Collaborative Ranking for Local Preferences

Berk Kapicioglu YP

David S. Rosenberg YP

Robert E. Schapire Princeton University

Tony Jebara Columbia University

Abstract

For many collaborative ranking tasks, we have access to relative preferences among subsets of items, but not to global preferences among all items. To address this, we introduce a matrix factorization framework called Collaborative Local Ranking (CLR). We justify CLR by proving a bound on its generalization error, the first such bound for collaborative ranking that we know of. We then derive a simple alternating minimization algorithm and prove that its running time is independent of the number of training examples. We apply CLR to a novel venue recommendation task and demonstrate that it outperforms state-of-the-art collaborative ranking methods on real-world data sets.

1 Introduction

Since the early days of the Netflix Prize competition, matrix factorization (MF) [7] has become a popular method for modeling users' preferences over a set of items. MF achieves state-of-the-art performance on very large-scale datasets, such as the Netflix dataset [2], which comprises more than one hundred million ratings. MF also does not require user and item features [6], making it particularly useful for practitioners, as they can easily apply it to new domains without designing domain-specific features.

Most MF methods optimize root-mean-square error (RMSE) [15][12][11], largely because the winner of the Netflix Prize was determined by it. However, for many applications of collaborative filtering, RMSE is inappropriate because actual performance is based on predicting a rank rather than a rating. For example, when choosing which sci-fi movie to watch, a user might wonder how Star Trek ranks against Serenity, and predicting 3.8 for Star Trek and 4.5 for Serenity is relevant only in so far as helping the user infer a relative ranking.

As a remedy, researchers have devised various algorithms for collaborative ranking. Weimer et al. [17] proposed CofiRank, which is an MF method that optimizes ranking measures, such as normalized discounted cumulative gain (NDCG). Rendle et al. [9] analyzed the collaborative ranking problem from a Bayesian perspective and provided a meta-optimization criterion called Bayesian Personalized Ranking (BPR), which when coupled with MF, reduces into a differentiable version of CofiRank's objective. Balakrishnan and Chopra [1] used a two-stage model to rank; the first stage learns the latent factors via Probabilistic Matrix Factorization (PMF) [12], and the second stage processes these factors as features using supervised regression and ranking algorithms.

Existing collaborative ranking algorithms, including CofiRank and BPR, are derived under the assumption that users' preferences are totally ordered. Given a training dataset, even if it is not totally ordered, these algorithms process it by partitioning the entire universe of items into observed and unobserved items and ranking the observed items higher than the unobserved ones. However, there are many tasks where such a global partitioning is inappropriate. For example, for a movie recommendation task, when a user browses a small list of sci-fi titles and eventually chooses one, she prefers the chosen movie over the remaining sci-fi movies in that list, but she does not necessarily prefer the chosen movie over *all* other movies. Similarly, for a venue recommendation task, when a user considers nearby restaurants and settles on one, she demonstrates a local preference among a small set of nearby restaurants, rather than a global preference between the chosen restaurant and *all* other restaurants.

We are interested in an algorithm that not only learns from local preferences encountered during training, but also is evaluated according to the local preferences it predicts during deployment. For example, for a venue recommendation task, the user would query

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33. Copyright 2014 by the authors.

the recommendation system when she is at a particular neighborhood, and the system would return a local, not global, ranking over nearby venues. Thus, a successful recommendation system does not just use local preferences it observes during training to piece together a global ranking; what the user ultimately cares about are local rankings themselves.

Recently, Yang et al. [18] studied the problem of learning local rankings in a collaborative ranking setting using random utility theory. They assumed that the error terms are distributed with respect to a Weibull distribution, and they derived a multinomial logit model whose likelihood they maximized. In contrast, we develop a more theoretically rigorous framework using computational learning theory. We base all our decisions on minimizing the loss function, and we motivate these decisions using learning-theoretic arguments that provably minimize the running time and maximize the generalization performance.

We proceed as follows. In Section 2, we formally set up the problem, make very general assumptions about the sample space (i.e. examples are generated i.i.d from an unknown probability distribution, etc.), and introduce the *local ranking loss*, which can be viewed as a local version of AUC. In Section 3, we prove a generalization error bound, which to the best of our knowledge, is the first bound of its kind for collaborative ranking. In Section 4, we describe the CLR objective, justify it using the generalization bound, and explore its relationship with other learning methods, including the objective proposed by Yang et al. [18]. In Section 5, we derive a simple alternating minimization algorithm for training CLR and prove that its running time is independent of the number of training examples. In Section 6, we present empirical results on a real-world venue recommendation task that naturally lends itself to our local ranking framework due to its spatial structure. We conclude in Section 7.

2 Problem Formulation

In this section, we formally set up the problem by describing our assumptions about the data, the hypothesis space, and the loss function. We assume that data is generated from a sample space $\mathcal{X}$, and each data point consists of a user, a candidate set of items, a local time, and a single item that the user prefers over the remaining candidates (i.e. the label). More formally, let $\mathcal{U} = \{1, \ldots, m\}$ be the set of users, let $\mathcal{V} = \{1, \ldots, n\}$ be the set of items, and let $\mathcal{T} = \{1, \ldots, T\}$ indicate the local time. Then, the sample space is defined as

$$\mathcal{X} = \{(u, C, i, t) \mid u \in \mathcal{U}, C \subseteq \mathcal{V}, i \in C, t \in \mathcal{T}\}. \quad (1)$$

Given a training dataset, the learning algorithm's goal is to choose a hypothesis $g : \mathbb{Z}^3 \to \mathbb{R}$, which would map a user u, item i, and time index t to a scalar value. Once the training is complete and the hypothesis is deployed, the recommendation system may be queried with a user u, candidate set C, and local time. In return, the hypothesis assigns a scalar value to each candidate item and induces a ranking over the candidate set, and the higher the value of an item, the higher its rank. For example, in case of venue recommendation, the user sends the system her geographical coordinates, the system identifies the nearby venues, forms the candidate set, applies the hypothesis, and ranks the candidate venues.

The performance of the recommendation system depends on how high it ranks the item that the user will ultimately choose. More formally, let $P[\cdot]$ denote probability, let C^i be the set C excluding element i, and let $c \overset{U}{\sim} C$ mean that c is sampled uniformly from C. Then, the *local ranking loss* associated with hypothesis g is

$$L_g(u, C, i, t) = \underset{c \overset{U}{\sim} C^i}{P}\left[g(u, i, t) - g(u, c, t) \leq 0\right]. \quad (2)$$

Intuitively, if the highest scalar value is assigned to the correct item i, there is no loss. Otherwise, the loss is proportional to the number of items in the candidate set that are ranked higher than the correct item. Alternatively, local ranking loss can be viewed as a local version of AUC loss, since if we generalize a candidate set to be the entire universe of items, we would obtain the regular AUC loss.

To simplify the discussion, for most of the paper, we will assume that the sample space, hypotheses, and loss functions all exclude references to local time. We will revisit this issue in Section 4.

3 A Bound on the Generalization Error

Our ultimate goal is to devise algorithms that minimize the generalization error of collaborative ranking. In this section, we focus specifically on deriving a bound on the generalization error, which in turn will influence our algorithmic design.

We assume that the hypothesis class is based on the set of low-rank matrices. Given a low-rank matrix M, let $g_M \in \mathcal{F}$ be the associated hypothesis, where $g_M(u, i) = M_{u,i}$. Throughout the paper, we abuse notation and use g_M and M interchangeably. We assume that data is generated with respect to $\mathcal{D}$, which is an unknown probability distribution over the sample space $\mathcal{X}$, and we let $\mathbb{E}$ denote expectation. Then, the generalization error of hypothesis M

is $\mathop{\mathbb{E}}_{(u,C,i)\sim\mathcal{D}} L_M(u,C,i)$, which is the quantity we bound below.

We will derive the generalization bound in two steps. In the first step, we will bound the empirical Rademacher complexity of our loss class, defined below, with respect to samples that contain exactly 2 candidates, and in the second step, we will prove the generalization bound with a reduction to the previous step.

Lemma 1. *Let m be the number of users and let n be the number of items. Define $\mathcal{L}_r = \{L_M \mid M \in \mathbb{R}^{m\times n}$ has rank at most $r\}$ as the class of loss functions associated with low-rank matrices. Assume that $S_2 \subseteq \mathcal{X}$ is a set of d samples, where each sample contains exactly 2 candidate items; i.e. if $(u,C,i) \in S_2$, then $|C| = 2$. Let $R_{S_2}(\mathcal{L}_r)$ denote the Rademacher complexity of $\mathcal{L}_r$ with respect to S_2. Then,*

$$R_{S_2}(\mathcal{L}_r) \le \sqrt{\frac{2r(m+n)\ln\left(\frac{16emn^2}{r(m+n)}\right)}{d}}.$$

Proof. Because each sample in S_2 contains exactly 2 candidates, any hypothesis $L_M \in \mathcal{L}_r$ applied to a sample in S_2 outputs either 0 or 1. Thus, the set of dichotomies that are realized by $\mathcal{L}_r$ on S_2, called $\Pi_{\mathcal{L}_r}(S_2)$, is well-defined. Using Equation 6 from Boucheron et al. [3], we know that $R_{S_2}(\mathcal{L}_r) \le \sqrt{\frac{2\ln|\Pi_{\mathcal{L}_r}(S_2)|}{d}}$. Let $\mathcal{X}_2 \subseteq \mathcal{X}$ be the set of all samples that contain exactly 2 candidates, $|\Pi_{\mathcal{L}_r}(S_2)| \le |\Pi_{\mathcal{L}_r}(\mathcal{X}_2)|$, so it suffices to bound $|\Pi_{\mathcal{L}_r}(\mathcal{X}_2)|$.

We bound $|\Pi_{\mathcal{L}_r}(\mathcal{X}_2)|$ by counting the sign configurations of polynomials. Let $(u,\{i,j\},i) \in \mathcal{X}_2$ be a sample and let M be a hypothesis matrix. Because M has rank at most r, it can be written as $M = UV^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Let $[\![\cdot]\!]$ denote an indicator function that is 1 if and only if its argument is true. Then, the loss on the sample can also be rewritten as $L_M(u,\{i,j\},i) = [\![M_{u,i} - M_{u,j} \le 0]\!] = [\![\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,j} \le 0]\!] = [\![\sum_{a=1}^{r} U_{u,a}(V_{i,a} - V_{j,a}) \le 0]\!]$. Since cardinality of $\mathcal{X}_2$ is at most $2m\binom{n}{2} \le mn^2$, putting it all together, it follows that $|\Pi_{\mathcal{L}_r}(\mathcal{X}_2)|$ is bounded by the number of sign configurations of mn^2 polynomials, each of degree at most 2, over $r(m+n)$ variables. Applying Corollary 3 from Srebro et al. [14], we obtain $|\Pi_{\mathcal{L}_r}(\mathcal{X}_2)| \le \left(\frac{16emn^2}{r(m+n)}\right)^{r(m+n)}$. Taking logarithms and making basic substitutions yield the desired result. □

We proceed to proving the more general result via a reduction to Lemma 1. We will no longer assume that $|C| = 2$.

Theorem 1. *Let m be the number of users and let n be the number of items. Assume that S consists of d independently and identically distributed samples chosen from $\mathcal{X}$ with respect to a probability distribution $\mathcal{D}$. Let L_M be the loss function associated with a matrix M, as defined in Equation 2. Then, with probability at least $1-\delta$, for any matrix $M \in \mathbb{R}^{m\times n}$ with rank at most r,*

$$\mathop{\mathbb{E}}_{(u,C,i)\sim\mathcal{D}} L_M(u,C,i) \le \mathop{\mathbb{E}}_{(u,C,i)\overset{U}{\sim}S} L_M(u,C,i) + 2\sqrt{\frac{2r(m+n)\ln\left(\frac{16emn}{r}\right)}{d}} + \sqrt{\frac{2\ln\left(\frac{2}{\delta}\right)}{d}}. \quad (3)$$

Proof. We will manipulate the definition of Rademacher complexity [3] in order to use the bound given in Lemma 1:

$$\begin{aligned}
R_S(\mathcal{L}_r) &\doteq \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathcal{L}_r}\left(\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,C_a,i_a)\right)\right] \\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathcal{L}_r}\left(\frac{1}{d}\sum_{a=1}^{d}\sigma_a \mathop{\mathbb{E}}_{j_a\overset{U}{\sim}(C_a\setminus\{i_a\})} L_M(u_a,\{i_a,j_a\},i_a)\right)\right] \\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathcal{L}_r}\left(\mathop{\mathbb{E}}_{j_1,\ldots,j_d}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] \\
&\le \mathop{\mathbb{E}}_{\sigma}\left[\mathop{\mathbb{E}}_{j_1,\ldots,j_d}\left(\sup_{L_M\in\mathcal{L}_r}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] \\
&= \mathop{\mathbb{E}}_{j_1,\ldots,j_d}\left[\mathop{\mathbb{E}}_{\sigma}\left(\sup_{L_M\in\mathcal{L}_r}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] \\
&= \mathop{\mathbb{E}}_{j_1,\ldots,j_d}\left[R_{S_2}(\mathcal{L}_r)\right] \\
&\le \sqrt{\frac{2r(m+n)\ln\left(\frac{16emn^2}{r(m+n)}\right)}{d}}
\end{aligned}$$

Plugging the bound to Theorem 3.2 in Boucheron et al. [3] proves the theorem. □

Srebro et al. [14] used covering numbers to prove a generalization error bound for collaborative binary classification, which is matrix factorization framework where the matrix entries are binary and the loss function is 0-1. Even though their learning setting is different from ours, we can still compare our bound in Theorem 1 with their bound, and it can be seen that they match up to logarithmic factors. Thus, collaborative local ranking maintains the same generalization error as collaborative binary classification in exchange for a small increase in sample size.

We can improve the tightness of our generalization error bound by restricting the sample space $\mathcal{X}$. Let

$\mathcal{X}_2 \subseteq \mathcal{X}$ be the set of all samples that contain exactly 2 candidates. In our proof of Lemma 1, we loosely bounded the cardinality of the sample space with $|\mathcal{X}_2| \leq mn^2$, but many collaborative ranking tasks have additional structure which lead to better bounds. For example, in case of venue recommendation, cardinality of $\mathcal{X}_2$ is small, because two venues are only assigned to the same candidate set if they are near each other. If we assume that every venue has at most b nearby venues, then $|\mathcal{X}_2| \leq mnb$, and the n in the logarithm of Equation 3 gets replaced by a small constant. Thus, by making additional assumptions about the structure of the sample space, the generalization error bound in Theorem 1 becomes asymptotically equivalent to the bound provided in Srebro et al. [14], even though the latter bound was derived for the simpler collaborative classification setting.

4 Collaborative Local Ranking

In this section, we describe CLR by formulating its objective function, which we justify using the generalization error bound derived above. We also make connections between CLR and other learning methods, such as maximum-margin matrix factorization (MMMF) [10], ranking support vector machine (ranking SVM) [5], and collaborative competitive filtering (CCF) [18].

Given the generalization error bound in Theorem 1, a reasonable objective is to minimize the empirical local ranking loss, which is the first term on the right hand side of Equation 3. However, this function is discontinuous and difficult to minimize with respect to M, so we propose minimizing a more tractable upper bound instead. Let $h(x) = \max(0, 1-x)$ be the hinge function, let M be the hypothesis matrix with rank at most r, and let $M = UV^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Then, we can bound the empirical local ranking loss as

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(u,C,i)\overset{U}{\sim}S} L_M(u,C,i) = \frac{1}{|S|}\sum_{(u,C,i)\in S} L_M(u,C,i) \\
&= \frac{1}{|S|}\sum_{(u,C,i)\in S} \mathop{P}_{c \overset{U}{\sim} C^i}\left[M_{u,i} - M_{u,c} \leq 0\right] \\
&= \frac{1}{|S|}\sum_{(u,C,i)\in S} \mathop{\mathbb{E}}_{c \overset{U}{\sim} C^i}\left[\!\left[M_{u,i} - M_{u,c} \leq 0\right]\!\right] \\
&= \frac{1}{|S|}\sum_{(u,C,i)\in S} \frac{1}{|C^i|}\sum_{c\in C^i}\left[\!\left[\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} \leq 0\right]\!\right] \\
&\leq \frac{1}{|S|}\sum_{(u,C,i)\in S} \frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).
\end{aligned} \tag{4}$$

The resulting upper bound is not necessarily convex with respect to U and V jointly, but it is convex with respect to U if V is fixed, and vice versa.

We use trace norm regularization, in addition to the low-rank constraint, to further restrict the hypothesis class. Let $\|\cdot\|_T$ denote the trace norm and let $\|\cdot\|_F$ denote the Frobenius norm, then we use the equality $\|M\|_T = \min_{U,V,M=UV^T} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$ given in Lemma 6 of Mazumder et al. [8] to state the objective in its final form.

Definition 1 (CLR Objective)**.** Let S be the training data. Let $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ be the factor matrices, and let $\lambda > 0$ be the regularization parameter. The CLR objective is

$$\begin{aligned}
&f^{\mathrm{CLR}}(S; U, V) = \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \\
&+ \frac{1}{|S|}\sum_{(u,C,i)\in S}\frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).
\end{aligned}$$

If $\lambda = 0$, then the CLR objective is equivalent to minimizing the bound in Equation 4. If $\lambda \neq 0$, then it can be shown that the equivalence still holds, but under the constraint that matrices have bounded trace norm in addition to having bounded rank. Even though minimizing Equation 4 is more directly justified by our generalization bound, we use the CLR objective instead, which is more general, to allow additional regularization. Note that the resulting hypothesis still satisfies the assumptions of Theorem 1. Rank-truncated trace norms have been used as regularizers in other collaborative learning settings, such as by Foyget et al. [4] and by Rennie and Srebro [10], and have been demonstrated to work well.

We note that the CLR and the ranking SVM [5] objectives are closely related. If V is fixed and we only need to minimize U, then each row of V acts as a feature vector for the corresponding item, each row of U acts as a separate linear predictor, and the CLR objective decomposes into solving simultaneous ranking SVM problems. In particular, let $S_u = \{(a, C, i) \in S \mid a = u\}$ be the examples that correspond to user u, let U_u denote row u of U, and let f^{rSVM} denote the objective function of ranking SVM, then

$$f^{\mathrm{CLR}}(S; U, V) = \sum_{u=1}^{m} f^{\mathrm{rSVM}}(S_u; U_u, V).$$

This correspondence is not surprising, since in the ranking SVM setting, items have features, the training data consists of binary orderings between items, and the ranking SVM objective itself is a convex upper bound on the number of misordered pairs. In contrast, if U is fixed and we only need to minimize V, the resulting objective does not correspond to any objective function that we know of.

Next, we compare the CLR objective with the CCF objective proposed by Yang et al. [18]. Even though they look similar, the two objectives are actually different. In particular, given a sample (u, C, i), the loss term in the CCF objective is

$$h\left(\left(UV^T\right)_{u,i} - \frac{1}{|C^i|}\sum_{c\in C^i}\left(UV^T\right)_{u,c}\right),$$

whereas the loss term in the CLR objective is

$$\frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).$$

The CCF objective is not penalized if the score of the correct item is sufficiently higher than the average score of the incorrect items. However, such cases occur even when most of the scores of incorrect items are actually higher than the score of the correct item, as long as there exists some incorrect item with a low enough score that brings the overall average down. The CLR objective does not suffer from this drawback. Furthermore, we have already proved that minimizing the CLR objective is a reasonable proxy for minimizing the expected local ranking loss. It is harder to make a similar argument for the CCF objective.

Lastly, we extend the CLR objective to incorporate features. In the context of venue recommendation, such features might be obtained from an external database and indicate the venue type or the query time stamp. We assume an extended sample space, and given a sample $(u, C, i, t) \in \mathcal{X}$, we let t denote the query time stamp.

Definition 2 (CLR.F Objective)**.** Let S be the training data. Given a sample $(u, C, i, t) \in S$, assume that $F_{u,i,t} \in \mathbb{R}^q$ denotes the corresponding feature vector. Let $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ be the factor matrices, let $w \in \mathbb{R}^q$ be the feature coefficients, and let $\lambda, \gamma > 0$ be the regularization parameters. The CLR.F objective is

$$f^{\text{CLR.F}}(S; U, V, w) = \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \frac{\gamma}{2}\|w\|^2 + \frac{1}{|S|}\sum_{(u,C,i,t)\in S}\frac{1}{|C^i|}\times \sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} + w^T\left(F_{u,i,t} - F_{u,c,t}\right)\right).$$

Algorithm 1 Alternating minimization for optimizing the CLR objective.

```
Input: Training data $S \subseteq \mathcal{X}$, regularization parameter $\lambda > 0$, rank constraint $r$, number of iterations $T$.
1: $U_1 \leftarrow$ Sample matrix uniformly at random from $\left[-\frac{1}{\sqrt{\lambda m r}}, \frac{1}{\sqrt{\lambda m r}}\right]^{m\times r}$.
2: $V_1 \leftarrow$ Sample matrix uniformly at random from $\left[-\frac{1}{\sqrt{\lambda n r}}, \frac{1}{\sqrt{\lambda n r}}\right]^{n\times r}$.
3: for all t from 1 to $T-1$ do
4:     $U_{t+1} \leftarrow \arg\min_U f^{\text{CLR}}(S; U, V_t)$
5:     $V_{t+1} \leftarrow \arg\min_V f^{\text{CLR}}(S; U_{t+1}, V)$
6: return $U_T, V_T$.
```

5 Algorithms

5.1 Derivation

In this subsection, we derive and describe our algorithm for optimizing the CLR objective. As we noted before, the CLR objective is not necessarily jointly convex in U and V, but it is convex in U when V is fixed and vice versa. We minimize the CLR objective using alternating minimization, where we sequentially alternate between solving the convex subproblems, and we solve each such subproblem using projected stochastic subgradient descent. The pseudocode is depicted in Algorithm 1.

Now, we derive the projected stochastic subgradient descent algorithm for minimizing V while keeping U fixed. At each iteration, the algorithm approximates the objective function based on an example selected at random, updates the weight vector using the approximate subgradient, and projects the weights onto a bounded ball. Let $(u, C, i) \in S$ be an example, then the corresponding approximate objective function is

$$f^{\text{CLR}}((u, C, i); U, V) = \frac{\lambda}{2}\|V\|_F^2 + \frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).$$

We introduce various matrix notation to help us define the approximate subgradients. Given a matrix M, let $M_{k,\cdot}$ denote row k of M. Define the matrix $\hat{M}^{p,q,z}$, for $p \neq q$, as

$$\hat{M}^{p,q,z}_{s,\cdot} = \begin{cases} M_{z,\cdot} & \text{for } s = p, \\ -M_{z,\cdot} & \text{for } s = q, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

Algorithm 2 Projected stochastic subgradient descent for optimizing U.

```
Input: Factors $V \in \mathbb{R}^{n \times r}$, training data $S$, regularization parameter $\lambda$, rank constraint $r$, number of iterations $T$.
1: $U_1 \leftarrow 0^{m \times r}$
2: for all t from 1 to $T-1$ do
3:   Choose $(u, C, i) \in S$ uniformly at random.
4:   $\eta_t \leftarrow \frac{1}{\lambda t}$
5:   $C^+ \leftarrow \left\{ c \in C^i \mid \left(U_t V^T\right)_{u,i} - \left(U_t V^T\right)_{u,c} < 1 \right\}$
6:   $U_{t+1} \leftarrow (1 - \eta_t \lambda) U_t + \frac{\eta_t}{|C^i|} \sum_{c \in C^+} \check{V}^{i,c,u}$
7:   $U_{t+1} \leftarrow \min \left\{ 1, \frac{1}{\sqrt{\lambda} \|U_{t+1}\|_F} \right\} U_{t+1}$
8: return $U_T$.
```

Algorithm 3 Projected stochastic subgradient descent for optimizing V.

```
Input: Factors $U \in \mathbb{R}^{m \times r}$, training data $S$, regularization parameter $\lambda$, rank constraint $r$, number of iterations $T$.
1: $V_1 \leftarrow 0^{n \times r}$
2: for all t from 1 to $T-1$ do
3:   Choose $(u, C, i) \in S$ uniformly at random.
4:   $\eta_t \leftarrow \frac{1}{\lambda t}$
5:   $C^+ \leftarrow \left\{ c \in C^i \mid \left(U V_t^T\right)_{u,i} - \left(U V_t^T\right)_{u,c} < 1 \right\}$
6:   $V_{t+1} \leftarrow (1 - \eta_t \lambda) V_t + \frac{\eta_t}{|C^i|} \sum_{c \in C^+} \hat{U}^{i,c,u}$
7:   $V_{t+1} \leftarrow \min \left\{ 1, \frac{1}{\sqrt{\lambda} \|V_{t+1}\|_F} \right\} V_{t+1}$
8: return $V_T$.
```

and define the matrix $\check{M}_{s,\cdot}^{p,q,z}$ as

$$\check{M}_{s,\cdot}^{p,q,z} = \begin{cases} M_{p,\cdot} - M_{q,\cdot} & \text{for } s = z, \\ 0 & \text{otherwise.} \end{cases} \qquad (6)$$

Let $[\![\cdot]\!]$ denote an indicator function that is 1 if and only if its argument is true. Then, the subgradient of the approximate objective function with respect to V is

$$\nabla_V f^{\text{CLR}}\left((u, C, i); U, V\right) = \lambda V - \frac{1}{|C^i|} \sum_{c \in C^i} [\![\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} < 1]\!] \hat{U}^{i,c,u}. \qquad (7)$$

Setting $\eta_t = \frac{1}{\lambda t}$ as the learning rate at iteration t, the approximate subgradient update becomes $V_{t+1} = V_t - \eta_t \nabla_V f^{\text{CLR}}\left((u, C, i); U, V\right)$. After the update, the weights are projected onto a ball with radius $\frac{1}{\sqrt{\lambda}}$. The pseudocode for optimizing both convex subproblems is depicted in Algorithms 2 and 3. We prove the correctness of the algorithms and bound their running time in the next subsection.

We conclude the subsection by commenting on the implementation details. A naive implementation of Algorithm 2 would execute each iteration in time $\Omega(bmnr)$, where b denotes the size of the largest candidate set. One can reduce the running time of each iteration considerably by normalizing the matrix efficiently. We represent U as a triplet (W, a, ν), where a is a scalar, $U = aW$, and $\|U\|_F = \nu$. It can be verified that, using the new representation, a single iteration in both Algorithms 2 and 3 can be executed in time $O(br)$.

5.2 Analysis

In this subsection, we analyze the running time and correctness of our algorithms. In particular, we prove that projected stochastic subgradient has a running time that is independent of the number of training examples. We do not bound the total number of alternating minimization steps a priori; nevertheless, we demonstrate that our methods are especially suitable for large datasets since they solve each individual minimization problem efficiently.

The convex subproblems we analyze have the general form

$$\min_{X \in D} f(X; \ell) = \min_{X \in D} \frac{\lambda}{2} \|X\|_F^2 + \frac{1}{|S|} \sum_{(u,C,i) \in S} \ell\left(X; (u, C, i)\right). \qquad (8)$$

One can obtain the individual subproblems by specifying the domain D and the loss function ℓ. For example, in case of Algorithm 2, the corresponding minimization problem is specified by

$$\min_{X \in \mathbb{R}^{m \times r}} f(X; \ell_V) \text{ where}$$
$$\ell_V = \frac{1}{|C^i|} \sum_{c \in C^i} h\left(\left(XV^T\right)_{u,i} - \left(XV^T\right)_{u,c} \right), \qquad (9)$$

and in case of Algorithm 3, it is specified by

$$\min_{X \in \mathbb{R}^{n \times r}} f(X; \ell_U) \text{ where}$$
$$\ell_U = \frac{1}{|C^i|} \sum_{c \in C^i} h\left(\left(UX^T\right)_{u,i} - \left(UX^T\right)_{u,c} \right). \qquad (10)$$

Let $U^\star = \arg\min_U f(U; \ell_V)$ and $V^\star = \arg\min_V f(V; \ell_U)$ denote the solution matrices of Equations 9 and 10, respectively. Also, given a general convex loss ℓ and domain D, let $\bar{X} \in D$ be an ϵ-accurate solution for the corresponding minimization problem if $f\left(\bar{X}; \ell\right) \leq \min_{X \in D} f(X; \ell) + \epsilon$.

In the remainder of this subsection, we show that Algorithms 2 and 3 are adaptations of the Pegasos [13] algorithm to the CLR setting. Then, we prove certain properties that are prerequisites for obtaining Pegasos's performance guarantees. In particular, we show that the approximate subgradients computed by Algorithms 2 and 3 are bounded and the loss functions associated with Equations 9 and 10 are convex. In the end, we combine these results with previous results obtained by Shalev-Shwartz et al. [13] to show that our algorithms reach an ϵ-accurate solution with respect to their corresponding minimization problems in $\tilde{O}\left(\frac{1}{\lambda^2\epsilon}\right)$ iterations.

The main difficulty in extending the results of Shalev-Shwartz et al. [13] from the supervised learning setting to the collaborative ranking setting is the the treatment of features. In the supervised learning setting, features are constant, but in our setting, the factor matrices U and V act like features, and their values vary from one step of alternating minimization to the next. For example, when optimizing U, the factor matrix V acts like a feature matrix, and vice versa, and both matrices change in value during subsequent steps of alternating minimization. As a result, feature values that are treated as constants in the supervised setting cannot be treated as constants in the collaborative setting. We deal with these issues in our proofs, which are provided in the supplement. Below, we only state the main theorem of our analysis.

Theorem 2. *Assume that the conditions and the bound in Lemma 4 hold. Let t be an iteration index selected uniformly at random from $\{1,\ldots,T\}$. Then, with probability at least $\frac{1}{2}$,*

$$f\left(U_t;\ell_V\right) \leq f\left(U^\star;\ell_V\right) + \frac{42\left(\sqrt{\lambda}+2\sqrt{\frac{1}{\lambda}}\right)^2 \ln\left(\frac{T}{\delta}\right)}{\lambda T}.$$

The analogous result holds for Algorithm 3 as well.

Thus, with high probability, our algorithms reach an ϵ-accurate solution in $\tilde{O}\left(\frac{1}{\lambda^2\epsilon}\right)$ iterations. Since we argued in Subsection 5.1 that the running time of each stochastic update is $O(br)$, it follows that a complete run of projected stochastic subgradient descent takes $\tilde{O}\left(\frac{br}{\lambda^2\epsilon}\right)$ time, and the total running time is *independent* of the number of training examples.

6 Experiments

In this section, we provide an empirical analysis of the CLR framework by applying it to a venue recommendation task. We assess CLR's generalization and running time performance by comparing it against CofiRank, an algorithm which is representative of the state-of-the-art in collaborative ranking.

We created our dataset by collecting publicly available "check-ins" originating from New York City via the Twitter and Foursquare APIs. A check-in is a virtual announcement where a user shares her whereabouts with other people on her social network. Our dataset ranged over a 9 month period and consisted of 13750 users, 11700 venues, and 269597 check-ins. We note that all the check-ins we collected were shared with the entire Internet, so our dataset does not contain any private information.

At a high level, our goal is to build a venue recommendation application for mobile platforms, where the user specifies a geographical region she is interested in exploring, and the application returns a personalized ranking over the venues in that region. We use the Foursquare dataset as a proxy, and assume that when a user checks into a venue, she is implicitly preferring that venue over the remaining venues in a specified radius. We also note that many of the publicly available collaborative filtering datasets, such as the Netflix dataset, are not appropriate for our problem, since they usually do not contain enough information for us to reconstruct the candidate sets.

We partition the check-ins chronologically, and place the earliest 60% of check-ins to train, the subsequent 20% to validation, and final 20% to test sets. Each check-in corresponds to a tuple (u,i), where u is the user and i is the checked-in venue. In order to apply CLR, we augment each check-in with a candidate set C, which consists of venues within a specified distance to i, and form the CLR sample (u,C,i). We train a variety of CLR and CofiRank models on a combined train and validation set, determine the best performing model parameters using the validation set, and report the performance of the chosen parameters on the test set. The model parameters include the rank $r \in \{2,5,10,20\}$, CofiRank regularization parameter $\lambda \in \{10,100,1000,10000\}$, and CLR regularization parameter $\lambda \in \{0.01,0.001,0.0001,0.00001,0.000001\}$. For testing, we form queries with a user u and a candidate set C, hide the label i, and measure the algorithms' performances accordingly. We vary the radius that determines the size of the candidate set C from 50m to 300m, in 50m increments.

Figure 1 displays the local ranking loss of various algorithms on the held-out test set. We compare CLR against both CofiRank and an algorithm called Popular, which outputs the most frequently checked in venue from the candidate set. We execute CofiRank with three different loss functions: ordinal, NDCG, and squared [16]. CofiRank performs best when coupled with the squared loss, where it effectively mimics MMMF [10]. Our algorithm, CLR, outperforms all methods and achieves the lowest local ranking loss. We note that, unlike competing methods, CLR does not suffer an additional loss when the size of its candidate

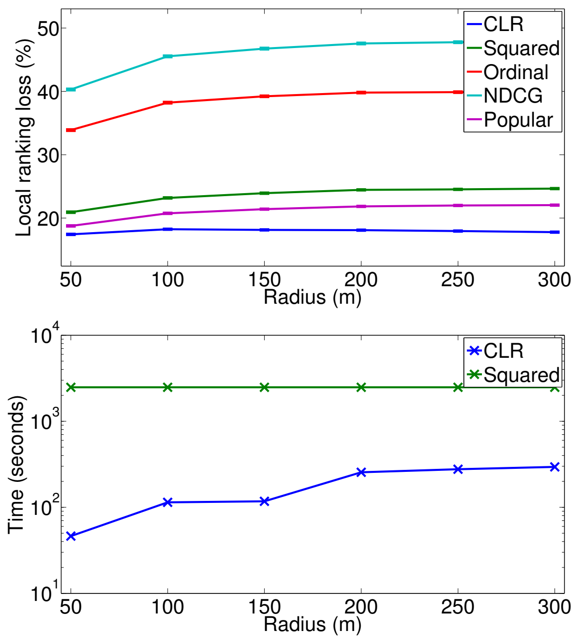

Figure 1: **Top:** The local ranking loss of various algorithms on the held-out test set. CLR outperforms all competing methods. The error bars correspond to standard error and are barely noticeable. **Bottom:** Training time of CLR and CofiRank in log-scale. CLR trains orders of magnitude faster than CofiRank-squared, the best performing variant of CofiRank, even when we increase the size of the radius and candidate sets.

sets increases.

Figure 2 displays the recall@k performance on the held-out test set, for $k = \{1, 5, 10\}$. The order of algorithms with respect to recall performance is exactly the same as their order with respect to local ranking loss. For recall@1, Popular outperforms most versions of CofiRank, but as the radius of candidate sets increase beyond 150m, the performance of CofiRank-squared matches that of Popular. As k increases, the performance gap between CLR and the remaining methods diminishes, especially for lower radii, but CLR continues to outperform for higher radii. Unlike CLR, the remaining collaborative ranking methods do not take advantage of the local feedback, which leads to poor performance.

We also empirically analyze the running times of CofiRank and CLR. For both algorithms, for each radius, we choose the parameter settings that perform the best on the validation set, and record their respective training times. We plot the results using log-scale in Figure 1. Even though the running time of CLR has a linear dependence on the size of the candidate sets, CLR trains orders of magnitude faster than CofiRank, as suggested by our theoretical analysis in Subsection 5.2.

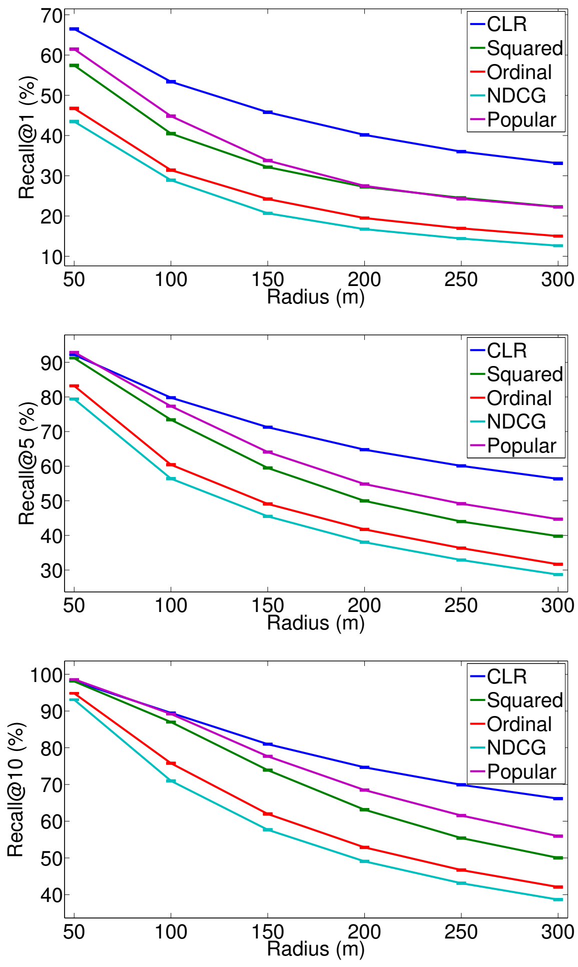

Figure 2: Recall@k of various algorithms on the held-out test set, for $k = \{1, 5, 10\}$. CLR outperforms all competing methods. The error bars correspond to standard error and are barely noticeable.

7 Conclusion

In this paper, we formulated a novel matrix factorization framework, called Collaborative Local Ranking, which allows us to formulate a new set of real-world ranking tasks in a collaborative setting. We justified CLR with a bound on its generalization error, which to the best of our knowledge, is the first bound of its kind for collaborative ranking. We also derived a simple alternating minimization algorithm and showed that each minimization step can be efficiently computed in time independent of the number of training examples. We applied CLR to a venue recommendation task and demonstrated that it outperforms state-of-the-art collaborative ranking methods, such as CofiRank, both in terms of generalization performance and running time.

References

[1] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM.

[2] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.

[3] Stéphane Boucheron, Olivier Bousquet, and Gabor Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

[4] Rina Foygel, Ruslan Salakhutdinov, Ohad Shamir, and Nati Srebro. Learning with the weighted trace-norm under arbitrary sampling distributions. In John S. Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, Kilian Q. Weinberger, John S. Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 2133–2141, 2011.

[5] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[6] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 5, pages 145–186. Springer US, Boston, MA, 2011.

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

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

[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars S. Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[10] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA, 2005. ACM.

[11] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 25, 2008.

[12] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[13] Shai S. Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, March 2011.

[14] Nathan Srebro, Noga Alon, and Tommi Jaakkola. Generalization error bounds for collaborative prediction with Low-Rank matrices. In *NIPS*, 2004.

[15] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola. Maximum-Margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, volume 17, pages 1329–1336, 2005.

[16] Markus Weimer, Alexandroos Karatzoglou, and Alex Smola. Improving maximum margin matrix factorization. In *ECML*, 2008.

[17] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.

[18] Shuang H. Yang, Bo Long, Alexander J. Smola, Hongyuan Zha, and Zhaohui Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 295–304, New York, NY, USA, 2011. ACM.