International Journal of Machine Learning and Cybernetics (2020) 11:231–248
https://doi.org/10.1007/s13042-019-00969-8

ORIGINAL ARTICLE

Ensemble with estimation: seeking for optimization in class noisy data

Ruifeng Xu[1] · Zhiyuan Wen[1] · Lin Gui[2] · Qin Lu[4] · Binyang Li[3] · Xizhao Wang[5]

Received: 26 November 2018 / Accepted: 22 May 2019 / Published online: 12 June 2019

Abstract
Class noise, as know as the mislabeled data in training set, can lead to poor accuracy in classification no matter what machine learning methods are used. A reasonable estimation of class noise has a significant impact on the performance of learning methods. However, the error in existing estimation is inevitable theoretically and infer the performance of optimal classifier trained on noisy data. Instead of seeking a single optimal classifier on noisy data, in this work, we use a set of weak classifiers, which are caused by negative impacts of noisy data, to learn an ensemble strong classifier which is based on the training error and estimation of class noise. By this strategy, the proposed ensemble with estimation method overcomes the gap between the estimation and true distribution of class noise. Our proposed method does not require any a priori knowledge about class noises. We prove that the optimal ensemble classifier on the noisy distribution can approximate the optimal classifier on the clean distribution when the training set grows. Comparisons with existing algorithms show that our methods outperform state-of-the-art approaches on a large number of benchmark datasets in different domains. Both the theoretical analysis and the experimental result reveal that our method can improve the performance, works well on clean data and is robust on the algorithm parameter.

Keywords Class Noise · Ensemble Learning · Machine Learning

1 Introduction

Typical machine learning method uses a classifier learned from a labeled dataset (i.e., the training data) to predict the class labels of new samples (i.e., the testing data). In most of classification applications, labels of the training data are assumed correct. However, real-world datasets often contain noise which may occur either in the features of the data, defined as the attribute noise, or in the labels of the data, defined as the class noise.

Many studies have focused on handling attribute noise since it is quite common in machine learning and data mining tasks. However, researchers, such as [1, 2], have indicated that class noise can be potentially more detrimental than attribute noise. The study on class noise problem has an essential impact on classification performance improvement [1]. We must point out that class noise is unavoidable in many real world applications such as disease prediction in medical applications [3], food labeling for the food industry [4], and manual data labeling in some natural language processing applications [5–7].

Generally speaking, there are two types of strategies to deal with class noise. The first entails learning with noisy data and the second is based on noise elimination.

Learning with noise assumes that each training sample is assigned to a weight based on an estimated probability of class noise, that is, the class noise rate. The learning algorithm will consider the class noise rate while learning from the original noisy training data [8]. Unfortunately, this method requires a priori knowledge of the class noise rate in the training data.

Noise elimination strategy attempts to detect and remove erroneous data from the training set [9–11]. This method can reduce the rate of class noise in the training data, yet it often leads to an overall reduction of training samples.

✉ Lin Gui
Lin.gui@warwick.ac.uk

[1] Harbin Institute of Technology Shenzhen, Shenzhen, China
[2] Department of Computer Science, University of Warwick, Coventry, UK
[3] University of International Relations, Beijing, China
[4] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
[5] Big Data Institute, ShenZhen University, Shenzhen, China

Even though the reduced training data may be less noisy, reduced training data may result in reduced performance of a learning algorithm compared to the result from using the original noisy data. Therefore, good noise estimation methods are important to avoid overrating class noises which can lead to large data reduction. One successful approach for class noise elimination is called kernel density estimation including methods such as k-nearest neighbors (kNN) or Parzen Window to detect and remove noisy data [10, 12, 13]. However, there is a theoretical flaw in these methods. Their work reposes on kernel density estimation, which requires a small radius of neighborhood ε, to satisfy both the manifold assumption and the central-limit theorem which requires a large ε to estimate the parameter for Guassian distribution. The contradicting assumptions will limit the performance of their methods in applications.

In this paper, we propose a novel method to estimate the noise rate for each training sample. In order to avoid the overrating of class noise rate and the contradict assumptions, we introduce the sum of Rademacher distribution [14] instead of the central-limit theorem to estimate class noise. We choose the kNN graph for the kernel density estimation because it is more sensitive to class noise.

Based on this noise estimation method, we then modify a given surrogate loss function based on the estimated class noise rate. The modified surrogate loss function is the optimization objective of the classifier. According to the theoretical analysis, this loss function is sensitive to the parameter in the estimation algorithm. So, we propose a sampling based algorithms to obtain a strong classifier through a series of weak classifiers as an ensemble to overcome the sensibility of the parameter,which is adopted to optimize the loss function on the training data. Traditionally, the ensemble method is not suitable to handle class noise because it will also enhance the noise in the learning process [15, 16]. However, in our method, we take into consideration of the noise rate and make the algorithm to adapt the noisy distribution. Both the theoretical analysis and the experimental result reveal that our method can improve the performance, works well on clean data and is robust on the algorithm parameter.

The main contributions of this paper are:

- A class noise estimation approach is proposed based and a new weighted loss function is given;
- The proposed loss function based on the estimated class noise experimentally demonstrates better class noise estimation performance than the existing popular algorithms;
- In comparison with the existing methods of noise estimation, our approach requires no a priori knowledge about the class noise and it makes up for the contradiction of the currently used theory. Performance evaluation also show that we can indeed achieve state-of-the art results.

The remainder of this paper is organized as follows. Section 2 provides a review of related works on class noise estimation and the two types of strategies to deal with class noise. Section 3 presents the problem setup and background. Sections 4 and 5 present our proposed class noise estimation method and our method to incorporate the estimation into class noise elimination, respectively. Section 6 shows the performance evaluation on real-world public datasets in different domains. Section 7 lists our conclusion and future work.

2 Related work

Identifying class noise is an important issue in machine learning. Previous publications cover both the theory [17] and the application aspects [12, 13] of this topic. Especially, in recent years, with the development of deep learning, how to train a rubost neural network becomes a new hot topic in the designing of learning architecture [18, 18–21]. In this section, we briefly introduce related works from three perspectives: the source of class noise, the handling of class noise and the application of class noise.

Class noises can exist for different reasons. When used for disease prediction in medical applications [3], training data contains a probability of false positive or negative because data comes from medical experiments. In other words, class noises naturally exist and cannot be avoided. Food labeling for the food industry [4] also faces class noise problem. As shown by [4], beef has a higher price than mutton in some countries in South Africa. Miss-labeling is thus an aforethought to achieve more benefit. The manual labeling of data in some natural language processing applications [5–7] also contains class noise because there is always a possibility of inter-annotator inconsistency.

Given x, being the set of features of a sample, let $\widetilde{y}$ be the observed label of x; y the true label of the sample x and p is the probability to flip the true label into a noise label (thus p is the class noise rate, or noise rate for short). For any training algorithm, only x and $\widetilde{y}$ can be observed. In general, class noises can be simply categorized into three different models based the dependence of noise to y and p [22]. The first model in is called the noise completely at random model [17, 23, 24]. The basic idea is that the class noise rate of a sample is completely random and independent of the labels and the feature set. Thus, an observed label is only determined by the true label and the class noise rate. The second model shown is called the Noise at Random Model [25–27]. In this model, the class noise rate is dependent on the true label of a sample and is independent of the feature set of the sample. In other words, different labels have different probabilities to flip to the wrong label. The third model shown is the noise not at random model [28, 29]. This model assumes the class noise rate should be affected by both the label and the feature set of the sample. Informally, this model can be described as: a sample will be miss-labeled to the most similar category.

The theoretical research on class noise and learning was first proposed by Angluin and Laird in 1988 [17]. In their work, all instances of the labeled data for binary classification have a uniform invert probability $p \in [0, 1/2)$. This is referred to as the random classification noise.

Class noises are typically assumed to be stochastic in algorithms that can handle class noises. The work by Ref. [8] assumed a learned noise rate from a priori knowledge and every sample was given the same probability, a simple assumption that may not be reasonable in all scenarios. The Cut Edge Weight Statistic method [10] also required prior assumed noise rate. This method used the prior probability as a hypothesis to test if the training sample satisfies the null hypothesis. The method also required the neighbors of a training sample to follow the central limit theorem, which is not reasonable because the set of neighbors are too small. Other works simply did not consider noise rate [5, 9, 11, 30, 31].

There are two basic categories of strategies to deal with class noise in training data: either learning with class noise [5, 8] or class noise elimination [9–11]. The learning with noise strategy approximates a distribution of noiseless training data using the distribution of the original training data with class noise and must have a priori knowledge about the class noise [8]. The problem, however, is that a priori knowledge of the class noise is not typically available, limiting the applicability of this method. Li [24] uses the Kernel Fisher method to estimate the class noise. The estimation is then used to use a robust algorithm that can tolerate noise.

The class noise elimination strategy attempts to detect the samples with high noise probabilities and remove them from the training set. There are different methods to detect class noise and they can be categorized as classification-based methods and graph-based methods. The classification-based method was first proposed by Brodley in 1999 [9]. He used k-fold cross-validation to split the training set into k subsets, and used any $k-1$ sets as the training data to classify the remaining data. If the classification result for a sample is different from the original label, that sample is considered class noise and is removed. Zhu et al. proposed a more efficient algorithm suitable for large datasets [30]. Zhu also proposed a cost-sensitive approach based on k-fold cross-validation [1]. Sluba employed a 10-fold cross-validation to detect class noise [11]. The graph-based method, known as the Cut Edge Weight Statistic method, was proposed by Zighed [10]. The principal idea is based on the manifold assumption and Bernoulli distribution assumption. A similar approach was proposed by Jiang and Zhou, who used a kNN graph to detect class noise without considering noise rate [32]. There are three major problems with the elimination approach. First, some correctly labeled training data can be eliminated because of potentially inaccurate class noise identification. Second, the number of samples in the training data will be reduced, potentially leading to an adverse effect on the learning algorithm performance. Third, the manifold assumption requires small k and the sum of Bernoulli distribution converges to Gaussian distribution if and only if k is larger than 25. The paradox leads to a limited performance gain in the class noise estimation. In elimination-based methods, reliable noise estimation is crucial and inaccurate estimations can potentially degrade performance compared to no noise elimination.

Some works also use some revision to the AdaBoost learning algorithm to handle noise data [33–35]. In principle, AdaBoost [36] is not a suitable learning algorithm for noisy data. The reason is that during the learning process AdaBoost will enhance the misclassified training samples in the next iteration. For a noisy sample, which is actually classified correctly, may be regarded as a misclassified sample due to the noisy label to lead to worsened performance [15, 16]. However, some strategies can be used to smooth out weight updating in the learning steps of AdaBoost to avoid over-fitting on noisy data [33, 34]. Another method simply allows boosting to miss-classify some of the training samples to obtain a more robust boosting [35]. There are also adaptive methods by using a confidence score and removing a sample if its noise estimation confidence is higher than a threshold [37–39].

3 Problem setup and background

Before presenting our proposed algorithm, we need to explain the background and the problem setup first. The fundamental problem is that each sample in the training data has the probability to be a miss-labeled sample, which means that the samples have class noise. Hence, we need to estimate the probability for the miss-labeling of each training sample, referred to as the class noise rate in the rest of this paper. After that, we can train a classifier on the noisy training data based on the estimated probability.

The second problem is how to measure the learning ability of the classifier with the obtained noise rate? Is it possible that our classifier, trained on noisy distribution, can achieve similar performance to the classifier trained on the clean distribution? If the answer to this question is yes, we can then use the differential between the two classifiers to measure the learning ability of the classifier trained on noisy distribution. If the classifier trained on noisy distribution is the optimal classifier on the clean distribution, this differential should be a small number theoretically.

In this section, we first give a strict formalized definition of the class noise rate, followed by the learning ability measure of the classifiers.

Let D denote the clean distribution without class noise, and $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ denote n training samples from D with true binary label y_i $(y_i = \pm 1, i = 1, 2, \dots, n)$ When there are class noises, $\widetilde{D}$ denotes the observed distribution which contains class noise, and the training sample from a noisy distribution $\widetilde{D}$ are $(x_1, \widetilde{y}_1), (x_2, \widetilde{y}_2), \dots, (x_n, \widetilde{y}_n)$, where the label $\widetilde{y}_i$ may be different from the true label y_i. In this paper, we want to estimate noise rate at the level of the individual samples. Here, we give the definition of class noise rate as follows:

Definition 1 Let $(x_i, \widetilde{y}_i) \in \widetilde{D}$ be a sample from the noisy distribution.The class noise rate is the probability of the observed label different from the true label of x_i, denoted by $P(\widetilde{y}_i \neq y_i | x_i)$.

According this definition, class noise rate defines on individual data samples. In other words, we allow different samples to have different noise rate.Thus, the first issue is how to estimate the class noise rate for each training sample. The main challenge, however, is that a learner can only observe the noisy data $(x_i, \widetilde{y}_i) \in \widetilde{D}$ and there is no a priori knowledge about the class noise rate and the clean distribution D.

For the time being, let us assume that we have a reasonable estimation method. Then the second issue is how to measure the performance of a classifier. Generally speaking, the objective of a classifier is to minimize a given loss function on a given set of training data. However, in this paper, the observed training data contains class noise. The minimized loss function on noisy training data may not be the minimized loss function on the clean data. So the issue is whether we can use the estimation of the class noise rate for each individual sample to modify the loss function on the noisy distribution so that the modified loss function can also minimize the loss on the clean distribution.

In order to address the problem in a formally, we give some definitions below.

Definition 2 Let $f : X \to \mathbb{R}$ be a real-value decision function, defined as $f(x) = P(y = 1|x) - 1/2$. The risk of f for each sample on the clean distribution is 0–1 loss given by $R_D(f) = E_{(x,y)\sim D}(1_{sign(f(x)\neq y)})$

Let $l(f(x), y)$ denote a loss function with a real-value prediction, for the clean distribution where $y = \pm 1$ is the true label of x. We can then use the estimation of the class noise rate for each individual sample to modify the loss function on the noisy distribution with an observed label, denoted as $\widetilde{l}(f(x), \widetilde{y})$. The modified loss function is marked with a hat $\widetilde{\cdot}$. Because the loss function is defined under the noisy distribution $\widetilde{D}$. Then, we can define three related risks as follows:

Definition 3.1 The empirical $\widetilde{l}$-risk on the training data: $\widehat{R}_{\widetilde{l}}(f) = \frac{1}{n}\sum_{i=1}^{n} \widetilde{l}(f(x_i), \widetilde{y})$.

Definition 3.2 The excepted $\widetilde{l}$-risk under noisy distribution $\widetilde{D}$: $R_{\widetilde{l},\widetilde{D}}(f) = E_{(x,\widetilde{y})\sim\widetilde{D}}(\widetilde{l}(f(x), \widetilde{y}))$.

Definition 3.2 The excepted l-risk under clean distribution D: $R_{l,D}(f) = E_{(x,y)\sim D}(l(f(x), y))$.

(Here, the hat $\widehat{\cdot}$ means that the marked object is a estimated result. The hat $\widetilde{\cdot}$ means that the noisy label will influence the marked object.)

Here the empirical $\widetilde{l}$-risk $\widehat{R}_{\widetilde{l}}(f)$ is is the expected error of the trained classifier on noisy distribution, and the expectation of l-risk $R_{l,D}(f)$ is the expected error of a classifier training on clean distribution. The learning ability of a training classifier is the difference between the two risks: $|\widehat{R}_{\widetilde{l}}(f) - R_{l,D}(f)|$ It indicates the distance between our trained classifier on noisy data and the optimal classifier on the clean distribution. The objective of our algorithm is to make $|\widehat{R}_{\widetilde{l}}(f) - R_{l,D}(f)|$ approaching 0 when the size of noisy training set grows.

4 Class noise estimation

In this section, we deploy a class noise estimation method proposed in our previous work [40]. Due to the length limitation, we only introduce the basic idea and theorem. The details of mathematical proof can be found in paper [40]. The idea is based on the kernel density estimation method. In this method, the label of an individual sample should be similar to the most similar neighbors even if there is class noise in the data. Therefore, we first present a class noise model based on the kernel density estimation method. We mainly introduce the kNN graph as kernel density estimation method, and we will also introduce the formula based on Parzen Window (known as e-graph).Since the kernel density estimation method is sensitive to class noise and it can overrate the class noise. To avoid overrating of noises, we introduce a loose distribution called the Sum of Random Noise in this section. As will be seen in the evaluation, our method is more reasonable theoretically and shows a higher performance in the experiments when compare to the existing estimation methods [10, 12, 13].

For any $(x_i, \widetilde{y}_i)$ from the training set with true label y_i, we can define a sign function on the individual sample $(x_i, \widetilde{y}_i)$ and its k nearest samples $(x_j, \widetilde{y}_j)$:

$$I_{ij} = \begin{cases} 1, & \textbf{if} \quad \widetilde{y}_j \neq \widetilde{y}_i \\ -1, & \textbf{if} \quad \widetilde{y}_j = \widetilde{y}_i. \end{cases} \tag{1}$$

I_{ij} indicates the difference between an individual sample and its nearest labels. Even $\widetilde{y}_j$ can also contain noise with some probability, the noise level of $\widetilde{y}_i$ should still be similar to that of $\widetilde{y}_j$. Furthermore, the noise estimation method should also consider similarity. Thus, we define a statistic, S_i, referred to as the sum of noisy similarity, which can be used to measure the total noise level.

Definition 4 For any individual sample $(x_i, \widetilde{y}_i) \in \widetilde{D}$ and the top k nearest sample $(x_j, \widetilde{y}_j), j = 1, 2, \dots, k$ under Formula (1) and a normalized similarity w_{ij}, the sum of noisy similarity is:

$$S_i = \sum_{j=1}^{k} I_{ij} w_{ij}. \quad (2)$$

Now, we can define the class noise rate for $(x_i, \widetilde{y}_i)$ as the probability of S_i being opposite from *SRN*. Because $I_{ij} = 1$ indicates that the sample x_i has different label to its nearest neighbor and the similarity metric is between 0 to 1, the larger S_i is, the higher the probability it should be that x_i has a noisy label. So, we should only consider the upper quantile of *SRN* here. Now, we can give the class noise rate under Definition 5.

Definition 5 For any individual sample $(x_i, \widetilde{y}_i) \in \widetilde{D}$ and the sum of noisy similarity S_i, the probability of *SRN* denoted as P_{SRN}, the class noise rate of $(x_i, \widetilde{y}_i)$ is $1 - P_{SRN}(s \geq S_i)$.

The formal definition above would be easier to understand using the following explanation. If the principle of entropy maximum reveals a best guessed label, the upper quantile of *SRN* is the probability of the individual sample having a "worse" label than a guessed one. So Definition 5 can be presented as a descriptive definition as: the class noise rate of a sample is the probability of the observed label being worse than a guessed label.

Definition 5 defines the class noise rate of the sample $(x_i, \widetilde{y}_i)$ as the probability of S_i not following *SRN*. We can explain the definition in a different way. If the label of a training samples totally random under PEM, the label of the training sample can be considered as a label from guessing. Then we can get the distribution of S_i from a "guessing result". If the individual sample $(x_i, \widetilde{y}_i)$ contains class noise, the corresponding S_i should be larger than the "guessing result". Thus, we can say the probability of S_i does not follow SRN. The formal theorem is given below:

Theorem 1 *The estimated class noise rate of* $(x_i, \widetilde{y}_i) \in \widetilde{D}$ *denoted by* $P_c(x_i)$, *is:*

$$P_c(x_i) \geq 1 - 0.5 * \exp\left(- \frac{\left(\sum_{j=1}^{K} w_{ij} I_{ij}\right)^4}{2\left(\| w_i \|_1 \| w_i \|_2\right)^2} \right). \quad (3)$$

Lemma 1 *Let* $I_{i1}, I_{i2}, \dots, I_{ik}$ *be independent Bernoulli random variables* ($P(I_{ij} = 1) = P(I_{ij} = -1) = 1/2$), $w_{i1}, w_{i2}, \dots, w_{ik}$ *be a sequence of real number such that* $w_i = (w_{i1}, w_{i2}, \dots, w_{ik}) \in \mathcal{L}_2$ *and* $t > 0$. *We can the conclude that*

$$P\left(\sum_{j=1}^{k} I_{ij} w_{ij} > K_{1,2}(w_i, t) \right) \leq e^{-\frac{t^2}{2}}, \quad (4)$$

where $K_{1,2}(w_i, t)$ *is defined as:*

$$K_{1,2}(w_i, t) = \inf \left\{ \| w_i' \|_1 + t \| w_i'' \|_2 \right\}.$$

Here $\| \cdot \|_1$ and $\| \cdot \|_2$ are the $\mathcal{L}_1$ and $\mathcal{L}_2$ norms; and $w_i' + w_i'' = w_i$. The formula of $K_{1,2}(w_i, t)$ is well known as the K-method of real interpolation for Banach Space [41]. The proof of Lemma 1. was given by [14] in details. According to Lemma 1, we can easily get the Lemma 2 by a sub-optimal solution of $K_{1,2}(w_i, t)$.

Lemma 2 *Let* $I_{i1}, I_{i2}, \dots, I_{ik}$ *be independent Bernoulli random variables* ($P(I_{ij} = 1) = P(I_{ij} = -1) = 1/2$), $w_{i1}, w_{i2}, \dots, w_{ik}$ *be a sequence of real number such that* $w_i = (w_{i1}, w_{i2}, \dots, w_{ik}) \in \mathcal{L}_2$ *and* $t > 0$, *we have,*

$$P\left(\sum_{j=1}^{k} I_{ij} w_{ij} > \sqrt{t \| w_i \|_1 \| w_i \|_2} \right) \leq e^{-\frac{t^2}{2}},$$

where $\| \cdot \|_1$ *and* $\| \cdot \|_2$ *are the* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *norms.*

The proof of Lemma 2 is given in our previouse work [40].

According to Definition 4, the estimated class noise rate $P_c(x_i)$ for $(x_i, \widetilde{y}_i) \in \widetilde{D}$ is the probability of $(x_i, \widetilde{y}_i)$ not following *SRN*. According to Lemma 2, the probability of $(x_i, \widetilde{y}_i)$ from *SRN* is less than $\exp\left(- \frac{\left(\sum_{j=1}^{K} w_{ij} I_{ij}\right)^4}{2\left(\| w_i \|_1 \| w_i \|_2\right)^2} \right)$ when $t > 0$.

Thus, consider the assumption of $P(I_{ij} = 1) = P(I_{ij} = -1) = 1/2$, the estimated class noise rate is:

$$P_c(x_i) \geq 1 - 0.5 * \exp\left(- \frac{\left(\sum_{j=1}^{K} w_{ij} I_{ij}\right)^4}{2\left(\| w_i \|_1 \| w_i \|_2\right)^2} \right) \blacksquare$$

For noise detection, the lower boundary is more significant since it pertains to the minimum noise rate of a labeled training sample. In practice, we can then use

$$P_c(x_i) = 1 - 0.5 \times e^{-r(x_i)/2} \quad (5)$$

as the estimation for each training sample. Here,

$$r(x_i) = \left(\frac{\left(\sum_{j=1}^{K} w_{ij} I_{ij} \right)^4}{\left(\| w_i \|_1 \| w_i \|_2 \right)^2} \right).$$

$P_c(x_i)$ can be either be incorporated in the learning algorithms to weight the importance of a training sample or used to identify for elimination. It should be noticed that $r(x_i)$ is symmetrical on S_i. It means that S_i is less than 0, which indicates the label of the individual is similar to the nearest label, the upper quantile of S_i needs to be solved by the lower quantile of *SRN* which is introduced by opposite sign function.

5 Learning in noise data

In this section, we introduce our learning algorithm based on the estimated class noise rate. We propose to modify a given surrogate loss function based on the estimated class noise rate. The modified surrogate loss function is the optimization objective of the classifier. In this section, two training algorithms are used to optimize the loss function on the training data. One is a perceptron based method, which is based on the learning with noise strategy and aims to reduce the impact of noisy data. The other one is a sampling based Adaboost method, which is based on the class noise elimination strategy and focuses on selecting high quality training data rather than to identify low quality data.

5.1 Class noise estimation based on loss function

Based on the class noise estimation given in Formula (5), we propose a surrogate loss function based on the estimated class noise rate. The key is to ensure that the surrogate loss function can adequately approximate the loss function for the clean training data. Then the loss function on the observed distribution is defined as

$$\widetilde{l}(f(x_i), \widetilde{y}_i) = \frac{(1 - P_c(x_i)) l(f(x_i), \widetilde{y}_i) - P_c(x_i) l(f(x_i), -\widetilde{y}_i)}{1 - 2\frac{\sum_i P_c(x_i)}{n}}. \tag{6}$$

Without loss of generality, we do not specify any particular loss function on the observed distribution. Theoretically, any loss function can be used in Formula (6) to modified a surrogate loss function. In Formula (6), the numerator is formed by two parts. The first part is the original loss function with the observed labels weighted by their label correctness probabilities. The second is a penalty for the loss function with an inverted label (i.e., the probability of the observed label is incorrect) weighted by the class noise rate. The denominator is based on the average class noise rate to ensure that the expectation of loss on noisy training data approximates the expectation of loss on the clean data. This is an updated version of the original loss function on noisy data.

Our question is whether the minimum risk of the proposed surrogate loss function on the noisy training data can approximate the minimum risk of the original loss function on the clean data. In general, the main question is whether we can train a classifier for the cleaning distribution by the noisy data only without any prior knowledge? Let us assume that the training data is clean. Then, the risk of the loss function on the clean distribution and noisy distribution are defined under Definitions 3.1, 3.2 and 3.3. Based on these definitions, the main question can be split into two questions:

- Will $\widehat{R}_{\widetilde{l}}(f)$ converge to $R_{\widetilde{l},\widetilde{D}}(f)$ under the noisy distribution when n grows? If the answer is "Yes", it means that we could train a stable classifier on the noisy training data.
- Will $R_{\widetilde{l},\widetilde{D}}(f)$ converge to $R_{l,D}(f)$ under the clean distribution? If the answer is "Yes", it means that the trained classifier by noisy training data is also the optimal classifier on the clean distribution.

For the frist question, we can use the Chebyshev law of large numbers to prove. Given n independent $(x_i, \widetilde{y}_i)$, $P_c(x_i)$ denotes the class noise rate estimated by Formula (5) with the expectation $E(P_c(x_i))$. According to the Chebyshev law of large numbers, $\forall \varepsilon > 0, \exists \delta > 0$, and $N > 0$, when $n > N$, it is true that

$$\left| \sum_i \frac{P_c(x_i)}{n} - E(P_c(x_i)) \right| \le \varepsilon.$$

For the same reason, $\forall \varepsilon > 0, \exists \delta > 0$, and $N > 0$, when $n > N$, it is true that

$$\left| \frac{1}{n} \sum_i \widetilde{l}(f(x_i), \widetilde{y}_i) - E\left(\widetilde{l}(f(x_i), \widetilde{y}_i) \right) \right| \le \varepsilon$$

with a probability of at least $1 - \delta$.

Then the risk on $\widetilde{D}$ and the empirical $\widetilde{l}$-risk on the training data satisfies

$$|\widehat{R}_{\widetilde{l}}(f) - R_{\widetilde{l},\widetilde{D}}(f)| \le 2\varepsilon$$

with probability at least $1 - \delta$. ■

In other word, if we have a perfectly correct estimation of the class noise rate, the empirical $\widetilde{l}$-risk on the training data $\widehat{R}_{\widetilde{l}}(f)$ will converge to the risk $R_{\widetilde{l},\widetilde{D}}(f)$ of the loss function on the noise data when the size of the training data is sufficiently large.

For the second question, the proof is given in detail by Ref. [8], which will not repeated here, Based on the proof in Ref. [8], the following inequality holds:

$$|R_{l,D}(\widehat{f}_p) - R_{l,D}(f_*)| \leq 2\mathfrak{R}(\widetilde{l}\circ\mathcal{F}) + 2\sqrt{\frac{\log(1/\delta)}{2n}} \tag{7}$$

with probability of at least $1-\delta$.

Here, f_* is the minimizer of $R_{l,D}(f)$ on the clean distribution, $\widehat{f}_p$ is the minimizer of $R_{l,D}(f_p)$ on the clean distribution and

$$\mathfrak{R}(\widetilde{l}\circ\mathcal{F}) = \mathbb{E}_{x_i,\widetilde{y}_l,\epsilon_i}\left[\sup_{f\in\mathcal{F}} = \frac{1}{n}\sum\nolimits_{i=1}^{n}\epsilon_i\widetilde{l}(f(x_i),\widetilde{y}_l)\right],$$

where ϵ_i is an independent and identically distributed Rademancher random variable. The right side of the inequality in Formula (7) is bound by the richness of the function family $\mathcal{F}$ and the sample size of n.

Thus the risk of our estimation method $R_{l,D}(\widehat{f})$ satisfies:

$$|R_{l,D}(\widehat{f}) - R_{l,D}(f_*)| \leq \left(|R_{l,D}(\widehat{f}) - R_{l,D}(\widehat{f}_p)| + |R_{l,D}(\widehat{f}_p) - R_{l,D}(f_*)|\right). \tag{8}$$

Because the second item is bound, the risk of our estimation is determined by the first item which is related to the estimation result. This can be interpreted as that the performance of the classifier trained on the noisy data is determined by the estimation result when the size of the training set grows.

5.2 Learning in class noise

According to Sect. 5.1, the basic idea is to use the surrogate loss function defined in Formula (6) to train a classifier on the noisy training data. In a previous work [40], a simple Perceptron based on-line learning method is used with noisy class data. However, the theoretical analysis has revealed that the result of the surrogate loss function would be affected by the estimated class noise rate. It is shown in Formula (8).

In Formula (8), $|R_{l,D}(\widehat{f}_p) - R_{l,D}(f_*)|$ is dependent on the size of the training dataset. Theoretically speaking, when the size of the training set grows, this absolute value will approximate to 0. It means that the risk of the surrogate loss function is decided by $|R_{l,D}(\widehat{f}) - R_{l,D}(\widehat{f}_p)|$, which is related to the estimation result. If we have a good estimation, this item should be small. Otherwise, we will face the problem with a risky surrogate loss function.

Algorithm 1: ALGORITHM: *AdaBC*

```
Input :
        $S$: The training dataset with $N$ samples;
        $(x_i, \widetilde{y}_i)$: The training sample $(x_i, \widetilde{y}_i) \in S$;
        $T$: Number of iterations.
begin
    For each $(x_i, \widetilde{y}_i)$ in $S$, build a kNN graph and calculate
      $P_c(x_i)$ according to formula (5);
    Initialize the weight vector: $w_i^1 = 1 - P_c(x_i)$ ;
    for $j = 1$ to $T$ do
        Set $p_i^j = \frac{w_i^j}{\sum_i w_i^j}$ ;
        Sampling based on $p^j$ for each $(x_i, \widetilde{y}_i)$, the result is
          $S_j$ ;
        Use the learner on $S_j$ to get a classifier
          $h_j : X \rightarrow [0, 1]$ ;
        Calculate the error of $h_j : \widetilde{\varepsilon}_j = \sum_{i=1}^{N} p_i^j f(x_i, \widetilde{y}_i)$
        (Here, $f_{error} = (x_i, \widetilde{y}_i) = |(1 - P_c(x_i))(h_j(x_i) -$
          $\widetilde{y}_i) - P_c(x_i)(h_j(x_i) - (-\widetilde{y}_i))|)$ ;
        Set $\beta_j = \widetilde{\varepsilon}_j/(1 - \widetilde{\varepsilon})$ ;
        Set the new weight as $w_i^{j+1} = w_i^j \beta_j^{1-f_{error}(x_i,\widetilde{y}_i)}$
    end
end
Output:
        The classifier:
```

$$h_f(x_i) = \begin{cases} 1 & \textbf{if } \sum\limits_{j=1}^{T}(\log(1/\beta_j))h_j(x) \geq \frac{1}{2}\sum\limits_{j=1}^{T}(log(1/\beta_j)) \\ 0 & \texttt{otherwise} \end{cases}$$

According to theoretical analysis, direct use of a perceptron or any other linear optimizer as the basic classifier (proposed in our previous work [40]) can face one problem: when the estimation is unreliable, the performance of the surrogate loss function may be limited because the first item in $|R_{l,D}(\widehat{f}) - R_{l,D}(\widehat{f}_p)|$ may not be sufficiently small. Since the estimation method proposed in Section 4 is based on kNN graph, estimation is related to the parameter k and the similarity measure. The incorrect parameter of the similarity measure may misguide the estimation result, and the error in the estimation will affect the learning algorithm according to our analysis.

Hence we propose an Adaboost based method based on the class noise elimination strategy to whittle the impact of estimation. The reason is that Adaboost makes use of a bag of weak classifiers to achieve a better classification by enhancing the "misclassified" sample in the learning process. However, for a noisy data, "misclassified" by a base classifier may actually indicate a good performance because the observed label is incorrect. These samples should be given a lower weight since they have been trained well. If we give these samples a higher weight as traditional AdaBoost, it may lead to worsened performance because the algorithm enhances the noisy data with noisy label causing overfitting on these samples.

In our method, we solve the problem by adjusting the weight. The basic idea is to use the surrogate loss function given in Formula (6) as the objective function. We then use the AdaBoost.M1 [36] method to achieve optimal result on the noisy training data. In the learning step, we use the surrogate loss function to avoid overfitting on the noisy data without the need to eliminate any training sample. In the optimization, the samples with high class noise probability is given a low weight when "misclassified" by the base classifier so that even if the noisy sample is "misclassified" in the training, it will not obtain a higher weight in the next iteration to enhance learning. By using this strategy, our algorithm can adaptively handle class noise in the training instead of overfitting on noisy data. The pseudo code is shown in Algorithm 1. The weight of each sample is initialized to the probability of the correct label. Then, this weight is used to sample the training data. That is, we use the samples seemed "correct" as the training data to train the base classifier. Based on the error of the whole dataset, we will update the weight to make sure that the sample with high correct label probability but misclassified will have a high weight so that it will be added into training in the next round.

Next, we need to provide the training error of Algorithm 1. Since the observed labels are noisy, we can only get the training error on noisy data. That is, $\frac{1}{N}\sum_{i=1}^{N}\left[h_f(x_i) \neq \widetilde{y}_i\right]$, We need to prove that the error on the clean data is also optimized in our algorithm. Reference [42] has given the boundary of training error for Adaboost.M1 on the clean training data. Take the training error in each iteration j as ε_j, take $\gamma_t = 1/2 - \varepsilon_j$. If $\forall \gamma_j > 0, \exists \gamma_j > \gamma$. Then the training error is:

$$\frac{1}{N}\sum_{i=1}^{N}\left[h_f(x_i) \neq y_i\right] \leq \exp(-2T\gamma^2).$$

However, in our task, we cannot observe the correct label of training data and we do not know the training error in each iteration neither. So, we cannot obtain the training error on the cleaning data directly. Fortunately, Formula (8) reveals that the gap between the training error on the clean data and the noisy data is decided by the estimation result. In each iteration, if we define the training error on the clean data as: $\varepsilon_j = \left[h_j(x_i) \neq y_i\right]$, and the training error on the noisy data as: $\widetilde{\varepsilon}_j = \left[h_j(x_i) \neq \widetilde{y}_i\right]$. There is a positive real number g to indicate the gap between the two errors: $|\varepsilon_j - \widetilde{\varepsilon}_j| \leq g$. So, we can obtain the inequality:

$$\widetilde{\varepsilon}_j + \frac{1}{2} - g \geq \frac{1}{2} - \varepsilon_j \geq \frac{1}{2} - \widetilde{\varepsilon}_j - g.$$

According to the Hoeffding boundary from Angluin and Laird [17], we have $\frac{1}{2} - \varepsilon_j \geq 0$ here. It means that if we take $\gamma_t = 1/2 - \varepsilon_j$, then $\forall \gamma_j > 0, \exists \gamma_j > \gamma$ such that the training error is:

$$\frac{1}{N}\sum_{i=1}^{N}\left[h_f(x_i) \neq y_i\right] \leq \exp(-2T\gamma^2). \tag{9}$$

This means that the classifier can be optimized on the noisy distribution and achieves an optimal result on the clean distribution.

6 Performance evaluation

6.1 Experimental setup

Our experiments evaluate the performance of the noise estimation method and show its usefulness in improving the learning performance. The evaluations are based on experiments with varying class-noise rates and training-set sizes compared to other state-of-the-art systems. We use seven public datasets for binary classification with different class-noise rates: (1) the LEU [43] set of cancer data, we reduce the dimensions into 20 by PCA; (2) the Splice for DNA sequence splice-junction classification; (3) the UCI Adult dataset collection containing seven subsets (referred to as UCI.a1a to UCI.a7a in this paper) of independent training and testing data [44]; (4) the DBWorld e-mails DataSet in English (in short, DB), which consists of 64 e-mails manually collected from DBWorld mailing list and are classified into two classes: "announces of conferences" and "everything else"; (5) the Farm Ads DataSet in English (in short, FADS), which is collected from text ads found on twelve websites that deal with various farm animal related topics with binary labels based on whether the content owner approves of the ad; (6) the Twitter Dataset for Arabic Sentiment Analysis Dataset (in short, TDA), the class labels are opinion polarity; (7) the Product Reviews from Amazon in three categories, Book, DVD and Music (in short, PRA) with class labels being the opinion polarities. (8) Banknote is the banknote authentication Data Set. Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. (9) Haberman contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer [45]. (10) ILPD contains 416 liver patient records and 167 non liver patient records [46]. (11) QSAR contains values for 41 attributes (molecular descriptors) used to classify 1055 chemicals into 2 classes (ready and not ready biodegradable) [47]. (12) SPEC cardiac Single Proton Emission Computed Tomography (SPECT) images. Each

Table 1 Overview of datasets

Data	Type	Dimension	Training size (+/−)	Testing size (+/−)
LEU	Cancer	7129 (20)	38 (0.71/0.29)	34 (0.59/0.41)
Splice	DNA	60	1000 (0.52/0.48)	2175 (0.52/0.48)
UCI.a1a	Adult	123	1605 (0.37/0.63)	30956 (0.24/0.76)
UCI.a2a	Adult	123	2265 (0.25/0.75)	30296 (0.24/0.76)
UCI.a3a	Adult	123	3185 (0.24/0.76)	29376 (0.24/0.76)
UCI.a4a	Adult	123	4781 (0.25/0.75)	27780 (0.24/0.76)
UCI.a5a	Adult	123	6414 (0.24/0.76)	26147 (0.24/0.76)
UCI.a6a	Adult	123	11220 (0.24/0.76)	21341 (0.24/0.76)
UCI.a7a	Adult	123	16100 (0.24/0.76)	16461 (0.24/0.76)
DB	English	4698	64 (0.45/0.55)	–
FADS	English	54877	4143 (0.51/0.49)	–
TDA	Arabic	7415	2000 (0.50/0.50)	–
PRABook	Chinese	74643	4000 (0.50/0.50)	–
PRADVD	Chinese	74638	4000 (0.50/0.50)	–
PRAMusic	Chinese	74638	4000 (0.50/0.50)	–
Banknote	Image	5	1372 (0.56/0.44)	–
Haberman	Medical	3	306 (0.26/0.74)	–
ILPD	Medical	10	583 (0.29/0.71)	–
QSAR	Chemicals	41	1055 (0.34/0.66)	–
Spect	Medical	22	80 (0.50/0.50)	187 (0.92/0.08)

patient classified into two categories: normal and abnormal [48].

The datasets (1) and (2) can be downloaded from the website of LibSVM[1]. The datasets (3) to (6) and (8) to (12) are from UCI[2]. Dataset (7) is from NLPCC 2013 cross lingual opinion analysis evaluation task. Datasets (4)–(7) are text data. The size, class ratio, types and dimensions of the datasets are listed Table 1.

To introduce class noise into the training set, we stochastically invert the binary labels of training samples with probability of 10%, 20%, and 30%. For datasets (1)–(3), we train the binary classification algorithm on the inverted noisy data. The algorithm is tuned on a development set before being used on the testing set. We use SVM$^{\text{light}}$[3] as the basic classifier for this experiment. We choose two kind of similarity measures, the cosine similarity for text data, and the Euclidean Distance based similarity for other data. The parameter k in the kNN graph is 5, set experimentally. For datasets (4)–(11), the experiment result is from a fivefolds cross-validation as they do not have separate testing data.

We compare the performance of ouralgorithm with other state-of-the-art learning algorithms for noisy data:

1. *LiC* [40]: a Perceptron based on-line learning method from our previous work ;
2. NHERD [49]: a widely used open source robust method;
3. ℓ_{log} [8] : a log loss method using the same loss function as LiC;
4. IMTD [9]: a method using [9];
5. CEWS [10]: a method using cut edge weight statistics. Furthermore, in order to see the necessity to use an ensemble or not, we also compare our methods with:
6. the original SVMs;
7. an ensemble SVMs without class noise handling;
8. an e-graph based method (two samples will be linked if the similarity between them is larger than e, including three different parameters, $e = 0.2, 0.1$ and 0.05);
9. Laplace distribution based CEWS;

Notice that ℓ_{log}, CEWS, and Laplace distribution based CEWS require prior knowledge of class-noise rate in training data. Thus, they are more appropriate for use as benchmarks rather than direct comparison to other algorithms, and should be discussed separately.

6.1.1 Comparison with other methods

Table 2 shows the performance of our proposed algorithms compared to the other state-of-the-art methods using the 12 datasets. Results show that our proposed algorithms outperform other algorithms in most of the cases. Both the macro average and the weighted micro average over the size of the

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.

[2] http://www.ics.uci.edu.

[3] http://svmlight.joachims.org.

Table 2 Performance of the 10 systems

Data	Noise	*AdaBC*	NHERD	ℓ^*_{log}	*Lic*	IMTD	CEWS*	E-graph (0.2)	E-graph (0.1)	E-graph (0.05)	Laplace*	SVMs	Boosting
LEU	10	**90.91**	81.62	**90.91**	87.88	78.79	73.53	58.82	73.53	76.47	73.53	73.53	73.53
	20	**87.88**	76.62	84.85	90.91	67.65	73.53	58.82	73.53	73.53	67.65	67.65	67.65
	30	**76.62**	58.68	57.58	78.79	58.82	58.82	58.82	58.82	67.64	58.82	58.82	58.82
SpliceUCI	10	**85.56**	72.14	83.99	84.67	83.42	83.15	85.28	85.05	85.37	83.07	83.15	83.15
	20	**83.44**	66.63	83.02	83.05	82.91	82.39	82.76	82.80	82.94	81.99	82.34	82.34
	30	**81.71**	61.14	78.31	79.53	80.83	73.54	79.54	79.53	79.54	73.65	75.68	78.34
UCI.a1a	10	**83.68**	81.23	83.33	83.40	83.40	82.86	82.93	83.17	83.48	83.01	82.87	83.20
	20	**83.49**	79.30	81.85	81.40	79.79	82.95	83.34	82.95	83.27	82.99	80.18	80.12
	30	**81.46**	74.78	77.10	77.30	78.39	81.31	79.11	78.30	78.30	81.46	79.65	78.30
UCI.a2a	10	**84.40**	82.25	83.92	84.24	83.79	83.71	84.19	84.10	84.11	84.00	83.85	83.79
	20	**83.53**	79.34	82.72	83.16	82.32	81.54	81.72	82.63	82.99	80.96	80.86	82.62
	30	**82.28**	74.13	76.22	80.66	76.58	81.05	76.50	77.19	77.19	81.09	77.79	77.19
UCI.a3a	10	84.06	82.79	83.93	**84.17**	83.96	83.07	83.91	83.97	83.83	83.22	83.70	83.60
	20	83.45	81.14	82.23	**83.49**	78.02	82.83	81.18	82.19	82.17	82.59	78.52	78.39
	30	**82.28**	77.52	80.23	81.02	79.28	81.35	76.91	78.30	78.30	81.53	77.25	78.31
UCI.a4a	10	84.19	83.48	84.05	**84.28**	83.95	83.37	84.33	84.31	84.24	83.02	84.13	83.95
	20	**84.17**	82.34	82.90	84.10	82.70	83.51	83.11	83.96	84.01	83.33	82.64	82.14
	30	**82.62**	80.05	81.54	81.81	78.09	82.31	78.66	79.06	78.06	82.12	78.31	78.06
UCI.a5a	10	**84.60**	83.57	83.78	84.45	84.03	83.87	84.47	84.23	84.22	83.64	84.40	84.12
	20	84.04	82.88	83.06	**84.20**	83.55	83.23	83.62	84.05	83.68	83.12	83.07	83.46
	30	82.12	80.42	74.71	79.20	80.30	**82.22**	79.45	79.52	78.23	82.27	79.49	79.52
UCI.a6a	10	84.64	83.97	83.20	**84.77**	83.83	84.44	84.50	84.56	84.61	84.05	84.46	83.75
	20	**84.04**	82.45	80.37	83.38	78.00	81.48	80.21	78.00	78.65	81.64	83.96	78.00
	30	**81.60**	80.22	77.83	81.83	78.00	78.00	78.36	78.00	77.99	78.00	77.99	78.00
UCI.a7a	10	84.96	84.58	82.93	**85.65**	84.45	83.14	84.66	84.75	84.81	82.99	84.45	84.64
	20	**84.53**	82.87	80.43	84.27	78.17	81.65	83.90	78.18	78.18	82.01	78.18	78.18
	30	83.26	80.28	78.67	**83.49**	79.33	80.86	79.33	80.33	79.63	81.13	79.33	77.35
DB	10	**91.66**	**91.66**	87.98	91.66	66.67	73.53	91.66	87.98	73.53	74.01	91.66	91.66
	20	58.33	75.00	84.50	**91.66**	58.33	67.65	58.33	58.33	58.33	67.65	58.33	58.33
	30	58.33	58.33	80.52	**81.82**	58.33	58.33	58.33	58.33	58.33	58.33	58.33	58.33
FADS	10	**89.55**	86.44	90.91	82.81	75.15	60.74	86.74	85.66	86.97	61.06	89.12	90.04
	20	**85.23**	81.27	81.82	84.38	73.47	53.78	82.81	81.43	84.36	53.66	83.61	84.99
	30	83.31	78.42	63.64	**85.58**	71.07	49.58	71.07	49.58	49.58	49.68	79.95	80.82
TDA	10	83.61	82.92	82.35	**84.06**	71.88	47.68	81.25	80.97	79.66	48.06	82.56	82.13
	20	79.46	76.81	77.70	**79.66**	65.28	45.97	76.53	75.45	62.33	45.68	75.41	73.82
	30	79.64	68.70	76.23	**77.94**	58.92	47.19	71.22	69.91	48.56	47.99	73.22	73.56

Table 2 (continued)

Data	Noise	*AdaBC*	NHERD	ℓ^*_{log}	*Lic*	IMTD	CEWS*	E-graph (0.2)	E-graph (0.1)	E-graph (0.05)	Laplace*	SVMs	Boosting
PRA book	10	**79.55**	75.65	78.02	77.72	63.38	64.36	78.43	78.26	78.00	65.24	76.91	77.36
	20	**78.06**	74.31	76.29	75.55	61.65	56.97	76.59	77.94	74.51	57.91	77.56	77.25
	30	**77.93**	69.52	76.97	77.03	59.43	70.04	77.03	76.21	73.22	73.33	74.31	72.51
PRA DV	10	78.93	79.21	79.55	**80.42**	69.24	80.20	79.21	79.06	74.69	79.99	78.03	78.09
	20	78.42	74.33	79.30	**79.43**	73.10	69.86	77.98	77.98	71.29	70.21	74.66	74.94
	30	77.24	70.16	77.55	**78.96**	62.39	71.98	78.00	76.92	69.55	71.78	74.31	72.76
PRA music	10	**80.65**	71.39	79.35	78.83	72.16	52.06	79.65	80.21	74.69	52.06	80.25	79.33
	20	79.36	77.27	76.25	**79.84**	68.30	52.45	78.99	79.22	75.44	52.45	74.69	74.76
	30	74.88	69.78	72.13	**75.23**	71.26	70.75	73.21	75.46	71.06	70.75	73.99	72.96
Banknote	10	**98.29**	93.25	90.07	89.38	94.54	97.95	97.95	98.29	98.29	97.95	98.29	98.29
	20	**98.29**	91.44	87.32	91.43	93.86	97.61	97.26	97.61	97.26	97.61	97.61	97.61
	30	**97.61**	83.21	76.38	80.13	91.47	97.61	92.74	97.61	97.61	97.61	97.61	97.61
Haberman	10	**79.81**	70.29	72.46	73.42	72.52	73.91	71.43	71.43	71.43	73.91	71.43	71.43
	20	**75.44**	69.71	73.91	74.21	70.96	72.46	69.71	69.71	66.53	72.46	71.43	71.43
	30	**77.69**	66.53	75.36	75.96	70.11	71.43	66.53	66.53	66.53	71.43	71.43	71.43
ILPD	10	**77.26**	53.77	64.35	73.26	74.25	74.25	50.98	50.98	50.98	74.69	50.98	72.55
	20	**75.25**	49.27	74.25	74.25	72.55	70.13	50.33	50.33	50.33	69.96	50.33	72.55
	30	**76.24**	48.51	62.37	75.24	66.45	69.14	46.21	50.33	50.33	70.04	46.21	72.55
QSAR	10	**74.89**	72.67	67.26	67.26	71.36	72.54	71.81	71.81	71.81	72.54	70.40	71.81
	20	**71.36**	69.35	67.26	67.26	70.91	69.51	66.96	66.96	66.96	69.51	66.96	66.96
	30	**66.96**	66.96	66.96	66.96	66.96	66.96	66.96	66.96	66.96	66.96	66.96	66.96
Spect	10	**94.86**	73.25	82.79	90.32	62.30	70.59	74.33	74.33	74.33	70.36	74.86	74.33
	20	**85.93**	80.98	83.87	94.08	56.68	77.01	70.05	71.42	72.33	76.44	77.01	70.05
	30	**81.49**	69.69	71.51	78.49	50.27	59.36	58.82	69.51	66.35	60.21	59.89	58.82
Macro-average	10	**84.68**	79.18	81.27	82.36	77.07	75.55	80.93	80.69	79.42	75.62	80.82	81.96
	20	**82.73**	76.66	79.95	82.04	74.24	73.31	77.12	76.90	75.55	73.27	76.60	77.15
	30	**79.41**	71.49	74.96	78.85	71.45	72.26	73.05	73.03	70.80	72.60	73.77	74.91
Micro-average	10	**84.23**	82.38	83.40	83.88	82.61	81.61	83.73	83.72	83.57	81.59	83.58	83.56
	20	**83.55**	80.62	81.69	82.96	79.52	80.24	81.99	81.70	81.48	80.16	80.44	80.11
	30	**81.87**	77.00	77.45	80.27	77.32	79.46	77.82	77.72	76.87	79.59	78.15	77.86

Bold values indicate the best results for each benchmark

*Means require prior distribution of class noise in the training process different class labels clearly show that our algorithm outperform all other methods. Note that ℓ_{log}, CEWS, and Laplace are provided with the class-noise rate. So the comparison to our method is not completely fair. Even with provided class-noise rate to ℓ_{log}, CEWS, and Laplace, they outperform *LiC* and *AdaBC* only for the relatively low class noise levels (10% and 20%) in two to three datasets only. This is because these methods require all samples to have the same class-noise rate for the probability weighting.

For the tiny training sets, such as LEU which has only 37 training samples and DB which has only 64 samples for training, the size effect is very prominent. The noisy data elimination based method does not work well in these data because the removal of noisy data also removed useful training data. This is particularly true when the class noise rate increases to 20% and 30%. Obviously, the high percentage of noise has a big effect on the training data. Most methods perform well on 10% class-noise rate but have large degradation in the 20% and the 30% class-noise rates. Different from these methods, *LiC* performs well and shows a significant advantage in these two sets of data at all three levels of class-noise rate. This is because when the training data is small, the size of the training data is more important than the quality, *LiC* does not remove any training data. Thus, it can make full use of the data for training. Due to its good class noise estimation, *LiC* also shows a better performance than ℓ_{log} even though ℓ_{log} does not remove any training data either.

When the training set grows, such as Splice which has 1000 samples or TDA which has 2000 samples, the quality of training samples becomes much more important. In these relatively large datasets, *AdaBC* shows better performance than the other methods. When the training set size becomes larger, the advantages of our methods become even more obviously. For example, the UCI.Adult at most 16,100 samples, or the review text from Amazon with 4000 samples. When the noisy level is at the 30% level, *AdaBC* can achieve a 5% higher accuracy than other methods which can be shown in the micro average of accuracy with 30% class noise.

We also compare the performance of SVMs and Boosting of SVMs on noisy data with other methods.When class noise rate is low, the class noise handing approach does not show significant performance gain compared to SVMs. But, as class noise increases, our proposed method and other class noise handling approach start to work and have marked improvements over SVMs [22]. have claimed that IMTD is cheap and easy toimplement.However,it is also likely to remove a substantialamount of data. CEWS has similar problem. When the class noise is 30%, which is at a high level, our proposed method is better than other methods.

Note that the *e*-graph based method and the Laplace based CEWS are also compared to our proposed methods. Since an *e*-graph does not need the ranking processing, it is more convenient than a knn graph. However, experimental results shows that the value of e should be different for different tasks. How to choose a reasonable *e* is an important issue. This is even more serious in our experiments because of the diversity of the datasets. By tuning *e* for each task, we did not get any better performance compared to *LiC* nor to *AdaBC*. This, again, shows that knn graph is a better choice because *k* is much easier to find. For the Laplace based CEWS, it uses the Laplace distribution and the Bernoulli distribution both are similar to Gaussian distribution. Since CEWS is a discretized result of these distributions, the performance of this method is quite similar to the original CEWS.

Generally speaking, *AdaBC* shows a more stable performance in most cases as shown in Table.IV. For the cases which *AdaBC* cannot achieve the best performance, they are at least as competitive as the other methods. Considering the performance across all datasets, our proposed *AdaBC* method has achieved the state-of-the-art performance.

6.2 Further discussion and experimental analysis

We further answer remaining questions: in this section. (1) “Can we use kNN directly?”; (2)“If the SRN distribution seems to be similar to the Gaussian distribution, what is the difference between them?”; (3) “Since the performance of *AdaBC* looks similar to *LiC*, is it necessary to use *AdaBC*?”; and (4)“Can we use the estimation method on the clean data?”.

6.2.1 kNN, SRN and Gaussian

In the estimation method proposed in section 4, kNN graph is to comparean individual sample with its most similar neighbors. kNN graph is used because it is sensitive to class noise [50, 51]. The Gaussian distribution based estimation method also used kNN graph [10, 12]. To see the differences, we conduct the following experiment to estimate the class noise rate for each individual sample by different methods on the UCI.Adult dataset. The parameter *k* is 5 in this experiment because a small *k* always leads to better performance. After the estimation, the mean value of the class noise rate for each disagreement in kNN is shown in Fig. 1. For the kNN method, we take the class noise rate equal to 0 when the number of different labels is less than 3, and the class noise rate is 1 otherwise. In Fig. 1a, the Gaussian based method is sensitive to class noise. When the number of different labels is 2, the probability of class noise estimation is near 95%. If this result is used as a probability, it will lead to an overrated class noise rate. Note that in Refs. [10, 12], the author takes the Gaussian distribution based method as a hypothesis testing method, and the threshold is indeed 0.95. If the individual sample achieves a higher p value,

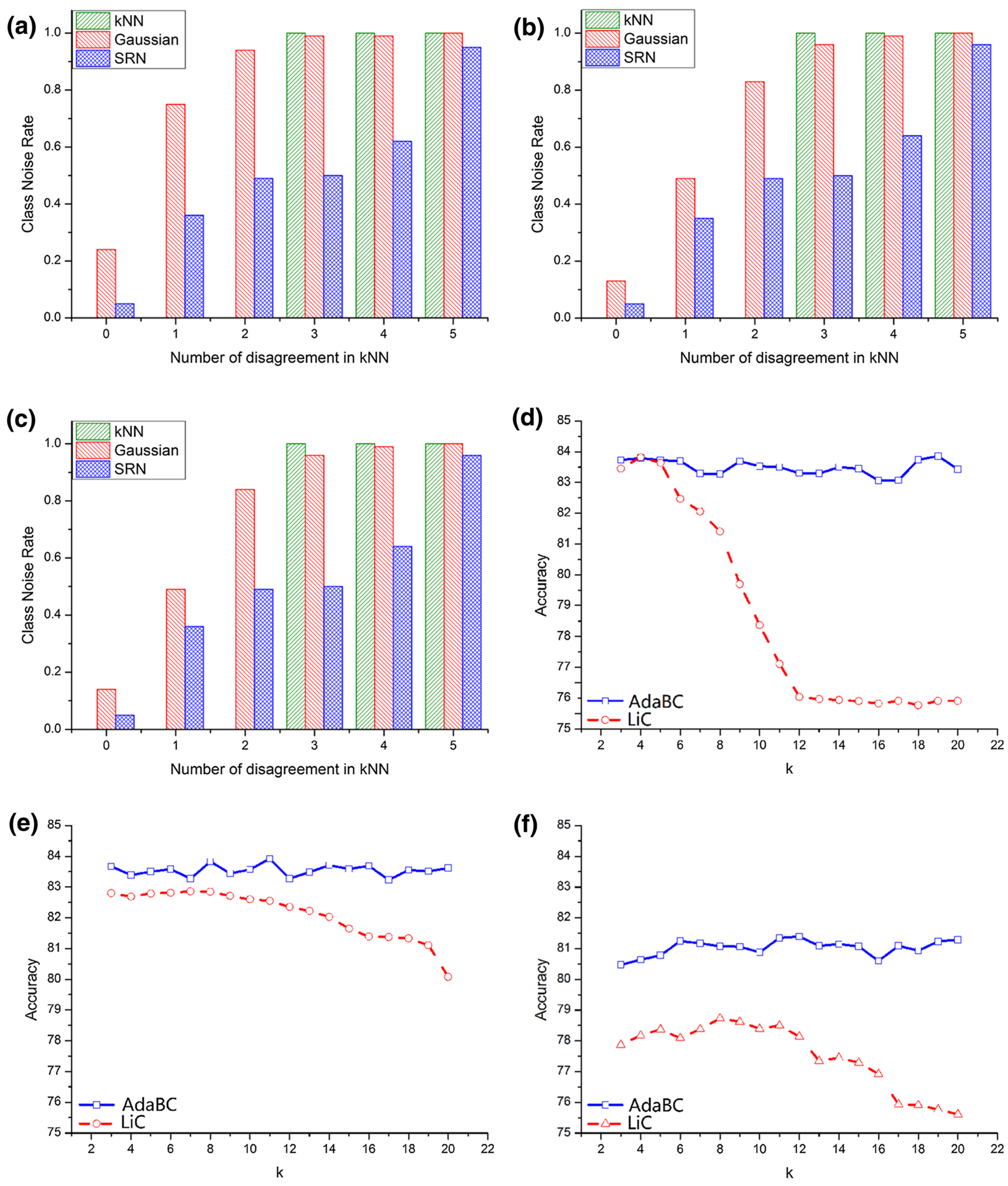

Fig. 1 Distribution of three estimation results (**a** 10% class noise, **b** 20% class noise, **c** 30% class noise) and Different k in the kNN graph for the two methods (**d** 10% class noise, **e** 20% class noise, **f** 30% class noise)

the individual sample will be considered as a noisy sample. We should note that, the threshold of an estimation method is the same as the experimental threshold for kNN. Due to the sensibility of kNN to class noise, this threshold is an overrated result for class noise rate. It will lead to misclassified noisy samples, which have the correct label. When the class noise increasing, the threshold of kNN and Gaussian distribution based method do not change. However, SRN is quite different. First, SRN will not over-rate class noise. It is obvious that even if there is no class noise, the number of disagreement labels can still be 3 when $k = 5$ in kNN graph. So SRN gives quite a reasonable estimation according to common sense. Another benefit is that, when the class-noise rate increases, SRN also gives a higher estimation of class-noise as indicated in Fig. 1a–c, with increasing noise levels.

From the experiment above, we can see that the estimation method proposed in this paper is more flexible. Only when most of neighbors have different labels, the individual sample is considered a noisy sample with high probability. If the training data has high probability of noise label, the number of disagreements should be high. In addition, comparing to kNN and Gaussian distribution, SRN gives a more reasonable estimation based on our analysis above. That is why the performance of our method is better than CEWS (Gaussian distribution based method) in the experiment.We do not compare SRN with the original kNN based method because the performance of kNN is similar to CEWS, and CEWS is a more sound method theoretically speaking.

6.2.2 *Lic* v.s *AdaBC* on different parameters

In the experiments of Sect. 4. C, *LiC* achieves better performance in most datasets than *AdaBC*. Then, why we need the AdaBoost based method, which seems to be more complicated? Figure 1 shows the set of performance evaluations of CUI. Adult using different k of the kNN graph under 10%, 20% and 30% class noise, respectively. Figure 1 shows very clearly that when k increases, the performance of *LiC* shows a sharp degradation. That is the gap between the noisy training data and clean training data revealed by the theoretical analysis. The performance of *LiC* is dependent on the estimation result. In fact, it is a well-known conclusion that the best k in kNN should be no larger than 5 (in most textbook of A.I or machine learning such as [52]), or the precision of this method will be limited and proven in Fig. 1 here again. When k is no larger than 5, *LiC* achieves the best performance among all k values.

In the Fig. 1d–f the gap between the two methods becomes larger when k increases, it also provides proof of our analysis. *AdaBC*, seems to be quite robust to k value.This is because even if the estimation is wrong, it still reveals some truth of the clean distribution, and the adaptive sampling method ensemble a series of weak classifiers into a strong classifier.The error boundary in Formula (9) also is affected by the class noise rate. The low class noise rate does less harm to the performance of the base classifier obviously. That is why *AdaBC* is robust to the k value.

6.2.3 Stability of *LiC* and *AdaBC*

Both *LiC* and *AdaBC* are iteration based learning algorithms. With the growth of iterations, the risk of overfitting on the training data becomes higher. With noisy training data, overfitting will certainly lead to misclassification directly. In the following set of experiments, we compare *LiC* and *AdaBC* with iterations as the variable to identify the stability of the two algorithms.

Figure 2a–c show the performance of *LiC* at noise level of 10%, 20%, and 30% on the UCI.Adult data, respectively. Since *LiC* is a perceptron based method, we also care about the effect of different learning rates. We take three different learning rate of 0.02, 0.01 and 0.005 in this experiment. Different learning rate can achieve similar top performance. It is also obvious that a smaller learning rate picks up performance slower, but it will outperform the higher learning rate after iteration 5. The performance gain with smaller learning rate is much more obvious when the noise level increases. In fact, for the 30% class noise case, there is a 4% gap between different learning rates.

Figure 2d–f show the boxplots of the respective noise levels for *AdaBC*. Since *AdaBC* is an AdaBoost based method, the experiment about *AdaBC* focuses on the mean value and variance of accuracy. In Fig. 2, the top performances of the two methods are similar. But *AdaBC* gives a more stable performance because the mean value of accuracy is in a similar level when iteration number increases. The variance is also small in the figure. However, the performance of *LiC* will peak at certain iteration number and then degrade because the accumulated noise will take its tolls on performance.

Now we can answer the question proposed at the beginning of this section. Even though the performances of the two methods are similar, we still have reason to use *AdaBC* because *LiC* needs to choose the optimal iteration number and learning rate to achieve top performance. The stability of *LiC* is also not as good as *AdaBC*. If the parameter is not suitable for a dataset, the performance degradation is obvious. Comparing to *LiC*, *AdaBC* shows a stable and robust result and that is the reason why we propose *AdaBC* although *AdaBC* may not be suited for small training datasets.

6.2.4 Estimating the noise on clean data

In practice, we do not know if the data contains noise or not. So an interesting question is that if we estimate the noise rate on the clean data, what will happen. In this experiment, we run both *LiC* and *AdaBC* on clean data. As a comparison,

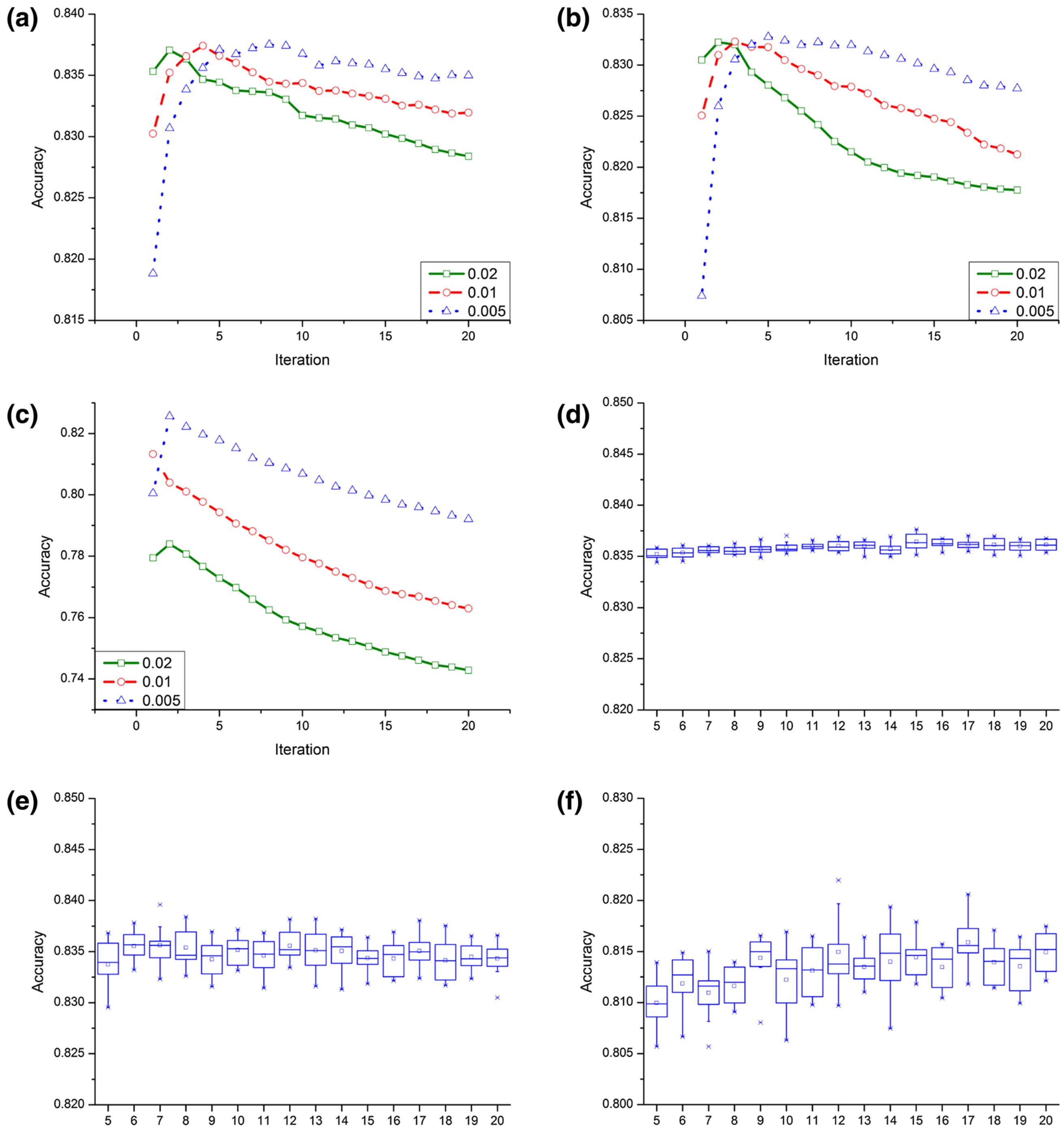

Fig. 2 Comparison of performance based on iterations (**a** *LiC*, 10% class noise; **b** *LiC*, 20% class noise; **c** *LiC*, 30% class noise; **d** *AdaBC*, 10% class noise; **e** *AdaBC*, 20% class noise; **d** *AdaBC*, 30% class noise)

a linear kernel SVMs is used as the baseline method. The result is show below in Table 3.

Note that *LiC* performs worse than the original SVMs on the clean dataset. The main reason is that, in the loss function of *LiC* given by Formula (6), the samples with high class noise rate will have a penalty. When the estimated class noise rate is higher than 50%, the weight of the penalty item will be larger than the original loss function item. It is actually an operation of label inversing on the training sample. Since *LiC* can introduce class noise into clean data this way, it performs worse than the original SVMs is reasonable. Different from *LiC*, *AdaBC* is an ensemble method. The basic classifier is still SVMs. The loss function is to estimate the error rate of the basic classifier and calculate the weight

Table 3 Performance on clean data

Data	*LiC*	*AdaBC*	SVMs	Data	*LiC*	*AdaBC*	SVMs
LEU	57.58	73.53	73.53	FADS	78.42	89.55	90.91
Splice	66.63	84.97	85.29	TDA	68.70	84.97	85.64
UCI.a1a	78.45	84.57	84.29	PRABook	72.03	79.98	80.13
UCI.a2a	77.21	84.15	84.57	PRADVD	69.96	81.25	81.63
UCI.a3a	77.96	83.96	84.51	PRAMusic	68.71	79.61	78.61
UCI.a4a	78.33	84.11	84.51	Banknote	83.21	97.95	97.95
UCI.a5a	78.41	84.27	84.39	Haberman	72.52	80.34	81.43
UCI.a6a	77.25	83.59	84.71	ILPD	74.25	81.25	82.55
UCI.a7a	78.00	84.66	84.80	QSAR	67.26	77.35	77.35
DB	58.33	91.66	91.66	Spect	77.00	92.34	97.64

based on this error rate. Usually, the error rate is less than 0.5. It means that each basic classifier will have a positive weight. So, the *AdaBC* algorithm becomes a bagging method of SVMs. Each basic classifier in the bagging has a weight, but the weight is meaningless since the training data is clean. Since each classifier trains on only part of the training set, the performance is no better than the original SVMs, but it is still comparable.

In conclusion, the estimation of class noise is used to weigh the samples in *LiC* but to weigh classifiers in *AdaBC*. In clean data, the incorrect weight on samples is much more harmful than the incorrect weight on classifiers. This is because the former leads to a miss-labeled sample, yet the later only introduces an incorrect weight. Fortunately, the weight is still with correct polarity and the training data of classifier is clean. That is the reason why AdaBC also works well on clean data.

6.3 Summary

Generally speaking, *AdaBC* achieves the best performance in the evaluation. However, by examining their performance in details in the experiments, we can see that *LiC* is sensitive to algorithm parameters including the k value in the kNN graph and the learning rate of the perceptron as well as the termination point of the algorithm, it cannot work on the clean data neither. *AdaBC* is better than *LiC* in this perspective. *AdaBC* is robust to all theabove parameters. Another advantage of *AdaBC* is the performance on clean data. Even the data does not contain noise, the *AdaBC* still perform well. Therefore, the Adaboost based improvement is necessary.

7 Conclusion and future work

In this paper, we present a novel class noise estimation method. We apply our estimation result into an Adaboost based algorithm to handle class noise. The algorithm is competitive compared to the state-of-the-art techniques and show superior performance on real datasets. We analyze the algorithm performance on different training dataset sizes and class-noise rates. Results confirm to the learning theorem provided in Eq. (8). In future works, we will investigate noise handling in semi-supervised tasks such as semi-supervised classification, transductive transfer learning, and also look into the domain adaptation problem. We will also consider different noise rate for different classes since label noise rates are often class label dependent in practice.

Acknowledgements This work was supported by the National Natural Science Foundation of China U1636103, 61632011, 61876053, EU-H2020 (Grant no. 794196), Key Technologies Research and Development Program of Shenzhen JSGG20170817140856618, Shenzhen Foundational Research Funding 20170307150024907.

References

1. Zhu X, Wu X (2004) Class noise vs. attribute noise: a quantitative study. Artif Intell Rev 22(3):177–210
2. Sáez JA, Galar M, Luengo J, Herrera F (2014) Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. Knowl Inf Syst 38(1):179–206
3. Joseph L, Gyorkos TW, Coupal L (1995) Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard. Am J Epidemiol 141(3):263–272
4. Cawthorn D-M, Steinman HA, Hoffman LC (2013) A high incidence of species substitution and mislabelling detected in meat products sold in South Africa. Food Control 32(2):440–449
5. Beigman E, Klebanov BB (2009) Learning with annotation noise. In: Proceedings of the joint conference of the 47th annual meeting of the acl and the 4th international joint conference on natural language processing of the AFNLP, vol. 1. Association for Computational Linguistics, pp 280–287

6. Du J, Gui L, He Y, Xu R, Wang X (2019) Convolution-based neural attention with applications to sentiment classification. In: IEEE Access
7. Gui L, Zhou Y, Xu R, He Y, Lu Q (2017) Learning representations from heterogeneous network for sentiment classification of product reviews. Knowl Based Syst 124:34–45
8. Natarajan N, Dhillon IS, Ravikumar PK, Tewari A (2013) Learning with noisy labels. In: Advances in neural information processing systems, pp 1196–1204
9. Brodley CE, Friedl MA (1999) Identifying mislabeled training data. J Artif Intell Res 11:131–167
10. Zighed DA, Lallich S, Muhlenbach F (2005) A statistical approach to class separability. Appl Stoch Models Bus Ind 21(2):187–197
11. Sluban B, Gamberger D, Lavra N (2010) Advances in class noise detection. In: Proceedings of the 2010 conference on ECAI 2010: 19th European conference on artificial intelligence. IOS Press, pp 1105–1106
12. Zhang M-L, Zhou Z-H (2011) Cotrade: confident co-training with data editing. IEEE Trans Syst Man Cybern Part B Cybern 41(6):1612–1626
13. Gui L, Xu R, Lu Q, Xu J, Xu J, Liu B, Wang X (2014) Cross-lingual opinion analysis via negative transfer detection. In: ACL (2), pp 860–865
14. Montgomery-Smith SJ (1990) The distribution of rademacher sums. Proc Am Math Soc 109(2):517–522
15. McDonald RA, Hand DJ, Eckley IA (2003) An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In: International workshop on multiple classifier systems. Springer, pp 35–44
16. Melville P, Shah N, Mihalkova L, Mooney RJ (2004) Experiments on ensembles with missing and noisy data. In: International workshop on multiple classifier systems. Springer, pp 293–302
17. Angluin D, Laird P (1988) Learning from noisy examples. Mach Learn 2(4):343–370
18. Hendrycks D, Mazeika M, Wilson D, Gimpel K (2018) Using trusteddata to train deep networks on labels corrupted by severe noise. In: Advances in neural information processing systems, pp 10456–10465
19. Ren M, Zeng W, Yang B, Urtasun R (2018) Learning to reweight examples for robust deep learning. arXiv preprint arXiv:1803.09050
20. Han B, Yao Q, Yu X, Niu G, Xu M, Hu W, Tsang I, Sugiyama M (2018) Co-teaching: robust training of deep neural networks with extremely noisy labels. In: Advances in neural information processing systems, pp 8527–8537
21. Zhang Z, Sabuncu M (2018) Generalized cross entropy loss for training deep neural networks with noisy labels. In: Advances in neural information processing systems, pp 8778–8788
22. Frénay B, Verleysen M (2014) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869
23. Heskes T (2000) The use of being stubborn and introspective. In: Prerational intelligence: adaptive behavior and intelligent systems without symbols and logic, volume 1, volume 2 prerational intelligence: interdisciplinary perspectives on the behavior of natural and artificial systems, vol 3. Springer, pp 1184–1200
24. Li Y, Wessels LF, de Ridder D, Reinders MJ (2007) Classification in the presence of class noise using a probabilistic kernel fisher method. Pattern Recognit 40(12):3349–3357
25. Scott C, Blanchard G, Handy G (2013) Classification with asymmetric label noise: consistency and maximal denoising. In: COLT, pp 489–511
26. Lawrence ND, Schölkopf B (2001) Estimating a kernel fisher discriminant in the presence of label noise. In: ICML, vol 1. Citeseer, pp 306–313
27. Pérez CJ, González-Torre FJG, Martín J, Ruiz M, Rojano C (2007) Misclassified multinomial data: a bayesian approach. Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A Matemáticas (RACSAM) 101(1):71–80
28. Klebanov BB, Beigman E (2009) From annotator agreement to noise models. Comput Linguist 35(4):495–503
29. Kolcz A, Cormack GV (2009) Genre-based decomposition of email class noise. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 427–436
30. Zhu X, Wu X, Chen Q (2003) Eliminating class noise in large datasets. In: ICML, vol 3, pp 920–927
31. Chen K, Guan D, Yuan W, Li B, Khattak AM, Alfandi O (2018) A novel feature selection-based sequential ensemble learning method for class noise detection in high-dimensional data. In: International conference on advanced data mining and applications. Springer, pp 55–65
32. Jiang Y, Zhou Z-H (2004) Editing training data for kNN classifiers with neural network ensemble. In: International symposium on neural networks. Springer, pp 356–361
33. Oza NC (2003) Boosting with averaged weight vectors. In: International workshop on multiple classifier systems. Springer, pp 15–24
34. Oza NC (2004) Aveboost2: boosting for noisy data. In: International workshop on multiple classifier systems. Springer, pp 31–40
35. Rätsch G, Schölkopf B, Smola AJ, Mika S, Onoda T, Müller K-R (2000) Robust ensemble learning for data mining. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 341–344
36. Freund Y, Schapire RE (1995) A desicion-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory. Springer, pp 23–37
37. Gao Y, Gao F, Guan X (2010) Improved boosting algorithm with adaptive filtration. In: Intelligent control and automation (WCICA), 2010 8th world congress on. IEEE, pp 3173–3178
38. Wheway V (2000) Using boosting to detect noisy data. In: Pacific rim international conference on artificial intelligence. Springer, pp 123–130
39. Breiman L (1997) Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, Technical Report
40. Gui L, Lu Q, Xu R, Li M, Wei Q (2015) A novel class noise estimation method and application in classification. In: Proceedings of the 24th ACM international on conference on information and knowledge management. ACM, pp 1081–1090
41. Bennett C, Sharpley RC (1988) Interpolation of operators, vol 129. Academic Press, Cambridge
42. Li H (1982) Method of statistical learning. Prentice hall, Upper Saddle River
43. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
44. Platt JC (1999) 12 fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods, pp 185–208
45. Haberman SJ (1976) Generalized residuals for log-linear models. In: Proceedings of the 9th international biometrics conference, pp 104–122

46. Ramana BV, Babu MSP, Venkateswarlu N (2012) A critical comparative study of liver patients from usa and india: an exploratory analysis. Int J Comput Sci Issues 9(2):506–516
47. Mansouri K, Ringsted T, Ballabio D, Todeschini R, Consonni V (2013) Quantitative structure-activity relationship models for ready biodegradability of chemicals. J Chem Inf Model 53(4):867–878
48. Kurgan LA, Cios KJ, Tadeusiewicz R, Ogiela M, Goodenday LS (2001) Knowledge discovery approach to automated cardiac spect diagnosis. Artif Intell Med 23(2):149–169
49. Crammer K, Lee DD (2010) Learning via Gaussian herding. In: Advances in neural information processing systems, pp 451–459
50. Devijver PA, Kittler J (1982) Pattern recognition: a statistical approach. Prentice Hall, Upper Saddle River
51. Wilson DR, Martinez TR (1997) Instance pruning techniques. In: ICML, vol 97, pp 403–411
52. Russell S, Norvig P, Intelligence A (1995) A modern approach. Artificial Intelligence, vol 25. Prentice-Hall, Egnlewood Cliffs