On the Effectiveness of Linear Models for One-Class Collaborative Filtering

Suvash Sedhain[†*], Aditya Krishna Menon[*†], Scott Sanner[‡†], Darius Braziunas[§]
{[†]Australian National University, [*]NICTA}, Canberra, ACT, Australia
[‡] Oregon State University, Corvallis, OR, USA
[§] Rakuten Kobo Inc., Toronto, Canada
{suvash.sedhain, aditya.menon}@nicta.com.au, scott.sanner@oregonstate.edu, dbraziunas@kobo.com

Abstract

In many personalised recommendation problems, there are examples of items users prefer or like, but *no* examples of items they dislike. A state-of-the-art method for such *implicit feedback*, or *one-class collaborative filtering* (OC-CF), problems is SLIM, which makes recommendations based on a learned item-item similarity matrix. While SLIM has been shown to perform well on implicit feedback tasks, we argue that it is hindered by two limitations: first, it does not produce user-personalised predictions, which hampers recommendation performance; second, it involves solving a constrained optimisation problem, which impedes fast training. In this paper, we propose LRec, a variant of SLIM that overcomes these limitations without sacrificing any of SLIM's strengths. At its core, LRec employs linear logistic regression; despite this simplicity, LRec consistently and significantly outperforms all existing methods on a range of datasets. Our results thus illustrate that *the OC-CF problem can be effectively tackled via linear classification models*.

Introduction

Personalised recommendation systems are a core component of many modern e-commerce services. *Collaborative filtering* (*CF*) is the *de facto* approach to algorithmic recommendation, based on extracted information from a database of *preferences* for a *collection* of users (Goldberg et al. 1992). Most work on CF has considered the *explicit feedback* setting, where users express positive and negative preferences for items in terms of ratings or likes/dislikes. By contrast, in the *implicit feedback* setting, we do *not* have explicit negative preference information. For example, consider recommending items to users of an e-commerce website, based on their purchase history. One can assume that a purchase indicates a positive preference for an item. However, the *lack* of a purchase does *not* necessarily indicate a negative preference; it may just be that the user is unaware of an item. Similar problems arise when modelling users' play counts of songs, viewing of videos, *et cetera*. Implicit feedback scenarios are also referred to as *one-class collaborative filtering* (*OC-CF*) problems (Pan et al. 2008; Pan and Scholz 2009); we use the terms interchangeably.

While there is a rich literature on OC-CF (discussed subsequently), to our knowledge, all existing methods lack one or more desiderata. For example, methods that do not employ learning (such as neighbourhood approaches) cannot ensure full exploitation of available data. On the other hand, amongst methods that employ learning (such as matrix factorisation), it is common to employ learning objectives that are non-convex, which limits the range of methods for efficient global optimisation (if even possible); further, parallel training of such models is challenging, due to strong coupling between parameters.

SLIM (Ning and Karypis 2011) is a state-of-the-art method for OC-CF problems. Unlike neighbourhood and matrix factorisation approaches, SLIM is has a convex learning objective. While SLIM has been shown to perform well on OC-CF problems, we shall argue that the method is hindered by two limitations: first, it does not produce *user-personalised* predictions, which hampers its predictive performance; and second, it involves a *constrained optimisation*, which hampers fast training.

In this paper, we propose *LRec*, a variant of SLIM that overcomes these limitations without sacrificing any of SLIM's strengths: LRec produces user-personalised recommendations by training a convex, unconstrained, objective which is embarrassingly parallelisable (i.e., without distributed communication) across users. Experimentation on a range of real-world datasets reveals that LRec demonstrates dramatically superior performance to SLIM, as well as other more complicated approaches. At its core, LRec employs linear logistic regression; thus, our results illustrate the (perhaps surprising) level of effectiveness of simple yet powerful linear classification models for OC-CF problems.

Background and Notation

Let $\mathcal{U}$ denote a set of users, and $\mathcal{I}$ a set of items, with $m = |\mathcal{U}|$ and $n = |\mathcal{I}|$. In one-class collaborative filtering (OC-CF), we have a purchase[1] matrix $\mathbf{R} \in \{0, 1\}^{m \times n}$, where $\mathbf{R}_{ui}$ indicates whether user u purchased item i or not. We use $\mathbf{R}_{:i}$ to denote the indicator vector of each users' purchase of an item, and similarly $\mathbf{R}_{u:}$ to denote the vector of a

[1]We use the word "purchase" simply for the purposes of exposition. The method described in this paper is equally applicable in settings such as modelling users' play counts of songs.

user's preference for each item. We denote by $\mathcal{R}(u)$ the set of the items purchased by user u.

The goal in OC-CF is to learn a recommender, which for our purposes is simply a matrix $\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$ of the same size as $\mathbf{R}$. We call $\hat{\mathbf{R}}$ the *recommendation matrix*. As the entries are real-valued, for each user u, we can sort the entries of $\hat{\mathbf{R}}_{u:}$ to obtain a predicted ranking over items. We evaluate $\hat{\mathbf{R}}$ assuming we are in the *top-N recommendation* scenario (Deshpande and Karypis 2004): here, the interest is in finding $\hat{\mathbf{R}}$ such that the head of the ranked list for each user comprises items she will enjoy.

The challenge in OC-CF is the lack of explicit negative preference. A user not purchasing an item does not necessarily indicate a negative preference: the user may simply be unaware of the item. Henceforth, we say that (u, i) pairs with $\mathbf{R}_{ui} = 1$ denote "known positive preference", while pairs with $\mathbf{R}_{ui} = 0$ denote "unknown preference".

We now review some existing approaches to OC-CF. We focus on approaches that rely only on information provided in $\mathbf{R}$, and thus do not discuss approaches that exploit additional information, e.g. those that exploit an exogenous network amongst users or items (Yu et al. 2014; Sedhain et al. 2014).

Neighbourhood methods

There are two broad types of neighbourhood methods. The first type is *item-based* neighbourhood methods, where we produce a recommendation matrix of the form

$$\hat{\mathbf{R}} = \mathbf{R}\mathbf{S}_I, \tag{1}$$

where $\mathbf{S}_I \in \mathbb{R}^{n \times n}$ is an item-item similarity matrix. The second type is *user-based* neighbourhood methods, which effectively operate on $\mathbf{R}^T$ instead of $\mathbf{R}$, producing a recommendation matrix of the form

$$\hat{\mathbf{R}} = \mathbf{S}_U\mathbf{R} \tag{2}$$

where $\mathbf{S}_U \in \mathbb{R}^{m \times m}$ is a user-user similarity matrix. The intuition for both types of method is simple: observe that in item-based methods,

$$\hat{\mathbf{R}}_{ui} = \sum_{i' \in \mathcal{I}} \mathbf{R}_{ui'} \cdot \mathbf{S}_{i'i}.$$

That is, for a user u, one scores an item i as the *sum of i's similarity to all other items that u enjoys*. A similar intuition holds for user-based methods.

Typically, one chooses the similarity matrix $\mathbf{S}_*$ as a predefined function of $\mathbf{R}$, such as cosine similarity (Sarwar et al. 2001; Linden, Smith, and York 2003): for item-based methods, this is given by

$$\mathbf{S}_{i'i} = \frac{\mathbf{R}_{:i}^T \mathbf{R}_{:i'}}{||\mathbf{R}_{:i}||_2 ||\mathbf{R}_{:i'}||_2}.$$

Other examples are the Pearson correlation, Jaccard coefficient and conditional probability (Deshpande and Karypis 2004). It is typical to sparsify $\mathbf{S}$ by retaining only top-k similar entries in the similarity matrix.

Recommendation performance can be quite sensitive to the choice of $\mathbf{S}$, and the choice generally depends on the problem domain. Therefore, neighbourhood methods are unable to adapt to the characteristics of the data at hand.

Matrix factorisation

Matrix factorisation methods are the *de facto* approach to collaborative filtering with explicit feedback. The basic idea is to embed users and items into some shared latent space (Srebro and Jaakkola 2003; Koren, Bell, and Volinsky 2009). Formally, let $\mathbf{J} \in \mathbb{R}_+^{m \times n}$ be some weighting matrix to be defined shortly. Let $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be some loss function, typically squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. Then, matrix factorisation methods optimise

$$\min_\theta \sum_{u \in \mathcal{U}, i \in \mathcal{I}} \mathbf{J}_{ui} \cdot \ell(\mathbf{R}_{ui}, \hat{\mathbf{R}}_{ui}(\theta)) + \Omega(\theta), \tag{3}$$

where the recommendation matrix is[2]

$$\hat{\mathbf{R}}(\theta) = \mathbf{A}^T\mathbf{B} \tag{4}$$

for $\theta = \{\mathbf{A}, \mathbf{B}\}$, and the *regulariser* $\Omega(\theta)$ is typically

$$\Omega(\theta) = \frac{\lambda}{2} \cdot (||\mathbf{A}||_F^2 + ||\mathbf{B}||_F^2)$$

for some $\lambda > 0$. The matrices $\mathbf{A} \in \mathbb{R}^{K \times m}, \mathbf{B} \in \mathbb{R}^{K \times n}$ are the *latent representations* of users and items respectively, and $K \in \mathbb{N}_+$ the *latent dimension* of the factorisation.

In standard CF applications with explicit negative feedback, one typically sets (Koren, Bell, and Volinsky 2009) $\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} > 0]\!]$, so that one only considers (user, item) pairs with known preference information. In implicit feedback problems, this is not appropriate, as one will simply learn on the known positive preferences, and thus predict $\hat{\mathbf{R}}_{ui} = 1$ uninformatively. An alternative is to set $\mathbf{J}_{ui} = 1$ uniformly. This treats all absent purchases as indications of a negative preference. The resulting approach is termed PureSVD in (Cremonesi, Koren, and Turrin 2010), and has been shown to perform surprisingly well in top-N recommendation task for both explicit and implicit feedback datasets (Cremonesi, Turrin, and Airoldi 2011).

As an intermediate between the two extreme weighting schemes above, the WRMF method (Pan et al. 2008) sets $\mathbf{J}_{ui}$ to be

$$\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} = 0]\!] + \alpha \cdot [\![\mathbf{R}_{ui} > 0]\!] \tag{5}$$

where α assigns an importance weight to the observed ratings. More sophisticated weighting schemes have also been explored. In scenarios where there are multiple observations for each (user, item) pair, (Hu, Koren, and Volinsky 2008) considered a logarithmic weighting of these counts.

Bayesian personalised ranking (BPR)

An alternate strategy to adapt matrix factorisation to the implicit feedback setting is the Bayesian Personalised Ranking (BPR) model (Rendle et al. 2009). BPR optimises a loss over (user, item) pairs, so that the known positive preferences score at least as high as the unknown preferences:

$$\min_\theta \sum_{u \in \mathcal{U}, i \in \mathcal{R}(u), i' \notin \mathcal{R}(u)} \ell(1, \hat{\mathbf{R}}_{ui}(\theta) - \hat{\mathbf{R}}_{ui'}(\theta)) + \Omega(\theta), \tag{6}$$

where $\ell(1, v) = \log(1 + e^{-v})$ is the logistic loss, and $\hat{\mathbf{R}}$ is as per Equation 4. Intuitively, this does not disallow high scores for items with unknown preferences, but simply places these scores below that of an item with a positive preference.

[2] We omit user- and item- bias terms for brevity.

SLIM

SLIM (Ning and Karypis 2011) directly learns an item-similarity matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ via

$$\min_{\mathbf{W}\in\mathcal{C}} ||\mathbf{R} - \mathbf{R}\mathbf{W}||_F^2 + \frac{\lambda}{2}||\mathbf{W}||_F^2 + \mu||\mathbf{W}||_1, \quad (7)$$

where $\lambda, \mu > 0$ are appropriate constants, and

$$\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{n\times n} \colon \operatorname{diag}(\mathbf{W}) = 0, \mathbf{W} \geq 0\}. \quad (8)$$

Here, $||\cdot||_1$ is the elementwise ℓ_1 norm of $\mathbf{W}$ which encourages sparsity, and the constraint $\operatorname{diag}(\mathbf{W}) = 0$ prevents a trivial solution of $\mathbf{W} = \mathbf{I}_{n\times n}$. The nonnegativity constraint offers interpretability, but (Levy and Jack 2013) showed that it can be ignored without affecting performance. Given a learned $\mathbf{W}$, SLIM produces a recommendation matrix

$$\hat{\mathbf{R}}(\theta) = \mathbf{R}\mathbf{W}.$$

Thus, SLIM can be seen as an item-based neighbourhood approach where the similarity matrix $\mathbf{S}$ is *learned* from data.

The LRec Model

We now describe LRec, our approach to OC-CF based on linear classification. Define a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ by

$$\mathbf{X} = \mathbf{R}^T,$$

and, for each $u \in \mathcal{U}$, define a vector $\mathbf{y}^{(u)} \in \{\pm 1\}^n$ by:

$$\mathbf{y}^{(u)} = 2\mathbf{R}_{u:} - 1.$$

Then, LRec solves

$$\min_{\theta} \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{I}} \ell(\mathbf{y}_i^{(u)}, \mathbf{X}_{i:}\mathbf{w}^{(u)}) + \Omega(\theta), \quad (9)$$

where ℓ is some convex loss function, and $\theta = \{\mathbf{w}^{(u)}\}_{u\in\mathcal{U}}$. Each $\mathbf{w}^{(u)} \in \mathbb{R}^m$, and so we can equivalently think of $\theta = \{\mathbf{W}\}$ for some $\mathbf{W} \in \mathbb{R}^{m\times m}$, with $\mathbf{w}^{(u)} = \mathbf{W}_{:u}$. Given a learned $\mathbf{W}$, LRec produces a recommendation matrix

$$\hat{\mathbf{R}}(\theta) = \mathbf{W}^T\mathbf{R}. \quad (10)$$

In the sequel, we will employ $\Omega(\theta) = \frac{\lambda}{2}||\mathbf{W}||_F^2$. We shall also choose ℓ to be the logistic loss, $\ell(y, v) = \log(1+e^{-yv})$, which we favour due to the existence of efficient solvers for linear logistic regression, such as LibLinear (Fan et al. 2008), and its suitability for estimating class-probabilities, which we shall see is appropriate for OC-CF problems.

Equation 9 can be interpreted as a linear classification model for each user u. For each such model, we have a separate training example corresponding to each item in the dataset. The feature vector for each item comprises the purchase history for *all* users, *including* the one in consideration[3]. The label for each item is simply whether or not the user in consideration purchased it. Each $\mathbf{W}_{uu'}$ can be seen as a weight indicating the relevance of the purchases of user u in predicting the purchases of the user u'.

The above assumes the use of a linear kernel for the underlying classification model. In principle, one could also use a nonlinear kernel. However, we did not explore this owing to the strong performance of linear logistic regression, and the comparatively slower training of kernel methods.

[3]This is only done for the *training* purchases of each user. Therefore, there is no leakage of test purchases into the model.

Properties of LRec

Several observations are worth emphasising at this point. First, training of the model is highly parallelisable across the users, as the weights $\mathbf{w}^{(u)}$ do not depend on each other. Parallelisation may thus be conducted *without distributed communication*. Further, the feature matrix $\mathbf{X}$ is identical across all users u. Therefore, this matrix can be *shared* across all such parallel executions of a per-user model, which is useful on a multi-core architecture.

Second, the feature matrix for each model is highly sparse: we expect that for most items, only a small fraction of users will have purchased them. This means that the optimisation of each user's model can be done efficiently, employing e.g. sparse matrix-vector computations. Also, when $m \gg n$, optimising the dual objective reduces training time significantly over optimising the primal.

Third, with a large number of users, we have a high-dimensional feature representation, which introduces the apparent risk of overfitting. Our use of ℓ_2 regularisation mitigates this issue. Further, observe that if we optimise the dual objective, the effective number of parameters for each user is simply the total number of items n, which may be considerably smaller than m.

Fourth, the use of ℓ_2 regularisation is crucial beyond its value in preventing overfitting. Recall that for each user u's model, that user's purchase history is included as a feature. Without regularisation, then, the solution to Equation 9 is trivial: we simply let $\mathbf{W}_{uu'} = 0$ for all $u \neq u'$, and let $\mathbf{W}_{uu} \to +\infty$. However, with ℓ_2 regularisation, such a solution is penalised. What is favoured instead is a spreading of weight across other *similar* users.

Fifth, the objective in Equation 9 is strictly convex. Therefore, there are no issues of local optima.

Sixth, in the model for each user, LRec treats unknown preferences as negative examples. It appears that this falls into the trap of suppressing preferences for items that the user is unaware of. However, we can view the learning problem for each user as one of learning from *positive and unlabelled data*. (Elkan and Noto 2008) proved a surprising fact about learning from such data, namely, that for the purposes of *ranking*, it suffices to simply treat the unlabelled examples as negative, and estimate class-probabilities. As we are only interested in ranking performance for top-N recommendation, this justifies treating unknown preferences as negative, provided we choose an ℓ in Equation 9 that is capable of estimating class-probabilities, e.g. logistic ($\ell(y, v) = \log(1 + e^{-yv})$) or squared ($\ell(y, v) = (1 - yv)^2$) loss.

Relation to existing methods

We now contrast LRec to existing methods for recommendation with implicit feedback. We shall argue that LRec is the only proposed method (to our knowledge) that is learning based, with a convex objective that is efficient to train in parallel, and which produces user-personalised predictions.

Relation to SLIM The model most closely related to LRec is SLIM (Equation 7). To see the connection, observe that we may rewrite SLIM as (see also (Ning and Karypis

2011, Equation 4))

$$\min_{\mathbf{W} \in \mathcal{C}} \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}} \ell(\mathbf{y}_u^{(i)}, \mathbf{X}_{u:}^{(i)} \mathbf{w}^{(i)}) + \Omega(\mathbf{W}),$$

where $\mathcal{C}$ is as in Equation 8, $\ell(y, v) = (y - v)^2$ is the square loss, and the feature matrix $\mathbf{X}$ and label vector $\mathbf{y}^{(i)}$ is

$$\mathbf{X} = \mathbf{R} \ ; \ \mathbf{y}^{(i)} = \mathbf{R}_{:i}.$$

LRec can thus be seen as a variant of SLIM where: (1) one learns a user-user rather than item-item weight matrix; (2) there are no constraints in performing the optimisation; and (3) the target user is included when training each model. As a result of these design choices, LRec has three main advantages over SLIM.

User-focussed modelling. We can see SLIM as learning a separate model for each *item*, where in each such model we treat each user as a training instance, represented by their purchase history. This is in contrast to LRec, which learns a separate model for each *user*. While this difference appears trivial, it is very important in the context of recommendation. LRec ensures that its predictions are *personalised to each user*: for each user, the model attempts to produce the best possible ranking over items. By contrast, SLIM solves a subtly different problem, namely, ensuring that for each *item*, there is a good ranking over the users that would be interested in it. That is, SLIM can be seen as optimising a ranking for a fictional *average user*, and thus does not produce genuinely personalised predictions. Our experiments demonstrate that personalising the results towards each user, as in LRec, yield significant performance improvements.

Note that there may be recommendation scenarios where user-focussed modelling is not possible due to limited feedback. For example, a news site may wish to recommend articles to visitors without having any information about their viewing or preference history. In these scenarios, item-based recommendation may be superior. But in the many important applications where users do provide sufficient amounts of feedback, user-focussed modelling can be significantly more accurate and useful.

Optimisation. With the choice of logistic loss, the unconstrained LRec objective (Equation 9) can be optimised via standard logistic regression solvers. By contrast, optimisation of SLIM requires respecting the nonnegativity constraint on the weights, as well as performing ℓ_1 regularisation. The resulting optimisation is not as efficient as the unconstrained version (Levy and Jack 2013), and thus LRec can scale to larger datasets. This is illustrated in our results, where SLIM is unable to run on a large real-world dataset (the Million Song Dataset), unlike LRec.

Inclusion of target user. Another subtle difference between LRec and SLIM is the definition of the constraint set $\mathcal{C}$. In SLIM, the use of $\text{diag}(\mathbf{W}) = 0$ is crucial, because otherwise as noted we can just set $\mathbf{W} = \mathbf{I}$ and attain zero loss. However, such a solution will *not* be optimal when using logistic loss, as in LRec. Therefore, this means that the constraint $\text{diag}(\mathbf{W}) = 0$ is unnecessary. Dropping the constraint is beneficial from the perspective of parallelisation, as we can share the feature matrix $\mathbf{X}$ across multiple models.

Relation to neighbourhood methods As item-based neighbourhood approaches are subsumed by SLIM, the above advantages of LRec over SLIM apply to such methods as well. Compared to user-based neighbourhood methods, LRec uses a similarity matrix that is *learned from data*: the weight matrix $\mathbf{W}$ learned by LRec is a user-user similarity matrix, chosen so as to optimise recommendation performance for each individual user.

Relation to matrix factorisation Compared to matrix factorisation approaches, LRec does not involve projection of users and items into a shared latent space. However, we can interpret factorisation approaches as follows. Consider WRMF with $\mathbf{J}_{ui} = 1$ uniformly. Then, the optimal solution for the item latent factors is clearly

$$\mathbf{B} = (\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^{-1} \mathbf{A}\mathbf{R}.$$

This means that the factorisation equally predicts

$$\hat{\mathbf{R}} = \mathbf{S}\mathbf{R},$$

where $\mathbf{S} = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^{-1} \mathbf{A}$ is a rank K matrix. (A similar analysis was performed in (Hu, Koren, and Volinsky 2008).) Thus, we can view matrix factorisation as solving a surrogate to

$$\min_{\mathbf{W} \in \mathcal{C}} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \ell(\mathbf{y}_i^{(u)}, \mathbf{X}_{i:} \mathbf{w}^{(u)}) + \Omega(\mathbf{W}),$$

where $\mathcal{C} = \{\mathbf{W} \in \mathbb{R}^{m \times m} \colon \text{rank}(\mathbf{W}) = K\}$. There are three main advantages to LRec over such approaches.

Convexity of objective. Matrix factorisation imposes the constraint that the weight matrix is rank K. This can be viewed as performing reduced rank regression. While a low-rank assumption is plausible, it makes the resulting optimisation non-convex. By contrast, LRec allows $\mathbf{S}$ to be full-rank, and uses an ℓ_2 penalty to prevent overfitting, resulting in a strongly convex objective.

Parallelisation w/o distributed communication. Matrix factorisation models can be trained via alternating least squares (ALS) (Koren, Bell, and Volinsky 2009). For a fixed set of item latent features $\mathbf{B}$, we can learn the user latent features $\mathbf{A}$ by performing least squares using a design matrix $\mathbf{X} = \mathbf{B}$ and, for each $u \in \mathcal{U}$, a target vector $\mathbf{y}^{(u)} = \mathbf{R}_{u:}^T$. As in LRec, this operation can be parallelised across users. However, when it comes to learning the item latent features, all distributed nodes must send their estimates of $\mathbf{A}$ to a central source. As ALS typically requires several such alternation iterations, this communication may become expensive. In LRec, such distributed communication is not required: both learning and recommendation for all users can be done completely independently.

Richer model class. Matrix factorisation considers weight matrices $\mathbf{W}$ that are low-rank. This might induce *underfitting* if one does not choose sufficiently many latent features K in the factorisation. As LRec does not constrain $\mathbf{W}$ explicitly, but instead merely penalises its Frobenius norm, it employs a richer model class. As we shall see in our experiments, this endows LRec with better predictive performance.

Incorporating side-information

As a final comment, we note that it is easy to incorporate any side-information in the form of feature vectors for users and items into LRec. Suppose each item has a feature vector $\mathbf{v} \in \mathbb{R}^d$. Let $\mathbf{V} \in \mathbb{R}^{n \times d}$ denote the matrix of features for all items. Then, we can use exactly the same core LRec model as Equation 9, but change the feature representation to

$$\mathbf{X} = \begin{bmatrix} \mathbf{R}^T & \mathbf{V} \end{bmatrix} \in \mathbb{R}^{n \times (m+d)}.$$

The learned weights are now $\mathbf{W} \in \mathbb{R}^{m \times (m+d)}$, so that for each user we additionally learn affinities to the item features.

Experimental Evaluation

We now report experimental results of LRec against competing methods on a number of implicit feedback datasets.

Methods compared

We compared LRec trained with the logistic loss (Equation 9) to a number of baselines:

- Recommending to each user the most popular items they have not purchased (**Popularity**).
- User- and item-based nearest neighbour (**U-KNN** and **I-KNN**), using the best performing of the cosine similarity and Jaccard coefficient to define $\mathbf{S}$.
- **PureSVD** of (Cremonesi, Koren, and Turrin 2010).
- Weighted matrix factorisation (**WRMF**), Equation 3.
- Logistic matrix factorisation (**LogisticMF**), as per Equation 3 with the weights $\mathbf{J}_{ui} = [\![\mathbf{R}_{ui} > 0]\!]$, and ℓ being the logistic loss.
- Bayesian Personalised Ranking (**BPR**), Equation 6.
- **SLIM**, as per Equation 7. For computational convenience, we used the SGDReg variant (Levy and Jack 2013), which removes the nonnegativity constraint.
- Two variants of LRec, one which uses square rather than logistic loss (**LRec+Sq**), and one which additionally employs ℓ_1 regularisation and nonnegativity of weights (**LRec+Sq+ℓ_1+NN**). The weight constraints in the latter approach are as employed by SLIM.

For the neighbourhood methods, we tuned the neighbourhood size from $\{10, 20, 40, 80, 120, 160, n/m\}$. For methods relying on a latent dimension K, we tuned this from $\{10, 20, 40, 80, 120, 160\}$. For methods relying on ℓ_2 regularisation, we tuned this from $\lambda \in \{1000, 100, 10, 1, 0.1, 0.01, 0.001, 0.0001\}$. For WRMF, the weight α (Equation 5) was tuned from $\{1, 5, 10, 20, 40, 80\}$.

Datasets used

We evaluated all methods on four datasets: (a) ML1M, the MovieLens 1M dataset, (b) KOBO, an e-book purchase dataset from Rakuten Kobo Inc.[4], (c) LASTFM, a song play count dataset from LastFM (Celma 2008), and (d) MSD, the Million Song dataset (Bertin-Mahieux et al. 2011). For ML1M, we binarised ratings of 4 and 5 to be positive preferences. For the LASTFM and MSD datasets, we binarised by setting all nonzero playcounts to 1. Table 1 summarises statistics of these datasets.

[4] http://www.kobo.com

Table 1: Summary of datasets used in evaluation.

Dataset	m	n	$\lvert\mathbf{R}_{ui} > 0\rvert$
ML1M	6,038	3,533	575,281
KOBO	38,868	170,394	89,815
LASTFM	992	107,398	821,011
MSD	1,019,318	384,546	48,373,586

Evaluation protocol

For the ML1M and LASTFM datasets, we estimate performance based on 10 random train-test splits, similar to (Pan et al. 2008; Johnson 2014). For each split, we remove a random 20% subset of all (user, item) pairs (regardless of whether purchased or not), and place them in the test set. We then make predictions using the training set, and evaluate the performance against the test set. We report the mean test split performance, along with standard errors corresponding to 95% confidence intervals. To evaluate performance, we report Precision@20 (averaged across all test fold users), and mean average precision (mAP@100).

For KOBO, we followed the above except using *temporally* divided train and test splits. In each split, the training set includes all purchases prior to a specific date, and the test set all purchases in the following week.

For MSD, we use the provided training set to learn all models. We evaluate performance on 10 *random subsets* of the provided test set. In each such random subset, we pick 500 random test users to evaluate performance on.

Tables of Results

The experimental results (Table 2) indicate the following:

- LRec and LRec+Sq consistently outperform *all* existing methods by a statistically significant margin. In terms of the mAP@100 score, the % improvement over the next best existing method is 3.4% on ML1M, 7.7% on KOBO, 11.5% on LASTFM, and 14.2% on MSD.
- LRec+Sq consistently outperforms LRec+Sq+ℓ_1+NN. This suggests that the constraints on $\mathbf{W}$ and ℓ_1 regularisation (as imposed by SLIM) are harmful for retrieval.
- On all but the ML1M dataset, the number of (test) users is significantly fewer than the number of items. Therefore, methods that are item-focussed tend to strongly underperform. For example, on the LASTFM dataset, the otherwise competitive SLIM method performs much worse than all other methods.
- Amongst the matrix factorisation methods, WRMF performs the best in all datasets, and is competitive with the neighbourhood methods. The better performance of LRec suggests that the rank restriction of WRMF makes the model under fit to the data.
- BPR generally underperforms, likely since it optimises AUC that does not target the head of the ranked list.
- The scale of MSD is challenging: several methods, including WRMF and SLIM, did not finish running in a day[5]. Compared to similarly scalable neighbourhood methods, LRec's performance is significantly superior.

[5] LRec was executed in a parallel environment, whereas the

Table 2: Results of OC-CF methods on benchmark datasets. "DNF" indicates a method did not finish running in 24 hours.

	mAP@100				prec@20			
	ML1M	LASTFM	KOBO	MSD	ML1M	LASTFM	KOBO	MSD
Popularity	0.069 ± 0.001	0.086 ± 0.001	0.036 ± 0.015	0.018 ± 0.006	0.111 ± 0.001	0.272 ± 0.002	0.010 ± 0.005	0.024 ± 0.002
U-KNN	0.156 ± 0.001	0.148 ± 0.002	0.116 ± 0.029	0.108 ± 0.008	0.182 ± 0.001	0.397 ± 0.002	0.017 ± 0.003	0.091 ± 0.007
I-KNN	0.147 ± 0.001	0.168 ± 0.002	0.112 ± 0.010	0.155 ± 0.011	0.183 ± 0.001	0.433 ± 0.001	0.016 ± 0.002	0.148 ± 0.016
PureSVD	0.161 ± 0.001	0.170 ± 0.001	0.090 ± 0.009	DNF	0.193 ± 0.002	0.444 ± 0.002	0.010 ± 0.009	DNF
WRMF	0.168 ± 0.002	0.183 ± 0.001	0.097 ± 0.022	DNF	0.198 ± 0.002	0.459 ± 0.002	0.012 ± 0.004	DNF
LogisticMF	0.068 ± 0.002	0.085 ± 0.001	0.042 ± 0.021	DNF	0.110 ± 0.001	0.271 ± 0.001	0.010 ± 0.003	DNF
BPR	0.161 ± 0.001	0.150 ± 0.002	0.062 ± 0.025	DNF	0.193 ± 0.002	0.389 ± 0.012	0.014 ± 0.003	DNF
SLIM	0.175 ± 0.001	0.031 ± 0.003	0.109 ± 0.018	DNF	0.201 ± 0.001	0.032 ± 0.004	0.016 ± 0.002	DNF
LRec+Sq+ℓ_1+NN	0.165 ± 0.003	0.203 ± 0.001	0.125 ± 0.013	0.148 ± 0.020	0.198 ± 0.002	0.480 ± 0.001	0.018 ± 0.001	0.145 ± 0.018
LRec+Sq	0.176 ± 0.001	**0.204 ± 0.001**	**0.128 ± 0.017**	0.153 ± 0.012	0.203 ± 0.001	**0.494 ± 0.004**	**0.019 ± 0.002**	0.152 ± 0.015
LRec	**0.181 ± 0.001**	0.183 ± 0.001	0.125 ± 0.020	**0.177 ± 0.013**	**0.208 ± 0.001**	0.453 ± 0.002	0.018 ± 0.002	**0.164 ± 0.016**

Near cold-start recommendation

Table 2 summarises retrieval performance across all users. However, different users will have rated different numbers of items. It is of interest to see how LRec compares to baselines as a function of the numbers of ratings provided by a user. To do this, we segment users on the ML1M dataset based on the 1%, 25%, 50%, 75% and 99% quantiles of the number of training ratings. For each segment, we then plot the % improvement in terms of Prec@20 of LRec over WRMF.

Figure 1 shows the improvements of LRec across these segments. We see that, reassuringly, LRec is significantly superior to WRMF over all user segments. Of interest is that we see dramatic improvements in Prec@20 for LRec over WRMF for users with a small number (1 – 6) of ratings. This shows that LRec is especially useful in near cold-start recommendation scenarios.

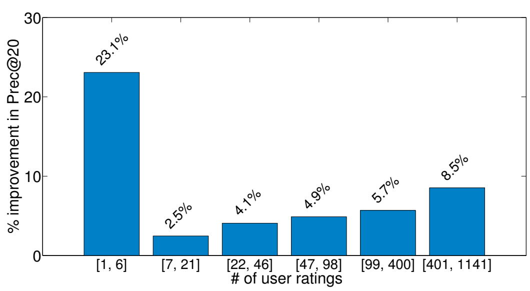

Figure 1: Improvement in Prec@20 of LRec over WRMF for varying number of training ratings, ML1M dataset.

Case-study

We now study the qualitative impact of the above performance gains. On the ML1M dataset, we compare the top 10 recommendations generated by LRec and WRMF for a user with relatively few training ratings (16). Table 3 shows a selection of this user's preferred movies on the training set, as well as the resulting recommendations. We see that the user's training preferences are towards sci-fi movies, which are picked up by both WRMF and LRec. However, we see that more specifically, the user enjoys "B-grade" Hollywood sci-fi movies, often themed on "alien invasion", with movies such as "It Came from Beneath the Sea". This is not picked up by WRMF, which recommends more "cerebral" sci-fi movies such as "2001". By contrast, LRec successfully captures all three preferred movies in the user's test set.

The above illustrates a general phenomenon: LRec can produce accurate recommendations for *non-mainstream users*, i.e. users whose preferences are for items that are not popular. As an analogue to Figure 1, Figure 2 shows the % improvement in Prec@20 scores for LRec over WRMF, where users are segmented according to the average popularity of their training set movies. Here, the popularity of a movie is simply the number of training purchases of the movie. We see that for non-mainstream users, LRec achieves significant improvement over WRMF. This illustrates that LRec captures user preferences at a finer granularity.

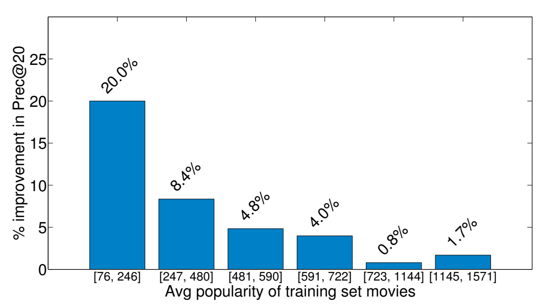

Figure 2: Improvement in Prec@20 of LRec over WRMF for users with varying average popularity of their training set movies, ML1M dataset.

other baselines were run on single machines. Hence, it is not meaningful to directly compare runtimes of LRec and other methods. But the ability of LRec to be parallelised with ease accounts for its ability to scale to datasets such as MSD.

Table 3: Subset of 16 preferred training set movies for a candidate user, and resulting top 10 recommendations on ML1M dataset. Bolded entries are actually enjoyed by user on test set, which is shown as the last column.

Preferred training movies	WRMF recommendations	LRec recommendations	Preferred test movies
• Day the Earth Stood Still, The	• Planet of the Apes	• **Them!**	• Blob, The
• Forbidden Planet	• Thing, The	• Godzilla (Gojira)	• Them!
• Kronos	• Night of the Living Dead	• **Blob, The**	• It Came from Outer Space
• Tarantula	• Star Trek: The Wrath of Khan	• 20,000 Leagues Under the Sea	
• Thing From Another World, The	• Fly, The	• Soylent Green	
• War of the Worlds, The	• Alien	• Village of the Damned	
• It Came from Beneath the Sea	• Dark City	• Metropolis	
• Invasion of the Body Snatchers	• Star Trek IV: The Voyage Home	• Quatermass and the Pit	
• Earth Vs. the Flying Saucers	• 2001: A Space Odyssey	• **It Came from Outer Space**	
• It Conquered the World	• Gattaca	• Plan 9 from Outer Space	

Conclusion

We propose LRec, a variant of SLIM for OC-CF problems based on linear classifiers, where we identify and address key limitations of SLIM. LRec has several desirable properties such as being learning-based, yielding a convex optimisation objective, producing user-personalised recommendations, and being embarrassingly parallelisable. Experiments on a range of datasets show that LRec significantly outperforms existing state-of-the-art methods for OC-CF.

An important avenue for future work is exploring techniques to further scale LRec and reduce its memory burden when dealing with a large number of users.

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The authors thank the anonymous reviewers for their valuable feedback.

References

Bertin-Mahieux, T.; Ellis, D. P.; Whitman, B.; and Lamere, P. 2011. The million song dataset. In *ISMIR' 11*.

Celma, Ò. 2008. *Music Recommendation and Discovery in the Long Tail*. Ph.D. Dissertation, Universitat Pompeu Fabra.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*.

Cremonesi, P.; Turrin, R.; and Airoldi, F. 2011. Hybrid algorithms for recommending new items. In *HetRec '11*, 33–40. New York, NY, USA: ACM.

Deshpande, M., and Karypis, G. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22.

Elkan, C., and Noto, K. 2008. Learning classifiers from only positive and unlabeled data. In *KDD '08*.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9.

Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM '08*. IEEE Computer Society.

Johnson, C. C. 2014. Logistic matrix factorization for implicit feedback data. In *NIPS'14 2014 Workshop on Distributed Machine Learning and Matrix Computations*.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42.

Levy, M., and Jack, K. 2013. Efficient top-n recommendation by linear regression. In *RecSys'13 Large Scale Recommender Systems Workshop*.

Linden, G.; Smith, B.; and York, J. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7.

Ning, X., and Karypis, G. 2011. SLIM: Sparse linear methods for top-n recommender systems. In *ICDM'11*. IEEE.

Pan, R., and Scholz, M. 2009. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *KDD '09*.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *ICDM'08*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Lars, S.-T. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Uncertainty in Artificial Intelligence (UAI)*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01*.

Sedhain, S.; Sanner, S.; Braziunas, D.; Xie, L.; and Christensen, J. 2014. Social collaborative filtering for cold-start recommendations. In *RecSys'14*.

Srebro, N., and Jaakkola, T. 2003. Weighted low-rank approximations. In *ICML*.

Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; and Han, J. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM'14*.