Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions

Gediminas Adomavicius

Member – IEEE

Alexander Tuzhilin

Member – IEEE

Published in IEEE, 2005

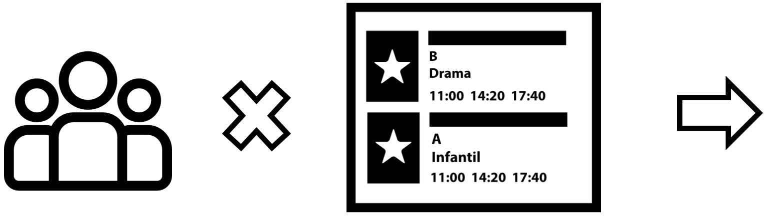
2025
HoonUi Lee

Contents

What is a Recommendation System?

- **How to predict and suggest items that users will like**
  - In daily life, there are many types of items to recommend
    - Movie, Shopping item, Book, …
  - Influenced by concepts from various academic fields
    - Approximation, Forecasting theory, Information retrieval, …
  - Focus on recommendation problems rely on rating structure after mid-1990s
    - How to estimate unseen item ratings for a user?
    - Recommend item with the highest estimated rating

Estimating Unseen Utility



Set of Users — C

Set of Items — S

TABLE 1
A Fragment of a Rating Matrix for a Movie Recommender System

S

C	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	∅	4	5	5
Cindy	2	2	4	∅
David	3	∅	5	2

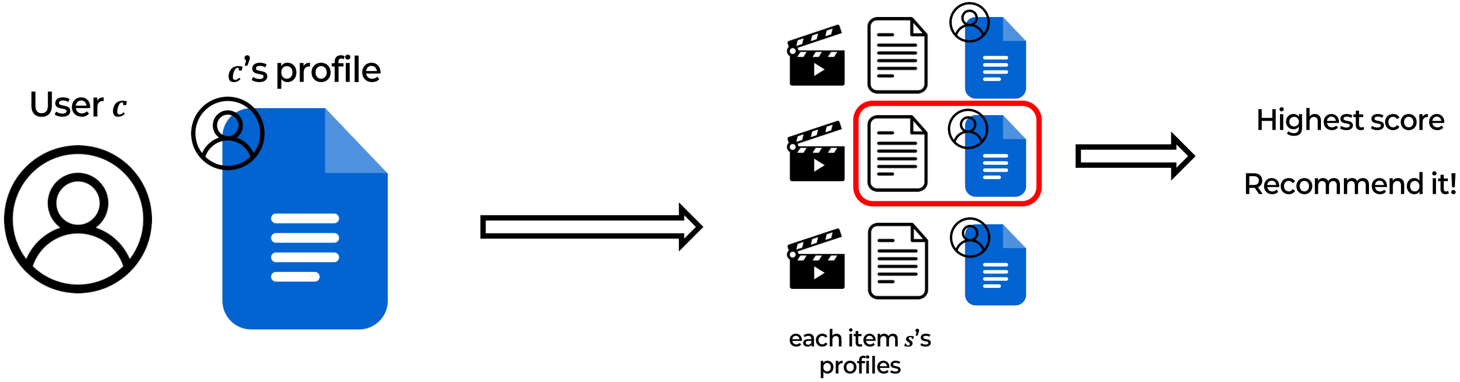
utility

- **Utility, the usefulness of an item for a user**
  - Defined through a utility function
  - Usually represented by a ratings
  - Utility is shaped by how the function is defined
    - Scroll depth, Purchased, Shared, ...

Category of Recommendation

- **By the source of information used for recommendation**
  - Content-based Approach
    - Using inherent features of users and items
  - Collaborative Approach
    - Using user-item interactions from other users
      - “collaborative”
  - Hybrid Approach
    - Combine collaborative & content-based methods

Content-based Approach



- ❑ Selecting items to recommend using the profile
- ❑ Computing similarity score between user and item profile
    - Recommend items with high scores
    - Similarity-based utility

Profile Construction

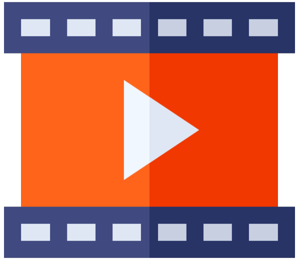
- Item's profile

Text-based features

(Year of release, Genre, Director, ...)

Image features

(Image embedding, Object, Style, ...)



Video features

(Keyframe's Embedding, Object, ...)

Audio features

(Embedding, Lyric, Metadata, ...)

Profile Construction

- User's profile

Age

Gender

Income

Selected Items' Profile

Text Description Example

- **Constructing the profile using item's text descriptions**
  - Item profile $Content(s)$
    - Represented by the weight of each $keyword$ in item
    - Computing each $keyword's\ weight$ with **TF-IDF**
      - High frequency within the item, low frequency across items
  - User Profile $ContentBasedProfile(c)$
    - Profile construction using only user-selected items
      1. Averaging items' keyword weight
      2. Strengthen or weaken specific keyword weights by online updating

Heuristic Approach

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{||\vec{w}_c||_2 \times ||\vec{w}_s||_2}$$

$$= \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2}\sqrt{\sum_{i=1}^{K} w_{i,s}^2}},$$

$\vec{w_c}, \vec{w_s}$: user, item's profile vector

K : the number of keywords

❖ Similarity Score Between Profiles

- ❑ Using cosine similarity between user and item keyword weight vectors
  - Recommending items with high similarity
- ❑ Heuristic utility function based on similarity score
  - Assumption that users prefer items with similar profiles

Model-based Approach

❖ **Bayesian classifier (e.g.)**

$$P(C_i|k_{1,j}\&\ldots\&k_{n,j}) \Longrightarrow P(C_i)\prod_x P(k_{x,j}|C_i)$$

C_i : Relevant / Irrelevant class (Supposed)

$k_{n,j}$: jth item's n'th keyword

- Estimating the "Relevant" probability of an item to a user given its keywords
  - Based on keyword frequency in items the user liked
  - Utility as probability

Heuristic-based vs. Model-based

Heuristic-based (Using similarity)

- Recommendation criteria defined by human intuition and knowledge
- Used for output validation rather than guiding the recommendation criteria

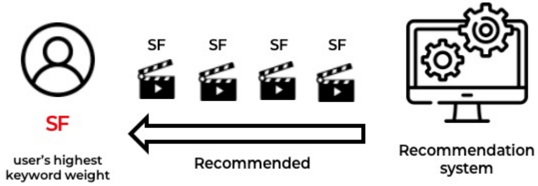
Model-based

- Recommendation criteria are learned from data-driven rules
  - Discovering rules that which conditions an item is likely to be recommended
- non-heuristic

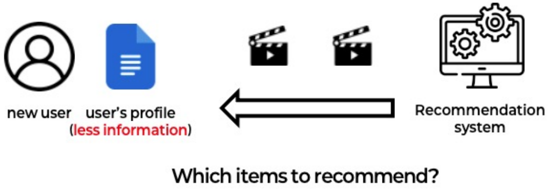
Limitations of Content-based

- **Limited Content Analysis**
  - Non-uniqueness of items with the same feature vector
    - Need for richer embeddings or additional discriminative information

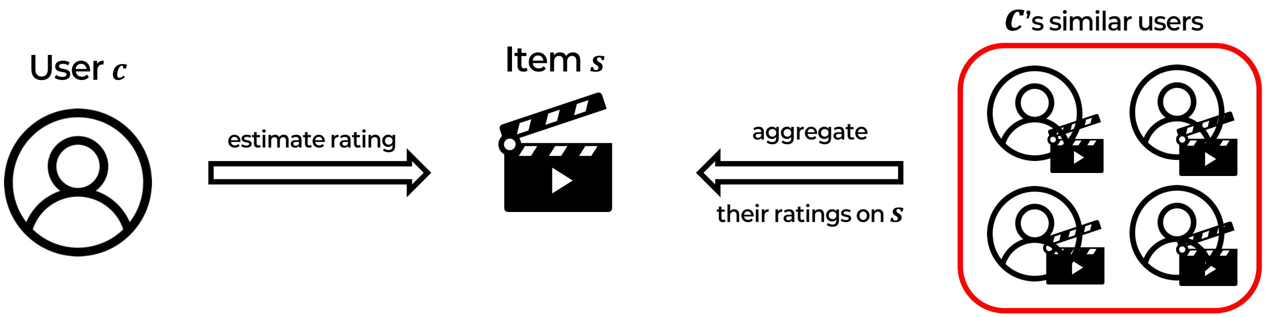
- **Overspecialization**
  - Trivial recommendation, Diversity needed
    - Need randomness, avoiding duplicates



- **New user problem**
  - Cold-start problem from few user ratings



Collaborative Methods



- **Based on User-Item interactions**
  - How did other users rate the items?
  - How were other items rated by users?

Memory-based Approach

- **Find similar users with rating**
  - User similarity based on co-rated items
    - similarity used as heuristic artifact
      - pearson correlation coefficient & cosine similarity

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}}$$

pearson correlation coefficient

$$sim(x,y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||_2 \times ||\vec{y}||_2} = \frac{\sum_{s \in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}}$$

cosine similarity

- **Aggregating ratings from similar users**
  - Various aggregation functions can be used
    - Simple average (a), Weighted by similarity (b), Adjusted for user bias (c)
  - Estimated rating obtained through aggregation

$$\text{(a)}\ r_{c,s} = \frac{1}{N} \sum_{c' \in \hat{C}} r_{c',s},$$

$$\text{(b)}\ r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \times r_{c',s},$$

$$\text{(c)}\ r_{c,s} = \bar{r}_c + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r}_{c'})$$

Some examples of
aggregation function

Item-based Approach

- **Recommend based on item similarity, not user similarity**
  - Item similarity based on co-rating users
    - Similar ratings from other users → high similarity

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{u,j} - \bar{r}_j)^2}}$$

pearson correlation coefficient

- **Why the item-based is better?**
  - Large and dynamic user base
    - Frequent model recalculation
  - Sparsity robustness
    - More rating data available for items than for users

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U_{ij}} r_{u,i}^2} \sqrt{\sum_{u \in U_{ij}} r_{u,j}^2}}$$

cosine similarity

Model-based Approach

❖ **Criterion based on rating patterns of other users**

Estimate this probability

$$r_{c,s} = E(r_{c,s}) = \sum_{i=0}^{n} i \times \boxed{\Pr(r_{c,s} = i | r_{c,s'}, s' \in S_c)}$$

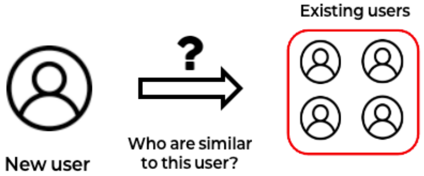
- User's past ratings used as features to identify similar users

- Clustering model
  - Classifying the user into a cluster and use that cluster's rating distribution
- Bayesian network
  - Estimate probabilities from past ratings via other users' distributions

Limitations of Collaborative Approach

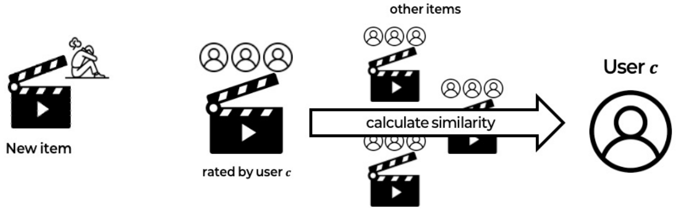
❖ New user problem

- ❑ Lack of user-provided ratings to learn preferences
  - ▸ Fast preference discovery strategy, Content-based information needed



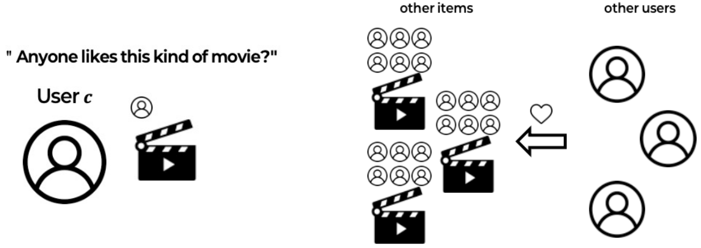
❖ New item problem

- ❑ Cold-start issue for unrated items



❖ Sparsity

- ❑ Users with niche preferences and niche items



Hybrid Approach

- **Combining results from separately implemented CB and CF**
  - Linear combination of ratings
  - Voting
  - Choices based on quality metric
    - High confidence, Better aligned with the user's past ratings

- **Single unifying recommendation model**
  - Rule-based classifier with user, item features
  - Statistical model with user, item parameters

Hybrid Approach

❖ Adding Content to Collaborative

- Using profile for calculating user similarity
- Content-based rating imputation
  - Other model's output / Filterbot

❖ Adding Collaborative to Content

- Dimensionality reduction on the set of profiles
  - Discovering hidden patterns in user profile space

Knowledge-based Recommendation System

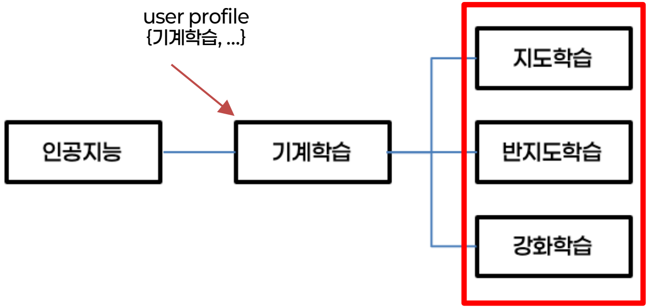
❖ Recommendation using domain knowledge

- Constraint-based filtering with external knowledge
  - Preference for child-friendly content
    - Rated-R = not suitable for watching with children
    - Animations are generally family-friendly

Movie.rated == Rated-R -> except

Movie.genre == ‘Animation’ -> score++

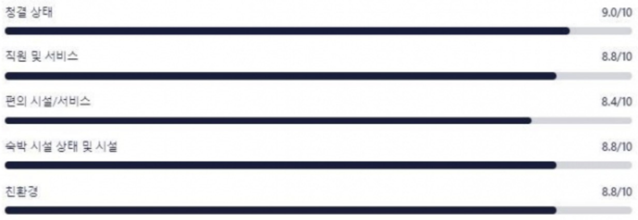
- Ontology-based recommendation expansion
  - Assuming the user is interested in “machine learning



Extending Capabilities of Recommender Systems

- **Integration of contextual information**
  - Utilizing contextual factors such as time, location, and companions

- **Multi-criteria rating**
  - When ratings involve multiple aspects
    - e.g., hotel rated by location, service, cleanliness, breakfast, etc.
  - Using Pareto optimal solutions, consecutive optimizing, ...



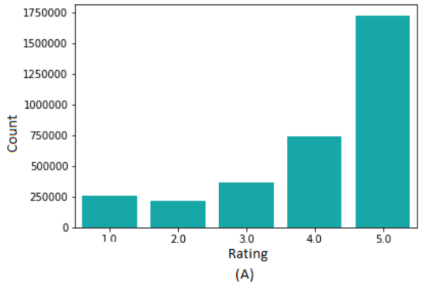
- **Nonintrusiveness**
  - Less intrusive collection of user's explicit feedback
  - Strategic incorporation of implicit feedback

Commonly Used Metrics

- **Coverage**
  - Coverage of the item space in prediction
  - Limited scope reduces recommendation diversity
    - Popularity bias in recommendation
- **Accuracy**
  - Statistical : Deviation between predicted and true ratings
    - RMSE, MAE
  - Decision-support : Usefulness of recommendations in real decision-making
    - Precision, Recall, F1 score

Limitations of Conventional Metrics

- **Training on positive sample, Evaluation with positive sample**
  - Biased observed ratings toward user preferences
  - Sparse feedback on not prefer
  - ➔ Is this model truly intelligent?



- **Measuring "Usefulness", "Quality"**
  - Is it always a good thing to recommend only items the user is certain to like?
  - Is a model that simply recommends popular items necessarily a good one?

Maria Stratigi (2019), Ratings vs. Reviews in Recommender Systems: A Case Study on the Amazon Movies Dataset

Conclusion

- **Recommendation is how to predict and suggest items that users will like**
  - Estimate utility

- **Content-based Approach**
  - Using profiles based on the intrinsic features of the User and Item

- **Collaborative Approach**
  - Using interactions with items from other users

- **The combination of the two approaches can also be considered**
- **We've also looked at some of the Expansion Capabilities**

Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model

Yehuda Koren

AT&T Labs – Research

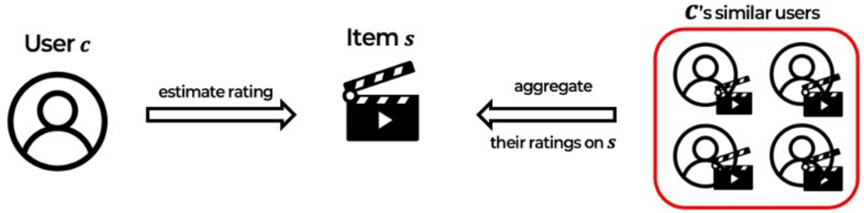
Published in KDD, 2008

2025

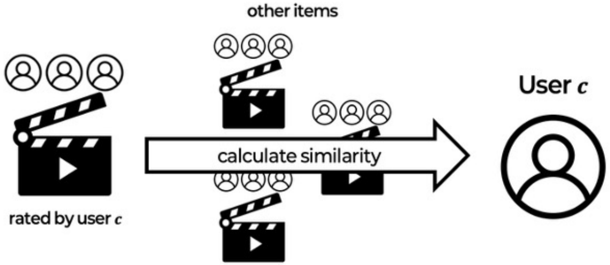
HoonUi Lee

Contents

Collaborative Approach



User-based



Item-based

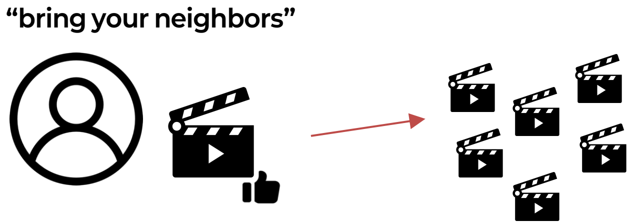
❖ Recommendation Based on User-Item interactions

- ❑ No use of explicit user profiles
  - No need for domain knowledge for profiles
  - No data collection required for profile construction

- ❑ Able to discover patterns not explainable by user/item attributes alone

Neighborhood method

- **Focus on computing relationships between items or users**
  - In item-based filtering…
    - Recommending items most similar to those rated by the user
    - Similarity based on **co-rating users**
  - Effective for detecting **local relationships**
    - Relies on a **few important neighbor**
    - Ignore most of the users' ratings
      - Fails to capture weak signals in most ratings
      - A signal of 3 out of 5



Why are Weak Signals Important?

❖ Weak signals as valid indicators of user preference

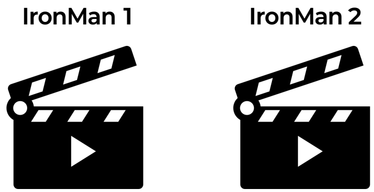
- Some users **rarely** express **strong reactions**
  - Frequent moderate ratings (e.g., 3 or 4 stars)
    - Mild positive feedback, such as 'not bad’
  - Identifying latent interests not strongly expressed

- Low diversity when recommending based only on top-rated items
  - To expand the user’s preference space

Latent factor model (SVD)

❖ Embedding User and Item into latent factor space

- ❑ Automatically discover hidden characteristics in user-item interactions
    - Comedy vs. Drama, Action scene ratio
    - Bizarreness, Tension, Peacefulness
    - Latent factors that may not be interpretable by humans



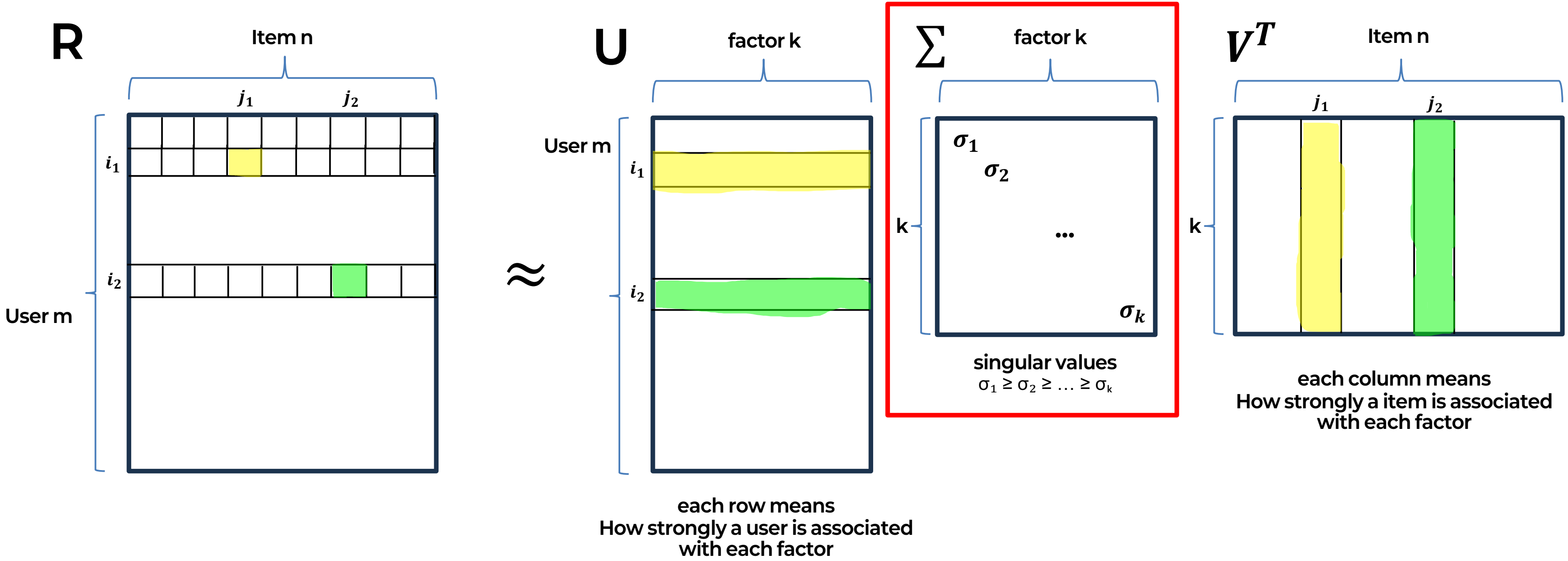
They can be similar factor

but not “highlighted”

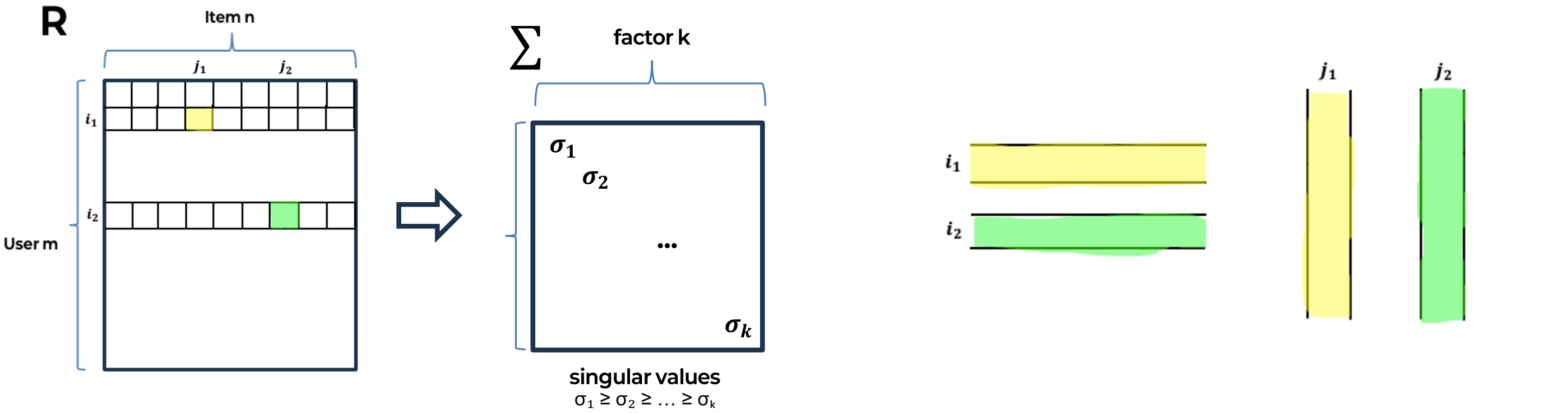
- ❑ Effective in estimating the overall structure from most or all items
    - Weak in detecting strong relationships among a small number of closely related items

Singular Value Decomposition

Approximate the entire given user-item matrix to $R \approx U\,\Sigma\,V^T$



Singular Value Decomposition



Latent factors are derived from the overall user-item interaction matrix

And each user and item representation is derived from the factors learned through overall user-item interactions

➔ Collaborative Approach!

Factorization Meets the Neighborhood

- **Each method captures different levels of structure from the data**
  - Neighborhood models capture local, explicit similarity
    - Between a small set of neighbors
  - Latent factor models uncover the hidden structure
    - From the entire user-item rating matrix

- **A model that combines both can leverage their strengths and improve accuracy**
  - Integrated model

About Implicit Feedback

- **Explicit feedback and Implicit feedback**
  - Explicit feedback includes ratings and like/dislike buttons
  - Implicit feedback includes purchase history, search logs, and click behavior
    - This paper uses rated/not rated as a form of feedback

- **The importance of integrate various forms of user input**
  - Leverage both explicit and implicit feedback
  - Fallback to implicit signals when explicit data is sparse

Overview of upcoming discussion

- **Enhancing Neighborhood and Latent Factor models**
  - With implicit feedback

- **Propose an integrated model combining both approaches**

- **Empirical evaluation and performance comparison**

Preliminary

- **Baseline Estimates for Rating Prediction**
  - Accounts for user and item bias on ratings
    - Some users tend to give higher(lower) ratings
    - Some items tend to receive higher(lower) ratings

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

- **Baseline estimate b_{ui} about unseen rating r_{ui}**
  - Overall average rating μ
  - b_u and b_i : deviation from global mean

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2)$$

Preliminary: Neighborhood models

- **Focusing on Item-based approach**
  - Recommend items similar to those already rated by the user

- **Advantages of Item-based over User-based approach**
  - Better scalability and accuracy
    - Fewer items than users
    - Item features and similarities are more stable
  - More interpretable predictions
    - Users are more familiar with their previously liked items
    - Other similar users are less relatable

Item-based approach

- **Measuring Item-to-Item Similarity**
  - Similarity based on user rating patterns
    - Pearson correlation coefficient
    - Cosine similarity

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{u,j} - \bar{r}_j)^2}}$$

pearson correlation coefficient

$$s_{ij} \stackrel{\text{def}}{=} \frac{n_{ij}}{n_{ij} + \lambda_2} \rho_{ij}$$

n_{ij}: co-rating users on item i, j

$\lambda_2 = 100$

$n_{ij} = 200$	$n_{ij} = 10$
$\frac{200}{200+100} \cdot \rho_{ij}$	$\frac{10}{10+100} \cdot \rho_{ij}$

  - Similarity score s_{ij}
  - Require sufficient number of co-raters

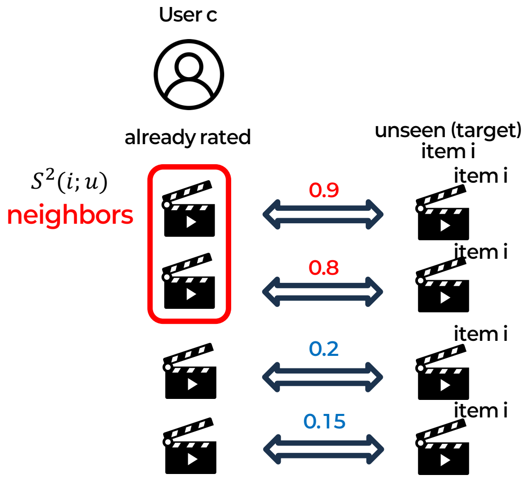
Correlation-based Neighborhood

❖ **Estimate unseen rating r_{ui} with correlation (CorNgbr)**

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}}$$

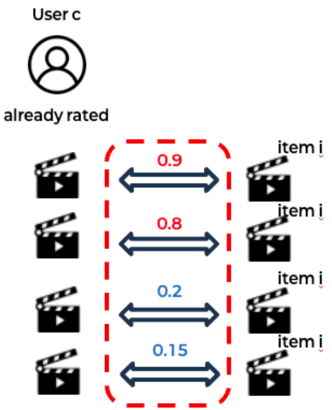
$S^k(i;u)$: Top-k items rated by user u that are most similar to the target item i

- ❑ $r_{uj} - b_{uj}$: How different is it from the usual ratings
  - Emphasizing the influence of items that user truly liked
- ❑ Weighted average using Pearson similarity

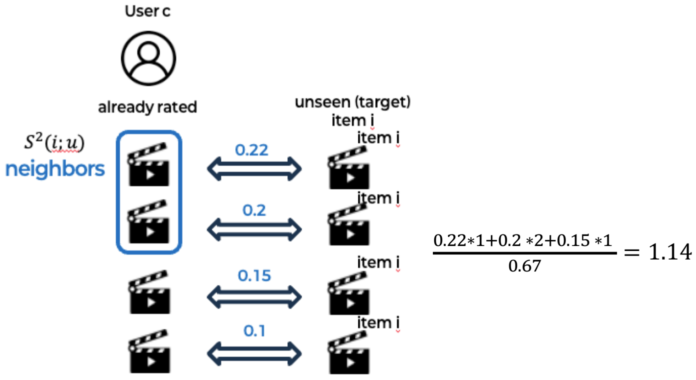


Limitations of CorNgbr

- ❖ **Independent analysis of item–item relationships**
  - ❑ No modeling of joint effects among neighbor items
    - ‣ Between neighbor items or among all items rated by the user



- ❖ **No mechanism that relies solely on the baseline**
  - ❑ No similar items to item *i* among those the user has rated
  - ❑ Low similarities still affect the prediction
    - ‣ Similarities are forced to sum to 1



Upgraded Interpolation Weight

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in S^k(i;u)} \theta_{ij}^{u} (r_{uj} - b_{uj})$$

$S^k(i;u)$: Top-k items rated by user u that are most similar to the target item i

θ_{ij}^{u} : Contribution of item j to the prediction of item i (user-specific)

- **Representing how much each neighbor *j* contributes to the prediction of target *i***
  - Previous methods force interpolation using all neighbors similarity
  - If two items are unrelated, the weight naturally becomes small
    - Fallback to baseline prediction if needed
  - Learned using the matrix of all neighbors $S^k(i;u)$

“Interpolation” to “offset”

- **Global weight neighborhood model**

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

$R(u)$: Set of all items that user has rated

- Learn a global weight $\mathbf{w_{ij}}$ shared across all users
  - w_{ij} represents how much item j helps predict item i
  - Interpreted as an offset coefficient determining how much $r_{uj} - b_{uj}$ contributes to the prediction
    - $residual \times weight \Rightarrow offset$

- **No need for compatibility between b_{ui} and b_{uj}**
  - b_{ui} can be extended to a richer representation

Using Implicit Feedback

- **User's opinion is reflected even in missing ratings**
  - Meaningful item-to-item weights are unusable without explicit ratings

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

$R(u)$: Set of all items that user has rated

$\mathrm{N}(u)$: Set of all items that user has not rated (but implicit interaction occurred)

- **Offset for implicit signal**
  - Adjusting item i's contribution using only implicit feedback
    - e.g., viewed or clicked item j

Final prediction rule

Adjust about rating count

Regularization term

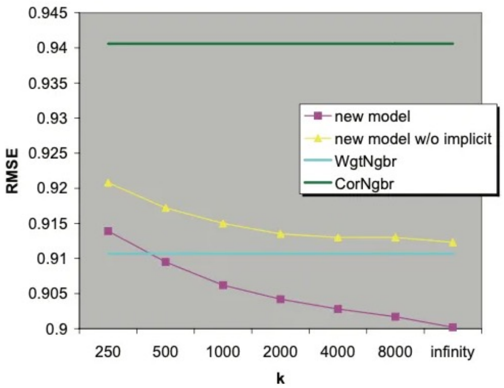
$$\min_{b_*,w_*,c_*} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - \boxed{|\mathrm{N}^k(i;u)|^{-\frac{1}{2}}} \sum_{\boxed{j\in\mathrm{N}^k(i;u)}} c_{ij} - \boxed{|\mathrm{R}^k(i;u)|^{-\frac{1}{2}}} \sum_{\boxed{j\in\mathrm{R}^k(i;u)}} (r_{uj} - b_{uj})w_{ij} \right)^2 + \boxed{\lambda_4 \left(b_u^2 + b_i^2 + \sum_{j\in\mathrm{R}^k(i;u)} w_{ij}^2 + \sum_{j\in\mathrm{N}^k(i;u)} c_{ij}^2 \right) \quad (11)}$$

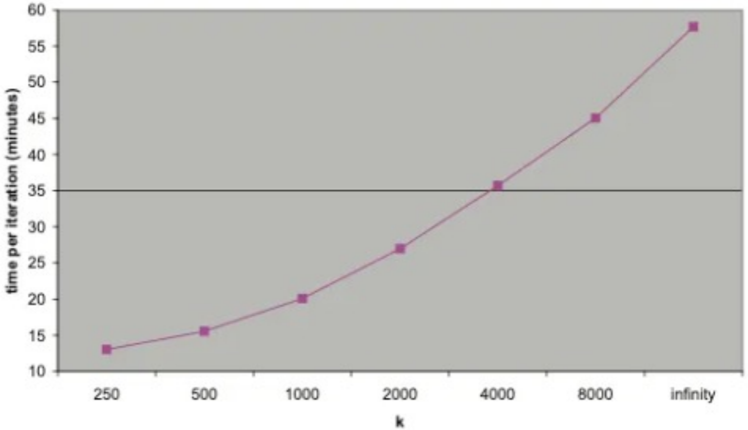
Pruning item-item relations

- **Adjusting for rating count difference between users**
  - Apply inverse square root of rating count
  - Mitigates overemphasis due to rating volume differences
- **Pruning unlikely item-item relations**
  - Ignore item $i's$ neighbors if user hasn't rated items similar to item i
- **All parameters are optimized via gradient descent**

Experiment about K

❖ **While find item i's top-K neighbors..**





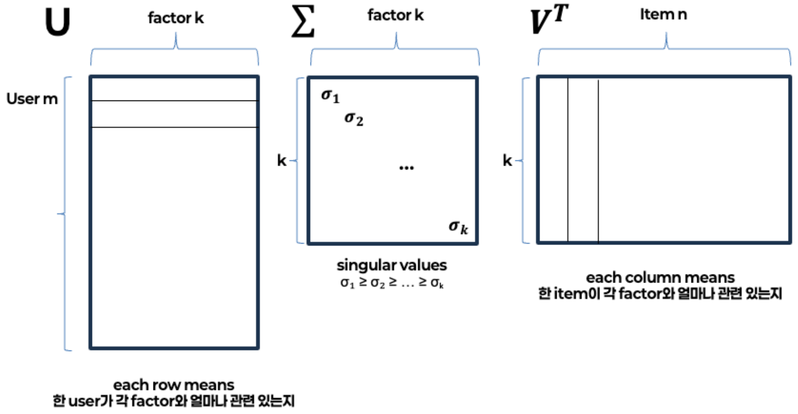
- Performance improves proportionally with larger k
  - Trade-off exists with increased running time
- Significant performance drop without implicit feedback

Preliminary: Latent Factor Models

- **Predict ratings by discovering latent factors between users and items**
  - Apply SVD to the user-item rating matrix

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

  - Represent each user and item as vectors p_u and q_i
  - Estimate preference via dot product of vectors



Regularized SVD

- **Many missing values in the user-item matrix**
  - Traditional SVD requires a fully filled matrix
  - Imputing missing values causes distortion and increases computation

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda_3 (\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

- **Optimization to minimize error over observed ratings**
  - Under a factorized SVD structure
  - Learn p_u and q_i based on observed rating pairs

NSVD model (Paterek's idea)

- **Learn a separate vector p_u for each user**
  - Overfitting occurs when user ratings are sparse

$$b_{ui} + q_i^T \boxed{\left(\sum_{j \in \mathrm{R}(u)} x_j\right) / \sqrt{|\mathrm{R}(u)|}}$$

- **Approach without learning user vectors directly**
  - Replace with average of rated item vectors
  - Represent the user based on items they rated

Asymmetric-SVD

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) x_j + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$

x_j, y_j : latent factor vector for item j

- **Apply Paterek's idea**
  - User vector p_u is explained by items and the ratings given to them
- **Include parameters for implicit feedback**
  - Even without explicit ratings, implicit interactions are reflected in training

Benefits of Asymmetric-SVD

- **Reduced number of parameters**
  - Users are much more numerous than items
  - Replace user parameters with item parameters to reduce complexity

- **New users**
  - New users can be recommended items without model update when feedback is provided
  - New items require model update -> *Asymmetric!*

Benefits of Asymmetric-SVD

- **Improved interpretability of recommendations**
  - SVD-based models are black-box models
    - Abstraction of users into intermediate user factors
  - Asymmetric-SVD does not apply user-side abstraction
    - Can identify the contribution of a user's ratings to the prediction

- **Efficient integration of implicit feedback**
  - Using Implicit & explicit feedback with optimal weights without mixing them
  - Flexibly adjusted based on the type of feedback provided more frequently

SVD++

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u + \boxed{|\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j} \right)$$

- **Adds only parameters for implicit feedback to the original Regularized SVD**

- **Loses the advantages introduced in Asymmetric-SVD**
  - However, it performs well
  - A model that demonstrates the importance of implicit feedback

Integrated model

- Combine SVD++ & Neighborhood model

$$\hat{r}_{ui} = \underbrace{\boxed{\mu + b_u + b_i}}_{\text{Baseline}} + \underbrace{\boxed{q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)}}_{\text{Factor}}$$
$$\underbrace{+ |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij}}_{\text{Neighborhood}}$$

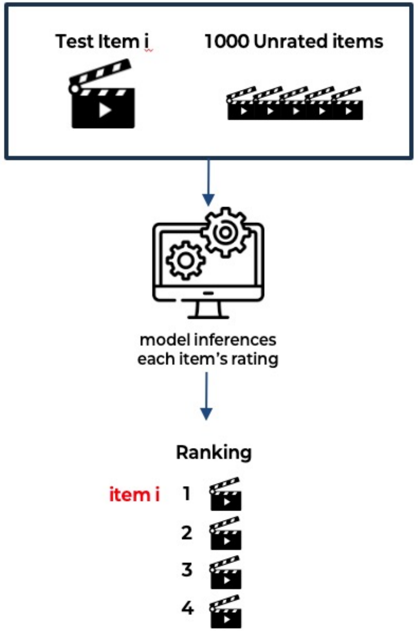
- Baseline tier + Factor tier + Neighborhood tier
  - Describes general tendencies of users and items without interactions
  - Analyzes user-item interactions in more detail
  - Refines predicted ratings based on item's neighbors

Evaluation through a top-k recommender

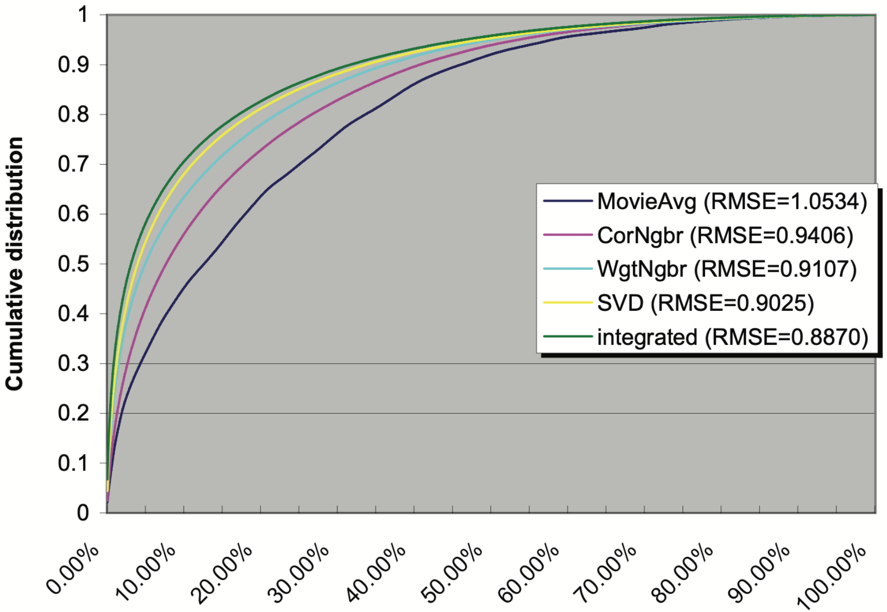
$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2}$$

- **Did the RMSE reduction lead to better user experience?**
  - Trained to minimize the error between true and predicted ratings
  - Does this error reduction actually improve recommendation quality?

- **Validated through top-K recommendation experiment**
  - A movie i rated 5 by user u
  - How high does it rank among 1000 random movies?
    - 1001-item ranking task
    - Best case: movie i ranks first among 1001 (0% position)

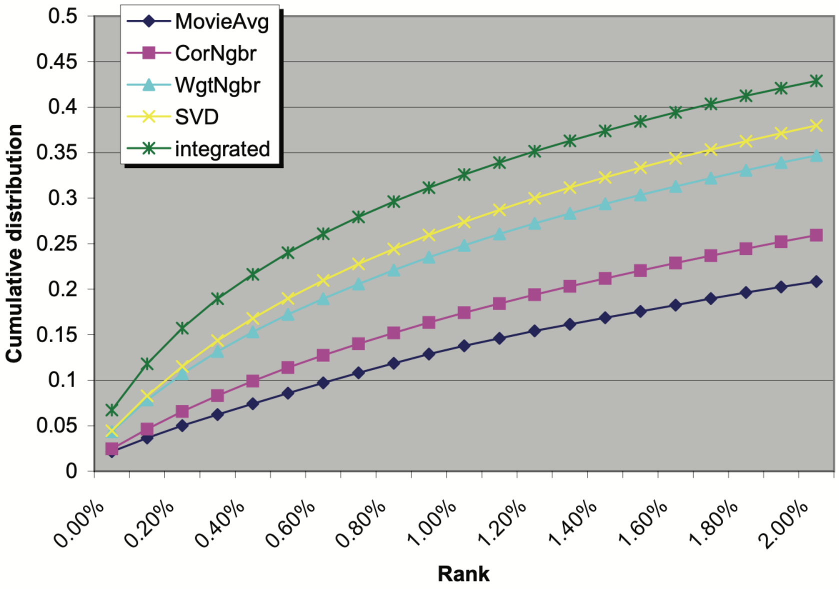


Experiment



- Y-axis represents the cumulative proportion of all experiments
  - Y% of cases where the 5-star item i is ranked within the top X%

Experiment



- Zoomed-in version of the x-axis within 2%
  - The integrated model significantly outperforms the others

Conclusion

- **Neighborhood Method captures local relationships**
  - Recommended using a few similar neighbors
- **Latent factor model captures overall user-item interactions**
- **Integrate two models strengthens with implicit feedback**
  - Use both explicit feedback and implicated feedback

Performance of Recommender Algorithms on Top-N Recommendation Tasks

Paolo Cremonesi
Politecnico di Milano

Yehuda Koren
Yahoo! Research

Roberto Turrin
Neptuny

Published in Recsys, 2010

2025
HoonUi Lee

Contents

Is the "RMSE" suitable?

- **The fundamental goal is not "predicted ratings" but "top-N recommendation"**
  - Existing methods evaluated based on the error between actual and predicted ratings
    - The goal is not to present predicted ratings directly to users
    - Recommendation of top-N items

- **Proposed to properly evaluate top-N recommendation performance**
  - Evaluate the performance of models trained with RMSE using this new metric
    - By showing that these models perform poorly
    - Showing error metrics do not accurately reflect top-N recommendation performance

New Metric for Top-N Recommendation

❖ Precision & Recall

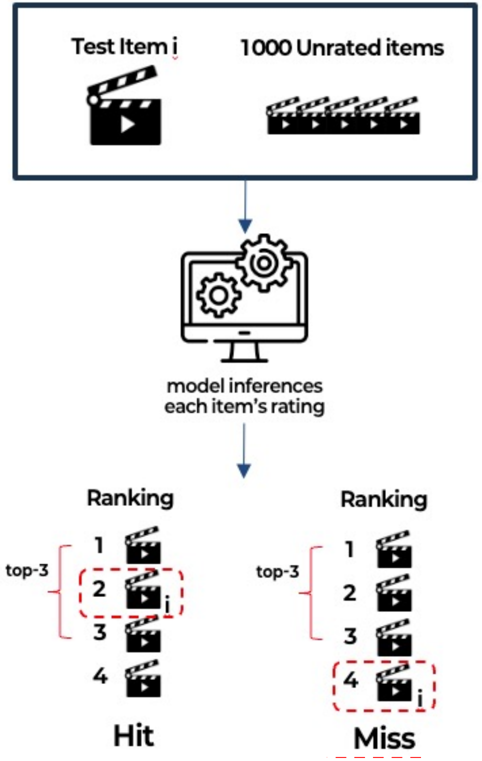
- ❑ Metric based on the number of **“hits”** where test item i is included in the Top-N

$$\mathrm{recall}(N) = \frac{\#\mathrm{hits}}{|T|}$$

- ❑ The number of successful cases among all test cases

$$\mathrm{precision}(N) = \frac{\#\mathrm{hits}}{N \cdot |T|} = \frac{\mathrm{recall}(N)}{N}$$

- ❑ The proportion of test items among all “recommended” items



Long-Tail in Rating Distribution

- Popular items vs long-tail
  - The majority of ratings are concentrated on a small number of popular items
    - 33% of all user ratings were concentrated on the top 1.7% of items
      - These items are referred to as the short-head
      - The remaining less-rated items are called the long-tail

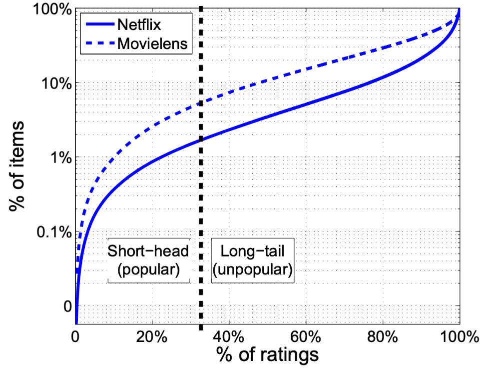


Figure 1: Rating distribution for Netflix (solid line) and Movielens (dashed line). Items are ordered according to popularity (most popular at the bottom).

Why should we consider long-tail?

- **Recommending popular items is trivial**
  - In the case of non-personalized models
    - Recommending mainly popular items can still be considered a good model
    - This does not bring significant benefit to users or content providers

- **Evaluate the accuracy of recommendation algorithms on non-trivial items**
  - Splitting the test set T into two parts
    - T_{head} : short-head items
    - T_{long} : long-tail items

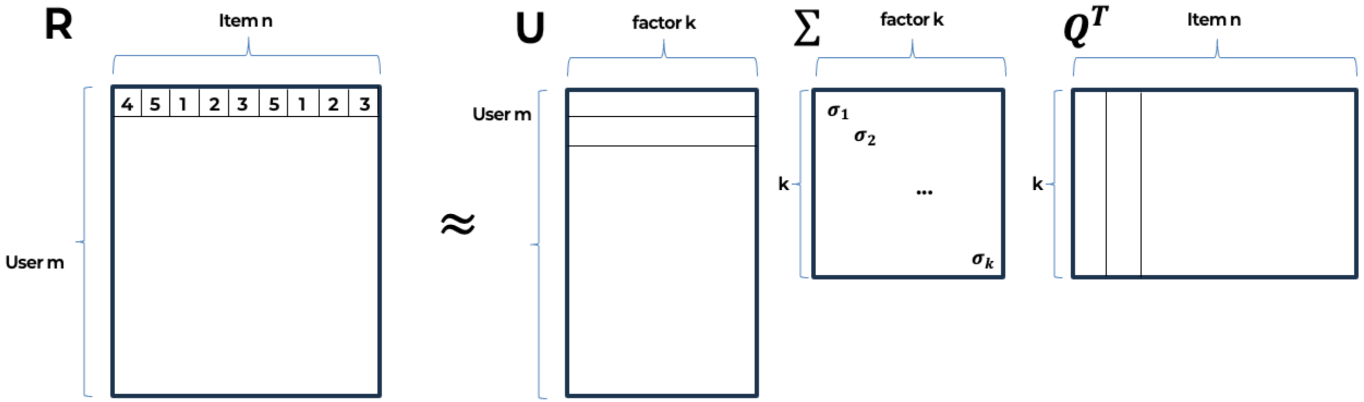
PureSVD

- **No Necessity of "exact" rating prediction for Top-N**
  - Here, flexibility is allowed
    - Missing values in the user-item rating matrix are set to 0
      - RMSE-based models claimed that this distorts the values
    - This flexibility allows the use of traditional SVD

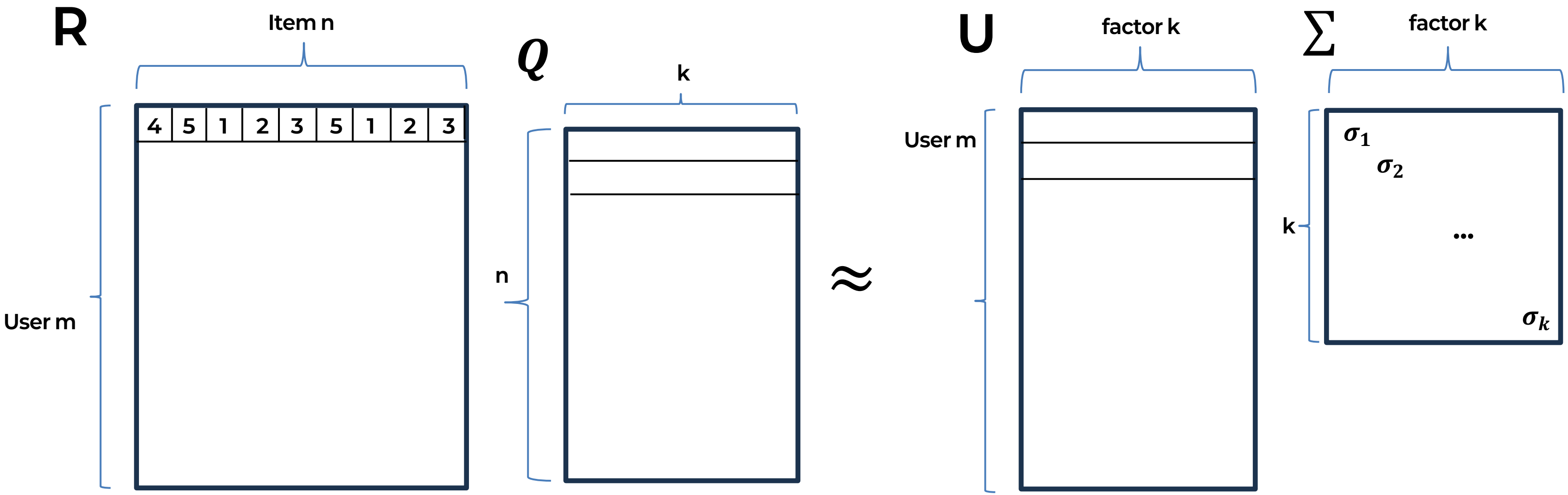
- **Propose the PureSVD model**
  - Using traditional SVD
  - Using only the rating matrix and item information
      - Like Asymmetric-SVD

How to derive

- **Replace user factors vector**
  - Compute p_u from $\hat{r}_{ui} = p_u \cdot q_i^T$
    - Suppose $\mathbf{P} = \mathbf{U} \cdot \mathbf{\Sigma}$, every row of P is user factors vector p_u
  - Predict rating in the form of $\hat{r}_{ui} = \mathbf{r}_u \cdot \mathbf{Q} \cdot \mathbf{q}_i{}^{\mathrm{T}}$



How to derive



Experiment – Dataset

Dataset	Users	Items	Ratings	Density
Movielens	6,040	3,883	1M	4.26%
Netflix	480,189	17,770	100M	1.18%

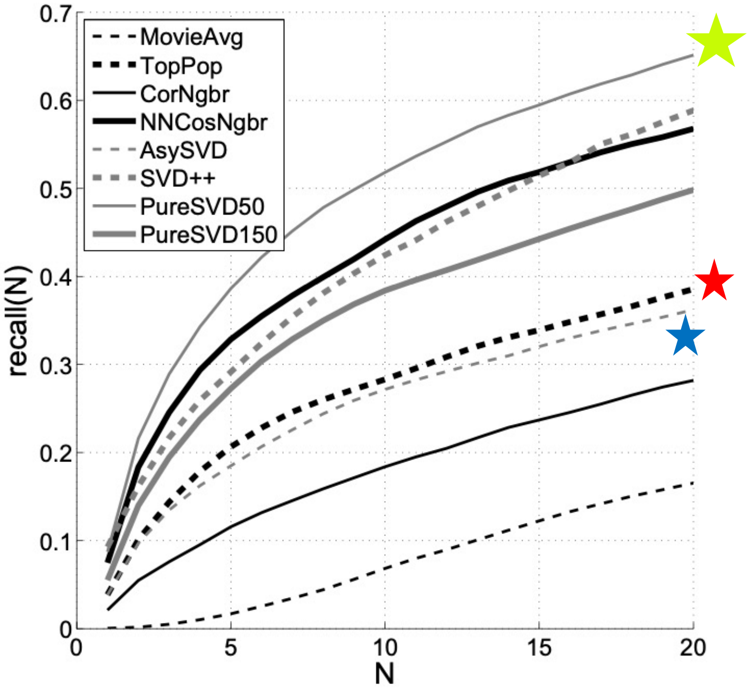
Table 1: Statistical properties of Movielens and Netflix.

- Evaluation results on MovieLens and Netflix data
  - Netflix data is much larger and sparser
- Experiments conducted on both full and long-tail test sets for each dataset
  - Recall and precision presented with respect to the number of recommended items N
    - N ranges from 1 to 20

Experiment - Model

- Non-personalized algorithms
  - MovieAvg
    - Top-N recommendation of the highest average rating
  - TopPop
    - Top-N recommendation of the most rated items
- RMSE-oriented
  - CorNgbr
  - AsySVD (with 200 factors)
  - SVD++ (with 200 factors)
- non-RMSE-oriented
  - NNCosNgbr
    - Using cosine similarity instead of Pearson correlation in CorNgbr + Removing normalization to accumulate similarity scores
  - PureSVD
    - Model proposed in the paper
    - Two options presented: one with 50 factors, another with larger factors

Movielens Dataset - recall



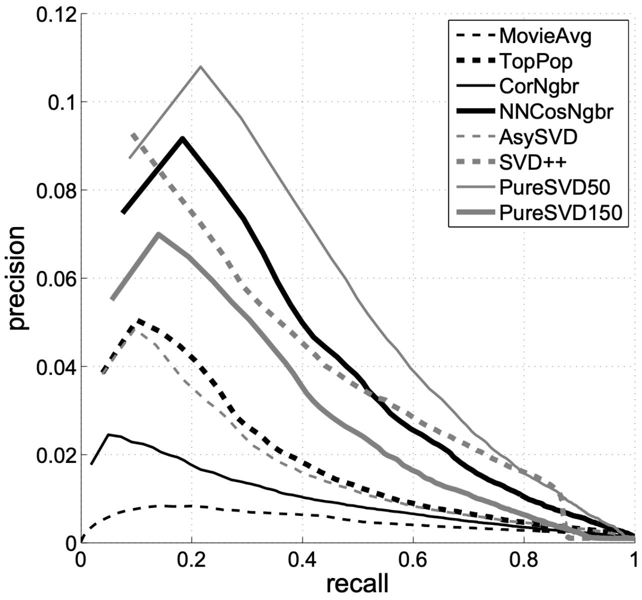
(a) recall

- ❑ AsySVD's recall is about 0.28 at N=10
  - 28% chance of putting an appealing movie in the top-10
  - Not much better than TopPop, a non-personalized algorithm

➔ Motivates the need for a long-tail test set

- ❑ Non-RMSE-oriented algorithms performed best in terms of recall

Movielens Dataset – precision-recall



(b) precision vs recall

- Non-RMSE-oriented algorithms also performed best in terms of precision

Movielens Dataset – Long-tail

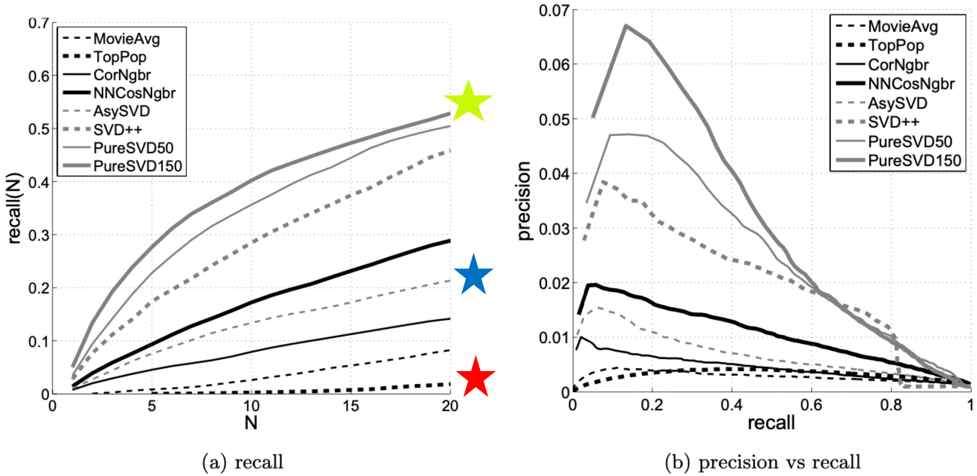
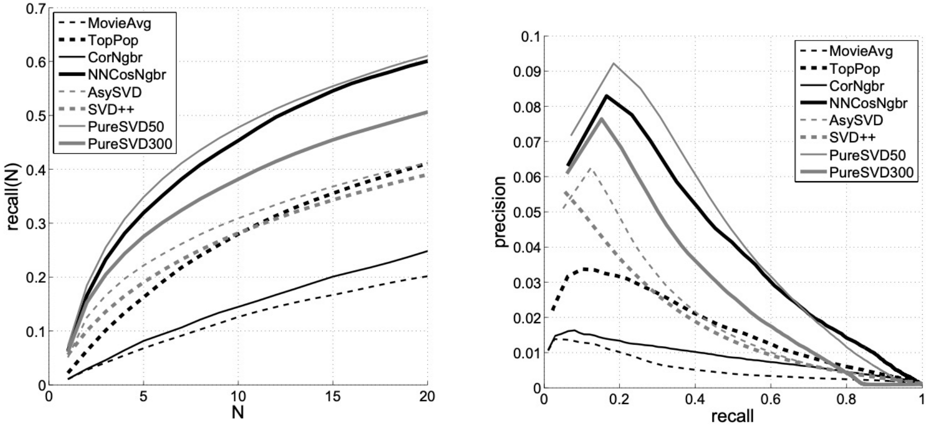


Figure 3: Movielens: (a) recall-at-N and (b) precision-versus-recall on long-tail (94% of items).

- ❑ The ranking of the algorithms normalized somewhat
- ❑ With only long-tail data, the larger-factor PureSVD performed better
  - Rich latent-factor representation helped in the long-tail

Netflix Dataset – All items



(a) recall (b) precision vs recall

Figure 4: Netflix: (a) recall-at-N and (b) precision-versus-recall on all items.

- ❑ TopPop dominates CorNgbr in performance — a strange result
  - Again, need to compare in the long-tail

Netflix Dataset – Long-tail

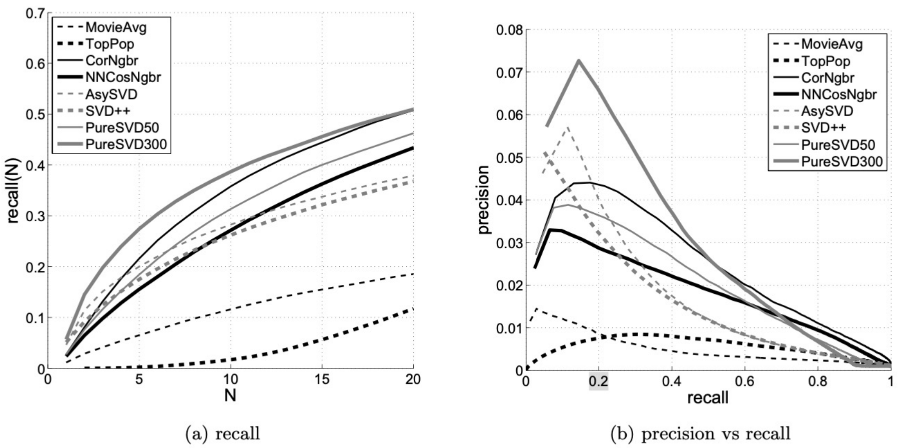


Figure 5: Netflix: (a) recall-at-N and (b) precision-versus-recall on long-tail (98% of items).

- Among non-RMSE-oriented algorithms, PureSVD performs best
  - Another non-RMSE method, NNCosNgbr, shows lower performance
- CorNgbr improves in long-tail cases, while others drop in performance
  - because it effectively captures strong similarities among unpopular items

Advantages of PureSVD

- **PureSVD performs best regardless of whether popular items are included or not**

- **Simple implementation without any hyperparameters that require manual tuning**
  - Computation is easy using off-the-shelf optimized SVD packages

- **Users can be represented as combinations of item characteristics**
  - Interpretability (or Explainability)
  - Easy to handle new users or new rating data from existing users

Conclusion

- **Existing metrics do not properly reflect top-N recommended performance**
  - Evaluated based on the error between actual and predicted ratings

- **Long-tail distribution exists in the rating data**
  - In order to prevent trivial recommendations,
    it is necessary to evaluate the recommendation model considering this

- **Propose PureSVD**
  - Use traditional SVD with an approach that does not focus on accurate rating figures