Classification in the Presence of Heavy Label Noise: A Markov Chain Sampling Framework

by

Zijin Zhao

B.Eng., Nankai University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2017

Approval

Name:	**Zijin Zhao**
Degree:	**Master of Science (Computing Science)**
Title:	***Classification in the Presence of Heavy Label Noise: A Markov Chain Sampling Framework***
Examining Committee:	**Chair:** Dr. Qianping Gu Professor School of Computing Science

Dr. Jian Pei
Senior Supervisor
Professor
School of Computing Science _____

Dr. Liangliang Wang
Supervisor
Assistant Professor
Department of Statistics and Actuarial Science _____

Dr. Jiannan Wang
Internal Examiner
Assistant Professor
School of Computing Science _____

Date Defended: June 19, 2017

Abstract

Heavy label noise is often present in many practical scenarios where observed labels of instances are corrupted. Classification with heavy label noise has great significance and attracts a lot of attention, since label noise may lead to many potential negative consequences. Many state-of-the-art approaches assume that label noise is class-dependent, and thus cannot be generalized to situations without this assumption. In this thesis, we propose a Markov chain sampling framework, MCS, to conquer the limitations of the existing methods in the binary classification problem. The main idea is to utilize the predictions of a sequence of classifiers in an ensemble way to detect mislabeled instances, the sequence of classifiers is trained on different subsets of the training data by sampling the states of a carefully designed Markov chain with random walk. Our proposed MCS framework is general and can entertain a wide spectrum of classification algorithms. We theoretically prove the correctness and effectiveness of the MCS framework. We further present experimental results showing the effectiveness and efficiency of the proposed framework and derivative algorithms.

Keywords: classification; label noise; Markov chain; sampling

To my family.

Acknowledgements

I would like to express my deepest gratitude to my senior supervisor, Dr. Jian Pei, for his wisdom guidance, continuous support and warm encouragement throughout my Master's studies and throughout the process of researching and writing this thesis. His love and passion for science and research profoundly influenced me and led me to devote myself to scientific research.

I would like to thank Dr. Liangliang Wang for being my supervisor and offering me helpful suggestions on this thesis.

My sincere gratitude also goes to Dr. Jiannan Wang, the internal examiner of my supervisory committee, for his kind help and insightful comments.

I am deeply grateful to Dr. Lingyang Chu for his continuous help and support throughout my research.

I am also grateful to my lab mates for their kind help and encouragement. A particular acknowledgement gose to Dong-Wan Choi, Juhua Hu, Xiangbo Mao, Yu Yang, Chuancong Gao, Xiao Meng, Xiaojian Wang, Zicun Cong, Chen Lin, Zhefeng Wang, Mingtao Lei, Xiang Zhu, Dongxiang Zhang, Yajie Zhou, Xia Hu, Yanyan Zhang.

Last but not least, I would like to express my sincerest gratitude to my parents, Xianhong Zhao and Zhifang Yang for giving birth to me and offering unconditional love and support to me. Also, I would like to thank Xiuyu Wu, for his love and company.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we first introduce the background and motivations of this thesis. Then, we summarize the major ideas and contributions of our work. Finally, we introduce the structure of the thesis.

1.1 Background and Motivations

Clean data is rare in practice, real-world data sets are easily corrupted and thus contain noise. Sometimes real-world data sets even contain heavy noise. However, a classification task highly relies on the reliability and accuracy of data sets. Thus, designing and developing classification algorithms that take heavy noise into consideration is of great practical importance.

There are two types of noise in real data sets for classification tasks: feature (or attribute) noise and label (or class) noise [41]. Among these two kinds of noise in data sets, label noise draws particular attention in literature since it negatively affects classification results more easily than feature noise [10].

Many studies were dedicated to studying classification methods that can reduce the undesirable influence of label noise. To be specific, some studies focus on comparing different classification algorithms and selecting the ones that are relatively robust against label noise [3, 10, 25]. Unfortunately, most of the popular classification algorithms in machine learning are not completely robust against label noise. Besides, another group of work is inspired by an obvious and natural idea: we can clean the corrupted data by detecting the instances with label noise. The most challenging part of this approach is finding ways to detect noisy labels [8, 40]. Moreover, one can also gain the information of label noise by studying and modeling it before or during the learning process. The challenge in this approach is that modeling label noise increases the time complexity and easily causes overfitting [27, 24].

Although some cutting-edge methods have already accomplished improvement compared to the traditional standard classification algorithms in alleviating the adverse impact of

label noise, we notice that there are some limitations. Notice that in order to learn the nature of the label noise, the state-of-the-art algorithms assume that the label noise is class-dependent. However, the work of our thesis does not need any prior knowledge of noise distribution.

1.2 Major Ideas

We propose a Markov chain sampling framework, MCS, to tackle the aforementioned problem. The major ideas of our proposed method are described as follows:

First, the prediction results of the trained classifier contain information about which labels are corrupted and can be used to identify noisy data. Many methods [10, 13, 23, 36, 21, 26, 17, 2] take advantage of this idea. These methods identify instances misclassified by the classifier as corrupted instances. However, as explained in [10], the learned classifier is trained based on corrupted training data, and it may not be accurate. Since one-time training and identifying is not accurate, we come up with an ensemble method. The key idea is to identify the corrupted instances using the ensemble outputs of a sequence of classifiers. The sequence of classifiers is trained on different subsets of training data by sampling the states of a carefully designed Markov chain with random walk. Each state of the Markov chain uniquely relates to one subset of training data.

1.3 Contributions

The main results and contributions of the thesis are summarized as bellow:

- We develop a novel framework, MCS, to deal with classification with heavy label noise, such that a wide spectrum of classification algorithms can be plugged in. We use Logistic Regression and Support Vector Machine in this thesis to demonstrate and assess the effect and generalizability. The proposed MCS framework does not need any prior knowledge of noise distribution. Even in situations where label noise is not class-dependent, the MCS framework can still produce adequate results.

- We show the correctness and efficiency of our proposed framework theoretically and experimentally. We theoretically prove that the designed Markov chain holds one unique stationary distribution, and the probabilities of good states in the stationary distribution of the Markov chain are much larger than that of bad states. We also show that the time complexity of MSC is $\mathcal{O}(NT)$, where T is the running time of the classifier plugged in. The MCS framework is scalable with respect to input size. We experimentally show that sampling 100 steps in the Markov chain is good enough to approximate the stationary distribution and can achieve satisfactory results in practice.

- We experimentally show that MCS outperforms the original classifier plugged in as well as some cutting-edge algorithms on both synthetic data sets and real data sets. In our experiments, the MCS framework can handle heavy label noise, and can achieve satisfactory results, even at high noise rates.
- Our proposed MCS framework is insensitive to parameters, the only three parameters, N, α and β, do not affect the classification results in a wide range.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we review and analyze the related work on classification with label noise. In Chapter 3, we introduce the problem definition as well as the MCS framework. We also theoretically prove the correctness and effectiveness of MCS. We present our experimental study in Chapter 4, and conclude the thesis in Chapter 5.

Chapter 2

Related Work

Classification with label noise draws a lot of attention since label noise may lead to many negative consequences, such as decrease in classification performance, increase in the complexity of models and difficulties in selecting features [10, 41, 32]. The existing studies related to this problem can be grouped into three main categories [10]. The first category studies the robustness of different classification algorithms in the presence of label noise [25, 28, 9]. Second, some studies filter out noisy data to improve data quality thus the classification results [13, 23, 36, 7, 4, 29, 33, 39, 8, 40, 38, 17, 21]. The third category aims at modeling label noise to gain and utilize information about label noise [27, 24].

2.1 Robust Model Approach

This approach studies the robustness properties of different classification algorithms.

In [25], Manwani *et al*. analyzed the label noise-tolerance properties of the empirical risk minimization (ERM) framework under different loss functions for the binary classification problem. ERM is a popular methodology in classification which selects a suitable loss function and tries to minimize the risk (expectation of the loss). They proposed that if a learning algorithm achieves the same classification accuracy after learning on clean data and noisy data, then this learning algorithm is noise tolerant. They theoretically showed that the recent loss functions, including the log loss and the hinge loss, are not completely noise tolerant. The 0-1 loss function has the best noise-tolerant property, however, the 0-1 loss is non-convex and intractable in practice. There is an assumption of label noise in [25], the probability that the label of an instance is corrupted is a function of the feature vector of the instance.

Some work empirically studied the robustness of different learning methods in the presence of label noise. Nettleton *et al*. [28] systematically compared the effect of different types of noise, including label noise and attribute noise [41], to four different classification models. Specifically, they investigated the Naïve Bayes probabilistic classifier [22], the C4.5

decision tree [31], the IBK instance-based learner [1] and the SMO [30] support vector machine [37]. They found that Naïve Bayes classifier is most robust against noise among these four classifiers. Folleco *et al*. [9] showed that imbalanced or skewed data sets may aggravate the negative effect of label noise. They studied the robustness performance of 11 different commonly used classifiers in imbalanced noisy data sets. The experiment results showed that all classifiers are negatively influenced by label noise, though to different degrees. They experimentally showed that the random forest ensemble classifier [5] holds the best noise robustness properties.

Nonetheless, the limitations of this approach are obvious. First, this approach is passive. Instead of improving or modifying standard algorithms, they only investigate the robustness properties of existing commonly used classifiers. Second, even the most robust classifier is only effective when the portion of label noise is small. The performance of the robust classifier declines dramatically when the portion of label noise is large. Common classification algorithms are not completely robust against label noise [10, 3].

2.2 Data Filtering Approach

In this approach, instances with noisy labels are identified and treated before the training process. The corrupted labels can be simply removed or relabeled at the very first step [10].

One idea is to use the predictions from classifiers to detect mislabeled instances, which also inspires the work of this thesis. This category is called the classification filtering methods [10, 13, 23]. In [36], Thongkam *et al*. trained a SVM on all training data and then identified the instances misclassified by SVM as mislabeled ones. This work was further improved by training four classifiers and voting to determine mislabeled instances [26]. However, filtering all the instances misclassified by classifiers is too rigid and dangerous, since the classifiers learned from noisy data may not be always correct.

Another category relates to the kNN classifiers [7, 4]. It is showed in [29, 33, 39] that the kNN classifiers are sensitive to label noise, thus some methods aim at adding or modifying the rules of the kNN algorithms to identify noisy labels, then remove or relabel the corresponding instances [8, 40, 39, 38]. For instance, Delany *et al*. [8] proposed the blame-based noise reduction (BBNR) method. If an instance causes misclassification of another instance, it will be filtered out by the BBNR method. Wilson *et al*. [40] introduced the rule that filters out instances whose labels mismatch the majority labels of their neighbors. More related methods can be found in [39, 38], Wilson *et al*. investigated some kNN-based data filtering methods and analyzed the comparison results. However, the methods belonging to this category are heuristics and may not be effective near the classification boundary [14]. Moreover, the performance of these methods may be significantly affected by the selection of k.

Some data filtering methods are based on thresholds. They calculate a score for each instance based on some measures and filter out the instances which outstrip a certain threshold [35, 12, 11]. However, this category is similar to the outlier or anomaly detection problem, which is never trivial to deal with [10]. Sometimes it is hard to distinguish true exceptions from mislabeled instances.

The filtering/cleansing algorithms are inclined to remove a considerable amount of instances, which may contain important information for classification [10].

2.3 Inherently Noise-Tolerant Learning Approach

This approach models label noise before or during the training step to take label noise into consideration. The model built for label noise contains the information of noise and can be embedded in the classification model [10].

Many standard classifiers, including SVM and neural networks, can be modified to combine the learned information of label noise to alleviate the influence of noisy labels. Natarajan *et al.* [27] studied the class-dependent random label noise in the binary classification problem. They proposed two possible methods to modify any given surrogate loss function to make the classifier more robust against label noise. They provided guarantees for risk minimization under random label noise. In [24], Liu *et al.* studied the same type of label noise and proved that the proposed importance reweighting method [24] can be used in this problem for any surrogate loss function. They also provided a method to estimate the noise rate ρ. Nonetheless, these methods take class-dependent random label noise as an assumption and lack the capability to deal with more general situations.

In order to build the model for label noise, these methods need additional parameters, which may increase the time complexity and model complexity. High model complexity sometimes causes over-fitting.

In our thesis, the proposed MCS framework can learn the probability of being correctly labeled for each instance by using ensemble outputs of a sequence of classifiers. The learned probabilities can be used in two directions. First, the learned probabilities can be used to filter out data. Moreover, the sequence of classifiers is learned by using the predictions from classifiers. Thus, the MCS framework is similar to the data filtering approach mentioned in Section 2.2. Our proposed MCS framework is especially similar to the method proposed in [36]. However, as mentioned in Section 2.2, the data filtering method in [36] conducts classification only once and filters out all the instances misclassified by classifiers, which is too rigid and dangerous. Thus, we propose the MCS framework that uses predictions from classifiers in a fuzzy way to avoid the aforementioned danger. The MCS framework uses ensemble outputs of a sequence of classifiers to obtain a more robust classification model. Second, the learned probabilities can be embedded in the classification model to modify the standard model. Thus, the MCS framework is also similar to the inherently noise-tolerant

learning approach mentioned in Section 2.3. Nevertheless, the methods mentioned in Section 2.3 assume that the label noise is class-dependent. Thus, these methods cannot handle the scenarios where the label noise is not class-dependent. Our proposed MCS framework does not assume any prior knowledge of noise distribution.

Chapter 3

Problem Definition and MCS Framework

In this chapter, we formulate our problem of classification with heavy label noise. Then, we introduce the MCS framework.

3.1 Classification with Heavy 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 $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ 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.

In many real-world applications, the true labels may be corrupted by noise. We model this by a sample of label-corrupted instances $\hat{S} = \{(x_1, \hat{y}_1), \cdots, (x_n, \hat{y}_n)\}$, where $\hat{y}_i$ is the possibly *corrupted label* of instance $(x_i, \hat{y}_i)$. 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 *wrongly labeled* instance if $y_i \neq \hat{y}_i$.

Let $\hat{S}^+$ and $\hat{S}^-$ denote the set of positive instances and the set of negative instances in $\hat{S}$, respectively. We denote by ρ^+ and ρ^- the proportions of wrongly labeled instances in $\hat{S}^+$ and $\hat{S}^-$, respectively. Similar to the conventional classification tasks with label noise [24, 27], we assume that the proportion of wrongly labeled instances are less than or equal to 50%, that is, $\rho^+ \leq 0.5$ and $\rho^- \leq 0.5$. We regard $\rho^+ \geq 0.4$ and $\rho^- \geq 0.4$ as heavy label noise.

Now, we define the problem of Classification with Heavy Label Noise (CHLN) as follows.

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

The CHLN problem is more challenging than the conventional classification tasks with label noise [24, 27], since no prior knowledge on the noise distribution is available and any instance in $\hat{S}$ may be wrongly labeled.

3.2 The Markov Chain Sampling Framework

In this section, we introduce the Markov Chain Sampling (MCS) framework to solve the CHLN problem. The key idea is to identify the wrongly labeled instances in $\hat{S}$ by using ensemble outputs of a sequence of classifiers. The sequence of classifiers is trained on different subsets of $\hat{S}$ obtained by sampling the states of 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 $\hat{S}$.

In the following, we first introduce the structure of the Markov chain, then we prove that the Markov chain has a stationary distribution, which can be used to effectively identify the wrongly labeled instances. Finally, we present the MCS framework, which samples the Markov chain by random walk to robustly identify the wrongly labeled instances and trains the predictive model $\mathcal{M}$.

3.2.1 The Structure of the Markov Chain

We consider a Discrete Time Markov Chain (DTMC), 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 *instance subsets* of $\hat{S}$. We assign a unique index to each non-empty instance subset of $\hat{S}$ and denote the i-th instance 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 k-th dimension of $\mathbf{c}_i$, denoted by $\mathbf{c}_{ik}$ in Equation 3.1, indicates whether the k-th instance $(x_k, \hat{y}_k)$ is contained by the i-th instance subset $\hat{S}_i$. In this way, we can write $\hat{S}_i = \{(x_k, \hat{y}_k) \in \hat{S} \mid \mathbf{c}_{ik} = 1\}$.

$$\mathbf{c}_{ik} = \begin{cases} 1 & \text{if } (x_k, \hat{y}_k) \in \hat{S}_i \\ 0 & \text{if } (x_k, \hat{y}_k) \notin \hat{S}_i \end{cases} \tag{3.1}$$

For two states $\mathbf{c}_i$ and $\mathbf{c}_j$, a transition from $\mathbf{c}_i$ to $\mathbf{c}_j$ is equivalently a transition from $\hat{S}_i$ to $\hat{S}_j$. We perform this transition by first training a classification model $\mathcal{M}_i$ on $\hat{S}_i$, then obtaining $\hat{S}_j$ by sampling instances from $\hat{S}$ using the probabilities of classification results produced by $\mathcal{M}_i$. Traditional models that produce probabilities of classification results can all be used to train $\mathcal{M}_i$. In this thesis, we use Logistic Regression (LR) [18] and the kernelized Support Vector Machine (KSVM) [34] to demonstrate the feasibility and effectiveness.

Denote by $\tilde{y}_k \in \{+1, -1\}$ the classification result of feature $x_k \in \mathcal{X}$ produced by model $\mathcal{M}_i$. Let $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ represent the probability that model $\mathcal{M}_i$ outputs label $\tilde{y}_k = \hat{y}_k$

for feature x_k. Since $\mathcal{M}_i$ is trained using all instances in $\hat{S}_i$, $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ can be regarded as the belief that $(x_k, \hat{y}_k)$ is correctly labeled and this belief is supported by all instances in $\hat{S}_i$. Although $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ can be affected by the wrongly labeled instances in $\hat{S}_i$, we prove in Section 3.2.2 that, under some simple conditions, incorporating such probabilities into the proposed Markov chain framework effectively identifies wrongly labeled instances and leads to better classification performance.

To obtain $\hat{S}_j$, we sample each instance $(x_k, \hat{y}_k) \in \hat{S}$ with probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$. That is, $\mathbb{P}((x_k, \hat{y}_k) \in \hat{S}_j \mid \mathbf{c}_i) = \mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$. Intuitively, a larger $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ indicates a stronger belief that instance $(x_k, \hat{y}_k)$ is correctly labeled. Therefore, $(x_k, \hat{y}_k)$ has a higher probability to be sampled.

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

$$\mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i) = \prod_{k=1}^{|\hat{S}|} \mathbb{P}(\mathbf{c}_{jk} = 1 \mid \mathbf{c}_i) \tag{3.2}$$

Until now, we have defined the $2^{|\hat{S}|} - 1$ 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 i-th row and the j-th column is $P_{ij} = \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)$.

Next, we prove that the Markov chain C has a stationary distribution, which can be used to effectively identify wrongly labeled instances in $\hat{S}$.

3.2.2 Why Stationary Distribution Works

In this section, we first introduce an assumption of the Markov chain C proposed in Section 3.2.1. Then, we prove that C has a unique stationary distribution, which can be used to effectively identify the correctly labeled and wrongly labeled instances in $\hat{S}$.

Given an instance set $\hat{S}_i \subseteq \hat{S}$ and the corresponding model $\mathcal{M}_i$, we say $\hat{S}_i$ is *more separable* by $\mathcal{M}_i$ if, for most of the instances $(x_k, \hat{y}_k) \in \hat{S}_i$, there is a larger difference between $\mathbb{P}(\tilde{y}_k = +1 \mid x_k, \mathbf{c}_i)$ and $\mathbb{P}(\tilde{y}_k = -1 \mid x_k, \mathbf{c}_i)$.

Intuitively, the states in the Markov chain C can be divided into two categories. The *good states*, denoted by C^+, are the set of states dominated by correctly labeled instances. The *bad states*, denoted by C^-, are the set of states corrupted by wrongly labeled instances. Since the wrongly labeled instances are randomly corrupted, it is reasonable to assume that the set of instances associated with a good state are more separable than that of a bad state.

Recall that each instance $(x_k, \hat{y}_k)$ in $\hat{S}$ is sampled with probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ produced by model $\mathcal{M}_i$, therefore, for a good state $\mathbf{c}_i \in C^+$, the probability of sampling a

correctly labeled instance is much larger than the probability of sampling a wrongly labeled instance. This means that, starting from a good state in the Markov chain C, the probability of jumping to a good state is much larger than the probability of jumping to a bad state.

Since the set of instances associated with a bad state is less separable than that of a good state, the difference between the probability of sampling a correctly labeled instance and the probability of sampling a wrongly labeled instance for a bad state $\mathbf{c}_j \in C^-$ is smaller than the difference of such probabilities for a good state $\mathbf{c}_i \in C^+$. This further indicates that the difference between the probability of jumping to a good state and the probability of jumping to a bad state when starting from a bad state is smaller than the difference of probabilities when starting from a good state. Mathematically, we write this as

$$\forall \mathbf{c}_i \in C^+, \forall \mathbf{c}_j \in C^-, \frac{\mathbb{P}(C^- \mid \mathbf{c}_i)}{\mathbb{P}(C^+ \mid \mathbf{c}_i)} \ll \frac{\mathbb{P}(C^+ \mid \mathbf{c}_j)}{\mathbb{P}(C^- \mid \mathbf{c}_j)} \tag{3.3}$$

where $\mathbb{P}(C^+ \mid \cdot)$ and $\mathbb{P}(C^- \mid \cdot)$ represent the probabilities of jumping from a state in C to any good state in C^+ and to any bad state in C^-, respectively.

Since $\mathbb{P}(C^+ \mid \mathbf{c}_i) + \mathbb{P}(C^- \mid \mathbf{c}_i) = 1$ and $\mathbb{P}(C^+ \mid \mathbf{c}_j) + \mathbb{P}(C^- \mid \mathbf{c}_j) = 1$, we can derive from Equation 3.3 that

$$\forall \mathbf{c}_i \in C^+, \forall \mathbf{c}_j \in C^-, \mathbb{P}(C^- \mid \mathbf{c}_i) \ll \mathbb{P}(C^+ \mid \mathbf{c}_j) \tag{3.4}$$

Next, we prove in Theorem 3.1 that the proposed Markov chain C has a unique stationary distribution.

Theorem 3.1. The Discrete Time Markov Chain C as defined above has a unique stationary distribution $\boldsymbol{\pi}$ which satisfies $\boldsymbol{\pi} P = \boldsymbol{\pi}$.

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

First, since every state in C has a self-loop, C is aperiodic. Second, since $P_{ij} > 0$ for all i and j, the proposed Markov chain C is *irreducible* [15]. According to Grimmett *et al.* [16], since C is irreducible and has a finite number of states, C is a positive recurrent Markov chain. In sum, the proposed Markov chain C is aperiodic and positive recurrent.

According to [15], every aperiodic and positive recurrent Markov chain has one unique stationary distribution $\boldsymbol{\pi}$ which satisfies $\boldsymbol{\pi} P = \boldsymbol{\pi}$. The proof follows. □

In Theorem 3.2, we prove that the stationary distribution $\boldsymbol{\pi}$ is a robust identifier for both correctly and wrongly labeled instances in $\hat{S}$ when the assumption in Equation 3.4 holds.

Theorem 3.2. Denote by $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$ the i-th and the j-th entries of the stationary distribution $\boldsymbol{\pi}$ of the Markov chain C, respectively. If Equation 3.4 holds, then $\sum_{\mathbf{c}_i \in C^+} \boldsymbol{\pi}_i \gg \sum_{\mathbf{c}_j \in C^-} \boldsymbol{\pi}_j$.

Proof. Since $\boldsymbol{\pi}$ is the stationary distribution of the Markov chain C, it follows that for any $\mathbf{c}_i \in C^+$ and $\mathbf{c}_j \in C^-$, the following equation holds.

$$\mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)\boldsymbol{\pi}_i = \mathbb{P}(\mathbf{c}_i \mid \mathbf{c}_j)\boldsymbol{\pi}_j$$

Therefore, we have

$$\sum_{\mathbf{c}_i \in C^+} \sum_{\mathbf{c}_j \in C^-} \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i)\boldsymbol{\pi}_i = \sum_{\mathbf{c}_j \in C^-} \sum_{\mathbf{c}_i \in C^+} \mathbb{P}(\mathbf{c}_i \mid \mathbf{c}_j)\boldsymbol{\pi}_j. \tag{3.5}$$

Recall that

$$\begin{cases} \mathbb{P}(C^- \mid \mathbf{c}_i) = \sum_{\mathbf{c}_j \in C^-} \mathbb{P}(\mathbf{c}_j \mid \mathbf{c}_i) \\ \mathbb{P}(C^+ \mid \mathbf{c}_j) = \sum_{\mathbf{c}_i \in C^+} \mathbb{P}(\mathbf{c}_i \mid \mathbf{c}_j) \end{cases}$$

we can derive from Equation 3.5 that

$$\sum_{\mathbf{c}_i \in C^+} \mathbb{P}(C^- \mid \mathbf{c}_i)\boldsymbol{\pi}_i = \sum_{\mathbf{c}_j \in C^-} \mathbb{P}(C^+ \mid \mathbf{c}_j)\boldsymbol{\pi}_j \tag{3.6}$$

Define $\mathbb{P}^*(C^- \mid \mathbf{c}_i) = \max_{\mathbf{c}_i \in C^+} \mathbb{P}(C^- \mid c_i)$, it follows from Equation 3.4 that

$$\sum_{\mathbf{c}_i \in C^+} \frac{\mathbb{P}(C^- \mid \mathbf{c}_i)}{\mathbb{P}^*(C^- \mid \mathbf{c}_i)}\boldsymbol{\pi}_i = \sum_{\mathbf{c}_j \in C^-} \frac{\mathbb{P}(C^+ \mid \mathbf{c}_j)}{\mathbb{P}^*(C^- \mid \mathbf{c}_i)}\boldsymbol{\pi}_j \gg \sum_{\mathbf{c}_j \in C^-} \boldsymbol{\pi}_j$$

Considering that

$$\sum_{\mathbf{c}_i \in C^+} \boldsymbol{\pi}_i > \sum_{\mathbf{c}_i \in C^+} \frac{\mathbb{P}(C^- \mid \mathbf{c}_i)}{\mathbb{P}^*(C^- \mid \mathbf{c}_i)}\boldsymbol{\pi}_i$$

we have

$$\sum_{\mathbf{c}_i \in C^+} \boldsymbol{\pi}_i \gg \sum_{\mathbf{c}_j \in C^-} \boldsymbol{\pi}_j$$

The proof follows. □

According to Theorem 3.2, the probabilities of the good states in the stationary distribution of C are much larger than that of the bad states. Therefore, we can easily identify the good states and the bad states of C using the stationary distribution $\boldsymbol{\pi}$.

The stationary distribution $\boldsymbol{\pi}$ can be further used to compute

$$\overline{\mathbf{c}} = \sum_{\mathbf{c}_i \in C} \mathbf{c}_i \boldsymbol{\pi}_i \tag{3.7}$$

where the k-th entry $\overline{\mathbf{c}}_k$ is the probability that instance $(x_k, \hat{y}_k) \in \hat{S}$ is correctly labeled. We use $\overline{\mathbf{c}}$ to identify correctly and wrongly labeled instances, because $\overline{\mathbf{c}}$ comprehensively embeds the rich distribution information in $\boldsymbol{\pi}$, which leads to more robust identification performance. We can use $\overline{\mathbf{c}}_k$ of each instance to filter out noisy instances in training data.

If $\overline{\mathbf{c}}_k$ is larger than a threshold α, $(x_k, \hat{y}_k)$ is identified as a correctly labeled instance. Otherwise, $(x_k, \hat{y}_k)$ is identified as a wrongly labeled instance. We will discuss the effect of threshold α in Section 4.3. We can also embed $\overline{\mathbf{c}}_k$ in the standard classification model to make the model more robust against label noise.

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 how to approximate $\boldsymbol{\pi}$ by sampling the proposed Markov chain.

3.2.3 The Markov Chain Sampling Method

According to [16], the stationary distribution $\boldsymbol{\pi}$ of the Markov chain C can be approximated by iteratively sampling the states of C using random walk.

Considering that we have no prior knowledge on the noise distribution in $\hat{S}$, a good starting state to start the random walk is the state associated with all instances in $\hat{S}$. We denote this state by $\mathbf{c}_1 = \mathbf{1}$, which is a $|\hat{S}|$-dimensional vector of all ones.

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 obtain 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 $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$. Last, we 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 exponentially with respect to the volume of $\hat{S}$, the probability of redoing the sampling is very small, especially when the volume of $\hat{S}$ is large.

As shown in Algorithm 1, we continue the above sampling process to sample N times. Then we use the empirical distribution of the sampled states as the approximated stationary distribution, denoted by $\hat{\boldsymbol{\pi}}$. We can substitute $\boldsymbol{\pi}_i$ in Equation 3.7 with $\hat{\boldsymbol{\pi}}_i$ to compute $\overline{\mathbf{c}}$ and identify the correctly labeled and wrongly labeled instances. At last, we can train our final classification model $\mathcal{M}$ with the learned probability set $\overline{\mathbf{c}}$. As mentioned in Section 3.2.2, the probability set $\overline{\mathbf{c}}$ can be used to filter data according to the threshold α, which is obtained by either flipping the wrong labels or discarding the wrongly labeled instances. The probability set $\overline{\mathbf{c}}$ can also be embedded in standard classification model to make the model more robust to label noise.

As mentioned before, we use LR [18] and KSVM [34] to train $\mathcal{M}_i$ in this thesis. However, the initial classification result of KSVM for each instance is not in the form of $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$. Thus, we map the initial classification result to the sigmoid function $f(r) = \frac{1}{1+e^{\beta r}}$, where r is the initial classification result of KSVM and β is a parameter need to tune.

We explain the algorithms derived from the MCS framework as follows. First, we plug LR [18] in the MCS framework and get the MCS-LR algorithm. The MCS-LR algorithm discards instances whose $\overline{\mathbf{c}}_k$ are less than the threshold α. Second, we plug KSVM [34] in the MCS framework and get three MCS-KSVM algorithms. The MCS-KSVMD algorithm discards instances whose $\overline{\mathbf{c}}_k$ are less than the threshold α. The MCS-KSVMF algorithm flips labels of instances whose $\overline{\mathbf{c}}_k$ are less than the threshold α. The MCS-KSVMW algorithm takes $\overline{\mathbf{c}}$ as the weight to train the final weighted KSVM model.

Algorithm 1 Markov Chain Sampling Method

Input: The set of label-corrupted instances $\hat{S}$.
Output: The approximated stationary distribution $\hat{\boldsymbol{\pi}}$.

Initialization: $i \leftarrow 1$, $\mathbf{c}_1 \leftarrow \mathbf{1}$, $\mathbb{C} \leftarrow \emptyset$.
for $i \leq N$ **do**
  Train model $\mathcal{M}_i$ on $\hat{S}_i$.
  $j \leftarrow i+1$, $\mathbf{c}_j \leftarrow \mathbf{0}$.
  for each $(x_k, \hat{y}_k) \in \hat{S}_i$ **do**
    Set $\mathbf{c}_{jk} = 1$ with probability $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$.
  end for
  if $\sum_k \mathbf{c}_{jk} > 0$ **then**
    $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbf{c}_j$.
    $i \leftarrow i+1$.
  end if
end for
Obtain $\hat{\boldsymbol{\pi}}$ by the empirical distribution of the sampled states in $\mathbb{C}$.
return $\hat{\boldsymbol{\pi}}$.

The proposed Markov chain 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 $\mathcal{M}_i$. As demonstrated by the experiments in Chapter 4, the performance of the proposed method is scalable to the value of N, and $N = 100$ is good enough to achieve a clearly better classification performance than the state-of-the-art methods. The method is summarized in Algorithm 1.

Chapter 4

Experiments

In this Chapter, we evaluate the performance of our proposed MCS framework and compare it with the state-of-the-art noisy label classification methods including (1) the kernelized SVM (KSVM) [34], (2) C-Support Vector Classification Filter (C-SVCF) [36], (3) Importance Reweighting with cross-validation method in hinge loss (IWℓ) [24], and (4) Importance Reweighting method in hinge loss (eIWℓ) [24]. The code for KSVM is provided by Chang *et al*. [6]. We carefully implement C-SVCF in MATLAB R2015a. The code for IWℓ and eIWℓ is provided by Liu *et al*. [24].

For the proposed MCS framework, we derive two methods MCS-LR and MCS-KSVM, which are obtained by training model $\mathcal{M}_i$ with Logistic Regression (LR) [18] and KSVM [34], respectively.

All compared methods are evaluated with default parameter settings. For the proposed MCS methods, we use $N = 100$, $\alpha = 0.5$, $\beta = 2$. We discuss the effects of parameters N, α, β in detail in Section 4.3.

All experiments are implemented using MATLAB R2015a. We use a PC with Intel Core i7-3770 CPU (3.40GHz) and 16 GB memory running Windows 7.

4.1 Data Sets

The following data sets are used in our experiments.

- *Synthetic Data Set SD1.* The synthetic data set SD1 contains 200 instances in two dimensions. There are 100 true positive instances and 100 true negative instances. These two classes are generated by two different Gaussian Distributions. We add noise to the training sets by setting $\rho^+ = \rho^- = 0.2$. The labels of training instances are flipped randomly according to given noise rates ρ^+ and ρ^-.

- *Synthetic Data Set SD2.* The synthetic data set SD2 also contains 200 instances in two dimensions, including 100 true positive instances and 100 true negative instances.

The noise rates are $\rho^+ = \rho^- = 0.2$ as well. However, instead of sampling the noise data uniformly, we add the noise according to a 2-dimensional Gaussian distribution with positive correlation between the two feature dimensions.

- *Synthetic Data Set SD3.* The synthetic data set SD3 consists of 200 instances in ten dimensions. We add noise to SD3 in three different noise rate settings. The labels of training instances are flipped randomly according to given noise rates ρ^+ and ρ^-.

- *Banknote Authentication Data Set.* The Banknote authentication data set is a public data set in the UCI Machine Learning Repository [19]. It contains 1372 instances in four dimensions. The labels of training instances are flipped randomly according to given noise rates ρ^+ and ρ^-.

- *UCI Benchmark Data Set.* The UCI Benchmark data set is a standard classification data set provided by Gunnar Rätsch [20], it contains 13 data sets. For each Benchmark data set, we conduct experiments on six different settings of ρ^+ and ρ^-. The labels of training instances are flipped randomly according to given noise rates ρ^+ and ρ^-.

For each data set, we randomly select 75% instances as training data and 25% instances as testing data. Only the training data is corrupted by label noise. We test all compared methods on each data set for ten times and report their mean classification accuracies and variances.

4.2 Evaluation Metrics

To fairly evaluate the classification performance, we use the same evaluation criteria as Liu *et al.* [24], that is

$$Accuracy = \frac{\#\text{Correctly Classified Instances}}{\#\text{All Classified Instances}} \tag{4.1}$$

In the data filtering category in Chapter 2, there are two types of errors [10]. The first type is removing clean instances by mistake. We denote it as

$$ER_1 = \frac{\#\text{ of correctly labeled instances which are removed}}{\#\text{ of correctly labeled instances}} \tag{4.2}$$

The second type is retaining noisy instances and can be represented as

$$ER_2 = \frac{\#\text{ of mislabeled instances which are not removed}}{\#\text{ of mislabeled instances}} \tag{4.3}$$

To test the running time of the algorithms, we use second as the unit.

4.3 Effect of Parameters

We analyze the effect of parameters including N and α in the MCS framework, and β in MCS-KSVM algorithms.

4.3.1 Effect of Parameters for MCS Framework

N is a significant parameter of the MCS framework, and it determines the number of jumping steps in the constructed Markov chain. This experiment is conducted by setting N to different values ranging from 1 to 150 for the MCS-LR algorithm and the MCS-KSVMD algorithm. The experiment results for MCS-LR and MCS-KSVMD are shown in Figure 4.1 and Figure 4.2, respectively. We observe that the classification accuracies increase as N increasing from 1 to 40 and stay stable after 40 steps on most data sets. The accuracies stay stable after 100 steps on all data sets. Thus, we set $N = 100$ as default in our experiments. Therefore, we experimentally show that conducting 100 steps of random walk is good enough to approximate the stationary distribution of the constructed Markov chain.

As mentioned in Chapter 3, if we use $\overline{\mathbf{c}}$ to filter out data, α is the threshold determining which instances are filtered out. The parameter α has an effect on classification accuracies. In this experiment, we set α to different values between 0 and 1 for the MCS-LR algorithm and the MCS-KSVMD algorithm to investigate the effect of α on accuracies. The experiment results are shown in Figure 4.3 and Figure 4.4. From Figure 4.3 and Figure 4.4, we can observe that the accuracies of MCS-LR and MCS-KSVMD lack stability only when α is close to 0 or 1. The reason is that $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i)$ in $\hat{\boldsymbol{\pi}}$ for most instances satisfy either $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i) \approx 0$ or $\mathbb{P}(\tilde{y}_k = \hat{y}_k \mid x_k, \mathbf{c}_i) \approx 1$ in our experiments. Thus, our algorithm is capable of predicting the true labels of most instances. We also observe that the remaining instances are close to the classification boundary. In conclusion, the classification accuracy of the proposed algorithm stays stable when α varies in a considerable range. Thus, MCS is insensitive to parameter α, and we set $\alpha = 0.5$ as default in our experiments.

4.3.2 Effect of Parameters for MCS-KSVM

Recall in Section 3.2.3, β is a peculiar parameter for the MCS-KSVM algorithms. Figure 4.5 shows the effect of parameter β on the performance of MCS-KSVMD. In general, the MCS-KSVMD algorithm achieves stable results when β alters between 0.5 and 5. We can also observe that the accuracies increase slightly before $\beta = 1$, whereas the accuracies decrease slightly after $\beta = 3$ for most data sets. Thus, MCS-KSVMD performs best when β is set between 1 and 3. In conclusion, the MCS-KSVMD algorithm is insensitive to parameter β, and we set $\beta = 2$ as default in our experiments.

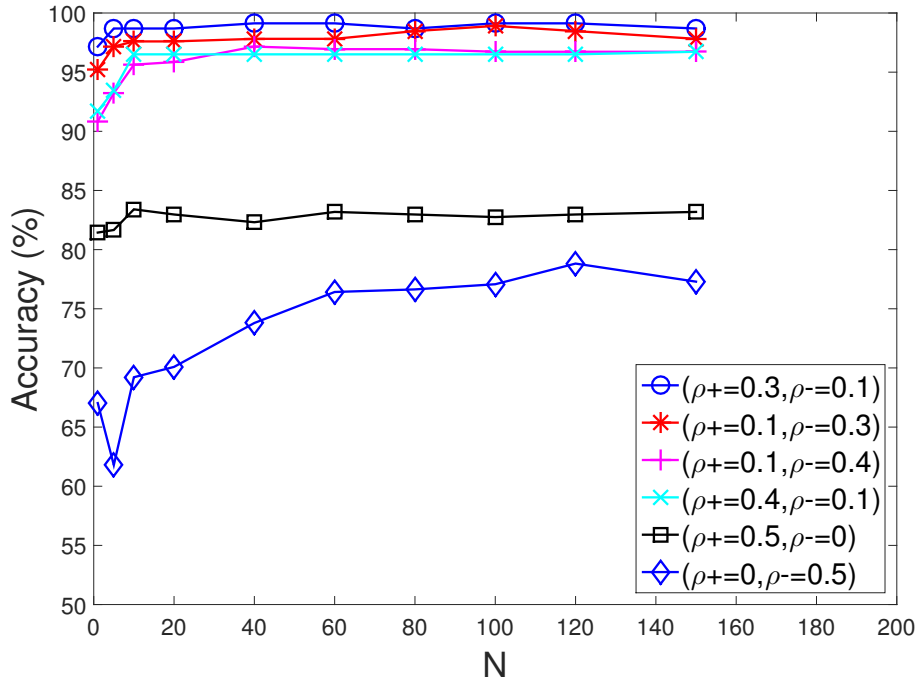


Figure 4.1: Classification accuracies (percentage) of MCS-LR on the Banknote authentication data set in different settings of N.

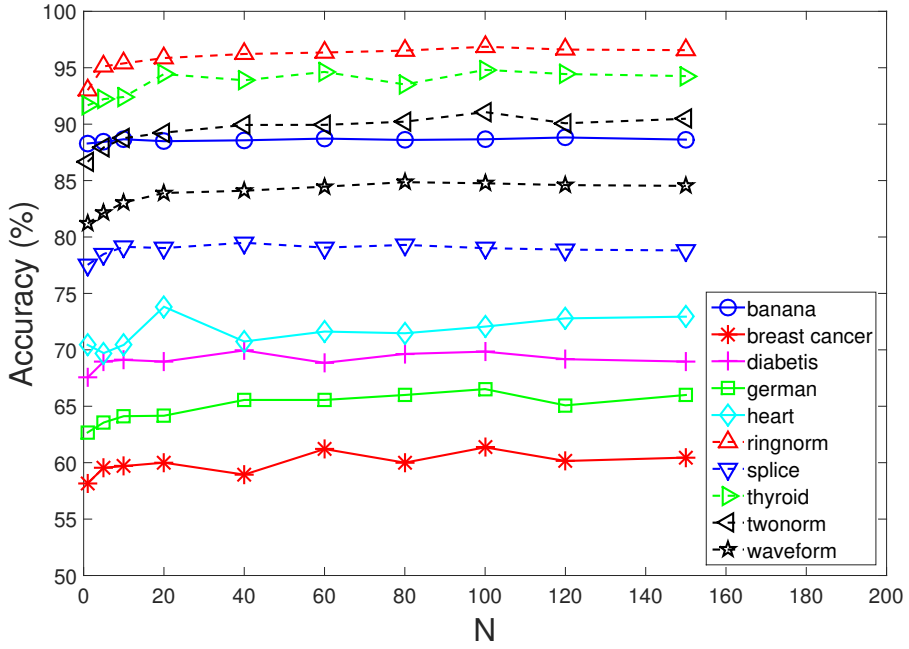


Figure 4.2: Classification accuracies (percentage) of MCS-KSVMD on Benchmark data sets in different settings of N, the noise rate is $\rho^+ = 0.1$ and $\rho^- = 0.3$.

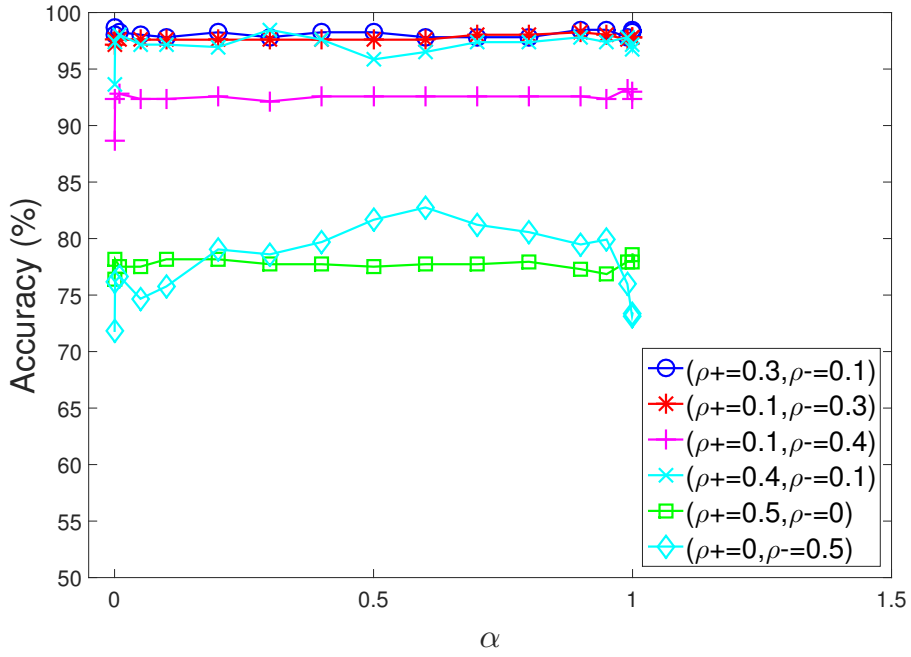


Figure 4.3: Classification accuracies (percentage) of MCS-LR on the Banknote authentication data set in different settings of α.

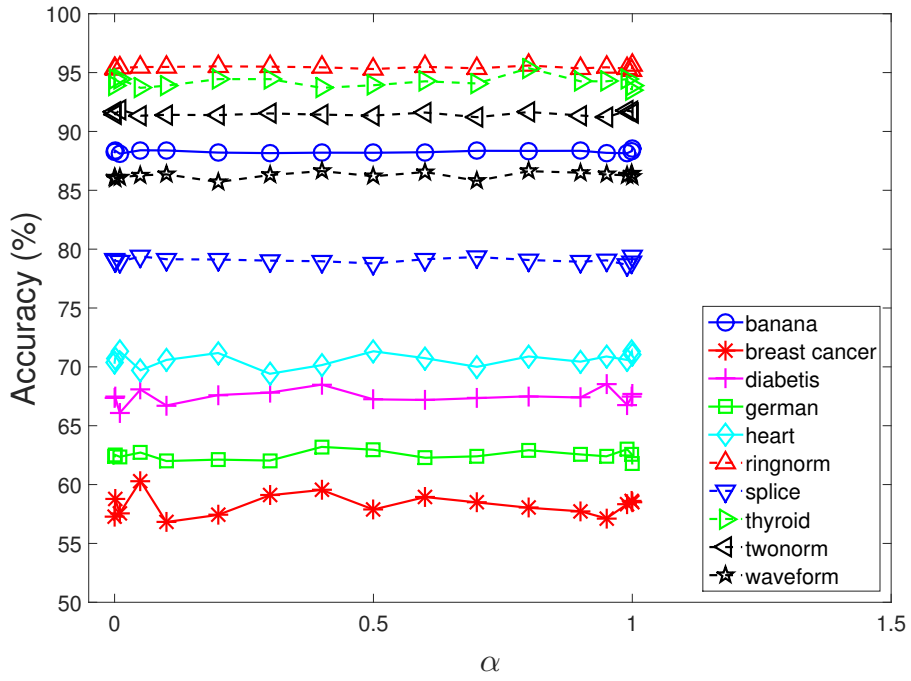


Figure 4.4: Classification accuracies (percentage) of MCS-KSVMD on Benchmark data sets in different settings of α, the noise rate is $\rho^+ = 0.1$ and $\rho^- = 0.3$.

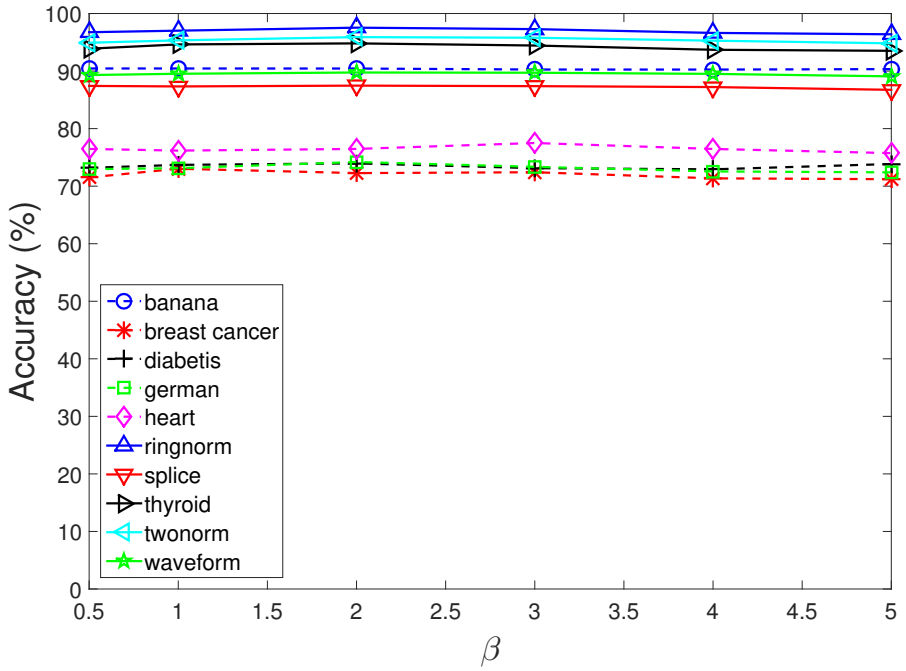


Figure 4.5: Classification accuracies (percentage) of MCS-KSVMD on Benchmark data sets in different settings of β, the noise rate is $\rho^+ = 0.1$ and $\rho^- = 0.1$.

4.4 Results on Synthetic Data

To explain the mechanism and effectiveness of the proposed MCS framework, we conduct experiments on the synthetic data sets for the MCS-LR algorithm. The classification results of the Logistic Regression classifier are easily observed and understood. Thus, we plug the Logistic Regression classifier in the MCS framework to explain our proposed framework.

4.4.1 Results on Synthetic Data Set SD1

We conduct experiments on the synthetic data set SD1. The clean SD1 without label noise is shown in Figure 4.6. Notice that the noise is class-dependent in SD1. Figure 4.7 shows the results of MCS-LR. The circled instances are identified as noise, and the squared ones are noisy instances missed by our algorithm. Figure 4.7 shows that most of the corrupted instances are diagnosed by MCS-LR, especially the ones that are relatively far away from the classification boundary.

4.4.2 Results on Synthetic Data Set SD2

We show the experiment results on the synthetic data set SD2. Figure 4.8 shows the clean SD2 without label noise. It is noticeable in Figure 4.9 and Figure 4.10 that the noise in corrupted SD2 is not class-dependent. As mentioned before, the noise is added according

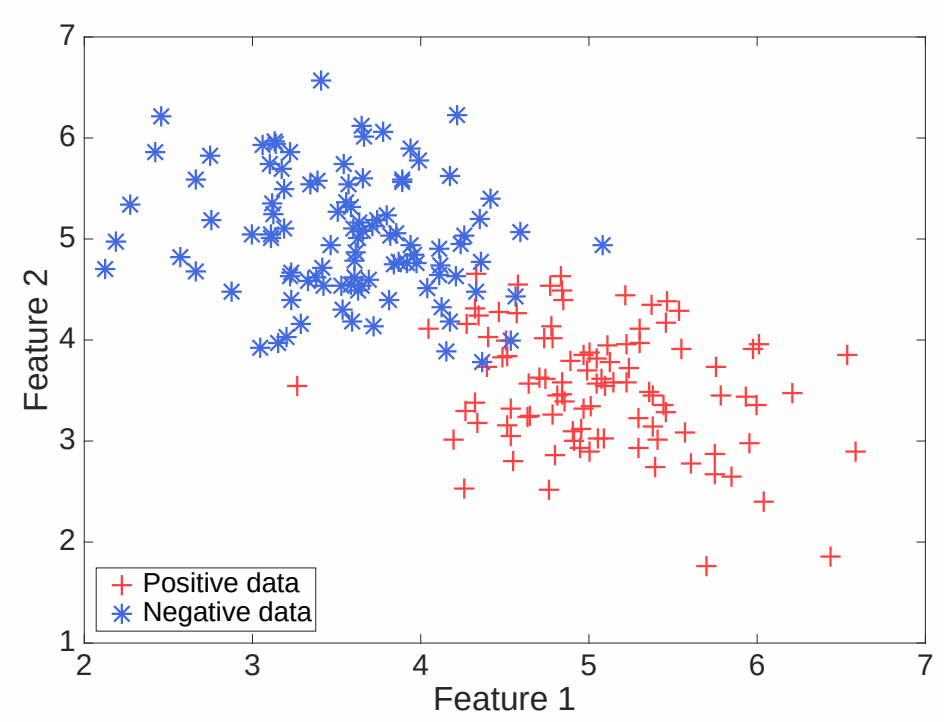


Figure 4.6: The clean synthetic data set SD1 without label noise.

to a 2-dimensional Gaussian distribution with positive correlation between the two feature dimensions.

The noise in the positive class is concentrated at the upper right, and the noise in the negative class is concentrated at the lower left. The classification results of the standard Logistic Regression algorithm are shown in Figure 4.9. The classification boundary is severely distorted by the noisy instances. Figure 4.10 demonstrates the classification results of MCS-LR. The classification boundary is almost unaffected by noise. The different classification boundaries in Figure 4.9 and Figure 4.10 show that MCS-LR outperforms LR in this data set. Furthermore, the accuracy in Figure 4.9 is 86.5% and the accuracy in Figure 4.10 is 95.5%. In summary, we experimentally show that even in situations where the data noise is not class-dependent, and LR is badly affected by noise, MCS-LR can still achieve satisfactory performance.

4.4.3 Results on Synthetic Data Set SD3

We conduct experiments on the synthetic data set SD3 for the standard Logistic Regression algorithm and the MCS-LR algorithm. The experiment results are illustrated in Table 4.1, with the highest results shown in bold in each row. As shown in Table 4.1, our proposed MCS-LR algorithm outperforms the standard Logistic Regression algorithm. The reason is that MCS-LR can identify most of the corrupted instances and filter them out. Thus,

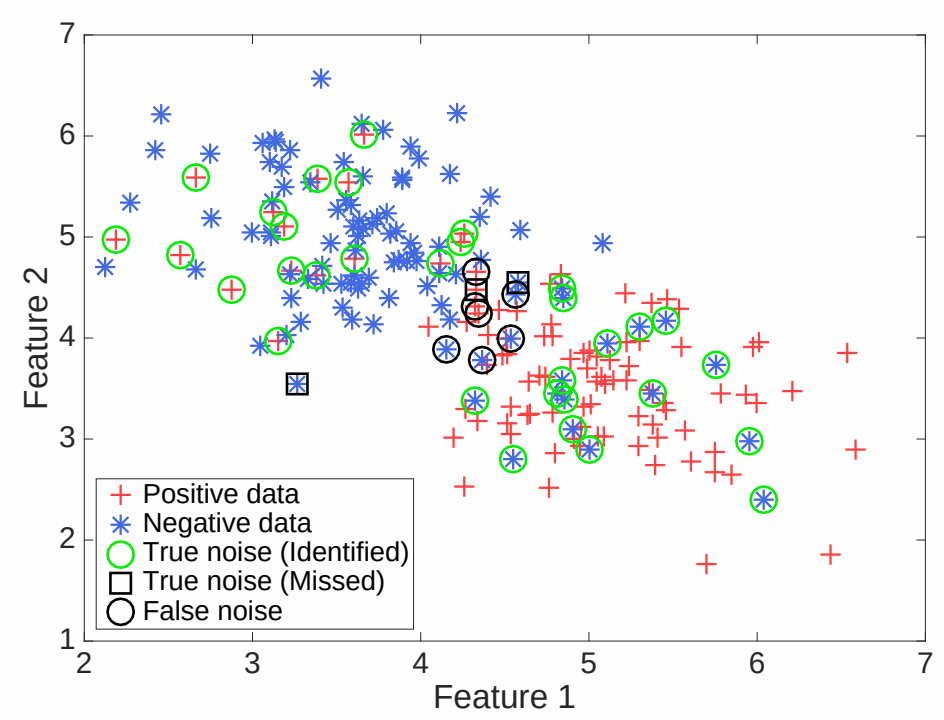


Figure 4.7: The performance of MCS-LR on the synthetic data set SD1 with 20% label noise.

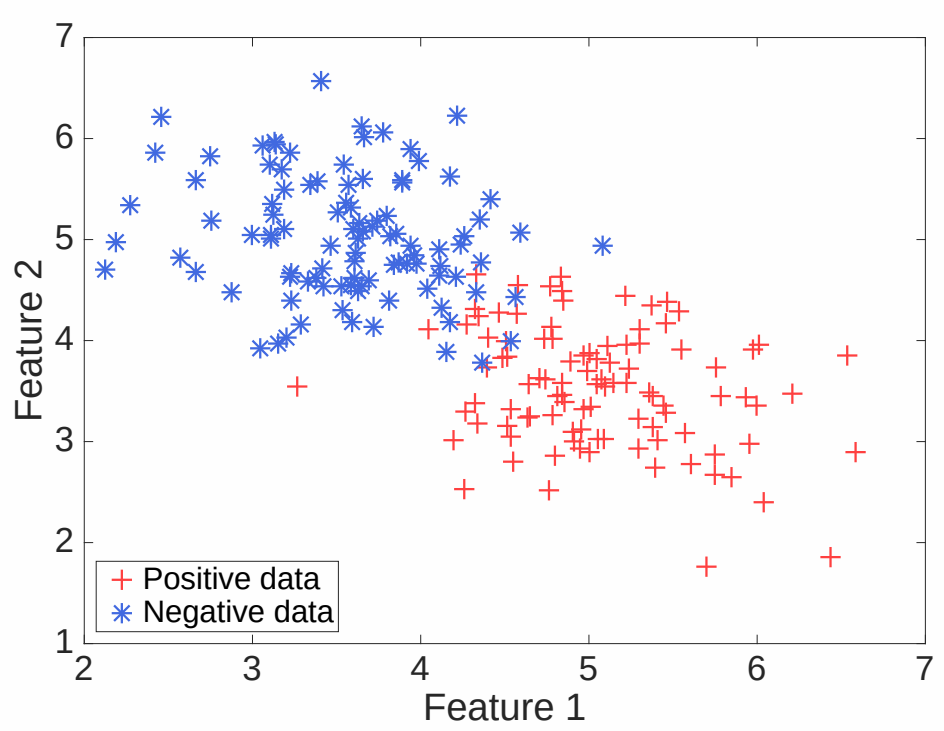


Figure 4.8: The clean synthetic data set SD2 without label noise.

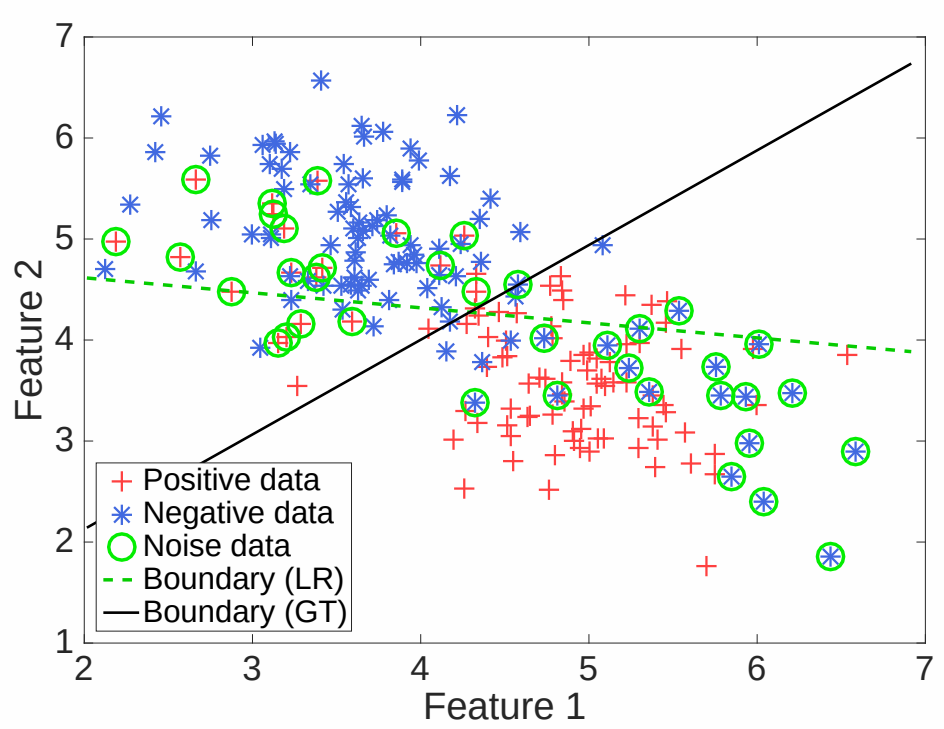


Figure 4.9: The performance of LR on the synthetic data set SD2 with 20% label noise.

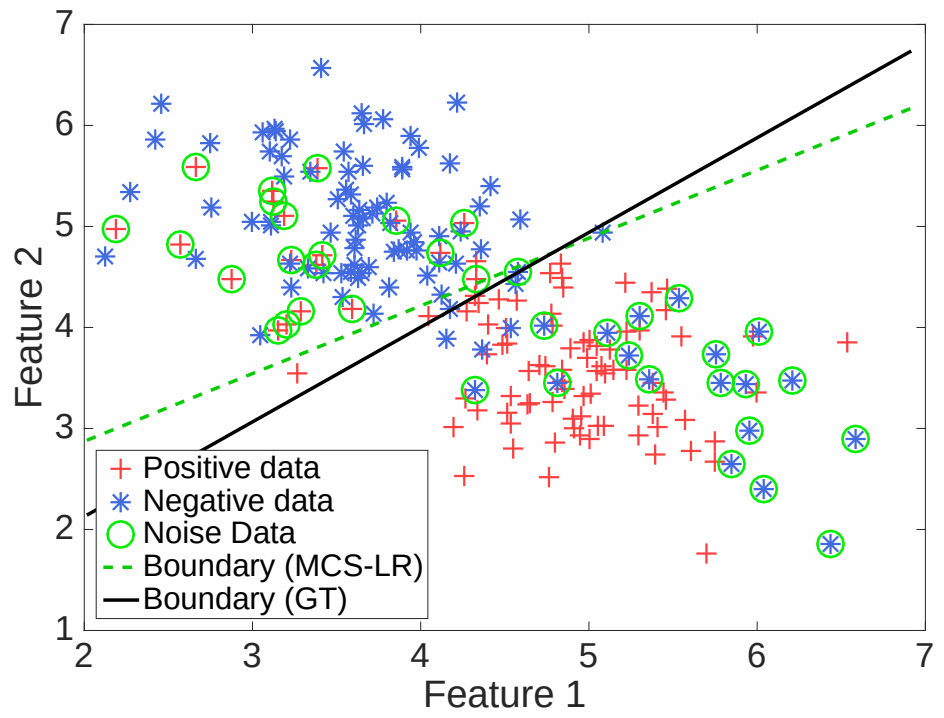


Figure 4.10: The performance of MCS-LR on the synthetic data set SD2 with 20% label noise.

the negative effects of the corrupted instances can be decreased. However, the standard Logistic Regression algorithm cannot differentiate the corrupted instances from the clean instances and takes all instances as training data.

Noise rate $(\rho^+,\ \rho^-)$	LR	MCS-LR
(0.3, 0)	89.60	**95.89**
(0.3, 0.1)	91.55	**95.36**
(0.4, 0.1)	80.44	**89.83**

Table 4.1: Means of Classification Accuracies (Percentage) of LR and MCS-LR on the Synthetic Data Set SD3.

4.5 Results on Real Data

4.5.1 Results of Accuracy

We compare the classification accuracy of our proposed MCS framework and the state-of-the-art methods [24] on the UCI Benchmark data sets. Since the methods in [24] are based on the KSVM algorithm, we plug KSVM in the MCS framework and get MCS-KSVM algorithms. As mentioned in Section 3.2.3, MCS-KSVM algorithms include MCS-KSVMD, MCS-KSVMF and MCS-KSVMW. Table 4.2 shows the experiment results, with the highest results in bold in each row.

In general, MCS-KSVM algorithms outperform other baselines in most data sets. Moreover, MCS-KSVM algorithms perform better than baselines in 6 different noise rate settings, including light noise rate like $\rho^+ = 0.1,\ \rho^- = 0.1$ and heavy noise rate like $\rho^+ = 0.4,\ \rho^- = 0.4$. MCS-KSVMD, MCS-KSVMF and MCS-KSVMW get similar results in most cases.

The data sets in Table 4.2 are sorted in the ascending order of the number of instances. Thus, we conclude that MCS-KSVM performs better on relatively large data sets. As mentioned in Theorem 3.2, the MCS framework satisfies that $\sum_{\mathbf{c}_i \in C^+} \boldsymbol{\pi}_i \gg \sum_{\mathbf{c}_j \in C^-} \boldsymbol{\pi}_j$. Therefore, one possible reason is that the difference between $\sum_{\mathbf{c}_i \in C^+} \boldsymbol{\pi}_i$ and $\sum_{\mathbf{c}_j \in C^-} \boldsymbol{\pi}_j$ on a large data set is larger than that on a small data set. In other words, the transition matrix P of a large data set is more concentrated on good states, which means the transition probabilities among good states in the Markov chain on a large data set are larger than that on a small data set.

We also show the comparison of accuracies between KSVM and MCS-KSVMF as the noise rate increasing on the UCI Benchmark data sets. The comparison results are shown in Figure 4.11. The noise rate ρ^+ and ρ^- are equivalent in Figure 4.11. We can observe that MCS-KSVMF generates higher accuracies than KSVM on all the data sets. More-

over, accuracies of MCS-KSVMF decrease slower than accuracies of KSVM as noise rate increasing. Thus, we show that MCS is more robust against label noise than KSVM.

Data set n	Noise $(\rho^+,\ \rho^-)$	KSVM	C-SVCF	eIWℓ	IWℓ	MCS -KSVMD	MCS -KSVMF	MCS -KSVMW
	(0.1, 0.1)	94.81±2.87	93.89±3.03	94.26±3.08	95.00±3.15	94.81±3.36	95.19±2.79	**95.19±2.65**
	(0.2, 0.2)	91.11±6.04	90.19±6.76	90.00±6.54	**91.48±6.43**	89.07±7.17	89.07±7.22	90.00±6.88
Thyroid	(0.3, 0.1)	87.22±5.56	86.48±6.30	85.56±6.46	87.41±5.71	**89.26±3.88**	88.33±4.79	87.96±4.21
215	(0.1, 0.3)	91.30±3.71	91.85±3.92	91.30±4.94	91.85±4.11	94.81±2.87	**94.81±2.73**	94.44±3.49
	(0.3, 0.3)	87.40±4.17	88.52±4.60	86.48±3.60	85.93±4.88	**92.41±3.32**	90.37±3.47	89.63±4.55
	(0.4, 0.4)	80.00±6.52	82.59±6.31	79.26±11.96	80.37±11.39	85.00±7.53	85.37±6.79	**85.93±6.83**
	(0.1, 0.1)	71.36±4.60	71.21±4.34	71.82±6.78	**75.60±4.76**	72.27±5.30	72.12±5.72	72.58±5.69
	(0.2, 0.2)	63.63±5.85	64.09±5.98	64.24±6.15	**70.76±6.63**	68.64±7.76	68.79±8.02	68.48±7.45
Breast	(0.3, 0.1)	67.12±4.04	67.42±3.59	69.39±4.39	**70.15±3.98**	69.55±3.67	69.55±3.60	69.39±3.56
cancer	(0.1, 0.3)	58.18±5.98	58.79±5.38	**64.09±4.41**	64.09±6.02	61.36±8.00	59.70±7.08	57.42±6.56
263	(0.3, 0.3)	59.55±5.20	60.91±4.09	63.18±6.19	**67.58±5.72**	63.18±5.58	63.64±6.74	61.52±5.63
	(0.4, 0.4)	57.12±7.49	58.18±8.34	59.70±7.15	57.58±12.10	59.70±7.77	**60.76±8.43**	60.45±8.70
	(0.1, 0.1)	72.50±4.39	73.68±3.89	73.68±3.83	73.38±3.57	76.47±2.86	**77.35±2.88**	76.32±3.43
	(0.2, 0.2)	69.26±6.22	68.53±5.68	71.03±5.50	69.26±5.21	**72.94±5.24**	72.50±4.90	71.62±6.36
Heart	(0.3, 0.1)	70.00±7.34	68.82±7.78	70.15±7.53	68.68±8.38	**72.79±9.33**	71.03±7.53	71.47±7.24
270	(0.1, 0.3)	69.41±5.08	70.29±3.45	70.00±4.56	69.26±4.30	72.06±4.60	**74.85±5.52**	72.50±5.84
	(0.3, 0.3)	64.71±7.14	64.71±6.20	65.44±5.69	65.88±6.96	67.35±6.23	68.82±7.68	**68.82±7.30**
	(0.4, 0.4)	56.62±10.72	55.88±9.38	59.71±11.33	58.68±11.59	**60.00±10.21**	59.41±8.80	58.97±11.48
	(0.1, 0.1)	72.55±5.35	72.24±4.52	71.67±4.99	71.56±4.87	**73.96±5.19**	73.70±4.79	73.59±5.87
	(0.2, 0.2)	70.89±2.74	71.72±2.69	72.08±2.52	70.68±3.19	73.54±1.53	**74.17±2.17**	73.70±1.52
Diabetes	(0.3, 0.1)	69.27±1.89	69.79±2.24	70.16±1.70	68.75±2.34	69.69±2.13	**70.57±2.51**	69.79±2.08
768	(0.1, 0.3)	66.93±2.45	67.92±3.40	**71.20±1.92**	69.53±3.08	69.84±2.65	69.22±3.35	70.21±2.93
	(0.3, 0.3)	68.70±3.66	70.10±3.61	70.52±3.17	69.95±2.80	71.93±3.78	71.04±3.26	**71.98±2.86**
	(0.4, 0.4)	58.18±3.27	59.11±3.46	60.83±3.71	**67.08±3.76**	61.98±3.18	61.78±3.21	61.93±3.39
	(0.1, 0.1)	72.32±2.97	71.84±2.64	72.44±2.85	72.72±3.36	74.20±4.09	**74.28±3.79**	73.00±3.67
	(0.2, 0.2)	64.84±3.52	65.96±3.62	65.08±2.98	66.68±3.68	**68.80±3.30**	68.12±2.85	67.76±3.38
German	(0.3, 0.1)	69.80±2.46	70.64±2.57	69.84±2.97	71.28±2.53	72.40±2.37	**72.72±3.39**	72.44±2.69
1,000	(0.1, 0.3)	61.12±2.21	62.64±2.30	61.44±1.81	62.92±1.52	66.52±4.15	**66.60±3.35**	65.96±3.43
	(0.3, 0.3)	61.60±3.27	62.24±3.83	62.04±3.19	63.36±4.74	65.20±4.17	**65.60±4.76**	63.52±3.32
	(0.4, 0.4)	55.32±2.57	55.76±3.30	56.24±2.45	56.76±4.40	**57.12±2.97**	56.60±3.97	56.36±3.01
	(0.1, 0.1)	86.50±1.35	86.39±1.46	86.63±1.25	86.60±1.43	87.47±1.06	**87.81±1.12**	86.95±1.20
	(0.2, 0.2)	80.43±1.43	80.90±1.29	80.40±1.53	80.53±1.65	**82.41±1.22**	82.39±1.44	81.35±1.65
Splice	(0.3, 0.1)	77.99±1.37	77.75±1.45	78.24±1.36	78.25±1.54	78.57±1.56	**78.66±1.86**	78.07±1.80
2,991	(0.1, 0.3)	77.62±1.17	77.53±1.17	77.78±1.18	77.57±1.35	**79.01±1.30**	78.57±0.98	78.05±1.23
	(0.3, 0.3)	71.97±2.09	72.22±1.93	72.06±2.10	72.21±2.29	74.33±2.10	**74.39±2.39**	73.16±2.32
	(0.4, 0.4)	62.33±1.85	62.41±1.56	62.35±1.79	61.89±2.30	63.69±2.12	**63.96±2.10**	62.91±1.60
	(0.1, 0.1)	87.94±1.16	88.34±0.44	87.94±1.11	88.30±0.85	89.75±0.84	**89.98±0.40**	89.77±0.70
	(0.2, 0.2)	84.18±1.41	84.75±1.21	84.26±1.51	86.50±0.97	88.18±1.28	**88.44±1.16**	87.84±1.23
Waveform	(0.3, 0.1)	84.48±1.29	84.82±1.29	85.78±1.38	85.09±1.51	**87.50±1.33**	87.33±1.68	86.66±1.66
5,000	(0.1, 0.3)	79.98±1.28	81.18±1.45	81.23±1.50	83.38±2.04	84.76±0.67	**85.33±0.73**	84.77±0.74
	(0.3, 0.3)	78.04±1.77	79.14±1.71	79.06±1.66	81.10±1.36	**84.32±1.26**	84.17±1.29	83.70±1.46
	(0.4, 0.4)	65.37±2.26	66.75±1.98	69.16±3.95	70.95±2.80	73.08±2.11	**73.46±2.06**	72.03±2.29
	(0.1, 0.1)	90.39±0.77	90.42±0.81	89.02±2.36	90.27±0.85	**90.44±0.70**	90.39±0.74	90.33±0.65
	(0.2, 0.2)	89.86±0.66	89.92±0.87	88.69±1.07	89.70±0.76	90.01±0.74	90.03±0.73	**90.04±0.69**
Banana	(0.3, 0.1)	88.16±0.52	88.42±0.58	87.22±1.13	88.76±0.91	89.07±0.53	89.04±0.60	**89.10±0.60**
5,300	(0.1, 0.3)	88.47±0.76	88.61±0.78	87.25±1.57	88.62±0.78	**88.66±0.95**	88.57±0.86	88.35±0.66
	(0.3, 0.3)	89.53±0.81	89.73±0.77	84.22±1.96	89.55±0.82	**89.97±0.74**	89.77±0.78	89.89±0.86
	(0.4, 0.4)	87.58±1.77	87.80±1.54	72.26±11.47	87.42±1.72	**88.23±1.52**	88.18±1.63	88.23±1.78
	(0.1, 0.1)	94.54±0.56	95.91±0.36	94.76±0.58	94.81±0.53	97.54±0.38	**97.62±0.29**	97.04±0.25
	(0.2, 0.2)	89.64±0.90	91.50±0.94	89.98±1.07	90.03±1.06	**94.70±0.75**	94.67±0.90	93.17±0.87
Ringnorm	(0.3, 0.1)	85.81±0.92	87.43±0.52	85.68±0.82	85.62±0.90	**90.01±0.69**	89.91±0.77	88.49±0.86
7,400	(0.1, 0.3)	91.39±0.64	94.03±0.76	93.63±0.60	93.55±0.72	**96.86±0.57**	96.69±0.47	96.13±0.52
	(0.3, 0.3)	84.01±1.29	86.01±1.15	85.00±1.32	85.17±1.36	90.03±1.19	**90.32±1.23**	88.01±1.22
	(0.4, 0.4)	73.40±1.89	76.75±1.60	74.65±1.67	77.86±1.58	**82.23±1.61**	82.03±1.78	80.26±1.56
	(0.1, 0.1)	93.26±0.44	93.28±0.50	93.38±0.49	94.25±0.47	95.90±0.42	**96.05±0.38**	95.56±0.36
	(0.2, 0.2)	87.60±0.92	87.90±0.81	88.42±0.90	89.16±0.94	92.74±0.75	**92.98±0.61**	92.11±0.71
Twonorm	(0.3, 0.1)	85.43±1.23	86.11±0.98	86.68±1.21	87.57±1.41	**90.74±0.85**	90.74±1.18	89.94±0.77
7,400	(0.1, 0.3)	85.86±1.07	86.67±1.17	87.34±1.05	88.05±1.46	**91.07±1.02**	90.96±1.02	90.43±1.08
	(0.3, 0.3)	78.66±1.26	80.28±1.04	80.94±1.08	80.69±1.39	86.41±0.94	**86.49±0.91**	85.16±0.95
	(0.4, 0.4)	65.43±2.09	67.22±2.04	70.10±1.69	69.94±1.74	74.04±1.79	**74.49±2.00**	73.04±1.72

Table 4.2: Means and Standard Deviations of Classification Accuracies (Percentage) of KSVM, C-SVCF, IWℓ, eIWℓ, MCS-KSVMD, MCS-KSVMF and MCS-KSVMW on the UCI Benchmark Data Sets.

4.5.2 Results of Error Rates

We also compare two types of errors, ER_1 and ER_2, in data filtering methods. The C-SVCF algorithm and the MCS-KSVMD algorithm belong to the data filtering category. The results of ER_1, ER_2 and ER_1+ER_2 of C-SVCF and MCS-KSVMD are shown in Table 4.3, with the lowest ER_1, ER_2 and ER_1+ER_2 in bold in each row. We can observe that ER_1 of C-SVCF are lower than that of MCS-KSVMD in most noise rate settings and most data sets. On the contrary, ER_2 of MCS-KSVMD are lower than that of C-SVCF in most noise rate settings and most data sets.

Since both ER_1 and ER_2 have an effect on the performance of algorithms, we compare the sum of ER_1 and ER_2 between C-SVCF and MCS-KSVMD. The sum of ER_1 and $ER2$ of MCS-KSVMD are lower than that of C-SVCF in almost all the cases. In other words, MCS-KSVMD makes fewer mistakes in filtering out noisy instances, thus resulting in better classification performance. Thus, we experimentally show the effectiveness of the MCS framework in identifying wrongly labeled instances.

Data set (n)	Noise (ρ^+, ρ^-)	ER_1		ER_2		ER_1+ER_2	
		C-SVCF	MCS-KSVMD	C-SVCF	MCS-KSVMD	C-SVCF	MCS-KSVMD
	(0.1, 0.1)	0.02±0.01	0.02±0.01	0.20±0.07	**0.11±0.08**	0.22	**0.13**
	(0.2, 0.2)	0.02±0.01	0.02±0.01	0.28±0.06	**0.17±0.08**	0.30	**0.19**
Thyroid	(0.3, 0.1)	**0.03±0.01**	0.03±0.02	0.55±0.06	**0.47±0.08**	0.58	**0.50**
215	(0.1, 0.3)	0.02±0.01	0.02±0.01	0.21±0.06	**0.14±0.05**	0.23	**0.16**
	(0.3, 0.3)	**0.04±0.01**	0.04±0.02	0.31±0.03	**0.20±0.10**	0.35	**0.24**
	(0.4, 0.4)	**0.11±0.10**	0.11±0.11	0.45±0.14	**0.38±0.19**	0.56	**0.49**
	(0.1, 0.1)	**0.08±0.01**	0.12±0.02	0.71±0.06	**0.60±0.13**	0.79	**0.72**
	(0.2, 0.2)	**0.09±0.02**	0.12±0.01	0.77±0.07	**0.66±0.10**	0.86	**0.78**
Breast cancer	(0.3, 0.1)	**0.07±0.01**	0.11±0.02	0.78±0.06	**0.66±0.08**	0.85	**0.77**
263	(0.1, 0.3)	**0.10±0.02**	0.14±0.04	0.80±0.07	**0.65±0.08**	0.90	**0.79**
	(0.3, 0.3)	**0.09±0.01**	0.12±0.03	0.77±0.05	**0.67±0.06**	0.86	**0.79**
	(0.4, 0.4)	**0.11±0.03**	0.16±0.04	0.83±0.04	**0.76±0.05**	0.94	**0.92**
	(0.1, 0.1)	**0.01±0.01**	0.04±0.01	0.83±0.12	**0.61±0.14**	0.84	**0.65**
	(0.2, 0.2)	**0.01±0.01**	0.04±0.01	0.89±0.06	**0.71±0.07**	0.90	**0.75**
Heart	(0.3, 0.1)	**0.01±0.01**	0.04±0.02	0.85±0.08	**0.67±0.13**	0.86	**0.71**
270	(0.1, 0.3)	**0.02±0.01**	0.04±0.02	0.86±0.06	**0.69±0.08**	0.88	**0.73**
	(0.3, 0.3)	**0.02±0.01**	0.05±0.02	0.91±0.03	**0.77±0.05**	0.93	**0.82**
	(0.4, 0.4)	**0.02±0.02**	0.05±0.02	0.92±0.04	**0.80±0.04**	0.94	**0.85**
	(0.1, 0.1)	**0.11±0.01**	0.13±0.01	0.48±0.04	**0.41±0.07**	0.59	**0.54**
	(0.2, 0.2)	**0.11±0.02**	0.14±0.01	0.57±0.05	**0.47±0.06**	0.68	**0.61**
Diabetes	(0.3, 0.1)	**0.11±0.01**	0.14±0.01	0.69±0.06	**0.64±0.06**	0.80	**0.78**
768	(0.1, 0.3)	**0.13±0.02**	0.15±0.02	0.56±0.05	**0.46±0.07**	0.69	**0.61**
	(0.3, 0.3)	**0.11±0.02**	0.14±0.02	0.60±0.03	**0.51±0.05**	0.71	**0.65**
	(0.4, 0.4)	**0.15±0.02**	0.17±0.03	0.70±0.06	**0.60±0.05**	0.85	**0.77**
	(0.1, 0.1)	**0.03±0.01**	0.07±0.01	0.84±0.05	**0.64±0.08**	0.87	**0.71**
	(0.2, 0.2)	**0.03±0.01**	0.07±0.01	0.86±0.02	**0.69±0.03**	0.89	**0.76**
German	(0.3, 0.1)	**0.03±0.01**	0.08±0.01	0.88±0.03	**0.74±0.03**	0.91	**0.82**
1000	(0.1, 0.3)	**0.04±0.01**	0.09±0.01	0.88±0.02	**0.73±0.04**	0.92	**0.82**
	(0.3, 0.3)	**0.04±0.01**	0.08±0.01	0.89±0.02	**0.76±0.03**	0.93	**0.84**
	(0.4, 0.4)	**0.04±0.01**	0.09±0.01	0.92±0.02	**0.81±0.03**	0.96	**0.90**
	(0.1, 0.1)	0.00±0.00	0.00±0.00	0.98±0.01	**0.76±0.03**	0.98	**0.76**
	(0.2, 0.2)	0.00±0.00	0.00±0.00	0.98±0.01	**0.86±0.02**	0.98	**0.86**
Splice	(0.3, 0.1)	**0.00±0.00**	0.01±0.00	0.98±0.01	**0.86±0.02**	0.98	**0.87**
2991	(0.1, 0.3)	0.00±0.00	0.00±0.00	0.99±0.00	**0.91±0.02**	0.99	**0.91**
	(0.3, 0.3)	**0.00±0.00**	0.01±0.00	0.99±0.01	**0.92±0.01**	0.99	**0.93**
	(0.4, 0.4)	**0.00±0.00**	0.01±0.00	0.99±0.00	**0.96±0.01**	0.99	**0.97**
	(0.1, 0.1)	**0.02±0.00**	0.04±0.00	0.54±0.03	**0.26±0.03**	0.56	**0.30**
	(0.2, 0.2)	**0.02±0.00**	0.04±0.00	0.61±0.03	**0.34±0.03**	0.63	**0.38**
Waveform	(0.3, 0.1)	**0.02±0.00**	0.04±0.00	0.66±0.02	**0.41±0.02**	0.68	**0.45**

5000	(0.1, 0.3)	**0.03±0.00**	0.05±0.00	0.68±0.01	**0.44±0.03**	0.71	**0.49**
	(0.3, 0.3)	**0.03±0.00**	0.05±0.00	0.69±0.01	**0.45±0.03**	0.72	**0.50**
	(0.4, 0.4)	**0.06±0.01**	0.08±0.01	0.78±0.01	**0.59±0.02**	0.84	**0.67**
	(0.1, 0.1)	0.10±0.00	**0.09±0.00**	**0.10±0.02**	0.11±0.02	0.20	0.20
	(0.2, 0.2)	0.10±0.00	**0.09±0.00**	0.10±0.01	0.10±0.01	0.20	**0.19**
Banana	(0.3, 0.1)	0.10±0.00	**0.09±0.00**	0.17±0.01	**0.16±0.02**	0.27	**0.25**
5300	(0.1, 0.3)	0.11±0.00	0.11±0.00	**0.13±0.02**	0.14±0.03	**0.24**	0.25
	(0.3, 0.3)	0.10±0.01	**0.10±0.00**	0.11±0.01	**0.11±0.00**	0.21	0.21
	(0.4, 0.4)	0.11±0.01	**0.10±0.01**	0.13±0.01	**0.12±0.01**	0.24	**0.22**
	(0.1, 0.1)	0.00±0.00	0.00±0.00	0.55±0.01	**0.17±0.02**	0.55	**0.17**
	(0.2, 0.2)	**0.00±0.00**	0.01±0.00	0.56±0.02	**0.34±0.02**	0.56	**0.35**
Ringnorm	(0.3, 0.1)	**0.00±0.00**	0.01±0.00	0.76±0.01	**0.60±0.02**	0.76	**0.61**
7400	(0.1, 0.3)	0.00±0.00	0.00±0.00	0.42±0.02	**0.15±0.01**	0.42	**0.15**
	(0.3, 0.3)	0.01±0.00	0.01±0.00	0.58±0.01	**0.41±0.01**	0.59	**0.42**
	(0.4, 0.4)	0.01±0.00	0.01±0.00	0.65±0.01	**0.49±0.01**	0.66	**0.50**
	(0.1, 0.1)	**0.00±0.00**	0.01±0.00	0.66±0.02	**0.25±0.02**	0.66	**0.26**
	(0.2, 0.2)	**0.00±0.00**	0.01±0.00	0.71±0.02	**0.39±0.02**	0.71	**0.40**
Twonorm	(0.3, 0.1)	**0.00±0.00**	0.01±0.00	0.75±0.01	**0.46±0.02**	0.75	**0.47**
7400	(0.1, 0.3)	**0.01±0.00**	0.01±0.00	0.76±0.02	**0.46±0.02**	0.77	**0.47**
	(0.3, 0.3)	**0.01±0.00**	0.01±0.00	0.77±0.02	**0.51±0.01**	0.78	**0.52**
	(0.4, 0.4)	**0.02±0.00**	0.04±0.00	0.85±0.01	**0.64±0.01**	0.87	**0.68**

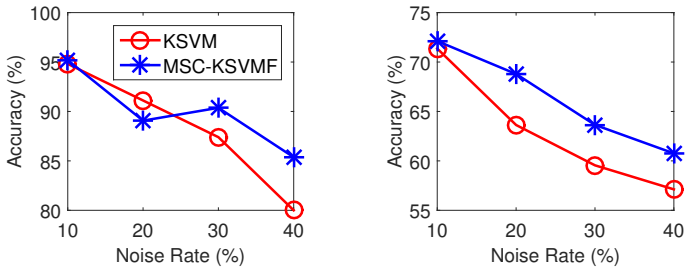
Table 4.3: ER_1, ER_2 and ER_1+ER_2 of C-SVCF and MCS-KSVMD on the UCI Benchmark Data Sets.

In conclusion, the MCS method outperforms the state-of-the-art methods on both synthetic data sets and real data sets. MCS is effective in identifying wrongly labeled instances and alleviating the negative effects of heavy label noise.

4.6 Scalability Test

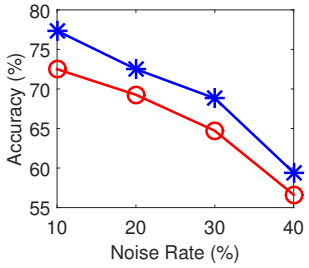
As stated in Chapter 3, the time complexity of the MCS framework is only $\mathcal{O}(NT)$. Denote by t_{KSVM} and $t_{MCS-KSVM}$ the running time of KSVM and MCS-KSVM, respectively. We conduct experiments on the UCI Benchmark data set. We compare $N \times t_{KSVM}$ and $t_{MCS-KSVM}$ by setting N to different values between 1 and 100.

The experiment results are demonstrated in Figure 4.12, where each running time is the total time of running experiments on 11 Benchmark data sets in six different noise rate settings. As shown in Figure 4.12, as N increases, $t_{MCS-KSVM}$ increases slower than $N \times t_{KSVM}$. The reason is that the training data in each state is a sampled subset of the original training set. Thus, the running time of MCS in each state is less than t_{KSVM}, and the total running time of MCS is less than $N \times t_{KSVM}$. Thus, we experimentally show the scalability of the proposed method.

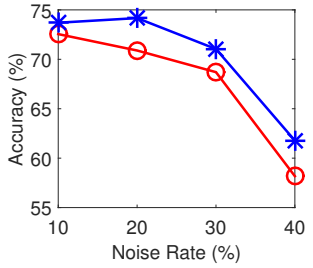


(a) Thyroid

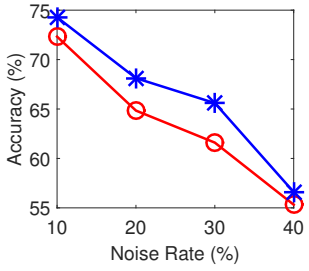
(b) Breast cancer



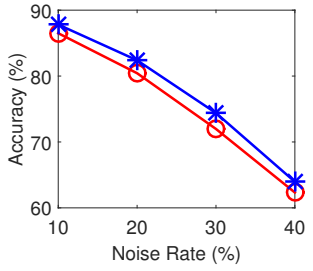
(c) Heart



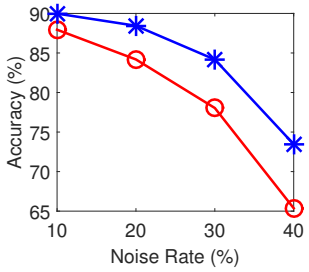
(d) Diabetes



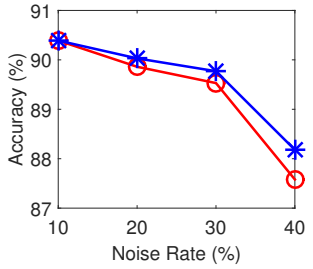
(e) German



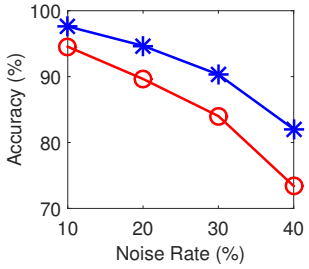
(f) Splice



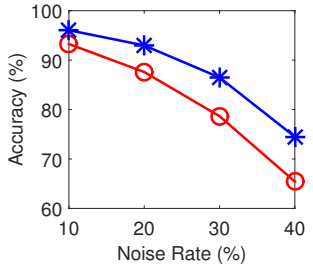
(g) Waveform



(h) Banana



(i) Ringnorm



(j) Twonorm

Figure 4.11: Classification accuracies (percentage) of KSVM and MCS-KSVMF on the UCI Benchmarks data sets as noise rate increasing.

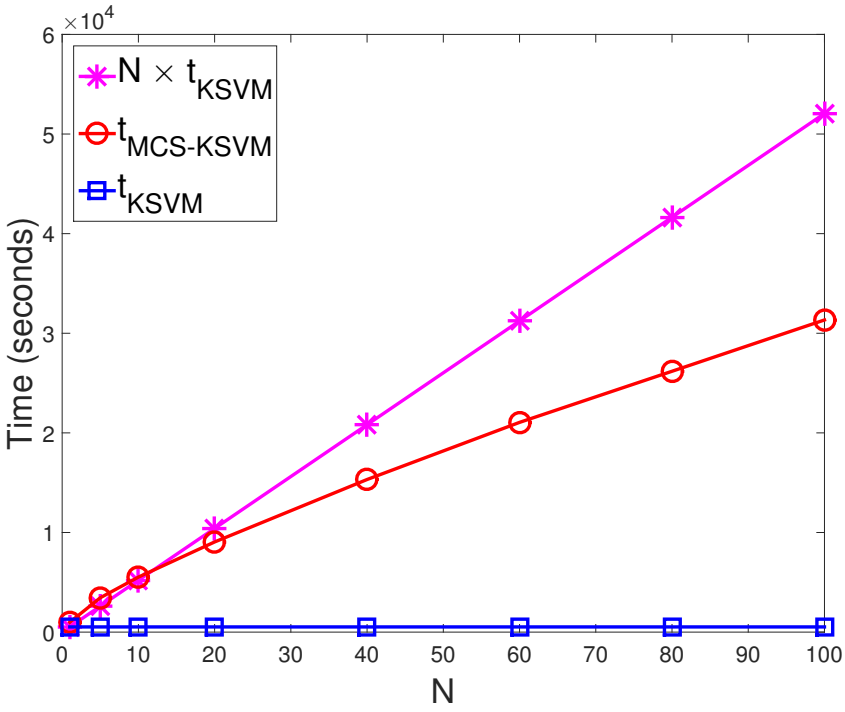


Figure 4.12: The running time of KSVM and MCS-KSVM.

Chapter 5

Conclusions

Classification with heavy label noise is important in practice, since real-world data sets are easily corrupted and thus contain noise [10]. In this thesis, we formulate the problem of classification with heavy label noise. To tackle this problem, we propose a novel framework, MCS, where a wide spectrum of classification algorithms can be plugged in. We use Logistic Regression and Support Vector Machine to demonstrate the effectiveness and generalizability of MCS in this thesis. In the MCS framework, we identify the wrongly labeled instances using ensemble outputs of a sequence of classifiers. The sequence of classifiers is trained on different subsets of the training data by sampling the states of a carefully designed Markov chain with random walk. Each state of the Markov chain uniquely relates to one of the subsets of the training data. The MCS framework largely reduces the number of states to sample in the Markov chain and is efficient in time complexity. The algorithms derived from the MCS framework are effective and efficient. We theoretically prove the correctness and effectiveness of our proposed framework. We also experimentally show that the derivative algorithms can achieve better performance than the corresponding standard algorithms as well as some cutting-edge algorithms. Furthermore, the MCS framework can handle some tough scenarios which the corresponding standard algorithms fail to tackle.

For future work, the following directions can be considered:

- *Plugging other classifiers in the MCS framework.* The MCS framework can entertain a wide spectrum of classifiers. A classifier can be plugged in the MCS framework as long as it provides different probabilities or scores for instances in classification results. In this thesis, we use Logistic Regression [18] and the kernelized Support Vector Machine [34] to demonstrate the effectiveness of MCS. A possible direction is to plug Naïve Bayes classifier in the MCS framework to further explore the effectiveness and generalizability of MCS.

- *Improving the classification performance on small data sets.* As shown in Section 4.5, the classification performance of MCS on small data sets is not as good as that on

large data sets. Moreover, the performance of MCS is not as good as that of baselines in some cases on small data sets. For future work, we can investigate the reason and try to improve the performance of MCS on small data sets.

- *Applying the major idea of MCS to the problem of classification with feature noise.* In this thesis, we only deal with the problem of classification with heavy label noise. Can the major idea of MCS be applied to the problem of classification with feature noise? This is an interesting topic to be studied in the future.

Bibliography

[1] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[2] Anelia Angelova, Yaser Abu-Mostafam, and Pietro Perona. Pruning training sets for learning of object categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 494–501. IEEE, 2005.

[3] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[4] VD Belur. Nearest neighbor (nn) norms: Nn pattern classification techniques. *IEEE Computer Society Press, New York: IEEE press*, 1991.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[8] Sarah Jane Delany and Pádraig Cunningham. An analysis of case-base editing in a spam filtering system. In *European Conference on Case-Based Reasoning*, pages 128–141. Springer, 2004.

[9] Andres Folleco, Taghi M Khoshgoftaar, Jason Van Hulse, and Lofton Bullard. Identifying learners robust to low quality data. In *Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on*, pages 190–195. IEEE, 2008.

[10] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[11] Dragan Gamberger and Nada Lavrač. Noise detection and elimination applied to noise handling in a krk chess endgame. In *International Conference on Inductive Logic Programming*, pages 72–88. Springer, 1996.

[12] Dragan Gamberger, Nada Lavrač, and Sašo Džeroski. Noise elimination in inductive concept learning: A case study in medical diagnosis. In *Algorithmic Learning Theory*, pages 199–212. Springer, 1996.

[13] Dragan Gamberger, Nada Lavrac, and Ciril Groselj. Experiments with noise filtering in a medical domain. In *ICML*, pages 143–151, 1999.

[14] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[15] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[16] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[17] Isabelle Guyon, Nada Matic, Vladimir Vapnik, et al. Discovering informative patterns and data cleaning., 1996.

[18] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[19] https://archive.ics.uci.edu/ml/datasets/banknote+authentication.

[20] http://theoval.cmp.uea.ac.uk/matlab.

[21] Piyasak Jeatrakul, Kok Wai Wong, and Chun Che Fung. Data cleaning for classification using misclassification analysis. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 14(3):297–302, 2010.

[22] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[23] Taghi M Khoshgoftaar and Pierre Rebours. Generating multiple noise elimination filters with the ensemble-partitioning filter. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 369–375. IEEE, 2004.

[24] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

[25] Naresh Manwani and PS Sastry. Noise tolerance under risk minimization. *IEEE transactions on cybernetics*, 43(3):1146–1151, 2013.

[26] André LB Miranda, Luís Paulo F Garcia, André CPLF Carvalho, and Ana C Lorena. Use of classification algorithms in noise detection and elimination. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 417–424. Springer, 2009.

[27] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[28] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[29] Seishi Okamoto and Nobuhiro Yugami. An average-case analysis of the k-nearest neighbor classifier for noisy domains. In *IJCAI (1)*, pages 238–245, 1997.

[30] John C Platt. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208, 1999.

[31] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[32] José A Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, 38(1):179–206, 2014.

[33] José Salvador Sánchez, Filiberto Pla, and Francesc J Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18(6):507–513, 1997.

[34] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[35] Jiang-wen Sun, Feng-ying Zhao, Chong-jun Wang, and Shi-fu Chen. Identifying and correcting mislabeled training instances. In *Future generation communication and networking (FGCN 2007)*, volume 1, pages 244–250. IEEE, 2007.

[36] Jaree Thongkam, Guandong Xu, Yanchun Zhang, and Fuchun Huang. Support vector machine for outlier detection in breast cancer survivability prediction. In *Asia-Pacific Web Conference*, pages 99–109. Springer, 2008.

[37] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[38] D Randall Wilson and Tony R Martinez. Instance pruning techniques. In *ICML*, volume 97, pages 403–411, 1997.

[39] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

[40] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3):408–421, 1972.

[41] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, 2004.