Introduction to Mahout

Yun Liaw
Vpon Inc.

What is Mahout

A person who drives an elephant(?)

Mahout is…

- A machine learning libraries
- Scalable and robust
  - Scalable to large data
  - Scalable to support business cases
- Based on Hadoop
  - MapReduce Based
  - But will shift to Apache Spark

Mahout's Currently Use Cases

- Classification
- Clustering
- Recommendation

Classification Algorithms

- Naive Bayes / Complementary Naive Bayes
- Random Forest
- Logistic Regression (trained via SGD)
  - Online learning
- Hidden Markov Models
- Multilayer Perceptron

Clustering Algorithms

- Canopy Clustering
  - deprecated, will be removed once Streaming k-Means is stable enough
- L-Means Clustering
- Fuzzy k-Means
- Streaming k-Means
- Spectral Clustering

Recommendation Algorithms

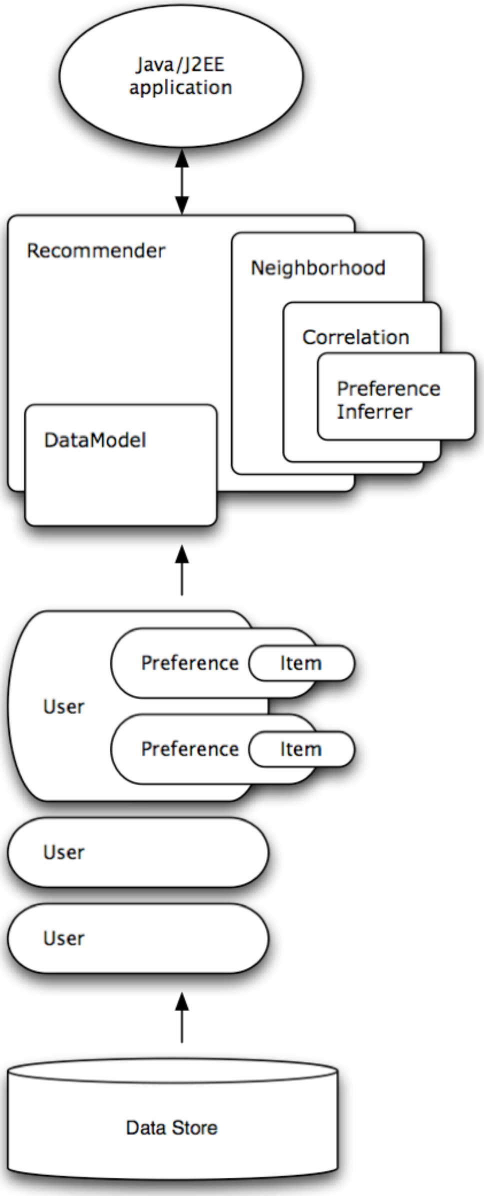
- User-Based Collaborative Filtering
- Item-Based Collaborative Filtering
- Matrix Factorization with Alternating Least Squares (Implicit Feedback)
- Weighted Matrix Factorization, SVD++, Parallel SGD

Others…

- Dimensionality Reduction
  - Singular Value Decomposition
  - Lanczos Algorithm
  - Stochastic SVD
  - Principal Component Analysis (via Stochastic SVD)
- Topic Models
  - Latent Dirichlet Allocation
- Miscellaneous
  - Frequent Pattern Mining
  - RowSimilarityJob
    - compute pairwise similarities between the rows of a matrix
  - ConcatMatrices
    - combine 2 matrices or vectors into a single matrix
  - Collocations
    - find co-locations of tokens in text

Recommender!

Recommendation Architecture



A Java/J2EE Application that invokes..

Mahout Recommendation Engine..

Whose DataModel is based on a set of user preferences..

That are built on the ground of physical datasource

Recommendation in Mahout

- Input: raw data (user preference)
- Output: preferences estimation

- Step1: raw data -> DataModel
- Step2: Recommendation
- Step3: Compute rating estimation
- Step4: Evaluating Recommendation

Recommendation Components

- DataModel Interface
  - Methods for mapping raw data to a Mahout-compliant form
- Similarity Interface
  - Methods to calculate the degree of correlation between two objects (user, item, etc.)
- Neighborhood Interface
  - Methods to define the concept of ‘neighborhood’
- Recommender Interface
  - Methods to implement the recommendation step itself

Component: DataModel

- Example: FileDataModel

Consider a CSV File:

1, 101, 5.0

1, 102, 3.0 ← User 1 has preference 3.0 for item 102

2, 101, 2.0

2, 102, 2.5

2, 103, 3.0

2, 104, 5.0

…

Basic object: Preference

A Triple Object -> (user, item, rating)

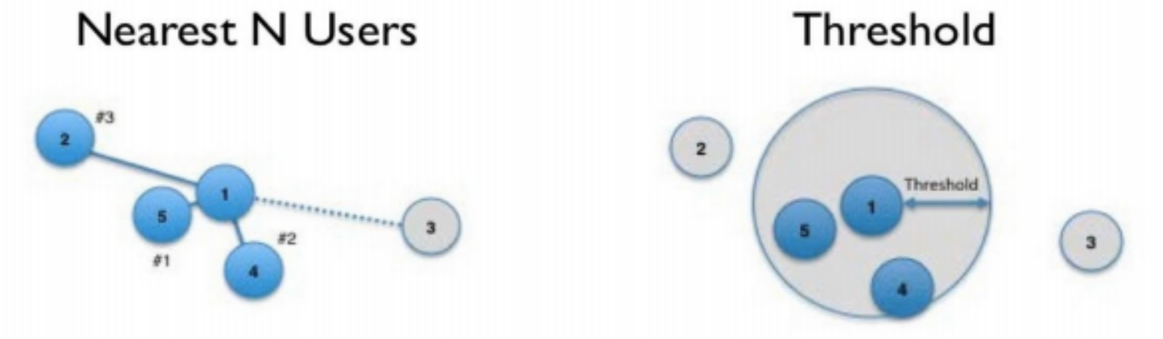
Stored in **UserPreferenceArray**

Component: Similarity

- Many different similarity measures
  - Pearson Correlation
  - Euclidean Distance
  - LogLikelyhood
  - etc.
- Different Similarity influence the **Neighborhood** formation

Component: Neighborhood

- Two definitions of Neighborhood:
  - Nearest N users
    - The first N users with the highest similarity are labeled as ‘neighbors’
  - Thresholds
    - Users whose similarity is above a threshold are labeled as ‘neighbors’



Component: Recommender

- Given a **DataModel**, a **Similarity** measure between objects (user or item), and a definition of **Neighborhood**, a **recommender** can produce the preference estimation as the output
- Recommenders:
  - User-based CF
  - Item-based CF
  - SVD (single value decomposition)
  - ALS (alternating least square)
  - etc.

Evaluation

- Prediction-based estimator
  - (MAE) Mean Absolute Error
  - RMSE (Root Mean Square Error)
- IR (information retrieval) based estimator
  - Precision = TP / (TP + FP)
  - Recall = TP / (TP + FN)
  - F1Measure = 2PR / (P + R)

Let’s Try Mahout!

Environment Setup

- Download Eclipse
  - http://www.eclipse.org/downloads/
- Install Maven plugin for Eclipse
  - http://www.eclipse.org/m2e/download/
- Create Maven Project in Eclipse
- Add Maven Dependency
  - https://mahout.apache.org/users/basics/quickstart.html
- HADOOP IS NOT MANDATORY

JAVADOC

https://builds.apache.org/job/Mahout-Quality/javadoc/

Exercise 1

- Create a Preference object
- Set some preferences
- Print it!

- Hint:
  - GenericUserPreferenceArray
  - Preference

Exercise 2

- Create DataModel that stores multiple user's preference
- PreferenceArray stores preference of a single user
  - HashMap?

- Hint:
  - FastByIDMap

Exercise3

- Create DataModel
- Feed the data from CSV file
- Calculate Pearson Cooreation and Euclidean Distance

- Hint:
  - FileDataModel
  - PearsonCorrelationSimilarity
  - EuclideanDistanceSimilarity

Exercise4

- Create a DataModel
- Feed the DataModel through a CSV file
- Generate Neighborhood
- Generate Recommendations

- Hint:
  - NearestNUserNeighborhood
  - GenericUserBasedRecommender

Exercise5

- Let's start with some real data!
  - Download GroupLens Dataset
    - http://files.grouplens.org/datasets/movielens/ml-100k.zip
- Play this dataset with Exercise3 and Exercise4!
- Tweak the parameters
  - Different similarity measures
  - Different Neighborhood
- But which one is better?
  - We need evaluation!

Exercise6

- Evaluate different recommender configurations on MovieLens data
- Metrics: MAE, RMSE

- Hint:
  - RandomUtils.useTestSeed()
    - To ensure that random seed is the same
  - RecommenderEvaluator
    - AverageAbsoluteDifferenceRecommenderEvaluator (MAE)
    - RMSRecommenderEvaluator (RMSE)
  - RecommenderBuilder
    - To generate the recommender instance

Exercise7

- Evaluate different recommender configurations on MovieLens data
- Metrics: Precision, Recall, F1-Score

- Hint:
  - GenericRecommenderIRStatsEvaluator

Exercise8

- Item-based Recommender

- Hint:
  - ItemSimilarity
  - GenericItemBasedRecommender

Running on Hadoop

- Package into Jar
  - – Maven will help A LOT
- Using Hadoop jar command to run

```
hadoop jar your_jar.jar \
org.apache.mahout.cf.taste.hadoop.pseudo.RecommenderJob \
-Dmapred.input.dir=hdfs/position/for/input/data \
-Dmapred.output.dir=output \
--recommenderClassName \
your.own.implement.very.special.AwsomeRecommender
```

Future

- Hadoop -> Spark
  - http://spark.apache.org/
  - Less code with Scala implementation
  - Faster computation
    - Utilize memories in cluster
  - Machine learning is built in core (MLLib)
  - With streaming processing (Spark Streaming)
  - With Hive-compatible SQL (Spark SQL)
  - With graph algorithms (Graph X)
  - AWSOME!

Mahout Resources

- Official Site
  - https://mahout.apache.org/
- JAVADOC
  - https://builds.apache.org/job/Mahout-Quality/javadoc/
- Github
  - https://github.com/apache/mahout
- Books
  - Mahout in Action (2011)
  - Apache Mahout Cookbook (2013)