10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Recommender Systems

Matt Gormley
Lecture 26.5
Apr. 20, 2022

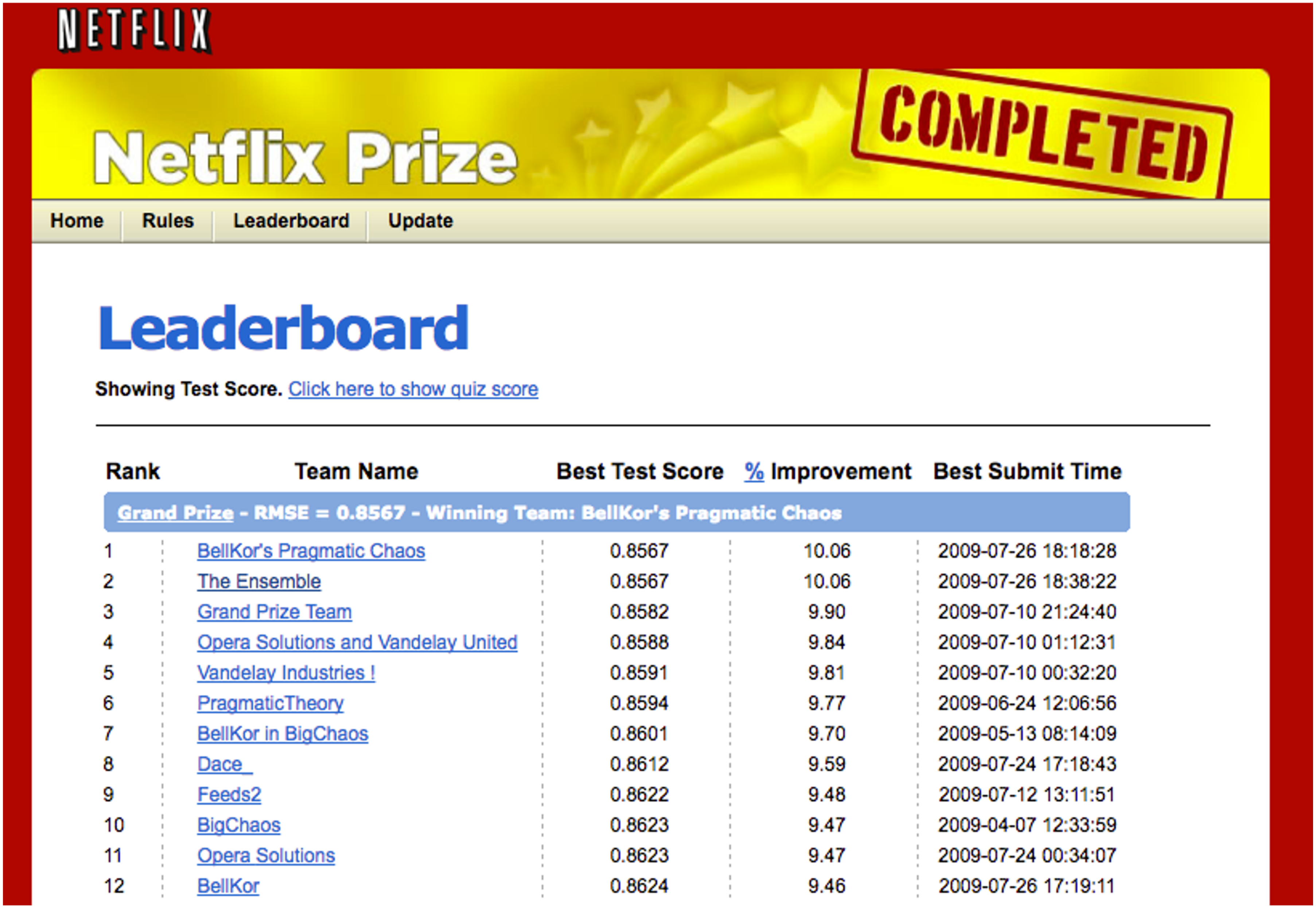
RECOMMENDER SYSTEMS

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

- *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

- *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

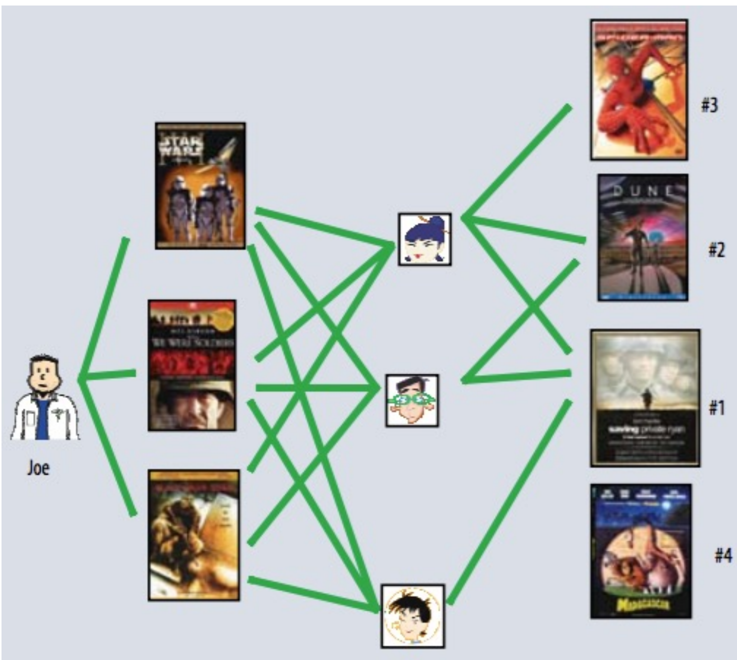
Collaborative Filtering

- **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

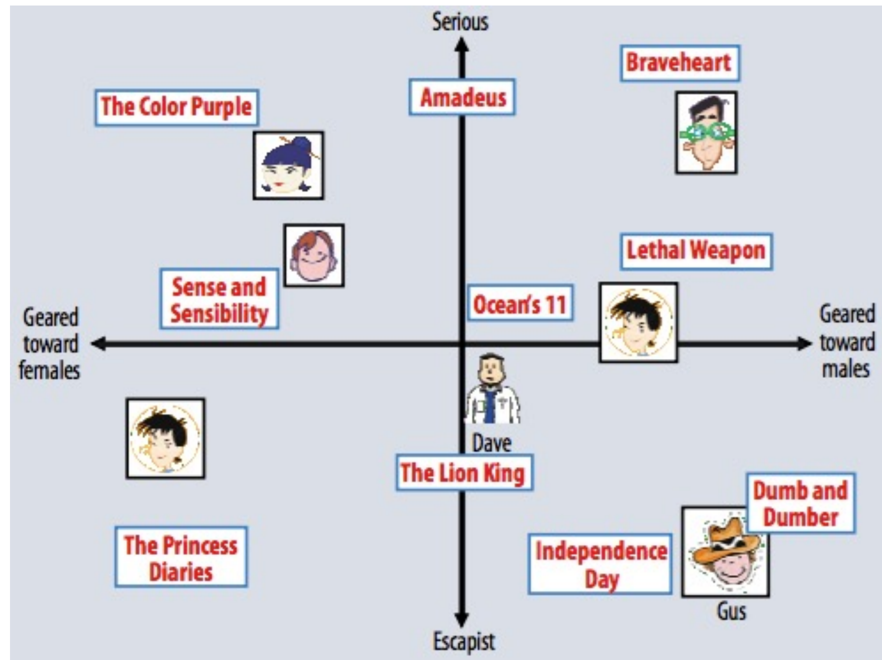
Slide from William Cohen

Two Types of Collaborative Filtering

1. Neighborhood Methods



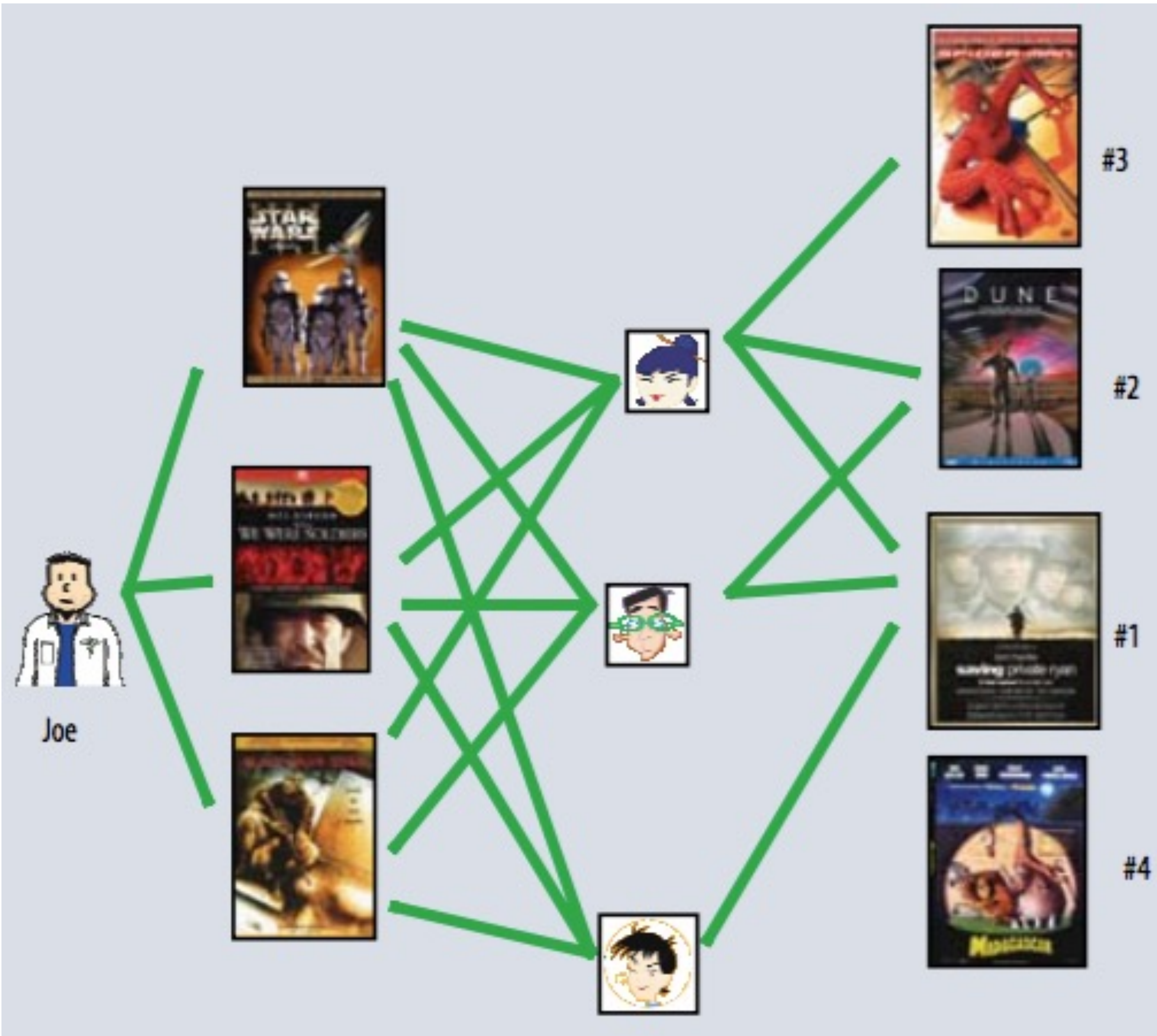
2. Latent Factor Methods



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:

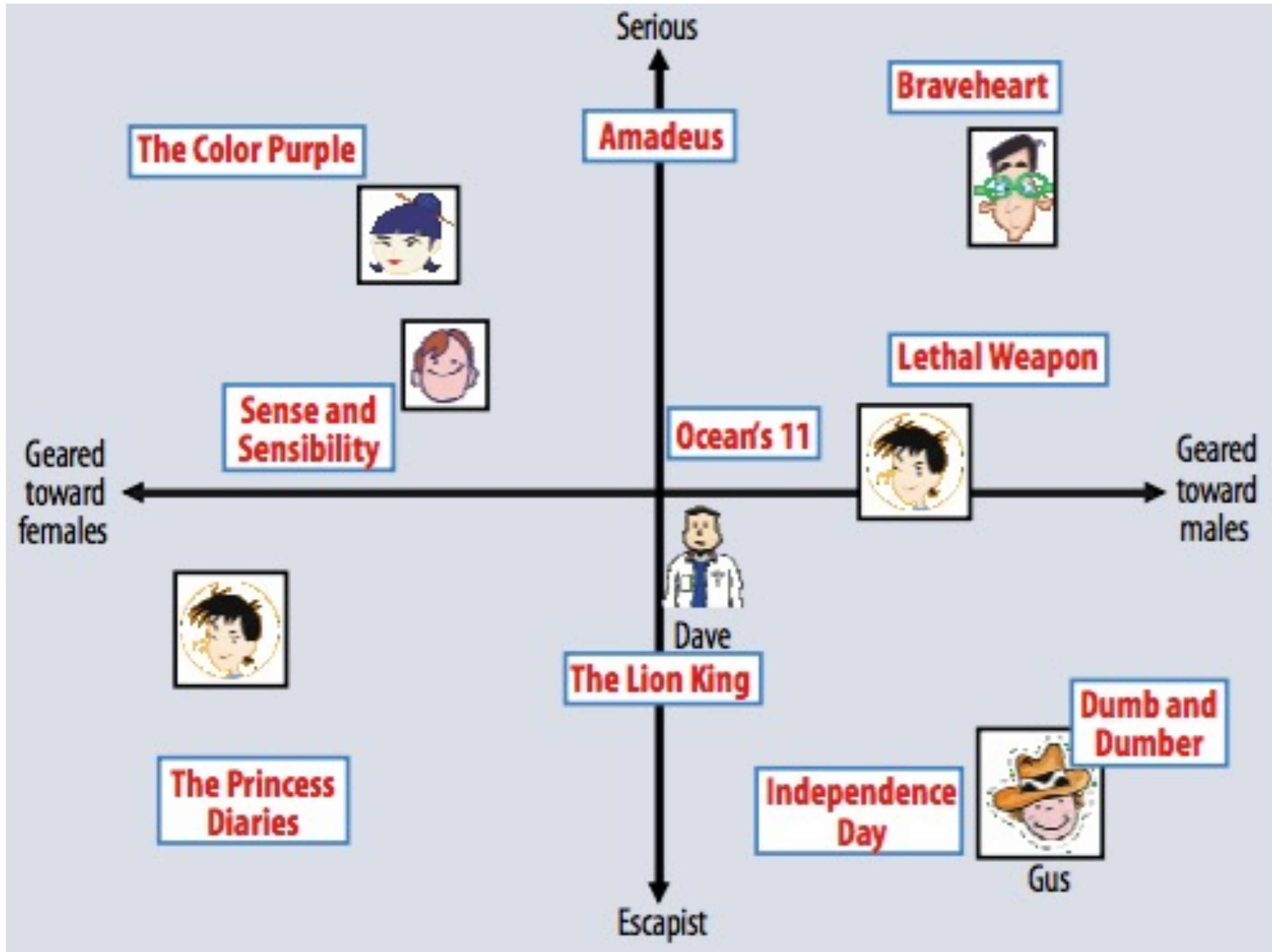
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Figures from Koren et al. (2009)

Two Types of Collaborative Filtering

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



Figures from Koren et al. (2009)

Recommending Movies

Question:

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

Select all that apply

A. K-Means
B. collaborative filtering
C. latent factor methods
D. ensemble methods
E. content filtering
F. neighborhood methods
G. recommender systems

Answer:

MATRIX FACTORIZATION

Matrix Factorization

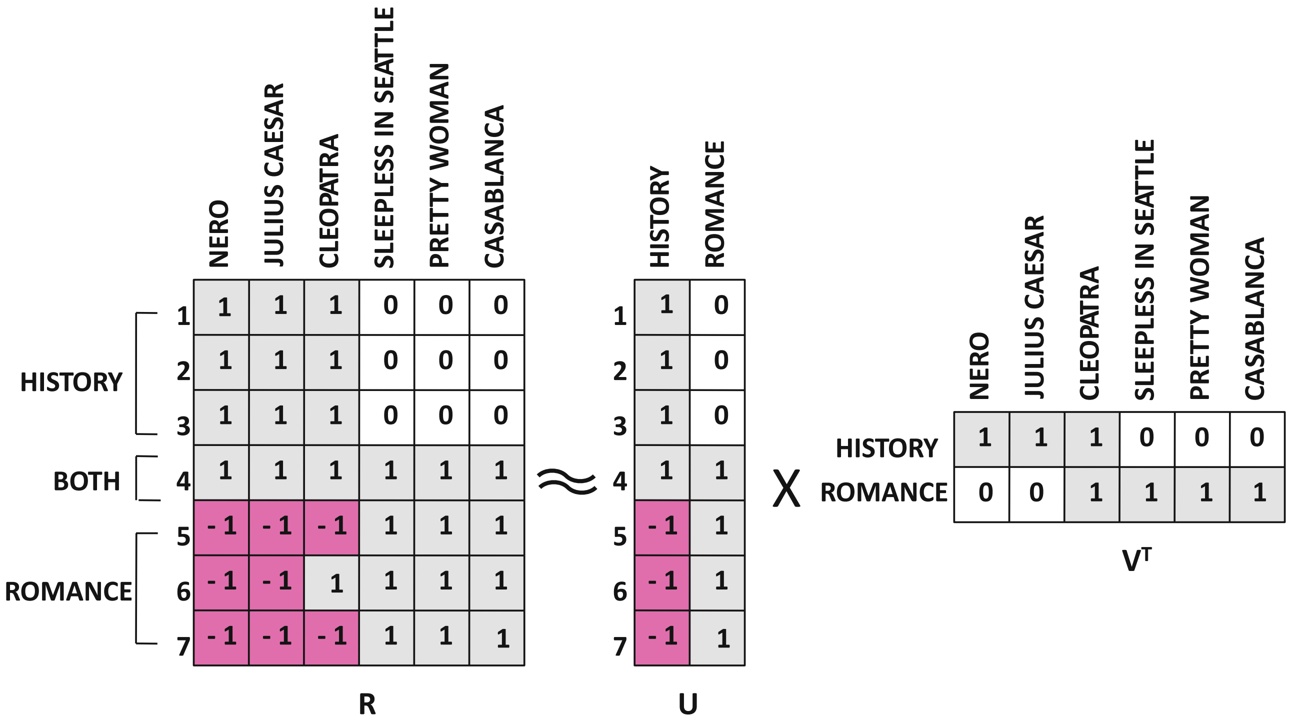
- Many different ways of factorizing a matrix
- We'll consider three:
    1. Unconstrained Matrix Factorization
    2. Singular Value Decomposition
    3. Non-negative Matrix Factorization
- MF is just another example of a **common recipe**:
    1. define a model
    2. define an objective function
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

Matrix Factorization

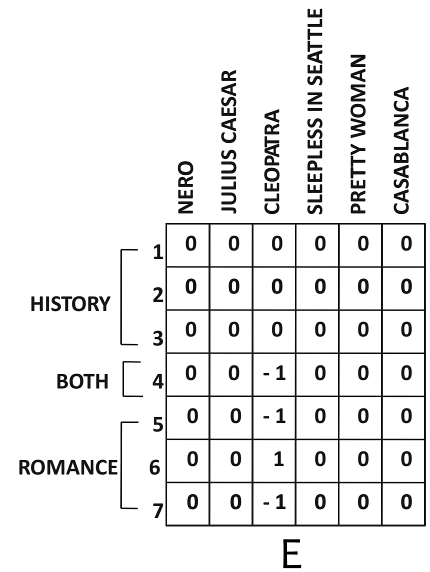
Whiteboard

- Background: Low-rank Factorizations
- Residual matrix

Example: MF for Netflix Problem



(a) Example of rank-2 matrix factorization

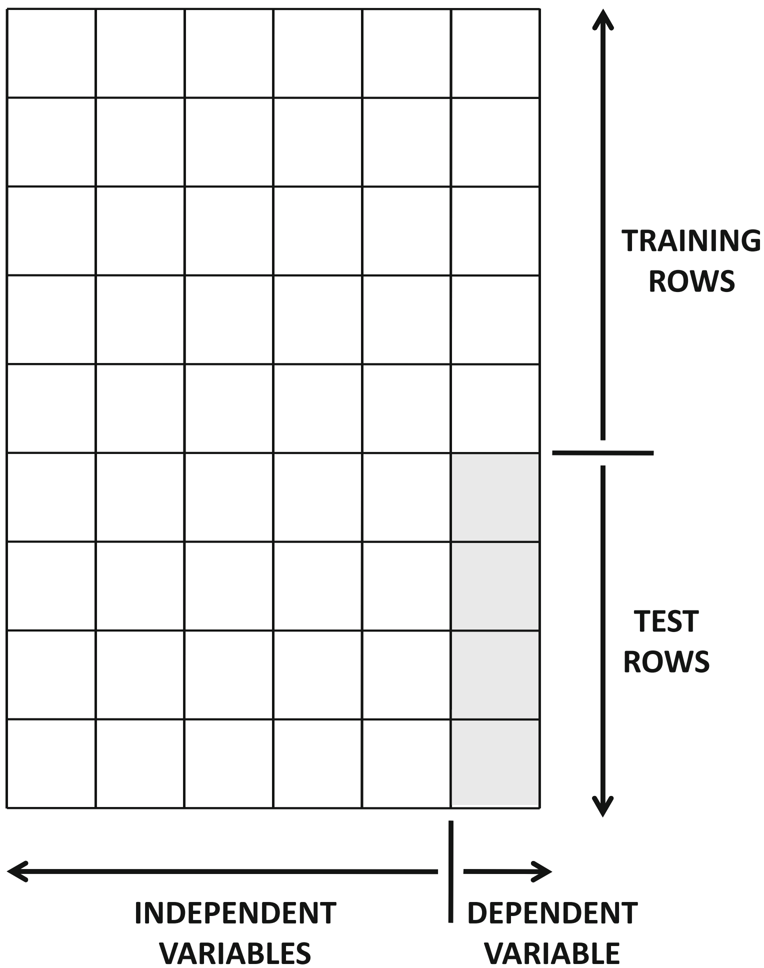


(b) Residual matrix

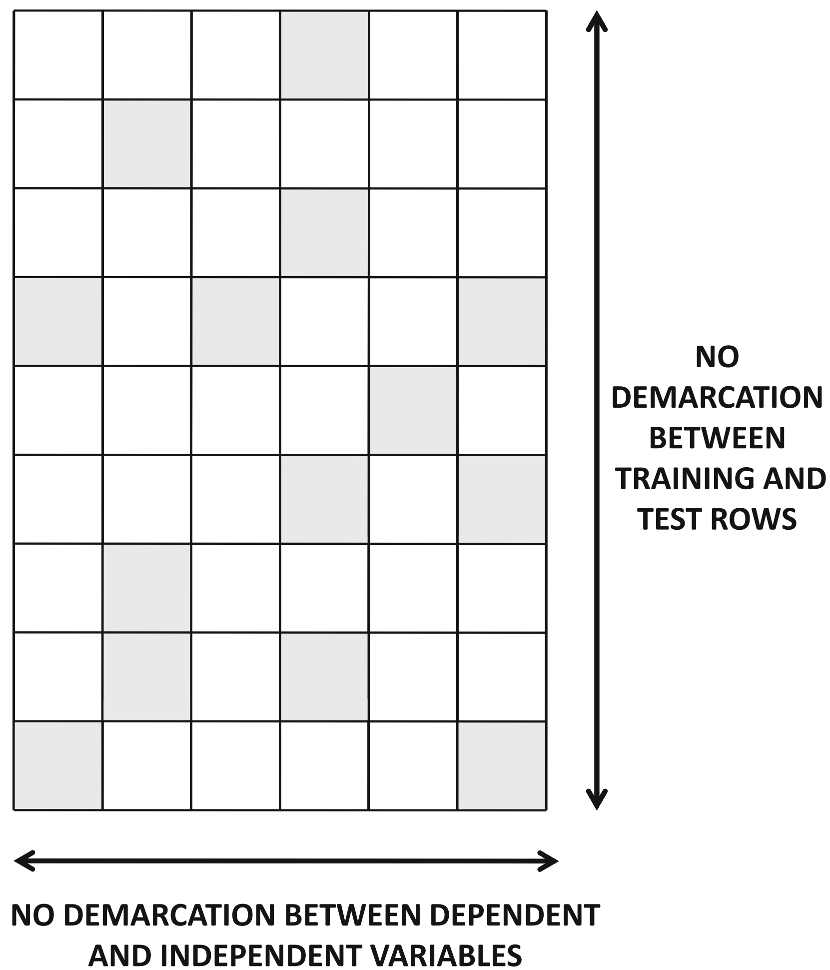
Figures from Aggarwal (2016)

Regression vs. Collaborative Filtering

Regression



Collaborative Filtering



Figures from Aggarwal (2016)

UNCONSTRAINED MATRIX FACTORIZATION

Unconstrained Matrix Factorization

Whiteboard

– Optimization problem
– SGD
– SGD with Regularization
– Alternating Least Squares
– User/item bias terms (matrix trick)

Unconstrained Matrix Factorization

SGD for UMF:

While not converged:

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

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

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

W/ Regularization

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

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

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

where $e_{ij} = r_{ij} - \vec{u}_i^T \vec{v}_j$

Unconstrained Matrix Factorization

SGD for UMF:

User/Item Bias terms

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

matrix trick:

$$U = \begin{bmatrix} \cdots & o_1 & 1 \\ \cdots & o_2 & 1 \\ & \vdots & \vdots \\ \cdots & o_m & 1 \end{bmatrix} \qquad V = \begin{bmatrix} \cdots & 1 & p_1 \\ \cdots & 1 & p_2 \\ & \vdots & \vdots \\ \cdots & 1 & p_n \end{bmatrix}$$

Unconstrained Matrix Factorization

Alternating Least Squares (ALS) for UMF:

Block Coord. Descent:

while not converged:

① $U = \underset{U}{\operatorname{argmin}} J(U,V)$

② $V = \underset{V}{\operatorname{argmin}} J(U,V)$

convex and easy to solve in closed form

Lin. Reg.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \vec{\theta}^T \vec{x}_i)^2$$

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

if U is fixed Least Squares in V

if V is fixed Least Squares in U

Option #1: take derivatives, set to zero and solve in closed form

★ solving $J(U,V)$ in closed form directly isn't easy and $J(U,V)$ is nonconvex

Matrix Factorization

Example Factors

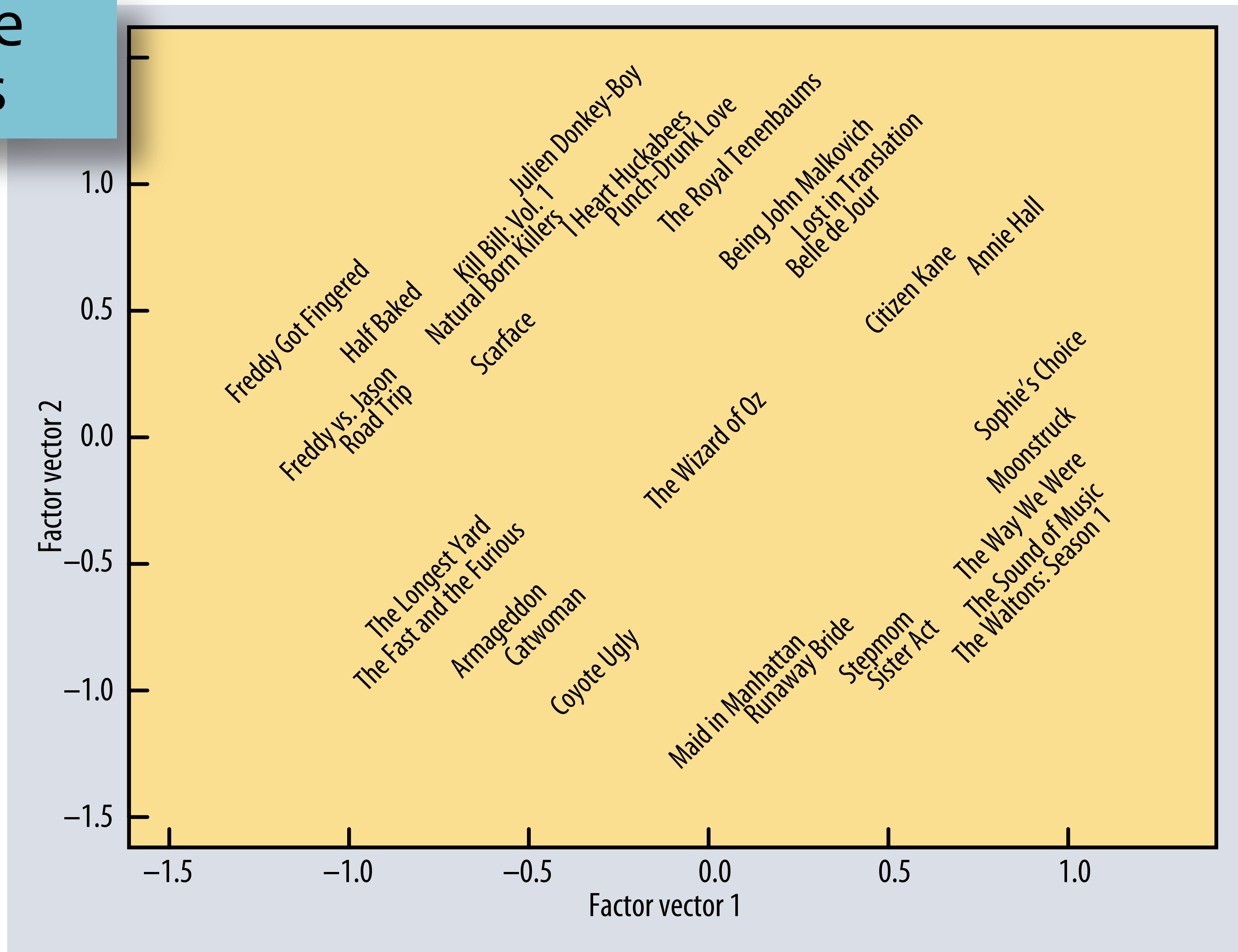


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky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

Figure from Koren et al. (2009)

Matrix Factorization

Comparison of Optimization Algorithms

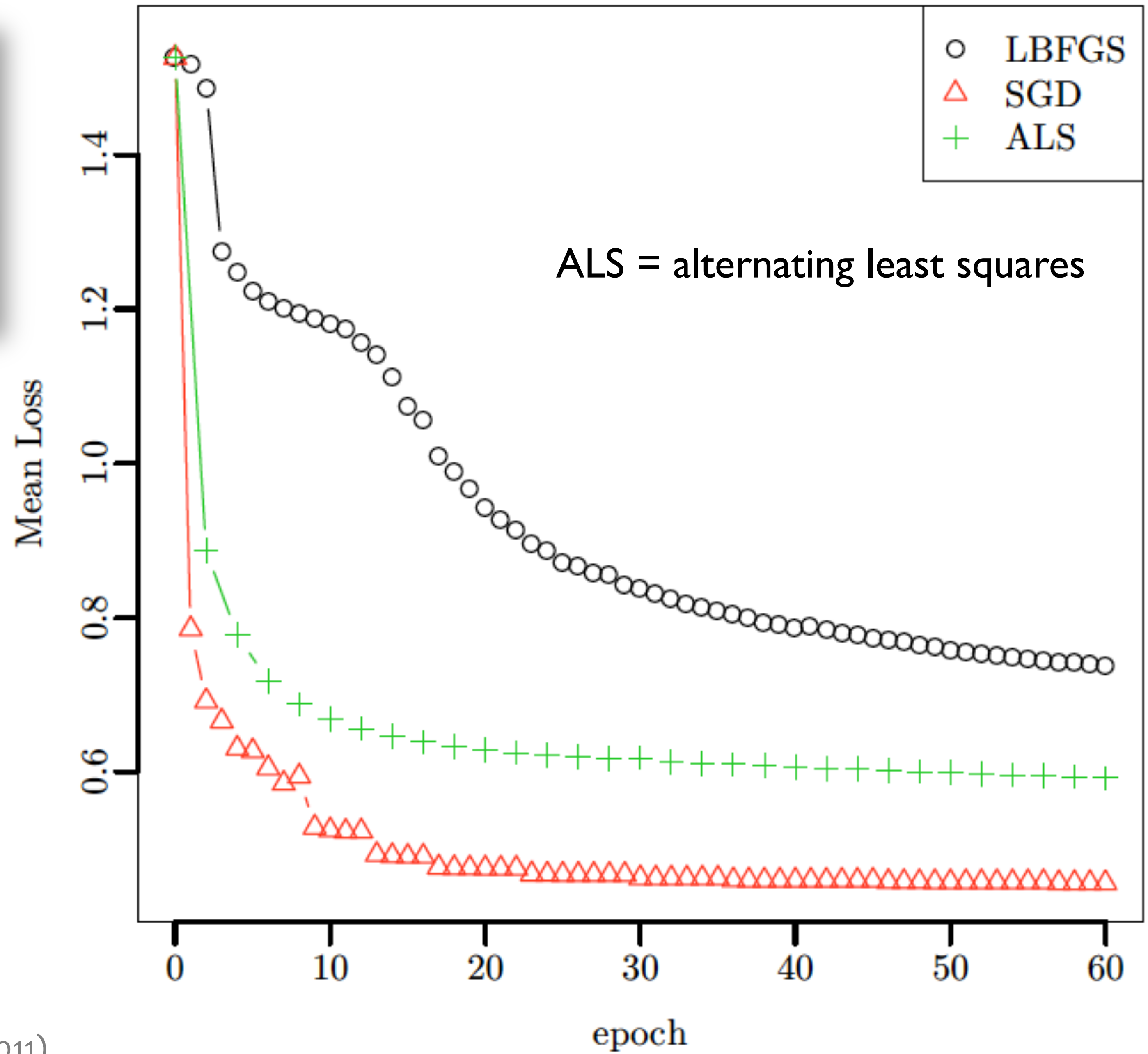


Figure from Gemulla et al. (2011)

SVD FOR COLLABORATIVE FILTERING

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:

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

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

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

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 $G = \{G_1 \dots, G_g\}$ be a partition of users/items into g groups, i.e., $I = \bigcup_{i \in \{1,\dots,g\}} G_i$. 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 $x_1, \dots, x_n$ 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

EXTRA SLIDES ON UMF

Unconstrained Matrix Factorization

In-Class Exercise

Derive a block coordinate descent algorithm for the Unconstrained Matrix Factorization problem.

- User vectors:

  $$\mathbf{w}_u \in \mathbb{R}^r$$

- Item vectors:

  $$\mathbf{h}_i \in \mathbb{R}^r$$

- Rating prediction:

  $$v_{ui} = \mathbf{w}_u^T \mathbf{h}_i$$

- Set of non-missing entries:

  $\mathcal{Z} = \{(u,i) : v_{ui}$ is observed$\}$

- Objective:

  $$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

Matrix Factorization
(with matrices)

- User vectors:

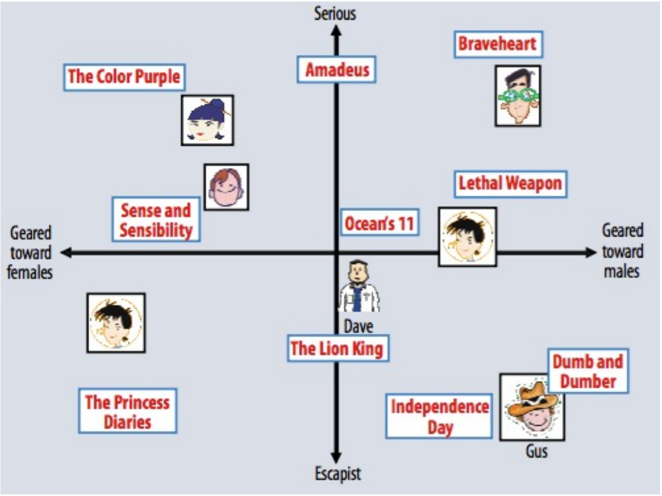
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

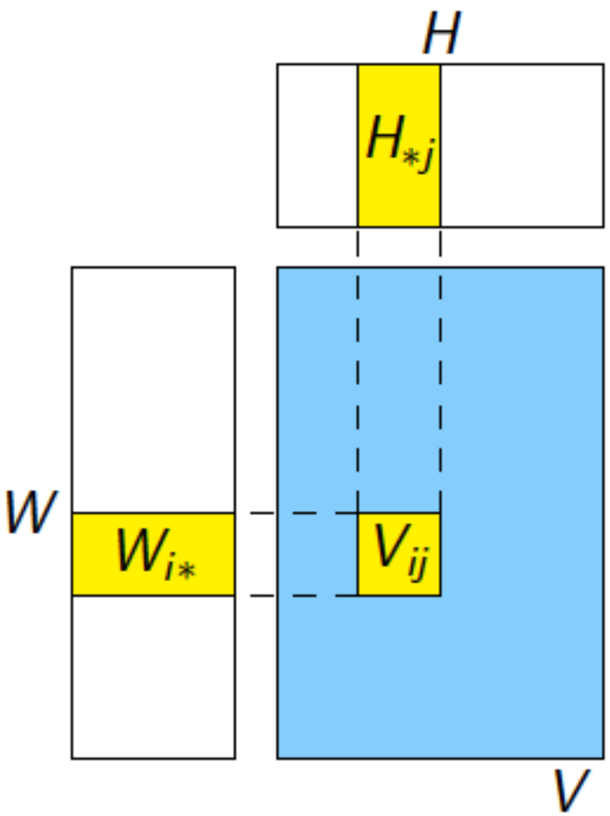
$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$V_{ui} = W_{u*}H_{*i}$$
$$= [WH]_{ui}$$



Figures from Koren et al. (2009)



Figures from Gemulla et al. (2011)

Matrix Factorization
(with vectors)

- User vectors:

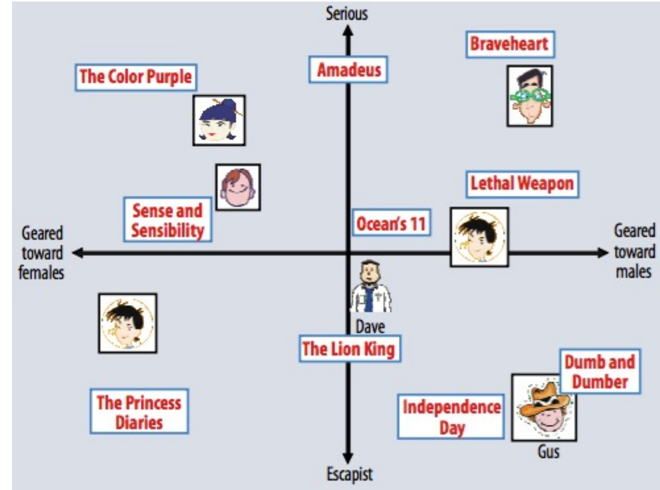
$$\mathbf{w}_u \in \mathbb{R}^r$$

- Item vectors:

$$\mathbf{h}_i \in \mathbb{R}^r$$

- Rating prediction:

$$v_{ui} = \mathbf{w}_u^T \mathbf{h}_i$$



Figures from Koren et al. (2009)

Matrix Factorization

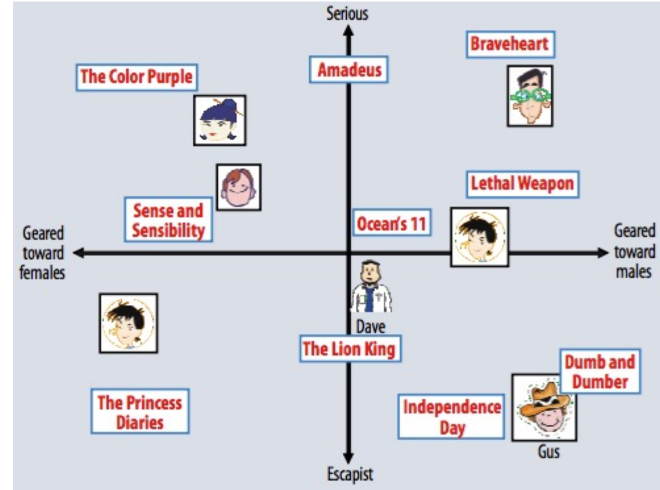
(with vectors)

- Set of non-missing entries:

$$\mathcal{Z} = \{(u, i) : v_{ui} \text{ is observed}\}$$

- Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\text{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$



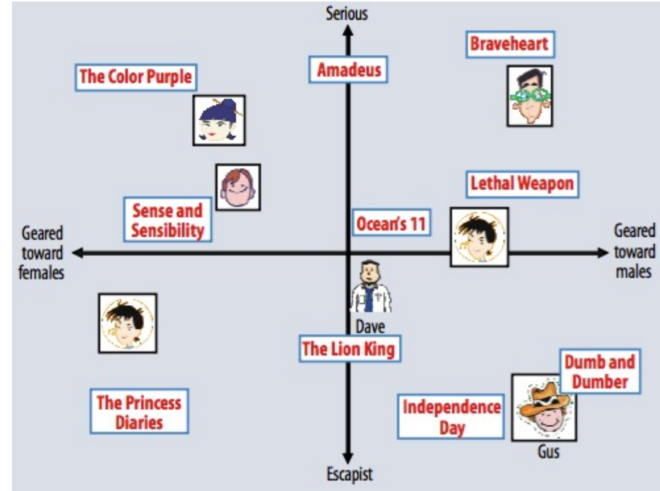
Figures from Koren et al. (2009)

Matrix Factorization
(with vectors)

- Regularized Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\text{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

$$+ \lambda(\sum_i ||\mathbf{w}_i||^2 + \sum_u ||\mathbf{h}_u||^2)$$



Figures from Koren et al. (2009)

Matrix Factorization
(with vectors)

- Regularized Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

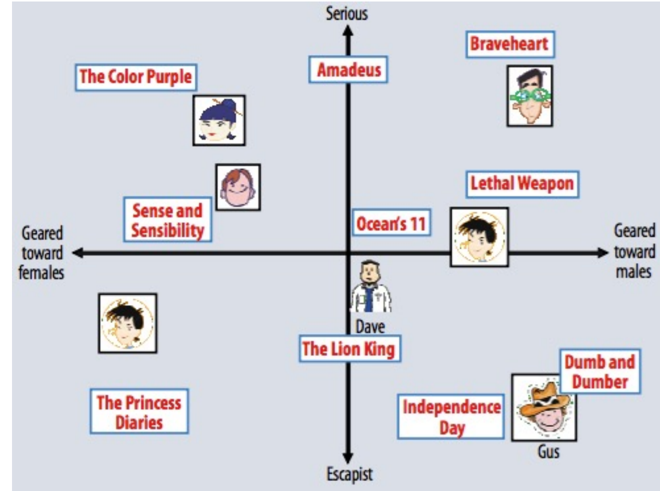
$$+ \lambda(\sum_i ||\mathbf{w}_i||^2 + \sum_u ||\mathbf{h}_u||^2)$$

- SGD update for random (u,i):

$$e_{ui} \leftarrow v_{ui} - \mathbf{w}_u^T \mathbf{h}_i$$

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \gamma(e_{ui}\mathbf{h}_i - \lambda \mathbf{w}_u)$$

$$\mathbf{h}_i \leftarrow \mathbf{h}_i + \gamma(e_{ui}\mathbf{w}_u - \lambda \mathbf{h}_i)$$



Figures from Koren et al. (2009)

Matrix Factorization
(with matrices)

- User vectors:

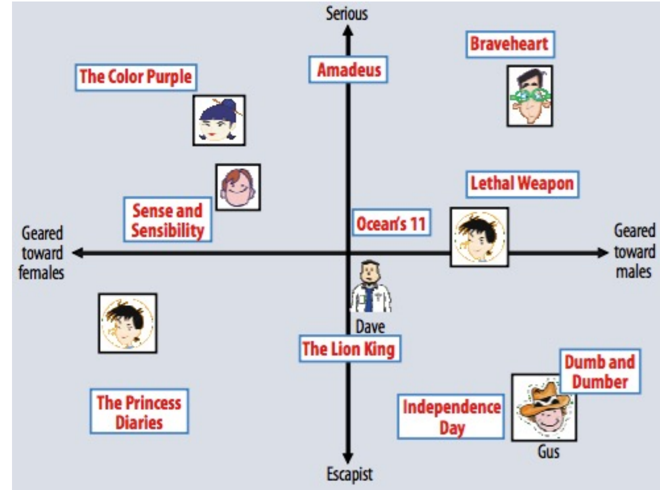
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

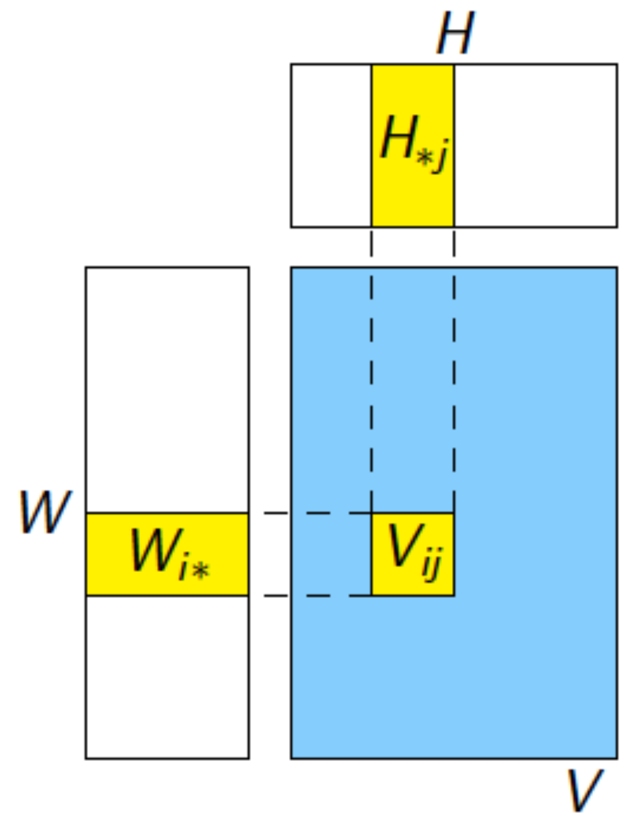
$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$\begin{aligned} V_{ui} &= W_{u*} H_{*i} \\ &= [WH]_{ui} \end{aligned}$$



Figures from Koren et al. (2009)



Figures from Gemulla et al. (2011)

Matrix Factorization
(with matrices)

- SGD

require that the loss can be written as

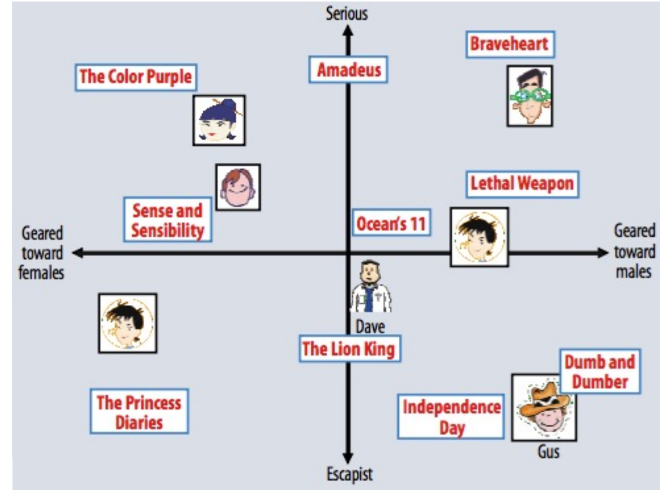
$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

Algorithm 1 SGD for Matrix Factorization

Require: A training set Z, initial values $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$
  while not converged **do** {step}
    Select a training point $(i, j) \in Z$ uniformly at random.
    $\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
    $\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$
  end while

step size

Figure from Gemulla et al. (2011)



Figures from Koren et al. (2009)

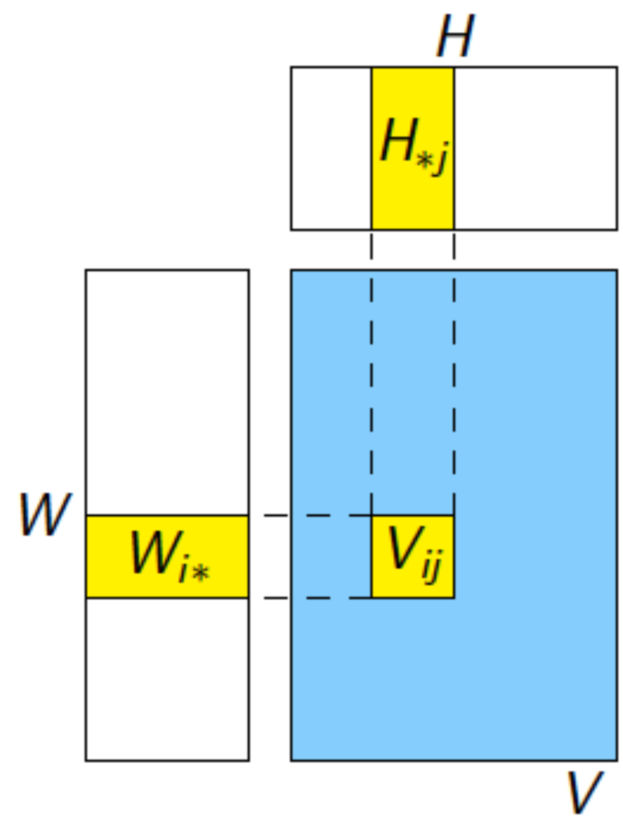


Figure from Gemulla et al. (2011)