Active Collaborative Permutation Learning

Jialei Wang
Dept. of Computer Science
University of Chicago
jialei@uchicago.edu

Nathan Srebro
Toyota Technological Institute
at Chicago
nati@ttic.edu

James A. Evans
Dept. of Sociology
University of Chicago
jevans@uchicago.edu

ABSTRACT

We consider the problem of Collaborative Permutation Recovery, i.e. recovering multiple permutations over objects (e.g. preference rankings over different options) from limited pairwise comparisons. We tackle both the problem of how to recover multiple related permutations from limited observations, and the active learning problem of which pairwise comparison queries to ask so as to allow better recovery. There has been much work on recovering single permutations from pairwise comparisons, but we show that considering several related permutations jointly we can leverage their relatedness so as to reduce the number of comparisons needed compared to reconstructing each permutation separately. To do so, we take a collaborative filtering / matrix completion approach and use a trace-norm or max-norm regularized matrix learning model. Our approach can also be seen as a collaborative learning version of Jamieson and Nowak's recent work on constrained permutation recovery, where instead of basing the recovery on known features, we learn the best features de novo.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Theory, Algorithms, and Experimentation

Keywords

Collaborative Ranking; Active Learning; Matrix Factorization

1. INTRODUCTION

Recovering permutations or rankings from pairwise comparisons is an extensively studied problem with wide applications in information retrieval, knowledge discovery and machine learning. The standard setup is that of recovering a *single* permutation π over m objects based on information of the sort "A appears before B in the permutation". It is well known that if comparisons are indeed consistent with some underlying permutation (i.e. there are no discrepancies or errors) then $\Theta(m^2)$ random queries or $\Theta(m \log m)$ adaptively chosen queries are sufficient for recovering the permutation (this is essentially a sorting problem). Also when dealing with noisy comparisons, or when the comparisons are not transitive and consistent with some underlying permutation, methods are available for reconstructing the best consensus permutation from both random and adaptively chosen queries [2, 21].

But in many scenarios, we would like to recover multiple related permutations. Consider for example many people, each with their own tastes or preferences. We could try to recover a single permutation that best tries to explain the consensus among the people, treating comparisons made by different people as not-entirely-consistent noisy comparisons. This approach, seems absurd in the context of survey research, because it would fail to capture individual preferences and miss out entirely on the diversity of preferences in the population.

Alternatively, we could independently learn a separate permutation for each person. While this corresponds to the most common approach to imputing missing survey data, it breaks down in the context of sparse or much missing data. If we can only obtain a limited number of comparisons from each user, perhaps even less than the number of items m, this might not be enough to recover the permutations when each users' ranking is considered separately. By contrast, in this paper, we investigate the problem of *collaborative permutation learning*. That is, of jointly recovering multiple related, but not identical, permutations based on pairwise comparisons from different permutations. We will show through theoretical and empirical demonstrations that the solution of these problems could widen the application of learning with permutations to the design of novel, intelligent surveys that only require respondents to rank a few items in order to approximately recover their entire preference ranking. This would allow social and information science researchers to rapidly identify individually preferred products, services, opinions, policies and solutions, without wasting questions on answers that are likely predictable.

This type of collaborative ranking survey has become common for organizations that seek to identify preferred solutions to problems. Consider the survey platform "All our ideas" [1] in which individuals and organizations can field surveys or "idea marketplaces" of precisely the type we describe here. Individual respondents are posed with a question and an associated pair of possible responses. Respondents can either add a new response, or compare the presented pair. For example, in 2011, New York City Mayor Michael Bloomberg's office fielded such a survey that posed the question "Which do you think is better for creating a greener, greater New York City?" with possible responses including "open schoolyards across the city as public playgrounds", "increase targeted tree plantings in neighborhoods with high asthma rates", and "keep

[1] http://allourideas.org

KDD '14, August 24-27, 2014, New York City, USA.

http://dx.doi.org/10.1145/2623330.2623730.

NYC's drinking water clean by banning fracking in NYC's watershed" [25]. Collaborative ranking has been used in other academic projects to "crowdsource" common knowledge and impressions about the world. Consider the 2013 article, "the Collaborative Image of the City"[24], which employed a collaborative ranking survey through which thousands of respondents ranked pairs of urban street images from the United States and Europe to determine urban areas of more and less perceived safety and prosperity. In all of this research, however, comparison pairs are chosen randomly and are densely annotated by many respondents. Our research could dramatically improve the efficiency of these efforts.

More formally, we consider the following setup: we have n people and would like to learn for each person i a permutation π_i over m items. The information we obtain is through pairwise comparisons of the form "user i prefers item a over item b", i.e. that a appears before b in π_i ($\pi_i(a) > \pi_i(b)$). However, these pairwise comparisons might be noisy, or only partially consistent with some underlying permutation. Furthermore, we assume the permutations π_i are all related in some way, and modeling this relationship is part of our contribution here. In any case, based on such, possibly noisy and non-transitive, pairwise comparisons, our goal is to recover $\pi_1, \ldots, \pi_n$. We also consider the "active" or "adaptive" setup where we can actively choose which queries to make, i.e. at each step we can choose a person i and a pair of items (a, b), and query user i as to whether the person prefers a over b (i.e. ask whether $\pi_i(a) < \pi_i(b)$ or $\pi_i(b) < \pi_i(a)$). The two questions we thus ask, and suggests answers for, are: (1) how can we recover the permutations for every person based on these pairwise preferences, and, (2) how should we adaptively choose queries so as to make our predictions more accurate and reduce the overall number of queries required.

In order to tackle the problem of collaborative permutation learning, we formulate the problem as a matrix completion problem with pairwise constraints, and use standard matrix-factorization regularizers, such as the trace-norm or max-norm, to reconstruct the matrix based on limited pairwise constraints. In Section 4 we discuss this matrix factorization approach, its underlying assumption about the relatedness of the permutation, and its relationship to the constrained single-permutation learning approach of [12]. We then leverage the existing understanding of methodology on matrix factorization regularizers to suggest efficient optimization methods for fitting our suggested models (Section 5) and to analyze the number of queries required for recovery—in Section 7 we show that under our model, $O(n+m)$ random queries are sufficient for approximate recovery, significantly less than the $O(nm)$ required for approximate recovery if learning each permutation independently. In order to further reduce this query complexity we turn to active learning and suggest an adaptive query heuristic based on biasing the queries towards more ambiguous pairs (Section 6).

Notation Let $[n] = \{1, 2, ..., n\}$. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, let $\mathbf{X}_{i,j}$ be the (i, j)-th element; $\mathbf{X}_{i,\cdot}$ be the i-th row of $\mathbf{X}$, $\|\mathbf{X}\|_F = \sqrt{\sum_{i\in[n]} \sum_{j\in[m]} \mathbf{X}^2_{i,j}}$ be the Frobenius norm, $A \circ B = \text{tr}(A^T B)$ be the matrix inner product of A and B, and denote $A \succeq 0$ to mean A is positive-semi-definite.

2. RELATED WORK

Rank aggregation from pairwise comparisons.

There has been much work on constructing a permutation based on pairwise comparisons. If the comparisons are transitive, that is consistent with some global permutation, then outputting this global permutation given enough pairwise comparisons is easy. Once the comparisons are noisy, or are not fully consistent, the task becomes trickier and one must both decide on a criteria for measuring the quality of a permutation (i.e. its degree of agreement with the comparisons) and devise an algorithm for finding the optimal permutation under this criteria. Some criteria and algorithms suggested for the problem include Borda Count [7], Rank Centrality [17], HodgeRank [13], Balanced Rank Estimation [31], and others. There are, however, two major differences between rank aggregation and the problem we pose here. For rank aggregation, it is typically assumed that: 1) there is only one globally optimal ranking, while here we assume each user holds a personal ranking; and 2) there are dense observations, i.e., with all, or at least many, pairwise comparisons being observed, while in our setting the observations are usually sparse, with only a very small fraction of pairwise preferences revealed. In particular, there is no hope of obtaining a meaningful global permutations with less than $m-1$ comparisons, as even for perfectly consistent data, this does not enable linking relationships among all m items (more typically, in rank aggregation one has $O(m^2)$ comparisons).

Active learning for ranking.

Beyond the problem of rank aggregation given dense static data, there has also been interest in adaptive query strategies for learning the optimal distribution. Again, if all comparisons are guaranteed to be transitive and so consistent with some permutation, the problem is easy and boils down to sorting by comparisons, and can the permutation can be recovered using $O(m \log m)$ comparisons, e.g. using merge-sort. When the comparisons are not necessarily consistent, one might wish to obtain the most agreeing (optimal) permutation for all $\binom{m}{2}$ possible comparisons, but without actually making all these comparisons. I.e. the assumption is a user holds a system of pairwise preferences that is not necessarily consistent with a permutation, and we would like to uncover the most consistent permutation by making the least amount of pairwise queries. Recently, [1] showed that this problem too can be solved using only $O(m\text{poly}\log m)$ adaptively chosen queries. Furthermore, if we are willing to tolerate discrepancies in ϵ fraction of comparisons, relative to the optimal permutation, we need only $O(m\text{poly}\log 1/\epsilon)$ queries. This is in contrast to the $O(m/\epsilon^2)$ random queries required for solving the same task. This demonstrates the power of active learning in ranking, but is still limited to learning a single independent permutation.

(Active) Collaborative filtering.

The problem of collaborative ranking, as we consider it here, is related to widely studied problem of *collaborative filtering* as formalized through *matrix completion*, where we observe some ratings of items by users and would like to use these to predict the ratings of users on items they have not yet rated. Matrix factorization approaches have long been used collaborative filtering problems [11, 23], including using the trace-norm and max-norm [27] as well as other related matrix factorization regularizers. Active queries have also been investigated in this context, with heuristics suggested based on the *expected value of information*(EVIQ) [5] or the prediction margin [22], as well as using a Bayesian approach [14].

The main difference between standard collaborative filtering and the problem studied here, is that in the standard setting single-item ratings are observed, providing absolute information about a single entry in the unknown matrix. In contrast, we consider learning from pairwise comparisons, where we can only obtain relative information comparing two items. Our goal is also different, in that instead of seeking accuracy of specific ratings, we seek to capture a permutation for each user. Recent research [3, 32, 30] has also

studied the problem of ranking in a collaborative filtering setup, but the objective of this work has been to optimize global ranking measures, such as NDCG or MAP. Moreover, their inputs are still rating scores, rather than pairwise information.

3. BACKGROUND: CONSTRAINED PERMUTATION LEARNING

In order to reduce the query complexity of ranking from pairwise comparisons, and allow learning permutations with less than m comparisons, we must use some external information and make assumptions on the permutation to be learned. [12] suggest associating with each item a a feature vector $v_a \in \mathbb{R}^d$ which encodes our prior information about a. Thinking of the feature vectors v_a as points in $\mathbb{R}^d$, a permutation is then specified as a direction in $\mathbb{R}^d$, where the rank order of items is given by their order when projected to this direction. Representing the direction in space as a vector $u \in \mathbb{R}^d$, the permutation π_u associated with u is defined by $\pi_u(a) < \pi_u(b)$ iff $\langle u, v_a \rangle < \langle u, v_b \rangle$ (we assume that u is in general position relative to the vectors v, i.e. that for no two items a, b, $\langle u, v_a \rangle = \langle u, v_b \rangle$). Alternatively, one can think of u as specifying a point in $\mathbb{R}^d$ and order items according to their distances from u: $\pi_u(a) < \pi_u(b)$ iff $\|u - v_a\| < \|u - v_b\|$. Ordering by the norm or by the projection are equivalent if u and all vectors v are normalized (i.e. on the unit sphere), and otherwise the difference between them amounts to adding one additional dimension. Although [12] focused mostly on ordering by distance, here it will be more convenient for us to order by projection.

In either case, for a given feature map, only a subset of permutations can be represented this way, allowing a significant reduction in the query complexity if we focus only on such permutations—[12] showed how $O(d \log m)$ adaptive queries are enough for learning the optimal permutation (among permutations of this form).

One can think of the feature map as specifying an embedding of items into a possible "preference space", with different axes specifying different attributes one might prefer or not prefer, and a direction u in this space specifying the preferences over these attributes, which in turn defines the preferences over items. The query complexity is then proportional to the dimensionality, or number of "attributes", d, instead of to the number of items m.

But the approach of [12] is still limited to a learning a single permutation at a time. Furthermore, the reduced query complexity relies on external information in the form of the feature embedding v_a. [12] did consider scenarios with individualized permutations, where each of many users has a different permutation over items (e.g. beer preferences in their application). But their proposed approach was to first obtain good features for each item using some external information source (in their case, the text of product reviews and descriptions) and then learn the permutation π_i for each user i separately, using only pairwise comparisons by this user i, based on the fixed feature maps. Here, we would like to avoid using a pre-determined feature map based on external information, and instead learn this map de novo by considering all users jointly, and leveraging information gleaned from one user's comparisons to improve the ranking of other users.

4. MATRIX COMPLETION BASED CPL

As in the model discussed above, we associate with each item a a vector $v_a \in \mathbb{R}^d$ and with each user a vector $u_i \in \mathbb{R}^d$, such that the permutation for user i is specified by $\pi_i(a) < \pi_i(b)$ iff $\langle u_i, v_a \rangle < \langle u_i, v_b \rangle$. However, instead of basing the model on prespecified feature vectors v_a, we do not assume any prior knowledge on the items. Rather, following a collaborative approach, we jointly learn both the user vectors u_i *and* and item vectors v_a. Since the item vectors v_a are learned, the permutation for any single user is *not* constrained, unlike [12]. However, having observations on multiple users constrains the possible setting of the item vectors v_a and thus constraints the *relationship* between the multiple permutations. In other words, based on the observed comparisons from multiple users, we are learning the population space of possible permutations, which in turn allows us to learn individual permutations with significantly fewer observations.

$\pi_i(a) < \pi_i(b)$ iff $\mathbf{X}_{i,a} < \mathbf{X}_{i,b}$. Requiring that X decomposes as $X = UV^T$ of the appropriate dimensions is equivalent to requiring that it has rank at most d. We can thus consider performing collaborative permutation learning as searching for a low-rank matrix $\mathbf{X}$ such that $\mathrm{sgn}(\mathbf{X}_{i,a} < \mathbf{X}_{i,b})$ matches our observed pairwise comparisons, or at least matches as many as possible to our observed pairwise comparisons.

However, as was suggested by [9, 27], we instead consider an infinite-dimensional model, where the dimensionality d is unbounded (one can think of $U_{a,\cdot}$ and $V_{b,\cdot}$ as vectors in an infinite-dimensional Hilbert space, or simply allow them to be vectors in an arbitrarily high dimensional space), where we instead constrain the *norms* $\|U_{a,\cdot}\|_2, \|V_{b,\cdot}\|_2$. Constraining the average squared-norm of these vectors corresponds to constraining the *trace-norm* (aka nuclear norm) of X, which is defined as[2]:

$$\|X\|_\Sigma = \min_{X=UV^T} \frac{1}{2}\left(\sum_{a=1}^{n} \|U_{a,\cdot}\|_2^2 + \sum_{b=1}^{m} \|V_{b,\cdot}\|_2^2\right). \quad (1)$$

Similarly, constraining the norm of all vectors (i.e. constraining the maximal norm) corresponds to constraining the *max-norm* of X, defined as:

$$\|X\|_{\max} = \min_{X=UV^T} \max\left(\max_a \|U_{a,\cdot}\|_2^2, \max_b \|V_{b,\cdot}\|_2^2\right). \quad (2)$$

Both the max-norm and the trace-norm can be thought of either as convex relaxations to the rank [10], or as more refined models, allowing an infinite-dimensional embedding, regularized through norm-regularization rather than a parametric constraint as in, e.g., Support Vector Machines [27]. Either way, in order for the norm-regularization to be meaningful, we must require not only that $X_{i,a} > X_{i,b}$ whenever we observe $\pi_i(a) > \pi_i(b)$, but that the inequality holds with a *margin*. And in order to allow for noisy or inconsistent observations, instead imposing a hard margin constraint, we seek to minimize an empirical *hinge loss*, defined below, which penalizes the extent of margin violations.

Let $\mathcal{S} : (i_1, a_1, b_1), (i_2, a_2, b_2), ..., (i_{|\mathcal{S}|}, a_{|\mathcal{S}|}, b_{|\mathcal{S}|})$ be the set of observed pairwise preferences, where (i, a, b) represents the comparison of user i prefers item a over b. $|\mathcal{S}|$ is the total set of observed pairwise comparisons. The empirical hinge loss of X is then defined as:

$$\hat{L}_{\text{hinge}}(\mathbf{X}) = \frac{\sum_{(i,a,b)\in\mathcal{S}} \max(1 - (\mathbf{X}_{i,a} - \mathbf{X}_{i,b}), 0)}{|\mathcal{S}|}$$

And our training objectives, with the trace-norm and max-norm respectively are:

$$\min_{\mathbf{X}} \|\mathbf{X}\|_\Sigma + \lambda \hat{L}_{\text{hinge}}(\mathbf{X}) \quad \text{and} \quad \min_{\mathbf{X}} \|\mathbf{X}\|_{\max} + \lambda \hat{L}_{\text{hinge}}(\mathbf{X}) \quad (3)$$

where λ is a regularization tradeoff parameter.

Both the trace-norm and max-norm are semi-definite-representable [9, 27] and thus both problems above can be rewritten and solved as semi-definite programs (SDP). This allows us to use standard SDP

[2]The trace-norm of X is also equal to the sum of the singular values of X, but we prefer thinking in terms of the matrix-factorization characterization of the trace-norm

solvers. However, in order to handle large-scale problem, special-purpose first-order optimization methods are required. Fortunately, in the past few years, there has been much progress in developing such methods for both trace-norm and max-norm regularized problems, and in the next Section we describe the methods we use here.

5. LARGE SCALE OPTIMIZATION

In this Section, we describe the accelerated first order methods we use to solve the optimization problems (3) and fit our models. For both the trace-norm and the max-norm we adapt recently proposed accelerated proximal methods.

5.1 Accelerated proximal gradient methods for trace-norm regularized CPL

For trace-norm regularized collaborative permutation learning, we use an accelerated version of Singular Value Thresholding (SVT): SVT optimization [6] consists of iterative updates corresponding to the optimization of a partial linearization of the objective function, where loss is linearized but the regularizer is not:

$$\begin{aligned}\mathbf{X}_k &= \arg\min_{\mathbf{X}} \hat{L}_{\text{hinge}}(\mathbf{X}_{k-1}) + (\mathbf{X}-\mathbf{X}_{k-1}) \circ \nabla \hat{L}_{\text{hinge}}(\mathbf{X}_{k-1}) \\ &+ \frac{1}{2\eta_k}\|\mathbf{X}-\mathbf{X}_{k-1}\|_F^2 + \frac{1}{\lambda}\|\mathbf{X}\|_\Sigma\end{aligned}$$

Such an update can be performed by soft-thresholding the singular-values of $\mathbf{X}_{k-1} - \eta_k \nabla \hat{L}_{\text{hinge}}(\mathbf{X}_{k-1})$, requiring a singular value decomposition at each iteration [6]. For a smooth loss function, such updates can be *accelerated* by combining two sequences of iterates $\mathbf{X}_k$ and $\mathbf{Z}_k$, as in [18, 20]. Although our objective function is not smooth, empirically we found that accelerated gradient methods can usually obtain better convergence than simple gradient descent for our problem. In particular, each iteration consists of the following updates:

$$\begin{aligned}\mathbf{X}_k &= \text{SVT}_{\frac{1}{2\lambda}}(\mathbf{Z}_k - \eta_k \nabla \hat{L}_{\text{hinge}}(\mathbf{Z}_k)) \\ \mathbf{Z}_{k+1} &= \mathbf{X}_k + (\frac{\alpha_k - 1}{\alpha_{k+1}})(\mathbf{X}_k - \mathbf{X}_{k-1})\end{aligned}$$

where the SVT operator is defined as: $\text{SVT}_\lambda(\mathbf{X}) = U\Sigma_\lambda V^T$, where $\mathbf{X} = U\Sigma V^T$ is the singular value decomposition of $\mathbf{X}$, and $(\Sigma_\lambda)_{i,i} = \max\{0, \Sigma_{i,i} - \lambda\}$, and α_k is the parameter recursively defined as $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$ with $\alpha_1 = 1$.

5.2 Proximal iterative smoothing algorithm for max-norm regularized CPL

For the max-norm, we cannot take this approach, because the minimum of a quadratic function plus a max-norm regularizer is *not* given analytically in terms of the SVD. Instead, we take the proximal iterative smoothing (PRISMA) technique proposed by [19], where the goal is to solve a convex optimization which decomposes into three parts: a smooth part, a simple Lipschitz part, and a non-Lipschitz part. Although the pairwise hinge loss functions in max-norm regularized CPL problems are not smooth, we can also apply PRISMA to our problem because our objective function is

$$\min_{\mathbf{X}} \|\mathbf{X}\|_{\max} + \lambda \hat{L}_{\text{hinge}}(\mathbf{X})$$

. This problem can be rewritten as:

$$\min_{\mathbf{X},A,B} \frac{1}{\lambda} \max \text{diag} \begin{bmatrix} A & \mathbf{X} \\ \mathbf{X}^T & B \end{bmatrix} + \hat{L}_{\text{hinge}}(\mathbf{X}) + \delta_{\mathbb{S}_+^{m+n}} \begin{bmatrix} A & \mathbf{X} \\ \mathbf{X}^T & B \end{bmatrix}$$

where $\delta_{\mathbb{S}_+^{m+n}}$ is the indicator function of set of $(m+n) \times (m+n)$ positive semi-definite matrices (zeros inside $\mathbb{S}_+^{m+n}$ and infinite outside).

To apply PRISMA for this problem, we learn $A, B, \mathbf{X}$ simultaneously as a $(m+n) \times (m+n)$ matrix (we denote it as $\mathbf{Z}$): at each step, we first perform a gradient descent step on the function $\hat{L}_{\text{hinge}}(\mathbf{X})$, then add the proximal operator solution of the function $\frac{1}{\lambda} \max \text{diag}(\mathbf{Z})$, and then perform a proximal operator of the function $\delta_{\mathbb{S}_+^{m+n}} \mathbf{Z}$. Note that the proximal operator of $\max \text{diag}(\mathbf{Z})$ is equivalent to the proximal operator of the $\|\cdot\|_\infty$ on the diagonal vector of the matrix $\mathbf{Z}$, which has closed-form solution [8], and the proximal operator of $\delta_{\mathbb{S}_+^{m+n}} \mathbf{Z}$ can be solved by setting all negative eigenvalues of $\mathbf{Z}$ to zero. Thus the core updating steps are:

$$\begin{aligned}\mathbf{Y}_k &= \text{EVT}((1 - \frac{1}{\lambda\beta_k})\mathbf{Z}_k - \eta_k \nabla \hat{L}_{\text{hinge}}(\mathbf{Z}_k) + \frac{1}{\lambda\beta_k}\text{DVT}_{\beta_k}(\mathbf{Z}_k)) \\ \mathbf{Z}_{k+1} &= \mathbf{Y}_k + (\frac{\alpha_k - 1}{\alpha_{k+1}})(\mathbf{Y}_k - \mathbf{Y}_{k-1})\end{aligned}$$

where DVT is the diagonal value thresholding operator: $\text{DVT}_{\beta_k}(\mathbf{X})_{i,i} = \text{sgn}(\mathbf{X}_{i,i}) \min(\mathbf{X}_{i,i}, \beta_k)$ and $\text{DVT}_{\beta_k}(\mathbf{X})_{i,j} = \mathbf{X}_{i,j}$ for all $i \neq j$; and EVT is the Eigenvalue thresholding operator obtained by setting all negative eigenvalues to zero.

6. ACTIVE LEARNING STRATEGIES

In this section we discuss some active learning strategies for collaborative permutation learning. Although our goal is to create a system that actively makes queries to all possible pairs, for presentation clarity we consider the setting of pool-based batch mode active learning [26]. Suppose at learning stage t, we already have our learned model $\mathbf{X}^t$, as well as a candidate pool $\mathcal{P}_t = \{(i,a,b)\}$. We then need to ask users to reveal their comparison labels for a small batch of instances $\mathcal{Q}_{t+1} \subseteq \mathcal{P}_t$, and add these new training data to our current training set $\mathcal{S}_t$. Then we re-train the model using the combined training set $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \mathcal{Q}_{t+1}$ to obtain improvement.

The baseline solution is uniform sampling where we randomly sample instances in $\mathcal{P}_t$ uniformly. We can also adopt some active ranking methods for single permutation learning in our setting: as inspired by [12], which actively query only the "ambiguous" pairs for labeling. "Ambiguous" is defined as the pair that can not be imputed from known pairwise comparisons based on the consistency assumption. (Suppose we know "a" is preferred to "b", "b" is preferred to "c"; pair "a" and "c" is not an "ambiguous" pair because we can transitively impute that "a" is preferred than "c"). We can use this approach in our setting: we uniformly sample "ambiguous" pairs for all users independently.

For our proposed approaches, the margin provides important information about the uncertainty of our learned model on the data. [29] proposed to actively query the instances with smallest margin for Support Vector Machines (SVMs), and [22] adopted this strategy and showed it's effectiveness for maximum-margin matrix factorization [27]. Likewise, we can use similar ideas to select pairs with the smallest difference in their estimated scores: $f_t(i,a,b) = |\mathbf{X}_{i,a}^t - \mathbf{X}_{i,b}^t|$.

Since our goal is to interactively estimate the model from data, when the model is inaccurate, the margin information could be misleading. As a result, we propose a stochastic sampling alternative. Given a temperature parameter T_t, we randomly sample query instances with probability proportional to $p_{(i,a,b)} = e^{-\frac{|\mathbf{X}_{i,a}^t - \mathbf{X}_{i,b}^t|}{T_t}}$. Thus we favor instances with smaller margin. At the beginning, we set the temperature high, which means we select instances tending to uniform randomness, and then we decrease the temperature during the learning process. In this way, we are more confident about our model and bias toward more of the instances with small margin. 1 described the detailed process of our proposed "CPL-Margin Sampling" algorithm.

Algorithm 1 CPL-Margin Sampling — Algorithm of active collaborative permutation learning from pairwise comparisons

Input: $\mathcal{P}_0 \in \mathbb{N}^{|\mathcal{P}|\times 3}$: the set of candidate pairwise preferences.
Initialization: The initialized model X^0 at time 0.
for $t = 1, 2, \ldots$ **do**
  Computing the margin score achieved by current model.
  Sampling k triplets from $\mathcal{P}_{t-1}$ with probability proportional to $e^{-\frac{|\mathbf{x}_{i,a}^{t-1} - \mathbf{x}_{i,b}^{t-1}|}{T_{t-1}}}$ to form the query set $\mathcal{Q}_t$, update the training set $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \mathcal{Q}_t$.
  Learn the new $\mathbf{X}^t$ by solving the problem (3).
end for

7. THEORETICAL ANALYSIS

In this section we provide some theoretical analysis of the proposed algorithms. Suppose the true permutations are generated by the underlying matrix $\mathbf{X}^*$, we can define the expected loss for any matrix $\mathbf{X}$:

$$L(\mathbf{X}) = \frac{2\sum_{i=1}^{n}\sum_{a=1}^{m}\sum_{b\neq a}^{b\in[m]} \mathbb{I}_{\text{sgn}(\mathbf{x}^*_{i,a} - \mathbf{x}^*_{i,b}) \neq \text{sgn}(\mathbf{x}_{i,a} - \mathbf{x}_{i,b})}}{nm(m-1)}$$

and the empirical loss for matrix $\mathbf{X}$:

$$\hat{L}(\mathbf{X}) = \frac{1}{|\mathcal{S}|}\sum_{(i,a,b)\in\mathcal{S}} \mathbb{I}_{\mathbf{x}_{i,a} < \mathbf{x}_{i,b}}$$

and the expected hinge loss for matrix $\mathbf{X}$:

$$L_{\text{hinge}}(\mathbf{X}) = \frac{2\sum_{i=1}^{n}\sum_{a=1}^{m}\sum_{b\neq j}^{b\in[m]} l_{\text{hinge}}(i,a,b)}{nm(m-1)}$$

where $l_{\text{hinge}}(i,a,b) \max(1 - (\mathbf{X}_{i,a} - \mathbf{X}_{i,b}), 0)$ and the empirical hinge loss for matrix $\mathbf{X}$:

$$\hat{L}_{\text{hinge}}(\mathbf{X}) = \frac{1}{|\mathcal{S}|}\sum_{(i,a,b)\in\mathcal{S}} \max(1 - (\mathbf{X}_{i,a} - \mathbf{X}_{i,b}), 0)$$

7.1 Generalization bounds of CPL

The following theorems give generalization guarantees for the proposed collaborative permutation learning models.

THEOREM 1. *Let $\mathcal{X}^{\max}[A]$ be the set of matrices that with bounded max-norm A. Then for any $\delta > 0$, with probability at least $1-\delta$ over the choice of a sample $\mathcal{S}$, for every $\mathbf{X} \in \mathcal{X}^{\max}[A]$, the following holds:*

$$L(\mathbf{X}) \leq \hat{L}_{hinge}(\mathbf{X}) + 48\sqrt{\frac{A^2(n+m)}{|\mathcal{S}|}} + 3\sqrt{\frac{\ln\frac{2}{\delta}}{2|\mathcal{S}|}}$$

Proof Sketch. We treat the matrix $\mathbf{X}$ as a function $[n] \times [m] \to \mathbb{R}$, and use existing bounds on the Rademacher complexity [4] of matrices with bounded trace-norm or max-norm [28, 10]. The main difference here is that the instances, and hence the loss, depend on *two*, rather than only *one* evaluation of $\mathbf{X}$. Nevertheless, this only results in an increase by a factor of two in the Rademacher complexity of the loss class, and we can use standard arguments for obtaining uniform concentration and generalization guarantees as a function of the Rademacher complexity.

And we get the following generalization bound for trace-norm based methods:

THEOREM 2. *Let $\mathcal{X}^{\Sigma}[A]$ be the set of matrices with bounded trace-norm A. Then there exists a constant K, for any $\delta > 0$, with probability at least $1-\delta$ over the choice of a sample $\mathcal{S}$, for every $\mathbf{X} \in \mathcal{X}^{\Sigma}[A]$, the following holds:*

$$L(\mathbf{X}) \leq \hat{L}_{hinge}(\mathbf{X}) + 4K\sqrt{\frac{\frac{A^2}{nm}(n+m)\ln n}{|\mathcal{S}|}} + \sqrt{\frac{\ln\frac{1}{\delta}}{2|\mathcal{S}|}}$$

7.2 Sample complexity for approximate recovery

COROLLARY 3. *Suppose $\mathbf{X}^*$ is rank-r with bounded entries, let $\mathbf{X}_{\max}(\mathcal{S})$ be the max-norm regularized empirical loss minimizer: $\mathbf{X}_{\max}(\mathcal{S}) = \arg\min_{\mathbf{X}\in\mathcal{X}^{\max}[\sqrt{r}]} \hat{L}_{hinge}(\mathbf{X})$, with the sample size $|\mathcal{S}| \geq \frac{4608r(n+m)+18\ln\frac{2}{\delta}}{\epsilon^2}$, with probability at least $1-\delta$, we have*

$$L(\mathbf{X}_{\max}(\mathcal{S})) \leq \hat{L}_{hinge}(\mathbf{X}^*) + \epsilon$$

Proof Sketch. By error decomposition $L(\mathbf{X}_{\max}(\mathcal{S})) = L(\mathbf{X}_{\max}(\mathcal{S})) - \hat{L}_{\text{hinge}}(\mathbf{X}_{\max}(\mathcal{S})) + (\hat{L}_{\text{hinge}}(\mathbf{X}_{\max}(\mathcal{S})) - \hat{L}_{\text{hinge}}(\mathbf{X}^*)) + \hat{L}_{\text{hinge}}(\mathbf{X}^*)$, combining the fact of empirical loss minimizer, as well as basic mean inequalities.

Remark. If we further assume that the true scoring matrix suffers 0 hinge loss, i.e., that the following realizable condition holds: $\hat{L}_{\text{hinge}}(\mathbf{X}^*) = 0$, we get $L(\mathbf{X}_{\max}(\mathcal{S})) \leq \epsilon$, which means we obtain approximate recovery of all the permutations with ϵ-error.

Similarly, for the trace-norm regularized solution, we obtain the following result:

COROLLARY 4. *Suppose $\mathbf{X}^*$ is rank-r with bounded entries, let $\mathbf{X}_{\Sigma}(\mathcal{S})$ be the trace-norm regularized empirical loss minimizer: $\mathbf{X}_{\Sigma}(\mathcal{S}) = \arg\min_{\|\mathbf{X}\|_{\Sigma}\leq\sqrt{rnm}} \hat{L}_{hinge}(\mathbf{X})$. Then there exists a constant K, with the sample size $|\mathcal{S}| \geq \frac{32K^2r(n+m)\ln n + \ln\frac{1}{\delta}}{\epsilon^2}$, with probability at least $1-\delta$, we have*

$$L(\mathbf{X}_{\Sigma}(\mathcal{S})) \leq \hat{L}_{hinge}(\mathbf{X}^*) + \epsilon$$

Remark. By comparing the sample complexity of regularized CPL with trace-norm versus max-norm, we can see that max-norm is slightly superior by avoiding a $\log n$ factor, which is consistent with the analysis of classical collaborative filtering problems [28].

8. EMPIRICAL STUDIES

In this section, we conduct a set of experiments to demonstrate the effectiveness of our proposed approaches.

8.1 Experimental setting

Table 1: Statistics of data sets used in our experiments

Data	#Users	#Items	Size	Sparsity
Sushi	5,000	10	225,000	1
Vote200	16	67	6,459	0.182
Vote-wbc	339	369	10,658	0.0004
ML100K-500	500	500	1,659,704	0.027
ML100K-50	50	50	1,185	0.019
HetRec-600	600	600	1,042,049	0.010
HetRec-70	70	70	1,234	0.007

We test the performance of various algorithms for the CPL task on both real-world and synthetic data sets including actual pairwise comparison surveys, including the following:

- **Sushi [15]**: which is a data set contains 5,000 users' order lists over 10 kinds of sushi [3].
- **Vote-200**: a pairwise comparison survey associated with Washington Post's idea marketplace [25] implemented through the online platform [4]. In this survey, people were asked to propose who had the worst year in Washington in 2010, and then vote their comparisons between various people, agencies, organizations or classes according to the following form: "Entity A had a worse year than Entity B". Entities included "Representative Charlie Rangel", "The American Homeowner", "Senator Blanche Lincoln", "D.C. schoolchildren", and "The Political Center". Because the voting was anonymous, we have information on voting sessions, but not individual users. While it is very unlikely that multiple users compared items within a single "session," it is possible that the same person came back to the site and voted in multiple sessions, although we expect this behavior, if it occurred, was rare. As such, we treat each session as a singer user following the traditional collaborative filtering approach. There are 67 entities (items) in total, and voting results are very sparse. Thus we select a subset that each session contains at least 200 preferences.
- **Vote-wbc**: a pairwise comparison survey regarding the relative appeal of different Wikipedia advertisement banners [5] using the same platform as in the Vote-200 data. Here Wikipedia users propose and then compare pairs of possible banners for Wikipedia fundraising. Banner possibilities include "Knowledge is power. Keep the access to it free"; "A small donation for a world of information"; "A penny a thought? How many times has Wikipedia helped you?"; and "Open. Honest. Free."
- **Movielens data** [6]: a widely used collaborative filtering data set. We choose two subsets of 'Movielens 100K': 'ML100K-50' is a subset that contains 50 users on 50 items; 'ML100K-ML500' is a subset that contains 500 users on 500 movies. We generated all pairwise preferences in the subsets according to user ratings.
- **HetRec 2011 Movie data** [7]: another popular movie rating data set. We choose two subsets: 'HetRec-600' is the subset that contains 600 users on 600 movies, 'HetRec-70' is the subset that contains 70 users on 70 movies. We generated all pairwise preferences in the subsets according to user ratings.

Table 1 summarizes the statistics associated with these data sets, and shows that our data contains pairwise comparisons ranging up to several million. We measure the performance of algorithms using mean Kendell-Tau distance (MKTD), a generalization of Kendell-Tau distance to our multiple permutations setting. Kendll-Tau distance is widely used in single permutation learning evaluation [12] as it measure the fraction of pairwise preferences from the true permutation π recovered by the learned permutation $\hat{\pi}$: $d(\pi, \hat{\pi}) = \frac{\sum_{(a,b):\pi(a)<\pi(b)} \mathbb{I}\{\hat{\pi}(a)>\hat{\pi}(b)\}}{\binom{m}{2}}$. Suppose our model learns n ranking lists: $\pi_1, ...\pi_n$, given a ground truth set of test pairwise preferences $\mathcal{S}$, the MKTD is defined as:

$$\text{MKTD} = \frac{1}{|\mathcal{S}|} \sum_{(i,a,b)\in\mathcal{S}} \mathbb{I}_{(\pi_i(b)-\pi_i(a))>0}.$$

[3] http://www.kamishima.net/sushi/
[4] http://allourideas.org
[5] http://blog.allourideas.org/post/16175975017/
[6] http://movielens.umn.edu
[7] http://grouplens.org/datasets/hetrec-2011/

We test the performance of our model on the entire data set, and evaluate performance with a learning curve.

8.2 Compare CPL with rank aggregation

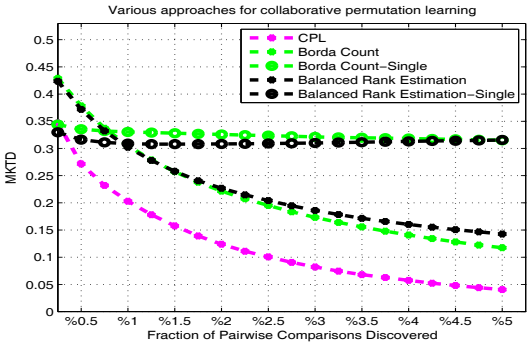
This set of experiments is designed to demonstrate how collaborative information is useful for our task. We compared the proposed CPL with a set of single permutation learning methods, including the following two that represent rank aggregation approaches: **Borda Count** [7] and **Balanced Rank Estimation** [31]. Because there are no natural active learning strategies for these approaches, we compare all algorithms to a uniform sampling of queries for fair comparison. We tune the λ parameter in our algorithms from $\{2^{-10}, 2^{-9}, ...2^{10}\}$ for all data sets. Note that we can use rank aggregation methods to learn multiple rankings: each user has one, and we can follow the traditional rank aggregation setting which learns a global ranking list. We add "Single" to denote the latter in the figures [8].

Figure 1 show the learning curves associated with our algorithms on various data sets. When comparing with Borda Count and Balanced Rank Estimation, we found that they perform reasonably well, and Balanced Rank Estimation usually performs slightly worse than Borda Count; When comparing the ranking aggregation method which learns multiple and single rankings, we find that the multiple rankings approach usually performs better, although single permutation methods might perform better in circumstances where we do not have enough observed data; when comparing the previous method with our collaborative approach, we can see that our algorithms performs significantly better. Finally, our method can improve rank aggregation approaches more on the voting data when we have more observations (Vote200 case), which illustrates the intrinsic difficulty of collaborative ranking on highly sparse and noisy data sets.

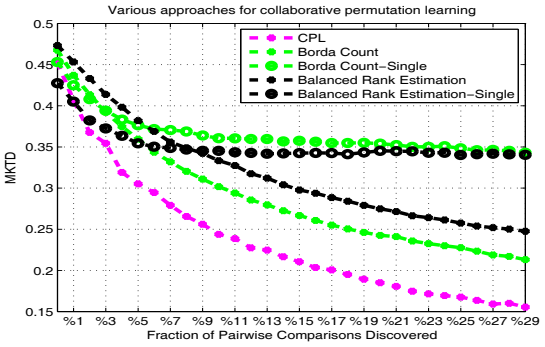
8.3 Comparison of active learning strategies

Knowing that CPL methods are superior, this set of experiments compares various active learning strategies. We compare the following methods: i) **Uniform Sampling**; ii) **Active Ranking**: as proposed in [12] for single permutation learning, note in our setting there are no given features for items, thus we perform a reduced SVD of our learned score matrix $X = U_r\Sigma_r V_r^T$, and use V_r as features for the items; iii) **SRRA**: the Smooth Relative Regret Approximation method proposed in [2], which suggests uniform sampling of the pairs for which rank position differences are within a certain range, as suggested by [2]. At first we narrow the range by 2 (which means we query the pairs with rank position differences of at most 2), then we double this range at each query stage; iv) **Simple Margin**: deterministic selection of pairs with the smallest margin; v) **Margin Sampling**: For simplicity, we begin with the following temperature attenuation scheme as $T_t = \frac{1}{t}$, where t is the learning stage. Since Active Ranking [12] needs to solve two linear programming problems in order to decide whether a pair is "Ambiguous", when the query pool is large (as in data sets with millions of pairs), the time cost becomes very high. Thus, we omit comparisons with Active Ranking on some large data sets. Figure 2 graphs the learning curves for various learning strategies. We can draw the following conclusions: i) Both Simple Margin and Margin Sampling perform significantly better than Uniform Sampling, which demonstrate the advantages that come from exploring a range of active learning strategies. While Active Ranking did not perform much better than Random Sampling, this is likely due

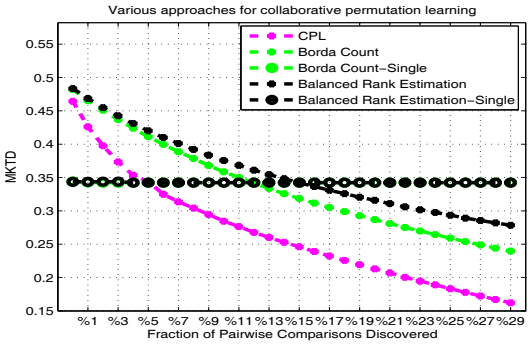
[8] We also tried to compare against Rank Centrality [17] and HodgeRank [13], but they failed in our setting due to the sparseness of the observations



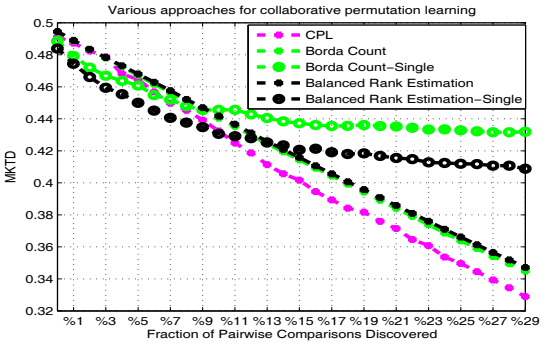
(a) *ML100K-500*



(b) *Vote200*

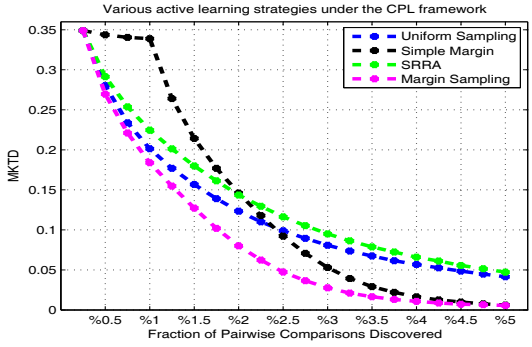


(c) *Sushi*

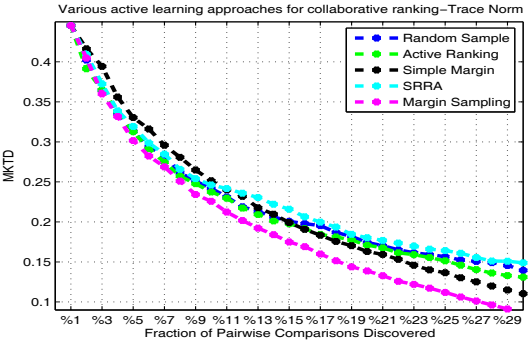


(d) *Vote-wbc*

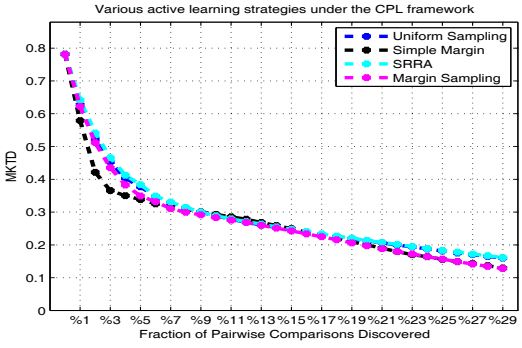
Figure 1: Comparison of various learning methods for CPL



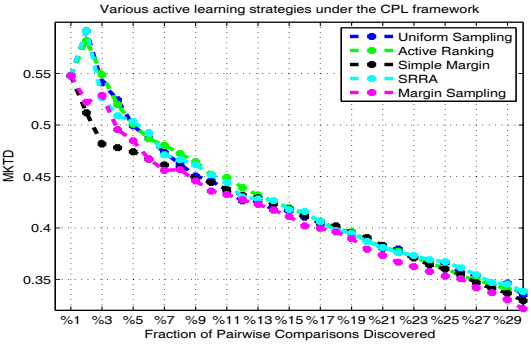
(a) *ML100K-500*



(b) *Vote200*

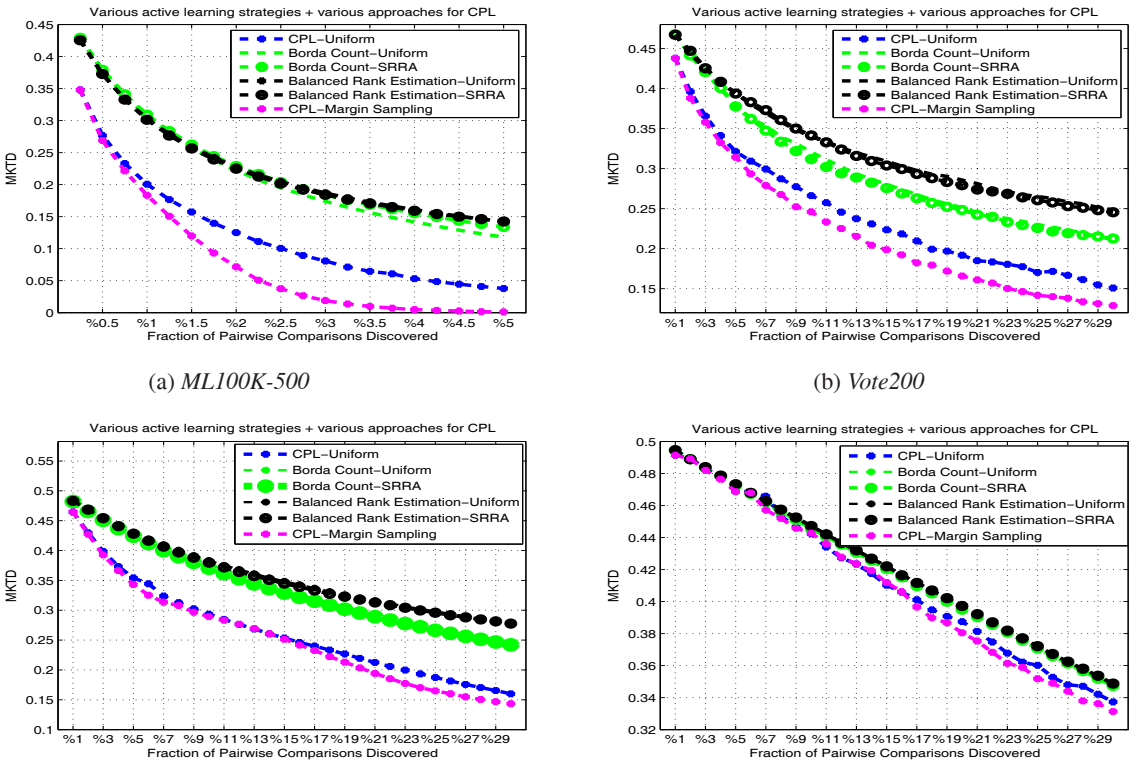


(c) *Sushi*



(d) *Vote-wbc*

Figure 2: Comparison of various active learning strategies for CPL



(a) *ML100K-500* (b) *Vote200*

(c) *Sushi* (d) *Vote-wbc*

Figure 3: Comparison of our system with single permutation based active learning + rank aggregation approaches

to 1) The active ranking approach, which is designed for the single ranking problem and so did not utilize collaborative information; 2) The "embedding" and "consistency" assumptions are not adequate for these real world data sets, especially when the features are not given; but most critically 3) the data are sparse and we can only test our algorithms on questions that were actually posed to and answered by users. Moreover, the SRRA method does not work in our setting, which may source from the reasons why Active Ranking performed poorly. ii) When comparing Simple Margin and Margin Sampling, we found that Margin Sampling performs significantly better, especially at the beginning of the learning process. This is likely because at the beginning, the model learned is not accurate as the margin information is somewhat misleading and thus we prefer more randomness. As the model becomes more accurate, we can put more trust in the margin information.

8.4 Comparison of the whole system

In this section we compare our proposed active collaborative permutation learning system(CPL+Margin Sampling) with some traditional rank aggregation methods, equipped with single permutation based active learning approaches. Figure 3 summarizes our results. We can see a significant advantage of our system over traditional approaches. SRRA based active methods cannot do better than uniform sampling in our collaborative setting, which further validates the emerging requirements of our collaborative permutation learning techniques on multiple permutation learning problems.

8.5 Simulations of sample complexity

The following experiment uses simulation on the sample complexity to evaluate our proposed methods. As shown in our theoretical analysis, under realizable condition, we only need $\mathcal{O}(\frac{r(n+m)}{\epsilon^2})$ pairwise observations to recover the multiple rankings list to ϵ error. We aim to validate the linear growth rates in our proven sample complexity bound. For simplicity, we set $\#\text{users} = \#\text{items} = n$, generalize two $n \times 5$ matrices U, V, where every elements of the two matrices are i.i.d. generated by a uniform distribution over $[0, 1]$, then we multiply these two matrices to form an $n \times n$ "rating" matrix, which is rank-5, and generate all pairwise comparisons according to the rating. Then, we test the MKTD on pair with margin at least 0.2(since we require realizable conditions for our analysis). We test the samples required to approximately recover $\%97$ of the whole pairwise preferences, test the CPL method using both uniform sampling and margin sampling strategies, and also compare with the single permutation learning methods. Figure 4 shows the sample complexity when n grows from 100 to 300. In the left figures, we see that the sample complexities of our methods look linear as n grows, and CPL requires significantly less samples than traditional methods. When comparing the sample complexities of uniform sampling and margin sampling, we found that margin sampling can usually reduce the samples needed, and it seems that sample complexity of margin sampling also grows linearly with n. We also compare the sample size needed for uniform sampling vs. margin sampling to achieve various ϵ-approximate recoveries when fixing $n = 200$, as left figure of 4 shows, the curves further validate the point that margin sampling does significantly better when we seek a very accurate solution.

8.6 CPL with Trace Norm vs Max Norm

In this section we compare the CPL with trace norm versus max norm. Figure 5 shows the curves learned by both trace-norm and max-norm regularized CPL algorithms. The basic observation is that trace-norm based methods and max-norm based algorithms usually perform similarly, and max-norm methods often performing slightly better.

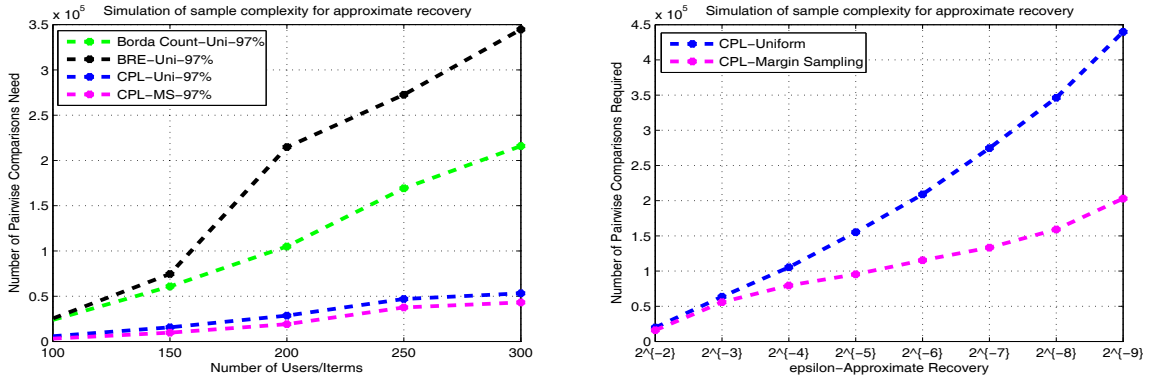


Figure 4: Simulation of Sample Complexity for Approximate Recovery, left: comparison of sample complexity as number of user-s/items increase; right: comparison of sample complexity for various ϵ-approximate recovery.

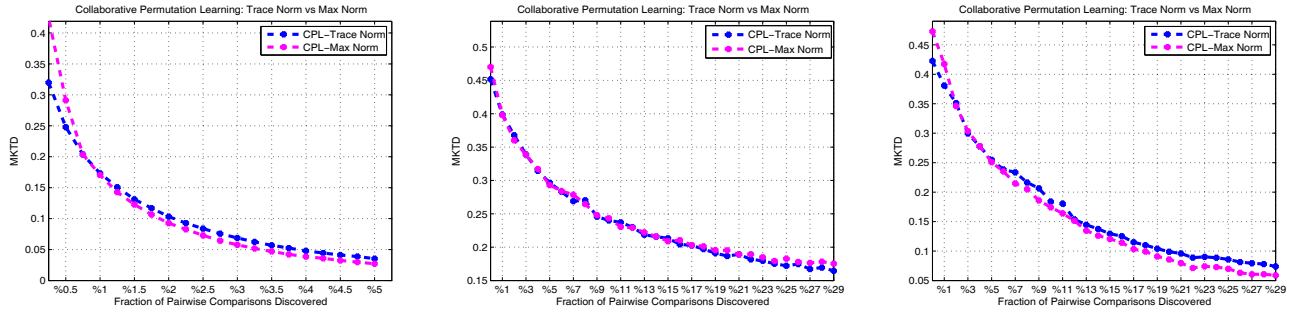


Figure 5: Evaluation on Trace Norm vs Max Norm for CPL: data sets from left to right: ML100K-500, Vote200, Hetrec 70.

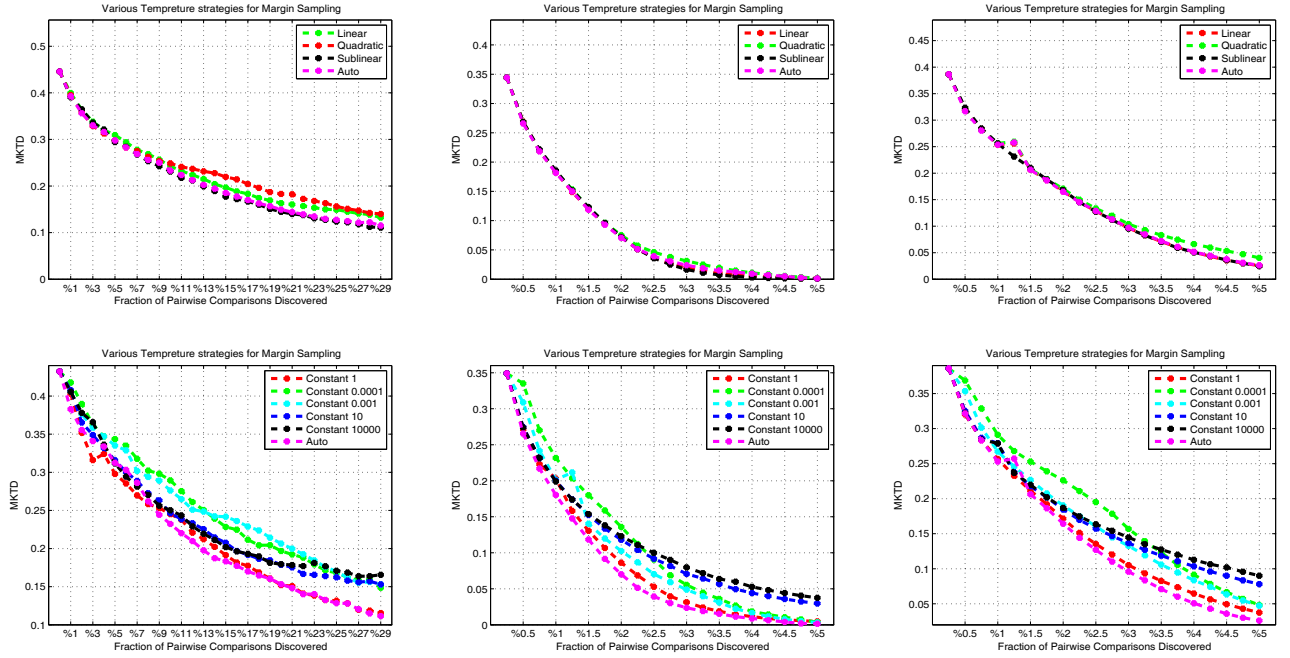


Figure 6: Evaluation of temperature decreasing schemes, top: evaluation of decreasing with different rates; bottom: evaluation of various constant temperatures. Data sets from left to right: Vote200, ML100K-50, Hetrec-600.

8.7 Temperature Decreasing Scheme

This experiment aims to study various temperature decreasing schemes for margin sampling. We consider the following general temperature model: $T_t = \frac{1}{c \cdot t^p}$, compared to the strategy we used in previous experiments where $c = 1, p = 1$, here we also test several schemes by varying c and p: first we fix $c = 1$, and vary $p = 0.5, 1, 2$, which represents sub-linear, linear, and quadratic temperature decreasing schemes, respectively. Moreover, we also adopt a "performance-driven" strategy by setting the temperature according to the MKTD we achieve: $T_t = \text{MKTD}_t$(We denote this method "auto" in the figures). Figure 6 summarizes the results. We can see that our margin sampling strategy is robust: it performs similar and quite well with different decreasing rates, when comparing different constant temperatures, we found that very low temperature tends to behave like simple margin(bad at the beginning), very high temperature tends to behave like uniform(bad at the end), adequate constant can performs reasonably well, although not so good as decreasing scheme. In addition, the proposed "performance-driven" temperature scheme appears promising. Thus, our margin sampling algorithms could be easily used in practical applications.

9. CONCLUSION

This paper studies the problem of collaborative permutation learning from pairwise comparisons, both passively and actively. We demonstrated that collaborative information is important, and propose to utilize collaborative information by matrix completion based algorithms. We then analyzed the generalization ability and sample complexity needed for approximate recovery of our proposed algorithms, and empirically demonstrated that our methods perform much better than traditional approaches. To reduce the number of comparisons required from users, we proposed various active learning strategies, and showed that active querying is very useful in reducing label costs. These approaches provide immediate application to a range of information retrieval and survey tasks. In particular, we highlight their power for creating efficient, just-in-time comparison surveys that can predict user preferences from small samples of comparisons.

We also identify several promising directions for further research, including: i) More scalable algorithms: we aim to apply our proposed methods on very large scale problems, where even the SVD is computationally prohibitive. Some existing large-scale trace-norm (and max-norm) optimization methods might be able to be adopted, e.g. [16]; ii) More theoretical analysis: currently we only explore generalization ability and approximate recovery analysis, but interesting theoretical questions remain, including: 1) the sample complexity required for exact recovery of the permutations (which might require additional assumptions); 2) the gap between active learning label complexity and uniform sample complexity.

References

[1] N. Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13:137–164, 2012.

[2] N. Ailon, R. Begleiter, and E. Ezra. Active learning using smooth relative regret approximations with applications. *COLT*, 23:19.1–19.20, 2012.

[3] S. Balakrishnan and S. Chopra. Collaborative ranking. In *WSDM*, pages 143–152, 2012.

[4] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[5] C. Boutilier, R. S. Zemel, and B. M. Marlin. Active collaborative filtering. In *UAI*, pages 98–106, 2003.

[6] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[7] J. C. de Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1784.

[8] J. C. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[9] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *In Proceedings of the 2001 American Control Conference*, pages 4734–4739, 2001.

[10] R. Foygel and N. Srebro. Concentration-based guarantees for low-rank matrix reconstruction. In *COLT*, pages 315–340, 2011.

[11] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[12] K. G. Jamieson and R. D. Nowak. Active ranking using pairwise comparisons. In *NIPS*, pages 2240–2248, 2011.

[13] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Math. Program.*, 127(1):203–244, 2011.

[14] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *UAI*, pages 278–285, 2004.

[15] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *KDD*, pages 583–588, 2003.

[16] J. D. Lee, B. Recht, R. Salakhutdinov, N. Srebro, and J. A. Tropp. Practical large-scale optimization for max-norm regularization. In *NIPS*, pages 1297–1305, 2010.

[17] S. Negahban, S. Oh, and D. Shah. Iterative ranking from pair-wise comparisons. In *NIPS*, pages 2483–2491, 2012.

[18] Y. Nesterov. A method for solving a convex programming problem with convergence rate $o(\frac{1}{k^2})$. 1983.

[19] F. Orabona, A. Argyriou, and N. Srebro. Prisma: Proximal iterative smoothing algorithm. *CoRR*, abs/1206.2372, 2012.

[20] T. K. Pong, P. Tseng, S. Ji, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

[21] A. Rajkumar and S. Agarwal. A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *ICML*, 2014.

[22] I. Rish and G. Tesauro. Active collaborative prediction with maximum margin matrix factorization. In *ISAIM*, 2008.

[23] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[24] P. Salesses, K. Schechtner, and C. A. Hidalgo. The collaborative image of the city: Mapping the inequality of urban perception. 8(7):e68400, 2013.

[25] M. J. Salganik and K. E. C. Levy. Wiki surveys: Open and quantifiable social data collection. *arXiv:1202.0500*, 2012.

[26] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[27] N. Srebro, J. D. M. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[28] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *COLT*, pages 545–560, 2005.

[29] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[30] M. Volkovs and R. S. Zemel. Collaborative ranking with 17 parameters. In *NIPS*, pages 2303–2311, 2012.

[31] F. L. Wauthier, M. I. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *ICML (3)*, pages 109–117, 2013.

[32] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *NIPS*, 2007.