# MiSC: Mixed Strategies Crowdsourcing

**Ching-Yun Ko**[1] , **Rui Lin**[1] , **Shu Li**[2] and **Ngai Wong**[1]
[1]The University of Hong Kong, Hong Kong
[2]Nanjing University, Nanjing 210023, China
{cyko, linrui, nwong}@eee.hku.hk, lis@smail.nju.edu.cn

## Abstract

Popular crowdsourcing techniques mostly focus on evaluating workers' labeling quality before adjusting their weights during label aggregation. Recently, another cohort of models regard crowdsourced annotations as incomplete tensors and recover unfilled labels by tensor completion. However, mixed strategies of the two methodologies have never been comprehensively investigated, leaving them as rather independent approaches. In this work, we propose *MiSC* (**Mi**xed **S**trategies **C**rowdsourcing), a versatile framework integrating arbitrary conventional crowdsourcing and tensor completion techniques. In particular, we propose a novel iterative Tucker label aggregation algorithm that outperforms state-of-the-art methods in extensive experiments.

## 1 Introduction

In recent years, with the advent of many complex machine learning models, the need for large amounts of labeled data is boosted. Though we can obtain unlabeled data abundantly and cheaply, acquiring labeled data from domain experts or well-trained workers with specific background knowledge is usually expensive and time-consuming. Crowdsourcing provides an effective way of collecting labeled data quickly and inexpensively. However, as crowd workers are usually non-experts or even spammers, the individual contribution from one worker can be unreliable. To improve the label quality, each item is presented to multiple workers. Therefore, inferring the true labels from a large sum of noisy labels is a key challenge in crowdsourcing.

Existing learning-from-crowds works mainly focus on eliminating annotations [Li and Jiang, 2018] from unreliable workers or reducing their weights. In line with this goal, a series of work focusing on evaluating workers' quality are proposed based on accuracy [Whitehill *et al.*, 2009; Karger *et al.*, 2011], and confusion matrix [Dawid and Skene, 1979; Raykar *et al.*, 2009; Liu *et al.*, 2012; Zhou *et al.*, 2012; Zhang *et al.*, 2016]. Apart from that, two emerged works [Zhou and He, 2016] and [Li and Jiang, 2018] aim at completing annotations by translating single/multi-label crowdsouring tasks to tensor completion problems. However, their improvements in performance over conventional annotations methodologies are limited. In this work, we present a label aggregation algorithm by mixing the two strategies: We view the label tensor as an incomplete and noisy tensor, which is coherent to the reality where incompleteness comes from heavy workloads and noise stems from mislabelling. In a nutshell, we capture the structural information and filter out noisy information from the label tensor through tensor completion. This is followed by a conventional label aggregation procedure. We iterate over the above two steps until convergence. Despite the generality of the proposed framework, we highlight our use of tensor decomposition techniques.

Tensors are a higher-order generalization of vectors and matrices and constitute a natural representation for many real-life data that are intrinsically multi-way. In analogy to the significance of matrix QR factorization and singular value decomposition in matrix preconditioning and principal component analysis, tensor decomposition concepts have been deployed in modern machine learning models. Iconic examples include text analysis [Collins and Cohen, 2012; Liu *et al.*, 2015], compression of neural networks [Denton *et al.*, 2014; Novikov *et al.*, 2015], and tensor completion [Suzuki, 2015; Imaizumi *et al.*, 2017]. Among various popular tensor decomposition techniques, we specifically have interests in the Tucker model [Tucker, 1963; Levin, 1963; De Lathauwer *et al.*, 2000a]. The Tucker decomposition factorizes a tensor into a core tensor of generally smaller size, along with factor matrices for each of the tensor modes. We exemplify through a case study and show that for a perfectly labeled matrix, its corresponding label tensor will have an intrinsically low-rank Tucker decomposition. The main contributions of this article are:

- A general mixed strategies framework for crowdsourcing tasks is proposed, in which annotations are completed by tensor recovery and ground-truth labels are estimated by conventional deduction algorithms.
- To our knowledge, this is the *first work* that introduces tensor decomposition methods to exploit the structural information in the label tensor. This scheme also bears a clear physical interpretation.
- Experimental results demonstrate that the proposed algorithms manage to improve existing methods to achieve higher aggregation accuracy.

- Most importantly, the proposed algorithms are shown to be particularly powerful compared to conventional pure label aggregation methods when the annotations are highly sparse and severely noisy. This is crucial since obtaining low-sparsity/ high-quality annotations can be arduous and expensive.

In the following, Section 2 briefly summarizes the related work. Then, Section 3 presents the necessary tensor preliminaries. Section 4 focuses on one of the proposed models, mixed low-rank Tucker DS-EM aggregation, and showcases the algorithm. Next, numerical experiments comparing the proposed MiSC (mixed strategies crowdsourcing) with pure label aggregation methods are given in Section 5. Finally, Section 6 concludes this work.

## 2 Related Work

Our two-stage mixed algorithms build upon the standard label aggregation algorithms. Majority voting, as the most straightforward crowdsourcing technique, has served as a baseline method for years. Besides, the seminal work of Dawid and Skene based on expectation maximization (EM), denoted as DS-EM henceforth, is also among the earliest work of crowdsourcing [Dawid and Skene, 1979] which is a generative method by assuming the performance of each worker is consistent across different tasks. Moreover, the crowdsourcing problem is further translated into a variational Bayesian inference problem of a graphical model in [Liu *et al.*, 2012]. Using Mean Field algorithm, the parameters are tuned to maximize the marginal likelihood. [Zhou *et al.*, 2012] introduces the minimax entropy principle into crowdsourcing tasks and infers true labels by minimizing the Kullback–Leibler (KL) divergence between the probability distribution over workers, items, labels, and the ground truth.

In matrix completion, most problems are formulated into the construction of a structurally low-rank matrix $\boldsymbol{X}$ having the same observed entries: $\min_{\boldsymbol{X}} \text{rank}(\boldsymbol{X})$, s.t. $(\boldsymbol{X}-\boldsymbol{A})_\Omega = \boldsymbol{0}$, where $\boldsymbol{A}$ represents the matrix with missing values filled by zeros and $\Omega$ is the mapping that specifies the locations of originally non-zero elements. Directly solving the above optimization problem is NP-hard, which results in extensive research on solving alternative formulations. One of the two popular candidates is to minimize the nuclear norm as the convex envelope of the matrix rank-operator [Candès and Recht, 2009; Chen, 2015]. This nuclear norm minimization idea is then further generalized to tensor completion problems by computing matricizations of the tensor along its $k$ modes and summing over the nuclear norm of the resulting $k$ matrices (abbreviated as LRTC in the remainder of this paper) [Liu *et al.*, 2013; Signoretto *et al.*, 2014]. Methods that exploit tensor decomposition formats were also introduced to tensor completion problems in recent years. In [Jain and Oh, 2014; Suzuki, 2015; Zhao *et al.*, 2015], the authors use the Canonical Polyadic (CP) decomposition for tensor estimators.

The pioneers of introducing tensor completion concepts to crowdsourcing problems are [Zhou and He, 2016] and [Li and Jiang, 2018], where authors build label aggregation algorithms upon the methodologies introduced in [Liu *et al.*, 2013]. This work contrasts with them in two main aspects: (1) *Aims* - we focus on introducing a versatile complete-aggregate two-step looping structure for crowdsourcing tasks, where completion techniques adopted can be of any kind. (2) *Approaches* - we introduce tensor decomposition driven completion algorithms and showcase its advantages over LRTC methods in crowdsourcing tasks.

## 3 Preliminaries

Tensors are high-dimensional arrays that generalize vectors and matrices. In this paper, boldface capital calligraphic letters $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots$ are used to denote tensors, boldface capital letters $\boldsymbol{A}, \boldsymbol{B}, \ldots$ denote matrices, boldface letters $\boldsymbol{a}, \boldsymbol{b}, \ldots$ denote vectors, and Roman letters $a, b, \ldots$ denote scalars. A $d$-way or $d$-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ is an array where each entry is indexed by $d$ indices $i_1, i_2, \ldots, i_d$. We use the convention $1 \leq i_k \leq I_k$ for $k = 1, \ldots, d$, where the $I_k$ is called the $k$th dimension of tensor $\boldsymbol{\mathcal{A}}$. MATLAB notation $\boldsymbol{\mathcal{A}}(i_1, i_2, \ldots, i_d)$ is used to denote entries of tensors. Fibers are high-dimensional analogue of rows and columns in matrices. In a matrix $\boldsymbol{A}$, a matrix column can be referred by fixing a column index. By again adopting MATLAB convention, the $i_2$th column of matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}(:, i_2)$, which is also called a 1-mode fiber of the matrix. Similarly, a matrix row $\boldsymbol{A}(i_1, :)$ is known as a 2-mode fiber of the matrix. For a $d$-way tensor, a $k$-mode fiber is determined by fixing all the indices but the $k$th one, which we denote as $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_d)$.

Tensor $k$-mode matricization, also known as tensor $k$-mode unfolding/flattening, reorders the elements of a $d$-way tensor into a matrix. It is formally defined as:

**Definition 1** *[Kolda and Bader, 2009, p. 459] The $k$-mode matricization of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, denoted by $\boldsymbol{A}_{(k)} \in \mathbb{R}^{I_k \times I_1 \cdots I_{k-1} I_{k+1} \cdots I_d}$, arranges the $k$-mode fibers to be the columns of the resulting matrix. Tensor element $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d)$ maps to matrix element $\boldsymbol{A}_{(k)}(i_k, j)$, where $j = 1 + \sum_{p=1, p \neq k}^{d} (i_p - 1) J_p$ and $J_p = \prod_{m=1, m \neq k}^{p-1} I_m$.*

The generalization of the matrix-matrix multiplication to tensor involves a multiplication of a matrix with a $d$-way tensor along one of its $d$ modes:

**Definition 2** *[Kolda and Bader, 2009, p. 460] The $k$-mode product of a tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$ with a matrix $\boldsymbol{U} \in \mathbb{R}^{J \times R_k}$ is denoted $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_k \boldsymbol{U}$ and defined by*

$$\boldsymbol{\mathcal{A}}(r_1, \cdots, r_{k-1}, j, r_{k+1}, \cdots, r_d) = \sum_{r_k=1}^{R_k} \boldsymbol{U}(j, r_k) \boldsymbol{\mathcal{G}}(r_1, \cdots, r_{k-1}, r_k, r_{k+1}, \cdots, r_d),$$

*where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{R_1 \times \cdots \times R_{k-1} \times J \times R_{k+1} \times \cdots \times R_d}$.*

Following Definition 2, the full multilinear product of a $d$-way tensor with $d$ matrices is:

**Definition 3** *[Cichocki* et al.*, 2015, p. 147] The full multilinear product of a tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$ with matrices $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}$, where $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$, is denoted $\boldsymbol{\mathcal{A}} = [[\boldsymbol{\mathcal{G}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}]]$ and defined by $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_d \boldsymbol{U}^{(d)}$, where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$.*

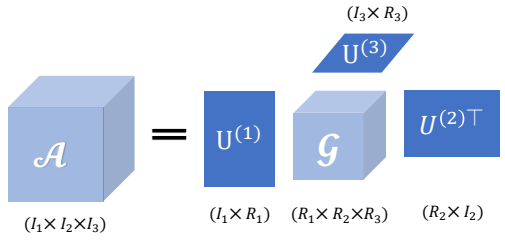


Figure 1: Graphical depiction of the Tucker decomposition of a 3-way tensor $\boldsymbol{\mathcal{A}}$. $\boldsymbol{\mathcal{G}}$ represents the core tensor and $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}$ are the factor matrices.

Following the definition of the full multilinear product, we define the Tucker decomposition as follows:

**Definition 4** *The Tucker decomposition decomposes a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ into a core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$ multiplied by a matrix $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ along the kth mode for $k = 1, \ldots, d$. That is, a $d$-way tensor $\boldsymbol{\mathcal{A}}$ is regarded as a multilinear transformation of a small core tensor $\boldsymbol{\mathcal{G}}$ by $d$ factor matrices $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}$. By writing out $\boldsymbol{U}^{(k)} = [\boldsymbol{u}_1^{(k)}, \boldsymbol{u}_2^{(k)}, \ldots, \boldsymbol{u}_{R_k}^{(k)}]$ for $k = 1, 2, \ldots, d$, we have*

$$\begin{aligned} \boldsymbol{\mathcal{A}} &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_d=1}^{R_d} \boldsymbol{\mathcal{G}}(r_1, \ldots, r_d)(\boldsymbol{u}_{r_1}^{(1)} \circ \cdots \circ \boldsymbol{u}_{r_d}^{(d)}), \quad (1) \\ &= \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_d \boldsymbol{U}^{(d)}, \\ &= [[\boldsymbol{\mathcal{G}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}]], \end{aligned}$$

*where $r_1, r_2, \ldots, r_d$ are auxiliary indices that are summed over, and $\circ$ denotes the outer product.*

The dimensions $(R_1, R_2, \ldots, R_d)$ of these auxiliary indices are called the Tucker ranks. It is worth noting that $R_k$ is in general no bigger than $rank(\boldsymbol{A}_{(k)})$, which is also called the multilinear rank. In other words, for a Tucker representation with Tucker ranks $(S_1, S_2, \ldots, S_d)$, if there exists a $k$, where $1 \leq k \leq d$ and $S_k > rank(\boldsymbol{A}_{(k)})$, then we can always find an equivalent Tucker representation with Tucker ranks no bigger than the multilinear ranks.

When the core tensor $\boldsymbol{\mathcal{S}}$ is cubical and diagonal, a Tucker model reduces to a CP model. By writing out the formula, a CP decomposition expresses a $d$-way tensor $\boldsymbol{\mathcal{A}}$ as

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \boldsymbol{\mathcal{G}}(r, \ldots, r)(\boldsymbol{u}_r^{(1)} \circ \cdots \circ \boldsymbol{u}_r^{(d)}).$$

# 4 When Label Aggregation Meets Tensor Completion

We now exemplify the proposed complete-aggregate two-step looping scheme using the case of mixed low-rank Tucker DS-EM aggregation. Firstly, we showcase in Section 4.1 that Tucker model bears a clear physical meaning in completing label tensor. In Section 4.2, we explain in detail how to combine a low-rank Tensor completion with traditional label aggregation to form one loop. The MiSC algorithms are then given in 4.3.

## 4.1 Basic Idea

We use $\boldsymbol{A} \in \mathbb{R}^{N_w \times N_i}$ to denote the label matrix, where $N_w$ is the number of workers and $N_i$ is the number of items. If item $i$ is labeled by worker $w$ as class $c$, then we have $\boldsymbol{A}(w, i) = c$. Now we proposed to form a three-way binary tensor $\boldsymbol{\mathcal{A}}$ of size $N_w \times N_i \times N_c$ by using the label matrix. This is done by using the canonical basis vectors to denote the non-zero elements in the matrix and all-zero vectors to denote unlabeled entries. Due to the fact that each worker gives at most one label for one item, there then contains at most one 1 in each of the 3-mode fibers of tensor $\boldsymbol{\mathcal{A}}$. We use a small example to illustrate this process.

**Example 1** *Suppose there are two workers labeling for three items among four classes. The first worker believes the first item and the third item belong to the first class and the fourth class, respectively. The second worker labels the first item as the first class and the second item as the third class. We use label matrix $\boldsymbol{A} \in \mathbb{R}^{2 \times 3}$ to describe the above setting, where*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 4 \\ 1 & 3 & 0 \end{pmatrix}.$$

*Considering matrix $\boldsymbol{A}$ contains four non-zero elements, four 3-mode fibers of the resulting tensor contain a 1, while the remaining two fibers are initiated with zeros vectors. For $\boldsymbol{A}(1,1) = 1$ and $\boldsymbol{A}(2,1) = 1$, the canonical basis vector $\boldsymbol{e}_1 = (1,0,0,0)^T$ is used. For $\boldsymbol{A}(1,3) = 4$ and $\boldsymbol{A}(2,2) = 3$, the canonical basis vectors $\boldsymbol{e}_4 = (0,0,0,1)^T$ and $\boldsymbol{e}_3 = (0,0,1,0)^T$ are needed respectively. Collecting these fibers renders the following tensor*

$$\boldsymbol{\mathcal{A}}(1,:,:) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \boldsymbol{\mathcal{A}}(2,:,:) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

We call the above-defined tensor $\boldsymbol{\mathcal{A}}$ label tensor. Our goal is to complete the label tensor $\boldsymbol{\mathcal{A}}$ by assuming a specific underlying structure. Hence the problem can be regarded as a tensor completion task. As is well-known, the quality of the label matrix is in general not guaranteed. Therefore, it is of great importance that the proposed method should not only fill in entries but also serve as a pre-processor to de-noise the label tensor before doing deduction.

Similar to the traditional matrix completion problem formulations, in this paper we propose to form the following optimization problem:

$$\min_{\boldsymbol{\mathcal{G}}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}} ||[[\boldsymbol{\mathcal{G}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}]] - \boldsymbol{\mathcal{A}}||, \quad (2)$$

$$\text{s.t. } \text{rank}_{\text{Tucker}}([[\boldsymbol{\mathcal{G}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}]]) = (R_1, R_2, R_3). \quad (3)$$

The optimization problem is solved via a truncated Tucker decomposition. It is worth noting that finding an exact Tucker decomposition of rank $(R_1, R_2, R_3)$, where $R_k = rank(\boldsymbol{A}_{(k)})$, of $\boldsymbol{\mathcal{A}}$ is easy. However, finding an optimal truncated Tucker decomposition is nontrivial [Kolda and Bader, 2009]. In this paper, the Tucker model is initialized using a truncated higher-order SVD [De Lathauwer *et al.*, 2000a] and updated by a higher-order orthogonal iteration algorithm [De Lathauwer *et al.*, 2000b]. We now use an extreme case to demonstrate the vivid physical meaning behind the choice of a Tucker model.

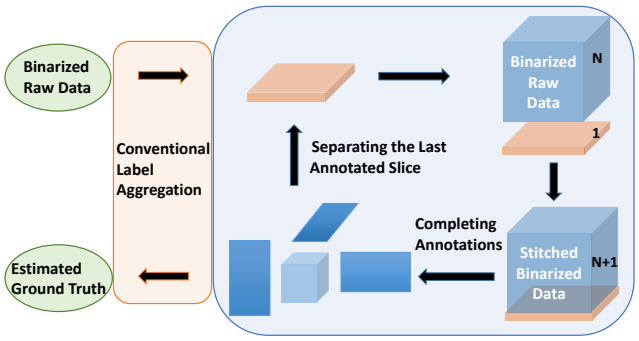


Figure 2: The MiSC work flow.

**Example 2** *Suppose the label matrix is perfectly given by two workers for three items among four classes as follows*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 3 & 4 \\ 1 & 3 & 4 \end{pmatrix}.$$

*The label tensor is then constructed as*

$$\boldsymbol{\mathcal{A}}(1,:,:) = \boldsymbol{\mathcal{A}}(2,:,:) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

*Now we consider the exact Tucker decomposition of this label tensor. By taking three different mode matricizations, we obtain* $rank(\boldsymbol{\mathcal{A}}_{(1)}) = 1, rank(\boldsymbol{\mathcal{A}}_{(2)}) = 3, rank(\boldsymbol{\mathcal{A}}_{(3)}) = 3$*, where* $\min(N_i, N_c) = \min(3, 4) = 3$.

As is demonstrated in the above example, if given a noiseless label tensor, it can be naturally decomposed into an exact low-rank Tucker format with Tucker ranks equal to $(1, \min(N_i, N_c), \min(N_i, N_c))$.

## 4.2 Methodology

In general, the proposed MiSC consists of two phases: label tensor completion and deduction. In each phase, a wide range of algorithms are available as will be seen in Section 5. We visualize the architecture of MiSC in Figure 2.

The method starts from a binarized label tensor constructed as described in Section 4.1. A conventional label aggregation algorithm is adopted to infer a $1 \times N_i$ resultant initial guess of the ground-truth labels. Similar to the idea explained in Example 1, the initial guess can then be encoded to a $1 \times N_i \times N_c$ slice $\boldsymbol{\mathcal{S}}$ and concatenated with the raw binarized label tensor. By doing so, we enlarge the size of the label tensor from $N_w \times N_i \times N_c$ to $(N_w + 1) \times N_i \times N_c$ and form a target tensor $\boldsymbol{\mathcal{T}} = [\boldsymbol{\mathcal{A}}; \boldsymbol{\mathcal{S}}]$, which follows by completion iterations. The all-zero 3-mode fibers of tensor $\boldsymbol{\mathcal{A}}$ are filled in during the annotation completion step as depicted in (2) and (3). At the same time, the imposed low-rank constraint also automatically smooths the noisy label tensor. When the stopping criteria are not satisfied, the bottom annotated slice of the completed tensor is separated and concatenated again with the binarized raw data to form an iterative refinement loop.

## 4.3 Algorithms

Before proposing the pseudocode of the MiSC algorithms, we will briefly introduce the truncated higher-order SVD (HOSVD) procedure [De Lathauwer *et al.*, 2000a] (denoted as Algorithm 1) and the higher-order orthogonal iteration (HOOI) algorithm [De Lathauwer *et al.*, 2000b] (denoted as Algorithm 2), which serve as two cornerstones for the low-rank Tucker completion.

**Algorithm 1** Truncated higher-order singular value decomposition (SVD)

**Input:** Tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, ranks: $R_1, \ldots, R_d$
**Output:** Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$, factor matrices $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}$, where $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for $1 \leq k \leq d$.
**for** $i = 1$ **to** $d$ **do**
  $[\boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R}^T] \leftarrow$ SVD decomposition of $\boldsymbol{T}_{(i)}$
  $\boldsymbol{U}^{(i)} \leftarrow R_i$ leading column vectors of $\boldsymbol{L}$
**end for**
$\boldsymbol{\mathcal{G}} \leftarrow [[\boldsymbol{\mathcal{T}}; \boldsymbol{U}^{(1)T}, \boldsymbol{U}^{(2)T}, \ldots, \boldsymbol{U}^{(d)T}]]$

**Algorithm 2** Higher-order orthogonal iteration

**Input:** Tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, initial rank: $R^0$, ranks: $R_1, \ldots, R_d$.
**Output:** Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$, factor matrices $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}$, where $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for $1 \leq k \leq d$.
Initial factor matrices $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R^0}$ for $1 \leq k \leq d$ using Algorithm 1 with $\boldsymbol{\mathcal{T}}$ and $R_1^0 = \ldots = R_d^0 = R^0$ as the input.
**while** stopping criteria not satisfied **do**
  **for** $i = 1$ **to** $d$ **do**
    $\boldsymbol{\mathcal{A}} \leftarrow \boldsymbol{\mathcal{T}} \times_1 \boldsymbol{U}^{(1)T} \cdots \times_{i-1} \boldsymbol{U}^{(i-1)T} \times_{i+1} \boldsymbol{U}^{(i+1)T} \cdots \times_d \boldsymbol{U}^{(d)T}$
    $[\boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R}^T] \leftarrow$ SVD decomposition of $\boldsymbol{A}_{(i)}$
    $\boldsymbol{U}^{(i)} \leftarrow R_i$ leading column vectors of $\boldsymbol{L}$
  **end for**
**end while**
$\boldsymbol{\mathcal{G}} \leftarrow [[\boldsymbol{\mathcal{T}}; \boldsymbol{U}^{(1)T}, \boldsymbol{U}^{(2)T}, \ldots, \boldsymbol{U}^{(d)T}]]$

The ultimate goal of Algorithms 1 and 2 is to find a (nearly) optimal low-rank Tucker decomposition approximation of the target tensor $\boldsymbol{\mathcal{T}}$. While finding an exact Tucker decomposition of the multilinear ranks is rather straightforward by directly employing HOSVD algorithm, the Tucker decomposition of ranks smaller than the multilinear ranks found by truncated HOSVD is demonstrated to be non-optimal in [Kolda and Bader, 2009], in terms of the norm of the difference. Hence we only use a truncated HOSVD step as an initialization procedure. Algorithm 1 takes in a tensor $\boldsymbol{\mathcal{T}}$ and prescribed Tucker ranks as inputs, and followed by consecutive SVDs operating on different mode matricizations. Factor matrices of the Tucker model are determined by the leading left-singular vectors within each SVD step. Lastly, the core tensor $\boldsymbol{\mathcal{G}}$ is computed through the full multilinear product of the tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ with matrices $\boldsymbol{U}^{(1)T}, \boldsymbol{U}^{(2)T}, \ldots, \boldsymbol{U}^{(d)T}$. An initial Tucker decomposition is thereby obtained. It is worth noting that in the HOSVD algorithm, the mode matricizations are always computed directly from the input tensor.

**Algorithm 3** Mixed Strategies Crowdsourcing (MiSC)

**Input:** Label matrix $\boldsymbol{A} \in \mathbb{R}^{N_w \times N_i}$, prior statistics $\boldsymbol{\mathcal{S}}$, initial rank: $R^0$, ranks: $R_1, R_2, R_3$.
**Output:** Inferred true labels.
Find $N_c$ by checking the maximum entry of $\boldsymbol{A}$
Initialize conventional aggregation result $\boldsymbol{s}$ from $\boldsymbol{A}$
Construct $\boldsymbol{\mathcal{A}}$ through the following for-loop:
**for** $c = 1$ **to** $N_c$ **do**
  $\boldsymbol{\mathcal{A}}(:,:,c) \leftarrow \text{double}(\boldsymbol{A} == c)$
**end for**
**while** stopping criteria not satisfied **do**
  $\boldsymbol{\mathcal{S}} \leftarrow$ binarize $\boldsymbol{s}$ as discussed in Example 1
  $\boldsymbol{\mathcal{T}} \leftarrow$ concatenate $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{S}}$
  $[\boldsymbol{\mathcal{G}}, \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}] \leftarrow \text{HOOI}(\boldsymbol{\mathcal{T}}, R^0, R_1, R_2, R_3)$
  $\boldsymbol{\mathcal{L}} \leftarrow [[\boldsymbol{\mathcal{G}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}]]$
  $\boldsymbol{s} \leftarrow$ separate the last slice from $\boldsymbol{\mathcal{L}}$
**end while**
$\hat{\boldsymbol{A}} \leftarrow$ choose the indices of the maximum values of all 3-mode vectors in $\boldsymbol{\mathcal{L}}$
Infer the true labels from the completed label matrix $\hat{\boldsymbol{A}}$ by conventional label aggregation techniques

Hence the computation of each of the factor matrices is completely independent of other factor matrices.

The higher-order orthogonal iteration (HOOI) algorithm was proposed in [De Lathauwer *et al.*, 2000b] to give a low-rank Tucker decomposition with a smaller norm of the difference, without guarantees to converge to the global optimum. In Algorithm 2, a Tucker decomposition of prescribed initial ranks is firstly constructed using Algorithm 1. Then the updates follow an alternating linear scheme (ALS): the factor matrices are updated iteratively by updating one at a time and keeping all the others fixed. In contrast to HOSVD, the update of a factor matrix in HOOI is not independent of others. Specifically, the $i$-mode matricization $\boldsymbol{A}_{(i)}$ in HOOI is computed from tensor $\boldsymbol{\mathcal{A}}$, where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{R_1 \times \cdots \times R_{i-1} \times I_i \times R^0 \times \cdots \times R^0}$ in the first ALS sweep (a full for-loop corresponds to one sweep). Starting from the second ALS sweep, the size of tensor $\boldsymbol{\mathcal{A}}$ before $i$-mode matricization becomes $R_1 \times \cdots \times R_{i-1} \times I_i \times R_{i+1} \times \cdots \times R_d$. One can update the Tucker decomposition for a fixed amount of sweeps or until the residual stops decreasing.

We summarize the proposed MiSC algorithms with an exemplary Tucker completion case in Algorithm 3. The for-loop realizes the label matrix to label tensor conversion process explained in Example 1. Within the while-loop, HOOI subroutines are employed to fill in and augment the label tensor. One can iterate over the filling process for a prescribed amount of sweeps or until the last slice stops evolving. After the loops, we binarize the completed 3-way tensor which then follows by a deduction step. The binarization is realized by choosing the indices of the maximum values in each of the 3-mode vectors. A new $(N_w + 1) \times N_i$ label matrix $\hat{\boldsymbol{A}}$ is thereby constructed. The most computationally expensive steps in Algorithm 3 are SVDs in the HOSVD and HOOI subroutines, which have computational complexities of approximately $O((I_1 I_2 I_3)^2 / I_i)$ and $O(I_i (R^0)^4)$ flops, respectively.

# 5 Experimental Results

In this section, the proposed mixed complete-aggregate strategies crowdsourcing algorithms are compared with conventional label aggregation methods on six popular datasets, including *Web* dataset [Zhou *et al.*, 2012], *BM* dataset [Mozafari *et al.*, 2014], *RTE* dataset [Snow *et al.*, 2008], *Dog* dataset [Deng *et al.*, 2009; Zhou *et al.*, 2012], *Temp* dataset [Snow *et al.*, 2008], and *Bluebirds* dataset [Welinder *et al.*, 2010]. Details of their nonzero rates defined by $\frac{\#\text{worker labels}}{\#\text{items} \times \#\text{workers}}$, and annotation error rates defined by $\frac{\#\text{wrong labels}}{\#\text{worker labels}}$ are recorded in Table 1.

Five conventional crowdsourcing methods are considered in the aggregation step of the proposed MiSC. They are: Majority Voting (MV), Dawid-Skene model + Expectation Maximization (DS-EM), Dawid-Skene model + mean field (DS-MF) [Liu *et al.*, 2012], and Categorical/ Ordinal Minimax Conditional Entropy ($\text{MMCE}_{(\text{C})}$, $\text{MMCE}_{(\text{O})}$).

We consider three tensor completion algorithms in the MiSC algorithms: LRTC[1], TenALS[2], and Tucker. These methods represent three different approaches towards the completion problem. LRTC [Liu *et al.*, 2013] aims at minimizing the sum of nuclear norms of the unfolded matrices. While TenALS [Jain and Oh, 2014] and Tucker utilize CP decomposition and Tucker factorization, respectively.

## 5.1 Comparison with State-of-the-Arts

The estimation errors of all five pure conventional label aggregation algorithms and fifteen MiSC approaches on six real-life datasets are reported in Table 1. It is shown that the proposed MiSC algorithms achieve lower estimation errors than traditional pure crowdsourcing methods on *all* six datasets. We observe the greatest breakthrough in the Web dataset, where the state-of-the-art estimation errors of around $10.33\%$ are brought down to around $5.24\%$. The second noticeable refinement is in the Bluebirds dataset, where MiSC algorithms produce error rates lower than $5\%$, in contrast to the $> 8\%$ state-of-the-art records. For the BM, RTE, Dog datasets, the MiSC algorithm via DS-MF+Tucker also reaches the lowest errors of $26.2\%$, $6.75\%$, and $15.37\%$, respectively, among others.

## 5.2 Pure vs. Mixed Strategies Crowdsourcing

Besides evaluating all MiSC algorithms together with pure label aggregation methods, we also zoom in on pairwise comparisons between pure and mixed strategies that use the same label deduction approaches. Specifically, in the Web dataset, the MiSC algorithms stacked by DS-EM/ DS-MF and Tucker completion have estimation errors of $5.77\%/5.73\%$, compared with the initial $16.92\%/16.10\%$ by pure DS-EM/ DS-MF. In the Temp dataset, both DS-EM+Tucker and DS-MF+Tucker improve the performance of their corresponding pure counterparts by approximately one point. In the Bluebirds dataset, mixing MMCE aggregations with Tucker completion helps the crowdsourcing reach the lowest error rate of $4.63\%$ compared with the original $8.33\%$.

[1] http://www.cs.rochester.edu/u/jliu/code/TensorCompletion.zip
[2] http://web.engr.illinois.edu/~swoh/software/optspace

| **Web** (3.3/ 63.4) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ | **BM** (6.0/ 31.1) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| pure | 26.93 | 16.92 | 16.10 | 11.12 | 10.33 | pure | 30.4 | 27.60 | 26.90 | 27.10 | 27.10 |
| LRTC | 26.76 | 16.55 | 16.09 | 11.12 | 10.33 | LRTC | 29.25 | 27.60 | 26.90 | 27.10 | 27.10 |
| TenALS | 26.93 | 16.77 | 15.83 | 11.12 | 10.33 | TenALS | 27.60 | 27.60 | 26.90 | 27.10 | 27.10 |
| Tucker | 10.87 | 5.77 | 5.73 | 6.97 | **5.24** | Tucker | 26.50 | 27.00 | **26.20** | 26.40 | 26.40 |
| **RTE** (6.1/ 16.3) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ | **Dog** (9.2/ 30.0) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ |
| pure | 10.31 | 7.25 | 7.13 | 7.50 | 7.50 | pure | 17.78 | 15.86 | 15.61 | 16.23 | 16.73 |
| LRTC | 9.25 | 7.25 | 7.00 | 7.50 | 7.50 | LRTC | 15.61 | 15.61 | 15.61 | 15.61 | 15.61 |
| TenALS | 10.25 | 7.25 | 7.13 | 7.50 | 7.50 | TenALS | 15.86 | 15.74 | 15.61 | 15.86 | 15.86 |
| Tucker | 8.38 | 6.88 | **6.75** | 7.50 | 7.50 | Tucker | 15.61 | 15.49 | **15.37** | 15.86 | 15.86 |
| **Temp** (13.2/ 15.9) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ | **Bluebirds** (100.0/ 36.4) | MV | DS-EM | DS-MF | $\text{MMCE}_{(C)}$ | $\text{MMCE}_{(O)}$ |
| pure | 6.39 | 5.84 | 5.84 | 5.63 | 5.63 | pure | 24.07 | 10.19 | 10.19 | 8.33 | 8.33 |
| LRTC | 5.19 | 5.63 | 5.63 | 5.63 | 5.63 | LRTC | 20.37 | 9.26 | 9.26 | 6.48 | 6.48 |
| TenALS | 5.41 | 5.63 | 5.84 | 5.63 | 5.63 | TenALS | 23.15 | 9.26 | 9.26 | 6.48 | 6.48 |
| Tucker | 5.19 | **4.98** | **4.98** | 5.41 | 5.41 | Tucker | 19.91 | 8.33 | 9.26 | **4.63** | **4.63** |

Table 1: Estimation errors (%) of pure and mixed strategies on Web, BM, RTE, Gog, Temp, and Bluebirds datasets. Nonzero rates and annotation error rates of datasets are given after their names ($\cdot\%/\cdot\%$). As an example, the lowest estimation error in the Web dataset comes from the low-rank Tucker completion + $\text{MMCE}_{(O)}$ aggregation strategies.

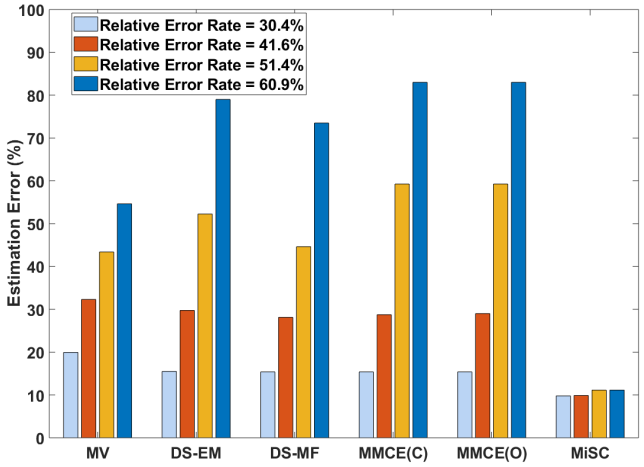


Figure 3: Estimation errors (%) of pure and mixed strategies on highly sparse and severely noisy annotations in the RTE dataset.

Consequently, we have empirically verified that the proposed MiSC algorithms can consistently improve the performance on top of conventional label aggregation schemes. By and large, we also remark that this complete-aggregate two-step looping structure is readily compatible with other state-of-the-art label deduction and tensor completion algorithms.

### 5.3 MiSC for Sparse and Noisy Annotations

As noticed in Section 5.1, although MiSC successfully improves the annotation quality in all six datasets, the striking refinement obtained in Web dataset stands out and raises the question: when will MiSC be remarkably advantageous? Referencing Table 1, one can see that the Web dataset has the sparsest and "poorest" annotations. Only 88 items out of 2665 are labeled on average per worker, and 63.4% of the total 15567 labels are misleading. We emulate similar scenarios by simultaneously eliminating worker labels from and adding noise to the RTE dataset. The resultant label matrix has a nonzero rate of 3.7%, and four different degrees of annotations error rates: $30.4\%, 41.6\%, 51.4\%, 60.9\%$. Figure 3 records the corresponding error rates. We can then see that MiSC is significantly more robust to high sparsity and severe noise. Replacing reliable annotations by erroneous labels only slightly affects the accuracy of MiSC, while the performance of traditional pure label aggregation degrades rapidly. We highlight this important finding since the number and quality of worker labels have a huge implication on the crowdsourcing cost. To this end, the proposed MiSC approach provides a means to substantially relax the labor and labeling quality requirements while maintaining superior accuracy.

## 6 Conclusion

This paper has introduced the mixed strategies crowdsourcing (MiSC) framework, a versatile complete-aggregate two-step iterative procedure, for crowdsourcing tasks. MiSC is readily compatible with various completion techniques and deduction schemes. By integrating tensor completion procedures, and importantly, tensor decomposition methods, into label aggregation, the proposed methods can largely improve the crowdsourcing performance. By further assuming a low-rank Tucker structure, the mixed low-rank Tucker model with conventional label aggregation approaches are shown to be particularly favorable when the annotations are highly sparse and severely noisy. Experiments have shown that MiSC consistently outperforms state-of-the-art methods in terms of estimation errors.

## Acknowledgments

This work is partially supported by the General Research Fund (Project 17246416) of the Hong Kong Research Grants Council.

## References

[Candès and Recht, 2009] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[Chen, 2015] Yudong Chen. Incoherence-optimal matrix completion. *IEEE Transactions on Information Theory*, 61(5):2909–2923, 2015.

[Cichocki *et al.*, 2015] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Anh-Huy Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

[Collins and Cohen, 2012] Michael Collins and Shay B. Cohen. Tensor decomposition for fast parsing with latent-variable PCFGs. In *NIPS*, 2012.

[Dawid and Skene, 1979] Alexander P. Dawid and Allan M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[De Lathauwer *et al.*, 2000a] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[De Lathauwer *et al.*, 2000b] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[Denton *et al.*, 2014] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[Imaizumi *et al.*, 2017] Masaaki Imaizumi, Takanori Maehara, and Kohei Hayashi. On tensor train rank minimization: Statistical efficiency and scalable algorithm. In *NIPS*, 2017.

[Jain and Oh, 2014] Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. In *NIPS*, 2014.

[Karger *et al.*, 2011] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 2011.

[Kolda and Bader, 2009] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[Levin, 1963] Joseph Levin. *Three-Mode Factor Analysis*. PhD thesis, University of Illinois at Urbana-Champaign, 1963.

[Li and Jiang, 2018] Shao-Yuan Li and Yuan Jiang. Multi-label crowdsourcing learning with incomplete annotations. In *PRICAI*, 2018.

[Liu *et al.*, 2012] Qiang Liu, Jian Peng, and Alexander T. Ihler. Variational inference for crowdsourcing. In *NIPS*, 2012.

[Liu *et al.*, 2013] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[Liu *et al.*, 2015] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *IJCAI*, 2015.

[Mozafari *et al.*, 2014] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *PVLDB*, 8(2):125–136, 2014.

[Novikov *et al.*, 2015] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. In *NIPS*, 2015.

[Raykar *et al.*, 2009] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Anna Jerebko, Charles Florin, Gerardo H. Valadez, Luca Bogoni, and Linda Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*, 2009.

[Signoretto *et al.*, 2014] Marco Signoretto, Quoc Tran Dinh, Lieven De Lathauwer, and Johan A. K. Suykens. Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, 94(3):303–351, 2014.

[Snow *et al.*, 2008] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[Suzuki, 2015] Taiji Suzuki. Convergence rate of Bayesian tensor estimator and its minimax optimality. In *ICML*, 2015.

[Tucker, 1963] Ledyard R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137, 1963.

[Welinder *et al.*, 2010] Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. The multidimensional wisdom of crowds. In *NIPS*, 2010.

[Whitehill *et al.*, 2009] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009.

[Zhang *et al.*, 2016] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *The Journal of Machine Learning Research*, 17(1):3537–3580, 2016.

[Zhao *et al.*, 2015] Q. Zhao, L. Zhang, and A. Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1751–1763, 2015.

[Zhou and He, 2016] Yao Zhou and Jingrui He. Crowdsourcing via tensor augmentation and completion. In *IJCAI*, 2016.

[Zhou *et al.*, 2012] Denny Zhou, Sumit Basu, Yi Mao, and John C. Platt. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, 2012.