10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Recommender Systems
+
Matrix Factorization

Matt Gormley & Geoff Gordon
Lecture 23
Nov. 17, 2025

Reminders

- **Homework 8: Deep RL**
  - Out: Sun, Nov. 16
  - Due: Mon, Nov. 24 at 11:59pm

Learning Paradigms

Paradigm	**Data**
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \ldots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$
$\hookrightarrow$ Clustering	predict $\{z^{(i)}\}_{i=1}^N$ where $z^{(i)} \in \{1, \ldots, K\}$
$\hookrightarrow$ Dimensionality Reduction	convert each $\mathbf{x}^{(i)} \in \mathbb{R}^M$ to $\mathbf{u}^{(i)} \in \mathbb{R}^K$ with $K << M$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots\}$

ML Big Picture

Learning Paradigms:

What data is available and when? What form of prediction?

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

Problem Formulation:

What is the structure of our output prediction?

boolean	Binary Classification
categorical	Multiclass Classification
ordinal	Ordinal Classification
real	Regression
ordering	Ranking
multiple discrete	Structured Prediction
multiple continuous	(e.g. dynamical systems)
both discrete & cont.	(e.g. mixed graphical models)

Application Areas

Key challenges?

NLP, Speech, Computer Vision, Robotics, Medicine, Search

Theoretical Foundations:

What principles guide learning?

- ❑ probabilistic
- ❑ information theoretic
- ❑ evolutionary search
- ❑ ML as optimization

Facets of Building ML Systems:

How to build systems that are robust, efficient, adaptive, effective?

1. *Data prep*
2. *Model selection*
3. *Training (optimization / search)*
4. *Hyperparameter tuning on validation data*
5. *(Blind) Assessment on test data*

Big Ideas in ML:

Which are the ideas driving development of the field?

- *inductive bias*
- *generalization / overfitting*
- *bias-variance decomposition*
- *generative vs. discriminative*
- *deep nets, graphical models*
- *PAC learning*
- *distant rewards*

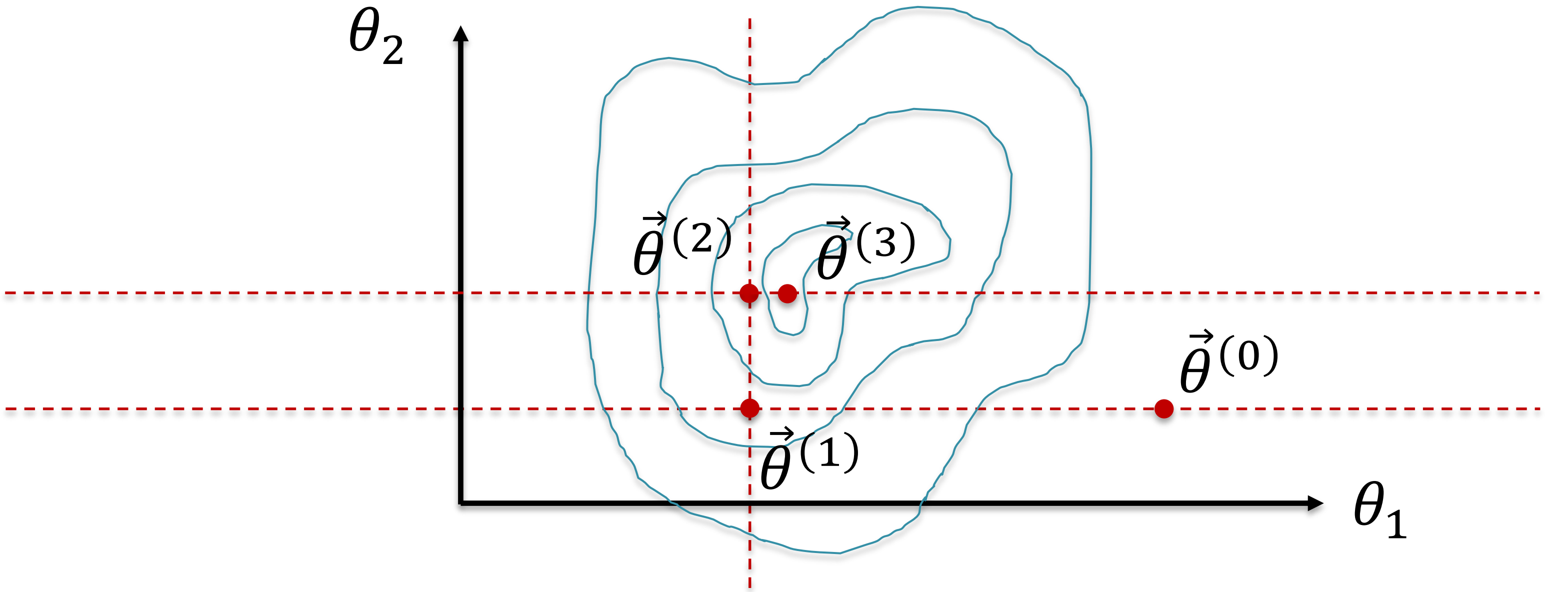
OPTIMIZATION BACKGROUND

Coordinate Descent

- Goal: minimize some objective

$$\vec{\theta}^* = \underset{\vec{\theta}}{\operatorname{argmin}}\, J(\vec{\theta})$$

- Idea: iteratively pick one variable and minimize the objective w.r.t. just that one variable, *keeping all the others fixed.*



Block Coordinate Descent

- Goal: minimize some objective (with 2 blocks)

$$\vec{\alpha}^*, \vec{\beta}^* = \underset{\vec{\alpha}, \vec{\beta}}{\operatorname{argmin}} J(\vec{\alpha}, \vec{\beta})$$

- Idea: iteratively pick one *block* of variables ($\vec{\alpha}$ or $\vec{\beta}$) and minimize the objective w.r.t. that block, keeping the other(s) fixed.

while not converged:

$$\vec{\alpha} = \underset{\vec{\alpha}}{\operatorname{argmin}} J(\vec{\alpha}, \vec{\beta})$$

$$\vec{\beta} = \underset{\vec{\beta}}{\operatorname{argmin}} J(\vec{\alpha}, \vec{\beta})$$

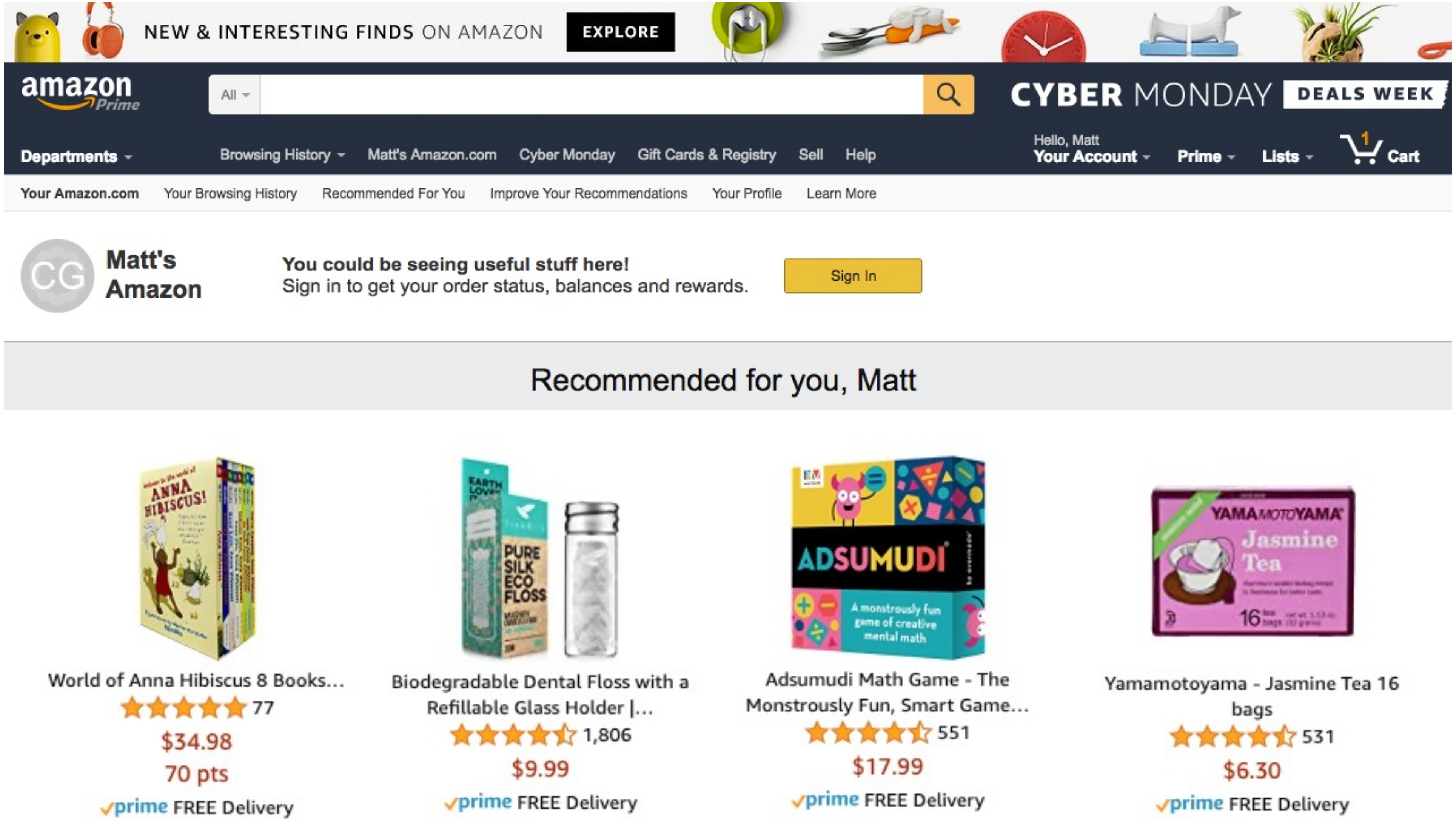
RECOMMENDER SYSTEMS

Recommender Systems

A Common Challenge:

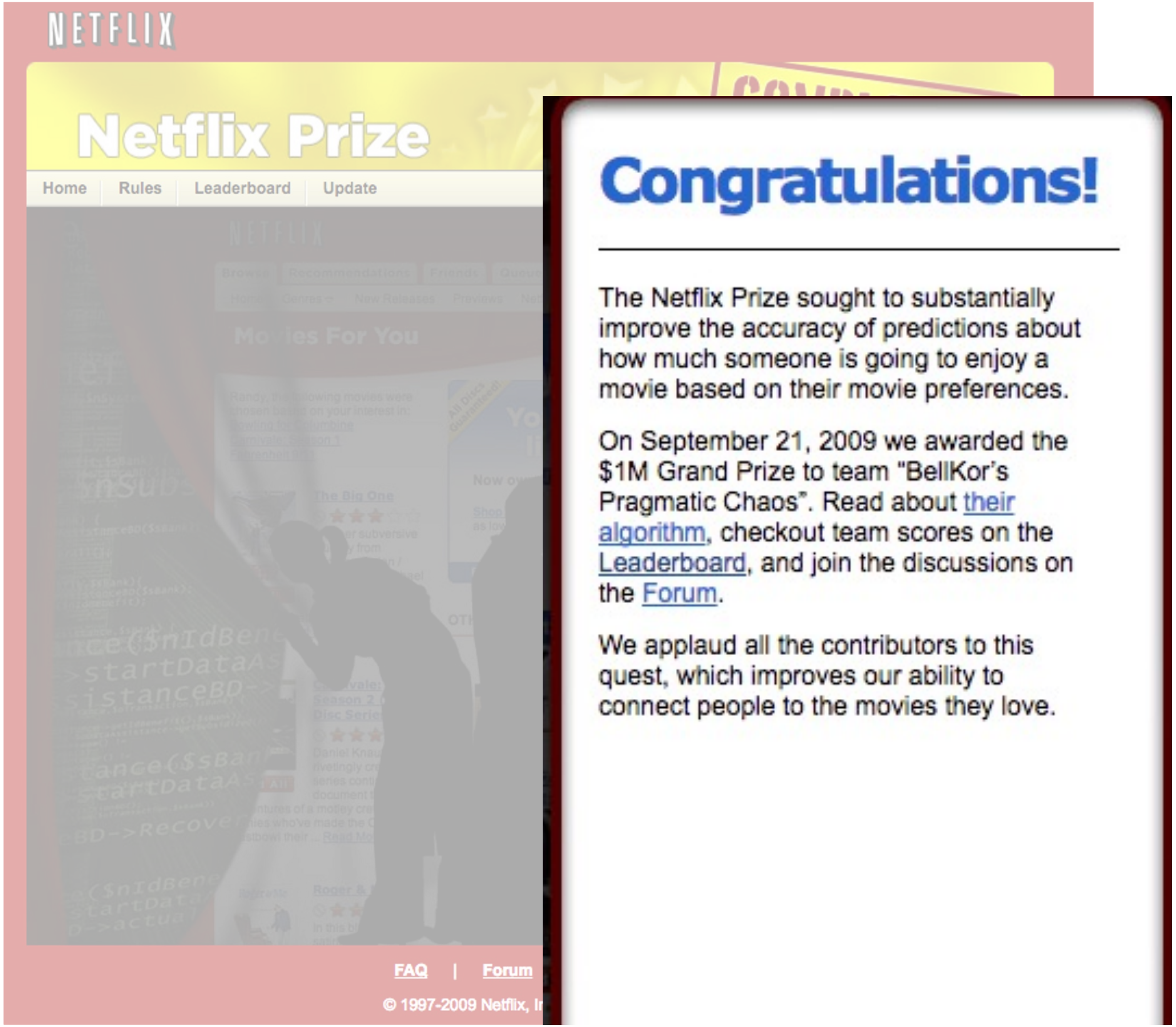
- Assume you're a company selling **items** of some sort: movies, songs, products, etc.
- Company collects millions of **ratings** from **users** of their **items**
- To maximize profit / user happiness, you want to **recommend** items that users are likely to want

Recommender Systems

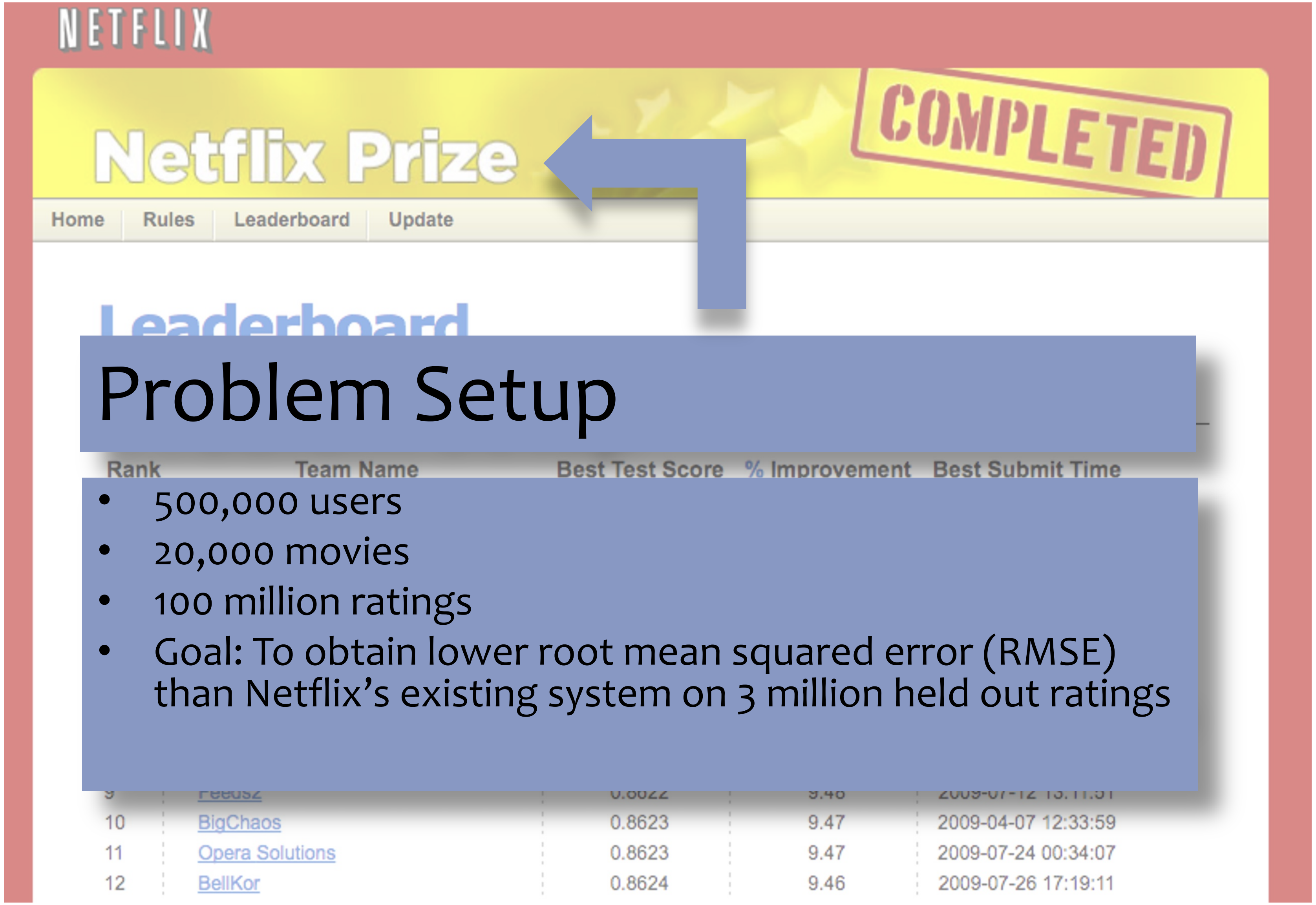


Recommender Systems

Recommender Systems



Recommender Systems



Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix's existing system on 3 million held out ratings

Recommender Systems

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

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

Recommender Systems

- **Setup:**
  - Items:
    movies, songs, products, etc.
    (often many thousands)
  - Users:
    watchers, listeners, purchasers, etc.
    (often many millions)
  - Feedback:
    5-star ratings, not-clicking 'next',
    purchases, etc.
- **Key Assumptions:**
  - Can represent ratings numerically as a user/item matrix
  - Users only rate a small number of items (the matrix is sparse)

	Doctor Strange	Star Trek: Beyond	Zootopia
Alice	1		5
Bob	3	4	
Charlie	3	5	2

Two Types of Recommender Systems

Content Filtering

- **Definition:** Recommendation approach that analyzes **item attributes** (features, metadata, keywords, categories) and matches them to a **user's known preferences**.
- **Example:** Pandora.com music recommendations (Music Genome Project)
- **Con:** Assumes access to side information about items (e.g. properties of a song)
- **Pro:** Got a new item to add? No problem, just be sure to include the side information

Collaborative Filtering

- **Definition:** A recommendation approach that analyzes **user–item interaction patterns** (ratings, clicks, purchases) to find similar users or items based on behavior.
- **Example:** Netflix movie recommendations
- **Pro:** Does not assume access to side information about items (e.g. does not need to know about movie genres)
- **Con:** Does not work on new items that have no ratings

COLLABORATIVE FILTERING

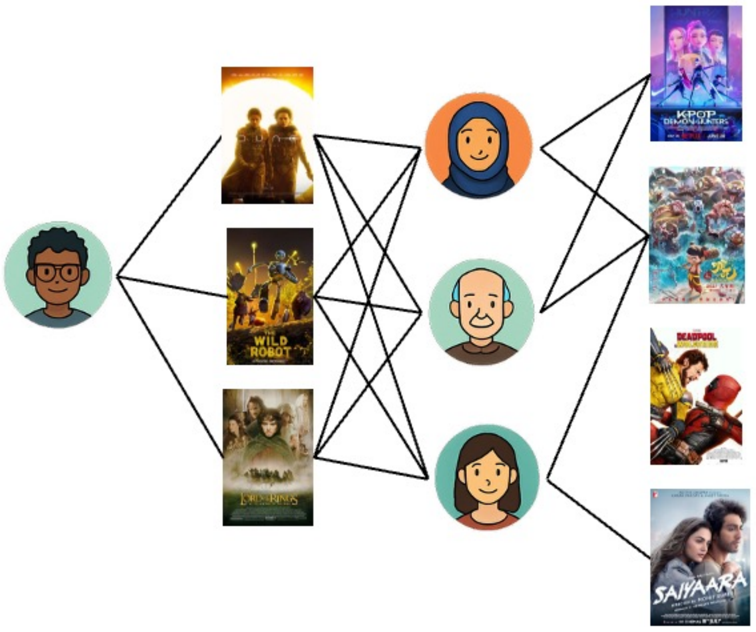
Collaborative Filtering

- **Everyday Examples of Collaborative Filtering...**
  - Bestseller lists
  - Top 40 music lists
  - The "recent returns" shelf at the library
  - Unmarked but well-used paths thru the woods
  - The printer room at work
  - "Read any good books lately?"
  - …
- **Common insight:** personal tastes are correlated
  - If Alice and Bob both like X and Alice likes Y then Bob is more likely to like Y
  - especially (perhaps) if Bob knows Alice

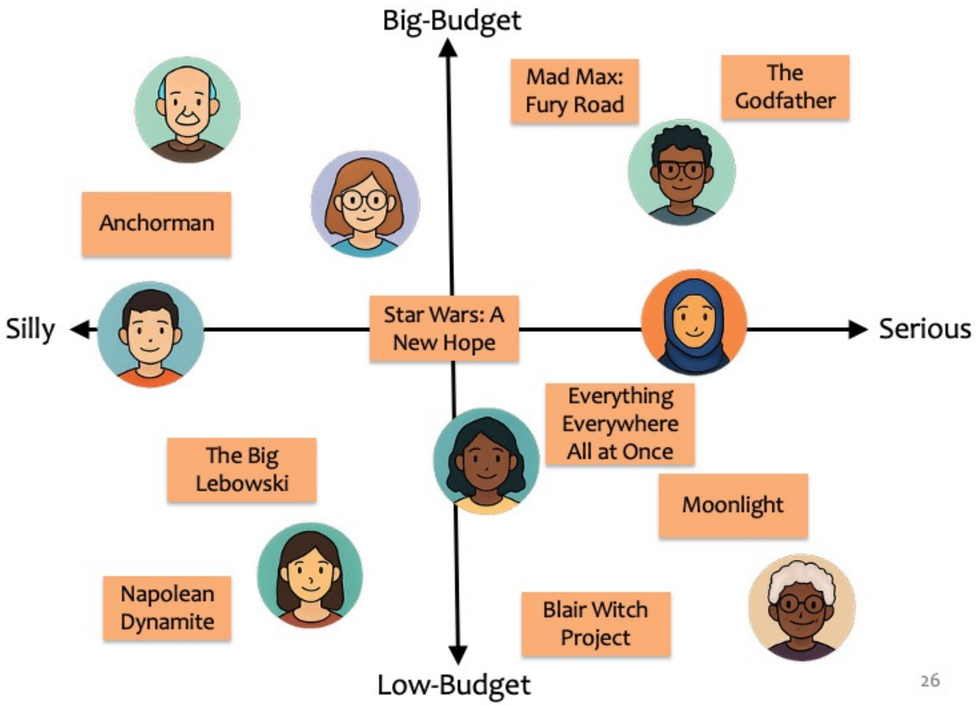
Slide from William Cohen

Two Types of Collaborative Filtering

1. Neighborhood Methods



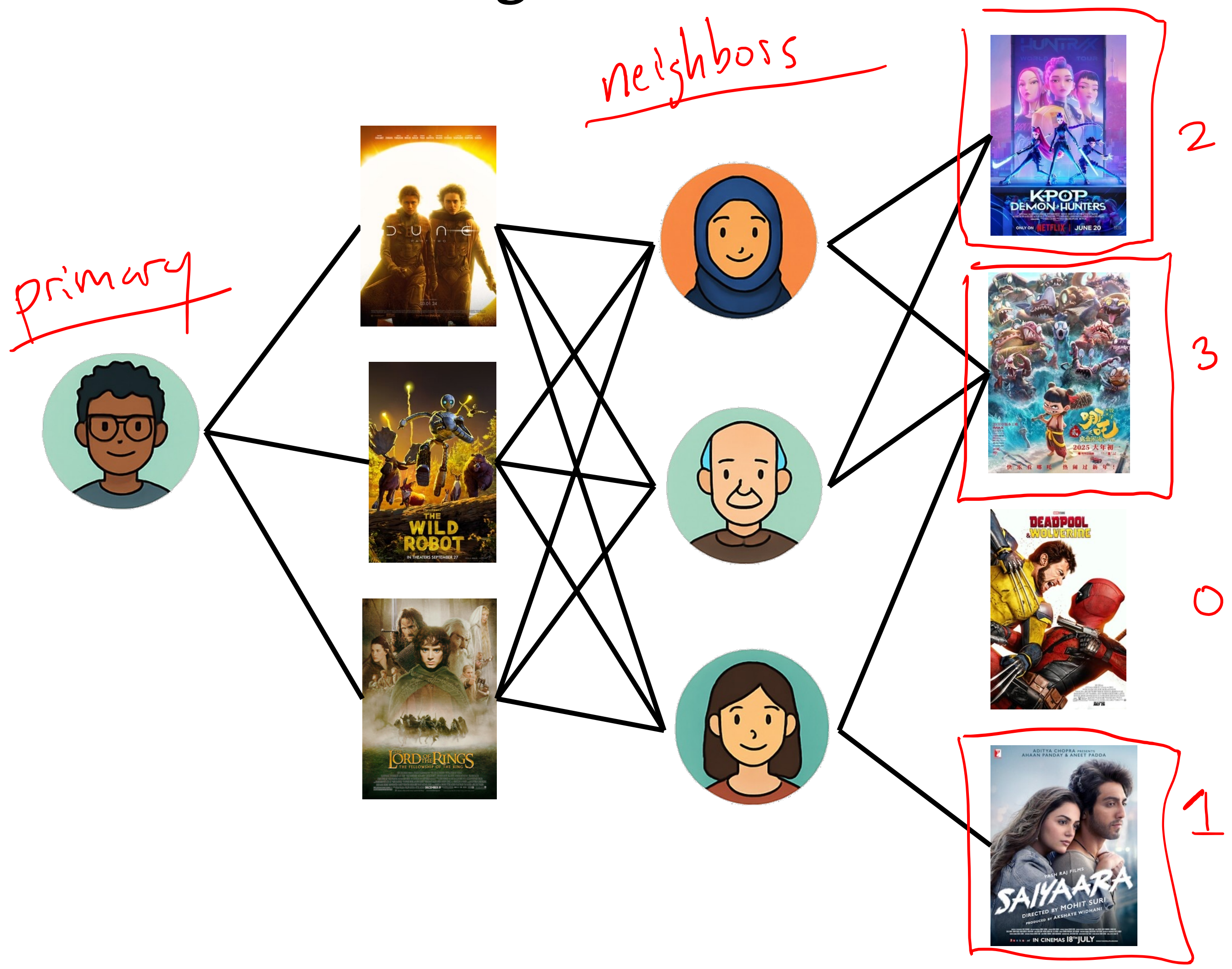
2. Latent Factor Methods



Figures from Koren et al. (2009)

Two Types of Collaborative Filtering

1. Neighborhood Methods



In the figure, assume that a ~~green~~ line indicates the movie was **watched**

Algorithm:

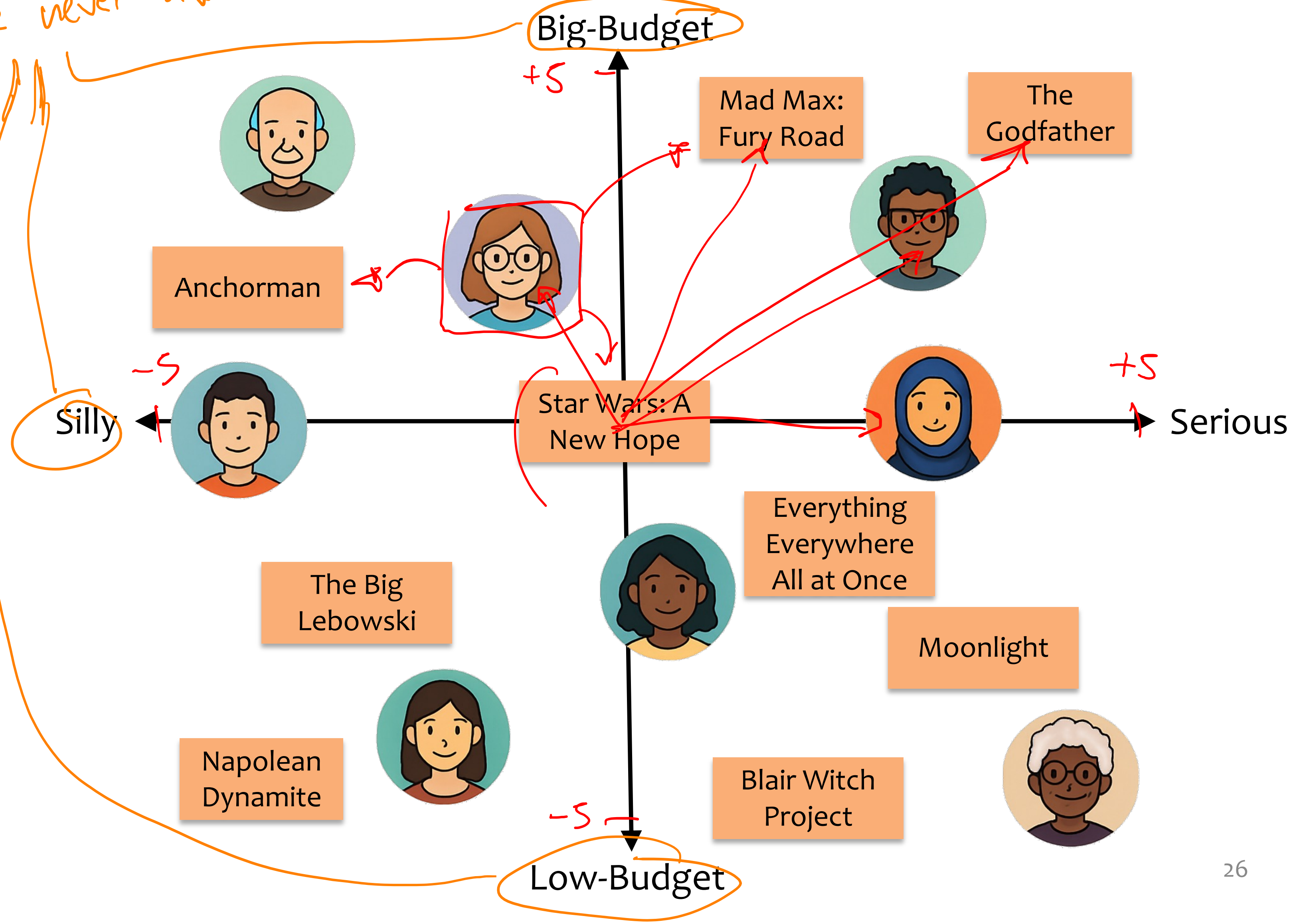
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Two Types of Collaborative Filtering

these labels are never available

2. Latent Factor Methods

- Assume that both movies and users live in some **low-dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space
- **Example Algorithm:** Matrix Factorization



Recommending Movies

Poll

Question: 1

Applied to the Netflix Prize problem, which of the following methods *always* requires side information about the users and movies?

Select all that apply

toxic → A. ~~principal component analysis~~

26% B. collaborative filtering

25% C. latent factor methods

15% D. ensemble methods

73% E. content filtering

26% F. neighborhood methods

20% G. recommender systems

Answer:

MATRIX FACTORIZATION

Matrix Factorization

- Many different ways of factorizing a matrix
- We'll consider three:
    1. Unconstrained Matrix Factorization
    2. Singular Value Decomposition
    3. Non-negative Matrix Factorization
- MF is just another example of a **common recipe**:
    1. define a model
    2. define an objective function
    3. optimize with SGD

Linear Algebra Background

Rank of a Matrix: For an $m \times n$ matrix $\mathbf{R}$, the *rank* is the number of linearly independent columns (equivalently, rows).

$\mathrm{rank}(\mathbf{R}) = k \iff$ $\mathbf{R}$ has k linearly independent columns.

Example.

$$\mathbf{R} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

The first two columns are independent; the third is their sum. Thus $\mathrm{rank}(\mathbf{R}) = 2$.

Basis Vectors: A *basis* is a smallest set of independent vectors that can generate all vectors in a space. If $\mathbf{R}$ has rank k, then there exist k basis vectors that span all its columns.

Example. A basis for the $\mathbf{R}$ above is given by its first two columns:

$$\mathbf{u}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{u}_2 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}.$$

The third column satisfies

$$\mathbf{r}_3 = 1 \cdot \mathbf{u}_1 + 1 \cdot \mathbf{u}_2.$$

Low-rank Matrix Factorization

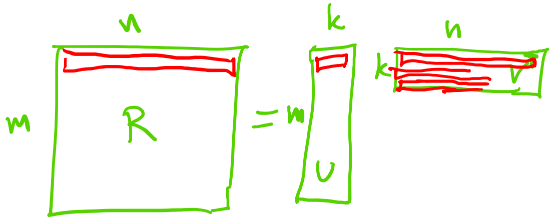
Case 1: Exact Factorization

Given: $m \times n$ matrix R of rank $k \ll \min(m,n)$

Claim: $\exists$ $m \times k$ matrix U

$n \times k$ matrix V

s.t. $R = UV^T$



Note: ① columns of U are the k basis vectors of the cols. of R

② rows of V^T are " " " " " " rows. of R

each row of R is a linear combination of the rows of V^T

Case 2: Approximate Factorization

Given: $m \times n$ matrix R of rank $\ell > k$

where k = # of cols of U and V

Claim: Can approximate R by some U and V as

$R \approx UV^T$

Approximation Error:

Def: residual matrix $E = R - UV^T$

MSE: $(\|E\|_2)^2 = (\|R - UV^T\|_2)^2$

where $\|E\|_2 = \sqrt{\sum_i \sum_j (E_{ij})^2}$

is the Frobenius Norm

Low-rank Matrix Factorization

Low-Rank Factorization: We seek matrices $\mathbf{U}$ and $\mathbf{V}$ with k columns such that

$$\mathbf{R} = \mathbf{U}\mathbf{V}^\top, \qquad \operatorname{rank}(\mathbf{U}\mathbf{V}^\top) \le k.$$

Example (Exact Rank–2 Factorization). Let

$$\mathbf{U} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{V}^\top = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Then

$$\mathbf{U}\mathbf{V}^\top = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{R}.$$

Approximation Viewpoint: If $\mathbf{R}$ has rank $\ell > k$, then it cannot be represented exactly with rank k, but we can choose $\mathbf{U}, \mathbf{V}$ to make the residual $\mathbf{E} = \mathbf{R} - \mathbf{U}\mathbf{V}^\top$ as small as possible.

Example (Forced Rank–1 Approximation). Let

$$\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{V}^\top = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}.$$

Then

$$\mathbf{U}\mathbf{V}^\top = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The residual is

$$\mathbf{E} = \mathbf{R} - \mathbf{U}\mathbf{V}^\top = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \|\mathbf{E}\|_F = \sqrt{2}.$$

Example: MF for Netflix Problem

not availble

history

romance

		NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
HISTORY	1	1	1	1	0	0	0
HISTORY	2	1	1	1	0	0	0
HISTORY	3	1	1	1	0	0	0
BOTH	4	1	1	1	1	1	1
ROMANCE	5	-1	-1	-1	1	1	1
ROMANCE	6	-1	-1	1	1	1	1
ROMANCE	7	-1	-1	-1	1	1	1

R

$\approx$

1	1	0
2	1	0
3	1	0
4	1	1
5	-1	1
6	-1	1
7	-1	1

U

X

NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
1	1	1	0	0	0
0	0	1	1	1	1

V^T

(a) Example of rank-2 matrix factorization

		NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
HISTORY	1	0	0	0	0	0	0
HISTORY	2	0	0	0	0	0	0
HISTORY	3	0	0	0	0	0	0
BOTH	4	0	0	-1	0	0	0
ROMANCE	5	0	0	-1	0	0	0
ROMANCE	6	0	0	1	0	0	0
ROMANCE	7	0	0	-1	0	0	0

E

E

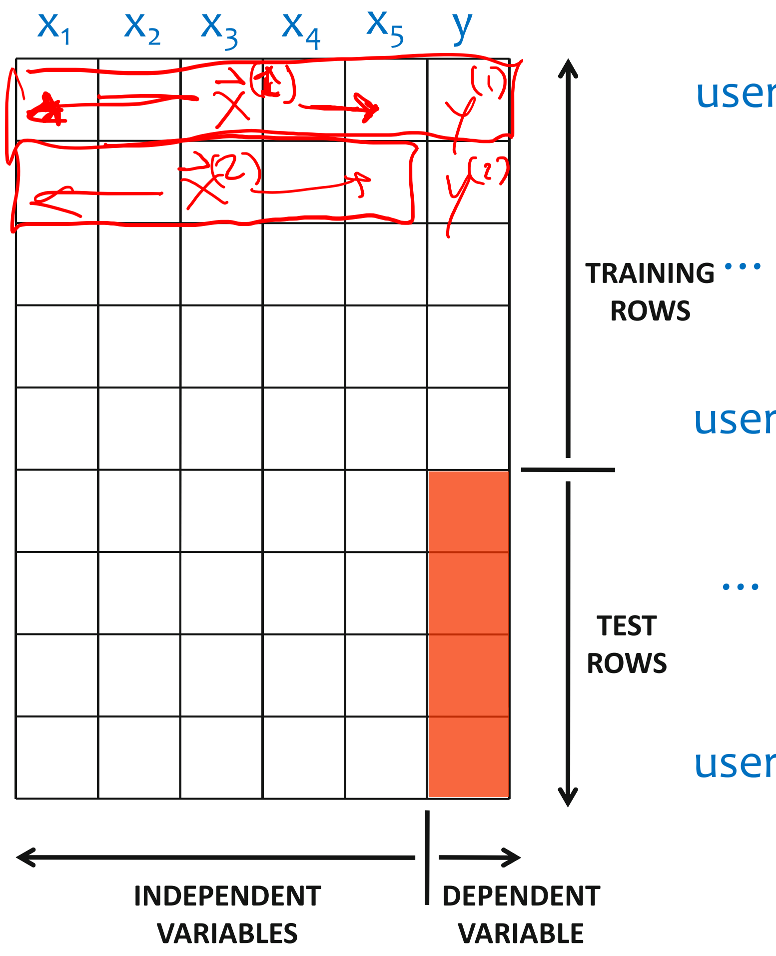
(b) Residual matrix

Figures from Aggarwal (2016)

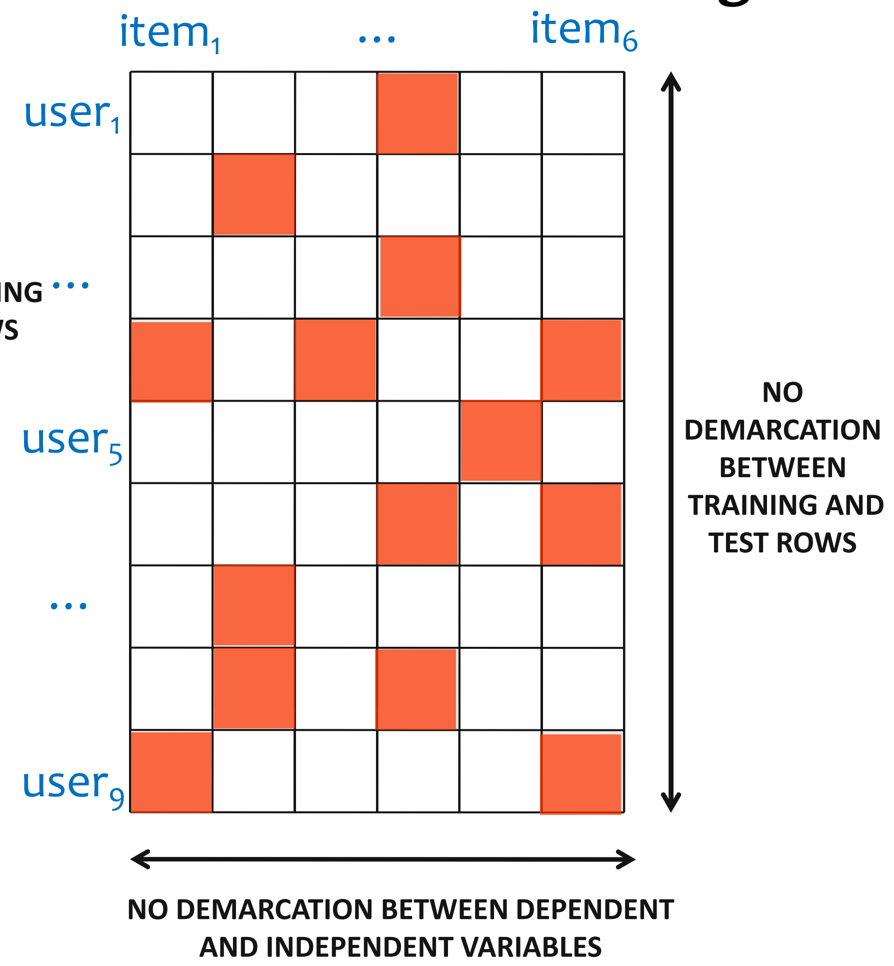
Regression vs. Collaborative Filtering

Goal of each problem is to predict the values of the missing squares

Regression



Collaborative Filtering



Figures from Aggarwal (2016)

UNCONSTRAINED MATRIX FACTORIZATION

Unconstrained Matrix Factorization

Opt. Problem #1 $\overbrace{\text{(fully observed R)}}^{\text{silly}}$

$$\hat{U}, \hat{V} = \underset{U,V}{\operatorname{argmin}}\ J(U,V)$$

$$J(U,V) = \frac{1}{2}\| R - UV^T \|_2^2$$

Opt. Problem #2 $\overbrace{\text{(partially observed R)}}^{\text{real}}$

Let $r_{ij} \triangleq R_{ij}$ ← rating of item j by user i

$\vec{u}_i \triangleq U_{i,\cdot}$ ← user factor

$\vec{v}_j \triangleq V_{j,\cdot} = (V^T)_{\cdot,j}$ ← item factor

learned feature vector, latent

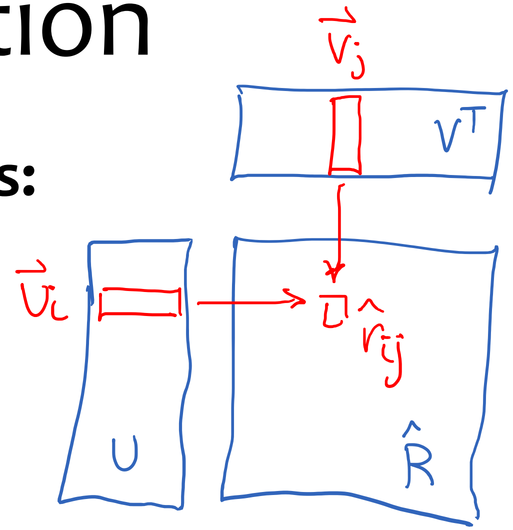
Let $Z = \{(i,j) : r_{ij} \text{ is observed}\}$

$$J(U,V) = \frac{1}{2} \sum_{(i,j) \in Z} (r_{ij} - \vec{u}_i^T \vec{v}_j)^2$$

Model Predictions:

for missing $(i,j) \notin Z$

$$\hat{r}_{ij} = \vec{u}_i^T \vec{v}_j$$



Gradient Descent:

while not converged:

$$g_U = \nabla_U J(U,V)$$

$$g_V = \nabla_V J(U,V)$$

$$U \leftarrow U - \eta g_U$$

$$V \leftarrow V - \eta g_V$$

Unconstrained Matrix Factorization

SGD for UMF:

while not converged:

1. Sample (i, j) from $\mathcal{Z}$ uniformly at random
2. Compute $e_{ij} = r_{ij} - \mathbf{u}_i^T \mathbf{v}_j$
3. Update:

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta \nabla_{\mathbf{u}_i} J_{ij}(\mathbf{U}, \mathbf{V})$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta \nabla_{\mathbf{v}_j} J_{ij}(\mathbf{U}, \mathbf{V})$$

where:

with Regularization

$$J_{ij}(\mathbf{U}, \mathbf{V}) = \frac{1}{2}(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda(\|\mathbf{u}_i\|_2^2 + \|\mathbf{v}_j\|_2^2)$$

$$\nabla_{\mathbf{u}_i} J_{ij}(\mathbf{U}, \mathbf{V}) = -e_{ij}\mathbf{v}_j + \lambda \mathbf{u}_i$$

$$\nabla_{\mathbf{v}_j} J_{ij}(\mathbf{U}, \mathbf{V}) = -e_{ij}\mathbf{u}_i + \lambda \mathbf{v}_j$$

User/Item Bias terms

$$\hat{r}_{ij} = \underline{o_i} + \underline{p_j} + \underline{\mathbf{u}_i^T \mathbf{v}_j}$$

matrix trick:

$$\mathbf{U} = \begin{bmatrix} — & \mathbf{u}_1 & — & o_1 & 1 \\ — & \mathbf{u}_2 & — & o_2 & 1 \\ & \vdots & & & \\ — & \mathbf{u}_m & — & o_m & 1 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} — & \mathbf{p}_1 & — & 1 & p_1 \\ — & \mathbf{p}_2 & — & 1 & p_2 \\ & \vdots & & & \\ — & \mathbf{p}_n & — & 1 & p_n \end{bmatrix}$$

Unconstrained Matrix Factorization

Alternating Least Squares (ALS) for UMF:

Block Coordinate Descent:

while not converged:

1. $\mathbf{U} = \arg\min_{\mathbf{U}} J(\mathbf{U}, \mathbf{V})$
2. $\mathbf{V} = \arg\min_{\mathbf{V}} J(\mathbf{U}, \mathbf{V})$

Applied to UMF:

$$\boxed{J(\mathbf{U}, \mathbf{V}) = \frac{1}{2} \sum_{(i,j) \in \mathcal{Z}} \left(r_{ij} - \mathbf{u}_i^\top \mathbf{v}_j\right)^2}$$

$\rightarrow$ If $\mathbf{U}$ is fixed: Least Squares to solve for $\mathbf{V}$

$\rightarrow$ If $\mathbf{V}$ is fixed: Least Squares to solve for $\mathbf{U}$

The Least Squares Problem:

(i.e. solving Linear Regression in closed form)

$$J(\mathbf{\Theta}) = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \hat{\mathbf{\Theta}}^\top \mathbf{x}_i\right)^2$$

take derivatives, set to zero, and solve in closed form.

Solving $J(\mathbf{U}, \mathbf{V})$ in closed form directly isn't easy and $J(\mathbf{U}, \mathbf{V})$ is nonconvex.

Example Factors

Matrix Factorization

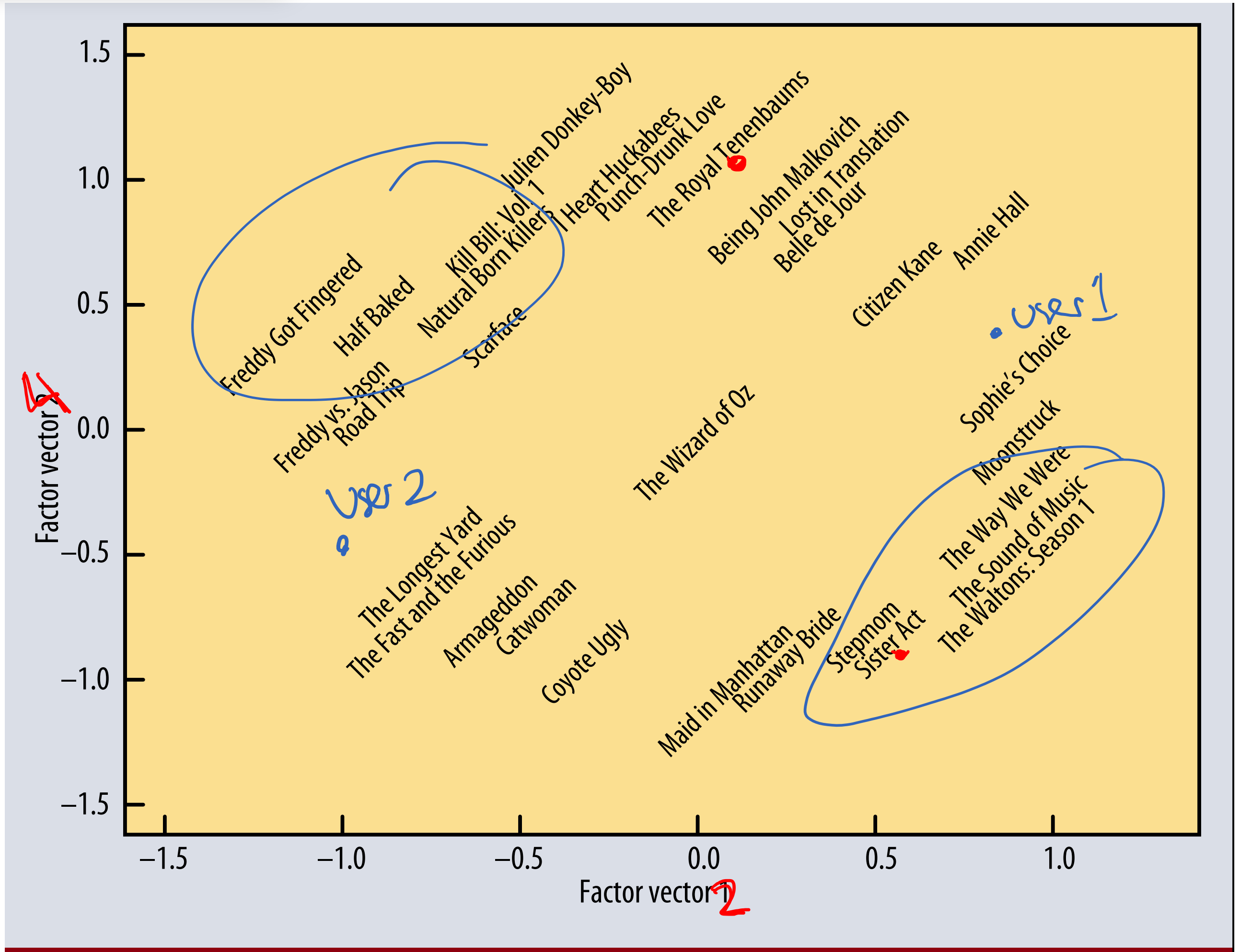


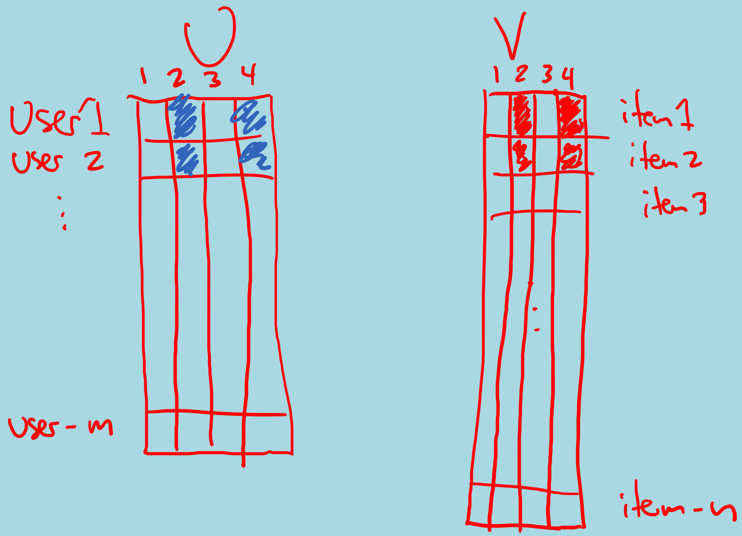
Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Figure from Koren et al. (2009)

Poll 2

Question: Write an algorithm for visualizing two latent factors.

Answer:



Matrix Factorization

Comparison of Optimization Algorithms

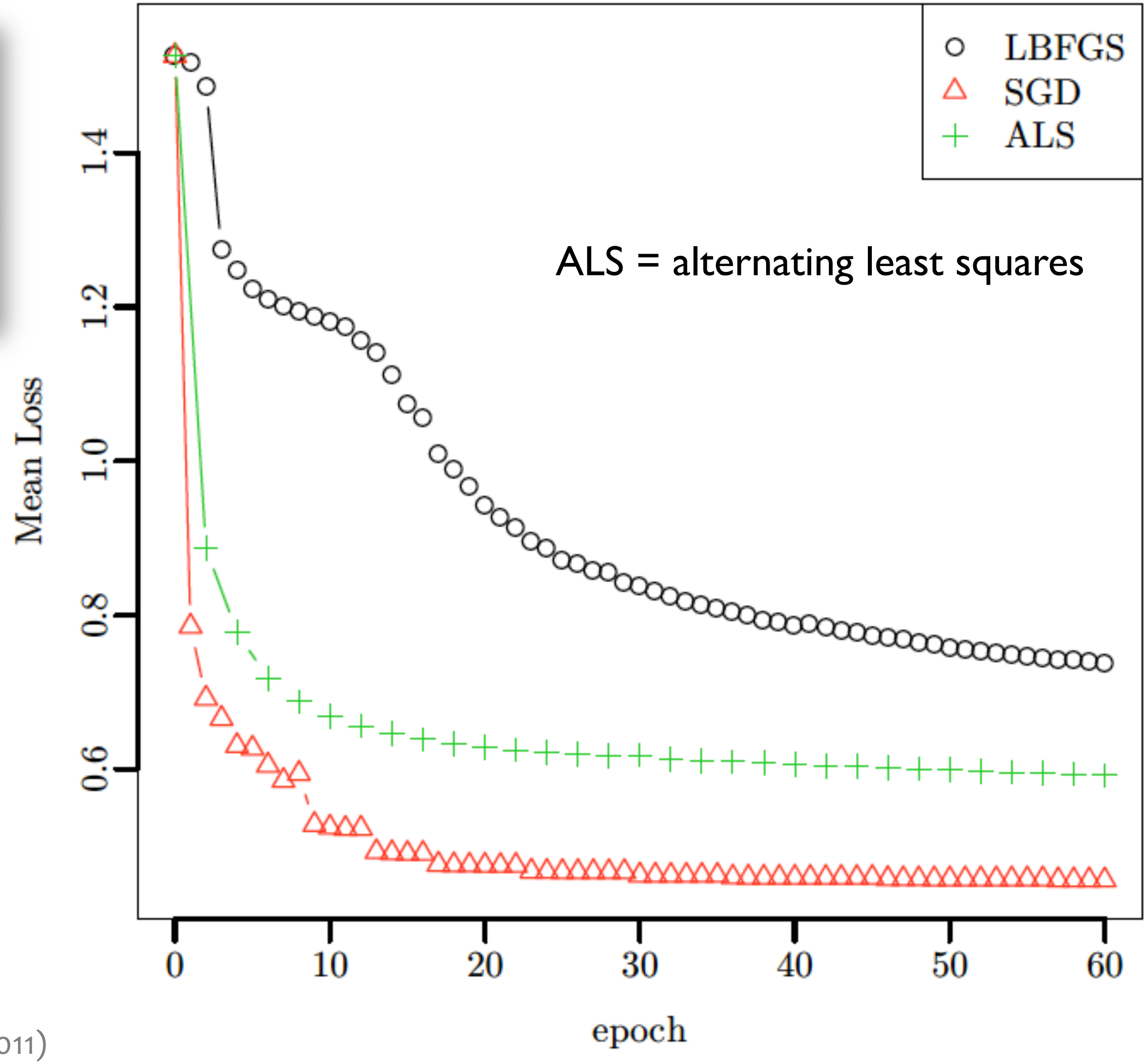


Figure from Gemulla et al. (2011)

SVD FOR COLLABORATIVE FILTERING

Singular Value Decomposition

Definition: Any $m \times n$ matrix $\mathbf{R}$ can be factored as

$$\mathbf{R} = \mathbf{U}\,\boldsymbol{\Sigma}\,\mathbf{V}^\top,$$

where

$$\mathbf{U} \in \mathbb{R}^{m\times m}, \qquad \mathbf{V} \in \mathbb{R}^{n\times n}, \qquad \boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}.$$

Properties:

- Columns of $\mathbf{U}$ are orthonormal $\Rightarrow$ $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$.
- Columns of $\mathbf{V}$ are orthonormal $\Rightarrow$ $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$.
- $\boldsymbol{\Sigma}$ is diagonal (possibly rectangular) with entries
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq 0,$$
called the *singular values*.

Interpretation: SVD writes $\mathbf{R}$ as a sum of rank–1 pieces:

$$\mathbf{R} = \sum_{k=1}^{\mathrm{rank}(\mathbf{R})} \sigma_k\,\mathbf{u}_k\,\mathbf{v}_k^\top.$$

The first few components (largest σ_k) capture the most important structure; smaller ones refine details.

Best Rank–k Approximation: Keeping only the first k singular values gives

$$\mathbf{R}_k = \sum_{i=1}^{k} \sigma_i\,\mathbf{u}_i\,\mathbf{v}_i^\top,$$

the closest rank–k matrix to $\mathbf{R}$ under the Frobenius norm.

Singular Value Decomposition for Collaborative Filtering

For any arbitrary matrix $\mathbf{A}$, SVD gives a decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$$

where $\mathbf{\Lambda}$ is a diagonal matrix, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

Suppose we have the SVD of our ratings matrix

$$R = Q\Sigma P^T,$$

but then we truncate each of Q, Σ, and P s.t. Q and P have only k columns and Σ is $k \times k$:

$$R \approx Q_k \Sigma_k P_k^T$$

For collaborative filtering, let:

$$U \triangleq Q_k \Sigma_k$$
$$V \triangleq P_k$$
$$\Rightarrow U, V = \underset{U,V}{\operatorname{argmin}} \frac{1}{2} ||R - UV^T||_2^2$$

s.t. columns of U are mutually orthogonal

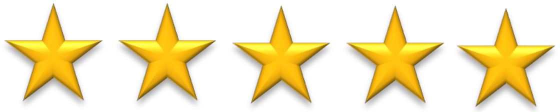
s.t. columns of V are mutually orthogonal

Theorem: If *R* fully observed and no regularization, the optimal UV^T from SVD equals the optimal UV^T from Unconstrained MF

NON-NEGATIVE MATRIX FACTORIZATION

Implicit Feedback Datasets

- What information does a five-star rating contain?



- Implicit Feedback Datasets:
  - In many settings, users don't have a way of expressing *dislike* for an item (e.g. can't provide negative ratings)
  - The only mechanism for feedback is to "like" something
- Examples:
  - Facebook has a "Like" button, but no "Dislike" button
  - Google's "+1" button
  - Pinterest pins
  - Purchasing an item on Amazon indicates a preference for it, but there are many reasons you might *not* purchase an item (besides dislike)
  - Search engines collect click data but don't have a clear mechanism for observing dislike of a webpage

Examples from Aggarwal (2016)

Non-negative Matrix Factorization

Constrained Optimization Problem:

$$U, V = \operatorname*{argmin}_{U,V} \frac{1}{2} ||R - UV^T||_2^2$$

$$\text{s.t. } U_{ij} \geq 0$$

$$\text{s.t. } V_{ij} \geq 0$$

Multiplicative Updates: simple iterative algorithm for solving just involves multiplying a few entries together

Fighting Fire with Fire: Using Antidote Data to Improve Polarization and Fairness of Recommender Systems

Bashir Rastegarpanah
Boston University
bashir@bu.edu

Krishna P. Gummadi
MPI-SWS
gummadi@mpi-sws.org

Mark Crovella
Boston University
crovella@bu.edu

where $\mathbf{S}_j = \sum_{i \in \Omega_j} \mathbf{u}_i \mathbf{u}_i^\top + \tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top + \lambda \mathbf{I}_\ell$.

By using (9) instead of the general formula in (5) we can significantly reduce the number of computations required for finding the gradient of the utility function with respect to the antidote data. Furthermore, the term ${\mathbf{g}_j}^\top \mathbf{U}^\top \mathbf{S}_j^{-1}$ appears in all the partial derivatives that correspond to elements in column j of $\tilde{\mathbf{X}}$ and can be precomputed in each iteration of the algorithm and reused for computing partial derivatives with respect to different antidote users.

5 SOCIAL OBJECTIVE FUNCTIONS

The previous section developed a general framework for improving various properties of recommender systems; in this section we show how to apply that framework specifically to issues of polarization and fairness.

As described in Section 2, polarization is the degree to which opinions, views, and sentiments diverge within a population. Recommender systems can capture this effect through the ratings that they present for items. To formalize this notion, we define polarization in terms of the variability of predicted ratings when compared across users. In fact, we note that both very high variability, and very low variability of ratings may be undesirable. In the case of high variability, users have strongly divergent opinions, leading to conflict. Recent analyses of the YouTube recommendation system have suggested that it can enhance this effect [29, 30]. On the other hand, the convergence of user preferences, i.e., very low variability of ratings given to each item across users, corresponds to increased homogeneity, an undesirable phenomenon that may occur as users interact with a recommender system [11]. As a result, in what follows we consider using antidote data in both ways: to either increase or decrease polarization.

As also described in Section 2, unfairness is a topic of growing interest in machine learning. Following the discussion in that section, we consider a recommender system fair if it provides equal quality of service (i.e., prediction accuracy) to all users or all groups of users [36].

Next we formally define the metrics that specify the objective functions associated with each of the above objectives. Since the gradient of each objective function is used in the optimization algorithm, for reproducibility we provide the details about derivation of the gradients in appendix A.2.

5.1 Polarization

To capture polarization, we seek to measure the extent to which the user ratings *disagree*. Thus, to measure user polarization we consider the estimated ratings $\hat{\mathbf{X}}$, and we define the polarization metric as the normalized sum of pairwise euclidean distances between estimated user ratings, i.e., between rows of $\hat{\mathbf{X}}$. In particular:

$$R_{pol}(\hat{\mathbf{X}}) = \frac{1}{n^2 d} \sum_{k=1}^{n} \sum_{l>k} ||\hat{\mathbf{x}}^k - \hat{\mathbf{x}}^l||^2 \tag{10}$$

The normalization term $\frac{1}{n^2 d}$ in (10) makes the polarization metric identical to the following definition: [4]

$$R_{pol}(\hat{\mathbf{X}}) = \frac{1}{d} \sum_{j=1}^{d} \sigma_j^2 \tag{11}$$

where σ_j^2 is the variance of estimated user ratings for item j. Thus this polarization metric can be interpreted either as the average of the variances of estimated ratings in each item, or equivalently as the average user disagreement over all items.

5.2 Fairness

Individual fairness. For each user i, we define ℓ_i, the loss of user i, as the mean squared estimation error over known ratings of user i:

$$\ell_i = \frac{||P_{\Omega^i}(\hat{\mathbf{x}}^i - \mathbf{x}^i)||_2^2}{|\Omega^i|} \tag{12}$$

Then we define the individual unfairness as the variance of the user losses:[5]

$$R_{indv}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l>k} (\ell_k - \ell_l)^2 \tag{13}$$

To improve individual fairness, we seek to minimize R_{indv}.

Group fairness. Let I be the set of all users/items and $G = \{G_1 \ldots, G_g\}$ be a partition of users/items into g groups, i.e., $I = \bigcup_{i \in \{1,\ldots,g\}} G_i$. We define the loss of group i as the mean squared estimation error over all known ratings in group i:

$$L_i = \frac{||P_{\Omega_{G_i}}(\hat{\mathbf{X}} - \mathbf{X})||_2^2}{|\Omega_{G_i}|} \tag{14}$$

For a given partition G, we define the group unfairness as the variance of all group losses:

$$R_{grp}(\mathbf{X}, \hat{\mathbf{X}}, G) = \frac{1}{g^2} \sum_{k=1}^{g} \sum_{l>k} (L_k - L_l)^2 \tag{15}$$

Again, to improve group fairness, we seek to minimize R_{grp}.

5.3 Accuracy vs. Social Welfare

Adding antidote data to the system to improve a social utility will also have an effect on the overall prediction accuracy. Previous works have considered social objectives as regularizers or constraints added to the recommender model (eg, [8, 25, 37]), implying a trade-off between the prediction accuracy and a social objective.

However, in the case of the metrics we define here, the relationship is not as simple. Considering polarization, we find that in general, increasing or decreasing polarization will tend to decrease system accuracy. In either case we find that system accuracy only declines slightly in our experiments; we report on the specific values in Section 6. Considering either individual or group unfairness, the situation is more subtle. Note that our unfairness metrics will be exactly zero for a system with zero error (perfect accuracy). As a

[4] We can derive it by rewriting (10) as $R_{pol}(\hat{\mathbf{X}}) = \frac{1}{d} \sum_{j=1}^{d} \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l>k} (\hat{x}_{kj} - \hat{x}_{lj})^2$.

[5] Note that for a set of equally likely values $x_1, \ldots, x_n$ the variance can be expressed without referring to the mean as: $\frac{1}{n^2} \sum_i \sum_{j>i} (x_i - x_j)^2$.

Summary

- Recommender systems solve many **real-world** (*large-scale) **problems**
- Collaborative filtering by Matrix Factorization (MF) is an **efficient** and **effective** approach
- MF is just another example of a **common recipe**:
    1. define a model
    2. define an objective function
    3. optimize with your favorite black box optimizer
       (e.g. SGD, Gradient Descent, Block Coordinate Descent aka. Alternating Least Squares)

Learning Objectives

Recommender Systems

You should be able to…

1. Compare and contrast the properties of various families of recommender system algorithms: content filtering, collaborative filtering, neighborhood methods, latent factor methods
2. Formulate a squared error objective function for the matrix factorization problem
3. Implement unconstrained matrix factorization with a variety of different optimization techniques: gradient descent, stochastic gradient descent, alternating least squares
4. Offer intuitions for why the parameters learned by matrix factorization can be understood as user factors and item factors

Recommender Systems

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Top performing systems were ensembles

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