CMPSC 448: Machine Learning

Lecture 15. Matrix Factorization

Rui Zhang

Fall 2021

PennState

Outline

- Recommender systems and The Netflix challenge
- Collaborative Filtering
- Latent Factor Models
- Matrix Factorization

Netflix challenge: the story

Netflix launched a competition in 2006 to try to improve their system for recommending movies to their customers.

The Netflix dataset has 17,770 movies and 480,189 customers

Each user rates some subset of the movies with a score in have rated some of the movies on a scale from 1 to 5, where 1 is worst and 5 is best.

On average, each user rates around 200 movies, though the variance is high (e.g., some user has rated more than 17000 movies! :-))

The goal is to predict the ratings for unrated movies, so as to better recommend movies to customers.

The competition was held starting in 2006, with the winner being the first algorithm that could improve this Root Mean Square Error (RMSE) by at least 10%.

The input rating data

Here is how rating data looks like (a partially observed matrix with a row for every user and a column for every movie)

The data matrix is very sparse with “only” 100 million (1%) of the ratings present in the training set!

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmahill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				
Cinematch score - RMSE = 0.9525				

There are currently 51051 contestants on 41305 teams from 186 different countries.
We have received 44014 valid submissions from 5169 different teams; 0 submissions in the last 24 hours.

Questions about interpreting the leaderboard? Please read this.

The Netflix competition leaderboard at the close of the competition.

The competition was finally won in 2009 by a large group of researchers called “Bellkor’s Pragmatic Chaos,” which was the combined effort of three individual groups where **latent factor models** was the key building blocks!

Problem Formalization

There are n users 

There are m movies (items) 

Problem Formalization

2	1	4	4
3	2	1	5
1	2	4	2
5	3	1	2
3	1	5	2

There are n users

There are m movies (items) 

The unknown rating matrix:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

Problem Formalization

There are n users

There are m movies (items) 

The unknown rating matrix:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

But we only observed subset of ratings:

$$\Omega = \{(i, j) \text{ such that } r_{i,j} \text{ is observed}\}$$

Goal: predict unknown ratings!

2	1	4	4
3	2	1	5
1	2	4	2
5	3	1	2
3	1	5	2

2	?	4	?
3	?	1	?
?	2	?	2
?	3	1	?
3	?	5	2

Example: Netflix challenge

Training data

- 100 million ratings (6 years of data: 2000-2005)
- 480,189 users
- 17,770 movies
- only 1.2% observed

Test data

- Last few ratings of each user (2.8 million)

Evaluation criterion

- Root Mean Square Error (RMSE): sqrt($\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} (\text{prediction}_{i,j} - r_{i,j})^2$)

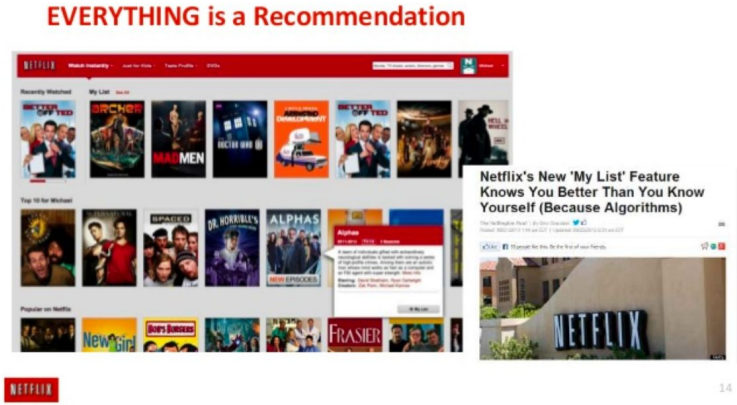
Netflix's system RMSE: 0.9514

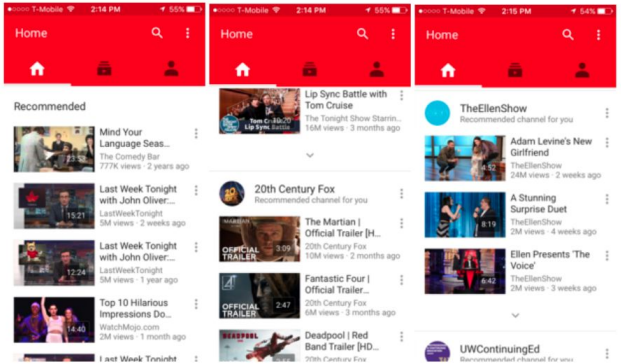
Competition: 2,700+ teams

$1 million prize for 10% improvement on Netflix

Everyday examples

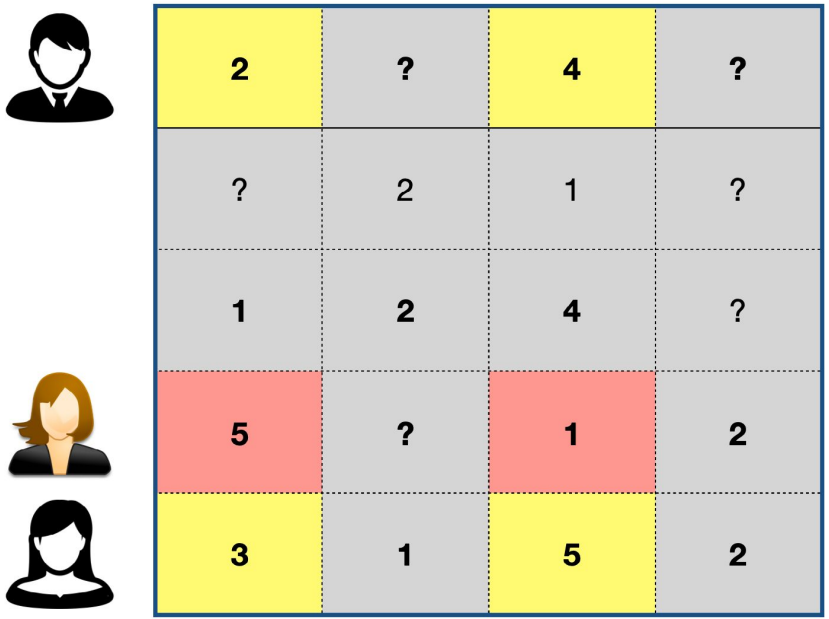
"Judging by Amazon's success, the recommendation system works. The company reported a 29% sales increase to $12.83 billion during its second fiscal quarter, up from $9.9 billion during the same time last year. A lot of that growth arguably has to do with ***the way Amazon has integrated recommendations*** *into nearly every part of the purchasing process…"*





Collaborative filtering

1. Users who rate the same movie similarly are likely to be similar.

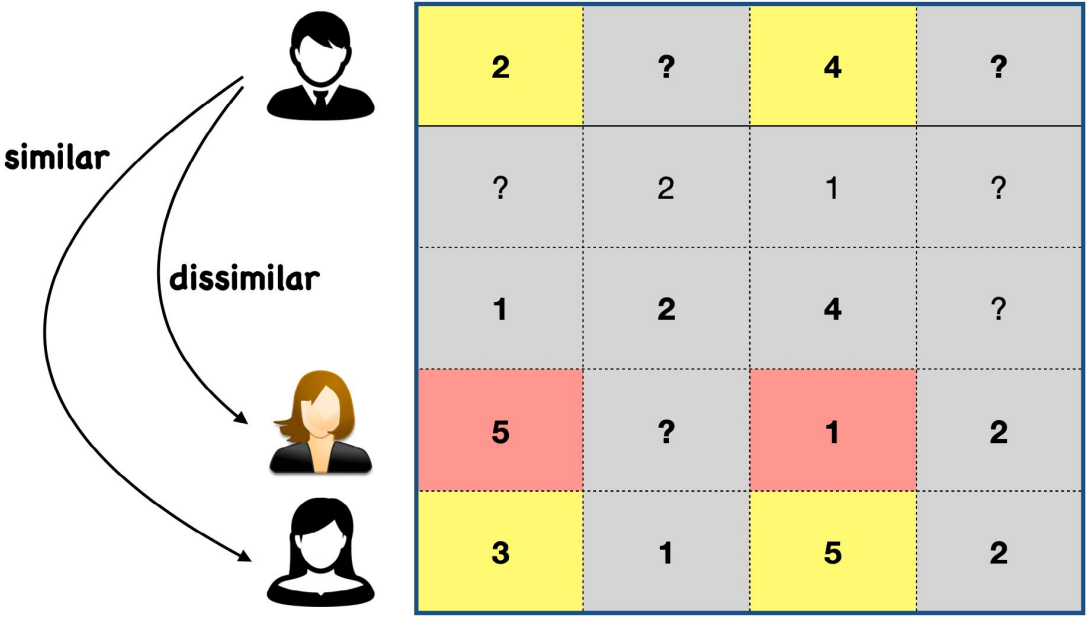


2	?	4	?
?	2	1	?
1	2	4	?
5	?	1	2
3	1	5	2

Model every user by a vector as his/ her ratings for all movies

Collaborative filtering

1. Users who rate the same movie similarly are likely to be similar.



Model every user by a vector as his/ her ratings for all movies

We can compute the similarity between users by computing the similarity of their rating vectors (e.g., Euclidean distance)

Collaborative filtering

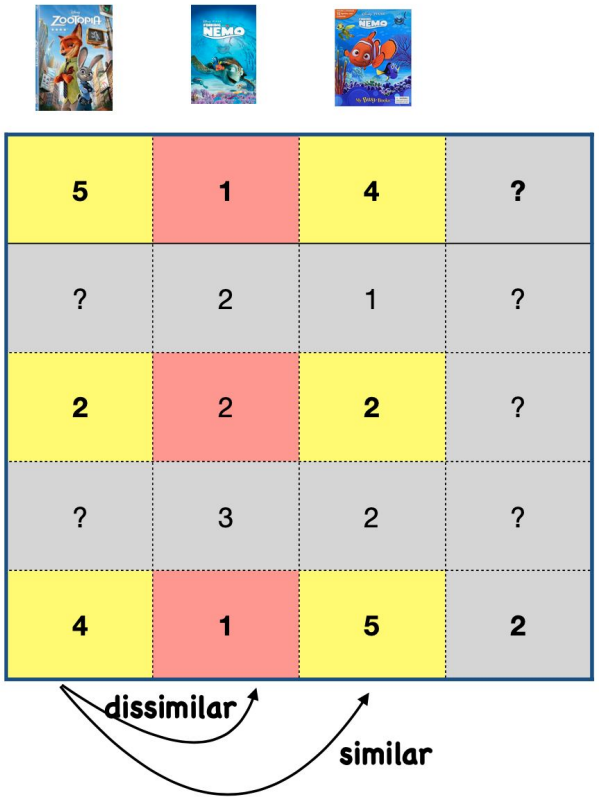
2. Movies that are rated similarly by the same user are likely to be similar.

5	**1**	**4**	**?**
?	2	1	?
2	2	**2**	?
?	3	2	?
4	**1**	**5**	**2**

Model every movie by a vector as the ratings issued for movie by all users

Collaborative filtering

2. Movies that are rated similarly by the same user are likely to be similar.



Model every movie by a vector as the ratings issued for movie by all users

Compute the similarity between movies by computing the similarity of their rating vectors (e.g., Euclidean distance)

Collaborative filtering

1. Users who rate the same movie similarly are likely to be similar.
2. Movies that are rated similarly by the same user are likely to be similar.

Personal tastes are correlated! If Alice and Bob both like X and Alice likes Y, then Bob is more likely to like Y as well! [collaborative]

There may be a wealth of information just in the ratings themselves

Collaborative filtering vs Content-based filtering

1. Users who rate the same movie similarly are likely to be similar.
2. Movies that are rated similarly by the same user are likely to be similar.

Personal tastes are correlated! If Alice and Bob both like X and Alice likes Y, then Bob is more likely to like Y as well! [collaborative]

There may be a wealth of information just in the ratings themselves

- But we cannot make recommendation for new items.
- This is in contrast to **content-based filtering**.

Question: How can we formalize this intuition to learn from observed ratings and predict unknown ratings?

How do you rate a movie?

How do you rate a movie?

FACTOR	WEIGHT
Comedy	0
Science	0.6
Action	-0.5
…	

You have few factors in your mind that you most care about movies and a weight for each factor.

How do you rate a movie?

You have few factors in your mind that you most care about movies and a weight for each factor.

FACTOR	WEIGHT
Comedy	0
Science	0.6
Action	-0.5
…	

WEIGHT	FACTOR
-1	Comedy
2	Science
0	Action
	…

The movie can be also get a weight for the factors.

How do you rate a movie?

You have few factors in your mind that you most care about movies and a weight for each factor.

FACTOR	WEIGHT	WEIGHT	FACTOR
Comedy	0	-1	Comedy
Science	0.6	2	Science
Action	-0.5	0	Action
…			…

Rating would be:

$$0 * (-1) + 0.6 * 2 + (-0.5) * 0 = 1.2$$

The movie can be also get a weight for the factors.

How do you rate a movie?

A different user with different taste weights the factors differently!

FACTOR	WEIGHT	WEIGHT	FACTOR
Comedy	-2	-1	Comedy
Science	3	2	Science
Action	0	0	Action
…			…

Rating would be:

$$(-2) * (-1) + 3 * 2 + 0 * 0 = 4$$

The movie can be also get a weight for the factors.

How you rate a movie?

Let assume we have k factors in total for all users

How you rate a movie?

Let assume we have k factors in total for all users

We can represent the user's and movie's weights for all factors as a vector:

$$\boldsymbol{u} = \begin{bmatrix} u_1(\text{weight for 1st factor}) \\ u_2(\text{weight for 2nd factor}) \\ \vdots \\ u_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k$$

How you rate a movie?

Let assume we have k factors in total for all users

We can represent the user's and movie's weights for all factors as a vector:

$$\boldsymbol{u} = \begin{bmatrix} u_1(\text{weight for 1st factor}) \\ u_2(\text{weight for 2nd factor}) \\ \vdots \\ u_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k \quad \boldsymbol{v} = \begin{bmatrix} v_1(\text{weight for 1st factor}) \\ v_2(\text{weight for 2nd factor}) \\ \vdots \\ v_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k$$

How you rate a movie?

Let assume we have k factors in total for all users

We can represent the user's and movie's weights for all factors as a vector:

$$\boldsymbol{u} = \begin{bmatrix} u_1(\text{weight for 1st factor}) \\ u_2(\text{weight for 2nd factor}) \\ \vdots \\ u_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k \quad \boldsymbol{v} = \begin{bmatrix} v_1(\text{weight for 1st factor}) \\ v_2(\text{weight for 2nd factor}) \\ \vdots \\ v_k(\text{weight for } k\text{th factor}) \end{bmatrix} \in \mathbb{R}^k$$

Then, the rating of the user for the movie is just:

$$\textbf{RATING}_{\text{  }} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{f=1}^{k} u_f v_f$$

From factors to ratings

If we know the factor weight for all movies and users, then we can compute the ratings

Given: $(n = 6, m = 12, k = 3)$

From factors to ratings

If we know the factor weight for all movies and users, then we can compute the ratings

Given: $(n = 6, m = 12, k = 3)$

$\boldsymbol{u}_1$	0.52	0.19	0.90
$\boldsymbol{u}_2$	0.25	1.17	0.56
$\boldsymbol{u}_3$	1.11	0.52	0.13
$\boldsymbol{u}_4$	2.21	1.04	0.26
$\boldsymbol{u}_5$	0.51	2.35	1.11
$\boldsymbol{u}_6$	1.05	0.38	1.79

$\boldsymbol{U}$

$\boldsymbol{v}_1$	$\boldsymbol{v}_2$				...						$\boldsymbol{v}_{12}$
1.56	0.20	0.77	0.44	0.53	1.17	1.23	0.20	1.59	0.61	0.27	0.63
0.47	1.31	-0.15	1.33	0.84	1.01	1.16	1.31	0.34	0.44	1.02	0.40
0.11	0.72	1.77	0.58	0.62	1.29	0.15	0.71	0.37	0.66	1.30	0.67

$\boldsymbol{V}^\top$

From factors to ratings

If we know the factor weight for all movies and users, then we can compute the ratings

Given: $(n = 6, m = 12, k = 3)$

u_1	0.52	0.19	0.90
u_2	0.25	1.17	0.56
u_3	1.11	0.52	0.13
u_4	2.21	1.04	0.26
u_5	0.51	2.35	1.11
u_6	1.05	0.38	1.79

$\boldsymbol{U}$

v_1	v_2				...						v_{12}
1.56	0.20	0.77	0.44	0.53	1.17	1.23	0.20	1.59	0.61	0.27	0.63
0.47	1.31	-0.15	1.33	0.84	1.01	1.16	1.31	0.34	0.44	1.02	0.40
0.11	0.72	1.77	0.58	0.62	1.29	0.15	0.71	0.37	0.66	1.30	0.67

$\boldsymbol{V}^\top$

Easy to compute ratings:

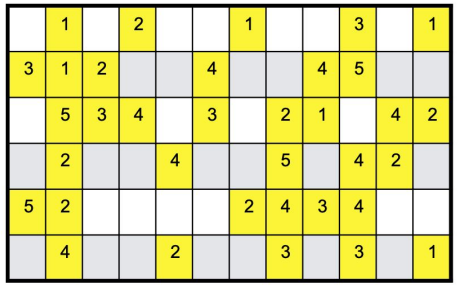
$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^\top =$$

1	1	2	1	1	0	1	1	0	1	0	1
1	2	1	2	0	0	0	2	1	0	2	1
2	1	1	0	0	2	2	1	2	1	1	1
4	2	2	0	0	4	4	2	4	2	2	2
2	4	2	4	0	0	0	4	2	0	4	2
2	2	4	2	2	0	2	2	0	2	0	2

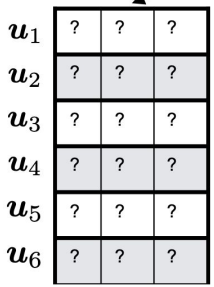
From ratings to factors?

But, our problem is the inverse!

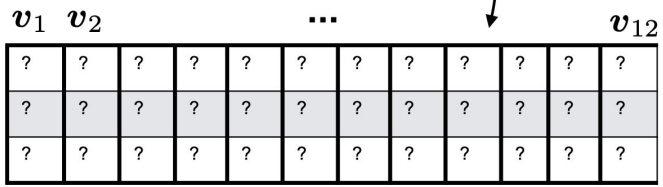
Can we extract factors for users and movies from a partially observed rating matrix?



R



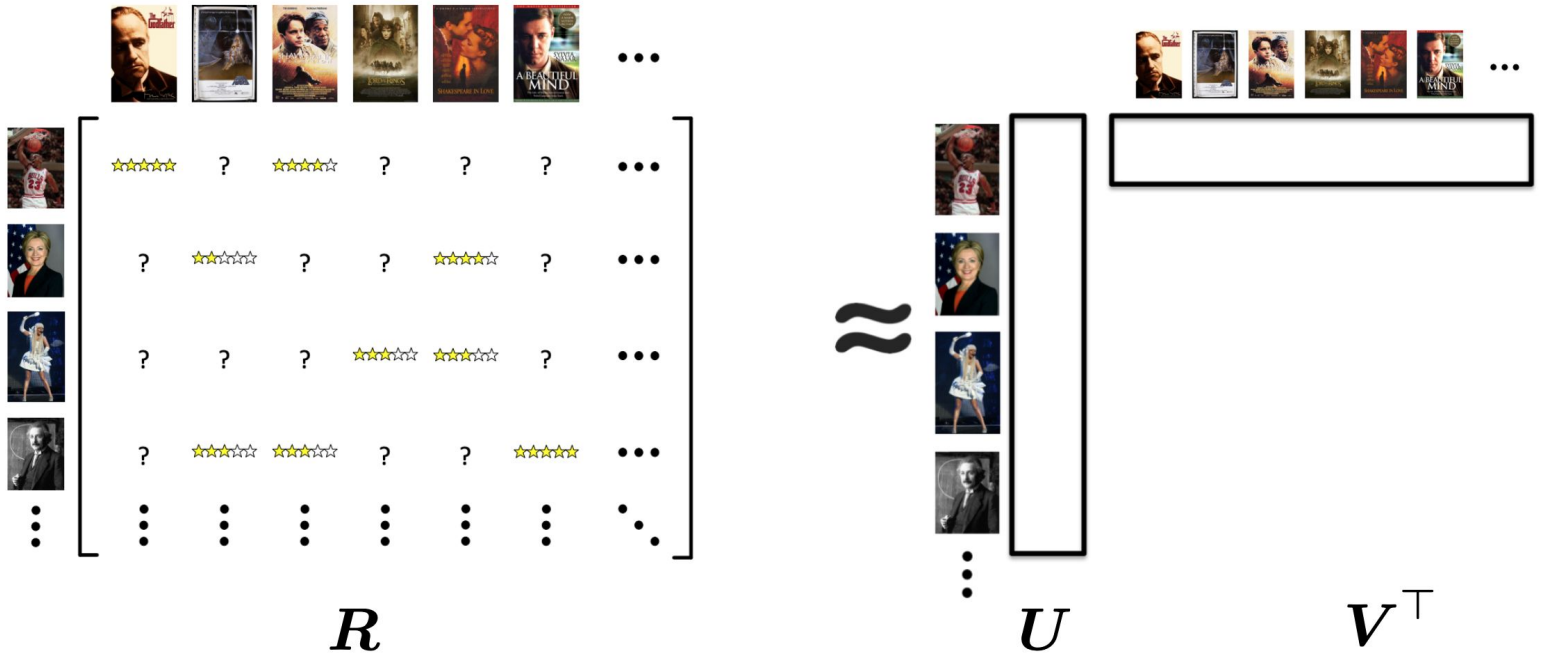
U



$V^\top$

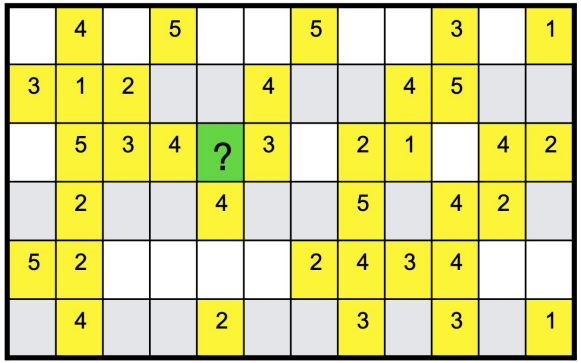
Latent factor models

We need to approximate the partially observed rating matrix with the **product of two low-rank metrics** (factor matrices for users and movies)!



For Netflix, we can write the rating matrix as product of two matrices!

From ratings to factors?



$$\boldsymbol{R}$$

From ratings to factors?

	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4	?	3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

$\boldsymbol{R}$

0.2	-0.4	0.1
0.5	**0.6**	**0.5**
0.5	0.3	-0.2
0.3	2.1	1.1
-2	2.1	-0.7
0.3	0.7	-1

$\boldsymbol{U}$

-0.9	2.4	1.4	0.3	-0.4	0.8	-0.5	**2**	0.5	0.3	-0.2	1.1
1.3	-0.1	1.2	-0.7	2.9	1.4	-1	**-0.3**	1.4	0.5	0.7	-0.8
0.1	-0.6	0.7	0.8	0.4	-0.3	0.9	**2.4**	1.7	0.6	-0.4	2.1

$\boldsymbol{V}^\top$

From ratings to factors?

	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4	?	3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

R

0.2	-0.4	0.1
0.5	**0.6**	**0.5**
0.5	0.3	-0.2
0.3	2.1	1.1
-2	2.1	-0.7
0.3	0.7	-1

U

-0.9	2.4	1.4	0.3	-0.4	0.8	-0.5	**2**	0.5	0.3	-0.2	1.1
1.3	-0.1	1.2	-0.7	2.9	1.4	-1	**-0.3**	1.4	0.5	0.7	-0.8
0.1	-0.6	0.7	0.8	0.4	-0.3	0.9	**2.4**	1.7	0.6	-0.4	2.1

$V^\top$

	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4	**2.1**	3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

If we can extract factors, then we can use the factors to predict unseen ratings and make recommendations!

Simple latent factor model

Simple latent factor model:

-  Each user i has some hidden feature vector $\boldsymbol{u}_i \in \mathbb{R}^k$
-  Each movie j has some hidden feature vector $\boldsymbol{v}_j \in \mathbb{R}^k$
-  Rating r_{ij} assigned by user i to movie j is, in expectation $r_{ij} \approx \boldsymbol{u}_i^\top \boldsymbol{v}_j$

Simple latent factor model

Simple latent factor model:

Each user i has some hidden feature vector $\boldsymbol{u}_i \in \mathbb{R}^k$

Each movie j has some hidden feature vector $\boldsymbol{v}_j \in \mathbb{R}^k$

Rating r_{ij} assigned by user i to movie j is, in expectation $r_{ij} \approx \boldsymbol{u}_i^\top \boldsymbol{v}_j$

$$\mathbb{E}\left\{ \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix} \right\} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_m \\ | & | & & | \end{bmatrix}$$

Simple latent factor model

Simple latent factor model:

Each user i has some hidden feature vector $\boldsymbol{u}_i \in \mathbb{R}^k$

Each movie j has some hidden feature vector $\boldsymbol{v}_j \in \mathbb{R}^k$

Rating r_{ij} assigned by user i to movie j is, in expectation $r_{ij} \approx \boldsymbol{u}_i^\top \boldsymbol{v}_j$

$$\mathbb{E}\left\{ \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix} \right\} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_m \\ | & | & & | \end{bmatrix}$$

We observe $k \leq \min(n, m)$, which becomes more clear later.

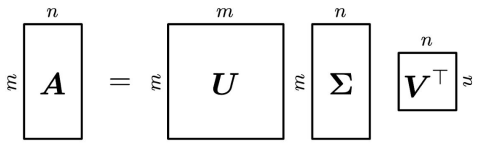
Hypothetical situation 1: Complete ratings

Hypothetical situation: suppose every user rates every movie

$$\mathbb{E}\left\{\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix}\right\} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_m \\ | & | & & | \end{bmatrix}$$

recall: full SVD

Theorem 4.22 (SVD Theorem). *Let $\boldsymbol{A}^{m\times n}$ be a rectangular matrix of rank $r \in [0, \min(m,n)]$. The SVD of $\boldsymbol{A}$ is a decomposition of the form*



(4.64)

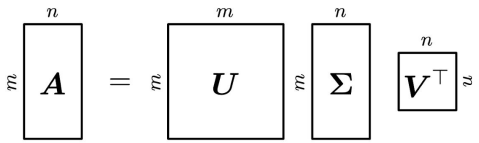
with an orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ with column vectors $\boldsymbol{u}_i$, $i = 1, \ldots, m$, and an orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ with column vectors $\boldsymbol{v}_j$, $j = 1, \ldots, n$. Moreover, $\boldsymbol{\Sigma}$ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geqslant 0$ and $\Sigma_{ij} = 0,\ i \neq j$.

The diagonal entries σ_i, $i = 1, \ldots, r$, of $\boldsymbol{\Sigma}$ are called the *singular values*, $\boldsymbol{u}_i$ are called the *left-singular vectors*, and $\boldsymbol{v}_j$ are called the *right-singular vectors*. By convention, the singular values are ordered, i.e., $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_r \geqslant 0$.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & \ddots & 0 & 0 & & 0 \\ 0 & 0 & \sigma_m & 0 & \ldots & 0 \end{bmatrix}$$

a bit more on SVD: reduced SVD

Theorem 4.22 (SVD Theorem). *Let $\boldsymbol{A}^{m\times n}$ be a rectangular matrix of rank $r \in [0, \min(m,n)]$. The SVD of $\boldsymbol{A}$ is a decomposition of the form*



(4.64)

with an orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ with column vectors $\boldsymbol{u}_i$, $i = 1, \ldots, m$, and an orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ with column vectors $\boldsymbol{v}_j$, $j = 1, \ldots, n$. Moreover, $\boldsymbol{\Sigma}$ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geqslant 0$ and $\Sigma_{ij} = 0,\ i \neq j$.

The diagonal entries σ_i, $i = 1, \ldots, r$, of $\boldsymbol{\Sigma}$ are called the *singular values*, $\boldsymbol{u}_i$ are called the *left-singular vectors*, and $\boldsymbol{v}_j$ are called the *right-singular vectors*. By convention, the singular values are ordered, i.e., $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_r \geqslant 0$.

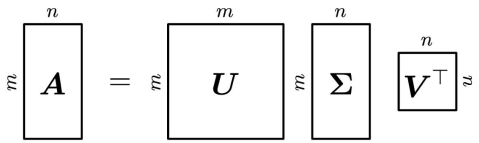
for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $m \geqslant n$,

$$\underset{m\times n}{\boldsymbol{A}} = \underset{m\times n}{\boldsymbol{U}}\ \underset{n\times n}{\boldsymbol{\Sigma}}\ \underset{n\times n}{\boldsymbol{V}^\top}.$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ & & \\ & & \end{bmatrix}$$

reduced SVD

Theorem 4.22 (SVD Theorem). *Let $\boldsymbol{A}^{m\times n}$ be a rectangular matrix of rank $r \in [0, \min(m,n)]$. The SVD of $\boldsymbol{A}$ is a decomposition of the form*

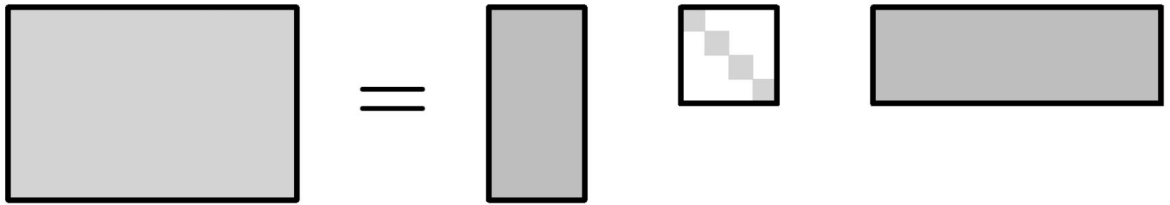


(4.64)

$$\text{for } \boldsymbol{A} \in \mathbb{R}^{m\times n} \text{ and } m \geqslant n,$$

$$\underset{m\times n}{\boldsymbol{A}} = \underset{m\times n}{\boldsymbol{U}}\ \underset{n\times n}{\boldsymbol{\Sigma}}\ \underset{n\times n}{\boldsymbol{V}^\top}.$$

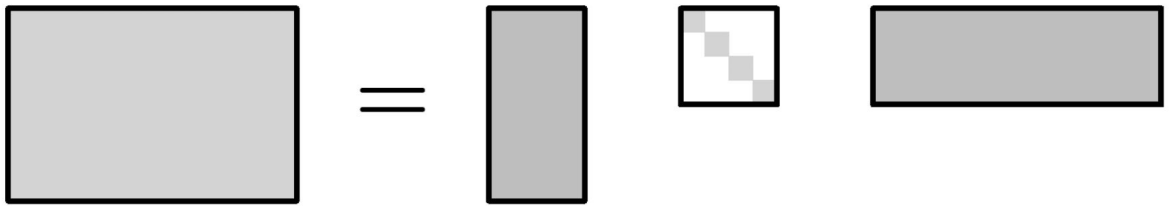
$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ & & \\ & & \end{bmatrix}$$



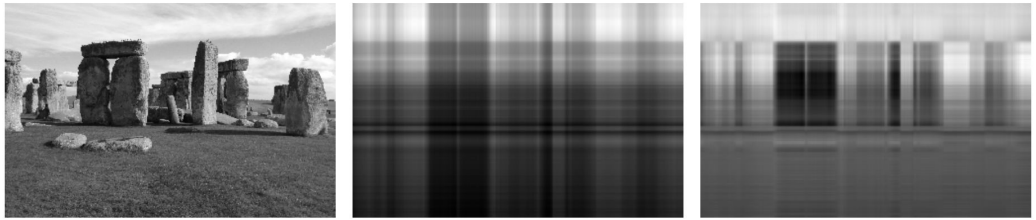
SVD with rank-r matrix

Theorem. The **rank** of A equals the **number of non-zero singular values**, which is the same as the number of non-zero diagonal elements Σ

It is possible to define the SVD of a rank-r matrix $\boldsymbol{A}$ so that $\boldsymbol{U}$ is an $m \times r$ matrix, $\boldsymbol{\Sigma}$ a diagonal matrix $r \times r$, and $\boldsymbol{V}$ an $r \times n$ matrix. This construction is very similar to our definition, and ensures that the diagonal matrix $\boldsymbol{\Sigma}$ has only nonzero entries along the diagonal. The



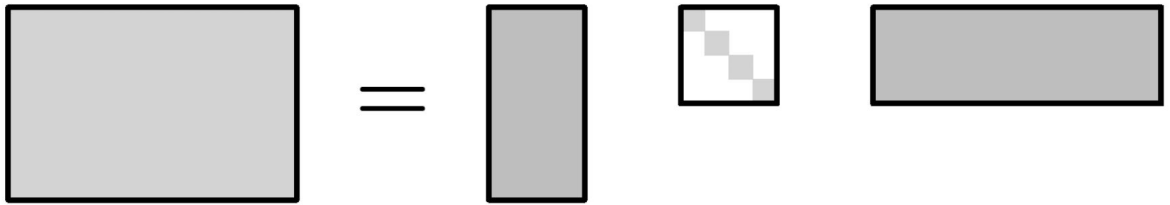
SVD for low-rank matrix approximation



(a) Original image $\boldsymbol{A}$. (b) Rank-1 approximation $\widehat{\boldsymbol{A}}(1)$. (c) Rank-2 approximation $\widehat{\boldsymbol{A}}(2)$.

(d) Rank-3 approximation $\widehat{\boldsymbol{A}}(3)$. (e) Rank-4 approximation $\widehat{\boldsymbol{A}}(4)$. (f) Rank-5 approximation $\widehat{\boldsymbol{A}}(5)$.

Figure 4.12 Image reconstruction with the SVD. (a) Original image. (b)–(f) Image reconstruction using the low-rank approximation of the SVD, where the rank-k approximation is given by $\widehat{\boldsymbol{A}}(k) = \sum_{i=1}^{k} \sigma_i \boldsymbol{A}_i$.

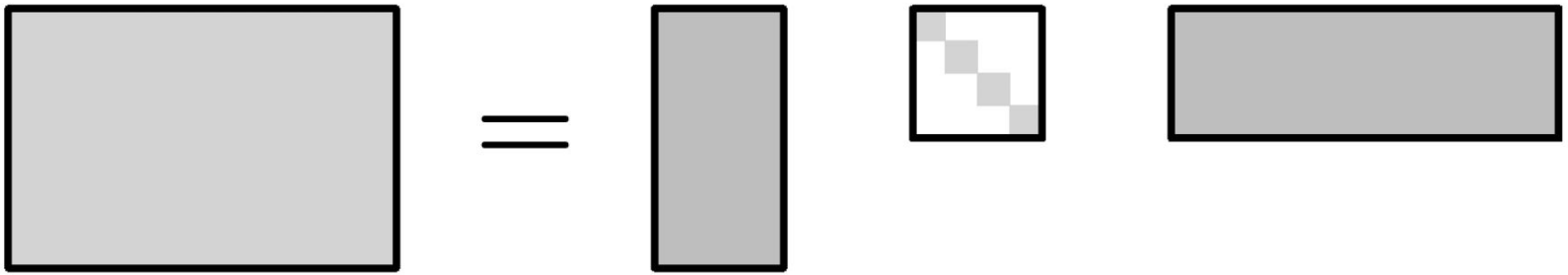


Hypothetical situation 1: Complete ratings

Hypothetical situation: suppose every user rates every movie

$$\mathbb{E}\left\{\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix}\right\} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_m \\ | & | & & | \end{bmatrix}$$

Can use SVD in this case:



Hypothetical situation 2: Known user features

$$\mathbb{E}\left\{\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1m} \\ r_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ r_{n1} & r_{n2} & r_{n3} & \dots & r_{nm} \end{bmatrix}\right\} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_m \\ | & | & & | \end{bmatrix}$$

Hypothetical situation: suppose the users' features were *not* hidden.

Could try to *estimate* each movie vector $\boldsymbol{v}_j$ using linear regression:

$$\begin{bmatrix} r_{1,j} \\ r_{2,j} \\ \vdots \\ r_{n,j} \end{bmatrix} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{v}_j \\ | \end{bmatrix} + \begin{bmatrix} \varepsilon_{1,j} \\ \varepsilon_{2,j} \\ \vdots \\ \varepsilon_{n,j} \end{bmatrix}$$

The catch: we don't observe all of the entries of rating matrix.

Missing Data of Ratings $r_{i,j}$

$$\begin{bmatrix} r_{1,j} \\ r_{2,j} \\ \vdots \\ r_{n,j} \end{bmatrix} = \begin{bmatrix} — & \boldsymbol{u}_1^\top & — \\ — & \boldsymbol{u}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{u}_n^\top & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{v}_j \\ | \end{bmatrix} + \begin{bmatrix} \varepsilon_{1,j} \\ \varepsilon_{2,j} \\ \vdots \\ \varepsilon_{n,j} \end{bmatrix}$$

We don't observe all of the entries of rating matrix.

So, for each movie j, only use $\boldsymbol{u}_i$ for which we have observed rating $r_{i,j}$

Real situation: Don't know either user or movie features

Let Ω be the entries for which the rating $r_{i,j}$ is observed

Regard these $r_{i,j}$ as training data

Training objective: find $\boldsymbol{U} = [\boldsymbol{u}_1 \mid \boldsymbol{u}_1 \mid \ \ldots \ \boldsymbol{u}_n] \in \mathbb{R}^{n \times k}$

$$\boldsymbol{V} = [\boldsymbol{v}_1 \mid \boldsymbol{v}_2 \mid \ \ldots \ \mid \boldsymbol{v}_m] \in \mathbb{R}^{m \times k}$$

Minimizing error on training data

Let Ω be the entries for which the rating $r_{i,j}$ is observed

Regard these $r_{i,j}$ as training data

Training objective: find $\boldsymbol{U} = [\boldsymbol{u}_1 \mid \boldsymbol{u}_1 \mid \ \ldots \ \boldsymbol{u}_n] \in \mathbb{R}^{n \times k}$

$$\boldsymbol{V} = [\boldsymbol{v}_1 \mid \boldsymbol{v}_2 \mid \ \ldots \ \mid \boldsymbol{v}_m] \in \mathbb{R}^{m \times k}$$

with squared error as small as possible:

$$f(\boldsymbol{U}, \boldsymbol{V}) = \sum_{(i,j) \in \Omega} \left(r_{i,j} - \boldsymbol{u}_i^\top \boldsymbol{v}_j \right)^2$$

Often called (low-rank) matrix completion, because this is equivalent to

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times m}} \quad \sum_{(i,j) \in \Omega} \left(r_{i,j} - X_{i,j} \right)^2$$

$$\text{such that} \quad \operatorname{rank}(\boldsymbol{X}) \leq k$$

Minimizing error on training data

$$f(\boldsymbol{U}, \boldsymbol{V}) = \sum_{(i,j)\in\Omega} \left(r_{i,j} - \boldsymbol{u}_i^\top \boldsymbol{v}_j\right)^2$$

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} \sum_{(i,j)\in\Omega} (r_{i,j} - X_{i,j})^2 \quad \text{such that} \quad \operatorname{rank}(\boldsymbol{X}) \leq k$$

Unfortunately, objective is not convex due to rank constraint.

Nevertheless, we can derive an alternating minimization algorithm to iteratively update $\boldsymbol{U}$ and $\boldsymbol{V}$

Alternating minimization

Initialize

$$\boldsymbol{u}_i \in \mathbb{R}^k \text{ for each } i = 1, 2, \ldots, n \text{ and } \boldsymbol{v}_j \in \mathbb{R}^k \text{ for each } j = 1, 2, \ldots, m$$

Repeat until convergence

Alternating minimization

Initialize

$$\boldsymbol{u}_i \in \mathbb{R}^k \text{ for each } i = 1, 2, \ldots, n \text{ and } \boldsymbol{v}_j \in \mathbb{R}^k \text{ for each } j = 1, 2, \ldots, m$$

Repeat until convergence

For each user $i = 1, 2, \ldots, n$

$$\widehat{\boldsymbol{u}_i} = \arg\min_{\boldsymbol{u}_i \in \mathbb{R}^k} f(\boldsymbol{U}, \boldsymbol{V})$$

For each movie $j = 1, 2, \ldots, m$

$$\widehat{\boldsymbol{v}_i} = \arg\min_{\boldsymbol{v}_i \in \mathbb{R}^k} f(\boldsymbol{U}, \boldsymbol{V})$$

Alternating minimization

Initialize

$$\boldsymbol{u}_i \in \mathbb{R}^k \text{ for each } i = 1, 2, \ldots, n \text{ and } \boldsymbol{v}_j \in \mathbb{R}^k \text{ for each } j = 1, 2, \ldots, m$$

Repeat until convergence

For each user $i = 1, 2, \ldots, n$

$$\widehat{\boldsymbol{u}_i} = \left(\sum_{j \in [m]:(i,j) \in \Omega} \boldsymbol{v}_j \boldsymbol{v}_j^\top \right)^{-1} \sum_{j \in [m]:(i,j) \in \Omega} r_{i,j} \boldsymbol{v}_j$$

For each movie $j = 1, 2, \ldots, m$

$$\widehat{\boldsymbol{v}_j} = \left(\sum_{i \in [n]:(i,j) \in \Omega} \boldsymbol{u}_i \boldsymbol{u}_i^\top \right)^{-1} \sum_{i \in [n]:(i,j) \in \Omega} r_{i,j} \boldsymbol{u}_i$$

You will derive this solution in HW

Adding regularization

Regularized training objective:

$$f(\boldsymbol{U}, \boldsymbol{V}) = \sum_{(i,j)\in\Omega} \left(r_{i,j} - \boldsymbol{u}_i^\top \boldsymbol{v}_j\right)^2 + \lambda \left(\sum_{i=1}^{n} \|\boldsymbol{u}_i\|_2^2 + \sum_{j=1}^{m} \|\boldsymbol{v}_j\|_2^2 \right)$$

$$\widehat{\boldsymbol{u}_i} = \left(\sum_{j\in[m]:(i,j)\in\Omega} \boldsymbol{v}_j \boldsymbol{v}_j^\top + \lambda \boldsymbol{I} \right)^{-1} \sum_{j\in[m]:(i,j)\in\Omega} r_{i,j} \boldsymbol{v}_j$$

$$\widehat{\boldsymbol{v}_j} = \left(\sum_{i\in[n]:(i,j)\in\Omega} \boldsymbol{u}_i \boldsymbol{u}_i^\top + \lambda \boldsymbol{I} \right)^{-1} \sum_{i\in[n]:(i,j)\in\Omega} r_{i,j} \boldsymbol{u}_i$$

Incorporating bias

We can also incorporating biases in users and movie ratings:

b_i

c_j

$$f(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{c}, \mu) = \sum_{(i,j)\in\Omega} \left(r_{i,j} - \boldsymbol{u}_i^\top \boldsymbol{v}_j - b_i - c_j - \mu\right)^2 + \lambda \left(\sum_{i=1}^{n} \|\boldsymbol{u}_i\|_2^2 + \sum_{j=1}^{m} \|\boldsymbol{v}_j\|_2^2 \right)$$

Prediction

After learning the latent factors for all users and movies, we can make new predictions by:

$$\widehat{r}_{i,j} = \widehat{\boldsymbol{u}_i}^{\top} \widehat{\boldsymbol{v}_j}$$

Unknown Rating

Initialization

Initialize

$$\boldsymbol{u}_i \in \mathbb{R}^k \text{ for each } i = 1, 2, \ldots, n \text{ and } \boldsymbol{v}_j \in \mathbb{R}^k \text{ for each } j = 1, 2, \ldots, m$$

Initialization

Initialize

$$\boldsymbol{u}_i \in \mathbb{R}^k \text{ for each } i = 1, 2, \ldots, n \text{ and } \boldsymbol{v}_j \in \mathbb{R}^k \text{ for each } j = 1, 2, \ldots, m$$

Initialization: can't initialize with all $\boldsymbol{u}_i = \boldsymbol{v}_j = \mathbf{0}$. (Try random.)

How to pick k or λ?

Why? Think about this and you will answer in HW.

Answer is obvious by now, but there are some subtle issues.

- If $\lambda = 0$, then for updates to work, every user must have seen at least k movies, and every movie must have been rated by at least k users.
- If $\lambda > 0$, then technically updates work for any k.
- Perhaps should have separate λ for each user/movie, and perhaps it should change as the $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ change!

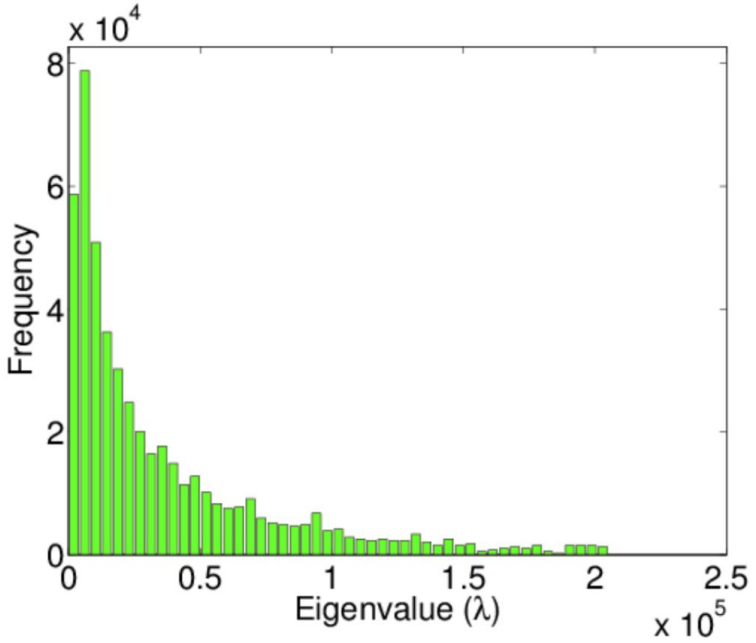
Matrix completion

Given an partially observed matrix, can you fill in the blanks?
In general, the answer is NO!

$$\begin{bmatrix} \checkmark & ? & ? & ? & \checkmark & ? \\ ? & ? & \checkmark & \checkmark & ? & ? \\ \checkmark & ? & ? & \checkmark & ? & ? \\ ? & ? & \checkmark & ? & ? & \checkmark \\ \checkmark & ? & ? & ? & ? & ? \\ ? & \checkmark & ? & ? & \checkmark & ? \\ ? & ? & \checkmark & \checkmark & ? & ? \end{bmatrix}$$

- Underdetermined system (more unknowns than revealed entries)
- Seems hopeless

Matrix completion for Netflix



For Netflix, we can because the rating matrix is low-rank!

Data in reality

images

videos

U.S. COMMERCE'S ORTNER SAYS YEN UNDERVALUED

Commerce Dept. undersecretary of economic a@airs Robert Ortner said that he believed the dollar at current levels was fairly priced against most European currencies.

In a wide ranging address sponsored by the Export-Import Bank, Ortner, the bank's senior economist also said he believed that the yen was undervalued and could go up by 10 or 15 pct.

"I do not regard the dollar as undervalued at this point against the yen," he said.

On the other hand, Ortner said that he thought that "the yen is still a little bit undervalued," and "could go up another 10 or 15 pct."

In addition, Ortner, who said he was speaking personally, said he thought that the dollar against most European currencies was "fairly priced."

Ortner said his analysis of the various exchange rate values was based on such economic particulars as wage rate di@erentiations.

Ortner said there had been little impact on U.S. trade de~cit by the decline of the dollar because at the time of the Plaza Accord, the dollar was extremely overvalued and that the ~rst 15 pct decline had little impact.

He said there were indications now that the trade de~cit was beginning to level o@.

Turning to Brazil and Mexico, Ortner made it clear that it would be almost impossible for those countries to earn enough foreign exchange to pay the service on their debts. He said the best way to deal with this was to use the policies outlined in Treasury Secretary James Baker's debt initiative.

text

web data

Huge data sizes
but often
low-dimensional structure