8. Recommender systems

Based on slides by Irena Koprinska et al.

Outline

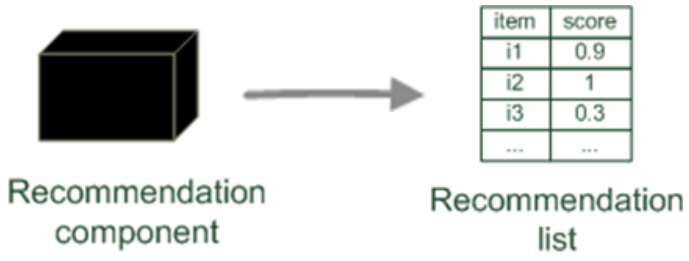
1. Recommending: What and why?
2. Collaborative filtering approaches
3. Content-based approaches

Recommending: What and why?

Recommender Systems

Recommend items to users

- Which digital camera should I buy?
- What is the best holiday for me?
- Which movie should I rent?
- Which book should I buy?
- Which degree is best for my future?



Sometimes, items are people too:

- Which Twitter users should I follow?
- Which writers/bloggers should I read?

Why? The Context

How do we find good items?

- Friends, Experts, Searchers...

The paradox of choice:

- 4 types of jam vs 24 types of jam?

The Problem

Huge volume of data makes it difficult to find "best" items. We can't see all options.

How to recommend

The recommendation problem:

Try to predict items that will interest this user

1. Top-N items (ranked)
2. All interesting items (few false positives)
3. A sequence of items (music playlist)

Based on what information?

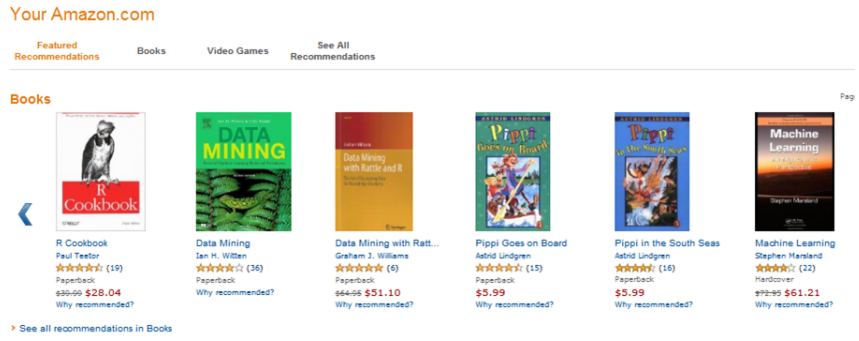
User profiles vs. Ratings

User Profiles

- Explicit info (demographics, interests).
- Problem: People won't bother or have multiple profiles.

Ratings

- **Explicit** (1..5 stars): Hard to obtain.
- **Implicit** (clicks, views): Unreliable/Noisy.



Methods Overview

- **Baseline:** Recommend most popular items
- **Collaborative filtering (CF):** The crowd knows best.
- **Content-based:** Item features matter.
- **Hybrid:** Best of both worlds.

Collaborative filtering approaches

Collaborative Filtering (CF)

- Trusts wisdom of the crowd

- **Input:** a matrix of user-to-item ratings, an active user.
- **Output:** top-N recommendations for active user.

Main CF methods

- **Nearest neighbors:** User-to-user or Item-to-item.
- **Latent Factor Models:** Matrix factorization (SVD).

User-to-user CF: Intuition

Recommend to you what is rated high by people with ratings similar to yours.

1. If you, Joe and Jane like band X,
2. and if you, Joe and Jane like band Y,
3. and if Joe and Jane like band Z...
4. ...then band Z is a good recommendation for you.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Algorithm: User-to-user Nearest Neighbors

1. **Find** k **nearest neighbors** of active user.
   - Needs "distance" or "similarity" metric.
2. **Identify set** C of items bought by these k users.
3. **Recommend top-**N items in C that active user has not purchased.

Similarity Metric

We use correlation as similarity.

Pearson Correlation / Cosine Similarity

$$sim(a,b) = \frac{\sum_{s \in S}(r_{a,s} - \bar{r}_a) \cdot (r_{b,s} - \bar{r}_b)}{\sqrt{\sum_{s \in S}(r_{a,s} - \bar{r}_a)^2} \cdot \sqrt{\sum_{s \in S}(r_{b,s} - \bar{r}_b)^2}}$$

Where S is the set of items rated both by user a and user b.

Prediction Calculation

How much will user a like item s?

Adjusted Weighted Sum:

$$pred(a, s) = \bar{r}_a + \frac{\sum_b sim(a, b) \cdot (r_{b,s} - \bar{r}_b)}{\sum_b sim(a, b)}$$

Note: We subtract the user's average rating to normalize for "optimistic" or "pessimistic" users.

Evaluation

How do we measure success?

Main Metric: Mean Absolute Error (MAE)

$$|\, pred(a, s) - r_{a,s}|$$

Evaluated on a separate **test subset**.

Beyond Accuracy:

- Diversity: Don't recommend Star Wars 3 after 1 and 2.
- Serendipity/Surprise: Don't recommend "milk" in a supermarket.
- Trust/Explainability: Why are you showing me this?

Item-to-item CF

- Look at **columns** of the matrix.
- Find set of items similar to the target one.
- *Why?* Items are more stable than users.
- Amazon uses this for scalability.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Matrix Factorization (SVD)

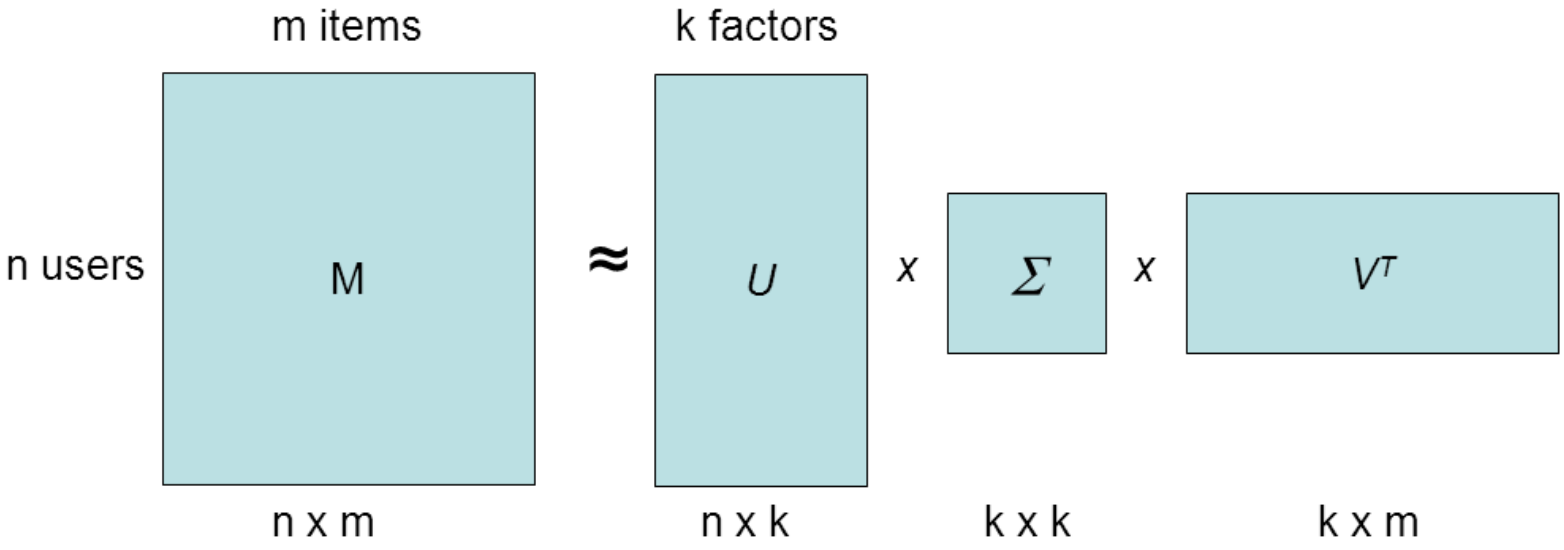
Theorem

Every $n \times m$ matrix M of rank K can be decomposed as $M = U\Sigma V^T$.

U (Users)	Σ (Weights)	V^T (Items)
$n \times K$	$K \times K$	$m \times K$
Affinity to factors	Strength of factors	Relation to factors

Idea: Keep only $k < K$ highest values of Σ for the best approximation (Low-rank approximation).

Latent Factors Interpretation



k latent factors represent hidden topics (e.g., "Action amount", "Romance amount", "Serious vs Funny").

Pros and Cons of CF

Pros

- **No domain knowledge needed.**
- Works for any type of item (music, books, jokes) as long as we have ratings.

Cons

- Requires user community.
- Sparsity issues.
- **Cold start problem:**
  - New user (no history).
  - New item (no ratings).
- No explanation ("Black box").

Content-based approaches

Content-based methods

Use information about the items, not just the crowd.

Logic: Recommend fantasy novels to people who liked fantasy novels in the past.

Requirements:

1. *Item Features:* Genre, actors, director, tf-idf vectors...
2. *User Profile:* Preferences matching those features.

This becomes a standard **Machine Learning Classification/Regression** problem.

Content-based: Pros and Cons

Pros

- No user base required (works for first user).
- **No item cold start:** We can recommend a new book immediately based on its summary.

Cons

- **Feature Engineering:** Hard work to define features.
- Overspecialization (User never sees anything outside their "profile").
- Hard to transfer across domains.

Advanced Topics

- **Scalability:** Handling "zillions" of ratings in real-time.
- **Context-aware:** Don't recommend a NY restaurant when I am in Barcelona.
- **Group Recommendations:** Making a group of friends happy (minimizing misery).
- **Privacy:** Differential privacy to protect user history.
- **Robustness:** Detecting shilling attacks (fake reviews).