

Building Recommendation Systems in Python using Apache Spark

December 3, 2016
Galvanize, Seattle



Recommender Systems



Recommending **products**
depending on your previous **purchases**



Recommending **movies**
depending on your previous **views & ratings**



Recommending **content, updates, posts**
depending on your previous **engagement**
(likes, reactions, shares, clicks)

**Recommender systems
are a key asset
for these industries**

**Any increase/decrease
in performance
can have a
direct impact
on churn / c.t.r.**

**A key component of the
economy of attention**

General Principles of Recommendation



User-User similarity

Recommend content that has been liked/bought/viewed by people similar to you.

This similarity is based on commonality of likes/buys/views btw two users.

The diagram shows a rating matrix with columns for Movie 1, Movie 2, Movie 3, ..., Movie m and rows for User 1, User 2, User 3, ..., User n. A blue arrow points to the User 2 row, and a blue box highlights the row for User 2. An orange arrow points to the Movie 2 column, and an orange box highlights the column for Movie 2. The matrix data is as follows:

	Movie 1	Movie 2	Movie 3	...	Movie m
User 1	4	?	?	...	1
User 2	3	3	2	?	2
User 3	?	3	?	?	?
...	?	?	...
User n	?	5	4	...	5

Item-Item similarity

Recommend content that is similar to the content you've liked/bought/viewed.

This similarity is based on commonality of likes/buys/views btw two items.

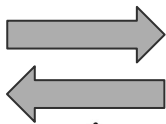
Pb: this is SPARSE and HUGE

Most items are unrated
(1-2% of ratings on Netflix)

Matrix Factorization in a nutshell



	Movie 1	Movie 2	Movie 3	...	Movie m
User 1	4	?	?	...	1
User 2	3	3	2	?	2
User 3	?	3	?	?	?
...	?	?	...
User n	?	5	4	...	5

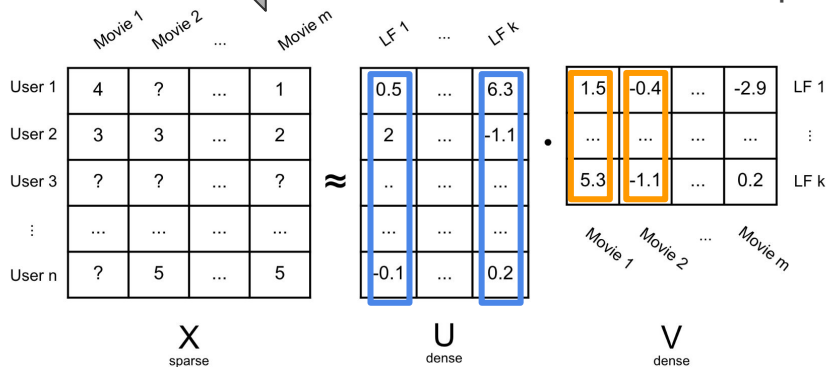


Factorization: finding U, V from X

Reconstruction: computing X from U, V

Factorization:

Finding “latent features” that let us factorize this sparse matrix as the product of two dense matrices



Matrix Factorization : Alternating Least Squares



1) WHAT are Latent Vectors ?

Item vectors and **user vectors** used to decompose X as $U.V$

2) What are “GOOD” Latent Vectors ?

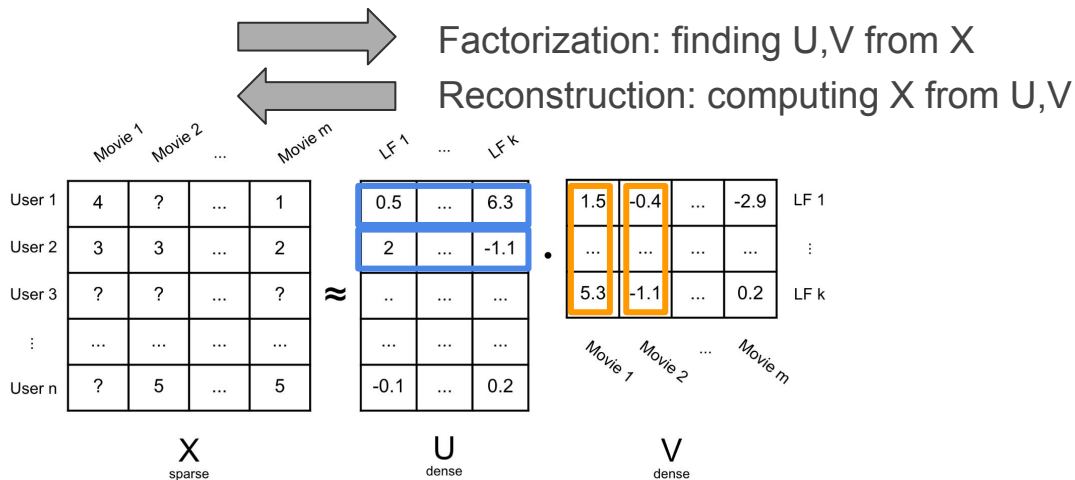
A couple of **item-vectors** **user-vectors** that minimize the error in reconstruction of X

3) HOW TO find “Good” Latent Vectors ?

Fix **item-vectors** and optimize **user-vectors** based on reconstruction error, then do the opposite

Factorization:

Finding “latent features” that let us factorize this sparse matrix as the product of two dense matrices





Building a recommender in Apache Spark