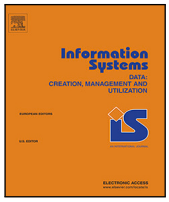
Information Systems 121 (2024) 102321

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Worker similarity-based noise correction for crowdsourcing

Yufei Hu [a], Liangxiao Jiang [a,b,*], Wenjun Zhang [a,b]

[a] School of Computer Science, China University of Geosciences, Wuhan, 430074, Hubei, China
[b] Key Laboratory of Artificial Intelligence, Ministry of Education, Shanghai, 200240, China

ARTICLE INFO

Dataset link: https://github.com/jiangliangxiao/WSNC

Keywords:
Crowdsourcing learning
Noise correction
Worker similarity
Instance similarity

ABSTRACT

Crowdsourcing offers a cost-effective way to obtain multiple noisy labels for each instance by employing multiple crowd workers. Then label integration is used to infer its integrated label. Despite the effectiveness of label integration algorithms, there always remains a certain degree of noise in the integrated labels. Thus noise correction algorithms have been proposed to reduce the impact of noise. However, almost all existing noise correction algorithms only focus on individual workers but ignore the correlations among workers. In this paper, we argue that similar workers have similar annotating skills and tend to be consistent in annotating same or similar instances. Based on this premise, we propose a novel noise correction algorithm called worker similarity-based noise correction (WSNC). At first, WSNC exploits the annotating information of similar workers on similar instances to estimate the quality of each label annotated by each worker on each instance. Then, WSNC re-infers the integrated label of each instance based on the qualities of its multiple noisy labels. Finally, WSNC considers the instance whose re-inferred integrated label differs from its original integrated label as a noise instance and further corrects it. The extensive experiments on a large number of simulated and three real-world crowdsourced datasets verify the effectiveness of WSNC.

1. Introduction

In supervised learning [1], a large number of annotated data is a prerequisite for training learning models. However, obtaining high-quality annotated labels is usually expensive and time-consuming as it requires the participation of domain experts. Moreover, as artificial intelligence receives more and more attention in different fields, many real-world practical applications require millions of annotated data, which is impossible for domain experts to annotate them [2–5].

Fortunately, the advent of crowdsourcing provides a new solution for these practical applications, which makes it possible to obtain large amounts of annotated data in a time-effective and cost-effective manner [6]. The widespread use of crowdsourcing has given rise to some large-scale online crowdsourcing platforms [7], such as Amazon Mechanical Turk (AMT),[1] Crowdflower[2] and Clickworker.[3] We can get extensive annotated data by employing different crowd workers through these crowdsourcing platforms, which has been widely used in ImageNet [8], computer vision [9], natural language processing [10] and other fields.

From these platforms, we can obtain a multiple noise label set for each instance, then label integration is used to infer its integrated label. So far, scholars have proposed numerous label integration algorithms. The simplest but usually effective integration algorithm is Majority Voting (MV) [11]. However, MV considers the quality of each label to be the same. But the collected labels have different qualities due to the imbalance ability of the employed workers, the difficulty of the instances and other factors. In order to relax the assumption of MV, scholars have proposed a large number of relatively complex algorithms, such as Positive Label Frequency Threshold (PLAT) [12], Domain-Weighted Majority Voting (DWMV) [13], Enhanced Bayesian Classifier Combination (EBCC) [14], Multiple Noisy Label Distribution Propagation (MNLDP) [15], Label Augmented and Weighted Majority Voting (LAWMV) [16], Decision Tree-based Weighted Majority Voting (DTWMV) [17] and Attribute Augmentation-based Label Integration (AALI) [18].

Despite the effectiveness of label integration algorithms, there always remains a certain degree of noise in the integrated labels. Here, noise means that the integrated label of an instance is not consistent with its true label. Noise reduces the quality of the trained models to a certain extent, resulting in inaccurate predicted labels. Thus, noise correction algorithms [19–23] have been proposed to reduce the impact of noise in recent years. They filter out the possible noise instances

* Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan, 430074, Hubei, China.
E-mail addresses: yufeihu@cug.edu.cn (Y. Hu), ljiang@cug.edu.cn (L. Jiang), wjzhang@cug.edu.cn (W. Zhang).

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

https://doi.org/10.1016/j.is.2023.102321
Received 15 April 2023; Received in revised form 3 September 2023; Accepted 28 November 2023
Available online 30 November 2023

based on the originally integrated labels, so as to obtain the noise set and clean set, and then correct the instances in the noise set. However, to the best of our knowledge, almost all existing noise correction algorithms only focus on individual workers but ignore the correlations among workers. We argue that it is equally necessary to focus on the correlations among workers. The more similar two workers are, the more similar annotating skills these two workers have, so they tend to annotate consistent labels to same or similar instances.

Based on this premise, we propose a novel noise correction algorithm called worker similarity-based noise correction (WSNC). At first, WSNC exploits the annotating information of similar workers on similar instances to estimate the quality of each label annotated by each worker on each instance. Then, WSNC re-infers the integrated label of each instance based on the qualities of its multiple noisy labels. Subsequently, WSNC considers the instance whose re-inferred integrated label differs from its original integrated label as a noise instance and thus obtains a clean set and a noise set. Finally, WSNC trains two different classifiers on the clean set to further correct the noise instances. In summary, the contributions of this paper are as follows:

- In order to explore the correlations among workers, we propose a method to measure worker similarities, so as to find workers who are similar to a certain worker.
- We propose a worker similarity-based noise correction (WSNC) algorithm. WSNC exploits the annotating information of similar workers on similar instances to estimate the quality of each label annotated by each worker on each instance.
- We conduct a large number of experiments to verify the effectiveness of the proposed WSNC. The results on 34 simulated and three real-world crowdsourced datasets show that WSNC performs much better than all the other competitors.

The rest of this paper is organized as follows. Section 2 reviews the previous work on noise correction. Section 3 presents our WSNC in detail. Section 4 conducts extensive experiments on simulated and real-world datasets. Section 5 concludes this paper.

2. Related work

Although label integration is effective, there still remains a certain degree of noise in the integrated labels no matter which label integration algorithm is used. It is important to handle noise to improve label quality. In crowdsourcing scenarios, noise handling can be divided into two steps, noise filtering and noise correction.

Noise filtering refers to filtering possible noise instances with filtering algorithms. Gamberger et al. [24] proposed a filtering algorithm called Classifier Filter (CF). CF first divides the original dataset into N subsets. Then, CF takes out one subset as the test set and trains a classifier on the remaining $N-1$ subsets to predict instances in the test set. In this way, each instance has a predicted label. Finally, CF considers the instance whose predicted label differs from its original integrated label as a noise instance. In addition, the commonly used filtering algorithms are Majority Vote Filter (MVF) [25], Iterative-Partitioning Filter (IPF) [26] and so on.

Although direct deletion of noise instances can reduce the noise of the integrated labels, it will cause a lot of information waste due to cost and other reasons. Therefore, to make better use of the information of the data, scholars have proposed numerous noise correction algorithms to correct the instances in the noise set, which can be divided into to two categories based on whether they exploit the annotating information.

The first category does not exploit the annotating information, including Polishing Labels (PL), Self-Training Correction (STC), Cluster-based Correction (CC) [19] and so on. PL first divides the dataset into 10 subsets and trains a classifier on each subset. Then, PL makes predictions for each instance using the 10 trained classifiers. Finally, PL takes the predicted label that gets the most votes as its final integrated label. STC first uses a filtering algorithm to generate a clean set and a noise set. Then, STC builds a classifier on the clean set and calculates the probability of each instance belonging to each class in the noise set. If the label with the maximum probability differs from its integrated label, the predicted label is taken as its new label and the instance is added to the clean set. Finally, the process is repeated until a certain proportion of noise instances are corrected. CC first performs clustering algorithms on a dataset. Then, the instances in each cluster are given the same weight. Finally, CC sums the multiple weights for each instance and regards the label with the largest weight as its final integrated label.

The second category exploits the annotating information, including Adaptive Voting Noise Correction (AVNC) [20], Cross-Entropy-based Noise Correction (CENC) [21], Label Distribution Noise Correction (LDNC) [22] and so on. AVNC first uses the integrated labels to calculate the qualities of workers, and then obtains the qualities of the integrated labels and the proportion of noise instances. Then, AVNC uses the trained classifiers to predict the probability of each instance being noise. Finally, AVNC corrects the noise instances using an ensemble learning algorithm. CENC first uses each instance's multiple noisy labels to calculate the entropy. Then, CENC uses the entropy to filter noise instances. Finally, CENC builds classifiers on the clean set by cross-validation and uses the cross-entropy to correct instances in the noise set. LDNC first converts the multiple noisy labels of each instance into label distribution. Then, LDNC uses the difference between the first and second largest label probabilities to filter noise instances. Finally, LDNC builds a classifier on the clean set to relabel all noise instances.

Based on the above descriptions, we find that existing noise correction algorithms do not focus on the influence of workers or only focus on individual workers, but ignore the correlations among workers. So in this paper, we explore the correlations among workers and propose WSNC, which is described in detail in Section 3.

3. Worker similarity-based noise correction

3.1. Motivation

In crowdsourcing scenarios, we usually denote a crowdsourced dataset as $D = \{(\mathbf{x}_i, \mathbf{L}_i)\}_{i=1}^{N}$. Each instance $\mathbf{x}_i$ is annotated by multiple different crowd workers u_r $(r = 1, 2, \ldots, R)$, so $\mathbf{x}_i$ obtains multiple noisy labels $\mathbf{L}_i = \{l_{ir}\}_{r=1}^{R}$. l_{ir} is the label of $\mathbf{x}_i$ annotated by u_r and it takes the value from a fixed set $\{-1, c_1, c_2, \ldots, c_Q\}$. Q is the number of classes and -1 means that u_r does not annotate $\mathbf{x}_i$. Subsequently, each instance $\mathbf{x}_i$ obtains the original integrated label $\hat{y}_i$ using a label integration algorithm. Then, the integrated crowdsourced dataset $\hat{D} = \{(\mathbf{x}_i, \mathbf{L}_i, \hat{y}_i)\}_{i=1}^{N}$ is used for noise correction.

In crowdsourcing platforms, the employed workers $\{u_r\}_{r=1}^{R}$ come from different places, receive different educations and have different knowledge, so they usually have different influence when annotating same crowdsourcing tasks. However, as discussed in Section 2, existing noise correction algorithms do not focus on the influence of workers or only focus on individual workers, but ignore the correlations among workers. We argue that it is equally necessary to focus on the correlations among workers. In general crowdsourcing scenarios, workers annotate crowdsourcing tasks according to their real ability. If two workers are similar, which means that they have similar annotating skills, so when they annotate same or similar instances, they tend to annotate consistent labels. In this paper, we try to exploit the annotating information of similar workers on similar instances to estimate the quality of each label annotated by each worker on each instance. If the label annotated by a worker on an instance is more consistent with the labels annotated by his similar workers on same or similar instances, we consider the quality of the label to be higher. Conversely, the label is highly likely to be annotated incorrectly and of low quality.

Table 1
The multiple noisy labels of $\mathbf{x}_1$'s similar instances.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7
$\mathbf{x}_1$	c_1	c_2	c_3	c_1	c_1	c_1	c_3
$\mathbf{x}_2$	c_1	c_3	c_1	c_1	c_2	c_1	c_2
$\mathbf{x}_3$	c_1	c_1	c_1	c_1	c_1	c_3	c_1

Table 2
The multiple noisy labels annotated by u_8, u_9 and u_{10}.

	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
u_8	c_1	c_2	c_3	c_1	c_2	c_1	c_2
u_9	c_1	c_2	c_1	c_1	c_2	c_1	c_2
u_{10}	c_1	c_1	c_1	c_2	c_1	c_2	c_3

For example, there exists an instance $\mathbf{x}_1$, whose similar instances are $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ ($\mathbf{x}_1$ is most similar to itself). Seven different crowd workers annotate $\mathbf{x}_1$ with $\{l_{11} = c_1, l_{12} = c_2, l_{13} = c_3, l_{14} = c_1, l_{15} = c_1, l_{16} = c_1, l_{17} = c_3\}$. Table 1 shows the multiple noisy labels of $\mathbf{x}_1$'s similar instances. Suppose u_1's similar workers are u_1, u_3 and u_4 (u_1 is most similar to himself). To estimate the quality of l_{11}, we exploit the annotating information of u_1, u_3 and u_4 on $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. As can be seen in Table 1, most of the labels annotated by u_1's similar workers on these instances are consistent with l_{11}, and only the label annotated by u_3 on $\mathbf{x}_1$ differs from l_{11}. It indicates that l_{11} is of high quality. Likewise, suppose u_2's similar workers are u_2, u_5 and u_6 (u_2 is most similar to himself), most of the labels annotated by u_2's similar workers on these instances differ from l_{12}, and only the label annotated by u_5 on $\mathbf{x}_2$ is consistent with l_{12}. It indicates that u_2 is highly likely to annotate incorrectly on $\mathbf{x}_1$, that is, l_{12} is of low quality.

So if worker u_r annotates instance $\mathbf{x}_i$ with l_{ir}, the more consistent the labels annotated by u_r's similar workers on $\mathbf{x}_i$'s similar instances are with l_{ir}, the higher quality l_{ir} has. This motivated us to propose a new noise correction algorithm called worker similarity-based noise correction (WSNC).

3.2. The proposed WSNC

As mentioned above, to estimate the quality of each label l_{ir} which annotated by u_r on $\mathbf{x}_i$, WSNC exploits the annotating information of u_r's similar workers on $\mathbf{x}_i$'s similar instances. There exist two problems: (1) How to find $\mathbf{x}_i$'s similar instances. (2) How to find u_r's similar workers.

First, we solve the problem of how to find $\mathbf{x}_i$'s similar instances. The more consistent the features of two instances are, the more similar the two instances are. So we need to calculate the distances between $\mathbf{x}_i$ and all instances (including $\mathbf{x}_i$), and then find some instances with the shortest distances to $\mathbf{x}_i$, which are the similar instances of $\mathbf{x}_i$. Given a crowdsourced dataset $\hat{D} = \{(\mathbf{x}_i, \mathbf{L}_i, \hat{y}_i)\}_{i=1}^N$, suppose each instance $\mathbf{x}_i$ is described by M features $\{a_{i1}, \dots, a_{im}, \dots, a_{iM}\}$, where a_{im} represents the mth feature value of $\mathbf{x}_i$. In this paper, we use Heterogeneous Euclidean-Overlap Metric (HEOM) [27] as distance metric. HEOM uses euclidean distance for numerical features and overlap metric for nominal features. So HEOM calculates the distance between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ as

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{m=1}^{M} d_m(\mathbf{x}_i, \mathbf{x}_j)^2}, \tag{1}$$

where $d_m(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ on the mth feature, which can be formulated as

$$d_m(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1, & \text{if } a_{im} \text{ or } a_{jm} \text{ is unknown} \\ 1 - I(a_{im} = a_{jm}), & \text{if } a_{im} \text{ is nominal} \\ \dfrac{|a_{im} - a_{jm}|}{max_m - min_m}, & \text{if } a_{im} \text{ is numeric} \end{cases}, \tag{2}$$

where $I(\cdot)$ is an indicator function which returns 1 when the condition in parentheses is true, and returns 0 otherwise. max_m and min_m represent the maximum and minimum values of the mth feature in $\hat{D}$, respectively. HEOM uses $max_m - min_m$ to normalize the difference of the mth feature values between two instances.

After calculating the distances between $\mathbf{x}_i$ and all instances (including $\mathbf{x}_i$), we sort the calculated distances in ascending order and then take the top K instances $\{\mathbf{x}_{ik}\}_{k=1}^K$, which are the K similar instances of $\mathbf{x}_i$.

Second, we solve the problem of how to find u_r's similar workers. If two workers annotate consistent labels on most instances, they are likely to be similar workers. On the contrary, if two workers annotate inconsistent labels on most instances, it indicates that the two workers are very different. For example, as shown in Table 2, there exist three workers: u_8, u_9 and u_{10}, they all annotate seven instances: $\mathbf{x}_4$, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$, $\mathbf{x}_8$, $\mathbf{x}_9$, $\mathbf{x}_{10}$. For workers u_8 and u_9, they annotate consistent labels on six instances except $\mathbf{x}_6$. But for workers u_8 and u_{10}, they only annotate consistent labels on $\mathbf{x}_4$. It is obvious that the correlation between u_8 and u_9 is higher than that between u_8 and u_{10}. That is to say, u_8 and u_9 are more likely to be similar workers.

Based on above discussions, we propose a method to measure worker similarity between u_r and $u_{r'}$ as

$$s(u_r, u_{r'}) = \begin{cases} 0, & \text{if } \sum_{i=1}^{N} I(l_{ir} \neq -1 \wedge l_{ir'} \neq -1) < N/10 \\ \dfrac{\sum_{i=1}^{N} I(l_{ir} \neq -1 \wedge l_{ir'} \neq -1) \cdot I(l_{ir} = l_{ir'})}{\sum_{i=1}^{N} I(l_{ir} \neq -1 \wedge l_{ir'} \neq -1)}, & \text{otherwise} \end{cases}, \tag{3}$$

where $I(l_{ir} \neq -1 \wedge l_{ir'} \neq -1)$ denotes that u_r and $u_{r'}$ both annotate instance $\mathbf{x}_i$, and $I(l_{ir} = l_{ir'})$ denotes that u_r and $u_{r'}$ annotate consistent labels on $\mathbf{x}_i$. In crowdsourcing scenarios, workers usually do not annotate all instances. Some workers annotate only a few instances. When there are few instances that two workers both annotate, it is inaccurate to calculate the similarity between two workers. So when the number of instances annotated by both two workers is less than $N/10$, we consider the similarity between two workers is 0. The more instances that two workers annotate consistent labels, the more similar their annotating skills are, and the more likely they are similar workers.

After calculating the similarities between u_r and all workers (including u_r), we sort the calculated similarities in descending order and then take the top T workers $\{u_{rt}\}_{t=1}^T$, which are the similar workers of u_r.

Subsequently, we exploit the annotating information of workers $\{u_{rt}\}_{t=1}^T$ on instances $\{\mathbf{x}_{ik}\}_{k=1}^K$ to estimate the quality of the label l_{ir}, and the formula is

$$w_{ir} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T} I(l_{ikrt} = l_{ir}) \cdot s(u_r, u_{rt})}{\sum_{k=1}^{K} \sum_{t=1}^{T} I(l_{ikrt} \neq -1) \cdot s(u_r, u_{rt})}, \tag{4}$$

where l_{ikrt} denotes the label annotated by the tth similar worker of u_r on the kth similar instance of $\mathbf{x}_i$. The more consistent the labels annotated by $\{u_{rt}\}_{t=1}^T$ on $\{\mathbf{x}_{ik}\}_{k=1}^K$ are with l_{ir}, the higher the quality of l_{ir}. Particularly, we add worker similarity $s(u_r, u_{rt})$ in the calculation, and more similar worker should be assigned higher weight.

Based on the calculated quality of each label annotated by each worker on each instance, we can re-infer the integrated label of each instance by

$$\hat{y}_i' = \underset{c_q \in \{c_1, c_2, \dots, c_Q\}}{\arg\max} \sum_{r=1}^{R} I(l_{ir} = c_q) \cdot w_{ir}. \tag{5}$$

If the re-inferred integrated label $\hat{y}_i'$ of $\mathbf{x}_i$ differs from its original integrated label $\hat{y}_i$, we consider $\mathbf{x}_i$ as a noise instance and add it into the noise set, otherwise add it into the clean set. Thus we obtain a clean set and a noise set and then further correct noise instances. Because the instances in the clean set are of high quality, we use these instances to train two different classifiers. Then we use the trained classifiers to predict each instance in the noise set, if the predicted labels are same but different from its re-inferred integrated label, we correct this instance and consider the predicted label as its final integrated label. Otherwise, correct the instance with its re-inferred integrated label. Algorithm 1 shows the detailed learning process of WSNC.

Algorithm 1 WSNC

Input: $\hat{D} = \{(\mathbf{x}_i, \mathbf{L}_i, \hat{y}_i)\}_{i=1}^{N}$ - an integrated crowdsourced dataset; K - the number of similar instances; T - the number of similar workers
Output: $\tilde{D}$ - the corrected dataset

1: **for** $i = 1$ to N **do**
2: **for** $j = 1$ to N **do**
3: Calculate the distance $d(\mathbf{x}_i, \mathbf{x}_j)$ by Equations (1) and (2)
4: **end for**
5: Find K similar instances $\{\mathbf{x}_{ik}\}_{k=1}^{K}$ of $\mathbf{x}_i$
6: **for** $r = 1$ to R **do**
7: **for** $r' = 1$ to R **do**
8: Calculate the similarity $s(u_r, u_{r'})$ by Equation (3)
9: **end for**
10: Find T similar workers $\{u_{rt}\}_{t=1}^{T}$ of u_r
11: Calculate l_{ir}'s quality w_{ir} by Equation (4)
12: **end for**
13: Re-infer the integrated label $\hat{y}'_i$ of $\mathbf{x}_i$ by Equation (5)
14: **if** $(\hat{y}'_i \neq \hat{y}_i)$ **then**
15: Add the instance $\mathbf{x}_i$ into the noise set $\hat{D}_n$
16: **else**
17: Add the instance $\mathbf{x}_i$ into the clean set $\hat{D}_c$
18: **end if**
19: **end for**
20: Build two different classifiers f_1 and f_2 on $\hat{D}_c$
21: **for** $i = 1$ to $|\hat{D}_n|$ **do**
22: **if** $(f_1(\mathbf{x}_i) == f_2(\mathbf{x}_i))$ and $(f_1(\mathbf{x}_i) \neq \hat{y}'_i)$ **then**
23: Correct integrated label of $\mathbf{x}_i$ with $f_1(\mathbf{x}_i)$
24: **else**
25: Correct integrated label of $\mathbf{x}_i$ with $\hat{y}'_i$
26: **end if**
27: **end for**
28: Update the noise set $\hat{D}_n$ to $\tilde{D}_n$
29: $\tilde{D} = \hat{D}_c + \tilde{D}_n$
30: Return $\tilde{D}$ as the corrected dataset

3.3. Time and space complexity analysis

As shown in Algorithm 1, WSNC can be divided into two main steps: noise filtering and noise correction.

The noise filtering step is in lines 1–19, which including: (1) Lines 2–5 find K similar instances of $\mathbf{x}_i$, the time complexity of calculating distances is $O(MN)$, the time complexity of sorting the calculated distances is $O(N \log N)$, so the time complexity of lines 2–5 is $O(MN + N \log N)$. (2) Lines 7–10 find T similar workers of u_r, the time complexity of calculating similarities is $O(RN)$, the time complexity of sorting the calculated similarities is $O(R \log R)$, so the time complexity of lines 7–10 is $O(RN + R \log R)$. (3) Line 11 calculates the quality of each label l_{ir}, the time complexity is $O(KT)$. (4) Lines 13–18 re-infer the integrated label of $\mathbf{x}_i$ and add $\mathbf{x}_i$ into the corresponding dataset, the time complexity is $O(RQ+1)$. Each instance follows the above steps, so the time complexity of noise filtering is $O(N(MN+N \log N+R(RN+R \log R + KT) + RQ + 1))$, namely $O(N^2(M+ \log N + R^2) + N(R^2 \log R + RKT + RQ + 1))$.

The noise correction step is in lines 20–29, which including: (1) Line 20 builds two different classifiers f_1 and f_2 on the clean set. (2) Lines 21–27 predict each instance in the noise set and correct it via the consensus voting. (3) Lines 28–29 update the noise set and merge it with the clean set. Assuming that the training time complexities of f_1 and f_2 are $O(f_1)$ and $O(f_2)$, the classification time complexities of f_1 and f_2 are $O(t_1)$ and $O(t_2)$. So the total time complexity of noise correction is $O(f_1 + f_2 + |\hat{D}_n|(t_1 + t_2) + N)$.

According to the above analysis, the time complexity of WSNC is $O(N^2(M+ \log N+R^2)+N(R^2 \log R+RKT+RQ+2)+f_1+f_2+|\hat{D}_n|(t_1+t_2))$. If we only take the highest order terms, the time complexity of WSNC is $O(N^2(M+ \log N + R^2))$.

The space complexity of the algorithm can be summarized as follows: (1) The input dataset denoted as $\hat{D}$, occupies $O(N(M + R + 2))$ space as it contains N instances. Because each instance is described by M features, requiring $O(M)$ space for storage. The multiple noisy label set of each instance requires $O(R)$ space for storage, where R represents the number of workers. The true label and integrated label of each instance require $O(2)$ space for storage. (2) The output dataset denoted as $\tilde{D}$, also requires $O(N(M+R+2))$ space as it also contains N instances. (3) There are certain computations that may require temporary variables to store intermediate results. The space complexity of these steps can be estimated as $O(K+T)$, where $O(K)$ represents the storage space required for storing K similar instances, $O(T)$ represents the storage space required for storing T similar workers. Thus these temporary variables do not significantly impact the overall space complexity.

According to the above analysis, the overall space complexity of the algorithm can be approximated as $O(N(M + R))$.

Table 3
The description of the whole 34 simulated datasets.

Dataset	#Instances	#Features	#Classes	Missing values	Feature type
anneal	898	38	6	yes	hybrid
audiology	226	69	24	yes	nominal
autos	205	25	7	yes	hybrid
balance-scale	625	4	3	no	numeric
biodeg	1055	41	2	no	numeric
breast-cancer	286	9	2	yes	nominal
breast-w	699	9	2	yes	numeric
car	1728	6	4	no	nominal
credit-a	690	15	2	yes	hybrid
credit-g	1000	20	2	no	hybrid
diabetes	768	8	2	no	numeric
heart-c	303	13	5	yes	hybrid
heart-h	294	13	5	yes	hybrid
heart-statlog	270	13	2	no	numeric
hepatitis	155	19	2	yes	hybrid
horse-colic	368	22	2	yes	hybrid
hypothyroid	3772	29	4	yes	hybrid
ionosphere	351	34	2	no	numeric
iris	150	4	3	no	numeric
kr-vs-kp	3196	36	2	no	nominal
labor	57	16	2	yes	hybrid
letter	20 000	16	26	no	numeric
lymph	148	18	4	no	hybrid
mushroom	8124	22	2	yes	nominal
segment	2310	19	7	no	numeric
sick	3772	29	2	yes	hybrid
sonar	208	60	2	no	numeric
spambase	4601	57	2	no	numeric
tic-tac-toe	958	9	2	no	nominal
vehicle	846	18	4	no	numeric
vote	435	16	2	yes	nominal
vowel	990	13	11	no	hybrid
waveform	5000	40	3	no	numeric
zoo	101	17	7	no	hybrid

4. Experiments and results

In this section, we conduct a large number of experiments to verify the effectiveness of WSNC. We compare WSNC with the existing six noise correction algorithms, including PL, CC, STC, AVNC, CENC and LDNC on simulated and real-world datasets in terms of the noise ratio. Here, the noise ratio represents the percentage of instances whose integrated labels differ from their true labels.

We implement WSNC, CENC and LDNC on the Crowd Environment and its Knowledge Analysis (CEKA) [28] platform. PL, CC, STC and AVNC already exist on the CEKA platform, so we can use them directly. In our experiments, we use a simple but effective label integration algorithm MV [11] to obtain each instance's original integrated label. The

Table 4
The noise ratio (%) comparisons on the uniform distribution.

Dataset	PL	STC	CC	AVNC	CENC	LDNC	WSNC
anneal	19.82	4.48	13.15	9.4	7.81	3.05	2.36
audiology	30.93	13.72	22.57	24.03	24.73	16.19	16.19
autos	44.05	3.46	22.15	16.2	15.56	4.63	12.34
balance-scale	22.13	11.01	12.77	16.24	15.6	8.43	7.01
biodeg	30.13	16.32	13.42	12.45	13.05	10.24	9.26
breast-cancer	28.15	21.68	22.38	24.06	24.3	13.04	12.94
breast-w	3.98	8.67	3.33	4.48	4.61	6.87	5.26
car	20.17	10.4	20.12	7.65	7.88	4.7	5.39
credit-a	15.1	14.06	12.71	12.48	12.61	9.38	9.46
credit-g	25.39	20.29	20.54	21.71	21.27	12.04	11.47
diabetes	21.91	20.78	20.33	21.88	21.21	13.11	12.58
heart-c	16.67	16.93	16.6	18.25	17.92	13.66	12.28
heart-h	33.91	15.85	17.55	18.37	17.89	12.01	10.95
heart-statlog	16	17.59	17.96	17.41	16.81	12.11	12.22
hepatitis	18.06	14.65	13.94	14.9	14.58	12.52	10.97
horse-colic	14.86	15.24	17.17	14.57	14.13	9.95	11.2
hypothyroid	1.09	6.94	7.23	0.46	0.42	1.27	3.09
ionosphere	12.36	11.99	8.32	8.83	9.32	8.72	8.21
iris	20.8	5.07	3.47	5.4	4.47	2.47	1.6
kr-vs-kp	22.96	5.96	11.26	1.26	1.32	5.91	7.34
labor	30.35	20.53	14.39	19.65	19.82	14.04	12.28
letter	10.22	9.03	3.25	8.51	7.41	0.47	0.02
lymph	22.36	14.73	15.14	19.12	18.38	11.96	10.07
mushroom	5.49	3.79	0.74	0.01	0.02	5.1	4.37
segment	4.1	2.26	3.17	2.71	1.99	0.3	0.15
sick	1.81	5.44	4.57	1.44	1.48	5.85	5.8
sonar	29.9	21.3	18.41	18.03	20.53	13.22	9.28
spambase	31.99	12.12	8.84	6.5	6.81	8.17	7.47
tic-tac-toe	34.76	16.02	15.41	13.31	12.6	10.44	8.81
vehicle	20.5	19.52	18.66	17.47	16.73	4.7	2.13
vote	5.33	9.84	8.34	3.98	3.93	7.47	8.69
vowel	29.82	13.02	3.07	8.64	9.98	1.31	0.09
waveform	12.14	15.93	14.24	12.6	12.47	5.52	2.48
zoo	19.6	9.31	4.06	9.7	9.01	5.64	5.45
Average	19.91	12.58	12.63	12.11	11.96	8.07	7.62

experimental settings and parameters of the competitors are consistent with the original papers.

In WSNC, we use NaiveBayes (NB) [29] and C4.5 [30] as the two classifiers. For simplicity, we set the number of similar instances K and the number of similar workers T to be the same and empirically set them to 5.

4.1. Experiments on simulated datasets

In this subsection, extensive experiments are conducted on the whole 34 simulated datasets published on the CEKA platform. Table 3 shows the detailed description of these datasets.

To simulate the process of annotating, we need to hide the true label of each instance first. Since we exploit the annotating information of similar workers on similar instances, it is inaccurate to find similar workers of a certain worker when the total number of workers is too small. So we simulate 11 workers to annotate each instance and the annotating quality of each worker is denoted as $p_r(r = 1, 2, \ldots, 11)$, which is randomly generated from a uniform distribution in the interval $[0.55, 0.75]$.

So each instance obtains 11 noisy labels annotated by different workers, then we use a simple but effective label integration algorithm MV to obtain the original integrated label of each instance. Thus, we get the integrated dataset for noise correction. And then we use WSNC and the other six noise correction algorithms to filter possible noise instances and further correct them. In this way, we can obtain the noise ratio of each algorithm applied on each dataset. As we all know, the simulated annotating process has a certain randomness, so we repeat each experiment 10 times independently and show the average results.

Tables 4 and 5 show the detailed comparison results of the noise ratio on the whole 34 simulated datasets. The last row in Table 4 shows the arithmetic average value of each algorithm applied on the whole datasets, representing the overall performance of each algorithm. To further compare each pair of algorithms, we also perform the Wilcoxon tests [31,32], which is shown in Table 5. The symbol • indicates that the algorithm in the row is significantly superior to the algorithm in the corresponding column, and the meaning of the symbol ∘ is the opposite of the symbol •. The significance level of the lower diagonal is $\alpha = 0.05$, and the significance level of the upper diagonal is $\alpha = 0.1$. From the results shown in Tables 4 and 5, we can conclude as follows:

Table 5
The noise ratio (%) comparisons on the uniform distribution using Wilcoxon tests.

	PL	STC	CC	AVNC	CENC	LDNC	WSNC
PL	–	∘	∘	∘	∘	∘	∘
STC	•	–			∘	∘	∘
CC	•		–			∘	∘
AVNC	•			–	∘	∘	∘
CENC	•				–	∘	∘
LDNC	•	•	•	•	•	–	∘
WSNC	•	•	•	•	•	•	–

1. From Table 4, we can see that the average noise ratio of PL (19.91%), STC (12.58%) and CC (12.63%) are considerably higher than AVNC (12.11%), CENC (11.96%), LDNC (8.07%) and WSNC (7.62%). It indicates that the algorithms exploiting the annotating information have better performance than those without exploiting the annotating information.
2. From Table 4, the average noise ratio of WSNC on all datasets is 7.62%, which is considerably lower than those of PL (19.91%), STC (12.58%), CC (12.63%), AVNC (12.11%), CENC (11.96%) and LDNC (8.07%). It indicates that WSNC has the best performance among other competitors.
3. From Table 5, the results of the Wilcoxon tests indicates that WSNC is significantly superior to PL, STC, CC, AVNC, CENC

Table 6
The noise ratio (%) comparisons on the Gaussian distribution.

Dataset	PL	STC	CC	AVNC	CENC	LDNC	WSNC
anneal	19.73	4.58	13.6	9.25	8.16	2.78	2.1
audiology	32.12	13.94	23.81	24.65	25.66	15.53	16.24
autos	43.76	4.2	24.05	16.54	17.51	4.78	12.63
balance-scale	21.78	9.62	12.42	15.86	15.73	7.38	6.56
biodeg	29.05	15.92	12.97	12.61	12.69	9.93	8.83
breast-cancer	27.13	22.48	21.89	23.99	24.51	12.87	12.73
breast-w	4.42	9.63	3.51	4.25	4.25	7.02	5.55
car	20.27	10.97	19.99	7.8	7.86	5.06	5.53
credit-a	16.78	14.81	13.01	12.36	11.96	8.78	9.33
credit-g	25.46	20.41	20.19	21.48	21.59	12.47	11.39
diabetes	21.34	19.96	19.51	22.23	20.72	12.1	11.78
heart-c	16.24	15.48	16.07	16.96	17.72	13.7	11.62
heart-h	33.71	16.26	16.63	17.82	17.21	11.02	10.44
heart-statlog	17.44	19.52	19	17.56	17.44	12.67	12.07
hepatitis	17.03	14.84	14.13	17.23	15.68	11.94	11.16
horse-colic	15	14.4	16.55	14.35	14.05	8.94	10.27
hypothyroid	1.16	7.14	7.21	0.46	0.44	1.35	3.33
ionosphere	12.76	11.48	8.29	9.29	9.4	7.46	7.29
iris	20.4	4.07	3.33	5.13	4.13	1.93	1.4
kr-vs-kp	21.76	5.15	10.74	1.09	1.08	5.29	6.22
labor	32.81	21.93	13.86	21.93	22.81	14.21	15.26
letter	9.68	9.24	3.27	9.22	7.64	0.64	0.03
lymph	20.47	14.73	15	19.59	19.53	12.64	10.47
mushroom	5.24	4.02	0.88	0.03	0.04	5.35	4.95
segment	4.37	2.38	3.13	2.66	2.05	0.29	0.12
sick	1.8	5.14	4.39	1.41	1.46	5.28	5.4
sonar	35.14	20.24	16.92	18.37	21.11	15.63	9.81
spambase	31.83	11.87	8.75	6.54	6.6	7.66	6.88
tic-tac-toe	34.66	16	15.73	12.49	12.97	10.05	8.2
vehicle	20.17	19.33	18.65	17.8	16.65	4.67	1.93
vote	4.71	8.41	8.55	4.23	3.95	6.8	7.13
vowel	29.06	12.58	2.91	8.95	10.6	1.78	0.09
waveform	12.29	17	14.21	12.95	12.86	6.34	2.93
zoo	17.92	9.11	4.65	10.4	9.9	6.24	6.63
Average	19.93	12.55	12.58	12.28	12.23	7.96	7.54

Table 7
The noise ratio (%) comparisons on the Gaussian distribution using Wilcoxon tests.

	PL	STC	CC	AVNC	CENC	LDNC	WSNC
PL	–	∘	∘	∘	∘	∘	∘
STC	•	–				∘	∘
CC	•		–			∘	∘
AVNC	•			–		∘	∘
CENC	•				–	∘	∘
LDNC	•	•	•	•	•	–	∘
WSNC	•	•	•	•	•	•	–

and LDNC. The noise ratio can be significantly reduced by exploiting the annotating information of similar workers on similar instances.

In crowdsourcing scenarios, the quality distributions of workers are diversified due to the differences in the education they received. Therefore, we design another set of simulated experiments to verify the effectiveness of our proposed WSNC. In the new experiments, the annotating qualities of simulated workers are generated from a Gaussian distribution with $N(0.65, 0.1^2)$. The detailed comparison results are shown in Tables 6 and 7. From these results, we can draw the same conclusions that WSNC performs much better than PL, STC, CC, AVNC, CENC and LDNC. So whether the simulated worker qualities belong to a uniform or Gaussian distribution, WSNC significantly outperforms all the other algorithms.

4.2. Experiments on real-world datasets

In this subsection, we aim to verify the effectiveness of WSNC in real-world crowdsourcing scenarios, so we design experiments on three real-world crowdsourced datasets, which are collected from AMT platform. The detailed descriptions are shown in Table 8.

Table 8
The description of three real-world datasets.

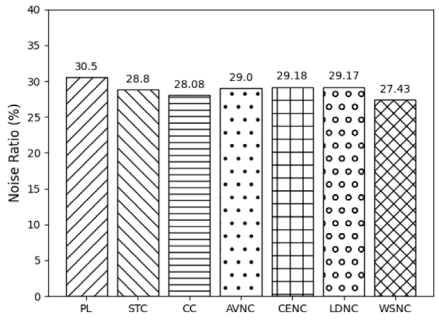
Dataset	#Instances	#Features	#Classes	#Workers	#Labels
Income	600	10	2	67	6000
Labelme	1000	512	8	59	2547
Music_genre	700	124	10	44	2946

The "Income" dataset is a binary classification dataset, which contains 600 instances and each instance is described by 10 features. 67 different crowd workers are employed to annotate these instances. In total, the "Income" dataset gets 6000 labels.

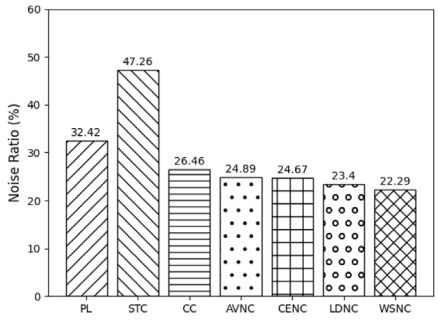
The "Labelme" dataset is a multi-class classification dataset, which contains 1000 instances and each instance is described by 512 features. The platform employs 59 different crowd workers to annotate these instances. In total, the "Labelme" dataset gets 2547 labels finally.

The "Music_genre" dataset is also a multi-class classification dataset, which contains 700 instances and each instance is described by 124 features. In total, the dataset gets 2946 labels annotated by 44 different crowd workers.

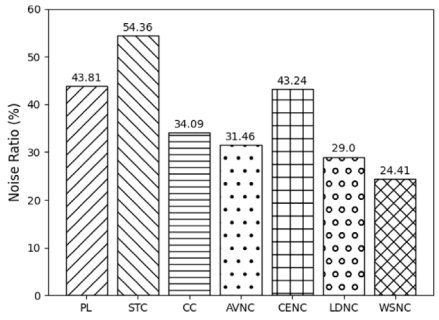
The detailed comparison results on three datasets in terms of the noise ratio are shown in Fig. 1. On the "Income" dataset, the noise ratio of WSNC is 27.43%, which is significantly lower than those of PL (30.5%), STC (28.8%), CC (28.08%), AVNC (29.0%), CENC (29.18%) and LDNC (29.17%). On the "Labelme" dataset, the noise ratio of WSNC is 22.29%, which is significantly lower than those of PL (32.42%), STC (47.26%), CC (26.46%), AVNC (24.89%), CENC (24.67%) and LDNC (23.4%). On the "Music_genre" dataset, the noise ratio of WSNC is 24.41%, which is significantly lower than those of PL (43.81%), STC (54.36%), CC (34.09%), AVNC (31.46%), CENC (43.24%) and LDNC (29.0%). As can be seen from these results, WSNC outperforms all the other competitors on three real-world crowdsourced



(a) Income

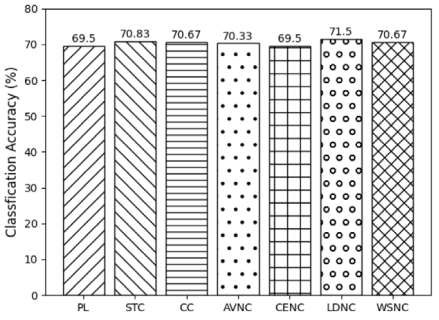


(b) Labelme

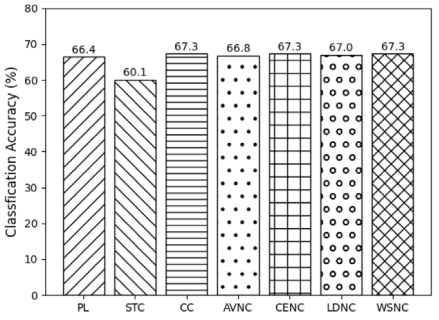


(c) Music_genre

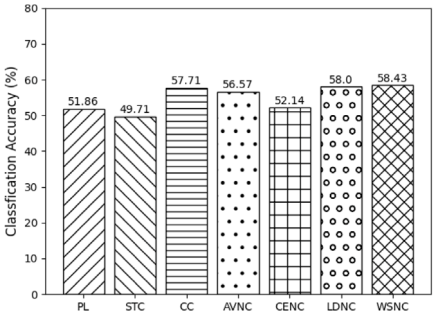
Fig. 1. The noise ratio (%) comparisons on three real-world datasets.



(a) Income



(b) Labelme



(c) Music_genre

Fig. 2. The classification accuracy (%) comparisons on three real-world datasets.

datasets. So we can conclude that WSNC has better performance on both simulated and real-world datasets.

To further validate the performance of WSNC, we conduct experiments on three real-world datasets in terms of the classification accuracy of the used target classifier. The detailed steps are as follows. Firstly, we use each noise correction algorithm to obtain the corrected dataset. Secondly, we divide the corrected dataset into training set and test set using ten-fold cross-validation. Thirdly, we build a classifier on the training set with the corrected labels. In this paper, we use NB as the target classifier. Finally, we use the built classifier to predict the instances in the test set and evaluate its performance with the true labels. We repeat each experiment 10 times and show the average results in Fig. 2. On the "Income" dataset, the classification accuracy of WSNC is 70.67%, which is higher than those of PL (69.5%), AVNC (70.33%) and CENC (69.5%). On the "Labelme" dataset, the classification accuracy of WSNC is 67.3%, which is higher than those of PL (66.4%), STC (60.1%), AVNC (66.8%) and LDNC (67.0%). On the "Music_genre" dataset, the classification accuracy of WSNC is 58.43%, which is higher than those of PL (51.86%), STC (49.71%), CC (57.71%), AVNC (56.57%), CENC (52.14%) and LDNC (58.0%). As can be seen from these results, WSNC also has good performance on three real-world datasets in terms of the classification accuracy of the used target classifier.

4.3. Parameter sensitivity analysis

In order to explore the impact of the number of similar instances K and the number of similar workers T in WSNC, we design our experiments to analyze the sensitivity of the parameters in this subsection. For simplicity, we set K and T to be the same value and conduct experiments on three real-world datasets. Fig. 3 shows the noise ratio comparisons when K and T vary from 3 to 9. From these results, we find that WSNC can obtain relatively low noise ratios when K and T vary from 3 to 9 on these datasets, which indicates that WSNC is not

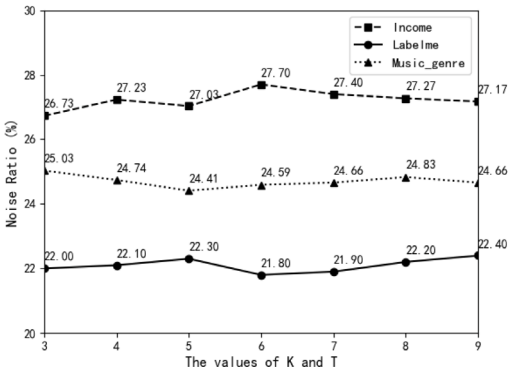


Fig. 3. The noise ratio (%) comparisons for WSNC when K and T vary from 3 to 9 on three real-world datasets.

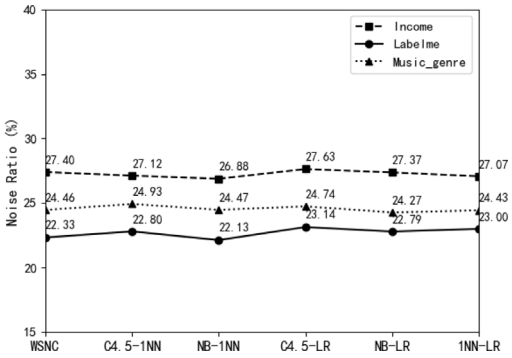


Fig. 4. The noise ratio (%) comparisons for WSNC versus its 5 variants on three real-world datasets.

very sensitive to the values of K and T. Therefore, we set the values of K and T to 5 in our experiments.

In addition, we use NB and C4.5 as the classifiers in our experiments. Now we analyze the generality of WSNC across different classification methods. We choose four commonly used classification methods, including Naive Bayes (NB), C4.5, 1-Nearest Neighbor (1NN) and Logistic Regression (LR). Then we combine the four algorithms in pairs to get six combinations. We name the other combinations with their algorithm names, which are C4.5-1NN, NB-1NN, C4.5-LR, NB-LR and 1NN-LR. We compare WSNC with its 5 variants in terms of the noise ratio. Fig. 4 shows the comparisons results for WSNC versus its 5 variants. From these results, we find that regardless of which two classification methods are chosen, satisfactory results can be achieved. Therefore, we conclude that WSNC is not sensitive to the choice of classification methods. So in WSNC, we choose NB and C4.5 as the classifiers.

5. Conclusions and future work

In this paper, we explore the correlations among workers and propose a novel noise correction algorithm called worker similarity-based noise correction (WSNC). At first, WSNC exploits the annotating information of similar workers on similar instances to estimate the quality of each label annotated by each worker on each instance. Then, WSNC re-infers the integrated label of each instance based on the qualities of its multiple noisy labels. Subsequently, WSNC considers the instance whose re-inferred integrated label differs from its original integrated label as a noise instance and thus obtains a clean set and a noise set. Finally, WSNC builds two different classifiers on the clean set to further correct the noise instances. The extensive experiments on 34 simulated and three real-world crowdsourced datasets show that WSNC is superior to all the other state-of-the-art noise correction algorithms used for comparison.

In the current version, we define a simple worker similarity metric based on label consistency. However, in complex crowdsourcing scenarios, worker similarity might be influenced by various factors beyond label consistency, such as expertise, biases and domain knowledge. We believe that a more sophisticated similarity metric that takes these factors into account could potentially yield better results. This is our main research direction in the future.

Declaration of competing interest

I confirm that there is no conflict-of-interest in the submission, and the manuscript has been approved by all authors for publication.

Data availability

The source code of WSNC and the datasets used in this work can be obtained from the website: https://github.com/jiangliangxiao/WSNC.

Acknowledgments

The work was partially supported by National Natural Science Foundation of China (62276241) and Foundation of Key Laboratory of Artificial Intelligence, China, Ministry of Education, P.R. China (AI2022004).

References

[1] L. Jiang, L. Zhang, L. Yu, D. Wang, Class-specific attribute weighted naive Bayes, Pattern Recognit. 88 (2019) 321–330.
[2] J. Zhang, Knowledge learning with crowdsourcing: A brief review and systematic perspective, IEEE CAA J. Autom. Sinica 9 (5) (2022) 749–762.
[3] K. Zhu, S. Xue, L. Jiang, Improving label quality in crowdsourcing using deep co-teaching-based noise correction, Int. J. Mach. Learn. Cybern. 14 (10) (2023) 3641–3654.
[4] W. Yang, C. Li, L. Jiang, Learning from crowds with robust support vector machines, Sci. China Inf. Sci. 66 (3) (2023) 132103.
[5] W. Zhang, L. Jiang, Z. Chen, C. Li, FNNWV: Farthest-nearest neighbor-based weighted voting for class-imbalanced crowdsourcing, Sci. China Inf. Sci. (2023) http://dx.doi.org/10.1007/s11432-023-3854-7.
[6] T. Buecheler, J.H. Sieg, R.M. Füchslin, R. Pfeifer, Crowdsourcing, open innovation and collective intelligence in the scientific method - A research agenda and operational framework, in: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2010, Odense, Denmark, August 19–23, 2010, MIT Press, 2010, pp. 679–686.
[7] M. Buhrmester, T. Kwang, S.D. Gosling, Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? Perspect. Psychol. Sci. 6 (1) (2011) 3–5.
[8] X. Kang, G. Yu, C. Domeniconi, J. Wang, W. Guo, Y. Ren, L. Cui, Crowdsourcing with self-paced workers, in: J. Bailey, P. Miettinen, Y.S. Koh, D. Tao, X. Wu (Eds.), IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7–10, 2021, IEEE, 2021, pp. 280–289.
[9] J. Xie, M. Reddie, S. Lee, S.M. Billah, Z. Zhou, C. Tsai, J.M. Carroll, Iterative design and prototyping of computer vision mediated remote sighted assistance, ACM Trans. Comput. Hum. Interact. 29 (4) (2022) 36:1–36:40.
[10] S. Mishra, D. Khashabi, C. Baral, H. Hajishirzi, Cross-task generalization via natural language crowdsourcing instructions, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22–27, 2022, Association for Computational Linguistics, 2022, pp. 3470–3487.
[11] V.S. Sheng, F.J. Provost, P.G. Ipeirotis, Get another label? improving data quality and data mining using multiple, noisy labelers, in: Y. Li, B. Liu, S. Sarawagi (Eds.), Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24–27, 2008, ACM, 2008, pp. 614–622.
[12] J. Zhang, X. Wu, V.S. Sheng, Imbalanced multiple noisy labeling, IEEE Trans. Knowl. Data Eng. 27 (2) (2015) 489–503.
[13] D. Tao, J. Cheng, Z. Yu, K. Yue, L. Wang, Domain-weighted majority voting for crowdsourcing, IEEE Trans. Neural Networks Learn. Syst. 30 (1) (2019) 163–174.
[14] Y. Li, B.I.P. Rubinstein, T. Cohn, Exploiting worker correlation for label aggregation in crowdsourcing, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA, in: Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 3886–3895.

[15] L. Jiang, H. Zhang, F. Tao, C. Li, Learning from crowds with multiple noisy label distribution propagation, IEEE Trans. Neural Networks Learn. Syst. 33 (11) (2022) 6558–6568.
[16] Z. Chen, L. Jiang, C. Li, Label augmented and weighted majority voting for crowdsourcing, Inform. Sci. 606 (2022) 397–409.
[17] W. Yang, C. Li, L. Jiang, Learning from crowds with decision trees, Knowl. Inf. Syst. 64 (8) (2022) 2123–2140.
[18] Y. Zhang, L. Jiang, C. Li, Attribute augmentation-based label integration for crowdsourcing, Front. Comput. Sci. 17 (5) (2023) 175331.
[19] B. Nicholson, V.S. Sheng, J. Zhang, Label noise correction and application in crowdsourcing, Expert Syst. Appl. 66 (2016) 149–162.
[20] J. Zhang, V.S. Sheng, T. Li, X. Wu, Improving crowdsourced label quality using noise correction, IEEE Trans. Neural Netw. Learn. Syst. 29 (5) (2018) 1675–1688.
[21] W. Xu, L. Jiang, C. Li, Improving data and model quality in crowdsourcing using cross-entropy-based noise correction, Inform. Sci. 546 (2021) 803–814.
[22] Z. Chen, L. Jiang, C. Li, Label distribution-based noise correction for multiclass crowdsourcing, Int. J. Intell. Syst. 37 (9) (2022) 5752–5767.
[23] H. Li, L. Jiang, S. Xue, Neighborhood weighted voting-based noise correction for crowdsourcing, ACM Trans. Knowl. Discov. Data 17 (7) (2023) 96.
[24] D. Gamberger, N. Lavrac, C. Groselj, Experiments with noise filtering in a medical domain, in: I. Bratko, S. Dzeroski (Eds.), Proceedings of the Sixteenth International Conference on Machine Learning, ICML 1999, Bled, Slovenia, June 27–30, 1999, Morgan Kaufmann, 1999, pp. 143–151.
[25] C.E. Brodley, M.A. Friedl, Identifying mislabeled training data, J. Artificial Intelligence Res. 11 (1999) 131–167.
[26] T.M. Khoshgoftaar, P. Rebours, Improving software quality prediction by noise filtering techniques, J. Comput. Sci. Tech. 22 (3) (2007) 387–396.
[27] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, J. Artificial Intelligence Res. 6 (1997) 1–34.
[28] J. Zhang, V.S. Sheng, B. Nicholson, X. Wu, CEKA: a tool for mining the wisdom of crowds, J. Mach. Learn. Res. 16 (2015) 2853–2858.
[29] P. Langley, W. Iba, K. Thompson, An analysis of Bayesian classifiers, in: W.R. Swartout (Ed.), Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12–16, 1992, AAAI Press / The MIT Press, 1992, pp. 223–228.
[30] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[31] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[32] L. Jiang, L. Zhang, C. Li, J. Wu, A correlation-based feature weighting filter for naive Bayes, IEEE Trans. Knowl. Data Eng. 31 (2) (2019) 201–213.