Ranking and Recommendation

LambdaMF: Learning Nonsmooth Ranking Function in Matirx Factorization Using Lambda

By M. Parsa Mohammadian

CLiMF: Learning to Maximize Resiprocal Rank with Collabrative Less-is-more Filterring

By Vanessa Zydorek

6. December 2016

Motivation on Topic: Learning to rank(LTR)”

- **LTR** is the application of ML in the construction of ranking models for information retrieval(IR) systems
- **IR** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
- In **Recommendation System**: for identifying a ranked list of related items to recommend to a user after he or she has seen current items
- The problem is to rank, that is, sort, the documents according to some criterion so that the "best" results appear early in the result list displayed to the user.
- Ranking criteria are phrased in terms of relevance of documents with respect to an information need expressed in the query.

Motivation on Topic(Cont.)

- **Relevance** denote how well a retrieved document or set of documents meets the information need of the user.
- **Training data** is used by a learning algorithm to produce a ranking model which computes relevance of documents for actual queries.
- **Evaluation measures:** There are several measures which are commonly used to judge how well an algorithm is doing on training data and to compare performance of different LTR algorithms such as Normalized Discounted cumulative gain (NDCG)

Learning to rank in CF

- Pointwise approach $f(user, item) \rightarrow \mathbb{R}$
  - ➢ Reduce Ranking to Regression, Classification, or Ordinal Regression problem

- Pairwise approach $f(user, item_1, item_2) \rightarrow \mathbb{R}$
  - ➢ Reduce Ranking to pair-wise classification

- Listwise approach $f(user, item_1, item_2, \ldots, item_n) \rightarrow \mathbb{R}$
  - ➢ Direct optimization of IR measures, List-wise loss minimization

Ranking and Recommendation

LambdaMF: Learning Nonsmooth Ranking Function in Matirx Factorization Using Lambda

By M. Parsa Mohammadian

6. December 2016

Outline

Motivation

- Creating a new matrix factorization algorithm modeling the gradient of ranking measure
- Proving that some ranking-oriented matrix factorization model would suffer from unlimited norm growing due to popular item effect
- Introducing LambdaMF as a novel regularization term to deal with the popular item effect on recommendation systems
- Conducting experiments to show that the model is scalable to large recommendation dataset and outperforms several state-of-the-art models

Introduction

- Ranking measures which used commonly are discontinuous
- Usage of common gradient descent optimization techniques leads to property of zero and non-optimal solutions.

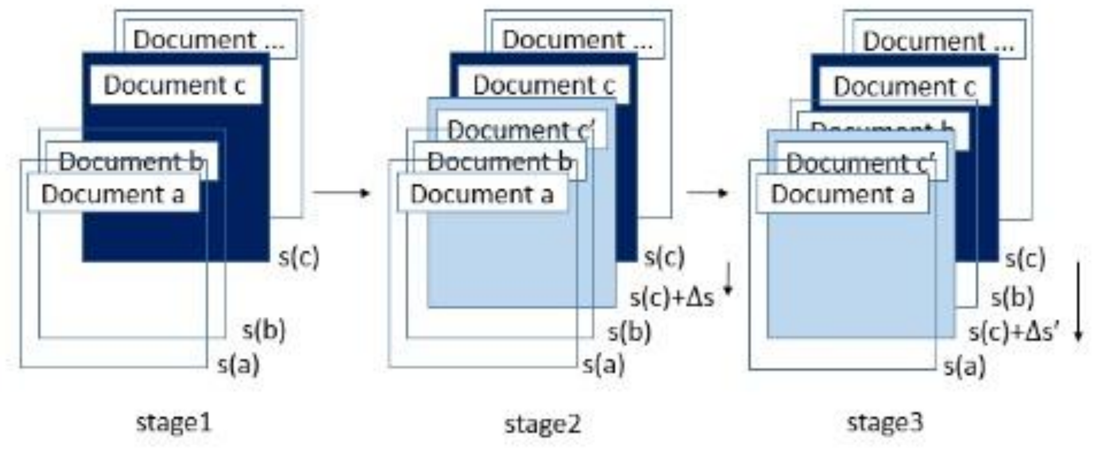


Figure 1. Ranking based on different model score

Handling non-optimal solutions

- Designing new loss function for optimization
- Mismatch between the loss function and the ranking measure would result in a gap between the optimal and learned models
- Using lambda-based methods to bypass converting ranking measures to loss function
- Designing a new recommendation model named Lambda Matrix Factorization (LambdaMF)
- LambdaMF as first ranking matrix factorization model to achieve global training optimum of a ranking measure

“Learning to rank(LTR)”

- Formulated with queries Q, documents D and relevance levels R
- A feature vector for each document-query pair as the focus is on optimizing the ranking measure for each query
- Essential difference from a recommendation: No queries or features in a collaborative filtering
- Evaluating the results of top-N recommendation by correlating users to queries and items to documents.
- Using the same user to query and item to document analogy to describe LTR task

“Learning to rank(LTR)”

- Dividing simply several LTR measures into:
  - ➢Implicit relevance

❖Explicit feedback

- ✓Providing explicitly a specific score for each observed item by users
- ✓Evaluating the performance using Normalized discounted cumulative gain(NDCG) with cut off value K
- ✓Generating the NDCG@K as follows

$$NDCG@K = \frac{DCG@K}{\max(DCG@K)}, DCG@K = \sum_{l_i}^{K} \frac{2^{R_i} - 1}{\log(l_i + 1)}$$

Normalized Discounted Cumulative Gain (NDCG@K)

- measures the performance of a recommendation system based on the graded relevance of the recommended entities
- It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities
- Normalize DCG at rank *n* by the DCG value at rank *n* of the ideal ranking
- The ideal ranking would first return the documents with the highest relevance level, then the next highest relevance level, etc
- This metric is commonly used in information retrieval and to evaluate the performance of web search engines.
- Normalization useful for contrasting queries with varying numbers of relevant results
- It handles multiple levels of relevance (MAP doesn’t)
- It seems to have the right kinds of properties in how it scores system rankings

Lambda Matrix Factorization

- RankNet
  - Proposing firstly to handle the problem of learning to rank using gradient
  - Promoting the relative predicted probability of every pair of documents i and j with different relevance level $R_{q,i} > R_{q,j}$ in terms of their model score
  - The Relative probability of document i ranked higher than document j is defined as

$$P_{i,j}^{q} = \frac{1}{1 + \exp(-\sigma(s_i - s_j))}$$

Lambda Matrix Factorization(Cont.)

- RankNet- Cont.
  - ➢Loss function and gradient for the query q, document i, j is defined as

$$C_{i,j}^{q} = -\overline{P_{i,j}^{q}} \log - \left(1 - \overline{P_{i,j}^{q}}\right) \log\left(1 - P_{i,j}^{q}\right)$$

$$= \log(1 + \exp(-\sigma(s_i - s_j)))$$

$$\frac{\partial C_{i,j}^{q}}{\partial s_i} = -\frac{\sigma}{1 + \exp\left(\sigma\left(s_i - s_j\right)\right)} = \frac{\partial C_{i,j}^{q}}{\partial s_j}$$

Lambda Matrix Factorization(Cont.)

- LambdaRank
  - ➢Based on the idea of RankNet, formulating gradient of a list-wise ranking measure, which is called λ
  - ➢Using best Lambda for optimizing NDCG as below with $|\Delta NDCG|$ defined as the absolute NDCG gain from swapping documents i and j

$$\frac{\partial C_{i,j}^{q}}{\partial s_i} = \lambda_{i,j}^{q} = \frac{\sigma|\Delta NDCG|}{1 + \exp\left(\sigma\left(s_i - s_j\right)\right)} = -\frac{\partial C_{i,j}^{q}}{\partial s_j}$$

$$\frac{\partial C_{i,j}^{q}}{w} = \frac{\partial C_{i,j}^{q}}{\partial s_i}\frac{\partial s_i}{\partial w} + \frac{\partial C_{i,j}^{q}}{\partial s_j}\frac{\partial s_j}{\partial w} = \lambda_{i,j}^{q}\frac{\partial s_i}{\partial w} - \lambda_{i,j}^{q}\frac{\partial s_j}{\partial w}$$

Lambda in Matrix Factorization

- Adopting MF as our latent factor model representing features for users and items
- The values in the sparse rating matrix R with M users and N items are only available when the corresponding ratings exist in the training dataset, and the task of MF is to guess the remaining ratings
- Formulating rating prediction(the predicted score) as below with inner-product of U_u and V_i for each (u, i) pair:

$$\bar{R} = UV^T$$

Lambda in Matrix Factorization

- Introducing LambdaMF by using as the basis of our latent model
- Goal is to optimize a ranking measure function f by using cost function C and lambda for modeling score

$$\frac{\partial C_{i,j}^{u}}{\partial s_i} = \lambda_{i,j}^{u}$$

Differences between LambdaRank and LambdaMF

1. Given a user u, when a pair of items (i , j) is chosen, unlike in LTR which only model weight vector is required to be updated, in a recommendation task we have item latent factors of i and j and user latent factor of u to update
2. In the original LambdaRank model, the score of the pair (u , i) in the model is generated from the prediction outputs a neural network, while in MF such score is generated from the inner-product of U_u and V_i. Hence, we have $\frac{\partial s_i}{\partial U_u}=V_i$ and $\frac{\partial s_i}{\partial V_i}=U_u$

Apply Stochastic Gradient Descent to learn model parameters

- With 2 differences between LambdaRank and LambdaMF, computing the gradient as below given a user u, a pair item i and j, with $R_{u,i} > R_{u,j}$

$$\frac{\partial C^u_{i,j}}{V_i} = \frac{\partial C^u_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial V_i} = \lambda^u_{i,j} U_u$$

$$\frac{\partial C^u_{i,j}}{V_j} = \frac{\partial C^u_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial V_j} = -\lambda^u_{i,j} U_u$$

$$\frac{\partial C^u_{i,j}}{U_u} = \frac{\partial C^u_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial U_u} + \frac{\partial C^u_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial U_u} = \lambda^u_{i,j}(V_i - V_j)$$

- The specific definition of $\lambda^u_{i,j}$ is not given this section. In contrast, we want to convey that the design of $\lambda^u_{i,j}$ can be simple and generic.

Popular Item Effect

- Theorem: If there exists an item $\widehat{i}$, such that all users $k \epsilon Rel(\widehat{i}), R_{u,i} > R_{u,j}$ for all other observed item j of user K. Furthermore, if after certain iteration τ , latent factors of all users $k \epsilon Rel(\widehat{i})$ converge to certain extent. That is, there exists a vector $\bar{U}^t$ such that for all $k \epsilon Rel(\widehat{i})$ I all iteration t $> \tau$, inner-product (U_k^t ,$\bar{U}^\tau$)>0. Then the norm of $V_{\widehat{i}}$ will eventually grow to infinity for any MF model satisfying the constraint that $\frac{\partial C^u_{\widehat{i},j}}{\partial s_{\widehat{i}}}$>0 for all j with $R_{u,\widehat{i}} > R_{u,j}$ as shown below:

$$\lim_{n \to \infty} \|V_{\hat{i}}^n\|^2 = \infty$$

Regularization in LambdaMF

- To address the previous concern, we add a regularization term to the original cost C ($\dot{C}$) as below and thus restrict the complexity of model

$$\dot{C} = \sum_{all(u,i,j)\epsilon data} C_{i,j}^{u} - \frac{\alpha}{2}(\|U\|_2^2 + \|V\|_2^2)$$

- If α is large, the capability of the model is strongly restricted even when the norm of all parameters are small. Hence, we argue that the utilization of norm restriction based method is blind.

Regularization in LambdaMF(Cont.)

- Proposing another regularization as below to confine the inner-product $U_u V_i^T$(Which is the prediction outcome of the model) to be as close to the actual rating $R_{u,i}$ for every observed rating u, i as possible

$$\hat{C} = \sum_{all(u,i,j)\epsilon data} C_{i,j}^{u} - \frac{\alpha}{2} \sum_{all(u,i,j)\epsilon data} (R_{i,j} - U_u \, V_i^T)^2$$

$$\frac{\partial \widehat{C_{i,j}^{u}}}{\partial V_i} = \lambda_{i,j}^{u} U_i + \alpha(R_{u,i} - U_u V_i^T)U_u$$

$$\frac{\partial \widehat{C_{i,j}^{u}}}{\partial V_j} = -\lambda_{i,j}^{u} U_i + \alpha(R_{u,j} - U_u V_j^T)U_u$$

$$\frac{\partial \widehat{C_{i,j}^{u}}}{\partial U_u} = \lambda_{i,j}^{u}(V_i - V_j) + \alpha(R_{u,i} - U_u V_i^T)V_i + \alpha(R_{u,j} - U_u V_j^T)V_j$$

Advantages in adopting the MSE regularization term

1. Intensity-adaptive regularization term

- The adaption power of MSE regularization is automatic with no utiliazation of value α needed.

2. Inductive transfer with MSE

- Inductive transfer is a technique to use some source task T_S(MSE) to help the learning of target task T_T(The ranking measure $\boldsymbol{f}$).
- MSE presents a way to model learning to rank with point-wise learning.
- Optimal MSE=0 also suggests an optimal NDCG=1
- The proposed regularization as adopting an inductive transfer strategy to enhance the performance of LambdaMF.

Algorithm and Complexity Analysis

- By deriving lambda gradient and regularization, ready to define the algorithm and analyze its time complexity
- Choosing lambda as below

$$\lambda_{i,j}^{u} = |f(\xi) - f(\xi')|$$

- By applying NDCG as the ranking measure(f) and denoting the ranking of i as l_i , will be:

$$|f(\xi) - f(\xi')| = \frac{(2^{R_i} - 2^{R_j})\left(\frac{1}{\log(1 + l_i)} - \frac{1}{\log\left(1 + l_j\right)}\right)}{\max(DCG)}$$

Algorithm and Complexity Analysis

- Given $\bar{N}$ observed items for a user u, the computation of ξ takes $\mathrm{O}(\bar{N}\log\bar{N})$ time since it requires sorting, so as the computation of $\lambda_{i,j}^{u}$.
- Updating a pair of items for a user is expensive if SGD is applied
- To overcome such deficiency, proposed to product the mini-batch gradient descent. After sorting the observed item list for a user, we compute the gradient for all observed items. It tacks $O(\bar{N}^2)$
- Effectively, updating each pair of items takes $\mathrm{O}(1)$ time as the sorting time is hidden with mini-batch learning structure

Algorithm:Learning LambdaMF

- **input**: $R_{M,N}$, learning rate η, α , latent factor dimension d,# of iteration n_iter, and target ranking measure f

```
Initialize $U_{M,N}$, $V_{M,N}$ randomly;
**for** $t \leftarrow 1$ **to** $n\_iter$ **do**
    **for** $u \leftarrow 1$ **to** $M$ **do**
        $\frac{\partial \widehat{C^u}}{\partial U_u} \leftarrow 0$;
        **for** $all\ absorved\ i\ \epsilon\ R_u$ **do**
            $\frac{\partial \widehat{C^u}}{\partial V_i} \leftarrow 0$
        $\xi$ = observed item list of $u$ sorted by its predicted scores
        **for** $all\ absorved\ pair\ i, j\ \epsilon\ R_u, R_{u,i} > R_{u,j}$
        **do**
            $\xi' = \xi$ with $i, j$ swapped
            $\frac{\partial \widehat{C_u}}{\partial U_u} += \frac{\partial \widehat{C_{i,j}^u}}{\partial U_u}$
            $\frac{\partial \widehat{C_u}}{\partial V_i} += \frac{\partial \widehat{C_{i,j}^u}}{\partial V_i}$
            $\frac{\partial \widehat{C_u}}{\partial V_j} += \frac{\partial \widehat{C_{i,j}^u}}{\partial V_j}$
        $U_u += \eta \frac{\partial \widehat{C_u}}{\partial U_u}$
        **for** $all\ absorved\ i\ \epsilon\ R_u$ **do**
            $V_i += \eta \frac{\partial \widehat{C_u}}{\partial V_i}$
**ruturn** $U, V$
```

Experiment

- Comparisons Methods
  - Adopting NDCG as the target ranking function to demonstrate the performance of LambdaMF
- Compare LambdaMF with several state-of-the-art LTR recommendation system methods:
  - **PopRec**: Popular Recommendation(PopRec) is a strong non-CF baseline
  - **CoFi Rank**: CoFi Rank is a state-to-the-art MF method minimizing a convex upper bound of (1-NDGC)
  - **ListRank MF**: ListRank MF is another state-to-the-art MF method optimizing cross-entropy of top-one probability, which implemented using softmax function.

Experiment Results

(MEAN,STATNDARD DEVIATION) OF NDCG@10 IN MOVIELEBS100K

	N=10	N=20	N=50
PopRec	(0.5995,0.0000)*	(0.6202,0.0000)*	(0.6310,0.0000)*
CoFi NDCG	(0.6400,0.0061)*	(0.6307,0.0062)*	(0.6076,0.0077)*
CoFi Best	(0.6420,0.0252)*	(0.6686,0.0058)*	(0.7169,0.0059)
ListRank MF	(0.6943,0.0050)*	(0.6940,0.0036)*	(0.6881,0.0052)*
LambdaMF L2	(0.5518,0.0066)*	(0.5813,0.0074)*	(0.6471,0.0074)*
LambdaMF MSE	**(0.7119,0.0005)**	**(0.7126,0.0008)**	**(0.7172,0.0013)**

- Running LambdaMF 10 times
- Reporting average result of NDCG@10 across 10 experimental datasets for rating N=10,20,50

STATISTICS OF DATASETS

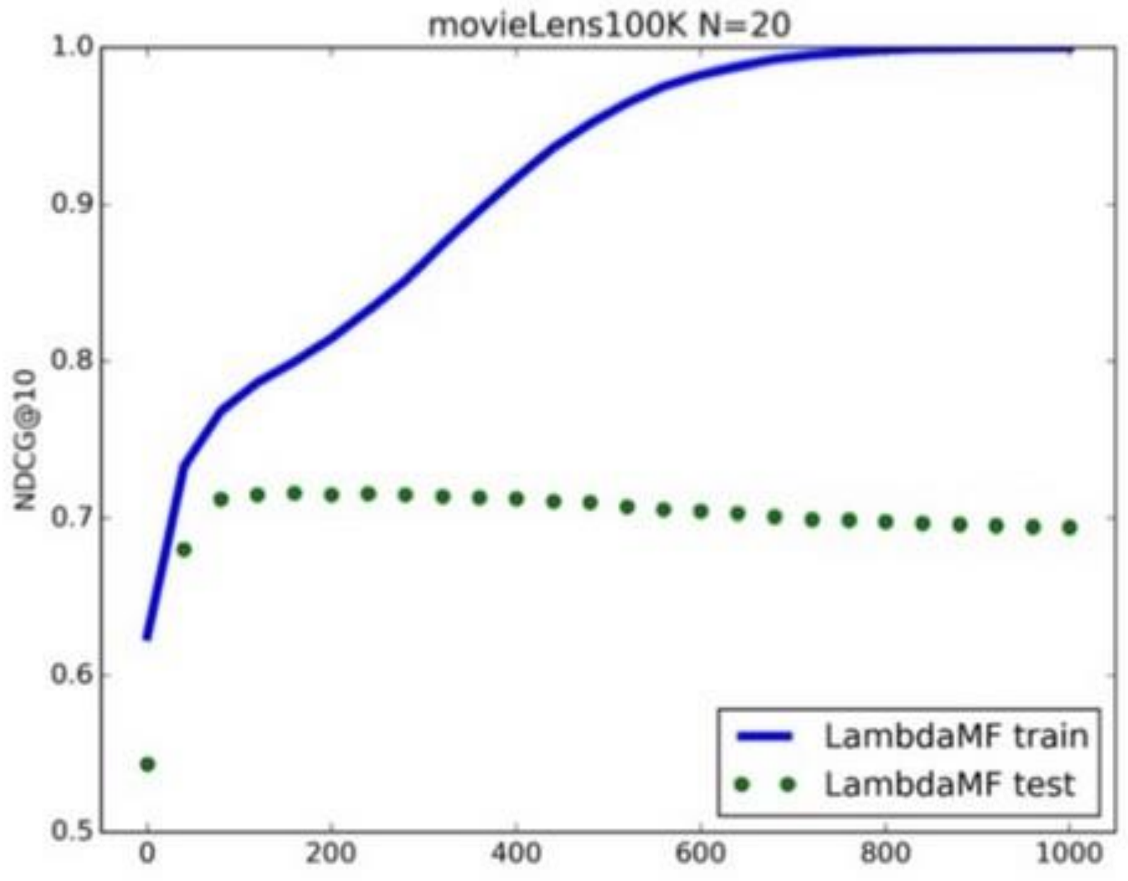
	Netflix	MovieLens
# of users	480189	943
# of items	17770	1682
sparsity	98.82%	93.70%

(MEAN) OF NDCG@10 IN NETFLIX

	N=10	N=20	N=50
PopRec	(0.5175)	(0.5163)	(0.5293)
CoFi NDCG	(0.6081)	(0.6204)	NA
CoFi Best	(0.6082)	(0.6287)	NA
ListRank MF	(0.7550)	(0.7586)	(0.7464)
LambdaMF L2	(0.7171)	(0.7163)	(0.7218)
LambdaMF MSE	**(0.7558)**	**(0.7586)**	**(0.7664)**

Stiftung Universität Hildesheim 2003

Our solution optimizes non-smooth NDCG directly



Conclusion

- Purposing a new framework for lambda in a recommendation scenario
- Proving that ranking-based MF model can be ineffective due to overflow in certain circumstance
- Proposing an adjustment of our model using regularization
- Updating step for each pair of items takes effectively O(1) time in LambdaMF.
- Showing empirically that LambdaMF outperforms the state-of-the-art in terms of NDCG@10.
- Demonstrating that LambdaMF exhibits global optimality and directly optimizes target ranking function.
- Considering the stability of MSE regularization

References

- [1] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Han-jalic, “Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering,” in Proceedings of the sixth ACM conference on Recommender systems. ACM, 2012, pp. 139–146.
- [2] C. Quoc and V. Le, “Learning to rank with nonsmooth cost functions,” Proceedings of the Advances in Neural Information Processing Systems, vol. 19, pp. 193–200, 2007.
- [3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009, pp. 452–461.
- [4] Y. Shi, M. Larson, and A. Hanjalic, “List-wise learning to rank with matrix factorization for collaborative filtering,” in Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010, pp. 269–272.
- [5] P. Donmez, K. M. Svore, and C. J. Burges, “On the local optimality of lambdarank,” in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. ACM, 2009, pp. 460–467.

Ranking and Recommendation

CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering

by Vanessa-Chantal Zydorek

Overview

Motivation

- Problem of recommendation: binary relevance data
- Finding a better solution than CF and BPR with CLiMF
- Top-k recommendation lists are valuable (less-is-more)
- Introducing a lower bound of the smoothed RRM, a linear computational complexity is achieved
- CLiMF outperforms old Collaborative Filtering methods

Main Idea of Collaborative Filtering

- CF methods are the core of recommendation engines
- The main idea: users that shared common interests prefer similar products/items
- CF is valuable in scenarios with only implicit feedback or the duration of the presence of a user on a particular website
- Problem: in some scenarios this count information is not available

- Conclusion: Use binary information
- e.g. 1 = friendship and 0 = no friendship/not observed

Bayesian Personalized Ranking

- In the past: BPR proposed as state of the art recommendation algorithm
- Based on pair-wise comparisons between the binary information of relevant and irrelevant items
- Optimization of the Area Under the Curve
- Problem: AUC measure doesn’t reflect well the quality of the recommendation lists
- The position at which the pairwise comparisons are made, is irrelevant
- Problem: Mistakes at lower ranked positions are penalized to mistakes in higher ranked positions

Collaborative Less-is-More Filtering

- Better approach on recommendation algorithm with binary data
- Data is modeled by means of directly optimizing the MRR
- Is a important measure of recommendation quality for domains with only few recommendations (like top-3)
- “The mean reciprocal rank is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness.”[2]

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

- Rank$_i$: rank position of first relevant document for i-th query
- By measuring how early in the list the first relevant recommended item is ranked, the Reciprocal Rank is defined for a given recommendation list of a single user

Collaborative Less-is-More Filtering

- First, insights are taken from the area of learning to rank and integrating latent factor models from CF
- Subsequently, CLiMF optimizes a lower bound of the smoothed Reciprocal Rank for learning the model parameters
- This model parameters are used to generate item recommendations for individual users

Ranking-oriented Collaborative Filtering

- Previous work:
- Matrix Factorization with latent factor U_i, V_j vectors for each user and item
- CF uses a ranking oriented objective function to learn the latent factors of items and users
- CLiMF: extension to general Matrix Factorization with new characteristics
- First model-based methods for the use scenarios with only implicit feedback data was introduced in 2008

Learning to Rank

- CLiMF is closely related to Learning to Rank which focuses on direct optimization of IR metrics
- LTR minimizes the convex upper bounds of loss functions that are based on the evaluation measures like SVM
- Optimizes a smoothed version of an evaluation measure like SoftRank
- CLiMF targets the application scenario of recommendation rather than query-document search and
- An algorithm is proposed that makes the optimization of the smoothed MRR tractable and scalable

Smoothing the Reciprocal Rank

$$RR_i = \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij}))$$

Figure: 1

- User = *i*   Item = *j*
- Number of items = *N*
- Binary relevance score of item *j* to user *i* = Y_{ij}
- Where Y_{ij} is 1 when the item *j* is relevant to user *i*, otherwise it is 0
- Indicator function $\mathbb{I}(x)$
- When *x* is true = 1, otherwise it is 0
- Rank of the item *j* in the ranked list of items for user *i* = R_{ij}

Smoothing the Reciprocal Rank

$$RR_i = \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij}))$$

Figure: 1

- The user *i* has predicted relevance scores which are ranked descending
- This model is dependent on the ranking of relevant item
- RR_i is a non-smooth function over the model parameters
- Problem: RR measure makes it impossible to use standard optimization methods

Smoothing the Reciprocal Rank

$$RR_i = \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij}))$$

Figure: 1

- Therefore an approximation of $\mathbb{I}(R_{ik} < R_{ij})$ is derived by using a logistic function:

$$\mathbb{I}(R_{ik} < R_{ij}) \approx g(f_{ik} - f_{ij})$$

Figure: 2

- Where $g(x) = 1/(1+e^{-x})$, f_{ij} defines the predictor function that maps the parameters from user *i* and item *j* to a predicted relevance score
- Reminder: Logistic regression = $1/(1+e^{-x})$

Smoothing the Reciprocal Rank

$$RR_i = \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij}))$$

Figure: 1

- The predictor function is the factor model

$$f_{ij} = \langle U_i, V_j \rangle$$

Figure: 3

- Where U_i denotes a *d*-dimensional latent factor vector for user *i* and V_j a *d*-dimensional latent factor vector for item *j*
- Y_{ij} can only be either 1 or 0
- Only $1/R_{ij}$ is in use

Smoothing the Reciprocal Rank

- Therefore $1/R_{ij}$ is approximated by another logistic function:

$$\frac{1}{R_{ij}} \approx g(f_{ij})$$ Figure: 4

- It assumptions that the lower the item rank, the higher the predicted relevance score
- Now assimilating the given thought, the model is changed to a finally smooth version

$$RR_i \approx \sum_{j=1}^{N} Y_{ij} g(f_{ij}) \prod_{k=1}^{N} \left(1 - Y_{ik} g(f_{ik} - f_{ij})\right)$$ Figure: 5

Smoothing the Reciprocal Rank

- In other words:
- Only the items that are relevant to an user are taken into account
- With the advanced and smooth function RR_{ij} the items that are lower in the rank, have a higher predicted relevance score

$$RR_i \approx \sum_{j=1}^{N} Y_{ij} g(f_{ij}) \prod_{k=1}^{N} \left(1 - Y_{ik} g(f_{ik} - f_{ij})\right)$$

Smoothing the Reciprocal Rank

$$RR_i \approx \sum_{j=1}^{N} Y_{ij} g(f_{ij}) \prod_{k=1}^{N} \left(1 - Y_{ik} g(f_{ik} - f_{ij})\right)$$

Figure: 5

- The problem is:
- The complexity of the gradient with respect to V_j is $O(N^2)$
- -> Computational cost grows quadratically with the number of items N
- The number of items is generally huge in recommender systems (e.g. thousands of movies in a database)
- To avoid this threat of large computational cost, there is a lower bound of an similar model

Lower Bound of Smooth Reciprocal Rank

- Assuming that the number of relevant items for user *i* is n_i^+
- The model parameters that maximize the previous model are equivalent to the parameters that maximize $ln((1/n_i^+)RR_i)$
- Ensuing this model:

$$
\begin{aligned}
U_i, V &= \arg\max_{U_i,V}\{RR_i\} = \arg\max_{U_i,V}\{\ln(\frac{1}{n_i^+}RR_i)\} \\
&= \arg\max_{U_i,V}\{\ln\Big(\sum_{j=1}^{N}\frac{Y_{ij}}{n_i^+}g(f_{ij})\prod_{k=1}^{N}\big(1 - Y_{ik}g(f_{ik}-f_{ij})\big)\Big)\}
\end{aligned}
$$

Figure: 6

Lower Bound of Smooth Reciprocal Rank

- The lower bound of $\ln((1/n_i^+)RR_i)$ is derived as follows:

$$
\begin{aligned}
&\ln(\frac{1}{n_i^+}RR_i)\\
&=\ln\left(\sum_{j=1}^{N}\frac{Y_{ij}}{\sum_{l=1}^{N}Y_{il}}g(f_{ij})\prod_{k=1}^{N}\left(1-Y_{ik}g(f_{ik}-f_{ij})\right)\right)\\
&\geq\frac{1}{n_i^+}\sum_{j=1}^{N}Y_{ij}\ln\left(g(f_{ij})\prod_{k=1}^{N}\left(1-Y_{ik}g(f_{ik}-f_{ij})\right)\right)\\
&=\frac{1}{n_i^+}\sum_{j=1}^{N}Y_{ij}\left(\ln g(f_{ij})+\sum_{k=1}^{N}\ln\left(1-Y_{ik}g(f_{ik}-f_{ij})\right)\right)
\end{aligned}
$$

Figure: 7

Lower Bound of Smooth Reciprocal Rank

$$\geq \frac{-}{n_i^+} \sum_{j=1} Y_{ij} \ln \Big(g(f_{ij}) \prod_{k=1} (1 - Y_{ik} g(f_{ik} - f_{ij})) \Big)$$

$$= \frac{1}{n_i^+} \sum_{j=1}^{N} Y_{ij} \Big(\ln g(f_{ij}) + \sum_{k=1}^{N} \ln (1 - Y_{ik} g(f_{ik} - f_{ij})) \Big)$$

Figure: 7

- The derivation above uses the definition of the $n_i^+ = \sum_{l=1}^{N} Y_{il}$
- The constant $1/n_i^+$ can be neglected in the lower bound
- The new function is now:

$$L(U_i, V) = \sum_{j=1}^{N} Y_{ij} \big[\ln g(f_{ij}) + \sum_{k=1}^{N} \ln (1 - Y_{ik} g(f_{ik} - f_{ij})) \big]$$

Figure: 8

Lower Bound of Smooth Reciprocal Rank

- CLiMF leads to a recommendation where some relevant items are at the very top of the recommendation list for a user
- With the regularization terms that serve to control the complexity of the model, the new objective function is:

$$\begin{aligned} F(U,V) = &\sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}\Big[\ln g(U_i^T V_j) \\ &+ \sum_{k=1}^{N} \ln\big(1 - Y_{ik}\, g(U_i^T V_k - U_i^T V_j)\big)\Big] \\ &- \frac{\lambda}{2}(\|U\|^2 + \|V\|^2) \end{aligned}$$

Figure: 9

Lower Bound of Smooth Reciprocal Rank

$$F(U,V) = \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}\Big[\ln g(U_i^T V_j) + \sum_{k=1}^{N} \ln\big(1 - Y_{ik}\, g(U_i^T V_k - U_i^T V_j)\big)\Big] - \frac{\lambda}{2}(\|U\|^2 + \|V\|^2)$$

Figure: 9

- Lambda denotes the regularization coefficient
- $\|U\|$ is identified as Frobenius norm of *U*
- This leads to the fact that the lower bound *F*(*U*,*V*) is much less complex than the original objective function RR$_i$:

$$RR_i \approx \sum_{j=1}^{N} Y_{ij}\, g(f_{ij}) \prod_{k=1}^{N} \big(1 - Y_{ik}\, g(f_{ik} - f_{ij})\big)$$

Figure: 5

- The standard optimization methods (e.g. gradient ascend) can be used to learn the optimal model parameters *U* and *V*

Optimization

- To maximize the function,

  the stochastic gradient ascent is used

$$F(U,V) = \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}\Big[\ln g(U_i^T V_j) + \sum_{k=1}^{N} \ln\big(1 - Y_{ik}\, g(U_i^T V_k - U_i^T V_j)\big)\Big] - \frac{\lambda}{2}\left(\|U\|^2 + \|V\|^2\right)$$

Figure: 9

- For each user *i*, *F*(*U*$_i$, *V*) is optimized
- The gradients can be computed as follows:

$$\frac{\partial F}{\partial U_i} = \sum_{j=1}^{N} Y_{ij}\Big[g(-f_{ij})V_j + \sum_{k=1}^{N} \frac{Y_{ik}\, g'(f_{ik} - f_{ij})}{1 - Y_{ik}\, g(f_{ik} - f_{ij})}(V_j - V_k)\Big] - \lambda U_i$$

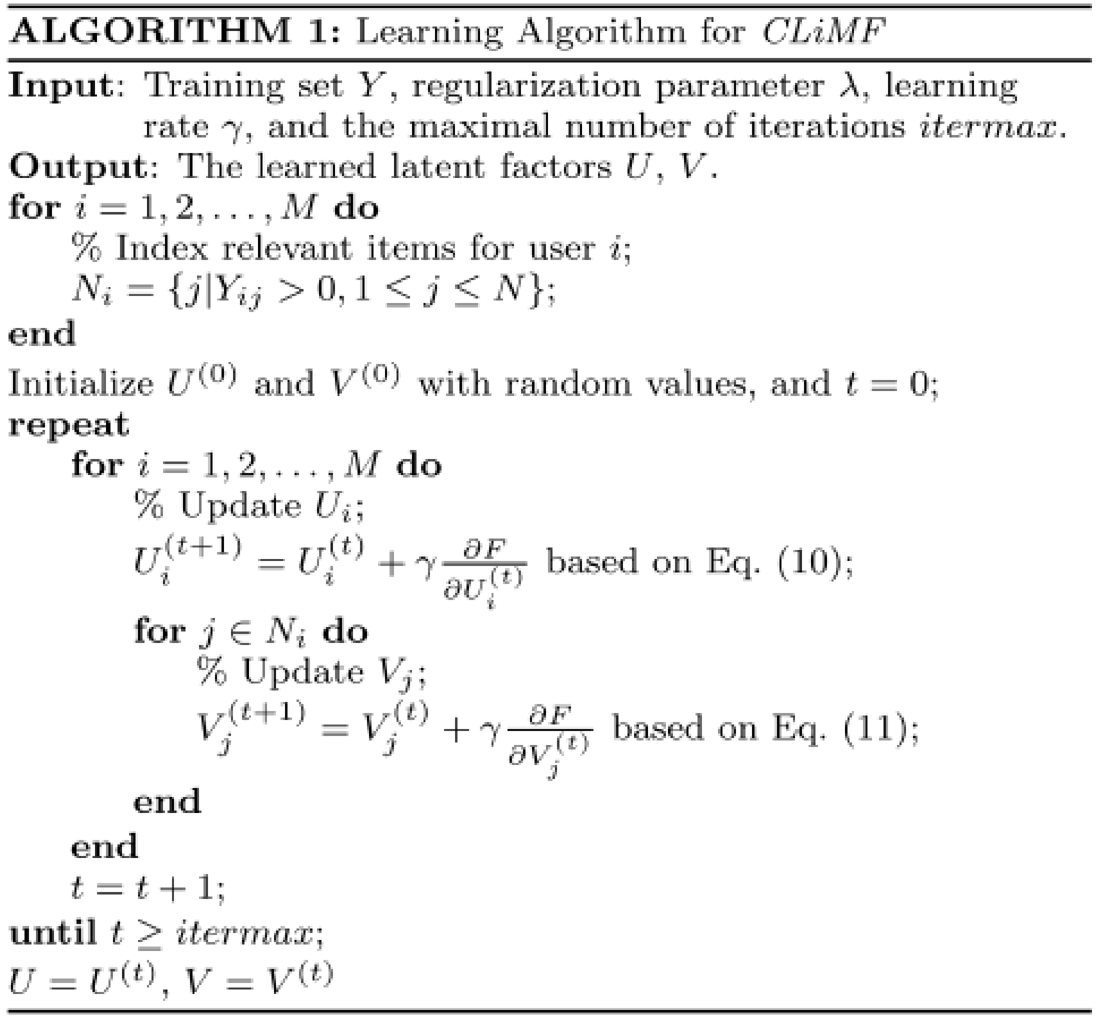
- Figure: 10

$$\frac{\partial F}{\partial V_j} = Y_{ij}\Big[g(-f_{ij}) + \sum_{k=1}^{N} Y_{ik}\, g'(f_{ij} - f_{ik})\Big(\frac{1}{1 - Y_{ik}\, g(f_{ik} - f_{ij})} - \frac{1}{1 - Y_{ij}\, g(f_{ij} - f_{ik})}\Big)\Big]U_i - \lambda V_j$$

Figure: 11

Optimization

•



ALGORITHM 1: Learning Algorithm for *CLiMF*

```
Input: Training set $Y$, regularization parameter $\lambda$, learning
       rate $\gamma$, and the maximal number of iterations $itermax$.
Output: The learned latent factors $U$, $V$.
for $i = 1, 2, \ldots, M$ do
    % Index relevant items for user $i$;
    $N_i = \{j | Y_{ij} > 0, 1 \leq j \leq N\}$;
end
Initialize $U^{(0)}$ and $V^{(0)}$ with random values, and $t = 0$;
repeat
    for $i = 1, 2, \ldots, M$ do
        % Update $U_i$;
        $U_i^{(t+1)} = U_i^{(t)} + \gamma \frac{\partial F}{\partial U_i^{(t)}}$ based on Eq. (10);
        for $j \in N_i$ do
            % Update $V_j$;
            $V_j^{(t+1)} = V_j^{(t)} + \gamma \frac{\partial F}{\partial V_j^{(t)}}$ based on Eq. (11);
        end
    end
    $t = t + 1$;
until $t \geq itermax$;
$U = U^{(t)}$, $V = V^{(t)}$
```

Optimization

- By exploiting the data sparseness in Y, the computational complexity of the gradient in Figure 10 is *O(dñ² M +dM)*

Figure: 10

$$\frac{\partial F}{\partial U_i} = \sum_{j=1}^{N} Y_{ij}\Big[g(-f_{ij})V_j + \sum_{k=1}^{N} \frac{Y_{ik}g'(f_{ik}-f_{ij})}{1-Y_{ik}g(f_{ik}-f_{ij})}(V_j - V_k)\Big] - \lambda U_i$$

- ñ is the average number of relevant items across all the users
- Whereas the complexity of the gradient in Figure 11 is *O(dñ² M +dñM)*

Figure: 11

$$\frac{\partial F}{\partial V_j} = Y_{ij}\Big[g(-f_{ij}) + \sum_{k=1}^{N} Y_{ik}g'(f_{ij}-f_{ik})\Big(\frac{1}{1-Y_{ik}g(f_{ik}-f_{ij})} - \frac{1}{1-Y_{ij}g(f_{ij}-f_{ik})}\Big)\Big]U_i - \lambda V_j$$

- Conclusion: CLiMF is suitable for large use cases

Relation to other state of the art recommendation models

- Similarity to CofiRank, Collaborative Competitive Filtering, OrdRec and Bayesian Personalized Ranking
- All past models do ranking
- Relative pair-wise constraints in learning the latent factors

- Difference and simultaneously advantages:
- CLiMF optimizes a ranking loss (Area Under the Curve) and deals with binary relevance data
- CLiMF first smooths the evaluation metric RR, and then optimizes the smoothed version via lower bound
- CLiMF only requires relevant items from users
- CLiMF promotes and scatters relevant items at the same time
- CLiMF focuses on recommending items that are few in number, but relevant at top-k positions of the list
- CLiMF is able to recommend relevant items to the top positions of a recommendation list
- CLiMF's computational complexity is linear and therefore suited for large scale use cases

Experimental Evaluation

- Approach:

1. Describing the datasets used in the experiment and the setup
2. Comparing the recommendation performance of CLiMF with two baseline approaches
3. Analyzing the effectiveness and the scalability of the CLiMF model

Experimental Evaluation

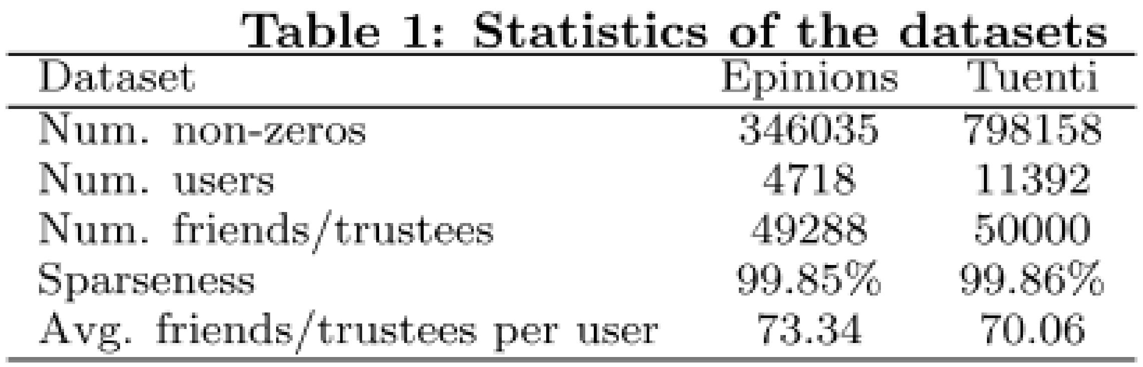


Table 1: Statistics of the datasets

Dataset	Epinions	Tuenti
Num. non-zeros	346035	798158
Num. users	4718	11392
Num. friends/trustees	49288	50000
Sparseness	99.85%	99.86%
Avg. friends/trustees per user	73.34	70.06

- Two social network datasets from Epinions and Tuenti
- Number of trust relationships between
- 49,288 users on Epinions 50,000 users on Tuenti
- Friends/trustees are declared as “items” of a user
- Assuming that the items are relevant to the user
- Task: Generate friend recommendations for individual users

Experimental Evaluation

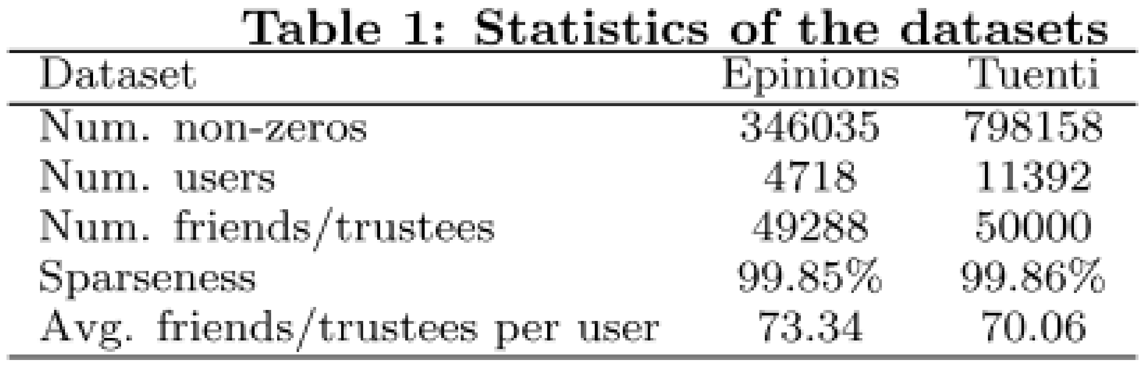


Table 1: Statistics of the datasets

Dataset	Epinions	Tuenti
Num. non-zeros	346035	798158
Num. users	4718	11392
Num. friends/trustees	49288	50000
Sparseness	99.85%	99.86%
Avg. friends/trustees per user	73.34	70.06

- Separating each dataset into training and test set
- Using the training dataset to generate recommendation lists
- Using the test dataset to measure the performance

Experimental Evaluation

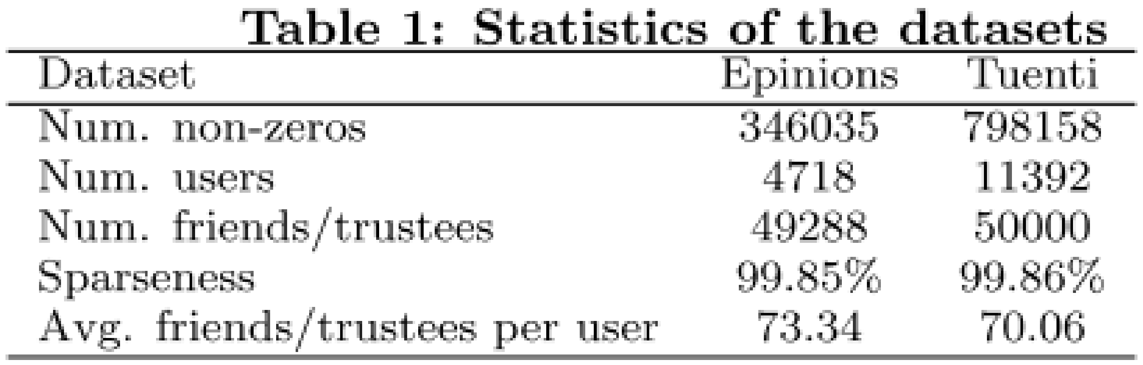


Table 1: Statistics of the datasets

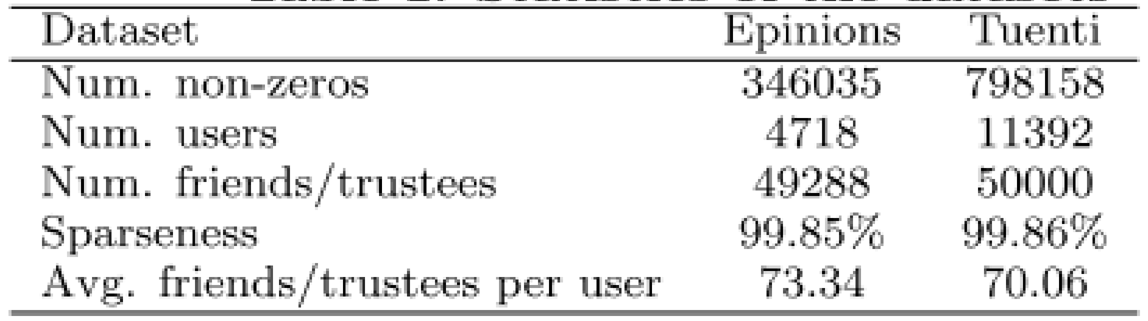
Dataset	Epinions	Tuenti
Num. non-zeros	346035	798158
Num. users	4718	11392
Num. friends/trustees	49288	50000
Sparseness	99.85%	99.86%
Avg. friends/trustees per user	73.34	70.06

- Condition example: “given 30”
- -> 30 random selected friends for training dataset and the rest for test dataset

- Repeating experiment five times with different conditions and each dataset
- The results are averaged across five runs

Experimental Evaluation

Table 1: Statistics of the datasets



Dataset	Epinions	Tuenti
Num. non-zeros	346035	798158
Num. users	4718	11392
Num. friends/trustees	49288	50000
Sparseness	99.85%	99.86%
Avg. friends/trustees per user	73.34	70.06

- Using Mean Reciprocal Rank (MRR)
- Measuring performance by precision at top-ranked items
- precision at top-5 (P@5) -> reflects ratio of number of relevant items in top-5 recommended items
- Measuring of 1-call at top-ranked items
- (1-call@5) -> reflects ratio of test users who have at least one relevant item in their top-5

Experimental Evaluation

- Avoiding that popular friends/trustees heavily dominate recommendation performance
- -> The top-3 are considered irrelevant to reduce the influence
- Condition set to “Given 5” for validation
- Values of the parameters with best performance on validation set:
- Regularization parameter $\lambda = 0.001$
- Latent dimensionality $d = 10$
- Learning rate $\gamma = 0.0001$

Experimental Evaluation

- Comparing performance of CLiMF with three baselines PopRec, iMF and BPR
- PopRec: Native baseline that recommends a user to be a friend in terms of popularity (training dataset)
- iMF: State of the art matric factorization technique for implicit feedback data (Regularization parameter set to 1)
- Bayesian Personalized Ranking: represents state of the art optimization framework (binary relevance data)

Experimental Evaluation

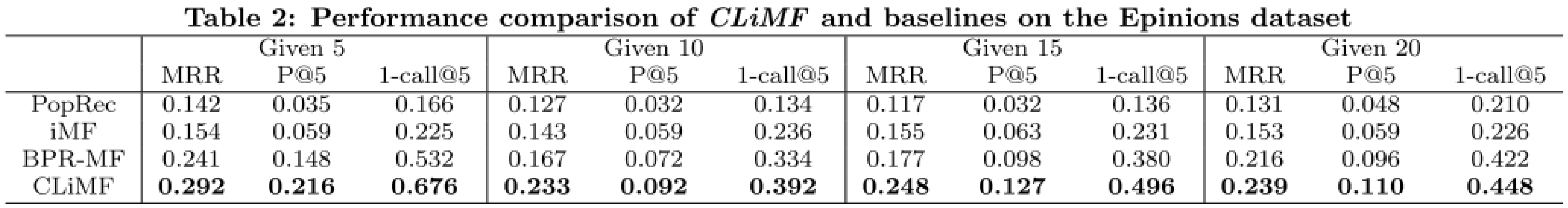


Table 2: Performance comparison of *CLiMF* and baselines on the Epinions dataset

	Given 5			Given 10			Given 15			Given 20		
	MRR	P@5	1-call@5	MRR	P@5	1-call@5	MRR	P@5	1-call@5	MRR	P@5	1-call@5
PopRec	0.142	0.035	0.166	0.127	0.032	0.134	0.117	0.032	0.136	0.131	0.048	0.210
iMF	0.154	0.059	0.225	0.143	0.059	0.236	0.155	0.063	0.231	0.153	0.059	0.226
BPR-MF	0.241	0.148	0.532	0.167	0.072	0.334	0.177	0.098	0.380	0.216	0.096	0.422
CLiMF	**0.292**	**0.216**	**0.676**	**0.233**	**0.092**	**0.392**	**0.248**	**0.127**	**0.496**	**0.239**	**0.110**	**0.448**

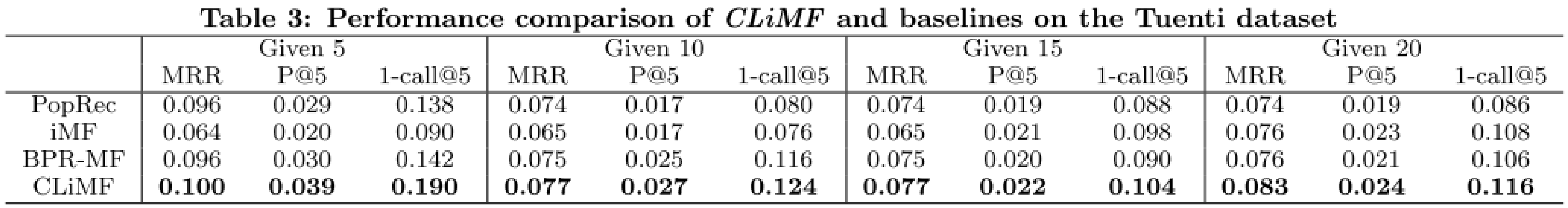


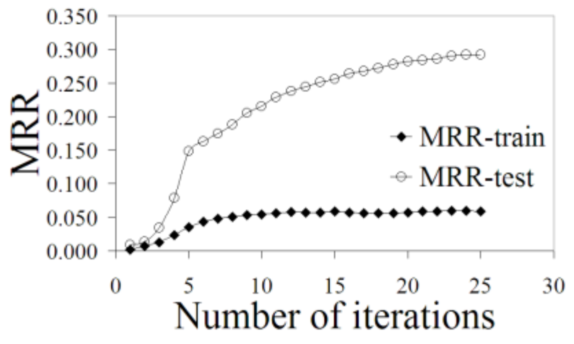
Table 3: Performance comparison of *CLiMF* and baselines on the Tuenti dataset

	Given 5			Given 10			Given 15			Given 20		
	MRR	P@5	1-call@5	MRR	P@5	1-call@5	MRR	P@5	1-call@5	MRR	P@5	1-call@5
PopRec	0.096	0.029	0.138	0.074	0.017	0.080	0.074	0.019	0.088	0.074	0.019	0.086
iMF	0.064	0.020	0.090	0.065	0.017	0.076	0.065	0.021	0.098	0.076	0.023	0.108
BPR-MF	0.096	0.030	0.142	0.075	0.025	0.116	0.075	0.020	0.090	0.076	0.021	0.106
CLiMF	**0.100**	**0.039**	**0.190**	**0.077**	**0.027**	**0.124**	**0.077**	**0.022**	**0.104**	**0.083**	**0.024**	**0.116**

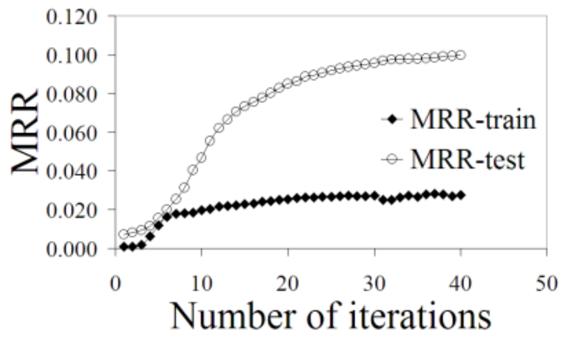
- Not possible to compare results across condition (containing different numbers of items)
- CLiMF model outperforms the three baselines (in MRR)
- measured based on results from individual test users with $p<0.01$
- CLiMF achieves improvement over baselines (in P@5 and 1-call@5)
- By optimizing MRR, CLiMF improves quality of recommendations among top-ranked items

Experimental Evaluation

- Investigating the effectiveness of CLiMF



(a) Epinions



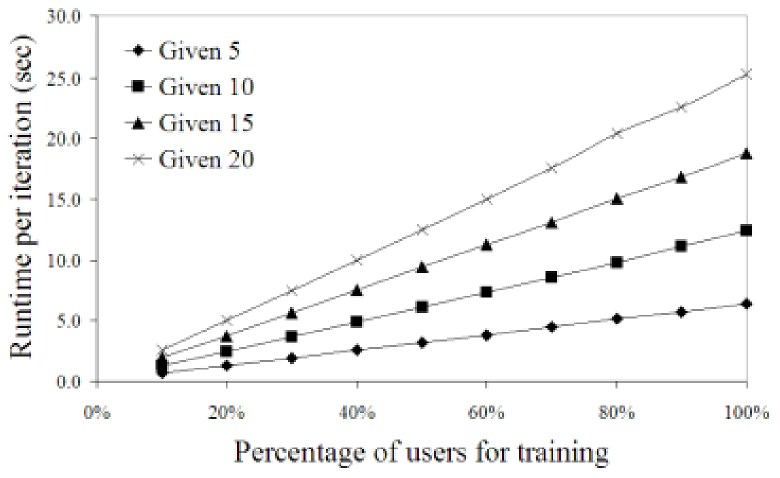
(b) Tuenti

Figure: 12 and 13

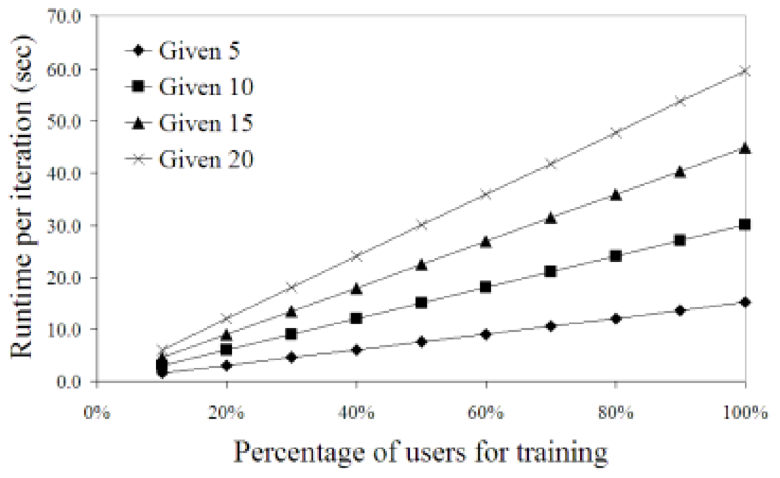
- Evolution of MRR with each iteration under “Given 5” condition
- Increasing measures with each iteration
- Reaching convergence after few iterations (25 and 30)

Experimental Evaluation

- Investigating the scalability of CLiMF



(a) Epinions



(b) Tuenti

-

Figure: 14 and 15

- Measuring the training time that is required
- Computational time is almost linearly to the increase of number of users

Conclusion

- CLiMF learns latent factors of users and items by directly maximizing MRR
- The model is specialized to improve the performance of top-k recommendations for usage scenarios with only binary relevant data
- CLiMF is superior to the past state of the art baseline models
- MRR can be optimized trough CLiMF

- CLiMF's basic idea of the less-is-more is valuable to the performance of computation time and prediction of what a single user's recommendation list could be

References

- Figures 1-15, Algorithm 1, Table 1-3
- [1]Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., & Hanjalic, A. (2012). CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In RecSys '12 Proceedings of the sixth ACM conference on Recommender systems (S. 139-146). Dublin, Ireland: ACM New York. doi:10.1145/2365952.2365981
- [2]Mean reciprocal rank: https://en.wikipedia.org/wiki/Mean_reciprocal_rank (October 20th 2016, at 13:05)

Ranking-oriented CF methods (CLiMF & LambdaMF)

- Attracting more attention
- CLiMF :
  - Optimizing an approximation of mean reciprocal rank(MRR)
- CoFi Rank:
  - Optimizing a bound of NDCG
  - Deriving loss function from a bound of NDCG
  - No guarantee about the distance between bound and actual real ranking function
- LambdaMF:
  - exhibits global optimality and directly optimizes target ranking function.
  - Dealing with actual ranking due to is incorporation of sorting
  - Formulating directly the ranking function into its gradient descent and no need to worry about the similar extent of approximation or bound

CLiMF & LambdaMF

- Similarity

➢ Both of methods are list-wise Approach and approximation of Rankings

- Differences

➢ CLiMF $< -$Smooth approach

➢ LambdaMF $< -$ Non-smooth approach

Winning Method : LambdaMF

- It presents a template model which can be fine-tuned for different types of ranking functions given sparse data

Questions