

Functional matrix factorizations for cold-start recommendation

College of Computing Georgia Institute of Technology



psu.edu

<https://citeseerx.ist.psu.edu> › viewdoc › download PDF

Functional Matrix Factorizations for Cold-Start ... - CiteSeerX

K Zhou 저술 · 2011 · 335회 인용 — In this paper, we present **functional matrix factorization (fMF)**, a novel **cold-start recommendation** method that solves the problem of initial interview ...

목차

1. Matrix Factorization Recommender Systems and Cold Start
2. Abstract
3. Introduction
4. Related Work
5. Functional Matrix Factorization
6. Experiments
7. Concluding Remarks

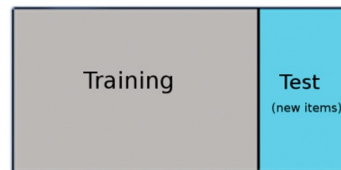
Matrix Factorization Recommender Systems and Cold Start

- Cold Start Problem Overview
 - How to give ratings for new users and new items for which no ratings are recorded in the dataset?
- User Cold Start
- Item Cold Start



Functional Matrix Factorization [3]

- Hybrid MF-Decision Tree Recommender System
- Uses MF to find best queries to estimate user profiles



Local Collective Embeddings [4]

- Uses collective factorization to link user ratings to item text.

Abstract

- A key challenge in recsys is how to effectively profile new users (a.k.a cold-start rec)
- 최근(옛날임)에는 initial interview 를 통해서 user responses 를 querying 하여 유저의 선호도를 이끌어내고자 함
- 본 논문에서 function matrix factorization (fMF) 제안
 - A novel cold-start recommendation method **that solves the problem of initial interview construction within the context of learning user and item profiles**
 - 즉, user 와 item profiles 을 활용해서 초기 인터뷰를 구축하는 방법에 대해서 논함
 - fMF 는 decision tree 형태인데, 각 노드는 interview question 이고, user adaptively 하게 쿼리를 추천하는 형태
 - 또한 decision tree 구축 시, 각 노드를 latent profile 과 연관지어, profile 이 유저의 인터뷰 응답 과정을 통해 점진적으로 정제되도록 함
 - Decision tree construction 과 latent profiles extraction 이 번갈아가며 진행하는 반복적인 optimization 알고리즘 제안

Introduction

- A natural approach to solving the cold start problem
 - 초기 인터뷰의 유저 답변을 통해서 유저의 선호도를 명시적으로 확인
 - 초기 인터뷰는 고정된 아이템을 제공
 - 초기 인터뷰 답변을 통해 유저 profile 을 점진으로 구축해나감
 - 좋은 초기 인터뷰란?
 - should not be time-consuming
 - should be effective
 - should ask the interview questions adaptively

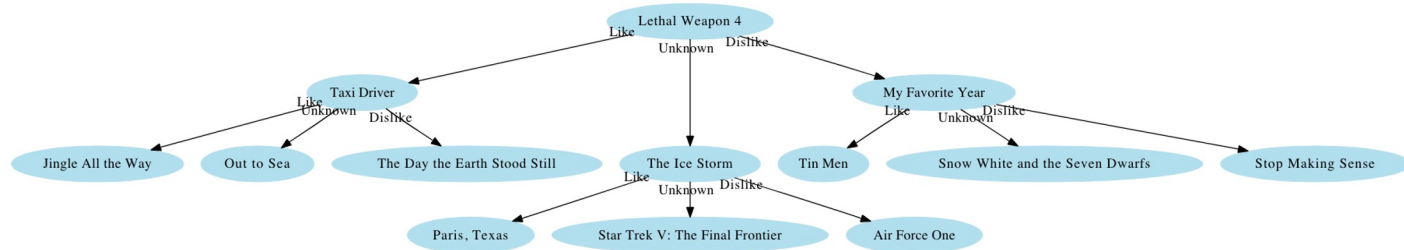


Figure 1: An example for decision trees: Top three levels of a model of depth 6 for MovieLens data set

Introduction

- fMF
 - 앞선 interview decision tree 를 구축하는 새로운 방법
 - Single framework : Matrix factorization model + decision tree
 - Initial interview 를 수행하는 function 은 decision tree 형태
 - Decision tree 와 user profile 은 학습되어야 하며, 이를 위해 iterative optimizaiton 알고리즘 제안함

Related Work

- Collaborative Filtering

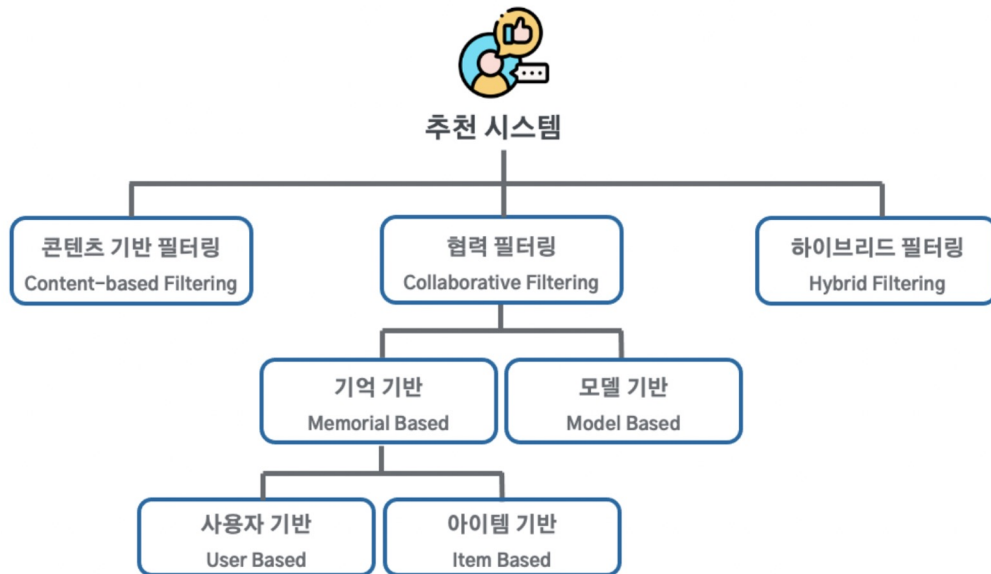
- Collaborative Filtering 이란?

- 사용자가 아이템에 매긴 평점, 상품 구매 이력 등의 사용자 **행동 양식 (user behavior)** 를 기반으로 추천 즉, user, item 의 상호작용 (구매, 평가 등) 의 이력을 바탕으로 추천 진행
- 예를 들어, A 에게 새로운 작품을 추천 해주고 싶을 때, A 와 비슷한 사람 B 가 좋은 점수를 매긴 작품을 추천

- CF 장 / 단점

- 😊 상호작용 (rating, 구매 빈도 ..) 데이터만 있으면 적용할 수 있음. 따라서 도메인에 의존되지 않고 사용 가능
- 😊 일반적으로 content based 보다 좋은 성능
- 😞 Cold start 문제. 새로운 item, user 에 대한 feedback (상호작용) 부족

- CF 종류 (가운데 협력 필터링 파트)



Related Work

- Cold Start Collaborative Filtering
 - cold start user 의 선호도를 알아내기 위한 다양한 노력이 있었음
 - A static seed 로 인터뷰 프로세스를 구축함 => 이 때, informativeness, popularity and coverage 측정
 - Greedy seed selection algorithm by optimizing the prediction performance
 - They do not reflect the user responses during the interview process
 - IGCN 알고리즘
 - 기타 등등
- 제안 모델은 한 단계 더 나아가 matrix factorization framework 와 decision tree 를 결합한 프레임워크 제안함

Functional Matrix Factorization

- Cold start collaborative filtering for constructing the interview process
- 핵심은, 초기 인터뷰 과정으로 user profiles 을 파라미터화 하는 것!
- **P possible interview questions**
 - Answer to a question takes value in the finite set $\{0,1,Unknown\}$
- **$a\{i\}$ denote the P dimensional vector** representing the answers of user i to the P questions
 - 즉, 학습 유저가 p 개 질문에 답변한 value, 학습 유저는 모든 질문에 대한 답을 가지고 있음
- 유저 i 의 $a\{i\}$ 를 T 함수(decision tree)에 태워 **user profile $u\{i\}$ 생성**
 - $u\{i\} = T(a\{i\})$, T is a function that maps the responses a_i to the user profile

Functional Matrix Factorization

assuming $u_i = T(a_i)$, where T is a function that maps the responses a_i to the user profile $u_i \in \mathbb{R}^k$. To make recommendations for user i , we simply use $r_{ij} = v_j^T T(a_i)$.

Our goal is to learn both T and v_j from the observed ratings \mathcal{K} . To this end, substituting $u_i = T(a_i)$ into the low-rank matrix factorization model, we have the following optimization problem:

$$T, V = \operatorname{argmin}_{T \in \mathcal{H}, V} \sum_{(i,j) \in \mathcal{O}} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|V\|^2, \quad (1)$$

- Decision tree $\mathbf{T}(\mathbf{a})$ 를 통해서 user profile 을 생성
- 이렇게 생성한 user profile 로 Matrix Factorization 수행
 - User profile $u\{i\}$, $v\{j\}$

Functional Matrix Factorization

- 인터뷰 프로세스를 구축하는 것에는 몇 가지 이슈가 있음
 1. 인터뷰 질문 수량은 적어야 함
 2. 인터뷰 과정은 유저 응답에 따라 adaptive 해야함
 3. 또한, 우리는 unknown 답변을 허용했기 때문에 이러한 missing value 도 함께 다룰 수 있어야 함
- T(a) 를 위해 Ternary decision tree 사용.
- Decision tree 의 각 노드는 하나의 interview question 에 해당하고, three child node 를 가짐.
- 최종적으로 유저는 인터뷰 프로세스 과정동안 root node 에서부터 leaf node 에 도착함
- User profile 은 leaf node 에서 추정됨 (이전까지의 응답을 기반으로)

Functional MAtrix Factorization

Alternative Optimization

- $T(a)$ 로 일단 어떻게든지 user profile 이 만들어짐. 그리고 item profile 은 초기에 random init 설정하고. User profile (latent factor) 과 item profile (latent factor) 를 matrix factorization + ALS 학습

- 1 Step

1. Given $T(a)$, we can compute v_j by regularized least square regression. Formally, for each j , we find v_j such that

$$v_j = \operatorname{argmin}_{v_j} \sum_{(i,j) \in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|v_j\|^2.$$

This problem has a closed-form solution given by

$$v_j = \left(\sum_{(i,j) \in O} T(a_i)T(a_i)^T + \lambda I \right)^{-1} \left(\sum_{(i,j) \in O} r_{ij} T(a_i) \right), \quad (2)$$

where I is the identity matrix of appropriate size.

Functional Matrix Factorization

Alternative Optimization

- $T(a)$ 로 일단 어떻게든지 user profile 이 만들어짐. 그리고 item profile 은 초기에 random init 설정하고. User profile (latent factor) 과 item profile (latent factor) 를 matrix factorization + ALS 학습

- 2 Step

2. Given v_j , we try to fit a decision tree T such that

$$T = \operatorname{argmin}_{T \in \mathcal{H}} \sum_{(i,j) \in O} (r_{ij} - T(a_i)^T v_j)^2. \quad (3)$$

- Tree 가 깊어지면 exponentially 하게 수가 많아짐
- Global optimal solution 을 찾기는 too extensive
- 이를 해결하기 위해서 greedy algorithm 을 통해서 계산함

Functional MAtrix Factorization

Decision Tree Construction

- Decision Tree 는 어떻게 구축하고, 어떻게 user profile 을 생성하는가?
 - 각 노드는 다음 수식을 최소로 하는 **최적의 질문을 선택한다**.

$$\begin{aligned} R_L(p) &= \{i | a_{ip} = \text{"Like"}\}, \\ R_D(p) &= \{i | a_{ip} = \text{"Dislike"}\}, \\ R_U(p) &= \{i | a_{ip} = \text{"Unknown"}\}. \end{aligned}$$

질문 p 에 대해
User 답변에 따라 세 그룹으로 그룹핑

To find the optimal question p that leads to the best split, we minimize the following objective:

$$\begin{aligned} \min_p \quad & \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u_L^T v_j)^2 + \sum_{i \in R_D(p)} \sum_{(i,j) \in O} (r_{ij} - u_D^T v_j)^2 \\ & + \sum_{i \in R_U(p)} \sum_{(i,j) \in O} (r_{ij} - u_U^T v_j)^2, \end{aligned} \quad (4)$$

where u_L , u_D and u_U are the optimal profiles for users in the child nodes corresponds to the answers of "Like", "Dislike" and "Unknown", respectively:

$$\begin{aligned} u_L &= \argmin_u \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2, \\ u_D &= \argmin_u \sum_{i \in R_D(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2, \\ u_U &= \argmin_u \sum_{i \in R_U(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2. \end{aligned}$$

Functional Matrix Factorization

Decision Tree Construction

Algorithm 1 Alternating Optimization for Functional Matrix Factorization

Require: The training data $\mathcal{K} = \{r_{ij} \mid (i, j) \in O\}$.

Ensure: Estimated decision tree $T(a)$ and item profiles $v_j, j = 1, 2, \dots, M$.

- 1: Initialize v_j randomly for $j = 1, \dots, M$.
 - 2: **while** not converge **do**
 - 3: Fit a decision tree $T(a)$ using Algorithm 2.
 - 4: Update v_j with Equation (2).
 - 5: **end while**
 - 6: **return** $T(a)$ and $v_j, j = 1, 2, \dots, M$.
-


Algorithm 2 Greedy Decision Tree Construction

- 1: **function** FitTree(UserSet)
 - 2: // UserSet: the set of users in current node.
 - 3: Calculate u_L, u_D, u_U by Equation (5), (6) and (7) for $p = 1, \dots, P$.
 - 4: Compute the split criteria L_p by Equation (4) for $p = 1, \dots, P$.
 - 5: Find the best interview question $p = \operatorname{argmax}_p L_p$.
 - 6: Split user into three groups $R_L(p), R_D(p)$ and $R_U(p)$.
 - 7: **if** square error reduces after split **and** depth < maxDepth **then**
 - 8: Call FitTree($R_L(p)$), FitTree($R_D(p)$) and
 FitTree($R_U(p)$) to construct the child nodes.
 - 9: **end if**
 - 10: **return** $T(a)$.
 - 11: **end function**
-

Functional Matrix Factorization

Hierarchical Regularization

- Overfitting 방지하기 위함
 - 특히 노드 아래로 내려갈 수록 유저 수는 작아지기 때문에 overfitting 가능성이 높음
- 본 논문에서는 decision tree 구조를 활용한 hierarchical regularization 사용
 - $u_{\{c\}}$ is the estimation at the current node
 - $u_{\{L\}}$ is the estimation at its child node corresponding to the answer Like


$$u_L = \underset{u}{\operatorname{argmin}} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2 + \lambda_h \|u - u_C\|^2,$$

Experiments

- Datasets
 - MovieLens, EachMovie and Netflix
- **The MovieLens1 data set** contains 3900 movies, 6040 users and about 1 million ratings. In this data set, about 4% of the user-movie dyads are observed. The ratings are integers ranging from 1 (bad) to 5 (good)
- **The EachMovie data set**, collected by HP/Compaq Research, contains about 2.8 million ratings for 1628 movies by 72,916 users. The ratings are integers ranging from 1 to 6
- **The Netflix2** is one of the largest test bed for collaborative filtering. It contains over 100 million ratings and was split into one training and one test subsets. The training set consists of 100,480,507 ratings for 17,770 movies by 480,189 users. The test set contains about 2.8 million ratings. All the ratings are ranged from 1 to 5

Experiments

- Datasets
 - MovieLens, EachMovie and Netflix
- Ratings in 1-5 scale, assume that the responses a_{ij} of user i to the the question “Do you like item j ?” can be inferred from her rating r_{ij} as follows:

$$a_{ij} = \begin{cases} 0, & \text{if } (i, j) \in O \text{ and } r_{ij} \leq 3 \\ 1, & \text{if } (i, j) \in O \text{ and } r_{ij} > 3 \\ \text{"Unknown"}, & \text{if } (i, j) \notin O \end{cases}$$

Experiments

- Experiment Design
 1. How does the proposed algorithm perform?
i.e. How effective is the interview process?
 2. How do missing values in user ratings impact the performance of the proposed algorithm?
 3. How does fMF perform in warm-start settings compared to traditional collaborative filtering methods such as plain matrix factorization?
 4. How do the parameters impact the performance of the proposed model?

Experiments

Cold-start Performance

- Train 75%
- Test 25%
 - Answer set 75%
 - Evaluation set 25%

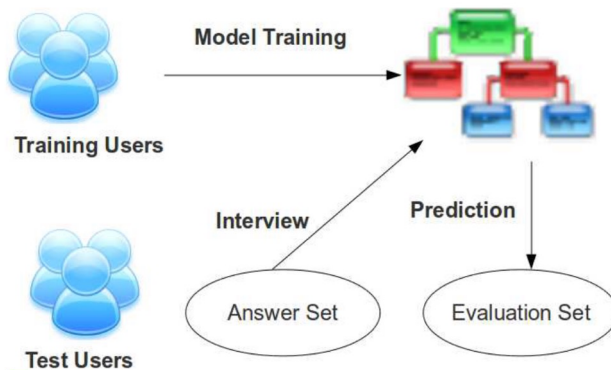


Figure 2: Evaluation process for cold-start users

Experiments

Cold-start Performance (Results)

Table 2: RMSE on MovieLens data set for cold-start users with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	0.9509	0.9480	0.9437	0.9429	0.9410
Tree	0.9767	0.9683	0.9615	0.9512	0.9523
TreeU	0.9913	0.9887	0.9789	0.9690	0.9657

Table 3: RMSE on EachMovie data set for cold-start users with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	1.2853	1.2717	1.2715	1.2691	1.2673
Tree	1.2989	1.2842	1.2800	1.2779	1.2750
TreeU	1.3134	1.3009	1.2928	1.2907	1.2905

Table 4: RMSE on Netflix data set for cold-start user with respect to the number of interview questions

No. Questions	3	4	5	6	7
fMF	0.9205	0.9189	0.91582	0.9123	0.9121
Tree	0.9320	0.9252	0.9239	0.9228	0.9219
TreeU	0.9422	0.9382	0.9359	0.9322	0.9323

Table 5: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Dislike
2	Armageddon	Like
3	Maid in Manhattan	Dislike
4	Reservoir Dogs	Like
5	Collateral Damage	Dislike
6	2 Fast 2 Furious	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	Lord of the Rings: The Return of the King
2	Lord of the Rings: The Two Towers
3	Mobile Suit Gundam: Char's Counterattack
4	Star Wars: Episode V: The Empire Strikes Back
5	Raiders of the Lost Ark

Table 6: Examples of interview process with 6 questions

(a) Interview process

Round	Question	Response
1	Being John Malkovich	Like
2	Armageddon	Dislike
3	What Women Want	Dislike
4	The Royal Tenenbaums	Unknown
5	Pretty Woman	Like
6	Cats: The Ultimate Edition	Unknown

(b) Top-5 recommendations

Rank	Movie Title
1	The Shawshank Redemption
2	Schindler's List
3	Sex and the City: Season 5
4	Sex and the City: Season 4
5	The Usual Suspects

Experiments

Cold-start Performance (Results)

- The impact of Non-responses

- 실험에서는 유저가 해당 영화에 점수를 주지 않았을 때 Unknown 이라고 가정했지만, 실제로 학습 데이터에서는 그렇지 않을수도 있다. 실제 유저는 영화를 알고도 rating 을 주지 않았을 수 있어서. 이런 경우, 해당 유저는 초기 인터뷰에서는 해당 영화를 like, unlike 선택할 것이다.

○ 이러한 missing rate 이 초기 인터뷰 프로세스에 얼마나 영향을 주었는지 분석

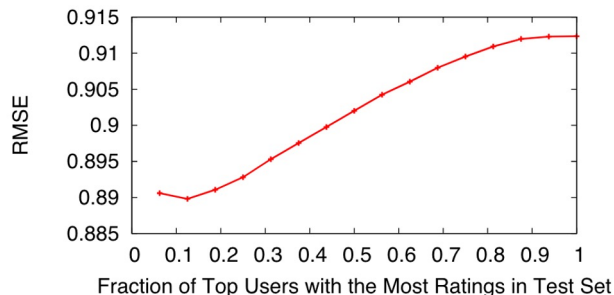


Figure 3: Performance measured by RMSE on Netflix data set with respect to the fraction of users with the most known ratings.

- 평점이 적은 사용자를 포함할수록 RMSE 증가
- 옆의 그래프 : 유저를 평점이 많은 수 대로 내림차순 한 후, 위에서부터 0.1, 0.2 , ... 1. 즉, 0.1 구간에서 평점이 많은 유저임
- 평점이 많을수록 unknown 을 선택할 가능성이 적을것이고, 학습 데이터에서 누락된 값으로 인해 bias 의 영향을 덜 받아 성능이 좋을 것

Experiments

Cold-start Performance (Results)

- The impact of Non-responses
 - M1 : 20%, M2: 40%, .., M5: 100%

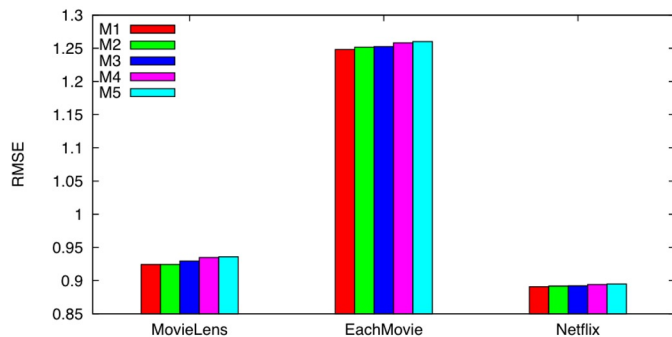


Figure 4: Performance measured by RMSE on three data sets with respect to the fraction of users with the most known ratings.

- 학습 데이터에 Unknown value (missing value)
가 많을수록 test set 의 RMSE 가 높아진다.

Experiments

Cold-start Performance (Results)

- Warm-Start Performance
 - Warm-start problem 에 적용시킨 후, cold-start method 와 warm-start method 의 상대적인 성능 비교 (Tree, TreeU, fMF)
 - We can observe that all the cold-start methods perform worse than the matrix factorization method
 - The goal of the cold-start algorithms is to provide cold-start users with reasonable predictions **within a few quick interview questions.**

Table 7: RMSE on MovieLens data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9240	0.9216	0.9190	0.9134	0.9098
Tree	0.9458	0.9439	0.9409	0.9321	0.9329
TreeU	0.9906	0.9850	0.9771	0.9706	0.9728
MF	0.8702				

Table 8: RMSE on EachMovie data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	1.2548	1.2499	1.2449	1.2366	1.2226
Tree	1.2709	1.2614	1.2664	1.2570	1.2549
TreeU	1.28742	1.2813	1.2790	1.2772	1.2751
MF	1.1790				

Table 9: RMSE on Netflix data set in warm-start setting

No. Questions	3	4	5	6	7
fMF	0.9770	0.9749	0.9721	0.9715	0.9703
Tree	0.9772	0.9761	0.9726	0.9717	0.9713
TreeU	0.9914	0.9879	0.9842	0.9792	0.9752
MF	0.9399				

Experiments

Cold-start Performance (Results)

- Warm-Start Performance

- 그러나, cold-start problem 에서 depth 를 제한하지 않고 늘린다면, MF와 성능이 비슷해짐을 확인
- 또한 fMF user profile 이 MF user profile 과 비슷해짐
- 이 결과로, cold-start problem 에서 성능이 비슷한 것

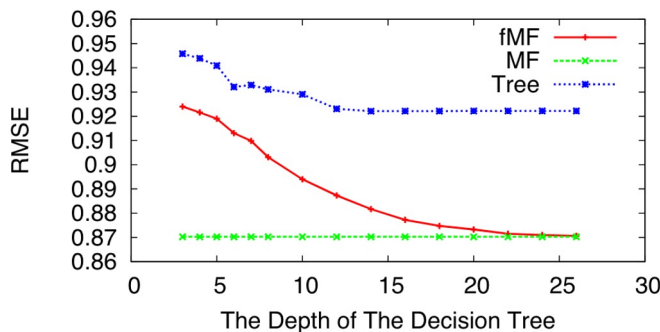
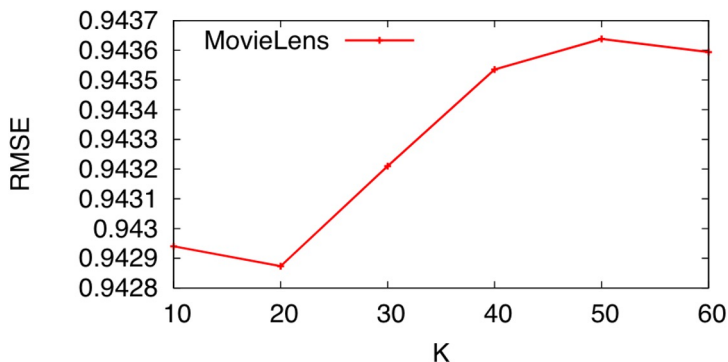


Figure 5: RMSE of fMF, Tree and MF on MovieLens data set with respect to the number of interview questions

Experiments

Cold-start Performance (Results)

- Impact of Model Parameters
 - parameter K : dimension of the user profiles and item profiles (default : 6)



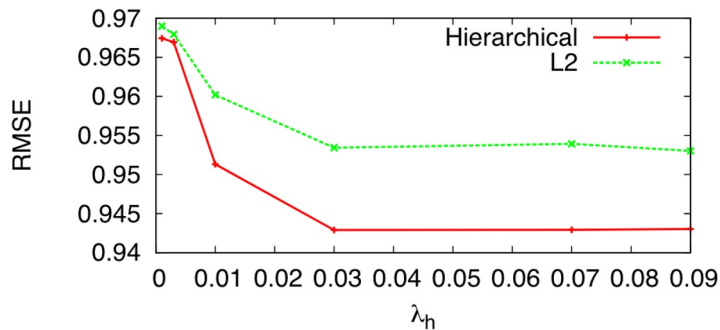
$K=20$ 까지는 optima 를 찾아가다가,
그 이상부터는 overparameterized 됨

Figure 6: Performance measured by RMSE with respect to different values of K on MovieLens data set. The performance is reported by setting the depth of the decision tree to be 6.

Experiments

Cold-start Performance (Results)

- Hierarchical regularization to avoid overfitting (parameter λ_h)



Decisions tree 에서는 하이어나키컬 정규화가 L2 정규화보다는 성능이 더 좋음.

Figure 7: The performance of hierarchical regularization. The performance is reported by setting the depth of the decision tree to be 6.

Experiments

Cold-start Performance (Results)

- Regularization weight λ for the item profiles $v_{\{j\}}$

$$v_j = \left(\sum_{(i,j) \in O} T(a_i)T(a_i)^T + \lambda I \right)^{-1} \left(\sum_{(i,j) \in O} r_{ij}T(a_i) \right), \quad (2)$$

where I is the identity matrix of appropriate size.

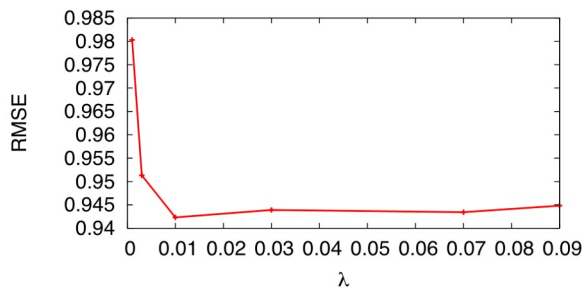


Figure 8: Performance measured by RMSE with respect to different values of λ on MovieLens data set.

The model has a high risk of overfitting if λ is too small.

Experiments

Cold-start Performance (Results)

- Iterative process

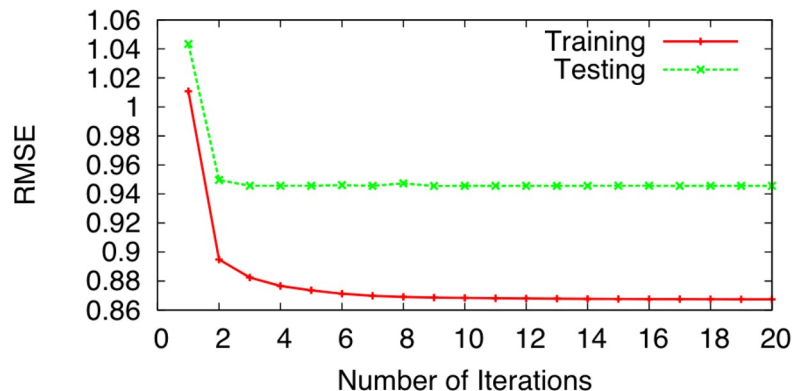


Figure 9: Performance measured by RMSE with respect to the number of iterations on MovieLens data set.

Algorithm 1 Alternating Optimization for Functional Matrix Factorization

Require: The training data $\mathcal{K} = \{r_{ij} \mid (i, j) \in \mathcal{O}\}$.

Ensure: Estimated decision tree $T(a)$ and item profiles

$v_j, j = 1, 2, \dots, M$.

1: Initialize v_j randomly for $j = 1, \dots, M$.

2: **while** not converge **do**

3: Fit a decision tree $T(a)$ using Algorithm 2.

4: Update v_j with Equation (2).

5: **end while**

6: **return** $T(a)$ and $v_j, j = 1, 2, \dots, M$.

The algorithm converges very quickly, usually within 5 iterations.

Concluding remarks

- 본 논문은 cold-start problem 해결에 초점을 맞춘 fMF 프레임워크 제안
- 제안된 fMF 알고리즘은 MF 와 decision tree 기반으로 user/item profile 을 학습하며, 신규 인터뷰를 구축함
- fMF 는 basic MF 를 기반으로 하고 있기 때문에, demographical information 과 같은 content feature 는 사용하지 않음. 따라서 future works 로 content feature 를 사용하여 fMF 성능 향상을 기대할 수 있음
- 또한 missing value problem 문제를 해결하고자 함

Thank You