Data Mining and Knowledge Discovery (2019) 33:1468–1504
https://doi.org/10.1007/s10618-018-0592-8

Classification with label noise: a Markov chain sampling framework

Zijin Zhao[1] · Lingyang Chu[1] · Dacheng Tao[2] · Jian Pei[1]

Received: 2 January 2018 / Accepted: 27 September 2018 / Published online: 6 October 2018

Abstract

The effectiveness of classification methods relies largely on the correctness of instance labels. In real applications, however, the labels of instances are often not highly reliable due to the presence of label noise. Training effective classifiers in the presence of label noise is a challenging task that enjoys many real-world applications. In this paper, we propose a Markov chain sampling (MCS) framework that accurately identifies mislabeled instances and robustly learns effective classifiers. MCS builds a Markov chain where each state uniquely represents a set of randomly sampled instances. We show that the Markov chain has a unique stationary distribution, which puts much larger probability weights on the states dominated by correctly labeled instances than the states dominated by mislabeled instances. We propose a Markov Chain Monte Carlo sampling algorithm to approximate the stationary distribution, which is further used to compute the mislabeling probability for each instance, and train noise-resistant classifiers. The MCS framework is highly compatible with a wide spectrum of classifiers that produce probabilistic classification results. Extensive experiments on both real and synthetic data sets demonstrate the superior effectiveness and efficiency of the proposed MCS framework.

Keywords Label noise · Classification · Markov Chain Monte Carlo · Sampling framework

Responsible editor: Jesse Davis, Elisa Fromont, Derek Greene, Bjorn Bringmann.

Lingyang Chu and Zijin Zhao contribute equally in this work.

✉ Lingyang Chu
lca117@sfu.ca

Extended author information available on the last page of the article

1 Introduction

The good performance of a classification method crucially relies on the correctness of the labeled training data. Unfortunately, in real-world data the labels of instances are usually corrupted by random noise, which largely limits the performance of conventional machine learning methods. Training effective classifiers in the presence of label noise is a challenging and urgent task of great practical importance.

The problem of *learning with label noise* was first proved PAC learnable by Valiant (1984) and Angluin and Laird (1988). Many related studies have been developed since then. The most straightforward idea is to comprehensively analyze the noise-tolerating performance of existing classification algorithms (Bartlett et al. 2006; Frénay and Verleysen 2014; Manwani and Sastry 2013), and select the ones that are relatively robust against label noise. Nettleton et al. (2010) compared the effect of different types of noise on conventional classification methods, and found that the Naïve Bayes classifier is the most robust method against noise. However, according to the comprehensive study by Frénay and Verleysen (2014), the performance of most conventional classification methods is not robust to label noise.

Some studies attempt to remove mislabeled instances and clean corrupted data using the classification result of conventional classification algorithms (Delany and Cunningham 2004; Wilson 1972; Cover and Hart 1967; Belur 1991). For example, Thongkam et al. (2008) removed the instances misclassified by the support vector machine trained on all the training data. Wilson (1972) proposed a kNN-based method to filter out every instance whose label mismatches the majority labels of its neighbours. The performance of such methods highly depends on the noise-tolerance of the employed conventional classifier, whose classification result may be unreliable in the presence of label noise (Frénay and Verleysen 2014).

There are also many methods that improve the classification performance by customizing conventional classification models with additional assumptions on noise distribution (Biggio et al. 2011; Lawrence and Schölkopf 2001; Yang et al. 2012; Liu and Tao 2016). For example, Biggio et al. (2011) improved the robustness of support vector machine by a kernel matrix correction method, which is based on the assumption that the label of each training instance is independently flipped with a fixed probability across the whole training data set. Yang et al. (2012) assumed an expected mislabelling probability of every training instance, and applied stochastic programming to improve the noise-tolerance of multiple kernel learning algorithms. By assuming a class-conditional noise distribution, Liu and Tao (2016) developed a noise-robust classifier based on density ratio estimation, which relies heavily on the feature dimensionality and the width of estimation kernel. By assuming label noise is asymmetric and class conditional, Scott et al. (2013) proposed a universally consistent estimator of the maximal proportion of overlap between different class-conditional distributions. Those methods achieve good noise-tolerating performance by making strong assumptions on noise distribution. However, the good performance of these methods is of little comfort when the assumptions of their methods do not apply to the complex label noise in real-world scenarios.

Can we build a more general noise-resistant classification model without assuming any specific noise distribution? In this paper, we provide an affirmative answer. Our

approach follows a simple yet reasonable intuition. Correctly labeled data follows a consistent distribution. Noise (that is, mislabeled data) is caused by diverse and inconsistent mistakes and thus does not follow a consistent distribution. Therefore, a model trained on a sample may fit the correctly labeled data in another sample well, but may not fit the mislabeled noise data in another sample. By training a set of models on a set of selected samples of data, we can infer the likelihood of a data instance being correctly labeled by the proportion of the trained models that fit the data instance. Importantly, this natural difference between noise data and clean data does not rely on any specific data distribution, and can be used to effectively identify noise data. Therefore, we can build a more general noise-resistant classification model without assuming any specific distribution of the noise data instances.

In this paper, we tackle the challenging problem of classification in the presence of label noise without assuming any specific data distribution. Our only assumption is that all clean data is sampled from the same unknown distribution while the mislabeled data is mislabeled in various ways. In other words, we assume that various mistakes may lead to mislabeled instances. We propose a Markov Chain Sampling (MCS) framework that trains a set of classifiers on different sets of randomly sampled training instances. By assembling the probabilistic outputs of the set of classifiers, MCS learns the mislabeling probability of every instance, which is further applied for label correction and noise-tolerant classifier training.

We make the following contributions.

First, we design a Markov chain to assemble the probabilistic outputs of a set of classifiers. Each state of the Markov chain uniquely corresponds to a set of randomly sampled training instances. We train a classifier for each state using the corresponding training instances. The probabilistic outputs of the classifiers are used to compute the transition probabilities of the Markov chain. Any classification model that produces probabilistic classification results can be smoothly incorporated in the Markov chain. This makes MCS compatible with a wide spectrum of conventional classification algorithms.

Second, we show that the proposed Markov chain has a unique stationary distribution, which puts much larger probability weights on the states dominated by correctly labeled instances than the states dominated by mislabeled instances. Based on the stationary distribution, we further design an algorithm to compute the mislabeling probability for each instance, and train noise-resistant classifiers.

Third, we design three methods to train noise-tolerating classifiers using the mislabeling probability of each instance in different ways. The first method trains the final classifier after cleaning the data by removing the mislabeled data instances. The second method trains the final classifier after cleaning the data by flipping the labels of mislabeled data instances. The third method directly uses the mislabeling probability as supervision information to train the final classification model. All methods achieve superior classification performance under different types of label noise.

Last, we report an extensive experimental study on both synthetic and real-world benchmark data sets. The results clearly show that the MCS framework achieves good classification accuracy in the presence of label noise.

The rest of the paper is organized as follows. In Sect. 2, we define the problem and review related works on classification with label noise. In Sect. 3, we introduce

the MCS framework and prove the correctness and effectiveness of MCS. We present comprehensive experimental studies in Sect. 4 and conclude the paper in Sect. 5.

2 Problem definition and related work

In this section, we first define the problem of classification with label noise. Then, we review the related work.

2.1 The problem of classification with label noise

Let D be the distribution of a pair of random variables $(X, Y) \in \mathcal{X} \times \{+1, -1\}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is a d-dimensional feature space, $X \in \mathcal{X}$ is a *feature* and $Y \in \{+1, -1\}$ is the *label* of X. Denote by $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ a sample of n *instances* drawn i.i.d. from D. For each instance $(x_i, y_i) \in S$, y_i is the *true label* of x_i. Here, a *true label* is the ground truth label that is not affected by any label noise.

In many real-world applications, the observed labels of instances may be corrupted by noise. We model this by a sample of *observed instances* $\hat{S} = \{(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n)\}$, where $\hat{y}_i$ is the *observed label* of instance (x_i, y_i). An instance $(x_i, \hat{y}_i) \in \hat{S}$ is called a *positive instance* if $y_i = +1$, a *negative instance* if $y_i = -1$, a *correctly labeled* instance if $y_i = \hat{y}_i$, and a *mislabeled* instance if $y_i \neq \hat{y}_i$.

Denote by $\hat{S}^+ = \{(x_i, \hat{y}_i) \in \hat{S} \mid y_i = 1\}$ and $\hat{S}^- = \{(x_i, \hat{y}_i) \in \hat{S} \mid y_i = -1\}$ the set of positive instances and the set of negative instances in $\hat{S}$, respectively. Denote by $\rho^+ = \frac{|\{(x_i, \hat{y}_i) \in \hat{S}^+ | \hat{y}_i = -1\}|}{|\hat{S}^+|}$ and $\rho^- = \frac{|\{(x_i, \hat{y}_i) \in \hat{S}^- | \hat{y}_i = 1\}|}{|\hat{S}^-|}$ the proportions of mislabeled instances in $\hat{S}^+$ and $\hat{S}^-$, respectively. Similar to the typical tasks of classification with label noise (Liu and Tao 2016; Natarajan et al. 2013), we assume that the proportion of mislabeled instances is less than or equal to 50%, that is, $\rho^+ \leq 0.5$ and $\rho^- \leq 0.5$. Please note that by assuming the two proportion rates at least 0.5, we do not assume we have the values of ρ^+ or ρ^-. Indeed, we do not assume we can know $\hat{S}^+$ or $\hat{S}^-$ exactly.

Now, we define the problem of Classification with Label Noise (CLN).

Problem 1 (*Classification with Label Noise*) Given a set of observed instances $\hat{S} = \{(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n)\}$ that satisfies $\rho^+ \leq 0.5$ and $\rho^- \leq 0.5$, the problem of classification with label noise (CLN for short) is to train a classification model $\mathcal{M}^*$ that predicts the true label of any feature $x \in \mathcal{X}$.

The CLN problem is challenging, since little prior knowledge on the noise distribution is available and any instance in $\hat{S}$ may be mislabeled.

2.2 Related work

Classification with label noise is a challenging task that attracts much research attention. The existing studies can be grouped into three major categories: the *robust model methods*, the *data cleaning methods*, the *noise-tolerant methods* and the *crowdsourcing methods*. We review the four categories briefly as follows.

2.2.1 Robust model methods

The robust model methods comprehensively study the robustness of different classification models in the presence of label noise, and select the model with the best noise-tolerating performance (Manwani and Sastry 2013; Nettleton et al. 2010; Folleco et al. 2008).

Manwani and Sastry (2013) analyzed the noise-tolerating performance of the empirical risk minimization framework with different loss functions for the binary classification problem. According to their study, conventional loss functions, such as log loss and hinge loss, do not perform well with corrupted instance labels. The 0-1 loss achieves the best noise-tolerating performance, however, it is non-convex and intractable in practice. Ghosh et al. (2015) assumed the noise-free data to be separable, and proved a sufficient condition for risk minimization loss functions, such as sigmoid loss, ramp loss and probit loss, to be tolerant to random label noise. Patrini et al. (2016) proved that stochastic gradient descent and proximal methods can be adapted with minimal effort to train robust classifiers using asymmetric noisy labels. They also showed that most model losses are robust to a data-dependent label noise. Nettleton et al. (2010) systematically compared the effect of label noise and feature noise on four different classification models, namely the Naïve Bayes model (John and Langley 1995), the C4.5 algorithm (Quinlan 2014), the IBK instance-based model (Aha et al. 1991) and the Support Vector Machine (SVM) (Vapnik 2013). They found that the Naïve Bayes model is most robust against noise. Long and Servedio (2008) studied the effect of random classification noise on a broad class of boosting algorithms. They showed that the widely used boosting methods, such as AdaBoost and LogitBoost, are highly susceptible to random classification noise. Folleco et al. (2008) showed that imbalanced or skewed data sets may aggravate the influence of label noise. They studied the robustness performance of eleven different commonly used classifiers in imbalanced noisy data sets. According to their experiment results, all classifiers are affected by label noise to some extent, and the Random Forest method (Breiman 2001) achieves the best noise-tolerating performance.

It is worthwhile to comprehensively study the noise-tolerating performance of existing conventional classification methods. We can conclude from these studies that most conventional classification methods are not robust to label noise (Frénay and Verleysen 2014).

2.2.2 Data cleaning methods

The data cleaning methods improve the classification performance of conventional classification models by training classifiers on the data that is cleaned by filtering out or relabelling the mislabeled data instances (Khoshgoftaar and Rebours 2004; Thongkam et al. 2008; Cover and Hart 1967; Belur 1991; Wilson 1972; Guyon et al. 1994). There are three sub-groups in this category, namely the classification filtering methods, the kNN based methods, and the threshold-based methods.

The classification filtering methods (Frénay and Verleysen 2014; Gamberger et al. 1999; Khoshgoftaar and Rebours 2004) filter out the instances misclassified by a classifier that is trained on label-corrupted data. Thongkam et al. (2008) trained a

SVM on the label-corrupted data and removed the instances misclassified by the SVM. Miranda et al. (2009) extended the work of Thongkam et al. (2008) by employing four classifiers to vote for the mislabeled instances. However, filtering out all the instances that are misclassified by a single classifier is rigid and risky, since a single classifier learned from the entire set of label-corrupted data is usually unreliable (Frénay and Verleysen 2014). The proposed MCS method also employs the classifiers trained on label-corrupted data to identify mislabeled instances. However, unlike the conventional classification filtering methods, the proposed MCS method trains a set of classifiers on selected samples of the original label-corrupted data, and learns a more reliable mislabeling probability of every instance by using a Markov chain to assemble the probabilistic outputs of the set of classifiers.

The kNN-based methods use a kNN classifier enhanced by noise-tolerating rules to remove or relabel misclassified instances. Delany and Cunningham (2004) proposed the Blame-Based Noise Reduction method to filter out the instances that cause the misclassification of the other instances. Wilson (1972) proposed to filter out the instances whose labels mismatch the majority label of their neighbours. Wilson and Martinez (1997, 2000) comprehensively investigated the noise-tolerating performance of kNN-based data filtering methods. Most kNN-based methods are heuristic whose performance are significantly affected by the selection of the parameter k.

The threshold-based methods (Sun et al. 2007; Gamberger et al. 1996; Gamberger and Lavrač 1996) calculate a (heuristic) mislabeling probability score for each instance and filter out the instances whose score pass a certain threshold. For such methods, it is often difficult to find a proper threshold that effectively distinguishes true exceptions from mislabeled instances (Frénay and Verleysen 2014).

2.2.3 Noise-tolerant methods

The noise-tolerant methods (Lawrence and Schölkopf 2001; Liu and Tao 2016; Natarajan et al. 2013; Stempfel and Ralaivola 2009) build inherently noise-tolerant classifiers by modeling the distribution of label noise before or during model training.

Lawrence and Schölkopf (2001) proposed KLDA to model the generating process of class-conditional label noise as one component of a generative classification model. By extending the conventional noise-free SVM functional to a tailored nonconvex functional, Stempfel and Ralaivola (2009) developed SLOPPYSVM to learn robust SVM classifiers in the presence of class-conditional label noise. Natarajan et al. (2013) studied the class-dependent random label noise in the binary classification problem. They proposed two methods that modify any given surrogate loss function to make the classifier more robust against label noise. Liu and Tao (2016) studied the same type of label noise and applied density ratio estimation to estimate the noise rate and improve classification performance. As claimed by Liu and Tao (2016), however, the classification performance of their method relies crucially on the accuracy of density ratio estimation, which may be affected by high feature dimensionality and inappropriate estimation kernels.

In summary, the good classification performance of noise-tolerant methods is usually based on a complex classification model that is supported by strong model

assumptions. A high model complexity increases the chance of over-fitting. The strong assumptions often limit the ability of generalization in dealing with complex label noise in real-world scenarios.

2.2.4 Crowdsourcing methods

In terms of identifying true labels of instances, our work is also related to a conventional crowdsourcing task, that is, to find the true label of an instance by collecting multiple labels with label noise (Li 2015; Vaughan 2018; Sheng et al. 2008).

Sheng et al. (2008) developed a majority voting method and studied its effectiveness on improving the quality of labels that are repeatedly collected from unreliable labelers. However, as claimed by Karger et al. (2011), majority voting is error-prone and inefficient because it treats the labels provided by each labeler equally. To tackle this problem, many effective improvements were developed to jointly estimate the reliabilities of labelers and the labels of instances (Li 2015; Whitehill et al. 2009; Liu et al. 2012; Zhou et al. 2012; Raykar et al. 2010).

The above crowdsourcing methods are effective in finding the true label of an instance when the instance is associated with multiple labels. However, these methods cannot be straightforwardly extended to train a robust classifier in the presence of label noise, because, for the task of classification with label noise, each instance is only associated with a single unreliable label.

3 The Markov chain sampling framework

In this section, we introduce the Markov Chain Sampling (MCS) framework to solve the CLN problem. The key idea is to identify the mislabeled instances in $\hat{S}$ by assembling the outputs of a set of classifiers. The set of classifiers are trained on different subsets of $\hat{S}$ that are obtained by sampling a carefully designed Markov chain with random walk. Each state of the Markov chain uniquely corresponds to one of the $2^{|\hat{S}|} - 1$ non-empty subsets of instances of $\hat{S}$.

In the following, we first introduce the structure of the Markov chain and prove that the proposed Markov chain has a unique stationary distribution. Then, we prove that the stationary distribution is effective in identifying the mislabeled instances and present three methods to train the classification model $\mathcal{M}^*$. Finally, we introduce a MCMC sampling algorithm to efficiently approximate the stationary distribution. Table 1 shows the frequently used notations.

3.1 The structure of the Markov chain

We consider a discrete time Markov chain, denoted by C, that consists of the $2^{|\hat{S}|} - 1$ discrete states. Each state is uniquely associated with one of the $2^{|\hat{S}|} - 1$ non-empty subsets of $\hat{S}$. We assign a unique index to each non-empty subset of $\hat{S}$ and denote the ith subset by $\hat{S}_i \subseteq \hat{S}$. The state uniquely associated with $\hat{S}_i$ is denoted by $\mathbf{c}_i \in C$, which is a binary indicator vector with $|\hat{S}|$ dimensions. The kth dimension of $\mathbf{c}_i$, denoted by

Table 1 Frequently used notations

Notation	Description
$\lvert\cdot\rvert$	The set volume operator
$\hat{S}$	The set of observed instances
C	The discrete time Markov chain
$\mathbf{c}_i$	The ith state of the Markov chain C
$\hat{S}_i$	The subset of observed instances associated with state $\mathbf{c}_i$
$\mathcal{M}_i$	The ith model that is trained on $\hat{S}_i$
$\mathcal{M}^*$	The final classifier of the MCS framework
P	The transition matrix of the Markov chain C

$\mathbf{c}_{ik}$ in Eq. 1, indicates whether the kth instance $(x_k, \hat{y}_k)$ is contained by the ith subset of instances $\hat{S}_i$. We can write $\hat{S}_i = \{(x_k, \hat{y}_k) \in \hat{S} \mid \mathbf{c}_{ik} = 1\}$.

$$\mathbf{c}_{ik} = \begin{cases} 1, & \text{if } (x_k, \hat{y}_k) \in \hat{S}_i \\ 0, & \text{if } (x_k, \hat{y}_k) \notin \hat{S}_i \end{cases} \tag{1}$$

For two states $\mathbf{c}_i$ and $\mathbf{c}_j$, a transition from $\mathbf{c}_i$ to $\mathbf{c}_j$ indicates a transition from $\hat{S}_i$ to $\hat{S}_j$. We perform such transition in two steps. First, we train a conventional classification model $\mathcal{M}_i$ on $\hat{S}_i$, and use $\mathcal{M}_i$ to compute the probability of predicted label for every instance in $\hat{S}$. Second, we obtain $\hat{S}_j$ by sampling instances from $\hat{S}$ according to the probabilities of predicted labels computed by $\mathcal{M}_i$.

Conventional models that produce positive probabilities of predicted labels can all be used to train $\mathcal{M}_i$. In this paper, we use Kernelized Support Vector Machine (KSVM) (Scholkopf and Smola 2001) with Platt scaling and Logistic Regression (LR) (Hosmer Jr et al. 2013) to demonstrate the feasibility and effectiveness.

The details of the sampling method are as follows.

Given an instance $(x_k, \hat{y}_k)$, let $\tilde{y}_{ik} \in \{+1, -1\}$ represent the predicted label of feature $x_k \in \mathcal{X}$ that is computed by model $\mathcal{M}_i$. If $\tilde{y}_{ik} = \hat{y}_k$, we say model $\mathcal{M}_i$ **fits** $(x_k, \hat{y}_k)$; if $\tilde{y}_{ik} \neq \hat{y}_k$, we say $\mathcal{M}_i$ **does not fit** $(x_k, \hat{y}_k)$.

Denote by $\mathbb{P}(\tilde{y}_{ik} \mid x_k, \mathbf{c}_i)$ the probability that model $\mathcal{M}_i$ predicts label $\tilde{y}_{ik}$ for feature x_k. For each instance $(x_k, \hat{y}_k)$ in $\hat{S}_i$, we define $\mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$ as the probability that $\mathcal{M}_i$ fits $(x_k, \hat{y}_k)$.

Then, we obtain $\hat{S}_j$ by sampling each instance $(x_k, \hat{y}_k) \in \hat{S}$ with probability $\mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$. That is, $\mathbb{P}((x_k, \hat{y}_k) \in \hat{S}_j \mid \mathbf{c}_i) = \mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$.

In this way, if $\mathcal{M}_i$ fits $(x_k, \hat{y}_k)$, $(x_k, \hat{y}_k)$ is sampled with probability $\mathbb{P}(\tilde{y}_{ik} \mid x_k, \mathbf{c}_i)$; otherwise, it is sampled with probability $1 - \mathbb{P}(\tilde{y}_{ik} \mid x_k, \mathbf{c}_i)$. Since $\mathbb{P}(\tilde{y}_{ik} \mid x_k, \mathbf{c}_i) > 1 - \mathbb{P}(\tilde{y}_{ik} \mid x_k, \mathbf{c}_i)$, the instance that $\mathcal{M}_i$ fits has a higher probability to be sampled than the instance that $\mathcal{M}_i$ does not fit.

The intuition of the above sampling process is that, since $\mathcal{M}_i$ is trained using all instances in $\hat{S}_i$, $\mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$ can be regarded as the belief that $(x_k, \hat{y}_k)$ is correctly labeled. This belief is **supported** by all instances in $\hat{S}_i$.

However, since $\hat{S}_i$ may contain some mislabeled instances, the belief $\mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$ may not be fully reliable. To tackle this problem, we embed $\mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$ into a carefully designed Markov chain and prove in Sect. 3.2 that the stationary distribution of the Markov chain is reliable and effective in identifying mislabeled instances.

Next, we define the transition probability $\mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)$ of the Markov chain C.

Recall that $\mathbf{c}_{jk} = 1$ when $(x_k, \hat{y}_k) \in \hat{S}_j$, it follows that $\mathbb{P}(\mathbf{c}_{jk} = 1 \mid \mathbf{c}_i) = \mathbb{P}(\tilde{y}_{ik} = \hat{y}_k \mid x_k, \mathbf{c}_i)$. This means $\mathbf{c}_{jk}$ is a random variable following the Bernoulli distribution $\mathbf{c}_{jk} \sim Bern(\mathbb{P}(\mathbf{c}_{jk} = 1 \mid \mathbf{c}_i))$, and $\mathbf{c}_j$ is the ensemble of $|\hat{S}|$ independent random variables that follow Bernoulli distributions. The transition probability from $\mathbf{c}_i$ to $\mathbf{c}_j$ is

$$\mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i) = \prod_{k=1}^{|\hat{S}|} \mathbb{P}(\mathbf{c}_{jk} = 1 \mid \mathbf{c}_i) \tag{2}$$

So far, we have defined the discrete states of C and the transition probability between each pair of states in C. The *transition matrix* of C, denoted by P, is a square matrix, where the entry at the ith row and the jth column is $P_{ij} = \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)$.

Next, we show that the proposed Markov chain C has a unique stationary distribution.

Theorem 1 *The proposed Discrete Time Markov Chain C has a unique stationary distribution $\boldsymbol{\pi}$.*

Proof We prove this by showing that the proposed Markov chain C is *aperiodic* and *positive recurrent*.

(Aperiodic) Since model $\mathcal{M}_i$ produces positive probabilities for predicted labels, $P_{ii} > 0$ for all i. Therefore, every state in C has a self-loop. Thus C is aperiodic.

(Positive recurrent) By Definition 3.1 in (Gilks et al. 1995), since $P_{ij} > 0$ for all i and j, the proposed Markov chain C is *irreducible*. According to Lemma 6.3.5 in (Grimmett and Stirzaker 2001), since C is irreducible and has a finite number of states, C is a positive recurrent Markov chain. According to Theorem 3.1 in (Gilks et al. 1995), every aperiodic and positive recurrent Markov chain has one unique stationary distribution $\boldsymbol{\pi}$. □

3.2 Why does stationary distribution work?

In this section, we prove that the stationary distribution $\boldsymbol{\pi}$ of C is effective in identifying the mislabeled instances in $\hat{S}$.

Without loss of generality, we divide the states in the Markov chain C into two subsets. The set of *good states*, denoted by C^g, is the set of states dominated by correctly labeled instances. The set of *bad states*, denoted by C^b, is the set of states dominated by mislabeled instances.

Denote by $\mathbb{P}(C^g \mid \mathbf{c}_i) = \sum_{j:\mathbf{c}_j \in C^g} \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)$ the probability of jumping from a state $\mathbf{c}_i \in C$ to a good state, and by $\mathbb{P}(C^b \mid \mathbf{c}_i) = \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)$ the probability of jumping from $\mathbf{c}_i \in C$ to a bad state.

For a good state $\mathbf{c}_i \in C^g$, since most correctly labeled instances consistently follow the same distribution, the corresponding model $\mathcal{M}_i$ has a high probability to fit most of the correctly labeled instances in $\hat{S}$. Recall that an instance that $\mathcal{M}_i$ fits has a higher probability to be sampled than an instance that $\mathcal{M}_i$ does not fit, the correctly labeled instances have a much higher probability to be sampled than the mislabeled instances. Therefore, from a good state $\mathbf{c}_i \in C^g$, the probability of sampling a correctly labeled instance is much greater than the probability of sampling a mislabeled instance. Thus, we have a much higher probability of jumping to a good state than to a bad state. In other words, the ratio $\frac{\mathbb{P}(C^g|\mathbf{c}_i \in C^g)}{\mathbb{P}(C^b|\mathbf{c}_i \in C^g)}$ is large.

However, for a bad state $\mathbf{c}_j \in C^b$, the corresponding model $\mathcal{M}_j$ does not have a high probability to fit most of the mislabeled instances in $\hat{S}$. This is because the noise data instances are more often than not corrupted by diverse corruption factors, thus they do not consistently follow the same data distribution. Therefore, from a bad state $\mathbf{c}_j \in C^b$, the probability of sampling a mislabeled instance may not be greater than the probability of sampling a correctly labeled instance. Recall that $\mathbb{P}(C^g \mid \mathbf{c}_i \in C^g)$ is much larger than $\mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)$, the ratio of $\frac{\mathbb{P}(C^b|\mathbf{c}_j \in C^b)}{\mathbb{P}(C^g|\mathbf{c}_j \in C^b)}$ is much smaller than $\frac{\mathbb{P}(C^g|\mathbf{c}_i \in C^g)}{\mathbb{P}(C^b|\mathbf{c}_i \in C^g)}$.

Inspired by the above insight, we make the following assumption.

$$\frac{\mathbb{P}(C^b \mid \mathbf{c}_j \in C^b)}{\mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)} \ll \frac{\mathbb{P}(C^g \mid \mathbf{c}_i \in C^g)}{\mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)} \tag{3}$$

Since $\mathbb{P}(C^g \mid \mathbf{c}_i) + \mathbb{P}(C^b \mid \mathbf{c}_i) = 1$ and $\mathbb{P}(C^g \mid \mathbf{c}_j) + \mathbb{P}(C^b \mid \mathbf{c}_j) = 1$, we can derive from Eq. 3 that

$$\mathbb{P}(C^b \mid \mathbf{c}_i \in C^g) \ll \mathbb{P}(C^g \mid \mathbf{c}_j \in C^b) \tag{4}$$

Next, we prove that, according to the stationary distribution $\boldsymbol{\pi}$ of the Markov chain C, the probability that a random walk visits the good states of C is larger than the probability of visiting the bad states of C.

Theorem 2 *Denote by $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$ the ith and the jth entries of the stationary distribution $\boldsymbol{\pi}$ of the Markov chain C, respectively. If Eq. 4 holds, then $\sum_{i:\mathbf{c}_i \in C^g} \boldsymbol{\pi}_i \gg \sum_{j:\mathbf{c}_j \in C^b} \boldsymbol{\pi}_j$.*

Proof Denote by P the transition matrix of the Markov chain C. Since $\boldsymbol{\pi}$ is the stationary distribution of C, $\boldsymbol{\pi} = \boldsymbol{\pi} P$.

Therefore, denote by $\boldsymbol{\pi}_h$ the hth entry of $\boldsymbol{\pi}$, the following equation holds for any $\mathbf{c}_h \in C$.

$$\boldsymbol{\pi}_h = \sum_{i:\mathbf{c}_i \in C^g} \mathbb{P}(\mathbf{c}_h \mid \mathbf{c}_i \in C^g)\boldsymbol{\pi}_i + \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(\mathbf{c}_h \mid \mathbf{c}_j \in C^b)\boldsymbol{\pi}_j$$

Therefore, we have

$$\sum_{h:\mathbf{c}_h \in C^g} \boldsymbol{\pi}_h = \sum_{h:\mathbf{c}_h \in C^g} \left(\sum_{i:\mathbf{c}_i \in C^g} \mathbb{P}(\mathbf{c}_h \mid \mathbf{c}_i \in C^g)\boldsymbol{\pi}_i + \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(\mathbf{c}_h \mid \mathbf{c}_j \in C^b)\boldsymbol{\pi}_j \right)$$
$$= \sum_{i:\mathbf{c}_i \in C^g} \mathbb{P}(C^g \mid \mathbf{c}_i \in C^g)\boldsymbol{\pi}_i + \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)\boldsymbol{\pi}_j$$

Thus, the following equation follows.

$$\sum_{i:\mathbf{c}_i \in C^g} \left(1 - \mathbb{P}(C^g \mid \mathbf{c}_i \in C^g)\right)\boldsymbol{\pi}_i = \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)\boldsymbol{\pi}_j$$

Since $\forall \mathbf{c}_i \in C^g, 1 - \mathbb{P}(C^g \mid \mathbf{c}_i \in C^g) = \mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)$, we have

$$\sum_{i:\mathbf{c}_i \in C^g} \mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)\boldsymbol{\pi}_i = \sum_{j:\mathbf{c}_j \in C^b} \mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)\boldsymbol{\pi}_j \tag{5}$$

Define $\mathbb{P}^*(C^b \mid \mathbf{c}_i \in C^g) = \max_{\mathbf{c}_i \in C^g} \mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)$, it follows from Eq. 4

$$\frac{\mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)}{\mathbb{P}^*(C^b \mid \mathbf{c}_i \in C^g)} \gg 1$$

Then, we can derive from Eq. 5

$$\sum_{i:\mathbf{c}_i \in C^g} \frac{\mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)}{\mathbb{P}^*(C^b \mid \mathbf{c}_i \in C^g)}\boldsymbol{\pi}_i = \sum_{j:\mathbf{c}_j \in C^b} \frac{\mathbb{P}(C^g \mid \mathbf{c}_j \in C^b)}{\mathbb{P}^*(C^b \mid \mathbf{c}_i \in C^g)}\boldsymbol{\pi}_j$$
$$\gg \sum_{j:\mathbf{c}_j \in C^b} \boldsymbol{\pi}_j$$

Consider

$$\sum_{i:\mathbf{c}_i \in C^g} \boldsymbol{\pi}_i \gg \sum_{i:\mathbf{c}_i \in C^g} \frac{\mathbb{P}(C^b \mid \mathbf{c}_i \in C^g)}{\mathbb{P}^*(C^b \mid \mathbf{c}_i \in C^g)}\boldsymbol{\pi}_i,$$

we have

$$\sum_{i:\mathbf{c}_i \in C^g} \boldsymbol{\pi}_i \gg \sum_{j:\mathbf{c}_j \in C^b} \boldsymbol{\pi}_j$$

□

According to Theorem 2, the sum of the probabilities of the good states in the stationary distribution of C is much larger than that of the bad states. Therefore, we

can easily identify the correctly labeled instances and mislabeled instances by the expectation of $\mathbf{c}$ over the stationary distribution $\boldsymbol{\pi}$, that is,

$$\overline{\mathbf{c}} = \sum_{i:\mathbf{c}_i \in C} \mathbf{c}_i \boldsymbol{\pi}_i \tag{6}$$

where the kth entry of $\overline{\mathbf{c}}$, denoted by $\overline{\mathbf{c}}(k)$, represents the porbability that instance $(x_k, \hat{y}_k) \in \hat{S}$ is correctly labeled. Apparently, an instance that is always contained by many good states has a larger probability to be correctly labeled than an instance that is always contained by bad states.

The expectation $\overline{\mathbf{c}}$ can be used in three different ways to train the final classification model $\mathcal{M}^*$.

- The *discarding method* trains $\mathcal{M}^*$ on the data set that is obtained by removing every instance $(x_k, \hat{y}_k) \in \hat{S}$ whose $\overline{\mathbf{c}}(k)$ is smaller than a given threshold α.
- The *flipping method* trains $\mathcal{M}^*$ on the data set that is obtained by flipping the label of every instance $(x_k, \hat{y}_k) \in \hat{S}$ whose $\overline{\mathbf{c}}(k)$ is smaller than a given threshold α.
- The *weighting method* uses $\overline{\mathbf{c}}$ as the importance weight of every instance to train $\mathcal{M}^*$.

As demonstrated later in Sect. 4, all these methods achieve state-of-the-art classification performance.

A typical way to obtain the stationary distribution $\boldsymbol{\pi}$ is to solve the linear equation $\boldsymbol{\pi} P = \boldsymbol{\pi}$. However, since the Markov chain C has $2^{|\hat{S}|} - 1$ states, it is intractable to compute and store the full transition matrix P. Therefore, we cannot obtain $\boldsymbol{\pi}$ by directly solving the equation $\boldsymbol{\pi} P = \boldsymbol{\pi}$.

Next, we introduce the Markov Chain Monte Carlo (MCMC) sampling algorithm that approximates $\boldsymbol{\pi}$ by sampling the proposed Markov chain without explicitly computing the full transition matrix P.

3.3 Approximation of the stationary distribution

According to Grimmett and Stirzaker (2001), the stationary distribution $\boldsymbol{\pi}$ of the Markov chain C can be approximated by iteratively sampling the states of C using random walk.

Since we have no prior knowledge on the noise distribution in $\hat{S}$, a good starting state to begin random walk is the state containing all instances in $\hat{S}$. We denote this state by $\mathbf{c}_1 = \mathbf{1}$, which is a $|\hat{S}|$-dimensional vector of all 1's.

Starting from a state $\mathbf{c}_i \in C$ $(i \geq 1)$, we sample the next state $\mathbf{c}_{i+1} \in C$ in the following steps. First, we train a model $\mathcal{M}_i$ on the set of instances $\hat{S}_i$ and compute the probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ for each instance $(x_k, \hat{y}_k) \in \hat{S}$. Second, we obtain the set of instances $\hat{S}_j$ by sampling each instance $(x_k, \hat{y}_k) \in \hat{S}$ with probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$, and get the next state $\mathbf{c}_j$ by setting $\mathbf{c}_{jk} = 1$ for each $(x_k, \hat{y}_k) \in \hat{S}_j$.

It is possible that $\hat{S}_j = \emptyset$. In this case, we simply redo the sampling until $\hat{S}_j \neq \emptyset$. However, since $\mathbb{P}(\hat{S}_j = \emptyset) = \prod_{k=1}^{n}(1 - \mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i))$, which decreases

Algorithm 1 Markov Chain Monte Carlo Sampling Method

Input: The set of label-corrupted instances $\hat{S}$.
Output: The approximated stationary distribution $\hat{\boldsymbol{\pi}}$.
1: Initialization: $i \leftarrow 1, \mathbf{c}_1 \leftarrow \mathbf{1}, \mathbb{C} \leftarrow \emptyset$.
2: **for** $i \leq N$ **do**
3: Train model $\mathcal{M}_i$ on $\hat{S}_i = \{(x_k, \hat{y}_k) \in \hat{S} \mid \mathbf{c}_{ik} = 1\}$.
4: $j \leftarrow i + 1, \mathbf{c}_j \leftarrow \mathbf{0}$.
5: **for** each $(x_k, \hat{y}_k) \in \hat{S}$ **do**
6: Set $\mathbf{c}_{jk} = 1$ with probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$.
7: **end for**
8: **if** $\sum_k \mathbf{c}_{jk} > 0$ **then**
9: $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbf{c}_j$.
10: $i \leftarrow i + 1$.
11: **end if**
12: **end for**
13: Compute $\hat{\boldsymbol{\pi}}$ by the distribution of the states in $\mathbb{C}$.
14: **return** $\hat{\boldsymbol{\pi}}$.

exponentially with respect to the volume of $\hat{S}$, the probability that we have to redo the sampling is very small.

Algorithm 1 shows the details of the proposed MCMC sampling method. In step 3, we start from a state $\mathbf{c}_i$, and train a model $\mathcal{M}_i$ on the corresponding set of instances $\hat{S}_i$. In steps 5–7, we perform a transition from the state $\mathbf{c}_i$ to the next state $\mathbf{c}_j$, $j = i+1$ by sampling the set of instances $\hat{S}_j$ from $\hat{S}$. To obtain $\hat{S}_j$, we use $\mathcal{M}_i$ to compute the probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$, and sample $(x_k, \hat{y}_k) \in \hat{S}$ according to $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$. After repeating the above sampling process to sample N times, we use the empirical distribution of the sampled states as the approximated stationary distribution.

The proposed MCMC sampling method approximates the stationary distribution without explicitly computing the full transition matrix P. The time complexity is only $\mathcal{O}(NT)$, where T is the training time of model $\mathcal{M}_i$ and N is the sampling size.

As demonstrated by the experiments in Sect. 4, the performance of the proposed method is scalable with respect to the sample size N, and $N = 200$ is good enough to achieve a better classification performance than the state-of-the-art methods.

4 Experiments

In this section, we evaluate the performance of our proposed MCS framework and compare it with the state-of-the-art methods including (1) Kernelized Support Vector Machine (KSVM) (Scholkopf and Smola 2001); (2) C-Support Vector Classification Filter (C-SVCF) (Thongkam et al. 2008); (3) Kernel Fisher Discriminant (KLDA) (Lawrence and Schölkopf 2001); (4) the importance re-Weighting method (IWℓ) that estimates noise rate by cross-validation (Liu and Tao 2016); and (5) the importance re-Weighting method (eIWℓ) that estimates noise rate jointly (Liu and Tao 2016). We use the LIBSVM (Chang and Lin 2011) implementation of KSVM and carefully implement C-SVCF in MATLAB R2015a. The code for IWℓ and eIWℓ is provided by Liu and Tao (2016).

Since most compared methods are based on SVM, for the fairness of comparison, we use KSVM (Scholkopf and Smola 2001) to train each model $\mathcal{M}_i$ and the final model $\mathcal{M}^*$ in the MCS framework.

As introduced in Sect. 3.2, we train $\mathcal{M}^*$ by three different methods. This derives three different versions of the MCS-based methods, denoted by MCS-KSVMD, MCS-KSVMF and MCS-KSVMW, which train $\mathcal{M}^*$ by the discarding method, the flipping method and the weighting method, respectively.

All compared methods are evaluated with default parameter settings. For the proposed MCS-based methods, we use $N = 200$, $\alpha = 0.5$. The effects of parameters N and α are discussed in Sect. 4.2.

All experiments are conducted using MATLAB R2015a on a PC that runs Windows 7 with an Intel Core i7-3770 CPU (3.40 GHz) and 16 GB main memory.

4.1 Data sets and evaluation metrics

We use the same UCI benchmark data sets used by Cawley and Talbot (2006), Liu and Tao (2016) and Natarajan et al. (2013) to evaluate the performance of all methods. The statistics of the data sets are shown in Table 2.

To generate class-conditional label noise, we randomly flip the labels of training instances according to the noise rate, denoted by a tuple $\boldsymbol{\rho} = (\rho^+, \rho^-)$. We conduct experiments on seven different noise rates, namely $\boldsymbol{\rho}_0 = (0.0, 0.0)$, $\boldsymbol{\rho}_1 = (0.1, 0.1)$, $\boldsymbol{\rho}_2 = (0.2, 0.2)$, $\boldsymbol{\rho}_3 = (0.3, 0.3)$, $\boldsymbol{\rho}_4 = (0.4, 0.4)$, $\boldsymbol{\rho}_5 = (0.1, 0.3)$ and $\boldsymbol{\rho}_6 = (0.3, 0.1)$. $\boldsymbol{\rho}_0$ is the setting for no label noise, $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\rho}_3$ and $\boldsymbol{\rho}_4$ are the settings for balanced label noise, $\boldsymbol{\rho}_5$ and $\boldsymbol{\rho}_6$ are the settings for unbalanced label noise.

To further evaluate the performance of all compared methods under adversarial noise, we employ the ALFASVMLib toolbox developed by Xiao et al. (2015) to generate 4 types of adversarial label noise, that is, $\boldsymbol{\gamma}_1 =$ ALFA attack, $\boldsymbol{\gamma}_2 =$ far first attack, $\boldsymbol{\gamma}_3 =$ nearest first attack, $\boldsymbol{\gamma}_4 =$ random attack.

For each data set, we randomly select 75% of the instances as training data, and use the rest 25% instances are used as testing data. Only the training data is corrupted

Table 2 The statistics of the UCI benchmark data sets

Data set	#Features	#Positive instances	#Negative instances	#Total instances
Breast cancer	9	77	186	263
Heart	13	120	150	270
Diabetis	8	268	500	768
German	20	300	700	1000
Splice	60	1344	1647	2991
Waveform	21	1647	3353	5000
Banana	2	2376	2924	5300
Ringnorm	20	3664	3736	7400
Twonorm	20	3703	3697	7400

by label noise. On each data set, we run every compared method 50 times and report the average performance, the variance and the p value.

The classification performance of all compared methods are evaluated by the same evaluation metric used by Liu and Tao (2016), that is

$$\text{Accuracy} = \frac{\#\text{correctly classified instances}}{\#\text{all instances}}$$

For the data cleaning methods C-SVCF and MCS-KSVMD that remove mislabeled instances, we further evaluate the data cleaning performance by the following two standard evaluation metrics (Frénay and Verleysen 2014).

$$\text{ER}_1 = \frac{\#\text{correctly labeled instances that are removed}}{\#\text{correctly labeled instances}}$$
$$\text{ER}_2 = \frac{\#\text{mislabeled instances that are not removed}}{\#\text{mislabeled instances}}$$

ER_1 is the error rate of removing correctly labeled instances. ER_2 is the error rate of retaining mislabeled instances. For both ER_1 and ER_2, a smaller value indicates a better performance.

4.2 Effect of parameters

In this section, we analyze the effect of parameters N and α on the MCS-KSVMD method. Since all MCS-based methods achieve comparable performance that are highly stable with respect to the parameters N and α, we use MCS-KSVMD as an representative and analyze its performance in this subsection. A comprehensive evaluation of all MCS-based methods is presented later in Sect. 4.3.

Figure 1 shows the effect of N on the accuracy of MCS-KSVMD. We can see that the accuracy on the large data sets, such as Splice, Waveform, Banana, Ringnorm and Twonorm, are more stable than the accuracy on the small data sets, such as Breast cancer, Heart, Diabetis and German. The reason why the accuracy is less stable on small data sets is that a small data set provides less training data in $\hat{S}_i$. When $\hat{S}_i$ is small, the model $\mathcal{M}_i$ may overfit a small subset of the correctly labeled instance, and may not fit the other correctly labeled instances well. In such a case, our assumption in Eq. 3 may not hold quite well, which leads to degenerated stableness of accuracy.

We can also see in Fig. 1 that the accuracy of MCS-KSVMD increases when N increases from 1 to 40. This is because, when N increases, the MCMC sampling method in Algorithm 1 samples more states, thus computes a better approximation of the stationary distribution $\boldsymbol{\pi}$. When $N > 40$, the accuracy converges on most of the data sets. According to the results in Fig. 1, we set $N = 200$ as the default parameter for all MCS-based methods in the rest of the experiments.

The default sample size $N = 200$ is extremely small comparing to $2^{|\hat{S}|} - 1$, the total number of states of the Markov chain C. We study this phenomenon by analyzing the influence of N on the set of instances that remains after MCS-KSVMD removes every instance $(x_k, \hat{y}_k) \in \hat{S}$ whose $\overline{\mathbf{c}}(k)$ is smaller than α.

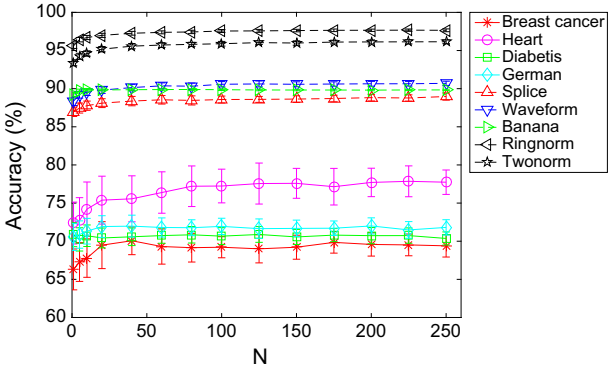

Fig. 1 The effect of parameter N on the accuracy of MCS-KSVMD. The parameters are set to $\boldsymbol{\rho}_1, \alpha = 0.5$ and $N = \{1, 5, 10, 20, 40, 60, 80, 100, 125, 150, 175, 200, 225, 250\}$

Denote by $\mathbf{b}_N$ a binary indicator vector induced by $\overline{\mathbf{c}}$ obtained at the Nth iteration of MCS-KSVMD. The kth entry of $\mathbf{b}_N$, denoted by $\mathbf{b}_N(k)$, is defined as

$$\mathbf{b}_N(k) = \begin{cases} 1, & \text{if } \overline{\mathbf{c}}(k) \geq \alpha \\ 0, & \text{if } \overline{\mathbf{c}}(k) < \alpha \end{cases} \tag{7}$$

The non-zero entries and zero entries of $\mathbf{b}_N$ identify the set of remaining instances and the set of removed instances, respectively.

To analyze the convergence of $\mathbf{b}_N$, we show the L1-distance between $\mathbf{b}_N (N \leq 1000)$ and $\mathbf{b}_{1000}$ in Fig. 2. We can see that $\mathbf{b}_N$ converges fast when N increases from 1 to 40, and the L1-distance between $\mathbf{b}_{200}$ and $\mathbf{b}_{1000}$ is very small. This means the set of remaining instances when $N = 200$ is quite similar with the set of remaining instances when $N = 1000$. Since the final classification model $\mathcal{M}^*$ is trained on the set of remaining instances, the classification accuracy when $N = 200$ is close to the classification accuracy when $N = 1000$. As a result, $\mathbf{b}_N$ converges fast when N increases. Therefore, the classification accuracy of MCS-KSVMD converges fast as well.

There are two possible reasons why $\mathbf{b}_N$ converges fast. First, for any model $\mathcal{M}_i$, most of the instances are far from the decision boundary of $\mathcal{M}_i$ in the high dimensional feature space. Each of these instances is either sampled with a very high probability if $\mathcal{M}_i$ fits it, or sampled with a very low probability if $\mathcal{M}_i$ does not fit it. Therefore, the random walk conducted by the MCMC sampling method focuses on visiting the states that are dominated by the instances far from the decision boundary of $\mathcal{M}_i$, and most of the $2^{|\hat{S}|} - 1$ states of C have an extremely low probability to be visited. Second, since most of the visited states are dominated by the same set of instances that are far from the decision boundaries, the models of these states produce similar classification results and sample probabilities for most of the instances that are far from the decision boundaries. As a result, many transition probabilities are close to zero in practice and the state transitions of the MCMC method tend to form a clique. Therefore, a small

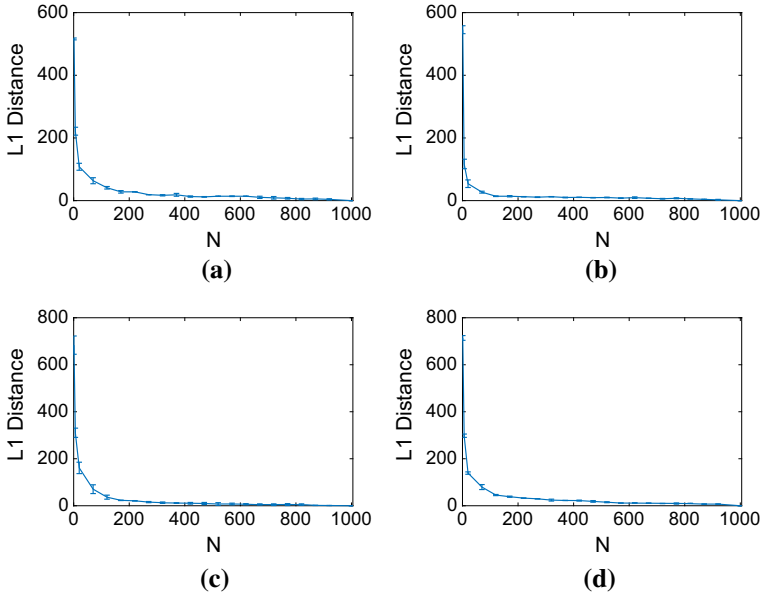

Fig. 2 The influence of N on the L1 distance between $\mathbf{b}_N\,(N \leq 1000)$ and $\mathbf{b}_{1000}$. **a** Waveform. **b** Banana. **c** Ringnorm. **d** Twonorm

sample size N is enough for the proposed MCMC sampling method to achieve a good classification performance without applying the burn-in phase or subsequent thinning.

Figure 3 shows the mean and deviation of the sample probabilities of every instance in four UCI data sets. For each instance, we show the mean and deviation of its sample probabilities, which are produced by the models of all the states visited by MCS-KSVMD. Such mean and deviation of sample probabilities are experimental evidences that support the two reasons for the fast convergence speed of MCS-KSVMD. We illustrate this as follows.

For the first reason, we can observe in Fig. 3 that a large proportion of instances have high sample probabilities; these instances are far from the decision boundaries, and are fitted by most of the models of the visited states. At the tail of each curve, there are some instances that have extremely low sample probabilities. Those instances are far from the decision boundaries, however, do not fit most of the models of visited states. Only a few instances have moderate sample probabilities; these are the instances that are near the boundaries.

For the second reason, we can see in Fig. 3 that the deviations are small for the instances with very large or small sample probabilities. This indicates that most of the models of the visited states produce similar classification results and sample probabilities for the instances that are far from the decision boundaries. We can also see that the deviations are large for the instances with moderate sample probabilities. This is because the classification results of most models are not stable for the instances that are close to the decision boundaries.

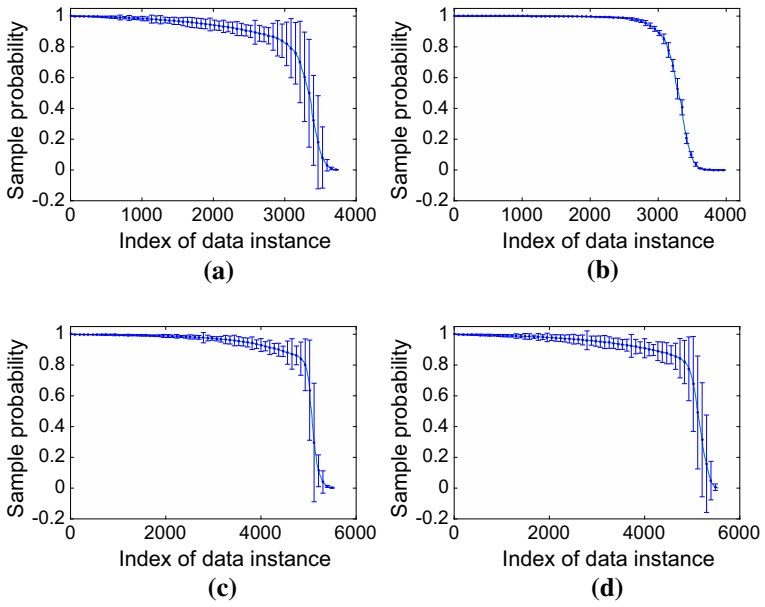

Fig. 3 The mean and standard deviation of the sample probability of every instance. The instances are sorted by their means in descending order. To make the error bars of deviation less crowded for better viewing, we show the deviations of 60 uniformly sampled instances. **a** Waveform. **b** Banana. **c** Ringnorm. **d** Twonorm

Interestingly, among all the data sets in Fig. 3, the Banana data set in Fig. 3b contains the largest proportion of instances whose means are highly close to either 1.0 or 0.0. The deviations of these instances are nearly 0.0. This means most instances are either sampled with a probability close to 1.0, or sampled with a probability close to 0.0. In this case, $\mathbf{b}_N$ converges very fast, which further leads to a fast convergence of classification accuracy. Accordingly, as shown in Fig. 1, MCS-KSVMD converges much faster on the Banana data set than on the data sets Waveform, Ringnorm and Twonorm.

Figure 4 shows the effect of parameter α on the accuracy of MCS-KSVMD. On most of the data sets, the accuracy of MCS-KSVMD first increases when α increases, then drops when α approaches 1. This is because the correctly labeled instances that are far from the decision boundary of model $\mathcal{M}_i$ are usually sampled with very high probabilities, and the mislabeled instances that are far from the decision boundary always carry a very low probability for sampling. Therefore, as shown in Fig. 5, the $\overline{\mathbf{c}}(k)$ value of most data instances are around 0 or 1, only a few data instances fall within the wide gap between 0 and 1. Recall that MCS-KSVMD filters out all instances whose $\overline{\mathbf{c}}(k)$ is smaller than threshold α. When $\alpha = 0$, no mislabeled instances are filtered out and MCS-KSVMD degenerates to KSVM, which gives a low accuracy performance. When α increases, more mislabeled instances will be filtered out, thus the performance of MCS-KSVMD increases for a wide range of α. When α approaches 1, a large proportion of correctly labeled instances are filtered out. In this case, MCS-KSVMD may not have enough data to train a good classifier, thus the accuracy performance drops.

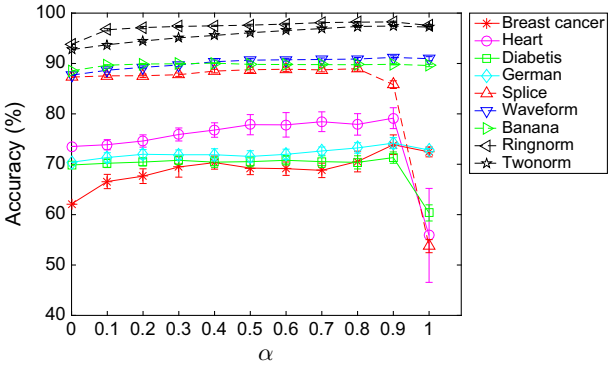

Fig. 4 The effect of parameter α on the accuracy of MCS-KSVMD. The parameters are set to $\boldsymbol{\rho}_1$, $N = 200$ and $\alpha = \{0.0, 0.1, \ldots, 1.0\}$

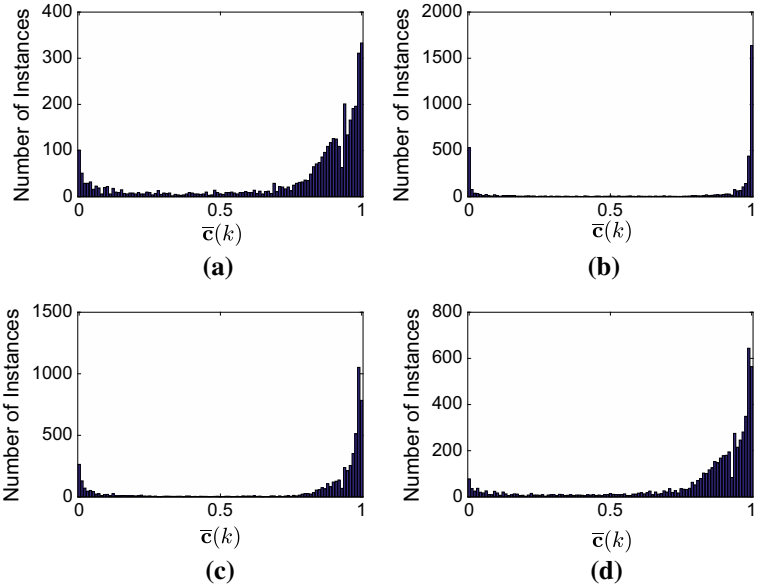

Fig. 5 The frequency distribution histograms of $\overline{\mathbf{c}}(k)$ on the data sets of Waveform, Banana, Ringnorm and Twonorm. **a** Waveform. **b** Banana. **c** Ringnorm. **d** Twonorm

4.3 Comparison on the UCI benchmark data sets

In this subsection, we analyze the performance in accuracy, ER1 and ER2 of all compared methods on the UCI benchmark data sets.

Tables 3, 4 and Fig. 6 show the accuracy of all methods for the seven settings of class-conditional noise $\boldsymbol{\rho}_0 \sim \boldsymbol{\rho}_6$ and the four settings of adversarial noise $\boldsymbol{\gamma}_1 \sim \boldsymbol{\gamma}_4$, respectively. For the noise setting of $\boldsymbol{\rho}_0$, there is no label noise, and we can see that all compared methods achieve comparable performance in accuracy. For the other noise

Table 3 The accuracy (in percentage) of all methods

Data set (n)	ρ	KSVM	C-SVCF	eIW	IW	MCS-KSVMD	MCS-KSVMF	MCS-KSVMW	p value
Breast cancer (268)	ρ_0	69.8 ± 5.6	70.2 ± 5.2	70.0 ± 5.5	**71.8 ± 5.4**	70.3 ± 4.7	70.3 ± 4.9	70.2 ± 4.6	–
	ρ_1	67.7 ± 4.8	68.2 ± 5.0	68.3 ± 6.3	**71.2 ± 5.5**	70.0 ± 5.2	70.1 ± 5.1	70.4 ± 4.9	–
	ρ_2	66.0 ± 5.9	67.1 ± 5.5	68.0 ± 6.4	**70.1 ± 5.6**	68.6 ± 5.1	69.0 ± 5.4	69.2 ± 5.2	–
	ρ_3	58.6 ± 6.1	59.8 ± 6.9	63.1 ± 7.2	**65.9 ± 7.5**	61.4 ± 7.5	61.7 ± 7.1	62.0 ± 7.1	–
	ρ_4	56.2 ± 8.0	57.0 ± 7.4	59.3 ± 9.4	**61.2 ± 11.1**	59.1 ± 8.4	59.2 ± 8.5	59.4 ± 8.7	–
	ρ_5	58.1 ± 6.6	59.2 ± 6.9	**63.3 ± 6.7**	63.1 ± 7.8	60.1 ± 6.6	60.5 ± 6.6	60.8 ± 6.3	–
	ρ_6	68.0 ± 5.8	68.9 ± 5.3	68.7 ± 6.3	**70.9 ± 5.0**	70.0 ± 3.9	70.5 ± 3.8	70.4 ± 4.0	–
Heart (270)	ρ_0	79.4 ± 5.5	79.6 ± 5.2	79.5 ± 5.2	79.0 ± 5.2	81.2 ± 4.5	**81.3 ± 4.4**	81.1 ± 4.1	**0.042**
	ρ_1	74.7 ± 4.3	74.5 ± 4.0	75.1 ± 4.5	74.7 ± 4.5	77.3 ± 4.9	77.4 ± 5.0	**77.4 ± 4.6**	**0.007**
	ρ_2	69.9 ± 6.1	70.3 ± 6.5	71.1 ± 6.6	70.9 ± 7.0	73.3 ± 5.9	73.1 ± 5.8	**74.0 ± 5.7**	**0.010**
	ρ_3	64.6 ± 6.9	64.6 ± 6.6	66.2 ± 6.2	66.1 ± 7.2	68.0 ± 6.2	68.1 ± 6.4	**68.6 ± 6.5**	**0.034**
	ρ_4	57.3 ± 6.8	58.1 ± 6.6	59.0 ± 7.0	59.3 ± 7.9	60.6 ± 8.0	60.8 ± 7.5	**61.1 ± 7.7**	0.126
	ρ_5	66.6 ± 5.6	67.6 ± 5.2	69.1 ± 4.8	67.6 ± 5.7	70.3 ± 6.3	**70.7 ± 6.6**	70.6 ± 6.6	0.090
	ρ_6	70.3 ± 6.4	71.1 ± 6.2	71.0 ± 7.0	70.3 ± 6.7	72.7 ± 6.4	72.9 ± 6.6	**73.2 ± 6.1**	**0.048**
Diabetis (768)	ρ_0	**75.1 ± 2.5**	75.0 ± 2.4	75.1 ± 2.7	74.8 ± 2.8	74.6 ± 2.6	74.6 ± 2.4	74.9 ± 2.5	–
	ρ_1	72.9 ± 2.7	72.9 ± 2.5	73.3 ± 2.5	72.8 ± 2.7	73.6 ± 2.9	**73.8 ± 2.7**	73.7 ± 2.6	0.175
	ρ_2	70.4 ± 3.2	70.6 ± 3.0	71.4 ± 3.4	71.1 ± 3.1	72.0 ± 3.4	**72.3 ± 3.3**	72.2 ± 3.1	0.102
	ρ_3	66.6 ± 3.7	67.9 ± 3.4	68.9 ± 3.6	68.5 ± 3.4	69.8 ± 3.4	70.0 ± 3.3	**70.3 ± 3.0**	**0.024**
	ρ_4	60.8 ± 3.9	61.4 ± 4.6	63.6 ± 5.2	**64.8 ± 6.1**	63.5 ± 5.0	63.6 ± 4.9	63.9 ± 4.7	–
	ρ_5	67.9 ± 3.4	68.9 ± 3.8	**71.4 ± 2.9**	69.1 ± 3.4	69.9 ± 3.3	70.1 ± 3.4	70.4 ± 3.3	–
	ρ_6	71.5 ± 2.9	71.7 ± 3.0	71.4 ± 3.1	69.9 ± 3.4	72.0 ± 3.2	71.9 ± 3.2	**72.4 ± 3.0**	0.132

Table 3 continued

Data set (n)	ρ	KSVM	C-SVCF	eIW	IW	MCS-KSVMD	MCS-KSVMF	MCS-KSVMW	p value
German (1000)	ρ_0	72.6 ± 2.5	72.9 ± 2.4	72.6 ± 2.6	73.3 ± 2.7	74.0 ± 2.5	74.0 ± 2.6	**74.4 ± 2.5**	**0.015**
	ρ_1	68.9 ± 2.7	69.1 ± 2.9	69.1 ± 2.8	70.1 ± 3.0	71.8 ± 2.4	71.8 ± 2.3	**72.3 ± 2.6**	**0.000**
	ρ_2	66.2 ± 4.1	66.7 ± 3.9	66.3 ± 4.1	66.9 ± 4.0	69.0 ± 3.7	69.0 ± 3.8	**69.5 ± 3.7**	**0.001**
	ρ_3	62.3 ± 3.2	63.3 ± 3.1	62.6 ± 3.5	63.3 ± 3.6	65.7 ± 3.2	65.7 ± 3.2	**66.0 ± 3.1**	**0.000**
	ρ_4	56.9 ± 3.4	57.4 ± 3.1	56.9 ± 3.2	58.2 ± 3.9	59.0 ± 3.1	59.1 ± 3.3	**59.6 ± 3.3**	**0.036**
	ρ_5	61.4 ± 3.3	61.9 ± 3.2	62.4 ± 3.4	62.4 ± 3.7	63.5 ± 2.9	63.8 ± 2.9	**64.2 ± 3.0**	**0.005**
	ρ_6	69.7 ± 2.8	70.1 ± 3.2	69.8 ± 2.8	70.8 ± 2.6	71.8 ± 2.7	71.8 ± 2.8	**71.9 ± 2.8**	**0.022**
Splice (2991)	ρ_0	**90.8 ± 0.9**	**90.8 ± 0.9**	**90.8 ± 0.9**	**90.8 ± 0.9**	90.7 ± 1.0	90.6 ± 1.0	90.6 ± 1.0	–
	ρ_1	86.4 ± 1.2	86.5 ± 1.2	86.5 ± 1.2	86.5 ± 1.2	87.8 ± 1.2	87.8 ± 1.2	**87.8 ± 1.1**	**0.000**
	ρ_2	80.3 ± 1.8	80.4 ± 1.7	80.4 ± 1.8	80.3 ± 1.8	82.2 ± 1.7	82.2 ± 1.6	**82.4 ± 1.7**	**0.000**
	ρ_3	71.3 ± 2.2	71.4 ± 2.1	71.4 ± 2.2	71.3 ± 2.2	73.2 ± 1.9	73.2 ± 1.9	**73.4 ± 1.9**	**0.000**
	ρ_4	61.4 ± 2.7	61.3 ± 2.7	61.4 ± 2.6	61.3 ± 2.5	62.2 ± 2.7	62.1 ± 2.7	**62.3 ± 2.7**	0.054
	ρ_5	78.3 ± 1.8	78.3 ± 1.7	78.3 ± 1.7	78.4 ± 1.7	79.7 ± 1.6	**79.8 ± 1.6**	**79.8 ± 1.6**	**0.000**
	ρ_6	79.2 ± 1.6	79.2 ± 1.5	79.5 ± 1.5	79.3 ± 1.6	80.3 ± 1.4	80.4 ± 1.4	**80.4 ± 1.5**	**0.001**
Waveform (5000)	ρ_0	90.2 ± 0.8	90.3 ± 0.9	90.2 ± 0.8	90.3 ± 0.9	90.7 ± 0.8	90.7 ± 0.8	**90.9 ± 0.8**	**0.000**
	ρ_1	88.0 ± 0.8	88.2 ± 0.6	88.0 ± 0.8	88.5 ± 0.8	89.8 ± 0.6	89.9 ± 0.6	**90.1 ± 0.7**	**0.000**
	ρ_2	84.0 ± 1.2	84.7 ± 1.1	84.2 ± 1.2	85.8 ± 0.9	87.7 ± 0.9	87.8 ± 0.9	**88.2 ± 0.9**	**0.000**
	ρ_3	76.2 ± 1.7	77.9 ± 1.6	78.0 ± 1.6	79.4 ± 1.8	82.6 ± 1.7	82.9 ± 1.7	**83.5 ± 1.6**	**0.000**
	ρ_4	65.0 ± 1.6	66.8 ± 1.8	68.5 ± 2.2	70.6 ± 2.5	71.7 ± 2.4	72.2 ± 2.4	**72.9 ± 2.4**	**0.000**
	ρ_5	79.4 ± 1.6	80.6 ± 1.5	80.2 ± 1.6	83.5 ± 1.6	84.3 ± 1.1	84.4 ± 1.2	**85.1 ± 1.2**	**0.000**
	ρ_6	84.1 ± 1.1	84.7 ± 0.9	85.8 ± 0.9	84.9 ± 1.4	87.2 ± 1.0	87.3 ± 1.0	**87.6 ± 1.0**	**0.000**

Table 3 continued

Data set (n)	$\boldsymbol{\rho}$	KSVM	C-SVCF	eIW	IW	MCS-KSVMD	MCS-KSVMF	MCS-KSVMW	p value
Banana (5300)	$\boldsymbol{\rho}_0$	89.9 ± 0.7	89.7 ± 0.7	88.9 ± 1.7	89.9 ± 0.7	90.0 ± 0.7	89.9 ± 0.7	$\mathbf{90.1 \pm 0.7}$	0.078
	$\boldsymbol{\rho}_1$	89.7 ± 0.6	89.7 ± 0.6	88.9 ± 1.0	89.6 ± 0.6	89.9 ± 0.6	89.9 ± 0.6	$\mathbf{90.0 \pm 0.6}$	**0.015**
	$\boldsymbol{\rho}_2$	89.6 ± 0.8	89.6 ± 0.8	87.5 ± 1.6	89.5 ± 0.8	89.8 ± 0.7	89.7 ± 0.7	$\mathbf{89.9 \pm 0.7}$	**0.012**
	$\boldsymbol{\rho}_3$	88.8 ± 0.9	89.0 ± 0.8	83.6 ± 3.4	88.6 ± 0.9	89.3 ± 1.0	89.2 ± 1.0	$\mathbf{89.4 \pm 1.1}$	**0.018**
	$\boldsymbol{\rho}_4$	87.1 ± 1.4	87.4 ± 1.5	71.8 ± 9.1	86.9 ± 1.8	87.7 ± 1.6	87.7 ± 1.6	$\mathbf{87.8 \pm 1.6}$	0.091
	$\boldsymbol{\rho}_5$	88.1 ± 0.8	88.3 ± 0.8	87.1 ± 1.8	88.1 ± 1.1	88.5 ± 1.0	88.5 ± 1.0	$\mathbf{88.6 \pm 1.0}$	**0.042**
	$\boldsymbol{\rho}_6$	88.0 ± 0.8	88.3 ± 0.9	85.4 ± 6.2	88.5 ± 1.1	89.2 ± 0.8	89.2 ± 0.8	$\mathbf{89.4 \pm 0.8}$	**0.000**
Ringnorm (7400)	$\boldsymbol{\rho}_0$	98.1 ± 0.3	98.1 ± 0.3	98.1 ± 0.3	98.1 ± 0.3	$\mathbf{98.3 \pm 0.3}$	$\mathbf{98.3 \pm 0.3}$	$\mathbf{98.3 \pm 0.3}$	**0.011**
	$\boldsymbol{\rho}_1$	94.4 ± 0.5	95.8 ± 0.4	94.6 ± 0.5	94.7 ± 0.5	97.6 ± 0.3	97.6 ± 0.4	$\mathbf{97.7 \pm 0.3}$	**0.000**
	$\boldsymbol{\rho}_2$	90.0 ± 0.8	91.9 ± 0.7	90.2 ± 0.8	90.2 ± 0.8	94.8 ± 0.6	94.8 ± 0.6	$\mathbf{94.9 \pm 0.6}$	**0.000**
	$\boldsymbol{\rho}_3$	83.9 ± 0.9	86.0 ± 1.0	84.8 ± 1.0	84.7 ± 1.0	89.3 ± 0.9	89.5 ± 1.0	$\mathbf{89.7 \pm 1.0}$	**0.000**
	$\boldsymbol{\rho}_4$	73.9 ± 1.7	77.4 ± 1.5	75.2 ± 1.8	78.4 ± 1.1	82.1 ± 1.2	82.4 ± 1.3	$\mathbf{82.6 \pm 1.3}$	**0.000**
	$\boldsymbol{\rho}_5$	91.8 ± 0.8	94.2 ± 0.8	93.7 ± 0.7	93.6 ± 0.8	96.6 ± 0.5	96.7 ± 0.5	$\mathbf{96.9 \pm 0.5}$	**0.000**
	$\boldsymbol{\rho}_6$	85.6 ± 1.0	86.9 ± 1.0	85.5 ± 1.0	85.4 ± 1.0	89.6 ± 1.0	89.6 ± 1.0	$\mathbf{89.7 \pm 1.0}$	**0.000**
Twonorm (7400)	$\boldsymbol{\rho}_0$	97.0 ± 0.4	97.1 ± 0.4	97.0 ± 0.4	97.0 ± 0.4	97.2 ± 0.4	97.2 ± 0.4	$\mathbf{97.3 \pm 0.4}$	**0.001**
	$\boldsymbol{\rho}_1$	93.3 ± 0.8	93.2 ± 0.8	93.6 ± 0.7	94.3 ± 0.7	96.1 ± 0.5	96.1 ± 0.5	$\mathbf{96.3 \pm 0.5}$	**0.000**
	$\boldsymbol{\rho}_2$	87.5 ± 0.9	88.1 ± 1.0	88.4 ± 0.9	89.0 ± 1.0	92.2 ± 0.7	92.4 ± 0.7	$\mathbf{93.1 \pm 0.7}$	**0.000**
	$\boldsymbol{\rho}_3$	78.5 ± 1.0	79.9 ± 1.0	80.8 ± 1.0	80.5 ± 1.0	85.3 ± 0.9	85.6 ± 1.0	$\mathbf{86.8 \pm 1.0}$	**0.000**
	$\boldsymbol{\rho}_4$	65.8 ± 1.6	68.0 ± 1.7	70.6 ± 1.4	69.9 ± 1.9	73.4 ± 1.8	73.9 ± 1.9	$\mathbf{75.0 \pm 1.9}$	**0.000**
	$\boldsymbol{\rho}_5$	85.8 ± 1.0	86.4 ± 0.8	87.0 ± 0.9	87.6 ± 0.9	90.3 ± 0.9	90.4 ± 0.9	$\mathbf{91.1 \pm 0.9}$	**0.000**
	$\boldsymbol{\rho}_6$	85.6 ± 1.0	86.3 ± 1.0	87.0 ± 0.9	87.7 ± 1.1	90.1 ± 0.9	90.3 ± 0.9	$\mathbf{90.9 \pm 0.8}$	**0.000**

n is the number of instances. $\boldsymbol{\rho} = (\rho^+, \rho^-)$ is the tuple of noise rates. $\boldsymbol{\rho}_0 = (0.0, 0.0)$, $\boldsymbol{\rho}_1 = (0.1, 0.1)$, $\boldsymbol{\rho}_2 = (0.2, 0.2)$, $\boldsymbol{\rho}_3 = (0.3, 0.3)$, $\boldsymbol{\rho}_4 = (0.4, 0.4)$, $\boldsymbol{\rho}_5 = (0.1, 0.3)$ and $\boldsymbol{\rho}_6 = (0.3, 0.1)$. The p value is computed by performing a paired t test between the best baseline method and the best MCS-based method
Bold values indicate the best performance

Table 4 The accuracy (in percentage) of all methods

Data set (n)	γ	KSVM	C-SVCF	eIW	IW	MCS-KSVMD	MCS-KSVMF	MCS-KSVMW	p value
Breast cancer (268)	γ_1	68.8 ± 4.7	68.2 ± 4.8	69.0 ± 5.1	$\mathbf{70.4 \pm 5.5}$	68.9 ± 5.4	69.0 ± 5.2	69.0 ± 5.5	–
	γ_2	69.5 ± 5.3	68.7 ± 5.5	70.5 ± 5.4	$\mathbf{72.1 \pm 5.8}$	69.7 ± 6.4	69.9 ± 6.4	69.8 ± 6.2	–
	γ_3	69.3 ± 4.3	69.5 ± 4.9	68.7 ± 6.1	$\mathbf{70.7 \pm 5.5}$	69.5 ± 5.4	69.5 ± 5.2	69.7 ± 5.6	–
	γ_4	69.8 ± 5.0	69.8 ± 4.8	70.3 ± 5.5	$\mathbf{72.2 \pm 5.2}$	70.6 ± 5.6	70.7 ± 5.6	71.0 ± 5.3	–
Heart (270)	γ_1	75.0 ± 5.7	76.1 ± 5.6	75.8 ± 5.6	75.1 ± 5.7	78.8 ± 5.5	78.7 ± 5.5	$\mathbf{79.4 \pm 5.5}$	**0.002**
	γ_2	73.3 ± 4.6	75.1 ± 5.0	74.4 ± 5.1	74.2 ± 4.9	79.9 ± 5.4	79.9 ± 5.5	$\mathbf{80.2 \pm 5.5}$	**0.000**
	γ_3	81.2 ± 3.8	80.9 ± 3.8	81.0 ± 3.7	81.1 ± 3.6	82.8 ± 3.9	82.7 ± 3.8	$\mathbf{82.9 \pm 3.8}$	**0.013**
	γ_4	77.1 ± 4.7	77.6 ± 4.8	77.4 ± 4.7	77.7 ± 4.2	81.7 ± 4.3	81.6 ± 4.3	$\mathbf{81.8 \pm 4.7}$	**0.000**
Diabetis (768)	γ_1	72.0 ± 2.9	71.8 ± 2.5	$\mathbf{72.5 \pm 2.4}$	71.8 ± 2.5	72.2 ± 2.6	72.4 ± 2.8	72.5 ± 2.6	–
	γ_2	71.5 ± 3.1	72.0 ± 3.1	72.4 ± 2.5	70.7 ± 2.7	72.8 ± 2.7	73.0 ± 2.8	$\mathbf{73.0 \pm 2.6}$	0.129
	γ_3	74.2 ± 3.0	73.8 ± 3.1	73.8 ± 3.0	73.5 ± 2.9	74.0 ± 2.8	73.9 ± 2.7	$\mathbf{74.2 \pm 2.6}$	0.470
	γ_4	73.4 ± 2.9	73.6 ± 2.7	73.6 ± 2.3	72.9 ± 2.8	73.9 ± 2.3	74.0 ± 2.3	$\mathbf{74.1 \pm 2.4}$	0.146
German (1000)	γ_1	70.4 ± 2.7	70.4 ± 2.4	70.5 ± 2.4	71.3 ± 2.0	71.6 ± 2.2	71.6 ± 2.2	$\mathbf{72.1 \pm 2.1}$	**0.042**
	γ_2	71.0 ± 3.1	70.6 ± 2.2	70.7 ± 2.7	71.9 ± 2.7	72.4 ± 2.0	72.3 ± 2.1	$\mathbf{72.8 \pm 2.3}$	**0.039**
	γ_3	73.2 ± 2.7	73.6 ± 2.3	73.0 ± 2.7	73.5 ± 2.6	74.4 ± 2.6	$\mathbf{74.5 \pm 2.6}$	74.4 ± 2.7	**0.037**
	γ_4	71.4 ± 2.5	71.4 ± 2.4	71.5 ± 2.6	72.4 ± 2.3	72.7 ± 2.3	72.9 ± 2.4	$\mathbf{73.2 \pm 2.4}$	**0.039**
Splice (2991)	γ_1	88.1 ± 1.1	88.1 ± 1.1	88.1 ± 1.1	88.1 ± 1.0	89.8 ± 1.1	89.8 ± 1.1	$\mathbf{89.9 \pm 1.1}$	**0.000**
	γ_2	86.5 ± 1.2	86.4 ± 1.2	86.6 ± 1.3	86.5 ± 1.2	90.1 ± 1.0	90.1 ± 1.0	$\mathbf{90.2 \pm 1.0}$	**0.000**
	γ_3	88.5 ± 1.0	88.5 ± 1.0	88.5 ± 1.1	88.5 ± 1.0	$\mathbf{88.8 \pm 1.0}$	88.7 ± 1.0	$\mathbf{88.8 \pm 1.0}$	**0.047**
	γ_4	88.9 ± 1.2	88.8 ± 1.2	88.9 ± 1.2	88.8 ± 1.2	$\mathbf{89.5 \pm 1.1}$	$\mathbf{89.5 \pm 1.1}$	$\mathbf{89.5 \pm 1.1}$	**0.003**

Table 4 continued

Data set (n)	γ	KSVM	C-SVCF	eIW	IW	MCS-KSVMD	MCS-KSVMF	MCS-KSVMW	p value
Waveform (5000)	γ_1	85.3 ± 0.8	85.6 ± 0.8	85.2 ± 0.7	85.4 ± 0.9	86.0 ± 0.8	86.0 ± 0.8	**86.3 ± 0.8**	**0.000**
	γ_2	86.0 ± 0.9	86.2 ± 0.9	85.6 ± 0.8	86.1 ± 0.8	86.8 ± 0.8	86.9 ± 0.8	**87.1 ± 0.9**	**0.000**
	γ_3	90.2 ± 0.6	90.3 ± 0.7	90.3 ± 0.6	90.2 ± 0.6	**90.7 ± 0.6**	90.6 ± 0.7	**90.7 ± 0.6**	**0.007**
	γ_4	89.2 ± 0.7	89.6 ± 0.7	89.1 ± 0.6	89.5 ± 0.7	90.5 ± 0.7	90.5 ± 0.7	**90.7 ± 0.6**	**0.000**
Banana (5300)	γ_1	86.1 ± 1.2	86.0 ± 1.2	85.2 ± 1.7	85.5 ± 1.5	86.1 ± 1.2	86.1 ± 1.2	**86.3 ± 1.2**	0.296
	γ_2	88.9 ± 1.2	89.0 ± 1.1	87.2 ± 2.1	88.5 ± 2.1	89.2 ± 1.8	89.1 ± 1.7	**89.3 ± 1.8**	0.125
	γ_3	89.5 ± 0.7	89.4 ± 0.7	88.9 ± 1.2	89.4 ± 0.7	89.9 ± 0.7	89.9 ± 0.7	**90.0 ± 0.7**	**0.001**
	γ_4	89.6 ± 0.7	89.5 ± 0.7	88.2 ± 2.2	89.5 ± 0.7	90.1 ± 0.7	90.0 ± 0.7	**90.2 ± 0.7**	**0.000**
Ringnorm (7400)	γ_1	96.2 ± 1.3	96.2 ± 1.3	96.3 ± 1.3	96.2 ± 1.3	96.6 ± 1.7	96.6 ± 1.7	**96.7 ± 1.7**	0.063
	γ_2	97.1 ± 0.3	97.2 ± 0.3	97.3 ± 0.3	97.2 ± 0.3	97.9 ± 0.3	97.9 ± 0.3	**98.0 ± 0.3**	**0.000**
	γ_3	97.5 ± 0.6	97.5 ± 0.6	**97.6 ± 0.7**	**97.6 ± 0.7**	97.3 ± 0.7	97.4 ± 0.7	97.5 ± 0.8	–
	γ_4	96.5 ± 0.4	97.1 ± 0.3	96.7 ± 0.4	96.8 ± 0.4	98.0 ± 0.3	98.0 ± 0.3	**98.1 ± 0.3**	**0.000**
Twonorm (7400)	γ_1	92.7 ± 0.5	92.7 ± 0.6	92.7 ± 0.5	92.7 ± 0.6	**92.9 ± 0.6**	**92.9 ± 0.6**	**92.9 ± 0.6**	**0.046**
	γ_2	92.8 ± 0.6	92.8 ± 0.6	92.8 ± 0.6	92.6 ± 0.8	93.0 ± 0.5	93.1 ± 0.5	**93.4 ± 0.5**	**0.000**
	γ_3	96.2 ± 0.4	96.3 ± 0.4	96.2 ± 0.4	96.2 ± 0.4	96.4 ± 0.3	96.4 ± 0.3	**96.5 ± 0.3**	**0.000**
	γ_4	95.3 ± 0.4	95.2 ± 0.5	95.3 ± 0.4	95.9 ± 0.4	96.9 ± 0.4	96.9 ± 0.4	**97.0 ± 0.4**	**0.000**

n is the number of instances. $\gamma_1, \gamma_2, \gamma_3$ and γ_4 are the four types of adversarial noise. The p value is computed by performing a paired t test between the best baseline method and the best MCS-based method

Bold values indicate the best performance

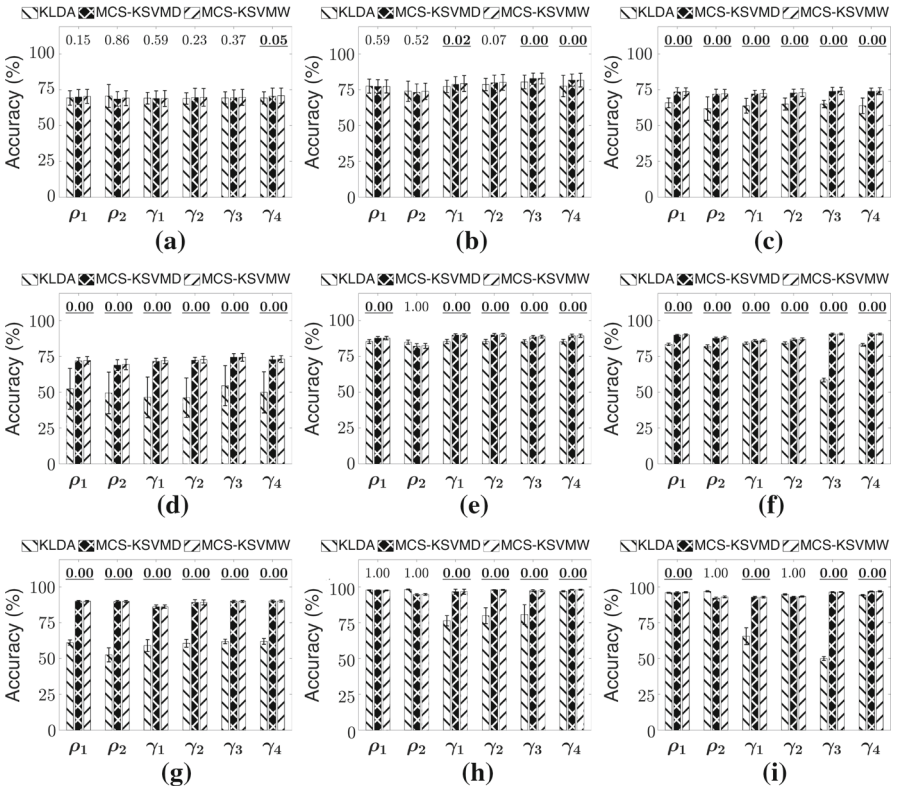

Fig. 6 The comparison of accuracy performance between KLDA and MCS-based methods on the noise settings of $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$, $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$, $\boldsymbol{\gamma}_3$ and $\boldsymbol{\gamma}_4$. The real numbers on top of each group of bars is the p values obtained by performing a paired t test between the best MCS-based method and KLDA. **a** Breast cancer. **b** Heart. **c** Diabetis. **d** German. **e** Splice. **f** Waveform. **g** Banana. **h** Ringnorm. **i** Twonorm

settings, such as $\boldsymbol{\rho}_1 \sim \boldsymbol{\rho}_6$ in Table 3 and $\boldsymbol{\gamma}_1 \sim \boldsymbol{\gamma}_4$ in Table 4, we can observe the following results.

First, C-SVCF slightly improves the accuracy of KSVM by training a KSVM on all label-corrupted instances, and removing the instances misclassified by the KSVM. This demonstrates that KSVM trained on label-corrupted instances can provide some useful information in identifying mislabeled instances. However, since the reliability of single KSVM is inevitably affected by the mislabeled instances, the accuracy of C-SVCF does not always outperform KSVM.

Second, the importance re-weighting methods IWℓ and eIWℓ achieve better accuracy than KSVM and C-SVCF. The improvement of accuracy is largely achieved by applying density ratio estimation to estimate the label noise rate. However, as claimed by Liu and Tao (2016), the classification performance of IWℓ and eIWℓ heavily relies on the accuracy of density ratio estimation, and it is difficult to choose an appropriate density estimation kernel to get accurate density ratio estimation in high dimensional space.

Third, as shown in Fig. 6, the MCS-based methods achieve a better accuracy than KLDA on most of the benchmark data sets. This demonstrate the superior noise resistance performance of the MCS-based methods over KLDA.

In summary, the proposed MCS-based methods MCS-KSVMD, MCS-KSVMF, MCS-KSVMW achieve comparable performance in accuracy, and the best MCS-based method outperforms the other compared methods on most of the data sets in a statistically significant manner. The good performance is achieved by assembling the probabilistic output of multiple KSVMs in a Markov chain to effectively identify mislabeled instances. Although a single KSVM may not be highly reliable, each of the multiple KSVM models provides some useful information in identifying mislabeled instances. The MCS framework comprehensively utilizes such information to achieve outstanding noise-resisting classification performance.

We can also observe that the MCS-based methods perform worse than the importance re-weighting methods on some small data sets, such as Breast cancer and Diabetis. This is because $\mathcal{M}_i$ overfits S_i when a data set is small. In such a case, the correctly labeled instances do not consistently support the sampling of each other, which leads to limited performance in accuracy.

Tables 5 and 6 show the performance in ER_1, ER_2 and $\mathrm{ER}_1+\mathrm{ER}_2$ of the data filtering methods C-SVCF and MCS-KSVMD for the seven settings of class-conditional noise and the four settings of adversarial noise, respectively. Since IWℓ, eIWℓ, MCS-KSVMF and MCS-KSVMW do not remove mislabeled instances, ER_1 and ER_2 are not applicable evaluation metrics to them.

On most of the data sets, C-SVCF slightly outperforms MCS-KSVMD in ER_1, however, MCS-KSVMD significantly outperforms C-SVCF in ER2 and $\mathrm{ER}_1+\mathrm{ER}_2$. These results indicate that, comparing with C-SVCF, MCS-KSVMD removes a little bit more correctly labeled instances and, at the same time, much more mislabeled instances.

To investigate why MCS-KSVMD removes some correctly labeled instances, we locate the removed correctly labeled instances in the feature space. It turns out that most of the correctly labeled instances removed are close to the decision boundary of the model $\mathcal{M}_i$ of many states. This is because an instance close to the decision boundary of $\mathcal{M}_i$ is sampled with a probability around 0.5. Recall that the threshold to remove an instance is set to $\alpha = 0.5$. It is likely that MCS-KSVMD removes some correctly labeled instances that are close to the decision boundary of $\mathcal{M}_i$.

More often than not, the classification performance of most conventional classification models is more sensitive to existing mislabeled instances than missing correctly labeled instances. Therefore, the improvement of classification performance achieved by removing more mislabeled instances out weights the negative influence of removing a few correctly labeled instances. As a result, MCS-KSVMD achieves much better classification performance than C-SVCF, which has been demonstrated by the performance in accuracy in Table 3 and Table 4.

Table 5 The ER_1, ER_2 and ER_1+ER_2 of C-SVCF and MCS-KSVMD

Data set (n)	ρ	C-SVCF ER_1	MCS-KSVMD ER_1	C-SVCF ER_2	MCS-KSVMD ER_2	C-SVCF ER_1+ER_2	MCS-KSVMD ER_1+ER_2	p value
Breast cancer (268)	ρ_1	**0.08 ± 0.02**	0.12 ± 0.02	0.70 ± 0.12	**0.55 ± 0.12**	0.78 ± 0.12	**0.66 ± 0.12**	**0.000**
	ρ_2	**0.08 ± 0.02**	0.11 ± 0.02	0.72 ± 0.07	**0.60 ± 0.07**	0.81 ± 0.08	**0.71 ± 0.08**	**0.000**
	ρ_3	**0.09 ± 0.02**	0.13 ± 0.03	0.79 ± 0.07	**0.69 ± 0.08**	0.88 ± 0.07	**0.82 ± 0.10**	**0.000**
	ρ_4	**0.11 ± 0.03**	0.15 ± 0.05	0.83 ± 0.03	**0.74 ± 0.06**	0.95 ± 0.05	**0.88 ± 0.09**	**0.000**
	ρ_5	**0.10 ± 0.03**	0.15 ± 0.03	0.78 ± 0.07	**0.67 ± 0.11**	0.89 ± 0.09	**0.82 ± 0.13**	**0.001**
	ρ_6	**0.07 ± 0.02**	0.10 ± 0.02	0.78 ± 0.08	**0.69 ± 0.08**	0.85 ± 0.08	**0.79 ± 0.07**	**0.000**
Heart (270)	ρ_1	**0.01 ± 0.01**	0.04 ± 0.01	0.87 ± 0.07	**0.60 ± 0.11**	0.88 ± 0.06	**0.64 ± 0.11**	**0.000**
	ρ_2	**0.01 ± 0.01**	0.03 ± 0.01	0.88 ± 0.05	**0.67 ± 0.08**	0.89 ± 0.05	**0.70 ± 0.08**	**0.000**
	ρ_3	**0.02 ± 0.01**	0.04 ± 0.01	0.90 ± 0.04	**0.76 ± 0.06**	0.92 ± 0.04	**0.81 ± 0.06**	**0.000**
	ρ_4	**0.03 ± 0.02**	0.06 ± 0.03	0.93 ± 0.02	**0.82 ± 0.04**	0.95 ± 0.03	**0.88 ± 0.06**	**0.000**
	ρ_5	**0.01 ± 0.01**	0.05 ± 0.02	0.90 ± 0.04	**0.72 ± 0.07**	0.91 ± 0.04	**0.77 ± 0.08**	**0.000**
	ρ_6	**0.01 ± 0.01**	0.04 ± 0.01	0.90 ± 0.05	**0.76 ± 0.07**	0.91 ± 0.05	**0.80 ± 0.08**	**0.000**
Diabetis (768)	ρ_1	**0.11 ± 0.01**	0.13 ± 0.01	0.51 ± 0.06	**0.43 ± 0.06**	0.62 ± 0.06	**0.56 ± 0.06**	**0.000**
	ρ_2	**0.11 ± 0.01**	0.14 ± 0.01	0.54 ± 0.04	**0.46 ± 0.04**	0.65 ± 0.04	**0.60 ± 0.05**	**0.000**
	ρ_3	**0.12 ± 0.01**	0.14 ± 0.02	0.60 ± 0.04	**0.51 ± 0.04**	0.72 ± 0.05	**0.64 ± 0.05**	**0.000**
	ρ_4	**0.15 ± 0.03**	0.17 ± 0.04	0.69 ± 0.05	**0.59 ± 0.06**	0.84 ± 0.07	**0.76 ± 0.10**	**0.000**
	ρ_5	**0.12 ± 0.01**	0.15 ± 0.01	0.58 ± 0.05	**0.48 ± 0.05**	0.70 ± 0.05	**0.63 ± 0.06**	**0.000**
	ρ_6	**0.11 ± 0.01**	0.13 ± 0.01	0.69 ± 0.05	**0.63 ± 0.05**	0.79 ± 0.05	**0.76 ± 0.06**	**0.001**

Table 5 continued

Data set (n)	ρ	C-SVCF ER_1	MCS-KSVMD ER_1	C-SVCF ER_2	MCS-KSVMD ER_2	C-SVCF ER_1+ER_2	MCS-KSVMD ER_1+ER_2	p value
German (1000)	ρ_1	**0.03 ± 0.00**	0.07 ± 0.01	0.83 ± 0.04	**0.62 ± 0.06**	0.86 ± 0.04	**0.69 ± 0.06**	**0.000**
	ρ_2	**0.03 ± 0.01**	0.07 ± 0.01	0.86 ± 0.03	**0.69 ± 0.04**	0.89 ± 0.03	**0.76 ± 0.05**	**0.000**
	ρ_3	**0.03 ± 0.01**	0.07 ± 0.01	0.88 ± 0.03	**0.73 ± 0.04**	0.91 ± 0.03	**0.81 ± 0.04**	**0.000**
	ρ_4	**0.04 ± 0.01**	0.09 ± 0.02	0.91 ± 0.02	**0.81 ± 0.03**	0.96 ± 0.02	**0.90 ± 0.04**	**0.000**
	ρ_5	**0.04 ± 0.01**	0.08 ± 0.01	0.88 ± 0.02	**0.73 ± 0.03**	0.92 ± 0.02	**0.81 ± 0.04**	**0.000**
	ρ_6	**0.03 ± 0.01**	0.07 ± 0.01	0.88 ± 0.03	**0.72 ± 0.04**	0.91 ± 0.03	**0.80 ± 0.04**	**0.000**
Splice (2991)	ρ_1	0.00 ± 0.00	0.00 ± 0.00	0.98 ± 0.01	**0.76 ± 0.03**	0.98 ± 0.01	**0.76 ± 0.03**	**0.000**
	ρ_2	0.00 ± 0.00	0.00 ± 0.00	0.98 ± 0.01	**0.86 ± 0.02**	0.98 ± 0.01	**0.86 ± 0.02**	**0.000**
	ρ_3	0.00 ± 0.00	0.00 ± 0.00	0.99 ± 0.00	**0.93 ± 0.01**	0.99 ± 0.00	**0.94 ± 0.01**	**0.000**
	ρ_4	**0.00 ± 0.00**	0.01 ± 0.00	0.99 ± 0.00	**0.96 ± 0.01**	0.99 ± 0.00	**0.97 ± 0.01**	**0.000**
	ρ_5	0.00 ± 0.00	0.00 ± 0.00	0.99 ± 0.01	**0.91 ± 0.01**	0.99 ± 0.01	**0.91 ± 0.01**	**0.000**
	ρ_6	**0.00 ± 0.00**	0.01 ± 0.00	0.98 ± 0.01	**0.86 ± 0.02**	0.98 ± 0.01	**0.87 ± 0.02**	**0.000**
Waveform (5000)	ρ_1	**0.02 ± 0.00**	0.04 ± 0.00	0.54 ± 0.02	**0.25 ± 0.03**	0.56 ± 0.02	**0.29 ± 0.03**	**0.000**
	ρ_2	**0.02 ± 0.00**	0.04 ± 0.00	0.59 ± 0.02	**0.32 ± 0.02**	0.61 ± 0.02	**0.36 ± 0.02**	**0.000**
	ρ_3	**0.03 ± 0.00**	0.04 ± 0.00	0.69 ± 0.01	**0.44 ± 0.02**	0.72 ± 0.02	**0.48 ± 0.02**	**0.000**
	ρ_4	**0.05 ± 0.01**	0.08 ± 0.01	0.79 ± 0.02	**0.58 ± 0.03**	0.84 ± 0.02	**0.66 ± 0.04**	**0.000**
	ρ_5	**0.03 ± 0.00**	0.05 ± 0.00	0.67 ± 0.02	**0.43 ± 0.02**	0.70 ± 0.02	**0.47 ± 0.02**	**0.000**
	ρ_6	**0.02 ± 0.00**	0.04 ± 0.00	0.65 ± 0.02	**0.40 ± 0.02**	0.68 ± 0.02	**0.44 ± 0.03**	**0.000**

Table 5 continued

Data set (n)	$\boldsymbol{\rho}$	C-SVCF ER_1	MCS-KSVMD ER_1	C-SVCF ER_2	MCS-KSVMD ER_2	C-SVCF ER_1+ER_2	MCS-KSVMD ER_1+ER_2	p value
Banana (5300)	$\boldsymbol{\rho}_1$	0.10 ± 0.00	$\mathbf{0.09 \pm 0.00}$	0.11 ± 0.01	$\mathbf{0.10 \pm 0.01}$	0.22 ± 0.01	$\mathbf{0.20 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_2$	0.11 ± 0.00	$\mathbf{0.10 \pm 0.00}$	0.11 ± 0.01	$\mathbf{0.10 \pm 0.01}$	0.22 ± 0.01	$\mathbf{0.20 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_3$	0.11 ± 0.01	$\mathbf{0.10 \pm 0.01}$	0.12 ± 0.01	$\mathbf{0.11 \pm 0.01}$	0.23 ± 0.01	$\mathbf{0.21 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_4$	0.13 ± 0.01	$\mathbf{0.11 \pm 0.01}$	0.14 ± 0.02	$\mathbf{0.13 \pm 0.02}$	0.27 ± 0.03	$\mathbf{0.24 \pm 0.03}$	**0.000**
	$\boldsymbol{\rho}_5$	0.12 ± 0.01	$\mathbf{0.10 \pm 0.01}$	0.14 ± 0.01	$\mathbf{0.13 \pm 0.02}$	0.26 ± 0.01	$\mathbf{0.24 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_6$	0.11 ± 0.00	$\mathbf{0.09 \pm 0.00}$	0.18 ± 0.01	$\mathbf{0.15 \pm 0.01}$	0.29 ± 0.01	$\mathbf{0.25 \pm 0.01}$	**0.000**
Ringnorm (7400)	$\boldsymbol{\rho}_1$	0.00 ± 0.00	0.00 ± 0.00	0.55 ± 0.02	$\mathbf{0.14 \pm 0.02}$	0.56 ± 0.02	$\mathbf{0.14 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_2$	0.00 ± 0.00	0.00 ± 0.00	0.56 ± 0.01	$\mathbf{0.31 \pm 0.02}$	0.56 ± 0.01	$\mathbf{0.32 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_3$	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.59 ± 0.01	$\mathbf{0.43 \pm 0.01}$	0.59 ± 0.01	$\mathbf{0.44 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_4$	0.02 ± 0.00	$\mathbf{0.01 \pm 0.00}$	0.66 ± 0.01	$\mathbf{0.49 \pm 0.01}$	0.68 ± 0.02	$\mathbf{0.50 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_5$	0.00 ± 0.00	0.00 ± 0.00	0.42 ± 0.02	$\mathbf{0.14 \pm 0.01}$	0.43 ± 0.02	$\mathbf{0.14 \pm 0.01}$	**0.000**
	$\boldsymbol{\rho}_6$	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.77 ± 0.01	$\mathbf{0.60 \pm 0.02}$	0.77 ± 0.01	$\mathbf{0.61 \pm 0.02}$	**0.000**
Twonorm (7400)	$\boldsymbol{\rho}_1$	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.66 ± 0.02	$\mathbf{0.23 \pm 0.03}$	0.67 ± 0.02	$\mathbf{0.24 \pm 0.03}$	**0.000**
	$\boldsymbol{\rho}_2$	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.71 ± 0.01	$\mathbf{0.38 \pm 0.02}$	0.71 ± 0.01	$\mathbf{0.39 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_3$	0.01 ± 0.00	0.01 ± 0.00	0.77 ± 0.01	$\mathbf{0.50 \pm 0.02}$	0.78 ± 0.01	$\mathbf{0.52 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_4$	$\mathbf{0.02 \pm 0.00}$	0.04 ± 0.01	0.85 ± 0.01	$\mathbf{0.64 \pm 0.02}$	0.87 ± 0.01	$\mathbf{0.68 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_5$	0.01 ± 0.00	0.01 ± 0.00	0.75 ± 0.01	$\mathbf{0.45 \pm 0.02}$	0.75 ± 0.01	$\mathbf{0.45 \pm 0.02}$	**0.000**
	$\boldsymbol{\rho}_6$	0.01 ± 0.00	0.01 ± 0.00	0.75 ± 0.01	$\mathbf{0.45 \pm 0.02}$	0.76 ± 0.01	$\mathbf{0.46 \pm 0.02}$	**0.000**

n is the number of instances. $\boldsymbol{\rho} = (\rho^+, \rho^-)$ is the tuple of noise rates. $\boldsymbol{\rho}_0 = (0.0, 0.0)$, $\boldsymbol{\rho}_1 = (0.1, 0.1)$, $\boldsymbol{\rho}_2 = (0.2, 0.2)$, $\boldsymbol{\rho}_3 = (0.3, 0.3)$, $\boldsymbol{\rho}_4 = (0.4, 0.4)$, $\boldsymbol{\rho}_5 = (0.1, 0.3)$ and $\boldsymbol{\rho}_6 = (0.3, 0.1)$. The p value is computed by performing a paired t test between the ER_1+ER_2 performance of C-SVCF and MCS-KSVMD

Bold values indicate the best performance

Table 6 The ER_1, ER_2 and ER_1+ER_2 of C-SVCF and MCS-KSVMD

Data set (n)	γ	C-SVCF ER_1	MCS-KSVMD ER_1	C-SVCF ER_2	MCS-KSVMD ER_2	C-SVCF ER_1+ER_2	MCS-KSVMD ER_1+ER_2	p value
Breast cancer (268)	γ_1	**0.08 ± 0.01**	0.12 ± 0.01	1.00 ± 0.02	**0.98 ± 0.06**	**1.08 ± 0.03**	1.10 ± 0.06	–
	γ_2	**0.08 ± 0.01**	0.13 ± 0.02	0.82 ± 0.13	**0.59 ± 0.22**	0.90 ± 0.13	**0.71 ± 0.22**	**0.000**
	γ_3	**0.08 ± 0.01**	0.12 ± 0.02	0.96 ± 0.07	**0.95 ± 0.08**	**1.05 ± 0.06**	1.07 ± 0.08	–
	γ_4	**0.08 ± 0.01**	0.12 ± 0.02	0.60 ± 0.14	**0.50 ± 0.16**	0.68 ± 0.14	**0.62 ± 0.16**	**0.026**
Heart (270)	γ_1	**0.01 ± 0.01**	0.04 ± 0.01	0.98 ± 0.04	**0.83 ± 0.21**	0.99 ± 0.04	**0.87 ± 0.21**	**0.000**
	γ_2	**0.01 ± 0.01**	0.04 ± 0.01	0.91 ± 0.11	**0.51 ± 0.25**	0.92 ± 0.11	**0.55 ± 0.25**	**0.000**
	γ_3	**0.01 ± 0.00**	0.03 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	**1.01 ± 0.00**	1.03 ± 0.01	–
	γ_4	**0.01 ± 0.01**	0.03 ± 0.01	0.83 ± 0.12	**0.60 ± 0.16**	0.84 ± 0.12	**0.64 ± 0.16**	**0.000**
Diabetis (768)	γ_1	**0.12 ± 0.01**	0.14 ± 0.01	0.97 ± 0.05	**0.94 ± 0.07**	1.08 ± 0.05	1.08 ± 0.07	0.451
	γ_2	**0.12 ± 0.01**	0.15 ± 0.01	0.71 ± 0.11	**0.34 ± 0.15**	0.83 ± 0.11	**0.49 ± 0.15**	**0.000**
	γ_3	**0.11 ± 0.01**	0.13 ± 0.01	0.92 ± 0.05	**0.91 ± 0.04**	**1.03 ± 0.05**	1.04 ± 0.04	–
	γ_4	**0.11 ± 0.01**	0.14 ± 0.01	0.52 ± 0.09	**0.43 ± 0.08**	0.63 ± 0.09	**0.56 ± 0.08**	**0.000**
German (1000)	γ_1	**0.03 ± 0.01**	0.08 ± 0.01	0.96 ± 0.03	**0.80 ± 0.09**	1.00 ± 0.03	**0.89 ± 0.09**	**0.000**
	γ_2	**0.04 ± 0.01**	0.09 ± 0.01	0.79 ± 0.07	**0.41 ± 0.15**	0.83 ± 0.08	**0.50 ± 0.15**	**0.000**
	γ_3	**0.03 ± 0.00**	0.06 ± 0.01	0.99 ± 0.02	**0.98 ± 0.02**	**1.02 ± 0.02**	1.05 ± 0.02	–
	γ_4	**0.03 ± 0.00**	0.08 ± 0.01	0.81 ± 0.08	**0.59 ± 0.09**	0.84 ± 0.08	**0.67 ± 0.09**	**0.000**
Splice (2991)	γ_1	0.00 ± 0.00	0.00 ± 0.00	0.97 ± 0.02	**0.49 ± 0.04**	0.97 ± 0.02	**0.49 ± 0.04**	**0.000**
	γ_2	0.00 ± 0.00	0.00 ± 0.00	0.98 ± 0.01	**0.20 ± 0.05**	0.98 ± 0.01	**0.20 ± 0.05**	**0.000**
	γ_3	0.00 ± 0.00	0.00 ± 0.00	0.97 ± 0.01	**0.78 ± 0.04**	0.97 ± 0.01	**0.79 ± 0.04**	**0.000**
	γ_4	0.00 ± 0.00	0.00 ± 0.00	0.97 ± 0.02	**0.69 ± 0.04**	0.97 ± 0.02	**0.70 ± 0.04**	**0.000**

Table 6 continued

Data set (n)	γ	C-SVCF ER_1	MCS-KSVMD ER_1	C-SVCF ER_2	MCS-KSVMD ER_2	C-SVCF ER_1+ER_2	MCS-KSVMD ER_1+ER_2	p value
Waveform (5000)	γ_1	**0.02 ± 0.00**	0.04 ± 0.00	0.99 ± 0.01	**0.96 ± 0.03**	1.01 ± 0.01	**1.00 ± 0.03**	**0.002**
	γ_2	**0.02 ± 0.00**	0.04 ± 0.00	0.94 ± 0.02	**0.82 ± 0.04**	0.96 ± 0.02	**0.86 ± 0.04**	**0.000**
	γ_3	**0.01 ± 0.00**	0.02 ± 0.01	0.96 ± 0.01	**0.92 ± 0.03**	0.98 ± 0.01	**0.94 ± 0.02**	**0.000**
	γ_4	**0.02 ± 0.00**	0.04 ± 0.00	0.54 ± 0.04	**0.22 ± 0.03**	0.56 ± 0.04	**0.26 ± 0.03**	**0.000**
Banana (5300)	γ_1	0.13 ± 0.01	**0.10 ± 0.01**	**0.76 ± 0.28**	0.78 ± 0.25	0.89 ± 0.27	**0.88 ± 0.24**	0.453
	γ_2	0.13 ± 0.01	**0.10 ± 0.01**	**0.09 ± 0.07**	0.13 ± 0.17	**0.22 ± 0.07**	0.23 ± 0.18	–
	γ_3	0.10 ± 0.00	**0.08 ± 0.00**	0.56 ± 0.03	**0.54 ± 0.03**	0.66 ± 0.03	**0.61 ± 0.03**	**0.000**
	γ_4	0.12 ± 0.00	**0.09 ± 0.00**	0.13 ± 0.02	**0.11 ± 0.02**	0.26 ± 0.02	**0.20 ± 0.02**	**0.000**
Ringnorm (7400)	γ_1	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	**0.49 ± 0.41**	1.00 ± 0.00	**0.50 ± 0.41**	**0.000**
	γ_2	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	**0.33 ± 0.23**	1.00 ± 0.00	**0.33 ± 0.23**	**0.000**
	γ_3	0.00 ± 0.00	0.00 ± 0.00	0.95 ± 0.03	**0.78 ± 0.10**	0.95 ± 0.03	**0.78 ± 0.10**	**0.000**
	γ_4	0.00 ± 0.00	0.00 ± 0.00	0.56 ± 0.03	**0.08 ± 0.02**	0.56 ± 0.03	**0.09 ± 0.02**	**0.000**
Twonorm (7400)	γ_1	**0.00 ± 0.00**	0.01 ± 0.00	1.00 ± 0.00	**0.98 ± 0.01**	1.00 ± 0.00	**0.99 ± 0.01**	**0.000**
	γ_2	**0.00 ± 0.00**	0.01 ± 0.00	0.95 ± 0.01	**0.78 ± 0.02**	0.95 ± 0.01	**0.79 ± 0.02**	**0.000**
	γ_3	0.00 ± 0.00	0.00 ± 0.00	0.98 ± 0.01	**0.90 ± 0.02**	0.98 ± 0.01	**0.91 ± 0.02**	**0.000**
	γ_4	0.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.03	**0.15 ± 0.03**	0.67 ± 0.03	**0.15 ± 0.03**	**0.000**

n is the number of instances. γ_1, γ_2, γ_3 and γ_4 are the four types of adversarial noise. The p value is computed by performing a paired t test between the ER_1+ER_2 performance of C-SVCF and MCS-KSVMD

Bold values indicate the best performance

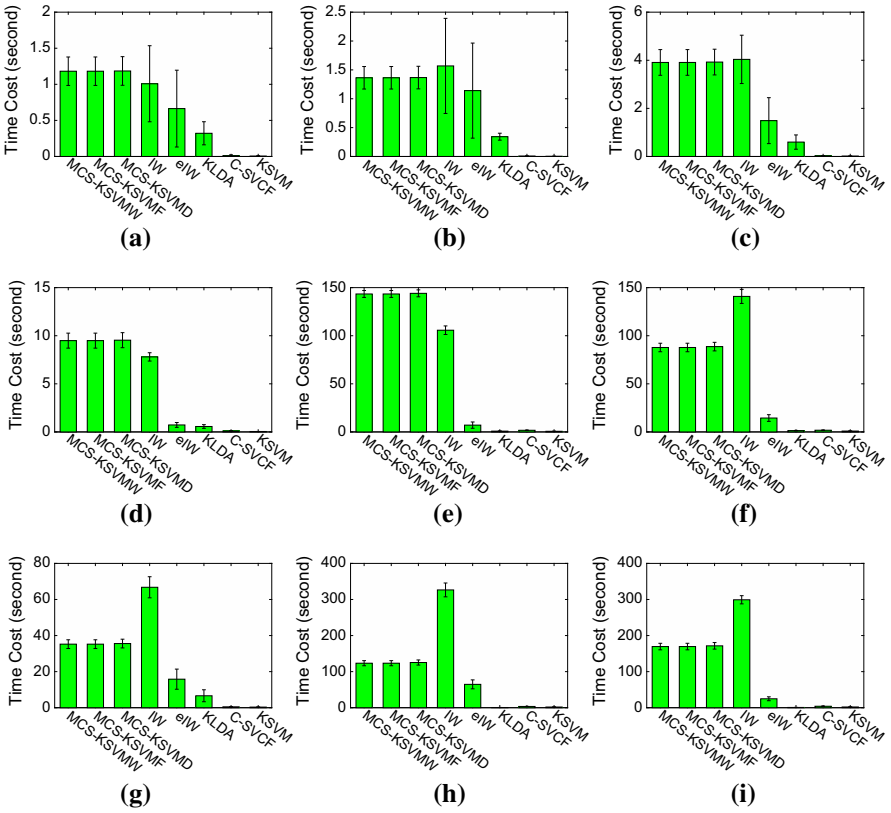

Fig. 7 The time cost of all compared methods on the UCI benchmark data sets in noise setting $\boldsymbol{\rho}_1$. For MCS-based methods, we use $N = 200$, $\alpha = 0.5$. For the other methods, we use their default parameters. **a** Breast cancer. **b** Heart. **c** Diabetis. **d** German. **e** Splice. **f** Waveform. **g** Banana. **h** Ringnorm. **i** Twonorm

4.4 Efficiency of Markov Chain Monte Carlo sampling

In this subsection, we compare the efficiency of all methods, and analyze the effect of the sample size N on the time cost of the proposed MCMC method in Algorithm 1.

Figure 7 shows the time cost of all compared methods on each of the UCI benchmark data sets. eIW, KLDA, C-SVCF and KSVM are efficient due to their low time complexity. The time cost of IW is much larger than eIW because the density ratio estimation of IW is time consuming (Liu and Tao 2016). We can also observe that the MCS-based methods achieve a comparable time cost with IW on the small data sets, such as Breast cancer, Heart and Diabetis. On the large data sets, such as Waveform, Banana, Ringnorm and Twonorm, the MCS-based methods even achieve a lower time cost than IW.

To evaluate how the time cost of the MCMC method in Algorithm 1 changes when the sample size N increases, we show in Fig. 8 the time cost of the MCMC sampling method, denoted by t_{MCMC}, as well as the time cost to train a single KSVM on $\hat{S}$, denoted by t_{KSVM}. For each value of N, t_{MCMC} and t_{KSVM} are the total time cost

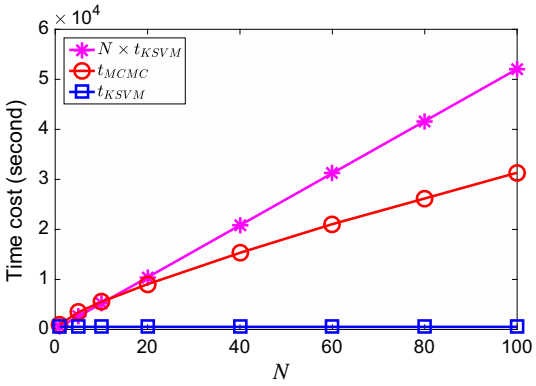

Fig. 8 The time cost of KSVM and the MCMC sampling method in Algorithm 1. t_{MCMC} and t_{KSVM} are the total time cost of running MCMC and KSVM, respectively, on all benchmark data sets in the six class-conditional noise settings $\boldsymbol{\rho}_1 \sim \boldsymbol{\rho}_6$

of running MCMC and KSVM, respectively, on all benchmark data sets in the six class-conditional noise settings $\boldsymbol{\rho}_1 \sim \boldsymbol{\rho}_6$.

As it is shown in Fig. 7, t_{MCMC} increases linearly with respect to N, and it is always smaller than Nt_{KSVM}. This is because the time complexity of the MCMC sampling method is $\mathcal{O}(NT)$, where T is the average time cost to train a single model $\mathcal{M}_i$ on $\hat{S}_i$. Since $|\hat{S}_i| \leq |\hat{S}|$, it follows that $T \leq t_{\text{KSVM}}$. Therefore, $NT \leq Nt_{\text{KSVM}}$.

In sum, since a small sample size N is enough for the MCS-based methods to achieve good classification performance, the proposed MCS framework is efficient.

4.5 Case study on synthetic data

In this section, we present a case study on a synthetic data set to show the identified mislabeled instances and the decision boundary of the MCS-based model. We do not use MCS-KSVM, since it does not provide an explicit decision boundary. Instead, we use the MCS-LR model, which employs Logistic Regression (LR) (Hosmer Jr et al. 2013) to train $\mathcal{M}_i$ and the final model $\mathcal{M}^*$ for the MCS framework. The discarding method is employed by MCS-LR. For comparison, we use LR as the baseline method.

We generate a *clean synthetic data set* by sampling 200 instances from the mixture of two 2-dimensional multivariate normal distributions in a standard 2-dimensional Cartesian coordinate system. The mean vectors of the multivariate normal distributions are $[5, 3]^\top$ and $[3, 5]^\top$, respectively. The covariance matrices are the same, which is a 2-by-2 diagonal matrix whose diagonal elements are set to 0.4. The data set consists of 100 positive instances and 100 negative instances. Each class of instances is generated by one of the multivariate normal distributions. The feature of each instance is a 2-dimensional vector denoted by $[x_1, x_2]^\top \in R^2$. To generate a *corrupted synthetic data set*, we randomly flip the label of each instance, whose second feature x_2 is larger than 3 and smaller than 5, with a probability of 0.5.

Figure 9a shows the distribution of the clean synthetic data set. Figure 9b shows the performance of MCS-LR in identifying the mislabeled instances in the corrupted synthetic data set. For all mislabeled instances, those that are correctly identified by MCS-LR are marked by circles, and those that are missed by MCS-LR are marked by squares. No correctly labeled instance is identified as mislabeled instance. Most of

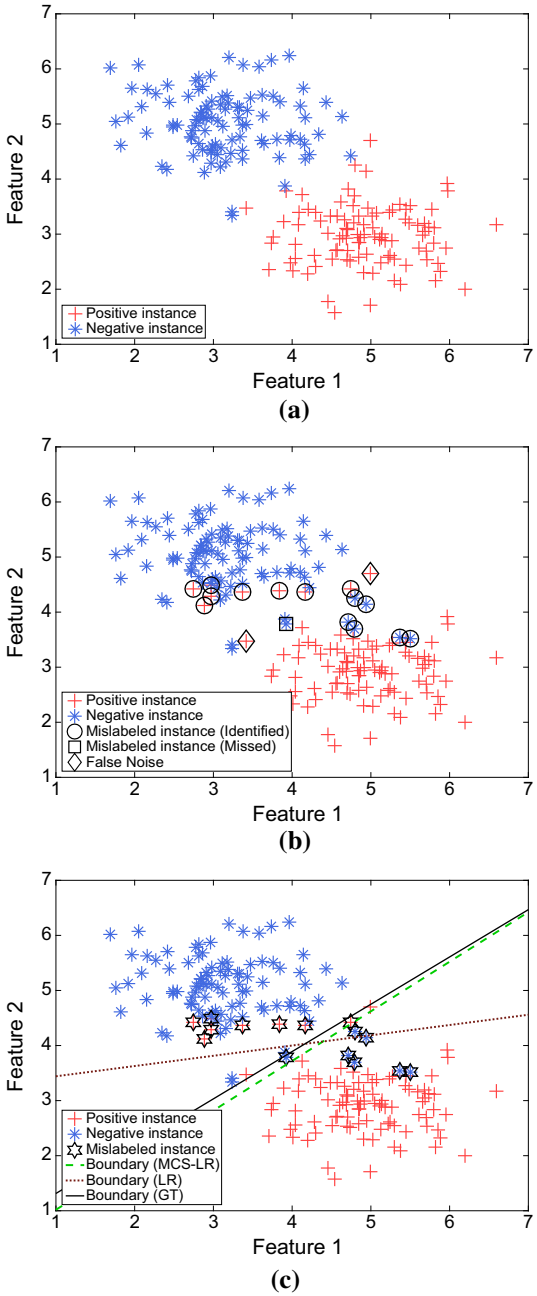

Fig. 9 A case study of LR and MCS-LR on the synthetic data set. A false noise is a correctly labeled instance that is identified as a mislabeled instance by MCS-LR. **a** The clean synthetic data set. **b** The noise identification performance. **c** The performance of MCS-LR and LR

the mislabeled instances are correctly identified by MCS-LR. This demonstrates the outstanding performance of MCS-LR in identifying mislabeled instances.

Figure 9c draws the decision boundaries of MCS-LR and LR, which are trained on the corrupted synthetic data set. The 'Boundary (GT)' is the ground truth boundary that is obtained by training LR on the clean synthetic data set. Due to the effect of mislabeled instances, "Boundary (LR)" significantly deviates from "Boundary (GT)". However, "Boundary (MCS-LR)" is very close to "Boundary (GT)", which demonstrates the superior noise-resisting performance of MCS-LR.

In summary, the proposed MCS framework achieves impressive performance in identifying mislabeled instances and training noise-resisting classifiers.

5 Conclusions

In this paper, we tackle the challenging problem of classification with label noise. We propose a novel MCS framework that embeds a conventional classification model into a carefully designed Markov chain to accurately identify mislabeled instances and train noise-resisting classifiers. The MCS framework smoothly works with a wide spectrum of conventional binary classification models, and achieves outstanding noise-resisting binary classification performance.

For future work, we will generalize the MCS framework to tackle the more challenging problem of multi-class classification in the presence of both label noise and feature noise.

Acknowledgements This work was supported in part by the NSERC Discovery Grant Program, the Australian Research Council Projects FL-170100117, DP-180103424, and IH180100002. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66

Angluin D, Laird P (1988) Learning from noisy examples. Mach Learn 2(4):343–370

Bartlett PL, Jordan MI, McAuliffe JD (2006) Convexity, classification, and risk bounds. J Am Stat Assoc 101(473):138–156

Belur V (1991) Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, Washington DC

Biggio B, Nelson B, Laskov P (2011) Support vector machines under adversarial label noise. Asian Conf Mach Learn 20:97–112

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Cawley G, Talbot N (2006) http://theoval.cmp.uea.ac.uk/matlab

Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27:1–27:27

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27

Delany SJ, Cunningham P (2004) An analysis of case-base editing in a spam filtering system. Eur Conf Case-Based Reason 3115:128–141

Folleco A, Khoshgoftaar TM, Van Hulse J, Bullard L (2008) Identifying learners robust to low quality data. In: International conference on information reuse and integration, pp 190–195

Frénay B, Verleysen M (2014) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869

Gamberger D, Lavrač N (1996) Noise detection and elimination applied to noise handling in a KRK chess endgame. In: International workshop on inductive logic programming, pp 72–88
Gamberger D, Lavrač N, Džeroski S (1996) Noise elimination in inductive concept learning: a case study in medical diagnosis. Int Workshop Algorithm Learn Theory 1160:199–212
Gamberger D, Lavrac N, Groselj C (1999) Experiments with noise filtering in a medical domain. In: International conference on machine learning, pp 143–151
Ghosh A, Manwani N, Sastry P (2015) Making risk minimization tolerant to label noise. Neurocomputing 160:93–107
Gilks WR, Richardson S, Spiegelhalter D (1995) Markov Chain Monte Carlo in practice. CRC Press, Boca Raton
Grimmett G, Stirzaker D (2001) Probability and random processes. Oxford University Press, Oxford
Guyon I, Matic N, Vapnik V, et al (1994) Discovering informative patterns and data cleaning. In: International conference on knowledge discovery and data mining, pp 145–156
Hosmer DW Jr, Lemeshow S, Sturdivant RX (2013) Applied logistic regression, vol 398. Wiley, Hoboken
John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Uncertainty in artificial intelligence, pp 338–345
Karger DR, Oh S, Shah D (2011) Iterative learning for reliable crowdsourcing systems. In: Advances in neural information processing systems, pp 1953–1961
Khoshgoftaar TM, Rebours P (2004) Generating multiple noise elimination filters with the ensemble-partitioning filter. In: International conference on information reuse and integration, pp 369–375
Lawrence ND, Schölkopf B (2001) Estimating a kernel fisher discriminant in the presence of label noise. Int Conf Mach Learn 1:306–313
Li H (2015) Theoretical analysis and efficient algorithms for crowdsourcing. University of California, Berkeley
Liu Q, Peng J, Ihler AT (2012) Variational inference for crowdsourcing. In: Advances in neural information processing systems, pp 692–700
Liu T, Tao D (2016) Classification with noisy labels by importance reweighting. IEEE Trans Pattern Anal Mach Intell 38(3):447–461
Long PM, Servedio RA (2008) Random classification noise defeats all convex potential boosters. In: International conference on machine learning, pp 608–615
Manwani N, Sastry P (2013) Noise tolerance under risk minimization. IEEE Trans Cybern 43(3):1146–1151
Miranda AL, Garcia LPF, Carvalho AC, Lorena AC (2009) Use of classification algorithms in noise detection and elimination. Int Conf Hybrid Artif Intell Syst 5572:417–424
Natarajan N, Dhillon IS, Ravikumar PK, Tewari A (2013) Learning with noisy labels. In: Advances in neural information processing systems, pp 1196–1204
Nettleton DF, Orriols-Puig A, Fornells A (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artif Intell Rev 33(4):275–306
Patrini G, Nielsen F, Nock R, Carioni M (2016) Loss factorization, weakly supervised learning and label noise robustness. In: International conference on machine learning, pp 708–717
Quinlan JR (2014) C4.5: programs for machine learning. Elsevier, Amsterdam
Raykar VC, Yu S, Zhao LH, Valadez GH, Florin C, Bogoni L, Moy L (2010) Learning from crowds. J Mach Learn Res 11(Apr):1297–1322
Scholkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
Scott C, Blanchard G, Handy G (2013) Classification with asymmetric label noise: consistency and maximal denoising. Conf Learn Theory 30:489–511
Sheng VS, Provost F, Ipeirotis PG (2008) Get another label? Improving data quality and data mining using multiple, noisy labelers. In: International conference on knowledge discovery and data mining, pp 614–622
Stempfel G, Ralaivola L (2009) Learning SVMs from sloppily labeled data. In: International conference on artificial neural networks, pp 884–893
Sun JW, Zhao Wang CI, Chen SF (2007) Identifying and correcting mislabeled training instances. Future Gener Commun Netw 1:244–250
Thongkam J, Xu G, Zhang Y, Huang F (2008) Support vector machine for outlier detection in breast cancer survivability prediction. Asia-Pac Web Conf 4977:99–109
Valiant LG (1984) A theory of the learnable. ACM Commun 27(11):1134–1142
Vapnik V (2013) The nature of statistical learning theory. Springer, New York

Vaughan JW (2018) Making better use of the crowd: how crowdsourcing can advance machine learning research. J Mach Learn Res 18(1):7026–7071
Whitehill J, Wu Tf, Bergsma J, Movellan JR, Ruvolo PL (2009) Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In: Advances in neural information processing systems, pp 2035–2043
Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybern 2(3):408–421
Wilson DR, Martinez TR (1997) Instance pruning techniques. Int Conf Mach Learn 97:403–411
Wilson DR, Martinez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38(3):257–286
Xiao H, Biggio B, Nelson B, Xiao H, Eckert C, Roli F (2015) Support vector machines under adversarial label contamination. Neurocomputing 160:53–62
Yang T, Mahdavi M, Jin R, Zhang L, Zhou Y (2012) Multiple kernel learning from noisy labels by stochastic programming. In: International conference on machine learning, pp 1–8
Zhou D, Basu S, Mao Y, Platt JC (2012) Learning from the wisdom of crowds by minimax entropy. In: Advances in neural information processing systems, pp 2195–2203

Affiliations

Zijin Zhao[1] · Lingyang Chu[1] · Dacheng Tao[2] · Jian Pei[1]

Zijin Zhao
zijinz@sfu.ca

Dacheng Tao
dacheng.tao@sydney.edu.au

Jian Pei
jpei@cs.sfu.ca

1 School of Computing Science, Simon Fraser University, Burnaby, Canada

2 The Faculty of Engineering and Information Technologies, The UBTECH Sydney Artificial Intelligence Centre and the School of Information Technologies, The University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia