AUC OPTIMISATION AND COLLABORATIVE FILTERING

CHARANPAL DHANJAL AND STÉPHAN CLÉMENÇON

Institut Mines-Télécom; Télécom ParisTech, CNRS LTCI, 46 rue Barrault, 75634 Paris Cedex 13, France

ROMARIC GAUDEL

Université Lille 3, Domaine Universitaire du Pont de Bois, 59653 Villeneuve d'Ascq Cedex, France

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 over matrix factorisations 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. The objectives are differentiable and optimised through a carefully designed stochastic gradient-descent-based algorithm which scales linearly with the size of the data. In the special case of square loss we show how to improve computational complexity by leveraging previously computed measures. To understand theoretically the underlying matrix factorisation approaches we study both the consistency of the loss functions with respect to AUC, and generalisation using Rademacher theory. The resulting generalisation analysis gives strong motivation for the optimisation under study. Finally, we provide computation results as to the efficacy of the proposed method using synthetic and real data.

1. Introduction

A recommendation system [1, 16] 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 [17], CDs [17], movies [27, 3] and news [7]. To formalise the recommendation problem let $\{w_1, \ldots, w_m\} \subseteq \mathcal{W}$ be the set of all users and let $\{y_1, \ldots, y_n\} \subseteq \mathcal{Y}$ be the items that can be recommended. Each user w_i rates item y_j with a value r_{ij} which measures whether item y_j is useful to w_i. In this work, we consider the *implicit recommendation problem* in which $r_{ij} \in \mathcal{R} = \{-1, +1\}$. For example, the rating value represents

E-mail addresses: {charanpal.dhanjal, stephan.clemencon}@telecom-paristech.fr, romaric.gaudel@univ-lille3.fr.

Date: August 21, 2015.

whether a person has read a particular research paper, or bought a product. We are thus presented with a set of observations $\{(w_i, y_j, r_{ij}) : w_i \in \mathcal{W}, y_j \in \mathcal{Y}\}$ as a training sample. The aim of recommendation systems is to find a scoring function $f : \mathcal{W} \times \mathcal{Y} \mapsto \mathbb{R}$ such that the score $f(w, y)$ is high when user w strongly prefers item y. This problem is challenging for a number of reasons: we are interested only in the top few items for each user, there is often a large fraction of missing observations (irrelevant items are generally unknown) and the sample of rated items is often drawn from a non-uniform distribution.

To address these issues, we propose a general framework for obtaining strong ranking of items based on the theoretically well studied local *Area Under the ROC Curve* (AUC) metric. The framework 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 an AUC-based objective and several different surrogate loss functions. In addition we show how to focus the optimisation to the items placed at the top of the list. The resulting methods have smooth differentiable objective functions and can be solved using stochastic gradient descent. We additionally show how to perform the optimisation in parallel so it can make effective use of modern multicore CPUs. The novel algorithms are studied empirically in relation to other state-of-the-art matrix factorisation methods on a selection of synthetic and real datasets. We also answer some theoretical questions about the proposed methods. The first question is whether optimising the surrogate functions will result in improving the AUC. The second question represents a generalisation analysis of the matrix factorisation approaches using *Rademacher Theory* [2], a data-dependent approach to obtaining error bounds. The analysis sheds some light onto whether the quantities optimised on a training sample will generalise to unseen observations and provides a bound between these values. Note that a preliminary version of this paper has been presented in [9] and here we extend the work by considering a much larger class of objectives, the theoretical study and further empirical analysis.

This paper is organised as follows. In the next section we motivate and derive the Matrix Factorisation with AUC (MFAUC) framework and present its specialisation according to a variety of objective functions. In Section 3 we present the theoretical study on consistency and generalisation of the proposed approaches. Following, there is a review on related work on matrix factorisation methods including those based on top-ℓ rank-based losses. The MFAUC algorithm is then evaluated empirically in Section 5 and finally we conclude with a summary and some perspectives.

Notation: A bold uppercase letter represents a matrix, e.g. $\mathbf{X}$, and a column vector is displayed using a bold lowercase letter, e.g. $\mathbf{x}$. The transpose of a matrix or vector is written $\mathbf{X}^T$. The indicator function is given by $I(\cdot)$ so that it is 1 when its input is true otherwise it is . The all ones vector is denoted by $\mathbf{j}$ and the corresponding matrix is $\mathbf{J}$.

2. Maximising AUC

The Area Under the ROC Curve 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 negative item y' under a scoring function $s : \mathcal{Y} \mapsto \mathbb{R}$

$$\mathrm{AUC}_{\mathcal{D}}(s) = \mathbb{E}_{\mathcal{D}}[\mathcal{I}(s(y) > s(y'))], \tag{1}$$

where $\mathcal{D}$ is a distribution over items for w, $\mathcal{I}$ is the *indicator function* that is 1 when its input is true and otherwise 0, and we assume scores are never equal. One cannot in general maximise AUC directly since the indicator function is non-smooth and non-differentiable and hence difficult to optimise. Our observations consist only of positive relevance for items, thus we maximise a quantity closely related to the AUC for all users, which is practical to optimise. The missing observations are considered as negative as in [6, 25] for example.

Accuracy experienced by users is closely related to performance on complete data rather than available data [25] and thus the distribution $\mathcal{D}$ is an important consideration in a practical recommendation system. This implies that a non-uniform sampling of items might be beneficial. Consider a user w and a rating function s, then the empirical AUC for this user is:

$$\widehat{\mathrm{AUC}}(s) = \sum_{p \in \omega} \sum_{q \in \bar{\omega}} \mathcal{I}(s(y_p) > s(y_q)) g(y_p) g'(y_q), \tag{2}$$

where ω is the set of indices of relevant items for user w, $\bar{\omega}$ is the complement of ω, and g and 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 and we return to this point later.

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 s and s' could have identical AUC value and yet s 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* [5] which is 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. **A General Framework.** As previously stated, direct maximisation of AUC is not practical and hence we use a selection of surrogate functions to approximate the indicator function. Here a scoring function is found for all users and items, in particular the score for the ith user and jth item is given by $s(w_i, y_j) = (\mathbf{U}\mathbf{V}^T)_{ij}$ where $\mathbf{U} \in \mathbb{R}^{m \times k}$ is a matrix of m *user factors* and $\mathbf{V} \in \mathbb{R}^{n \times k}$ is a matrix of n *item factors*. We will refer to the ith rows of these matrices as $\mathbf{u}_i$ and $\mathbf{v}_i$ respectively. Furthermore, let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be a matrix of ratings such that $\mathbf{X}_{ij} = +1$ if user w_i finds item y_j relevant otherwise $\mathbf{X}_{ij} = 0$ if the item is missing or irrelevant. In a sense which is made clear later, $\mathbf{X}$ is an approximation of the complete structure of learner scores so that $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$. The advantage of this matrix factorisation strategy is that the scoring function has $k(m + n)$ parameters and hence scales linearly with both the number of users and items.

The framework we propose is composed principally of solving the following general optimisation:

$$\min \quad \frac{1}{m}\sum_{i=1}^{m}\sum_{p\in\omega_i} g(y_p)\phi\left(\sum_{q\in\bar{\omega}_i} L(\gamma_{i,p,q})g'(y_q)\right) + \frac{\lambda}{2}\left(\frac{1}{m}\|\mathbf{U}\|_F^2 + \frac{1}{n}\|\mathbf{V}\|_F^2\right), \tag{3}$$

for a user-defined regularisation parameter λ, loss function L, 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 distributions for relevant and irrelevant items g and g'. The relevant items for the ith user are given by the set ω_i and irrelevant/missing items are indexed in $\bar{\omega}_i$. The first sum in the first term averages over users and the second sum computes ranking losses over positive items for the ith user. 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 but a simple case is using the uniform distributions $1/|\omega_i|$ and $1/|\bar{\omega}_i|$ respectively.

The loss function L can be specialised to one of the following options ($\beta > 0$ is a user-defined parameter):

$L(x) = \frac{1}{2}\max(0, 1-x)^2$	square hinge
$L(x) = \frac{1}{2}(1-x)^2$	square
$L(x) = -1/(1+e^{-\beta x})$	sigmoid
$L(x) = -\ln(1/(1+e^{-\beta x}))$	logistic

See Figure 1 for a graphical representation of each of these loss functions. On binary classification the square hinge loss is show to provide a strong AUC on both training and test data in [29]. Furthermore, the objective becomes convex in $\mathbf{U}$ and $\mathbf{V}$ when $\phi(x) = x$. The squared loss is shown to be consistent with the AUC [11] and the squared hinge loss is shown to be consistent in [12]. In contrast, the hinge loss is not consistent with AUC. The sigmoid function is perhaps the best approximation of the negative indicator function. As $\beta \to \infty$ it approaches the indicator function, and hence we get an objective exactly corresponding to maximising AUC. The sigmoid function is used in conjunction with AUC for bipartite ranking in [14]. An noted in this paper, if β is too small then corresponding objective is a poor approximation of the AUC and if it is too large then the objective approaches the sum of step functions making it hard to perform gradient descent. To alleviate the problem the training data is used to to choose a series of increasing β values.

For the weighting function $\phi(x) = x$, one can see that the first term in the objective follows directly from AUC (Equation 2) by replacing the indicator function with the appropriate loss. The optimisation in this case is convex in the square and hinge loss cases for $\mathbf{U}$ 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 $\phi(x) = \tanh(\rho x)$ for $x \geq 0$ and fixed $\rho > 0$, which is a concave function. The term inside ϕ in Optimisation 3 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.

The distributions on the items allow some flexibility into the expectation we ultimately compute. Although the obvious choice is a uniform distribution, in practice relevant item distributions often follow the so-called *power law* given by

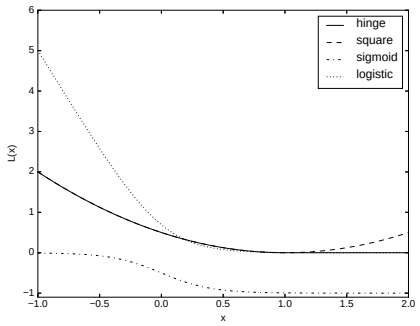


FIGURE 1. Plot of the loss functions L with $\beta = 5$.

$p(y) \propto n_y^{-\tau}$ for some exponent $\tau \geq 1$, where n_y is the number of times relevant item y occurs in the complete (fully observed) data. In words, the power law says that there are a few items which are rated very frequently whereas most items are not frequently rated, corresponding to a bias on observing a rating for popular items. However, recommendations for less well-known items are considered valuable for users. Since we have incomplete data the generating distribution can be modelled in a similar way to that of the complete data (see e.g. [26] for further motivation),

$$g(y) \propto \hat{p}(y)^{\tau'}.$$

where $\hat{p}(y)$ is the empirical probability of observing y and $\tau' \geq 0$ is a exponent used to control the weight of items in the objective. The irrelevant and missing items form the complement of the relevant ones and hence we have

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

Notice that when $\tau' = 0$ we sample from the uniform distribution. Since we expect $\hat{p}(y)$ to be related to n_y with a power law this implies that when $\tau' > 0$ we give preference to items for which n_y is small and hence focus on less popular items.

2.2. **Optimisation Algorithms.** To solve the above objectives one can use gradient descent methods which rely on computing the derivatives with respect to the parameters $\mathbf{U}$ and $\mathbf{V}$. Here we present the derivatives for choices of loss L and weighting function ϕ, starting with the squared hinge loss and $\phi(x) = x$. Denote the objective function of Optimisation 3 as θ then the derivatives are,

$$\frac{\delta\theta}{\delta\mathbf{u}_i} = \frac{1}{m}\sum_{p\in\omega_i} g(y_p)\left(\sum_{q\in\bar{\omega}_i}(\mathbf{v}_q - \mathbf{v}_p)h(\gamma_{i,p,q})g'(y_q)\right) + \frac{\lambda}{m}\mathbf{u}_i, \tag{4}$$

and

$$\begin{aligned}\frac{\delta\theta}{\delta\mathbf{v}_j} =& \frac{1}{m}\sum_{i=1}^{m}\mathbf{u}_i\left(\mathcal{I}_{j\in\bar{\omega}_i}g'(y_j)\sum_{p\in\omega_i} g(y_p)h(\gamma_{i,p,j})\right.\\ &\left.-\mathcal{I}_{j\in\omega_i}g(y_j)\sum_{q\in\bar{\omega}_i} g'(y_q)h(\gamma_{i,j,q})\right) + \frac{\lambda}{n}\mathbf{v}_j,\end{aligned} \tag{5}$$

where $h(x) = \max(0, 1-x)$ and for convenience we use the notation $\mathcal{I}_{j\in\bar{\omega}_i} = \mathcal{I}(j \in \bar{\omega}_i)$. The squared loss is identical except that $h(x) = (1-x)$ in this case. It is worth noting that the term inside the outer sum of this derivative in the squared loss case can be written as:

$$\begin{aligned}\frac{\delta\theta}{\delta\mathbf{v}_j} =& \frac{1}{m}\sum_{i=1}^{m}\mathbf{u}_i\left(\mathcal{I}_{j\in\bar{\omega}_i}g'(y_j)(1+\mathbf{u}_i^T(\mathbf{v}_j-\dot{\mathbf{v}}_i))\right.\\ &\left.-\mathcal{I}_{j\in\omega_i}g(y_j)(1+\mathbf{u}_i^T(\ddot{\mathbf{v}}_i-\mathbf{v}_j))\right)+\frac{\lambda}{n}\mathbf{v}_j\end{aligned} \tag{6}$$

where $\dot{\mathbf{v}}_i = \sum_{p\in\omega_i} g(y_p)\mathbf{v}_p$ and $\ddot{\mathbf{v}}_i = \sum_{q\in\bar{\omega}_i} g'(y_q)\mathbf{v}_q$ are empirical expectations. The derivative with respect to $\mathbf{u}_i$ can be treated in a similar way:

$$\frac{\delta\theta}{\delta\mathbf{u}_i} = \frac{1}{m}(\ddot{\mathbf{v}}_i - \dot{\mathbf{v}}_i + \ddot{\mathbf{w}}_i - \dot{\mathbf{v}}_i\ddot{\mathbf{v}}_i^T\mathbf{u}_i - \ddot{\mathbf{v}}_i\dot{\mathbf{v}}_i^T\mathbf{u}_i + \dot{\mathbf{w}}_i) + \frac{\lambda}{m}\mathbf{u}_i, \tag{7}$$

where $\dot{\mathbf{w}}_i = \sum_{p\in\omega_i} g(y_p)\mathbf{v}_p\mathbf{v}_p^T\mathbf{u}_i$ and $\ddot{\mathbf{w}}_i = \sum_{q\in\bar{\omega}_i} g'(y_q)\mathbf{v}_q\mathbf{v}_q^T\mathbf{u}_i$. Thus we have inexpensive ways to compute derivatives in the square loss case provided we have access to the expectations of particular vectors.

The logistic and sigmoid losses are similar functions and their derivatives are computed as follows (β is a user-defined parameter and $\phi(x) = x$ as before):

$$\frac{\delta\theta}{\delta\mathbf{u}_i} = -\frac{\beta}{m}\sum_{p\in\omega_i} g(y_p)\left(\sum_{q\in\bar{\omega}_i}(\mathbf{v}_q-\mathbf{v}_p)h(\gamma_{i,p,q})g'(y_q)\right)+\frac{\lambda}{n}\mathbf{u}_i, \tag{8}$$

and

$$\begin{aligned}\frac{\delta\theta}{\delta\mathbf{v}_j} =& -\frac{\beta}{m}\sum_{i=1}^{m}\mathbf{u}_i\left(\mathcal{I}_{j\in\bar{\omega}_i}g'(y_j)\sum_{p\in\omega_i}g(y_p)h(\gamma_{i,p,j})\right.\\ &\left.-\mathcal{I}_{j\in\omega_i}g(y_j)\sum_{q\in\bar{\omega}_i}g'(y_q)h(\gamma_{i,j,q})\right)+\frac{\lambda}{n}\mathbf{v}_j,\end{aligned} \tag{9}$$

in which $h(x) = \frac{e^{-\beta x}}{1+e^{-\beta x}}$ for the logistic loss and $h(x) = \frac{1}{(1+e^{-\beta x})^2}$ for the sigmoid loss.

We now consider the case in which we have the weighting function $\phi(x) = \tanh(\rho x)$ on the item losses in conjunction with a square or squared hinge loss. The gradients with respect to the objective θ are

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

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

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

with $h(x) = \max(0, 1 - x)$ in the square hinge loss case and $h(x) = (1 - x)$ for the square loss. When we compare the above derivatives to Equations 4 and 5 for the square/hinge loss functions we can see that there is an additional weighting on the relevant items given by $f(\mathbf{u}_i, \mathbf{v}_p) = 1 - \tanh^2\left(\frac{\rho}{2}\sum_{q\in\bar{\omega}_i} h(\mathbf{u}_i^T\mathbf{v}_p - \mathbf{u}_i^T\mathbf{v}_q)^2 g'(y_q)\right)$ which will naturally increase the size of gradient for items with small losses and hence those high up in the ranking.

Thus we have presented the derivatives for each of the loss functions we proposed earlier in this section as well the hyperbolic tangent weighting scheme. The derivative with respect to $\mathbf{v}_j$ is generally the most costly to find and we can see that the computational complexity of computing it is $\mathcal{O}(mn)$ and hence the derivative with respect to the complete matrix $\mathbf{V}$ is $\mathcal{O}(mn^2)$. The complexity of the derivative with respect to $\mathbf{U}$ is identical although in practice it is quicker to compute. Fortunately, the expectations for both derivatives can be estimated effectively using $\kappa_{\mathcal{W}}$ users, and $\kappa_{\mathcal{Y}}$ items from ω_i and $\bar{\omega}_i$. This reduces the complexity per iteration of computing the derivative with respect to $\mathbf{V}$ to $\mathcal{O}(\kappa_{\mathcal{W}}\kappa_{\mathcal{Y}}^2)$.

Using these approximate derivatives we can apply stochastic gradient descent to solve Optimisation 3. Define $\theta'(\mathbf{U}, \mathbf{V})$ as the approximate subsampled objective, then one then walks in the negative direction of the approximate derivatives using *average stochastic gradient descent* (average SGD, [22]). Algorithm 1 shows the pseudo code for solving the optimisation. Given initial values of $\mathbf{U}$ and $\mathbf{V}$ (using random Gaussian elements, for example) we define an iteration as $\max(m, n)$ gradient steps using a random permutation of indices $i \in [1, m]$ and $j \in [1, n]$. It follows that each iteration updates all of the rows of $\mathbf{U}$ and $\mathbf{V}$ at least once according to the learning rate $\alpha \in \mathbb{R}^+$ and the corresponding derivative. Note in line 6 that if t exceeds the size of the corresponding vector we wrap around to the start. Under average SGD one maintains, during the optimisation, matrices $\bar{\mathbf{U}}$ and $\bar{\mathbf{V}}$ which are weighted averages of the user and item factors. These matrices can be shown to converge more rapidly than $\mathbf{U}$ and $\mathbf{V}$, see [22] for further details. The algorithm concludes when the maximum number of iterations T has been reached or convergence is achieved by looking at the average objective value for the the most recent iterations. In the following text we refer to this algorithm as Matrix Factorisation with AUC (MFAUC).

2.2.1. *Parallel Algorithm.* 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 [13]. 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, see Figure 2. 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.

Algorithm 1 Pseudo code for Matrix Factorisation with AUC

Require: Ratings $\mathbf{X}$, solutions $\mathbf{U}$, $\mathbf{V}$, iterations T, average start T_0, learning rate α, convergence threshold ϵ

$\bar{\mathbf{U}} = \mathbf{U}$ and $\bar{\mathbf{V}} = \mathbf{V}$
Vector $\mathbf{z}$ is permutation of $\{1, \ldots, m\}$ and $\mathbf{z}'$ is permutation of $\{1, \ldots, n\}$
$\theta'_0 = \theta'(\mathbf{U}, \mathbf{V})$, $\theta'_1 = \theta'_0 + \epsilon$, $s = 1$
while $s \neq T$ and $|\theta'_s - \theta'_{s-1}| \geq \epsilon$ **do**
  for $t = 1, \ldots, \max(m, n)$ **do**
    Let $i = \mathbf{z}_t$ and $j = \mathbf{z}'_t$
    $\mathbf{u}_i \leftarrow \mathbf{u}_i - \alpha \frac{\delta \theta'}{\delta \mathbf{u}_i}$ and $\mathbf{v}_j \leftarrow \mathbf{v}_j - \alpha \frac{\delta \theta'}{\delta \mathbf{v}_j}$
    if $T \geq T_0$ **then**
      $\bar{\mathbf{u}}_i \leftarrow \frac{i}{i+1} \bar{\mathbf{u}}_i + \frac{1}{i+1} \mathbf{u}_i$ and $\bar{\mathbf{v}}_j \leftarrow \frac{i}{i+1} \bar{\mathbf{v}}_j + \frac{1}{i+1} \mathbf{v}_j$
    else
      $\bar{\mathbf{u}}_i = \mathbf{u}_i$ and $\bar{\mathbf{v}}_j = \mathbf{v}_j$
    end if
  end for
  Update $\theta'_s = \theta'(\bar{\mathbf{U}}, \bar{\mathbf{V}})$, $s = s + 1$
end while
return Solutions $\bar{\mathbf{U}}$, $\bar{\mathbf{V}}$

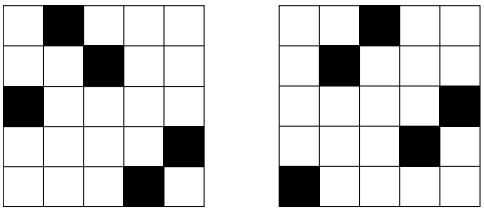


FIGURE 2. Illustration of the principal behind DSGD. Each of the two large squares represents a matrix divided into blocks, and each black square represents a process working on a block.

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.

3. Consistency and Generalisation

In this section we will discuss issues relating to the consistency of the above optimisation, as well as the generalisation performance on unseen data. First we address the question of whether optimising a surrogate of the AUC will lead to improving the AUC itself. We draw upon the work of [12] to show that our chosen loss functions are consistent. In the bipartite ranking case the square hinge loss is shown to be consistent in [12] and a similar result for the square loss is proven in [11].

Therefore we concentrate on the sigmoid and logistic functions and additionally show how the consistency results apply to the matrix factorisation scenario we consider in this paper.

To begin we define consistency in a more formal sense, starting with a more general definition of the AUC than Equation 1. In this case, the scoring function s can result in the same scores for different items,

$$\mathrm{AUC}_{\mathcal{D}}(s) = \mathbb{E}_{\mathcal{D}}[\mathcal{I}((r-r')(s(y)-s(y')) > 0) + \frac{1}{2}\mathcal{I}(s(y) = s(y'))|r \neq r')],$$

where $r, r' \in \{-1, +1\}$ are the labels respectively of items y, y'. We can alternatively state the AUC in terms of a risk so that maximising AUC corresponds to minimising this quantity:

$$R(s) = \mathbb{E}_{(y,r),(y',r')\sim\mathcal{D}}[\ell(s,(y,r),(y',r'))|r \neq r']$$

in which $\ell(s,(y,r),(y',r')) = \mathcal{I}((r-r')(s(y)-s(y')) < 0) + \frac{1}{2}\mathcal{I}(s(y) = s(y'))$. If we define $\eta(y) = P[r = +1|y]$ then we see that the risk can be expressed as

$$R(s) \propto \mathbb{E}_{(y,y')\sim\mathcal{D}^2_{\mathcal{Y}}}[\eta(y)(1-\eta(y'))\ell'(s,y,y') + \eta(y')(1-\eta(y))\ell'(s,y',y)],$$

where $\ell'(s,y,y') = \mathcal{I}(s(y)-s(y') < 0) + \frac{1}{2}\mathcal{I}(s(y) = s(y'))$ and $\mathcal{D}_{\mathcal{Y}}$ is the marginal distribution over $\mathcal{Y}$. Notice that if we assume that $\eta(y) > \eta(y')$ then we would prefer a function s such that $s(y) > s(y')$, and a similar results holds if we swap y and y'. This allows us to introduce the *Bayes risk* $R(s^*)$ where s^* is defined as follows:

$$\begin{aligned} s^* &= \mathrm{arginf}_s R(s) \\ &= \{s : (s(y)-s(y'))(\eta(y)-\eta(y')) > 0 \text{ if } \eta(y) \neq \eta(y')\}, \end{aligned}$$

where s is chosen from all measurable functions. Since we replace the indicator with a surrogate loss function, define $L'(s,y,y') = L(s(y)-s(y'))$ then we can write the corresponding risk as,

$$R_L(s) \propto \mathbb{E}_{(y,y')\sim\mathcal{D}^2_{\mathcal{Y}}}[\eta(y)(1-\eta(y'))L'(s,y,y') + \eta(y')(1-\eta(y))L'(s,y',y)],$$

and the optimal scoring function with respect to this risk is $s^*_L = \mathrm{arginf}_s R_L(s)$. With these definitions, we can define consistency.

Definition 3.1 (Consistency, [12])**.** *The surrogate loss L is said to be consistent with AUC if for every sequence $\{s^{\langle i\rangle}(y)\}_{i\geq 1}$, the following holds over all distributions $\mathcal{D}$ on $\mathcal{Y}\times\mathcal{R}$:*

$$R_L(s^{\langle i\rangle}) \to R_L(s^*_L) \text{ then } R(s^{\langle i\rangle}) \to R(s^*).$$

Thus, a consistent loss function will lead to an optimal Bayes risk in the limit of an infinite sample size. Furthermore, a sufficient condition is given for AUC as follows

Theorem 3.1 (Sufficiency for AUC consistency, [12])**.** *The surrogate loss $L'(s,y,y') = L(s(y)-s(y'))$ is consistent with AUC if $L : \mathbb{R} \to \mathbb{R}$ is a convex, differentiable and non-increasing function with $\Delta L(0) < 0$.*

These theorems allow us to prove that the sigmoid and logistic losses are consistent with AUC.

Theorem 3.2. *Both the sigmoid $-\sigma(x)$, $\sigma(x) = 1/(1+e^{-\beta x})$, and logistic loss $-\ln(\sigma(x)) = \ln(1/(1+e^{-\beta x}))$ functions are consistent with AUC.*

Proof. We will start with the logistic function. Assume that $\beta > 0$ and x is finite,

$$\frac{\delta - \ln(\sigma)}{\delta x} = -\frac{-\beta e^{-\beta x}}{(1 + e^{-\beta x})} < 0,$$

which implies a non-increasing function, in particular the derivative at zero is $-\beta/2$. In addition the logistic loss is convex since for all finite x,

$$\frac{\delta^2 - \ln(\sigma)}{\delta x^2} = \frac{\beta^2 e^{-\beta x}}{(1 + e^{-\beta x})}\left(1 - \frac{e^{-\beta x}}{(1 + e^{-\beta x})}\right) > 0,$$

where the first term in the derivative is greater than zero and the second term in parenthesis is between 0 and 1. An application of Theorem 3.1 gives the required result.

For the sigmoid function, consider a set of items $S = \{y_1, y_2, \ldots, y_n\}$ with corresponding conditional probabilities on being positive $\eta(y_1), \eta(y_2), \ldots, \eta(y_m)$ then we can find the following risk (we use notation $\eta_i = \eta(y_i)$ and $s_i = s(y_i)$ for convenience)

$$R_L(s) \propto R'_L(s) = -\sum_{i<j}\left(\eta_i(1-\eta_j)\frac{1}{1+e^{-(s_i - s_j)}} + \eta_j(1-\eta_i)\frac{1}{1+e^{-(s_j - s_i)}}\right).$$

For each pair of terms in this sum notice first that $1/(1 + e^{-x}) + 1/(1 + e^{x}) = 1$ for all x. It follows that if $\eta_i > \eta_j$ then the first term should be maximised by increasing $s_i - s_j$ otherwise the second one should be maximised to minimise the risk. Therefore, in the limiting case

$$R'_L(s) \to -\sum_{\eta_i > \eta_j} \eta_i(1 - \eta_j),$$

and we have $(s_i - s_j)(\eta_i > \eta_j) > 0$ when $\eta_i \neq \eta_j$ as in the Bayes risk. □

As we have not made any assumption on scoring functions in our previous results, the AUC is simply generalised to the expectation over users as follows

$$\mathrm{AUC}_{\mathcal{D}}(s) = \mathbb{E}_{\mathcal{W}\times\mathcal{Y}}[\mathcal{I}((r - r')(s(w, y) - s(w, y')) > 0) + \frac{1}{2}\mathcal{I}(s(y) = s(y'))|r \neq r')],$$

where $\mathcal{D}$ is now is a distribution over users and items and we redefine the scoring function as $s : \mathcal{W} \times \mathcal{Y} \mapsto \mathbb{R}$. Thus we have shown that all the loss functions we consider are consistent with the AUC in the multi-user scenario.

3.1. **Generalisation Bound.** We now turn to another critical question about our algorithm relating to the generalisation. In particular we look at Rademacher theory, which is a data-dependent way of studying generalisation performance. We assume that the observations are sampled from a distribution $\bar{\mathcal{D}}$ over $\mathcal{W} \times \mathcal{Y} \times \mathcal{R}$. Recall that only positive ratings are observed, hence the sample $S = \cup_{i=1}^{m}\{(w_i, y_{\omega_{i1}}), \ldots, (w_i, y_{\omega_{in_i}})\}$ is drawn from the distribution $\mathcal{D} = \mathcal{D}_1^{n_1} \times \cdots \times \mathcal{D}_m^{n_m}$ where we use the shorthand $n_i = |\omega_i|$. Now consider a class of functions $\mathcal{Q}$ mapping from $\mathcal{W} \times \mathcal{Y}^2$ to $[0, 1]$ then the AUC can be maximised by minimising

$$\hat{\mathbb{E}}_S[Q] = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i n'_i}\sum_{p\in\omega_i}\sum_{q\in\bar{\omega}_i} Q(w_i, y_p, y_q), \tag{10}$$

in the case that $\mathcal{Q} = \{Q : (w, y, y') \mapsto \mathcal{I}(s(w, y) \leq s(w, y')), s \in \mathcal{S}\}$. Likewise we can substitute any class of loss functions $\mathcal{Q}$ to be minimised, for example the surrogate functions defined above. Noting this, we can now present the idea of a

Rademacher variables which are uniformly distributed $\{-1,+1\}$ independent random variables. The following definition is an empirical Rademacher average for all users

$$\hat{R}^{AUC}(\mathcal{Q}) = 2\mathbb{E}_\nu \left[\sup_{Q\in\mathcal{Q}} \frac{1}{m} \sum_{i=1}^{m} \frac{\nu_i}{n_i n_i'} \sum_{p\in\omega_i} \sum_{q\in\bar{\omega}_i} Q(w_i, y_p, y_q) \right],$$

where the ν_i's are independent Rademacher random variables (note the close connection to Equation 10). This expectation is an empirical measure of the capacity of a function class because it measures how well functions in that class can correlate with random data. We define the Rademacher complexity of $\mathcal{Q}$ as

$$R^{AUC}(\mathcal{Q}) = \mathbb{E}_S[\hat{R}^{AUC}(\mathcal{Q})],$$

where S is drawn over $\mathcal{D}_1^{n_1} \times \cdots \times \mathcal{D}_m^{n_m}$. In the following lemma we provide a relationship between these two quantities using McDiarmid's Theorem.

Theorem 3.3 (McDiarmid, [19])**.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in a set A and assume $f : A^n \to \mathbb{R}$ satisfies*

$$\sup_{x_1,\ldots,x_n,\hat{x}_i\in A} |f(x_1,\ldots,x_n) - f(x_1,\ldots,\hat{x_i},x_{i+1},\ldots,x_n)| \leq c_i,$$

for all $i = 1,\ldots,n$. Then for all $\epsilon > 0$,

$$P\{f(X_1,\ldots,X_n) - \mathbb{E}[f(X_1,\ldots,X_n)] \geq \epsilon\} \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right),$$

and

$$P\{\mathbb{E}[f(X_1,\ldots,X_n)] - f(X_1,\ldots,X_n) \geq \epsilon\} \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).$$

We show that the Rademacher complexities are related as follows

Theorem 3.4. *Let $\mathcal{Q}$ be a function class mapping from $\mathcal{W}\times\mathcal{Y}^2$ to $[0,1]$ and define sample $S = \cup_{i=1}^m \{(w_i, y_{\omega_{i1}}),\ldots,(w_i,y_{\omega_{in_i}})\}$ drawn from distribution $\mathcal{D}_1^{n_1}\times\cdots\times\mathcal{D}_m^{n_m}$. Then with probability at least $1-\delta$ the following is true*

$$R^{AUC}(\mathcal{Q}) \leq \hat{R}^{AUC}(\mathcal{Q}) + \sqrt{\frac{2\ln(1/\delta)(n-1)^2}{m^2} \sum_{i=1}^{m} \frac{1}{|\omega_i||\bar{\omega}_i|^2}}.$$

Proof. As well as S consider another sample $\hat{S}_{ab} = S \setminus \{(w_a, y_b)\} \cup \{(w_a, y_{b'})\}$ for $a \in [1,m]$ and $b, b' \in [1,n]$. For notational convenience we call the nonzero elements of $\hat{S}_{ab}$, $\omega_i' = \omega_i$, $i \neq a$ and $\omega_a' = \omega_i \setminus \{y_b\} \cup \{y_{b'}\}$. Likewise for the zero elements $\bar{\omega}_i' = \bar{\omega}_i$, $i \neq a$ and $\bar{\omega}_a' = \bar{\omega}_i \setminus \{y_{b'}\} \cup \{y_b\}$. Define

$$f(S) = 2\mathbb{E}_\nu \left[\sup_{Q\in\mathcal{Q}} \frac{1}{m} \sum_{i=1}^{m} \frac{\nu_i}{|\omega_i||\bar{\omega}_i|} \sum_{p=1}^{|\omega_i|} \sum_{q=1}^{|\bar{\omega}_i|} Q(w_i, y_{\omega_{ip}}, y_{\bar{\omega}_{iq}}) \right].$$

Therefore we have that $(m/2)\cdot(f(S)-f(\hat{S}_{ab}))$ is equal to (the notation ω_{ij} denotes the jth element in ω_i)

$$
\begin{aligned}
&= \mathbb{E}_\nu\left[\sup_{Q\in\mathcal{Q}}\sum_{i=1}^{m}\frac{\nu_i}{|\omega_i||\bar{\omega}_i|}\sum_{p=1}^{|\omega_i|}\sum_{q=1}^{|\bar{\omega}_i|}Q(w_i,y_{\omega_{ip}},y_{\bar{\omega}_{iq}})-\sup_{Q\in\mathcal{Q}}\sum_{i=1}^{m}\frac{\nu_i}{|\omega'_i||\bar{\omega}'_i|}\sum_{p=1}^{|\omega'_i|}\sum_{q=1}^{|\bar{\omega}'_i|}Q(w_i,y_{\omega'_{ip}},y_{\bar{\omega}'_{iq}})\right]\\
&\leq \mathbb{E}_\nu\left[\sup_{Q\in\mathcal{Q}}\left(\sum_{i=1}^{m}\frac{\nu_i}{|\omega_i||\bar{\omega}_i|}\sum_{p=1}^{|\omega_i|}\sum_{q=1}^{|\bar{\omega}_i|}Q(w_i,y_{\omega_{ip}},y_{\bar{\omega}_{iq}})-\sum_{i=1}^{m}\frac{\nu_i}{|\omega'_i||\bar{\omega}'_i|}\sum_{p=1}^{|\omega'_i|}\sum_{q=1}^{|\bar{\omega}'_i|}Q(w_i,y_{\omega'_{ip}},y_{\bar{\omega}'_{iq}})\right)\right]\\
&= \mathbb{E}_\nu\left[\sup_{Q\in\mathcal{Q}}\left(\frac{\nu_a}{|\omega_a||\bar{\omega}_a|}\sum_{p=1}^{|\omega_a|}\sum_{q=1}^{|\bar{\omega}_a|}Q(w_a,y_{\omega_{ap}},y_{\bar{\omega}_{aq}})-\frac{\nu_a}{|\omega'_a||\bar{\omega}'_a|}\sum_{p=1}^{|\omega'_a|}\sum_{q=1}^{|\bar{\omega}'_a|}Q(w_a,y_{\omega'_{ap}},y_{\bar{\omega}'_{aq}})\right)\right]\\
&= \mathbb{E}_\nu\left[\sup_{Q\in\mathcal{Q}}\left(\frac{\nu_a}{|\omega_a||\bar{\omega}_a|}\left(\sum_{y_q\in\bar{\omega}_a\setminus\{y_{b'}\}}(Q(w_a,y_b,y_q)-Q(w_a,y_{b'},y_q))\right.\right.\right.\\
&\qquad\left.\left.\left.+\sum_{y_p\in\omega_a\setminus\{y_b\}}(Q(w_a,y_p,y_{b'})-Q(w_a,y_p,y_b))+Q(w_a,y_b,y_{b'})-Q(w_a,y_{b'},y_b)\right)\right)\right]\\
&\leq \frac{|\omega_a|+|\bar{\omega}_a|-1}{|\omega_a||\bar{\omega}_a|}\\
&= \frac{n-1}{|\omega_a||\bar{\omega}_a|}.
\end{aligned}
$$

A similar derivation can be shown to bound $m(f(\hat{S}_{ab})-f(S))$ using an identical term. If we let $c_{ab}=\frac{2(n-1)}{m|\omega_a||\bar{\omega}_a|}$ then we can apply Theorem 3.3 and write $\mathbb{E}[f(S)]\leq f(S)+\epsilon$ with a probability greater than $1-\delta$ for some $\delta\in[0,1]$ where

$$\delta=\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m}\sum_{j=1}^{n_i}c_{ij}^2}\right),$$

and rearranging gives $\epsilon=\sqrt{\frac{2\ln(1/\delta)(n-1)^2}{m^2}\sum_{i=1}^{m}\frac{1}{|\omega_i||\bar{\omega}_i|^2}}$ as required. □

To gain some insight into Theorem 3.4 we study the cases for which the empirical Rademacher complexity is close to its expectation. In the first setting, $|\omega_i|$ is fixed and non-zero. Therefore the second term of the theorem is $\sqrt{\frac{2\ln(1/\delta)(n-1)^2}{m|\omega_i|(n-|\omega_i|)^2}}$, which tends to zero as soon as the number of users m tends to infinity. Similarly, if $|\omega_i|=\Theta(n)$ when n tends to infinity, the second term of the theorem is $\Theta(1/\sqrt{mn})$ when n and m tends to infinity. Finally, when $|\bar{\omega}_i|$ is fixed and non-zero, the second term grows as $\Theta(\sqrt{\frac{n}{m}})$ when n and m tend to infinity. In the last setting we require m to grow faster than n so that n/m tends to zero to enforce the empirical Rademacher complexity to be close to its expectation.

We can now turn to concrete definitions of the function class $\mathcal{Q}$ relating to the matrix factorisation scenario we are interested in. Consider a specific form of the scoring function of the form $\mathcal{Q}_h=\{Q:(w_i,y_p,y_q)\mapsto h(\mathbf{u}_i^T\mathbf{v}_p-\mathbf{u}_i^T\mathbf{v}_q)\mid\|\mathbf{U}\|_F\leq R,\|\mathbf{V}\|_F\leq R\}$ which allows us to analyse more deeply the empirical Rademacher complexity. Our goal is to bound the empirical Rademacher complexity based on the observations S. Before presenting a result we introduce two useful theorems.

Theorem 3.5 ([20]). *Let $\phi_1, \ldots, \phi_n$ be functions with respective Lipschitz constants $\gamma_1, \ldots, \gamma_n$, then the following bound holds:*

$$\mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \sigma_i \phi_i(f(x_i)) \right] \leq \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \sigma_i \gamma_i f(x_i) \right],$$

where $\mathcal{F}$ is a function class on $x \in \mathcal{X}$ and $\sigma_1, \ldots, \sigma_n$ are independent Rademacher variables.

Theorem 3.6 ([10]). *Let $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ be a matrix with singular values $\sigma_1, \ldots, \sigma_r$ where r is the rank of $\boldsymbol{A}$, and $\tilde{\boldsymbol{B}} = \boldsymbol{D}_L \boldsymbol{B} \boldsymbol{D}_R$ be a multiplicative perturbation in which $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ are nonsingular matrices. The the following holds on the singular values $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_r$ of $\tilde{\boldsymbol{B}}$:*

$$\frac{\sigma_i}{\|\boldsymbol{D}_L^{-1}\|_2 \|\boldsymbol{D}_R^{-1}\|_2} \leq \tilde{\sigma}_i \leq \sigma_i \|\boldsymbol{D}_L\|_2 \|\boldsymbol{D}_R\|_2,$$

where $\|\boldsymbol{A}\|_2 = \sqrt{\lambda_{max}(\boldsymbol{A}^T \boldsymbol{A})} = \sigma_{max}(\boldsymbol{A})$ is the spectral norm.

We also introduce the following lemma whose utility will become apparently in the sequential theorem.

Lemma 3.1. *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, then for a fixed integer k the solutions $\boldsymbol{U} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ of,*

$$\begin{array}{ll} \sup & tr(\boldsymbol{U}^T \boldsymbol{A} \boldsymbol{V}) \\ s.t. & \|\boldsymbol{U}\|_F = 1 \\ & \|\boldsymbol{V}\|_F = 1, \end{array}$$

are given by $\boldsymbol{U} = \frac{1}{\sqrt{k}}[\boldsymbol{q}_1 \cdots \boldsymbol{q}_1]$ and $\boldsymbol{V} = \frac{1}{\sqrt{k}}[\boldsymbol{p}_1 \cdots \boldsymbol{p}_1]$ where $\boldsymbol{p}_1$ and $\boldsymbol{q}_1$ are the largest left and right singular vectors of $\boldsymbol{A}$ respectively. The corresponding value of the objective is σ_1, the largest singular value of $\boldsymbol{A}$.

Proof. We introduce $\phi(\mathbf{U}, \mathbf{V}) = \mathrm{tr}(\mathbf{U}^T \mathbf{A} \mathbf{V}) - \frac{1}{2}\lambda_1(\mathrm{tr}(\mathbf{U}^T \mathbf{U}) - 1) - \frac{1}{2}\lambda_2(\mathrm{tr}(\mathbf{V}^T \mathbf{V}) - 1)$ where λ_1 and λ_2 are Lagrange multipliers. Taking derivatives, one obtains the following equations:

$$\mathbf{A}\mathbf{V} = \lambda_1 \mathbf{U} \tag{11}$$

$$\mathbf{A}^T \mathbf{U} = \lambda_2 \mathbf{V}, \tag{12}$$

where λ_1 and λ_2 are Lagrange multipliers. By premultiplying the first equality by $\mathbf{U}^T$ and the second by $\mathbf{V}$ then taking the trace one can see that $\lambda_1 = \lambda_2 = \lambda$ is the objective value. By substitution of $\mathbf{U} = \lambda^{-1}\mathbf{A}\mathbf{V}$ into the second equality we obtain $\mathbf{A}^T \mathbf{A} \mathbf{V} = \lambda^2 \mathbf{V}$ which implies that columns of $\mathbf{V}$ must be composed of a particular eigenvector of $\mathbf{A}^T \mathbf{A}$. In a similar manner, one obtains $\mathbf{A}\mathbf{A}^T \mathbf{U} = \lambda^2 \mathbf{U}$. Denote the left and right singular values of $\mathbf{A}$ as $\mathbf{p}_1, \ldots, \mathbf{p}_r$ and $\mathbf{q}_1, \ldots, \mathbf{q}_r$, where r is the rank of $\mathbf{A}$, with corresponding singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$, then it is clear that $\mathbf{U} = a_1[\mathbf{q}_1 \cdots \mathbf{q}_1]$, $\mathbf{V} = a_2[\mathbf{p}_1 \cdots \mathbf{p}_1]$ and $\lambda = \sigma_1$ for some constants a_1 and a_2. To find the scaling factors a_1 and a_2 we can write $\mathrm{tr}(\mathbf{U}^T \mathbf{U}) = a_1 k^2 = 1$ and $\mathrm{tr}(\mathbf{V}^T \mathbf{V}) = a_2 k^2 = 1$ which implies $a_1 = a_2 = 1/\sqrt{k}$. □

These theorems allow us to make the following statement concerning the empirical Rademacher complexity under a matrix factorisation setting.

Theorem 3.7. *Consider a sample* $S = \cup_{i=1}^{m}\{(w_i, y_{\omega_{i1}}), \ldots, (w_i, y_{\omega_{in_i}})\}$ *drawn from distribution* $\mathcal{D}_1^{n_1} \times \cdots \times \mathcal{D}_m^{n_m}$. *Define scoring functions of the form* $\mathcal{Q}_h = \{Q : (w_i, y_p, y_q) \mapsto h(\boldsymbol{u}_i^T\boldsymbol{v}_p - \boldsymbol{u}_i^T\boldsymbol{v}_q) \mid \boldsymbol{U} \in \mathbb{R}^{m\times k}, \boldsymbol{V} \in \mathbb{R}^{n\times k} \| \boldsymbol{U}\|_F \leq R_{\boldsymbol{U}}, \| \boldsymbol{V}\|_F \leq R_{\boldsymbol{V}}\}$ *where* h *is Lipschitz with constant* B *and* k *is a fixed integer. Then the following bound holds on the empirical Rademacher complexity*

$$\hat{R}^{AUC}(\mathcal{Q}_h) \leq \frac{2BR_{\boldsymbol{U}}R_{\boldsymbol{V}}}{m}\|\boldsymbol{E} - \bar{\boldsymbol{E}}\|_2,$$

where $\boldsymbol{\Sigma}$ *is a diagonal matrix of the first* k *singular values of* $(\boldsymbol{E} - \bar{\boldsymbol{E}})$ *with* $\boldsymbol{E} = (diag(\boldsymbol{X}\boldsymbol{j})^{-1}\boldsymbol{X})^T$, $\bar{\boldsymbol{E}} = (diag(\bar{\boldsymbol{X}}\boldsymbol{j})^{-1}\bar{\boldsymbol{X}})^T$ *and* $\bar{\boldsymbol{X}} = \boldsymbol{J} - \boldsymbol{X}$ *where* $diag(\cdot)$ *is a diagonal matrix of its vector input.*

Proof. Consider the following

$$\begin{aligned}
\hat{R}^{AUC}(\mathcal{Q}) &= 2\mathbb{E}_\nu\left[\sup_{Q\in\mathcal{Q}}\frac{1}{m}\sum_{i=1}^{m}\frac{\nu_i}{n_i n_i'}\sum_{p=1}^{n_i}\sum_{q=1}^{n_i'}h(\mathbf{u}_i^T\mathbf{v}_p - \mathbf{u}_i^T\mathbf{v}_q)\right]\\
&\leq \frac{2B}{m}\mathbb{E}_\nu\left[\sup_{\|\mathbf{U}\|_F\leq R_{\mathbf{U}},\|\mathbf{V}\|_F\leq R_{\mathbf{V}}}\sum_{i=1}^{m}\frac{\nu_i}{n_i n_i'}\sum_{p\in\omega_i}\sum_{q\in\bar{\omega}_i}(\mathbf{u}_i^T\mathbf{v}_p - \mathbf{u}_i^T\mathbf{v}_q)\right]\\
&= \frac{2B}{m}\mathbb{E}_\nu\left[\sup_{\|\mathbf{U}\|_F\leq R_{\mathbf{U}},\|\mathbf{V}\|_F\leq R_{\mathbf{V}}}\sum_{i=1}^{m}\nu_i(\mathbf{u}_i^T\dot{\mathbf{v}}_i - \mathbf{u}_i^T\ddot{\mathbf{v}}_i)\right]\\
&= \frac{2B}{m}\mathbb{E}_\nu\left[\sup_{\|\mathbf{U}\|_F\leq R_{\mathbf{U}},\|\mathbf{V}\|_F\leq R_{\mathbf{V}}}\mathrm{tr}(\mathbf{\Pi}^\nu\mathbf{U}\mathbf{V}^T\mathbf{E} - \mathbf{\Pi}^\nu\mathbf{U}\mathbf{V}^T\bar{\mathbf{E}})\right]\\
&\leq \frac{2BR_{\mathbf{U}}R_{\mathbf{V}}}{m}\mathbb{E}_\nu\left[\|(\mathbf{E} - \bar{\mathbf{E}})\mathbf{\Pi}^\nu\|_2\right]\\
&\leq \frac{2BR_{\mathbf{U}}R_{\mathbf{V}}}{m}\|\mathbf{E} - \bar{\mathbf{E}}\|_2
\end{aligned}$$

where $\mathbf{\Pi}_{ii}^\nu = \nu_i$ for all i and off-diagonal elements are zero. The second line results from an application of Theorem 3.5. The fourth line is a matrix representation of the sum in the sup term and in the fifth line we use Lemma 3.1. For the final line, note that the diagonal elements of $\mathbf{\Pi}^\nu$ are selected from $\{-1, +1\}$ and so $\|\mathbf{\Pi}^\nu\|_2 = 1$ and an application of Theorem 3.6 shows the singular values do not change, giving the required result. □

Notice that this bound does not depend on the dimensionality of the factor matrices $\mathbf{U}$ and $\mathbf{V}$. Here we see the motivation for penalising Optimisation 3 by the norm of the weight matrices: doing so keeps the Rademacher complexity low. We would also benefit from choosing loss functions whose Lipschitz constant is small. Putting the parts together allows us to note the following.

Corollary 3.1. *Consider a sample* $S = \cup_{i=1}^{m}\{(w_i, y_{\omega_{i1}}), \ldots, (w_i, y_{\omega_{in_i}})\}$ *drawn from distribution* $\mathcal{D}_1^{n_1} \times \cdots \times \mathcal{D}_m^{n_m}$. *Using the notations defined above the following bound holds on the Rademacher complexity of* $\mathcal{Q}_h$, *with probability greater than* $1 - \delta$,

$$R^{AUC}(\mathcal{Q}_h) \leq \frac{2BR_{\boldsymbol{U}}R_{\boldsymbol{V}}}{m}\|\boldsymbol{E} - \bar{\boldsymbol{E}}\|_2 + \sqrt{\frac{2\ln(1/\delta)(n-1)^2}{m^2}\sum_{i=1}^{m}\frac{1}{|\omega_i||\bar{\omega}_i|^2}}.$$

We now want to make the connection between the expectation of the AUC and the Rademacher complexity. To do so we introduce the following lemma.

Lemma 3.2. *Let $\mathcal{Q}$ be a function class mapping from $\mathcal{W} \times \mathcal{Y}^2$ to $[0,1]$. Then with probability at least $1-\delta$ over all samples S drawn from $\mathcal{D}$, the following holds:*

$$\sup_{Q\in\mathcal{Q}}(\mathbb{E}_{\mathcal{D}}[Q]-\hat{\mathbb{E}}_S[Q]) \leq \mathbb{E}_{S\sim\mathcal{D}}[\sup_{Q\in\mathcal{Q}}(\mathbb{E}_{\mathcal{D}}[Q]-\hat{\mathbb{E}}_S[Q])]+\sqrt{\frac{\ln(1/\delta)}{2m^2}\sum_{i=1}^{m}\frac{1}{|\omega_i|}\left(\frac{n-1}{|\bar{\omega}_i|}\right)^2}.$$

Proof. We can use a similar proof technique to that of Theorem 3.4 to write

$$|\hat{\mathbb{E}}_S[Q]-\hat{\mathbb{E}}_{\hat{S}_{ab}}[Q]| \leq \frac{n-1}{m|\omega_a||\bar{\omega}_a|},$$

Noting that the left hand side can be written as $|(\mathbb{E}[Q]-\hat{\mathbb{E}}_{\hat{S}_{ab}}[Q])-(\mathbb{E}[Q]-\hat{\mathbb{E}}_S[Q])|$ and taking the supremum over Q allows us to bound $|\sup_{Q\in\mathcal{Q}}(\mathbb{E}[Q]-\hat{\mathbb{E}}_S[Q])-\sup_{Q\in\mathcal{Q}}(\mathbb{E}[Q]-\hat{\mathbb{E}}_{\hat{S}_{ab}}[Q])|$:

$$\begin{aligned}
&\leq |\sup_{Q\in\mathcal{Q}}(\mathbb{E}[Q]-\hat{\mathbb{E}}_S[Q])-(\mathbb{E}[Q]-\hat{\mathbb{E}}_{\hat{S}_{ab}}[Q])| \\
&\leq \frac{n-1}{m|\omega_a||\bar{\omega}_a|},
\end{aligned}$$

Define the following function

$$f'(S)=\sup_{Q\in\mathcal{Q}}(\mathbb{E}[Q]-\hat{\mathbb{E}}_S[Q]),$$

then the above bound can be used in conjunction with McDiarmid's theorem and we can say

$$P_{\mathcal{D}}(f'(S)-\mathbb{E}[f']>\epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m}\sum_{j=1}^{n_i}c_{ij}^2}\right),$$

where $c_{ab}=\frac{n-1}{m|\omega_a||\bar{\omega}_a|}$. If we equate the right side of the above to δ we have $\epsilon=\sqrt{\frac{\ln(1/\delta)}{2m^2}\sum_{i=1}^{m}\frac{1}{|\omega_i|}\left(\frac{n-1}{|\bar{\omega}_i|}\right)^2}$. □

Finally we show how the empirical expectations of $Q\in\mathcal{Q}$ found on different samples are related to the Rademacher complexity in the following lemma.

Lemma 3.3. *Let S and $\tilde{S}$ be sampled from the distribution $\mathcal{D}$ and consider a sequence of Rademacher variables $\nu_1,\ldots,\nu_m$, then the following statement is true*

$$\mathbb{E}_{S,\tilde{S}}\sup_{Q\in\mathcal{Q}}[\hat{\mathbb{E}}_{\tilde{S}}[Q]-\hat{\mathbb{E}}_S[Q]] \leq R^{AUC}(\mathcal{Q}).$$

Proof. We start by showing the following is equivalent to $\sup_{Q\in\mathcal{Q}}[\hat{\mathbb{E}}_{\tilde{S}}[Q]-\hat{\mathbb{E}}_S[Q]]$:

$$\mathbb{E}_{S,\tilde{S}\nu}\sup_{Q\in\mathcal{Q}}\left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i n_i'}\sum_{p=1}^{n_i}\sum_{q=1}^{n_i'}(\nu_i Q(w_i,y_{\omega_i p},y_{\bar{\omega}_i q})-\nu_i Q(w_i,y_{\omega_i p},y_{\bar{\omega}_i q}))\right],$$

since when $\nu_i=-1$ for all i the two expressions are clearly the same, and $\nu_i=1$ swaps the observations for the ith user in S with the corresponding ones in $\tilde{S}$. Since both S and $\tilde{S}$ are sampled from the the same distribution the expectation of the

supremum over these sets is the same. Note also that above term is upper bounded by the following:

$$\begin{aligned}
\sup_{Q\in\mathcal{Q}}[\hat{\mathbb{E}}_{\tilde{S}}[Q]-\hat{\mathbb{E}}_S[Q]] &\leq \mathbb{E}_{S,\tilde{S}\nu}\sup_{Q\in\mathcal{Q}}\left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i n_i'}\sum_{p=1}^{n_i}\sum_{q=1}^{n_i'}\nu_i Q(w_i, y_{\omega_i p}, y_{\bar{\omega}_i q})\right] \\
&\quad +\mathbb{E}_{S,\tilde{S}\nu}\sup_{Q\in\mathcal{Q}}\left[-\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i n_i'}\sum_{p=1}^{n_i}\sum_{q=1}^{n_i'}\nu_i Q(w_i, y_{\omega_i p}, y_{\bar{\omega}_i q})\right] \\
&= 2\mathbb{E}_{S,\tilde{S}\nu}\sup_{Q\in\mathcal{Q}}\left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i n_i'}\sum_{p=1}^{n_i}\sum_{q=1}^{n_i'}\nu_i Q(w_i, y_{\omega_i p}, y_{\bar{\omega}_i q})\right] \\
&= R^{AUC}(\mathcal{Q}),
\end{aligned}$$

where the second line comes from the symmetry of distribution of the Rademacher variables. □

We can now bound the expectation of the loss using the Rademacher complexity.

Theorem 3.8. *Let S and $\tilde{S}$ be sampled from the distribution $\mathcal{D}$ and let $\mathcal{Q}$ be a function class mapping from $\mathcal{W}\times\mathcal{Y}^2$ to $[0,1]$. The with probability at least $1-\delta$, the following holds:*

$$\mathbb{E}_{\mathcal{D}}[Q] \leq \hat{\mathbb{E}}_S[Q] + R^{AUC}(\mathcal{Q}) + \sqrt{\frac{\ln(1/\delta)(n-1)^2}{2m^2}\sum_{i=1}^{m}\frac{1}{|\omega_i||\bar{\omega}_i|^2}}.$$

Proof. The result follows directly from Lemmas 3.2 and 3.3. □

Given this result we can examine our loss functions in terms of the bound on the expectation. A key advantage of the bound is it is data-dependent and hence we can estimate the model complexity if we replace the Rademacher complexity term with the bound on its empirical estimate. We have already discussed when this quantity is small as well as a similar analysis for the final term. In practice one finds that the bound on the Rademacher term is larger than the last term hence once must be focus on this term. When studying the loss functions of Section 2.1 one can see that the logistic and sigmoid functions are Lipschitz with constant 1. If we say that the norms of $\mathbf{u}_i$ and $\mathbf{v}_i$ are bounded by D then we have $-2D^2 \leq \gamma_{i,p,q} \leq 2D^2$. The hinge loss is given by $\frac{1}{2}\max(0,1-x)^2$ which implies $D = 1/\sqrt{4}$ so that the loss term is in the range $[0,1]$ and the Lipschitz constant is $3/2$.

4. Related Work

In this section we will briefly review some works on finding low rank matrix factorisations for implicit rating matrices. An early attempt to deal with implicit ratings using a squared loss is presented in [15, 21]. The formulation is an adaptation of the Singular Value Decomposition (SVD), and minimizes

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

where $\mathbf{C}$ is a matrix of weights for user-item pairs and λ is a regularisation parameter. In [21] 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* [4, 18] 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 [23] 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. Furthermore, our framework specialises to BPR in logistical loss case and when the regularisation parameters above are equivalent.

Another paper which considers the AUC is [28] 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) [24] 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.

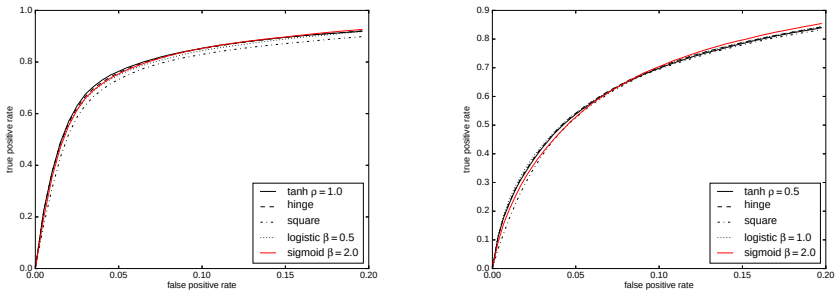
5. Computational Experiments

In this section we analyse various properties of MFAUC empirically in order to understand its behaviour and ranking performance. We consider the question of how well MFAUC optimises the AUC on the training set under specific loss functions and weightings. Another important question is whether the parallel optimisation procedure can speed up convergence of the objective. A final consideration is the evaluation of our ranking framework and other matrix factorisation methods when considering results at the very top of the list for each user. For this comparison we benchmark against other user-item matrix factorisation methods including Soft Impute [18] and Weighted Regularised Matrix Factorisation (WRMF, [15]). Notice that the choice of the logistic loss function and identity weighting implies a comparison to BPR. All experimental code is written in Python with critical sections written in Cython and C++.

We make use of the following 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 $\hat{\mathbf{X}}_{ij} = \mathcal{I}(\mathbf{u}_i^T\mathbf{v}_j > Q(s, 1-t))$ where $Q(s, 1-t_i)$ represents the quantile corresponding to the top t_i items. Here $t_i = .1$ so that there are on average 20 nonzero elements per row of $\hat{\mathbf{X}}$, and we 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 $\mathbf{Z}_{ij} \geq \hat{\mathbb{E}}[\mathbf{Z}]$, $\mathbf{Z} = \mathbf{U}\mathbf{V}^T$, otherwise it is zero. We continue this process until the density of the matrix is at least .1. The final matrix is of size 573×300 with 17796 non-zeros.

5.1. **ROC Analysis of Losses.** First we analyse the maximisation of AUC under the loss functions we outlined in Section 2.1 using the synthetic datasets. The MFAUC algorithm is set up with $k = 8$ using $\kappa_{\mathcal{W}} = 30$ row samples and $\kappa_{\mathcal{Y}} = 10$ column samples to compute approximate derivatives for $\mathbf{u}_i$ and $\mathbf{v}_i$. The initial values of $\mathbf{U}$ and $\mathbf{V}$ are set to zero mean Gaussian random values with a standard deviation of .1, the regularisation constant $\lambda = 0$, maximum iterations $T = 500$ and item distribution exponent $\tau = 0$. The learning rate is $\alpha = .05$ and we choose

$\beta \in \{.5, 1.0, 2.0\}$ for sigmoid and logistic losses and $\rho \in \{.5, 1.0, 2.0\}$ for the tanh item weighting. To make the problem harder we remove 5 items for each user and then train using the remaining items, recording the ROC curve at the end of the procedure. This process is repeated 5 times, averaging results. Since we are interested in the top items of the list, we consider items until a false positive rate of 20% is encountered. For clarity we only include the best ROC curves for logistic, sigmoid and tanh-based losses, defined as the largest true positive rate at 20% false positives.

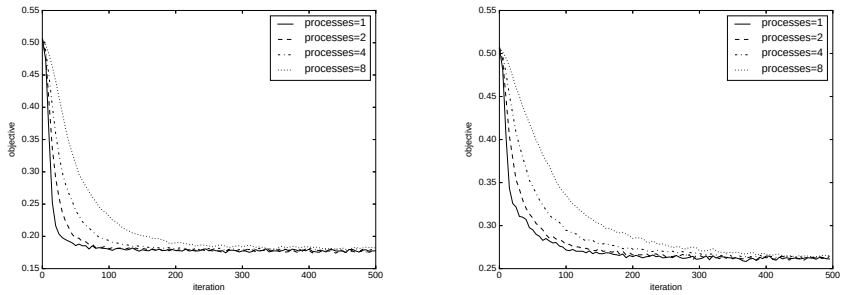


(a) `Synthetic1` (b) `Synthetic2`

FIGURE 3. Left side of ROC curves for the synthetic datasets using different loss functions.

The ROC plots are presented in Figure 3 and on the whole we see that differences are slight for both datasets. On `Synthetic1` the tanh weighting function is slightly preferential at the start of the curve relative to the other losses, followed by hinge, sigmoid and then the logistic loss. The square loss provides the worst of the results as it penalises only the difference in the scores for positive and negative items, however one does gain a speed advantage as outlined in Section 2.2. We can see from the curves on `Synthetic2` that this is a more challenging problem. We observe the logistic loss providing the fewest errors at the start of the ranking followed closely by hinge and then the tanh loss. The sigmoid and square loss curves are poor in this case, the latter for the reason we have already mentioned. Performing gradient descent over the sigmoid loss function can be challenging for the reasons we mentioned in Section 2.1.

5.2. **Optimisation Strategy.** We now examine the parallel optimisation proposed in Section 2.2 in terms of convergence and timing. The MFAUC algorithm is set up with $k = 8$ using $\kappa_{\mathcal{U}} = 30$ row samples and $\kappa_{\mathcal{Y}} = 10$ column samples to compute approximate derivatives for $\mathbf{u}_i$ and $\mathbf{v}_i$. The initial values of $\mathbf{U}$ and $\mathbf{V}$ are set to random Gaussian values, then we fix learning rate $\alpha = .05$, regularisation $\lambda = .1$, maximum iterations $T = 300$, and use the hinge loss. To see how the parallelisation affects convergence and timings, we record these values for the parallel optimisation with 2, 4, and 8 processes, as well as the standard SGD approach, repeating experiments 5 times to get average quantities for each parameter set. 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.



(a) `Synthetic1` (b) `Synthetic2`

FIGURE 4. Plots showing the objective function on the synthetic datasets with different optimisation routines.

	1	2	4	8
`Synthetic1`	607.1	309.1	163.6	106.9
`Synthetic2`	689.2	332.8	201.9	16.3

TABLE 1. Timings (in seconds) of the optimisation routines with the synthetic datasets by number of processes.

Figure 4 compares the objective values and Table 1 shows the timings of the parallel and non-parallel variants of SGD on both datasets. We observe approximate speedups of 3 times with 8 processes whilst converging to approximately the same objective values for both datasets. When we look at the objective against the iteration number, the parallel SGD converges slower than the non-parallel version particularly when using 8 cores. One of the reasons for this decrease in convergence rate is that the dataset is split into smaller blocks when more processes are applied. 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.

5.3. **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. For MFAUC the parameters are identical to the setup used above and we select learning rates from $\alpha \in \{2^{-1}, \ldots, 2^{-8}\}$ and $\lambda \in \{2^{0}, \ldots, 2^{-10}\}$. The initial factor matrices are computed using the randomised SVD. The F1 measure is recorded on validation items in conjunction with MFAUC and used to pick the best solution. We take a sample of 3 validation items from the users to form this validation set. With Soft Impute, we use regularisation parameters $\lambda \in \{1.0, .8, \ldots, 0\}$ and the randomised SVD to update solutions as proposed in [8]. The regularisation parameters for WRMF are chosen from $\lambda \in \{2^{1}, 2^{-1}, \ldots, 2^{-11}\}$. To select the best parameters we use the

		p@1	p@3	p@5	r@1	r@3	r@5	AUC
Syn1	SoftImpute	.831	.696	.549	.166	.417	.549	.914
	WRMF	**.893**	.754	.610	**.179**	.452	.610	**.924**
	MFAUC hinge	.855	.759	.606	.171	.455	.606	**.924**
	MFAUC square	.863	.755	.607	.173	.453	.607	.922
	MFAUC sigmoid	.867	.772	**.626**	.173	.463	**.626**	.923
	MFAUC logistic	.872	.762	.614	.174	.457	.614	**.924**
	MFAUC tanh $\rho = .5$	.854	.752	.602	.171	.451	.602	.923
	MFAUC tanh $\rho = 1.0$	.866	.760	.615	.173	.456	.615	**.924**
	MFAUC tanh $\rho = 2.0$	.874	**.775**	.618	.175	**.465**	.618	.921
	MFAUC tanh $\rho = 5.0$	.880	.764	.604	.176	.459	.604	.920
Syn2	SoftImpute	.225	.193	.172	.045	.116	.172	.766
	WRMF	.431	**.297**	**.241**	.086	**.178**	**.241**	**.817**
	MFAUC hinge	.428	.294	.240	.086	.176	.240	.796
	MFAUC square	.422	.290	.236	.084	.174	.236	.801
	MFAUC sigmoid	.416	.284	.228	.083	.170	.228	.795
	MFAUC logistic	**.444**	.295	.238	**.089**	.177	.238	.800
	MFAUC tanh $\rho = .5$	.415	.284	.232	.083	.171	.232	.791
	MFAUC tanh $\rho = 1.0$	.420	.284	.230	.084	.170	.230	.790
	MFAUC tanh $\rho = 2.0$	.395	.285	.231	.079	.171	.231	.797
	MFAUC tanh $\rho = 5.0$	.362	.259	.214	.072	.156	.214	.776

TABLE 2. Test errors on the synthetic datasets. Top represents `Synthetic1` and bottom is `Synthetic2`. Best results are in bold.

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 2 shows the performance of the matrix factorisation methods. On both datasets, MFAUC and WRMF perform better than Soft Impute and particularly on `Synthetic2`. One explanation is that WRMF and MLAUC do not make assumptions about the distributions of the relevant items like Soft Impute. On the harder `Synthetic2` dataset WRMF gives the best overall results closely followed by logistic and hinge loss MLAUC.

5.3.1. *Real Datasets.* Next we consider a set of real world datasets: MovieLens, Flixster, Epinions and Book Crossing. 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. Epinions ratings are given on the scale 0 to 5 and Book Crossing ones are given from 0 to 10, and we assign relevance to ratings greater than 3 and 4 respectively. For all datasets, we remove users with less than 10 items and items with less than 2 users, repeating this process until convergence is reached. Properties about the resulting matrices are shown in Table 3.

The experimental procedure is similar to before except that we select $k \in \{32, 64, 128\}$ in this case and use 2 model selection repetitions. 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,

Dataset	users	items	nonzeros	sparsity (%)
Book Crossing	9571	68,517	640,430	0.098
Epinions	12,663	38,499	371,969	0.076
Flixster	43,979	32,024	5,147,187	0.37
MovieLens	897	1281	54,883	4.78

TABLE 3. Properties of the real datasets

we pick users sequentially until the desired number of ratings is reached. Any items which then have no ratings are removed. The parameters for MFAUC are chosen from $\alpha \in \{2^1, \ldots, 2^{-2}\}$, $\lambda \in \{2^0, \ldots, 2^{-5}\}$ and $\kappa_{\mathcal{W}} = 15$. After the model selection step, we use the parallel SGD procedure in conjunction with MLAUC to compute the final matrix factorisation.

Table 4 shows the results on these datasets. It is clear that the non-ranking based methods perform reasonably well on the more sparse datasets Epinions and Book Crossing relative to MLAUC. We noticed in the model selection stage for example, MLAUC would not converge adequately for many parameter sets. Whilst this did not negatively impact the AUC, it did effect the items at the very top of the list, hence the low precision and recall scores for these datasets. In contrast, we see with Flixster and MovieLens that the ranking methods show their advantage particularly with the hinge and logistic losses. With MovieLens for example, all of the ranking losses improve over WRMF in the precisions at 3 and 5.

A useful comparison is between the hinge and *tanh* losses since it demonstrates the effectiveness of prioritising list elements. The results are mixed: on the Epinions and MovieLens datasets we gain an improvement with this prioritisation function, but on Flixster results are worse. A difficultly of the approach is that one must correctly set ρ for accurate results.

6. 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. Furthermore, 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. In addition we show that our chosen loss functions are consistent with AUC and gained insight into the generalisation of the algorithms using Rademacher Theory.

From the computational study we can conclude that the proposed parallelisation by block optimisation speeds up the convergence whilst keeping the quality of the objective relative to single process optimisation. The weighting of observations, which gives more importance to items which are more often relevant, can be effective for certain datasets. In general, the MFAUC approach is a useful tool when the sparsity of the underlying dataset under examination is not too high.

		p@1	p@3	p@5	r@1	r@3	r@5	AUC
Book Crossing	SoftImpute	**.040**	**.031**	**.027**	**.008**	**.019**	**.027**	.782
	WRMF	**.040**	.030	.026	**.008**	.018	.026	.754
	MFAUC hinge	.018	.016	.014	.004	.010	.014	.845
	MFAUC sigmoid	.015	.013	.012	.003	.008	.012	**.849**
	MFAUC logistic	.020	.016	.014	.004	.010	.014	.836
	MFAUC tanh $\rho = .5$	.015	.012	.010	.003	.007	.010	.797
	MFAUC tanh $\rho = 1.0$	.018	.015	.013	.004	.009	.013	.833
	MFAUC tanh $\rho = 2.0$	.018	.016	.014	.004	.009	.014	.846
Epinions	SoftImpute	.037	.030	.026	.007	.018	.026	.793
	WRMF	**.041**	**.031**	**.027**	**.008**	**.019**	**.027**	.771
	MFAUC hinge	.025	.020	.019	.005	.012	.019	.826
	MFAUC sigmoid	.025	.020	.018	.005	.012	.018	.819
	MFAUC logistic	.034	.027	.023	.007	.016	.023	.853
	MFAUC tanh $\rho = .5$	.027	.024	.021	.005	.014	.021	.824
	MFAUC tanh $\rho = 1.0$	.031	.026	.023	.006	.016	.023	.854
	MFAUC tanh $\rho = 2.0$	.033	.028	.024	.007	.017	.024	**.859**
Flixster	SoftImpute	.157	.111	.089	.031	.067	.089	.926
	WRMF	.167	.119	.096	.033	.071	.096	.891
	MFAUC hinge	**.168**	**.132**	**.112**	**.034**	**.079**	**.112**	**.984**
	MFAUC sigmoid	.121	.090	.076	.024	.054	.076	.980
	MFAUC logistic	.167	.126	.107	.033	.076	.107	**.984**
	MFAUC tanh $\rho = .5$	.119	.090	.077	.024	.054	.077	.981
	MFAUC tanh $\rho = 1.0$	.058	.049	.043	.012	.029	.043	.967
	MFAUC tanh $\rho = 2.0$	.069	.052	.047	.014	.031	.047	.969
MovieLens	SoftImpute	.209	.165	.140	.042	.099	.140	.880
	WRMF	.225	.174	.145	.045	.104	.145	.891
	MFAUC hinge	.217	.183	.158	.043	.110	.158	.919
	MFAUC square	.211	.176	.157	.042	.106	.157	.925
	MFAUC sigmoid	.237	**.193**	.165	.047	**.116**	.165	.924
	MFAUC logistic	**.241**	.191	**.166**	**.048**	.114	**.166**	**.926**
	MFAUC tanh $\rho = .5$	.223	.185	.161	.045	.111	.161	.922
	MFAUC tanh $\rho = 1.0$	.226	.183	.159	.045	.110	.159	.923
	MFAUC tanh $\rho = 2.0$	.222	.180	.157	.044	.108	.157	.918

TABLE 4. Test errors on the real datasets.

ACKNOWLEDGEMENTS

This work is funded by the Eurostars ERASM project.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 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, 2005.

[2] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2003.

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

[4] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[5] Stéphan Clémençon and Nicolas Vayatis. Ranking the best instances. *The Journal of Machine Learning Research*, 8:2671–2699, 2007.

[6] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

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

[8] Charanpal Dhanjal, Romaric Gaudel, and Stéphan Clémençon. Online matrix completion through nuclear norm regularisation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 623–631, 2014.

[9] Charanpal Dhanjal, Romaric Gaudel, and Stéphan Clémençon. Collaborative filtering with localised ranking. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[10] Stanley C Eisenstat and Ilse CF Ipsen. Relative perturbation techniques for singular value problems. *SIAM Journal on Numerical Analysis*, 32(6):1972–1988, 1995.

[11] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-pass auc optimization. In *Proceedings of The 30th International Conference on Machine Learning*, pages 906–914, 2013.

[12] Wei Gao and Zhi-Hua Zhou. On the consistency of auc optimization. *arXiv preprint arXiv:1208.0645*, 2012.

[13] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[14] Alan Herschtal and Bhavani Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*, page 49. ACM, 2004.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

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

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

[18] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[19] Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, Cambridge, 1989.

[20] Ron Meir and Tong Zhang. Generalization error bounds for bayesian mixture algorithms. *The Journal of Machine Learning Research*, 4:839–860, 2003.

[21] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

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

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[24] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.

[25] Harald Steck. 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*, pages 713–722. ACM, 2010.

[26] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 125–132. ACM, 2011.

[27] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V.D.P. Servedio. 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)*, 2007.

[28] Jason Weston, Hector Yee, and Ron J Weiss. Learning to rank recommendations with the k-order statistic loss. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 245–248. ACM, 2013.

[29] Lian Yan, Robert H Dodier, Michael Mozer, and Richard H Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 848–855, 2003.