»

Frequent Pattern Mining

Mining frequent items, itemsets, subsequences, or other substructures is usually among the first steps to analyze a large-scale dataset, which has been an active research topic in data mining for years. We refer users to Wikipedia's association rule learning for more information.

Table of Contents

FP-Growth

The FP-growth algorithm is described in the paper Han et al., Mining frequent patterns without candidate generation, where "FP" stands for frequent pattern. Given a dataset of transactions, the first step of FP-growth is to calculate item frequencies and identify frequent items. Different from Apriori-like algorithms designed for the same purpose, the second step of FP-growth uses a suffix tree (FP-tree) structure to encode transactions without generating candidate sets explicitly, which are usually expensive to generate. After the second step, the frequent itemsets can be extracted from the FP-tree. In `spark.mllib`, we implemented a parallel version of FP-growth called PFP, as described in Li et al., PFP: Parallel FP-growth for query recommendation. PFP distributes the work of growing FP-trees based on the suffixes of transactions, and hence is more scalable than a single-machine implementation. We refer users to the papers for more details.

`spark.ml`'s FP-growth implementation takes the following (hyper-)parameters:

- `minSupport`: the minimum support for an itemset to be identified as frequent. For example, if an item appears 3 out of 5 transactions, it has a support of 3/5=0.6.
- `minConfidence`: minimum confidence for generating Association Rule. Confidence is an indication of how often an association rule has been found to be true. For example, if in the transactions itemset `X` appears 4 times, `X` and `Y` co-occur only 2 times, the confidence for the rule `X => Y` is then 2/4 = 0.5. The parameter will not affect the mining for frequent itemsets, but specify the minimum confidence for generating association rules from frequent itemsets.
- `numPartitions`: the number of partitions used to distribute the work. By default the param is not set, and number of partitions of the input dataset is used.

The `FPGrowthModel` provides:

- `freqItemsets`: frequent itemsets in the format of DataFrame("items"[Array], "freq"[Long])
- `associationRules`: association rules generated with confidence above `minConfidence`, in the format of DataFrame("antecedent"[Array], "consequent"[Array], "confidence"[Double]).
- `transform`: For each transaction in `itemsCol`, the `transform` method will compare its items against the antecedents of each association rule. If the record contains all the antecedents of a specific association rule, the rule will be considered as applicable and its consequents will be added to the prediction result. The transform method will summarize the consequents from all the applicable rules as prediction. The prediction column has the same data type as `itemsCol` and does not contain existing items in the `itemsCol`.

Examples

Scala | Java | **Python** | R

Refer to the Python API docs for more details.

```
from pyspark.ml.fpm import FPGrowth

df = spark.createDataFrame([
    (0, [1, 2, 5]),
    (1, [1, 2, 3, 5]),
    (2, [1, 2])
], ["id", "items"])

fpGrowth = FPGrowth(itemsCol="items", minSupport=0.5, minConfidence=0.6)
model = fpGrowth.fit(df)

# Display frequent itemsets.
model.freqItemsets.show()

# Display generated association rules.
model.associationRules.show()

# transform examines the input items against all the association rules and summarize the
# consequents as prediction
model.transform(df).show()
```

Find full example code at "examples/src/main/python/ml/fpgrowth_example.py" in the Spark repo.