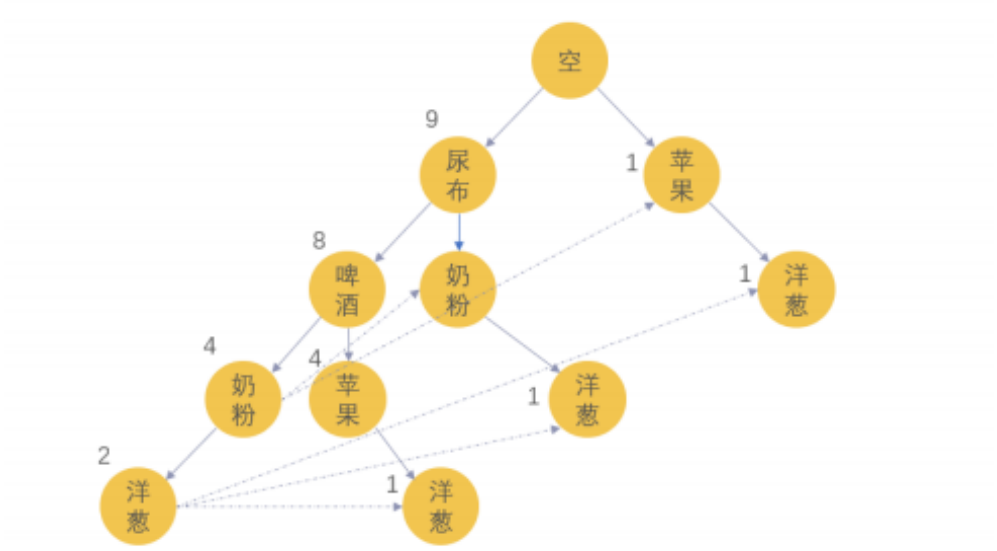
第二条数据与第一条一样，所以只有数字的变化，节点不会发生变化。

接着输入第三条数据。

输入完第二、三条数据之后的结果如下，其中由第一个洋葱和第二个洋葱直接画了一条虚线，标识它们是同一项。



接下来我们把所有的数据都插入到这棵树上。



这个时候我们得到的是完整的 FP 树，接下来我们要从这里面去寻找频繁项集。当然首先也要设定一个最小频度的阈值，然后从叶子节点开始，也就是最低频的节点。如果频度高于阈值那么就收录所有以该项结尾的项集，否则就向上继续检索。

至于后面的计算关联规则的方法跟上面就没有什么区别了，这里不再赘述。下面我们尝试使用代码来实现关联规则的发现。

尝试动手

在我们之前使用的 sklearn 算法工具包中没有 apriori 算法，所以这次我们需要先安装一个 efficient-apriori 算法包。而我们这次所使用的数据集，也就是我在上面提到的这个 “啤酒尿布” 数据。这个算法包使用起来也极其简单。

```
'''记得安装包

pip install efficient-apriori

'''

from efficient_apriori import apriori

# 设置数据集

data = [('尿布', '啤酒', '奶粉','洋葱'),

('尿布', '啤酒', '奶粉','洋葱'),

('尿布', '啤酒', '苹果','洋葱'),

('尿布', '啤酒', '苹果'),
```

```
('尿布', '啤酒', '奶粉'),

('尿布', '啤酒', '奶粉'),

('尿布', '啤酒', '苹果'),

('尿布', '啤酒', '苹果'),

('尿布', '奶粉', '洋葱'),

('奶粉', '洋葱')

]

# 挖掘频繁项集和规则

itemsets, rules = apriori(data, min_support=0.4, min_confidence=1)

print(itemsets)

print(rules)

#输出结果

{1: {('奶粉',): 6, ('洋葱',): 5, ('尿布',): 9, ('啤酒',): 8, ('苹果',): 4}, 2: {('啤酒', '奶粉'): 4, ('啤酒', '尿布'): 8, ('奶粉', '尿布'): 5, ('奶粉', '洋葱'): 4, ('尿布', '洋葱'): 4, ('啤酒', '苹果'): 4, ('尿布', '苹果'): 4}, 3: {('啤酒', '奶粉', '尿布'): 4, ('啤酒', '尿布', '苹果'): 4}}

[{啤酒} -> {尿布}, {苹果} -> {啤酒}, {苹果} -> {尿布}, {啤酒, 奶粉} -> {尿布}, {尿布, 苹果} -> {啤酒}, {啤酒, 苹果} -> {尿布}, {苹果} -> {啤酒, 尿布}]

#把min_support设置成0.5输出结果

{1: {('尿布',): 9, ('奶粉',): 6, ('啤酒',): 8, ('洋葱',): 5}, 2: {('啤酒', '尿布'): 8, ('奶粉', '尿布'): 5}}

[{啤酒} -> {尿布}]
```

通过上面的代码，我们就成功使用了 apriori 算法，对我们自己编造的数据集完成了关联关系挖掘，当在设置最小支持度为 0.4 的时候，我们找到了 7 条规则；而我们把最小支持度改为 0.5 以后，只剩下 {啤酒} -> {尿布} 这一条规则了。你学会了吗？

总结

这节课里，我们介绍了两种关联关系挖掘的方法，其中 Apriori 使用了穷举的方式，而 FP-Growth 使用了树形结构来提高速度。关联关系挖掘通常使用的算法都非常简单，或者我们可以把关联关系挖掘转化成分类问题、聚类问题来解决都是可以的。

在这节课中，我们还介绍了关联关系的评估指标，不管是用什么算法来挖掘的关联关系，都可以使用这些指标来进行评估。

下一课时，我们会进入关联关系挖掘的实践课程，看看如何使用关联关系挖掘来解决业务中的问题。