Multi-task Crowdsourcing via an Optimization Framework

YAO ZHOU, Arizona State University
LEI YING, Arizona State University
JINGRUI HE, Arizona State University

The unprecedented amounts of data have catalyzed the trend of combining human insights with machine learning techniques, which facilitate the use of crowdsourcing to enlist label information both effectively and efficiently. One crucial challenge in crowdsourcing is the diverse worker quality, which determines the accuracy of the label information provided by such workers. Motivated by the observations that same set of tasks are typically labeled by the same set of workers, we studied their behaviors across multiple related tasks and proposed an optimization framework for learning from task and worker dual heterogeneity. The proposed method uses a weight tensor to represent the workers' behaviors across multiple tasks, and seeks to find the optimal solution of the tensor by exploiting its structured information. Then, we propose an iterative algorithm to solve the optimization problem and analyze its computational complexity. To infer the true label of an example, we construct a worker ensemble based on the estimated tensor, whose decisions will be weighted using a set of entropy weight. We also prove that the gradient of the most time-consuming updating block is separable with respect to the workers, which leads to a randomized algorithm with faster speed. Moreover, we extend the learning framework to accommodate to the multi-class setting. Finally, we test the performance of our framework on several data sets, and demonstrate its superiority over state-of-the-art techniques.

CCS Concepts: •**Information systems** →**Data mining; Crowdsourcing;** •**Computer systems organization** →*Heterogeneous (hybrid) systems;*

Additional Key Words and Phrases: Multi-task learning, Crowdsourcing, Tensor representation, Optimization, Entropy Ensemble

ACM Reference format:
Yao Zhou, Lei Ying, and Jingrui He. 2019. Multi-task Crowdsourcing via an Optimization Framework. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2019), 26 pages.
DOI: 10.1145/3310227

1 INTRODUCTION

Many real world applications, ranging from natural language processing, speech recognition, computer vision, and spam filtering, etc., exhibit two types of data heterogeneities. From one hand, a learning problem, such as classification or regression, can often be viewed as a group

This work is supported by National Science Foundation under Grant No. IIS-1552654, Grant No. IIS-1813464, Grant No. CNS-1629888, Grant No. CNS-1618768 and Grant No. ECCS-1547294, the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001, and an IBM Faculty Award. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.
Author's addresses: Yao Zhou and Jingrui He, School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, 85281; Email: yzhou174@asu.edu and jingrui.he@asu.edu; Lei Ying, School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, 85281; Email: lei.ying.2@asu.edu .

DOI: 10.1145/3310227

of intrinsically correlated tasks that share a common representation. By exploiting the intrinsic relationships between tasks, it can improve the generalization performance by learning these related tasks jointly. This is frequently being referred as the task heterogeneity. From another hand, in the era of big data, unstructured data is getting piled up at an increasing speed, various approaches and platforms are emerging in order to help the researchers and businesses to make greater analytical use of data. Among the latest approaches, crowdsourcing becomes the resort that can leverage the wisdom of online workers to perform the customized micro-tasks, e.g. labeling the images for the training of a classification model, etc. However, the label qualities of these workers usually have large variation due to the various expertise of workers, which brings up the worker heterogeneity of the data.

Compared with the traditional machine learning techniques, addressing the learning problem with this type of data dual heterogeneity is challenging because how to jointly model the relations between multiple types of heterogeneities still remains an open research question. Multi-task learning (MTL) [2, 18, 19, 44, 52] has been proposed to solve the first type data heterogeneity and it can be characterized as the problem of learning multiple tasks jointly, as opposed to learning each task in isolation. Most multi-task learning methods focus on learning models under the supervised setting which usually requires large amounts of labeled examples for training, but the labor for the data labeling can be costly and time-consuming especially for the learning of a deep model. With the emergence of crowdsourcing services, researchers are able to collect large amounts of low-cost labels in a very short time. However, as a result of the tradeoff between quality and cost, these collected labels can be noisy and missing in most cases because they are normally annotated by imperfect online workers. In order to infer the ground truth labels from the large amounts of noisy and possible missing labels, many solutions have been proposed: Majority voting, Dawid and Skene EM method [12], Minimax conditional entropy method [49], Variational inference using mean field [28], Tensor augmentation and completion [53], etc. These crowdsourcing models can generally be categorized as generative models [12, 28, 49] and discriminative models [53]. Under some proper assumptions, many generative models perform well on real-world applications. However, no matter how complicated the generative model is designed, the true model that generated the crowd labels remains unknown. Therefore, the label inference problem can never achieve an accuracy as good as the ground truth.

To address the above challenges, we propose a novel structured framework extended from the multi-task classification approach using crowdsourcing labels (MultiC^2) [57]. Compared with the traditional multi-task learning, which requires ground truth labels, MultiC^2 aims to leverage the structural information between the learned classifiers of multiple tasks, various workers, and extracted features. Moreover, in this manuscript, we extend the original binary-class learning framework in multiple dimensions which includes proposing a randomized speedup algorithm, adapting to the multi-class setting, and adding the corresponding experiments. The main contributions of this manuscript are summarized as follows:

- *Formulation:* Instead of using the standard two-step procedure (label inference and model learning), we propose to learn the classifiers using the noisy and missing labels directly. We formulate the multi-task classification using crowdsourcing labels as a regularized optimization problem. The key idea is to jointly learn the task commonality, worker correlations, and feature similarity through a low-rank tensor regularization.
- *Algorithm:* We propose a blockwise iterative algorithm which jointly updates the weight tensors of all workers across all tasks. Next, the entropy-based ensemble coefficient is learned for the set of weak classifiers and the final prediction of a new data point is a weighted vote of their predictions.

- *Extensions:* The separability of the objective gradient is proved and it leads to a speedup randomized algorithm which converges faster and can be computed in parallel. We also design and extend the learning framework to accommodate the multi-class setting.
- *Data sets and evaluations:* We propose a crowd label generating model which includes four types of workers: Expert, layman, spammer and smart adversary. Then we evaluate the effectiveness, robustness, and efficiency of our framework on both semi-synthetic and real data set.

The rest of this paper is organized as follows. Section 2 is the brief review of the related work. In Sections 3 and 4, we formally present the proposed learning model, followed by the optimization algorithm and ensemble method. The crowd label generating model is introduced in Section 5. In Sections 6 and 7, we present the fast algorithm using randomized block coordinate descent and the multi-class extension of the proposed framework. Section 8 illustrates the experimental results on both semi-synthetic data set and real data sets. Finally, we conclude the paper in Section 9.

2 RELATED WORK

Multi-task learning. In many real world applications, the researchers are working on solving multiple related classification or regression problems. One successful example is the spam filtering, where all customers have different but similar distributions over the spam and normal emails, yet there is a commonality among all the customers that can be used for the design of spam filter. Another example is the house pricing prediction, where predicting the price using several combinations of house related factors are coherent learning tasks. The most naive approach is applying the conventional machine learning models to solve these tasks independently, one for each. However, this approach ignores the task relatedness and could not benefit from utilizing the shared information across tasks.

Start from the work proposed by [5], multi-task learning, which aims to exploiting the commonalities and differences across tasks and solving multiple learning tasks at the same time, has emerged as one important subdomain of machine learning. By simultaneously learning all tasks, multi-task learning has shown great performance improvement in several related applications. Many existing multi-task learning methods [2, 8, 9, 15, 17, 20, 26, 46, 51, 52] are formulated as the regularized optimization problems with an empirical loss term of the training data plus a regularization term. Their contributions usually focus on designing meaningful regularization terms in order to capture the underlying commonality among tasks. Different assumptions on task relatedness lead to various regularization formulations. Mean regularized multi-task learning, proposed by [15], assumes that the models of all tasks are generated as the variants of their mean model. The regularization term is designed to be the square loss of this mean model variances of all tasks. Multi-task learning with joint feature sparsity learning, proposed by [1, 26], assumes that the relatedness between multiple tasks can be addressed by constraining all models to have a shared set of features. The robust low-rank multi-task learning [8] has the assumption that the model can be decomposed into two components: a low-rank component that leverage the task commonality, and a group sparse structure that detect outliers. Clustered multi-task learning [51] assumes that the models of tasks have group structure so that tasks of the same group are closer than these of different groups, and they propose to formulate this intrinsic relationship with a spectral relaxed k-mean clustering regularization term. Instead of having the group truth labels like most MTL approaches do, our framework only require the noisy and possibly missing labels from crowdsourcing as the input, which is more suitable for real-world applications.

Crowdsourcing. Crowdsourcing is a special sourcing model in which pieces of micro-tasks

are distributed to a pool of online workers. It has become a popular research topic in the recent decades because of its widely commercial and academic adoption in areas such as machine learning [22, 29, 47, 49, 56], computer vision [14, 55], medical healthcare [25, 30, 45], and graph mining [6, 7, 42, 48, 50], etc. Modern machine learning tools such as deep models require massive amount of labeled data. Therefore, crowdsourcing is a desired resort for the researchers and scientist to collect large amounts of inexpensive and fast labels. The fundamental problem of interest is how to maximize the accuracy on the usage of crowdsourcing labels given the fact that the hired workers are non-experts. Despite that the use of crowdsourcing is becoming popular only in the recent decade, the idea of dividing workload among workers has successful examples for a long history. One of such earliest work is the Dawid-Skene EM model [12], which models the labeling ability of each worker as a latent confusion matrix and it is often referred as the two-coin worker model. Later on, inspired by this pioneering work, several extensions have been proposed. For example, the minimax conditional entropy (MMCE) model [49] is able to further infer the true labels, item difficulty, and worker ability jointly; Liu et al.[28] proposed a graphical model that performs variational inference method using belief propagation and mean field algorithms. The EM algorithm, proposed by Raykar et al.[36], imposes a beta prior over the worker confusion matrix and learns the classifier and true labels together. The tensor augmentation and completion method [53] proposed to use a tensor representation to capture the structural information in the crowd labeled data and augment it with a ground truth layer for label inference. The multi-task classification model using crowdsourcing labels (MultiC^2) proposed by Zhou et al. [57] uses a tensor rank minimization to capture the structural commonality between learning tasks and correlation between crowdsourcing workers and their final prediction model is an ensemble of each worker's weak classifier using entropy weights. In this paper, we extend the MultiC^2 model in multiple aspects: **(i)**. In the theoretical perspective, we have provided a detailed and logical smooth support for the convergence of the algorithm. We also give a full justification for the computational complexity analysis which considers the complexity of all optimization blocks; **(ii)**. In the efficiency perspective, We have proved the separability property of the gradient calculation w.r.t. the crowdsourcing workers and proposed a faster randomized algorithm which has lower computational complexity and can well adjust to the availability of the workers; **(iii)**. In the perspective of generalization, we have analyzed the possibility of extending the model to the multi-class learning setting and proposed a generalized framework with its effectiveness being verified; **(iv)**. In the perspective of the experiments, we have pointed out the pitfalls of the numerical implementation, increased one extra metric for evaluation, added a multi-class dataset for comparison, and verified the efficiency and scalability of the proposed randomized algorithm.

3 MULTI-TASK CLASSIFICATION USING CROWDSOURCING LABELS

In this section, we first summarize the notations and then we formally present the setting of the multi-task multi-worker learning problem.

3.1 Notation.

We use calligraphic letters (e.g. $\mathcal{X}$), to denote tensors and upper case letters (e.g. M), to denote matrices. Vectors and scalars are denoted by the bold lower case letters and lower case letters (e.g. $\mathbf{x}$ and x) respectively. Without specific mention, all vectors are assumed to be column vectors. For matrix indexing, we use $M(i, j)$ to denote the entry at the i-th row and j-th column of matrix M and use $M(i, :)$ and $M(:, j)$ to denote the i-th row and j-th column of matrix M. The matrix with subscript M_t denotes the t-th task of the learning problem and matrix transpose is denoted as M^T. The trace norm of a matrix M is defined as: $||M||_* = \sum_i \sigma_i(M)$ and $\sigma_i(M)$ denotes the i-th singular

Symbol	Definition
X_t	the data matrix of t-th task
Y_t	the crowd label matrix of t-th task
N_t	# of examples in the t-th task
N_w	# of workers in the crowd label matrix
P	# of features
$\mathbf{u}_k$	an unit vector with value of 1 on k-th element
$\mathcal{A}_{(l)} := unfold_l(\mathcal{A})$	matricization of a tensor $\mathcal{A}$ along l-th dimension
$\mathcal{A} := fold_l(\mathcal{A}_{(l)})$	transform a unfolded matrix $\mathcal{A}_{(l)}$ into a tensor
$\mathbb{I}(\cdot)$	element-wise indicator function, return 0 or 1
$M_1 \circ M_2$	hadamard product of two matrices

Table 1. Summary of symbols

value in descending order. An n-way tensor is denoted as $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_n}$. For tensor indexing, the (i, j, k)-th entry of a three-way tensor $\mathcal{X}$ is represented by $\mathcal{X}_{ijk}$. A slice of a three-way tensor $\mathcal{X}$ is denoted as $\mathcal{X}_{i::}, \mathcal{X}_{:j:}$ or $\mathcal{X}_{::k}$. A fiber of a three-way tensor is denoted as $\mathcal{X}_{:jk}, \mathcal{X}_{i:k}$ or $\mathcal{X}_{ij:}$. One important operation of a tensor $\mathcal{X}$ is called *matricization* or *unfold*, which reorders a n-way tensor into a matrix. We denote $\mathcal{X}_{(k)}$ as the output of *unfold* operation along the k-th dimension of a tensor $\mathcal{X}$, i.e., $\mathcal{X}_{(k)} = unfold_k(\mathcal{X})$. Similarly, the $fold_k(\mathcal{X}_{(k)})$ is the inverse operation of *unfold* and it returns the tensor $\mathcal{X}$. The Frobenius norm of a three-way tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is defined as $||\mathcal{A}||_F = \sqrt{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^{N_3} |\mathcal{A}_{ijk}|^2}$.

3.2 Learning framework.

In this article, we consider the following multi-task learning setting. We have T learning (classification) tasks and t-th task is associated with a set of training data:

$$\left\{ \left(X_t(1,:), Y_t(1,:)\right), ..., \left(X_t(N_t,:), Y_t(N_t,:)\right) \right\} \subset \mathbb{R}^P \times \mathbb{R}^{N_w} \tag{1}$$

where the data matrix $X_t \in \mathbb{R}^{N_t \times P}$ of t-th task has N_t examples and P-dimensional features. Crowd labels matrix $Y_t \in \{-1, 0, 1\}^{N_t \times N_w}$ also has N_t examples but with N_w workers providing the labels. In crowdsourcing, the workers do not have to label all items. Therefore, if j-th worker is willing to provide label for i-th item, $Y_t(i, j)$ is assigned with -1 (negative class) or $+1$ (positive class). Otherwise, $Y_t(i, j)$ is assigned with 0 to represent missing label.

Given the data matrices and crowd label matrices of all tasks, our target is to learn a prediction function $f : \mathbf{x} \rightarrow y$. To achieve that purpose, we propose to learn three-way weight tensor $\mathcal{W} \in \mathbb{R}^{T \times N_w \times P}$ as the first step. Each fiber $\mathbf{w} \in \mathbb{R}^P$ of this weight tensor is a weak classifier. If an new example $\mathbf{x}$ is given, the probabilistic label prediction made by this weak classifier is given as:

$$f(y = 1|\mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} \tag{2}$$

where the feature vector $\mathbf{x}$ and weight vector $\mathbf{w}$ are both augmented with a bias term.

In multi-task learning, the goal is to improve the performance of the classifiers by jointly learning from all the tasks [34]. In that case, it often leads to a better model for the general task because domain knowledge has been utilized to allow the learner to share the commonality among all tasks. In crowdsourcing, all workers are assigned to the same group of labeling tasks and naturally the labels gathered from these workers are intrinsically correlated. In order to capture this dual heterogeneous structural information, we propose to use the tensor rank minimization as the main

principle. Then, the following is the general minimization problem that we propose to solve:

$$\min_{\mathcal{W}} \sum_{t=1}^{T}\sum_{i=1}^{N_t}\sum_{j=1}^{N_w} \mathrm{L}\Big(Y_t(i,j), \mathcal{W}_{tj:}^T X_t(i,:)\Big) + \mathrm{Rank}(\mathcal{W}) + \mathrm{R}(\mathcal{W}) \tag{3}$$

where $\mathrm{L}\Big(Y_t(i,j), \mathcal{W}_{tj:}^T X_t(i,:)\Big)$ is the loss term, $\mathrm{Rank}(\mathcal{W})$ is the tensor rank, and $\mathrm{R}(\mathcal{W})$ is the tensor regularizer to prevent over-fitting or introduce feature sparsity. In crowdsourcing, most workers are not experts on the given tasks. As a result, if a worker attempt to finish a task that she/he is not sure of, the provided answer could be very unreliable [24, 38]. Therefore, a more effective mechanism design is to encourage the workers to select the unsure (missing) options if they don't have high confidence to label the item correctly. Following the same reward incentives of [38], we also assume that a qualifying loss term should satisfy following the property: *Loss of correct prediction* $\leq$ *Loss of missing label prediction* $\leq$ *Loss of incorrect prediction.* The selection of the loss term $\mathrm{L}\Big(Y_t(i,j), \mathcal{W}_{tj:}^T X_t(i,:)\Big)$ is flexible and we use the logistic loss function $\mathrm{L}(y, f(\mathbf{x})) = \log\big(1 + e^{-y\cdot f(\mathbf{x})}\big)$ in our model, although it can be naturally generalized to other loss functions. Logistic loss is a convex and monotonically decreasing function which is commonly used in practice for many real-world applications. Under the binary classification setting, when the a crowd label from a worker is missing, the logistic loss is $log\,2$. When the a crowd label from a worker is incorrect, the associated loss is greater than $log\,2$. Otherwise the loss is smaller than $log\,2$ if the label is correct.

In crowdsourcing, each item will be redundantly labeled by multiple workers such that the trained weak classifiers using the crowdsourcing labels provided by these workers should have correlations. From another perspective, in multi-task learning, the tasks are similar but different such that they share the commonalities in the learning process. These two types of intrinsic correlations can be captured using the low-rank structure [8, 23, 39, 53, 54], therefore, we also introduce the tensor rank term $\mathrm{Rank}(\mathcal{W})$ in the minimization objective. However, the rank minimization problem is NP-hard and non-convex [4] in general, one common alternative is to use trace norm to approximate the rank, which has been proved to be the closest convex envelope of the rank. Furthermore, there exist multiple ways to define the tensor trace norm. In this framework, we adopt the definition proposed by Liu et al. [27]. Following their convention, the trace norm of an n-way tensor is defined as the non-negative linear combination of the trace norms of tensor unfolded matrices along all dimensions:

$$\begin{aligned} ||\mathcal{X}||_* &= \sum_{l=1}^{3} \alpha_l ||\mathcal{X}_{(l)}||_* \\ s.t. : \sum_{l=1}^{3} \alpha_l &= 1,\ \alpha_l \geq 0,\ l = 1, ..., 3 \end{aligned} \tag{4}$$

By introducing some intermediate matrices $M_l, (l = 1, 2, 3)$ to relax tensor trace norm, the original problem is simplified and the unfolded matrices can be optimized independently. Then we add the Frobenius norm of weight tensor $\mathcal{W}$ as the regularization term to prevent overfitting, the final formulation of multi-task crowdsourcing problem becomes:

$$\min_{\mathcal{W},\{M_l\}} \gamma \sum_{t=1}^{T}\sum_{i=1}^{N_t}\sum_{j=1}^{N_w} \log\Big(1 + e^{-Y_t(i,j)\cdot W_{tj:}^T X_t(i,:)}\Big) + \sum_{l=1}^{3} \alpha_l ||M_l||_* + \frac{\beta_l}{2}||\mathcal{W}_{(l)} - M_l||_F^2 + \frac{\lambda}{2}||\mathcal{W}||_F^2 \tag{5}$$

3.3 MultiC^2 Algorithm

All terms in the objective are convex and the non-differentiable term is separable. Therefore we can apply the Block Coordinate Descent (BCD) algorithm, which guarantees to find the global optimal solution of this type of problem [40]. BCD is an iterative method which only optimizes one group of variables at a time while other variables are fixed. In our case, we have four groups of variables: $\mathcal{W}, \{M_l\}$, where $l = 1, 2, 3$, because the weight tensors have the dimension of three. There are two major sub-problems need to be solved in each BCD iteration: First sub-problem is to update the weight tensor $\mathcal{W}$ while fixing the other intermediate matrices M_1, M_2, M_3; Second sub-problem is to update one intermediate matrix M_l while $\mathcal{W}$ and other intermediate matrices are fixed.

3.3.1 Updating M_l: With some simplification, the optimization sub-problem of first BCD iteration becomes:

$$\min_{M_l} : \frac{\alpha_l}{\beta_l}||M_l||_* + \frac{1}{2}||\mathcal{W}_{(l)} - M_l||_F^2 \tag{6}$$

This problem is extensively studied in many recent work [3, 21, 32]. One of the earliest solutions of this problem, named singular value thresholding (SVT), is given by Theorem 3.1 [3].

Theorem 3.1. *For each $\tau \geq 0$, the singular value shrinkage operator of $\mathcal{W}_{(l)}$ that has the SVD decomposation as $\mathcal{W}_{(l)} = U\Sigma V^T$, obeys:*

$$D_\tau(\mathcal{W}_{(l)}) = \operatorname*{argmin}_{M_l} : \frac{\alpha_l}{\beta_l}||M_l||_* + \frac{1}{2}||\mathcal{W}_{(l)} - M_l||_F^2 \tag{7}$$

where $D_\tau(\mathcal{W}_{(l)}) = U\Sigma_\tau V^T$. It needs to compute the SVD of matrix $\mathcal{W}_{(l)} = U\Sigma V^T$, then replaces Σ with its shrinkage version: $\Sigma_\tau = diag(\{\sigma_i - \tau\}_+)$. Here $a_+ = \max(a, 0)$ and $\tau = \frac{\alpha}{\beta}$ is the threshold of the shrinkage SVD. The above discussion shows that the first sub-problem of updating M_l could be readily solved by utilizing SVD. Furthermore, this sub-problem exists an unique minimizer because the objective is strictly convex [3].

3.3.2 Updating $\mathcal{W}$: Similarly, with some simplification, the sub-problem of updating $\mathcal{W}$ becomes:

$$\min_{\mathcal{W}} \underbrace{\gamma \sum_{t=1}^{T}\sum_{i=1}^{N_t}\sum_{j=1}^{N_w} \log\left(1 + e^{-Y_t(i,j)W_{tj:}^T X_t(i,:)}\right)}_{\text{logistic loss } L(\mathcal{W})} + \underbrace{\sum_{l=1}^{3}\frac{\beta_l}{2}||\mathcal{W}_{(l)} - M_l||_F^2}_{\text{relaxation penalty } RP(\mathcal{W})} + \underbrace{\frac{\lambda}{2}||\mathcal{W}||_F^2}_{\text{regularization } R(\mathcal{W})} \tag{8}$$

In order to solve this convex optimization sub-problem, we apply gradient descent with back tracking line search to choose the step size. By taking the element-wise derivative with respect to one single entry of the weight tensor $\mathcal{W}_{tjp}$, the partial gradient of the logistic loss term is:

$$\begin{aligned}\frac{\partial L(\mathcal{W})}{\partial \mathcal{W}_{tjp}} &= -\gamma \sum_{i=1}^{N_t} \frac{e^{-Y_t(i,j)X_t(i,:)\cdot W_{tj:}}}{1 + e^{-Y_t(i,j)X_t(i,:)\cdot W_{tj:}}} Y_t(i,j)X_t(i,p) \\ &= -\gamma \left[\frac{e^{-Y_t(:,j)\circ(X_t \cdot W_{tj:})}}{1 + e^{-Y_t(:,j)\circ(X_t \cdot W_{tj:})}} \circ Y_t(:,j)\right]^T \cdot X_t(:,p)\end{aligned} \tag{9}$$

In a similar way, we can calculate the slice-wise (with respect to $\mathcal{W}_{t::} \in \mathbb{R}^{N_w \times P}$) derivative of logistic loss as follows:

$$\frac{\partial L(\mathcal{W})}{\partial \mathcal{W}_{t::}} = -\gamma \left[\frac{e^{-Y_t \circ (X_t \cdot W_{t::}^T)}}{1 + e^{-Y_t \circ (X_t \cdot W_{t::}^T)}} \circ Y_t\right]^T \cdot X_t \tag{10}$$

ALGORITHM 1: MultiC^2 Algorithm

Input: training data $\mathcal{D}_{train}$, $\boldsymbol{\alpha}, \boldsymbol{\beta}$, $\gamma, \lambda, \epsilon$, MaxIter
Output: $\mathcal{W}$, T groups of ensemble coefficient $\{c_1, ..., c_{N_w}\}$
Initialization:

$$\mathcal{W}^0 = \{0\}^{T \times N_w \times P}$$
$$m = 1$$

On training data $\mathcal{D}_{train}$:
repeat

$$\{M_l^m\} = \underset{M_l}{\text{argmin}} : \frac{\boldsymbol{\alpha}_l}{\boldsymbol{\beta}_l}||M_l||_* + \frac{1}{2}||\mathcal{W}_{(l)} - M_l||_F^2$$

$$\mathcal{W}^m = \underset{\mathcal{W}}{\text{argmin}} : \gamma \sum_{t=1}^{T}\sum_{i=1}^{N_t}\sum_{j=1}^{N_w} \log\left(1 + e^{-Y_t(i,j)\cdot W_{tj:}^T X_t(i,:)}\right) + \frac{\boldsymbol{\beta}_l}{2}||\mathcal{W}_{(l)} - M_l||_F^2 + \frac{\lambda}{2}||\mathcal{W}||_F^2$$

$$m = m + 1$$

until converged **or** m $\geq$ MaxIter

Ensemble coefficient learning for each task:
for j = 1, 2, ..., N_w **do**

$$c_j = \frac{1 - \kappa e^{H_j}}{\sum_{j=1}^{N_w}(1 - \kappa e^{H_j})}$$

end for

In multi-task learning, each task may have completely different number of examples, therefore we can only use the slice-wise gradient descent to update $f(\mathcal{W})$ as shown in Equation (10). As for the updating rules of relaxation penalty term $RP(\mathcal{W})$ and regularization term $R(\mathcal{W})$, the gradients of them are calculated as below:

$$\frac{\partial RP(\mathcal{W})}{\partial \mathcal{W}} = \sum_{l=1}^{3} \beta_l \left[\mathcal{W} - \text{fold}_l(M_l)\right] \tag{11}$$

$$\frac{\partial R(\mathcal{W})}{\partial \mathcal{W}} = \lambda \mathcal{W} \tag{12}$$

3.3.3 Worker ensemble with entropy weights: After the weight tensor $\mathcal{W}$ for all workers is jointly learned, there are multiple ways to apply these learned weights on a new testing example. One simple baseline is to apply these weights separately on testing data and then use majority voting to combine the predictions. Another baseline is taking the average of the weights of all workers and then apply the average weights on testing data. However, all of the above combining methods assume that the learned the classifiers of all workers are equally important, which is conceptually wrong because abilities of different workers are totally different.

In our framework, we propose an unsupervised weight ensemble method for the predictions of testing data. We assume the collection of item ground truth labels is generated according to an unknown model $g(y)$, and we endeavor to find a fitted parametric model which provides a suitable approximation to $g(y)$. As we defined in Section 2, $f(y|\mathbf{w}_j)$ is the probabilistic classification function for j-th worker. After the optimization algorithm converges, we have a collection of learned models $\mathcal{F} = \left\{f(y|\mathbf{w}_1), f(y|\mathbf{w}_2), ..., f(y|\mathbf{w}_{N_w})\right\}$. Then, similar to the average label entropy criterion proposed in [58] as a heuristic prediction confidence estimate, we proposed an entropy

based ensemble method to combine this set of classifiers as follows: for each learned classifier, we treat the probability predictions $P_j(\mathbf{x}_i) = f(y|\mathbf{x}_i, \mathbf{w}_j)$ of all examples as a sequence of Bernoulli trials. Here N_t is the number of training examples in t-th task, then the average entropy of j-th classifier is defined as:

$$H_j = -\frac{1}{N_t}\sum_{i=1}^{N_t} P_j(\mathbf{x}_i)\log(P_j(\mathbf{x}_i)) + (1 - P_j(\mathbf{x}_i))\log(1 - P_j(\mathbf{x}_i)) \tag{13}$$

The labeling abilities of different workers can have a wide difference, therefore the jointly learned classifiers will have various qualities. The motivation of the ensemble is to assign larger weights on these informative classifiers and smaller or zero weights on the others. Based on the information theory, the probability distribution with the largest entropy should be the least informative default. Thus, we define the ensemble coefficient as:

$$c_j = \frac{1 - \kappa e^{H_j}}{\sum_{j=1}^{N_w}(1 - \kappa e^{H_j})} \tag{14}$$

where $\kappa = e^{-H_{max}}$ and H_{max} is the largest average entropy among all the workers. At last, when a new testing example $\mathbf{x}_{test}$ is given, the predicted label $\tilde{y}$ of it is:

$$\tilde{y} = \text{sign}(\sum_{j=1}^{N_w} c_j \mathbf{w}_j^T \mathbf{x}_{test}) \tag{15}$$

The proposed algorithm is summarized in Algorithm 1 . It works as follows. We initialize the weight tensor $\mathcal{W}$ with all zeros. In each BCD iteration, we first optimize over the set of intermediate matrices $\{M_l\}$ in Problem (6) using the close-form solution and next optimize over $\mathcal{W}$ in Problem (8) using gradient descend. Then, upon convergence, entropy weights are learned separately for each task.

3.4 Convergence

Regarding the convergence property of the proposed Algorithm 1, our main interest is the block coordinate descent (BCD) method which cyclically optimize over $\mathcal{W}$ and $M_l, l = 1, 2, 3$. The overall objective of our approach is a regularized block multi-convex optimization problem, we will utilize the following definitions and theorem to analysis its convergence.

Definition 3.2. A function $f(\mathbf{x})$ is called block multi-convex, if the variable $\mathbf{x}$ can be decomposed into s blocks $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s)$, and for each block of variable $\mathbf{x}_i$, f is a convex function with respect to $\mathbf{x}_i$ while all the other block variables are fixed.

Definition 3.3. A set is called block multi-convex if its projection to each block of variables is convex.

Theorem 3.4. *For the optimization problem with the following format:*

$$\min_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{x}) := f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s) + \sum_{i=1}^{s} r_i(\boldsymbol{x}_i) \tag{16}$$

where the variable $\boldsymbol{x}$ (of a closed and multi-conex feasible set) can be decomposed into blocks $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$, f is differentiable and multi-convex, and $r_i, i = 1, \ldots, s$ are extended-value convex functions. The block coordinate descent (BCD) method which cyclically optimizes over $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$ can reach its global convergence [43].

The variable blocks of Algorithm 1 are $\mathcal{W}, M_1, M_2, M_3$. From Theorem 3.1, we know that the sub-problems of updating $M_l, l = 1, 2, 3$ are strictly convex. The second sub-problem of updating $\mathcal{W}$ is also convex because it includes the logistic loss term and two Frobenius norms as the regularization terms which are both convex. Overall, the logistic loss term $L(\mathcal{W})$ with the relaxation penalty term $RP(\mathcal{W})$ are block multi-convex, and the regularization terms $R(\mathcal{W})$ and $\sum_{l=1}^{3} \alpha_l ||M_l||_*$ are the extend-value convex functions for each block respectively. Therefore, our proposed approach is one of the practical instantiations of Theorem 3.4 and it is guaranteed to converge to the optimum. The same converging property also holds for many general applications, e.g. non-negative matrix and tensor factorization, matrix and tensor completion, which utilize the alternative least square [33] and block coordinate descent to optimize.

3.5 Computational complexity.

The first part of the algorithm is the M_l updating step in Problem (6), which involves the SVD computation of the unfolded matrix $\mathcal{W}_l$. In general, the computation cost of SVD is $O(nr^2)$ for matrix W_l of size[1] $n \times r$. Using the accelerated SVT of [32], the complexity is further reduced to $O(rq^2)$, where $q \ll r$ denotes the approximation of rank-q. If the BCD algorithm has m iterations before its convergence, then the complexity of the first sub-problem is $O(mN_w q^2)$.
The second part of the algorithm is the $\mathcal{W}$ updating step in Problem (8), which involves the repeated calculations of gradients. For each iteration, the number of floating points operation per second that the gradient calculations needed are:

$$\begin{aligned} O\left(\frac{\partial L(\mathcal{W})}{\partial \mathcal{W}}\right) &= O(TN_t N_w P) \\ O\left(\frac{\partial RP(\mathcal{W})}{\partial \mathcal{W}}\right) &= O(TN_w P) \\ O\left(\frac{\partial R(\mathcal{W})}{\partial \mathcal{W}}\right) &= O(1) \end{aligned} \tag{17}$$

Overall, the computation complexity of each iteration in the proposed Algorithm 1 is: $O\Big(mN_w(\tilde{m}TN_t P + q^2)\Big)$, where $\tilde{m}$ is the number of iterations needed for sub-problem (8) to converge. This analysis shows that the computational complexity is linear with respect to the number of workers.

4 CROWD LABELING MODEL

In this section, we introduce the crowd labels generating model. In practice, many models have been proposed, such as Dawid-Skene model [12] and Rasch Model [35] etc. In real world cases, a worker is not guaranteed to correctly label the given items because the labeling abilities of different workers can be significantly different.

In our labeling model, each worker is assumed to have an intrinsic probabilistic ability matrix to represent her labeling ability. If the number of classes K is given, then each worker will have a worker ability matrix of size $K \times K$ and the value of each entry in this matrix falls in the range of $[0, 1]$. The (i, j)-th entry of the ability matrix represents the probability that this worker will label one item belong to class i as class j. Obviously, the diagonal entries of this matrix denote the ability that a worker can correctly label one class of items. The off-diagonal elements represent the mislabeling probabilities. Besides considering the worker ability, the difficulty of labeling an item

[1]Without loss of generality, we assume $n > r$ in the unfolded matrix. In many real-world cases, number of workers N_w is a smaller number than the multiplication of number of tasks T and number of feature P. Here, we assume $r = N_w$ and $n = TP$.

should also be different from labeling another item no matter whether these two items belong to the same class or not. Therefore, we also assume there is a labeling difficulty associated with each item. The item difficulty is defined as the probability of this item being mislabeled as the incorrect classes and the value of it falls in the range of $[0, 1]$.

Based on the above assumptions, there are several types of workers: **Experts** are the type of workers who can provide correct labels with high probability no matter how difficult the item is. It is rare but possible that there are **experts** (with low payment) among the workers; **Spammers**, who are also rare in crowdsourcing, are the type of workers who randomly assign labels regardless the item difficulty. Their labeling results are analogous to random behavior. **Laymen** are the type of workers who are lacking the prior knowledge about the tasks and the qualities of their labels are reliable only when the item difficulty is relatively low. Otherwise, the labeling results of **laymen** are similar to these of the **spammers**. Sometimes **leymen** are also referred as the non-experts. Intuitively, the labeling accuracy of a worker on an item should be a function of item difficulty and worker ability. One of such functions for binary classification has already been proposed by Dai et al. [11] as:

$$p(d, a, 2) = \frac{1}{2}(1 + (1 - d)^{1-a}) \tag{18}$$

$$p(d, a, 2) = \frac{1}{2}(1 + (1 - d)^{\frac{1}{a}}) \tag{19}$$

We refer to the Equation (18) as the Experts preferred model and Equation (19) as the Laymen preferred model. d represents the item difficulty and a represents the worker ability of a given item. These two models are for binary setting, but the definitions of worker ability and item difficulty can be generalized in multi-class settings. Given a worker ability matrix $A \in [0, 1]^{K \times K}$, if the ground truth label of an item $y_{gt} = k, k \in \{1, ..., K\}$ is also given, the worker ability of a given item is defined as: $a = \frac{A_{kk} - 1/K}{1 - 1/K}$.

Names	Model formulation
Laymen and smart adversaries	$p(d, a, 2) = \begin{cases} \frac{1}{2}(1 + (1 - d)^{\frac{1}{a}}), & a \geq 0 \\ \frac{1}{2}(1 - d^{-\frac{1}{a}}), & a < 0 \end{cases}$

Table 2. Modified crowd labeling model.

Many existing crowdsourcing models have a basic assumption: the workers are better than random behavior (i.e. the labeling accuracy of the worker is 50% if the tasks are binary). But this assumption doesn't hold in general and there is one more type of workers named **smart adversaries**, whose prior knowledge about the tasks are biased and their labeling abilities are always worse than the random guess. The accuracies of their labels are getting close to zero when the difficulties of items are large and getting close to random guess when the difficulties of items are relatively small. The conventional adversaries are the type of malicious workers who intentionally sabotage labeling process and they can be eliminated by the filtering process in crowdsourcing platforms, e.g. randomly adding several testing questions on each working page as the evaluation mechanism. Different from conventional adversaries, **smart adversaries** still can exist even after the filtering procedure, thus a robust crowdsourcing framework should be able to tolerate a certain amount of this type of workers or even utilize the information contained in their labels. Under this new setting, we propose that the worker ability is in the range $[-1, +1]$ and the modified crowd labeling models are shown in Table 2.

ALGORITHM 2: Batch-RBCD Algorithm

Input: training data $\mathcal{D}_{train}$, intermediate matrices $\{M_l\}_{l=1,2,3}$, batch size N_b, $\mathcal{W}$, $\boldsymbol{\beta}$, λ, MaxIter
Output: W
Initialization: $\tilde{m} \leftarrow 0$
repeat:
  Choose batch of workers $\Phi \subset \{1, \cdots, N_w\}$ of size N_b with uniform probability $\frac{1}{N_w}$;
  Update W by Equation (21);
  $\tilde{m} \leftarrow \tilde{m} + 1$
until converged **or** $\tilde{m} >$ MaxIter

5 FAST GRADIENT USING RBCD

Solving the gradient descent of Problem (8) is time-consuming due to the high cost of large matrix multiplication and repeated line search steps got involved. In order to the optimization, we first prove that the gradient is separable with respect to each worker. Then, we propose a fast randomized block coordinate descent algorithm which solves the problem more efficiently and effectively in a manner of batch update.

The gradients of loss term, relaxation term, and regularization term are separable with respect to each worker and they are calculated as:

$$\begin{aligned}
\frac{\partial L(\mathcal{W})}{\partial \mathcal{W}_{:j:}} &= -\gamma \sum_{t=1}^{T} \left[\frac{e^{-Y_t(:,j) \circ (X_t \mathcal{W}_{tj:}^T)}}{1 + e^{-Y_t(:,j) \circ (X_t \mathcal{W}_{tj:}^T)}} \circ Y_t(:,j) \right]^T X_t \cdot u_t^T \\
\frac{\partial \mathrm{RP}(\mathcal{W})}{\partial \mathcal{W}_{:j:}} &= \sum_{l=1}^{3} \beta_l \left[\mathcal{W}_{:j:} - \left(\mathrm{fold}_l(M_l) \right)_{:j:} \right] \\
\frac{\partial \mathrm{R}(\mathcal{W})}{\partial \mathcal{W}_{:j:}} &= \lambda \mathcal{W}_{:j:}
\end{aligned} \tag{20}$$

Based on the above derivation, the gradient of the overall objective can be split into the sum of the gradients of all workers. It is an immediate proof of block separability of the overall objective in terms of the crowdsourcing workers. This is an appealing property because now we can design the algorithm that runs in parallel for each worker. From another perspective, the separated updating rules avoid the costly computation of large matrix multiplications. Thus, it is possible to design a scalable algorithm with high efficiency.

We propose a randomized block coordinate descent algorithm to update W in a batch manner in order to speed up the calculation. Compare with a full gradient descent algorithm, the block coordinate descent method is much cheaper and less memory demanding in the gradient computation especially when the workload got involved is excessive due to the size of the problem. On the other end, there are two less-costly strategies for choosing the updating block coordinate: cyclic or random. Interestingly, the theoretical analysis of cyclic coordinate descent that satisfying generality has not been done [31]. Many recent works suggest that the randomization can actually improve the convergence rate [31, 37], and choosing all blocks with certain probabilities should be able to avoid the worst-case order of block coordinates that cyclic updating may ends up with. In our multi-task multi-worker setting, updating one worker at an iteration could degenerate the performance because the crowdsourcing workers are mostly imperfect labelers. Therefore, in order to balance the tradeoff of computation speed and performance, our proposed RBCD algorithm

updates a batch of workers at each iteration and the updating rule of each worker is:

$$\mathcal{W}_{:j:} \leftarrow \mathcal{W}_{:j:} - \frac{\partial L(\mathcal{W})}{\partial \mathcal{W}_{:j:}} - \frac{\partial \mathrm{RP}(\mathcal{W})}{\partial \mathcal{W}_{:j:}} - \frac{\partial \mathrm{R}(\mathcal{W})}{\partial \mathcal{W}_{:j:}} \tag{21}$$

The detail of the fast RBCD algorithm is summarized in Algorithm 2. The computational complexity of solving sub-problem (8) using batch-RBCD is reduced to $O(\tilde{m}TN_tN_bP)$, where N_b is the batch size of the selected group of workers.

6 EXTENSION TO MULTICLASS SETTING

One way to extend the MultiC^2 to multi-class settings is by training multiple binary classifiers, one for each class, using the one-vs-other strategy. However such approach can not capture the correlations between different classes and can be very time consuming for training. Then, a natural and more efficient way of address this setting is to construct the multi-class classifier which considers all classes at once.

A natural extension is using the softmax loss in the objective. However, this learning setting, which involves taking the crowdsourcing labels as the input, requires the label space to have missing labels and it is different from the conventional softmax loss. On the other hand, our multi-task multi-worker learning scenario are tensor based approaches which requires more efforts to design and combine the objective with heterogeneous source of information. Let us denote the label space as $\mathcal{Y} = \{0, 1, \cdots, K\}$, where $K \geq 3$ represents the total number of classes and 0 represents for the missing labels. Then, for the t-th task and the j-th worker, the weak classifier $\mathcal{W}_{tj:}$ is a vector of size $K(P+1)$, which is denoted as ω for simplicity. Different from the binary setting, where we choose one class as the pivot and train the second class against it, we use the softmax function without pivot to represent the probabilistic prediction of each class. Here, we denote $\overline{\mathcal{W}}_{tjk} = \left(\omega_{k0}, \omega_{k1}, \cdots, \omega_{kP}\right)^T \in \mathbb{R}^{P+1}$ as the weak classifier of the t-th task and the j-th worker on the k-th class.

$$\mathcal{W}_{tj:} = \begin{bmatrix} \omega_{10} \\ \vdots \\ \omega_{1P} \\ \omega_{20} \\ \vdots \\ \omega_{2P} \\ \vdots \\ \omega_{K0} \\ \vdots \\ \omega_{KP} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \textit{1st class}: \ \overline{\mathcal{W}}_{tj1} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \textit{2nd class}: \ \overline{\mathcal{W}}_{tj2} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \textit{K-th class}: \ \overline{\mathcal{W}}_{tjK} \end{matrix}$$

For multi-class classification, the key challenge is looking for the proper way of defining the loss term in the objective. Let us define $P_{g_i}(\mathbf{x}_i; w) = Pr(Y = g_i | X = \mathbf{x}_i, w)$ as the probability of

predicting example $\mathbf{x}_i$ with label g_i. Then, the loss is defined as the negative log-likelihood:

$$\begin{aligned} L(\mathcal{W}) &= -\sum_{t=1}^{T}\sum_{j=1}^{N_w}\sum_{i=1}^{N_t} \log P_{g_i}(X_t(i,:);\mathcal{W}_{tj:}) \\ &= -\sum_{t=1}^{T}\sum_{j=1}^{N_w}\sum_{i=1}^{N_t} \log\left(\frac{e^{X_t(i,:)\overline{\mathcal{W}}_{tjg_i}}}{\sum_{k=1}^{K} e^{X_t(i,:)\overline{\mathcal{W}}_{tjk}}}\right) \\ &= \sum_{t=1}^{T}\sum_{j=1}^{N_w}\sum_{i=1}^{N_t} \left[\log\Big(\sum_{k=1}^{K} e^{X_t(i,:)\overline{\mathcal{W}}_{tjk}}\Big) - X_t(i,:)\overline{\mathcal{W}}_{tjg_i}\right] \end{aligned} \tag{22}$$

where, $g_i := Y_t(i,j) \in \{0,1,\cdots,K\}$ is the crowd label of i-th item provided by j-th worker. The terms of the new loss function have to satisfy the property that it penalizes less when the prediction is correct and penalizes more when the prediction is incorrect. However, the labels provided by the crowdsourcing workers can be missing. Thus, we multiply the numerator and the denominator with an indicator function on the linear prediction part to guarantee that the penalization is properly defined when the labels are missing. Then, the loss term becomes:

$$L(\mathcal{W}) = \sum_{t=1}^{T}\sum_{j=1}^{N_w}\sum_{i=1}^{N_t} \left[\log\Big(\sum_{k=1}^{K} e^{X_t(i,:)\overline{\mathcal{W}}_{tjk}\mathbb{I}(g_i\neq 0)}\Big) - X_t(i,:)\overline{\mathcal{W}}_{tjg_i}\mathbb{I}\big(g_i \neq 0\big)\right] \tag{23}$$

Similar to the binary setting, when the crowd label from a worker is missing, the associated loss is $log\, K$. When the prediction from a worker is incorrect, the loss is greater than $log\, K$. Otherwise the loss is smaller than $log\, K$ if the prediction is correct. One could simply skipping parts of the logistic loss terms by ignoring the item-worker pair that has missing labels. However, this simple solution has its drawbacks. First, the simplified model with the loss associated with the missing labels being removed has difficulty to converge. The reason is that the number of missing labels provided by one worker could be significantly different from the ones of other workers. Thus, it is reasonable to balance the loss between these workers by assigning a intermediate loss value to the item-work pair that has a missing label. Second, it is a common practice to include some 'gold standard' questions in the labeling tasks. These 'gold standard' questions are the ones to which the answers are known in advance to the mechanism designer. By interweaving the 'gold standard' questions with real crowdsourcing questions, the workers who intent to sabotage the tasks could be eliminated if they continuously provide wrong or random answers to these 'gold standard' questions. Due to the existence of the screening mechanism using 'gold standard' questions in crowdsourcing platforms, workers are more cautious on labeling and intend to provide correct answers with high confidence. These missing labels have indicated this implicit confidence information (missing means low confidence) which also supports the necessity to included loss with missing labels in the overall objective.

Then, as for the weak classifier t-th task and j-th worker (denote as ω), the partial gradient of the multi-class loss term with respect to r-th class and p-th feature is calculated as follows:

$$\begin{aligned}
\frac{\partial L(\mathcal{W})}{\partial \omega_{rp}} &= \sum_{j=1}^{N_w}\sum_{i=1}^{N_t}\left[\frac{e^{X_t(i,:)\overline{\mathcal{W}}_{tjr}\mathbb{I}(g_i\neq 0)}}{\sum_{k=1}^{K} e^{X_t(i,:)\overline{\mathcal{W}}_{tjk}\mathbb{I}(g_i\neq 0)}}X_t(i,p) - \mathbb{I}\big(g_i\neq 0 \,\&\&\, g_i = r\big)X_t(i,p)\right] \\
\frac{\partial L(\mathcal{W})}{\partial \omega_{r:}} &= \sum_{j=1}^{N_w}\sum_{i=1}^{N_t}\left[\frac{e^{X_t(i,:)\overline{\mathcal{W}}_{tjr}\mathbb{I}(g_i\neq 0)}}{\sum_{k=1}^{K} e^{X_t(i,:)\overline{\mathcal{W}}_{tjk}\mathbb{I}(g_i\neq 0)}}X_t(i,:) - \mathbb{I}\big(g_i\neq 0 \,\&\&\, g_i = r\big)X_t(i,:)\right] \\
&= \sum_{j=1}^{N_w}\left[\frac{e^{X_t\overline{\mathcal{W}}_{tjr}^T\circ\mathbb{I}\left(Y_t(:,j)\neq 0\right)}}{\sum_{k=1}^{K} e^{X_t\overline{\mathcal{W}}_{tjk}^T\circ\mathbb{I}\left(Y_t(:,j)\neq 0\right)}} - \mathbb{I}\Big(Y_t(:,j)\neq 0\circ Y_t(:,j) = r\Big)\right]^T\cdot X_t \\
\frac{\partial L(\mathcal{W})}{\partial \overline{\mathcal{W}}_{t:r}} &= \left[\frac{e^{X_t\overline{\mathcal{W}}_{t:r}^T\circ\mathbb{I}\left(Y_t\neq 0\right)}}{\sum_{k=1}^{K} e^{X_t\overline{\mathcal{W}}_{t:k}^T\circ\mathbb{I}\left(Y_t\neq 0\right)}} - \mathbb{I}\big(Y_t\neq 0\circ Y_t = r\big)\right]^T\cdot X_t
\end{aligned} \tag{24}$$

Similar to the binary classification setting, each task may have different number of examples, therefore we can only calculate the gradient of the new loss term in tensor slices with respect to each task and each class. The gradients of penalty term and regularization term remains the same as they were under binary setting. The algorithm for multi-class settings remains the same except that the loss term is replaced by Equation (23). As for the entropy weight ensemble, it is straightforward to extend to the multi-class scenario by defining the average entropy of j-th worker as:

$$H_j = -\frac{1}{N_t}\sum_{i=1}^{N_t}\sum_{k=1}^{K} P_k(\mathbf{x}_i;\overline{\mathcal{W}}_{tjk})\log\big(P_k(\mathbf{x}_i;\overline{\mathcal{W}}_{tjk})\big) \tag{25}$$

The definition of the ensemble coefficient c_j remains the same as the binary setting. At last, when a new testing example $\mathbf{x}_{test}$ given, the label prediction $\tilde{y}$ of it is given as:

$$\tilde{y} = \underset{k=1,\cdots,K}{\operatorname{argmax}}\left\{(\sum_{j=1}^{N_w} c_j\overline{\mathcal{W}}_{tj1})^T\mathbf{x}_{test},\cdots,(\sum_{j=1}^{N_w} c_j\overline{\mathcal{W}}_{tjK})^T\mathbf{x}_{test}\right\} \tag{26}$$

7 EXPERIMENTAL RESULTS

In this section, we formally present the details of the data sets, evaluation metrics, comparing methods, and the settings of the crowd labeling model. Then, we also introduce the numerical implementation details that need special attention. Next, we illustrate the experimental results and answer these questions on two binary semi-synthetic data set.

- *Effectiveness*: How effective is the proposed framework compare with other state-of-the-art approaches?
- *Robustness*: Is the entropy ensemble method robust enough to identify various types of workers?
- *Efficiency and scalability*: How fast and scalable are the proposed RBCD algorithm?

Moreover, we also show that this proposed framework can be applied on multi-class data set that uses labels from synthetic workers and the real data set that uses labels from real crowdsourcing workers.

7.1 Data set

In total, there are four data sets have been evaluated and compared. The statistics of these data set are listed in Table 3:

Data set	Tasks	# Examples (each class)	# Classes
Rec. vs Talk	Autos vs Guns	1844 (975/869)	2
	Baseball vs Mideast	1545 (860/685)	2
Comp. vs Sci.	Ms-windows.misc vs Crypt	1875 (967/908)	2
	Mac.hardware vs Space	1827 (871/956)	2
WebKB	Cornell	176 (42/32/19/83)	4
	Texas	186 (34/31/18/103)	4
	Washington	221 (66/27/21/107)	4
	Wisconsin	255 (76/35/22/122)	4
Animal Breed	Cat	439 (245/194)	2
	Canidae	514 (235/279)	2
	Horse	485 (266/219)	2

Table 3. Statistics of all data set

- *Rec. vs Talk*: One of the largest subset of 20 Newsgroups data set. It includes two binary document classification tasks that are related but different.
- *Comp. vs Sci.*: Another large subset of 20 Newsgroups data set, and it also has two binary classification tasks.
- *WebKB*: It is the collection of webpages from the computer science department of four universities. In total, there are four related multi-class classification tasks and each task has four categories.
- *Animal Breed*: It is the collection of animal images from ImageNet, which includes three related binary image classification tasks.

7.2 Evaluation metrics

We use the following metrics for the performance evaluations of the proposed methods:

- *Accuracy.* The accuracy for each class is defined as the proportion of the true positive predictions plus the true negative predictions over the total number of examples. In the multi-class case, we use the weighted average of the accuracy score of each class.
- *Average F1-score.* The F1-score is the harmonic mean of precision and recall, and it is a more strict and reliable evaluation metric than accuracy especially under learning setting where the data set is unbalanced. Average F1-score is the weighted mean of the F1-scores of all possible classes.

7.3 Comparing methods

In the comparison experiment, we employed four methods:

- *Single task learning (STL).* It learns a classifier for each task using logistic regression with ground truth (GT) labels;
- *Multi-task learning (MTL).* It also uses logistic regression and add L_{21} norm as the regularization term to do jointly feature learning with GT labels;
- *Multi-task learning with inferred labels (MMCE+MTL).* It uses the same multi-task learning algorithm except that labels are inferred from MMCE model;
- *MultiC*^{2}: Our proposed multi-task crowdsourcing classification framework which directly learns the classifiers of multiple tasks using crowd labels.

It should be noticed that the first two comparison methods STL and MTL uses the ground truth labels for learning, which is in general impossible in the real-world scenarios. The standard learning methods trained with the inferred labels, e.g. MMCE+MTL, are widely applied in many applications.

7.4 Numerical implementation.

To implement the logistic loss term in the objective, e.g. Problem (8), or its gradient, e.g. Equation (9), their mathematical formulations are not ideal from a practical standpoint. Specifically, the sigmoid function $\phi(t) = \frac{1}{1+e^{-t}}$ could easily overflow in the system even for some common input values, e.g. $t = -300$. This ends up assigning the loss term with an erroneous value of $+\infty$. For this reason, we split the loss term as follows to avoid overflow issues:

$$\phi(t) = \begin{cases} \frac{1}{1+e^{-t}}, & \text{if } t \geq 0 \\ \frac{e^t}{1+e^t}, & \text{otherwise} \end{cases} \tag{27}$$

As for the cases of training with unbalancing data set, we preprocess the training data by automatically adjusting data weights inversely proportional to class frequencies in the input data of each worker. During the learning process, these data weights will be applied on the corresponding loss of the prediction product of features $\mathbf{x}$ and classifier $\mathbf{w}$, e.g. in Equation (10), gradient of the loss $\left[\frac{e^{-Y_t \circ (X_t \cdot W_{t::}^T)}}{1+e^{-Y_t \circ (X_t \cdot W_{t::}^T)}} \circ Y_t\right]^T \cdot X_t$ will be replaced with its weighted version $\left[\frac{e^{-Y_t \circ (X_t \cdot W_{t::}^T)}}{1+e^{-Y_t \circ (X_t \cdot W_{t::}^T)}} \circ Y_t \circ \Omega\right]^T \cdot X_t$ represents the worker-wise inverse class frequency data weights. Similar reweighting strategy is also applicable in the multi-class scenario.

7.5 Binary semi-synthetic data set.

There are two binary semi-synthetic multi-task learning data set have been generated. They are the subset of the 20 News Groups data set[2], which is a collection of approximately 20k newsgroup documents that have been partitioned across 20 newsgroups. In our experiment, we have two cross-domain subsets [16]: *Rec. vs. Talk* and *Comp. vs. Sci.* Each subset includes two tasks and they belong to the top two categories in their subset. The splitting in each subset ensures that the tasks are related but different. The features we use are term frequency inverse document frequency (TF-IDF) based on the same bag-of-word dictionary. In practice, we eventually keep the top 150 TF-IDF features for each example.

In order to collect the crowd labels, we synthetically generate a group of workers. Generation of these workers is based on two parameters: number of workers N_w and number of classes K. In our semi-synthetic experiment, we have generated 50 workers. For each worker, we generate a $K \times K$ worker ability confusion matrix A as follows: the diagonal entries are independently and uniformly sampled from a certain probability range. If the worker is an **expert**, a **layman** or a **spammer**, then the probability range of diagonal entries is $[0.5, 1]$. Otherwise, the probability range is $[0, 0.5)$ for the **smart adversary**. The off-diagonal entries are randomly assigned with positive probabilities under the constraint that the sum of each row of the confusion matrix is equal to 1. Generation of the item difficulties is defined as follows: For each group of examples, we learn a logistic regression classifier $\mathbf{w}$. If the ground truth label of an item $\mathbf{x}_i$ is $+1$, then the corresponding item difficulty is $1 - P(y = +1|\mathbf{x}_i, \mathbf{w})$; Otherwise, the corresponding item difficulty is $1 - P(y = -1|\mathbf{x}_i, \mathbf{w})$. Since each worker does not have to label all items, for each worker, he/she will decide to label the given item if $p(d, a, 2) \geq \delta \cdot \min(A_{11}, A_{22}, ..., A_{CC})$ and we set $\delta = 0.9$ in the

[2] http://qwone.com/~jason/20Newsgroups/

implementation. After the crowd label matrix Y is obtained, we further randomly remove 70% of them as the missing labels and leave 30% of them as the labeled ones.

Worker	setting 1	setting 2	setting 3	setting 4	setting 5
Expert	10%	5%	0%	0%	0%
Layman	80%	85%	80%	70%	60%
Spammer	10%	10%	10%	10%	10%
Smart adversary	0%	0%	10%	20%	30%

Table 4. Five settings of the workers in semi-synthetic data set.

7.5.1 Qualitative study of inferred labels. To begin with, we report the quality of inferred labels. The purpose of this subsection is to verify our necessity of our modified crowd labeling model. There are three crowdsourcing methods being employed for labels inference: Dawid-Skene Expectation Maximization (DS-EM), Minimax Conditional Entropy (MMCE) and Majority Voting(MV). The evaluation metric is the error rate of inferred labels and all the outcomes are the results of 10 independent runs. There are five settings of semi-synthetic data set have been employed and each setting has a different proportion of all types of workers. The details of these settings are listed in Table 4. In real-world crowdsourcing applications, the majority of the workers are the laymen, i.e. the non-experts of the given tasks, whose labels will dominate the collected crowd labels. The rest of the workers could be the experts, the spammers, or the adversaries. Based on the characteristics of different types of the worker, we know that the labels collected from the experts and the laymen are helpful toward model learning. However, the labels collected from the spammers and the adversaries are either useless or could harm the model's the performance. As we can see from Table 4, from setting 1 to setting 5, the portion of the "good" workers gradually drops which means that the corresponding crowd label quality is also decreasing. In the meantime, the increasing amount of noisy (incorrect) labels collected from the adversaries could easily bias the model training. These five types of worker combinations could reflect the real-world crowdsourcing scenarios in a certain way and the rationale of designing these settings (with decreased labeling qualities) is to verify the robustness of the comparison methods.

The details of results are shown in Figure 1(a)-(d). As was expected, the MV label inference has the worst performance under all five settings because MV assumes all workers are equally good and all items are equally difficult. DS-EM and MMCE outperform MV because they both modeled the worker abilities. MMCE has even lower error rate than DS-EM because it also modeled the items difficulties and when the item difficult is ignored, the MMCE model is reduced to DS-EM model. One interesting thing we observed is that when smart adversary proportion is increasing, the error rate of DS-EM and MMCE decrease first and then start to increase again.

7.5.2 Effectiveness. For the effectiveness evaluation, each data set is randomly split into 50% for training and & 50% for testing. We also observe that the 20NG subsets we used are balanced data set and the values of True Positive (TP) and True Negative (TN) of two classes in the prediction results are almost the same, therefore accuracy and average F1-score have similar values of the same trend under various settings. Thus, we only present the average F1-score metric to illustrate the performance of all methods on the test data set. Figure 1 (e)-(h) summarize the average performances of these methods on two semi-synthetic data set. Each data set is randomly split into 50% for training and & 50% for testing. For evaluation, the average F1-score is applied as the metric to

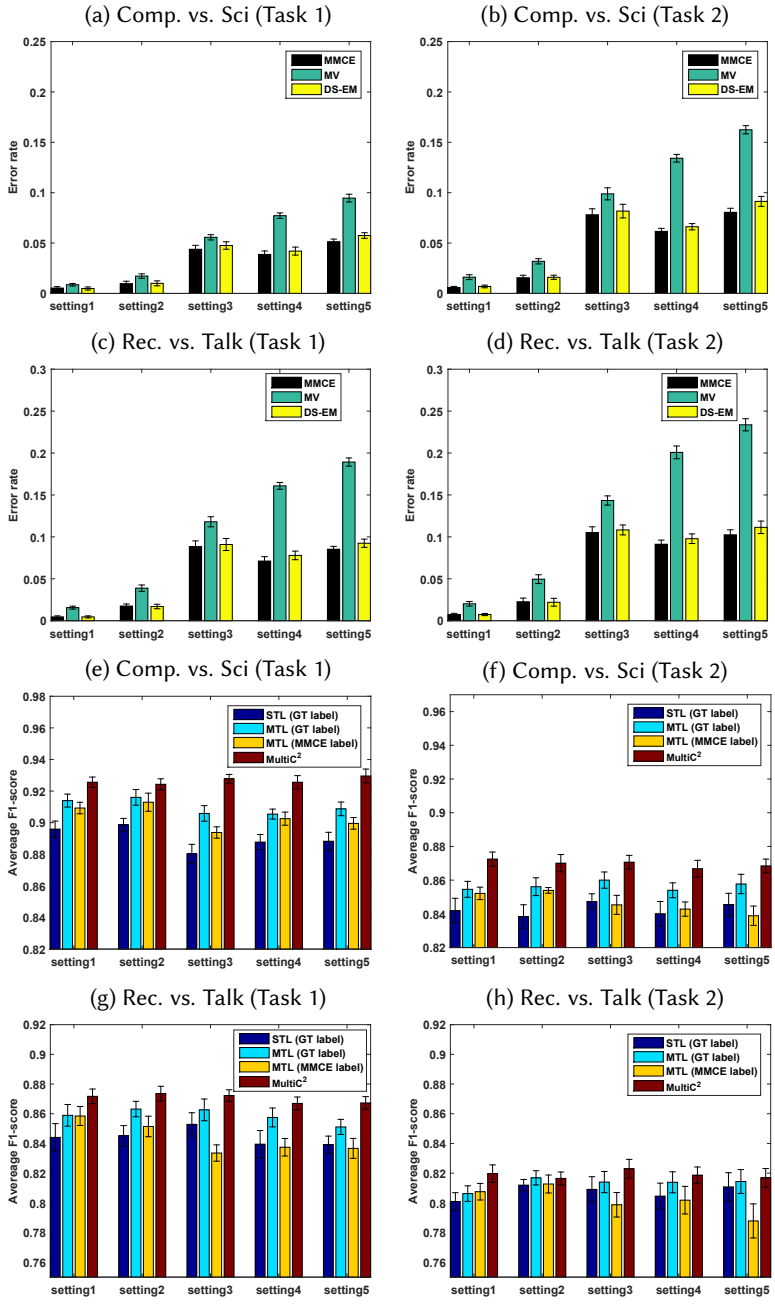

Fig. 1. (a) - (d): Error rate of labels inference using: MV, DS-EM and MMCE; (e) - (h): Classification performances on test data set using: STL and MTL with ground truth labels; MTL with MMCE inferred labels and our method (MultiC^2).

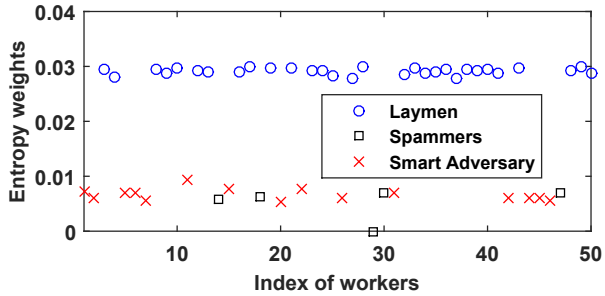

Fig. 2. Case study of entropy based weight learning

examine the performance of all methods on the test data set and all the outcomes are the results of ten independent runs. As we can see, The average F1-score of MMCE+MTL is comparable with MTL in settings 1 and 2 because the label inference of MMCE has a low error rate. However MMCE+MTL has a significant performance drop in settings 3, 4 and 5. Our proposed method MultiC^2 has consistent better performance than MMCE+MTL under all settings in every data set. It is because our proposed method can capture the structural information of all types of workers of across multiple tasks. We also observe that MultiC^2 can even outperform the MTL with ground truth labels. It is reasonable because the ensemble of these classifiers has reduced the chance of overfitting and decreased the variance. Therefore, MultiC^2, which uses the ensemble, could have a better performance.

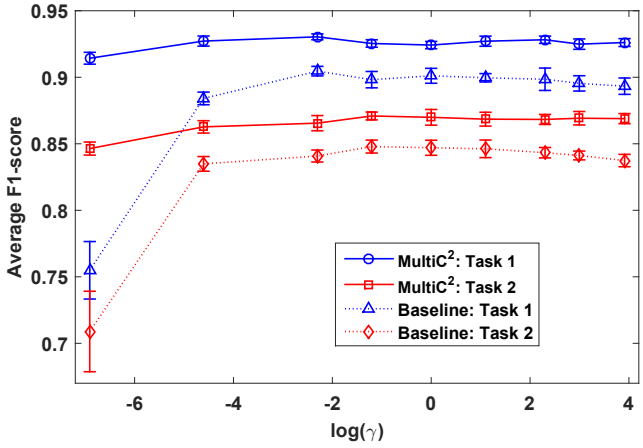

Fig. 3. Parameter study of gamma (logscale)

7.5.3 Robustness. In this section, we conduct a case study with respect to the robustness of the proposed method on task 1 of *Comp. vs. Sci* data set. The result comes from synthetic setting 5, which is the most difficulty setting because 30 percent of smart adversaries are employed. In Figure 2, the plot shows the entropy learned weights. As we can see, all the spammers and smart adversaries, who have poor classification performance, are founded and assigned with very low ensemble weights.

We also conduct a parameter study with respect to the γ, which controls the ratio of logistic loss, on two methods: MultiC^2 and the Baseline method (which has removed the tensor low-rank regularization term in MultiC^2). We change γ in the range $[0.001, 0.01, 0.1, 0.3, 1, 3, 10, 20, 50]$ while fixing the rest parameters, and plot the corresponding average F1-score of ten independent runs on a log-scaled γ. As we can see in Figure 3, our proposed model has consist better performance than Baseline method with varying γ values. The performance of Baseline has a significant drop when γ is extremely small, i.e. $\gamma = 10^{-3}$, because of lacking of information from crowd labels. However, the MultiC^2 only has a slightly performance drop under the same setting.

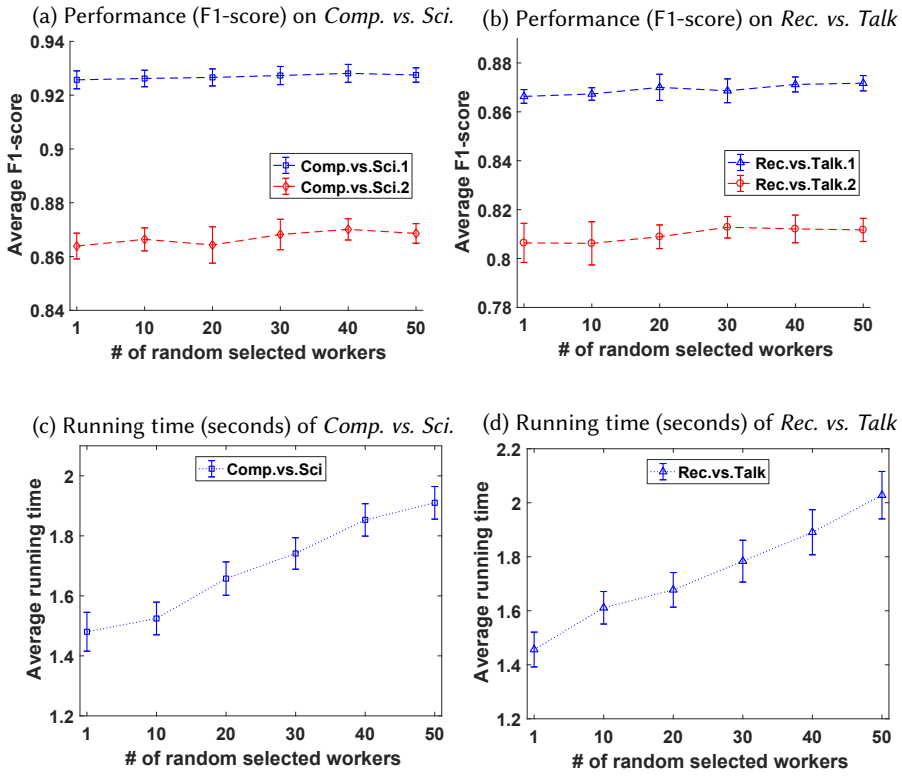

Fig. 4. Efficiency and scalability of Batch-RBCD

7.5.4 Efficiency and scalability. The efficiency plots of MultiC^2 using Batch-RBCD updating rule is presented in Figure 4. The horizontal axis represents for the number of randomly selected workers in each iteration, the vertical axis of (a) - (b) represents for the F1 score evaluation after convergence, and the vertical axis of (c) - (d) represents for the average running time of every training iteration. As we can see, the number of randomly selected workers is in a wide range. Start from updating the gradient using one random worker until updating the full gradient using all 50 workers, the performance of the framework shows that it is not very sensitive with respect to the number of randomly selected workers using the RBCD algorithm. From the bottom two plots (c) and (d), we also observe that the running time of MultiC^2 scales linearly with respect to the number of workers, which is consistent with our complexity analysis. It should be noticed that for the training of these two data set *Comp. vs. Sci.* and *Rec. vs. Talk*, the randomized BCD algorithm

will usually converge within 50 iterations. For the data set with a larger number of items and more crowdsourcing workers, the convergence may require more iterations. Under the learning scenario with more workers, the proposed randomized algorithm could have a more significant speedup. From another perspective, Batch-RBCD is not only a speed-up strategy that can improve the running speed of the gradient method by reduce computational intensity but also a realistic algorithm that considers the availability of the data defined in each worker block.

7.6 Multi-class semi-synthetic data set

The WebKB data set[3] contains the website pages from computer science departments of four universities: Cornell, Texas, Washington, and Wisconsin. Originally, there are 8282 website documents that are manually classified into seven categories. In our experiment, we have employed the top four categories (student, faculty, course, and project) for our learning problem. Under the multi-task learning setting, we consider classifying each group of documents of each university as a task. Overall, we have four related learning tasks and four classes. Since each university's web pages have their own idiosyncrasies, we follow the recommendation of the original authors [10] to train the model on the first three tasks and to test on the documents of the fourth task. The features we use are TF-IDF based on the same bag-of-word dictionary. We still keep same top 150 TF-IDF features for each example.

The process of collecting synthetic crowd labels is the similar to the one proposed in binary settings except the generation of item difficulties, which is modified as follows: if the ground truth label of an item $\mathbf{x}_i$ is class g_i, then its corresponding item difficulty is $1 - \sum_{k \neq g_i}^{K} P(y = k|\mathbf{x}_i, \mathbf{w})$. The implementation of the multi-class STL method is public available[4] and the multi-class MTL method is implemented using the standard one-vs-others strategy. The performances of the comparing methods on multi-class data set are evaluated using both types of metrics (Accuracy and average F-1 score) because the WebKB data set have multiple unbalanced classes. Figure 5 summarizes the performances of these comparing methods on the multi-class data set. As we expected, the MTL method have better performance than the STL method in terms of both evaluation metrics. The MultiC^2 framework consistently outperforms all other methods in terms of the accuracy but it has lower average F1-score than MTL with GT labels in the third learning task. The main reason is that WebKB data set is severely unbalanced, the MultiC^2 framework has high TP and TN scores in the class with majority examples but has relatively low performance in the minority classes. However, our proposed framework still outperforms the MTL method with MMCE inferred labels.

7.7 Real data set.

The Animal Breed data set [57] is designed for workers to label different animal images based on their breed type. Our approach builds on the insight that workers will get paid based on their labeling performance. Besides that, we have designed the questionnaire with the options of choosing *domestic*, *wild*, or *not sure*. Given this additional *unsure* option, the workers are provided with a conservative labeling alternative if she/he is not confident about the answer.

In detail, there are 1438 images being selected from ImageNet [13] and this subset includes three labeling tasks: Task 1: *Domestic cat v.s. Wildcat*; Task 2: *Domestic canidae v.s. Wild canidae*; Task 3: *Domestic horse v.s. Wild horse*. For the purpose of learning and evaluation, we randomly split 50% of the animal images as the training set, which is annotated by 31 real human crowdsourcing workers who have various labeling qualities, and the rest 50% images are left for testing. Regarding the features extractions on images, we adopt the standard bag-of-visual-words approach with the

[3] http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/
[4] https://github.com/PRML/PRMLT

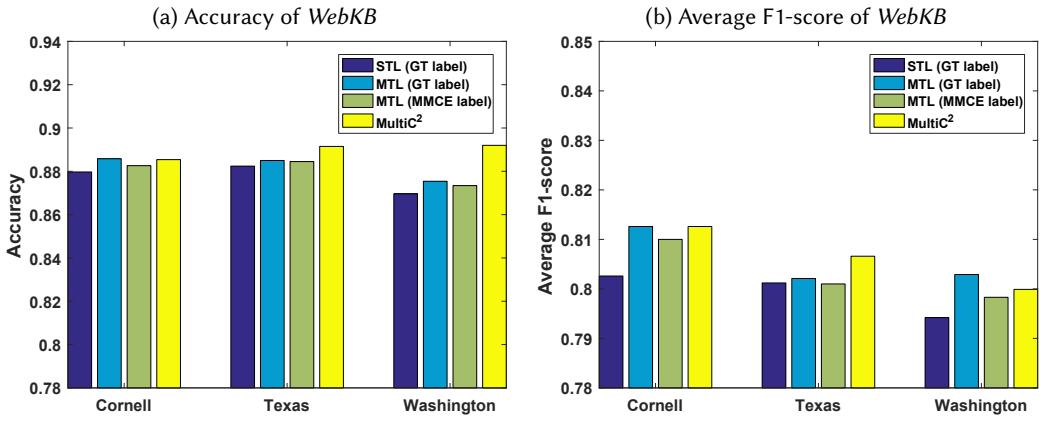

Fig. 5. Performance of comparing methods on multi-class data set (Evaluation performed on Winsconsin)

Fig. 6. Sample images of three tasks (columns) from the Animal Breed data set. First row are domestic animals and second row are wild animals.

following specific settings: Every image is resized to the one that has a maximum sides length of 300 pixels, then transformed into a grayscale image. Next, then a three-level image pyramid (resize to 1, 1/2 and 1/4 scale) is generated and the dense SIFT features of patch size 20×20 pixels with overlapping of 10 pixels are extracted on these image pyramids. All the collected SIFT features are clustered into 230 clusters using the accelerated Kmeans to generate the visual words dictionary. Then, the SIFT feature vector of each image is further quantized into the bag-of-visual-words histogram representation using the generated visual words dictionary. Eventually, the quantized features are transformed into the TF-IDF vectors and we found that using the top 110 TF-IDF features can usually guarantee a good classification performance. The feature extraction module is built on top of the VLFeat open source library [41]. The results of the real data set are summarized in Table 5. For this problem, we compare with the MTL that uses MMCE inferred labels because we can only get the inferred labels in practice. As we can see, our method outperforms the MTL method using MMCE labels in all three classification tasks in terms of both accuracy and average F1-score. The reason could be that the MTL with MMCE labels only utilizes the tasks commonality information in the learning process, however, our proposed MultiC^2 method explores the dual heterogeneity

Accuracy	MTL (MMCE label)	MultiC^2
Cat	0.7260	**0.8174**
Canidae	0.7743	**0.8132**
Horse	0.8093	**0.8430**

Average F1-score	MTL (MMCE label)	MultiC^2
Cat	0.7235	**0.8020**
Canidae	0.7701	**0.8131**
Horse	0.8086	**0.8402**

Table 5. Performance comparisons on real data set

correlations between tasks and workers. Also, the MTL uses the aggregated MMCE labels that will have incorrect inferred labels which could bias the learned classifiers. As a comparison, our proposed framework learns directly from all the crowdsourcing labels and it could keep the rich information of the redundant labels provided by these workers.

8 CONCLUSION

In this article, we formally defined the multi-task multi-worker dual heterogeneity learning problem in the context of classification. To address this challenge, we have developed a novel optimization framework (MultiC^2), which bypasses the standard two-step supervised learning procedure and learns the ensemble classifier directly using noisy and missing labels collected from crowdsourcing. Furthermore, we extend the solutions to multi-class settings and propose a fast gradient algorithm using RBCD. Then, we conduct several comparison experiments in terms of the effectiveness, robustness, and efficiency of our framework. The experimental results of three semi-synthetic data sets and one real data set have shown that our model outperforms the state-of-the-art techniques.

REFERENCES

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2006. Multi-Task Feature Learning. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*. 41–48.

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[3] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. 2010. A Singular Value Thresholding Algorithm for Matrix Completion. *SIAM Journal on Optimization* 20 (2010), 1956–1982.

[4] Emmanuel J. Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* (2009), 717–772.

[5] Rich Caruana. 1997. Multitask Learning. *Machine Learning* 28, 1 (1997), 41–75.

[6] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. 2016. FASCINATE: Fast Cross-Layer Dependency Inference on Multi-layered Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD*. 765–774.

[7] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. 2017. Cross-Dependency Inference in Multi-Layered Networks: A Collaborative Filtering Perspective. *TKDD* 11, 4 (2017), 42:1–42:26.

[8] Jianhui Chen, Jiayu Zhou, and Jieping Ye. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 42–50.

[9] Xi Chen, Jingrui He, Rick Lawrence, and Jaime G. Carbonell. 2012. Adaptive Multi-task Sparse Learning with an Application to fMRI Study. In *Proceedings of the Twelfth SIAM International Conference on Data Mining SDM*. 212–223.

[10] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence* 118, 1-2 (2000), 69–113.

[11] Peng Dai, Mausam, and Daniel S. Weld. 2010. Decision-Theoretic Control of Crowd-Sourced Workflows. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*.

[12] P. Dawid, A. M. Skene, A. P. Dawidt, and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics* (1979), 20–28.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[14] Jia Deng, Jonathan Krause, Michael Stark, and Fei-Fei Li. 2016. Leveraging the Wisdom of the Crowd for Fine-Grained Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 4 (2016), 666–676.

[15] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized Multi–task Learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 109–117.

[16] Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011. Learning a Kernel for Multi-Task Clustering. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*.

[17] Ruocheng Guo, Jundong Li, and Huan Liu. 2018. INITIATOR: Noise-contrastive Estimation for Marked Temporal Point Process. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*. 2191–2197.

[18] Jingrui He and Rick Lawrence. 2011. A Graphbased Framework for Multi-Task Multi-View Learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 25–32.

[19] Jingrui He, Yan Liu, and Qiang Yang. 2014. Linking Heterogeneous Input Spaces with Pivots for Multi-Task Learning. In *Proceedings of the 2014 SIAM International Conference on Data Mining SDM*. 181–189.

[20] Jingrui He and Yada Zhu. 2012. Hierarchical Multi-task Learning with Application to Wafer Quality Prediction. In *12th IEEE International Conference on Data Mining, ICDM*. 290–298.

[21] Yao Hu, Debing Zhang, Jun Liu, Jieping Ye, and Xiaofei He. 2012. Accelerated singular value thresholding for matrix completion. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 298–306.

[22] Pengfei Jiang, Weina Wang, Yao Zhou, Jingrui He, and Lei Ying. 2018. A Winners-Take-All Incentive Mechanism for Crowd-Powered Systems. In *Proceedings of the 13th Workshop on Economics of Networks, Systems and Computation, NetEcon@SIGMETRICS*. 3:1–3:6.

[23] David R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Budget-optimal crowdsourcing using low-rank matrix approximations. In *49th Annual Allerton Conference on Communication, Control, and Computing*. 284–291.

[24] Gabriella Kazai, Jaap Kamps, Marijn Koolen, and Natasa Milic-Frayling. 2011. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 205–214.

[25] Yaliang Li, Nan Du, Chaochun Liu, Yusheng Xie, Wei Fan, Qi Li, Jing Gao, and Huan Sun. 2017. Reliable Medical Diagnosis from Crowdsourcing: Discover Trustworthy Answers from Non-Experts. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM*. 253–261.

[26] Jun Liu, Shuiwang Ji, and Jieping Ye. 2009. Multi-Task Feature Learning Via Efficient l2, 1-Norm Minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*. 339–348.

[27] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2012. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 35 (2012), 208–220.

[28] Qiang Liu, Jian Peng, and Alexander T. Ihler. 2012. Variational Inference for Crowdsourcing. In *Advances in Neural Information Processing Systems (NIPS)*.

[29] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. FaitCrowd: Fine Grained Truth Discovery for Crowdsourced Data Aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD*. 745–754.

[30] Fenglong Ma, Chuishi Meng, Houping Xiao, Qi Li, Jing Gao, Lu Su, and Aidong Zhang. 2017. Unsupervised Discovery of Drug Side-Effects from Heterogeneous Data Sources. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD*. 967–976.

[31] Yurii Nesterov. 2012. Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems. *SIAM Journal on Optimization* 22 (2012).

[32] Tae Hyun Oh, Yasuyuki Matsushita, Yu-Wing Tai, and In-So Kweon. 2015. Fast randomized Singular Value Thresholding for Nuclear Norm Minimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4484–4493.

[33] Pentti Paatero and Unto Tapper. 1994. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* 5, 2 (1994), 111–126.

[34] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 22, 10 (2010), 1345–1359.

[35] Georg Rasch. 1961. On general laws and the meaning of measurement in psychology. (1961).

[36] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning From Crowds. *Journal of Machine Learning Research* 11 (2010), 1297–1322.

[37] Peter Richtárik and Martin Takác. 2014. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming* 144 (2014), 1–38.

[38] Nihar Bhadresh Shah and Denny Zhou. 2015. Double or Nothing: Multiplicative Incentive Mechanisms for Crowdsourcing. In *Advances in Neural Information Processing Systems (NIPS)*. 1–9.

[39] Chi Su, Fan Yang, Shiliang Zhang, Qi Tian, Larry S. Davis, and Wen Gao. 2015. Multi-Task Learning with Low Rank Attribute Embedding for Person Re-Identification. In *IEEE International Conference on Computer Vision (ICCV)*.

3739–3747.

[40] Paul Tseng. 2001. Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications* (2001), 475–494.

[41] A. Vedaldi and B. Fulkerson. 2008. VLFeat: An Open and Portable Library of Computer Vision Algorithms. http://www.vlfeat.org/. (2008).

[42] Jun Wu, Jingrui He, and Yongming Liu. 2018. ImVerde: Vertex-Diminished Random Walk for Learning Imbalanced Network Representation. In *IEEE International Conference on Big Data, Big Data*. 871–880.

[43] Yangyang Xu and Wotao Yin. 2013. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *SIAM J. Imaging Sciences* 6, 3 (2013), 1758–1789.

[44] Pei Yang and Jingrui He. 2016. Heterogeneous Representation Learning with Structured Sparsity Regularization. In *IEEE 16th International Conference on Data Mining (ICDM)*. 539–548.

[45] Chenwei Zhang, Sihong Xie, Yaliang Li, Jing Gao, Wei Fan, and Philip S. Yu. 2016. Multi-source Hierarchical Prediction Consolidation. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM*. 2251–2256.

[46] Lecheng Zheng, Yu Cheng, and Jingrui He. 2019. Deep Multimodality Model for Multi-task Multi-view Learning. *CoRR* abs/1901.08723 (2019).

[47] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth Inference in Crowdsourcing: Is the Problem Solved? *PVLDB* 10, 5 (2017), 541–552.

[48] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. 2018. SPARC: Self-Paced Network Representation for Few-Shot Rare Category Characterization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*. 2807–2816.

[49] Dengyong Zhou, John C. Platt, Sumit Basu, and Yi Mao. 2012. Learning from the Wisdom of Crowds by Minimax Entropy. In *Advances in Neural Information Processing Systems (NIPS)*.

[50] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD*. 655–664.

[51] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Clustered Multi-Task Learning Via Alternating Structure Optimization. In *Advances in Neural Information Processing Systems (NIPS)*. 702–710.

[52] Jiayu Zhou, Jianhui Chen, and J. Ye. 2011. *MALSAR: Multi-tAsk Learning via StructurAl Regularization.* Arizona State University. http://www.public.asu.edu/~jye02/Software/MALSAR

[53] Yao Zhou and Jingrui He. 2016. Crowdsourcing via Tensor Augmentation and Completion. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*.

[54] Yao Zhou and Jingrui He. 2017. A Randomized Approach for Crowdsourcing in the Presence of Multiple Views. In *IEEE International Conference on Data Mining, ICDM*. 685–694.

[55] Yao Zhou and Jiebo Luo. 2013. A practical method for counting arbitrary target objects in arbitrary scenes. In *Proceedings of the 2013 IEEE International Conference on Multimedia and Expo, ICME*. 1–6.

[56] Yao Zhou, Arun Reddy Nelakurthi, and Jingrui He. 2018. Unlearn What You Have Learned: Adaptive Crowd Teaching with Exponentially Decayed Memory Learners. In *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*. 2817–2826.

[57] Yao Zhou, Lei Ying, and Jingrui He. 2017. MultiC^2: an Optimization Framework for Learning from Task and Worker Dual Heterogeneity. In *Proceedings of SIAM International Conference on Data Mining (SDM)*.

[58] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML)*. 912–919.