Tomáš Horváth

BUSINESS ANALYTICS

Lecture 5

Recommendation Techniques

Information Systems and Machine Learning Lab

University of Hildesheim

Germany

Overview

The aim of this lecture is to describe personalization techniques and the main recommender techniques

- User feedback
- Evaluation measures
- Recommendation techniques
    - Demographic, Knowledge-based, Utility-based, Content-based and Collaborative-filtering
- Context-Aware recommendation

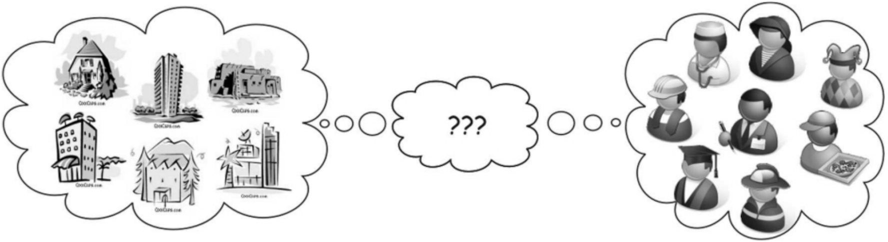
I guess, each of us has already met a real recommender system, thus, I expect more passionate discussions ☺

- It means a lot of questions on the slides.

Recommender Systems

A Recommender System (RS) aims to ease the user from an exhaustive process of seeking for relevant information by recommending her items she would be more likely interested in.

- personalization



The more important issues a RS have to deal with are

- large set of items (changing continuously in time)
- different types of users

User's Profile

User's **characteristics** contain personal information about the user

- age, income, marital status, education, profession, nationality, etc.
- also could contain information about the habits of a user
  - preferred sport, hobbies, favourite newspapers, etc.
- the most simple way to obtain these information are questionnaires
  - *What are the obstacles here?*
  - *Can You imagine other ways to obtain these information?*

User's **preferences** indicate which items, which properties of items and/or what combination of these properties are preferred by a user

- *Can questionnaires be used to obtain user's preferences?*
  - *If yes then how? If no then why?*

We would like to get feedback from users in a way such that they are less burdened.

Implicit Feedback

Information obtained about users by watching their natural interactions with the system[1].

Behaviour category	Minimum scope		
	Segment	Object	Class
Examine	View Listen Scroll Find Query	Select	Browse
Retain	Print	Bookmark Save Delete Purchase E-mail	Subscribe
Reference	Copy&Paste Quote	Forward Reply Link Cite	
Annotate	Mark-up	Rate Publish	Organize

[1] An augmented categorization and detailed description of implicit feedback can be found in (Kelly & Teevan 2003)

Explicit Feedback

Examples of explicit information collection include the following

- **rating** items on a rating scale[1]
- **scoring** items
- **ranking** a collection of items
- **choosing the better**[2] one from two presented items
- **provide** a list of preferred items

rating	scoring	ranking	choosing	provide
$r(A) = 3$ $r(B) = 2$ $r(C) = 2$ $r(D) = 4$ $r(E) = 5$	$r(A) = 15$ $r(B) = 10$ $r(C) = 8$ $r(D) = 20$ $r(E) = 50$	$E \succeq D \succeq A \succeq B \succeq C$	$E \succeq D, E \succeq A, E \succeq B, E \succeq C$ $D \succeq A, D \succeq B, D \succeq C$ $A \succeq B, A \succeq C$	$\{E, D, A\}$

Which type of EF is easier for a user to provide?
What is the difference in the semantics of these types of EF?

[1] Often a so-called *Likert's scale* is used.

[2] In other words, *pairwise ranking*

A Generic Recommendation Task

Given

- set of **users** and **items** U and I, respectively, and "transformed" **feedback** values F
  - in case of implicit feedback, usually $F = \{1\} \subset \mathbb{R}$
  - $F = [0, 1] \subset \mathbb{R}$ in case of explicit feedback
- partially observed **user-item interactions** $\phi|_{D \subset U \times I}$, where $\phi : U \times I \to F$
- **metadata** of users and items $\upsilon : U \to \mathcal{M}_U, \iota : I \to \mathcal{M}_I$, respectively, with $\mathcal{M}_U, \mathcal{M}_I$ be user and item metadata descriptions
- **background knowledge** $\mathcal{B}$ about the domain

The task is to

- learn a **user model** $\hat{\phi} : U \times I \to [0, 1]$ such that $acc(\hat{\phi}, \phi, D')$ is maximal, where acc is the **accuracy** of $\hat{\phi}$ w.r.t. ϕ measured on a set $D' \subseteq (U \times I) \setminus D$ of "unseen" (or future) user-item pairs.
  - $\hat{\phi}$ – predicted user's rating for an item (rating prediction)
  - $\hat{\phi}$ – predicted likelihood of user's "positive" implicit feedback for an item (item recommendation)

Recommendation tasks: Example

Rating prediction from explicit feedback

- How would Steve rate the movie Titanic more likely?

	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Joe	1	4	5		3
Ann	5	1		5	2
Mary	4	1	2	5	
Steve	?	3	4		4

Item recommendation from implicit feedback

- Which movie(s) would like Steve to see/buy more likely?

	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Joe	1	1	1		1
Ann	1	1		1	1
Mary	1	1	1	1	
Steve	?	1	1	?	1

Accuracy Measures for Rating Prediction

The goal is to measure the accuracy of predicted ratings.

- mean average error (MAE) or root mean squared error (RMSE) can be used

$$acc(\hat{\phi}, \phi, D') = 1 - \sqrt{\frac{\sum_{(u,i) \in D'} (\hat{\phi}(u,i) - \phi(u,i))^2}{|D'|}}$$

- average MAE or average RMSE
  - if data are imbalanced then accuracy is influenced by the eror on a few frequent items
  - compute MAE or RMSE for each item and average over all items
- use of distortion measure $d(\hat{\phi}, \phi)$, if the impact of the prediction error does not depends only on its magnitude

d(i,j)	1	2	3
1	–	2	1
2	3	–	1
3	5	3	–

Accuracy Measures for Item Recommendation

Goal is to measure the accuracy of predicted user's interest on items.

- we should be aware of items previously unseen by the user

	Recommended	Not recommended
Interested	true positive (TP)	false negative (FN)
Not interested	false positive (FP)	true negative (TN)

$$Precision^1 = \frac{|\{(u,i) \in D' \mid \phi(u,i) = 1, \hat{\phi}(u,i) > 0\}|}{|\{(u,i) \in D' \mid \hat{\phi}(u,i) > 0\}|} = \frac{TP}{TP + FP}$$

$$Recall^2 = \frac{|\{(u,i) \in D' \mid \phi(u,i) = 1, \hat{\phi}(u,i) > 0\}|}{|\{(u,i) \in D' \mid \phi(u,i) = 1\}|} = \frac{TP}{TP + FN}$$

- *Precision@K* if the number of recommended items are K
- longer recommendation lists improve recall while reduce precision
  - we need to find a tradeoff between precision and recall

[1] How many recommended items are interesting?

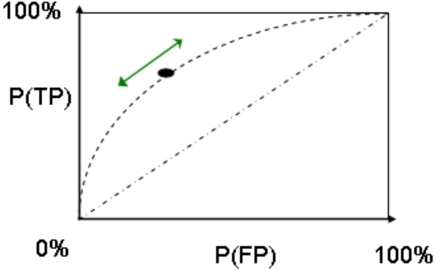
[2] How many interesting items are recommended?

Precision and Recall Trade-off

- F-measure

$$F = 2\frac{P \cdot R}{P + R}$$

- Precision-Recall and ROC curves[1] provided over a range of recommendation list lengths
    - Area under the ROC[2] curve (AUC)



[1] Do not forget, that P-R and ROC curves are different things.

[2] Image source: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Types of RS (1)

Demographic

- Recommendations are based on demographic classes of users.
  - Problem with gathering demographic data
  - Doesn't work well for users who fall in a border between existing classes ("gray sheeps")

Knowledge-based

- Recommendations based on knowledge about users needs and preferences.
  - need to acquire the required knowledge and manage an "expert system"

Utility-based

- The system helps the user to build her utility function, i.e. provide "weights" for item features and/or their preferred combination.
- often is exhaustive for users

Types of RS (2)

Content-based

- Learn user's interests based on the features of items previously rated by the user, using supervised machine learning techniques.
- Useful when we face a "new-item problem"
- often it is hard/expensive to get item features
- tend to "narrow" down the recommendation, i.e. can't recommend items with different features

Collaborative-filtering

- Recognize similarities between users according to their feedbacks and recommend objects preferred by the like-minded users.
- Doesn't work in the case of too few users or/and items ("cold-start" problem)
- Cross-genre recommendation ability
  - if users have similar taste in one domain they should have similar taste in other domains, too

Neighborhood-based CF

Recommendations for the "active" user u on the item i is made directly using feedback values stored in the system.

- user-based
  - computing $\hat{\phi}(u,i)$ from the feedback given by **most similar users** v to the user u, which have already given a feedback for i

$$\mathcal{N}_i^{u,k} = \arg\max_{\mathcal{U}} \sum_{\substack{v \in \mathcal{U}, v \neq u \\ \mathcal{U} \subseteq \mathcal{U}_i, |\mathcal{U}|=k}} sim(u,v)$$

    where $\mathcal{U}_i = \{v \in U \mid \phi(v,i) \text{ is defined on } D\}$

- item-based
  - computing $\hat{\phi}(u,i)$ using feedback values given by the user u to the **most similar items** j to the item i

$$\mathcal{N}_u^{i,k} = \arg\max_{\mathcal{I}} \sum_{\substack{j \in \mathcal{I}, j \neq i \\ \mathcal{I} \subseteq \mathcal{I}_u, |\mathcal{I}|=k}} sim(i,j)$$

    where $\mathcal{I}_u = \{j \in I \mid \phi(u,j) \text{ is defined on } D\}$

The Cosine Vector Similarity[1]

A row/column in a user-item interaction matrix is a sparse vector

$$sim_{cv}(u, v) = \frac{\sum_{i \in \mathcal{I}_{uv}} \phi_{ui} \cdot \phi_{vi}}{\sqrt{\sum_{i \in \mathcal{I}_u} \phi_{ui}^2 \, \sum_{i \in \mathcal{I}_v} \phi_{vi}^2}}$$

$$sim_{cv}(i, j) = \frac{\sum_{u \in \mathcal{U}_{ij}} \phi_{ui} \cdot \phi_{uj}}{\sqrt{\sum_{u \in \mathcal{U}_i} \phi_{ui}^2 \, \sum_{u \in \mathcal{U}_j} \phi_{uj}^2}}$$

More approppriate in case of item recommendation

- doesn't consider differences in mean and variance of the ratings

[1] Simplified notation: $\phi(u, i) \rightsquigarrow \phi_{ui}$, $\mathcal{I}_u \cap \mathcal{I}_v \rightsquigarrow \mathcal{I}_{uv}$, $\mathcal{U}_i \cap \mathcal{U}_j \rightsquigarrow \mathcal{U}_{ij}$

The Pearson Correlation Similarity

$$sim_{pc}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (\phi_{ui} - \overline{\phi}_u)(\phi_{vi} - \overline{\phi}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (\phi_{ui} - \overline{\phi}_u)^2 \ \sum_{i \in \mathcal{I}_{uv}} (\phi_{vi} - \overline{\phi}_v)^2}}$$

where $\overline{\phi}_u = \frac{\sum_{i \in \mathcal{I}_u} \phi(u,i)}{|\mathcal{I}_u|}$

$$sim_{pc}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (\phi_{ui} - \overline{\phi}_i)(\phi_{uj} - \overline{\phi}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (\phi_{ui} - \overline{\phi}_i)^2 \ \sum_{i \in \mathcal{U}_{ij}} (\phi_{uj} - \overline{\phi}_j)^2}}$$

where $\overline{\phi}_i = \frac{\sum_{u \in \mathcal{U}_i} \phi(u,i)}{|\mathcal{U}_i|}$

In which cases can't be this measure computed?

Similarity measures (rating prediction): Example

$sim_{pc}(i,j)$	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Titanic	1.0	-0.956	-0.815	NaN	-0.581
Pulp Fiction	–	1.0	0.948	NaN	0.621
Iron Man	–	–	1.0	NaN	0.243
Forrest Gump	–	–	–	1.0	NaN
The Mummy	–	–	–	–	1.0

NaN values are usually converted to zero

- such cases should be rare in case of enough data

$sim_{pc}(u,v)$	Joe	Ann	Mary	Steve
Joe	1.0	-0.716	-0.762	-0.005
Ann	–	1.0	0.972	0.565
Mary	–	–	1.0	0.6
Steve	–	–	–	1.0

$sim_{cv}(i,j)$	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Titanic	1.0	0.386	0.299	0.982	0.372
Pulp Fiction	–	1.0	0.975	0.272	0.929
Iron Man	–	–	1.0	0.211	0.858
Forrest Gump	–	–	–	1.0	263
The Mummy	–	–	–	–	1.0

$sim_{cv}(u,v)$	Joe	Ann	Mary	Steve
Joe	1.0	0.283	0.372	0.962
Ann	–	1.0	0.915	0.232
Mary	–	–	1.0	0.254
Steve	–	–	–	1.0

Similarity measures (item recommendation): Example

$sim_{cv}(i,j)$	Titanic	Pulp Fiction	Iron Man	Forrest Gump	The Mummy
Titanic	1.0	0.87	0.67	0.82	0.67
Pulp Fiction	–	1.0	0.87	0.71	0.87
Iron Man	–	–	1.0	0.41	0.67
Forrest Gump	–	–	–	1.0	0.41
The Mummy	–	–	–	–	1.0

$sim_{cv}(u,v)$	Joe	Ann	Mary	Steve
Joe	1.0	0.75	0.75	0.87
Ann	–	1.0	0.75	0.58
Mary	–	–	1.0	0.58
Steve	–	–	–	1.0

Mean-Centered Prediction[1]

rating prediction

- user-based

$$\hat{\phi}_{ui} = \overline{\phi}_u + \frac{\sum_{v \in \mathcal{N}_i^{u,k}} sim(u,v) \cdot (\phi_{vi} - \overline{\phi}_v)}{\sum_{v \in \mathcal{N}_i^{u,k}} |sim(u,v)|}$$

- item-based

$$\hat{\phi}_{ui} = \overline{\phi}_i + \frac{\sum_{j \in \mathcal{N}_u^{i,k}} sim(i,j) \cdot (\phi_{uj} - \overline{\phi}_j)}{\sum_{v \in \mathcal{N}_u^{i,k}} |sim(i,j)|}$$

item recommendation

- user-based

$$\hat{\phi}_{ui} = \frac{\sum_{v \in \mathcal{N}_i^{u,k}} sim(u,v)}{k}$$

- item-based

$$\hat{\phi}_{ui} = \frac{\sum_{j \in \mathcal{N}_u^{i,k}} sim(i,j)}{k}$$

[1] Simplified notation: $\hat{\phi}(u,i) \rightsquigarrow \hat{\phi}_{ui}$

Prediction: Example

rating prediction using two most similar users according to sim_{pc}

- $\mathcal{U}_{Titanic} = \{Joe, Ann, Mary\}$, $\mathcal{N}^{Steve,2}_{Titanic} = \{Mary, Ann\}$
- $\overline{\phi}_{Steve} = \frac{11}{3} = 3.67$, $\overline{\phi}_{Mary} = \frac{12}{4} = 3$, $\overline{\phi}_{Ann} = \frac{13}{4} = 3.25$
- $\hat{\phi}_{ST} = \overline{\phi}_S + \frac{s_{pc}(S,M)\cdot(\phi_{MT} - \overline{\phi}_M) + s_{pc}(S,A)\cdot(\phi_{AT} - \overline{\phi}_A)}{|s_{pc}(S,M)| + |s_{pc}(S,A)|} = 3.67 + \frac{0.6\cdot(4-3)+0.565\cdot(5-3.25)}{0.6+0.565} = 1.36$

rating prediction using two most similar items according to sim_{pc}

- $\mathcal{I}_{\underline{S}teve} = \{\underline{P}ulp\ Fiction, \underline{I}ron\ Man, The\ \underline{M}ummy\}$, $\mathcal{N}^{\underline{T}itanic,2}_{Steve} = \{\underline{I}ron\ Man, The\ \underline{M}ummy\}$
- $\overline{\phi}_T = \frac{10}{3} = 3.34$, $\overline{\phi}_I = \frac{11}{3} = 3.67$, $\overline{\phi}_M = \frac{9}{3} = 3$
- $\hat{\phi}_{ST} = \overline{\phi}_T + \frac{s_{pc}(T,I)\cdot(\phi_{SI} - \overline{\phi}_I) + s_{pc}(T,M)\cdot(\phi_{SM} - \overline{\phi}_M)}{|s_{pc}(T,I)| + |s_{pc}(T,M)|} = 3.34 + \frac{-.815\cdot(4-3.67)-.581\cdot(4-3)}{0.815+0.581} = 2.73$

item recommendation – two most similar users

- $\mathcal{N}^{Steve,2}_{Titanic} = \{Joe, Ann\}$, $\hat{\phi}_{ST} = \frac{s_{cv}(S,J)+s_{cv}(S,M)}{2} = \frac{0.87+0.58}{2} = 0.725$
- $\mathcal{N}^{Steve,2}_{Titanic} = \{Ann, Mary\}$, $\hat{\phi}_{ST} = \frac{s_{cv}(S,A)+s_{cv}(S,M)}{2} = \frac{0.58+0.58}{2} = 0.58$

item recommendation – two most similar items

- $\mathcal{N}^{Titanic,2}_{Steve} = \{PulpFiction, IronMan\}$, $\hat{\phi}_{ST} = \frac{s_{cv}(T,P)+s_{cv}(T,I)}{2} = \frac{0.87+0.67}{2} = 0.77$
- $\mathcal{N}^{ForrestGump,2}_{Steve} = \{PulpFiction, IronMan\}$, $\hat{\phi}_{ST} = \frac{s_{cv}(F,P)+s_{cv}(F,I)}{2} = \frac{0.71+0.41}{2} = 0.56$

Advantages of Neighborhood-based Recommendations

Simple

- intuitive and simple to implement, only one parameter to tune

Justifiable

- easy to users to understand recommendations

Efficient

- can be speeded-up e.g. by pre-computing nearest neighbors

Stable

- a small amount of new users, items and ratings affect the performance just a little

Serendipity

- ability to recommend an interesting item for a user which he might not have otherwise discovered
  - can be helpful in finding new type or class of interesting items

Matrix Factorization: A Model-based Approach

Latent space representation

- The idea is to map users and items to a common space, where the co-ordinates represent **latent factors**.
  - user's interests and item's implicit properties are both incorporated in ("expressed by") latent factors
    - e.g. amount of action/romance or orientation, in case of movies, ...

The task[1] of Matrix Factorization is to approximate the matrix[2] $\Phi^{n\times m}$ by the matrix $\hat{\Phi}^{n\times m}$ which is a product WH^T of two (smaller) matrices $W^{n\times k}$ and $H^{m\times k}$

$$\hat{\phi}_{ui} = w_u h_i^T = \sum_{k=1}^{K} w_{uk} h_{ik}$$

where K is the number of latent factors.

[1] Note, that there are several methods for Matrix factorization/decomposition, we'll discuss only the one most commonly used in recommender systems.

[2] $\Phi(u,i) = \phi_{ui}$

Training the Model for Rating Prediction

We would like **to minimalize**[1] **the squared error** of approximation

$$error = \sum_{(u,i) \in D} e_{ui}^2 = \sum_{(u,i) \in D} (\phi_{ui} - \hat{\phi}_{ui})^2 = \sum_{(u,i) \in D} (\phi_{ui} - w_u h_i^T)^2$$

Moreover, we add **regularization**[2] terms to the error function to prevent overfitting

$$error = \sum_{(u,i) \in D} (\phi_{ui} - w_u h_i^T)^2 + \lambda(\|W\|^2 + \|H\|^2)$$

where $\lambda \geq 0$ is a regularization term.

[1] When the model is trained, we have to minimalize the error on the training set, i.e. on the past user-item-interactions.

[2] To prevent Φ containing large numbers.

MF via Stochastic Gradient Descent

Training is an optimalizaiton problem of minimizing the **objective function** *error* with parameters W, H and a hyper-parameter λ.

- **updating** parameters **iteratively** for each data point ϕ_{ui} in the opposite direction of the **gradient** of the objective function at the given point until a **convergence** criterion is fulfilled.
  - updating the vectors w_u and h_i for the data point $(u, i) \in D$

$$\frac{\partial error}{\partial w_u}(u, i) = -2(e_{ui}h_i - \lambda w_u)$$

$$\frac{\partial error}{\partial h_i}(u, i) = -2(e_{ui}w_u - 2\lambda h_i)$$

$$w_u^{new}|u, i = w_u^{old} - \alpha \frac{\partial error}{\partial w_u}(u, i) = w_u^{old} + \alpha(e_{ui}h_i^{old} - \lambda w_u^{old})$$

$$h_i^{new}|u, i = h_i^{old} - \alpha \frac{\partial error}{\partial h_i}(u, i) = h_i^{old} + \alpha(e_{ui}w_u^{old} - \lambda h_i^{old})$$

where $\alpha > 0$ is a **learning rate**.

MF via SGD: Algorithm

Hyper-parameters: iter (the maximal number of iterations), $\alpha, \lambda, \sigma^2$
$W \leftarrow$ initialize with $\mathcal{N}(0, \sigma^2)$
$H \leftarrow$ initialize with $\mathcal{N}(0, \sigma^2)$
for $iter \leftarrow 1, \ldots, iter \cdot |D|$ **do**
    draw randomly (u, i) from D
    $\hat{\phi}_{ui} \leftarrow 0$
    for $k \leftarrow 1, \ldots, K$ **do**
        $\hat{\phi}_{ui} \leftarrow \hat{\phi}_{ui} + W[u][k] \cdot H[i][k]$
    end for
    $e_{ui} = \phi_{ui} - \hat{\phi}_{ui}$
    for $k \leftarrow 1, \ldots, K$ **do**
        $W[u][k] \leftarrow W[u][k] + \alpha \cdot (e_{ui} \cdot H[i][k] - \lambda \cdot W[u][k])$
        $H[i][k] \leftarrow H[i][k] + \alpha \cdot (e_{ui} \cdot W[u][k] - \lambda \cdot H[i][k])$
    end for
end for
return $\{W, H\}$

MF via SGD: Example[2]

$\Phi =$

1	4	5		3
5	1		5	2
4	1	2	5	
	3	4		4

Let's have the following hyper-parameters:
$K = 2,\ \alpha = 0.1,\ \lambda = 0.15,\ iter = 150,\ \sigma^2 = 0.01$

Results[1] are:

$W =$

1.1995242	1.1637173
1.8714619	-0.02266505
2.3267753	0.27602595
2.033842	0.539499

$H^T =$

1.6261001	1.1259034	2.131041	2.2285593	1.6074764
-0.40649664	0.7055319	1.0405376	0.39400166	0.49699315

$\hat{\Phi} =$

1.477499	2.171588	3.767126	3.131717	2.506566
3.052397	2.091094	3.964578	4.161733	2.997066
3.671365	2.814469	5.245668	5.294111	3.877419
3.087926	2.670543	4.895569	4.745101	3.537480

[1] Note, that these hyper-parameters are just picked up in an ad-hoc manner. One should search for the "best" hyper-parameter combinations using e.g. grid-search (a brute-force approach).

[2] Thanks to my colleague Thai-Nghe Nguyen for computing examples.

Biased Matrix Factorization via SGD

user bias $\overline{\phi}_u$ (**item bias** $\overline{\phi}_i$) measures how do ratings of the user u (ratings for the item i) differs from the **global average rating** $\overline{\phi}$.

- the "biased" prediction is $\hat{\phi}_{ui} = \overline{\phi} + \overline{\phi}_u + \overline{\phi}_i + w_u \cdot h_i$

The error function to minimize became

$$error = \sum_{(u,i) \in D} (\phi_{ui} - \overline{\phi} - \overline{\phi}_u - \overline{\phi}_i - w_u \cdot h_i)^2 + \lambda(\|W\|^2 + \|H\|^2 + \overline{\phi}_u^2 + \overline{\phi}_i^2)$$

Updates additional to w_u and h_i are

$$\overline{\phi}_u^{new}|u,i = \overline{\phi}_u^{old} - \alpha \frac{\partial error}{\partial \overline{\phi}_u}(u,i) = \overline{\phi}_u^{old} + \alpha(e_{ui} - \lambda \overline{\phi}_u^{old})$$

$$\overline{\phi}_i^{new}|u,i = \overline{\phi}_i^{old} - \alpha \frac{\partial error}{\partial \overline{\phi}_i}(u,i) = \overline{\phi}_i^{old} + \alpha(e_{ui} - \lambda \overline{\phi}_i^{old})$$

Biased MF with SGD: Algorithm

Hyper-parameters: iter (the maximal number of iterations), $\alpha, \lambda, \sigma^2$
$W \leftarrow$ initialize with $\mathcal{N}(0, \sigma^2)$
$H \leftarrow$ initialize with $\mathcal{N}(0, \sigma^2)$
$\overline{\phi} \leftarrow$ initialize with the global average
for $u \leftarrow 1, \ldots, |U|$ **do**
    $\overline{\phi}_u[u] \leftarrow$ average rating of user u
end for
for $i \leftarrow 1, \ldots, |I|$ **do**
    $\overline{\phi}_i[i] \leftarrow$ average rating of item i
end for
for $iter \leftarrow 1, \ldots, iter \cdot |D|$ **do**
    draw randomly (u, i) from D
    $\hat{\phi}_{ui} \leftarrow \overline{\phi} + \overline{\phi}_u[u] + \overline{\phi}_i[i]$
    for $k \leftarrow 1, \ldots, K$ **do**
        $\hat{\phi}_{ui} \leftarrow \hat{\phi}_{ui} + W[u][k] \cdot H[i][k]$
    end for
    $e_{ui} = \phi_{ui} - \hat{\phi}_{ui}$
    $\overline{\phi}_u^{new}[u] \leftarrow \overline{\phi}_u^{old}[u] + \alpha \cdot (e_{ui} - \lambda \cdot \overline{\phi}_u^{old}[u])$
    $\overline{\phi}_i^{new}[i] \leftarrow \overline{\phi}_i^{old}[i] + \alpha \cdot (e_{ui} - \lambda \cdot \overline{\phi}_i^{old}[i])$
    for $k \leftarrow 1, \ldots, K$ **do**
        $W[u][k] \leftarrow W[u][k] + \alpha \cdot (e_{ui} \cdot H[i][k] - \lambda \cdot W[u][k])$
        $H[i][k] \leftarrow H[i][k] + \alpha \cdot (e_{ui} \cdot W[u][k] - \lambda \cdot H[i][k])$
    end for
end for
return $\{W, H, \overline{\phi}_u, \overline{\phi}_i\}$

Biased MF with SGD: Example

Φ is the same as in the previous case.

Let's have the following hyper-parameters:

$K = 2$, $\alpha = 0.1$, $\lambda = 0.15$, $iter = 1000$, $\sigma^2 = 0.01$ Results are:

$W =$

-1.2818109	0.8797541
0.8263778	-0.658325
0.5540779	-0.37631336
0.48018292	-0.24728496

$H^T =$

1.3833797	-0.81226087	-0.82310724	0.122659974	-0.06878678
-0.9954762	0.51703054	0.5780823	-0.074271396	0.15422797

$WH^T =$

-2.649005	1.496024	1.563638	-0.222567	0.223854
1.798541	-1.011608	-1.060763	0.150258	-0.158375
1.141111	-0.644621	-0.673605	0.095912	-0.096151
0.910441	-0.517887	-0.538193	0.077265	-0.071168

$$\overline{\phi} = 3.2666667$$

$$\hat{\overline{\phi}}_u = (0.09477682, -0.45755777, -0.6332871, 1.2168586)$$

$$\hat{\overline{\phi}}_i = (0.3055541, -0.8959325, 0.04974971, 2.1113703, -0.548792)$$

$\hat{\phi}(Steve, Titanic) = 3.2666667 + 1.2168586 + 0.3055541 + 0.910441 = 5.6995204$

Summary

User feedback

- implicit vs. explicit

Evaluation measures for recommender systems Different approaches

- demographic, knowledge- and utility-based
  - quite old, need to maintain the knowledge-base, expert needed
- content-based
  - often is expensive to get item features
  - useful when the system faces with new user
- collaborative filtering
  - successful approach, no need for item features
  - "cold-start" problem
  - Matrix Factorization with Stochastic Gradient Descent
    - easy to implement
    - works well for sparse data
    - need to search for hyper-parameters

Many other, different techniques!

References

- D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. SIGIR Forum 37, 2 (September 2003), 18-28.
- F. Ricci, L Rokach, B. Shapira, P. B. Kantor: Recommender Systems Handbook. Springer, 2011, ISBN 978-0-387-85819-7, pp. 842.
- Y. Koren, R. Bell, Ch. Volinsky: Matrix Factorization Techniques for Recommender Systems. COMPUTER, 2009, IEEE Computer Society, p. 42-29.

Thanks for Your attention!

Questions?

horvath@ismll.de