Robust Crowdsourced Learning

Zhiquan Liu, Luo Luo, Wu-Jun Li
Shanghai Key Laboratory of Scalable Computing and Systems
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
liuzhiquan@sjtu.edu.cn, ricky@sjtu.edu.cn, liwujun@cs.sjtu.edu.cn

***Abstract*—In general, a large amount of labels are needed for supervised learning algorithms to achieve satisfactory performance. It's typically very time-consuming and money-consuming to get such kind of labeled data. Recently, crowdsourcing services provide an effective way to collect labeled data with much lower cost. Hence, crowdsourced learning (CL), which performs learning with labeled data collected from crowdsourcing services, has become a very hot and interesting research topic in recent years. Most existing CL methods exploit only the labels from different workers (annotators) for learning while ignoring the attributes of the instances. In many real applications, the attributes of the instances are actually the most discriminative information for learning. Hence, CL methods with attributes have attracted more and more attention from CL researchers. One representative model of such kind is the *personal classifier* (PC) model, which has achieved the state-of-the-art performance. However, the PC model makes an unreasonable assumption that all the workers contribute equally to the final classification. This contradicts the fact that different workers have different quality (ability) for data labeling. In this paper, we propose a novel model, called *robust personal classifier* (RPC), for robust crowdsourced learning. Our model can automatically learn an expertise score for each worker. This expertise score reflects the inherent quality of each worker. The final classifier of our RPC model gives high weights for good workers and low weights for poor workers or spammers, which is more reasonable than PC model with equal weights for all workers. Furthermore, the learned expertise score can be used to eliminate spammers or low-quality workers. Experiments on simulated datasets and UCI datasets show that the proposed model can dramatically outperform the baseline models such as PC model in terms of classification accuracy and ability to detect spammers.**

***Index Terms*—crowdsourcing; crowdsourced learning; supervised learning**

I. Introduction

The big data era brings a huge amount of data for analyzing, and consequently provides machine learning researchers with many new opportunities. In general, a large amount of labels are needed for supervised learning algorithms to achieve satisfactory performance. Traditionally, the labels are provided by domain experts and the labeling cost is high in terms of both time and money.

With the advent of crowdsourcing and human computation [1], [2] in recent years, it becomes practical to annotate large amounts of data with low cost. For example, with Internet-based crowdsourcing services, such as Amazon Mechanical Turk[1] and CrowdFlower[2], it has become relatively cheap and less time-consuming to acquire large amounts of labels from crowds. Another interesting case is the construction of ImageNet [3] during which a large number of images were efficiently labeled and classified by crowds. As crowdsourcing services become more and more popular, *crowdsourced learning* (CL), which performs learning with labeled data collected from crowdsourcing services, has become a very hot and interesting research topic in recent years [4]–[15].

Compared with the labels given by human experts, the labels collected from crowds are noisy and subjective because workers (annotators) vary widely in their quality and expertise. Previous work on crowdsourcing [16] has shown that there exist some workers who give labels randomly. Those workers who give labels randomly without considering the features (attributes) are called spammers. Spammers will give labels randomly to earn money. Besides the spammers, some low-quality workers will also give noisy labels for the data. Sorokin and Forsyth [17] reports that some of the errors come from sloppy annotations. Hence, the existence of noisy labels makes CL become a very challenging learning problem.

To handle the noise problem in CL, repeated labeling is proposed to estimate the correct labels from noisy labels [18]. Snow et al. [16] find that a small number of nonexpert annotations per instance can perform as well as an expert annotator. Hence, the typical setting of CL is that each training instance has multiple labels from multiple workers with different quality (ability).

The existing CL methods can be divided into two classes according to whether instance features (attributes) are exploited for learning or not. The first class of methods exploits only the labels from different workers (annotators) for learning while ignoring the attributes of the instances. Most of the existing methods, such as those in [19]–[22], belong to this class.

In many real applications, the attributes of the instances are actually the most discriminative information for learning. Hence, CL methods with attributes have attracted more and more attention from CL researchers. Very recently, some methods are proposed to exploit attributes for learning which have shown promising performance in real applications [6], [10], [11], [23]. Raykar et al. [6], [11], [23] propose a two-coin model and extensions which can learn a classifier from attributes and estimate ground truth labels simultaneously. The drawback of this two-coin model is that it fails to model the difficulty of each training instance. In [10], a *personal classifier* (PC) model is proposed which can model both the ability of workers and the difficulty of instances. Experimental

[1]https://www.mturk.com
[2]http://crowdflower.com/

results in [10] show that the PC model can achieve better performance than most state-of-the-art models, including the two-coin model.

However, the PC model makes an unreasonable assumption that all the workers contribute equally to the final classification. This contradicts the fact that different workers have different quality (ability) for data labeling. In this paper, we propose a novel model, called *robust personal classifier* (RPC), for robust crowdsourced learning. Our model can automatically learn an expertise score for each worker. This expertise score reflects the inherent quality of each worker. The final classifier of our RPC model gives high weights for good workers and low weights for poor workers or spammers, which is more reasonable than PC model with equal weights for all workers. Furthermore, the learned expertise score can be used to eliminate spammers or low-quality workers. Experiments on simulated datasets and UCI datasets show that the proposed model can dramatically outperform the baseline models such as PC model in terms of classification accuracy and ability to detect spammers.

II. Personal Classifier Model

In this section, we first introduce the setting of crowdsourced learning (CL), or learning from crowds [6]. Then we briefly introduce the *personal classifier* (PC) model [10].

A. Crowdsourced Learning

A typical CL problem consists of a training set $\mathcal{T} = \{\mathbf{X}, \mathbf{Y}, \mathcal{I}\}$, where $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{R}^D\}_{i=1}^M$ is the matrix representation of the M training instances with D features, the ith row of $\mathbf{X}$ corresponds to the training instance $\mathbf{x}_i$, the jth column of $\mathbf{X}$ corresponds to the jth feature of the instances, $\mathbf{Y}$ is a matrix of size $M \times N$ with N being the number of annotators, y_{ij} denotes the element at the ith row and the jth column of $\mathbf{Y}$ which is the label of instance i given by annotator j. Note $\mathbf{Y}$ may contain a lot of missing entries in practice, because it is not practical for each annotator to label all the instances. $\mathcal{I}$ is an indicator matrix with the same size as $\mathbf{Y}$, where $\mathcal{I}_{i,j} = 1$ denotes the ith instance is actually labeled by the jth annotator and $\mathcal{I}_{i,j} = 0$ otherwise. We also define $\mathcal{I}_j$ as the set of instances which are labeled by the jth annotator. We focus on the binary classification in this paper, although the algorithm can be easily extended to multiple-class cases.

B. Personal Classifier Model

The probabilistic graphical model for PC model is shown in Fig. 1 (a). It assumes that the final classifier (base model) is a logistic function parameterized by $\mathbf{w}_0$:

$$P(y = 1|\mathbf{x}, \mathbf{w_0}) = \sigma(\mathbf{w}_0^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}_0^T \mathbf{x})}. \quad (1)$$

To overcome overfitting, a zero-mean Gaussian prior is put on the parameter $\mathbf{w}_0$:

$$P(\mathbf{w}_0|\eta) = \mathcal{N}(\mathbf{0}, \eta^{-1}\mathbf{I}), \quad (2)$$

where $\mathcal{N}(\cdot)$ denotes the normal distribution, η is a hyperparameter, and $\mathbf{I}$ denotes an identity matrix whose dimensionality depends on context.

The jth annotator is assumed to give labels according to a logistic function parameterized by $\mathbf{w}_j$:

$$P(y = 1|\mathbf{x}, \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}_j^T \mathbf{x})}. \quad (3)$$

All the $\mathbf{w}_j$ are assumed to be generated from a Gaussian distribution with mean $\mathbf{w}_0$:

$$P(\mathbf{w}_j|\mathbf{w}_0, \lambda) = \mathcal{N}(\mathbf{w}_0, \lambda^{-1}\mathbf{I}), \quad (4)$$

with λ being a hyperparameter.

Putting together all the above assumptions, we can get the negative log-posteriori as follows:

$$\begin{aligned} f(\mathbf{w}_0, \mathbf{W}) = &- \sum_{j=1}^{N} \sum_{i \in \mathcal{I}_j} l(y_{ij}, \sigma(\mathbf{w}_j^T \mathbf{x}_i)) \\ &+ \sum_{j=1}^{N} \frac{\lambda}{2} ||\mathbf{w}_j - \mathbf{w}_0||^2 + \frac{\eta ||\mathbf{w}_0||^2}{2} + c_1, \end{aligned} \quad (5)$$

where $\mathbf{W} = \{\mathbf{w}_j\}_{j=1}^N$, $l(s,t) = s\log(t) + (1-s)\log(1-t)$ is the logistic loss, and c_1 is a constant independent of the parameters.

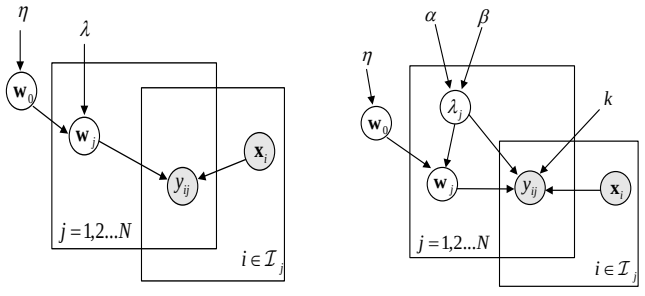
To solve the convex optimization problem in (5), an iteration algorithm with two steps is derived in [10]. The first step is to update $\mathbf{w}_0$ with $\mathbf{W}$ fixed:

$$\mathbf{w}_0 = \frac{\lambda \sum_{j=1}^N \mathbf{w}_j}{\eta + N\lambda}. \quad (6)$$

The second step is to update $\mathbf{W}$ with $\mathbf{w}_0$ fixed. The Newton-Raphson method is employed to update each $\mathbf{w}_j$ separately:

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \gamma [\mathbf{H}(\mathbf{w}_j^t)]^{-1} \mathbf{g}(\mathbf{w}_j^t), \quad (7)$$

where $\mathbf{w}_j^t$ denotes the value of iteration t, γ is the learning rate, $\mathbf{H}(\mathbf{w}_j^t)$ is the Hessian matrix and $\mathbf{g}(\mathbf{w}_j^t)$ is the gradient.



(a) PC model (b) RPC model

Fig. 1. Graphical models of PC and RPC.

III. Robust Personal Classifier

From (6), we can find that all the annotators ($\{\mathbf{w}_j\}_{j=1}^N$) contribute equally to the final classifier ($\mathbf{w}_0$), which is unreasonable because different annotators may have different ability. In this paper, we propose a *robust personal classifier* (RPC) model to learn the expertise score of each annotator. More specifically, each annotator will be associated with an expertise score, which can be automatically learned during the learning process of our model. The expertise score can be used to rank annotators and eliminate spammers.

A. Model

Fig. 1 (b) shows the probabilistic graphical model of our RPC model.

Like PC model, the final classifier (base model) of RPC for prediction is also a logistic function parameterized by $\mathbf{w}_0$:

$$P(y=1|\mathbf{x},\mathbf{w_0}) = \sigma(\mathbf{w}_0^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}_0^T\mathbf{x})},$$
$$P(\mathbf{w}_0|\eta) = \mathcal{N}(\mathbf{0},\eta^{-1}\mathbf{I}).$$

The jth annotator is also associated with a logistic function parameterized by $\mathbf{w}_j$. The prediction functions of the annotators in RPC model are as follows:

$$P(y_{ij}|\mathbf{w}_j,\mathbf{x}_i,\lambda_j) = \mathcal{N}(\sigma(\mathbf{w}_j^T\mathbf{x}_i),(k\lambda_j)^{-1}), \quad (8)$$
$$P(\mathbf{w}_j|\mathbf{w}_0,\lambda_j) = \mathcal{N}(\mathbf{w}_0,\lambda_j^{-1}\mathbf{I}), \quad (9)$$
$$P(\lambda_j|\alpha,\beta) = \mathcal{G}(\alpha,\beta) = \frac{\beta^\alpha\lambda_j^{\alpha-1}\exp(-\beta\lambda_j)}{\Gamma(\alpha)}, \quad (10)$$

where α, β and k are hyperparameters, and $\Gamma(\alpha) = \int_0^\infty s^{\alpha-1}\exp(-s)ds$ is the Gamma function.

The main difference between RPC model and PC model lies in the distributions of $P(y_{ij}|\mathbf{w}_j,\mathbf{x}_i,\lambda_j)$ and $P(\mathbf{w}_j|\mathbf{w}_0,\lambda_j)$. More specifically, all the annotators share the same λ in PC model. However, we associate different annotators with different values of $\{\lambda_j\}$s in RPC model, which actually reflect the expertise (ability) of annotators. This can be easily seen from the following learning procedure.

B. Learning

The maximum a posteriori (MAP) estimator of the model parameter $\mathbf{w}_0$ and $\mathbf{W}$ can be obtained by minimizing the following negative log-posteriori:

$$f(\mathbf{w}_0,\mathbf{W}) = \sum_{j=1}^N\sum_{i\in\mathcal{I}_j}\frac{k\lambda_j}{2}[y_{ij}-\sigma(\mathbf{w}_j^T\mathbf{x}_i)]^2 + \sum_{j=1}^N\frac{\lambda_j}{2}||\mathbf{w}_j-\mathbf{w}_0||^2 + \frac{\eta||\mathbf{w}_0||^2}{2} + c_2, \quad (11)$$

where c_2 is a constant independent of the parameters.

Solving the above optimization problem allows us to jointly learn the model parameters $\mathbf{w}_0$ and $\mathbf{W}$, and the expertise scores $\{\lambda_j\}$.

We devise an alternating algorithm with two steps to learn the parameters. In the first step, we fix $\{\lambda_j\}$, and then optimize $\mathbf{w}_0$ and $\mathbf{W}$. In the second step, we fix $\mathbf{w}_0$ and $\mathbf{W}$, and then optimize $\{\lambda_j\}$.

1) Optimization w.r.t. $\mathbf{w}_0$ and $\mathbf{W}$: We update $\mathbf{w}_0$ with $\mathbf{W}$ fixed, and then update $\mathbf{W}$ with $\mathbf{w}_0$ fixed. We repeat these two steps until convergence.

Given $\mathbf{W}$ is fixed, setting the gradient of (11) w.r.t. $\mathbf{w}_0$ to zero, we get

$$\mathbf{w}_0 = \frac{\sum_{j=1}^N(\lambda_j\mathbf{w}_j)}{\eta+\sum_{j=1}^N\lambda_j}, \quad (12)$$

where we can find that annotators with different expertise scores contribute unequally to the final classifier. This is different from the PC model in (6).

Given $\mathbf{w}_0$ is fixed, we can find that the parameters $\{\mathbf{w}_j\}_{j=1}^N$ are independent of each other. Hence, we can optimize each $\mathbf{w}_j$ separately. The PC model uses Newton-Raphson method to solve the problem w.r.t. $\mathbf{w}_j$, which is shown in (7). One problem with Newton-Raphson method is that we need to manually set the learning rate γ in (7). However, it is not easy to find a suitable learning rate in practice. Furthermore, the learning method in (7) cannot necessarily guarantee convergence, which will cause a problem about how to terminate the learning procedure.

In this paper, we design a surrogate optimization algorithm [24] for learning, which can guarantee convergence. Furthermore, there are no learning rate parameters for tuning in the surrogate algorithm, which can overcome the shortcomings of the learning algorithm in PC model.

The gradient $\mathbf{g}(\mathbf{w}_j)$ can be computed as follows:

$$\mathbf{g}(\mathbf{w}_j) = \lambda_j(\mathbf{w}_j-\mathbf{w}_0)+k\lambda_j\sum_{i\in\mathcal{I}_j}2(\sigma-y_{ij})\sigma(1-\sigma)\mathbf{x}_i, \quad (13)$$

where σ is short for $\sigma(\mathbf{w}_j^T\mathbf{x}_i)$.

The Hessian matrix $\mathbf{H}(\mathbf{w}_j)$ can be computed as follows:

$$\mathbf{H}(\mathbf{w}_j) = \lambda_j\mathbf{I}+ k\lambda_j\Big[\sum_{i\in\mathcal{I}_j}\sigma(\sigma-1)[3\sigma^2-2(y_{ij}+1)\sigma+y_{ij}]x_{im}x_{in}\Big]_{m,n}$$

where x_{im} denotes the mth element of $\mathbf{x}_i$, $[g(m,n)]_{m,n}$ denotes a matrix with the (m,n)th element being $g(m,n)$.

Let $s(\sigma) = \sigma(\sigma-1)[3\sigma^2-2(y_{ij}+1)\sigma+y_{ij}]$. Because $0<\sigma<1$, we can prove that $s(\sigma)\le 0.0770293$. Let $\tilde{\mathbf{H}}(\mathbf{w}_j) = \lambda_j\mathbf{I}+0.0770293k\lambda_j\sum_{i\in\mathcal{I}_j}\{\mathbf{x}_i\mathbf{x}_i^T\}$. We can prove that $\mathbf{H}(\mathbf{w}_j)\preceq\tilde{\mathbf{H}}(\mathbf{w}_j)$. With the surrogate optimization techniques [24], we can construct an upper bound of the original objective function. By optimizing the upper bound which is also called a surrogate function, we can get the following update rule:

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - [\tilde{\mathbf{H}}(\mathbf{w}_j^t)]^{-1}\mathbf{g}(\mathbf{w}_j^t). \quad (14)$$

Compared with the learning algorithm in (7), it is easy to find that there is no learning rate parameter for tuning in (14). We can also prove that this update rule can guarantee convergence. The detailed derivation and proof are omitted for space saving.

2) Optimization w.r.t. $\{\lambda_j\}$: Let $\mathbf{y}_j = \{y_{ij} | i \in \mathcal{I}_j\}$. We can update λ_j with the learned $\mathbf{w}_0$ and $\mathbf{W}$:

$$
\begin{aligned}
& P(\lambda_j | \mathbf{w}_j, \mathbf{w}_0, \mathbf{X}, \mathbf{y}_j) \\
& \propto P(\mathbf{y}_j | \mathbf{X}, \mathbf{w}_j, \lambda_j) \times P(\mathbf{w}_j | \mathbf{w}_0, \lambda_j) \times P(\lambda_j | \alpha, \beta) \\
& \propto [\prod_{i \in \mathcal{I}_j} \exp(\frac{k\lambda_j [y_{ij} - \sigma(\mathbf{w}_j^T \mathbf{x}_i)]^2}{2})] \times \exp(\frac{\lambda_j ||\mathbf{w}_j - \mathbf{w}_0||^2}{2}) \\
& \quad \times \lambda_j^{\alpha - 1} \exp(-\beta \lambda_j) \\
& \propto \lambda_j^{\frac{D + |\mathcal{I}_j|}{2} + \alpha - 1} \times \\
& \quad \exp[-(\beta + \frac{||\mathbf{w}_j - \mathbf{w}_0||^2 + k \sum_{i \in \mathcal{I}_j} [y_{ij} - \sigma(\mathbf{w}_j^T \mathbf{x}_i)]^2}{2}) \lambda_j] \ .
\end{aligned}
$$

Hence, $p(\lambda_j | \mathbf{w}_j, \mathbf{w}_0, \mathbf{X}, \mathbf{y}_j) = \mathcal{G}(\hat{\alpha}, \hat{\beta})$ where:

$$
\begin{aligned}
\hat{\alpha} &= \alpha + \frac{D + |\mathcal{I}_j|}{2}, \\
\hat{\beta} &= \beta + \frac{1}{2}(||\mathbf{w}_j - \mathbf{w}_0||^2 + k \sum_{i \in \mathcal{I}_j} [y_{ij} - \sigma(\mathbf{w}_j^T \mathbf{x}_i)]^2),
\end{aligned}
$$

where D is the dimensionality of instances, and $|\mathcal{I}_j|$ denotes the number of elements in the set $\mathcal{I}_j$.

The expectation of λ_j is:

$$
\begin{aligned}
\hat{\lambda}_j = & E(\lambda_j) = \frac{\hat{\alpha}}{\hat{\beta}} \\
= & \frac{2\alpha + D + |\mathcal{I}_j|}{2\beta + ||\mathbf{w}_j - \mathbf{w}_0||^2 + k \sum_{i \in \mathcal{I}_j} [y_{ij} - \sigma(\mathbf{w}_j^T \mathbf{x}_i)]^2}. \quad (15)
\end{aligned}
$$

We can get some intuition from (15). $||\mathbf{w}_j - \mathbf{w}_0||^2$ measures the difference from the parameter of the jth personal classifier to the parameter of final classifier (learned ground-truth classifier), and $\sum_{i \in \mathcal{I}_j} [y_{ij} - \sigma(\mathbf{w}_j^T \mathbf{x}_i)]^2$ denotes the error of the personal classifier on the training data. The larger these differences (errors) are, the smaller the corresponding λ_j will be. Thus, λ_j reflects the expertise score (ability) of the annotator j, which can be automatically learned from the training data.

By combining (15) with (12), we can get an algorithm which can automatically learn an expertise score for each worker. Based on these learned scores, our RPC model can learn a final classifier contributed more from good workers but less from poor workers or spammers. This is more reasonable than PC model with equal weights for all workers.

3) Summarization: We summarize the algorithm for RPC model in Algorithm 1.

During the learning of RPC, we can eliminate the spammers in each iteration after we have found them. The performance of the learned classifier in the following iterations can be expected to improve due to the reduced noise (spammer). In Algorithm 2, we present the variant of the PRC algorithm, called RPC2, that iteratively eliminates spammers in each iteration.

Algorithm 1 Robust personal classifier (RPC)

Input: features $\{\mathbf{x}_i\}_{i=1}^M$
  labels $y_{ij} \in \{0, 1\}, i = 1...M, j = 1...N$
  indicator matrix $\mathcal{I}$
  max_iter
  while $iter_num < max_iter$ **do**
    update $\mathbf{w}_0$ based on (12)
    update each $\mathbf{w}_j$ based on (14)
    update each λ_j based on (15)
  end while
Output: $\mathbf{w}_0$, $\mathbf{W}$, $\{\lambda_j\}_{j=1}^N$

Algorithm 2 Robust personal classifier with spammer elimination (RPC2)

Input: features $\{\mathbf{x}_i\}_{i=1}^M$
  labels $y_{ij} \in \{0, 1\}, i = 1...M, j = 1...N$
  indicator matrix $\mathcal{I}$
  max_iter
  $spammer_num$
  while $remove_num < spammer_num$ **do**
    while $iter_num < max_iter$ **do**
      update $\mathbf{w}_0$ based on (12)
      update each $\mathbf{w}_j$ based on (14)
      update each λ_j based on (15)
      sort $\{\lambda_j\}_{j=1}^N$, and remove z workers with the lowest values of λ_j
      $remove_num = remove_num + z$
    end while
  end while
Output: $\mathbf{w}_0$, $\mathbf{W}$, $\{\lambda_j\}_{j=1}^N$

IV. Experiments

In this section, we compare our model with some baseline methods including state-of-the-art methods on CL. We validate the proposed algorithms on both simulated datasets and UCI benchmark datasets. k is set to 1 in our experiments.

A. Baseline Methods

We compare our RPC model with two baseline methods, majority voting (MV) and PC model, to evaluate the effectiveness of RPC model. MV is a commonly used heuristic method in CL tasks, and PC model is the most related one. Furthermore, PC model has achieved the state-of-the-art performance according to the experiments in [10].

1) Majority Voting: In MV, all the annotators contribute equally and a training instance is assigned the label which gets the most vote. This method is very simple but strong in practice. We train a logistic regression classifier on the consensus labels. MV can be adapted to measure the expertise of each annotator (worker). We compute the similarity between the labels given by each worker and the majority voted labels. The similarity is treated as a measure of worker's expertise based on the fact that high-quality workers usually give similar labels as the ground-truth labels.

2) Personal Classifier (PC) Model: The PC model in [10] can learn a classifier for the underlying ground-truth labels. So we can measure its area under the curve (AUC) on the testing data. However, the PC model does not provide a direct mechanism to measure the ability (expertise) of each worker, so we only compare RPC with MV in terms of discriminating good workers from spammers in the experiment.

B. Simulated Data

We first validate our algorithm on simulated data. We assume there exist two types of annotators. The first type is called good annotators. Due to worker ability and understanding bias for the labeling task, the good annotators are assumed to give correct labels at a certain probability which ranges from 0.65 to 0.85 in the experiment. The second type of annotators is spammers. Spammers are assumed to give labels randomly regardless of the features.

The dimensionality of the feature vectors is 30 and each dimension is generated from a uniform distribution $U([-0.5, 0.5])$. The parameter of the base model $\mathbf{w}_0$ is generated from a Gaussian distribution with zero mean and identity covariance matrix. The ground truth labels are calculated from the logistic function in (8). The noisy labels given by each worker are then generated according to whether they are good annotators or spammers. For all the experiments, we run the experiments 10 times and report the average results.

Let M denote the number of training instances. For all the experiments, we generate $10M$ instances as the testing set. Let R denote the number of good annotators and S denote the total number of annotators. So the number of spammers is $S - R$. We use two metrics to evaluate our algorithms against baseline methods. As the ground truth labels are known, we compute the AUC for all the classifiers. Another evaluation metric is the ability to detect good annotators. We rank the expertise scores and fetch top n workers. Among them, we compute the precision of good annotators. The n is set to R in our experiment if there are R good annotators in the training set.

1) Classification Accuracy: We report the AUC on some datasets with different M, R and S in Table I.

TABLE I
AUC PERFORMANCE

Data Set	Parameters	MV	PC	RPC
Random Data 1	M=100,R=5,S=50	0.6014	0.6608	0.6718
Random Data 2	M=200,R=5,S=50	0.7544	0.8233	0.9032
Random Data 3	M=400,R=5,S=50	0.7760	0.8559	0.9520
Random Data 4	M=300,R=5,S=10	0.8838	0.9278	0.9695
Random Data 5	M=300,R=5,S=50	0.7153	0.8077	0.9091
Random Data 6	M=300,R=5,S=90	0.6641	0.7240	0.8346

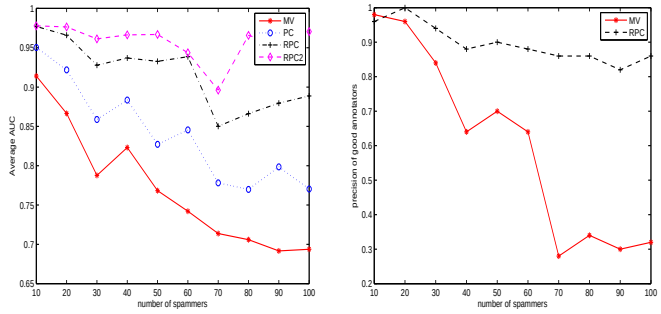
We can see from Table I that RPC outperforms PC model and MV method on all the datasets.

2) Ability to Discriminate Good Annotators from Spammers: We evaluate the ability of the proposed RPC model to discriminate good annotators from spammers. We generate 5 good annotators in this experiment. We rank the expertise scores and check the ratio of the 5 highest scores to be truly good annotators, we call this metric precision at top 5. Table II shows that RPC model can detect good annotators more accurately than MV method.

TABLE II
PRECISION OF DETECTING GOOD ANNOTATORS AT TOP 5

Data Set	Parameters	MV	RPC
Random Data 1	M=100,R=5,S=50	0.3200	0.5000
Random Data 2	M=200,R=5,S=50	0.4200	0.7600
Random Data 3	M=400,R=5,S=50	0.7400	0.9400
Random Data 4	M=300,R=5,S=10	0.9600	1.0000
Random Data 5	M=300,R=5,S=50	0.4800	0.7200
Random Data 6	M=300,R=5,S=90	0.3000	0.7600

3) Effect of the Number of Spammers: We are also interested in the sensitivity of performance when the number of spammers ranges from small to very large. Fig. 2 shows the AUC performance and the precision to detect good annotators in the top 5 positions with increasing number of spammers. The number of good annotators in the training set is 5. We can find that all the results degrade as more spammers are added. RPC model still performs better than PC model and MV method.



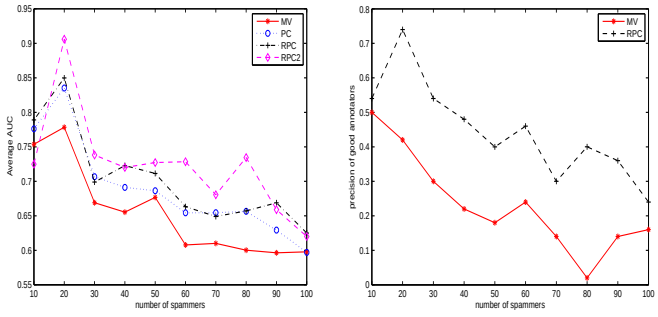
(a) AUC (b) precision

Fig. 2. AUC and precision of good annotators detected on simulated data. We vary the number of spammers from 10 to 100 by steps of 10.

4) Effect of Missing Labels: It is not practical for each worker to annotate all the instances in the dataset. We test our model in this scenario. Fig. 3 gives the AUC performance and the precision to detect good annotators in the top 5 positions with increasing number of spammers when each instance is labeled by 30% of the annotators. All the results drop significantly compared with those of complete labels, but the proposed RPC models still work better than the baselines.

C. UCI Benchmark Data

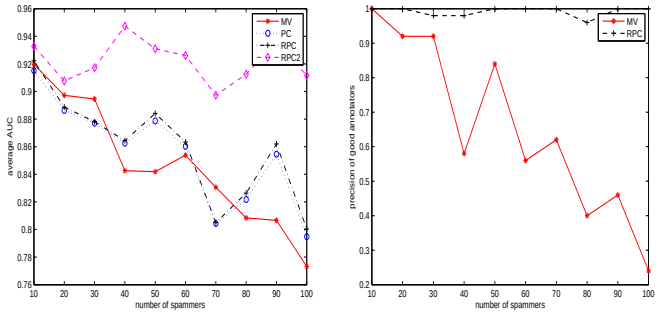
We use the breast cancer dataset [25] from the UCI machine learning repository [26] for evaluation. The cancer dataset contains 683 instances and each instance has 10 features (dimensions). In our experiments, 400 instances are used for training and the rest is for testing. We simulate the noisy labels with the same strategy as that in the previous section. We generate 5 good annotators and vary the number of spammers.



(a) AUC (b) precision

Fig. 3. AUC and precision of good annotators detected on simulated data with missing labels. We vary the number of spammers from 10 to 100 by steps of 10.

The AUC and precision of detecting good annotators are shown in Fig. 4. We can find that the proposed RPC can outperform both PC model and MV in terms of AUC, and RPC model does much better than MV in detecting good annotators. Moreover, the RPC2 can further improve the performance of PRC by eliminating the spammers during the learning procedure.



(a) AUC (b) precision

Fig. 4. AUC and precision of good annotators detected on UCI dataset. We vary the number of spammers from 10 to 100 by steps of 10.

V. Conclusion

A key problem in crowdsourced learning (CL) is about how to estimate accurate labels from noisy labels. To deal with this problem, we need to estimate the expertise level of each annotator (worker) and to eliminate the spammers who give random labels. In this paper, we propose a novel model, called *robust personal classifier* (RPC) model, to discriminate high-quality annotators from spammers. Extensive experimental results on several datasets have successfully verified the effectiveness of our model.

Future work will focus on empirical comparison between our model and other models, such as those in [11], on more real-world applications.

VI. Acknowledgements

This work is supported by the NSFC (No. 61100125), the 863 Program of China (No. 2012AA011003), and the Program for Changjiang Scholars and Innovative Research Team in University of China (IRT1158, PCSIRT).

References

[1] A. J. Quinn and B. B. Bederson, "Human computation: a survey and taxonomy of a growing field," in *Proceedings of the 2011 annual conference on Human factors in computing systems*. ACM, 2011, pp. 1403–1412.

[2] L. Von Ahn, "Human computation," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*. IEEE, 2009, pp. 418–419.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[4] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 20–28, 1979.

[5] P. Smyth, U. M. Fayyad, M. C. Burl, P. Perona, and P. Baldi, "Inferring ground truth from subjective labelling of venus images," in *NIPS*, 1994, pp. 1085–1092.

[6] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, pp. 1297–1322, 2010.

[7] Y. Yan, R. Rosales, G. Fung, and J. G. Dy, "Modeling multiple annotator expertise in the semi-supervised learning scenario," in *UAI*, 2010, pp. 674–682.

[8] ——, "Active learning from crowds," in *ICML*, 2011, pp. 1161–1168.

[9] J. Yi, R. Jin, A. K. Jain, S. Jain, and T. Yang, "Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning," in *NIPS*, 2012, pp. 1781–1789.

[10] H. Kajino, Y. Tsuboi, and H. Kashima, "A convex formulation for learning from crowds," in *AAAI*, 2012.

[11] V. C. Raykar and S. Yu, "Eliminating spammers and ranking annotators for crowdsourced labeling tasks," *Journal of Machine Learning Research*, vol. 13, pp. 491–518, 2012.

[12] Y. Baba and H. Kashima, "Statistical quality estimation for general crowdsourcing tasks," in *KDD*, 2013.

[13] H. Kajino, Y. Tsuboi, and H. Kashima, "Clustering crowds," in *AAAI*, 2013.

[14] S. Oyama, Y. Baba, Y. Sakurai, and H. Kashima, "Accurate integration of crowdsourced labels using workers' self-reported confidence scores," in *IJCAI*, 2013.

[15] K. Mo, E. Zhong, and Q. Yang, "Cross-task crowdsourcing," in *KDD*, 2013.

[16] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks," in *EMNLP*, 2008, pp. 254–263.

[17] A. Sorokin and D. Forsyth, "Utility data annotation with Amazon Mechanical Turk," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08*. IEEE, 2008, pp. 1–8.

[18] V. S. Sheng, F. J. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *KDD*, 2008, pp. 614–622.

[19] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *NIPS*, 2009, pp. 2035–2043.

[20] P. Welinder, S. Branson, S. Belongie, and P. Perona, "The multidimensional wisdom of crowds," in *NIPS*, 2010, pp. 2424–2432.

[21] Y. Tian and J. Zhu, "Learning from crowds in the presence of schools of thought," in *KDD*, 2012, pp. 226–234.

[22] D. Zhou, J. C. Platt, S. Basu, and Y. Mao, "Learning from the wisdom of crowds by minimax entropy," in *NIPS*, 2012, pp. 2204–2212.

[23] V. C. Raykar and S. Yu, "Ranking annotators for crowdsourced labeling tasks," in *NIPS*, 2011, pp. 1809–1817.

[24] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *Journal of Computational and Graphical Statistics*, vol. 9, no. 1, pp. 1–20, 2000.

[25] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology." *Proceedings of the National Academy of Sciences*, vol. 87, no. 23, pp. 9193–9196, 1990.

[26] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml