Apache Mahout

Scalable machine learning and data mining

Image

Origin of Mahout

Origin of Mahout

- Primary goal is creating scalable and efficient machine learning algorithms

Origin of Mahout

- Primary goal is creating scalable and efficient machine learning algorithms
- Ng et al.'s paper "**Map-Reduce for Machine Learning on Multicore**" was the driving force

Paper

Origin of Mahout

- Primary goal is creating scalable and efficient machine learning algorithms
- Ng et al.'s paper "**Map-Reduce for Machine Learning on Multicore**" was the driving force
- Developed as a ‘driver’ for Hadoop

Paper

Origin of Mahout

- Primary goal is creating scalable and efficient machine learning algorithms
- Ng et al.'s paper "**Map-Reduce for Machine Learning on Multicore**" was the driving force
- Developed as a ‘driver’ for Hadoop
- The latest release (Samsara),has shifted away from MapReduce

Paper

Origin of Mahout

- Primary goal is creating scalable and efficient machine learning algorithms
- Ng et al.'s paper "**Map-Reduce for Machine Learning on Multicore**" was the driving force
- Developed as a ‘driver’ for Hadoop
- The latest release (Samsara),has shifted away from MapReduce
- Evolved from being a collection of algorithms, to a scala based programming environment

Paper

Hadoop MapReduce vs Mahout on Spark

Hadoop MapReduce vs Mahout on Spark

MapReduce

Mahout Samsara

Hadoop MapReduce vs Mahout on Spark

MapReduce

- Real time streaming operations unsupported

Mahout Samsara

- In-memory storage makes streaming possible

Hadoop MapReduce vs Mahout on Spark

MapReduce

- Real time streaming operations unsupported
- Currently available algorithms :

Item based filtering, Matrix Factorization

**Most algorithms have been deprecated*

Mahout Samsara

- In-memory storage makes streaming possible
- Currently available algorithms :

Distributed BLAS, User & Item based filtering,Naive Bayes Classifier, SVD, PCA, RowSimilarity Job

Hadoop MapReduce vs Mahout on Spark

MapReduce

- Real time streaming operations unsupported
- Currently available algorithms :

  Item based filtering, Matrix Factorization

  **Most algorithms have been deprecated*

- Strong security measures exist

Mahout Samsara

- In-memory storage makes streaming possible
- Currently available algorithms :

  Distributed BLAS, User & Item based filtering,Naive Bayes Classifier, SVD, PCA, RowSimilarity Job

- Security is still in its infancy

Hadoop MapReduce vs Mahout on Spark

MapReduce

- Real time streaming operations unsupported
- Currently available algorithms :

  Item based filtering, Matrix Factorization

  **Most algorithms have been deprecated*

- Strong security measures exist
- Java is the primary choice

Mahout Samsara

- In-memory storage makes streaming possible
- Currently available algorithms :

  Distributed BLAS, User & Item based filtering,Naive Bayes Classifier, SVD, PCA, RowSimilarity Job

- Security is still in its infancy
- Scala, Java

Algorithms in Mahout

Algorithms in Mahout

- Algorithms are usually classified into 3 categories
    - Collaborative Filtering - Item Recommendation used in ecommerce (Amazon)

Algorithms in Mahout

- Algorithms are usually classified into 3 categories
  - Collaborative Filtering - Item Recommendation used in ecommerce (Amazon)
  - Clustering - K means clustering

Algorithms in Mahout

- Algorithms are usually classified into 3 categories
  - Collaborative Filtering - Item Recommendation used in ecommerce (Amazon)
  - Clustering - K means clustering
  - Categorization/Classification - Naive Bayes, Logistic Regression

Algorithms in Mahout

- Algorithms are usually classified into 3 categories
  - Collaborative Filtering - Item Recommendation used in ecommerce (Amazon)
  - Clustering - K means clustering
  - Categorization/Classification - Naive Bayes, Logistic Regression
- Dimensionality reduction algorithms are also provided by the Mahout Math-Scala Core library (SVD, Distributed RowMatrix)

Algorithms in Mahout

- Algorithms are usually classified into 3 categories
  - Collaborative Filtering - Item Recommendation used in ecommerce (Amazon)
  - Clustering - K means clustering
  - Categorization/Classification - Naive Bayes, Logistic Regression
- Dimensionality reduction algorithms are also provided by the Mahout Math-Scala Core library (SVD, Distributed RowMatrix)
- We shall try to implement a variant of Collaborative Filtering algorithm - Recommendation Engine

Installation of Mahout and Spark

- Install Java, git, maven, ssh
- Install Scala
- Install Spark
- Install Mahout
- Add JAVA_HOME, SCALA_HOME, SPARK_HOME,MAHOUT_HOME values to the path

Installation of Mahout and Spark

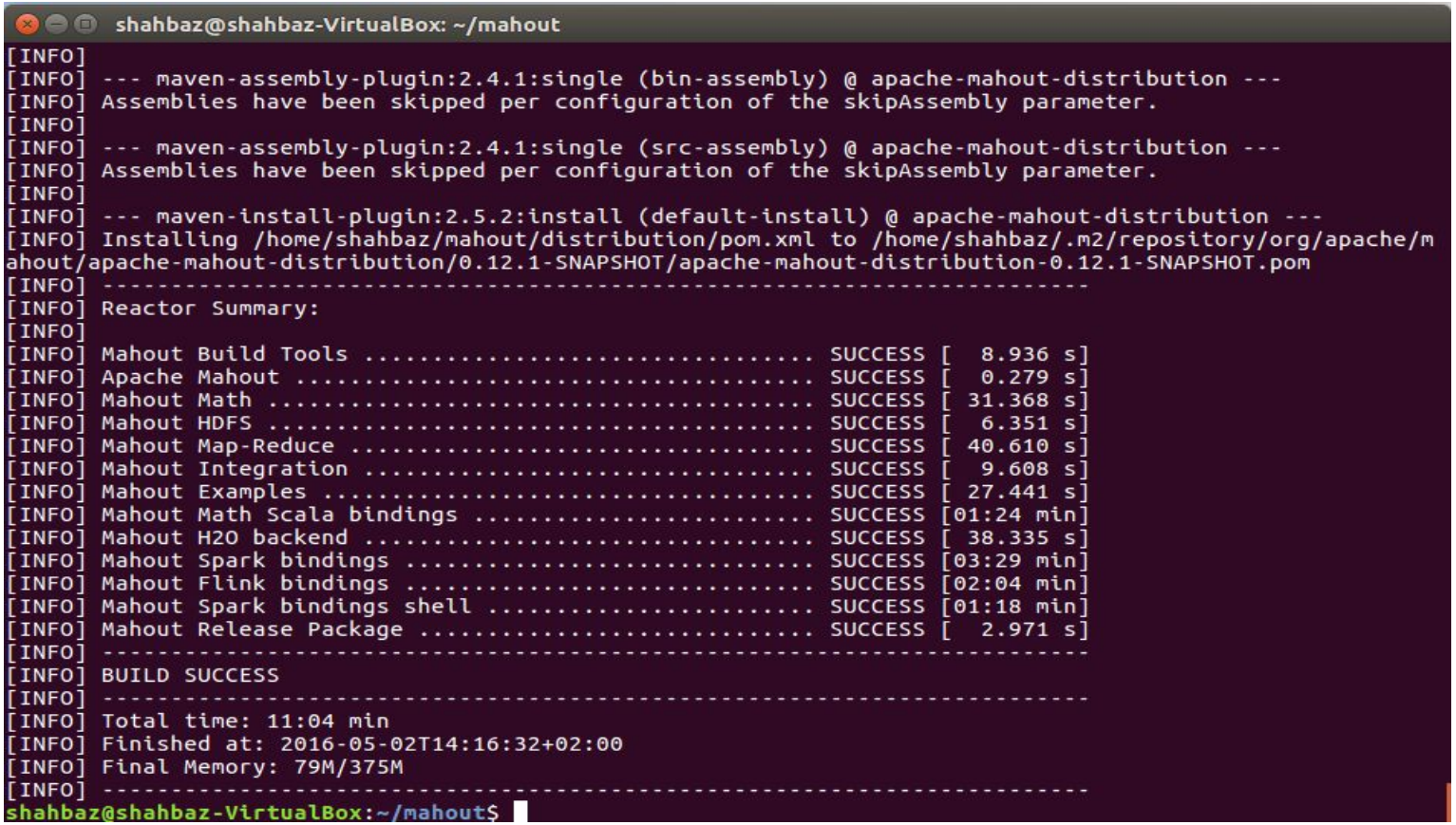
- Download Mahout and extract
- Enter the folder , use **mvn -DskipTests=True clean install**

Installation of Mahout and Spark

.bashrc file should contain the following paths

```
#exporting JAVA, MAHOUT, SCALA and SPARK home values
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
export MAHOUT_HOME=/home/username/mahout
export SCALA_HOME=/usr/local/src/scala
export SPARK_HOME=/home/username/spark-1.6.1
export MASTER=spark://username-VirtualBox:7077
PATH=$JAVA_HOME/bin:$PATH
PATH=$MAHOUT_HOME/bin:$PATH
PATH=$SCALA_HOME/bin:$PATH
PATH=$SPARK_HOME/bin:$PATH
export PATH
```

Installation of Mahout and Spark



In-Core Algebra

- In-core means In-memory which effectively refers to a shift from MapReduce paradigm
- Two basic interfaces for in-core algebra are **Matrix**, **Vector** (Tensor types)
- Vector Implementations
    - DenseVector, RandomAccessSparseVector,SequentialAccessSparseVector
- Matrix Implementations
    - DenseMatrix, SparseRowMatrix, SparseMatrix, DiagonalMatrix

Distributed Row Matrix

$$\mathbf{X} = \begin{bmatrix} 2 & 2 & 10.5 & 10 \\ 1 & 2 & 12 & 12 \\ 1 & 1 & 12 & 13 \\ 2 & 1 & 11 & 13 \\ 1 & 2 & 12 & 11 \\ 2 & 1 & 16 & 8 \\ 6 & 2 & 17 & 1 \\ 3 & 2 & 13 & 7 \\ 3 & 3 & 13 & 4 \end{bmatrix}$$

Image

Distributed Row Matrix

$$\mathbf{X} = \begin{bmatrix} 2 & 2 & 10.5 & 10 \\ 1 & 2 & 12 & 12 \\ 1 & 1 & 12 & 13 \\ 2 & 1 & 11 & 13 \\ 1 & 2 & 12 & 11 \\ 2 & 1 & 16 & 8 \\ 6 & 2 & 17 & 1 \\ 3 & 2 & 13 & 7 \\ 3 & 3 & 13 & 4 \end{bmatrix}$$

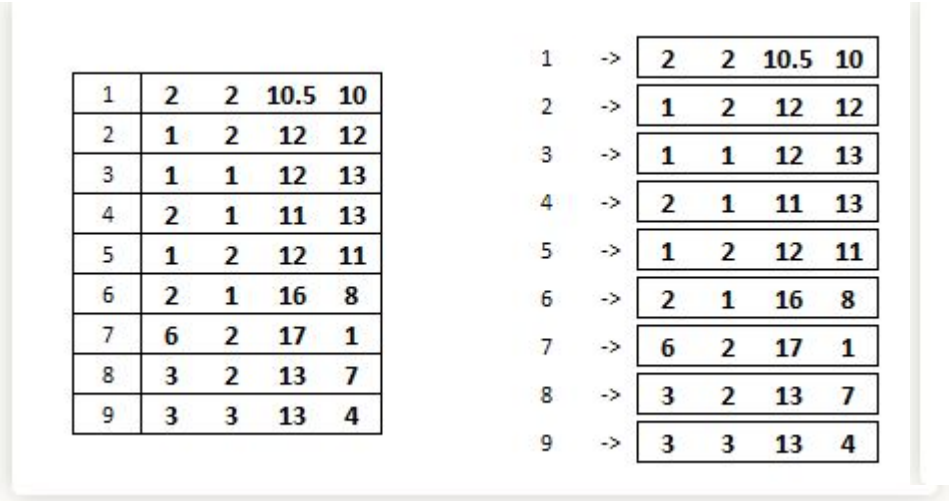
1	2	2	10.5	10
2	1	2	12	12
3	1	1	12	13
4	2	1	11	13
5	1	2	12	11
6	2	1	16	8
7	6	2	17	1
8	3	2	13	7
9	3	3	13	4

1	->	2	2	10.5	10
2	->	1	2	12	12
3	->	1	1	12	13
4	->	2	1	11	13
5	->	1	2	12	11
6	->	2	1	16	8
7	->	6	2	17	1
8	->	3	2	13	7
9	->	3	3	13	4

Index the matrix by rows

Image

Distributed Row Matrix



1	2	2	10.5	10
2	1	2	12	12
3	1	1	12	13
4	2	1	11	13
5	1	2	12	11
6	2	1	16	8
7	6	2	17	1
8	3	2	13	7
9	3	3	13	4

1	->	2	2	10.5	10
2	->	1	2	12	12
3	->	1	1	12	13
4	->	2	1	11	13
5	->	1	2	12	11
6	->	2	1	16	8
7	->	6	2	17	1
8	->	3	2	13	7
9	->	3	3	13	4

1	2	2	10.5	10
2	1	2	12	12
3	1	1	12	13
4	2	1	11	13
5	1	2	12	11
6	2	1	16	8
7	6	2	17	1
8	3	2	13	7
9	3	3	13	4

[1,2,3] ->	2	2	10.5	10
	1	2	12	12
	1	1	12	13
[4,5,6] ->	2	1	11	13
	1	2	12	11
	2	1	16	8
[7,8,9] ->	6	2	17	1
	3	2	13	7
	3	3	13	4

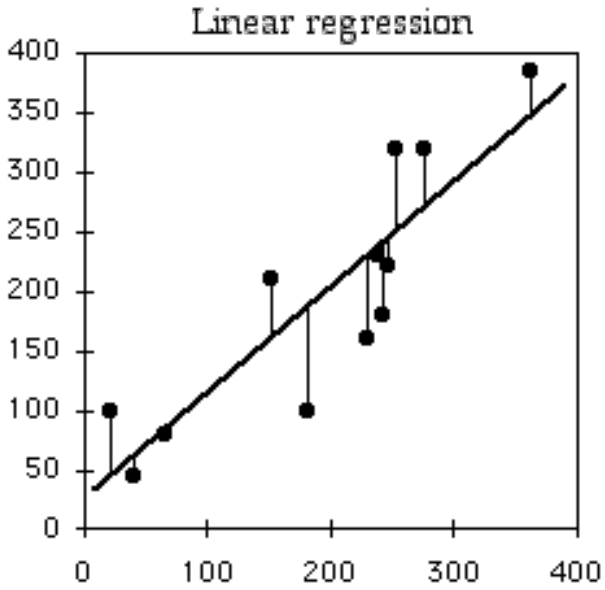
Blocks of the matrix can also be used as index to be distributed

Image

Mahout's interactive Spark shell

- Mahout is a scala DSL, which implicitly is optimized for Spark
- Lets, look at an example for performing Linear regression

Linear Regression



Image

Linear Regression

- Our goal is to find an estimate of **ß** in the equation **y= ßx + c** , where **y** is the target matrix , **x** is the feature matrix, **ß** is the parameter, **c** is noise, which explains the data very well
- **ß** can be estimated using least squares method which minimizes the sum of residual squares between the true target variable and the prediction of the target variable $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}\|^2$.

  We need to solve $\mathbf{X}^{\mathbf{T}}\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^{\mathbf{T}}\mathbf{y} \longrightarrow \hat{\mathbf{w}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}.$

Image

Recommendation Engine

Recommendation Engine

- An application of Collaborative filtering aspect of Machine learning

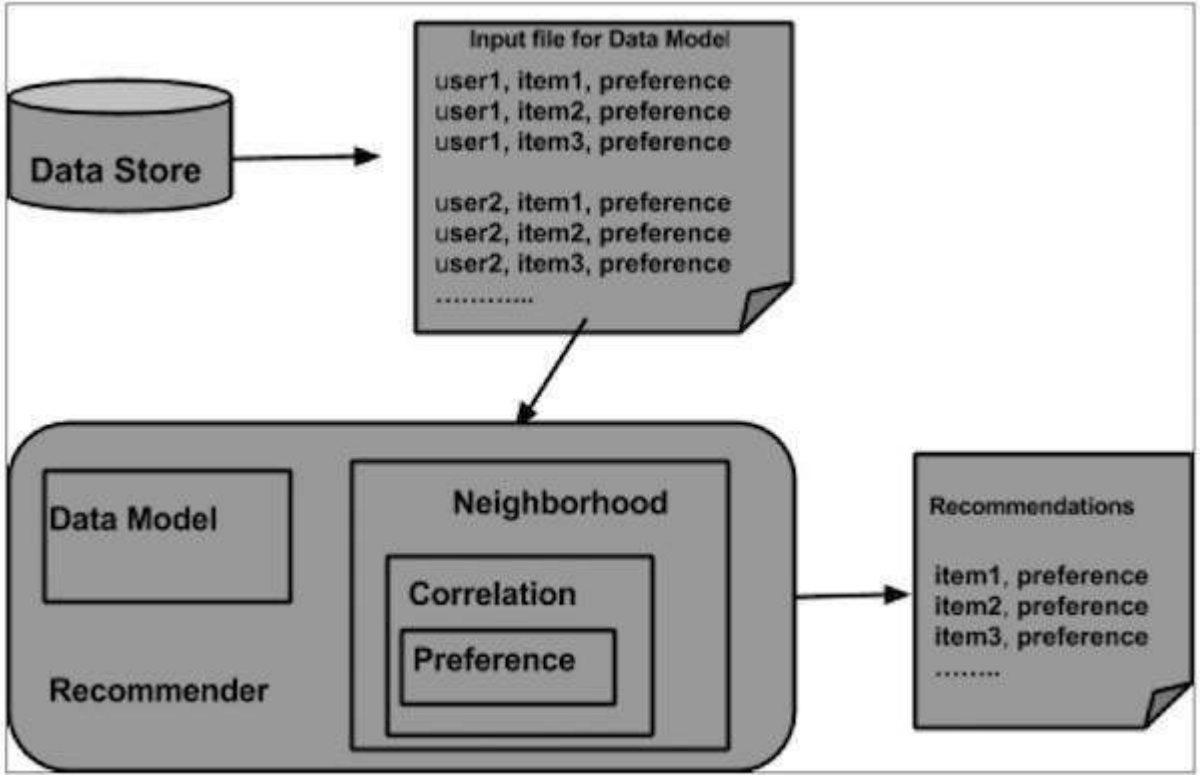
Recommendation Engine

- An application of Collaborative filtering aspect of Machine learning
- Can be based on 3 models
  - UserSimilarity
  - ItemSimilarity
  - Model based

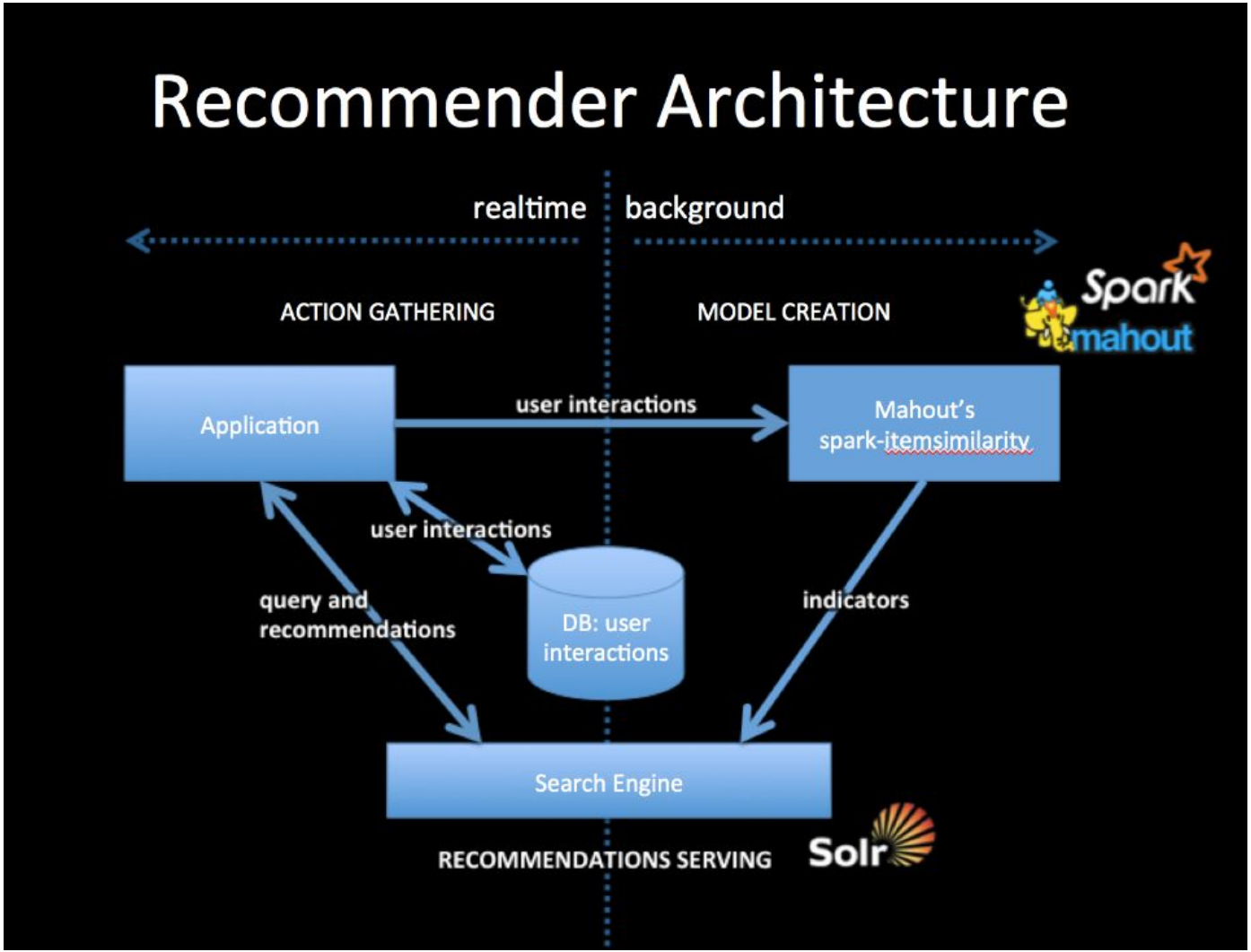
Recommendation Engine

- An application of Collaborative filtering aspect of Machine learning
- Can be based on 3 models
  - UserSimilarity
  - ItemSimilarity
  - Model based
- Key components provided by Mahout for building recommendations are :
  - DataModel, UserSimilarity, ItemSimilarity, Recommender

High level view of a Recommendation Engine



Image


Recommender Architecture
realtime
background
ACTION GATHERING
MODEL CREATION
Spark
mahout
Application
user interactions
Mahout's
spark-itemsimilarity
user interactions
query and
recommendations
DB: user
interactions
indicators
Search Engine
RECOMMENDATIONS SERVING
Solr

Image

Under the hood - Item Recommendation

- Given a set of items someone is known to like, recommend a new set of items which are similar to them

Under the hood - Item Recommendation

- Given a set of items someone is known to like, recommend a new set of items which are similar to them
- Two items can be regarded as similar if two different people like both of them

Under the hood - Item Recommendation

- Given a set of items someone is known to like, recommend a new set of items which are similar to them
- Two items can be regarded as similar if **two different people like both of them**

Algorithm

for every item **i** that **u** has no preference for yet
  for every item **j** that **u** has a preference for
    compute a similarity **s** between **i** and **j**
    add **u**‘s preference for **j**, weighted by **s**, to a running average
return top items, ranked by weighted average

– Owen, S., Anil, R., Dunning T., and Friedman E. 2010.

Under the hood - Item Recommendation

- Item similarity can be measured by Matrix multiplication operations
- Multiply item-item similarity matrix, with item-user vector

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} xa + yb \\ xc + yd \end{bmatrix}$$

ItemSim UserPref

Image

spark-itemsimilarity

spark-itemsimilarity

- Spark counterpart of Mapreduce **itemsimilarity**

spark-itemsimilarity

- Spark counterpart of Mapreduce **itemsimilarity**
- Takes in elements of user interactions(userid, itemid, value), and outputs indicator matrices by comparing user-user interactions

spark-itemsimilarity

- Spark counterpart of Mapreduce **itemsimilarity**
- Takes in elements of user interactions(userid, itemid, value), and outputs indicator matrices by comparing user-user interactions
- Indicator matrix is an item-item matrix where values are **log-likelihood** ratio strengths

spark-itemsimilarity

- Spark counterpart of Mapreduce **itemsimilarity**
- Takes in elements of user interactions(userid, itemid, value), and outputs indicator matrices by comparing user-user interactions
- Indicator matrix is an item-item matrix where values are **log-likelihood** ratio strengths
- Extends co-occurrence to cross-co-occurrence by allowing multi-model interactions

Conclusion

- Apache Mahout fully utilizes the Spark framework to move away from the fallbacks of MapReduce
- It also supports other backend frameworks such as H2o, Flink

Thank you