

Adam Warski

Writing recommender systems with Java

An introduction

Agenda

- ❖ What is a recommender?
- ❖ Types of recommenders, input data
- ❖ Recommendations with Mahout
- ❖ Recommendations with Spark
- ❖ Custom recommenders

Who Am I?

- ❖ **Day:** coding @ SoftwareMill
- ❖ **Afternoon:** playgrounds, Duplos, etc.
- ❖ **Evenings:** blogging, open-source
 - ❖ Original author of Hibernate Envers
 - ❖ MacWire, ElasticMQ, Veripacks
 - ❖ <http://www.warski.org>

Me + recommenders

- ❖ Yap.TV
- ❖ Recommends shows to users basing on their favorites
 - ❖ from Yap
 - ❖ from Facebook
- ❖ Some business rules



Some background

What can be recommended?

- ❖ Items to users
- ❖ Items to items
- ❖ Users to users
- ❖ Users to items

Input data

- ❖ (user, item, rating) tuples
 - ❖ rating: e.g. 1-5 stars
- ❖ (user, item) tuples
 - ❖ **unary**

Input data

- ❖ Implicit
 - ❖ clicks
 - ❖ reads
 - ❖ watching a video
- ❖ Explicit
 - ❖ explicit rating
 - ❖ favorites

Prediction vs recommendation

- ❖ Recommendations

- ❖ suggestions

- ❖ top-N

- ❖ Predictions

- ❖ ratings

Content-based recommenders

- ❖ Define features and feature values
- ❖ Describe each item as a vector of features
- ❖ Create a user vector
- ❖ Prediction: cosine between user and item vectors

Collaborative Filtering

- ❖ Something more than search keywords
 - ❖ tastes
 - ❖ quality
- ❖ Collaborative: data from many users

User-User CF

- ❖ Measure similarity between users
- ❖ Prediction: weighted combination of existing ratings

User-User CF

- ❖ Domain must be scoped so that there's agreement
- ❖ Individual preferences should be stable

User similarity

❖ Many different metrics

❖ Pearson correlation: $w_{au} = \frac{\sum_{i=1..m} (r_{ai} - \bar{r}_a)(r_{ui} - \bar{r}_u)}{\sigma_a * \sigma_u}$

User neighbourhood

- ❖ Threshold on similarity
- ❖ Top-N neighbours
 - ❖ 25-100: good starting point

Predictions in UU-CF

$$P_{ai} = \bar{r}_a + \frac{\sum_{u=1..n} (r_{ui} - \bar{r}_u) * w_{au}}{\sum_{u=1..n} w_{au}}$$

Item-Item CF

- ❖ Similar to UU-CF, only with items
- ❖ Item similarity
 - ❖ Pearson correlation on item ratings
 - ❖ co-occurrences

Evaluating recommenders

- ❖ How to tell if a recommender is good?
 - ❖ compare implementations
 - ❖ are the recommendations ok?
- ❖ Business metrics
 - ❖ what leads to increased sales

Leave one out

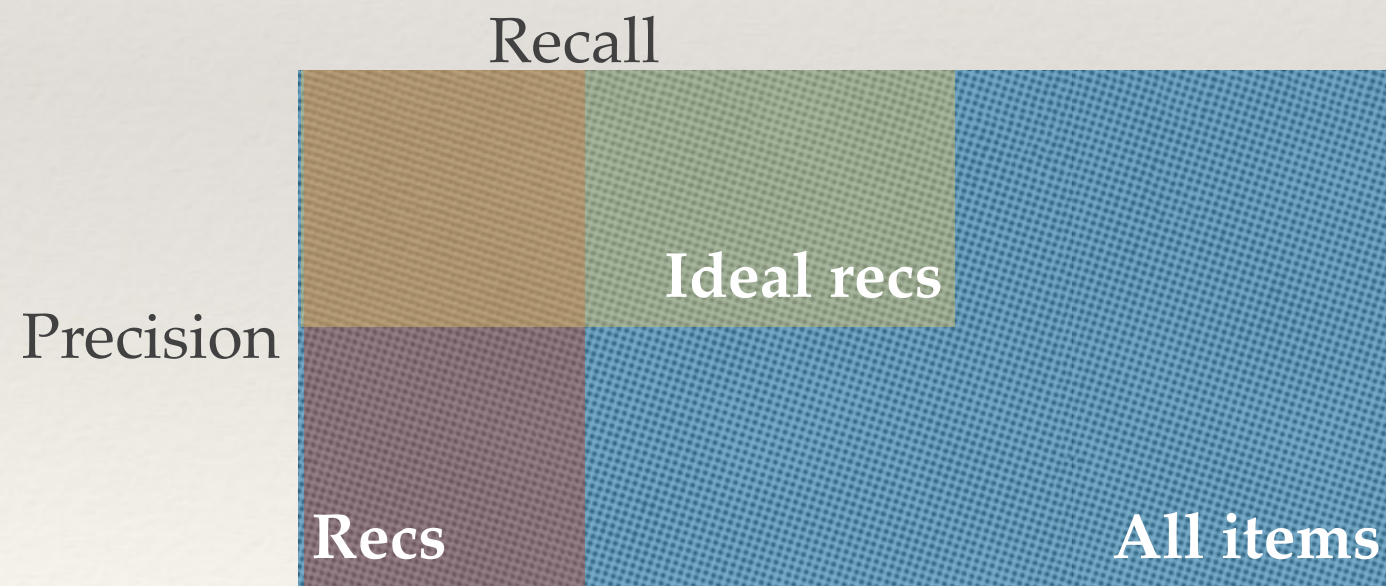
- ❖ Remove one preference
- ❖ Rebuild the model
- ❖ See if it is recommended
- ❖ Or, see what is the predicted rating

Some metrics

- ❖ Mean absolute error: $\frac{\sum_i |R_i - P_i|}{\text{ratings}}$
- ❖ Mean squared error

Precision/recall

- ❖ Precision: % of recommended items that are relevant
- ❖ Recall: % of relevant items that are recommended
- ❖ Precision@N, recall@N



Problems

- ❖ Novelty: hard to measure
 - ❖ leave one out: punishes recommenders which recommend **new** items
- ❖ Test on humans
 - ❖ A/B testing

Diversity/serendipity

- ❖ Increase diversity:
 - ❖ top-n
 - ❖ as items come in, remove the ones that are too similar to prior items
- ❖ Increase serendipity:
 - ❖ downgrade popular items

Netflix challenge

- ❖ \$1m for the best recommender
- ❖ Used mean error
- ❖ Winning algorithm won on predicting low ratings

Mahout single-node



Mahout

IN ACTION

Sean Owen
Robin Anil
Ted Dunning
Ellen Friedman



Requires Adobe Acrobat Reader to play audio and video links

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (<http://movielens.org>). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

MovieLens 100k

100,000 ratings from 1000 users on 1700 movies.

- [README.txt](#)
- [ml-100k.zip](#)
- [Index of unzipped files](#)

MovieLens 1M

1 million ratings from 6000 users on 4000 movies.

- [README.txt](#)
- [ml-1m.zip](#)

MovieLens 10M

10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

- [README.html](#)
- [ml-10m.zip](#)

Datasets

[MovieLens](#)

[HetRec 2011](#)

[WikiLens](#)

[Book-Crossing](#)

[Jester](#)

[EachMovie](#)

Mahout single-node

- ❖ User-User CF
- ❖ Item-Item CF
- ❖ Various metrics
 - ❖ Euclidean
 - ❖ Pearson
 - ❖ Log-likelihood
- ❖ Support for evaluation

Let's code

Mahout multi-node

- ❖ Based on Hadoop
- ❖ Pre-computed recommendations
- ❖ Item-based recommenders
 - ❖ Co-occurrence matrix
- ❖ Matrix decomposition
 - ❖ Alternating Least Squares

Running Mahout w/ Hadoop

```
hadoop jar mahout-core-0.9-job.jar  
  org.apache.mahout.cf.taste.hadoop.item.RecommenderJob  
  --booleanData  
  --similarityClassname SIMILARITY_LOGLIKELIHOOD  
  --output output  
  --input input/data.dat
```

Running Mahout @EMR

Create a New Job Flow

Cancel



Specify the location in Amazon S3 of your JAR. Hadoop executes the JAR. You can specify its main class in its manifest. If you don't you must specify a class name as the first argument of the JAR.

JAR Location*:

JAR Arguments*:

[< Back](#)

Continue



* Required field

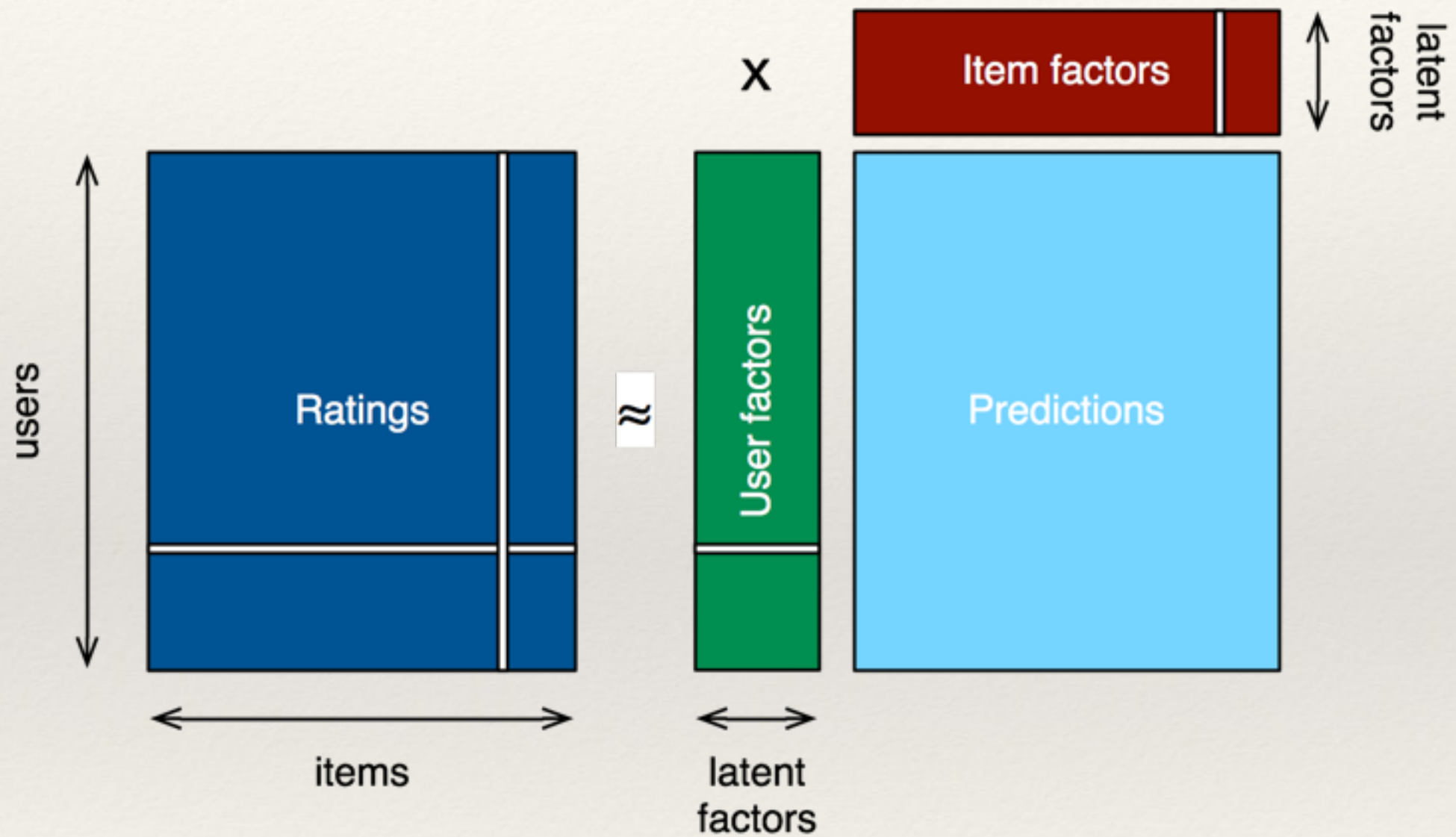
Co-occurrence on a single node

- ❖ In-memory computations
- ❖ Use a disk-based cache
 - ❖ e.g. MapDB
- ❖ Handles ~600 million datapoints easily

Matrix decomposition

- ❖ Dimensionality reduction
- ❖ A - user-item matrix
- ❖ Factor into: user-feature, weights, feature-item
- ❖ Alternating Least Squares
 - ❖ $A = XY^T$
- ❖ SVD
 - ❖ $A = U\Sigma V^T$

ALS



Apache Spark

- ❖ “in-memory Hadoop”
- ❖ MLlib
- ❖ Matrix factorization
 - ❖ Alternating Least Squares



Links

- ❖ <http://www.warski.org>
- ❖ <http://mahout.apache.org/>
- ❖ <https://spark.apache.org>
- ❖ <https://github.com/adamw/mahout-pres>

Thanks!

- ❖ Questions?
- ❖ Stickers ->
- ❖ adam@warski.org

