Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Collaborative Filtering with Localised Ranking

Charanpal Dhanjal and **Stéphan Clémençon**
Institut Mines-Télécom; Télécom ParisTech: CNRS LTCI
F-75634 Paris Cédex 13, France
charanpal.dhanjal@telecom-paristech.fr
stephan.clemencon@telecom-paristech.fr

Romaric Gaudel
LIFL: UMR University of Lille/CNRS 8022
& INRIA Lille - Nord Europe
F-59655 Villeneuve d'Ascq Cédex, France
romaric.gaudel@inria.fr

Abstract

In recommendation systems, one is interested in the ranking of the predicted items as opposed to other losses such as the mean squared error. Although a variety of ways to evaluate rankings exist in the literature, here we focus on the Area Under the ROC Curve (AUC) as it widely used and has a strong theoretical underpinning. In practical recommendation, only items at the top of the ranked list are presented to the users. With this in mind we propose a class of objective functions which primarily represent a smooth surrogate for the real AUC, and in a special case we show how to prioritise the top of the list. This loss is differentiable and is optimised through a carefully designed stochastic gradient-descent-based algorithm which scales linearly with the size of the data. We mitigate sample bias present in the data by sampling observations according to a certain power-law based distribution. In addition, we provide computation results as to the efficacy of the proposed method using synthetic and real data.

1 Introduction

A recommendation system (Adomavicius and Tuzhilin 2005; Koren, Bell, and Volinsky 2009) takes a set of items that a user has rated and recommends new items that the user may like in the future. Such systems have a broad range of applications such as recommending books (Linden, Smith, and York 2003), CDs (Linden, Smith, and York 2003), movies (Szomszor et al. 2007; Basu et al. 1998) and news (Das et al. 2007). To formalise the recommendation problem let $\{w_1, \dots, w_m\} \subseteq \mathcal{W}$ be the set of all users and let $\{y_1, \dots, y_n\} \subseteq \mathcal{Y}$ be the items that can be recommended. Each user w_i is associated to a utility function f_i^* which measures whether item y is useful to w_i. We consider the *implicit recommendation problem*, where $f_i^* : \mathcal{Y} \to \{-1, 1\}$. For example, the value of the utility function represents whether a person has read a particular research paper, or bought a product.

We are thus presented with a set of observations $\{(w_i, y, f_i^*(y)) : w_i \in \mathcal{W}, y \in \mathcal{Y}\}$ as a training sample. This training sample can also be represented as a matrix $\mathbf{X} \in \{-1, 0, 1\}^{m \times n}$ such that $\mathbf{X}_{ij} = 1$ if the ith user is interested in the jth item, and $\mathbf{X}_{ij} = -1$ if the item is irrelevant, otherwise the corresponding element is 0 to indicate a missing element. The aim of recommendation systems is to find for each user w_i a function f_i which is an approximation of f_i^* according to some loss function. This problem is challenging on real datasets since there is a large fraction of missing elements, and irrelevant items are generally unknown. In addition, the sample of observations is often drawn from a non-uniform distribution.

To address these issues, we propose a novel method to obtain a strong top ranking of items based on the theoretically well studied local Area Under the ROC Curve (local AUC) metric (Clémençon and Vayatis 2007). The method relies on matrix factorisation to represent parameters, which generally performs well and scales to large numbers of users and items. One essentially finds a low rank approximation of the unobserved entries according to a certain rank-based loss function. The resulting method has a smooth differentiable objective function and can be solved using stochastic gradient descent. We show how to perform the optimisation in parallel so it can make effective use of modern multicore CPUs. The resulting method is studied empirically in relation to other state-of-the-art matrix factorisation methods on a selection of synthetic and real datasets.

This paper is organised as follows. In the next section we motivate and derive the Maximum Local AUC (MLAUC) algorithm. Following, there is a review on related work on matrix factorisation methods including those based on top-ℓ rank-based losses. The MLAUC algorithm is then evaluated empirically in Section 4 and finally we conclude with a summary and some perspectives.

2 Maximum Local AUC

The Area Under the ROC Curve (AUC) is a common way to evaluate rankings. Consider user $w \in \mathcal{W}$ then the AUC is the expectation that a uniformly drawn positive item y is greater with respect to a uniformly drawn negative item y' under a scoring function $f : \mathcal{Y} \mapsto \mathbb{R}$

$$\text{AUC}(f) = \mathbb{E}_{\mathcal{U}}[\mathcal{I}(f(y) > f(y'))],$$

where $\mathcal{U}$ is the uniform distribution and $\mathcal{I}$ is the indicator function that is 1 when its input is true and otherwise 0. In the following text we try to maximise a quantity closely related to the AUC for all users, however there are two complications that make things more difficult. The first is that we cannot distinguish between irrelevant and unrated items.

A second challenge is that the number of ratings per item is non-uniform and hence we cannot easily compute the AUC as given above.

A common way to address the first challenge is simply to consider the missing elements as negative (Cremonesi, Koren, and Turrin 2010; Steck 2010). The second issue is more challenging, and sampling uniformly from the the data implies an expectation above with respect to some non-uniform distribution p'. In the recommendation case redefine $\mathbf{X} \in \{0,1\}^{m \times n}$ as the matrix of users and their relevant items where $+1$ signifies relevant items and 0 is an irrelevant or missing item. Consider a user w and a rating function f, then the empirical AUC for this user is:

$$\widehat{\text{AUC}}(f) = \sum_{p \in \omega} \sum_{q \in \bar{\omega}} \mathcal{I}(f(y_p) > f(y_q)) g(y_p) g'(y_q), \quad (1)$$

where ω is the set of indices of relevant items for user w, $\bar{\omega}$ is the complement of ω, and g, g' are distributions on the relevant and irrelevant items respectively. The most intuitive choices for g and g' are the uniform distributions. On real datasets however, item ratings often have a long tail distribution so that a small number of items represent large proportions of the total number of ratings. This opens up the question as to whether the so-called *popularity bias* might be an advantage to recommender systems that predict using the same distribution, however accuracy experienced by users is closely related to performance on complete data rather than available data, see (Steck 2010).

The AUC certainly captures a great deal of information about the ranking of items for each user, however as pointed out by previous authors, in practice one is only interested in the top items in that ranking. Two scoring functions f and f' could have identical AUC scores and yet f for example scores badly on the top few items but recovers lower down the list. One way of measuring this behaviour is through local AUC which measures the expectation that a positive item is scored higher than a negative one, and the positive one is in the top tth quantile. This is equivalent to saying that we ignore positive items low down in the ranking.

2.1 Surrogate Objective

Clearly the direct maximisation of AUC is not practical, since it is non-smooth and non-differentiable and hence we use the squared hinge loss to approximate the indicator function. First we define model scores for the ith user and jth item as $(\mathbf{U}\mathbf{V}^T)_{ij}$ in which $\mathbf{U} \in \mathbb{R}^{m \times k}$ and $\mathbf{V} \in \mathbb{R}^{n \times k}$ are user and item factors, and we will refer to the ith row of $\mathbf{U}$ and $\mathbf{V}$ as $\mathbf{u}_i$ and $\mathbf{v}_i$ respectively. The advantage of this strategy is that the scoring functions have $k(m+n)$ parameters and hence scale linearly with both the number of users and items. We thus propose solving the following optimisation (with penalties motivated by the data dependant bounds in (Usunier, Amini, and Gallinari 2005)):

$$\begin{aligned} \min \quad & \frac{1}{2m} \sum_{i=1}^{m} \sum_{p \in \omega_i} g(y_p) \phi \left(\sum_{q \in \bar{\omega}_i} h(\gamma_{i,p,q})^2 g'(y_q) \right) \\ & + \frac{\lambda}{2m} (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2), \end{aligned} \quad (2)$$

for a user-defined regularisation parameter λ, hinge loss $h(x) = \max(0, 1-x)$, rank weighting function ϕ, item difference $\gamma_{i,p,q} = \mathbf{u}_i^T \mathbf{v}_p - \mathbf{u}_i^T \mathbf{v}_q$ and independent item distributions g and g'. The first sum in the first term averages over users and the second sum computes ranking losses over positive items ω_i. The second term is a penalisation on the Frobenius norms ($\|\mathbf{A}\|_F^2 = \sum_{ij} \mathbf{A}_{ij}^2$) of the user and item factor matrices. Concrete item distributions g and g' are discussed later in this section.

Consider the weighting function $\phi(x) = x$, then after some rearrangement, one can see that the first term in the objective follows directly from AUC (Equation 1), replacing the indicator function with the squared hinge loss. The optimisation in this case is convex in $\mathbf{U}$ and $\mathbf{V}$ but not both simultaneously. This form of the objective however, does not take into account the preference for ranking items correctly at the top of the list. Consider instead $\phi(x) = \tanh(x)$ for $x \geq 0$, which is a concave function. The term inside ϕ in Optimisation 2 represents a ranking loss for ith user and pth item and thus y_p is high up in the ranked list if this quantity is small. The effect of choosing the hyperbolic tangent is that items with small losses have larger gradients towards the optimal solution and hence are prioritised.

2.2 Optimisation

Using this latter definition of ϕ, the gradients with respect to the objective θ are computed as

$$\begin{aligned} \frac{\delta \theta}{\delta \mathbf{u}_i} = \frac{1}{m} \sum_{p \in \omega_i} g(y_p) & \left(\sum_{q \in \bar{\omega}_i} (\mathbf{v}_q - \mathbf{v}_p) h(\gamma_{i,p,q}) g'(y_q) \right) \\ & \left(1 - \tanh^2 \left(\sum_{q \in \bar{\omega}_i} h(\gamma_{i,p,q})^2 g'(y_q) \right) \right) + \frac{\lambda}{m} \mathbf{u}_i, \end{aligned}$$

and the corresponding gradient with respect to $\mathbf{v}_j$ is

$$\begin{aligned} \frac{\delta \theta}{\delta \mathbf{v}_j} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{u}_i & \left(\mathcal{I}_{j \in \bar{\omega}_i} g'(y_j) \sum_{p \in \omega_i} g(y_p) h(\gamma_{i,p,j}) \cdot \right. \\ & \left(1 - \tanh^2 \left(\sum_{q \in \bar{\omega}_i} h(\gamma_{i,p,q})^2 g'(y_q) \right) \right) \\ & - \mathcal{I}_{j \in \omega_i} g(y_j) \sum_{q \in \bar{\omega}_i} g'(y_q) h(\gamma_{i,j,q}) \cdot \\ & \left. \left(1 - \tanh^2 \left(\sum_{\ell \in \bar{\omega}_i} h(\gamma_{i,j,\ell})^2 g'(y_\ell) \right) \right) \right) + \frac{\lambda \mathbf{v}}{m} \mathbf{v}_j. \end{aligned}$$

It is worth noting that the contribution to the derivatives of $\tanh$ lies in the $\left(1 - \tanh^2 \left(\sum_{q \in \bar{\omega}_i} h(\gamma_{i,p,q})^2 g'(y_q) \right) \right)$ terms.

The difficultly with this expression is that the derivative with respect to $\mathbf{v}_j$ requires $\mathcal{O}(mn^2)$ computations. Fortunately, the expectations for both derivatives can be estimated effectively using small samples of $s_{\mathcal{W}}$ users, and $s_{\mathcal{Y}}$ items from ω_i and $\bar{\omega}_i$. One then walks in the negative direction of these approximate derivatives using average stochastic gradient descent (average SGD, (Polyak and Juditsky 1992)).

Given initial values of $\mathbf{U}$ and $\mathbf{V}$ (using random Gaussian elements, for example) we define an iteration as $|\omega|/s_{\mathcal{Y}}$ gradient steps $\mathbf{u}_i \leftarrow \mathbf{u}_i - \alpha\frac{\delta\theta}{\delta\mathbf{u}_i}$ and $\mathbf{v}_j \leftarrow \mathbf{v}_j - \alpha\frac{\delta\theta}{\delta\mathbf{v}_j}$ using a random permutation of indices $i \in [1, m]$ and $j \in [1, n]$ and learning rate $\alpha \in \mathbb{R}^+$. Under average SGD, one maintains matrices $\mathbf{U}_\mu$ and $\mathbf{V}_\mu$ which are weighted averages of the user and item factors during the optimisation. These matrices can be shown to converge more rapidly than $\mathbf{U}$ and $\mathbf{V}$, see (Polyak and Juditsky 1992) for further details. The algorithm concludes when the maximum number of iterations T has been reached or convergence is achieved by looking at the objective value. In the sequential text we refer to this algorithm as Maximum Local AUC (MLAUC).

A disadvantage of the optimisation just described is that it cannot easily be implemented to make use of model parallel computer architectures as each intermediate solution depends on the previous one. To compensate, we build upon the Distributed SGD (DSDG) algorithm of (Gemulla et al. 2011). The algorithm works on the assumption that the loss function we are optimising can be split up into a sum of local losses, each local loss working on a subset of the elements of $\mathbf{X}_{ij}$. The algorithm proceeds by partitioning $\mathbf{X}$ into $d_1 \times d_2$ fixed blocks (sub-matrices) and each block is processed by a separate process/thread ensuring that no two concurrent blocks share the same row or column. This constraint is required so that updates to the rows of $\mathbf{U}$ and $\mathbf{V}$ are independent. The blocks do not have to be contiguous or of uniform size, and in our implementation blocks are sized so that the number of nonzero elements within each one is approximately constant. The algorithm terminates after every block has been processed at least T times.

For AUC-based losses we require a pairwise comparison of items for each row and therefore the loss cannot easily be split into local losses. To get around this issue we modify DSGD as follows: at each iteration we randomly assign rows/columns to blocks, i.e. blocks are no longer fixed throughout the algorithm. We then concurrently compute empirical estimates of the loss above within all blocks by restricting the corresponding positive and negative items, and repeat this process until convergence, or for T iterations.

2.3 Item Distributions

The distributions on the items allow some flexibility into the expectation we ultimately compute. The most simple choice, as we have already mentioned, is a uniform distribution. In practice, relevant item distributions often follow the so-called *power law* given by $p(n_y) \propto n_y^{-\alpha}$ for some exponent $\alpha \geq 1$, where n_y is the number of times relevant item y occurs in the complete data. Since we have incomplete data the generating distribution can be modelled in a similar way using (see e.g. (Steck 2011))

$$g(y) \propto \hat{p}(n_y)^\beta.$$

where $\hat{p}(n_y)$ is the empirical probability of observing y. The irrelevant and missing items form the complement of the relevant ones and hence we have

$$g'(y) \propto \hat{q}(n_y)^\beta = (1 - \hat{p}(n_y))^\beta$$

where $\beta \geq 0$ is a exponent used to control the weight of items in the objective.

3 Related Work

In this section we will briefly review some works on finding low rank matrix factorisations for implicit ratings matrices. An early attempt to deal with implicit ratings using a squared loss is presented in (Hu, Koren, and Volinsky 2008; Pan et al. 2008). The formulation is an adaptation of the Singular Value Decomposition (SVD), and minimises

$$\min \sum_{i=1}^m \sum_{j=1}^n \mathbf{C}_{ij}(\mathbf{u}_i^T\mathbf{v}_j - 1)^2 + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2),$$

where $\mathbf{C}$ is a matrix of weights for user-item pairs and λ is a regularisation parameter. In (Pan et al. 2008) the user-item values are set to 1 for positive items and lower constants for the rest. A related problem is that of *matrix completion* (Candès and Recht 2009; Mazumder, Hastie, and Tibshirani 2010) in which one minimises the Frobenius norm difference between real and predicted ratings using a trace norm penalisation (the sum of the singular values, denoted $\|\cdot\|_*$),

$$\min \frac{1}{2}\sum_{i,j\in\omega}(\mathbf{X}_{ij} - \mathbf{Z}_{ij})^2 + \lambda\|\mathbf{Z}\|_*,$$

where λ is a user-defined regularisation parameter and $\mathbf{Z}$ is the matrix factorisation. Notice that only nonzero elements of $\mathbf{X}$ are considered in the cost. A key strength of matrix completion is the strong theoretical guarantees on finding unknown entries with high accuracy. The disadvantage of both these approaches is that they do not take into account the orderings of the items.

In (Rendle et al. 2009) the authors study AUC maximisation for matrix factorisation in the context of recommendation, the primary motivation for which is a maximum posterior estimate for a Bayesian framing of the problem. The connection to AUC maximisation is made by replacing the indicator function in its computation with the log sigmoid denoted by $\ln\sigma(x) = \ln(1/(1 + e^{-x}))$. Solutions are obtained using a stochastic gradient descent algorithm based on bootstrap sampling. The particular optimisation considered takes the form

$$\max \sum_{i=1}^m \sum_{p\in\omega_i}\sum_{q\in\bar{\omega}_i} \log\sigma(\mathbf{u}_i^T\mathbf{v}_p - \mathbf{u}_i^T\mathbf{v}_q) - \frac{\lambda_\mathbf{U}}{2}\|\mathbf{U}\|_F^2$$
$$-\sum_i\left(\frac{\lambda_{\mathbf{V}_1}}{2}\sum_{p\in\omega_i}\mathbf{V}_{ip}^2 + \frac{\lambda_{\mathbf{V}_0}}{2}\sum_{q\in\bar{\omega}_i}\mathbf{V}_{iq}^2\right),$$

where $\lambda_\mathbf{U}, \lambda_{\mathbf{V}_1}, \lambda_{\mathbf{V}_0}$ are regularisation constants for the user factors, positive items and negative items respectively. The first term is a log-sigmoid unnormalised relaxation of the AUC criterion and the remaining terms are used for regularisation. Note that one optimises over the full list as opposed to prioritising the top few.

Another paper which considers the AUC is (Weston, Yee, and Weiss 2013) however it departs from other papers in using an item factor model of the form $\mathbf{Z}_{ij} = \frac{1}{|\omega_i|}\sum_{p\in\omega_i}\mathbf{v}_p^T\mathbf{v}_j$

which is the score for the ith user and jth item. The disadvantage of the item modelling approach is that it does not model users separately. Indeed, the connection between the user-item factor approach can be seen by setting $\mathbf{u}_i = \frac{1}{|\omega_i|}\sum_{p\in\omega_i}\mathbf{v}_p$. In the optimisation, the AUC is approximated by the hinge loss,

$$\min \sum_{i=1}^{m} \sum_{p\in\omega_i} \sum_{q\in\bar{\omega}_i} \max(0, 1 - \mathbf{Z}_{ip} + \mathbf{Z}_{iq}),$$

and one applies stochastic gradient descent using one user, one positive and one negative item chosen at random at each gradient step. As noted by the authors, this loss does not prioritise the top of the list and hence they propose another loss

$$\min \sum_{i=1}^{m} \sum_{p\in\omega_i} \sum_{q\in\bar{\omega}_i} \Phi\left(\frac{|\bar{\omega}_i|}{N}\right) \max(0, 1 - \mathbf{Z}_{ip} + \mathbf{Z}_{iq}),$$

where $\Phi(x) = \sum_{i=1}^{x} 1/x$ and N is a sampled approximation of $\sum_{q\in\bar{\omega}_i} \mathcal{I}(\mathbf{Z}_{ip} \leq 1 - \mathbf{Z}_{iq})$ i.e. the rank of pth item for ith user. Additionally, the algorithm samples ranks of positive items in non-uniform ways in order to prioritise the top of the list.

A related approach based on Mean Reciprocal Rank (MRR) (Shi et al. 2012) increases the importance of top k-ranked items in conjunction with maximising a lower bound on the smoothed MRR of the list. In words, the MRR is the average of the reciprocal of the rank of the first correct prediction for each user. The authors use a sigmoid function to approximate the indicator and after relaxation of a lower bound of the reciprocal rank, the following function is maximised

$$\sum_{ij} \mathbf{X}_{ij} \left(\ln \sigma(\mathbf{Z}_{ij}) + \sum_{k=1}^{n} \ln(1 - \mathbf{X}_{ik}\sigma(\mathbf{Z}_{ik} - \mathbf{Z}_{ij})) \right) - \\ - \frac{\lambda}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2),$$

where λ is a regularisation constant. The first term in the sum promotes elements in the factor model that correspond to relevant items and the second term degrades the relevance scores of the irrelevant items relative to relevant item y_j.

4 Computational Experiments

In this section we analyse various properties of MLAUC empirically as well as making comparisons between it and other user-item matrix factorisation methods including Soft Impute (Mazumder, Hastie, and Tibshirani 2010), Weighted Regularised Matrix Factorisation (WRMF, (Hu, Koren, and Volinsky 2008)) and BPR. The aim is to evaluate our ranking objective and optimisation algorithm relative to other loss functions when considering results at the very top of the list for each user.

4.1 Optimisation Strategy

First, we show that the parallel optimisation proposed in Section 2.2 does indeed give a performance advantage without impacting significantly the convergence. To do so we introduce two synthetic datasets. The first, `Synthetic1`, is generated in the following manner. Two random matrices $\mathbf{U}^* \in \mathbb{R}^{500\times 8}$ and $\mathbf{V}^* \in \mathbb{R}^{200\times 8}$ are constructed such that respectively their columns are orthogonal. We then compute the partially observed matrix $\hat{\mathbf{X}}_{ij} = \mathcal{I}(\mathbf{u}_i^T\mathbf{v}_j > Q(f_i, 1-t_i))$ in which $t_i = 0.1$ so that there are on average 20 nonzero elements per row of $\hat{\mathbf{X}}$, and additionally add an average of 5 random relevant ratings per row to form our final rating matrix $\mathbf{X}$. For the second dataset, `Synthetic2`, we generate $\mathbf{U}$ and $\mathbf{V}$ in a similar way, however this time the probability of observing a rating (relevant/irrelevant) is distributed according to the power law for each item/user with exponent 1. We iteratively sample observations according to this distribution of users and items, setting $\mathbf{X}_{ij}$ to be 1 if $\mathbf{Z}_{ij} \geq \mathbb{E}[\mathbf{Z}]$, $\mathbf{Z} = \mathbf{U}\mathbf{V}^T$, otherwise it is zero. We continue this process until the density of the matrix is at least 0.1.

The MLAUC algorithm is set up with $k = 8$ using $s_{\mathcal{U}} = 30$ row samples and $s_{\mathcal{Y}} = 10$ column samples to compute approximate derivatives for $\mathbf{u}_i$ and $\mathbf{v}_i$. The initial values of $\mathbf{U}$ and $\mathbf{V}$ are found using an SVD of the ratings matrix, then we fix learning rate $\alpha = 0.1$, regularisation $\lambda = 0.1$, maximum iterations $T = 100$, item distribution exponent $\beta = 0.5$ and positive item loss $\phi(x) = \tanh(x)$. The experiment is run on an Intel core i7-3740QM CPU with 8 cores and 16 GB of RAM and we record the objective value every 5 iterations.

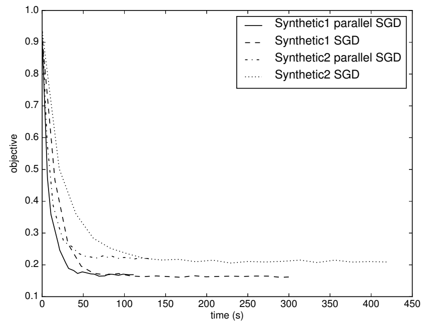

Figure 1: Plots showing the objective function on the synthetic datasets with different optimisation routines.

Figure 1 compares the timings of parallel and non-parallel variants of SGD on both datasets. We observe approximate speedups of 3 times whilst converging to approximately the same objective values for both datasets. Of course, when we look at the objective against the iteration number, the parallel SGD converges slower than the non-parallel version. To quantify the point, on `Synthetic1`, the objective falls below 0.16 after 40 iterations with parallel SGD, versus just 25 with the non-parallel variant. Overall however, parallel SGD matches the objective of SGD at a few smaller time cost and we naturally expect this improvement factor to increase with higher core CPUs.

4.2 Positive Item Loss

Next we explore the difference between $\phi_1(x) = x$ and $\phi_2(x) = \tanh(x)$ which weight the ranking losses for pos-

		p@1	p@3	p@5	r@1	r@3	r@5	AUC
Syn1	BPR	0.886	0.775	0.620	0.177	0.465	0.620	0.926
	MLAUC	0.894	0.800	0.639	0.179	0.480	0.639	0.926
	SoftImpute	0.828	0.699	0.548	0.166	0.419	0.548	0.913
	WRMF	0.898	0.756	0.609	0.180	0.454	0.609	0.923
Syn2	BPR	0.328	0.236	0.194	0.066	0.142	0.194	0.771
	MLAUC	0.445	0.299	0.241	0.089	0.179	0.241	0.794
	SoftImpute	0.201	0.184	0.165	0.040	0.111	0.165	0.754
	WRMF	0.431	0.297	0.241	0.086	0.178	0.241	0.815

Table 1: Test errors on the synthetic datasets. Top represents `Synthetic1` and bottom is `Synthetic2`.

itive items in different ways. The performance at the top of the list using `Synthetic2` as the optimisation proceeds is recorded using a validation set which is composed of 3 relevant items for each user. Let $\hat{S}_{i\ell}$ be the first ℓ predicted items for ith user, and S_i be the relevant items for the corresponding user, then we compute precision (p), recall (r) and F1 denoted by

$$r_i@\ell = \frac{|\hat{S}_{i\ell} \cap S_i|}{|S_i|}, \quad p_i@\ell = \frac{|\hat{S}_{i\ell} \cap S_i|}{\ell},$$
$$F1_i@\ell = \frac{2r_i@\ell \cdot p_i@\ell}{r_i@\ell + p_i@\ell}.$$

To compute $\hat{S}_{ik}$ and S_i, we remove the corresponding training items for each user. In this case, we consider $\ell \in \{3, 5, 10\}$ and report the average values of the above quantities across users. The setup of MLAUC is identical to above except we start with random initialisation of **U** and **V** and choose $\alpha = 0.01$, $\lambda = 0.01$. The experiment is repeated 5 times and we average the results, plotted in Figure 2.

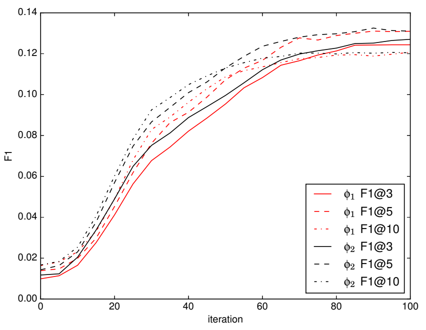

Figure 2: Plots showing the F1 scores using different weighting functions on rank losses.

The insight from the plots is that ϕ_2 does indeed improve the F1@ℓ scores for low ℓ faster than ϕ_1 due to the shape of the gradient for relevant items with good rankings. The AUC suffers however, with respective values of 0.757 and 0.742 for ϕ_1 and ϕ_2 at 100 iterations.

4.3 Comparative Ranking Performance

Here we concern ourselves with how well the different matrix factorisation methods can rank items in a top-ℓ recommendation task. From each user, 5 randomly selected relevant elements are removed and then a prediction is made for the top ℓ items. In this case, $\ell \in \{1, 3, 5\}$ and the average values of the precision, recall and AUC are recorded.

As an initial step in the learning process we perform model selection on the training nonzero elements using 3-fold cross validation with 3 relevant items taken from each user. For MLAUC the parameters are identical to the setup used above and we select learning rates from $\alpha \in \{2^{-3}, \ldots, 2^{-8}\}$ and $\lambda \in \{2^{-2}, \ldots, 2^{-6}\}$. The F1 measure is recorded on validation items in conjunction with MLAUC and used to pick the best solution. We take a sample of 3 validation items from a proportion of 0.2 of the users to form this validation set. With Soft Impute, we use regularisation parameters $\lambda \in \{1.0, 0.8, \ldots, 0\}$ and the randomised SVD to update solutions as proposed in (Dhanjal, Gaudel, and Clémençon 2014). The regularisation parameters for WRMF are chosen from $\lambda \in \{2^1, 2^{-1}, \ldots, 2^{-11}\}$. Finally, with BPR we select learning rate $\alpha \in \{2^{-3}, \ldots, 2^{-7}\}$, and regularisation parameters $\lambda_{\mathbf{U}} \in \{2^{-3}, \ldots, 2^{-8}\}$ and $\lambda_{\mathbf{v}_0} = \lambda_{\mathbf{v}_1} \in \{2^{-3}, \ldots, 2^{-8}\}$. To select the best parameters we use the maximum F1 scores on the test items averaged over all folds, fixing $k = 8$ as this is the dimension used to generate the data. Once we have found the optimal parameters we train using the training observations and test on the remaining elements to get estimates of precision, recall and AUC. The training is repeated 5 times with different random seeds and the resulting evaluation metrics are averaged.

Table 1 shows the performance of the matrix factorisation methods. On both datasets, MLAUC and WRMF perform better than BPR and in turn Soft Impute. The ingredient shared by MLAUC, BPR and WRMF is the choice of weighting on both positive and negative items summed in the objective function. We believe that this weighting corrects the bias of data toward observed relevant items, putting Soft Impute in comparison at a disadvantage. Note that MLAUC results in the best p@5 results on `Synthetic1`, and on the harder `Synthetic2` dataset has a clear advantage for p@1 despite having an AUC lower than WRMF.

Real Datasets Next we consider a set of real world datasets, MovieLens, Flixster and Mendeley coauthors. For the MovieLens and Flixster datasets, ratings are given on scales of 1 to 5 and those greater than 3 are considered relevant with the remaining ones set to zero. The Mendeley data is generated in the following manner: the raw data consists of authors and the documents stored in a matrix **Y** such

		p@1	p@3	p@5	r@1	r@3	r@5	AUC
ML	BPR	0.217	0.177	0.156	0.043	0.106	0.156	0.942
	MLAUC	0.230	0.183	0.158	0.046	0.110	0.158	0.923
	SoftImpute	0.164	0.139	0.121	0.033	0.083	0.121	0.835
	WRMF	0.199	0.163	0.144	0.040	0.098	0.144	0.869
Mend.	BPR	0.499	0.438	0.363	0.100	0.263	0.363	0.934
	MLAUC	0.828	0.799	0.714	0.166	0.480	0.714	0.924
	SoftImpute	0.828	0.790	0.697	0.166	0.474	0.697	0.932
	WRMF	0.829	0.806	0.722	0.166	0.483	0.722	0.925
Flix.	BPR	0.065	0.055	0.049	0.013	0.033	0.049	0.983
	MLAUC	0.180	0.131	0.106	0.036	0.079	0.106	0.910
	SoftImpute	0.190	0.138	0.112	0.038	0.083	0.112	0.937
	WRMF	0.197	0.146	0.119	0.039	0.087	0.119	0.929

Table 2: Test errors on the real datasets. Top represents MovieLens, middle is Mendeley and bottom is Flixster.

Dataset	users	items	nonzeros
MovieLens	897	1682	55,049
Mendeley	990	58536	998,385
Flixster	44,015	48,794	5,155,903

Table 3: Properties of the real datasets

that if ith author wrote jth document then $\mathbf{Y}_{ij} = 10$, and if it is referenced in his/her library then $\mathbf{Y}_{ij} = 1$. The rows of $\mathbf{Y}$ are normalised to have unit norm and an author-author matrix $\hat{\mathbf{X}} = \mathbf{Y}\mathbf{Y}^T$ is computed. Finally, values are thresholded such that $\mathbf{X}_{ij} = \mathcal{I}(\hat{\mathbf{X}}_{ij} > \sigma)$ where $\sigma = 0.05$ and we subsample 990 users randomly to form the dataset. For all datasets, we remove users with less than 10 items. Properties about the resulting matrices are shown in Table 3.

The experimental procedure is almost identical to before except that $k = 64$ in this case. Since the datasets are larger than the synthetic ones we perform model selection on a subsample of at most 10^5 ratings from the complete matrices. To form the model selection matrix, we pick users sequentially until the desired number of ratings is reached. The regularisation parameter for MLAUC is chosen from $\alpha \in \{2^{-3}, \ldots, 2^{-6}\}$ and $s_{\mathcal{U}} = 10$ for Flixster and Mendeley. For BPR, we limit the number of gradient steps per iteration to $|\omega|$, the number of nonzero elements.

Table 2 shows the results on these datasets. First consider the BPR results in which the AUC exceeds all other methods on the datasets. This is hardly surprising since the algorithm optimises a surrogate of the AUC criterion, and with MLAUC we use ϕ_2 to weight the loss on relevant items. On F1, BPR performs poorly for the larger datasets Mendeley and Flixster despite high AUCs, although it does exhibit a competitive performance on MovieLens. When considering MLAUC it provides the best results on MovieLens, competitive F1 scores on Mendeley and slightly worse results than the non-ranking based matrix factorisation methods on the Flixster dataset. One possible explanation for the this result is that model selection on the submatrix of the training ratings gives a poor estimation of the best hyperparameters on the full matrix. This issue does not seem to impact Soft Impute and WRMF as strongly.

5 Conclusion

Recommendation is a Learning To Rank (LTR) problem, in that for each user the system has to rank items according to the relevance for the user. Whilst early recommender systems based on matrix factorisation aimed to recover the complete matrix of ratings, recent advances focus on optimising scoring losses designed for LTR problems. The current paper pushes forward this domain by considering Local AUC maximisation, which focuses on the ranking of top items. The corresponding loss is handled with a smooth surrogate function which is minimised through stochastic gradient descent. The use of parallel architectures by a blockwise partitioning of the rating matrix in conjunction with stochastic gradient descent allows the algorithm to run on datasets with millions of known entries.

From the computational study we can draw three conclusions. First, the proposed parallelisation by block optimisation speeds up the convergence whilst keeping the quality of the objective relative to single process optimisation. Second, the effectiveness of the proposed MLAUC approach lies in the weighting of observations which gives more importance to items which are more often relevant. Third, MLAUC carefully selects the few top-ranked items, which are the one we want to recommend, and we saw the advantage of this strategy both by studying intermediate solutions and through its analysis on real and synthetic datasets.

A challenging problem highlighted by the current work is a more theoretical basis to model selection for MLAUC. It is not clear when a good model chosen on a subsample of the rating matrix will generalise well to a larger matrix, or if the approach we use is unbiased.

Acknowledgments

This work was supported by the Eurostars *ERASM* project, Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, FEDER through the *Contrat de Projets Etat Region (CPER) 2007-2013*, and *Hermes* project of the Pôle de compétitivité PICOM.

References

Adomavicius, G., and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on* 17(6):734–749.

Basu, C.; Hirsh, H.; Cohen, W.; et al. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the national conference on artificial intelligence*, 714–720. John Wiley & Sons LTD.

Candès, E. J., and Recht, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* 9(6):717–772.

Clémençon, S., and Vayatis, N. 2007. Ranking the best instances. *The Journal of Machine Learning Research* 8:2671–2699.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, 39–46. ACM.

Das, A.; Datar, M.; Garg, A.; and Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, 271–280. ACM.

Dhanjal, C.; Gaudel, R.; and Clémençon, S. 2014. Online matrix completion through nuclear norm regularisation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, 623–631.

Gemulla, R.; Nijkamp, E.; Haas, P. J.; and Sismanis, Y. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 69–77. ACM.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 263–272. IEEE.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7(1):76–80.

Mazumder, R.; Hastie, T.; and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research* 11:2287–2322.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 502–511. IEEE.

Polyak, B. T., and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization* 30(4):838–855.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 452–461. AUAI Press.

Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Oliver, N.; and Hanjalic, A. 2012. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, 139–146. ACM.

Steck, H. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 713–722. ACM.

Steck, H. 2011. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, 125–132. ACM.

Szomszor, M.; Cattuto, C.; Alani, H.; O'Hara, K.; Baldassarri, A.; Loreto, V.; and Servedio, V. 2007. Folksonomies, the semantic web, and movie recommendation. In *Workshop on Bridging the Gap between Semantic Web and Web 2.0, at 4th European Semantic Web Conference (ESWC2007).*

Usunier, N.; Amini, M.; and Gallinari, P. 2005. A data-dependent generalisation error bound for the auc. In *Proceedings of the Workshop on ROC analysis in Machine Learning (at ICML'05).*

Weston, J.; Yee, H.; and Weiss, R. J. 2013. Learning to rank recommendations with the k-order statistic loss. In *Proceedings of the 7th ACM conference on Recommender systems*, 245–248. ACM.