# Who Likes What? – SplitLBI in Exploring Preferential Diversity of Ratings

**Qianqian Xu,**[1] **Jiechao Xiong,**[2] **Zhiyong Yang,**[3,5]
**Xiaochun Cao,**[3,5,7] **Qingming Huang,**[1,4,6,7] **Yuan Yao**[8]
[1]Key Lab. of Intelligent Information Processing, ICT, CAS, Beijing, China
[2]Tencent AI Lab, Shenzhen, Guangdong, China
[3]State Key Lab. of Information Security, IIE, CAS, Beijing, China
[4]School of Computer Science and Technology, UCAS, Beijing, China
[5]School of Cyber Security, UCAS, Beijing, China
[6]Key Lab. of Big Data Mining and Knowledge Management, CAS, Beijing, China
[7]Peng Cheng Laboratory, Shenzhen, Guangdong, China [8]Department of Mathematics, HKUST, HongKong, China
xuqianqian@ict.ac.cn, jcxiong@tencent.com, {yangzhiyong, caoxiaochun}@iie.ac.cn, qmhuang@ucas.ac.cn, yuany@ust.hk

**Abstract**

In recent years, learning user preferences has received significant attention. A shortcoming of existing learning to rank work lies in that they do not take into account the multi-level hierarchies from social choice to individuals. In this paper, we propose a multi-level model which learns both the common preference or utility function over the population based on features of alternatives to-be-compared, and preferential diversity functions conditioning on user categories. Such a multi-level model, enables us to simultaneously learn a coarse-grained social preference function together with a fine-grained personalized diversity. It provides us prediction power for the choices of new users on new alternatives. The key algorithm in this paper is based on Split Linearized Bregman Iteration (SplitLBI) algorithm which generates a dynamic path from the common utility to personalized preferential diversity, at different levels of sparsity on personalization. A synchronized parallel version of SplitLBI is proposed to meet the needs of fast analysis of large-scale data. The validity of the methodology are supported by experiments with both simulated and real-world datasets such as movie and dining restaurant ratings which provides us a coarse-to-fine grained preference learning.

## Introduction

In an era of data deluge, people often confront with the "information overload" dilemma over the last decades and it is increasingly difficult for them to locate and access useful information for making decisions, ranging from marketing and advertisements to competitions and election. Yet the ubiquitous data flow of user behavior provides us a plethora of preference revelation of participants: which movie does a user like, which restaurant does a consumer like, and so on. All of these examples yield comparisons without explicitly revealing an underlying utility function or ranking score. That is, only the preference choice is observed, not necessarily the strength of the preference. In such data flow, additional features or attributes of users and/or ranking candidates are often equipped with the preference choice in applications. Such featured data have been seen widely in E-commerce such as recommendations of books, movies, and restaurants, based on their styles and categories of users. These applications often generate discrete preferential choices leading to pairwise comparisons, together with feature representations of candidates and users. Interesting cases include, but are not limited to, the following motivating examples.

**Example 1** *In movie ratings such as MovieLens, the goal is to find the potential movies that interest a user who did not watch it yet. It is possible to derive a common global ranking scores of movies based on the feedback provided by all users, then recommend based on such population preference or utility functions. However, in reality, movies can be described as various genres (e.g., Action, Adventure, Animation, Comedy, Drama, Romance) and users are with different demographics, such as gender, ages and occupations. This may lead to diverse preferences for different user groups/individuals. For example, users under the year of 24 may prefer Drama and Comedy. When users slowly waltz into their 25-34, they begin to enjoy the love story best. Can we simultaneously capture the common preference on social level and the preferential diversity over the variation of movie genres and user groups?*

**Example 2** *Another example can be found in dining restaurant and consumer datasets. Traditional dining preference studies generally fit into a schema of aggregating global rankings of restaurants for recommendation, such as Michelin-stars. However, dining behavior, in fact, can be easily influenced by user demographics (e.g., ages, occupations, living locations) and restaurant attributes (e.g., cuisine types, price) which are now available with online crowdsourcing data. Can one predict which restaurant a particular group/individual of consumers will come to dine? It will be of great commercial value.*

In all the examples above, we see the motivation to learn

utility or ranking functions from preferential choice data, where *feature representations of the candidates in comparison and the user types* must be used as inputs which may provide both *preferential diversity* and the *common global rank* aggregation. Can we develop a framework to achieve such goals simultaneously? Popular approaches for learning to rank with feature representations for pairwise comparison data include RankSVM (Joachims 2009), RankBoost (Freund et al. 2003), RankNet (Burges et al. 2005), and Unified Robust Learning to Rank (URLR) (Fu et al. 2016), etc. However, these methods are not a natural fit in the scenarios mentioned above where the majority of users share some common preference while some groups/users might deviate from that significantly.

Recently, (Xu et al. 2016) proposes a parsimonious mixed-effect HodgeRank, which considers that a majority of users may follow the common social preferences while some users may exhibit distinct personalized preferences which should not be regarded as abnormal or outliers. Such phenomena often show in social or psychological studies equipped with crowdsourcing tasks (e.g. subjective quality assessment of multimedia). However, it is restricted to derive the ranking score only and does not aim to predict the preferences (social or individual) based on features of alternatives to-be-compared and user categories as we need here.

To overcome this limitation, in this paper, we establish a unified framework, combining the strength of both the parsimonious mixed-effect HodgeRank in (Xu et al. 2016) and learning to rank. In brief, we propose a hierarchical model which learns both the common or social preference functions based on features of alternatives to-be-compared and preferential diversity conditioning on user categories. Such a two-level model, enables us to simultaneously learn a coarse-grained social preference function together with fine-grained preferential diversity ranking functions, equipped with the prediction power for the choices of new users on new alternatives.

To initiate a task of preferential diversity rank aggregation, here we assume the majority of users share a common preference interest and behave rationally, while deviations from that exist but are localized to different types of users and thus sparse. Therefore a fine-grained learning to rank model is adopted in this paper, with sparsity structure on personalized preference deviations. Due to the unknown amount of the sparse deviations in reality, it is natural to pursue a model at a variety levels of sparsity. Statistical models like Lasso (Tibshirani 1996) or BPDN (Chen, Donoho, and Saunders 1998) are often chosen in literature. However, we adopt a recently developed algorithm in machine learning, Split Linearized Bregman Iteration (SplitLBI) (Huang et al. 2016), which is a simple iterative procedure generating a sequence of sparse models, evolving from the common global ranking, to user's personalized ranking till a full model. One advantage of SplitLBI lies in its provable weaker incoherence requirement for model selection consistency than Lasso, as well as the improved accuracy in variable selection and prediction error (Sun et al. 2017). In practice when the number of users is large and sample size is relatively small, early stopping regularization is needed to prevent the over-fitting in full model. To meet the needs of large-scale data analysis, we further propose a synchronized parallel version of SplitLBI for speedup.

Equipped with such a new scheme, given a small number of item pairs and feature representation of items, we can not only derive the common preference on a global population level, but also estimate rapidly a multi-level family of user preferential deviations in local groups.

As a summary, our main contributions in this new framework are highlighted as follows:

(A) A novel multi-level learning-to-rank method is proposed for user preference prediction based on sparse pairwise comparison labels and features of alternatives and user types. In the core of the framework lies the two-level preference functions, which include both the fixed effect of common preference, and the sparse random effects of user's preferential deviations.

(B) An iterative algorithm based on Split Linearized Bregman Iteration (SplitLBI), is developed for the estimates of the preference deviation at different sparsity levels along a regularization path, which allows an almost linear speed-up with synchronized parallelization for large-scale data.

## Related Work

Existing studies propose various techniques to model annotators' behaviors in crowdsourcing (Li et al. 2016; 2017b; 2017a; Zheng et al. 2015a; 2017; Zheng, Li, and Cheng 2016; Hu et al. 2016; Zheng et al. 2015b; Sheshadri and Lease 2013; Tang, Yin, and Ho 2019; Aldahari, Shandilya, and Shiva 2018), etc. A typical way to exploit user behaviors in the crowd is to characterize users' quality using a certain type of user model. For example, user probability (Guo, Parameswaran, and Garcia-Molina 2012; Liu et al. 2012) models each user's quality as a single parameter, indicating the probability that the annotator correctly answers a task. While these methods can model the quality of the workers, they still fail to consider other potential biases, e.g., the tendency for a worker to consistently over or underrate items. Such biased labels can be captured in the form of a user's confusion matrix (Raykar et al. 2009; Whitehill et al. 2009; Dawid and Skene 1979; Venanzi et al. 2014). For example, Dawid and Skene's (DS) (Dawid and Skene 1979) classic approach models a confusion matrix for each user and a class prior, using EM to simultaneously estimate labels, confusion matrices, and the prior.

Typical DS models consider user quality via per-user confusion matrix separately. However, the practical annotation process might include more complicated factors such as 1) different levels of expertise among users; 2) different annotation difficulty among items; 3) user-item interactions; etc. Recently, there arises a new wave to exploit crowdsourcing annotation in such complicated scenarios. (Whitehill et al. 2009), as a typical work, proposes a generalized probabilistic model where both the users' expertise and the items' difficulties are explicitly modeled. To model user-item interaction, (Zhou et al. 2012) considers separate probabilistic

distributions for each worker-item pair which is able to automatically infer item confusability and worker expertise.

Besides, there are also some recent work that can better estimate the user bias via the grouping effect of workers (Venanzi et al. 2014; Kamar, Kapoor, and Horvitz 2015; Li, Rubinstein, and Cohn 2019). Instead of modeling per-worker confusion matrix separately, such studies generate confusion matrix for each user from (a mixture of) community-level priors. With this strategy, one can not only share valuable information across similar users but also avoid model over-fitting with smaller amount parameters.

Above all, we see that the confusion matrix serves as the major element to capture user behavior in crowdsourced probabilistic models for classification. However, in the context of ranking, the confusion matrix is no longer available since the ranking problem could not be simply regarded as a process of choosing one class out of a list of candidates.

Different from above-mentioned studies, we propose a pairwise learning model which can not only derive the consensus on population level, but also estimate annotator's utility deviation at an individual level. Instead of adopting the confusion matrix, we characterize personalized behavior and user correlations with the parsimonious structure of the model parameters, which is ignored by the majority of existing literature. Moreover, with an efficient implementation of the SplitLBI method, our proposed method could monitor the evolution dynamics of the model parameters which evolves from a simple and consensus model to a complicated and personalized model.

## Methodology

In this section, we systematically introduce the methodology for coarse-to-fine grained preference learning. Specifically, we first start from the basic problem description, followed by introducing the proposed multi-level preference learning model in detail. After that, we present a simple iterative algorithm to generate paths of sparse models at different sparsity levels. Finally, a synchronized parallel version of this method is proposed to meet the needs of large-scale data.

### Problem Description

Suppose there are $n$ alternatives or items to be ranked, represented by $n$ data points with each of which having a $d$-dimensional feature vector $\boldsymbol{X}_i$. The pairwise comparison labels collected from users can be naturally represented as a directed comparison graph $G = (V; E)$. Let $V = \{1, 2, \ldots, n\}$ be the vertex set of $n$ items and $E = \{(u, i, j) : i, j \in V, u \in U\}$ be the set of edges, where $U$ is the set of all users who take part in the annotation. User $u$ provides his/her preference between choice $i$ and $j$, such that $y^u_{ij} > 0$ means $u$ prefers $i$ to $j$ and $y^u_{ij} \leq 0$ otherwise. Hence we may assume $y : E \to \mathbb{R}$ with skew-symmetry (orientation) $y^u_{ij} = -y^u_{ji}$. The magnitude of $y^u_{ij}$ can represent the degree of preference and it varies in applications. The simplest setting is the binary choice, where $y^u_{ij} = 1$ if $u$ prefers $i$ to $j$ and $y^u_{ij} = -1$ otherwise. In applications, users are often categorized by their classifications, such as occupations and ages, hence $y^u_{ij}$ may be a summary statistics of all the pairwise comparisons between $i$ and $j$ among the same category of users.

Such kind of pairwise comparison data, together with feature representations of items, arise in a variety of applications, as is shown in the introduction section. In this paper, we hope to learn a preference model to predict the preference of users on the given dataset, taking into account both the common consensus and users' diversity.

### A Multi-level Preference Learning Model

In this paper, we consider the following basic two-level linear preference model:

$$y^u_{ij} = (\boldsymbol{X}_i - \boldsymbol{X}_j)^\top(\boldsymbol{\beta} + \boldsymbol{\delta}^u) + \varepsilon^u_{ij},\ \varepsilon^u_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2). \quad (1)$$

Here $\boldsymbol{\beta}$ is the population-level parameter which captures the common coefficient weight vector of the feature shared by all users. Practically, as the preference varies greatly across different types of users, we allow each type of user to have their personalized parameters. These personalized parameters can be obtained by adding a random effect factor $\boldsymbol{\delta}^u$ to the population parameter $\boldsymbol{\beta}$, representing personalized deviations from the population behavior.

Now we could formulate the preference score based on $\boldsymbol{\beta}$ and $\boldsymbol{\delta}^u$. First, we adopt the inner product of $\boldsymbol{\beta}$ with the $i^{th}$ feature, *i.e.*, $\boldsymbol{X}_i^\top \boldsymbol{\beta}$ as the common preference score on item $i$. Meanwhile, we employ $\theta^u_i = \boldsymbol{X}_i^\top(\boldsymbol{\beta} + \boldsymbol{\delta}^u)$ as user $u$'s personalized preference score, which adds a personalized response $\boldsymbol{X}_i^\top \boldsymbol{\delta}^u$ to the common result. From this viewpoint, preference functions containing only the common preference parameter $\boldsymbol{\beta}$ can be called "coarse-grained" patterns, while functions with $\boldsymbol{\beta}$ and $\boldsymbol{\delta}^u$ thus provide "fine-grained" patterns. The remaining term $\varepsilon^u_{ij}$ in (1) measures the random noise in sampling which is of zero mean and variance $\sigma^2$.

So far, we have finished the definition of our model. In the next two remarks, we proceed further with some important properties of the proposed model.

**Remark 1** *This model can be straightforwardly extended to multi-level models with more than two levels, by considering hierarchies of user types for example, which are particularly appropriate for research designs where user information are organized at more than two levels. Another extension is to the family of generalized linear models.*

**Remark 2** *After obtaining these two sets of parameters, we can not only predict the preference for seen items rated by different users but also solve the cold-start problem in the following sense. When a new item (which have not yet received any judgments from the community) comes, given the low-level feature $\boldsymbol{x}_{new}$ we can use a linear function $\boldsymbol{x}_{new}^\top(\boldsymbol{\beta}+\boldsymbol{\delta}^u)$ to predict his preference score. Besides, when a new active user comes, we can use the common score $f(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{\beta}$ to predict the user's preference.*

### Sparse Regularization Paths with Split Linearized Bregman Iteration

So far we have finished the elaboration of our model. In this subsection, our goal is to learn the model parameter $\boldsymbol{\beta}$ and $\boldsymbol{\delta}^u$ properly such that the predicted score $(\boldsymbol{X}_i - \boldsymbol{X}_j)^\top(\boldsymbol{\beta} +$

$\boldsymbol{\delta}^u$) approximates the ground-truth $y_{ij}^u$. For the whole training set, the linear model Eq.(1) can be rewritten in its matrix form

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\omega} + \boldsymbol{\varepsilon}, \tag{2}$$

where $\boldsymbol{\omega} = [\boldsymbol{\beta}, \boldsymbol{\delta}] \in \mathbb{R}^{d(1+|U|)}$, and $\boldsymbol{X} \in \mathbb{R}^{|E|\times d(1+|U|)}$ is the corresponding matrix for the linear operator $\mathcal{X} : \mathbb{R}^{d(1+|U|)} \to \mathbb{R}^E$ such that $\mathcal{X}(w(u,i,j)) = (\boldsymbol{X}_i^\top \boldsymbol{\beta} + \boldsymbol{X}_i^\top \boldsymbol{\delta}^u) - (\boldsymbol{X}_j^\top \boldsymbol{\beta} + \boldsymbol{X}_j^\top \boldsymbol{\delta}^u)$. To make the predicted score $\boldsymbol{X}\boldsymbol{\omega}$ consistent with the true labels in $\boldsymbol{y}$, we adopt a squared loss function:

$$\ell(\boldsymbol{y}, \boldsymbol{\omega}) = \frac{1}{2m}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega}||_2^2.$$

Now we turn our focus to finding a proper penalty scheme against overfitting. Practically, the feature dimension $d$ can be very large, which may lead to the notorious curse of dimensionality. As a standard remedy, one might adopt the LASSO model which poses an $\ell_1$ norm on $\boldsymbol{\omega}$ to directly penalize a dense parameter. However, in this setting, the reduction of dimensionality is always accompanied by the loss of weak-signals. In contrast, inspired by a recent method named SplitLBI (Huang et al. 2016), we then provide an alternative penalization scheme which preserves weak-signals together with a sparse parameter. Instead of directly penalizing the dense parameter $\boldsymbol{\omega}$, we here exert the $\ell_1$ penalty on an auxiliary variable $\boldsymbol{\gamma}$. Moreover, $\boldsymbol{\gamma}$ is forced to preserve the discriminative power of $\boldsymbol{\omega}$ with a proximity penalty $\frac{1}{2\nu}||\boldsymbol{\omega} - \boldsymbol{\gamma}||_2^2$. Putting all together, we come to an objective function:

$$\mathcal{L}(\boldsymbol{\omega}, \boldsymbol{\gamma}) = \frac{1}{2m}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\omega}\|_2^2 + \frac{1}{2\nu}\|\boldsymbol{\omega} - \boldsymbol{\gamma}\|_2^2. \tag{3}$$

According to the SplitLBI method, we could find a solution with the following iteration scheme:

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k - \alpha\nabla_{\boldsymbol{\gamma}} L(\boldsymbol{\omega}^k, \boldsymbol{\gamma}^k), \tag{4a}$$

$$\boldsymbol{\gamma}^{k+1} = \kappa \cdot \mathsf{prox}_{||\cdot||_1}(\boldsymbol{z}^{k+1}), \tag{4b}$$

$$\boldsymbol{\omega}^{k+1} = \boldsymbol{\omega}^k - \kappa\alpha\nabla_{\boldsymbol{\omega}} L(\boldsymbol{\omega}^k, \boldsymbol{\gamma}^k), \tag{4c}$$

where $\boldsymbol{\omega}^0 = \boldsymbol{\gamma}^0 = \boldsymbol{z}^0 = \mathbf{0}$, $k$ is the iteration index, variable $\boldsymbol{z}$ is an auxiliary parameter used for gradient descent, $\boldsymbol{z} = \boldsymbol{\rho} + \boldsymbol{\gamma}/\kappa, \boldsymbol{\rho} \in \partial||\boldsymbol{\gamma}|||_1$, and the proximal map associated with the penalty function is given by

$$\begin{aligned} \mathsf{prox}_P(z) &= \arg\min_{\boldsymbol{v}\in\mathbb{R}^{d(1+|U|)}} \left(\frac{1}{2}\|\boldsymbol{v} - \boldsymbol{z}\|^2 + ||\boldsymbol{v}||_1\right), \\ &=: \mathbf{Shrinkage}(\boldsymbol{z}). \end{aligned} \tag{5}$$

We now summarize this in Algorithm 1.

**Remark 3** *We have the following remarks for the implementation details:*

- *To accelerate the convergence, the update rule in (4c) can be replaced by the closed-form solution:*

$$\boldsymbol{\omega}^{k+1} = \arg\min_{\boldsymbol{\omega}} \ \mathcal{L}(\boldsymbol{\omega}, \boldsymbol{\gamma}^{k+1}), \tag{6}$$

*which yields:*

$$\boldsymbol{\omega}^{k+1} = (\frac{\nu}{m}\boldsymbol{X}^\top\boldsymbol{X} + \boldsymbol{I})^{-1}(\frac{\nu}{m}\boldsymbol{X}^\top\boldsymbol{y} + \boldsymbol{\gamma}^{k+1}). \tag{7}$$

*Thus (4a) can be further written as $\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \alpha\boldsymbol{H}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\gamma}^k)$, where $\boldsymbol{H} = (\nu\boldsymbol{X}^\top\boldsymbol{X} + m\boldsymbol{I})^{-1}\boldsymbol{X}^\top$. And the (4c) subroutine could thus be canceled out from the update rule.*

- *As $\boldsymbol{\omega}^k$ is the mixture of Least Square solution and sparse estimator $\boldsymbol{\gamma}$, it is usually not sparse. We will use $\boldsymbol{\gamma}^k$ as the final sparse estimator.*

**Algorithm 1** `Sparse regularization path`

**Input:** Data $(\boldsymbol{X}, \boldsymbol{y})$, damping factor $\kappa$, step size $\alpha$.
**Initialization:** $\gamma^0 = 0, z^0 = 0, t^0 = 0$
$\boldsymbol{H} \leftarrow (\nu\boldsymbol{X}^T\boldsymbol{X} + m\boldsymbol{I})^{-1}\boldsymbol{X}^\top$.
**for** $k = 0, \ldots, K$ **do**

$$\boldsymbol{z}^{k+1} \leftarrow \boldsymbol{z}^k + \alpha\boldsymbol{H}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\gamma}^k).$$
$$\boldsymbol{\gamma}^{k+1} \leftarrow \kappa \cdot \mathbf{Shrinkage}(\boldsymbol{z}^{k+1}).$$
$$t^{k+1} \leftarrow (k+1)\alpha.$$

**end for**
**Output:** Solution path $\{t^k, \gamma^k\}_{k=0,1,\ldots,K}$.

**Regularization Path Property and Cross Validation.** From a dynamical point-of-view, if we regard the stepsize as $\alpha = \Delta t$ a discrete time difference, Eq.(4a)-(4c) can actually be considered as a discretization of a dynamic system (realized when $\Delta t \to 0$):

$$\frac{d\boldsymbol{z}^t}{dt} = -\nabla_{\boldsymbol{\gamma}}\mathcal{L}(\boldsymbol{\omega}^t, \boldsymbol{\gamma}^t), \tag{9}$$

$$\boldsymbol{z}^t - \frac{\boldsymbol{\gamma}^t}{\kappa} \in \partial||\boldsymbol{\gamma}^t||_1, \tag{10}$$

$$\frac{d\boldsymbol{\omega}^t}{dt} = -\kappa \cdot \nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}^t, \boldsymbol{\gamma}^t). \tag{11}$$

Such dynamics are known as inverse-scale spaces (Burger et al. 2013; 2005), leveraging a regularization path consisting of sparse models at different levels from the null ($\mathsf{supp}(\gamma) = \phi$) to the full ($\mathsf{supp}(\gamma) = [d(1 + |U|)]$) . At iteration $k$, the cumulating time $\tau_k = k\alpha$ (recall $\alpha = \Delta t$ in each iteration) can be regarded as the inverse of the Lasso regularization parameter $\lambda$: the larger is $\tau_k$, the smaller is the regularization and hence the more nonzero parameters enter the model. Following the dynamic system, the model gradually grows from sparse to dense models with increasing complexity. This provides us a chance to obtain diverse models. However, without a stopping time control mechanism, we will finally encounter $\tau_k \to \infty$, where the dynamics may reach some over-fitting models when noise exists like our case. To prevent such overfitting models, early stopping is necessary to find an optimal model at an intermediate time. To this end, we adopt a standard cross-validation scheme to choose the stopping time $t$:

- Given the training data, fix $\kappa$ and $\alpha$, then split the data $\mathcal{S}$ into $\mathcal{S}_1, \cdots, \mathcal{S}_K$, where $\mathcal{S}_i \cup \mathcal{S}_j = \phi, i \neq j, \bigcup_{i=1}^K \mathcal{S}_i = \mathcal{S}$.
- **for** $k = 1, \ldots, K$ **do**

1. Run SplitLBI on the training data $\mathcal{S}\backslash\mathcal{S}_k$ to get the solution path.
2. For pre-decided parameter list of $t$, use a linear interpolation to get $\boldsymbol{\gamma}^t$.
3. Use the estimator $\boldsymbol{\gamma}^t$ to predict on $\mathcal{S}_k$, and then compute prediction error.

**end for**

- Return the optimal $t_{cv}$ with minimal average prediction error.

**Compatibility toward Weak Signals.** In our method, the dense parameter $\boldsymbol{\omega}$ coexists with the sparse parameter $\boldsymbol{\gamma}$. This scheme provides possibilities to embrace the weak signals that are missed by the LASSO model, which could be explained as follows. Let the support set of $\boldsymbol{\gamma}$ be defined as $\mathsf{Supp}(\boldsymbol{\gamma}) = \{(i,j) : \boldsymbol{\gamma}_{ij} \neq 0\}$. We can naturally decompose $\boldsymbol{\omega}$ into $\boldsymbol{\omega}_{\mathsf{supp}(\gamma)} + \boldsymbol{\omega}_{\mathsf{supp}(\gamma)^\perp}$. Here $\boldsymbol{\omega}_{\mathsf{A}}$ denotes the entry-wise projection onto the set $\mathsf{A}$:

$$\boldsymbol{\omega}_{\mathsf{A}i,j} = \begin{cases} \boldsymbol{\omega}_{i,j} & (i,j) \in \mathsf{A} \\ 0 & otherwise \end{cases}.$$

$\boldsymbol{\omega}_{\mathsf{supp}(\gamma)}$ then captures the strong signals that are supported by the sparse parameter $\boldsymbol{\gamma}$. Such strong signals induce a sharp reduction of the loss and dominate the sparse penalty function $||\boldsymbol{\gamma}||_1$. By contrast, $\boldsymbol{\omega}_{\mathsf{supp}(\gamma)^\perp}$ consists of parameters that fail to give a significant reduction of the loss function, that can be of either weak signals (the users are not of much difference to the common) or random noise.

**Synchronized Parallel SplitLBI.** To meet the needs of large-scale data analysis, we provide a synchronized parallel version of our method. According to (4a)-(4c) and Remark 3, we implement a parallel computation on $\boldsymbol{res}^k = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\gamma}^k$ and $\boldsymbol{H} \cdot \boldsymbol{res}^k$, where multiple threads simultaneously compute $\boldsymbol{z}, \boldsymbol{\gamma}$ on different sample subsets $I_i$ and feature subsets $J_i$. Subsequently, $\boldsymbol{res}$ is updated synchronously before the next parallel iteration. We summarize this process in Algorithm 2.

## Experiments

In this section, three examples are exhibited with both simulated and real-world data to illustrate the validity of the analysis above and applications of the methodology proposed. The first example is with simulated data while the latter two exploit real-world data. Because of the page limit, we only show the first two in the main body, while remaining the third one in the supplementary materials[1].

### Simulated Study

**Settings.** We validate the proposed algorithm on simulated data with $n = |V| = 50$ labeled by 100 users. Specifically, we first generate the feature matrix for each node: $\boldsymbol{X} = [\boldsymbol{X}_i^\top]_{i=1}^n \in \mathbb{R}^{n\times d}$, where $\boldsymbol{X}_i$ is a $d$-dimensional ($d = 20$ in this experiment) column feature vector drawn randomly from $\mathcal{N}(0,1)$ representing node $i$. Then each entry of the common coefficient $\boldsymbol{\beta}$ has a probability $p_1 = 0.4$ with nonzero value and they are drawn randomly from $\mathcal{N}(0,1)$. Besides, for each user $u$, each entry of his personalized deviation coefficient $\delta^u$ has a probability $p_2 = 0.4$ to be nonzero and is drawn randomly from $\mathcal{N}(0,1)$. At last, we draw $N^u$ samples for each user randomly with binary response $y_{ij}^u$ following the model $P(y_{ij}^u = 1) = \Psi((\boldsymbol{X}_i^\top\boldsymbol{\beta} + \boldsymbol{X}_i^\top\boldsymbol{\delta}^u) - (\boldsymbol{X}_j^\top\boldsymbol{\beta} + \boldsymbol{X}_j^\top\boldsymbol{\delta}^u))$, where $\Psi(t) = 1/(1+e^{-t})$. The sample number $N^u$ uniformly spans in $[N_1, N_2] = [100, 500]$. Finally, we obtain a multi-edge graph labeled by 100 users.

**Competitors.** We compare our fine-grained model with 8 competitors: RankSVM (Joachims 2009), RankBoost (Freund et al. 2003), RankNet (Burges et al. 2005), gdbt (Friedman 2001), dart (Vinayak and Gilad-Bachrach 2015), HodgeRank (Jiang et al. 2011), URLR (Fu et al. 2016), and Lasso (Tibshirani 1996).

**Comparative Results.** To see whether our proposed method could provide more precise preference function for users by introducing individual-specific parameters (i.e., $\delta^u$), we randomly split the whole data samples into training set (70% of the total comparisons) and testing set (the remaining 30%). To ensure the statistical stability, we repeat this procedure 20 times. Tab.1 shows the experimental

[1] https://github.com/qianqianxu010/AAAI2020/tree/master/supplementary

**Algorithm 2** `SynPar-SplitLBI of Algorithm 1`

**Initialization:** Given parameter $\kappa$, $\alpha$ and thread number $P$, $k = 0$, $z^0 = 0$, $\gamma^0 = 0$, $res^0 = y$.
$\boldsymbol{H} \leftarrow (\nu\boldsymbol{X}^T\boldsymbol{X} + m\boldsymbol{I})^{-1}\boldsymbol{X}^\top$.
**Split data and variables:**
$\{1,\ldots,m\} = \bigcup_{i=1}^P I_i$, $\{1,\ldots,d(1+|U|)\} = \bigcup_{i=1}^P J_i$.
**Iteration: For each thread** $i$

$$\boldsymbol{z}_{J_i}^{k+1} \leftarrow \boldsymbol{z}_{J_i}^k + \frac{\alpha}{\nu}\boldsymbol{H}_{J_i}\boldsymbol{res}^k. \tag{12a}$$

$$\boldsymbol{\gamma}_{J_i}^{k+1} \leftarrow \kappa\,\mathbf{Shrinkage}(\boldsymbol{z}_{J_i}^k). \tag{12b}$$

$$\boldsymbol{temp}^i \leftarrow \boldsymbol{X}_{J_i}\boldsymbol{\gamma}_{J_i}. \tag{12c}$$

**Synchronize.**

$$\boldsymbol{res}_{I_i}^{k+1} \leftarrow \boldsymbol{y}_{I_i} - \sum_{i=1}^P \boldsymbol{temp}_{I_i}^i. \tag{13}$$

**Stopping:** exit when stopping rules are met.

Table 1: Coarse-grained vs. fine-grained model (i.e., Ours) on test error (i.e. mismatch ratio) in simulated data.

| | min | mean | max | std |
|---|---|---|---|---|
| RankSVM | 0.1774 | 0.2547 | 0.3591 | 0.0521 |
| RankBoost | 0.1886 | 0.2638 | 0.3665 | 0.0504 |
| RankNet | 0.1741 | 0.2569 | 0.3633 | 0.0525 |
| gdbt | 0.1903 | 0.2648 | 0.3728 | 0.0529 |
| dart | 0.1896 | 0.2633 | 0.3715 | 0.0517 |
| HodgeRank | 0.1754 | 0.2537 | 0.3574 | 0.0520 |
| URLR | 0.1756 | 0.2561 | 0.3626 | 0.0535 |
| Lasso | 0.1745 | 0.2533 | 0.3560 | 0.0523 |
| **Ours** | 0.1189 | **0.1448** | 0.1722 | 0.0169 |

| M | T(M)(s) | M | T(M)(s) |
|---|---|---|---|
| 1 | 559.99 | 9 | 66.66 |
| 2 | 277.98 | 10 | 59.40 |
| 3 | 185.64 | 11 | 55.05 |
| 4 | 139.74 | 12 | 49.95 |
| 5 | 115.23 | 13 | 48.13 |
| 6 | 95.72 | 14 | 44.14 |
| 7 | 82.56 | 15 | 42.57 |
| 8 | 72.22 | 16 | 38.97 |

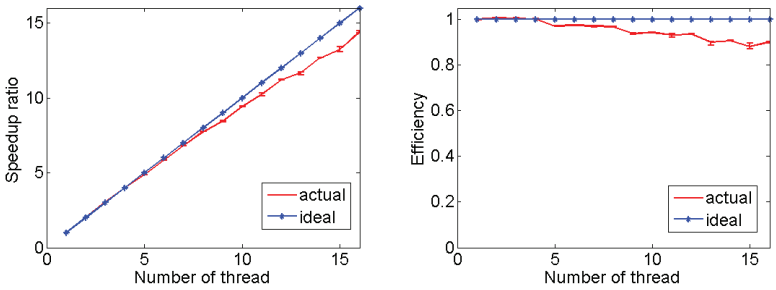



Figure 1: Left: Mean running time (20 times repeat) of SynPar-SplitLBI with thread number changing from 1 to 16 on simulated data. Middle: The linear speedup of parallel SplitLBI on simulated data. Right: The efficiency of parallel SplitLBI on simulated data.

results of the proposed fine-grained model compared with other 8 coarse-grained models with only the common preference parameter $\beta$, which indicates that our method exhibits smaller test error (i.e. mismatch ratio) with an average of $0.1448 \pm 0.0169$ due to its fine-grained property.

**Speedup of SynPar-LBI.** We then demonstrate the linear speedup of the synchronized parallel SplitLBI. In evaluating a parallel system, two performance measures of particular interest are *speedup* and *efficiency*. *Speedup* is defined as the ratio of the elapsed time when executing a program on a single thread (the single thread execution time) to the execution time when $M$ threads are available. Let $T(M)$ be the time required to complete the task on $M$ threads. The speedup $S(M)$ is the ratio: $S(M) = \frac{T(1)}{T(M)}$. *Efficiency* is defined as the average utilization of the $M$ allocated threads: $E(M) = \frac{T(1)}{MT(M)} = \frac{S(M)}{M}$.

In our setting, $M = 1, 2, 3, ..., 16$. Fig.1 (Left) shows the mean running time for 20 times repeat of SynPar-SplitLBI with thread number changing from 1 to 16 in a 16-core server with Intel(R) Xeon(R) E5-2670 2.60GHz CPU and 384GB of RAM. The server runs Linux 4.2.0 64bit. Furthermore, Fig.1 (Middle) shows the error bar of speedup with confidence interval [0.25, 0.75]. It is easy to find that the parallel SplitLBI could speed up the running time almost in a linear manner. Moreover, Fig.1 (Right) illustrates the parallel efficiency of the method. We can find that the efficiency is close to 1 in most cases. Note that Algorithm 2 is a synchronized parallel version of Algorithm 1, thus the test errors obtained by Algorithm 2 are exactly the same with the results in Tab.1.

## Movie Preference Prediction

**Dataset.** The MovieLens 1M DataSet [2] is comprised of 3952 movies rated by 6040 users. Each movie is rated on a scale from 1 to 5, with 5 indicating the best movie and 1 indicating the worst movie. There are a total of one million ratings in this dataset. Moreover, demographic information is provided voluntarily by the users, including gender, age range, occupation. Each movie titles is identical to titles provided by the IMDb [3] and each can be represented as a 18-dimensional genre feature vector, including Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.

Table 2: Coarse-grained vs. fine-grained (i.e., Ours) model in movie dataset.

| | min | mean | max | std |
|---|---|---|---|---|
| RankSVM | 0.4039 | 0.4304 | 0.4532 | 0.0131 |
| RankBoost | 0.4318 | 0.4554 | 0.4776 | 0.0131 |
| RankNet | 0.4056 | 0.4403 | 0.4625 | 0.0157 |
| gbdt | 0.3653 | 0.3850 | 0.4060 | 0.0115 |
| dart | 0.3704 | 0.3856 | 0.4023 | 0.0102 |
| HodgeRank | 0.4065 | 0.4303 | 0.4590 | 0.0126 |
| URLR | 0.4064 | 0.4300 | 0.4553 | 0.0124 |
| Lasso | 0.4089 | 0.4301 | 0.4557 | 0.0118 |
| **Ours** | 0.1204 | **0.1473** | 0.1814 | 0.0163 |

**Settings.** We then select a subset of this dataset containing 100 movies rated by 420 users, ensuring that each user has at least 20 ratings while each movie has been rated by at least 10 users. Since the proposed algorithm is designed for pairwise comparisons, we convert the rating information into a set of pairwise comparisons. More specifically, we create a pairwise comparison $(i, j)$ if item $i$ is rated higher by user $u$ than item $j$. Note that no pairwise comparison data is generated if two items are given the same rating.

**Individual Preference.** Following the experiment design in simulated study, we also split the dataset into training set and testing set. All the experiments were repeated 20 times with different training/testing partitions to reduce variance. Similar to the simulated dataset, the proposed fine-grained method could produce significant performance improvement than other 8 coarse-grained models with smaller mean test error, shown in Tab.2. Moreover, Fig.2 shows the running time, together with speedup ratio and efficiency of SynPar-SplitLBI on this movie dataset.

**Occupation and Age Preference.** Movie preference behav-

[2] https://grouplens.org/datasets/movielens/

[3] http://www.imdb.com/

| M | T(M)(s) | M | T(M)(s) |
|---|---|---|---|
| 1 | 3155.63 | 9 | 380.09 |
| 2 | 1575.51 | 10 | 337.28 |
| 3 | 1054.71 | 11 | 308.54 |
| 4 | 794.88 | 12 | 282.12 |
| 5 | 648.38 | 13 | 268.97 |
| 6 | 543.54 | 14 | 249.66 |
| 7 | 466.64 | 15 | 241.45 |
| 8 | 410.50 | 16 | 218.84 |

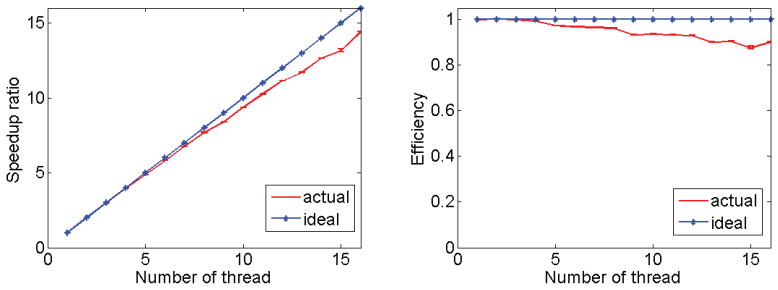
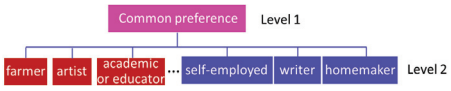
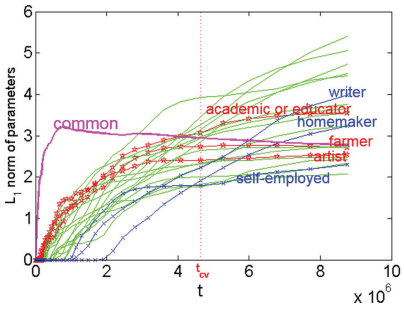



Figure 2: Experimental results of SynPar-SplitLBI with thread number changing from 1 to 16 in movie dataset.

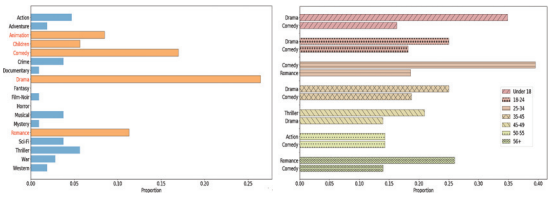



(a) Common preference with six representative group preferences




(b) Regularization path of SplitLBI

Figure 3: A two-level preference learning on movie dataset.



(a) Common preference. (b) Preference of 7 groups with different age ranges.

Figure 4: Experimental results on movie dataset.

ior may be influenced by the occupation and age factors. Tab.3 in supplementary materials show the occupation categories and the age ranges in this dataset. To exhibit the occupation influence of movie preference behavior, users from the same occupation are treated as a group. To further investigate the characteristics of groups with personalized preference, Fig.3 shows a two-level movie preference functions learned from this dataset: the common preference and 21 group preferences. Fig.3 (a) illustrates this two-level hierarchical model with six representative groups, among which farmer, artist, academic or educator are the top 3 groups exhibiting a large deviation from the common preferences, while the self-employed, writer, homemaker are those showing similar preference with the common. These are just the results derived from Fig.3 (b) using the methodology proposed in this paper. The purple curve represents the common preference, while the remaining 21 curves there represent the 21 occupation group preferences in regularization paths, of which the earlier popping up to be nonzero, the larger deviation is the group preference from the common. The purple curve indicates the path of the common preference parameter, being the first popping up. The red curves represent the top 3 groups (i.e., farmer, artist, and academic or educator) who jumped out early. Groups who jumped out earlier are those with a large deviation from the common ranking. Besides, the blue curves indicate the bottom 3 groups (i.e., homemaker, writer, and self-employed) jumped out later, and those often show similar preference with the common. Moreover, The red dotted line indicates the optimal $t$ (i.e., $t_{cv}$) obtained via cross-validation.

In particular, Fig.4(a) illustrates the common preference on this movie dataset, where the bars are the proportions of movie genres among top 50% movies ranked by common consensus preference. One can see that the top five genres in the common (social) preference are Drama, Comedy, Romance, Animation, and Children, respectively. Furthermore, user preferences on movies also change with age. Fig.4(b) illustrates the evolution of preference over age groups. One can see that users under the year of 18 and between 18-24 both prefer Drama and Comedy best. When users slowly waltz into their 25-34, they begin to enjoy the love story. However, when they get to their 40s, it happened that they grew to like the thriller movie best. Not surprisingly, as they continue into old age such as beyond 56, their retrospect on whole life cherishes love in a deep way and Romance movie returns to be their favorite again.

## Conclusions

In this paper, we propose a preference learning model that takes into account of both the common consensus preference and users' preferential diversity. Inspired by the newly developed Split Linearized Bregman Iteration, we establish a dynamic path from the common preference to personalized diversity, with different levels of sparsity on personalization.

A synchronized parallel version of our method is proposed to meet the needs of large-scale data analysis. Moreover, the basic two-level linear preference model can be easily extended to scenarios with multi-level groups. The effectiveness of our model has been validated on the movie preference prediction and dining restaurant preference datasets. It provides a coarse-to-fine grained characterization of user preferences with better precision in prediction.

## Acknowledgment

This work was supported in part by National Natural Science Foundation of China: 61620106009, 61861166002, 61931008, U1636214, 61836002, 61672514 and 61976202, in part by Key Research Program of Frontier Sciences, CAS: QYZDJ-SSW-SYS013, in part by the Science and Technology Development Fund of Macau SAR (File no. 0001/2018/AFJ) Joint Scientific Research Project, in part by Beijing Natural Science Foundation (No. 4172068 and 4182079), in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, in part by Youth Innovation Promotion Association CAS, and in part by Hong Kong Research Grant Council (HKRGC) grant 16303817.

## References

Aldahari, E.; Shandilya, V.; and Shiva, S. G. 2018. Crowdsourcing multi-objective recommendation system. In *WWW*, 1371–1379.

Burger, M.; Osher, S.; Xu, J.; and Gilboa, G. 2005. Nonlinear inverse scale space methods for image restoration. In *VLSM*, 25–36.

Burger, M.; Möller, M.; Benning, M.; and Osher, S. 2013. An adaptive inverse scale space method for compressed sensing. *Math. Comput.* 82(281):269–299.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, 89–96.

Chen, S. S.; Donoho, D. L.; and Saunders, M. A. 1998. Atomic decomposition by basis pursuit. *SISC* 20(1):33–61.

Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *JSTOR Series C (Applied Statistics)* 28(1):20–28.

Freund, Y.; Iyer, R.; Schapire, R. E.; and Singer, Y. 2003. An efficient boosting algorithm for combining preferences. *JMLR* 4(Nov):933–969.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist.* 29(5):1189–1232.

Fu, Y.; Hospedales, T. M.; Xiang, T.; Xiong, J.; Gong, S.; Wang, Y.; and Yao, Y. 2016. Robust subjective visual property prediction from crowdsourced pairwise labels. *TPAMI* 38(3):563–577.

Guo, S.; Parameswaran, A.; and Garcia-Molina, H. 2012. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, 385–396.

Hu, H.; Zheng, Y.; Bao, Z.; Li, G.; Feng, J.; and Cheng, R. 2016. Crowdsourced POI labelling: Location-aware result inference and task assignment. In *ICDE*, 61–72.

Huang, C.; Sun, X.; Xiong, J.; and Yao, Y. 2016. Split LBI: An iterative regularization path with structural sparsity. In *NeurIPS*, 3369–3377.

Jiang, X.; Lim, L.-H.; Yao, Y.; and Ye., Y. 2011. Statistical ranking and combinatorial Hodge theory. *Math. Program.* 127(6):203–244.

Joachims, T. 2009. SVM-rank: Support Vector Machine for ranking.

Kamar, E.; Kapoor, A.; and Horvitz, E. 2015. Identifying and accounting for task-dependent bias in crowdsourcing. In *HCOMP*, 92–101.

Li, G.; Wang, J.; Zheng, Y.; and Franklin, M. J. 2016. Crowdsourced data management: A survey. *TKDE* 28(9):2296–2319.

Li, G.; Chai, C.; Fan, J.; Weng, X.; Li, J.; Zheng, Y.; Li, Y.; Yu, X.; Zhang, X.; and Yuan, H. 2017a. CDB: optimizing queries with crowd-based selections and joins. In *SIGMOD*, 1463–1478.

Li, G.; Zheng, Y.; Fan, J.; Wang, J.; and Cheng, R. 2017b. Crowdsourced data management: Overview and challenges. In *SIGMOD*, 1711–1716.

Li, Y.; Rubinstein, B. I. P.; and Cohn, T. 2019. Exploiting worker correlation for label aggregation in crowdsourcing. In *ICML*, 3886–3895.

Liu, X.; Lu, M.; Ooi, B. C.; Shen, Y.; Wu, S.; and Zhang, M. 2012. CDAS: A crowdsourcing data analytics system. *PVLDB Endowment* 5(10):1040–1051.

Raykar, V. C.; Yu, S.; Zhao, L. H.; Jerebko, A. K.; Florin, C.; Valadez, G. H.; Bogoni, L.; and Moy, L. 2009. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*, 889–896.

Sheshadri, A., and Lease, M. 2013. SQUARE: A benchmark for research on computing crowd consensus. In *HCOMP*, 156–164.

Sun, X.; Hu, L.; Yao, Y.; and Wang, Y. 2017. GSplit LBI: Taming the procedural bias in neuroimaging for disease prediction. In *MICCAI*, 107–115.

Tang, W.; Yin, M.; and Ho, C. 2019. Leveraging peer communication to enhance crowdsourcing. In *WWW*, 1794–1805.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *JSTOR Series B (Methodological)* 267–288.

Venanzi, M.; Guiver, J.; Kazai, G.; Kohli, P.; and Shokouhi, M. 2014. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, 155–164.

Vinayak, R. K., and Gilad-Bachrach, R. 2015. DART: dropouts meet multiple additive regression trees. In *AISTATS*, 489–497.

Whitehill, J.; Ruvolo, P.; Wu, T.; Bergsma, J.; and Movellan, J. R. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NeurIPS*, 2035–2043.

Xu, Q.; Xiong, J.; Cao, X.; and Yao, Y. 2016. Parsimonious mixed-effects HodgeRank for crowdsourced preference aggregation. In *ACM MM*, 841–850.

Zheng, Y.; Cheng, R.; Maniu, S.; and Mo, L. 2015a. On optimality of jury selection in crowdsourcing. In *EDBT/ICDT*, 193–204.

Zheng, Y.; Wang, J.; Li, G.; Cheng, R.; and Feng, J. 2015b. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, 1031–1046.

Zheng, Y.; Li, G.; Li, Y.; Shan, C.; and Cheng, R. 2017. Truth inference in crowdsourcing: Is the problem solved? *PVLDB Endowment* 10(5):541–552.

Zheng, Y.; Li, G.; and Cheng, R. 2016. DOCS: domain-aware crowdsourcing system. *PVLDB Endowment* 10(4):361–372.

Zhou, D.; Platt, J. C.; Basu, S.; and Mao, Y. 2012. Learning from the wisdom of crowds by minimax entropy. In *NeurIPS*, 2195–2203.