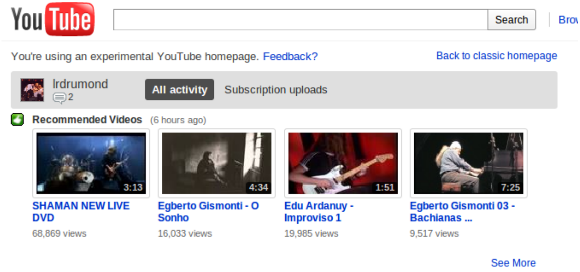
Recommender Systems

Dr. Josif Grabocka

ISMLL, University of Hildesheim

Business Analytics

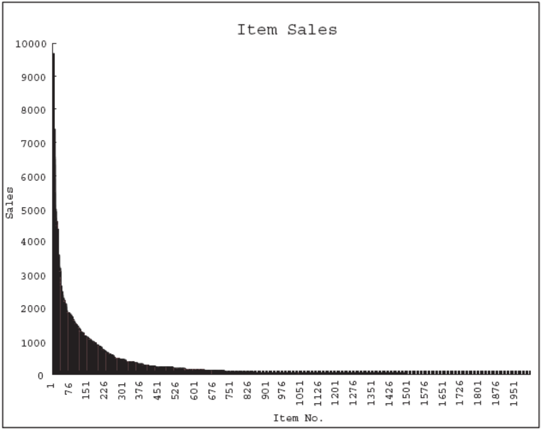
Recommender Systems



Why Recommender Systems?

- Powerful method for enabling users to filter large amounts of information
- Personalized recommendations can boost the revenue of an e-commerce system:
  - Amazon recommender systems
  - Netflix challgenge: 1 million dollars for improving their system on 10%
- Different applications:
  - E-commerce
  - Education
  - ...

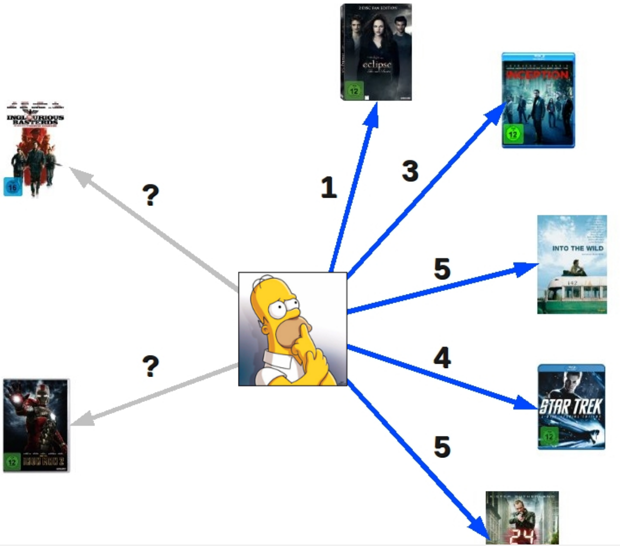
Why Personalization? - The Long Tail



Source: `http://www.ma.hw.ac.uk/esgi08/Unilever.html`

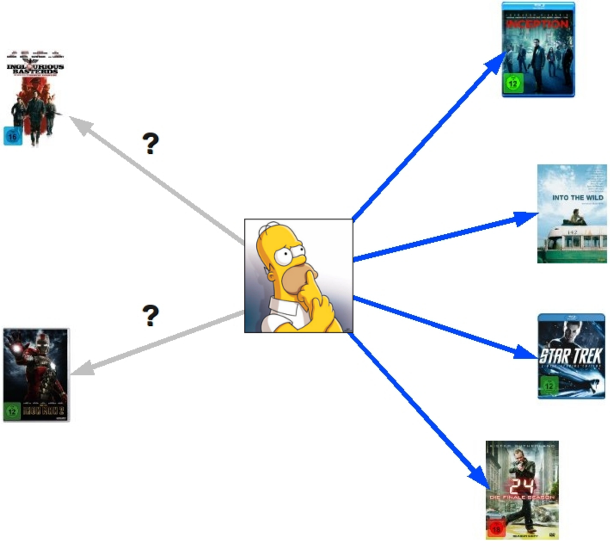
Prediction Version - Rating Prediction

Given the previously rated items, how the user will evaluate other items?



Ranking Version - Item Prediction

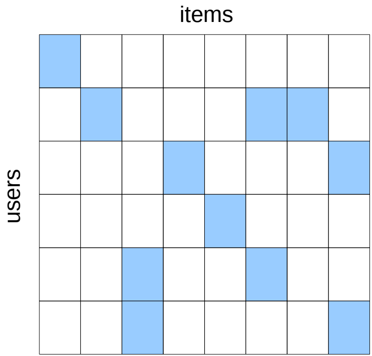
Which will be the next items to be consumed by a user?



Formalization

- U - Set of Users
- I - Set of Items
- Ratings data $D \subseteq U \times I \times \mathbb{R}$

Rating data D are typically represented as a sparse matrix $\mathbf{R} \in \mathbb{R}^{|U| \times |I|}$



Example

	Titanic (t)	Matrix (m)	The Godfather (g)	Once (o)
Alice (a)	4		2	5
Bob (b)		4	3	
John (j)		4		3

- Users $U := \{\text{Alice}, \text{Bob}, \text{John}\}$
- Items $I := \{\text{Titanic}, \text{Matrix}, \text{The Godfather}, \text{Once}\}$
- Ratings data $D := \{(\text{Alice}, \text{Titanic}, 4), (\text{Bob}, \text{Matrix}, 4), \ldots\}$

Recommender Systems - Some definitions

Some useful definitions:

- $\mathcal{N}(u)$ is the set of all items rated by user u
  - $\mathcal{N}(\text{Alice}) := \{\text{Titanic}, \text{The Godfather}, \text{Once}\}$
- $\mathcal{N}(i)$ is the set of all users that rated item i
  - $\mathcal{N}(\text{Once}) := \{\text{Alice}, \text{John}\}$

Recommender Systems - Task

Given a set of users U, items I and training data $D^{train} \subseteq U \times I \times \mathbb{R}$, find a function

$$\hat{r} : U \times I \rightarrow \mathbb{R}$$

such that the loss

$$error(\hat{r}, D^{train}) := \sum_{(u,i,r_{u,i}) \in D^{train}} \ell(r_{u,i}, \hat{r}_{u,i})$$

is minimal

Historical Approaches

Most recommender system approaches can be classified into:

- **Content-Based Filtering:** recommends items similar to the items liked by a user using textual similarity in metadata
- **Collaborative Filtering:** recommends items liked by users with similar behavior

We will focus on collaborative filtering!

Nearest Neighbor Approaches

Nearest neighbor approaches build on the concept of similarity between users and/or items.

The neighborhood N_u of a user u is the set of k most similar users to u

Analogously, the neighborhood N_i of an item i is the set of k most similar items to i

There are two main neighborhood based approaches

- User Based:
  - The rating of an item by a user is computed based on how similar users have rated the same item
- Item Based:
  - The rating of an item by a user is computed based on how similar items have been rated by the same the user

User Based Recommender

A user $u \in U$ is represented as a vector $\mathbf{u} \in \mathbb{R}^{|I|}$ containing user ratings.

	Titanic (t)	Matrix (m)	The Godfather (g)	Once (o)
Alice (a)	4		2	5
Bob (b)		4	3	
John (j)		4		3

Examples:

- $\mathbf{a} := [4, 0, 2, 5]$
- $\mathbf{b} := [0, 4, 3, 0]$
- $\mathbf{j} := [0, 4, 0, 3]$

User Based Recommender - Prediction Function

$$\hat{r}_{u,i} := \bar{r}_u + \frac{\sum_{v \in N_u} \text{sim}(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in N_u} |\text{sim}(u,v)|}$$

Where:

- $\bar{r}_u$ is the average rating of user u
- sim is a similarity function used to compute the neighborhood N_u

Item Based Recommender

An item $i \in I$ is represented as a vector $\mathbf{i} \in \mathbb{R}^{|U|}$ containing information on how items are rated by users.

	Titanic (t)	Matrix (m)	The Godfather (g)	Once (o)
Alice (a)	4		2	5
Bob (b)		4	3	
John (j)		4		3

Examples:

- $\mathbf{t} := [4, 0, 0]$
- $\mathbf{m} := [0, 4, 4]$
- $\mathbf{g} := [2, 3, 0]$
- $\mathbf{o} := [5, 0, 3]$

Item Based Recommender - Prediction Function

$$\hat{r}_{u,i} := \bar{r}_i + \frac{\sum_{j \in N_i} \text{sim}(i,j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in N_i} |\text{sim}(i,j)|}$$

Where:

- $\bar{r}_i$ is the average rating of item i
- sim is a similarity function used to compute the neighborhood N_i

Similarity Measures

On both user and item based recomenders the similarity measure plays an important role:

- It is used to compute the neighborhood of users and items (neighbors are most similar ones)
- It is used during the prediction of the ratings

Which similarity measure to use?

Similarity Measures

Commonly used similarity measures:
Cosine:

$$sim(u, v) = \cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

Pearson correlation:

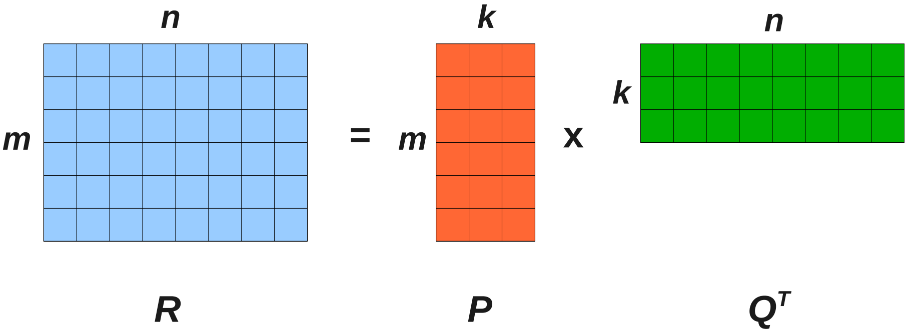
$$sim(u, v) = \frac{\sum\limits_{i \in \mathcal{N}(u) \cap \mathcal{N}(v)} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in \mathcal{N}(u) \cap \mathcal{N}(v)} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum\limits_{i \in \mathcal{N}(u) \cap \mathcal{N}(v)} (r_{v,i} - \bar{r}_v)^2}}$$

Factorization Models

Neighborhood based approaches have been shown to be effective but ...

- Computing and maintaining the neighborhoods is expensive
- In the last years, a number of models have been shown to outperform them
- One of the results of the Netflix Challenge was the power of factorization models when applied to recommender systems

Factorization Models

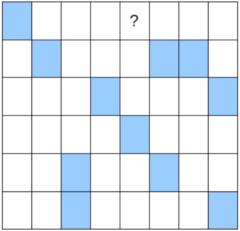


Partially observed matrices

The ratings matrix **R** is usually partially observed:

- No user is able to rate all items
- Most of the items are not rated by all users

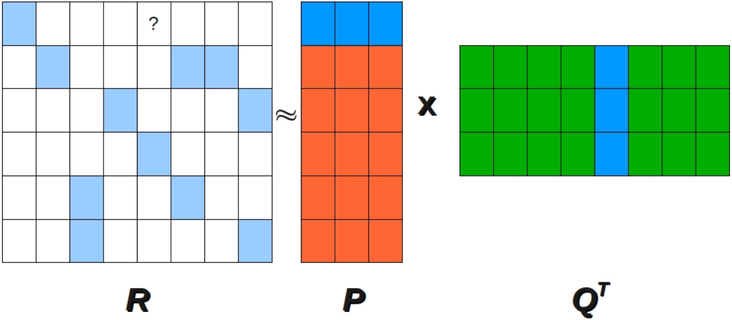
Can we estimate the factorization of a matrix from some observations to predict its unobserved part?



Factorization models

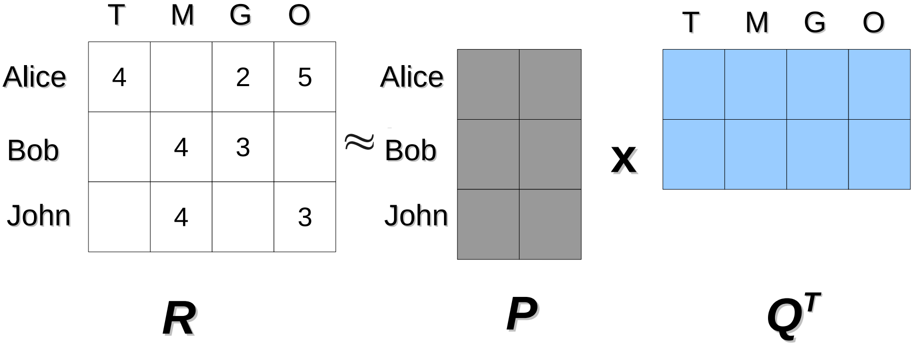
- Each item $i \in I$ is associated with a latent feature vector $\mathbf{Q}_i \in \mathbb{R}^K$
- Each user $u \in U$ is associated with a latent feature vector $\mathbf{P}_u \in \mathbb{R}^K$
- Each entry in the original matrix can be estimated by

$$\hat{r}_{u,i} = \mathbf{P}_u^\top \mathbf{Q}_i = \sum_{k=1}^{K} P_{u,k} Q_{i,k}$$

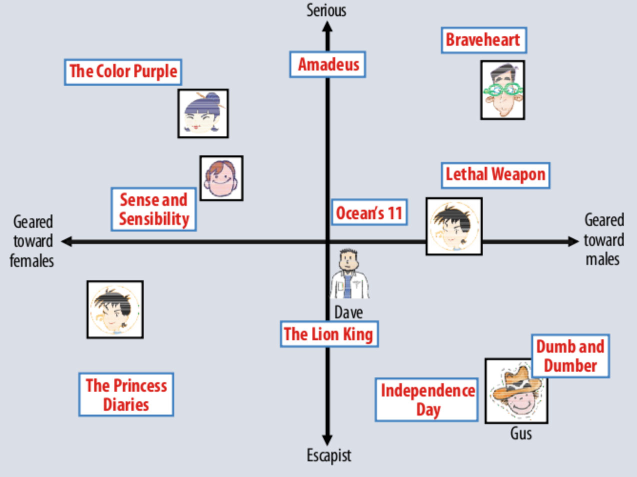


Example

	Titanic (t)	Matrix (m)	The Godfather (g)	Once (o)
Alice (a)	4		2	5
Bob (b)		4	3	
John (j)		4		3



Latent Factors



Source: Yehuda Koren, Robert Bell, Chris Volinsky: *Matrix Factorization Techniques for Recommender Systems*, Computer, v.42 n.8, p.30-37, August 2009

Learning a factorization model - Objective Function

Task:

$$\arg\min_{\mathbf{p},\mathbf{q}} \sum_{(u,i,r_{u,i})\in D^{train}} (r_{ui} - \hat{r}_{u,i})^2 + \lambda(||\mathbf{P}||^2 + ||\mathbf{Q}||^2)$$

Where:

- $\hat{r}_{u,i} := \mathbf{P}_u^\top \mathbf{Q}_i$
- D^{train} is the training data
- λ is a regularization constant

Optimization method

$$\mathcal{L} := \sum_{(u,i,r_{u,i}) \in D^{train}} (r_{ui} - \hat{r}_{u,i})^2 + \lambda(||\mathbf{P}||^2 + ||\mathbf{Q}||^2)$$

Stochastic Gradient Descent:

Conditions:

- Loss function should be decomposable into a sum of components
- The loss function should be differentiable

Procedure:

- Randomly draw one component of the sum
- Update the parameters in the opposite direction of the gradient

SGD: gradients

$$\mathcal{L} := \sum_{(u,i,r_{u,i}) \in D^{train}} (r_{ui} - \hat{r}_{u,i})^2 + \lambda(||\mathbf{P}||^2 + ||\mathbf{Q}||^2) \quad (1)$$

$$\mathcal{L} := \sum_{(u,i,r_{u,i}) \in D^{train}} \mathcal{L}_{u,i} \quad (2)$$

Gradients:

$$\frac{\partial \mathcal{L}_{u,i}}{\partial \mathbf{P}_{u,k}} = -2(r_{u,i} - \hat{r}_{u,i})\mathbf{Q}_{i,k} + 2\lambda \mathbf{P}_{u,k}$$

$$\frac{\partial \mathcal{L}_{u,i}}{\partial \mathbf{Q}_{i,k}} = -2(r_{u,i} - \hat{r}_{u,i})\mathbf{P}_{u,k} + 2\lambda \mathbf{Q}_{i,k}$$

Stochastic Gradient Descent Algorithm (Naive)

procedure LEARNLATENTFACTORS
input: D^{Train}, λ, α
    $(\mathbf{p}_u)_{u \in U} \sim N(0, \sigma \mathbf{I})$
    $(\mathbf{q}_i)_{i \in I} \sim N(0, \sigma \mathbf{I})$
    repeat
        for $(u, i, r_{u,i}) \in D^{Train}$ **do** ▷ In a random order
            for $k = 1, \ldots, K$ **do**
                $\mathbf{P}_{u,k} \leftarrow \mathbf{P}_{u,k} - \alpha \left(-2(r_{u,i} - \hat{r}_{u,i})\mathbf{Q}_{i,k} + 2\lambda \mathbf{P}_{u,k}\right)$
                $\mathbf{Q}_{i,k} \leftarrow \mathbf{Q}_{i,k} - \alpha \left(-2(r_{u,i} - \hat{r}_{u,i})\mathbf{P}_{u,k} + 2\lambda \mathbf{Q}_{i,k}\right)$
            end for
        end for
    until convergence
    return $\mathbf{P}, \mathbf{Q}$
end procedure

Stochastic Gradient Descent Algorithm

procedure LEARNLATENTFACTORS
input: D^{Train}, λ, α
    $(\mathbf{p}_u)_{u \in U} \sim N(0, \sigma \mathbf{I})$
    $(\mathbf{q}_i)_{i \in I} \sim N(0, \sigma \mathbf{I})$
    repeat
        for $(u, i, r_{u,i}) \in D^{Train}$ **do** ▷ In a random order
            $\xi_{u,i} = r_{u,i} - \hat{r}_{u,i}$
            for $k = 1, \ldots, K$ **do**
                $\mathbf{P}_{u,k} \leftarrow \mathbf{P}_{u,k} + \alpha \left(\xi_{u,i} \mathbf{Q}_{i,k} - \lambda \mathbf{P}_{u,k} \right)$
                $\mathbf{Q}_{i,k} \leftarrow \mathbf{Q}_{i,k} + \alpha \left(\xi_{u,i} \mathbf{P}_{u,k} - \lambda \mathbf{Q}_{i,k} \right)$
            end for
        end for
    until convergence
    return $\mathbf{P}, \mathbf{Q}$
end procedure

Factorization Models on practice

Dataset: MovieLens (ML1M)

- Users: 6040
- Movies: 3703
- Ratings:
  - From 1 (worst) to 5 (best)
  - 1.000.000 observed ratings (approx. 4.5% of possible ratings)

Evaluation

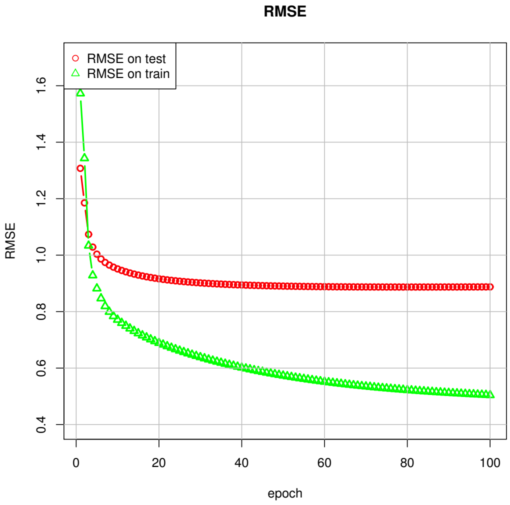
Evaluation protocol

- 10-fold cross validation
- Leave-one-out

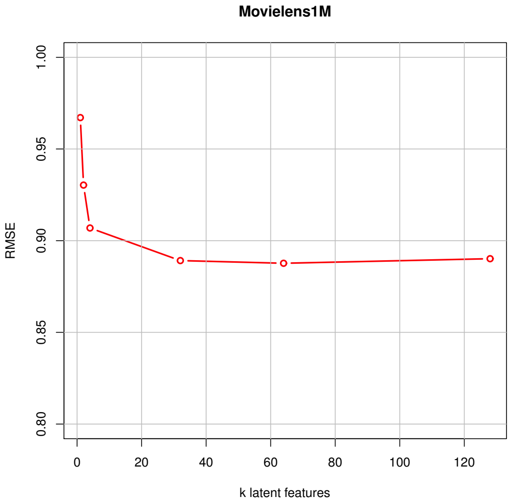
Measure: RMSE (Root Mean Squared Error)

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

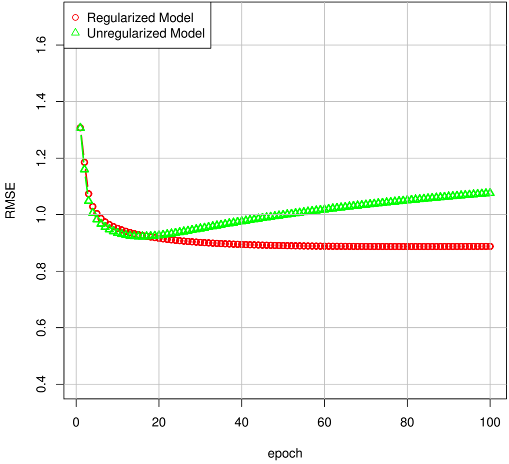
SGD for factorization Models - Performance over epochs



Factorization Models - Impact of the number of latent features



Factorization Models - Effect of regularization



Biased Matrix Factorization

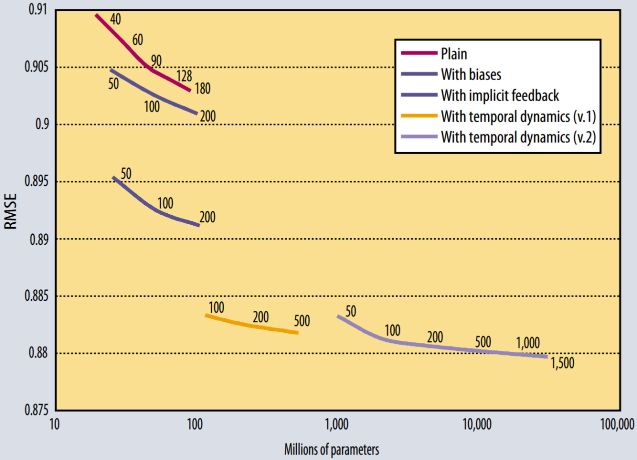
- Specific users tend to have specific rating behavior
  - Some users may tend to give higher (or lower) ratings
- The same can be said about items
- This can be easily modeled through bias terms for users b_u and for items b_i in the prediction function:

$$\hat{r}_{u,i} = b_u + b_i + \mathbf{P}_u^\top \mathbf{Q}_i$$

- Additionally a global bias can be added:

$$\hat{r}_{u,i} = g + b_u + b_i + \mathbf{P}_u^\top \mathbf{Q}_i$$

Effect of the Biases



Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. IEEE Computer, 42(8):30–37, 2009.

Integrating Implicit feedback

- In many situations we have information about items that the user has consumed but did not evaluate
  - Videos watched
  - Products bought
  - Webpages visited
  - ...
- The set of items $\mathcal{N}(u)$ consumed by a user u (rated or not) provides useful information about the tastes of the user

SVD++

SVD++ (Koren 2008) incorporates information about implicit feedback into user factors

User factors are represented as:

$$\mathbf{P}_u + \frac{1}{|\mathcal{N}(u)|} \sum_{j \in \mathcal{N}(u)} \mathbf{V}_j$$

The prediction function is then written as:

$$\hat{r}_{ui} := b_u + b_i + \mathbf{Q}_i^T \left(\mathbf{P}_u + \frac{1}{|\mathcal{N}(u)|} \sum_{j \in \mathcal{N}(u)} \mathbf{V}_j \right)$$

Where:

- $\mathbf{V}_j \in \mathbb{R}^k$ are item latent vectors used to construct user profile.
- $\mathcal{N}(u)$ is the set of items consumed by the user u.

SVD++ Performance

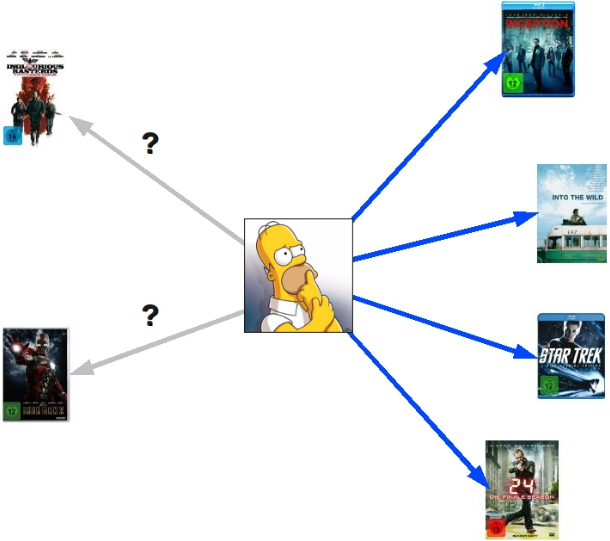
Dataset: Netflix
Measure: RMSE

Model	50 factors	100 factors	200 factors
MF	0.9046	0.9025	0.9009
SVD++	0.8952	0.8924	0.8911

Source: Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model, KDD 2008

Ranking-Version Item Prediction

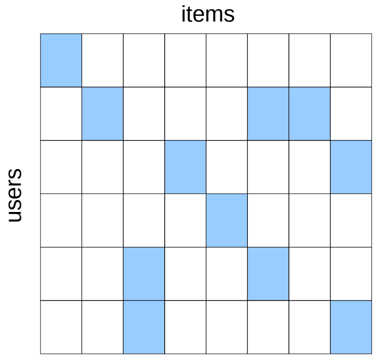
Which will be the next items to be consumed by a user?



Formalization

- U - Set of Users
- I - Set of Items
- Positive implicit feedback data $D \subseteq U \times I \times \{1\}$

We have available only information about $\mathcal{N}(u)$ which items the user has interacted with



Considerations

- We do not know whether a user has liked an item or not (how he rated it)
- The only information we have is which items the user has bought, watched, clicked, ...
- The task is to predict which will be the next items the user will interact with next
- We can assume that items already evaluated ($i \in \mathcal{N}(u)$) are preferred over the not evaluated ones ($i \notin \mathcal{N}(u)$)

Item Prediction Task

Assuming that items already evaluated are preferred over the not evaluated ones

$$i >_u j \text{ iff } i \in \mathcal{N}(u) \text{ and } j \notin \mathcal{N}(u)$$

Given a dataset $D_S \subseteq U \times I \times I$:

$$D_S := \{(u, i, j) | i \in \mathcal{N}(u) \wedge j \notin \mathcal{N}(u)\}$$

For each user, find a total order $>_u$ over items $j \notin \mathcal{N}(u)$ that reflects user preferences

Item Prediction Approach

- Learn a model $\hat{r} : U \times I \to \mathbb{R}$
- Sort items according to scores predicted by the model such that:

$$i >_u j \text{ iff } \hat{r}_{u,i} > \hat{r}_{u,j}$$

In a probabilistic setting, be Θ the model parameters, then

$$p(i >_u j|\Theta) := \sigma(\hat{y}_{u,i,j})$$

Where:

- $\sigma(x) := \frac{1}{1+e^{-x}}$
- $\hat{y}_{u,i,j} := \hat{r}_{u,i} - \hat{r}_{u,j}$

Bayesian Personalized Ranking (BPR)

The Maximum Likelihood Estimator:

$$\arg\max_{\Theta} p(\Theta| >_u) \propto \arg\max_{\Theta} p(>_u |\Theta) p(\Theta)$$

Prior:

$$p(\Theta) := N(0, \Sigma_\Theta)$$

The Bayesian Personalized Ranking Optimization Criterion (BPR-Opt)

$$
\begin{aligned}
\text{BPR-Opt} := & \ln \prod_{u \in U} p(\Theta| >_u) \\
& = \ln \prod_{u \in U} p(>_u |\Theta) p(\Theta) \\
& = \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{y}_{u,i,j}) p(\Theta) \\
& = \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{y}_{u,i,j}) + \ln p(\Theta) \\
& = \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{y}_{u,i,j}) - \lambda ||\Theta||^2
\end{aligned}
$$

Optimizing a factorization model for BPR:

Model:

$$\hat{r}_{u,i} = \mathbf{P}_u^\top \mathbf{Q}_i = \sum_{k=1}^{K} P_{u,k} Q_{i,k}$$

Loss Function:

$$\mathcal{L} := \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{y}_{u,i,j}) - \lambda ||\Theta||^2$$

Gradients:

$$\frac{\partial \text{BPR-Opt}}{\partial \theta} = \frac{e^{-\hat{y}_{u,i,j}}}{1 + e^{-\hat{y}_{u,i,j}}} \cdot \frac{\partial}{\partial \theta} \hat{y}_{u,i,j} - 2\lambda\theta$$

$$\frac{\partial}{\partial \theta} \hat{y}_{u,i,j} = \begin{cases} (Q_{i,k} - Q_{j,k}) & \text{if } \theta = P_{u,k} \\ P_{u,k} & \text{if } \theta = Q_{i,k} \\ -P_{u,k} & \text{if } \theta = Q_{j,k} \end{cases}$$

Stochastic Gradient Descent Algorithm

procedure LEARNBPR
input: D_S^{Train}, λ, η, Σ
  $(\mathbf{P}_u)_{u \in U} \sim N(0, \Sigma)$
  $(\mathbf{Q}_i)_{i \in I} \sim N(0, \Sigma)$
  repeat
    for $(u, i, j) \in D_S^{Train}$ **do** ▷ In a random order
      $\hat{y}_{u,i,j} := \hat{r}_{u,i} - \hat{r}_{u,j}$
      $\xi_{u,i,j} := \frac{e^{-\hat{y}_{u,i,j}}}{1+e^{-\hat{y}_{u,i,j}}}$
      for $k \in 1, \ldots, K$ **do**
        $P_{u,k} \leftarrow P_{u,k} + \eta \left(\xi_{u,i,j} \left(Q_{i,k} - Q_{j,k}\right) - 2\lambda P_{u,k}\right)$
        $Q_{i,k} \leftarrow Q_{i,k} + \eta \left(\xi_{u,i,j} P_{u,k} - 2\lambda Q_{i,k}\right)$
        $Q_{j,k} \leftarrow Q_{j,k} + \eta \left(-\xi_{u,i,j} P_{u,k} - 2\lambda Q_{j,k}\right)$
      end for
    end for
  until convergence
  return $\mathbf{P}, \mathbf{Q}$
end procedure

Exercise

- Given $R = \begin{bmatrix} 1 & \\ 1 & \end{bmatrix}$.
- Let $K = 1$ and initialized $P^T = \begin{bmatrix} 1 & 1 \end{bmatrix}$ and $Q = \begin{bmatrix} -1 & 1 \end{bmatrix}$.
- Let $\eta = 1, \lambda = 0$.

- **Compute P, Q after one BPR iteration.**