

# FP-growth笔记

浮生的笔记 1月5日

FP-growth (Frequent Pattern Growth) 模型，用于挖掘频繁项集和关联规则的算法，最近在学习这个算法，做一下笔记。

## 频繁项集

啤酒与尿布就是一个典型的例子。有研究发现，在超市的订单记录中，啤酒和尿布总是频繁共同出现在同一条订单记录里。这里啤酒和尿布就是频繁项集。

## 关联规则

关联规则是在频繁项集的基础上得到的。关联规则指由集合 A，可以在某置信度下推出集合 B。通俗来说，就是如果 A 发生了，那么 B 也很有可能会发生。举个例子，有关联规则如：{'鸡蛋', '面包'} -> {'牛奶'}，该规则的置信度是 0.9，意味着在所有买了鸡蛋和面包的客户中，有 90% 的客户还买了牛奶。这里涉及到两个概念：支持度和置信度。

## 支持度

支持度指某频繁项集在整个数据集中的比例。假设数据集有 10 条记录，包含{'鸡蛋', '面包'}的有 5 条记录，那么{'鸡蛋', '面包'}的支持度就是  $5/10 = 0.5$ 。

## 置信度

置信度是针对某个关联规则定义的。有关联规则如{'鸡蛋', '面包'} -> {'牛奶'}，它的置信度计算公式为{'鸡蛋', '面包', '牛奶'}的支持度 / {'鸡蛋', '面包'}的支持度。假设{'鸡蛋', '面包', '牛奶'}的支持度为 0.45，{'鸡蛋', '面包'}的支持度为 0.5，则{'鸡蛋', '面包'} -> {'牛奶'}的置信度为  $0.45 / 0.5 = 0.9$ 。

## FP-growth

假设事务数据库如下

1	A,B,E
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C

7	A,C
8	A,B,C,E
9	A,B,C

首先扫描一次数据库得到项目集

A	B	C	D	E
6	7	6	2	2

令最小支持度为2:  $\min\_sup=2$ ,

过滤支持度小于2的元素, 并对项目集内元素按频数重排, 得到项头表

B	A	C	D	E
7	6	6	2	2

按频数调整事务数据库

1	B,A,E
2	B,D
3	B,C
4	B,A,D
5	A,C
6	B,C
7	A,C
8	B,A,C,E
9	B,A,C

按照调整后的数据库 形成FP-tree

形成FP-tree后, 按照项头表的顺序, 从低到高依此求得各个元素的频繁项集

画图比较麻烦, 见下图笔记...

**事务数据库:**

1. A, B, E
2. B, D
3. B, C
4. A, B, D
5. A, C
6. B, C
7. A, C
8. A, B, C, E
9. A, B, C

**扫描一次得项目集:**

A B C D E  
6 7 6 2 2

min-sup = 2 重排

B A C D E  
7 6 6 2 2

**调整事务数据库:**

1. B, A, E
2. B, D
3. B, C
4. B, A, D
5. A, C
6. B, C
7. A, C
8. B, A, C, E
9. B, A, C

**形成 Tree**

Null

B:7

A:2

A:4

C:2

D:1

E:1

C:2

D:1

E:1

**项头表:**

B 7  
A 6  
C 6  
D 2  
E 2

**1. 首先考虑 E 的频繁项集:**

Null

B:2

A:2

E:1

min-sup

$\langle B, A:1 \rangle < \langle B, A, C:1 \rangle$

频繁项集:  $\{B, E:2\}, \{A, E:2\}, \{B, A, E:2\}$

$\{B, D:2\}$

**2. D:**

Null

B:2

D:1

$\langle B, A:1 \rangle < \langle B, A, C:1 \rangle$

**C:**

Null

B:4

A:2

C:2

$\langle B, A:2 \rangle < \langle B, C:2 \rangle < \langle A, C:2 \rangle$

频繁项集:  $\{B, C:4\}, \{A, C:4\}, \{B, A, C:2\}$

$\{B, A:6\}$

**最终频繁项集:**

min-sup = 2

B, A 4  
B, C 4  
A, C 4  
B, A, C 2  
B, D 2  
B, E 2  
A, E 2  
B, A, E 2

浮生的笔记

## Scala实现

```

1 def main(args: Array[String]): Unit = {
2     val appName = "FP-Growth关联规则"
3     val sparkConfig = new SparkConf().setAppName(appName).setMaster("local[*]")
4     val spark: SparkSession = SparkSession.builder().config(sparkConfig).getOrCreate()
5     var sqlContext = spark.sqlContext
6     // 调整开发输出的日志信息
7     val sc = spark.sparkContext
8     sc.setLogLevel("ERROR")
9
10    import spark.implicits._
11    val dataList: List[(String)] = List(
12        ("A,B,E"),
13        ("B,D"),
14        ("B,C"),
15        ("A,B,D"),
16        ("A,C"),

```

```

17      ("B,C"),
18      ("A,C"),
19      ("A,B,C,E"),
20      ("A,B,C")
21  )
22  val data = dataList.map(t => t.split(",")).toDF("items")
23
24  val fpgrowth = new FPGrowth().setItemsCol("items").setMinSupport(0.2).set
25  val model = fpgrowth.fit(data)
26
27  // Display frequent itemsets.
28  model.freqItemsets.show()
29
30  // Display generated association rules.
31  model.associationRules.show()
32
33  // transform examines the input items against all the association rules c
34  // consequents as prediction
35  model.transform(data).show()
36  }

```

## Python实现-pyspark

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # @Time      : 2021-01-05 17:52
4  # @File      : FP-Growth.py
5  # @Author    : FS
6
7
8  from pyspark.ml.fpm import FPGrowth
9  from pyspark.sql import SparkSession
10 from pyspark.sql import functions as F
11
12 spark = SparkSession.builder.appName("FP-Growth关联规则").getOrCreate()
13 df = spark.createDataFrame([
14     (0, 'A,B,E'),
15     (1, 'B,D'),

```

```
16     (2, 'B,C'),
17     (3, 'A,B,D'),
18     (4, 'A,C'),
19     (5, 'B,C'),
20     (6, 'A,C'),
21     (7, 'A,B,C,E'),
22     (8, 'A,B,C')
23 ], ["id", "items"])
24 df = df.withColumn("items", F.split(df.items, ","))
25 df.show()
26
27 fpGrowth = FPGrowth(itemsCol="items", minSupport=0.2, minConfidence=0.5)
28 model = fpGrowth.fit(df)
29
30 # Display frequent itemsets.
31 model.freqItemsets.show()
32
33 # Display generated association rules.
34 model.associationRules.show()
35
36 # transform examines the input items against all the association rules and su
37 # consequents as prediction
38 model.transform(df).show()
```

## Python实现-pyfpgrowth

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # @Time      : 2021-01-05 17:52
4  # @File      : FP-Growth.py
5  # @Author    : FS
6
7
8  import pandas as pd
9  import pyfpgrowth
10
11 df = pd.DataFrame([
12     'A,B,E',
```

```
13     'B,D',
14     'B,C',
15     'A,B,D',
16     'A,C',
17     'B,C',
18     'A,C',
19     'A,B,C,E',
20     'A,B,C'
21 ])
22 df.columns = ['items']
23 df['items'] = df['items'].map(lambda x: x.split(','))
24
25 res = []
26 for i in df.values.tolist():
27     res.append(i[0])
28
29 patterns = pyfpgrowth.find_frequent_patterns(res, 2)
30 print(patterns)
31
32 rules = pyfpgrowth.generate_association_rules(patterns, 0.7)
33 print(rules)
```

喜欢此内容的人还喜欢

数据真的真实？识破一本正经的胡说八道

浪子名臣

---

R绘图之蜂群图 (beeswarm)

科学家的司机

---

不荐|解构《俞军产品方法论》之三：决策框架

山间过堂风