Ensemble Learning from Crowds

Jing Zhang, *Member, IEEE*, Ming Wu, and Victor S. Sheng, *Senior Member, IEEE*

Abstract—Traditional learning from crowdsourced labeled data consists of two stages: inferring true labels for instances from their multiple noisy labels and building a learning model using these instances with the inferred labels. This straightforward two-stage learning scheme suffers from two weaknesses: (1) the accuracy of inference may be very low; (2) useful information may be lost during inference. In this paper, we proposed a novel ensemble method for learning from crowds. Our proposed method is a meta-learning scheme. It first uses a *bootstrapping* process to create M sub-datasets from an original crowdsourced labeled dataset. For each sub-dataset, each instance is duplicated with different weights according to the distribution and class memberships of its multiple noisy labels. A base classifier is then trained from this extended sub-dataset. Finally, unlabeled instances are predicted by aggregating the outputs of these M base classifiers. Because the proposed method gets rid of the inference procedure and uses the full dataset to train learning models, it preserves the useful information for learning as much as possible. Experimental results on nine simulated and two real-world crowdsourcing datasets consistently show that the proposed ensemble learning method significantly outperforms five state-of-the-art methods.

Index Terms—Bagging, classification, crowdsourcing, ensemble learning, learning from crowds

1 Introduction

The emergence of crowdsourcing provides a convenient and low cost solution to obtain class labels for training instances in supervised learning. However, due to the uncertainty of non-expert labelers, the quality of collected labels cannot be guaranteed, which has a negative impact on the subsequent model learning. In order to improve the quality of labels used in model training, we usually let each instance be labeled by multiple different labelers, and then an integrated label for each instance is inferred from these multiple noisy labels. We hope that the integrated labels for the training instances match their unknown true labels very well, so that the quality of a learning model trained from them could be improved. Sheng et al. [1] first systematically studied this problem and showed that even the simplest majority voting algorithm can improve the quality of the integrated labels and eventually improve the quality of the learned model. Since then, this two-stage learning scheme has been widely applied to different applications [2], [3], since it has two relatively independent steps, i.e., inference and model training, which is easy to be separately developed in practice. Obviously, under this scheme, if an inference algorithm (such as majority voting) can provide high-quality integrated labels for the training sample, the performance of learned models will be improved given using the off-the-shelf learning methods. Thus, developing new inference algorithms outperforming majority voting has been a fundamental research topic in crowdsourcing during the past decade [4].

A major problem of inferring true labels of instances by majority voting lies on the fact that it cannot deal with the complexity of crowdsourced labeling, such as biases of labeling, spam and adversarial labeling, expertise and intentions of labelers, difficulties of instances, and so on. Therefore, researchers made a lot of efforts in the past several years to design better inference algorithms [2], [5], [6], [7], [8], [9] which model the crowdsourced labeling from different aspects. For example, GLAD [8] models both the levels of expertise of labelers and the difficulties of instances, treating the probability of an instance being positive as a latent variable. A Bayesian approach RY [5] adds a specific prior to each class and models the biases of a labeler towards positive and negative classes as two parameters: sensitivity and specificity. ZenCrowd [2] only uses one parameter to model the reliability of labelers. By modeling the complexity of the crowd labeling, all of these inference algorithms were expected to achieve a high accuracy. However, a comprehensive empirical study on many real-world crowdsourcing datasets [10] has revealed some depressing facts that none of these algorithms significantly outperforms the others in most cases, and under a situation that the labelers exhibit low qualities, all of them are mediocre. Therefore, the performance of two-stage learning scheme can hardly be improved by designing more accurate inference algorithms.

In this paper, we propose a novel general ensemble method for learning from crowds, which does not infer the true labels of training instances and directly builds learning models from the crowdsourced labeled data to predict class labels of unlabeled instances. The motivations of the proposed method lie in three aspects. First, crowdsourcing scenario might be the lack of sufficient ground truth. Thus, in

- *J. Zhang and M. Wu are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Street, Nanjing 210094, P. R. China. E-mail: jzhang@njust.edu.cn, 349535311@qq.com.*
- *V.S. Sheng is with the Department of Computer Science, University of Central Arkansas, Conway, AR 72035. E-mail: ssheng@uca.edu.*

Manuscript received 2 July 2017; revised 31 May 2018; accepted 25 July 2018. Date of publication 31 July 2018; date of current version 3 July 2019. (Corresponding author: Jing Zhang.) Recommended for acceptance by N. Chawla. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2018.2860992

traditional two-stage learning scheme, truth inference is necessary. However, the accuracy of truth inference can hardly be improved in an agnostic scenario [10] and always restricts the improvement of the quality of learned models [11]. Second, useful information may be lost during the inference procedure, which is probably utilized by some classification algorithms (such as cost-sensitive decision tree and neural network) to obtain a better-learned model. For example, when an instance $\mathbf{x}_i$ with a multiple noisy label set $\{+++--\}$ is inferred as $\mathbf{x}_i\{+\}$ by majority voting, the proportions of different judgments that exactly reflect the whole picture of its class membership is missing. However, a neural network can simultaneously accept $\mathbf{x}_i\{+, 0.6\}$ and $\mathbf{x}_i\{-, 0.4\}$ and store the information as weights on the connections among its internal neurons, which makes it not only robust to label noises but also performs well [12]. Finally, ensemble learning has shown a significant advantage when dealing with uncertainties on instances [13]. Our proposed method is a meta-learning scheme, which is easily adapted for different kinds of features if a learner is carefully chosen. Generally, it first uses a *bootstrapping* process to create M sub-datasets from an original crowdsourced labeled dataset. For each sub-dataset, each instance is duplicated with different weights according to the distribution and class memberships of its multiple noisy labels. A base classifier is then trained from this extended sub-dataset. The class label of an unlabeled instance is predicted by aggregating the outputs of these M base classifiers.

This is the first paper that investigates ensemble learning under crowdsourcing scenarios. The contributions of the paper lie in: (1) it first proposes a novel ensemble learning method for learning from crowds without inferring true labels of training instances, which can serve as a meta-learning scheme for general purpose; (2) it proposes a novel instance duplication method for both binary and multi-class labeling; (3) it first shows that the neural network exhibits its power under this scheme; and (4) it shows that directly applying traditional ensemble learning methods to crowdsourcing probably do not work.

The remaining of the paper is organized as follows. Section 2 comprehensively reviews related work, noting the differences between the proposed method and others. Section 3 describes the proposed method in detail. Section 4 presents our experiments on both simulated and real-world datasets. Section 5 concludes the paper.

2 Related Work

After noisy labels are collected from crowdsourcing systems, classifiers can be trained using the instances with the integrated labels inferred from the crowdsourced labels. This is a straightforward two-stage meta-learning scheme, where both truth inference and model learning algorithms can be individually designed, implemented and tuned because they are mutually independent. Intuitively, a better truth inference will increase the label quality of training data, resulting in better-learned models. Thus, many studies focused on truth inference algorithms in the past decade. Some early approaches were directly developed from the classic Dawid & Skene (DS) model [14]. Raykar et al. [5] introduced a Bayesian method to model the *specificity* and *sensitivity* of each labeler during inference. Zhang et al. [15] utilized spectral methods to optimize the initial parameter settings of the DS model. Yan et al. [6] proposed a simple inference algorithm to model the expertise of labelers. Zhou et al. [16] optimized the quality of integrated labels using a minimax entropy principle. Besides these general models, some work investigated truth inference under specific scenarios. Li et al. [17] proposed an optimization framework to minimize the overall weighted deviation between truths and multi-source observations on heterogeneous data. Ma et al. [18] proposed a method, namely FaitCrowd, which jointly models topical expertise of labelers and the process of generating contents. Huang and Wang [19] modeled the relevance of unreliable sources towards the topics using graphical models. Zhao et al. [20] studied the truth inference in data streams. Although significant efforts were made to improve the performance of truth inference, both theoretical [21] and empirical [10] studies showed that it might be impossible to design a perfect inference algorithm. Thus, our method does not infer the true labels of training data.

For the model learning in the two-stage learning scheme, Sheng et al. [1], [22] first studied the learning model construction methods whereas the integrated labels are inferred by majority voting. Raykar et al. [5] built a logistic regression model after the true labels are inferred. Yan et al. [23], [24] extended the model learning to an active learning paradigm. Another similar method was proposed by Dekel et al. [25], where each labeler is modeled as an expert on a different subtopic, and learning models are trained in an active learning paradigm. Bi et al. [9] proposed a more complicated unified method, where the expertise, dedication of labelers, and the difficulties of instances are inferred during the learning model training. Compared with these two-stage meta-learning methods that only focus on binary classification, our method not only gets rid of the inference on the training data but also is suitable for multi-class classification.

Several methods directly build learning models from the crowdsourced labeled data without ground truth inference. Kajino et al. proposed two methods Personal Classifier [26] and Clustered Personal Classifier [27] to learn logistic regression models using convex optimization. In their work, each labeler is treated as an independent classifier, and these multiple classifiers can be modeled by a multi-task learning paradigm [28] with an objective function that can be globally optimized. Their methods avoid the EM-based inference that may easily trap into a local optimum. However, neither methods follow the meta-learning scheme because different classification algorithms other than logistic regression cannot guarantee the convexity of the objective function. Thus, they may not perform well when features of training data are not suitable for logistic regression. Moreover, their methods were only proposed for binary classification. Proactive learning [29] is another method that does not include inference, but it merely works under simple scenarios where two labelers present. To avoid inference, Sheng [22] proposed a pairwise training strategy for decision tree modeling, where each instance has a positive and a negative copy with different weights. Compared with the above, our proposed method is a more general meta-learning scheme, because it does not restrict learning algorithms, the number of labelers and the number of classes.

Recently, some other studies focused on learning from noisy crowds in specific domains. In particular, these studies aimed to learn from different types of crowd sensors using the noisy crowdsourcing data collected from various applications, such as online review system and social network platform. For example, Li et al. [30] proposed a budget allocation framework based on Markov decision process to set the specific requirements on labeling quality and maximize the number of labeled instances, which achieves higher labeling quality under a tight budget. Wu et al. [31] proposed a framework that models the temporal patterns of users' review behavior and detects the fake reviews. Li et al. [32] proposed an optimization framework to resolve conflicts among multiple sources of heterogeneous data types using truth discovery, which can be applied to the large-scale streaming data in the domains of forecasting weather, stocks, and flights. Huang et al. [33] proposed a semi-supervised framework to accurately infer the home locations of people from noisy and sparse crowdsourcing data they contribute. Compared with these studies, our proposed framework aims to build ensemble learning models from crowdsourcing data in a general way, which has potential to be applied in many application domains.

Learning from crowdsourced labeled data is implicitly related to *learning with noisy labels*, since crowdsourced labels are imperfect. Existing studies on this topic can be classified into three categories. *Label noise cleansing* methods [34], [35], [36] aim to identify mislabeled instances in the training set, then filter them out or correct them. These methods cannot be applied to the crowdsourcing dataset, because they suffer from two defects: removing too many correct labeled instances and a poor accuracy of predicting true values of noisy labels [11]. The second type of studies focuses on designing *label noise tolerant models*. Li and Long [37] mimicked soft-margin versions of support vector machines within a perceptron algorithm that allows it to tolerate noisy data. Sukhbaatar et al. [38] introduced an extra noise layer into a convolutional neural network which adapts its outputs to match the distribution of noisy labels. Natarajan et al. [39] proposed two approaches to suitably modify any given surrogate loss function to minimize empirical risk in presence of a small portion of noisy labels. However, as a recent survey [13] has concluded, many empirical studies reveal the facts that the performance of classifiers trained with label noise tolerant algorithms is still affected by noises. The third type of methods resorts to *ensemble learning*. Both *bagging* and *boosting* methods have been adopted to train learning models with the presence of label noises [40], [41], [42]. Simpson et al. [43] proposed a dynamic Bayesian combination of multiple imperfect classifiers and applied them to crowdsourced labeled data. Our proposed method is similar to *bagging*, since empirical studies [44], [45] have shown that *bagging* achieves better results than *boosting* when labels are imperfect. Furthermore, our method also incorporates with the crowdsourced multiple labels.

3 The Proposed Method

In this section, we present our novel ensemble learning method for crowdsourced labeled data in detail.

3.1 Problem Definition

A crowdsourced labeled dataset D consists of I instances, which is denoted by $D = \{ < \mathbf{x}_i, y_i, \mathbf{l}_i > \}_{i=1}^{I}$. $\mathbf{x}_i$ and y_i are the feature portion and the (unknown) true label of instance i respectively. $\mathbf{l}_i$ is the multiple noisy label set of instance i, each element of which l_{ij} is a noisy label provided by labeler j. Both noisy and true labels belongs to a class set $C = \{c_k\}_{k=1}^{K}$. The goal is to learn a hypothesis $h(\mathbf{x})$ that can minimize the generalization error

$$\varepsilon(h(\mathbf{x})) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}(h(\mathbf{x}) \neq y). \tag{1}$$

Since in our problem settings each instance associates with its multiple noisy labels, we duplicate each instance into several copies and let them associate with different kinds of labels. Our method does not estimate the true labels of instances. Instead, we find a hypothesis $\hat{h}$ to minimize the empirical risk in a training set, which can be calculated from the duplicated instances and their noisy labels as follows:

$$\hat{\varepsilon}(\hat{h}(\mathbf{x}')) = \frac{1}{m}\sum_{i=1}^{m} V(\hat{h}(\mathbf{x}'), l'), \tag{2}$$

where m is the number of instances in the training set and V is a loss function (usually we use 0-1 loss in classification).

3.2 Ensemble Learning Framework

Since our proposed method is a meta-learning scheme, we first present its skeleton as follows.

Algorithm 1. Ensemble Learning from Crowds (ELC)

Input: The crowdsourced labeled dataset D
the number of base classifiers M
Output: An ensemble classifier $H(\mathbf{x})$
1: load the original dataset D
2: **for** $i = 1$ to M **do**
3: create D'_i from D using *bootstrapping*
4: extend D'_i into a training set D_i^L which can be fed into a classification algorithm
5: training a base classifier $h_i(\mathbf{x})$ from D_i^L
6: **end for**
7: aggregate M base classifiers into a strong classifier, i.e., $H(\mathbf{x}) = \mathcal{F}(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_M(\mathbf{x}))$
8: **return** $H(\mathbf{x})$

Algo. ELC straightforward shows the main steps of the proposed method. First, we use *bootstrapping* procedures to create M sampled datasets $\{D'_1, D'_2, \ldots, D'_M\}$. The *bootstrapping* procedure is also known as sampling with replacement, which results in multiple identical instances in a sampled dataset. These sampled datasets cannot be used to train models, because each instance only has a multiple noisy label set which cannot be fed into classification algorithms. Then, we need change each of these datasets into a form that can be adopted by classification algorithms, which forms datasets $\{D_1^L, D_2^L, \ldots, D_M^L\}$. The details of this transformation will be discussed in the next section. We train M base classifiers $\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$ from $\{D_1^L, D_2^L, \ldots, D_M^L\}$ respectively. Finally, when making prediction for an unlabeled instance, its class label is obtained by aggregating M outputs of these base classifiers using function $\mathcal{F}$.

3.3 Refinement

In the above section, the details of two issues are untouched: (1) how do we change D_i' into D_i^L; and (2) how do we aggregate the outputs of M base classifiers.

3.3.1 Duplication of Instances

In the *bootstrapping* procedure, we only do not provide any treatment to the multiple noisy set of each instance. Since none of existing classification algorithms can accept instances with multiple noisy labels for model training, we must change their forms. If ground truth inference is involved, each instance will be assigned an integrated label which will be used in the training procedure. However, our method does not include an inference. For an instance $< \mathbf{x}_i, y_i, \mathbf{l}_i >$ (note that y_i is hidden and unknown), our method creates K copies of it, each of which has the form

$$< \mathbf{x}_i^{(k)}, \hat{y}_i^{(k)} = c_k, w_i^{(k)} >, k = 1, 2, \ldots, K.$$

That is, we create a copy of instance i for each class c_k (i.e., the estimation of its true label $\hat{y}_i$ is c_k) with a weight $w_i^{(k)}$. Suppose that J labelers provide the noisy labels for instance i, i.e., $\mathbf{l}_i = \{l_{i1}, l_{i2}, \ldots, l_{iJ}\}$. The weight $w_i^{(k)}$ for a copy of instance $\mathbf{x}_i^{(k)}$ can be calculated as follows:

$$w_i^{(k)} = \frac{1 + \sum_{j=1}^{J} \mathbb{I}(l_{ij} = c_k)}{J + K}, \tag{3}$$

where $\mathbb{I}(\cdot)$ is an indicator function, which returns 1 if the test condition is satisfied. Here, a Laplace smoothing is used to avoid zero weight. Weight reflects the importance of an instance in learning, which can be directly fed into a learning algorithm. Many mainstream classification algorithms, such as decision tree, support vector machine, neural networks, etc., adopt the weights assigned to training instances. That is why our method avoids an inference procedure and preserves information to the maximum extent.

3.3.2 Aggregation of Base Classifiers

When predicting the true label of an unlabeled instance, its final estimated class should be obtained through aggregating the outputs of M base classifiers. The simplest aggregation function is majority (or plurality) voting as follows:

$$H(\mathbf{x}) = \arg\max_{1 \le k \le K} \sum_{m=1}^{M} \mathbb{I}(h_m(\mathbf{x}) = c_k). \tag{4}$$

However, this simple voting scheme probably has a certain risk in crowdsourcing scenarios. Because each dataset D_i^L used in model training is imperfect, we cannot judge the goodness of each base classifier by cross validation as we usually do in traditional supervised learning. That is, the performance of each base classifier is still uncertain. Therefore, neither plurality voting nor weighted plurality voting is reliable. When we use a voting scheme, we deem that all unlabeled instances to be predicted are independent from one another as well as all base classifiers. Thus, a series of voting results does not reflect the class distribution of unlabeled instances. These results neither reach a global optimum or a local optimum.

In order to improve the performance of prediction, we adopt a grouped prediction scheme which is based on a maximum likelihood estimation (MLE). Simply speaking, when we predict the tth unlabeled instance, we simultaneously re-predict a group of historical unlabeled instances, e.g., from the first to the $(t-1)$th.

Suppose that M base classifiers predict T unlabeled instances. The outputs of all classifiers with respect to all unlabeled instances form a matrix A. Then, the full likelihood is,

$$L = \Pr(A|\Pi, P) = \prod_{t=1}^{T} \left(\sum_{k=1}^{K} p_k \prod_{m=1}^{M} \prod_{l=1}^{K} \left(\pi_{kl}^{(m)} \right)^{\lambda_{tl}^{(m)}} \right), \tag{5}$$

where $\Pi = \{\pi_{kl}^{(m)}\}_{m=1}^{M}$ is the set of confusion matrices for all-base classifiers ($\pi_{kl}^{(m)}$ is the probability that classifier m predict class c_k to class c_l), $P = \{p_k\}_{k=1}^{K}$ is the set of all class prior probabilities, and $\lambda_{tl}^{(m)} \in \{0, 1\}$ is whether classifier m predicts instance $\mathbf{x}_t$ as class c_l. Although the maximization of L is a complicated task, all parameters can be easily estimated with the help of EM algorithm. In E-step, probabilities of each unlabeled instance t belonging to class k are estimated as follows.

$$\Pr(\hat{y}_t = c_k|\Pi, P) \propto p_k \prod_{m=1}^{M} \prod_{l=1}^{K} \left(\pi_{kl}^{(m)} \right)^{\lambda_{tl}^{(m)}}. \tag{6}$$

In M-step, all confusion matrices of base classifiers and the priori probabilities of all classes are updated as follows.

$$\hat{\pi}_{kl}^{(m)} = \sum_{t=1}^{T} \mathbb{I}(\hat{y}_t = c_k) \lambda_{tl}^{(m)} / \sum_{l=1}^{K} \sum_{t=1}^{T} \mathbb{I}(\hat{y}_t = c_k) \lambda_{tl}^{(m)} \tag{7}$$

$$\hat{p}_k = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(\hat{y}_t = c_k). \tag{8}$$

Through the MLE-based grouped prediction method, a local optimum solution can be achieved.

3.4 Theoretical Analysis

In this section, we demonstrate some special features of our MLE weak classifier aggregation from theoretical perspective, comparing with the widely-used majority voting aggregation method.

We take the simple one-coin model for consideration, that is, the prediction of a weak classifier is either correct or wrong, the class label is either 0 (negative) or 1 (positive), and the predictive correctness probabilities over the negative and positive classes are the same. Our solution includes M weak classifiers under a basic assumption that the reliability of a weak classifier is better than the random guess. We summarize our analytical results as two theorem.

Theorem 1 (Error bound for EM). *Suppose M weak classifiers trained from crowdsourced data predict T unlabeled instances. Let $\mathbf{A} = (A_{mt})^{M \times T} (A_{mt} \in \{0, 1\})$ be the predicted labels of the T instances provided by the M weak classifiers, and $r_m \in [0.5, 1]$ be the reliability of weak classifier m. Under an one-coin model, the upper boundary of the error rate in each step during the EM iteration procedure is*

$$\epsilon \leq \frac{1}{T}\sum_{t\in[t]} exp\left(-\sum_{m\in[M]}(2r_m-1)log\frac{r_m}{1-r_m}\right). \quad (9)$$

Proof. In each iteration of the EM algorithm, our aggregation model maximizes the log-likelihood function

$$L(\mathbf{A};\mathbf{r}) = \sum_t \log\Big[\prod_m r_m^{A_{mt}}(1-r_m)^{(1-A_{mt})} + \prod_m (1-r_m)^{A_{mt}} r_m^{(1-A_{mt})}\Big]. \quad (10)$$

Using Jensen's inequality, we have

$$\begin{aligned}\mathcal{L}(\mathbf{A};\mathbf{r}) &= \sum_t \log\left[\frac{\hat{y}_t}{\hat{y}_t}\prod_m r_m^{A_{mt}}(1-r_m)^{(1-A_{mt})} + \frac{1-\hat{y}_t}{1-\hat{y}_t}\prod_m (1-r_m)^{A_{mt}} r_m^{(1-A_{mt})}\right] \\ &\geq \sum_{m,t}\hat{y}_t[A_{mt}\log r_m + (1-A_{mt})\log(1-r_m)] \\ &+ \sum_{m,t}(1-\hat{y}_t)[A_{mt}\log(1-r_m) + (1-A_{mt})\log r_m] \\ &- \sum_t[\hat{y}_t\log\hat{y}_t + (1-\hat{y}_t)\log(1-\hat{y}_t)] \equiv \mathcal{F}(\mathbf{r},\hat{\mathbf{y}}).\end{aligned} \quad (11)$$

Since A_{mt} can be written as

$$A_{mt} = y_t\mathbb{I}(A_{mt}=y_t) + (1-y_t)(1-\mathbb{I}(A_{mt}=y_t)), \quad (12)$$

we can estimate y_t by maximizing $\mathcal{F}(\mathbf{r},\hat{\mathbf{y}})$. Plugging (12) into $\mathcal{F}(\mathbf{r},\hat{\mathbf{y}})$, we have

$$\begin{aligned}\mathcal{F}(\mathbf{r},\hat{\mathbf{y}}) &= \sum_m\sum_t \mathbb{I}(A_{mt}=y_t)[((1-\hat{y}_t)(1-y_t)+\hat{y}_t y_t)\log r_m \\ &+ ((1-\hat{y}_t)y_t + \hat{y}_t(1-y_t))\log(1-r_m)] \\ &+ \sum_m\sum_t(1-\mathbb{I}(A_{mt}=y_t))[((1-\hat{y}_t)y_t+\hat{y}_t(1-y_t))\log r_m] \\ &+ ((1-\hat{y}_t)(1-y_t)+\hat{y}_t y_t)\log(1-r_m)] \\ &+ \sum_t(\hat{y}_t\log\frac{1}{\hat{y}_t} + (1-\hat{y}_t)\log\frac{1}{1-\hat{y}_t}).\end{aligned} \quad (13)$$

When maximizing $\mathcal{F}(\mathbf{r},\hat{\mathbf{y}})$, we have

$$\begin{cases}\frac{\partial\mathcal{F}(\mathbf{r},\hat{\mathbf{y}})}{\partial\mathbf{r}} = 0 \\ \frac{\partial\mathcal{F}(\mathbf{r},\hat{\mathbf{y}})}{\partial\hat{\mathbf{y}}} = 0.\end{cases} \quad (14)$$

Because y_t in (13) only has two value 0 and 1, we plug $y_t=0$ and $y_t=1$ into (13) respectively, and then solve (14). Then, we can obtain a uniform relation between estimated label $\hat{y}_t$ and its true value $\hat{y}_t$ as follows:

$$|\hat{y}_t - y_t| = \frac{1}{1+exp\Big(\sum_{m\in[M]}(2\mathbb{I}(A_{mt}=y_t)-1)log\frac{r_m}{1-r_m}\Big)}. \quad (15)$$

Finally, we can obtain the average error rate of all predicted unlabeled instances as follows:

$$\begin{aligned}\epsilon &= \frac{1}{T}\sum_{t\in[T]}|\hat{y}_t - y_t| \\ &= \frac{1}{T}\sum_{t\in[T]}\frac{1}{1+exp\Big(\sum_{m\in[M]}(2\mathbb{I}(A_{mt}=y_t)-1)log\frac{r_m}{1-r_m}\Big)} \\ &\leq \frac{1}{T}\sum_{t\in[t]} exp\left(-\sum_{m\in[M]}(2r_m-1)log\frac{r_m}{1-r_m}\right).\end{aligned}$$

□

Note that the error rates of both EM-MLE and EM-MAP aggregation of multiple uncertain predictors has been comprehensively studied in [46]. In their work, a hyperplane estimation rule, a rectified linear function of the observation matrix **A** in a high dimensional space, was used to derive the error rate bounds in more general forms. The error rate bounds of both EM-MLE and EM-MAP aggregation have the form $O(e^{-f(r_m^2)})$ in general. They also showed that if a predictor m has the same correctness probability r_m over all classes and $r_m \in [0.5, 1.0]$, tighter bounds with the complexity $O(e^{-f(r_m \log r_m)})$ could be obtained. Similarly, Eq. (9) also provides a tighter bound under the one-coin model and $r_m \in [0.5, 1.0]$ assumptions.

Theorem 2 (Error rate for aggregation). *Given an instance is predicted by M weak classifiers with the reliabilities $r_1, r_2, \ldots, r_M$, the error rate for majority voting aggregation of M weak classifiers is*

$$\epsilon = 1 - \sum_{k=\lfloor M/2\rfloor+1}^{M} PB(k;M,\mathbf{r}). \quad (16)$$

Proof. For M weak classifiers with the reliabilities $r_1, r_2, \ldots, r_M$, the occurrence of k positive labels obeys a *Poisson Binomial* distribution, whose probability mass function (*p.m.f.*), denoted by $BP(k;M,\mathbf{r})$, is defined as follows:

$$\begin{aligned}\Pr(K=k) &= PB(k;M,\mathbf{r}) \\ &= \sum_{O\subset\{1,\ldots,M\}|O|=k}\left(\prod_{i\in A} r_i\right)\left(\prod_{j\in O^c}(1-r_j)\right),\end{aligned} \quad (17)$$

where O is a subset of $\{1,\ldots,M\}$ and O^c is the complement of O ($O^c = \{1,\ldots,M\}\backslash O$). If more than a half of M weak classifiers predict the instance as positive (i.e., true label), the aggregated label will be positive. Under this *Poisson Binomial* model, the error rate of the majority voting aggregation can be estimated as Eq. (16). □

When $r_1 = r_2 = \ldots = r_M$, the *Poisson Binomial* distribution will be specialized as the *Binomial* distribution, where the error rate will reach its minimum value. In the worst case, the upper boundary of error rate will be $1 - min\{r_1, \ldots r_m\}$, which meaning that aggregation cannot increase the performance.

Comparing Eq. (9) with Eq. (16), it is easy to find two features of our MLE aggregation, which make it potentially superior to the majority voting aggregation. On one hand, increasing the number of weak classifiers will decrease the upper boundary of the error rate. On the other hand, increasing the number of unlabeled instances to be predicted will also decrease the upper boundary of the error rate.

4 Experiments

In this section, we compare our proposed algorithms with several state-of-the-art algorithms on both synthetic and real-world datasets.

4.1 Algorithms for Comparisons

In our experiments, seven algorithms are used for comparisons. They are:

- *MV* proposed in [1] is a baseline two-stage learning scheme. Its inference algorithm is majority voting.
- *ZenCrowd* proposed in [2] is a two-stage learning scheme, where reliabilities of labelers are modeled. In our experiments, we use the code of *ZenCrowd* implemented by Sheshadri and Lease[1] with default parameter settings, where the prior distribution on worker reliability has a mean of 0.7 and a variance of 0.3.
- *SDS* proposed in [15] only includes a spectral and EM based inference algorithm, where each labeler is modeled by a confusion matrix. We extended it as a two-stage learning scheme by adding a model learning procedure after inference. *SDS* only has one parameter δ used as a threshold for the elements in confusion matrix. In our experiments, we use the suggested value $\delta = 10^{-6}$ in [15].
- *Duplication* is very similar to algorithm *PairedFreq* proposed in [22] which only handles binary classification. Compared with our proposed method, it also duplicates instances with weights but only uses a single classifier to make prediction.
- *MVBagging* directly applies a traditional *bagging* algorithm to the dataset in which integrated labels of instances are inferred by majority voting. We directly use the implementation of *bagging* in WEKA 3.8.1.[2] All the parameters of *bagging* are set to their default values in WEKA, except the number of iteration which is set to the same value as M in our proposed method.
- *EnsembleMV* is a variant of our proposed method. It use a simple aggregation function plurality voting.
- *EnsembleMLE* is a more complicated variant of our proposed method. When predicting an unlabeled instance, a set of historical predicted instances are re-calculated again using the MLE method.

All of these algorithms are meta-learning schemes and can deal with both binary and multi-class labeling. Moreover, we use BP neural network implemented in WEKA to build base classifiers. We adopt the default parameter settings of the BP neural network in all experiments, in which the learning rate is set to 0.3, momentum is set to 0.2, and the number of hidden layers is set to $(\text{the number of features} + \text{the number of classes})/2$. Our ensemble learning algorithms (*EnsembleMV* and *EnsembleMLE*) include one common parameter (number of weak classifiers), which is set to 9 in most simulations and real-world evaluations. For *EnsembleMLE*, the initial setting of each confusion matrix $\boldsymbol{\pi}^{(m)}$ is that the diagonal elements are set to 0.8 and the off-diagonal elements are set to $0.2/(k-1)$ (k is the number of classes).

1. http://ir.ischool.utexas.edu/square/
2. http://www.cs.waikato.ac.nz/ml/weka/

TABLE 1
Nine UCI Datasets for Simulation

dataset	#cls	#inst	#ftr	ftr type
kr-vs-kp	2	3,196	37	nominal
spambase	2	4,601	58	numeric
sick	2	3,772	30	nominal+numeric
waveform	3	5,000	40	numeric
cmc	3	1,473	9	nominal+numeric
car	4	1728	6	nominal
vehicle	4	846	18	numeric
page-blocks	5	5,473	10	numeric
satimage	6	4,435	36	numeric

In order to provide a comprehensive comparison, we use Area Under ROC Curve (AUC) to evaluate the performance of algorithms. Since the AUC metric is original proposed for binary classification, Hand and Till [47] simply extended this measure into multi-class classification by calculating the AUC of every pair of classes (c_i and c_j) as follows.

$$\text{M-AUC} = \frac{2}{K(K-1)} \sum_{i<j} \text{AUC}(c_i, c_j). \tag{18}$$

In the absence of ambiguity, we still use the term AUC to represent M-AUC in multi-class classification for simplicity.

4.2 Simulations on UCI Datasets

In this section, nine datasets from the UCI machine learning repository[3] are used to simulate the crowdsourced labeling. These datasets are chosen, because they have different class distributions, different numbers of instances, different numbers of and different types of features, which can verify the performance of the algorithms and their applicability to the greatest extent. Table 1 lists some basic information of these datasets. Note that we do not preprocess the features of the instances in our simulations, and just use them as they were.

4.2.1 Simulation Setup

Since the cost of obtaining labels from crowdsourcing systems is still significant, in practice we always expect to save the cost as much as possible (i.e., each instance is labeled as few as possible). In order to evaluate the performance of the algorithms under the different numbers of noisy labels assigned on each instances, our comparison process is presented in *Proc. SUD* for clarity.

After we load a UCI dataset, we randomly hold out 30 percent of its instances with respect to each class for testing and the remaining 70 percent of instances are used to train models. Then we begin to simulate crowdsourcing workers. The number of workers increase from one to ten. Each worker labels all instances in the training set. In order to obtain experimental results under different numbers of workers, after a new worker has labeled all instances, we evaluate the performance of seven algorithms and record their results, as lines 5-7 in *Proc. SUD* show. Since the dataset is randomly partitioned, we repeat the above experimental procedure ten times (line 2). Finally, the average AUCs and their standard deviations of all compared algorithms

3. https://archive.ics.uci.edu/ml/index.html

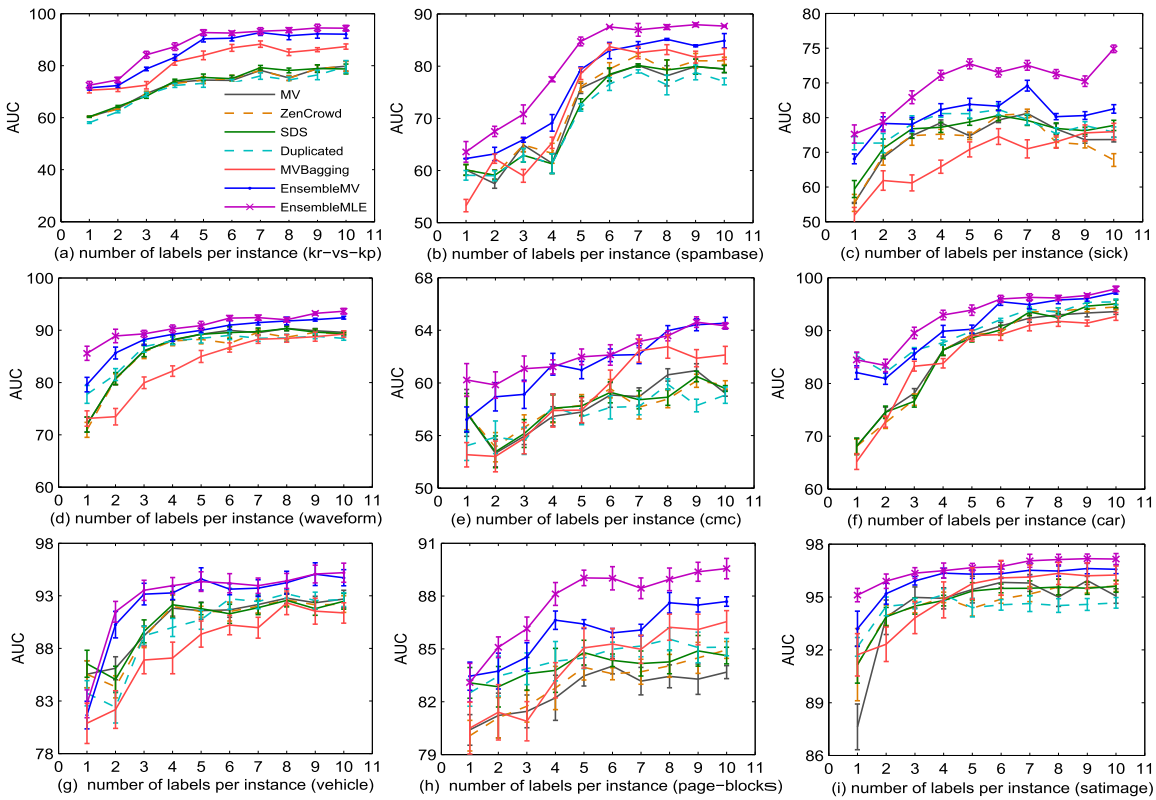


Fig. 1. The comparison results of six algorithms on the nine synthetic crowdsourced labeled datasets.

under different numbers of workers on each instance are reported (line 10).

Procedure 1. Simulations on a UCI Dataset (SUD)

1: load a UCI dataset $\mathscr{D}$
2: **for** $r = 1$ to 10 **do**
3: randomly partition $\mathscr{D}$ into a training set $\mathscr{D}^L(70\%)$ and a test set $\mathscr{D}^T(30\%)$ with respect to each class
4: simulate 10 labelers $\{u_1, u_2, \ldots, u_{10}\}$
5: **for** $j = 1$ to 10 **do**
6: let u_j label all instances in $\mathscr{D}^L$
7: run seven algorithms on $\mathscr{D}^L$ and $\mathscr{D}^T$ and record their results respectively
8: **end for**
9: **end for**
10: report average AUC values and their standard deviations of all compared algorithms

The simulation of the labeling behaviors of crowdsourcing workers is as follows. The distribution of the labeling quality (in accuracy) of these labelers obeys a truncated normal distribution with the form $\mathcal{N}(\mu, \sigma^2, a, b)$, where $\mu = 0.6$, $\sigma = 0.1$, $a = 0.5$, and $b = 0.7$. That is, each quality of labeling falls into a range $[0.5, 0.7]$. Furthermore, we assume that the labeling errors uniformly distribute across all classes. That is, if the quality of a labeler is p percent, each class of the dataset will equally obtain p percent error labels. Although in reality this is a very idealized situation, when we do not have more knowledge about the labelers, this is a widely accepted scheme.

4.2.2 Overall Comparison Results on Simulations

Fig. 1 shows the comparison results of seven learning algorithms on the nine synthetic crowdsourced labeled datasets. The x-axis represents the number of noisy labels on each instance and the y-axis represents the performance of learned models in term of AUC. From this figure, we have the following observations.

We first look at the results of three traditional two-stage learning algorithms *MV*, *ZenCrowd* and *SDS*. The performance of these three algorithms does not show any obvious difference in most cases. Only on *page-blocks*, *ZenCrowd* and *SDS* are slightly better than the base line *MV* when the number of noisy labels on each instance is greater than four. Although the inference algorithms *ZenCrowd* and *SDS* are thought to be more accurate than *MV*, their superiority on inference does not reflect to the performance of model learning, because agnostic inference algorithms completely ignore the features of instances which play a decisive role in model learning. The results of three two-stage learning algorithms confirm our argument that it is very hard to improve the quality of a learned model by designing more accurate inference algorithms. Then, we turn our attention to the results of *Duplicated*, a learning algorithm without inference. We found only in a small proportion of cases (on *sick*, *car* and *page-blocks*) *Duplicated* is slight better than *MV*, *ZenCrowd* and *SDS*. On the other datasets, *Duplicated* does not show any advantage, and sometimes it is even worse (e.g., on *satimage*).

MVBagging stands for a class of methods that a traditional *bagging* algorithm directly follows an inference algorithm. Because the performance of all two-stage methods mentioned above does not show obvious difference, the performance of *MVBagging* can fully represent the performance of this class of methods (i.e., *inference+bagging*). *MVBagging* may outperform *MV*, *ZenCrowd*, *SDS* and *Duplicated* as the number of labels increase on *spambase*, *cmc*, *page-blocks*, and *satimage*, but in most cases it performs even worse on the

other five datasets. From these results we can draw a conclusion that directly applying traditional ensemble learning methods to crowdsourced labels probably does not work well, because noisy inferred labels in training sets prohibit obtaining multiple accurate base classifiers. Thus, the performance of aggregated base classifier is not good. Compared with the traditional *Bagging* method, our proposed algorithms *EnsembleMV* and *EnsembleMLE* perform much better on all synthetic datasets.

Compared with *MV*, *ZenCrowd*, *SDS*, *Duplicated* and *MVBagging*, the performance of one of our proposed ensemble algorithm *EnsembleMV* is much better on seven out of nine datasets (*kr-vs-kp*, *sick*, *cmc*, *car*, *vehicle*, *page-blocks* and *satiamge*). The maximum increment of AUC can reach more than 10 percent (absolute value, similarly hereinafter). On *waveform*, it is slightly better than the other five existing algorithms. Therefore, the proposed ensemble learning scheme indeed significantly improves the performance of learned models. *EnsembleMLE* is an enhanced version of the ensemble learning method, where the aggregation of the predictions of the base classifiers resorts to the maximum likelihood estimation. It is obvious that *EnsembleMLE* wins *EnsembleMV* on eight out of nine datasets (except *vehicle*). The average AUC increment of *EnsembleMLE* reaches 3-7 percent. Thus, the proposed *EnsembleMLE* overwhelmingly outperforms the five existing learning algorithms on all synthetic datasets.

Let us get into some details of our proposed algorithms. Clearly, *EnsembleMV* is not very stable, because majority voting is not an optimal solution to aggregate the predictions of base classifiers. Sometime, majority voting has a large vibration. For example, on the dataset *spambase*, in most cases *EnsembleMV* is better than *MV*, *ZenCrowd*, *SDS*, *Duplicated* and *MVBagging*. However, in the cases that there are 3, 5 and 7 labels on each instance, *EnsembleMV* does not have an obvious advantage. Compared with aggregating the predictions using majority voting, our MLE aggregation method is more stable, which shows a significant superiority on the datasets *spambase*. The average AUC increment of *EnsembleMLE* even reaches 4-8 percent. Furthermore, its learning curves are smoother than that of *EnsembleMV*. Another advantage of *EnsembleMLE* is that when the numbers of noisy labels on each instances is small $(2 \sim 5)$, its performance is much better than that of *EnsembleMV* in most cases (e.g., on *spambase*, *waveform*, *cmc*, *car*, *page-blocks*, *satimage*). That is, *EnsembleMLE* potentially reduces the costs of label acquisition. In some cases (e.g., at some points on datasets *cmc* and *vehicle*), the performance of *EnsembleMV* and *EnsembleMLE* is very close, which is a normal phenomenon. When using an EM algorithm to solve an MLE aggregation problem, the object function usually traps into a local optimum. Compared to the MV aggregation that does not include any optimization, from a statistical perspective, MLE aggregation must be better than MV method, especially, when there are a large number of instances and/or classes. However, this does not mean that MLE aggregation will be significantly better than MV method in any individual case. Examining the attributes of these two datasets (*cmc* and *vehicle*), we find the numbers of instances in them are relatively small. That may be why the statistical advantages of MLE aggregation cannot be played at some points on them.

Pay attention to the results on *sick*, where all learning curves of these algorithms have a large vibration. This is because that the class distribution of this dataset is extremely imbalanced. The negative instances occupy 93.9 percent, which deteriorates the performance of inference algorithms and base classifiers. When using a majority voting to aggregate the predictions of all base classifiers, the imbalanced distribution of classes makes instances more easily to be predicted as majority class. Thus, *EnsembleMV* in this case loses its advantage. As Fig. 1c shows, it is only slightly better than *MV*, *ZenCrowd*, *SDS*, and *Duplicated*, when the numbers of labels are greater than seven. Compared with *EnsembleMV*, the MLE aggregation shows a great advantage. Its performance is sometimes over 8 point greater than that of *EnsembleMV*. It seems that the MLE aggregation of the predictions of base classifiers performs much better on the datasets with imbalanced class distribution. Similar results can also be found on *page-blocks*, where instances belonging to class c_0 occupy 89.8 percent and those belonging to the other three classes only occupy 10.2 percent.

Overall, compared with the existing five learning algorithms, our proposed ensemble learning algorithms *EnsembleMV* and *EnsembleMLE* significantly improve the quality of learned models. The MLE aggregation of the predictions of the base classifiers significantly outperforms the majority voting aggregation method, especially under the situations that the class distribution of a dataset exhibits some imbalance and the number of noisy labels on each instance is small.

4.2.3 Tuning the Number of Weak Classifiers

As the error bound analysis in Section 3 shows, the error rate of *EnsembleMLE* decreases with more added weak classifiers. To verify this point, we investigate the changes in the performance of *EnsembleMLE* when the different numbers of base classifiers increase. In our experiment, the number of weak classifiers increases from 3 to 15, increasing by two each time. Because we are only interested in the relationship between the performance and the number of weak classifiers, we simply fix the number of crowdsourced workers to five. The experimental procedure, classification algorithm, and the other parameter settings are the same as they are in Section 4.2.2.

Table 2 shows the experimental results under different numbers of weak classifiers on the nine UCI datasets. From the table, we have an overall observation that the tendency of the performance of *EnsembleMLE* goes up on each dataset when the number of weak classifiers increases, which is consistent with our theoretical analysis. When the numbers of weak classifiers are less then seven (i.e., $M = 3$ and $M = 5$), the performance on all datasets is obviously lower than that under $M = 7$. We can observe a significant increase in performance on each dataset under $M \leq 7$. On most datasets (except *spambase* and *satimage*), the average performance under $M = 7$ is less than that under $M = 9$. Therefore, $M = 9$ is a watershed of the performance. When we continue increasing the value of M (i.e., $M > 9$), we still can find some slight increase in the performance. For example, on *spambase*, *sick*, *car*, *vehicle*, *page-blocks* and *satimage*, the average performance under $M = 11$ is sightly greater than that under $M = 9$. However, considering the standard deviations, the excesses are not statistically significant. In

TABLE 2
Testing AUC Results of *EnsembleMLE* under Different Numbers of Weak Classifiers

Dataset	M=3	M=5	M=7	M=9	M=11	M=13	M=15
kr-vs-kp	84.86 ± 1.13	88.66 ± 1.02	91.33 ± 1.03	92.73 ± 1.08	92.43 ± 1.03	92.64 ± 0.96	92.68 ± 0.78
spambase	80.65 ± 1.02	82.48 ± 0.98	85.02 ± 1.00	84.74 ± 0.85	85.23 ± 0.82	84.62 ± 0.78	85.88 ± 0.89
sick	62.63 ± 0.89	67.06 ± 0.92	72.75 ± 0.76	73.29 ± 0.86	78.38 ± 0.90	77.98 ± 1.01	78.30 ± 0.90
waveform	84.12 ± 1.80	89.26 ± 1.41	90.30 ± 0.98	90.88 ± 1.02	90.11 ± 0.87	90.54 ± 1.00	91.08 ± 0.95
cmc	52.55 ± 0.98	57.94 ± 0.88	59.83 ± 0.82	62.47 ± 0.88	61.93 ± 0.78	62.20 ± 0.82	62.60 ± 0.88
car	67.36 ± 2.38	91.18 ± 1.54	94.37 ± 1.10	95.97 ± 1.18	96.33 ± 1.01	96.03 ± 0.98	96.33 ± 0.88
vehicle	73.42 ± 2.41	89.53 ± 1.38	93.81 ± 0.98	94.35 ± 0.93	94.65 ± 0.88	94.84 ± 0.98	94.92 ± 0.94
page-blocks	78.60 ± 1.82	85.72 ± 1.30	87.81 ± 0.89	89.63 ± 0.72	90.05 ± 0.88	89.02 ± 1.00	89.86 ± 0.97
satimage	92.10 ± 1.34	94.43 ± 0.83	96.41 ± 0.90	96.37 ± 0.68	96.68 ± 0.71	96.54 ± 0.68	96.70 ± 0.72

addition, the performance under $M = 13$ and $M = 15$ is not always better than that under $M = 9$. In summary, when $M > 9$, we deem that the performance has reached its ceiling, which may fluctuate within a small range. Therefore, in the rest of our experiments, we fix the number of weak classifiers to 9 to reduce the running time with near-optimal performance.

4.2.4 Tuning the Learning Rate of Classifiers

In our simulation, we use neural networks to learn classifiers. It is known that the learning rate is an essential parameter for neural networks. Theoretically, too small a learning rate (η) will significantly slow down the speed of model training, and even cause the learning process to fail to converge, while an excessively large learning rate will cause the learner to oscillate around its optimal point. In this section, we investigate the impact of the learning rate on the performance of the proposed *EnsembleMLE* algorithm. Usually, the learning rate of a neural network is a real value in the range of $(0, 1)$. In our experiments, the learning rate increases from 0.1 to 0.6, increasing by 0.1 each time. To facilitate the performance tuning, we still simply fix the number of crowdsourced workers to five and the number of weak classifiers to nine. The experimental procedure, classification algorithm, and the other parameter settings (except the learning rate) are the same as they are in Section 4.2.2.

Table 3 lists the experimental results of the performance of *EnsembleMLE* when tuning the learning rates. Obviously, we have the consistent observations on all the nine datasets. When the learning rates are less than and equal to 0.4, the performance of *EnsembleMLE* in different learning rates is persistently at a very high level and does not show significant differences. When the learning rate is set to 0.5, on most datasets (except *cmc*, *car* and *satimage*) the performance slightly declines 1-2 points. When the learning rate is set to 0.6, we can observe significant declines on almost all datasets (except on *waveform*). In addition, when we set $\eta \geq 0.5$, on some datasets (such as *spambase*, *sick*, *vehicle* and *page-blocks*) the standard deviations obviously increases. If we only consider the average AUC metric of the classifiers, setting the learning rate to 0.1 and 0.3 is not essentially different. However, setting it to 0.3 will greatly accelerate the learning speed. Thus, in the rest of our experiments, we set it to 0.3 for all compared algorithms.

4.2.5 Comparison in Different Percentages of Training Sets

In above experiments, we randomly hold out 30 percent instances of a dataset for test and the remaining 70 percent are for training models. In this section, we investigate the performance of all compared algorithm when varying the percentage of the training set and test set. For each dataset, the proportion of the training instances increases from 20 to 80 percent, increasing by 10 percent (absolute value) each time. Here, we still simply fix the number of crowdsourced workers to five and the number of weak classifiers to nine. The experimental procedure, classification algorithm, and the other parameter settings are the same as they are in Section 4.2.2.

Fig. 2 shows the comparison results of six algorithms on the nine synthetic datasets when varying the percentage of the training set. On all the datasets, the overall tendency of the performance of all algorithms goes up when the percentage of the training set increases. However, we still notice that on some datasets (*sick*, *waveform*, and *page-block*), when the percentage of the training set is up to 80 percent, the performance of the other baseline algorithms obviously decreases. This is because when the test set is small, several

TABLE 3
Testing AUC Results of *EnsembleMLE* under Different Learning Rates of Neural Networks

Dataset	$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.3$	$\eta = 0.4$	$\eta = 0.5$	$\eta = 0.6$
kr-vs-kp	$\mathbf{93.66 \pm 0.77}$	92.23 ± 0.85	92.43 ± 0.93	91.44 ± 1.01	89.56 ± 0.98	88.31 ± 1.16
spambase	86.02 ± 0.47	86.56 ± 0.78	85.23 ± 0.82	$\mathbf{87.03 \pm 1.20}$	84.45 ± 1.45	81.82 ± 1.42
sick	77.35 ± 1.06	78.09 ± 1.02	$\mathbf{78.38 \pm 0.90}$	78.09 ± 1.01	75.95 ± 1.26	72.16 ± 1.50
waveform	$\mathbf{91.19 \pm 0.55}$	90.32 ± 0.88	90.11 ± 0.87	90.63 ± 0.98	88.49 ± 1.02	88.71 ± 0.98
cmc	61.71 ± 0.97	$\mathbf{62.03 \pm 0.84}$	61.93 ± 0.78	61.87 ± 0.92	61.03 ± 1.01	59.92 ± 0.98
car	95.77 ± 0.65	95.85 ± 0.89	$\mathbf{96.33 \pm 0.78}$	96.16 ± 0.82	95.42 ± 0.89	92.89 ± 0.93
vehicle	93.86 ± 0.56	92.93 ± 0.98	$\mathbf{94.65 \pm 0.88}$	93.40 ± 1.14	92.02 ± 1.19	90.09 ± 1.33
page-blocks	89.43 ± 0.87	89.40 ± 0.75	$\mathbf{90.05 \pm 0.88}$	88.80 ± 0.89	87.81 ± 1.01	85.40 ± 1.28
satimage	$\mathbf{96.80 \pm 0.67}$	96.71 ± 0.72	96.68 ± 0.82	96.48 ± 0.88	95.75 ± 0.78	95.26 ± 0.85

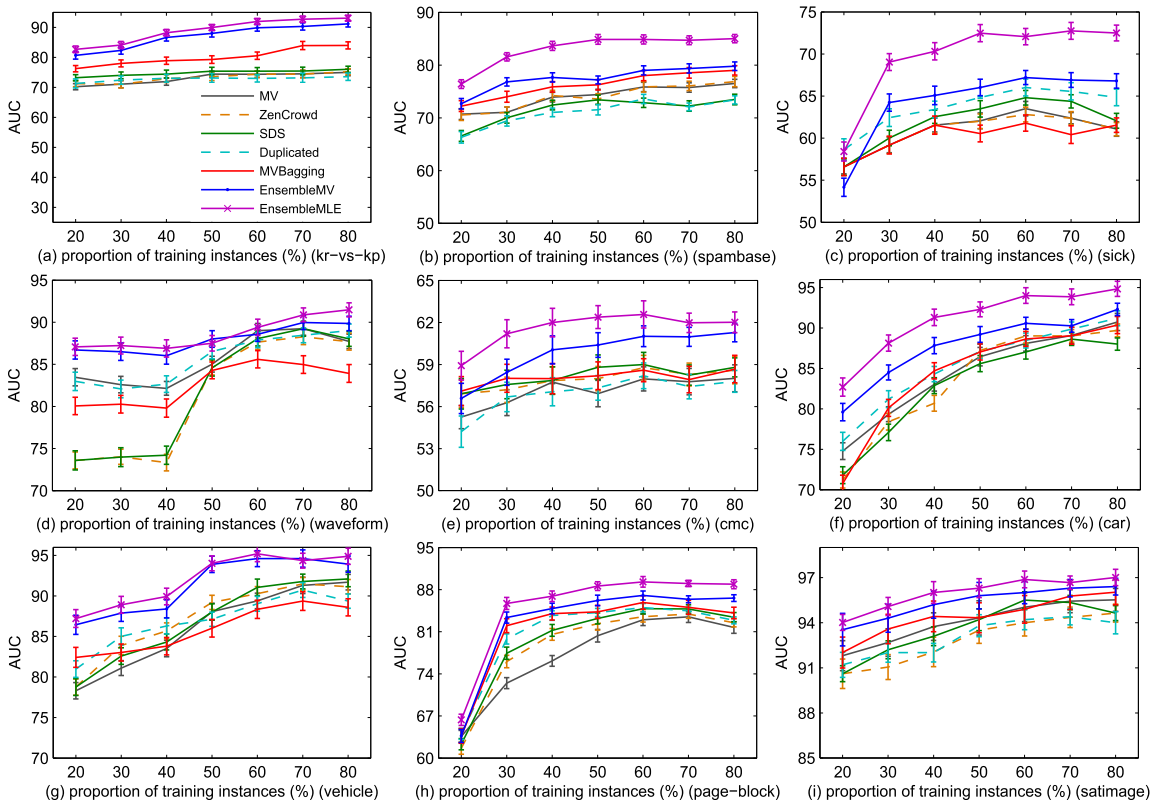


Fig. 2. The comparison results of six algorithms on the nine synthetic datasets when varying the percentage of the training set.

wrong predictions may result in a noticeable decrease in the evaluation matric. For both of our *EnsembleMV* and *EnsembleMLE* algorithms, we do not observe such deterioration of their performance. When the percentage of the training set is greater than and equal to 30 percent, our *EnsembleMV* and *EnsembleMLE* algorithms significantly outperform the others on all datasets. On six out of nine datasets (i.e., except *kr-vs-kp*, *waveform*, and *vehicle*), *EnsembleMLE* significantly outperforms *EnsembleMV*. When the percentage of the training set is only 20 percent, on five out of nine datasets (except *spambase*, *sick*, *cmc*, and *page-block*), *EnsembleMV* outperforms the baseline algorithms, and on seven out of nine datasets (except *sick* and *page-block*) *EnsembleMLE* outperforms them. Comparisons on some datasets (*waveform*, *vehicle*, and *satimage*) also show that when the percentage of the training set is less than 50 percent, the performance gaps between the proposed algorithms and baselines are larger. This is because our algorithms duplicate the instances, which partially alleviates scarcity of the training samples.

4.2.6 Comparison with Other Inference-Free Methods

Kajino et al. proposed two methods Personal Classifier [26] (*PC*) and Clustered Personal Classifier [27] (*CPC*) to learn logistic regression models for binary classification using convex optimization. We compare our algorithms with *PC* and *CPC* on three binary datasets *kr-vs-kp*, *spambase*, and *sick*. To make fair comparisons, our algorithms also train logistic regression models. We use the Python code of the *CPC* method provided by the original authors and implement the *PC* method ourselves.[4] There are several hyperparameters in *PC* and *CPC*. We search the values of each hyperparameter in a wide range with a small increment and report the best performance as the original articles [26] and [27] do. Specifically, λ is in a range of $[0.001, 0.1]$ with increment 0.05, ρ is in a range of $[1, 100]$ with increment 1, μ is in a range of $[0, 200]$ with increment 10, and η is fixed to the value 1.0 (as it is in [27]).

Fig. 3 shows the comparison results among these methods. As the number of workers increases, the performance of all methods is improved. Generally, the performance of *CPC* is only slightly better than that of *PC* in some cases. Our proposed *EnsembleMV* and *EnsembleMLE* significantly outperform *PC* and *CPC* in all cases, which confirms that an ensemble model usually outperforms a single model. Furthermore, *EnsembleMLE* is consistently better than *EnsembleMV*.

4.3 Evaluation on Two Real-World Datasets

In this section, we evaluate our proposed algorithms on two real-world datasets.

4.3.1 Two Real-World Datasets

LeavesCrowd. We create a real-world crowdsourced labeled dataset, namely *LeavesCrowd*. We extracted six different classes of instances from the one-hundred plant species leaves dataset:[5] Maple (96), Alder (48), Eucalyptus (48), Poplar (48), Oak (96), and Tilia (48). These classes are chosen because these leaf species are not very hard to be identified by non-expert workers. Each instance in the dataset is an image, in which a white shape of a leaf is presented in a

4. Source code of the CPC method is available at https://github.com/kanojikajino/clustering-crowds.

5. http://archive.ics.uci.edu/ml/datasets/One-hundred+plant+species+leaves+data+set

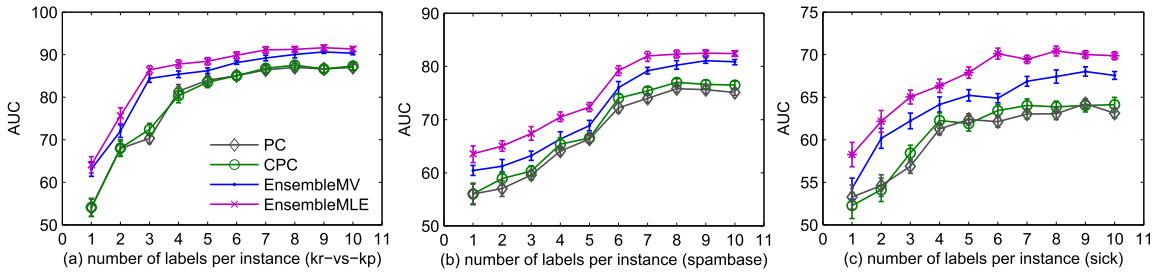


Fig. 3. Comparison results with Kajino's PC and CPC methods on three binary-classification datasets.

black background, as Fig. 4 shows. We use the 64 dimensional marginal features of the images provided in [48] to build the learning model in our experiments.

We posted selected 384 instances on Amazon Mechanical Turk (AMT)[6] and asked crowdsourcing workers to first look at some examples of these six leaf species (examples are not included in the dataset to be labeled) and then classify the instances into the most proper classes. Each instance is exactly labeled by ten different workers. Thus, the dataset totally contains 3840 labels. After noisy labels were collected, we analyzed the accuracies of the workers against ground truths. There are totally 83 workers labeling the dataset. The distribution of the labeling accuracies of these workers are shown in Fig. 5. It is obvious that the labeling accuracies of most workers are in the range $[0.4, 0.7]$.

ChihuahuaDog. This binary labeled dataset was collected by Bi et al. [9]. The crowd workers on AMT were asked to judge whether dogs in pictures are Chihuahua (a breed of dog). This dataset includes 299 instances (157 negative and 142 positive ones), each of which has 4,096 numeric features and was labeled by 6 workers. There are totally 21 workers. The distribution of the labeling accuracies of the workers are shown in Fig. 6. The labeling accuracies of most workers are in the range $[0.7, 1.0]$.

4.3.2 Experimental Setup

Setup for LeavesCrowd. Since each instance in the dataset obtained exactly 10 noisy labels, it is convenient for us to evaluate the performance of the seven algorithms under different numbers of noisy labels on each instance as we did in the simulations. We treat the multiple noisy label set of each instance as a label pool, from which we can pick up a number of labels. Similar to *Proc. SUD*, we increase the number of noisy labels on each instance from 1 to 10. Specifically, each time we randomly select n $(1 \leq n \leq 10)$ labels from the label pool of each instance, and then evaluate the performance of the seven algorithms. We still use the BP neural network as the classification algorithm and its parameter settings are the same as they are in the above simulations. Again, a random 70/30 (train/test) split across each class was performed on the dataset. We repeat the experiment 10 times and report the average AUCs and their standard deviations.

Setup for ChihuahuaDog. Different from the dataset *LeavesCrowd*, instances in this dataset have a small number of labels and the qualities of noisy labels are relatively high. Therefore, we directly evaluate the performance of the seven algorithms with all noisy labels presented. Because the number of features are quite large, we reduced the number of hidden layers of the BP neural network to 2 for faster model training. The other experimental setups are the same as those on *LeavesCrowd*.

Comparison using Different Learning Algorithms. As mentioned above, the proposed method is a meta-learning scheme. It means that we can choose proper classification algorithms according to the features of instances. To verify this point, we evaluate the proposed algorithms using other two famous classification algorithms: decision tree and SVM. A cost-sensitive decision tree [49] can accept weights of instances as the costs of misclassifications and SVM also can accept weights of instances as the penalty coefficients. The parameters of the decision tree and SVM (implemented as SMO [50]) are set to their default values in WEKA.

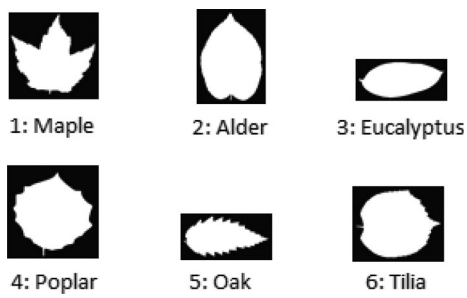


Fig. 4. Six species of leaves. Workers from AMT are asked to identify them according to their shapes.

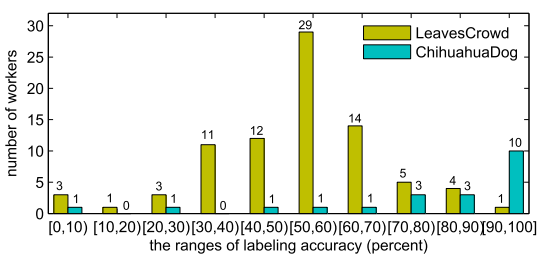


Fig. 5. The distribution of the labeling accuracies of 83 workers on AMT for leaf species tasks.

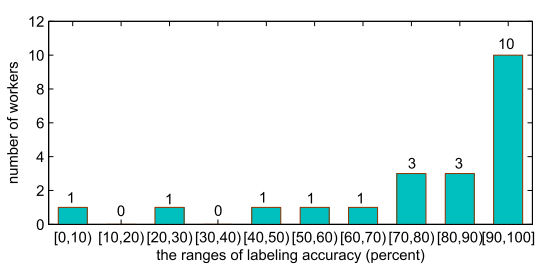


Fig. 6. The distribution of the labeling accuracies of 21 workers on AMT for dog breed judgment tasks.

6. http://www.mturk.com

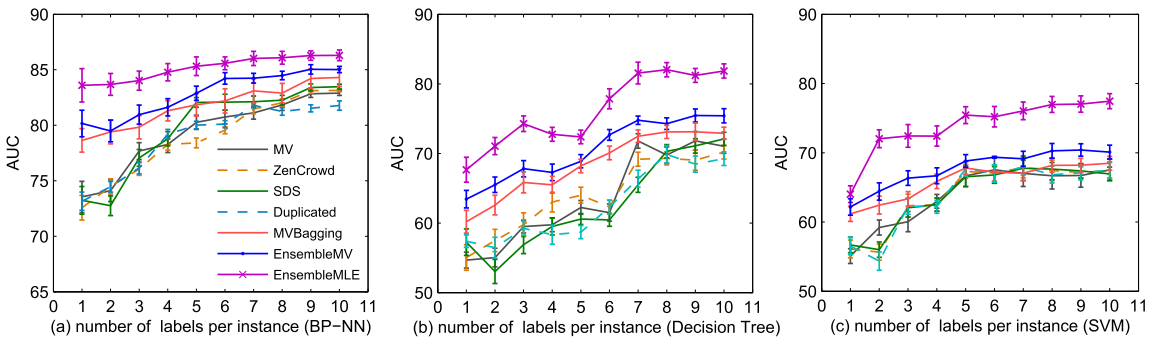


Fig. 7. Comparison results on *LeavesCrowd* under different numbers of noisy labels per instance, when the learning algorithms are (a) BP neural network, (b) decision tree, and (c) SVM.

4.3.3 Experimental Results

The comparison results of the seven algorithms on *LeavesCrowd* are shown in Fig. 7. First, we focus on Fig. 7a, where the learning algorithm is BP neural network. The learning curves in Fig. 7a basically follow the conclusions that we have drawn in the above simulations. When the number of noisy labels on each instance is less than five, the performance of three two-stage algorithms has no obvious differences. When the number of noisy labels on each instance becomes larger (≥ 5), *SDS* outperforms *MV* and *ZenCrowd*. The one stage *Duplicated* does not show any advantage on this real-world dataset. *MVBagging* outperforms the other three two-stage algorithms and *Duplicated*. Compared with the five existing algorithms, our proposed ensemble learning algorithms *EnsembleMV* and *EnsembleMLE* are much better. The average AUC increments can reach as much as 3 and 7 percent on average respectively. Furthermore, *EnsembleMLE* significantly outperforms *EnsembleMV*, especially when the number of noisy labels is less than 5. For example, the maximum AUC increase of *EnsembleMLE* reaches 6 percent, when there are 2 noisy labels on each instance.

Figs. 7b and 7c show the results when the learning algorithm is replaced with decision tree and SVM, respectively. In Fig. 7b, the performance of both proposed *EnsembleMV* and *EnsembleMLE* are significantly better than that of the five existing algorithms. In this case, the performance of *EnsembleMLE* is improved at least 4 points. In Fig. 7c, the performance of *EnsembleMV* is only slight better than that of the five existing algorithms. Surprisingly, the performance of *EnsembleMLE* still maintains at a high level. Compared with *EnsembleMV*, the AUC increase of *EnsembleMLE* exceeds 10 points in most cases. Thus, we can conclude that the proposed ensemble learning algorithms are superior to the existing ones when different learning algorithms are chosen. At last, compared with Fig. 7a, the performance of the models trained by decision tree and SVM is worse than that of the models trained by the BP neural network.

TABLE 4
Testing AUC Results on *ChihuahuaDog* using Different Classification Algorithms

algorithm	BP NN	Dec. Tree	SVM
MV	72.55 ± 1.12	67.03 ± 1.54	70.33 ± 1.30
ZenCrowd	72.81 ± 0.96	66.48 ± 1.28	68.27 ± 1.16
SDS	71.63 ± 1.05	67.01 ± 1.92	71.51 ± 1.32
Duplicated	71.75 ± 0.84	65.66 ± 1.60	70.59 ± 1.28
MVBagging	73.55 ± 1.19	65.94 ± 1.34	71.80 ± 0.92
EnsembleMV	75.00 ± 1.16	68.94 ± 1.54	$\mathbf{75.07 \pm 1.18}$
EnsembleMLE	$\mathbf{76.96 \pm 1.04}$	$\mathbf{71.63 \pm 1.32}$	74.64 ± 0.92

Table 4 lists the comparison results on *ChihuahuaDog* when BP neural network, decision tree and SVM are chosen for model training respectively. Again, we observe the consistent results. *MVBagging* is slightly better than *MV*, *ZenCrowd*, *SDS* and *Duplicated* only when the learning algorithm is neural network. No matter which learning algorithm is used for model training, our proposed algorithms *EnsembleMV* and *EnsembleMLE* are significantly superior to the other five algorithms. *EnsembleMLE* outperforms *EnsembleMV* when the learning algorithm is either neural network or decision tree. However, the former is slightly inferior to the later when SVM is used for model training. Nevertheless, the BP neural network outperforms the other two learning algorithms. On *ChihuahuaDog*, we also observe that the difference of the performance between *EnsembleMV* and *EnsembleMLE* is small. The reasons are the same as we have mentioned in simulation: *ChihuahuaDog* includes a small number of instances (299) and two classes, where MLE aggregation may not significantly exceed MV method. In contrast, on the multi-class dataset *LeavesCrowd*, although its number of instances is not large (348), the gap of the performance between *EnsembleMLE* and *EnsembleMV* is quite large because the dataset includes six classes. Both *LeavesCrowd* and *ChihuahuaDog* demonstrate that, when different learning algorithms are used as the base learner, neural network performs the best in our proposed ensemble learning scheme.

5 Conclusion

Repeated labeling has been widely utilized in label acquisition from crowdsourcing. When training models from repeated labeled instances, the traditional two-stage learning scheme consists two independent inference and model training phases. Due to the weaknesses of low accuracy and information lost during the inference, the quality of learned models cannot be easily improved. In this paper, we proposed a novel meta-learning ensemble solution for learning from crowds. The proposed solution first uses a *bootstrapping* process to create multiple sub-datasets from an original crowdsourced labeled dataset, in each of which instances are duplicated with different weights according to the distribution and class memberships of their repeated labels. A base classifier is then trained from each extended sub-dataset.

Finally, unlabeled instances are predicted by aggregating the outputs of multiple base classifiers. We proposed two different versions of the method *EnsembleMV* and *EnsembleMLE*, which aggregate the predictions of the base classifiers based on majority voting and maximum likelihood estimation respectively. Experimental results on nine simulated and two real-world crowdsourcing datasets consistently show that both of our proposed ensemble learning algorithms *EnsembleMV* and *EnsembleMLE* significantly outperform five state-of-the-art algorithms. Compared with *EnsembleMV*, *EnsembleMLE* performs much better.

Acknowledgments

The authors thank the anonymous reviewers for their constructive and insightful comments that help improve the quality of the paper. This research has been supported by the National Natural Science Foundation of China under grants 91846104 and 61603186, the Natural Science Foundation of Jiangsu Province, China, under grant BK20160843, the China Postdoctoral Science Foundation under grants 2016M590457 and 2017T100370, and the Postdoctoral Science Foundation of Jiangsu Province, China, under grant 1601199C.

References

[1] V. S. Sheng, F. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 614–622.

[2] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 469–478.

[3] B. Zhong, H. Yao, S. Chen, R. Ji, T.-J. Chin, and H. Wang, "Visual tracking via weakly supervised learning from multiple imperfect oracles," *Pattern Recognit.*, vol. 47, no. 3, pp. 1395–1410, 2014.

[4] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: A survey," *Artif. Intell. Rev.*, vol. 46, no. 4, pp. 543–576, 2016.

[5] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, 2010.

[6] Y. Yan, R. Rosales, G. Fung, M. W. Schmidt, G. H. Valadez, L. Bogoni, L. Moy, and J. G. Dy, "Modeling annotator expertise: Learning when everybody knows a bit of something," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 932–939.

[7] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, "The multidimensional wisdom of crowds," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 2424–2432.

[8] J. Whitehill, T.-F. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 2035–2043.

[9] W. Bi, L. Wang, J. T. Kwok, and Z. Tu, "Learning to predict from crowdsourced data," in *Proc. Conf. Uncertainty Artif. Intell.*, 2014, pp. 82–91.

[10] A. Sheshadri and M. Lease, "Square: A benchmark for research on computing crowd consensus," in *Proc. 1st AAAI Conf. Human Comput. Crowdsourcing*, 2013, pp. 156–164.

[11] J. Zhang, V. S. Sheng, J. Wu, X. Fu, and X. Wu, "Improving label quality in crowdsourcing using noise correction," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1931–1934.

[12] S. G. Pierce, Y. Ben-Haim, K. Worden, and G. Manson, "Evaluation of neural network robust reliability using information-gap theory," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1349–1361, Nov. 2006.

[13] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.

[14] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Appl. Statist.*, vol. 28, pp. 20–28, 1979.

[15] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan, "Spectral methods meet EM: A provably optimal algorithm for crowdsourcing," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1260–1268.

[16] D. Zhou, S. Basu, Y. Mao, and J. C. Platt, "Learning from the wisdom of crowds by minimax entropy," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2195–2203.

[17] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 1187–1198.

[18] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han, "FaitCrowd: Fine grained truth discovery for crowdsourced data aggregation," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 745–754.

[19] C. Huang and D. Wang, "Topic-aware social sensing with arbitrary source dependency graphs," in *Proc. 15th Int. Conf. Inf. Process. Sensor Netw.*, 2016, Art. no. 7.

[20] Z. Zhao, J. Cheng, and W. Ng, "Truth discovery in data streams: A single-pass probabilistic approach," in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 1589–1598.

[21] N. B. Shah and D. Zhou, "On the impossibility of convex inference in human computation," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1291–1297.

[22] V. S. Sheng, "Simple multiple noisy label utilization strategies," in *Proc. IEEE 11th Int. Conf. Data Mining*, 2011, pp. 635–644.

[23] Y. Yan, G. M. Fung, R. Rosales, and J. G. Dy, "Active learning from crowds," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 1161–1168.

[24] Y. Yan, R. Rosales, G. Fung, F. Farooq, B. Rao, J. G. Dy, and P. Malvern, "Active learning from multiple knowledge sources," in *Proc. 15th Int. Conf. Artif. Intell. Statist.*, 2012, Art. no. 6.

[25] O. Dekel, C. Gentile, and K. Sridharan,"Selective sampling and active learning from single and multiple teachers," *J. Mach. Learn. Res.*, vol. 13, pp. 2655–2697, 2012.

[26] H. Kajino, Y. Tsuboi, and H. Kashima, "Convex formulations of learning from crowds," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 73–79.

[27] H. Kajino, Y. Tsuboi, and H. Kashima, "Clustering crowds," in *Proc. 27th AAAI Conf. Artif. Intell.*, 2013, pp. 1120–1127.

[28] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 109–117.

[29] P. Donmez and J. G. Carbonell, "Proactive learning: Cost-sensitive active learning with multiple imperfect oracles," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 619–628.

[30] Q. Li, F. Ma, J. Gao, L. Su, and C. J. Quinn, "Crowdsourcing high quality labels with a tight budget," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, 2016, pp. 237–246.

[31] X. Wu, Y. Dong, J. Tao, C. Huang, and N. V. Chawla, "Reliable fake review detection via modeling temporal and behavioral patterns," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 494–499.

[32] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, "Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 1986–1999, Aug. 2016.

[33] C. Huang, D. Wang, and S. Zhu, "Where are you from: Home location profiling of crowd sensors from noisy and sparse crowdsourcing data," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[34] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *J. Artif. Intell. Res.*, vol. 11, pp. 131–167, 1999.

[35] X. Zhu, X. Wu, and Q. Chen, "Eliminating class noise in large datasets," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 920–927.

[36] I. Triguero, J. A. Sáez, J. Luengo, S. García, and F. Herrera, "On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification," *Neurocomput.*, vol. 132, pp. 30–41, 2014.

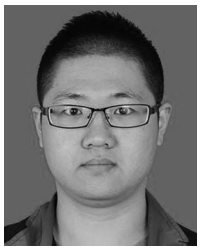
[37] Y. Li and P. M. Long, "The relaxed online maximum margin algorithm," *Mach. Learn.*, vol. 46, no. 1/3, pp. 361–387, 2002.

[38] S. Sukhbaatar, J. Bruna, M. Paluri, and L. B. R. Fergus, "Training convolutional networks with noisy labels," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2015, pp. 1–11.

[39] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1196–1204.

[40] W. Jiang, "Some theoretical aspects of boosting in the presence of noisy data," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 234–241.
[41] Y. Freund, "An adaptive version of the boost by majority algorithm," *Mach. Learn.*, vol. 43, no. 3, pp. 293–318, 2001.
[42] J. Abellán and A. R. Masegosa, "Bagging decision trees on data sets with classification noise," in *Proc. Int. Symp. Found. Inf. Knowl. Syst.*, 2010, pp. 248–265.
[43] E. Simpson, S. Roberts, I. Psorakis, and A. Smith, "Dynamic Bayesian combination of multiple imperfect classifiers," in *Decision Making and Imperfection*. Berlin, Germany: Springer, 2013, pp. 1–35.
[44] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, 2000.
[45] R. A. McDonald, D. J. Hand, and I. A. Eckley, "An empirical comparison of three boosting algorithms on real data sets with artificial class noise," in *Proc. Int. Workshop Multiple Classifier Syst.*, 2003, pp. 35–44.
[46] H. Li, B. Yu, and D. Zhou, "Error rate analysis of labeling by crowdsourcing," in *Proc. ICML Workshop: Mach. Learn. Meets Crowdsourcing*, 2013.
[47] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001.
[48] C. Mallah, J. Cope, and J. Orwell, "Plant leaf classification using probabilistic integration of shape, texture and margin features," *Signal Process. Pattern Recognit. Appl.*, vol. 5, pp. 45–54, 2013.
[49] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 3, pp. 659–665, May/Jun. 2002.
[50] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*, Cambridge, MA, USA, MIT Press, 1999, pp. 185–208.

Jing Zhang received the MS degree in computer science from the Graduate University of Chinese Academy of Sciences, Beijing, China, in 2006, and the PhD degree in computer science from the Hefei University of Technology, Hefei, China, in 2015. He is an associate professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include data mining, machine learning, and their applications in business and industry. He has published several articles in some prestigious journals, such as the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Neural Networks and Learning Systems*, the *Journal of Machine Learning Research*, the *IEEE Transactions on Cybernetics*, and top-tier conferences, such as SIGKDD, AAAI, SIGIR, CIKM, etc. He is a member of the IEEE. He serves as a PC member for a number of international conferences and a reviewer for several international journals.



Ming Wu received the bachelor's degree in software engineering from the Nanjing University of Science and Technology, China, in 2014. He is working toward the PhD degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include machine learning and data mining.

Victor S. Sheng received the master's degree in computer science from the University of New Brunswick, Canada, in 2003, and the PhD degree in computer science from Western University, Ontario, Canada, in 2007. He is an associate professor of computer science, University of Central Arkansas, and the founding director of the Data Analytics Lab (DAL). His research interests include data mining, machine learning, and related applications. He was an associate research scientist and NSERC postdoctoral fellow in information systems at Stern Business School, New York University, after he obtained his PhD. He is a senior member of the IEEE and a lifetime member of the ACM. He received the best paper award runner-up from KDD '08, and the best paper award from ICDM '11. He is a PC member for a number of international conferences and a reviewer for several international journals.