A Quality-Sensitive Method for Learning from Crowds

Jinhong Zhong, *Student Member, IEEE*, Peng Yang, *Student Member, IEEE*, and Ke Tang, *Senior Member, IEEE*

Abstract—In real-world applications, the oracle who can label all instances correctly may not exist or may be too expensive to acquire. Alternatively, crowdsourcing provides an easy way to get labels at a low cost from multiple non-expert annotators. During the past few years, much attention has been paid to learning from such crowdsourcing data, namely *Learning from Crowds* (LFC). Despite their proper statistical foundations, the existing methods for LFC still suffer from several disadvantages, such as needing prior knowledge to select the expertise model to represent the behavior of annotators, involving non-convex optimization problems, or restricting the classifier type being used. This paper addresses LFC from a quality-sensitive perspective and presents a novel framework named QS-LFC. Through reformulating the original LFC problem as a quality-sensitive learning problem, the above-mentioned disadvantages of existing methods can be avoided. Further, a support vector machine (SVM) implementation of QS-LFC is proposed. Experimental results on both synthetic and real-world data sets demonstrate that QS-LFC can achieve better generalization performance and is more robust to the noisy labels, than the existing methods.

Index Terms—Learning from crowds, crowdsourcing, learning from multiple annotators, quality-sensitive learning

1 Introduction

Many machine learning algorithms need labeled data to train a model for making predictions on new data. Conventional supervised learning techniques assume that there exists an oracle, from which the ground-truth labels of all instances can be obtained. This may not be true for many real-world applications, i.e., the oracle may not exist, or obtaining ground-truth labels from the oracle may be too expensive. Thus, it is infeasible to obtain the ground-truth labels. Moreover, in the Big Data era, many real-world applications of data mining need millions of labeled instances. And this kind of labeling tasks cannot be finished by one human annotator. For example, it may take overwhelmingly long time for a person to label millions of pictures. In this situation, crowdsourcing offers a tractable way to get labels from multiple annotators. A lot of works have shown that properly combining a group of non-expert annotators can be as good as the experts/oracles in many application domains [1], [2]. Recently, due to the development of crowdsourcing techniques, it has become possible that many human annotators work together on the same task online, e.g., via the Amazon's Mechanical Turk platform.[1]

On the other hand, the annotators who use crowdsourcing platform to label instances are not perfect, and their labels may be incorrect. As a result, there may be a substantial amount of disagreement among these annotators on the same instance. Hence, *Learning from Crowds* (LFC), which focus on how to learn a good prediction model with the labels collected from multiple imperfect annotators, has attracted much attention during the past few years [3], [4], [5].

Existing LFC methods fall into two categories, namely two-stage methods and direct methods. The two-stage methods contain two steps. In the first step, the ground-truth labels of the training instances are estimated from multiple noisy labels. And in the second step, the estimated labels are treated as the ground-truth labels such that traditional supervised learning approaches could be applied to train a classifier which was referred to as the target classifier in this paper. Among all the existing methods that estimate the ground-truth labels, Majority Voting (MV) may be the most straight-forward way that has often been employed as base-line method [6], [7], and many research efforts have also been taken to improve it [8], [9], [4]. Based on some behavior assumptions about annotators, some other methods build flexible probabilistic models for generating the noisy observations conditioned on the unknown true labels. Such as the Dawid-Skene (DS) estimator [10], Bayesian learning [11], minimax entropy [12] and variational inference [13]. Albeit simple, the disadvantages of two-stage methods are also apparent. First, the methods mentioned

- *J. Zhong and K. Tang are with the Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, and the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui 230027, China.*
  E-mail: zhongjinhong4213@gmail.com, tangk3@sustc.edu.cn.
- *P. Yang is with the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui 230027, China. E-mail: trevor@mail.ustc.edu.cn.*

Manuscript received 28 Aug. 2016; revised 30 July 2017; accepted 31 July 2017. Date of publication 11 Aug. 2017; date of current version 3 Nov. 2017. (Corresponding author: Ke Tang.)
Recommended for acceptance by R. Cheng.
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.2017.2738643

1. https://www.mturk.com/mturk/welcome

above do not use features when they estimate the ground-truth labels. Thus, they cannot utilize the relationship between data features and the ground-truth labels, which is a waste of the information. Second, during the training phase, only the estimated label of each instance will be reserved, and the detailed labels of each annotator will be discarded, which may lead to information loss again.

The other kind of methods for LFC is the direct methods, which focus on directly learning the target classifier from data instances together with the noisy labels collected from crowds (all annotators). Most of the existing direct methods use maximum likelihood criterion to infer the target classifier, and different methods make different assumptions about the annotators to build different likelihood functions. Raykar et al. [14], [15] use a two-coin model to represent the performance of each annotator in terms of sensitivity (true positive rate) and specificity (1-false positive rate) with respect to the unknown ground-truth labels. After imposing prior knowledge on sensitivity and specificity, the algorithms proposed in [14] and [15] iteratively estimate these two terms by Expectation Maximization (EM) method. However, these methods have a strong assumption about the expertise model of annotators. Specifically, they assume that the probability of an annotator providing the correct label of an instance is her/his sensitivity (for positive instances) or specificity (for negative instances). That means, for each annotator, she/he has only two different accuracies for all instances. In [16], this idea is generalized to multi-class problems. Yan et al. [17] relaxes this assumption, while at the same time introduces another assumption about the performance of annotators. More specifically, it is assumed that the performance of each annotator can be represented by a linear logistic regression. Bi et al. [18] introduces the dedication of annotator and the difficulty of instance to build another model. However, it suffers from the same problem, i.e., it introduces additional assumptions that leads to a more difficult optimization problem. It should be noted that all these methods involve solving non-convex optimization problems. Hence, there is no guarantee of obtaining optimal solutions.

Another group of direct methods is named Personal-Classifier (PC) methods. PC approaches use the personal classifier of each annotator to represent her/his labeling behavior. Kajino et al. [19] propose the first PC method. In [19], the target classifier and the personal classifiers are obtained at the same time by optimizing three goals: 1). the target classifier and each personal classifier should be as close as possible. 2). The difference between the output of each personal classifier and the labels of the corresponding annotator should be as small as possible. 3). The regularization term of target classifier should be as small as possible. Two hyper-parameters are introduced as the trade-off among these three goals. One assumption of [19] is that all annotators are equally important, hence each personal classifier has the same weight to determine the target classifier. Valizadegan et al. [20] introduces the self-consistency parameter and the consensus consistency parameter for each annotator to differentiate the reliabilities of different annotators. In [21], Kajino modifies the model to utilize the clusters of the annotators to distinguish different kinds of annotators. However, as a side effect, the latter two PC methods introduce additional hyper-parameters. Thus, the additional hyper-parameters make the problem of determining hyper-parameters become much more difficult, especially in the condition of LFC, since no ground-truth labels would be available for tuning these hyper-parameters.

The above introduction shows that direct methods may suffer from a number of drawbacks, such as 1). in direct methods, the expertise model, which is used to represent the reliabilities of annotators on instances, needs to be selected in prior, while it is non-trivial to select a suitable expertise model for a real-world application. 2). Most of the direct methods need to solve a hard non-convex problem. 3). Most of the existing direct methods are limited to linear classifier.

Motivated by the cost-sensitive learning, a quality-sensitive method is proposed in this paper. Specifically, the key difference between a LFC task and a traditional supervised learning task is that the former involves multiple imperfect annotators for each training instance rather than a single perfect oracle. During the course of LFC, the qualities of different labels may be different. Thus, the cost (or loss) induced by a classifier should be dependent on the reliability of annotator on the specific instance. For example, the classifier should not be punished (with a cost) if its output is inconsistent with a wrong label. This paper shows that under the novel framework named QS-LFC (Quality-Sensitive method for LFC), where the original LFC problem is reformulated as a quality-sensitive learning problem, there will be many advantages compared with existing methods.

The main difficulties of LFC include: (1) how to identify the different qualities of different labels, and (2) how to obtain the target classifier taking into account these qualities. As mentioned above, although these two issues have been tackled by several previous works, existing methods will meet different problems in real-world applications. Motivated by these problems, the main contributions of this work include: 1). Address the problem of LFC from a new perspective, i.e., quality-sensitive, and propose the QS-LFC framework that requires solving convex optimization problem regardless of the reliability model used. 2). Propose a new method to estimate the reliability of each label, such that the reliability of an annotator can be different over instances. 3). Propose a support vector machine implementation of QS-LFC.

The rest of this paper is organized as follows. The problem description of LFC and the analysis of LFC from the perspective of quality-sensitive learning are presented in Section 2. Then in Section 3, the proposed quality-sensitive framework QS-LFC for LFC is introduced. The SVM implementation of QS-LFC is detailed in Section 4. After that, Section 5 analyzes experimental studies. And finally, conclusions are given as the end of this paper in Section 6.

2 A Quality-Sensitive Framework for Learning from Crowds

2.1 Problem Formulation

Given a set of labeled data $\{(\mathbf{x}_i, z_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{R}^d$, a typical supervised learning problem is to train a good classifier to predict the labels of new instances. In case the true labels $\mathbf{Z} = \{z_i\}$ are unavailable at hand, multiple non-expert annotators are employed to label the data. Let y_i^t denote the label that the tth $(t \leq T)$ annotator assigns to the ith instance $\mathbf{x}_i$. For simplicity, this paper restricts the discussions on the

binary class classification. In this case, y_i^t may be one of three different values. y_i^t = +1/-1 represents that the tth annotator labels the ith instance positive/negative. $y_i^t = 0$ means that the tth annotator does not label the ith instance, and this kind of label is named missing label. $\mathbf{Y}^j = \{y_i^t\}_{t=j}$ are the labels of annotator j, and $\mathbf{Y} = \{y_i^t\}$ are all the labels from crowds. The goal of LFC is to train a classifier $f(\mathbf{x})$, which is named target classifier, from training data $\mathbf{X} = \{\mathbf{x}_i\}$ and $\mathbf{Y}$, to correctly predict the true labels of new instances. It is noted that the labels from these non-expert annotators might be inconsistent with respect to other annotators. Finally, let r_i^t be the reliability of label y_i^t, i.e., the probability of y_i^t being correct.

2.2 A Quality-Sensitive View on LFC

In conventional learning problems, a learning algorithm seeks a function $f : \mathcal{R}^d \to \mathcal{Z}$, where $\mathcal{R}^d$ is the input space, and $\mathcal{Z}$ is the output space. The function f is an element of the hypothesis space $\mathcal{H}$. The most common approach of traditional supervised learning to choose f is structural minimization. In structural minimization, there are two objectives. One is to make the function fit the training data. The other one is the function should be as simple as possible. In order to measure how well a function fits the training data, a loss function $l : \mathcal{R} \times \mathcal{Z} \to \mathcal{R}$ is defined, where $\mathcal{R}$ is the real space. The risk $R(f)$ of function f is defined as the total loss of f, i.e., $R(f) = \sum_{i=1}^{N} l(f(\mathbf{x}_i), z_i)$. To make the optimal function be simple, the function will be assigned with a regularization penalty to its complexity. The structural risk minimization incorporates this regularization penalty into the optimization of minimizing the risk $R(f)$. On this basis, the supervised learning optimization problem can be formulated as follows:

$$f(\mathbf{x}) = \arg\min_{f(\mathbf{x})\in\mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{N} l(f(\mathbf{x}_i), z_i), \tag{1}$$

where $\frac{1}{2}\|f\|_{\mathcal{H}}^2$ is the regularization penalty of f, and C is the trade-off between the regularization penalty and the risk.

Eq. (1) can be generalized to LFC, albeit the ground-truth labels $\mathbf{Z} = \{z_i | i = 1, 2, \ldots, N\}$ are not available. Suppose that all annotators correctly label instance $\mathbf{x}_i$, the loss of f on instance $\mathbf{x}_i$ in LFC can be $\frac{1}{T}\sum_{t=1}^{T} l(f(\mathbf{x}_i), y_i^t)$. Let the loss of a missing label be always 0, i.e., $l(f(\mathbf{x}_i), 0) = 0$ no matter what the value of $f(\mathbf{x}_i)$ is. The above loss can be reformulated as $\frac{1}{\phi_i}\sum_{t=1}^{T} l(f(\mathbf{x}_i), y_i^t)$, where $\phi_i = \sum_{t=1}^{T}[y_i^t \neq 0]$ is a normalized constant for i-th instance. $[\cdot]$ stands for 1 if $\cdot$ is true, and 0 otherwise. Therefore, the risk $R(f)$ of function f in this situation is $\sum_{i=1}^{N}\frac{1}{\phi_i}\sum_{t=1}^{T} l(f(\mathbf{x}_i), y_i^t)$. Thus, Eq. (1) becomes

$$f(\mathbf{x}) = \arg\min_{f(\mathbf{x})\in\mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{N}\frac{1}{\phi_i}\sum_{t=1}^{T} l(f(\mathbf{x}_i), y_i^t). \tag{2}$$

In the above mentioned situation, it is supposed that all the labels are correct. However, in real-world applications, the annotators taking part in the labeling task do not guarantee the correctness of her/his labels. In fact, the qualities of these labels usually vary over annotators, because different annotators may have different expertise. That is, some annotators may be more reliable than the others. Even for the same annotator, she/he may show different reliabilities on different instances. For this reason, the labels should not be regarded as equally important. That means, if the prediction of target classifier $f(\mathbf{x}_i)$ is different from a label y_i^t, the reliability of the tth annotator on instance $\mathbf{x}_i$ should be considered when determine the punishment/loss. Hence, QS-LFC is motivated through introducing additional weights into Eq. (2) to control the importance of each label according to its reliability. Finally, Eq. (2) becomes

$$f(\mathbf{x}) = \arg\min_{f(\mathbf{x})\in\mathcal{H}} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{N}\frac{1}{\phi_i'}\sum_{t=1}^{T} w_i^t \cdot l(f(\mathbf{x}_i), y_i^t), \tag{3}$$

where w_i^t is the weight for label y_i^t, $\phi_i' = \sum_{t=1}^{T} w_i^t \cdot [y_i^t \neq 0]$ is the normalized constant for the i-th instances. Eq. (3) allows the inconsistency between $f(\mathbf{x}_i)$ and y_i^t to cause different cost for different i or different t.

From Eq. (3), it can be found that the quality-sensitive method can be compatible with many different types of classifiers. In other words, $f(\mathbf{x})$ in Eq. (3) is a general formulation of classifier, and a variety kinds of classifiers can be applied. Besides, optimizing this object function is equal to training a classifier on $N \cdot T$ instances, each of which has its label and weight. Specifically, each of N instances will be made T copies, and each of these copies will be with a label provided by one annotator. Thus, the quality-sensitive method will not introduce any additional non-convex optimization problem. If the supervised learning optimization problem of Eq. (1) is convex, the optimization problem of Eq. (3) will also be convex.

Furthermore, all two-stage methods in the context of LFC can be viewed as special cases of the QS-LFC in Eq. (3). In two-stage methods, for all instances, only one label y_i^t will be selected as the estimated ground-truth label, and its weight w_i^t will be set to 1. The weights of all the other $T - 1$ labels of this instance will be set to 0. Hence, in addition to the disadvantages mentioned in introduction, another disadvantage of two-stage methods is that w_i^t can just be set to 1 or 0, which is unreasonable. In reality, there should be a wide range in the quality of labels, which leads to the real-valued w_i^t.

3 The Proposed Method QS-LFC

Based on the above analysis, QS-LFC will be proposed in this section. Generally, it involves two steps, i.e., determining the weight w_i^t and solving the optimization problem in Eq. (3). Intuitively, a weight should be related with the reliability of the corresponding label. More specifically, a more reliable label should be assigned with a larger weight. Hence, the problem of determining the weight can be divided into two sub-problems, i.e., estimating the reliability of each label and determining the weight according to the estimated reliability of each label.

3.1 The Selection of Weight Function

If the reliabilities of all labels are known, a weight function $q(x)$ should be defined to determine the weights of the labels based on their corresponding reliabilities. This weight function maps the reliability of a label to the weight assigned to the corresponding label. It should be noted that this function should satisfy the following two conditions.

1). $q(x)$ is monotonically increasing when x falls into the range of $[0, 1]$. As to the analysis above, a harsher punishment should be assigned to the target classifier if it makes a mistake on a more reliable label.
2). $q(x) = 0$ when $x = 0.5$. Let the reliability of a label be 0.5, i.e., the probability of this label being true is 0.5. Hence, this label is not better than a random guessing, and no punishment should be assigned even if the output of the target classifier is different from this label.

There are a lot of functions that meet the above conditions, such as linear function $x - 0.5$, logarithmic function $log(\frac{x+\epsilon}{1-x+\epsilon})$ where ϵ is a smooth factor in order to avoid division by zero, and exponential function $e^x - e^{0.5}$. Our preliminary experiments have showed that the results of $q(x) = x - 0.5$ and $q(x) = e^x - e^{0.5}$ were similar. But overall, the results of $q(x) = e^x - e^{0.5}$ were a little bit better than that of $q(x) = x - 0.5$. The reason might be that a superlinear increasing of weight is more rational than linear increasing. The experiment results of $q(x) = log(\frac{x+\epsilon}{1-x+\epsilon})$ were far worse than that of other two cases. The reason might be that when x approaches 1, the value of $log(\frac{x+\epsilon}{1-x+\epsilon})$, i.e., the value of $q(x)$, will be very large. If the value of $q(x)$ becomes very large when x approaches 1, a wrong estimation of r_i^t near 1 may have serious consequences. Therefore, in this paper, the exponential function $e^x - e^{0.5}$ is selected as the weight function. It should be noted that if the reliability r_i^t of label y_i^t is smaller than 0.5, $-y_i^t$ will be more likely to be a true label than y_i^t. In this case, y_i^t will be replaced by $-y_i^t$, and the new reliability for this updated label will be set to $1 - r_i^t > 0.5$.

3.2 Estimate the Reliabilities of Labels

Determining the weights in the QS-LFC method needs the reliabilities of labels to be known. However, in most real-world applications, the annotators will not provide their reliabilities of their labels. Therefore, a method to estimate the reliability of each label is needed.

There are some existing methods that can be adopted to estimate the reliabilities of labels, such as MV and Dawid-Skene estimator [10]. MV assumes that all annotators will label all instances with the same reliability. Hence, MV sets all r_i^t to the same value. Dawid-Skene estimator assumes different annotators have different reliabilities, while each annotator has the same reliability over all instances. Thus r_i^t are set to the same value for the ith annotator on all instances. Generally, directly utilizing the existing methods to estimate the reliabilities of labels in LFC may suffer from two drawbacks. First, these methods do not use features, which leads to the result that they cannot utilize the information of data features when estimating the reliabilities of labels. Second, all of the existing methods make strong assumptions about the behavior of annotators, such as the assumptions made in MV and Dawid-Skene estimator.

To overcome the above disadvantages of the existing methods, a new method is proposed to estimate the reliabilities of labels. Remind that, when applying supervised learning on a dataset, it is supposed that there exists a pattern to represent the relationship between data features and its ground-truth labels. For a very reliable annotator, the difference between her/his labels and the ground-truth labels is small, and the pattern of data will be well inherited in her/his labels. On the contrary, for a less reliable annotator, her/his labels will be quite different from ground-truth labels, and the original pattern of data is likely to be broken. Hence, how well the pattern is inherited in the labels of one annotator can be regarded as an indicator to estimate the quality of this annotator. Specifically, the labeled data $(\mathbf{X}, \mathbf{Y}^t)$ of annotator t is first divided into a training set and a test set. Then the classifier $f^t(\mathbf{x})$ of annotator t can be learned on the training set, and its accuracy r^t can be tested on the test set. Finally, r^t is regarded as the indicator to measure the quality of annotator t.

In addition to the quality of annotator t, whether this label is labeled incorrectly by accident should also be considered. Suppose that the average labeling accuracy of one annotator is 0.9, she/he will also make mistakes on some instances. For these labels that s/he makes mistakes, their reliabilities certainly should not be estimated as 0.9. Since $f^t(\mathbf{x}_i)$ is learned from the labeled data provide by annotator t, it can be used to predict the labels of this annotator. Hence, y_i^t is (is not) different from the prediction of $f^t(\mathbf{x}_i)$ can be regarded as an indicator that this label is (is not) an exception label of annotator t. The average labeling accuracy of an annotator is regarded as a priori estimation of the reliability of her/his labels. After the value of one label is observed, its posteriori estimation can be obtained by adjusting this priori estimation according to whether this label is different from the output of her/his classifier. Based on the analysis above, the method to estimate the reliability of a label is proposed as follows:

$$r_i^t = (r^t)^{[f^t(\mathbf{x}_i)=y_i^t]}(1 - r^t)^{[f^t(\mathbf{x}_i)=-y_i^t]}. \tag{4}$$

To further improve this estimation, the bagging technique [22] is used to obtain multiple classifiers $f^{t,k}(\mathbf{x})$ $(k = 1, 2, \ldots, K)$ and their test accuracies $r^{t,k}$ of annotator t. With regard to these K classifiers, the final reliability estimation of each label is shown as follows:

$$r_i^t = \frac{1}{\Phi}\left(\prod_{k=1}^{K}(r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=y_i^t]}(1 - r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=-y_i^t]}\right)^{1/K}, \tag{5}$$

where Φ is a normalized constant

$$\Phi = \sum_{y\in\{-1,+1\}}\left(\prod_{k=1}^{K}(r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=y]}(1 - r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=-y]}\right)^{1/K}. \tag{6}$$

According to Eq. (5), the personal classifier $f^{t,k}(\mathbf{x})$ plays an important role in the estimation of r_i^t. Generally, this estimation method tries to utilize each personal classifier as "weak classifier", i.e., the classifier is better than random guessing but not strong enough, to help recognize which labels are more likely to be true or false. Hence, if the majority of the annotators perform better than random guessing, the proposed estimation method will work well. However, since these personal classifiers are trained from annotator's labels rather than ground-truth labels, some classifiers might even be worse than the random guessing, and thereby lead to the wrong estimation of the reliability. An extreme case is that the annotator always flips the label. In this situation, the personal classifiers of this annotator are likely to be "bad classifiers" (worse than random guessing),

and the reliability of this annotator may be wrongly estimated as near 1, which may lead to the failure of the proposed method. In summary, this estimation method works under the condition that this kind of "bad classifiers" are minor and their bad effect on training target classifier is small with respect to the effect of major "weak classifiers". If the majority of the annotators perform worse than random guessing, e.g., the majority of annotators always flip the label, the proposed method will not work just like all the existing LFC methods.

3.3 Handling the Missing Labels

In many real-world applications, usually there are a large number of instances. Hence, it is impossible for one annotator to label all the instances, which will lead to a lot of missing labels ($y_i^t = 0$). To handle these missing labels, an easy way is to discard them. More specifically, the corresponding weights w_i^t of these missing labels are set to 0. However, if this method is applied, the algorithm cannot get any benefit from these null-valued labels. Discarding the missing labels cannot make good use of the high qualified annotators. In fact, since the proposed algorithm will obtain multiple classifiers for each annotator, these classifiers can be used to approximate the labeling manner of this annotator. Therefore, in the proposed method, the missing labels can be inferred by the following equation:

$$y_i^t = \underset{y\in\{-1,+1\}}{\operatorname{argmax}} \left(\prod_{k=1}^{K} (r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=y]} (1 - r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=-y]} \right)^{1/K}, \quad (7)$$

the reliability of the label is as follows:

$$r_i^t = \max_{y\in\mathcal{Z}} \frac{1}{\Phi} \left(\prod_{k=1}^{K} (r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=y]} (1 - r^{t,k})^{[f^{t,k}(\mathbf{x}_i)=-y]} \right)^{1/K}. \quad (8)$$

3.4 The Algorithm of QS-LFC

A formal description of the framework QS-LFC is summarized in Algorithm 1. Given the learner and crowdsourcing data, K personal classifiers will first be trained using bagging for each annotator (line 1). With these personal classifiers, the reliability of each label will be estimated (line 3). Then, the missing labels will be inferred, and the reliabilities of these inferred labels will also be estimated (line 4-5). With these estimated reliabilities and weight function $q(x)$, the weight of each label will be calculated (line 6). At last, the target classifier will be trained with these labels and weights (line 8).

Compared with the existing LFC methods, QS-LFC has many merits: 1). The proposed method can utilize the latent relationship between data features and ground-truth labels to self-learn the reliabilities of both annotators and labels. Other existing methods need to predefine and have strong limitation to the formulation of expertise/reliability models. 2). QS-LFC will not introduce additional non-convex problem. 3). QS-LFC is available for different kinds of classifiers, thus greatly expanding the area of real-world applications.

4 A Support Vector Machine Implementation of QS-LFC

Eq. (3) shows that QS-LFC is a general framework for LFC, and many learning models can be applied. Without loss of generality, SVM is chosen as the learning model of QS-LFC, and the SVM Implementation of QS-LFC is proposed.

Algorithm 1. QS-LFC: A Framework

Data: Instances $\mathbf{X}$, observed labels $\mathbf{Y}$, a base learning algorithm **Learner**, Bagging parameter K, weight function $q(x)$
Result: target classifier $f(\mathbf{x})$
begin
1: train K classifiers $f^{t,k}(\mathbf{x})$ for each annotator with **Learner** and bagging.
2: calculate test accuracy $r^{t,k}$ of each classifier.
3: calculate the reliability of each label using Eq. (5).
4: infer missing labels using Eq. (7).
5: calculate r_i^t of inferred labels using Eq. (8).
6: calculate w_i^t of each label with $q(x)$.
7: calculate ϕ_i' with $\phi_i' = \sum_{t=1}^{T} w_i^t \cdot [y_i^t \neq 0]$
8: learn the target classifier $f(\mathbf{x})$ using Eq. (3)

After each w_i^t has been obtained, if Eq. (3) is directly optimized to get the target classifier, it may need huge computational cost, especially when the number of annotators is large. Because optimizing this objective function is equal to training a classifier on $N \cdot T$ instances. All these $N \cdot T$ instances are with different labels and weights. More specifically, each of the N instances will be made T copies, and each of these copies will be with a label provided by one annotator. Take $\mathbf{x}_i$ as an example, the corresponding T instances are $(\mathbf{x}_i, y_i^1), (\mathbf{x}_i, y_i^2), \ldots (\mathbf{x}_i, y_i^T)$. Fortunately, we can simplify this optimization by merging the terms that with both the same label and the same instance. The process of the simplification is described as follows:

$$\begin{aligned}
&\sum_{i=1}^{N} \frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t \cdot l(f(\mathbf{x}_i), y_i^t) \\
&= \sum_{i=1}^{N} \frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t \cdot l(f(\mathbf{x}_i), y_i^t)[y_i^t = 1] \\
&+ \sum_{i=1}^{N} \frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t \cdot l(f(\mathbf{x}_i), y_i^t)[y_i^t = -1] \\
&= \sum_{i=1}^{N} \left(\frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t [y_i^t = 1] \right) \cdot l(f(\mathbf{x}_i), 1) \\
&+ \sum_{i=1}^{N} \left(\frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t [y_i^t = -1] \right) \cdot l(f(\mathbf{x}_i), -1) \\
&= \sum_{i=1}^{N} u_i \cdot l(f(\mathbf{x}_i), 1) + \sum_{i=1}^{N} v_i \cdot l(f(\mathbf{x}_i), -1),
\end{aligned} \quad (9)$$

where u_i and v_i are the sum of weights of positive labels and negative labels of instance $\mathbf{x}_i$ respectively. They can be calculated as follows:

$$\begin{aligned}
u_i &= \frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t [y_i^t = 1] \\
v_i &= \frac{1}{\phi_i'} \sum_{t=1}^{T} w_i^t [y_i^t = -1].
\end{aligned} \quad (10)$$

From Eq. (9), it is easy to know that the original optimization has been reduced into training on $2N$ weighted instances. More specifically, for any $i \in \{1, 2, \ldots, N\}$, the feature value of the ith instance is $\mathbf{x}_i$, its label is 1, and its weight is u_i. The feature value of the $(N+i)$th instance is also $\mathbf{x}_i$, its label is -1, and its weight is v_i. Therefore, the new optimization problem is presented as follows:

$$f(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \frac{1}{2}||f(\mathbf{x})||^2_{\mathcal{H}} + C\sum_{i=1}^{N} u_i l(f(\mathbf{x}_i), 1) + C\sum_{i=1}^{N} v_i l(f(\mathbf{x}_i), -1). \tag{11}$$

According to the discussion above, optimizing Eq. (3) is equal to training on $2N$ instances with different weights as illustrated in Eq. (11). For SVM, the dual formulation of Eq. (11) is as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{2N}\sum_{j=1}^{2N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{2N}\alpha_i \\ \mathbf{x}_{N+i} = \mathbf{x}_i \qquad 1 \le i \le N \\ y_i = 1 \qquad 1 \le i \le N \\ y_i = -1 \qquad N+1 \le i \le 2N \\ 0 \le \alpha_i \le u_i \qquad 1 \le i \le N \\ 0 \le \alpha_i \le v_i \qquad N+1 \le i \le 2N, \end{aligned} \tag{12}$$

where $K(\cdot,\cdot)$ is the kernel function of SVM, and the target classifier is $f(\mathbf{x}) = \sum_{i=1}^{2N}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$. The parameter b can be obtained as $b = y_j - \sum_{i=1}^{2N}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$ for any j subject to $1 \le j \le N, 0 < \alpha_j < u_j$ or $N+1 \le j \le 2N, 0 < \alpha_j < v_j$.

According to the discussion above, the algorithm of the SVM implementation of QS-LFC is presented in Algorithm 2.

Algorithm 2. QS-LFC: A SVM Implementation

Data: Instances $\mathbf{X}$, observed labels $\mathbf{Y}$, kernel function $K(\cdot,\cdot)$, Bagging parameter K, weight function $q(x)$
Result: target classifier $f(\mathbf{x})$
begin
1: obtain each w_i^t and $\phi_i^{'}$ using Algorithm 1.
2: calculate u_i and v_i using Eq. (10).
3: solve the optimization of Eq. (12).
4: select a suitable j to obtain b. $b = y_j - \sum_{i=1}^{2N}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$
5: return the target classifier $f(\mathbf{x}) = \sum_{i=1}^{2N}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$

5 Experiments

As mentioned above, the proposed method takes a number of advantages over the existing methods. These merits can be easily verified from Algorithms 1 and 2. What remains to be answered is the effectiveness of QS-LFC. Hence, empirical studies on four artificial data sets and two real crowdsourcing data sets have been conducted. Overall, through these experiments, the following issues will be investigated:

a). The performance of target classifier learned with QS-LFC;
b). The computational time of QS-LFC.
c). The robustness of QS-LFC to low qualified annotators;

It is natural to investigate issue a), because the most important target of LFC is to infer a classifier that has good performance on predicting new instances. The reason of exploring issue b) is that the computational time is also an important indicator for the quality of an algorithm. Last but not least, in many real-world applications, some annotators may simply seek to make money with little effort or they are less skilled. Hence, the quality of labels provided by these annotators will be low. In this regard, the robustness to low qualified annotator is also a very important indicator to evaluate the availability of a method for real-world applications, which gives rise to the issue c).

5.1 Dataset

5.1.1 Original Dataset

We evaluate the effectiveness of QS-LFC on four data sets which are without real crowdsourcing labels and two data sets that have been labeled by multiple real human annotators. Among the former four datasets, Adult, Conect and Mushroom are from UCI Machine Learning Repository, and DNA is from Statlog. Both Conect and DNA have three classes, and their two minor classes are aggregated as one class in this experiment, thus turns Conect and DNA into binary class datasets. All these four data sets are also available in LIBSVM website.[2] Two real crowdsourcing data sets are Twitter Topic and Twitter NER. The description of these two data sets is as follows.

Twitter Topic. The UDI-TwitterCrawl-Aug2012-Tweets,[3] [23], [3] are the tweets collected from 2008 to 2011. The hashtag of a tweet is regarded as the topic of this tweet. Hence, all the tweets with hashtags can be regarded as the labeled instances, and the ground-truth labels of these instances are their hashtags. In our experiment, to produce the feature vectors of these tweets, the natural language toolkit (NLTK[4]) was used. Finally, each tweet was transferred into a 295 dimensional feature vector. The tweets whose hashtags were job, music, Royal Wedding (Prince William and Kate Middleton) or Osama bin Laden being killed were used. The tweets whose hashtags were "job" were regarded as positive instances and the other instances were regarded as negative instances. 1,000 tweets whose hashtags belonged to these four topics were chosen randomly and 5 real annotators (i.e., humans) of our university were invited to label the topics of these 1,000 tweets.

Twitter NER. This Twitter dataset was created by Finin [24] and has been used in many works [19], [21]. It was used for the task of Named Entity Recognition (NER). The goal of this NER task was to identify the names of persons, organizations, locations and similar entities in tweets. The real crowdsourcing labels of this dataset were obtained from the Amazon's Mechanical Turk platform. Each token of all tweets was labeled by at least two annotators, and there were total 269 annotators taking part in this labeling task. We followed the setting of Kajino [19] and simply considered the task as a binary classification problem. If one token was (was not) in a name entity, its label was 1 (-1). There were total 8,107 instances that have ground-truth labels, of which 736 instances were positive instances. As class imbalance problem was not the

2. http://www.csie.ntu.edu.tw/%7ecjlin/libsvmtools/datasets/
3. https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012
4. http://www.nltk.org

TABLE 1
Characteristics of the Data Sets

Data Sets	Dim	Pos	Neg	Total	With Real Multiple labels
Adult	123	11,687	37,155	48,842	NO
Conect	126	44,473	23,084	67,557	NO
Mushroom	112	3,916	4,208	8,124	NO
DNA	180	1,051	949	2,000	NO
Twitter Topic	295	500	500	1,000	YES
Twitter NER	1,471	736	736	1472	YES

focus of this paper, we randomly chose 736 negative instances from all negative instances and got a new data set in which there were 1,472 instances. The features representation for each token was the same with the works of Kajino [19] and Ritter, Clark, Etzioni [25], and we obtained 161,904-dimensional feature vector for each instance. With PCA [26], the dimension of features was reduced to 1,471.

Table 1 summarizes the characteristics of the data sets.

5.1.2 Generate Label

Due to the lack of data sets with real crowdsourcing labels, when conduct experiments, all the existing works of LFC would select one or two kinds of methods to produce the artificial labels of multiple annotators. In this paper, to fully evaluate the effectiveness of the proposed method, three popular methods will be used to produce different kinds of crowdsourcing labels.

Clustering Data. This method has been used in [27], [3] and [16]. In this method, k-means [28] will be used, and all the instances of each dataset will be clustered into five subsets. The clustering procedure will be repeated twice, thus 10 clusters will be obtained. Then, ten annotators will be simulated. For each instance that belongs to cluster t, annotator $t(t = 1, 2, \ldots, 10)$ will correctly label this instance with probability P_i^t and make mistake with probability $1 - P_i^t$. For the instances that does not belong to cluster t, annotator t has no enough knowledge about them. Thus, she/he will correctly label these instances with probability p_i^t and make mistakes with $1 - p_i^t$. In our experiment, $P_i^t \sim U(0.9, 1.0)$ and $p_i^t \sim U(0.5, 0.6)$.

Raykar's Data. This method follows the setting in [15] and [19]. In this method, when annotator t labels instance $\mathbf{x}_i$, she/he will label correctly with probability P_i^t. In our experiment, $P_i^t \sim U(0.7, 0.9)$ for $t = 1$, $P_i^t \sim U(0.6, 0.8)$ for $t = 2 - 3$, $P_i^t \sim U(0.5, 0.7)$ for $t = 4 - 6$ and $P_i^t \sim U(0.4, 0.6)$ for $t = 7 - 10$.

Whitehill's Data. This method is used in [29] and [30]. In this method, if annotator t labels instance $\mathbf{x}_i$, the probability of her/his label being correct is modeled by two terms, i.e., the expertise of annotator t which is denoted as a^t $(a^t \in (-\infty, +\infty))$ and the difficulty of instance $\mathbf{x}_i$ denoted as $\frac{1}{b_i}$ $(\frac{1}{b_i} \in (0, +\infty))$. $a^t = +\infty$ and $-\infty$ correspond to the situations that annotator t always labels the instances correctly and incorrectly, respectively. The parameter $\frac{1}{b_i} = +\infty$ corresponds to the situation that $\mathbf{x}_i$ is too difficult for all annotators to label, and even the most proficient expert may fail by a chance of 50 percent, while $\frac{1}{b_i} = 0$ means that this instance is such easy that even the most obtuse annotator can label it correctly. The probability that annotator t labels $\mathbf{x}_i$ correctly is determined as follow:

$$P_i^t = \frac{1}{1 + e^{-a^t b_i}}. \tag{13}$$

We create 10 annotators, the expertise of annotator t is determined by $a^t \sim N(1, 1)$, and the difficulty of instance $\mathbf{x}_i$ is also determined by $b_i \sim N(1, 1)$ following the setting of [29] and [30].

To simulate the real-world situation, each annotator will not label all instances. Instead, each of them will randomly and independently select 3/10 instances to label. Therefore, different instances will be labeled by different number of annotators, some of them will be labeled by none annotators, and other some instances will be labeled by multiple annotators or even by all annotators. After that, only the instances labeled by three or more annotators will be selected as the training instances, since the existing methods are proposed for the situation where there are multiple labels/annotators for each instance.

5.2 Configuration

For all experiments, 10 times 10-fold cross validations are carried out, i.e., we will repeat training and testing for 100 times for each crowdsourcing data set. In each time 9/10 instances with according labels from multiple annotators are regarded as training set, and the other 1/10 instances with their ground-truth labels are regarded as testing set. We compare QS-LFC with two two-stage methods and two direct methods, these four competing methods are as follows. 1) MV-LFC. This method uses majority voting method to estimate the ground-truth labels, and then learn the target classifier on instances with these estimated labels. When there is no clear majority among the multiple annotators, a fair coin toss is used. 2) M3V-LFC. This method is another two-stage method. it uses M3V to estimate the ground-truth labels, where M3V was proposed in [4]. 3) LC model. It is one direct method, which is proposed in [15]. 4) PC model. This method was proposed in [19]. It is another direct method which uses the personal classifier of each annotator to represent her/his labeling manner.

As it is difficult for the direct methods (e.g., PC model) to use non-linear classifier, for the sake of fairness, all competing methods are applied with linear model. Generally, it is very difficult to optimize the hyper-parameters of any method in LFC, because the ground-truth labels are not available in a learning phase, which leads to the standard cross validation technique cannot be applied. Besides, optimizing the hyper-parameters is not the focus of this paper. Hence, all the hyper-parameters of the proposed method are set as default through all experiments. More specifically, the bagging parameter K is set to 10, the trade-off C is set to 1. The hyper-parameters of PC model, i.e., λ and δ in [19], are both determined by searching in the range of $2^{[-5:5]}$ to get an overall satisfactory performance.

5.3 Comparisons with Existing Methods

There were total 6 original data sets and three methods were applied to simulate multiple annotators on each of them. Hence, there were total 18 different artificial crowdsourcing data sets. Besides, both Twitter Topic and Twitter NER have crowdsourcing labels collected from real human annotators. Thus, there are total 20 different crowdsourcing data

TABLE 2
The Averaged Results of Five Algorithms on 20 Data Sets Are Listed in the Form of 'Mean ± Standard Deviation'

Data Source	Labeling Method	Two-Stage Methods		Direct Methods		QS-LFC
		MV-LFC	M3V-LFC	LC Model	PC Model	
Adult	Clustering data	0.8395±0.0054†	0.8389±0.0055†	**0.8441**±0.0057†	0.8372±0.0060†	0.8317±0.0065
	Raykar's data	0.8355±0.0053	0.8319±0.0062	**0.8455**±0.0049†	0.8308±0.0052	0.8399±0.0051
	Whitehill's data	0.8392±0.0054†	0.8378±0.0058†	**0.8426**±0.0049†	0.8380±0.0061†	0.8267±0.0177
Conect	Clustering data	0.7843±0.0057	0.7844±0.0063	0.7685±0.0128	0.7785±0.0063	**0.7859±0.0056**
	Raykar's data	0.7795±0.0050	0.7824±0.0054	0.7681±0.0171	0.7589±0.0091	**0.7860±0.0052**
	Whitehill's data	0.7791±0.0083‡	**0.7835**±0.0070‡	0.7817±0.0078‡	0.7428±0.0325	0.7815±0.0052
Mushroom	Clustering data	0.9924±0.0050	0.9908±0.0057	0.9885±0.0130	**0.9956**±0.0028‡	0.9953±0.0032
	Raykar's data	0.9855±0.0060	0.9830±0.0073	0.9942±0.0038	0.9903±0.0053	**0.9967±0.0025**
	Whitehill's data	0.9882±0.0112	0.9879±0.0103	0.9956±0.0031	0.9925±0.0073	**0.9965±0.0030**
DNA	Clustering data	0.7446±0.0385	0.7493±0.0324	0.7421±0.0355	0.7712±0.0349	**0.7875±0.0362**
	Raykar's data	0.6961±0.0413	0.6963±0.0437	0.7552±0.0344	0.7129±0.0404	**0.7839±0.0391**
	Whitehill's data	0.7502±0.0653	0.7549±0.0600	0.7891±0.0352	0.7664±0.0646	**0.8125±0.0542**
Twitter Topic	Clustering data	0.6911±0.0474	0.6851±0.0514	0.6772±0.0473	0.7360±0.0468	**0.8019±0.0459**
	Raykar's data	0.6489±0.0537	0.6346±0.0562	0.6357±0.0606	0.6864±0.0532	**0.7661±0.0515**
	Whitehill's data	0.7063±0.0544	0.7039±0.0532	0.6826±0.0477	0.7468±0.0570	**0.7827±0.0608**
	Real label	0.8444±0.0355	0.8444±0.0355	0.8092±0.0394	0.8615±0.0342‡	**0.8616±0.0344**
Twitter NER	Clustering data	0.6723±0.0402	0.6760±0.0384	0.6732±0.0425	0.6407±0.0401	**0.8066±0.0390**
	Raykar's data	0.6340±0.0410	0.6407±0.0419	0.6767±0.0377	0.6037±0.0409	**0.8278±0.0371**
	Whitehill's data	0.6813±0.0587	0.6827±0.0650	0.7264±0.0420	0.6543±0.0569	**0.8452±0.0333**
	Real label	0.7430±0.0396	0.7522±0.0383	0.8254±0.0362	0.6909±0.0399	**0.8458±0.0314**
Friedman-Test		3.500	3.450	3.000	3.450	1.600
Win/Tie/Loss (t-Test)		17/1/2	17/1/2	16/1/3	17/1/2	-

All the results are presented in terms of test accuracy. The best results are in boldface. † and ‡ indicate the entry is superior to and not significantly different from QS-LFC, respectively. The last two rows provide the results of Friedman test and t-test, where 'Win-Tie-Loss' indicates QS-LFC is superior to, not significantly different from, or inferior to the corresponding compared algorithms.

sets. Table 2 reports the performance of five competing methods on all these 20 cases.

Table 2 presents that there is no any method that can achieve the best average test accuracy in all 20 cases. Therefore, t-test and Friedman test have been utilized to check whether the differences (in terms of average test accuracy) between QS-LFC and the other competing methods are statistically significant. Both tests were carried out with 5 percent significance level. t-test allows us to compare QS-LFC with the other algorithms individually, and Friedman test is used to compare all the 5 algorithms. As shown in the last row of Table 2, QS-LFC performed statistically significantly better than the other 4 methods among the 20 data sets in pairwise comparisons. Furthermore, the result of Friedman test in Table 2 is $\chi_F^2 = 20.92$ and $F_F = 6.7278$. With 5 algorithms and 20 data sets, F_F is distributed according to the F distribution with $5-1=4$ and $(5-1)\times(20-1)=76$ degrees of freedom. The critical value of $F(4,76)$ for $\alpha = 0.05$ is 2.49. $F_F > 2.49$, hence the Friedman test result demonstrates that QS-LFC statistically significantly ranked the highest among the 5 algorithms, and a statistically significant difference was confirmed.

Taking a closer look at Table 2, we can see that direct methods performed better than two-stage methods. Specifically, in terms of average test accuracy, LC model and PC model achieved the best performance in 3 and 1 cases respectively, while M3V-LFC and MV-LFC achieved the best performance in 1 and 0 cases respectively. In terms of pairwise comparison, LC model and PC model lost to QS-LFC in 16 and 17 cases respectively, while both M3V-LFC and MV-LFC lost to QS-LFC in 17 cases. In terms of Friedman test, the average rank of LC model and PC model are 3.00 and 3.45 respectively, while the average rank of M3V-LFC and MV-LFC are 3.45 and 3.50 respectively. The poor performances of two-stage methods show that investigating the specially tailored approaches for LFC might be more promising than transforming LFC into the conventional supervised learning problem by first estimating the ground-truth labels.

Table 2 also shows the highly competitive performance of QS-LFC compared with all two-stage methods and direct methods. Specifically, in terms of pairwise comparison, QS-LFC is found to be statistically significantly superior to the other methods in most cases. In terms of average test accuracy, QS-LFC achieved the best performance in 15 of 20 cases. And in terms of average rank, QS-LFC achieved the best performance again.

5.4 Computational Time

The comparisons of the running time of 5 competing algorithms are shown in Fig. 1. These figures show that direct methods needed much more computational time than two-stage methods on all data sets. QS-LFC fell between these two kinds of methods. Specifically, MV-LFC and M3V-LFC cost the least and second least computational time on all data sets except for the 4 cases of Twitter NER. In the four cases of Twitter NER, MV-LFC and M3V-LFC need similar computational time. the reason might be in Twitter NER, the second step (i.e., training target classifier with estimated labels) costed the most of their total computational time.

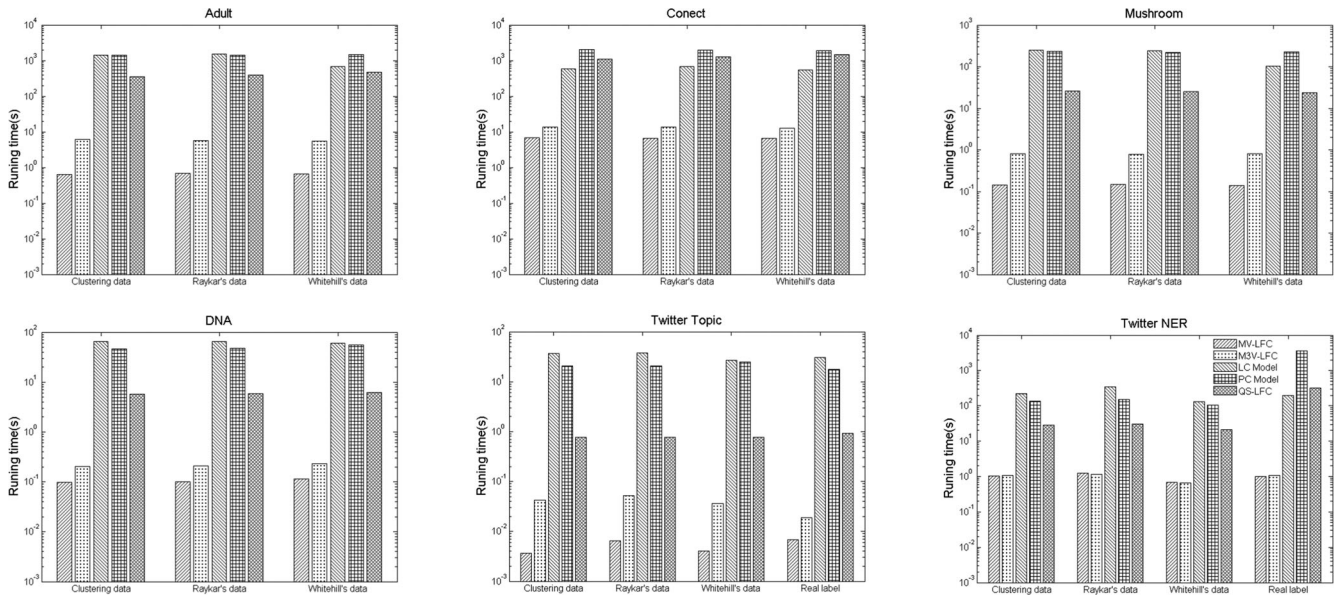


Fig. 1. The comparison of average running time.

But the second step is the same for MV-LFC and M3V-LFC. Hence, the difference of their time cost in the first step is small respect to the total time cost. Overall, the most time consuming algorithm were LC model and PC model. More specifically, LC model and PC model needed the most and second most computational time respectively in most cases, and the computational time of these two methods were similar. The computational time of QS-LFC was the third least in most cases. Besides, the gap between the computational time of QS-LFC and that of the direct methods is obvious.

To investigate how the computing time of each method changes when the number of annotators increases, different numbers of noisy annotators were simulated. These annotators would randomly label each instance. Fig. 2 shows the computational time of all the competing methods with different number of annotators. Fig. 2 shows that PC model was the most time consuming method and it needed much more computational time than the other methods. LC model and QS-LFC were the second-tier time consuming methods. Both MV-LFC and M3V-LFC needed much less time than all the other methods. The changes of computational time of M3V-LFC in Fig. 2d was very interesting. The computational time M3V-LFC spent when the number of annotators was even would be more than that when the number of annotators was odd. This result indicates that the even number of annotators would introduce some trouble in M3V-LFC. Besides, all computational time of MV-LFC, M3V-LFC and LC model kept stable and did not increase with the increasing number of annotators. Hence, these methods may show their efficiencies when the number of annotators is huge.

5.5 Robustness

As mentioned above, since there may be many low quality annotators in real-world applications, robustness is an important metric to measure the availability of LFC methods. In this part, three kinds of noise (random noise, concensus noise and systematic noise) will be utilized to test the robustness of competing methods. Random noise means that noisy annotators label the instances randomly and independently. Concensus noise means that the labels of all noisy annotators are the same thus they will make mistakes on the same instances. Systematic noise, essentially due to the behavioral biases of annotators, means the error made by a noisy annotator shows some regularity, e.g., an annotator is more likely to incorrectly label instances lying in a certain region in the feature space. For random noise and concensus noise, only the results of Twitter Topic data set will be presented and analyzed below, because the same conclusion can be drawn from other data sets. Introducing systematic noise into a real-world data set (e.g., the Twitter Topic data) might leads to problematic conclusions, since systematic noise depends on the features of the data. Hence, empirical studies on systematic noise were conducted on a possibly more general synthetic data set.

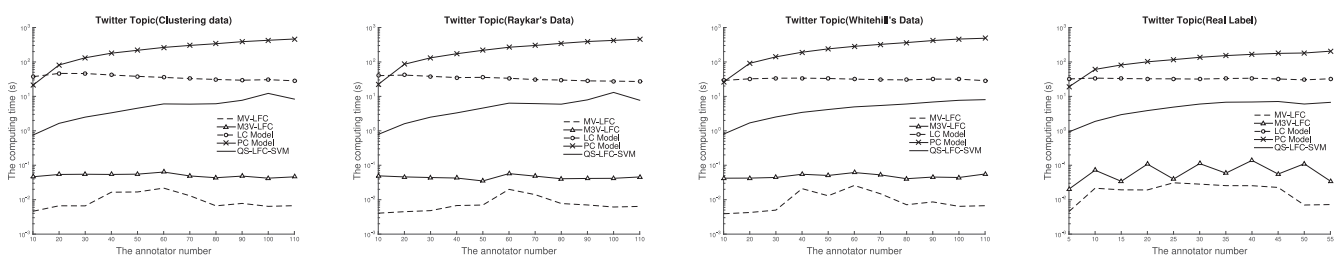


Fig. 2. The relationship between the annotator number and computational time.

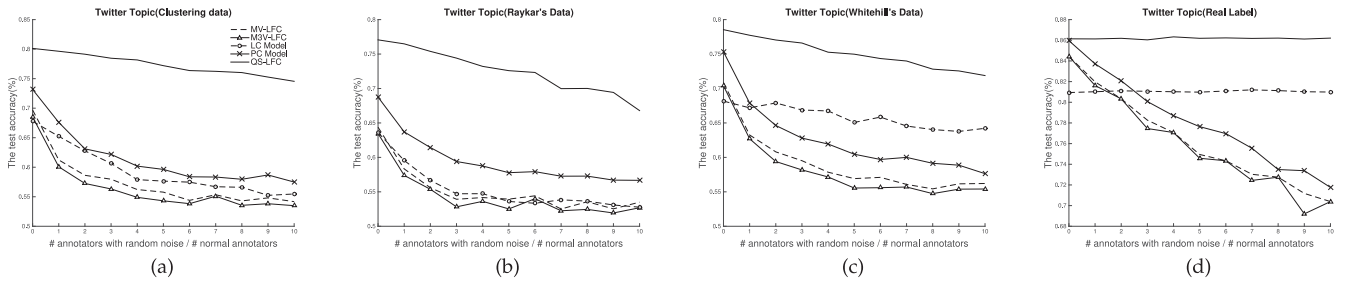


Fig. 3. The robustness comparison of competing methods to random noise.

To produce noise, different numbers of noisy annotators were produced. For the case of random noise, these annotators would randomly label all instances as label 1 or -1. For the case of concensus noise, half of the instances would be randomly selected as the consistent error. And all the noisy annotators would consistently provide wrong labels for these instances and provide correct labels for the other instances.

The synthetic data set for investigating systematic errors consists of 20 subsets/clusters of instances. Each subset contains 50 instances, which were sampled from a normal distribution $N(\boldsymbol{\mu}_i, diag(1,1))$, where $\boldsymbol{\mu}_i$ was randomly sampled from $[-1,1]\times[-1,1]$. For an instance $\mathbf{x}=[x_1,x_2]$, its ground-truth z was 1 if $x_1+x_2>0$, otherwise $z=-1$. That means, the ground-truth classifier is $f(\mathbf{x})=sgn(x_1+x_2)$. There were total three normal annotators. These annotators labeled all the instances with accuracy 0.7.

Three experiments were conducted to investigate the robustness of different algorithms against three types of systematic noisy annotators. The first experiment simulated the situation that noisy annotators were not good at some certain subsets of instances. In this case, each simulated systematic noisy annotator would randomly select 10 subsets as her/his system error, and the instances belonging to the selected subsets will be wrong labeled. The remained instances will be correctly labeled. The other two types of systematic noisy annotators were simulated with a similar approach. That is, the systematic error was explicitly determined by a function of instances. To be specific, a linear function is adopted in case of the second type. The systematic annotator t would label instance $\mathbf{x}$ as positive instance with probability $1/(1+e^{-(x_1+x_2+\zeta_t)})$, where $\zeta_t \sim U(0,2)$ was the bias of this noisy annotator. The third type of systematic noisy annotators were simulated with a nonlinear function. In this case, the systematic annotator t would label instance $\mathbf{x}$ as positive instance with probability $1/(1+e^{-(\varrho_t||\mathbf{x}||^2+\varsigma_t)})$ where both ϱ_t and ς_t were sampled from $U(-1,1)$. In experiments on systematic errors, an algorithm, including both the compared ones and the proposed one, was slightly modified with a preprocessing step to automatically eliminate the noisy annotators that are easy to detect. More specifically, in all the competing methods, if an annotator labeled more than 95 percent instances as positive/negative, her/his all labels would be discarded before training. For the proposed method, if one personal classifier labeled more than 95 percent instances as positive/negative, its weight would be set to 0.5.

Figs. 3, 4, and 5 show the results of all the five competing methods under the different noise conditions. These figures show that QS-LFC performed the best in terms of robustness. Specifically, its test accuracies were the highest in most cases. Besides, the degradation of its test accuracies were much slower than that of the other methods. The reason might be that, in most situation QS-LFC could effectively estimate the reliabilities of both annotators and labels. With these estimated reliabilities, it could identify which annotators were more likely to be poor-quality annotators and which labels were more likely to be wrong. Then it would assign much smaller weights to the less reliable labels.

The results of Fig. 3d, and Fig. 4d are very interesting. They show that the test accuracies of QS-LFC almost did not degrade with the increasing of noisy annotators number. The reason might be that the qualities of normal annotators are much better than the noisy annotators in these cases. Hence, it was easy for QS-LFC model to distinguish the highly qualified annotators. The real human annotators that labeled the Twitter Topic were the students from our university. These annotators can be seen as high qualified annotators since they were well-educated. The statistics of their results showed that the average labeling accuracy of these annotators was about 0.8, which is much better than the noisy annotators.

In most of the ten cases, the PC model kept the second place in terms of robustness. The degeneration of its test accuracy was also competitive. The reason might be that in the PC model the labels of annotators cannot directly affect

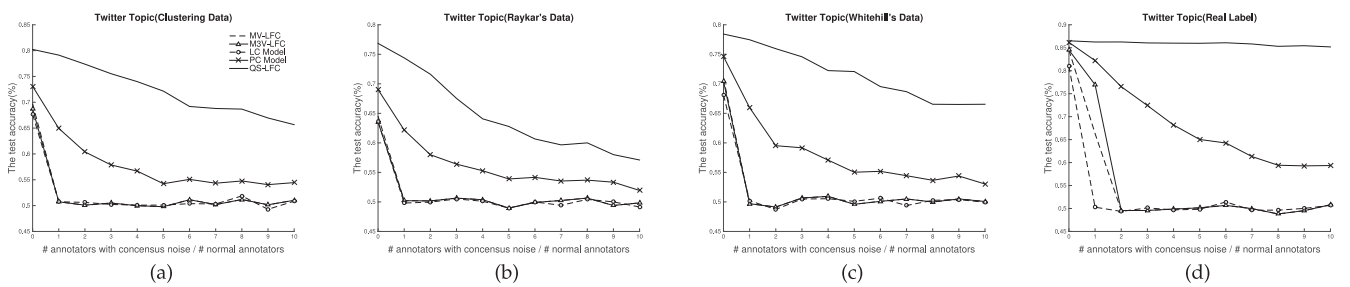


Fig. 4. The robustness comparison of competing methods to concensus noise.

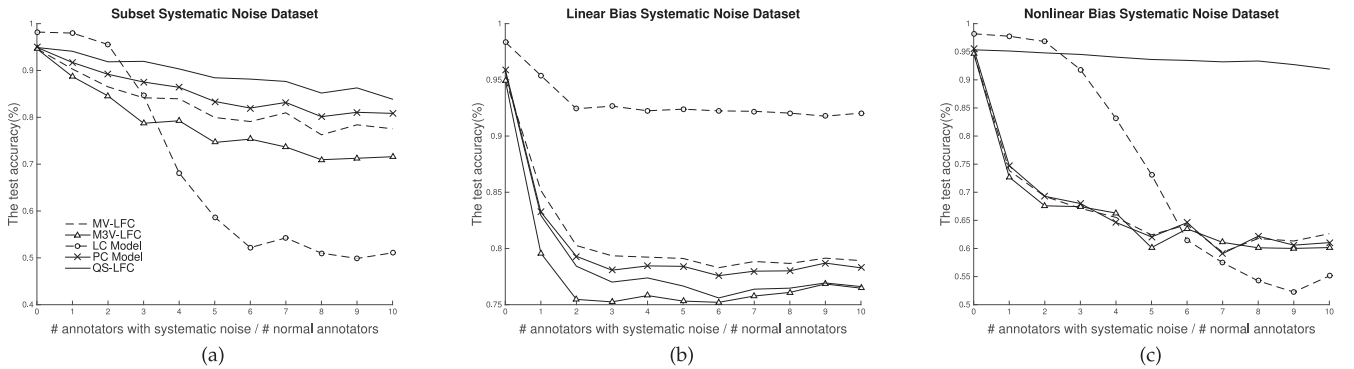


Fig. 5. The robustness comparison of competing methods to systematic noise. (a) Noisy annotators make errors on certain subsets of the data set. (b) Noisy annotators make errors following a linear bias/function. (c) Noisy annotators make errors following a nonlinear bias/function.

the target classifier. Only by personal classifiers, these labels can affect the target classifier. Hence, in the PC model, the effects of these noisy labels would be smoothed by the personal classifiers. The Fig. 5b is the only case that the proposed method did not achieve the best performance. Note that this figure was obtained by using a linear function to simulate the the systematic error. Since the ground-truth classifier is also a linear function (i.e., $f(\mathbf{x}) = sgn(x_1 + x_2)$), the behavioral bias of noisy annotators actually fit the ground-truth, and thus the estimation of annotators reliability r_t may fail to detect the noisy annotators. However, when the difference between the ground-truth and behavioral bias are large, e.g., the latter is inform of a nonlinear function, as shown in Fig. 5c, the proposed QS-LFC still performed more robust than the compared methods.

In most of ten cases, the robustness of two-stage methods were almost the worst. With the number of noisy annotators increasing, the test accuracies of both MV-LFC and M3V-LFC dropped very fast. Hence, the results indicate that both MV-LFC and M3V-LFC could not effectively tell which annotators and labels are more reliable.

6 Conclusion

Conventional supervised learning assumes that there exists an oracle to offer the ground-truth labels. This assumption may not be true for many real-world applications. Hence, how to learn from data with labels collected from multiple non-expert annotators has attracted much attention during the past few years, which is referred to as Learning From Crowds. However, the existing LFC methods suffer from some disadvantages such as needing prior knowledge to select the expertise model to represent the behavior of annotators, involving solving non-convex optimization problems, or having strict limits to the classifier type being used. The analysis about LFC in this paper shows that LFC is essentially a quality-sensitive learning problem. Hence, a framework named QS-LFC from the new perspective of quality-sensitive learning is proposed to overcome these disadvantages. Based on this framework, a SVM implementation of QS-LFC is proposed. Experiments on both artificial and real-world data sets show that, QS-LFC can achieve better generalization performance and more robust than the existing methods.

The success of QS-LFC can serve as an inspiration to help us explore how to learn from crowds. The most difficult problem in LFC is how to handle the labels from multiple annotators when the reliabilities of these labels are different. The proposed method provides a natural way for this problem, i.e., assigning different weights for different labels according to their reliabilities. Besides, the proposed method to estimate the reliabilities of labels also provides us a new thought about how to identify the qualities of both annotators and labels. The result suggests that exploiting the relationship between data features and its ground-truth would be very conducive for identifying the qualities of labels and leading to a better LFC method.

There are a few directions for further improving the QS-LFC. First, just like most of the existing literatures, the proposed method assumes that the annotators in the crowds are independent to each other. Suitable mining and utilizing the relationship between annotators can be helpful to improve the QS-LFC. Besides, a more powerful method may be proposed to estimate the reliability of each label. Thus, the quality-sensitive method may achieve a better performance. In additional to improving QS-LFC, it would also be interesting to utilize the latent relationship between data features and ground-truth labels on other research problems.

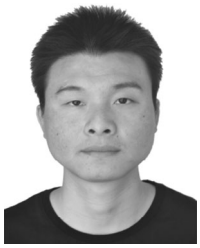
Acknowledgments

This work was supported in part by the Ministry of Science and Technology of China (Grant No. 2017YFC0804002), the National Natural Science Foundation of China (Grant Nos. 61672478 and 61329302), in part by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284), and in part by the Royal Society Newton Advanced Fellowship (Reference No. NA150123).

References

[1] J. Surowiecki and M. P. Silverman, "The wisdom of crowds," *Amer. J. Physics*, vol. 75, no. 2, pp. 190–192, 2007.
[2] 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.
[3] J. Zhong, K. Tang, and Z.-H. Zhou, "Active learning from crowds with unsure option," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 1061–1067.
[4] T. Tian and J. Zhu, "Max-margin majority voting for learning from crowds," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 1621–1629.
[5] L. Wang and Z.-H. Zhou, "Cost-saving effect of crowdsourcing learning," in *Proc. 25th Int. Conf. Artif. Intell.*, 2016, pp. 2111–2117.

[6] P. Donmez, J. G. Carbonell, and J. Schneider, "Efficiently learning the accuracy of labeling sources for selective sampling," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 259–268.
[7] A. Kumar and M. Lease, "Modeling annotator accuracies for supervised learning," in *Proc. Workshop Crowdsourcing Search Data Mining 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 19–22.
[8] V. S. Sheng, "Simple multiple noisy label utilization strategies," in *Proc. IEEE 11th Int. Conf. Data Mining*, 2011, pp. 635–644.
[9] F. K. Khattak and A. Salleb-Aouissi, "Quality control of crowd labeling through expert evaluation," in *Proc. NIPS 2nd Workshop Comput. Social Sci. Wisdom Crowds*, 2011, pp. 27–29.
[10] 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.
[11] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, "The multidimensional wisdom of crowds," in *Proc. Advances Neural Inf. Process. Syst.*, 2010, pp. 2424–2432.
[12] D. Zhou, S. Basu, Y. Mao, and J. C. Platt, "Learning from the wisdom of crowds by minimax entropy," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 2195–2203.
[13] Q. Liu, J. Peng, and A. T. Ihler, "Variational inference for crowdsourcing," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 692–700.
[14] V. C. Raykar, et al., "Supervised learning from multiple experts: Whom to trust when everyone lies a bit," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 889–896.
[15] V. C. Raykar, et al., "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, 2010.
[16] W. Wu, Y. Liu, M. Guo, C. Wang, and X. Liu, "A probabilistic model of active learning with multiple noisy oracles," *Neurocomputing*, vol. 118, pp. 253–262, 2013.
[17] Y. Yan, et al., "Modeling annotator expertise: Learning when everybody knows a bit of something," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 932–939.
[18] W. Bi, L. Wang, J. T. Kwok, and Z. Tu, "Learning to predict from crowdsourced data," in *Proc. 30th Int. Conf. Uncertainty Artif. Intell.*, 2014, pp. 82–91.
[19] H. Kajino, Y. Tsuboi, and H. Kashima, "A convex formulation for learning from crowds," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 73–79.
[20] H. Valizadegan, Q. Nguyen, and M. Hauskrecht, "Learning classification models from multiple experts," *J. Biomed. Informat.*, vol. 46, no. 6, pp. 1125–1135, 2013.
[21] H. Kajino, Y. Tsuboi, and H. Kashima, "Clustering crowds," in *Proc. AAAI Conf. Artif. Intell.*, 2013, pp. 1120–1127.
[22] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[23] R. Li, S. Wang, H. Deng, R. Wang, and K. C.-C. Chang, "Towards social user profiling: Unified and discriminative influence model for inferring home locations," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1023–1031.
[24] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze, "Annotating named entities in Twitter data with crowdsourcing," in *Proc. NAACL HLT Workshop Creating Speech Language Data Amazon's Mech. Turk*, 2010, pp. 80–88.
[25] A. Ritter, S. Clark, Mausam, and O. Etzioni, "Named entity recognition in tweets: An experimental study," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 1524–1534.
[26] I. Jolliffe, *Principal Component Analysis*. Hoboken, NJ, USA: Wiley, 2002.
[27] 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.
[28] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
[29] 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. Advances Neural Inf. Process. Syst.*, 2009, pp. 2035–2043.
[30] L. Zhao, Y. Zhang, and G. Sukthankar, "An active learning approach for jointly estimating worker performance and annotation reliability with crowdsourced data," arXiv:1401.3836, 2014.



Jinhong Zhong (S'14) received the BSc degree in mathematics and the BEng degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2012, and the PhD degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2017. He has also been a short-term visiting researcher in the Department of Computer Science and Engineering, Southern University of Science and Technology. Currently, he is with Huawei Technologies Co., Ltd. as a senior engineer. His research interests include machine learning and its real-world applications. He is a student member of the IEEE.

Peng Yang (S'14) received the BEng degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2012 and the PhD degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2017. Currently, he is with Huawei Technologies Co., Ltd. as a senior engineer. His research interests include optimization and various applications. He is a student member of the IEEE.

Ke Tang (M'07-SM'13) received the BEng degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and the PhD degree from Nanyang Technological University, Singapore, in 2007, respectively. From 2007 to 2017, he spent more than 10 years in the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, first as an associate professor (2007-2011) and later as a professor (2011-2017). Currently, he is a professor in the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, China. His major research interests include evolutionary computation, machine learning, and their real-world applications. He is an associate editor of the *IEEE Transactions on Evolutionary Computation* and served as a member of the editorial boards for a few other journals. He received the Royal Society Newton Advanced Fellowship (2015) and the 2018 IEEE Computational Intelligence Society Outstanding Early Career Award. He is a senior member of the IEEE.