

## **COEN281 -- Introduction to Pattern Recognition and Data Mining**

### **Lecture 17: Collaborative Filtering**

Instructor: Dr. Giovanni Seni  
*GSeni@scu.edu*

***Department of Computer Engineering  
Santa Clara University***

Fall/18

## **Syllabus**

Week 1	Introduction; R (Ch.1)
Week 2	Bayesian Decision Theory (Ch.2; DHS: 2.1-2.6, 2.9) Parameter Estimation (DHS: 3.1-3.4)
Week 3	Linear Discriminant Functions (Ch.3&4; DHS: 3.8.2, 5.1-5.8) Regularization (Ch.6; SE: Ch.3)
Week 4	Neural Networks (DHS: 6.1-6.6, 6.8);
Week 5	Support Vector Machines (Ch.9)
Week 6	Decision Trees (Ch. 8.1; DHS: 8.3; Ch 2 SE)
Week 7	Ensemble Methods (Ch. 8.2; SE: Ch 4, 5)
Week 8	Clustering (Ch. 10; DHS: 10.6, 10.7) Clustering (DHS: 10.9); How many clusters are there? (DHS: 10.10)
Week 9	Non-metric: Association Rules <b>Collaborative Filtering</b>
Week 10	Text Retrieval; Other topics

## Overview

- Recommender Systems Strategies
  - *Content filtering vs. Collaborative filtering*
- Collaborative Filtering Algorithms
  - Neighborhood Models
  - Latent Factor Models
  - Integrated Model
- Top-K Recommendations
- Summary

## Recommender Systems Strategies

- Personalization is key to enhancing user satisfaction and loyalty
- **Content filtering**
  - Creates a *profile* for each user or product to characterize its nature
    - E.g., Music Genome Project used by Pandora
      - Each song is represented by 150-500 “genes”
      - E.g., *gender of lead vocalist, level of distortion on the electric guitar, etc*
    - Profiles allow association of users with matching products
  - **Collaborative filtering**
    - Relies only on past user behavior – e.g., transactions or ratings
    - Domain free, yet can model data aspects that are often elusive and difficult to profile
    - More accurate... but suffers from *cold start problem*

## Ratings Matrix

- Numeric  $|U| \times |I|$  matrix

	Item A	Item B	Item C	Item D	Item E	Item F	Item G	Item H
User 1		5		3			2	
User 2	4		5			4		1
User 3		4		3			2	
User 4	1	2				5		3
User 5			3		4			2
User 6		2			1			2
User 7	4			5			4	

- Explicit preferences – e.g.,
  - 1: no interest... 5: strong interest
- Very sparse: 99% missing on Netflix data

## Collaborative Filtering

### Problem Definition

- Input:
  - Rating matrix ( $R_{|U| \times |I|}$ )
  - Active user  $u$  (user interacting with the system)
- Output
  - Prediction for all null entries of the active user  $r_{ui}$
  - Presented as “top K recommendations”
    - Ordered according to predicted suit of product to user’s taste/needs

## Collaborative Filtering

### Problem Definition (2)

- Top 10 Example

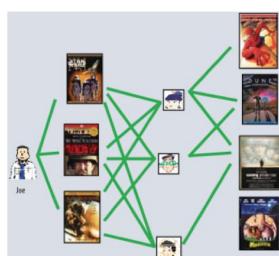


- Various elements to optimize for
  - Accuracy: many metrics possible:
    - Mean Average Precision (MAP), Fraction of Concordant Pairs (FCP), etc.
    - Need to relate to customer facing metric – e.g., CTR, retention
  - Diversity: Need to appeal to a range of interests and moods
  - Explainability: promotes trust and participation
  - Freshness, etc.

## Collaborative Filtering

### Overview

- Principle of *like mindedness*
  - Users similar in their historical preferences are likely to share similar tastes in the future
- Approach #1: **Neighborhood Models**
  - Require *similarity measure* between items (or users)
  - $r_{ui}$  is based on the  $k$  items rated by  $u$  that are most similar to  $i$



best at detecting strong associations among a small set of closely related items (i.e., localized relationships)

## Collaborative Filtering

### Overview (2)

- Approach #2: **Latent Factor Models**
  - Based on lower-rank decomposition of ratings matrix (*latent space*)
  - Matrix factorization:
$$r_{ui} = \sum_f userFeature[f][u] \times itemFeature[f][i]$$
  - Key Intuition : model item attributes as belonging to a set of unobserved topics and user preferences across these topics
    - E.g.,  $f$  = degree of action
  - Automatically inferred features ( $f$ 's) can be un-interpretable
  - Best at estimating overall structure that relates simultaneously to most or all items
    - But poor at detecting strong associations among a small set of related movies...

## Baseline Estimates

- Need to account for large user and item “effects” – i.e., systemic tendencies
  - Some users give higher ratings than others
  - Some items receive higher ratings than others

⇒ Encapsulate these effects within **baseline estimates**

$$b_{ui} = \mu + b_u + b_i$$

- $\mu$  : overall average rating
- $b_u$  : deviation of user  $u$  from the average
- $b_i$  : deviation of item  $i$  from the average

## Baseline Estimates Example

- Need  $b_{\text{Joe, Titanic}}$
- Assume
  - $\mu = 3.7$
  - $b_{\text{Titanic}} = 0.5$  -- i.e., Titanic tends to be rated higher
  - $b_{\text{Joe}} = -0.3$  -- i.e., Joe is a critical user
- Then  $b_{\text{Joe, Titanic}} = 3.7 - 0.3 + 0.5 = 3.9$
- Estimated by solving

$$\min_{b_*} \sum_{\substack{(u,i) \text{ s.t } r_{ui} \\ \text{is known}}} (r_{ui} - \mu + b_u + b_i)^2 + \lambda \left( \sum_u b_u^2 + \sum_i b_i^2 \right)$$

## Neighborhood Models

### User-oriented Illustration

	Item A	Item B	Item C	Item D	Item E
User 1	?	3	4	5	3
User 2	1	3	4	5	?

High similarity (*user1, user2*)

- If two users rate a subset of items similarly, then they might rate other items similarly as well

## Neighborhood Models

### Item-oriented Illustration

	Titanic	Batman	Inception	Super Man	Spider Man	Matrix
User 1	2.5	?	3	3.5	2.5	3
User 2	3	3.5	?	5	3	3.5
User 3	2.5	3	?	3.5	?	4
User 4	?	3.5	3	4	2.5	?
User 5	?	4	2	3	2	3
User 6	3	4	?	5	3.5	3
User 7	?	4.5	?	4	1	?

High similarity (*item4, item5*)



## Neighborhood Models

### User- vs. Item-oriented

- User-oriented

```

for every item  $i$  that  $u$  has no preference for yet
    for every other user  $v$  that has a preference for  $i$ 
         $s \leftarrow \text{similarity}(u, v)$ 
         $r_{u,i} \leftarrow r_{u,i} + s \times r_{v,i}$ 
    return the top items, ranked by weighted average

```

- Item-oriented

```

for every item  $i$  that  $u$  has no preference for yet
    for every item  $j$  that  $u$  has a preference for
         $s \leftarrow \text{similarity}(i, j)$ 
         $r_{u,i} \leftarrow r_{u,i} + s \times r_{u,j}$ 
    return the top items, ranked by weighted average

```

- Better scalability
- More amenable to explaining reasoning behind predictions

## Neighborhood Models

### K Neighbors – User-specific weights

- $S^k(i, u)$  : set of  $k$  items rated by  $u$  which are most similar to  $i$
- Predicted (residual) rating: weighted average of the ratings in  $S^k$ :

$$\begin{aligned}\hat{r}_{ui} &= b_{ui} + \frac{\sum_{j \in S^k(i, u)} s_{ij} (r_{uj} - b_{uj})}{\sum_{j \in S^k(i, u)} s_{ij}} \\ &= b_{ui} + \sum_{j \in S^k(i, u)} \theta_{ij}^u \cdot (r_{uj} - b_{uj})\end{aligned}$$

– User-specific weights  $\theta_{ij}^u = s_{ij} / \sum_{j \in S^k(i, u)} s_{ij}$

- |                       |                       |
|-----------------------|-----------------------|
| • Pros                | • Cons                |
| – Intuitive           | - Forced neighborhood |
| – simple to implement |                       |

## Neighborhood Models

### Global weights

- Weights independent of a specific user

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij}$$

- $R_u$  : set of all items rated by  $u$
- $w_{ij}$  : global relationship between items  $i$  and  $j$ 
  - $w_{ij} \approx 0$  when item  $j$  is not known to be predictive on  $i$
  - $w_{ij} >> 0$  when items  $i$  and  $j$  are related
- $w_{ij}$  learnt from data through optimization

- Generalization:  $\hat{r}_{ui} = \tilde{b}_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij}$

## Neighborhood Models

### Incorporating Implicit Feedback

- Indication of user preference without explicit rating  
E.g., rental and purchase history  $\Rightarrow$  Binary  $|U| \times |I|$  matrix
- $N(u)$ : set of items for which  $u$  provided an implicit preference
- Augmented neighborhood model:

$$\hat{r}_{ui} = \hat{b}_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} + \sum_{j \in N(u)} c_{ij}$$

- An implicit preference by  $u$  to  $j$  leads to a change in  $\hat{r}_{ui}$  by  $c_{ij}$

## Neighborhood Models

### Global Optimization

- Regularized Least-squares problem

$$\min_{b_u, w_{ij}, c_{ij}} \sum_{\substack{(u,i) \text{ s.t } r_{ui} \\ \text{is known}}} \left( r_{ui} - \mu - b_u - b_i - \frac{1}{\sqrt{R(u)}} \cdot \sum_{j \in R(u)} (r_{uj} - b_{uj}) \cdot w_{ij} - \frac{1}{\sqrt{N(u)}} \cdot \sum_{j \in N(u)} c_{ij} \right)^2 + \lambda \left( b_u^2 + b_i^2 + \sum_{j \in R(u)} w_{ij}^2 + \sum_{j \in N(u)} c_{ij}^2 \right)$$

–  $\frac{1}{\sqrt{R(u)}}$  and  $\frac{1}{\sqrt{N(u)}}$  moderate the dichotomy between heavy raters

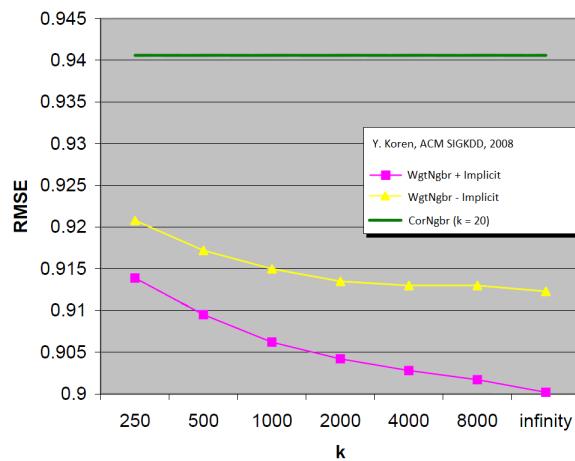
and those that rarely rate

– Simple gradient descend solver works well

– Parameter pruning:  $R(u) \rightarrow R^k(i; u) \stackrel{\text{def}}{=} R(u) \cap S^k(i)$ ; similarly for  $N(u)$

## Neighborhood Models

### Netflix Data

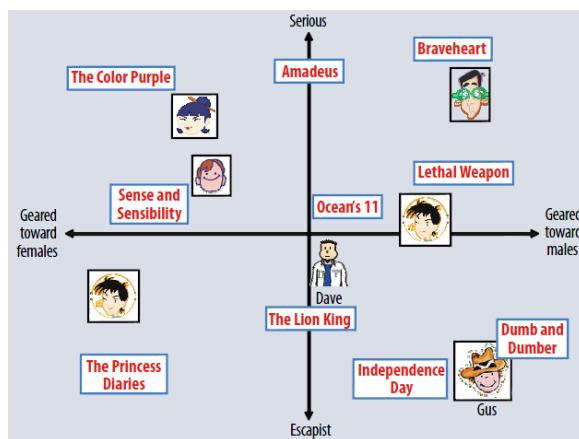


- Published Netflix's Cinematch RMSE: 0.9514

## Latent Factor Models

### Illustration

- Characterizes both users and items using factors (axes) inferred from the ratings data



## Latent Factor Models

### SVD-Like Matrix Factorization

- Approximate ratings matrix

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}' \quad \text{where } \mathbf{P} \text{ is } |\mathcal{U}| \times f \text{ matrix}$$

$$\mathbf{Q} \text{ is } |\mathcal{I}| \times f \text{ matrix}$$

$f$  is latent factor space dimensionality

- Each user  $u$  is associated with a vector  $\mathbf{p}_u \in \mathcal{R}^f$
- Each item  $i$  is associated with a vector  $\mathbf{q}_i \in \mathcal{R}^f$
- E.g.,  $f$  is *<Degree of seriousness, Geared toward males>*

$$\begin{aligned} \mathbf{p}_{\text{Gus}} &= <-5, 5> \\ \mathbf{q}_{\text{Dumb\_and\_Dumber}} &= <-10, 10> \Rightarrow \hat{r}_{\text{Gus, D\_and\_D}} = [-10 \quad 10] \cdot \begin{bmatrix} -5 \\ 5 \end{bmatrix} \end{aligned}$$

- Conventional SVD is undefined given the high NA rate

## Latent Factor Models

### Global Optimization

- Regularized Least-squares problem

$$\min_{q^*, p^*} \sum_{\substack{(u,i) \text{ s.t } r_{ui} \\ \text{is known}}} (r_{ui} - \mathbf{q}_i' \mathbf{p}_u)^2 + \lambda (\|\mathbf{q}_i\|^2 + \|\mathbf{p}_u\|^2)$$

- Stochastic Gradient Descent:

```

Initialize { $f, \mathbf{P}, \mathbf{Q}, \theta, \eta, \lambda$ }
do {
     $r_{ui} \leftarrow$  randomly chosen rating
     $e_{ui} \leftarrow r_{ui} - \mathbf{q}_i' \mathbf{p}_u$ 
     $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e_{ui} \cdot \mathbf{p}_u - \lambda \cdot \mathbf{q}_i)$ 
     $\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{p}_u)$ 
} until ( $\|e\| < \theta$ )

```

- Method of Alternating Least Squares also possible

## Latent Factor Models

### Incorporating Implicit Feedback

- Indication of user preference without explicit rating

E.g., rental and purchase history  $\Rightarrow$  Binary  $|U| \times |I|$  matrix

- $N(u)$  : set of items for which  $u$  provided an implicit preference

- Augmented model (**SVD++**):

$$\hat{r}_{ui} = \hat{b}_{ui} + \mathbf{q}_i^t \cdot \left( \mathbf{p}_u + \frac{1}{\sqrt{N(u)}} \sum_{j \in N(u)} \mathbf{y}_j \right) \quad \mathbf{y}_j \in \mathfrak{R}^f$$

- Item  $i$  is associated with two factor vectors:  $\mathbf{q}_i$  and  $\mathbf{y}_i$
- Fewer parameters

## Integrated Model

### Neighborhood & Factor Models Enrich Each Other

- Combined model:

$$\begin{aligned} \hat{r}_{ui} = & \boxed{\mu + b_u + b_i + \mathbf{q}_i^t \cdot \left( \mathbf{p}_u + \frac{1}{\sqrt{N(u)}} \cdot \sum_{j \in N(u)} \mathbf{y}_j \right)} \\ & + \boxed{\frac{1}{\sqrt{R^k(i; u)}} \cdot \sum_{j \in R^k(i; u)} (r_{uj} - b_{uj}) \cdot w_{ij}} + \boxed{\frac{1}{\sqrt{N^k(i; u)}} \cdot \sum_{j \in N^k(i; u)} c_{ij}} \end{aligned}$$

- Describes general properties of the item and the user
- Provides an interaction between the user profile and item profile
- Contributes fine-grained adjustments that are hard to profile

¶: Increasing  $k$  has no benefit as adding neighbors covers more global information, already captured by LF model

## Integrated Model

### Netflix Data

- Performance (RMSE):

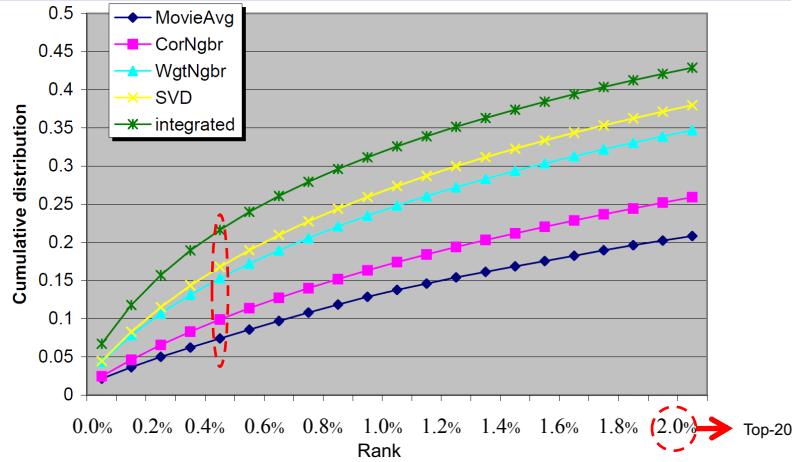
Model	50 factors	100 factors	200 factors
SVD++	0.8952	0.8924	0.8911
Integrated	0.8877	0.8870	0.8868

- Naïve ( $r_{ui}$  is mean rating of movie  $i$ ): 1.053
- Netflix's Cinematch: 0.9514
  - \$1M Prize competition: -10%
  - Integrated model: -7%
    - » Blended with several other solutions: 0.8645 (-9.1%, KorBell Team, Bell Labs)
- What effect on user experience should we expect by lowering the RMSE by 10%?

## Evaluation Through Top-K Recommendations

- Effect of lowering the RMSE on the quality of top K recommendations (user experience proxy)
- Goal: find relative place of 5-star rated movies
  - Proxy for movies that truly interest users
- K=1000
  - Randomly selected for each 5-star movie  $\langle u, i \rangle$
  - Order the 1001 movies based on their predicted rating  $\hat{r}_{ui}$
- Rank
  - 0%: none of the random movies appears before  $i$
  - 100%: all of the random movies appear before  $i$

## Evaluation Through Top-K Recommendations (2)



- Integrated method has a >20% prob of placing a 5-star movie in the top-5
  - 5x better than MovieAvg; 2x better than CorNgbr

## Summary

- The neighborhood and latent factor models address the data at different levels and complement each other
- Addressing different aspects of the data leads to improved recommendation quality – E.g., implicit user feedback
- Quality of a recommender system is expressed through multiple dimensions including: accuracy, top-K, diversity, ability to surprise, explainability, computational efficiency
  - But ultimately trust business metrics:
    - Member engagement
    - Retention