Big data: architectures and data analytics

Itemset and Association rule mining

Itemset and Association rule mining

- Spark MLlib provides
  - An itemset mining algorithm based on the FP-growth algorithm
    - That extracts all the sets of items (of any length) with a minimum frequency
  - A rule mining algorithm
    - That extracts the association rules with a minimum frequency and a minimum confidence
    - Only the rules with one single item in the consequent of the rules are extracted

Itemset and Association rule mining

- The input dataset in this case is a set of transactions
- Each transaction is defined as a set of items
- Transactional dataset: example

  A B C D

  A B

  B C

  A D E
- The example dataset contains 4 transactions

The FP-Growth algorithm and Association rule mining

The FP-Growth algorithm

- FP-growth is one of the most popular and efficient itemset mining algorithms
- It is characterized by one single parameter
  - The minimum support threshold **(minsup)**
    - i.e., the minimum frequency of the itemset in the input transational dataset
    - It is a real value in the range (0-1]
  - The minsup threshold is used to limit the number of mined itemsets
- The input dataset is a transactional dataset

Association Rule Mining

- Given a set of frequent itemsets, the frequent association rules can be mined
- An association rule is mined if
  - Its frequency is greater than the minimum support threshold (**minsup**)
    - i.e., a minimum frequency
    - The minsup value is specified during the itemset mining step and not during the association rule mining step
  - Its confidence is greater than the minimum confidence threshold (**minconf**)
    - i.e., a minimum "correlation"
    - It is a real value in the range [0-1]

The FP-Growth algorithm

- The MLlib implementation of FP-growth is based on DataFrames
- Differently from the other algorithms, the FP-growth algorithm is not invoked by using pipelines

Itemset and Association Rule Mining

- Itemset and association rule mining
  - Instantiate an FP-Growth object
  - Invoke the fit(input data) method on the FP-Growth object
  - Retrieve the sets of frequent itemset and association rules by invoking the following methods of on the FP-Growth object
    - Dataset<Row> freqItemsets()
    - Dataset<Row> associationRules()

Itemset and Association Rule Mining: Input data

- The input of the MLlib itemset and rule mining algorithm is a Dataset<Row> containing a column called **items**
  - Data type: java.util.List<String>
- Each record of the input DataFrame contains one transaction, i.e., a set of items
- Items are represented by means of String objects

Itemset and Association Rule Mining: Input data

- Example of input data

  A B C D

  A B

  B C

  A D E

- This example file contains 4 transactions

Itemset and Association Rule Mining: Input data

- Input data

A B C D
A B
B C
A D E

- Input Dataset<Row> that must be generated as input for the MLlib itemset mining algorithm

items
[A, B, C, D]
[A, B]
[B, C]
[A, D, E]

Itemset and Association Rule Mining: In

The input lines are "stored" in a List of strings (one list for each input line/transaction).
The generated Dataset<Row> contains a column called items containing the lists associated with the input data.

- Input data

```
A B C D
A B
B C
A D E
```

- Input Dataset<Row> that must be generated as input for the MLlib itemset mining algorithm

items
[A, B, C, D]
[A, B]
[B, C]
[A, D, E]

Itemset and Association Rule Mining: Example

- The following slides show how to
  - Extract the set of frequent itemsets from a transactional dataset and the association rules from the extracted frequent itemsets
- The input dataset is a transactional dataset
  - Each line of the input file contains a transaction, i.e., a set of items

Itemset and Association Rule Mining: Example

- Example of input file

  A B C D

  A B

  B C

  A D E

- This example file contains 4 transactions

Itemset and Association Rule Mining: Example

```
package it.polito.bigdata.spark.sparkmllib;
import java.io.Serializable;
import java.util.List;

public class Transaction implements Serializable {
    private List<String> items;
    public List<String> getItems() {
        return items;
    }

    public void setItems(List<String> items) {
        this.items = items;
    }

    public Transaction(List<String> items) {
        this.items = items;
    }
}
```

Itemset and Association Rule Mining: Example

```
package it.polito.bigdata.spark.sparkmllib;

import java.util.Arrays;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.ml.fpm.FPGrowth;
import org.apache.spark.ml.fpm.FPGrowthModel;
```

Itemset and Association Rule Mining: Example

```
public class SparkDriver {

 public static void main(String[] args) {
        String inputFile;
        String outputFolderItemsets;
        String outputFolderRules;
        double minSupport;
        double minConfidence;

        inputFile = args[0];
        outputFolderItemsets = args[1];
        outputFolderRules = args[2];
        minSupport = Double.parseDouble(args[3]);
        minConfidence = Double.parseDouble(args[4]);
```

Itemset and Association Rule Mining: Example

```
// Create a Spark Session object and set the name of the application
// We use some Spark SQL transformation in this program
SparkSession ss = SparkSession.builder()
.appName("MLlib - Itemset and Association rule mining").getOrCreate();

// Create a Java Spark Context from the Spark Session
// When a Spark Session has already been defined this method
// is used to create the Java Spark Context
JavaSparkContext sc = new JavaSparkContext(ss.sparkContext());
```

Itemset and Association Rule Mining: Example

```
// *************************
// Itemset and rule mining
// *************************

// Read input data
JavaRDD<String> inputData = sc.textFile(inputFile);

// Map each input line/data point of the input file to a Transaction.
//Transaction is characterized by the items "attribute" (data type:
// List<String>)
JavaRDD<Transaction> inputRDD = inputData.map(line -> {

        String[] items = line.split(" ");

        // Return a Transaction based on the content of the current line
        return new Transaction(Arrays.asList(items));
});
```

Itemset and Association Rule Mining: Example

```
// Create a DataFrame based on the input data.
Dataset<Row> transactionsData = ss
                        .createDataFrame(inputRDD, Transaction.class).cache();

// Create an FPGrowth object
FPGrowth fp = new FPGrowth();

// Set the value of min. support and min. confidence
fp.setMinSupport(minSupport)
   .setMinConfidence(minConfidence);

// Extract frequent itemsets and association rules by invoking the fit
// method of FPGrowth on the input data
FPGrowthModel itemsetsAndRulesModel = fp.fit(transactionsData);
```

Itemset and Association Rule Mining: Example

Instance of the FPGrowth object and setting of its parameters
-Minimum support threshold: double [0-1]
-Minimum confidence threshold: double [0-1]

```
// Create a
Dataset<R                                                    on.class).cache();
```

```
// Create an FPGrowth object
FPGrowth fp = new FPGrowth();

// Set the value of min. support and min. confidence
fp.setMinSupport(minSupport)
   .setMinConfidence(minConfidence);
```

```
// Extract frequent itemsets and association rules by invoking the fit
// method of FPGrowth on the input data
FPGrowthModel itemsetsAndRulesModel = fp.fit(transactionsData);
```

Itemset and Association Rule Mining: Example

```
// Create a DataFrame based on the input data.
Dataset<Row> transactionsData = ss
                        .createDataFrame(inputRDD, Transaction.class).cache();

// Create an FPGrowth object
FPGrowth fp = new FPGrowth();

// Set the value of min-support and min-confidence
```

The fit method is used to run the FPGrowth algorithm on the input data

```
// Extract frequent itemsets and association rules by invoking the fit
// method of FPGrowth on the input data
FPGrowthModel itemsetsAndRulesModel = fp.fit(transactionsData);
```

Itemset and Association Rule Mining: Example

```
        // Retrieve the set of frequent itemsets
        Dataset<Row> frequentItemsets = itemsetsAndRulesModel.freqItemsets();

        // Retrieve the set of association rules
        Dataset<Row> frequentRules = itemsetsAndRulesModel.associationRules();

        // Save the itemset in an HDFS output folder
        JavaRDD<Row> itemsetsRDD = frequentItemsets.javaRDD();
        itemsetsRDD.saveAsTextFile(outputFolderItemsets);

        // Save the rules in an HDFS output folder
        JavaRDD<Row> rulesRDD = frequentRules.javaRDD();
        rulesRDD.saveAsTextFile(outputFolderRules);

        sc.close();
        ss.stop();
    }
}
```

Itemset and Association Rule Mining: Example

```
        // Retrieve the set of frequent itemsets
        Dataset<Row> frequentItemsets = itemsetsAndRulesModel.freqItemsets();

        itemsetsRDD.saveAsTextFile(outputFolderItemsets);

        // Save the rules in an HDFS output folder
        JavaRDD<Row> rulesRDD = frequentRules.javaRDD();
        rulesRDD.saveAsTextFile(outputFolderRules);

        sc.close();
        ss.stop();
    }
}
```

The returned DataFrame contains one record for each frequent itemset.
Schema of the returned DataFrame:
- items: Array<String>
- freq: long

Itemset and Association Rule Mining: Example

```
        // Retrieve the set of frequent itemsets
        Dataset<Row> frequentItemsets = itemsetsAndRulesModel.freqItemsets();

        // Retrieve the set of association rules
        Dataset<Row> frequentRules = itemsetsAndRulesModel.associationRules();
```

The returned DataFrame contains one record for each association rule.
Schema of the returned DataFrame:
- antecedent: Array<String>
- consequent: Array<String>
- confidence: double

```
        rulesRDD.saveAsTextFile(outputFolderRules);

        sc.close();
        ss.stop();
    }
}
```