MultiC\textsuperscript{2}: an Optimization Framework for Learning from Task and Worker Dual Heterogeneity

Yao Zhou\textsuperscript{*} \hspace{1cm} Lei Ying\textsuperscript{*} \hspace{1cm} Jingrui He\textsuperscript{*}

Abstract

Nowadays, crowdsourcing has been commonly used to enlist label information both effectively and efficiently. One major challenge in crowdsourcing is the diverse worker quality, which determines the accuracy of the label information provided by such workers. Motivated by the observation that in many crowdsourcing platforms, the same set of workers typically work on the same set of tasks, we propose to model the diverse worker quality by studying their behaviors across multiple related tasks. To this end, we propose an optimization framework named MultiC\textsuperscript{2} for learning from task and worker dual heterogeneity. It 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. We then propose an iterative algorithm to solve the optimization framework 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. Finally, we test the performance of MultiC\textsuperscript{2} on various data sets, and demonstrate its superiority over state-of-the-art crowdsourcing techniques.

Keywords:
Multi-task learning, Crowdsourcing, Tensor Representation

1 Introduction

A real world classification problem can often be viewed as a group of intrinsically correlated classification tasks that have a shared representation. This idea emerges in a wide range of applications of biomedical informatics [1], computer vision [2] and data science such as medical diagnosis, speech classification, image classification and spam filtering etc. The most important design in multi-task learning is to find the shared information among tasks. By exploiting the intrinsic relationships between tasks, multi-task learning can improve the generalization performance by learning these related tasks jointly.

Most multi-task learning methods focus on learning models under the supervised setting by given the ground truth labels. For real world applications, it usually requires large amounts of labeled examples for training, but the labor for the data labeling can be costly and time-consuming. In recent years, with the emergence of crowdsourcing services, researchers are able to collect large amounts of low-cost noisy labels in a very short time. In order to infer the ground truth labels from the large amounts of noisy labels, many solutions have been proposed: Majority Voting, Dawid and Skene EM method [5], Minimax conditional entropy method [6], variational inference using mean field [7], tensor augmentation and completion [8], etc. These crowdsourcing models can generally be divided into generative models [5, 6, 7] and discriminative models [17, 8]. Under some proper assumptions, many generative models perform well on real-world cases. 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 challenge, for the first time, we propose a novel structured multi-task classification approach using crowdsourcing labels (MultiC\textsuperscript{2}). Compared with the traditional multi-task learning setting, which requires ground truth labels, MultiC\textsuperscript{2} aims to leverage the structural information between the learned classifiers of multiple tasks, various workers, and extracted features. The main contributions of this paper 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

\textsuperscript{*}Arizona State University. Email: yzhou174@asu.edu; lei.ying.2@asu.edu; jingrui.he@asu.edu.
classifiers and the final prediction of a new data point is a weighted vote of their predictions.

- **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 two semi-synthetic data set and one real data set.

The rest of this paper is organized as follows. In Sections 2 and 3, we present our proposed problem formulation, followed by the optimization algorithm and ensemble method. The crowd label generating model is introduced in Section 4. In Section 5, we present the experimental results on both semi-synthetic and real data sets. The related work is reviewed in Section 6. Finally we conclude the paper in Section 7.

## 2 Multi-task Classification Using Crowdsourcing Labels

In this section, we first summarize the notations and then we formally present the definition of multi-task crowdsourcing problem.

### 2.1 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. For matrix indexing, we use $M(i,j)$ to denote the entry at the $i$-th row and $j$-th column of matrix $M$. Similar to Matlab’s notation, we 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: $||M||_t = \sum_i \sigma_i(M)$ and $\sigma_i(M)$ denotes the $i$-th singular 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 $\mathcal{X}_{i,:}$. A fiber of a three-way tensor is denoted as $\mathcal{X}_{ij,:}$ or $\mathcal{X}_{i,:}$. A 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., $\mathcal{X}_{(k)} = \text{unfold}_k(\mathcal{X})$. Similarly, the $\text{fold}_k(\mathcal{X}_{(k)})$ is the inverse operation of unfold and it returns the tensor $\mathcal{X}$.

### 2.2 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:

\[
\{(X_t(1,:), Y_t(1,:)), \ldots, (X_t(N_t,:), Y_t(N_t,:))\} \subset \mathbb{R}^{P} \times \mathbb{R}^{N_w}
\]

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 $W \in \mathbb{R}^{T \times N_w \times P}$ 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:

\[
f(y = 1|\mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}
\]

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 for all the tasks [22]. 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 plus logistic loss as the main principle:

\[
\min_W \sum_{t=1}^{T} \sum_{i=1}^{N_t} \sum_{j=1}^{N_w} \log(1 + e^{-Y_t(i,j) \cdot W_t^T X_t(i,:)}) + \text{rank}(W)
\]

\[+ R(W)\]

where $\text{rank}(W)$ is the tensor rank and $R(W) = ||W||_F^2$ is the tensor regularizer to prevent over-fitting. We use


<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_w$</td>
<td># of workers</td>
</tr>
<tr>
<td>$P$</td>
<td># of features</td>
</tr>
<tr>
<td>$N_t$</td>
<td># of examples in $t$-th task</td>
</tr>
<tr>
<td>$X_t$</td>
<td>data matrix of $t$-th task</td>
</tr>
<tr>
<td>$Y_t$</td>
<td>crowd label matrix of $t$-th task</td>
</tr>
</tbody>
</table>

Table 1: Summary of symbols
the logistic loss function $L(f(x), y) = \log(1 + e^{-y \cdot f(x)})$ in our model, although it can be naturally generalized to other loss functions. Logistic loss is a convex and monotonically decreasing function, therefore 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.

However the rank minimization problem is NP-hard and non-convex [9], one common approach is to use trace norm to approximate the rank, which has been proved to be the closest convex envelope of the rank. There are multiple ways to define the tensor trace norm, in our formulation, we use the definition proposed by Liu et al. [11]. 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:

$$
||X||_* = \sum_{l=1}^{3} \alpha_l ||X_{(l)}||_*
$$

(2.4)

s.t. : $\sum_{l=1}^{3} \alpha_l = 1$, $\alpha_l \geq 0$, $l = 1, ..., 3$

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 $W$ as the regularization term to prevent overfitting, the final formulation of multi-task crowd-sourcing problem becomes:

$$
\min_{M_l} \gamma \sum_{t=1}^{T} \sum_{i=1}^{N_t} \sum_{j=1}^{N_u} \log(1 + e^{-Y_t(i,j)W_{tj}^T X_t(i,:)})
$$

(3.5)

\begin{align*}
&+ \sum_{l=1}^{3} \alpha_l ||M_l||_* + \frac{\beta_l}{2} ||W||_F^2 + \frac{\lambda}{2} ||W||_F^2
\end{align*}

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 [19]. 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: $W, \{M_l\}$, where $l = 1, 2, 3$, because the weight tensors have the dimension of three. There are two major sub-problems that need to be solved in each BCD iteration: First sub-problem is to update the weight tensor $W$ while fixing the other intermediate matrices $M_1, M_2, M_3$; Second sub-problem is to update one intermediate matrix $M_l$ while $W$ and other intermediate matrices are fixed.

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

(3.6)

$$
\min_{M_l} : \frac{\alpha_l}{\beta_l} ||M_l||_* + \frac{1}{2} ||W||_F^2 - \frac{\lambda}{2} ||W||_F^2.
$$

This problem is extensively studied in many recent works [12, 20]. One of the earliest solutions of this problem, named singular value thresholding (SVT), is given by Cai et al. [12] as $D_{\tau}(W(l)) = U \Sigma_{\tau} V^T$. It needs to compute the SVD of matrix $W(l) = U \Sigma V^T$, then replaces $\Sigma$ with its shrinkage version: $\Sigma_{\tau} = \text{diag}(\{\sigma_i - \tau\}_+)$. Here $a_+ = \max(a, 0)$ and $\tau = \frac{\alpha}{\beta}$ is the threshold of shrinkage SVD.

3.2 Updating $W$: Similarly, with some simplification, the sub-problem of updating $W$ becomes:

(3.7)

$$
\min_{W} \gamma \sum_{t=1}^{T} \sum_{i=1}^{N_t} \sum_{j=1}^{N_u} \log(1 + e^{-Y_t(i,j)W_{tj}^T X_t(i,:)})
$$

(3.7)

\begin{align*}
&+ \frac{\beta_l}{2} ||W||_F^2 - \frac{\lambda}{2} ||W||_F^2 + \frac{\lambda}{2} ||W||_F^2
\end{align*}

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 single entry of the weight tensor $W_{ijp}$, the partial gradient of the logistic loss term is:

(3.8)

$$
\frac{\partial f(W)}{\partial W_{ijp}} = -\gamma \sum_{i=1}^{N_t} \frac{e^{-Y_t(i,:)X_t(i,:) \odot W_{ijp}}}{1 + e^{-Y_t(i,:)X_t(i,:) \odot W_{ijp}}} Y_t(i,j) X_t(i,p)
$$

$$
= -\gamma \left[ \frac{e^{-Y_t(i,:) \odot (X_t \odot W_{ijp})}}{1 + e^{-Y_t(i,:) \odot (X_t \odot W_{ijp})}} \odot Y_t(i,:,:) \right]^T \cdot X_t(,:,p)
$$

where the $A \odot B$ denotes the Hadamard product between matrices $A$ and $B$ of the same size. In a similar way, we can calculate the slice-wise (with respect to $W_{t;} \in \mathbb{R}^{N_w \times P}$) derivative of logistic loss as follows:

(3.9)

$$
\frac{\partial f(W)}{\partial W_{t;}} = -\gamma \left[ \frac{e^{-Y_t \odot (X_t \odot W_{t;})}}{1 + e^{-Y_t \odot (X_t \odot W_{t;})}} \odot Y_t \right]^T \cdot X_t
$$

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 $f(W)$ as shown in equation (3.9). As for the updating rules of relaxation
penalty term $RP(W)$ and regularization term $R(W)$, the gradients of them are calculated as below:

$$\frac{\partial RP(W)}{\partial W} = \sum_{l=1}^{3} \beta_l [W - f_{old}(M_l)]$$  \hspace{1cm} (3.10)

$$\frac{\partial R(W)}{\partial W} = \lambda W$$  \hspace{1cm} (3.11)

The proposed algorithm is summarized in Algorithm 1. It works as follows. We initialize the weight tensor $W$ with all zeros. In each BCD iteration, we first optimize over the set of intermediate matrices $\{M_l\}$ in problem (3.6) using the close-form solution and next optimize over $W$ in problem (3.7) using gradient descend. Then, upon convergence, entropy weights are learned separately for each task.

3.3 Computational complexity. The bottleneck of the algorithm is the $M_l$ updating step in problem (3.6), which involves the SVD computation of the unfolded matrix $W_l$. In general, the computation cost of SVD is $O(nr^2)$ for matrix $W_l$ of size $n \times r$. Using the accelerated SVT of [20], the complexity is further reduced to $O(rkr^2)$, where $k \ll r$ denotes the approximation of rank-$k$. If the BCD algorithm has $m$ iterations, then the complexity of our proposed algorithm is $O(mrk^2)$.

3.4 Worker ensemble with entropy weights. After the weight tensor $W$ 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 to take 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, $f(y|w_j)$ is the probabilistic classification function for $j$-th worker. After the optimization algorithm converges, we have a collection of learned models $\mathcal{F} = \{f(y|w_1), f(y|w_2), ..., f(y|w_{N_w})\}$. Then we proposed an entropy based ensemble method to combine this set of classifiers as follows: for each learned classifier, we treat the probability predictions $P_j(x_i) = f(y|x_i, w_j)$ 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:

$$H_j = -\frac{1}{N_t} \sum_{i=1}^{N_t} P_j(x_i) \log(P_j(x_i)) + (1 - P_j(x_i))\log(1 - P_j(x_i))$$  \hspace{1cm} (3.12)

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 coef-

---

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 $n = TP$.

**Algorithm 1 MultiC² Algorithm**

1. **Input:** training data $\mathcal{D}_{train}$, $\alpha, \beta, \gamma, \lambda, \epsilon$, MaxIter
2. **Output:** $W$, $T$ groups of ensemble coefficient $\{c_1, ..., c_{N_w}\}$
3. **Initialization:**
   $$W^0 = \{0\}^{T \times N_w \times P}$$
   On training data $\mathcal{D}_{train}$:
4. **for** $m = 1, 2, ..., $ MaxIter **do**
   $$\{M^m_l\} = \arg\min_{M_l} \frac{\alpha_l}{\beta_l} \|M_l\|_* + \frac{1}{2} \|W(l) - M_l\|_F^2$$
   $W^m = \arg\min_{W} \frac{\beta_l}{2} \|W(l) - M_l\|_F^2 + \frac{\lambda}{2} \|W\|_F^2$
   $$+ \gamma \sum_{t=1}^{T} \sum_{i=1}^{N_t} \sum_{j=1}^{N_w} \log(1 + e^{-Y_t(i,j) \cdot W^m_{tj} \cdot x_t(i,:)})$$
5. **if** $\|W^m - W^{m-1}\|_F / \|W^{m-1}\|_F \leq \epsilon$
6. **break**
7. **end if**
8. **end for**
9. **for** $j = 1, 2, ..., N_w$ **do**
10. $$c_j = \frac{1 - \kappa e^{H_j}}{\sum_{j=1}^{N_w} (1 - \kappa e^{H_j})}$$
11. **end for**
Cj = \frac{1 - e^{-H_j}}{\sum_{j=1}^{N_w} (1 - e^{-H_j})}

where \( \kappa = e^{-H_{\text{max}}} \) and \( H_{\text{max}} \) is the largest average entropy among all the workers. At last, when a new testing example \( x_{\text{test}} \) is given, the predicted label \( \hat{y} \) of it is:

\[
\hat{y} = \text{sign}(\sum_{j=1}^{N_w} c_j w_j^T x_{\text{test}})
\]

4 Crowd Labeling Model

In this section, we introduce the crowd labels generating model. 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 \( C \) is given, then each worker will have a worker ability matrix of size \( C \times C \) 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 as belonging 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 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 of 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 laymen 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. [13] as:

\[
p(d, a, 2) = \frac{1}{2} \left(1 + (1 - d)^{1-a}\right)
\]

We refer to the equation (4.15) as the Experts preferred model and equation (4.16) 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 \( A \in [0, 1]^{C \times C} \), if the ground truth label of an item \( y_{gt} = c, c \in \{1, \ldots, C\} \) is also given, the worker ability of a given item is defined as:

\[
a = A_{c,c}^{-1/C}.
\]

<table>
<thead>
<tr>
<th>Names</th>
<th>Model formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laymen and smart</td>
<td>( p(d, a, 2) = \begin{cases} \frac{1}{2}(1 + (1 - d)^{\frac{1}{a}}), &amp; a \geq 0 \ \frac{1}{2}(1 - d^{\frac{1}{a}}), &amp; a &lt; 0 \end{cases} )</td>
</tr>
</tbody>
</table>

Table 3: Modified crowd labeling model.

Many existing crowdsourcing models have a basic assumption: the workers are better than random behavior (e.g., the labeling accuracies of the workers are at least 50 percent 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 3.
5 Experimental Results

In this section, we first study the label inference performance of various crowdsourcing methods. Next, we evaluate the effectiveness, robustness and efficiency of our proposed algorithm. In the end, we compare our algorithm with the state-of-the-art on the real-world data set.

5.1 Semi-synthetic data set. There are two semi-synthetic multi-task learning data set have been generated. They are the subset of the 20 News Group data set, 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 [14]: 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. We eventually keep same top 150 TF-IDF features for each example. The details of these two data sets are shown in Table 2.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Tasks</th>
<th>Class 1</th>
<th>Class 2</th>
<th># examples (Pos/Neg)</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rec. vs. Talk</td>
<td>Task 1: rec.autos</td>
<td>talk.politics.guns</td>
<td>1844 (975/869)</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task 2: rec.sport.baseball</td>
<td>talk.politics.mideast</td>
<td>1545 (860/685)</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Comp. vs. Sci.</td>
<td>Task 1: comp.os.ms-windows.misc</td>
<td>sci.crypt</td>
<td>1875 (967/908)</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task 2: comp.sys.mac.hardware</td>
<td>sci.space</td>
<td>1827 (871/956)</td>
<td>150</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Details of Multi-task Crowdsourcing semi-synthetic data sets.

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 $C$. We use the semi-synthetic experiment, we have generated 50 workers. For each worker, we generate a $C \times C$ 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 $w$. If the ground truth label of an item $x$ is $+1$, then the corresponding item difficulty is $1 - P(y = -1|x, w)$; Otherwise, the corresponding item difficulty is $1 - P(y = +1|x, w)$. Since each worker does not have to label all items, for each worker, they 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 implementation.

5.1.1 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 workers. The details of these settings are listed in Table 4.

<table>
<thead>
<tr>
<th>Worker</th>
<th>setting 1</th>
<th>setting 2</th>
<th>setting 3</th>
<th>setting 4</th>
<th>setting 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>10%</td>
<td>5%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Layman</td>
<td>80%</td>
<td>85%</td>
<td>80%</td>
<td>70%</td>
<td>60%</td>
</tr>
<tr>
<td>Spammer</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>Smart adversary</td>
<td>0%</td>
<td>0%</td>
<td>10%</td>
<td>20%</td>
<td>30%</td>
</tr>
</tbody>
</table>

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

The details of results are shown in Figure 5(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.

5.1.2 Effectiveness. In the comparison experiment, we employed four methods: Single task learning (STL), which learns a classifier for each task using logistic regression with ground truth labels; Multi-task learning (MTL), which also uses logistic regression and add
\( L_{21} \) norm as the regularization term to do jointly feature learning with ground truth labels; Multi-task learning with MMCE inferred labels (MMCE+MTL) has exactly the same multi-task learning algorithm except that labels are inferred from MMCE model; Our proposed multi-task crowdsourcing classification (MultiC\(^2\)) directly learns the classifiers of multiple tasks using crowd labels.

Figure 5(e)-(h) summarizes the accuracies of these methods on two semi-synthetic data set. Each data set is randomly split into 50% training & 50% testing. The evaluation metric is the accuracy of the predicted labels in testing data and all the outcomes are the results of ten independent runs. As we can see, The accuracy 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 large performance drop in settings 3, 4 and 5. Our proposed method MultiC\(^2\) has consistently higher accuracies 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. MTL using logistic regression tends to be very sensitive to small changes in the data. Therefore, MultiC\(^2\), which uses the ensemble, could have a better performance.

### 5.1.3 Robustness and efficiency

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 found and assigned with very low ensemble weights. Figure 3 shows that the running time of our algorithm increases linearly with respect to the number of workers.

We also conduct a parameter study with respect to the \( \gamma \), 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 \( \gamma \) 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 accuracy of ten independent runs on a log-scaled \( \gamma \). As we can see in Figure 4, our proposed model has consistently better performance than Baseline method with varying \( \gamma \) values. The performance of Baseline has a significant drop when \( \gamma \) 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.

#### 5.2 Real data set

Real data set are designed for workers to label different animal images based on the animal type. The label of each animal in these images is either domestic or wild. In total, there are three labeling tasks: Task 1: Domestic cat vs. Wildcat; Task 2: Domestic canidae vs. Wild canidae; Task 3: Domestic horse vs. Wild horse. All 1438 images are selected...
from ImageNet [16] and we split the data set into 50% training & 50% testing. The crowdsourcing labels of training are collected from 31 real workers. Each image is resized to maximum side length of 300 pixels, then a three-level image pyramid (resize to 1, 1/2 and 1/4 scale) is generated, next the dense SIFT features of patch size 20 $\times$ 20 with overlapping of 10 pixels are extracted on the image pyramid. All dense SIFT features are further quantized using Bag-of-Visual-Word (BoVW) representation and eventually the top 110 TF-IDF features are used. The result is summarized in Figure 5, for this problem, our method is better than MTL using MMCE labels.

### 6 Related Work

**Multi-task learning.** One important branch of supervised learning is multi-task learning, which treats the target task and the source task in the same manner. By simultaneously learning all tasks, multi-task learning has shown great performance improvement in many related domains over the last two decades. Many existing multi-task learning methods [3, 4] are formulated as the regularized optimization problems and 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 formulations. Examples include multi-task with joint feature learning [4, 25] and sparse and low-rank multi-task learning [21], etc.

**Crowdsourcing.** The commercial use of crowdsourcing becomes popular in the recent decade, but this idea of dividing work among workers has successes for a long history. One of the earliest work on crowdsourcing is the Dawid-Skene EM model, which treats the worker ability confusion matrix as a latent vector and it is often referred as the two-coin model. However, their model ignores the item difficulty variations and has limitations in real-world applications. Later on, more variant methods have been proposed on top of their work. Raykar’s EM algorithm [18] impose a beta prior over the worker confusion matrix and it jointly learns the classifier and true labels together. Zhou’s minimax entropy model is able to further infer the true labels, item difficulty, and worker ability jointly. Compared with the existing models, our proposed low-rank regularization model imposes a structural correlations of task and worker heterogeneities. This type of low-rank principle also has

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MTL (MMCE label)</th>
<th>MultiC$^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat (Domestic vs. Wild)</td>
<td>0.7032</td>
<td>0.8128</td>
</tr>
<tr>
<td>Canidae (Domestic vs. Wild)</td>
<td>0.7821</td>
<td>0.8132</td>
</tr>
<tr>
<td>Horse (Domestic vs. Wild)</td>
<td>0.8099</td>
<td>0.8430</td>
</tr>
</tbody>
</table>

Table 5: Accuracies of various methods on real data set
been widely applied in various domains, such as social network analysis [10, 15]. Recently, there are also some works [23, 27, 26] that focus on designing and estimating the various abilities of the workers and learning to hire the workers that can learn over time.

7 Conclusion

In this paper, 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 from crowdsourcing. We also conduct several comparison experiments with respect to the effectiveness, robustness, and efficiency of our framework. The experimental results of semi-synthetic data set and real data set have shown that our model outperforms the state-of-the-art techniques.

Acknowledgements

This work is supported by the NSF research grant IIS-1552654, CNS-1618768, ECCS-1547294, ONR research grant N00014-15-1-2821, 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.

References

[10] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. FASCINATE: Fast Cross-Layer Dependency Inference on Multi-layered Networks. ACM International Conference on Knowledge Discovery and Data Mining (KDD), 2016.
[14] Quanquan Gu, Zhenhui Li, and Jiawei Han. Learning a Kernel for Multi-Task Clustering. Proceedings of AAAI Conference on Artificial Intelligence (AAAI), 2011.
[26] Dawei Zhou, Jingrui He, Yu Cao, and Jee-sun Seo. Bi-level Rare Temporal Pattern Detection. IEEE International Conference on Data Mining (ICDM), 2016.