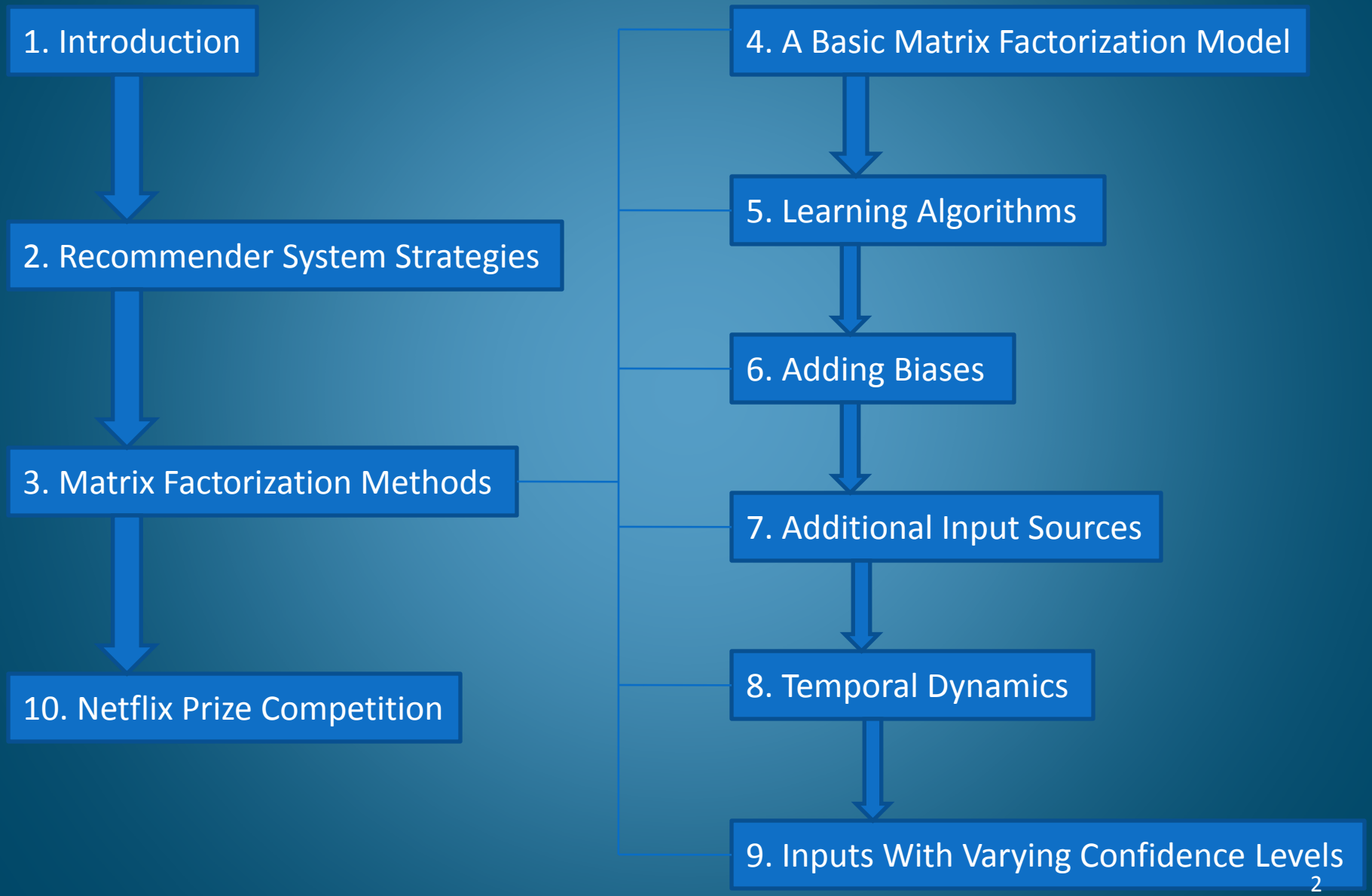
Matrix Factorization Techniques for Recommender Systems

By Yehuda Koren
Robert Bell
Chris Volinsky

Presented by Peng Xu
Supervised by Prof. Michel Desmarais

Contents


1. Introduction
2. Recommender System Strategies
3. Matrix Factorization Methods
10. Netflix Prize Competition
4. A Basic Matrix Factorization Model
5. Learning Algorithms
6. Adding Biases
7. Additional Input Sources
8. Temporal Dynamics
9. Inputs With Varying Confidence Levels

1. Introduction

- **Why recommender system? (for corporations)**
  - **Modern consumers are inundated with choices.**
  - **Key to enhance user satisfaction and loyalty**
  - **Particularly useful for entertainment products because of the available huge volume of data**

2. Recommender System Strategies

Content Filtering	**Collaborative Filtering**
Based on the ***profile***!	Based on the ***interactions***!
Main problem:	Main problem:
External information needed!	**Cold Start!**
Example: *Music Genome Project*	Example: *Netflix*

Collaborative Filtering

Major appeal:

- It is domain free!
- It can address data aspects that are often elusive and difficult to profile
- Generally more accurate than content-based techniques

Two areas of collaborative filtering:

- Neighborhood methods
- Latent factor models

Neighborhood Methods

This method centers on computing the relationships between items or between users.

- Item-oriented approach
- User-oriented approach (Illustrated by figure 1)

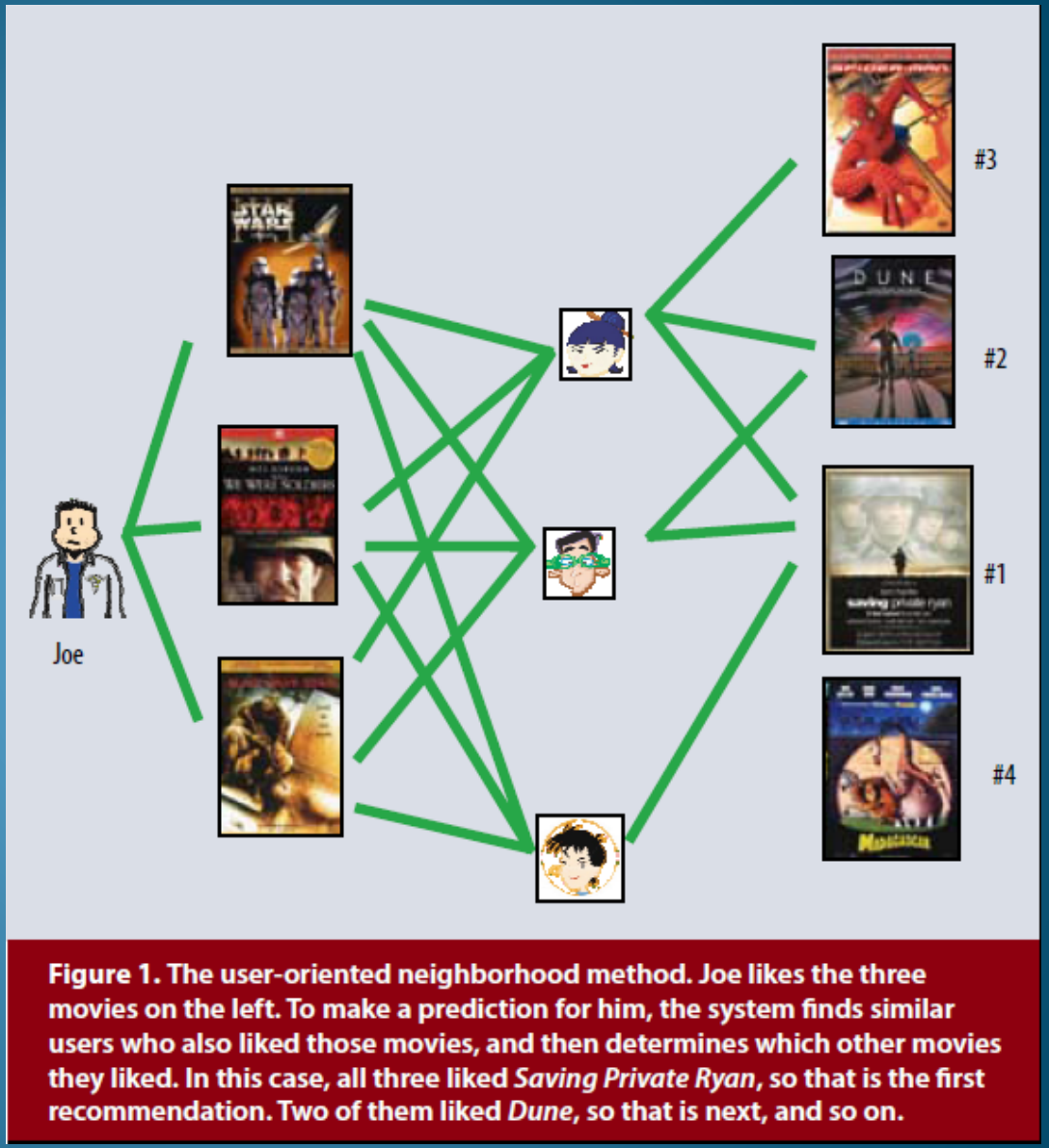


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

Latent factor models

This method tries to explain the user's ratings by characterizing both items and users on some factors inferred from the rating patterns.

That means you **don't** need to know the factors beforehand.

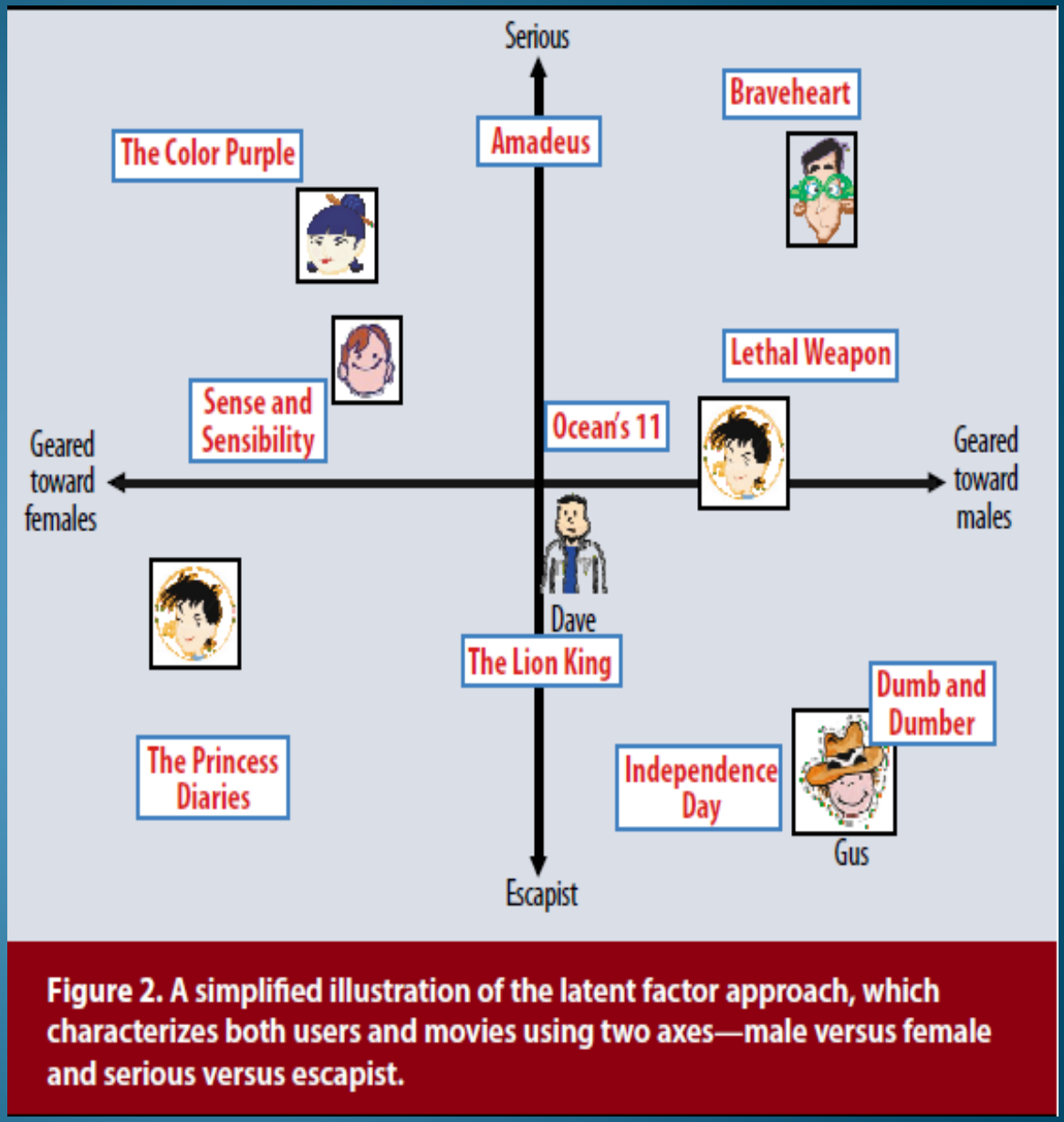


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

3. Matrix Factorization Methods

Q: How to realize latent factor models?

A: Matrix factorization

Q: Why it becomes popular?

A: 1. Good scalability

2. Accurate predication

3. Flexibility

4. A Basic Matrix Factorization Model

- Idea: $q_i \in \mathbb{R}^f$ represents the item i
  $p_u \in \mathbb{R}^f$ represents the user u
  and the inner product $q_i^T p_u$ would capture the interaction between user u and item i
- Thus user u 's rating of item i , which is denoted by r_{ui} , can be estimated by

$$\hat{r}_{ui} = q_i^T p_u$$

A Basic Matrix Factorization Model

- First, we would naturally think about the SVD (*Singular Value Decomposition*) method, a well-established technique for identifying latent factors.

- However, there comes two problems:
  1. Sparseness of the data
  2. Overfitting

A Basic Matrix Factorization Model

- Therefore, we would model directly the observed ratings only, while avoiding overfitting through a regularized model. That is: to learn the factor vectors(p_u and q_i), the system minimizes the regularized squared error on the set of known ratings:

$$\min_{q^*, p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

Here, κ is the set of the (u,i) pair for which r_{ui} is known (the training set).

5. Learning Algorithms

There are two approaches to minimizing the object function.

1. Stochastic gradient descent
   - Implementation ease
   - Relatively fast running time
2. Alternating least squares (ALS)
   - When the system can use parallelization
   - When system is centered on implicit data

6. Adding Biases

- Problems:

1. Some users used to rate high while some used to rate low.
2. Some products are widely perceived as better than others.

- Therefore, it would be unwise to explain the full rating value by an interaction of just the form $q_i^T p_u$

Adding Biases

- So, let's consider a first-order approximation of the bias involved in rating

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

- The overall average rating is denoted by μ;the parameters b_u and b_i indicate the observed deviations of user u and item i , respectively.

Adding Biases

- So, the estimation becomes

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

- And the object function becomes

$$\min_{p^*, q^*, b^*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

7. Additional Input Sources

- $N(u)$ denotes the set of items for which user expressed an implicit preference and item i is associated with $x_i \in \mathbb{R}^f$. Accordingly, a user who showed a preference for items in $N(u)$ is characterized by the vector $\sum_{i \in N(u)} x_i$

Normalize it we get $|N(u)|^{-0.5} \sum_{i \in N(u)} x_i$

Additional Input Sources

- Similarly, we use $A(u)$ and $y_a \in \mathbb{R}^f$ to model another type of information knowned as user attributes. $A(u)$ denotes the set of attributes of user u and y_a indicates the reaction because of the attribute a. Thus the user u's response to all his attributes is denoted by $\sum_{a \in A(u)} y_a$
- And thus our estimation becomes

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T [p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a]$$

8. Temporal Dynamics

- In reality, product perception and popularity, customers' inclinations may change thus the system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions.

$$\hat{r}_{ui} = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

9. Inputs With Varying Confidence Levels

- The matrix factorization model can readily accept varying confidence levels. If confidence in observing r_{ui} is denoted as c_{ui}, then the model becomes as follows:

$$\min_{p^*,q^*,b^*} \sum_{(u,i)\in\kappa} c_{ui}(r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2$$

$$+\lambda(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

10. Netflix Prize Competition

- Started in 2006, ended in 2009
- 1M prize for the team which has improved the algorithm's RMSE(Root Mean Square Error) performance by 10%
- Authors of this paper are the members of the final winner

Netflix Prize Competition

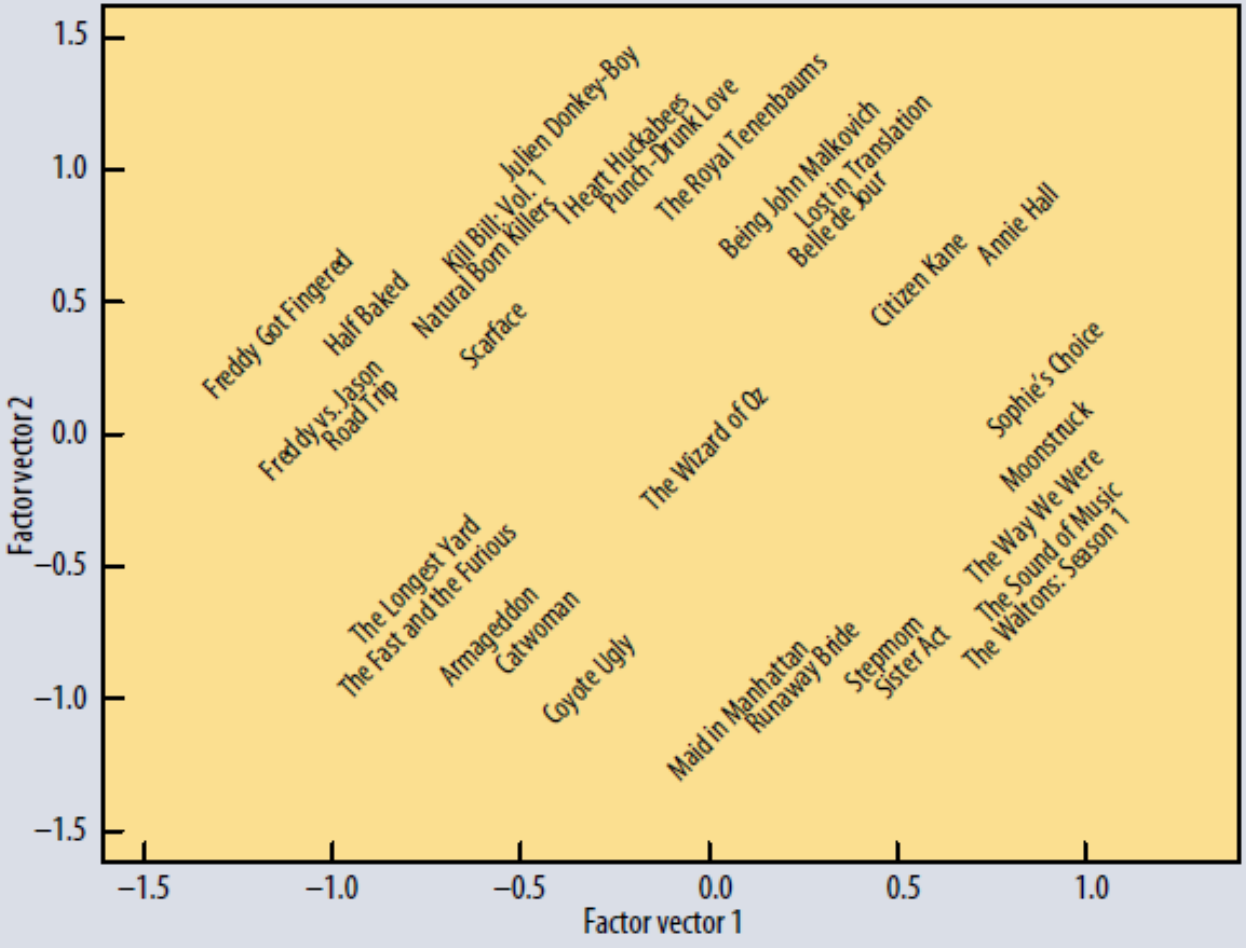


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky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

Netflix Prize Competition

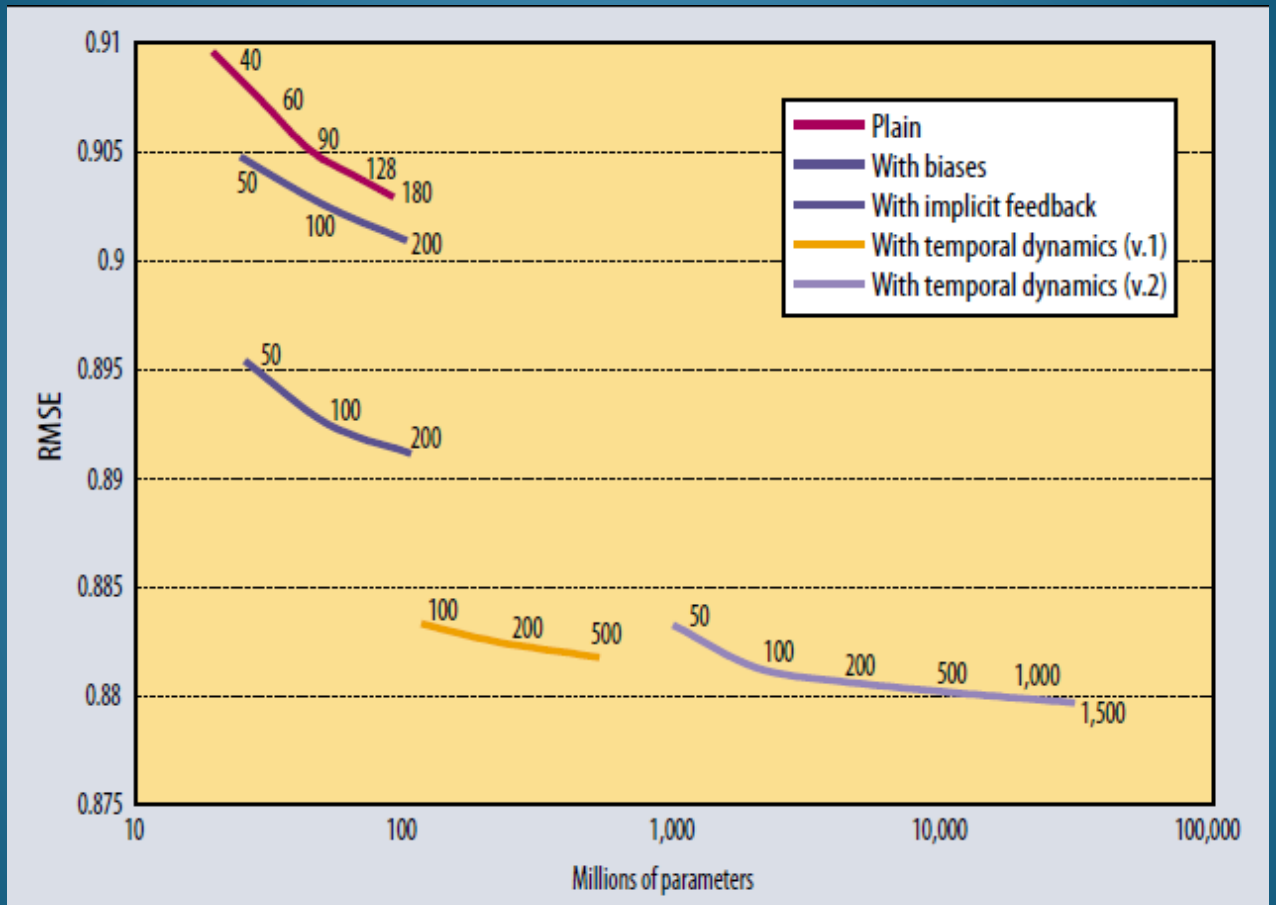


Figure 4. Matrix factorization models' accuracy. The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

Summary

- Matrix factorization techniques have become a dominant methodology within collaborative filtering recommenders.

- Experience with datasets such as the Netflix Prize data has shown that they deliver accuracy superior to classical nearest-neighbor techniques.

Summary

- At the same time, they offer a compact memory-efficient model that systems can learn relatively easily.

- What makes these techniques even more convenient is that models can integrate naturally many crucial aspects of the data, such as multiple forms of feedback, temporal dynamics, and confidence levels.

Any questions?

Thank you!