University of Zurich[UZH]

Two-Class Collaborative Filtering Problems

Thesis August 29, 2016

Thilo Haas
of Zurich ZH, Switzerland

Student-ID: 08-910-697
thilo@thilohaas.ch

Advisor: **Bibek Paudel**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to express my gratitude to Prof. Abraham Bernstein, for giving me the opportunity to pursue my master thesis at the Dynamic and Distributed Information Systems (DDIS) research group. I'm deeply grateful to my advisor Bibek Paudel, for all the feedback, fruitful discussions and reviews. He introduced me to the research field and provided invaluable advice to the herein presented work, thank you for your patience Bibek. I would also like to thank my family and friends for the support throughout my studies.

Zusammenfassung

Wir untersuchen Zwei-Klassen kollaborative Filter-Probleme mit positiver und negativer Klassen-Prognose. Unser Ziel ist die Erstellung von personalisierten Empfehlungslisten, wobei erwünschte Artikel zuoberst und unerwünschte Artikel zuunterst auf der Liste eingeordnet werden sollen.

Basierend auf Bayesian Personalized Ranking Matrix Factorization (BPRMF) von Rendle et al. (2012) und Logistic MF von Johnson (2014), präsentieren wir neue Methoden zur Lösung von Zwei-Klassen kollaborativen Filter-Problemen. Wir evaluieren unsere Methoden anhand der folgenden vier Datensätze: MovieLens 100K/1M, Slashdot-Zoo und Book Crossing. Wir vergleichen unsere Resultate mit den bestehenden Methoden BPRMF, Logistic MF, SGDReg (Levy and Jack, 2013) und GAUC-OPT (Song and Meyer, 2015).

Mit unseren Methoden übertreffen wir die bestehenden Methoden Logistic MF, BPRMF und GAUC-OPT in jeweils mindestens einem der folgenden Messwerte: AUC, Hit-Rate@10, Precision@20 und dem jeweiligen negativen Messwert. Dennoch werden alle unsere Methoden durch SGDReg übertroffen, welches in fast allen Messwerten und fast allen Datensätzen ausgezeichnete Resultate liefert.

Abstract

We study Two-Class Collaborative Filtering (TCCF) problems with positive and negative class prediction. Our goal is to distinguish between positive and negative samples by predicting positive samples at the top and negative samples at the bottom of a personalized ranking list.

Based on Bayesian Personalized Ranking Matrix Factorization (BPRMF) from Rendle et al. (2012) and Logistic MF from Johnson (2014), we introduce different new models to address TCCF problems. We evaluate our models on MovieLens 100K/1M, Slashdot-Zoo and Book Crossing datasets and compare the results with an evaluation of BPRMF, Logistic MF, SGDReg (Levy and Jack, 2013) and GAUC-OPT (Song and Meyer, 2015).

With our models we outperform Logistic MF, BPRMF and GAUC-OPT on either AUC, Hit-Rate@10, Precision@20 and their respective negative evaluation metrics. However all our evaluation results are surpassed by SGDReg, which excels in most evaluation metrics on the examined datasets.

Table of Contents

1

Introduction

Modern consumers are offered a vast variety of choices. Online retailers and content providers present a wide range of products and articles for different target audiences. Matching consumers with the most appropriate content is key to enhancing user satisfaction and loyalty (Koren et al., 2009). Therefore personalized recommender systems that suit a user's taste became popular and many e-commerce leaders like Amazon and Netflix added them as a salient part of their websites. By logging customers' implicit feedback such as site views and product purchases as well as explicit feedback like product ratings provided by the customers, a huge amount of data is available about which content appeals to which customer. By analyzing this data, companies can recommend the right content to the right customers.

Explicit and implicit negative feedback is increasingly available as well. For example we can extract negative feedback from users' ratings by treating ratings from 1-2 as negative feedback and 4-5 as positive feedback. Further examples of positive and negative feedback are: distinguishing friends from foes, spam from important mails and beneficial actions from harmful ones. The cost of a negative recommendation may not be significant in domains like movies or TV shows. But in case of jobs, places to live or friends to connect to, the cost of a negative recommendation could be high. In such domains, users are likely to be put off by recommender systems that cannot distinguish potentially negative items and occasionally put them at the top of the list. Therefore, producing personalised rankings with positive feedback at the top and negative feedback at the bottom is an increasingly important task.

In recommender systems oftentimes negative and unknown feedback is mixed together and not analyzed separately, known as the One-Class Collaborative Filtering (OCCF) problem. This leads to a bias towards positive feedback, whereas negative feedback cannot be distinguished from unknown feedback. Our goal is to examine how we can exploit the availability of negative feedback, to improve positive recommendations and also improve negative feedback prediction. We extend OCCF to Two-Class Collaborative Filtering (TCCF) to address this limitation and improve prediction quality, by considering positive, negative and unknown feedback separately.

This Chapter is structured as follows: The first Section of this Chapter gives an overview over the different research directions in the field of recommender systems. Section 1.2 will explain our motivation for Two-Class Collaborative Filtering problems and our contributions will be shown in Section 1.3.

In Chapter 2 we give a review of the existing literature and explain the relations to our work. In Chapter 3 we present the experimental setup and explain the different models and their adaptions to Two-Class Collaborative Filtering. We evaluate these models by applying them to the MovieLens Slashdot-Zoo and Book Crossing datasets and explain the datasets and results in Chapter 5.2. In Chapter 5 we conclude our experiments and summarize the results.

1.1 Background

There are two general types of recommender systems. There are *Content Filtering* and *Collaborative Filtering* approaches.

With *Content Filtering*, a profile is generated for each product/user which describes its characteristics/preferences. For example a movie contains features, describing its genre, whereas a user profile contains its demographic information. Oftentimes this data needs to be collected externally and is not always readily available. Content Filtering then matches user and content profiles.

Collaborative Filtering (Goldberg et al., 1992) uses past user behaviour like previous page views, product purchases or product ratings and doesn't need explicit profile creation. It analyzes dependencies between users and between products, to fill the blanks in the user-item matrix and to suggest and score new user-item recommendations. The main benefit of Collaborative Filtering is that it is domain independent and thus can model elusive features, which cannot be collected easily by user or item profiles. It assumes that, if user A has similar preferences as user B, user A is also more likely to prefer an item x from user B, than from any other randomly chosen user. Due to this quality estimation of items by peers, the model is more likely to produce a more diverse result set (Desrosiers and Karypis, 2011). On the other hand it suffers from the *cold-start problem*, its inability to address new items or users, since we cannot compute similar items or users for those.

Collaborative Filtering is mostly applied using *neighbourhood models* or *latent factor models* (Koren and Bell, 2011).

Neighborhood models calculate the similarity of either users or items, based upon the rating/feedback given by the same user. The model recommends "neighboring" items, which received similar feedback as another item the user already liked, or items which are liked by another user with similar ratings alternatively.

Latent factor models aim to explain users and items, by characterizing both with a low number of latent factors, inferred from the feedback patterns. A latent vector is assigned to each user and item. The dot product between those vectors will serve as the recommendation score for the given user and item. Some of the most sucessful latent factor models are based on matrix factorization. Matrix factorization aims to approximate the user-item feedback matrix with n users and m items $R \in \mathbb{R}^{n \times m}$ into a lower dimensional dot product approximation $\hat{R} = U^T V$ with $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times m}$ such that $\hat{R} \approx R$.

Traditionally, Collaborative Filtering problems focus on positive $(0, 1]$ and neutral/neg-

ative = 0 feedback only and treat the case, where no information is available the same as where negative feedback is given. This forms the basis of the *One-Class Collaborative Filtering (OCCF)*. With *Two-Class Collaborative Filtering (TCCF)*, we also consider negative feedback explicitly and therefore aim to predict the two classes positive or negative for any given item.

In this section, we gave a short overview of the different approaches to solve the recommender system problem. We pointed out the differences between content filtering and collaborative filtering, as well as the most successful realization of collaborative filtering as a latent factor model with matrix factorization. Due to the focus of this work, we emphasize on the developments in the field of latent factor collaborative filtering models with matrix factorization and how they perform in the Two-Class Collaborative Filtering case with positive and negative feedback.

1.2 Motivation

In this section, we will elaborate our interest in Two-Class Collaborative Filtering Problems (TCCF), as well as our goals that we wish to achieve.

Signed graphs, in which relationships between nodes can be either positive (like) or negative (dislike), are increasingly common in recent years. Other examples for datasets with not only positive but also negative feedback are: explicit up and down votes, friends and foes networks, item ratings with good (4-5 stars) and bad (1-2 stars) feedback. Also implicit negative feedback, such as a manual skip to the next song, aborting watching a movie or closing an ad display can be used.

For instance, MovieLens (Harper and Konstan, 2015) is a movie review website where users can indicate if they like or dislike a movie, by rating them with 1 to 5 stars. Another example is Slashdot-Zoo (Brzozowski et al., 2008; Kunegis et al., 2009), which is a technology related news website where users can tag each other as friends (like) or foes (dislike), based upon their comments on different articles.

Traditionally, collaborative filtering problems consider either binary One-Class Collaborative Filtering (OCCF) for ranking prediction or explicit rating prediction. Ranking prediction aims to rank items (people, in case of social networks), which the user is interested in, on the top of the predicted ranking list. Rating prediction algorithms aim to minimize the differences between the predicted and the actual rating. Whereas ranking prediction works on positive links and ignores negative links, by treating them the same as unknown links, rating prediction cannot explicitly differentiate between positive and negative ratings.

In signed networks however, the aim could be to rank items (people) the user is interested in (positive links, potential likes) on the top and items (people) the user is not interested in (negative links, potential dislikes) at the bottom of a personalized ranking list. We will present different metrics to measure the models' performance on ranking positive items on the top and negative items at the bottom. By explicitly incorporating negative ratings, we want to evaluate the performance of existing collaborative filtering algorithms on signed networks and also introduce new models to improve the algorithms

to these adjusted performance goals. Our goal is to first improve the performance of the models by explicitly incorporating negative samples and second to correctly classify positive and negative samples with high probability.

When a new user appears or a new item is added, we do not have many information nor ratings available to make our decision about recommending existing items for the new user or recommending the new item to existing users. This is referred to as the cold-start problem. We want to evaluate the performance of existing models and our newly introduced models with regard to the cold-start problem.

1.3 Contributions

In this section, we will describe our contributions to current research on recommender systems with this thesis.

In this thesis we present different methods to handle Two-Class Collaborative Filtering problems and analyze their performance on signed networks. Our main contributions are as follows:

1. Based on current state-of-the-art recommendation models, we introduce new methods for Two-Class Collaborative Filtering, by explicitly distinguishing between negative and unknown feedback.

2. We analyze the models' performance for different parameter settings.

3. We examine the performance of our methods on signed datasets.

4. We provide different evaluation measures to compare the models' performance on signed datasets and to measure efficiency of ranking negative feedback at the bottom of personalized ranking lists.

5. To measure the models' sensitivity to cold-start, we provide an analysis on the models' performance for the group with low number of feedback per user.

We adapt Bayesian Personalized Ranking Matrix Factorization BPRMF to explicitly incorporate negative ratings with different sampling methods and introduce Explicit Bayesian Personalized Ranking MF (E-BPRMF) and Proportional Bayesian Personalized Ranking MF (P-BPRMF). Based on Logistic Matrix Factorization, we introduce Multinominal Logistic Matrix Factorization (MLMF) and propose Logistic Likelihood Matrix Factorization (LLMF), to improve ranking of positive feedback and Explicit Logistic Likelihood Matrix Factorization (ELLMF), to distinguish between negative and unknown feedback. We compare our models with different state-of-the-art baseline recommenders, such as BPRMF, SGDReg, Logistic MF and GAUC-OPT. For the evaluation, we implement GAUC-OPT according to Song and Meyer (2015) and port Logistic MF, BPRMF and SGDReg to Python. We evaluate our models together with the baseline recommenders on four openly available benchmark datasets: MovieLens 100K, MovieLens 1M, Slashdot-Zoo and Book Crossing.

We focus on the difference between negative and unknown feedback to solve Two-Class Collaborative Filtering problems and fill the gap in the current research on personalized recommender systems. Based on state-of-the-art recommenders, we propose new methods and provide an analysis of the different approaches.

2

Related Work

In this chapter, we focus on recent research efforts on the topic of Collaborative Filtering and especially Two-Class Collaborative Filtering problems. The names of the models that we implement and use as baseline recommenders (see Section 4.1.1) are printed in bold face.

As shown in Section 1.1, Collaborative Filtering models and especially the use of Matrix Factorization has been widely applied as state-of-the-art method for recommendation systems. After the launch of the Netflix prize (Bennett et al., 2007), collaborative filtering has become a well studied topic and gained research attraction. The idea, that a small number of unobserved features can determine the preferences of a user forms the basis of low dimensional latent factor models, such as Matrix Factorization. Current research can be roughly categorized into two areas: First by introducing and incorporating additional information, like user or item context information within the model and second by modifying the models optimization function and for example, optimize a probabilistic loss function.

On the example of the Netflix prize competition, Koren et al. (2009) show the benefits of Matrix Factorization methods, which allow the incorporation of additional information and are thus superior to classic nearest-neighbor techniques. Using side information, such as implicit feedback, temporal effects and confidence levels, can improve the models' performance significantly. Especially weighting implicit feedback by their confidence and incorporating temporal effects drastically improved matrix factorization models on the Netflix dataset.

Probabilistic approaches introduced by Hofmann (1999), Marlin and Zemel (2004) and others attempt to encode the probability of a user choosing to interact with an item. But they were not meant to handle large datasets, due to exact inference (Welling et al., 2005). Mnih and Salakhutdinov (2007) introduce Probabilistic Matrix Factorization (PMF) which improves those traditional probabilistic approaches and handles very large datasets and users with very few ratings. Their model places zero-mean Gaussian priors on latent user and item features and optimizes the root-mean-square error (RMSE) with quadratic regularization terms, using gradient descent. By introducing constraints on the user-specific feature vectors, they improve their model for users with very few ratings significantly, by defining the user feature vector as a combination of user bias and relative item similarities of rated items for each given user.

Ma et al. (2008) extend PMF with addidtional information from a user-user adjacency matrix, which represents the users' social network. Due to the sparsity of explicit feedback matrices, Collaborative Filtering models often fail to find similar users and cannot handle users who have never rated any items. Traditional recommender systems assume that users and their ratings are independent and identically distributed, and therefore ignore the social connections and their importance for social recommendations, as shown by Sinha and Swearingen (2001) and Bedi et al. (2007). With the incorporation of the users' social network context information, they introduce the SoRec model to overcome these limitations. They factorize the user-item rating matrix together with the user-user adjacency matrix using a common latent user feature vector based on Probabilistic Matrix Factorization. By optimizing the RMSE using gradient descent, they evaluate their model on the Epinions dataset against Probabilistic Matrix Factorization (PMF), Constrained PMF and Maximum Margin Matrix Factorization (MMMF) and come to the conclusion, that SoRec outperforms those models on mean absolute error (MAE).

As most of the previous work on recommendation systems focuses on explicit feedback, which can be difficult to obtain, Oard and Kim (1998) propose to use implicit feedback, such as purchase or browsing history. Hu et al. (2008) introduce Implicit Feedback Matrix Factorization (IMF) to incorporate confidence levels of implicit user feedback and improve traditional Matrix Factorization models to handle implicit feedback. Their main finding is that for implicit observations a preference estimate, whether the user likes or dislikes an item, should be coupled with a confidence level.

By adapting Implicit Feedback Matrix Factorization to PMF, Johnson (2014) use a probabilistic approach on optimizing a logistic likelihood function and introduce the **Logistic Matrix Factorization (Logistic MF)**. They model the probability of a user choosing to interact with an item by a logistic function. They use alternating gradient descent to find a local optimum and drastically improve the convergence by adaptively choosing the gradient step size via AdaGrad (Duchi et al., 2011). Evaluated on the Spotify dataset and measuring Mean Percentage Ranking (MPR), the Logistic MF outperformed IMF.

Rendle et al. (2012) focus on ranked recommendations from implicit feedback and introduce the **Bayesian Personalized Ranking (BPR)** model. They use a boolean rating matrix, where a user has either seen an item or not. They provide a generic optimization criterion, based on the difference of user preferences between two items and build upon stochastic gradient descent with bootstrap sampling. The user's item preference is composed of the two estimators for the single items and thus allows the use of standard collaborative filtering models. The model is evaluated, first using nearest-neighbour and second using matrix factorization techniques. They show that their approach converges much faster and especially, that not only the choice of the right model, but also largely the choice of the right optimization criterion is critical for better prediction quality.

Also Song and Meyer (2015) propose a different optimization criterion, which explicitely takes negative samples into account. They adapt the area under the ROC curve metric (Hanley and McNeil, 1982), which is often used as AUC to measure the perfomance of recommendation systems, to also measure the models' performance on

negative samples. This adapted metric is used to quantify the ranking performance on a signed network graph with positive and negative samples, as a Two-Class Collaborative Filtering problem. They infere an algorithm, by minimizing the generalized AUC loss, using sub-gradient descent and introduce the **GAUC-OPT** optimization criterion. They evaluate their method on the MovieLens 1M dataset, by considering ratings of 4 and 5 as positive links and ratings of 1 and 2 as negative links.

Similar to BPR, Ning and Karypis (2011) looked into ranked recommendation lists and introduced Sparse Linear Methods for Top-N Recommender Systems (SLIM). They propose a simple model to generate top-N recommendations, by aggregating from the user-item rating history. The model consists of a sparse aggregation coefficient matrix, which is used to aggregate an item rank from the user's previous rating history. Under the assumption that the items are independent, the computation can be done in parallel and sparsity can be exploited, to reduce the computational complexity. Nevertheless, compared to matrix factorization models the training of SLIM involves considerably higher computational cost. Therefore Levy and Jack (2013) relax the non-negativity constraint and propose **SGDReg** which allows the learning of item similarities using stochastic gradient descent as described by Tsuruoka et al. (2009).

In this chapter, we described recent research efforts on the topic of Collaborative Filtering and pointed out different approaches, used to improve and adapt existing methods. We will base our work on some of these approaches and use the best recommenders as baseline to compare and evaluate our results.

3

Two-Class Collaborative Filtering Problems

In this chapter, we present the experimental setup and explain the different models and their adaptions for Two-Class Collaborative Filtering (TCCF). We focus on two different kinds of collaborative filtering models: Logistic matrix factorization and Bayesian Personalized Ranking.

We start with the problem definition in Section 3.1 and continue with an explanation of the used notations in Section 3.2.

For logistic matrix factorization in Section 3.3, we first show the adaption of the Logistic Matrix Factorization model from Johnson (2014) to the probabilistic approach, by optimizing the logistic likelihood function. We then adapt it to TCCF and introduce Explicit Logistic Likelihood Matrix Factorization (ELLMF) and add explicit two-class learning, using a Multinominal Logistic Matrix Factorization (MLMF).

In Section 3.4 we show the adaptions of Bayesian Personalized Ranking (BPR) from Rendle et al. (2012) to TCCF by using different sampling methods.

3.1 Problem Definition

Collaborative Filtering (CF) has been widely applied as the state-of-the-art method for recommendation systems. With the motivation of the Netflix Prize, the application of collaborative filtering to model explicit rating feedback has been extensively studied. Oftentimes explicit feedback is not or not yet available, whereas implicit feedback from views, clicks, buys or users' demographic information is readily available or can easily be collected.

Recommendation systems from implicit feedback have been studied by Rendle et al. (2012), Koren et al. (2009), Ning and Karypis (2011) and others. They all focus on solving the Collaborative Filtering problem, where only examples of the positive class are available and negative feedback is mixed together with unknown feedback. This approach is refered to as One-Class Collaborative Filtering, as shown by Pan et al. (2008).

In this work we focus on Two-Class Collaborative Filtering (TCCF) by considering examples from both positive and negative class. We aim to classify new unknown items

into the positive or negative class and therefore distinguish between unknown and negative feedback. We consider cases where explicit or implicit positive and negative feedback is available and aim to predict positive and negative items likewise.

3.2 Formalization

Let U be the set of all users and I the set of all items. We set R to be the user-item rating matrix with positive and negative feedback. Denote with $S^+ \subset U \times I$ the set of implicit/explicit positive feedback, $S^- \subset U \times I$ the set of negative feedback and $S^0 \subset U \times I$ the set of unknown feedback, all of which are non-overlapping.

We always use the subscripts $u \in U$ for an user, $i \in I$ for an item and the superscripts $+$, $-$ and 0 for positive, negative and unknown feedback respectively. $d \in [1, ..., D]$ is used as the index of a D-dimensional latent factor.

Let $\hat{R} \approx R \in \mathbb{R}^{N \times M}$ be the approximation of the user-item rating matrix with N users and M items. Using matrix factorization $A \in \mathbb{R}^{N \times D}$ and $B \in \mathbb{R}^{M \times D}$ will be the user and item latent factors respectively and D the dimensionality of the latent factors. With β_u, β_i we denote the user and item biases. Uppercase characters always represent matrices (e.g. the user-item rating matrix R) whereas lowercase characters represent vectors (e.g. the user latent vector a_u). Rows of matrices are denoted in lower-case, as for example r_u represents the row at position u in the matrix $R = [r_1, r_2, ..., r_N]^T$.

For convenience we define:

$$I_u^+ := \{i \in I : (u, i) \in S^+\}$$
$$I_u^- := \{i \in I : (u, i) \in S^-\}$$

We denote with $\|a_u\|_2^2$ the Euclidean norm, with $\|a_u\|_1$ the L1-norm and with $\delta(x)$ the indicator function as follows:

$$\|a_u\|_2^2 = \sqrt{\sum_{d=1}^{D} \mid a_{u,d} \mid^2}$$
$$\|a_u\|_1 = \sum_{d=1}^{D} \mid a_{u,d} \mid$$
$$\delta(x) = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{else} \end{cases}$$

3.3 Logistic Matrix Factorization

Inspired by Johnson (2014) and Mnih and Salakhutdinov (2007) we looked into probabilistic and logistic matrix factorization and how they could be extended for Two-Class Collaborative Filtering.

We will start with a short review of the Logistic Matrix Factorization model from Johnson (2014). We then introduce an adapted version, by optimizing the likelihood function and introduce Logistic Likelihood Matrix Factorization and further improve this model, to use explicit indicators with Explicit Logistic Likelihood Matrix Factorization. Finally we present the Multinominal Logistic Matrix Factorization, which optimizes a multinominal logistic model for the joint optimization on positive and negative class. We call them LLMF, ELLMF and MLMF respectively.

3.3.1 Logistic Matrix Factorization

The Logistic Matrix Factorization presented by Johnson (2014) uses two low dimensional matrices $A \in \mathbb{R}^{N \times D}$ and $B \in \mathbb{R}^{M \times D}$. It factorizes the observed user-item rating matrix $R \in \mathbb{R}^{N \times M}$ similar to Hu et al. (2008).

Let the probability that a user interacts with an item be distributed according to the logistic function as follows:

$$p(r_{ui}^{+} \mid a_u, b_i, \beta_u, \beta_i) = \frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1 + \exp(a_u b_i^T + \beta_u, + \beta_i)}$$

The bias terms β_u and β_i are bias factors associated with each user u and item i and account for the differences between users and items respectively.

Under the assumption that all observed ratings in R are independent, it optimizes the following likelihood function, where α defines a weighting factor to balance between positive and unkown ratings as suggested by Hu et al. (2008):

$$p^{LK} = \prod_{(u,i) \in U \times I} p(r_{ui}^{+} \mid a_u, b_i, \beta_u, \beta_i)^{\alpha r_{ui}} (1 - p(r_{ui}^{+} \mid a_u, b_i, \beta_u, \beta_i))$$

3.3.2 Logistic Likelihood Matrix Factorization

Based on Johnson (2014), we modify the Logistic Matrix Factorization to optimize the log likelihood function according to James et al. (2014). The goal of Logistic Likelihood Matrix Factorization (LLMF) is first to differentiate between positive and negative or unknown feedback and second to use the resulting matrix factorization, to provide a probability measure for classifying positive and negative feedback.

It differs from the model from Johnson (2014) in the definition of the likelihood function. Instead of always multiplying with the probability of a negative or unknown rating, we only consider the probability of a negative rating for truly negative or unknown samples within the likelihood function. We therefore only account for the loss of false positives, whereas Logistic MF also accounts for the loss of true positives and hence

lowers the probability measure for true positive classifications. With this adaption we expect the model to perform better on arranging truly positive samples on top of a user's ranking list.

To distinct between the positive and negative class, we denote the probability that a user u likes item i analogous to the Logistic Matrix Factorization as:

$$p(r_{ui}^{+} \mid a_u, b_i, \beta_u, \beta_i) = \frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1 + \exp(a_u b_i^T + \beta_u, +\beta_i)}$$

And the probability that user u dislikes item k as:

$$p(r_{ui}^{-} \mid a_u, b_k, \beta_u, \beta_k) = 1 - p(r_{ui}^{+} \mid a_u, b_k, \beta_u, \beta_k) = \frac{1}{1 + \exp(a_u b_k^T + \beta_u, +\beta_k)}$$

For better readability we omit the condintionals and use f to represent the matrix factorization result:

$$\begin{aligned} p(r_{ui}^{+}) &:= p(r_{ui}^{+} \mid a_u, b_i, \beta_u, \beta_i) \\ p(r_{ui}^{-}) &:= p(r_{ui}^{-} \mid a_u, b_k, \beta_u, \beta_k) \\ f &:= a_u b_i^T + \beta_u + \beta_i \end{aligned}$$

We consider the following likelihood function:

$$\begin{aligned} p^{LK} &= \prod_{(u,i) \in U \times I} p(r_{ui}^{+})^{\delta((u,i) \in S^{+})} (1 - p(r_{ui}^{+}))^{1 - \delta((u,i) \in S^{+})} \\ &= \prod_{(u,i) \in U \times I} \frac{\exp(f)}{1 + \exp(f)}^{\delta((u,i) \in S^{+})} \left(1 - \frac{\exp(f)}{1 + \exp(f)}\right)^{1 - \delta((u,i) \in S^{+})} \end{aligned}$$

And the resulting log-likelihood function:

$$\begin{aligned} \log p^{LK} &= \sum_{(u,i) \in U \times I} \log \left(\frac{\frac{\exp(f)}{1+\exp(f)}^{\delta((u,i) \in S^{+})} \frac{1}{1+\exp(f)}}{\frac{1}{1+\exp(f)}^{\delta((u,i) \in S^{+})}} \right) \\ &= \sum_{(u,i) \in U \times I} \delta((u,i) \in S^{+}) \log \left(\frac{\exp(f)}{1 + \exp(f)} \right) + \log \left(\frac{1}{1 + \exp(f)} \right) \\ &\qquad - \delta((u,i) \in S^{+}) \log \left(\frac{1}{1 + \exp(f)} \right) \\ &= \sum_{(u,i) \in U \times I} \delta((u,i) \in S^{+}) f - \delta((u,i) \in S^{+}) \log(1 + \exp(f)) \\ &\qquad - \log(1 + \exp(f)) + \delta((u,i) \in S^{+}) \log(1 + \exp(f)) \\ &= \sum_{(u,i) \in U \times I} \delta((u,i) \in S^{+}) f - \log(1 + \exp(f)) \\ &= \sum_{(u,i) \in U \times I} \delta((u,i) \in S^{+}) (a_u b_i^T + \beta_u + \beta_i) - \log(1 + \exp(a_u b_i^T + \beta_u + \beta_i)) \end{aligned}$$

And aim to maximize the following objective function:

$$\begin{aligned}\max_{a,b,\beta_u,\beta_i} \log p^{LK} - \frac{\lambda}{2}\|a\| - \frac{\lambda}{2}\|b\| = \\ \max_{a,b,\beta_u,\beta_i} \sum_{(u,i)\in U\times I} \delta((u,i)\in S^+)(a_u b_i^T + \beta_u + \beta_i) \\ - \log(1+\exp(a_u b_i^T + \beta_u + \beta_i)) \\ - \frac{\lambda}{2}\|a\| - \frac{\lambda}{2}\|b\|\end{aligned}$$

And therefore the partial derivatives for the user vectors and bias:

$$\begin{aligned}\frac{\partial}{\partial a} &= \delta((u,i)\in S^+)b - b\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)} - \lambda a \\ \frac{\partial}{\partial \beta_u} &= \delta((u,i)\in S^+) - \frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)}\end{aligned}$$

Similarly for the item vectors and item bias:

$$\begin{aligned}\frac{\partial}{\partial b} &= \delta((u,i)\in S^+)a - a\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)} - \lambda b \\ \frac{\partial}{\partial \beta_i} &= \delta((u,i)\in S^+) - \frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)}\end{aligned}$$

3.3.3 Explicit Logistic Likelihood Matrix Factorization

To distinguish between unknown and negative feedback we extend Logistic Likelihood Matrix Factorization by explicitly accounting for negative classes within the likelihood function and introduce the Explicit Logistic Likelihood Matrix Factorization (ELLMF).

In this approach, we modify the likelihood function, to account for the probability of a negative rating, only if the given user rated this item negative. We include positive and negative feedback and ignore unknown feedback within the likelihood function. With this adaption, we expect the model to be able to better differentiate between positive and negative feedback.

For readablility we omit the conditionals and denote the following as in Subsection 3.3.2:

$$\begin{aligned}p(r_{ui}^+) &:= p(r_{ui}^+ \mid a_u, b_i, \beta_u, \beta_i) \\ f &:= a_u b_i^T + \beta_u + \beta_i\end{aligned}$$

We consider the following likelihood function:

$$p^{LK} = \prod_{(u,i)\in U\times I} p(r_{ui}^+)^{\delta((u,i)\in S^+)}(1-p(r_{ui}^+))^{\delta((u,i)\in S^-)}$$

And the resulting log-likelihood function:

$$\begin{aligned}
\log p^{LK} &= \sum_{(u,i)\in U\times I} \log\Big(\frac{\exp(f)}{1+\exp(f)}^{\delta((u,i)\in S^+)} \frac{1}{1+\exp(f)}^{\delta((u,i)\in S^-)}\Big) \\
&= \sum_{(u,i)\in U\times I} \delta((u,i)\in S^+)\log\left(\frac{\exp(f)}{1+\exp(f)}\right) + \delta((u,i)\in S^-)\log\left(\frac{1}{1+\exp(f)}\right) \\
&= \sum_{(u,i)\in U\times I} \delta((u,i)\in S^+)f - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))\log(1+\exp(f)) \\
&= \sum_{(u,i)\in U\times I} \delta((u,i)\in S^+)(a_u b_i^T + \beta_u + \beta_i) \\
&\qquad - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))\log(1+\exp(a_u b_i^T + \beta_u + \beta_i))
\end{aligned}$$

And aim to maximize the following objective function:

$$\begin{aligned}
\max_{a,b,\beta_u,\beta_i} \sum_{(u,i)\in U\times I} & \delta((u,i)\in S^+)(a_u b_i^T + \beta_u + \beta_i) \\
& - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))\log(1+\exp(a_u b_i^T + \beta_u + \beta_i)) \\
& - \frac{\lambda}{2}\|a\| - \frac{\lambda}{2}\|b\|
\end{aligned}$$

And therefore the partial derivatives for the user vectors and bias:

$$\frac{\partial}{\partial a} = \delta((u,i)\in S^+)b - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))b\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)} - \lambda a$$

$$\frac{\partial}{\partial \beta_u} = \delta((u,i)\in S^+) - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)}$$

Analogue the partial derivatives for the item vectors and bias:

$$\frac{\partial}{\partial b} = \delta((u,i)\in S^+)a - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))a\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)} - \lambda b$$

$$\frac{\partial}{\partial \beta_i} = \delta((u,i)\in S^+) - (\delta((u,i)\in S^+) + \delta((u,i)\in S^-))\frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1+\exp(a_u b_i^T + \beta_u + \beta_i)}$$

3.3.4 Multinominal Logistic Matrix Factorization

Logistic MF, LLMF and ELLMF are all limited to predicting a single probability $p(r_{ui}^+)$ and cannot predict the difference between unknown and negative feedback. By introducing a second estimator, we now aim to predict the probabilities, that an item is positive or negative simultanously and are thus able to distinguish positive, negative and unknown feedback. We extend Logistic Matrix Factorization using multinominal logistic regressions as shown by Kwak and Clayton-Matthews (2002) and introduce Multinominal Logistic Matrix Factorization (MLMF). With this approach, we are able to separately predict the probability of positive, negative and unknown samples. Following

this method we expect an increase in prediction accuracy of both positive and negative feedback.

For all users we aim to predict each user-item pair to be part of one of the three classes: positive (+), negative (−) or unknown (0). Assuming independence of irrelevant alternatives, we will learn a combination of two independent logistic regressions, representing positive (P) and negative (N) item membership with common latent user features A and positive (B), respective negative latent item features (C), as shown in Figure 3.1:

$$\text{Positive (P): } \hat{R}^{+} = AB^{T} + \boldsymbol{\beta}_i^{+} + \boldsymbol{\beta}_u$$
$$\text{Negative (N): } \hat{R}^{-} = AC^{T} + \boldsymbol{\beta}_i^{-} + \boldsymbol{\beta}_u$$

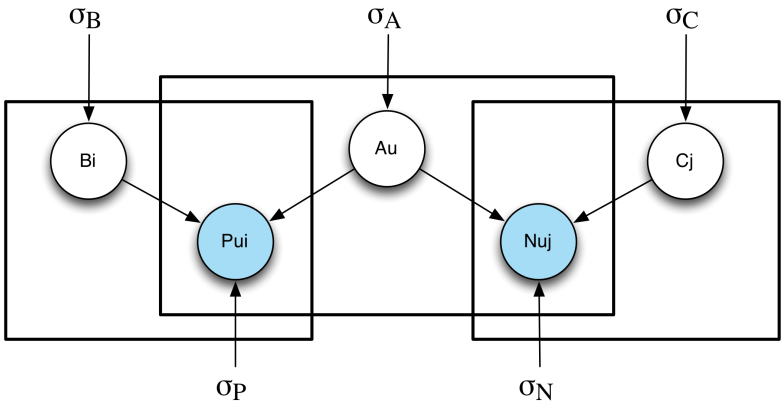


Figure 3.1: Graphical model for Multinominal Logistic Matrix Factorization

We denote the probability that user u likes item i as:

$$p(r_{ui}^{+} \mid a_u, b_i, c_i, \beta_i^{+}, \beta_i^{-}, \beta_u) = \frac{\exp(a_u b_i^T + \beta_i^{+} + \beta_u)}{1 + \exp(a_u b_i^T + \beta_i^{+} + \beta_u) + \exp(a_u c_i^T + \beta_i^{-} + \beta_u)}$$

And the probability that user u dislikes item k as:

$$p(r_{uk}^{-} \mid a_u, b_k, c_k, \beta_i^{+}, \beta_i^{-}, \beta_u) = \frac{\exp(a_u c_k^T + \beta_i^{-} + \beta_u)}{1 + \exp(a_u b_k^T + \beta_i^{+} + \beta_u) + \exp(a_u c_k^T + \beta_i^{-} + \beta_u)}$$

And therefore the probability that it is unknown if the user u likes or dislikes the item j as:

$$\begin{aligned} p(r_{uj}^{0} \mid a_u, b_j, c_j, \beta_i^{+}, \beta_i^{-}, \beta_u) &= 1 - p(r_{uj}^{+} \mid a_u, b_j, c_j, \beta_i^{+}, \beta_i^{-}, \beta_u) - p(r_{uj}^{-} \mid a_u, b_j, c_j, \beta_i^{+}, \beta_i^{-}, \beta_u) \\ &= \frac{1}{1 + \exp(a_u b_j^T + \beta_i^{+} + \beta_u) + \exp(a_u c_j^T + \beta_i^{-} + \beta_u)} \end{aligned}$$

Multinominal Logistic Optimization Criterion

Given the following likelihood function:

$$\begin{aligned} P^{LK} &= \prod_{(u,i)\in U\times I} p(r_{ui}^{+} \mid a_u, b_i, c_i, \beta_i^{+}, \beta_i^{-}, \beta_u)^{\delta((u,i)\in S^{+})} p(r_{ui}^{-})^{\delta((u,i)\in S^{-})} \\ &\qquad p(r_{ui}^{0} \mid a_u, b_i, c_i, \beta_i^{+}, \beta_i^{-}, \beta_u)^{\delta((u,i)\in S^{0})} \\ &= \prod_{(u,i)\in U\times I} p(r_{ui}^{+})^{\delta((u,i)\in S^{+})} p(r_{ui}^{-})^{\delta((u,i)\in S^{-})} (1 - p(r_{ui}^{+}) - p(r_{ui}^{-}))^{\delta((u,i)\in S^{0})} \end{aligned}$$

We get this logistic likelihood function which we want to maximize:

$$\begin{aligned} \log P^{LK} = \sum_{(u,i)\in U\times I} & \delta((u,i)\in S^{+})(a_u b_i^T + \beta_i^{+} + \beta_u) \\ & + \delta((u,i)\in S^{-})(a_u c_i^T + \beta_i^{-} + \beta_u) \\ & - \log\left(1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}\right) \end{aligned}$$

And therefore the partial derivatives for the user and item features and bias:

$$\frac{\partial}{\partial a_u} = \begin{cases} b_i - (b_i e^{a_u b_i^T + \beta_i^{+} + \beta_u} + c_i e^{a_u c_i^T + \beta_i^{-} + \beta_u}) \frac{1}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{+}, \\ c_i - (b_i e^{a_u b_i^T + \beta_i^{+} + \beta_u} + c_i e^{a_u c_i^T + \beta_i^{-} + \beta_u}) \frac{1}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{-}, \\ -(b_i e^{a_u b_i^T + \beta_i^{+} + \beta_u} + c_i e^{a_u c_i^T + \beta_i^{-} + \beta_u}) \frac{1}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial b_i} = \begin{cases} a_u - a_u \frac{e^{a_u b_i^T + \beta_i^{+} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{+}, \\ -a_u \frac{e^{a_u b_i^T + \beta_i^{+} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial c_i} = \begin{cases} a_u - a_u \frac{e^{a_u c_i^T + \beta_i^{-} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{-}, \\ -a_u \frac{e^{a_u c_i^T + \beta_i^{-} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial \beta_i^{+}} = \begin{cases} 1 - \frac{e^{a_u b_i^T + \beta_i^{+} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{+}, \\ -\frac{e^{a_u b_i^T + \beta_i^{+} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial \beta_i^{-}} = \begin{cases} 1 - \frac{e^{a_u c_i^T + \beta_i^{-} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{if } (u,i)\in S^{-}, \\ -\frac{e^{a_u c_i^T + \beta_i^{-} + \beta_u}}{1 + e^{a_u b_i^T + \beta_i^{+} + \beta_u} + e^{a_u c_i^T + \beta_i^{-} + \beta_u}} & \text{else} \end{cases}$$

3.4 Bayesian Personalized Ranking

Inspired by Rendle et al. (2012), we looked into Bayesian Personalized Ranking and how this could be adapted for Two-Class Collaborative Filtering.

We will start with a short review of the Bayesian Personalized Ranking (BPR) model from Rendle et al. (2012). We then introduce two adapted versions, by using different sampling techniques, to incorporate negative feedback into the BPR model.

3.4.1 Bayesian Personalized Ranking

The Bayesian Personalized Ranking as suggested by Rendle et al. (2012) presents a generic optimization criterion, derived from the maximum posterior estimator for optimal personalized ranking.

They optimize the total ranking of each user $u >_u$, such that if user u prefers item i over item j, $i >_u j$ holds. They let $\hat{r}_{uij}$ be the relationship between user u, item i and item j, such that $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$ and maximize the difference between preferred and unknown/disliked items:

$$\text{BPR-OPT} = \sum_{u \in U} \sum_{i \in I_u^+} \sum_{j \in I \backslash I_u^+} \log \frac{1}{1 + \exp(\hat{r}_{uj} - \hat{r}_{ui})} + \lambda_u \|a_u\|^2 + \lambda_i \|b_i\|^2$$

Θ represents the parameter vector of an arbitrary model class (e.g. matrix factorization).

They propose a LearnBPR algorithm which optimizes BPR-OPT by randomly sampling from $D_S := \{(u, i, j) \mid i \in I_u^+ \cap j \in I \backslash I_u^+\}$ using stochastic gradient descent:

Algorithm 1 LearnBPR (D_S, Θ)

1: initialize Θ
2: **repeat**
3: draw (u, i, j) from $D_S := \{(u, i, j) \mid i \in I_u^+ \cap j \in I \backslash I_u^+\}$
4: $\Theta \leftarrow \Theta + \alpha(\frac{e^{-\hat{r}_{uij}}}{1+e^{-\hat{r}_{uij}}} \frac{\partial}{\partial \Theta} \hat{r}_{uij} + \lambda_\Theta \Theta)$
5: **until** convergence

3.4.2 Explicit BPR (E-BPR)

Sine LearnBPR ignores the difference between known disliked items and unknown items, we propose a different sampling technique for LearnBPR and introduce Explicit BPR (E-BPR). We aim to maximize the difference between known positive and known negative samples and sample from triples, each consisting of one positive and one negative sample for each user. We therefore sample from the follwoing set D_S, which we define as follows:

$$D_S := \{(u, i, j) \mid i \in I_u^+ \cap j \in I_u^-\}$$

Algorithm 2 LearnEBPR (D_S, Θ)

initialize Θ
repeat
    draw (u, i, j) from $D_S := \{(u, i, j) \mid i \in I_u^+ \cap j \in I_u^-\}$
    $\Theta \leftarrow \Theta + \alpha(\frac{e^{-\hat{r}_{uij}}}{1+e^{-\hat{r}_{uij}}} \frac{\partial}{\partial \Theta} \hat{r}_{uij} + \lambda_\Theta \Theta)$
until convergence

3.4.3 Proportional BPR (P-BPR)

Since E-BPR only samples from known feedback and therefore has less data to train on, we introduce P-BPR to address this limitation.

With P-BPR we introduce two kind of triples: In the first class each triple consists of a positive and an unknown sample per user and in the second class each triple consists of a unknown and a negative sample. We then sample from positive and unknowns, as well as from negative and unkowns, proportional to ratio of positive versus negative ratings in the training dataset:

Let η be the proportion of number of positive to the number of total known samples in the training dataset:

$$\eta := \frac{\mid I_u^+ \mid}{\mid I_u^+ \mid + \mid I_u^- \mid}$$

With probability η we then sample from:

$$D_S^+ := \{(u, i, j) \mid i \in I_u^+ \cap j \in (I_u^0 \cup I_u^-)\}$$

And with probability $1 - \eta$ we sample from:

$$D_S^- := \{(u, i, j) \mid i \in (I_u^0 \cup I_u^+) \cap j \in I_u^-\}$$

Algorithm 3 LearnPBPR $(\eta, D_S^+, D_S^-, \Theta)$

initialize Θ
repeat
    if random <= η **then**
        draw (u, i, j) from D_S^+
    else
        draw (u, i, j) from D_S^-
    $\Theta \leftarrow \Theta + \alpha(\frac{e^{-\hat{r}_{uij}}}{1+e^{-\hat{r}_{uij}}} \frac{\partial}{\partial \Theta} \hat{r}_{uij} + \lambda_\Theta \Theta)$
until convergence

4

Evaluation

In the previous chapter we introduced new methods based upon Logistic MF by Johnson (2014) and Bayesian Personalized Ranking by Rendle et al. (2012). In the following sections we will evaluate our proposed methods and compare it to existing models, used as baseline in terms of accuracy and different other performance metrics.

We will start with a description of the methodology of our evaluations in Section 4.1, where we describe the baseline recommenders, present a descriptive analysis of the datasets and describe the performance metrics.

We then present our results of the models' performance on the chosen datasets in Section 4.2, as well as an evaluation, grouped by number of ratings per user, to analyze the cold-start problem and discuss the results.

4.1 Methodology

This section starts with an overview of the baseline recommenders considered in the evaluation, followed by a description of the dataset characteristics. We then describe the performance metrics, which we used to evaluate our models.

4.1.1 Baseline Recommenders

We compare our models, introduced in Chapter 3, in terms of accuracy and different other metrics, as described in Section 4.1.3, with Bayesian Personalized Ranking Matrix Factorization (**BPRMF**) from Rendle et al. (2012), **SGDReg** a variant of Sparse Linear Methods (SLIM) from Levy and Jack (2013), **Logistic MF** from Johnson (2014) and an optimization of a gerneralized AUC metric (**GAUC-OPT**) from Song and Meyer (2015).

We experimented with different numbers of latent factors, regularizers and learn rates to maximize the accuracy of each model. The complete set of parameters used for each recommender is shown in Appendix A.1.

The following paragraphs will give a short introduction of the baseline recommenders and their implementation used for the evaluations.

Bayesian Personalized Ranking Matrix Factorization

We use Rendle et al. (2012) Bayesian Personalized Ranking Matrix Factorization (BPRMF) implementation from MyMediaLite[1] and ported it to Python.

This implementation uses stochastic gradient descent, as suggested by the paper, as well as an item bias to capture the differences between items.

Sparse Linear Methods

We use Mendeley's Python implementation[2] of SGDReg as described by Levy and Jack (2013). SGDReg is a variant of Sparse Linear Methods (SLIM), proposed by Ning and Karypis (2011).

This implementation uses the *SGDRegressor* from scikit-learn for stochastic gradient descent, to compute the similarities between items. This implementation is considerably faster than the original SLIM, due to relaxed non-negativity constraints.

Logistic Matrix Factorization

For Logistic Matrix Factorization (Logistic MF) from Johnson (2014), we use his Python implementation[3].

This implementation uses gradient descent with AdaGrad as proposed by Duchi et al. (2011), to increase convergence and adds user and item biases.

GAUC-OPT

As described by Song and Meyer (2015), we implemented the optimization of a gerneralized AUC (GAUC-OPT) in Python, based on scikit-learn from Pedregosa et al. (2011).

4.1.2 Datasets

We evaluate the performance of the different methods on the datasets from MovieLens (100K and 1M), Slashdot-Zoo and Book Crossing. The characteristics of those datasets are summarized in Table 4.1.

We use 5-fold cross validation, to evaluate the models. For each of the 5 folds, the testset is generated by randomly sampling 30% of the total ratings from users which have at least five positive ratings and three negative ratings, and items which have at least one positive and one negative rating.

[1] MyMediaLite BPRMF Implementation - *https://github.com/zenogantner/MyMediaLite/blob/master/src/MyMediaLite/ItemRecommendation/BPRMF.cs*

[2] Mendeley SGDReg Implementation - *https://github.com/Mendeley/mrec/blob/master/mrec/item_similarity/slim.py*

[3] Johnson's Logistic MF Implementation - *https://github.com/MrChrisJohnson/logistic-mf/blob/master/logistic_mf.py*

Name	Ratings	Positive Ratings Share	Users	Items	Min User Degree	Max User Degree	Min Item Degree	Max Item Degree	Average User Degree	Average Item Degree	Median User Degree	Median Item Degree	Density
MovieLens 100K - total	99649	54.38%	943	1682	8	737	1	578	105.7	59.2	64	27	6.2825%
MovieLens 100K - train	69755	54.07%	943	1682	8	556	1	394	74	41.5	45	19	4.3978%
MovieLens 100K - test	29894	55.12%	943	1628	0	193	0	184	31.7	18.4	20	9	1.9472%
MovieLens 1M - total	998087	54.42%	6040	3706	9	2314	1	3416	165.2	269.3	95	123	4.4589%
MovieLens 1M - train	698661	54.36%	6040	3706	8	1613	1	2364	115.7	188.5	66	86	3.1212%
MovieLens 1M - test	299426	54.56%	6040	3706	0	701	0	1052	49.6	80.8	28	37	1.3377%
Slashdot-Zoo Small - total	202831	75.69%	3745	4769	20	345	3	972	54.2	42.5	38	25	1.1357%
Slashdot-Zoo Small - train	141982	82.00%	3745	4769	8	345	3	661	37.9	29.8	27	18	0.7950%
Slashdot-Zoo Small - test	60849	60.97%	3745	4769	0	171	0	311	16.2	12.8	9	8	0.3407%
Slashdot-Zoo Big - total	305583	75.74%	7896	9504	10	377	1	1520	38.7	32.2	22	16	0.4072%
Slashdot-Zoo Big - train	213909	82.64%	7896	9504	8	377	1	1096	27.1	22.5	16	12	0.2850%
Slashdot-Zoo Big - test	91674	59.63%	7895	9504	0	199	0	424	11.6	9.6	0	4	0.1222%
Book Crossing - total	306669	28.79%	3398	14841	15	4708	1	793	90.2	20.7	45	13	0.6081%
Book Crossing - train	214669	27.28%	3398	14841	8	3279	1	522	63.2	14.5	32	10	0.4257%
Book Crossing - test	92000	32.30%	3398	14839	0	1429	0	271	27.1	6.2	14	4	0.1825%

Table 4.1: Dataset Properties

MovieLens The MovieLens 1M (100K) dataset[4] contains 1,000,209 (100,000) movie ratings obtained from the MovieLens research project (Harper and Konstan, 2015).

On both datasets we apply mean-removal, as suggested by Wang et al. (2006); James et al. (2014) and others. For each user we subtract the mean rating of the given user from his ratings. The resulting ratings below zero are then mapped as negative samples and ratings above zero as positive samples, while zero-ratings are left out.

From the remaining 998,087 (99,649) ratings we select 299,426 (29,894) ratings for the testset at random, under the condition that each user in the trainset has at least five positive and three negative ratings and each item at least one positive and one negative rating.

The MovieLens dataset is the most dense dataset in our evaluation and positive and negative ratings are equally represented, with a share of positive to negative ratings of roughly 54%.

Slashdot-Zoo The Slashdot-Zoo[5] is a signed social network of users of the technology news site Slashdot (slashdot.org) (Kunegis et al., 2009). It consists of an adjacency matrix, describing the friends and foes relations between users of the news site.

For the Slashdot-Zoo Small dataset we remove all items with less than twenty ratings and then only select users with at least twenty ratings. From the remaining 202,831 ratings we select 60,849 ratings for the testset at random, under the condition that each user in the trainset has at least five positive and three negative ratings and each item at least one positive and one negative rating.

The Slashdot-Zoo Big dataset is constructed by first removing all items with less than ten ratings and then selecting all users with twenty or more ratings. From the remaining 305,583 ratings we select 91,674 ratings for the testset at random, under the condition that each user in the trainset has at least five positive and three negative ratings and each item at least one positive and one negative rating.

The Slashdot-Zoo Big dataset is the most sparse dataset in our evaluation and positive samples are over represented, with a share of positive to negative samples of roughly 75%.

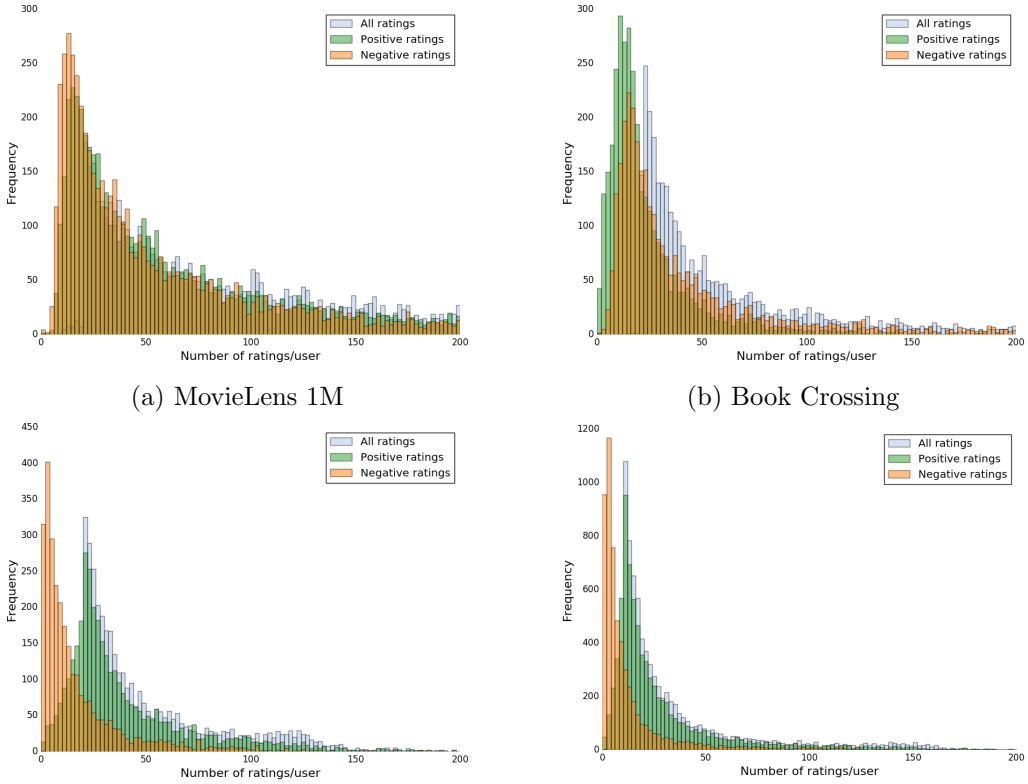
[4]MovieLens Dataset - *http://grouplens.org/datasets/movielens/*

[5]Slashdot-Zoo Dataset - *http://konect.uni-koblenz.de/networks/slashdot-zoo*

Book Crossing The Book Crossing dataset[6] contains book ratings from the Book Crossing community (Ziegler et al., 2005). We only consider users with at least fifteen ratings and then apply mean-removal. For each user we subtract the mean rating of the given user from his ratings. The resulting ratings below zero are then mapped as negative samples and ratings above zero as positive samples, while zero-ratings are left out.

From the remaining 306,669 ratings we select 92,000 ratings for the testset at random, under the condition that each user in the trainset has at least five positive and three negative ratings and each item at least one positive and one negative rating.

The Book Crossing dataset is relatively sparse on ratings per item, but relatively dense on ratings per user, as each user on average has roughly 90 ratings and negative ratings are over represented, with a share of positive to negative ratings of roughly 29%.



(a) MovieLens 1M

(b) Book Crossing

(c) Slashdot-Zoo Small

(d) Slashdot-Zoo Big

Figure 4.1: Dataset histograms of number of ratings per user

[6]Book Crossing Dataset - *http://www2.informatik.uni-freiburg.de/~cziegler/BX/*

4.1.3 Performance Metrics

To evaluate our models, we measure AUC, neg-AUC, GAUC, HR@10, neg-HR@10, MRR and positive, negative and overall classification performance. We then compare the evaluation results of our methods with the evaluation results of the baseline recommenders. In the following paragraphs, we will explain the different metrics, how they are measured and what they are used for.

AUC AUC measures the area under the receiver operating characteristic (ROC) curve, which is the curve plotted by the true positive rate against the false positive rate (Fawcett, 2006). It measures the expected proportion of positive samples which are ranked before a uniformly drawn random negative sample.

neg-AUC We define the negative-AUC as the ROC curve of negative samples on the inverse ranking list, as used by Song and Meyer (2015) as the second part of their generalized AUC measure (GAUC). With the negative-AUC we measure the efficiency of ranking negative samples at the bottom of a personalized ranking list.

Generalized AUC (GAUC) The generalized AUC (GAUC) was introduced by Song and Meyer (2015). It measures the combined efficiency of ranking positive samples at the top and negative samples at the bottom of a personalized ranking list. When $\eta \in [0, 1]$ is the fraction of positive samples in the dataset, $GAUC$ is defined as follows:

$$GAUC = \eta \text{AUC} + (1 - \eta)\text{neg-AUC}$$

Mean Reciprocal Rank (MRR) The mean reciprocal rank is the multiplicative inverse of the rank of the first correctly classified sample (Craswell, 2009). If we have N users and define rank_i as the rank of the first correctly classified sample of user i, the MRR is defined as:

$$MRR = \frac{1}{N} \sum_{i}^{N} \frac{1}{\text{rank}_i}$$

Precision (Prec) Precision@K measures the true positives within a personalized ranking list with K recommendations:

$$\text{Prec@K} = \frac{\text{Relevant true positives within top-K ranking list}}{K}$$

In all of our experiments we set $K = 20$ and look at the precision for top-20 ranked recommendations.

Negative Precision (neg-Prec) We define the negative Precision@K as the precision of the bottom-K ranking lists, where we are looking for correctly classified negative samples.

Hit Rate (HR) A *hit* is when a testset item of a user is correctly classified within the top-N items recommended by the model. The hit rate is the number of hits divided by the number of users:

$$\text{hit rate (HR)} = \frac{\text{Number of hits}}{\text{Number of users}}$$

In all of our experiments we set $N = 10$ and look at the hit-rate for top-10 ranked recommendations.

Negative Hit Rate (neg-HR) We define the negative hit rate as the hit-rate of the bottom-N ranking lists, where a hit correctly classifies a negative sample.

Classification Score The classification score is the fraction of correctly classified samples from the testset.

$$\text{Classification} = \frac{\text{Number of correctly classified samples}}{\text{Number of samples}}$$

For all logistic models we use the resulting probability approximation to classify each item for a given user. For simplicity we use 0.5 as the decision boundary to classify an item. We classify an item as positive, if the score is greater than 0.5 and negative, if it is lower than 0.5 and unknown else. For the non-logistic models we pass the resulting user-item score through the expit function and classify the item positive, if the result is greater than 0.5 and negative, if it is lower than 0.5 and unknown when equal to 0.5.

Positive Classification Score The positive classification score is the fraction of correctly classified positive samples from the testset.

$$\text{pos-Classification} = \frac{\text{Number of correctly classified positive samples}}{\text{Number of positive samples}}$$

Negative Classification Score The negative classification score is the fraction of correctly classified negative samples from the testset.

$$\text{neg-Classification} = \frac{\text{Number of correctly classified negative samples}}{\text{Number of negative samples}}$$

4.2 Results

In the following subsections we will analyze the results of our evaluations. We use the metrics as defined in Section 4.1.3, to evaluate our methods which we introduced in Chapter 3 and compare the results with an evaluation of the baseline recommenders which we presented in Section 4.1.1.

We first give an overview of the evaluation on the four benchmark datasets and then focus on the performance on negative samples and finally the performance for different groups of users, depending on how many samples they have.

We present our results in Table 4.2 which lists the performance of all models on all analyzed datasets. The model score with the best performance per dataset and metric is highlighted as bold text, the second best as bold italic text and the third best as italic text.

Table 4.2: Performance Evaluation Results

MODEL	AUC	neg-AUC	GAUC	PREC@20	neg-PREC@20	HR@10	neg-HR@10	MRR	CLASS	pos-CLASS	neg-CLASS
Slashdot Small											
LLMF	*0.724126*	0.393095	0.595003	***0.063111***	0.001687	***0.438996***	0.014649	**0.250971**	0.378585	0.002771	0.999243
ELLMF	0.657225	*0.684686*	***0.667937***	0.059432	**0.025977**	*0.421357*	**0.222619**	***0.228782***	**0.762338**	**0.846470**	0.442911
MLMF	0.600100	0.537168	0.575547	0.015396	0.003564	0.158395	0.046522	0.084970	0.623118	*0.748074*	0.395705
E-BPRMF	0.557043	0.661421	0.597758	0.032032	0.017745	0.291421	*0.169976*	0.146497	0.625323	0.597846	**0.680758**
P-BPRMF	0.654215	0.603060	0.634262	0.047937	0.017904	0.366660	0.167332	0.193450	*0.670455*	0.684180	*0.632747*
Logistic MF	0.537723	***0.687066***	0.595979	0.043098	0.018354	0.348925	0.162964	0.183917	0.377297	0.001293	0.999339
BPRMF	***0.749500***	0.385496	0.607516	**0.064524**	0.001427	**0.440860**	0.012762	*0.224639*	0.634791	***0.784717***	0.374940
SGDReg	**0.936589**	**0.858273**	**0.906040**	0.053235	***0.023822***	0.349490	***0.195122***	0.180746	0.287030	0.336997	0.145674
GAUC-OPT	0.695803	0.576150	*0.649133*	*0.061812*	*0.019449*	0.377762	0.156826	0.205772	***0.684816***	0.694211	***0.639348***
Slashdot Big											
LLMF	*0.757988*	0.337833	0.588815	0.048189	0.000663	*0.363434*	0.006879	*0.195566*	0.369203	0.000720	0.999586
ELLMF	0.672366	*0.654335*	***0.665108***	*0.051079*	**0.021179**	***0.372988***	**0.179684**	***0.201080***	**0.740065**	**0.861317**	0.397691
MLMF	0.564467	0.527741	0.549679	0.009846	0.002676	0.099375	0.033513	0.053732	0.619168	*0.739389*	0.397854
E-BPRMF	0.547193	***0.693258***	0.606007	0.030966	0.011553	0.293299	0.116929	0.165516	0.623269	0.575551	**0.712303**
P-BPRMF	0.671968	0.564819	*0.628826*	0.039502	*0.012366*	0.324489	0.119938	0.161641	***0.656001***	0.693319	***0.587936***
Logistic MF	0.650555	0.592037	0.626999	0.043451	0.012264	0.347415	*0.128597*	0.191414	0.368698	0.000158	0.999777
BPRMF	***0.785965***	0.315267	0.596441	***0.051358***	0.000543	**0.398408**	0.004543	**0.228243**	*0.629252*	***0.804285***	0.321379
SGDReg	**0.848944**	**0.725215**	**0.799124**	**0.054853**	***0.020956***	0.356565	***0.173430***	0.182425	0.145517	0.185980	0.059004
GAUC-OPT	0.710599	0.458956	0.609276	0.034270	0.006064	0.250252	0.049249	0.110556	0.626640	0.651901	*0.572283*
MovieLens 100K											
LLMF	***0.926161***	0.141509	0.574878	**0.216461**	0.000572	***0.820166***	0.006258	**0.539099**	0.473723	0.063706	0.982331
ELLMF	0.668456	0.489046	0.588125	0.066526	0.022670	0.433962	0.196342	0.223776	***0.620862***	0.652807	**0.541347**
MLMF	0.857779	***0.799525***	***0.831698***	0.117458	***0.054354***	0.668414	***0.426971***	0.399022	0.550823	0.579942	0.509421
E-BPRMF	0.712256	0.461402	0.599950	0.058829	0.017604	0.350672	0.140833	0.172039	**0.625844**	**0.762798**	0.456531
P-BPRMF	0.653401	0.520645	0.593965	0.103017	0.039051	0.597702	*0.312133*	0.322193	0.605784	*0.661808*	*0.536605*
Logistic MF	0.834838	*0.729845*	*0.787833*	0.074015	0.008318	0.450589	0.078209	0.253671	0.466776	0.050882	0.982376
BPRMF	*0.920216*	0.148978	0.574938	***0.201443***	0.000820	**0.843939**	0.008205	***0.536773***	0.575177	0.973313	0.079483
SGDReg	**0.935712**	**0.906247**	**0.922522**	*0.168282*	**0.065699**	*0.741169*	**0.461564**	*0.489900*	0.469766	0.575537	0.306098
GAUC-OPT	0.679730	0.497579	0.598171	0.114691	*0.041006*	0.534560	0.301295	0.305898	*0.619250*	***0.668715***	***0.540359***
MovieLens 1M											
LLMF	*0.909784*	0.181650	0.578293	*0.195740*	0.000359	*0.723537*	0.003355	*0.481469*	0.459819	0.028412	0.994418
ELLMF	0.732613	0.467373	0.611860	0.071138	0.009247	0.411667	0.074785	0.237644	**0.666695**	*0.698186*	**0.579731**
MLMF	0.826759	***0.766725***	***0.799424***	0.107078	***0.041287***	0.562421	***0.298915***	0.333294	0.552702	0.571933	0.529544
E-BPRMF	0.749255	0.450062	0.613044	0.059569	0.015914	0.363450	0.129379	0.181292	*0.637917*	**0.800220**	0.445077
P-BPRMF	0.620973	0.535946	0.582261	0.115649	0.033051	0.576353	*0.264986*	0.345146	0.595742	0.629230	*0.555121*
Logistic MF	0.825242	*0.718973*	*0.776862*	0.061672	0.005620	0.405542	0.047912	0.207474	0.465928	0.046130	0.982942
BPRMF	***0.935458***	0.149602	0.577689	**0.212219**	0.000451	**0.821569**	0.003758	***0.515051***	0.576689	0.985294	0.069779
SGDReg	**0.965631**	**0.921785**	**0.945671**	***0.197188***	**0.074244**	***0.774217***	**0.461403**	**0.526834**	0.311462	0.425746	0.154090
GAUC-OPT	0.715029	0.500948	0.617565	0.113455	*0.036221*	0.531363	0.235935	0.303964	***0.652390***	***0.701624***	***0.559945***
Book Crossing											
LLMF	*0.692916*	0.299469	0.426901	*0.006574*	0.000395	*0.108801*	0.003679	*0.057027*	*0.569662*	0.000001	0.999997
ELLMF	0.507893	0.517435	0.514344	*0.006574*	***0.014449***	0.071330	0.129566	0.037438	**0.587974**	0.362706	**0.659751**
MLMF	0.548991	*0.585068*	*0.573381*	0.002314	0.007374	0.024587	0.071570	0.011531	0.510171	0.439392	***0.552887***
E-BPRMF	0.544127	0.469225	0.493486	0.005838	0.012244	0.066268	*0.141542*	0.036671	0.499852	***0.546384***	0.465661
P-BPRMF	0.528529	0.488930	0.501754	0.005759	***0.014449***	0.069752	***0.148563***	0.037561	0.506110	*0.531108*	0.486498
Logistic MF	0.629020	***0.724790***	***0.693773***	0.000437	***0.014449***	0.004338	0.009769	0.003228	***0.569812***	0.000406	0.999777
BPRMF	***0.714570***	0.274596	0.417098	***0.013770***	0.000378	***0.137859***	0.003745	***0.074411***	0.456963	**0.721424**	0.257117
SGDReg	**0.866276**	**0.878845**	**0.874775**	**0.021583**	**0.030091**	**0.201760**	**0.226219**	**0.131181**	0.198660	0.144702	0.203727
GAUC-OPT	0.486546	0.521514	0.510184	0.004694	*0.013635*	0.049306	0.120667	0.025538	0.516304	0.489095	*0.520106*

Performance evaluation results of all models on all datasets. The best score per dataset and metric is highlighted as bold text, the second best as bold italic text and the third best as italic text.

4.2.1 Performance

In this subsection, we will first explain the results of the evaluation, measured by AUC, Precision@20, Hit-Rate@10 and MRR, followed by a discussion of the results.

Performance measured by AUC

When comparing the models' performance measured by AUC, we see that SGDReg always outperforms all other models, followed by BPRMF and LLMF.

Using Logistic Likelihood Matrix Factorization (LLMF), we see an increase in AUC, compared to Logistic MF. This is the expected increase, as we adapted LLMF to promote correctly classified positive samples more than the optimization used by Logistic MF.

The newly introduced methods ELLMF, E-BPRMF and P-PRMF all experience a performance drop in AUC, compared to their original models Logistic MF and BPRMF. This is mainly due to the fact that all of those models do not only optimize for the scores for positive samples, but also take negative samples into account. We will discuss later in Subsection 4.2.2 how the trade-off between optimizing for positive and negative samples will let those models outperform on other evaluation metrics.

The advantage of SGDReg is bigger for more sparse and skewed datasets like Slashdot-Zoo and Book Crossing, by scoring around 15 to 20 percentage points higher than the second best model. On the relatively dense and balanced MovieLens datasets, the difference between the models' performance is much smaller, with around 1 to 3 percentage points difference from SGDReg to the second best model.

Performance measured by Precision@20

When looking at Precision@20, BPRMF is always either the best or second best model, followed by SGDReg and LLMF. In Table 4.3 we list the Precision@20 results for better readability, together with the standard deviation in parenthesis.

Table 4.3: Precision@20

Model	Slashdot Small	Slashdot Big	MovieLens 100K	MovieLens 1M	Book Crossing
Logistic MF	0.043098 (1.0947e-03)	0.043451 (8.8543e-04)	0.074015 (8.1101e-03)	0.061672 (6.1222e-03)	0.000437 (6.9332e-05)
LLMF	***0.063111 (1.1673e-03)***	0.048189 (1.0437e-03)	**0.216461 (1.5342e-03)**	*0.195740 (1.3623e-03)*	*0.006574 (2.2689e-04)*
ELLMF	0.059432 (3.7589e-04)	*0.051079 (6.2329e-04)*	0.066526 (2.2938e-03)	0.071138 (5.8294e-04)	*0.006574 (2.2689e-04)*
Multinominal Log MF	0.015396 (1.3663e-03)	0.009846 (8.6182e-04)	0.117458 (4.2184e-03)	0.107078 (4.7601e-03)	0.002314 (1.2548e-04)
BPRMF	**0.064524 (5.7527e-04)**	***0.051358 (8.7915e-04)***	***0.201443 (1.2201e-03)***	**0.212219 (1.5865e-03)**	***0.013770 (3.3396e-04)***
E-BPRMF	0.032032 (6.6991e-04)	0.030966 (1.0051e-03)	0.058829 (9.5130e-04)	0.059569 (6.8765e-04)	0.005838 (1.8809e-04)
P-BPRMF	0.047937 (4.9809e-04)	0.039502 (9.5701e-04)	0.103017 (2.1777e-03)	0.115649 (9.2683e-04)	0.005759 (1.6083e-04)
SGDReg	0.053235 (9.8206e-04)	**0.054853 (9.3728e-04)**	*0.168282 (3.2847e-03)*	***0.197188 (3.4452e-04)***	**0.021583 (4.7490e-04)**
GAUC-OPT	*0.061812 (6.3063e-04)*	0.034270 (5.4378e-03)	0.114691 (4.0924e-03)	0.113455 (1.3474e-03)	0.004694 (2.1783e-04)

Precision@20 performance of all evaluated models on all datasets. The best score per dataset is highlighted as bold text, the second best as bold italic text and the third best as italic text. The numbers in parenthesis denote the standard deviation.

While LLMF performs well on the small and dense MovieLens 100K and Slashdot-Zoo Small datasets, SGDReg handles the skewed big Slashdot-Zoo Big and Book Crossing dataset better. Multinominal Log MF fails to list positive samples on the top of the ranking list, when the dataset is relatively sparse, like the Slashdot-Zoo Big and Book

Crossing datasets. While GAUC-OPT manages to perform well on the Slashdot-Zoo Small dataset, it fails on all other datasets.

LLMF and ELLMF both outperform Logistic MF on all except the small and dense MovieLens 100K dataset.

Performance measured by Hit-Rate@10

Comparing the models' performance on Hit-Rate@10, we examine whether a model is able to present at least one positive sample within the top 10 ranking list of each user.

Similar to the Precision@20 results, BPRMF outperforms most of the other models, as it explicitly optimizes the ranking list, followed by SGDReg and LLMF. Also ELLMF does a really good job on the Slashdot-Zoo datasets, while E-BPRMF and P-BPRMF do not perform as well as BPRMF, since they are shifted towards negative samples, as we will see in Subsection 4.2.2.

LLMF and ELLMF both outperform Logistic MF on all except the small and dense MovieLens 100K dataset.

Performance measured by MRR

While LLMF performs great for MRR, evaluated on the dense MovieLens 100K and Slashdot-Zoo Small datasets, SGDReg outperforms other models on the sparse Book Crossing and MovieLens 1M datasets.

LLMF and ELLMF both outperform Logistic MF on all except the small and dense MovieLens 100K dataset.

BPRMF is always within the top three models for MRR and has the best performance on the Slashdot-Zoo Big dataset. Again the Multinominal Log MF fails on sparse, large datasets.

Performance measured by Classification Score

ELLMF outperforms all other models, with an advantage of 3 to 9 percentage points to the second best model GAUC-OPT or P-BPRMF. Except on the small MovieLens 100K dataset, where they are slightly outmatched by E-BPRMF.

While ELLMF always outperforms Logistic MF, LLMF results are similar to the ones from Logistic MF. P-BPRMF outmatches BPRMF, except for the small MovieLens 100K dataset. Multinominal Log MF, E-BPRMF and GAUC-OPT perform nearly as good, whereas Logistic MF, LLMF and SGDReg are quite off.

Conclusion

Even though SGDReg is a rather simple model, it performs great for AUC optimization tasks. When looking for optimized Top-K ranking lists, BPRMF outperforms most of the other models on Hit-Rate@10.

By adapting Logistic MF to negative samples and introducing LLMF and ELLMF, we are able to outperform the original Logistic MF model in terms of AUC, Precision@20, Hit-Rate@10 and MRR on almost all datasets. The adaption of BPRMF to negative samples with the introduction of E-BPRMF and P-BPRMF did not yield an improvement on those metrics, but they still outperform BPRMF in classification accuracy. Multinominal Log MF, as well as GAUC-OPT do not perform well on the big and sparse Book Crossing and Slashdot-Zoo Big datasets.

4.2.2 Performance on Negative Samples

In this subsection, we will first explain the results of the evaluation, measured by negative-AUC, GAUC, negative-Precision@20, negative-Hit-Rate@10 and negative classification performance, followed by a discussion of the results.

Performance measured by negative-AUC

When comparing the models' performance on the negative-AUC metric, we see that SGDReg always outperforms all other models, followed by Logistic MF, ELLMF, E-BPRMF and Multinominal Log MF.

While E-BPRMF and ELLMF perform better on the Slashdot-Zoo datasets, Multinominal Log MF and Logistic MF are better on MovieLens and Book Crossing datasets. The adaption of BPRMF, to sample explicitly from negative samples, led to the result that E-BPRMF and P-BPRMF both outmatch BPRMF in terms of negative-AUC performance on all datasets. BPRMF, LLMF and GAUC-OPT are quite off, with up to 50 percentage points below the best performing model.

Performance measured by GAUC

The combination of AUC and negative-AUC in the GAUC metric is headed by SGDReg, followed by Logistic MF, ELLMF and Multinominal Log MF.

While Logistic MF and Multinominal Log MF perform quite well on Book Crossing and MovieLens datasets, P-BPRMF and ELLMF are better on the Slashdot-Zoo datasets. LLMF and BPRMF, which scored well on AUC, do not manage to transfer its superiority to the GAUC metric, since they both score bad on negative-AUC.

Performance measured by negative-Precision@20

When looking at negative-Precision@20, ELLMF performs great on the Slashdot-Zoo datasets, whereas SGDReg outmatches all other models on MovieLens and Book Crossing datasets.

On the dense MovieLens datasets, Multinominal Log MF performs quite good as well. BPRMF, Logistic MF and LLMF are rather bad at ranking negative samples at the top of the negative ranking list. ELLMF always beats Logistic MF and the adapted versions of BPRMF, E-BPRMF and P-BPRMF both always outmatch BPRMF.

Performance measured by negative-Hit-Rate@10

Negative-Hit-Rate@10, similar to negative-Precision@20, is headed by SGDReg and ELLMF.

Again ELLMF always outperforms Logistic MF and the adapted versions of BPRMF, E-BPRMF and P-BPRMF both always outmatch BPRMF. Except on the dense MovieLens datasets, Multinominal Log MF performs relatively bad. BPRMF and LLMF are not able to correctly rank negative samples at the bottom of the ranking list and therefore do not perform well on all datasets.

Performance measured by Classification

The analysis of the models' performance on classification score can be split into analyzing classification of positive and negative items separately.

If we look at how the classification performance of the models consists of negative and positive classification score, we notice that Logistic MF and LLMF are not able to classify correctly, but most of the time predict the negative class for any given sample. We therefore do not include these models in our classification evaluation, since a simple dummy recommender, which would always predict the same negative class, would be quite as good.

For classifying positive samples, BPRMF performs quite good on most datasets. ELLMF also performs quite well on positive classification, but is more sensitive to the share of positive samples within a dataset. It outperforms on Slashdot-Zoo with many positive samples, but is not as good on the Book Crossing dataset with many negative samples.

When looking at the classification score of negative items, ELLMF, P-BPRMF and GAUC-OPT are quite good. For skewed datasets with either many positive or many negative items, ELLMF outperforms all other models, since it catches those differences quite good and manages to balance positive and negative classification.

Conclusion

When evaluating the performance on negative samples by measuring negative-AUC, negative-Precision@20 and negative-Hit-Rate@10, ELLMF outmatches all other models on the sparse Slashdot-Zoo datasets with many positive and few negative samples. For the same evaluation metrics on the dense MovieLens datasets and the sparse Book Crossing dataset with many negative samples, SGDReg outperforms all other models.

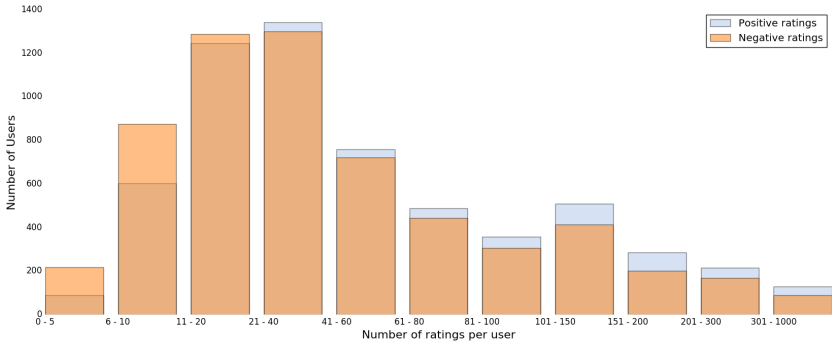
When looking at classification performance, ELLMF is the best choice, as it outperforms all other models on most datasets. Logistic MF and LLMF are not able to correctly classify most of the samples. This might be due to the proposed classification decision boundary which we presented in Subsection 4.1.3.

While BPRMF is great for positive metrics like AUC, it falls short on negative measures. E-BPRMF and P-BPRMF both outmatch BPRMF on all negative metrics, whereas E-BPRMF especially increases classification accuracy. P-BPRMF outmatches E-BPRMF on the sparse Slashdot-Zoo and Book Crossing datasets.

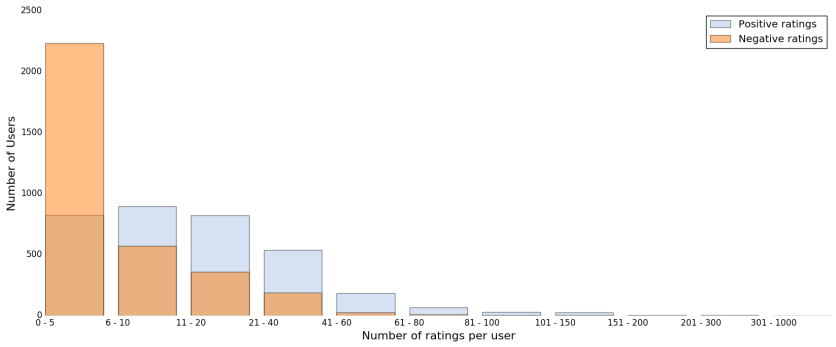
Overall SGDReg does a very good job for all measures on positive samples, especially for AUC. If one also needs a good performance on the negative samples, i.e. ranking negative samples at the bottom of a ranking list or for recommending a Bottom-K list, ELLMF is the best choice as it balances negative and positive samples the best and thus outperforms most of the other models. For classification tasks also ELLMF outperforms all other models on most of the datasets.

4.2.3 Performance per Groups of Ratings per User

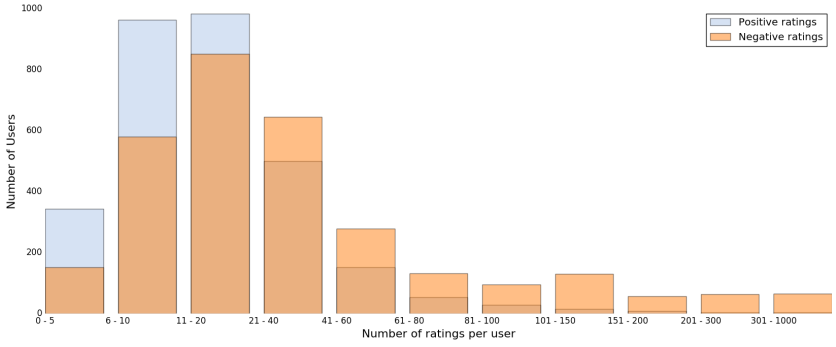
We will now analyze the models' performance on different groups of users, depending on how many samples they have. We assign each user into one of the following groups, according to how many positive or negative samples each user has: 0-10, 11-20, 21-40, 41-60, 61-80, 81-100, 101-150, 151-200, 201-300 and 301-1000 samples per user. We then calculate the Hit-Rate@10 and AUC measures for each group and analyze the results. With this approach, we want to examine the models' sensitivity to less samples per user and analyze the cold-start problem.



(a) MovieLens 1M



(b) Slashdot-Zoo Big



(c) Book Crossing

Figure 4.2: Distribution of Number of Ratings per User

We look at the performance of our methods on the three benchmark datasets MovieLens 1M, Book Crossing and Slashdot-Zoo Big. All of these datasets have a different distribution of number of ratings per user, as shown in Figure 4.2. The complete evaluation grouped by number of ratings per user of all models on all three datasets can be found in Appendix A.3.

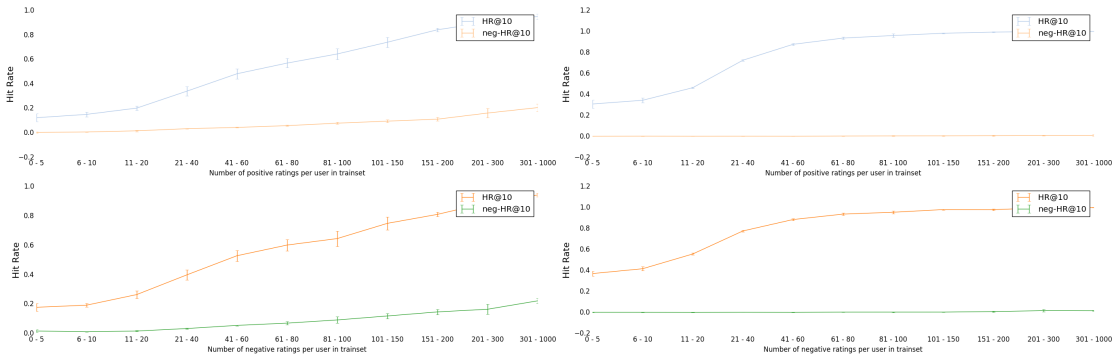
Sparse Linear Methods

SGDReg has nearly constant and high positive and negative AUC throughout all groups on nearly all datasets. Even for low numbers of samples per user (0-20 samples per user), SGDReg outperforms most of the other models, without any trade-off between positive and negative AUC.

Bayesian Personalized Ranking Models

Altough lower than SGDReg, BPRMF has nearly constant AUC over all groups of users, even for low numbers of samples. If there are enough positive samples, BPRMF has a really high Hit-Rate@10 score. While BPRMF excels on AUC and Hit-Rate@10 for low numbers of samples per user, E-BPRMF and P-BPRMF both outmatch BPRMF with higher neg-AUC and neg-Hit-Rate@10 scores which are nearly constant, even for low numbers of samples per user.

Logistic Models



(a) Logistic MF (b) Logistic Likelihood MF

Figure 4.3: Hit-Rate on MovieLens 1M

As shown in Figure 4.3, Logistic Likelihood MF nearly doubled the Hit-Rate@10 compared to Logistic MF for less than 10 samples on the MovieLens 1M dataset. But unlike Logistic MF, LLMF falls short on neg-Hit-Rate@10 and all other negative metrics.

Explicit Logistic Likelihood MF performs relatively bad on the dense MovieLens 1M dataset but excels on the sparse Slashdot-Zoo Big dataset, where it outmatches Logistic MF on all metrics. ELLMF has especially high negative and positive classification scores

for low numbers of samples per user. The negative classification score of ELLMF is lower, if there are more positive samples. If the number of negative samples increases, the positive classification score gets lower, as shown in Figure 4.4.

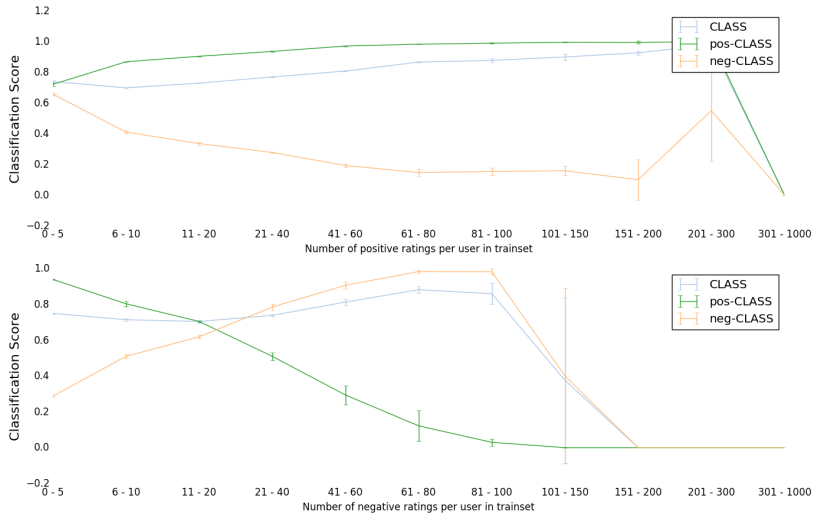


Figure 4.4: Classification performance of ELLMF on Slashdot-Zoo Big

Multinominal Logistic MF scores relatively bad on AUC, Hit-Rate and Classification for low numbers of samples to train on. For more than 150 positive samples MLMF scores rather good with an AUC score of around 95% on the Book Crossing dataset. Whereas an increasing number of positive samples lead to an increase in both positive and negative AUC, mainly negative AUC benefits from an increasing number of negative samples as shown in Figure 4.5.

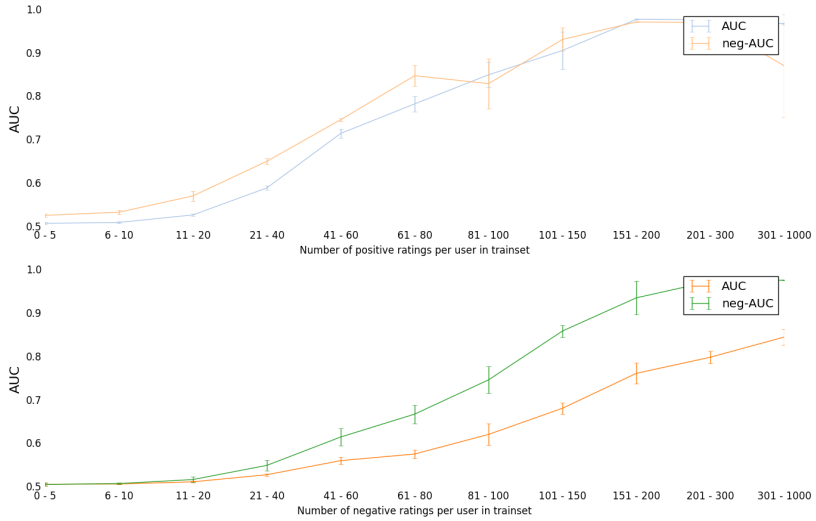


Figure 4.5: AUC performance of Multinominal Logistic MF on Book Crossing

Conclusion

In this section, we evaulated the models' performance on different number of samples per users. With focus on low number of samples per user (0-20 samples per user) we analyzed the models' capability of handling new users, the so called cold-start problem.

For low number of samples per user SGDReg outmatches all other models on AUC, negative AUC and negative Hit-Rate@10 on all datasets. BPRMF has the highest Hit-Rate@10 score across all evaluated methods. On the sparse Slashdot-Zoo Big dataset Log MF performs good for positive and negative Hit-Rate@10 and ELLMF outmatches other methods on positive and negative classification accuracy. On the dense MovieLens 1M dataset E-BPRMF and GAUC-OPT excel in positive and negative classification accuracy for low numbers of samples per user respectively. Only with a high number of samples per user Multinominal Logistic MF performs good on AUC and negative-AUC but is still outmatched by SGDReg.

Therefore depending on the task, different models are superior. For AUC SGDReg, for Hit-Rate@10 BPRMF and for classification tasks ELLMF are the best performing methods respectively, when evaluated on low number of samples per user.

5

Conclusions

Current research mainly focused on explicit and implicit positive feedback. Those One-Class Collaborative Filtering (OCCF) problems examine positive and unknown feedback and aim to correctly classify positive samples, whereas they ignore the difference between negative and unknown samples.

Moreoften not only positive but also negative relationships are of interest. For example: distinguishing friends from foes, spam from important mails and beneficial actions from harmful ones. The cost of a negative recommendation may not be significant in domains like movies or TV shows. But in case of jobs, places to live or friends to connect to, the cost of a negative recommendation could be high. In such domains, users are likely to be put off by recommender systems that cannot distinguish potentially negative items and occasionally put them at the top of the list. Therefore, producing personalised rankings with positive feedback at the top and negative feedback at the bottom is an increasingly important task (Song and Meyer, 2015).

We study positive and negative class prediction in Two-Class Collaborative Filtering (TCCF) problems, where we examine the efficiency of current state-of-the-art recommenders and propose new methods to better address these problems. Based on Bayesian Personalized Ranking Matrix Factorization (BPRMF) from Rendle et al. (2012) and Logistic MF from Johnson (2014) we introduce Explicit-BPRMF (E-BPRMF) and Proportional-BPRMF (P-BPRMF), as well as Logistic Likelihood MF (LLMF), Explicit Logisitc Likelihood MF (ELLMF) and Multinominal Logistic MF (MLMF) to address TCCF problems.

We evaluate our models on the four benchmark datasets: MovieLens 100K, MovieLens 1M, Slashdot-Zoo and Book Crossing. We compare the results with an evaluation of the baseline recommenders BPRMF, Logistic MF, SGDReg (Levy and Jack, 2013) and GAUC-OPT (Song and Meyer, 2015).

With our methods we outperform Logistic MF, BPRMF and GAUC-OPT on either AUC, Hit-Rate@10, Precision@20 and their respective negative evaluation metrics. However, all our evaluation results are surpassed by SGDReg, which excels in most evaluation metrics on the examined datasets.

In the following section we will discuss the evaluation results and finish with an overview of possible future work.

5.1 Discussion

As shown in Chapter 5.2, SGDReg a variant of Sparse Linear Methods (SLIM) excels in most evaluations. The advantage of SGDReg is bigger for more sparse and skewed datasets like Slashdot-Zoo and Book Crossing. It has an advantage of around 15 to 20 percentage points in AUC to the next best model.

With the introduction of Logistic Likelihood MF (LLMF) and Explicit Logistic Likelihood MF (ELLMF), we outperform Logistic MF on Percision@20 and Hit-Rate@10. LLMF even outperforms Logistic MF on AUC and ELLMF outperforms Logistic MF on most of the negative evaluation metrics, by ranking negative samples at the bottom of a personalized ranking list.

As described by Rendle et al. (2012), BPRMF is really good for personalized ranking lists. Whereas BPRMF excels on positive ranking lists, it cannot properly rank negative items at the bottom of a personalized ranking. With E-BPRMF and P-BPRMF we overcome this limitation and increase the performance on negative ranking lists. However, the increased performance of negative ranking lists leads to a decrease in the performance of positive ranking lists. We therefore experience a trade-off between positive and negative ranking lists on the Bayesian Personalized Ranking models.

By splitting the performance evaluation into groups of different number of samples per user, we analyze the models' behaviour on sparsity and cold-start. Especially for few samples per user (0-20) SGDReg outmatches all other models by providing very high AUC throughout all groups of users. Multinominal Logistic MF (MLMF) only performs good as soon as enough feedback for training is available.

Finally, we would like to share some insights and our observations which we accumulated during the work on this thesis. On the used methods we can note that user and item bias improve the models' performance as suggested by Koren et al. (2009), James et al. (2014) and Tintarev and Masthoff (2011). Also using AdaGrad to compute the learning rate in gradient descent algorithms, improves convergence of the algorithms as suggested by Duchi et al. (2011). Subtracting users' mean-rating by applying mean-removal seems to increase all Logistic MF models' performances, whereas it doesn't have a big impact on BPRMF models. Applying gridsearch, to search for the best parameter settings, increased the models' performance significantly, even for the baseline recommenders where the standard parameters didn't always yield the best performance. Although the SGDReg model is rather simple and relaxes SLIM's non-negativity constraint, it seems to be quite good in representing the ranking list and outperforms most of the other models.

5.2 Future Work

After our work on Two-Class Collaborative Filtering (TCCF) problems, we see opportunities for future research efforts in mainly three different directions: Adapting the optimization criterion, analyzing the classification decision boundary and changing the loss function used for gradient descent.

We have several ideas of adapting the optimization criterion, to better represent the desired ranking list, which lists positive samples at the top and negative samples at the bottom, to solve TCCF problems.

We focused on logistic models which predict the probability of an item belonging to a given class, rather than modelling the item's value directly (James et al., 2014). It would be interesting to see if using a logisitic model to predict the item similarity for SLIM and develop a Logistic-SLIM could improve the results.

As we've seen in the evaluation in Chapter , SGDReg performs great on numerous tasks on all datasets. Since SGDReg is based on SLIM but relaxes the non-negativity constraint and as suggested by Levy and Jack (2013), it would be interesting to investigate the theoretical basis for the success of the simple regression model SGDReg.

Another approach would be to adapt Logistic MF similar to Kabbur and Karypis (2014), to model positive and negative user interests in two separate user latent factors and combine it with one single item latent factor. This could be done analog to the Multinominal Logistic Matrix Factorization (MLMF) implementation.

The advantage of Bayesian Personalized Ranking models in representing the ranking list could be applied to other models as well. Using the difference between two user-item-rating triples ($r_{ui} - r_{uj}$), instead of focusing on one single user-item-rating triple (r_{ui}), could improve other models as well.

In this work, we classify samples according to a static decision boundary of 0.5. Depending on the model and the dataset, the decision boundary could be different. It would be interesting to analyze how a dynamic decision boundary can be developed, to improve classification performance.

Instead of using the direct optimization criterion loss within the gradient descent algorithms, it might be beneficial to use other loss functions like hinge-loss or softmax, to improve convergence and prevent over-fitting.

References

Bedi, P., Kaur, H., and Marwaha, S. (2007). Trust Based Recommender System for the Semantic Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2677–2682, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bennett, J., Lanning, S., and Netflix, N. (2007). The Netflix Prize. In *In KDD Cup and Workshop in conjunction with KDD*.

Brzozowski, M. J., Hogg, T., and Szabo, G. (2008). Friends and foes. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 817, New York, New York, USA. ACM Press.

Craswell, N. (2009). *Mean Reciprocal Rank*, page 1703. Springer US, Boston, MA.

Desrosiers, C. and Karypis, G. (2011). *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, chapter A Comprehe, pages 107–144. Springer US, Boston, MA.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36.

Harper, F. M. and Konstan, J. A. (2015). The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19.

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. *Proc of Uncertainty in Artificial Intelligence UAI'99*, 50(4):21.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. IEEE.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Johnson, C. C. (2014). Logistic Matrix Factorization for Implicit Feedback Data. *NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*, pages 1–9.

Kabbur, S. and Karypis, G. (2014). NLMF: NonLinear Matrix Factorization Methods for Top-N Recommender Systems. *2014 IEEE International Conference on Data Mining Workshop*, (ii):167–174.

Koren, Y. and Bell, R. (2011). Advances in Collaborative Filtering. In *Recommender Systems Handbook*, pages 145–186. Springer US, Boston, MA.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37.

Kunegis, J., Lommatzsch, A., and Bauckhage, C. (2009). The slashdot zoo. In *Proceedings of the 18th international conference on World wide web - WWW '09*, page 741, New York, New York, USA. ACM Press.

Kwak, C. and Clayton-Matthews, A. (2002). Multinomial Logistic Regression. *Nursing Research*, 51(6):404–410.

Levy, M. and Jack, K. (2013). Efficient Top-N Recommendation by Linear Regression. *Large Scale Recommender Systems Workshop in RecSys'13*, i(October).

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: Social Recommendation Using Probabilistic Matrix Factorization. In *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, volume 08pages, page 931, New York, New York, USA. ACM Press.

Marlin, B. and Zemel, R. S. (2004). The Multiple Multiplicative Factor Model for Collaborative Filtering. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 73—-, New York, NY, USA. ACM.

Mnih, A. and Salakhutdinov, R. (2007). Probabilistic matrix factorization. *Advances in neural information processing systems*, pages 1257–1264.

Ning, X. and Karypis, G. (2011). SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506. IEEE.

Oard, D. W. and Kim, J. (1998). Implicit Feedback for Recommender Systems. *Proceedings of the AAAI workshop on recommender systems*, pages 81–83.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-Class Collaborative Filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR*, abs/1205.2.

Sinha, R. and Swearingen, K. (2001). Comparing Recommendations Made by Online Systems and Friends. *DELOS Workshop on Personalisation and Recommender Systems in Digital Libraries*, pages http://www.ercim.org/publication/ws–proceedings/De.

Song, D. and Meyer, D. A. (2015). Recommending Positive Links in Signed Social Networks by Optimizing a Generalized AUC. *AAAI 2015: Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence*, pages 290–296.

Tintarev, N. and Masthoff, J. (2011). *Recommender Systems Handbook*, volume 54.

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - ACL-IJCNLP '09*, volume 1, page 477, Morristown, NJ, USA. Association for Computational Linguistics.

Wang, J., de Vries, A. P., and Reinders, M. J. T. (2006). Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 501–508, New York, NY, USA. ACM.

Welling, M., Rosen-zvi, M., and Hinton, G. E. (2005). Exponential Family Harmoniums with an Application to Information Retrieval. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 1481–1488. MIT Press.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving Recommendation Lists Through Topic Diversification. *Proceedings of the 14th International Conference on World Wide Web*, pages 22–32.

A

Appendix

A.1 Gridsearch Parameters

All experiments were performed on a Slurm[1] cluster with 16 nodes. Each machine node has 128 GB of RAM and two Intel R Xeon R E5-2680V2 processors (25 MB Cache, 2.80 GHz base frequency) with 10 cores per processor (40 threads per machine).

All models were evaluated with 30 iterations.

Bayesian Personalized Ranking (BPRMF), E-BPRMF, P-BPRMF Regularizer: [0.001, 0.0025, 0.01]. Learn-Rate: [0.01, 0.05, 0.1]. Latent Factors: [10, 30, 50].

GAUC-OPT Regularizer: [0.01, 0.1, 0.5]. Learn-Rate: [0.01, 0.05, 0.1]. Latent Factors: [10, 30, 50].

Logistic Matrix Factorization (Logistic MF), LLMF, ELLMF Regularizer: [0.01, 0.6, 0.1]. Gamma: [0.1, 1.0, 2.0]. Latent Factors: [10, 30, 50].

Multinominal Logistic Matrix Factorization (MLMF) Regularizer: [0.01, 0.06, 0.1]. Once with L1 regularizer whereas reg_L1 = regularizer, reg_L2 = L1/10.0 and once without L1 regularizer: reg_L1=0.0, reg_L2=regularizer. Gamma: [1.0, 2.0]. Latent Factors: [10, 30, 50].

SGDReg Regularizer: [0.001, 0.01, 0.1], whereas reg_L1 = regularizer, reg_L2 = reg_L1/10.0. Fit intercept: False.

[1] Slurm Workload Manager - *http://slurm.schedmd.com*

A.2 Optimal Parameter Sets of Recommenders

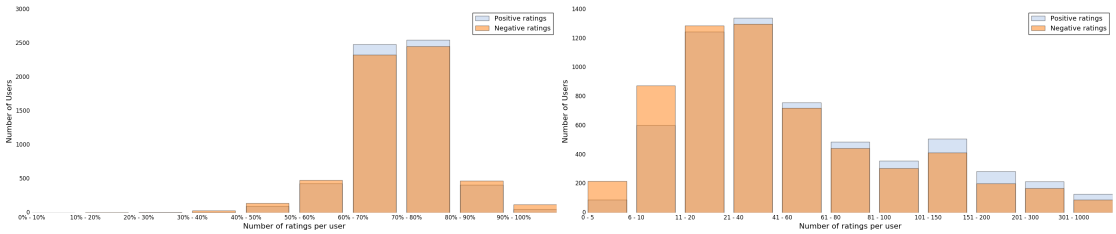
Table A.1: Optimal Parameter Sets of Recommenders

Model	MovieLens 1M	Slashdot-Zoo Big	Book Crossing
BPRMF	$D = 50, \lambda = 0.01, L = 0.05$	$D = 10, \lambda = 0.01, L = 0.01$	$D = 50, \lambda = 0.01, L = 0.01$
E-BPRMF	$D = 50, \lambda = 0.01, L = 0.01$	$D = 50, \lambda = 0.001, L = 0.01$	$D = 10, \lambda = 0.001, L = 0.01$
P-BPRMF	$D = 50, \lambda = 0.001, L = 0.05$	$D = 50, \lambda = 0.01, L = 0.05$	$D = 10, \lambda = 0.001, L = 0.01$
Logistic MF	$D = 50, \gamma = 2.0, \lambda = 0.1$	$D = 10, \gamma = 1.0, \lambda = 0.1$	$D = 50, \gamma = 2.0, \lambda = 0.6$
LLMF	$D = 10, \gamma = 1.0, \lambda = 0.1$	$D = 10, \gamma = 1.0, \lambda = 0.1$	$D = 10, \gamma = 1.0, \lambda = 0.6$
ELLMF	$D = 10, \gamma = 1.0, \lambda = 0.6$	$D = 50, \gamma = 2.0, \lambda = 0.6$	$D = 30, \gamma = 1.0, \lambda = 0.6$
MLMF	$D = 50, \gamma = 2.0, \lambda = 0.1$	$D = 50, \gamma = 2.0, \lambda = 0.1$	$D = 50, \gamma = 2.0, \lambda = 0.006$ with L1
GAUC-OPT	$D = 50, \lambda = 0.1, L = 0.01$	$D = 50, \lambda = 0.5, L = 0.05$	$D = 10, \lambda = 0.01, L = 0.01$
SGDReg	$\lambda = 0.01$	$\lambda = 0.001$	$\lambda = 0.001$

Optimal model parameter set for best AUC performance on each model. Where D denotes the dimensionality of latent vectors, λ the regularizer parameter and L the learning parameter.

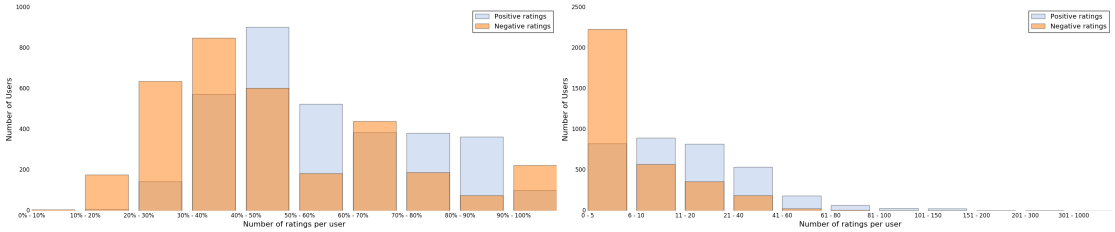
A.3 Evaluation Grouped By Number of Ratings per User

A.3.1 Distribution of Number of Ratings per User



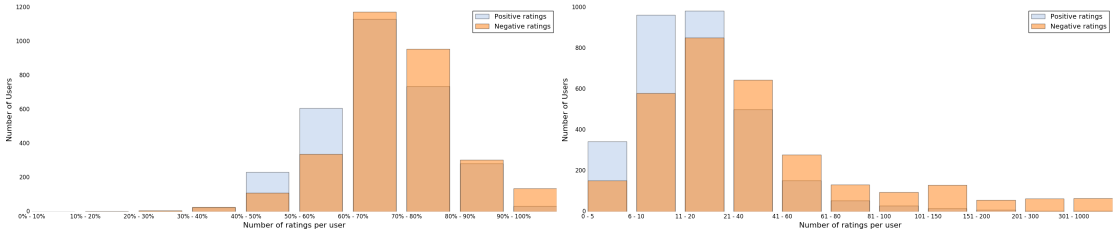
(a) Relative Number of Ratings per User (b) Number of Ratings per User

Figure A.1: MovieLens 1M - Distribution of Number of Ratings per User



(a) Relative Number of Ratings per User (b) Number of Ratings per User

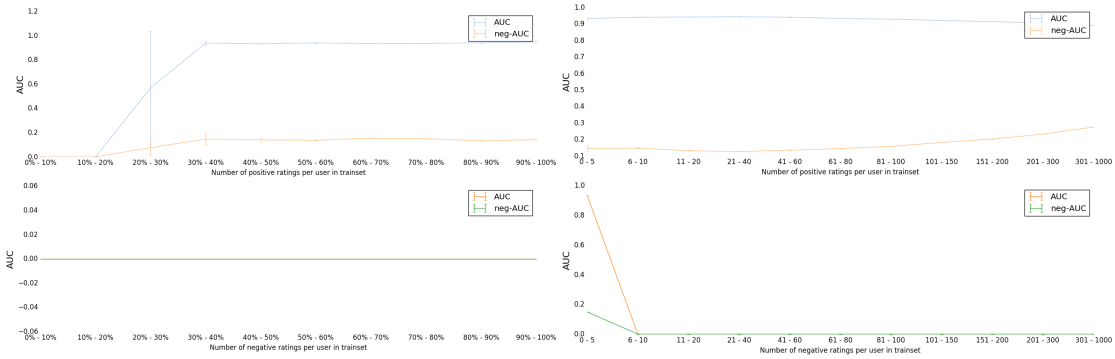
Figure A.2: Slashdot-Zoo Big - Distribution of Number of Ratings per User



(a) Relative Number of Ratings per User (b) Number of Ratings per User

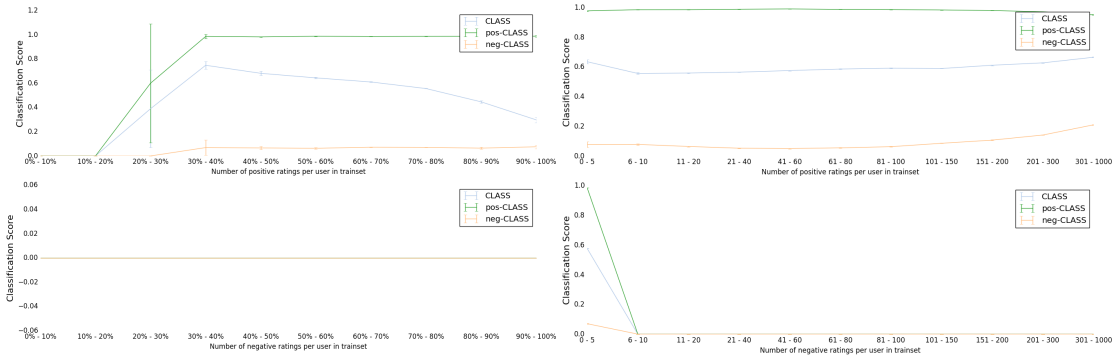
Figure A.3: Book Crossing - Distribution of Number of Ratings per User

A.3.2 MovieLens 1M



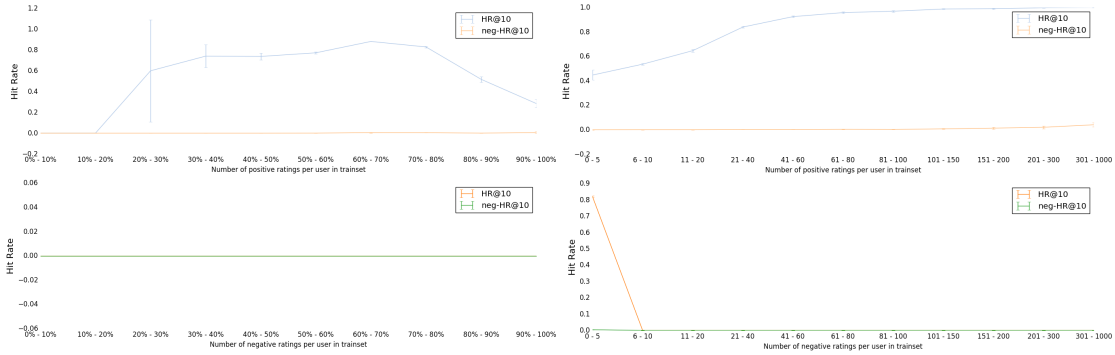
(a) Relative AUC

(b) AUC



(c) Relative Classification

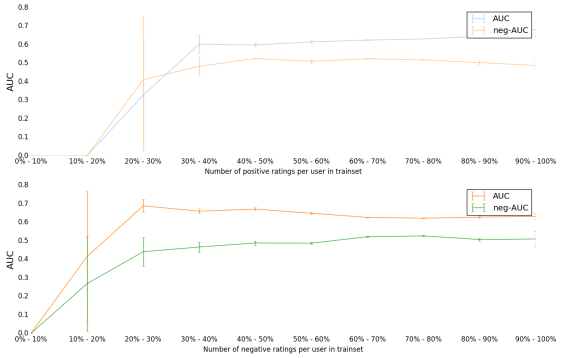
(d) Classification



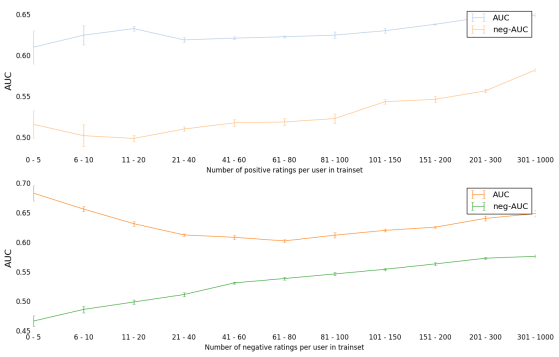
(e) Relative Hit-Rate

(f) Hit-Rate

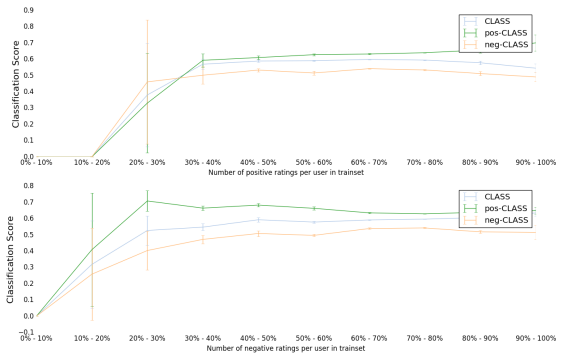
Figure A.4: MovieLens 1M - BPRMF Original



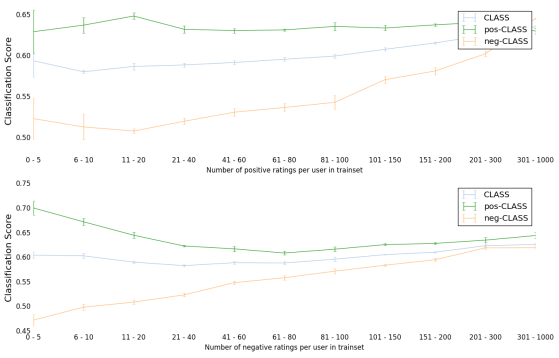
(a) Relative AUC



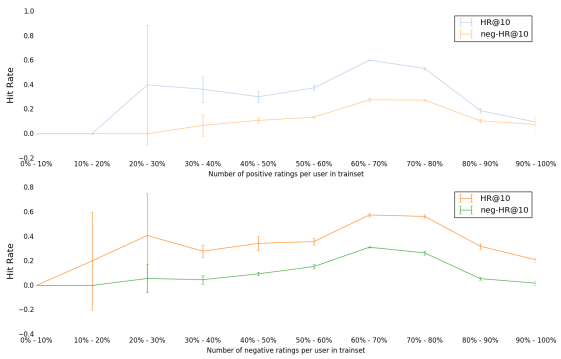
(b) AUC



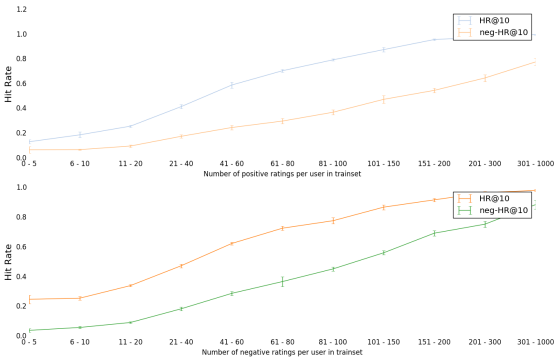
(c) Relative Classification



(d) Classification

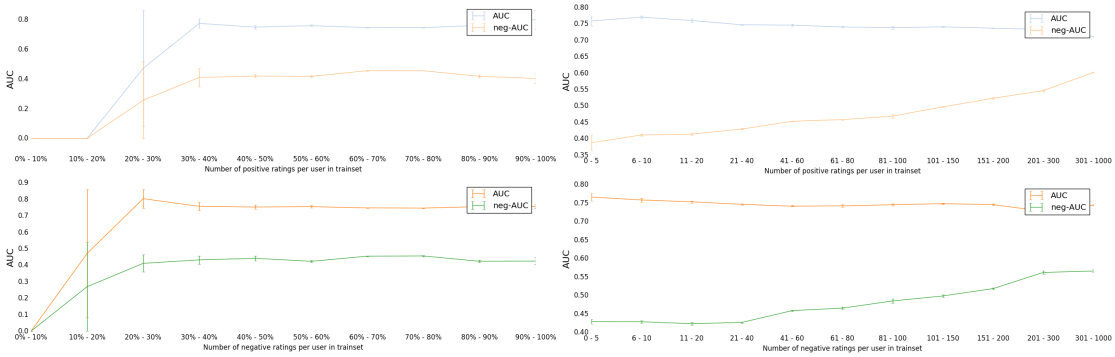


(e) Relative Hit-Rate



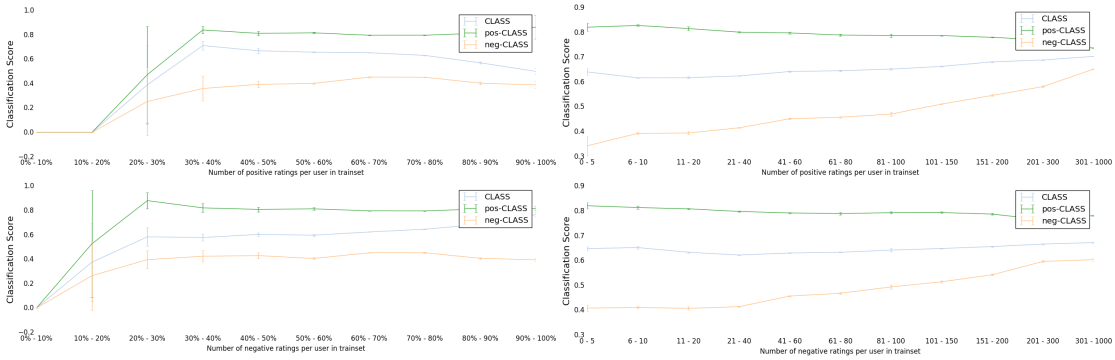
(f) Hit Rate

Figure A.5: MovieLens 1M - P-BPRMF



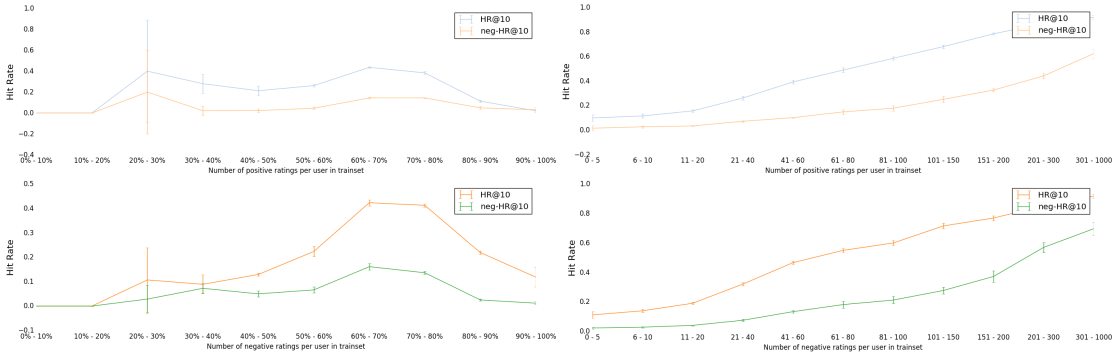
(a) Relative AUC

(b) AUC



(c) Relative Classification

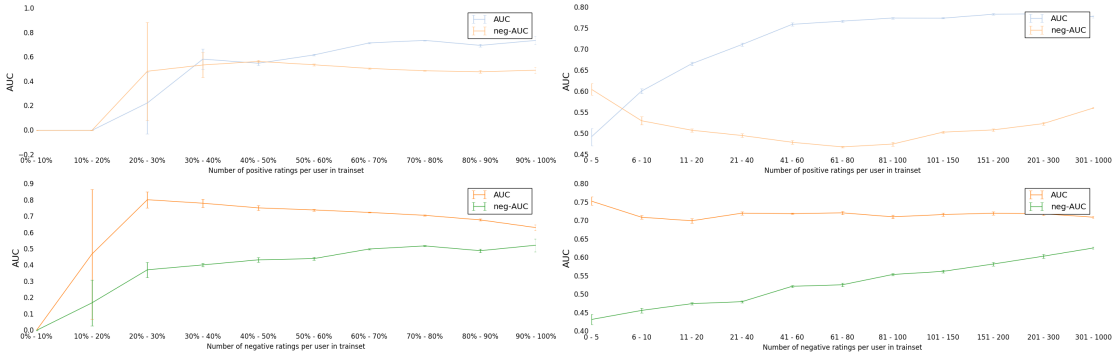
(d) Classification



(e) Relative Hit-Rate

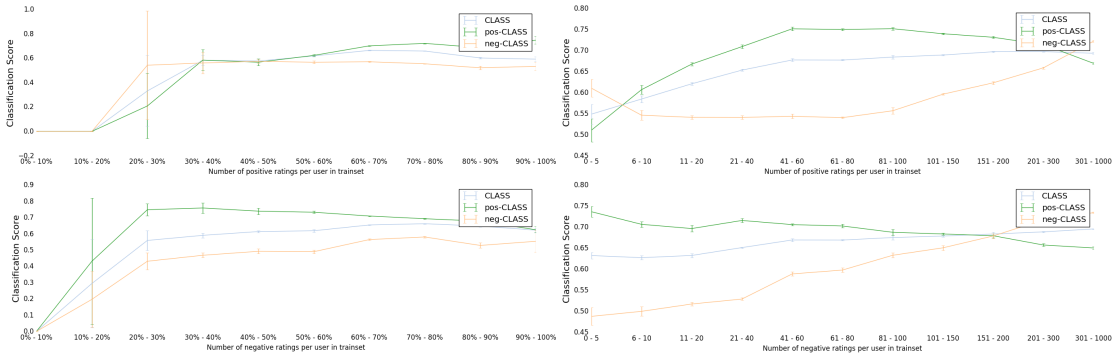
(f) Hit Rate

Figure A.6: MovieLens 1M - E-BPRMF



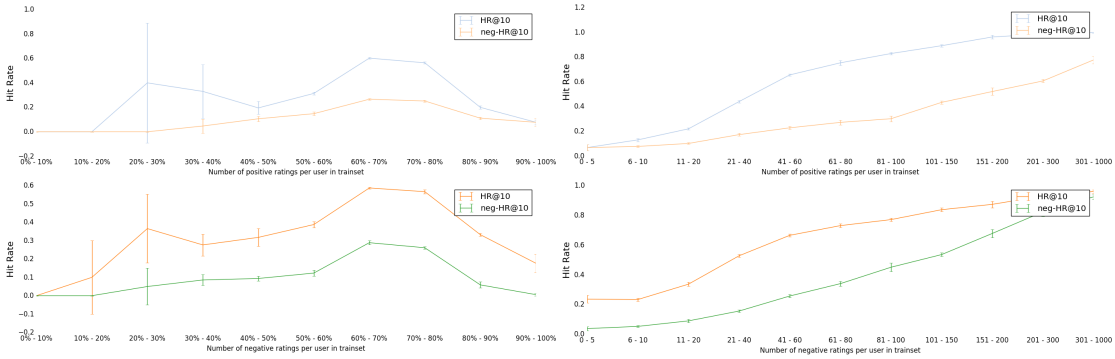
(a) Relative AUC

(b) AUC



(c) Relative Classification

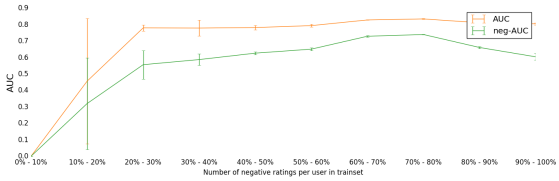
(d) Classification



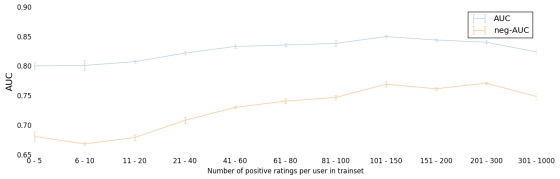
(e) Relative Hit-Rate

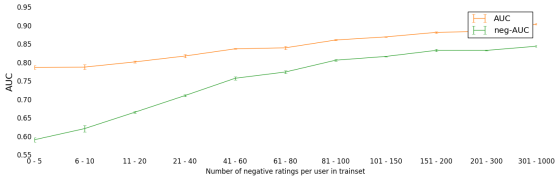
(f) Hit Rate

Figure A.7: MovieLens 1M - GAUC-OPT

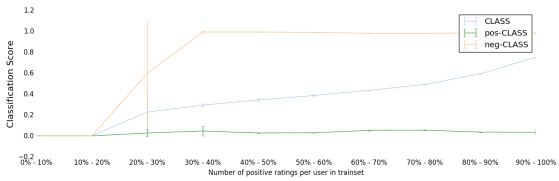


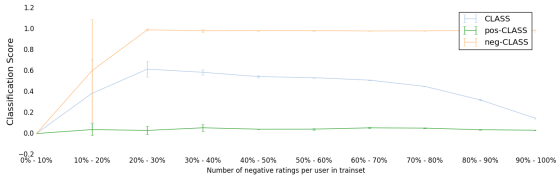
(a) Relative AUC



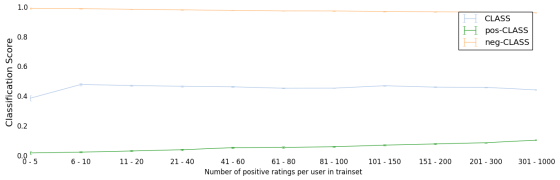


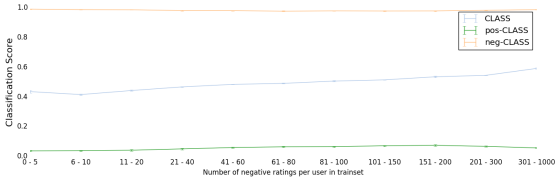
(b) AUC



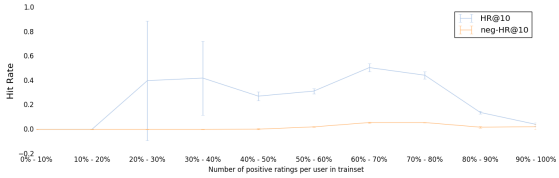


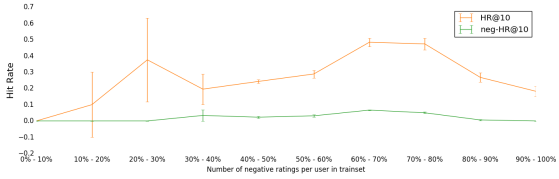
(c) Relative Classification



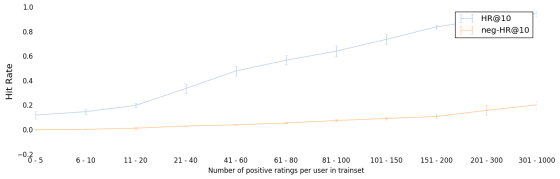


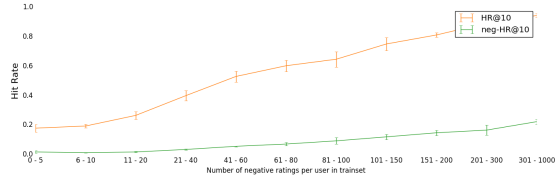
(d) Classification





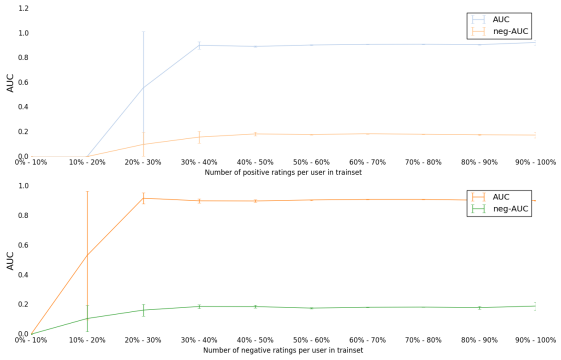
(e) Relative Hit-Rate



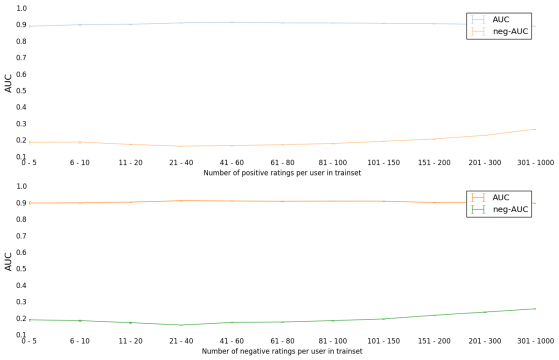


(f) Hit Rate

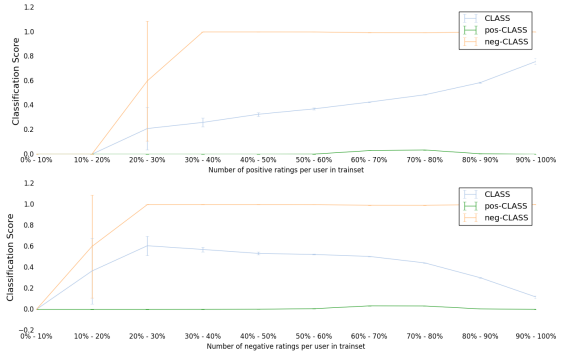
Figure A.8: MovieLens 1M - Logistic MF



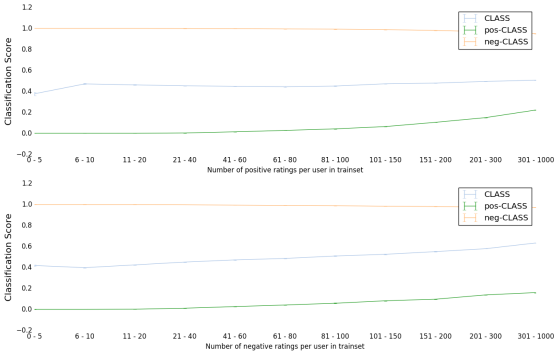
(a) Relative AUC



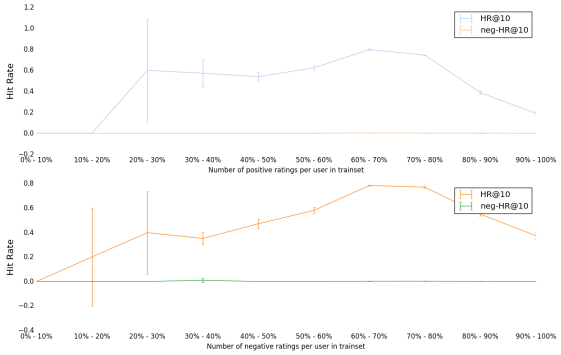
(b) AUC



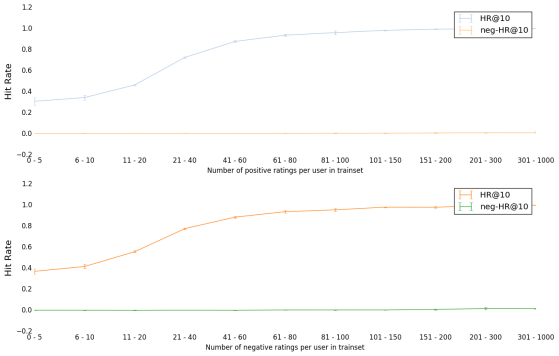
(c) Relative Classification



(d) Classification

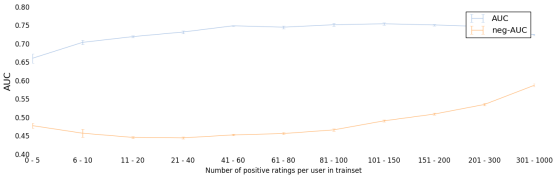


(e) Relative Hit-Rate

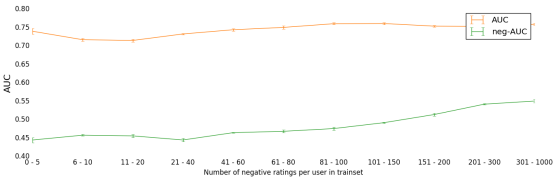


(f) Hit Rate

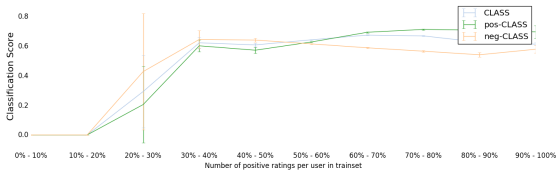
Figure A.9: MovieLens 1M - Logistic Likelihood MF

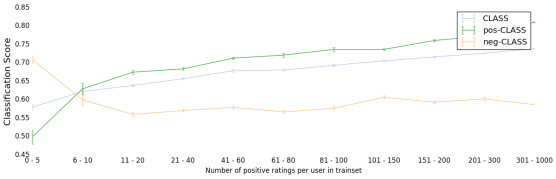


(a) Relative AUC

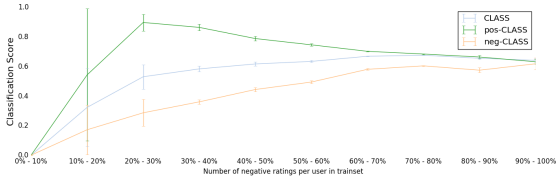


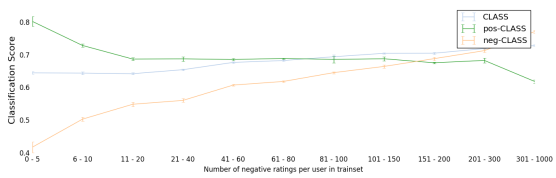
(b) AUC



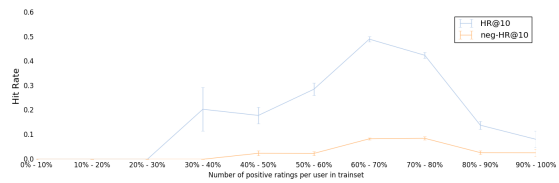


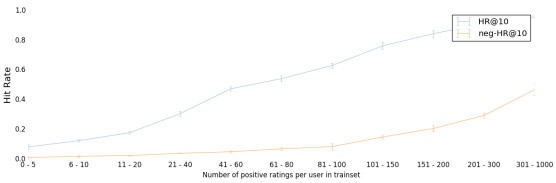
(c) Relative Classification



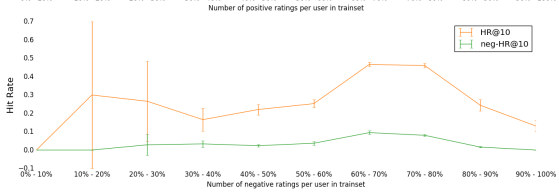


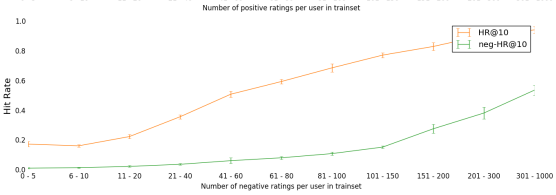
(d) Classification





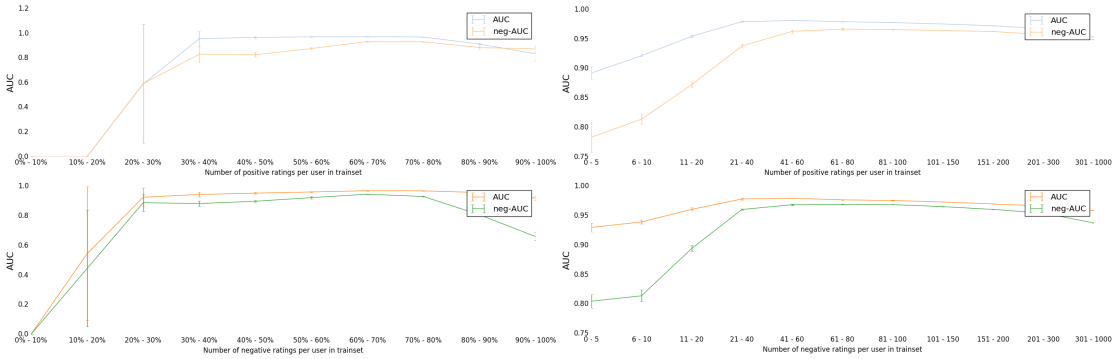
(e) Relative Hit-Rate





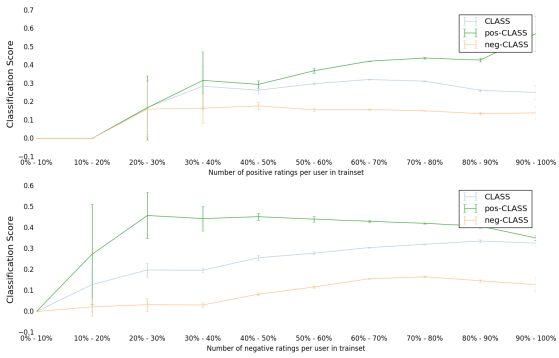
(f) Hit Rate

Figure A.10: MovieLens 1M - Explicit Logistic Likelihood MF

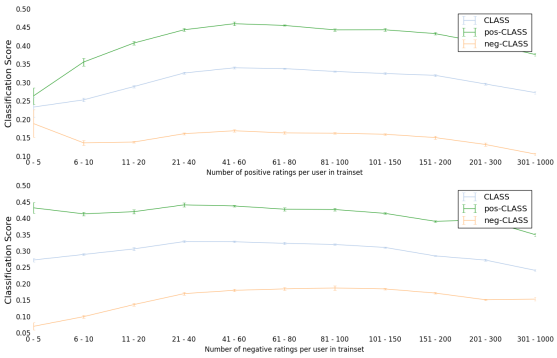


(a) Relative AUC

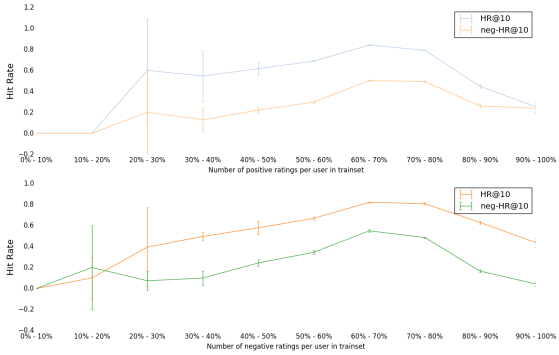
(b) AUC



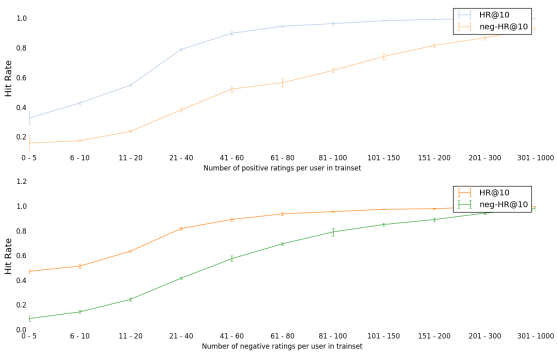
(c) Relative Classification



(d) Classification

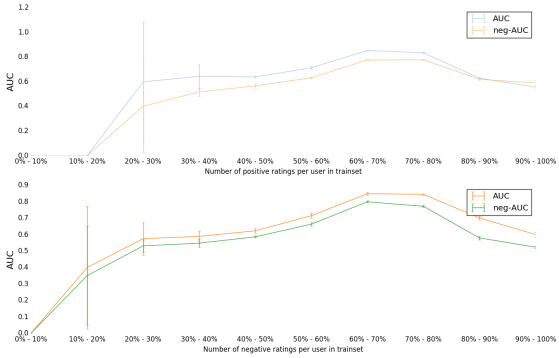


(e) Relative Hit-Rate

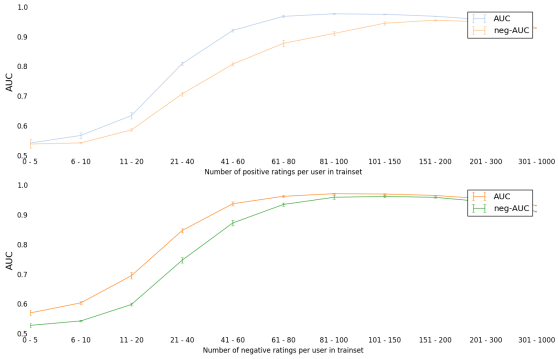


(f) Hit Rate

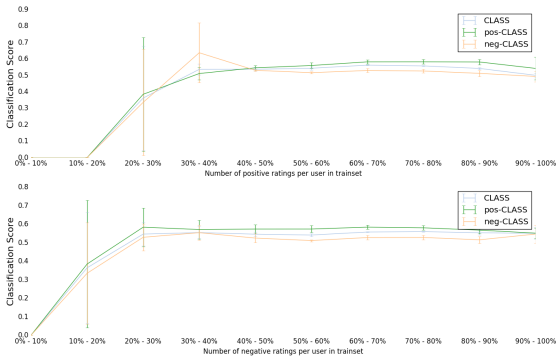
Figure A.11: MovieLens 1M - SGDReg



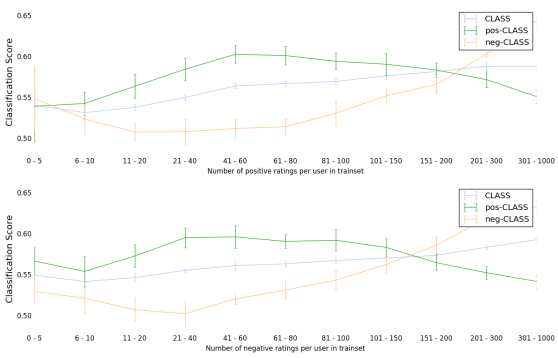
(a) Relative AUC



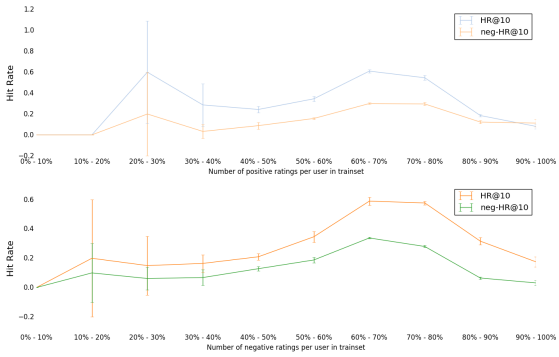
(b) AUC



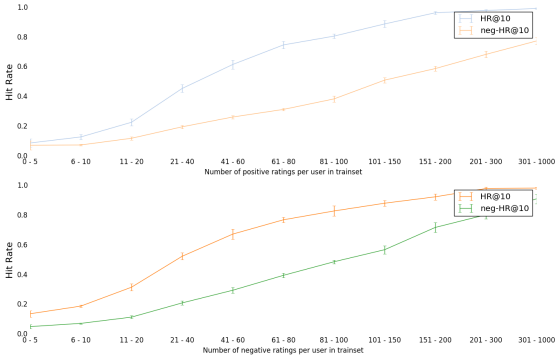
(c) Relative Classification



(d) Classification



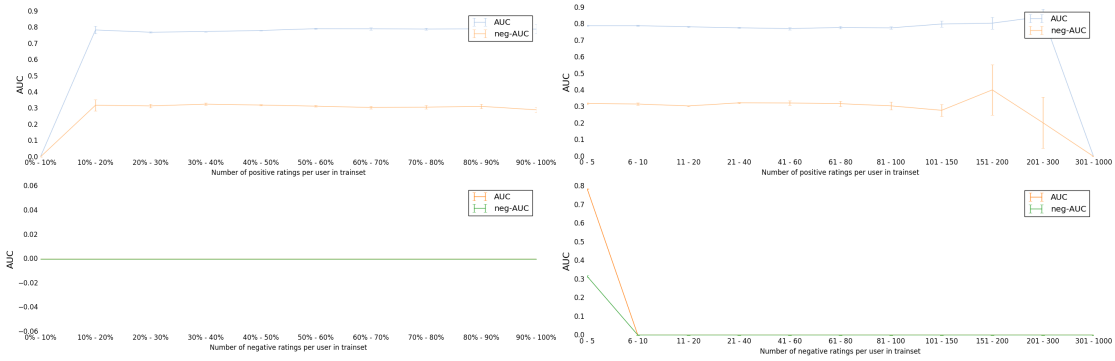
(e) Relative Hit-Rate



(f) Hit Rate

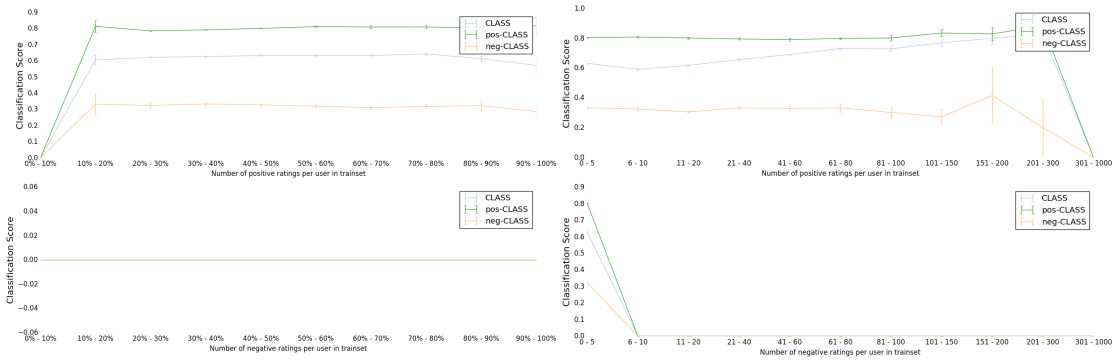
Figure A.12: MovieLens 1M - Multinominal Logistic MF

A.3.3 Slashdot-Zoo Big



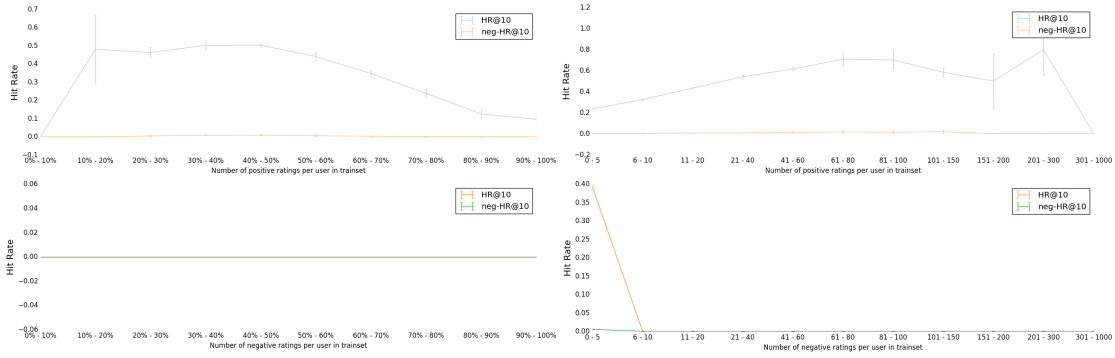
(a) Relative AUC

(b) AUC



(c) Relative Classification

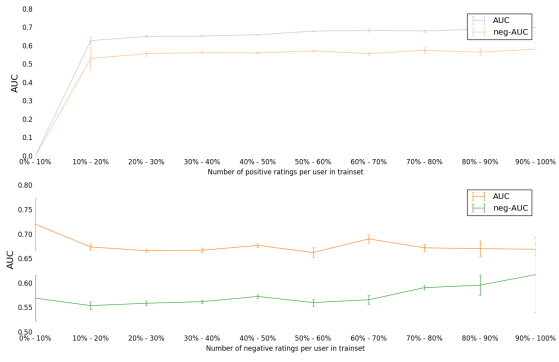
(d) Classification



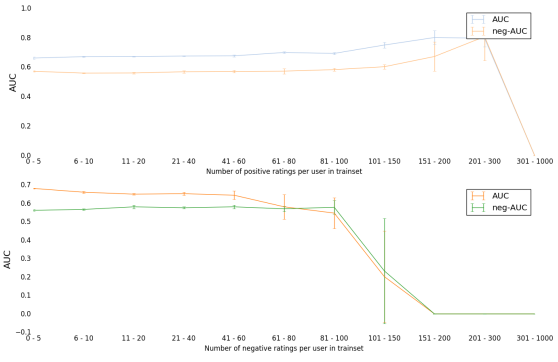
(e) Relative Hit-Rate

(f) Hit Rate

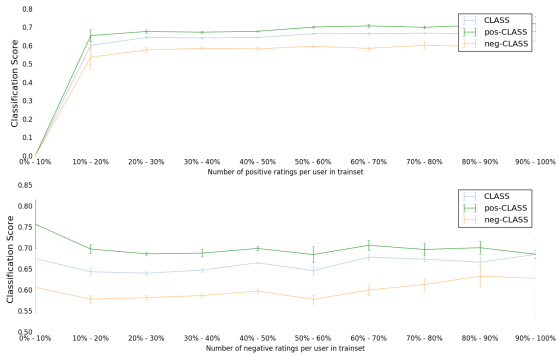
Figure A.13: Slashdot-Zoo Big - BPRMF Original



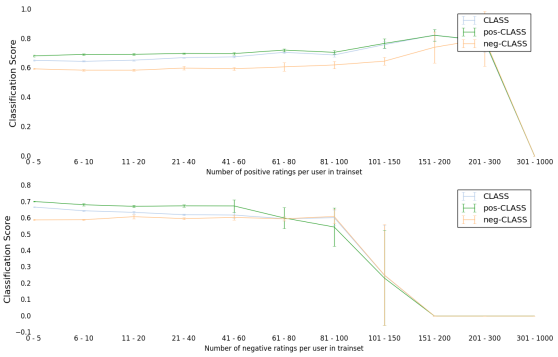
(a) Relative AUC



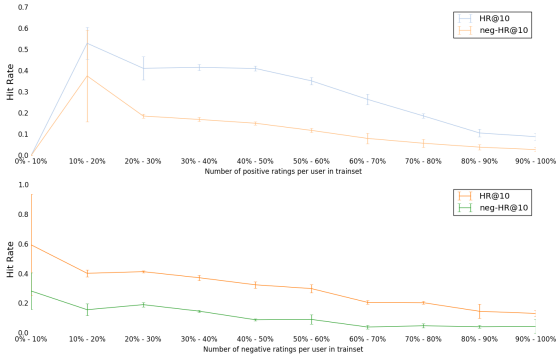
(b) AUC



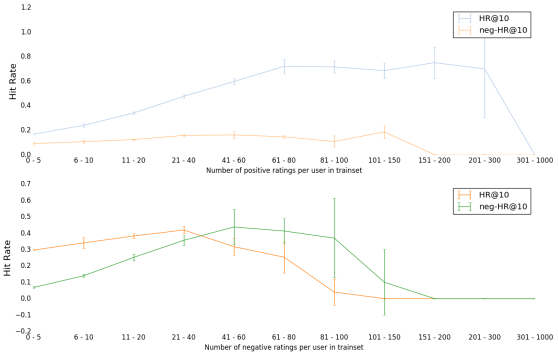
(c) Relative Classification



(d) Classification

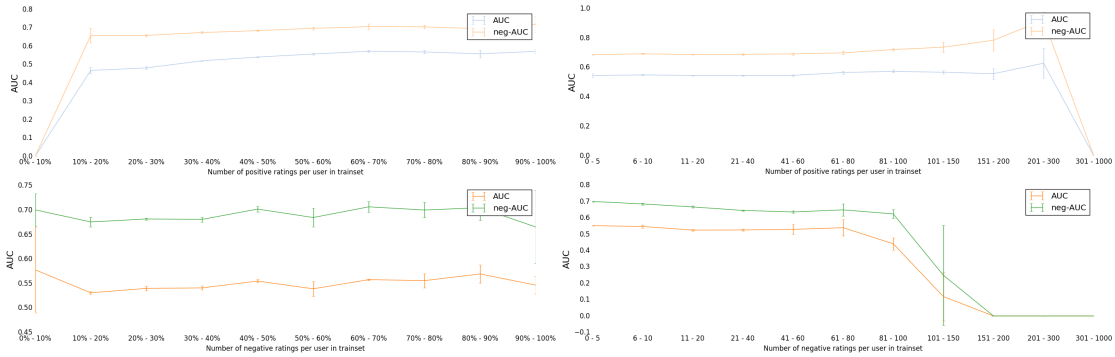


(e) Relative Hit-Rate



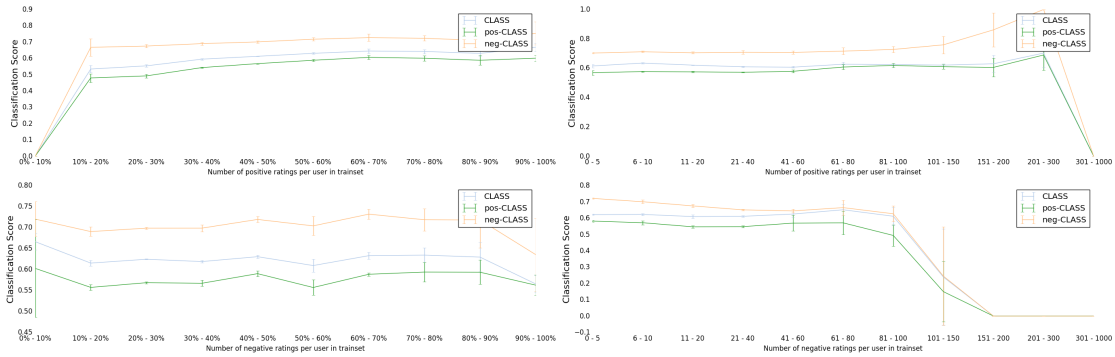
(f) Hit Rate

Figure A.14: Slashdot-Zoo Big - P-BPRMF



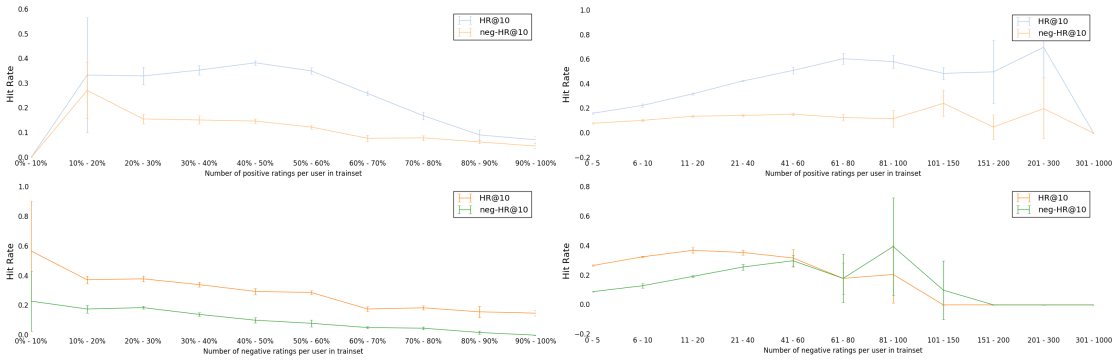
(a) Relative AUC

(b) AUC



(c) Relative Classification

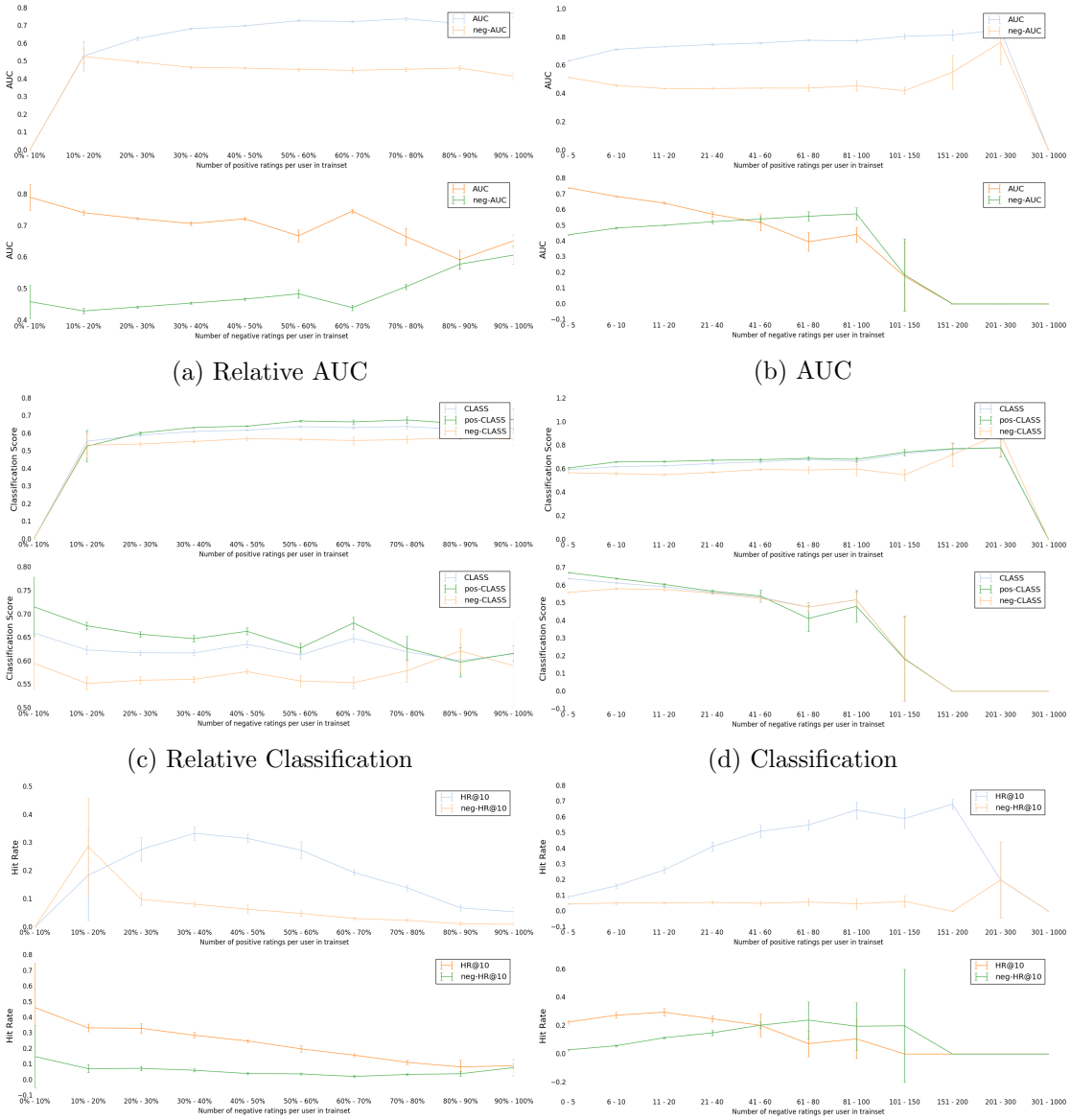
(d) Classification



(e) Relative Hit-Rate

(f) Hit Rate

Figure A.15: Slashdot-Zoo Big - E-BPRMF



(a) Relative AUC

(b) AUC

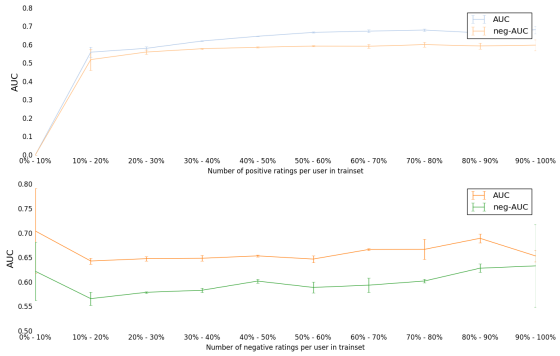
(c) Relative Classification

(d) Classification

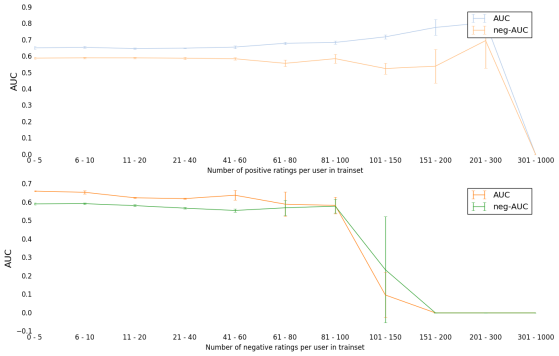
(e) Relative Hit-Rate

(f) Hit Rate

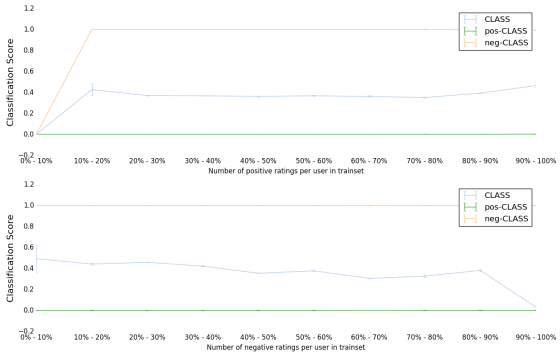
Figure A.16: Slashdot-Zoo Big - GAUC-OPT



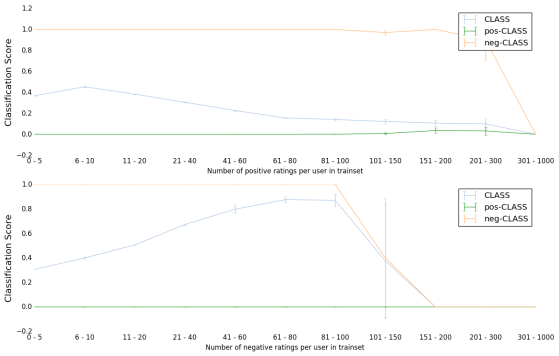
(a) Relative AUC



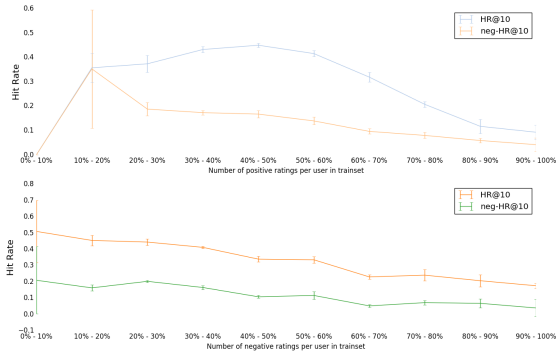
(b) AUC



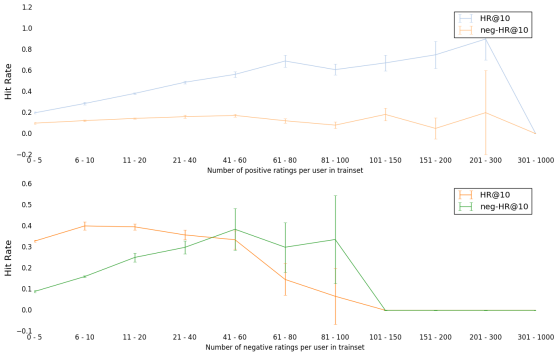
(c) Relative Classification



(d) Classification

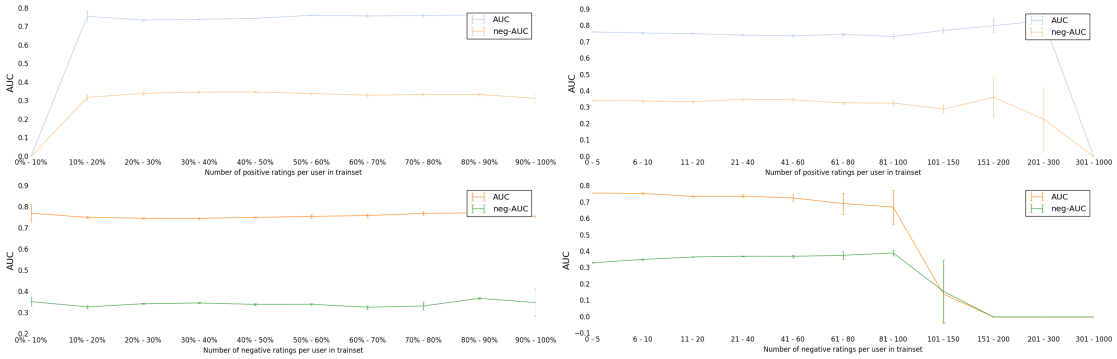


(e) Relative Hit-Rate



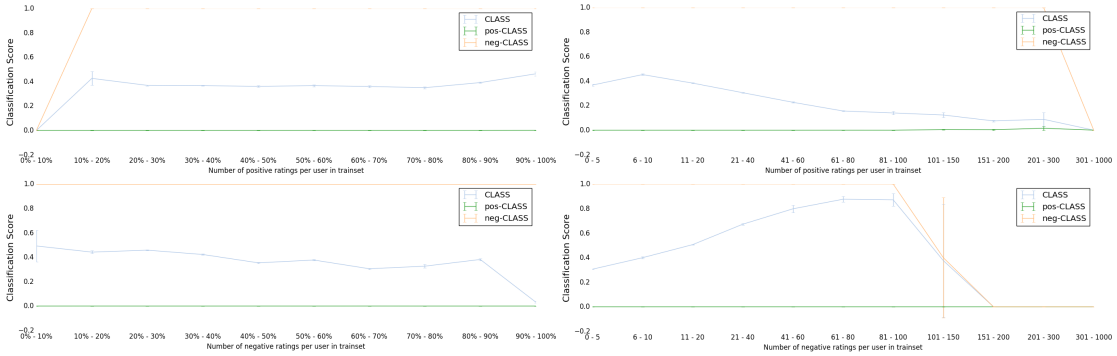
(f) Hit Rate

Figure A.17: Slashdot-Zoo Big - Logistic MF



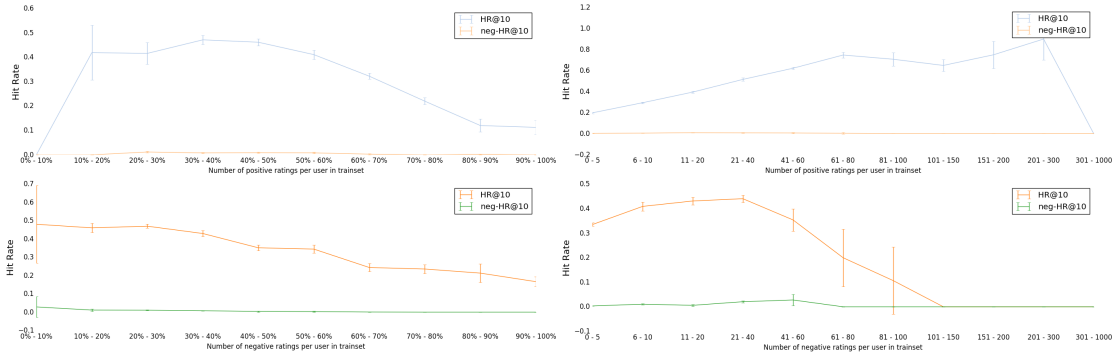
(a) Relative AUC

(b) AUC



(c) Relative Classification

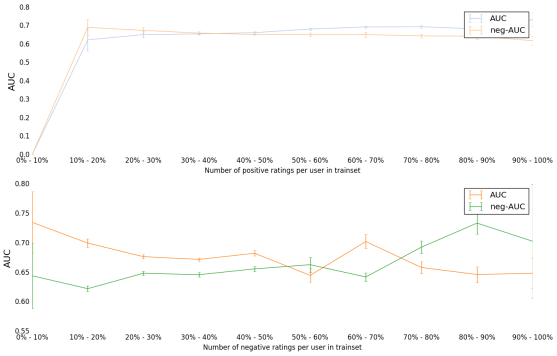
(d) Classification



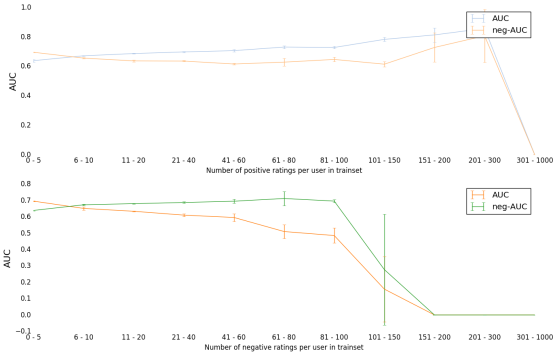
(e) Relative Hit-Rate

(f) Hit Rate

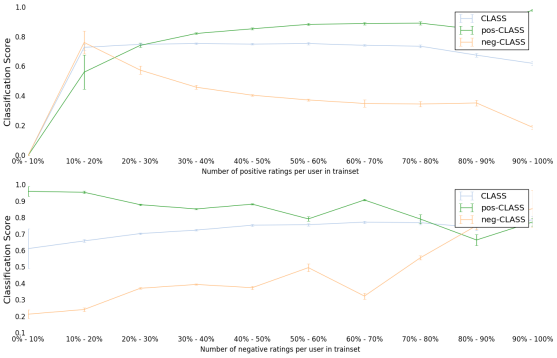
Figure A.18: Slashdot-Zoo Big - Logistic Likelihood MF



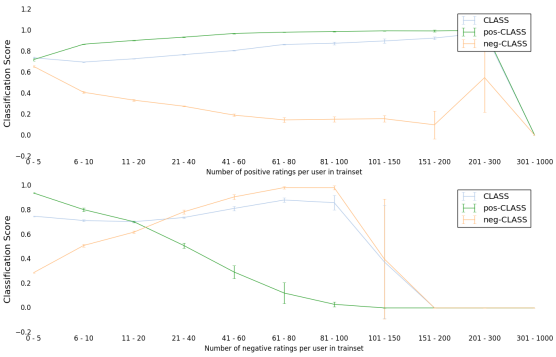
(a) Relative AUC



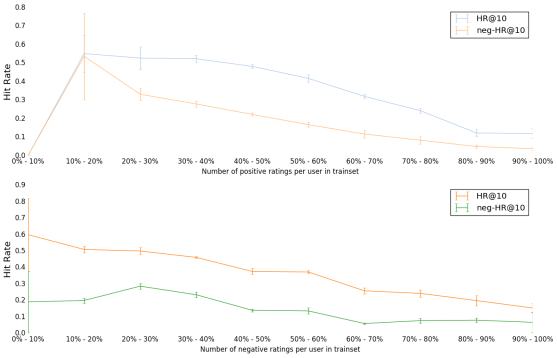
(b) AUC



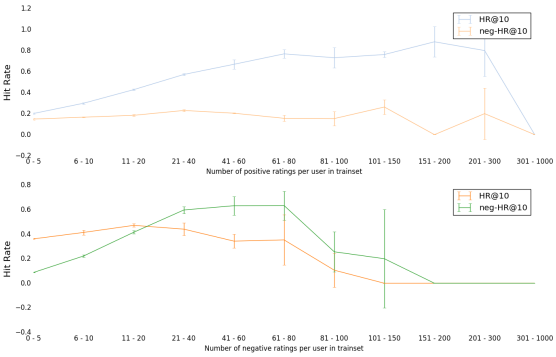
(c) Relative Classification



(d) Classification

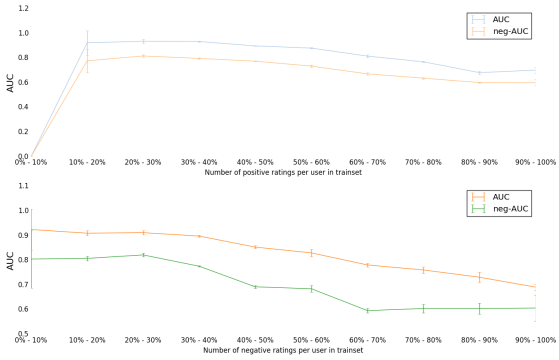


(e) Relative Hit-Rate

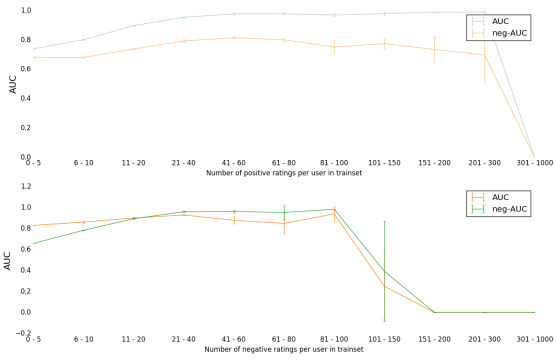


(f) Hit Rate

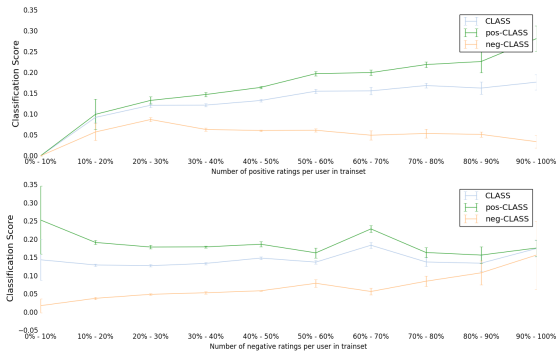
Figure A.19: Slashdot-Zoo Big - Explicit Logistik Likelihood MF



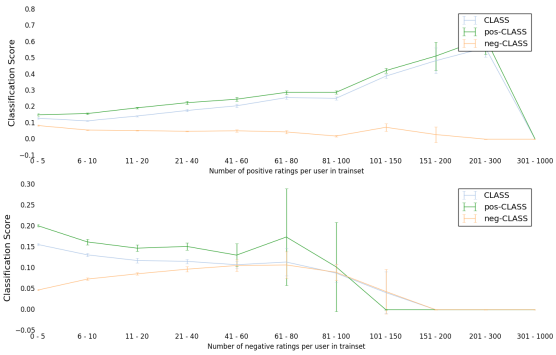
(a) Relative AUC



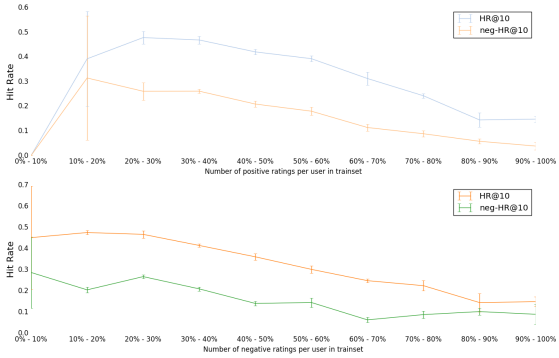
(b) AUC



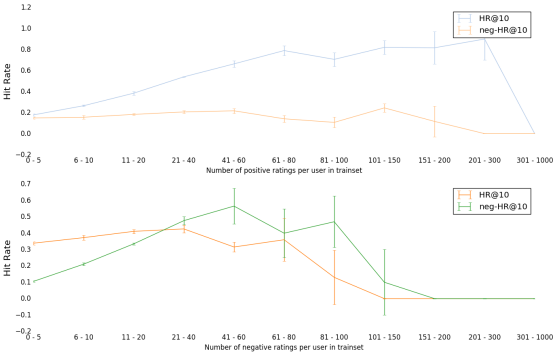
(c) Relative Classification



(d) Classification

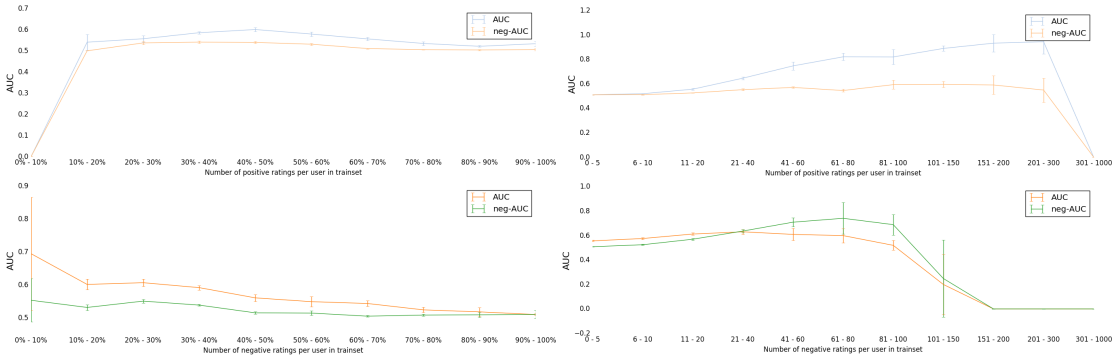


(e) Relative Hit-Rate



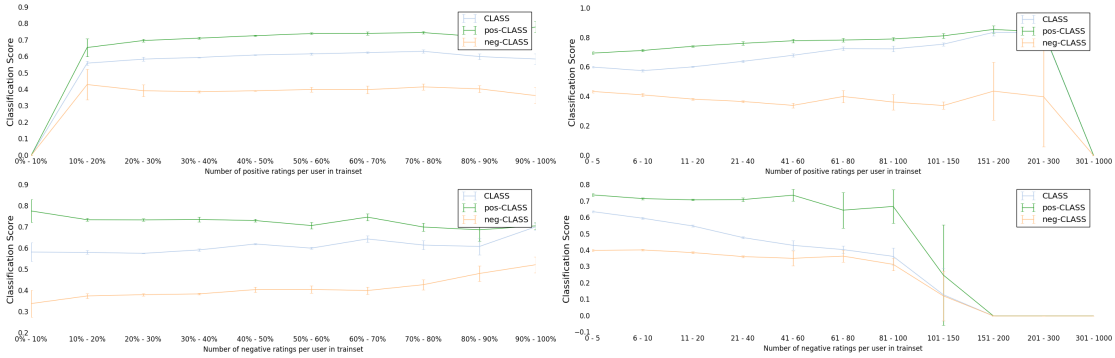
(f) Hit Rate

Figure A.20: Slashdot-Zoo Big - SGDReg



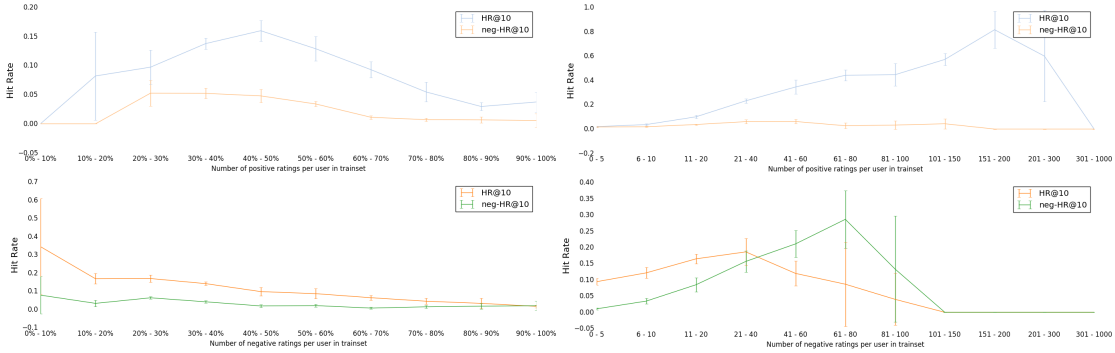
(a) Relative AUC

(b) AUC



(c) Relative Classification

(d) Classification

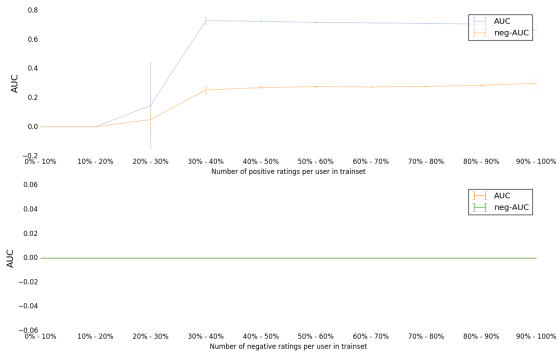


(e) Relative Hit-Rate

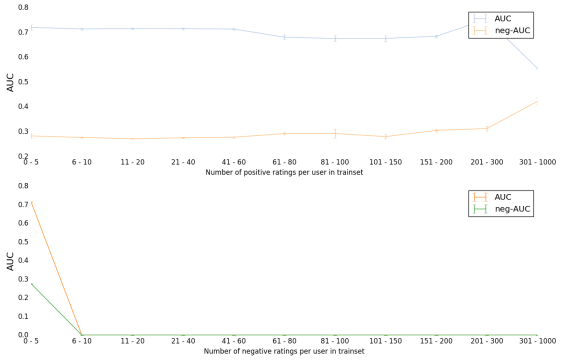
(f) Hit Rate

Figure A.21: Slashdot-Zoo Big - Multinominal Logistic MF

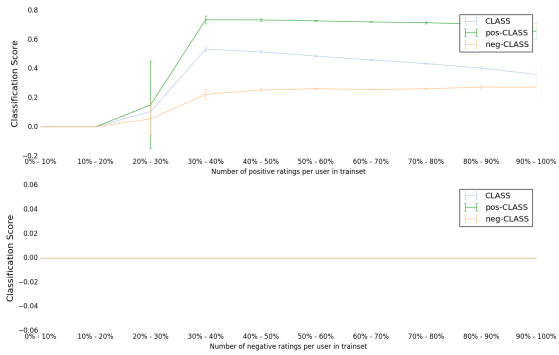
A.3.4 Book Crossing



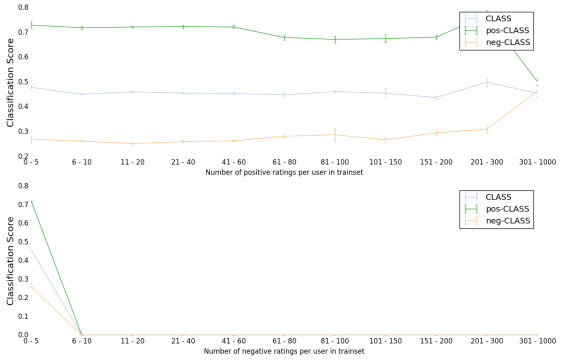
(a) Relative AUC



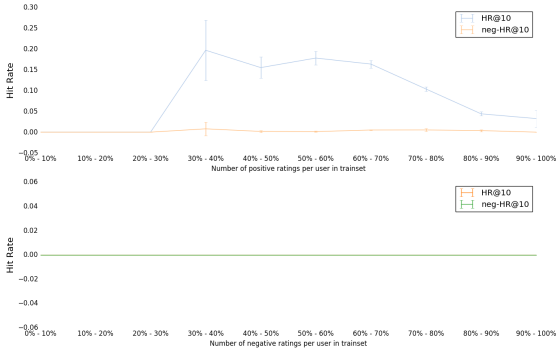
(b) AUC



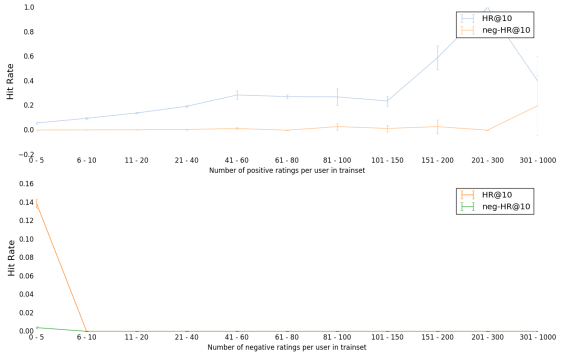
(c) Relative Classification



(d) Classification

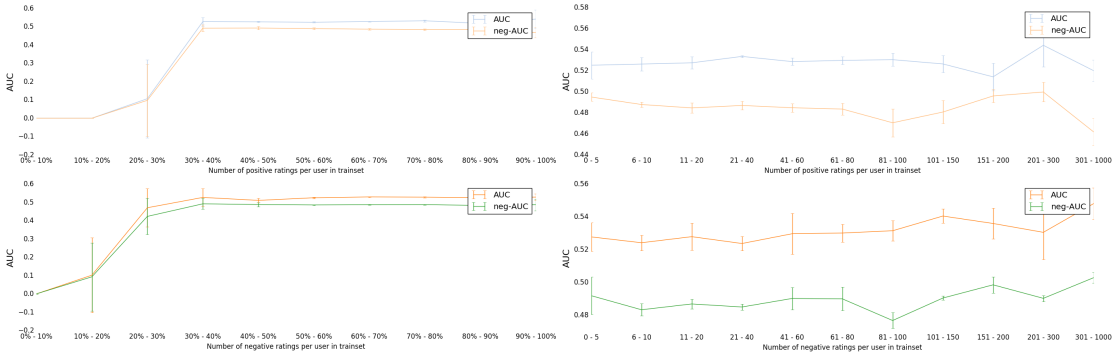


(e) Relative Hit-Rate



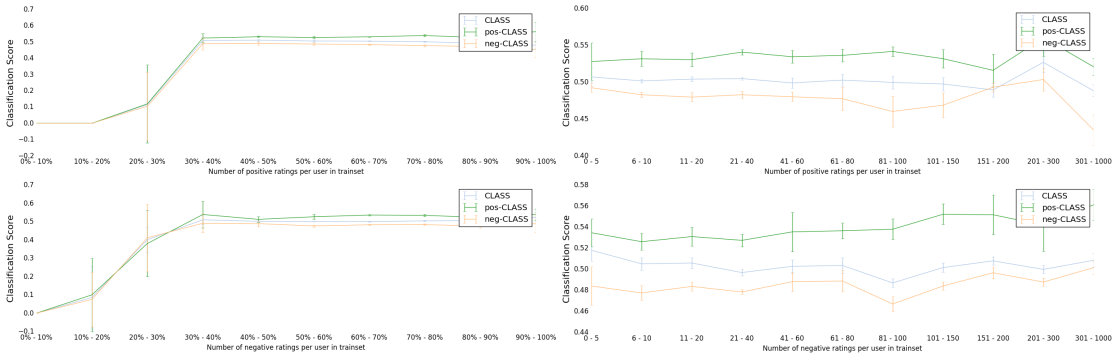
(f) Hit Rate

Figure A.22: Book Crossing - BPRMF Original



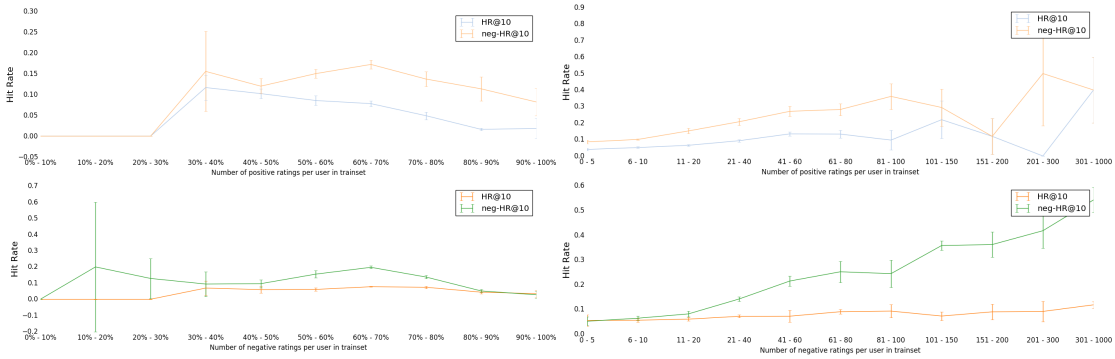
(a) Relative AUC

(b) AUC



(c) Relative Classification

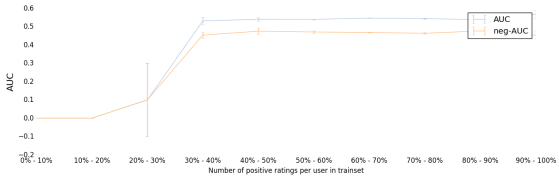
(d) Classification

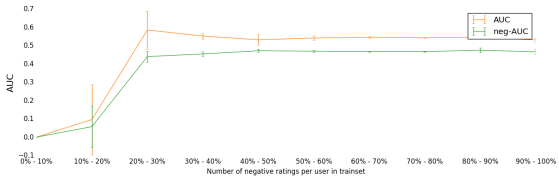


(e) Relative Hit-Rate

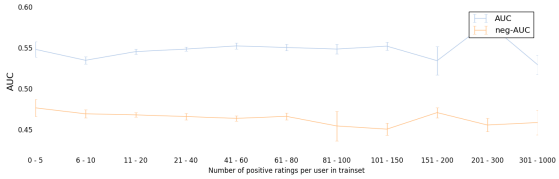
(f) Hit Rate

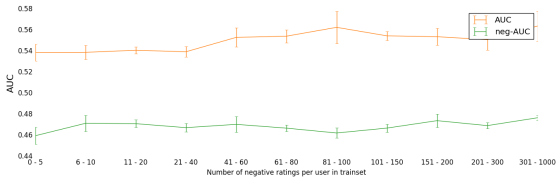
Figure A.23: Book Crossing - P-BPRMF



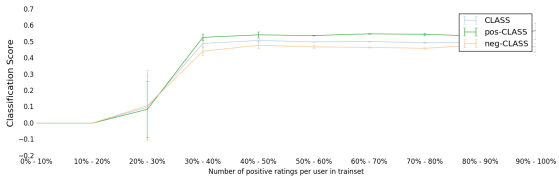


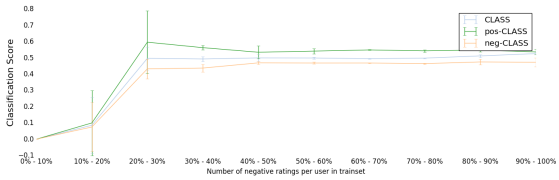
(a) Relative AUC



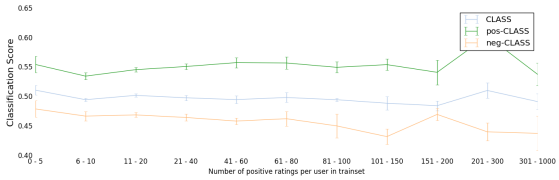


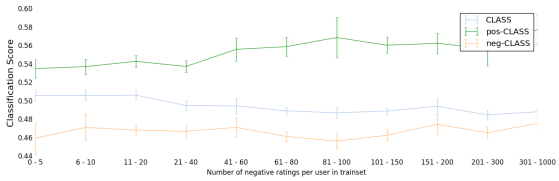
(b) AUC



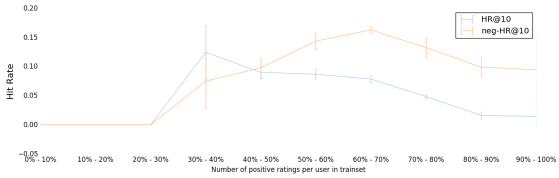


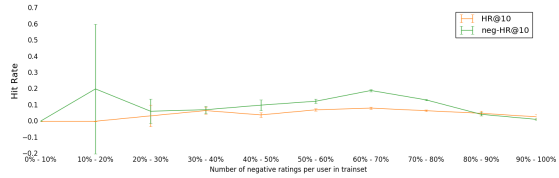
(c) Relative Classification



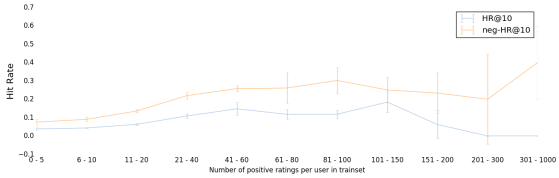


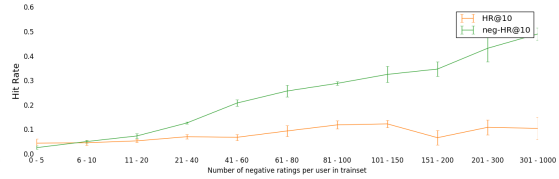
(d) Classification





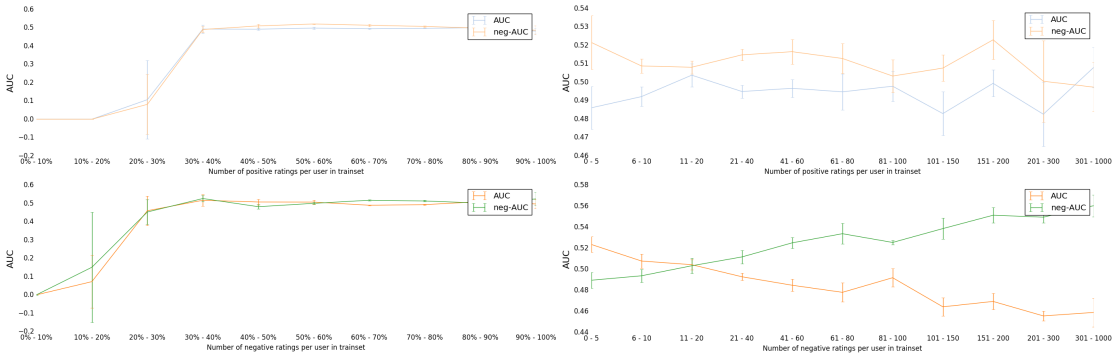
(e) Relative Hit-Rate





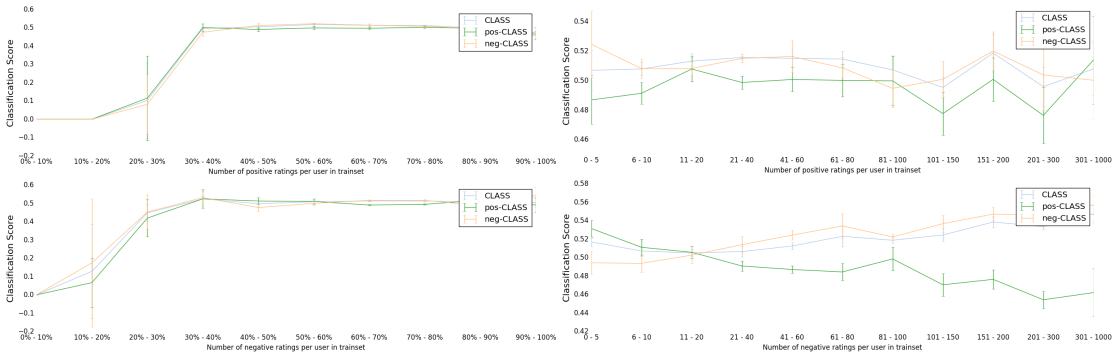
(f) Hit Rate

Figure A.24: Book Crossing - E-BPRMF



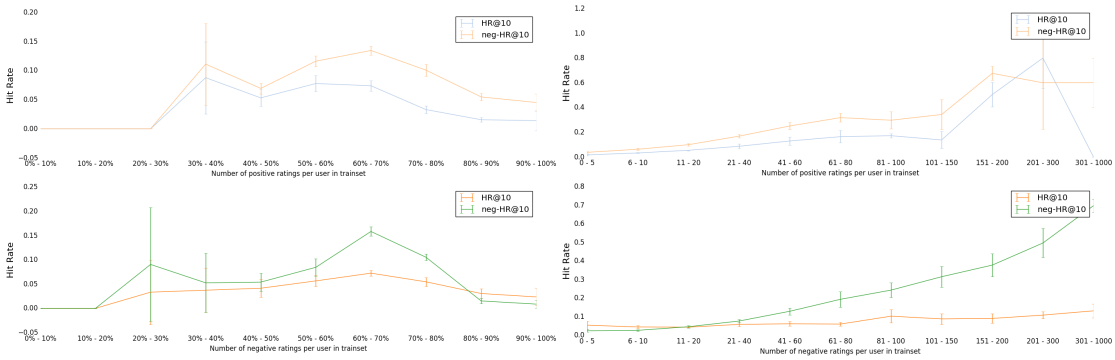
(a) Relative AUC

(b) AUC



(c) Relative Classification

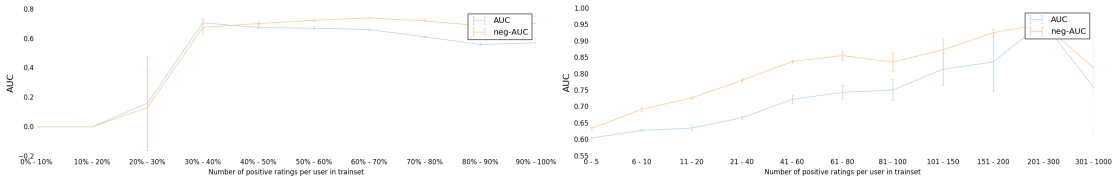
(d) Classification



(e) Relative Hit-Rate

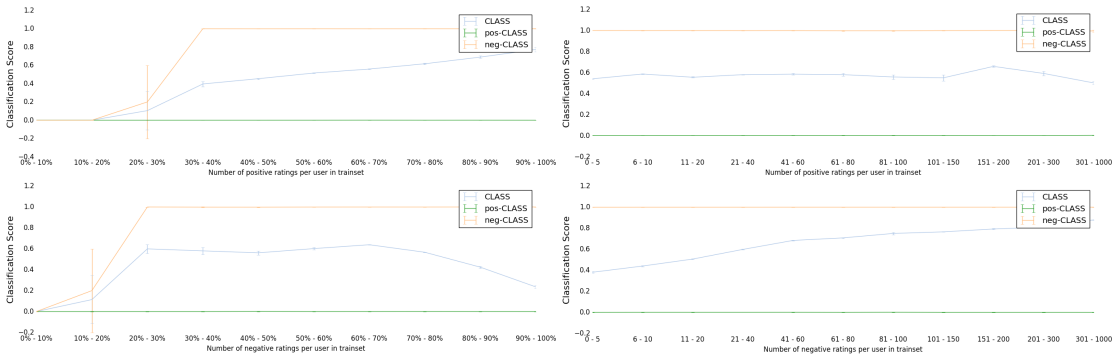
(f) Hit Rate

Figure A.25: Book Crossing - GAUC-OPT



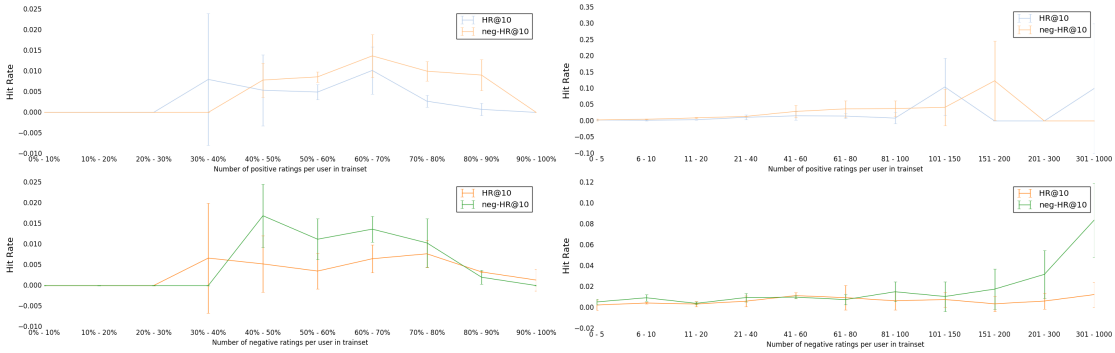
(a) Relative AUC

(b) AUC



(c) Relative Classification

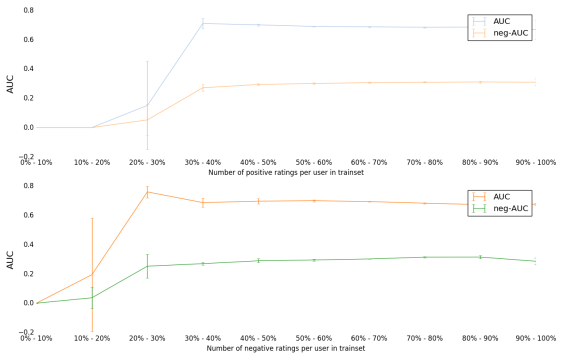
(d) Classification



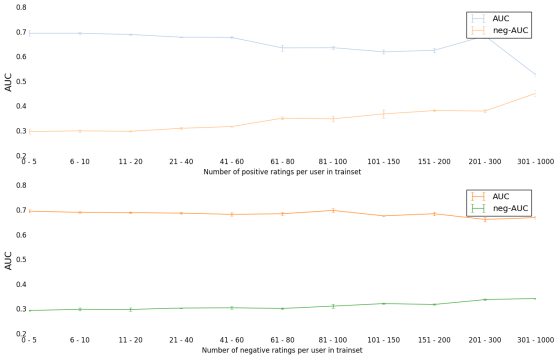
(e) Relative Hit-Rate

(f) Hit Rate

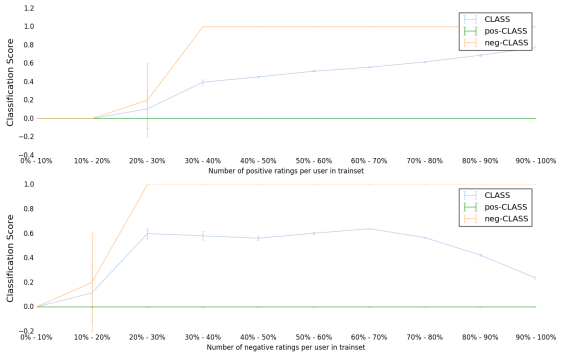
Figure A.26: Book Crossing - Logistic MF



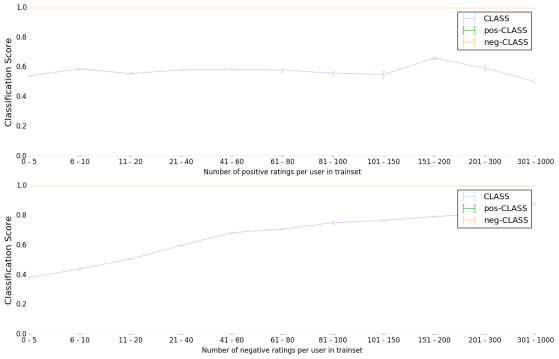
(a) Relative AUC



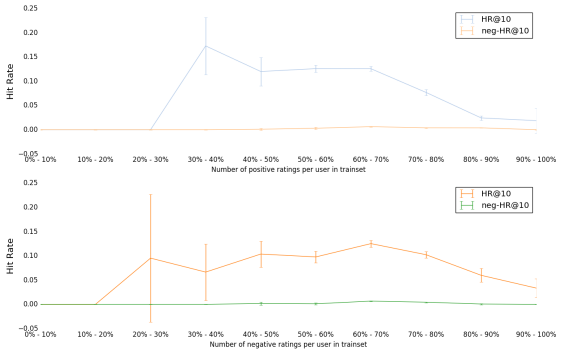
(b) AUC



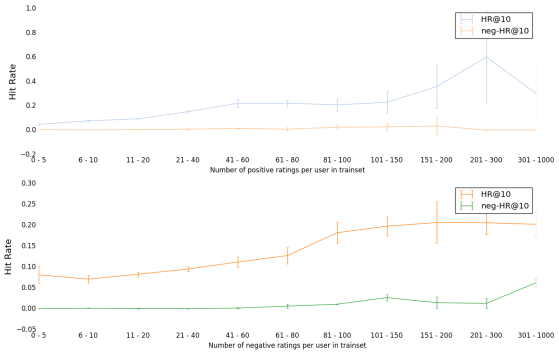
(c) Relative Classification



(d) Classification

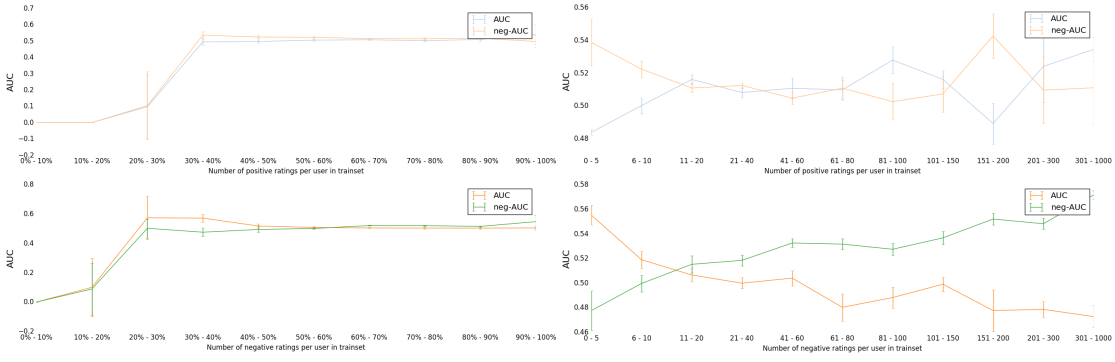


(e) Relative Hit-Rate



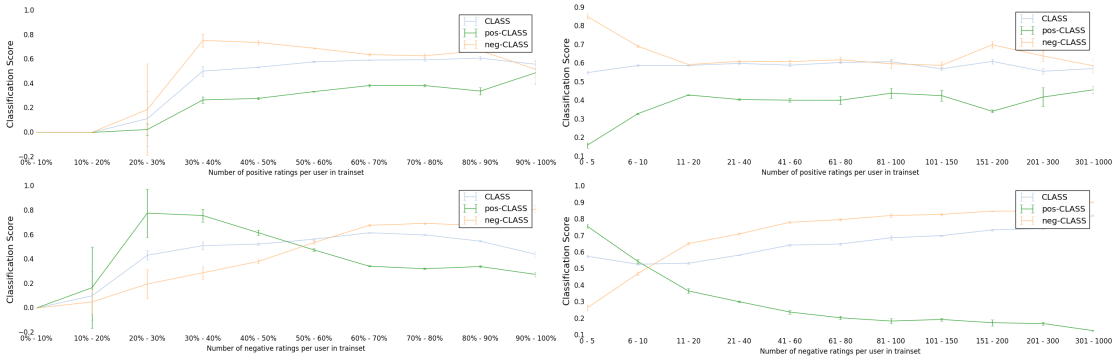
(f) Hit Rate

Figure A.27: Book Crossing - Logistic Likelihood MF



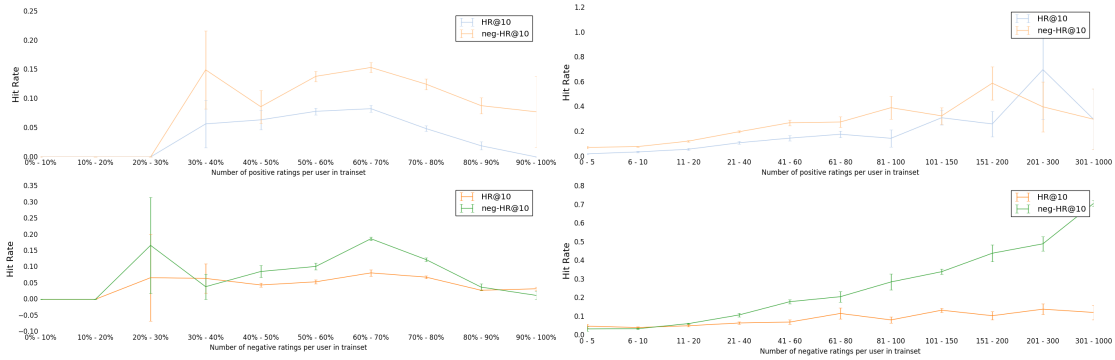
(a) Relative AUC

(b) AUC



(c) Relative Classification

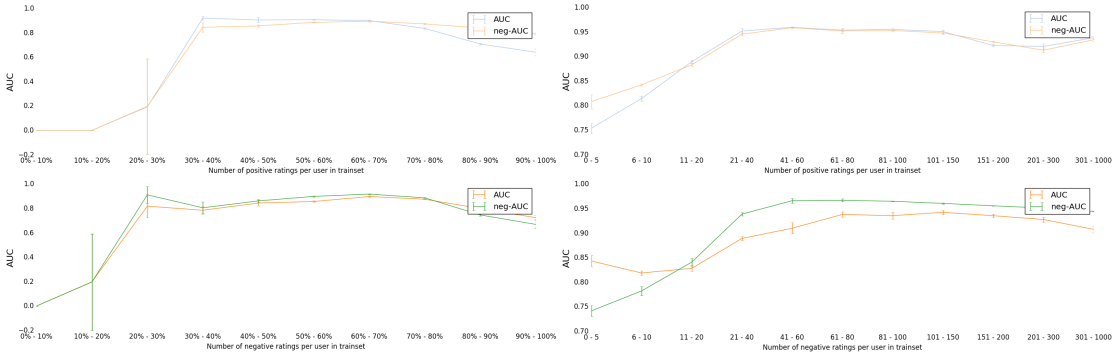
(d) Classification



(e) Relative Hit-Rate

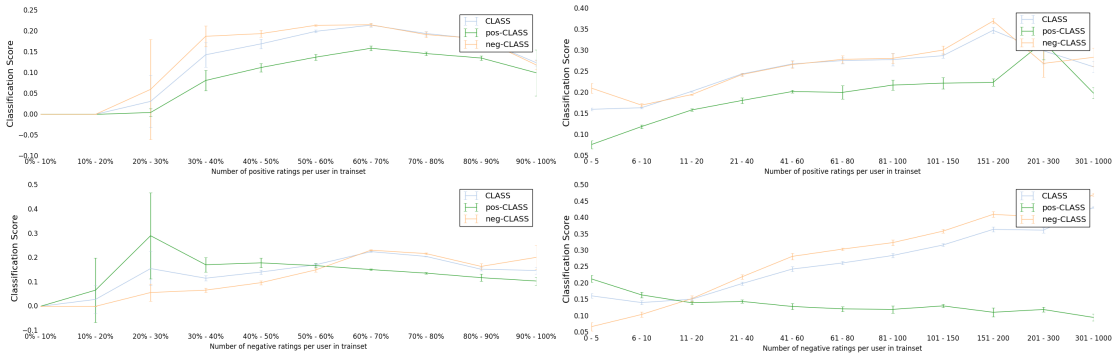
(f) Hit Rate

Figure A.28: Book Crossing - Explicit Logistik Likelihood MF



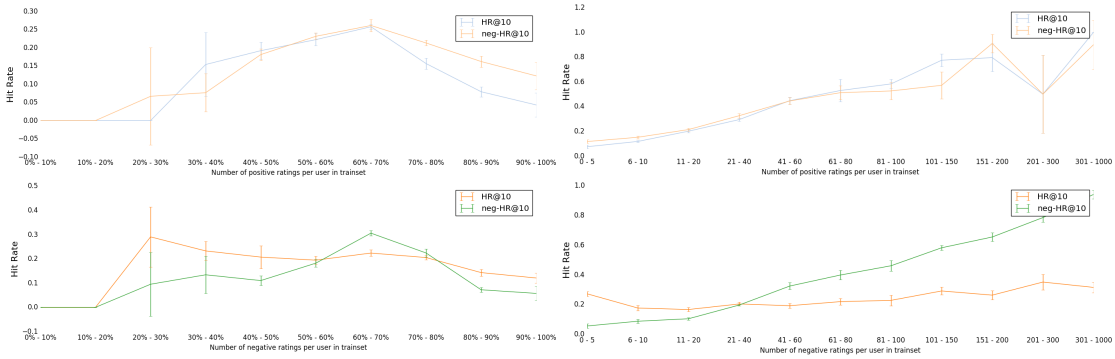
(a) Relative AUC

(b) AUC



(c) Relative Classification

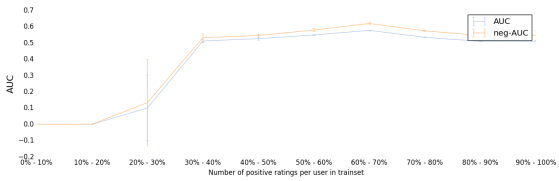
(d) Classification

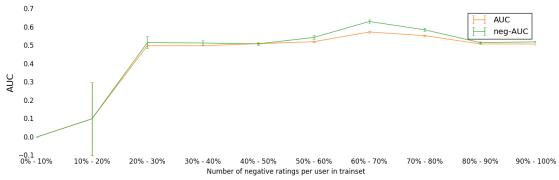


(e) Relative Hit-Rate

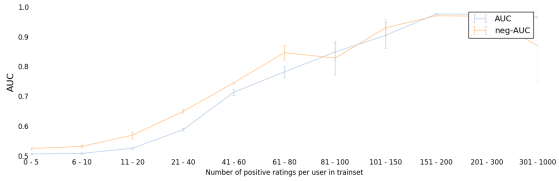
(f) Hit Rate

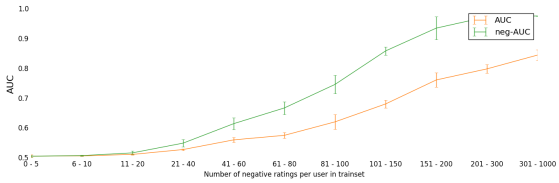
Figure A.29: Book Crossing - SGDReg



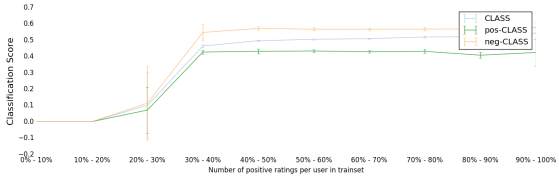


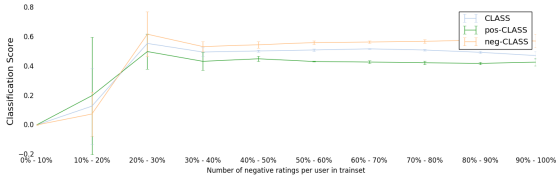
(a) Relative AUC



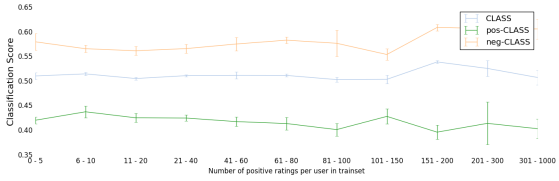


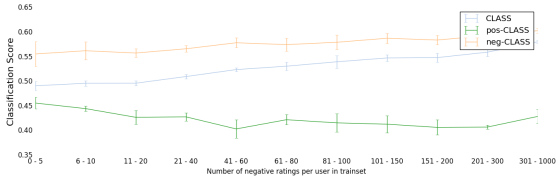
(b) AUC



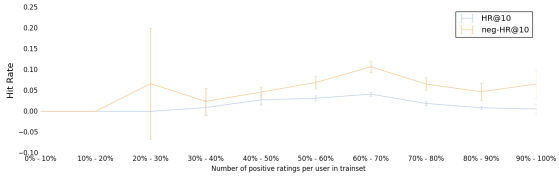


(c) Relative Classification

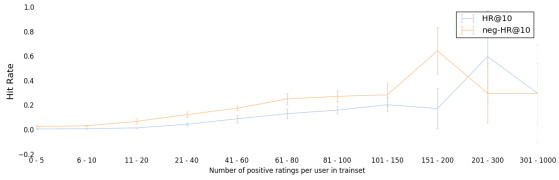


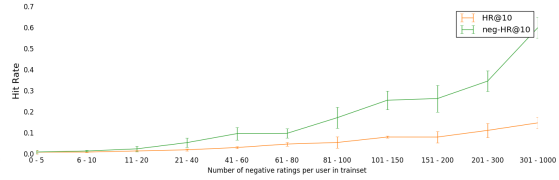


(d) Classification



(e) Relative Hit-Rate





(f) Hit Rate

Figure A.30: Book Crossing - Multinominal Logistic MF

List of Figures

List of Tables

List of Algorithms