

Matrix Factorization for Recommendation Engines

Dan Becker
April 1, 2015

Goals

- I. Conceptual Understanding of Latent Factor Models for Recommender Systems
- II. High level understanding of Latent Factor Model implementation using FunkSVD algorithm
- III. Ability to tune model performance

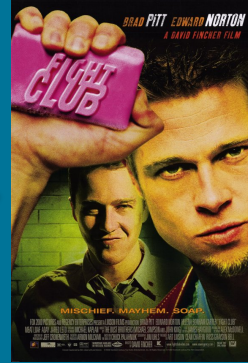
The Setup

User

Movie					
	A	B	C	D	...
Alice	1	?	2	?	
Bob	?	2	3	4	
Charlie	3	?	1	5	
Dan	?	2	?	?	
...					

The Problem With Item-Item Recommendations

I Like Surprising Endings



Violent movie recommendations



Movies (And Everything Else) Have Many Attributes

Movie Attributes

- Action
- Comedy
- Drama
- ...
- Tom Hanks
- Brad Pitt
- Jeff Bridges
- ...
- Happy ending
- Sad Ending
- ...
- Movie length
- Subtitles
- ...

A Familiar Looking Model

$$\begin{aligned}\text{Predicted rating} = & B_0 + B_1 * \text{level of action} \\ & + B_2 * \text{level of comedy} \\ & + B_3 * \text{level of drama} \\ & + \dots \\ & + B_n * \text{Tom Hanks} \\ & + B_{n+1} * \text{Brad Pitt} \\ & + \dots \\ & + B_k * \text{length} \\ & + \dots \\ & + e\end{aligned}$$

Singular Value Decomposition

User Matrix

Item Matrix

$$A_{m \times n} = \begin{bmatrix} | & | & | & | \\ \hline & & & \\ \hline | & | & | & | \\ \hline & & & \\ \hline \end{bmatrix}_{m \times k} \begin{bmatrix} \diagdown \\ \hline \\ \diagup \end{bmatrix}_{k \times k} \begin{bmatrix} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{bmatrix}_{k \times n}$$

SVD Doesn't Work With Missing Values

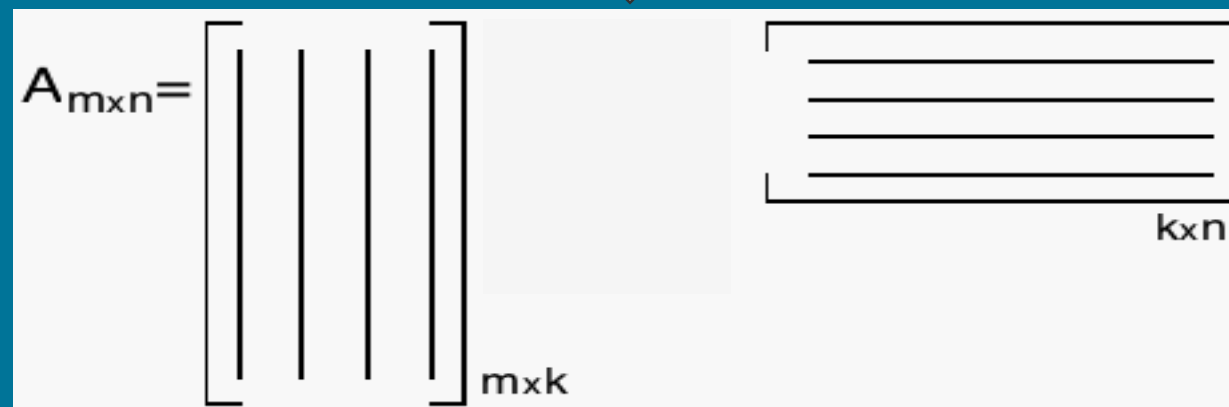
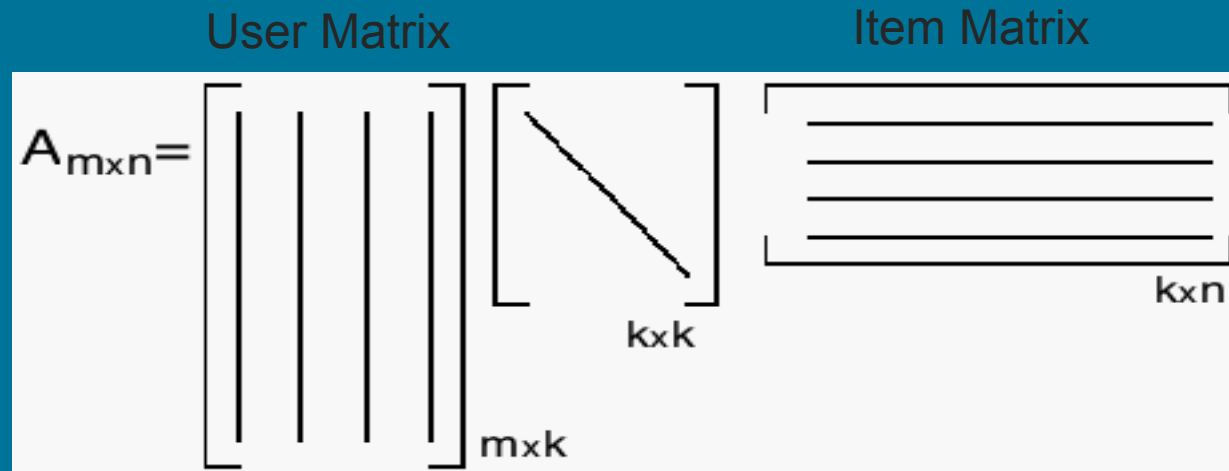


SO CLOSE

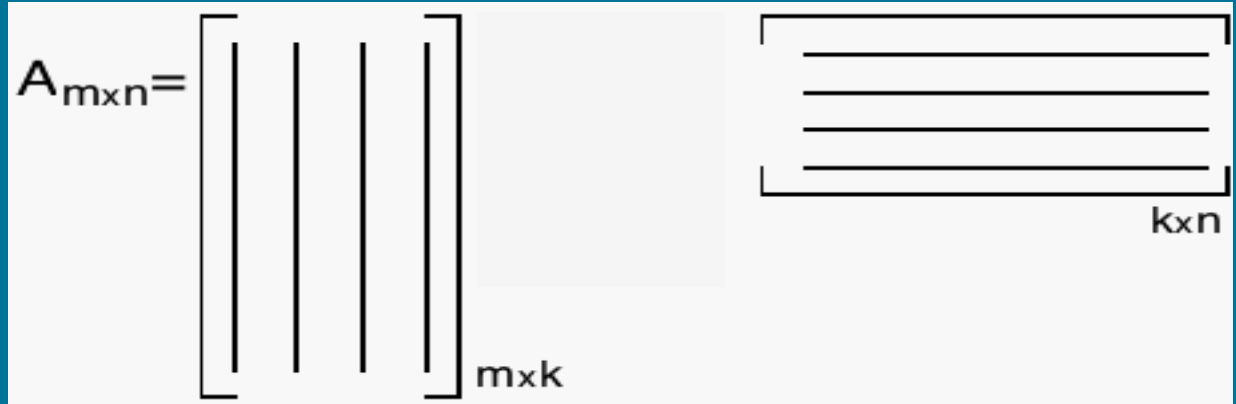
And yet so far away

Start By Simplifying
the Problem

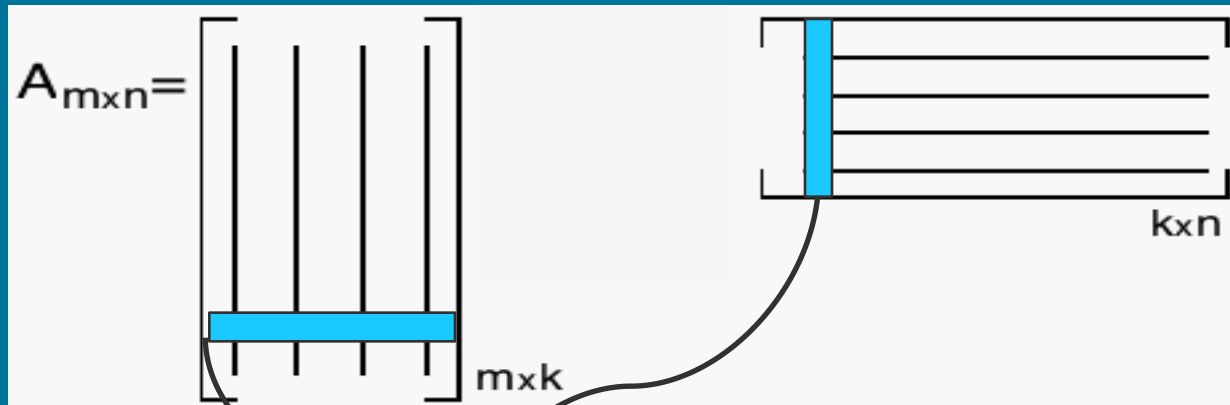
UV Decomposition



Evaluate Where we Can

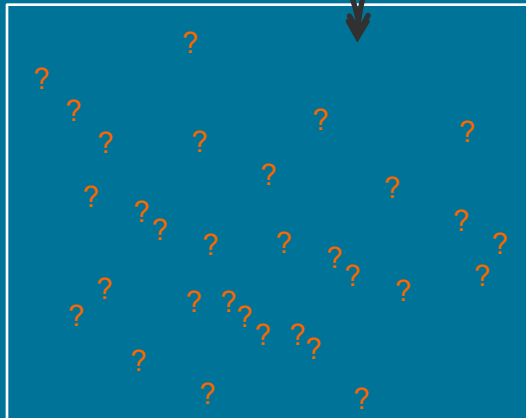


Every Rating is Product of Two Vectors



User

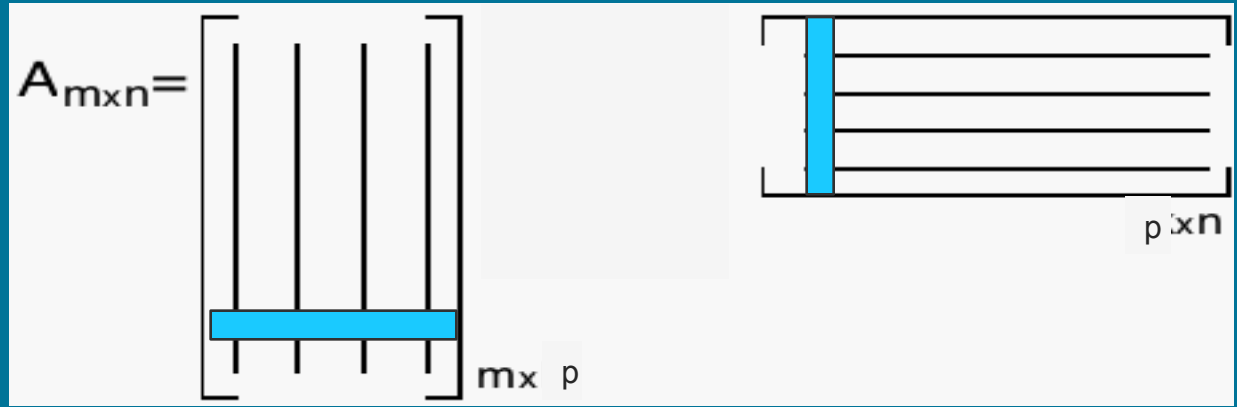
Item



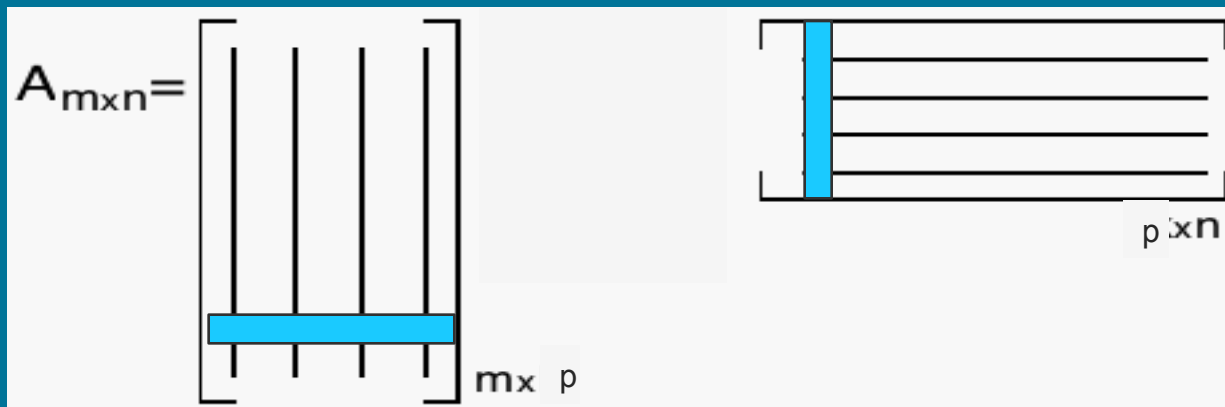
We can perfectly replicate every observed rating.

Want to approximate it by reducing k to avoid overfitting.

We Can Optimize Matrices



We Can Optimize Matrices



$$\bar{R}_{ai} = \mu + b_a + b_i + u_a + v_i$$

$$e_{ai} = \bar{R}_{ai} - R_{ai}$$

SGD Update Rules for Mean Squared Error

Without Regularization

$$\Delta u_{af} = \lambda e_{ai} v_{if}$$

$$\Delta v_{if} = \lambda e_{ai} u_{af}$$

With Regularization

$$\Delta u_{af} = \lambda (e_{ai} v_{if} - \gamma u_{af})$$

$$\Delta v_{if} = \lambda (e_{ai} u_{af} - \gamma v_{if})$$

FunkSVD Algorithm

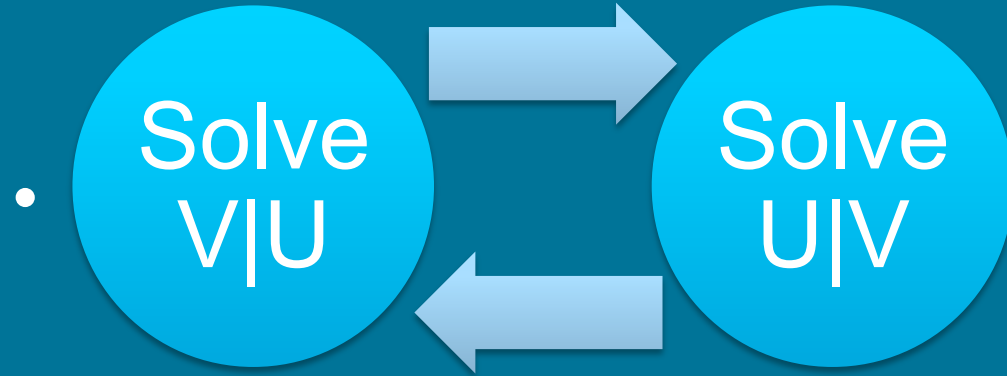
```
Initialize U and V matrices
for each feature:
    while objective function improves:
        for each user:
            for each item that user has rated:
                predict rating
                calculate error
                update U element for that user-feature pair
                update V element for that item-feature pair
```

Hints on Metaparameters:

- Learning rate around 0.001
- Regularization factor around 0.01

Alternating Least Squares SVD

- Initialize U and V



until convergence

Alternating Least Squares vs Stochastic Gradient Descent

ALS

SGD

Parallelizes
Better

Usually
faster

Available in
Spark/
MLlib

Anecdotes
of better
results

