mCRF and mRD: Two Classification Methods Based on a Novel Multiclass Label Noise Filtering Learning Framework

Article *in* IEEE Transactions on Neural Networks and Learning Systems · January 2021

DOI: 10.1109/TNNLS.2020.3047046

CITATIONS
4

READS
129

7 authors, including:

Shu-yin Xia
Chongqing University of Posts and Telecommunications
38 PUBLICATIONS **225** CITATIONS
SEE PROFILE

Guoyin Wang
Chongqing University of Posts and Telecommunications
83 PUBLICATIONS **1,731** CITATIONS
SEE PROFILE

Xinbo Gao
Xidian University
754 PUBLICATIONS **15,383** CITATIONS
SEE PROFILE

Elisabeth Giem
University of California, Riverside
6 PUBLICATIONS **42** CITATIONS
SEE PROFILE

mCRF and mRD: Two Classification Methods based on a Novel Multiclass Label Noise Filtering Learning Framework.

Shuyin Xia*, *Member, IEEE,* Baiyun Chen*, *Member, IEEE,* Guoyin Wang, Yong Zheng, Xinbo Gao, Elisabeth Giem, Zizhong Chen, *Senior Member, IEEE*

***Abstract*—Mitigating label noise is a crucial problem in classification. Noise filtering is an effect method of dealing with label noise which does not need to estimate the noise rate or rely on any loss function. However, most filtering methods focus mainly on binary classification, leaving the more difficult counterpart problem of multiclass classification relatively unexplored. To remedy this deficit, we present a definition for label noise in a multiclass setting and propose a general framework for a novel label noise filtering learning method for multiclass classification. Two examples of noise filtering methods for multiclass classification, multiclass complete random forest (mCRF) and multiclass relative density, are derived from their binary counterparts using our proposed framework. In addition, to optimize the *NI_threshold* hyperparameter in mCRF, we propose two new optimization methods: a new voting cross-validation method and an adaptive method which employs a 2-means clustering algorithm. Furthermore, we incorporate SMOTE into our label noise filtering learning framework to handle the ubiquitous problem of imbalanced data in multiclass classification. We report experiments on both synthetic data sets and UCI benchmarks to demonstrate our proposed methods are highly robust to label noise in comparison with state-of-the-art baselines. All code and data results are available at https://github.com/syxiaa/Multiclass-Label-Noise-Filtering-Learning.**

***Index Terms*—Label noise, multiclass classification, complete random forest, relative density.**

I. Introduction

CLASSIFICATION in the presence of label noise is a well-studied problem with a worldwide body of contributing literature. Label noise usually refers to labels that are corrupted; this corruption may be caused by insufficient tagging information, errors during expert tagging, subjectivity in the tagging process, and data coding or communication issues [1]. Label noise can deteriorate the performance of classifiers in many cases [2], [3], [4], [5]. The ubiquity of label noise in real world has prompted the development of various methods to eliminate this performance attrition.

A common approach for mitigating label noise is to remove the noisy instances in the training set by filtering or by relabeling the corrupted labels before the training process [6], [7], [8]. These methods make use of different characteristics to detect label noise by pre-defining an index and presetting a threshold to identify the noisy instances. For instance, in [9], the entropy of the conditional distribution $P(Y|X)$ is estimated using the probabilities obtained by the Bayesian classifier. Instances with low entropy and error prediction results are identified as mislabeled instances and then relabeled. In [10], a rank pruning algorithm is proposed to solve the $\hat{P}\hat{N}$ learning problem and the open problem of estimating noise rates. The accuracy of these types of methods greatly depends on the predicted probabilities of a specified classifier [11]. In [12], the relative density, defined as the ratio of the distances between the k-nearest homogeneous neighbors and the k-nearest heterogeneous neighbors, is estimated to identify noisy data. Instances with relative density greater than one are identified as mislabeled instances and are removed from the training set. However, the threshold value can be ineffective in an asymmetric data set, allowing Y. Liu $et\ al.$ to improve the relative density by performing an automatic search for an optimal threshold [13]. In [14], X. Liang $et\ al.$ improved algorithmic efficiency by calculating the distance ratio of each instance between homogeneous and heterogeneous centers, significantly reducing computational frequency. In [15], the novel completed random forest (CRF) method is proposed to detect class noise. The noise intensity (*NI*) in CRF is predefined to measure the stability of each instance; any instance with a label different from that of its closest ancestor node and which remains invariant during a backward trace of *NI_threshold* length is identified as a noisy sample. A distinct advantage of CRF is that the division process does not rely on any classifier or linear distribution, and is not influenced by different weightings of the features. In addition to these methods, some algorithms derived from the complexity of label noise, such as the Outlier Removal Boosting (ORBoost) method [16], the Classification-stability (CL-stability) algorithm, and the Leave-one-out-error-sensitivity (LOOE-sensitivity) algorithm based on the leave-one-out perturbed classification (LOOPC) matrix, were designed to detect label noise [17]. Ensemble classifiers also have excellent performance in classification tasks, and some ensemble-based noise detection methods have therefore been developed to solve noise-containing label classification [18], [19], [20], [21]. It is possible that editing

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61806030 and 61936001, the Natural Science Foundation of Chongqing under Grant Nos. cstc2019jcyj-msxmX0485 and cstc2019jcyj-cxttX0002, the National Key Research and Development Program of China under Grant Nos. 2019QY(Y)0301, the Doctor Training Program of the Chongqing University of Posts and Telecommunications (grant no. BYJS201901), and NICE: NRT for Integrated Computational Entomology, US NSF award 1631776.

S. Xia, B. Chen, Y. Zheng, and G. Wang are with the College of Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Telecommunications and Posts, 400065 Chongqing, China. E-mail: xiasy@cqupt.edu.cn, chenbaiyun.lovely@163.com; 413511280@qq.com; wanggy@cqupt.edu.cn; gaoxb@cqupt.edu.cn

E. Giem and Z. Chen are with Department of Computer Science and Engineering, University of California, Riverside. Email: gieme01@ucr.edu, zizhongchen@gmail.com.

*: co-corresponding authors.

the training set can reduce model complexity and alleviate overfitting engendered by label noise, but such editing may lead to over-cleansing, especially for unbalanced data sets [22], [23], [24], [25], [26].

Another approach is which obtains mixed results is convex potential loss functions in the presence of label noise. It has been demonstrated that any approach based on convex risk minimization is required to modify the loss function in order to obtain the same optimal classifier on a corrupted data set as it would on the corresponding clean set [27]. Based on this fact, a series of loss function modification algorithms have been developed. Natarajan, Dhillon, Ravikumar, and Tewari developed two approaches to modify the loss functions. The first approach employs an unbiased estimator of surrogate loss functions for empirical risk minimization, but it may be non-convex even when the original loss function is convex, except when the original loss satisfies a symmetric condition. The second approach constructs a cost-sensitive surrogate loss, which depends on a parameter that assumes *a priori* knowledge and is defined in terms of noise rates [28]. Because in most real cases noise rates cannot be known in advance, Scott proposed a method based on the mutually irreducible assumption to estimate the noise rate as a mixed proportion estimation (MPE) by calculating the minimized slope between the points of an induced receiver operating characteristic (ROC) curve and the point (1, 1). However, this method has the drawback that the convergence speed can be arbitrarily slow [29]. Yu, Liu, Gong, Batmanghelich, and Tao introduced an independence assumption to estimate the noise rate by using only a small number of examples from the components. This approach easily estimates the class priors by solving simple convex quadratic programming, and the convergence speed is guaranteed [30]. However, kernel parameter estimation may be difficult during the kernel mean embedding of distributions, which can greatly affect the eventual proportion estimation.

Another design approach researchers have investigated is developing naturally robust loss functions for learning with label noise. Ghosh, Manwani, and Sastry investigate how to create a risk minimization robust to label noise, and prove a sufficient condition on loss function in order to be tolerant to label noise [31]. This condition is expressed as $l(x,1)+l(x,-1)=C$, where $l(x,1)$ denotes the loss function of a sample x when it is predicted as 1. This implies that once the sum of a loss of an example to be classified as each class is equivalent to a constant C, the loss function l becomes noise-tolerant under uniform noise. This work was the foundational scientific theoretical basis for much of the following literature. To develop it further into symmetric label noise (SLN), Van Rooyen, Menon, and Williamson introduced a convex classification-calibrated loss [32]. This unhinged loss can be easily proven to be SLN-robust. T. Liu and Tao employed the abundant surrogate loss functions for importance reweighting in classification with label noise and consistently found that the label noise would not prevent the search for an optimal classifier for the noise-free samples [33]. This approach also provided a new method of obtaining the noise rate, and the upper bound of the noise rate was proven to be the conditional probability of a noise sample. Patrini, Nielsen, Nock, and Carioni proposed a simple loss factorization method to decompose a loss function into an even function and an odd function, and this makes it suitable for processing asymmetric label noise [34].

The approaches discussed above are primarily designed for binary classification; however, some algorithms have been developed for noise-containing multiple label classification. In [35], D. Hernández-Lobato, J. M. Hernández-Lobato, and Dupont introduced binary latent variables that indicate whether a given instance is considered to be an outlier (a wrongly-labeled instance) or not, and developed a robust multiclass Gaussian process classifier to alleviate the noisy label influence by neglecting the distances of wrongly-labeled samples to the decision boundaries. Based on a Gaussian assumption on class conditional distributions, Bootkrajang and Kabán developed a model-based approach that extends the multiclass quadratic normal discriminant analysis with a model of the mislabeling process [36]. Sukhbaatar *et al.* studied the performance of discriminatively-trained ConvNets trained with large scale noisy labels on ImageNet, introducing an extra noise layer into the network to adapt the network outputs to match noisy label distribution [37]. Decomposing strategies, which have proven to be successful in improving classification performance in multiclass classification problems, have also been used in noise filtering to simplify the noise identification process [38], [39], [40], [41]. A robust multiclass AdaBoost algorithm, Rob_MulAda, was developed in [42], where a noise-detection based multiclass loss function is formally designed and a new weight updating scheme is presented to mitigate the harmful effect of noisy examples. Learning on noisy data can closely reflect the results on noise-free data if the widely-used importance reweighting strategy is employed, which is applicable to any traditional surrogate loss function and many different multiclass classification settings [43]. In practical application, label noise is more common in multiclass classification, such as text classification [44], medical diagnosing [45], intrusion detection [46], and collaborative recommendation systems [47]. However, the above methods are all designed based on a specific algorithm for noise-containing multiclass label classification. Our work is focused on creating more generalized methods for processing noise-containing multiclass label classification problems, and the main contributions of this work can be summarized as follows:

1) We propose a general framework for label noise filtering in learning methods for multiclass classification. We first give the definition of label noise in multiclass scenarios, and develop noise-filtering methods from binary to multiclass classification.
2) We develop the multiclass versions of two state-of-the-art label noise-filtering methods for binary classification: the Complete Random Forest Model (CRF) and the Relative Density Model (RD). We call our multiclass versions multiclass Complete Random Forest Model (mCRF) and multiclass Relative Density Model (mRD).
3) We propose two methods, a novel voting cross-validation (VCV) method and an adaptive method, to improve the optimization of the *NI_threshold* parameter for mCRF.

The voting mechanism allows the methods to achieve higher classification accuracy because it leads to more stability in detecting noisy samples.

4) Our label noise-filtering learning framework can be easily incorporated into the current prevalent oversampling method to handle class-imbalance problems with label noise in multiclass classification, and can be easily combined with different classifiers, including Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Adaboost, LightGBM, Xgboost, and others. Experimental results on both synthetic and real data sets demonstrate the effectiveness of our proposed methods.

The rest of the paper is organized as follows. In Section II, two typical filtering-based methods in binary classification are introduced. We present the formulation of multiclass classification with label noise in Section III. In Sections IV and V, two instantiations of the noise-filtering methods, that is, mCRF and mRD, in multiclass classification are discussed in detail, and two methods to optimize the *NI_threshold* parameter in mCRF are also detailed. Experimental results are shown in Section VI, with our conclusions drawn in Section VII.

II. Related Work

In this section we introduce two excellent label noise-filtering methods—the complete random forest method and the relative density method—which have been theoretically and empirically proven to be effective in filtering label noise in binary classification scenarios.

A. Complete Random Forest Filtering Method

It is well known that the complete random forest method, inspired by simulating grid generation and expansion, is the most effective label noise detection method in binary classification. CRF does not rely on any specific classifiers, distance measures, or global distribution measurements. This allows CRF to effectively detect label noise in a complex data environment, even for high-dimensional data sets corrupted by label noise [15]. We therefore select CRF as our first model to extend into multiclass classification.

CRF filtering in a decision tree consists mainly of two steps. First, a complete random forest F is generated. Let *Ntree* denote the number of trees generated in F. In F, each tree is a binary tree. Contrary to the generation of a conventional random forest, the split points in F are designed to be a random value of a randomly selected feature instead of calculating the Gini Coefficient for each feature, a construction detail that boosts algorithmic time efficiency significantly, especially for high-dimensional data. When F is constructed, the label of each node is marked with the label of the majority of samples in each node. The second step is to detect label noise. Before detecting noisy samples, the noise intensity (*NI*) is defined to measure the level of a sample surrounded by heterogeneous samples. When traversing upward from a leaf node to the root or to a node with the same label as that of the leaf node, the *NI* value of each sample in the leaf node is determined by counting the number of successively different labels between the leaf node and its predecessor nodes. Thus, the greater the value of *NI*, the more likely a point is to be recognized as a noisy label, and vice versa. When a complete random forest is constructed, we can easily determine the *NI* value of each sample in each binary tree. To obtain an optimized value of *NI_threshold*, noisy samples can simply be identified with a majority voting mechanism.

CRF does not rely on any probabilistic classifiers, distance measures, or distribution of the data. Compared with existing noise detection approaches, it does not suffer from changing feature weights to different values or any disadvantage of specific classifier, resulting in an improvement in generalizability to a complex data environment. However, there is a hyperparameter, *NI_threshold*, in the algorithm. In [15], an optimal *NI_threshold* is specified according to the highest classification accuracy, with *NI_threshold* ranging from 2 to 11 with an interval of 1. In this paper, we not only develop CRF from binary classification to multiclass classification, but also propose a novel votingcross-validation method as well as an adaptive method to optimize the *NI_threshold* parameter.

B. Relative Density Filtering Method

The relative density model is inspired by the idea that samples surrounded by more heterogeneous points than homogeneous points are more likely to be noisy [2]. The relative density is based on the absolute density, and is calculated by the ratio of distances between the k-nearest heterogeneous neighbors and k-nearest homogeneous neighbors of each sample. RD performs very well on uniformly distributed data sets by utilizing the contrast characteristics. We review the following three classic definitions, which we will need later.

Definition 1: Absolute density.

Let $D \in R^d$ be a data set. For any two samples $p \in D$ and $q \in D$, $d(p,q)$ represents the distance between points p and q. The k nearest neighbors of p are given by $N_k(p)$, where $N_k(p) = \{q \in D \mid d(p,q) < k{-}distance(p)\}$. Then, the *absolute density* of p is given by:

$$AbsoluteDensity(p) = \frac{k}{\sum d(p,q)}, \text{ for } q \in N_k(p). \quad (1)$$

Definition 2: Heterogeneous and homogeneous k-nearest neighbors.

Let $D \in R^d$ be a data set, where D consists of two sets of samples: $D+$ and $D-$. For a sample point $p \in D+$, the k-nearest *homogeneous neighbors* of k are defined as $HONk(p) = \{q|q \in D + |d(p,q) \leq k - distance(p)\}$, and k-nearest *heterogeneous neighbors* of p are defined as $HENk(p) = \{q|q \in D - |d(p,q) \leq k - distance(p)\}$.

Definition 3: Relative density.

Let $D \in R^d$ be a data set, where D consists of two sets of samples: $D+$ and $D-$. For a sample point $o \in D+$, its relative density is defined as:

$$\begin{aligned} Relative\ Density(o) &= \frac{Absolutedensity(o, D-)}{Absolutedensity(o, D+)} \\ &= \frac{k/\sum(o,q)(q \in HENk(o))}{k/\sum(o,p)(p \in HONk(o))}. \\ &= \frac{\sum(o,p)(p \in HONk(o))}{\sum(o,q)(q \in HENk(o))} \end{aligned} \quad (2)$$

For a point $o \in D$, if $Relative\ Density(o)$ is greater than 1, the point o will be identified as noise.

III. Multiclass Label Noise Filtering Learning

Following common convention, let data set $\{(X_i, Y_i)\}_{i=1}^{n}$ be independently and identically drawn from the underlying true distribution D, with $X_i \in R^d$ and $Y = \left[Y^1, Y^2, \ldots, Y^m\right]^T$. $\sum_{k=1}^{m} Y^k = 1$, where n denotes the number of training samples, m denotes the number of classes, and d denotes the number of dimensions. $Y^k = 1$ when the corresponding example X_i belongs to the k^{th} class; otherwise $Y^k = 0$. A proportion $\gamma \in (0, 0.5)$ of sample labels are randomly flipped, and a corrupted data set $D' = \left\{\left(X_i, \hat{Y}_i\right)\right\}_{i=1}^{n}$ is obtained, where $\hat{Y}_i$ represents the label of X_i in D'.

Let D_γ denote the distribution of the corrupted data set, and $f_k\left(X_i, w\right)$ a real-valued decision function with w representing the parameters in the function. $1\left(f_k\left(X_i, w\right), \hat{Y}_i^k\right)$ represents the surrogate loss function, where $\hat{Y}_i^k \in \{0, 1\}$. The objective of multiclass classification is to find an optimal function $f_k^*\left(X_i, w\right)$ to distinguish the samples into m classes by leveraging n training samples. To make this goal quantitative, the label noise-filtering objective function is defined as:

$$\arg\min_{v,w} R_{1,D_\gamma}(f) = \min_{v,w} \sum_{k=1}^{M} \left(\frac{1}{n} \sum_{i=1}^{n} v_i 1\left(f_k\left(X_i, w\right), \hat{Y}_i^k\right) \right), \tag{3}$$

where $\boldsymbol{v} = [v_1, v_2, \ldots, v_n]^{\mathrm{T}}$. $v_i = 0$ when a sample is identified as a noise; otherwise, $v_i = 1$.

To extend a learning method designed for binary classification (such as the label noise detection methods in this paper) into multiclass classification, a common scheme is to use a one-vs-rest [39] or a one-vs-one [48] mechanism. Here, $M = m$ when the one-vs-rest strategy is adopted, and $M = m(m-1)/2$ when the one-vs-one strategy is adopted. However, both of these strategies will generate many median classifiers and increase computational overhead. In addition, the one-vs-rest strategy is likely to lead to an imbalanced problem. To address these problems, we give a general definition of label noise in multiclass classification and provide a general framework for label noise filtering learning methods that can be directly applied to multiclass classification tasks.

Definition 4: Label noise in multiclass classification.

Let $D' = \left\{\left(X_i, \hat{Y}_i\right)\right\}_{i=1}^{n}$ be a corrupted data set, where D' consists of m classes of samples D'_1, D'_2, D'_3, ..., D'_m. $\hat{Y}_i$ is the observed label of the sample X_i, and Y_i is the underlying true label of X_i. For any sample $X_i \in D_k, k \in \{1, 2, \ldots, \mathrm{m}\}$, X_i is defined as a noisy sample when the observed label of X_i is different from the underlying true label, i.e., $\hat{Y}_i \neq Y_i$.

Definition 4 implies that, once the observed label of a sample is different from its true label, it will be regarded as a noisy sample, regardless of what class the observed label $\hat{Y}_i$ belongs to. Based on this definition, we expand the label noise detection measurement from binary classification into multiclass classification in Definition 5.

Definition 5: Multiclass extension of binary noise detection.

Let $D' = \left\{\left(X_i, \hat{Y}_i\right)\right\}_{i=1}^{n}$ be a corrupted data set, where D' consists of m classes of samples D'_1, D'_2, D'_3,..., D'_m. $\hat{Y}_i$ is the observed label of the sample X_i, and Y_i is the underlying true label of X_i. Let $LN\left(X_i\right) = f\left(X_i, \hat{Y}_i, Num\left(-\hat{Y}_i\right)\right)$ be a label noise detection measurement for binary classification, where $Num\left(-\hat{Y}_i\right)$ represents the number of opposite labels of X_i when judging whether X_i is label noise or not under the given mapping function f. The multiclass version of LN is $LN\left(X_i\right) = f\left(X_i, \hat{Y}_i, Num\left(\hat{Y}_i \neq Y_i\right)\right)$, where $Num(\hat{Y}_i \neq Y_i)$ represents the number of samples whose labels are contaminated, and f is used to judge whether a sample is label noise or not.

By simply replacing opposing labels with heterogeneous labels, Definition 5 presents a general framework which extends the label-noise filtering methods designed for binary classification into multiclass classification. Accordingly, v_i in (3) can be directly calculated instead of decomposing the multi-class problems into binary classification problems. Thus, we may avoid generating the median classifiers altogether.

IV. mCRF: A Multiclass Version of CRF Noise Filtering

In this section, based on Definition 5, we develop the Complete Random Forest filtering learning method from a binary classification method into a multiclass classification method.

A. Multiclass Complete Random Forest Filtering

We first give the definitions of noise intensity (*NI*) and a measurement that identifies whether a sample is label noise or not for multiclass classification.

Definition 6: Noise intensity.

Let $D \in R^d$ be a data set consisting of m classes of samples, represented by D_1, D_2, D_3, ..., D_m. A multiclass complete random tree M is constructed on D. The *noise intensity* value *NI* of each sample in a leaf node of M is defined as the number of different labels between the leaf node and its ancestors.

Note that during the tree traversal process used to compute the *NI*, the labels of the ancestors must always be different from the label of the leaf node. The labels of the traversed nodes can be either the same or different from each other, but must be different from the label of the leaf node. If the traversal process reaches either an ancestor node with the same label as the leaf node or the root node, the traversal is terminated. Figure 1 illustrates the process of constructing a complete random tree. There are three classes labeled Y_1, Y_2, and Y_3, which respectively correspond to blue, orange, and green nodes. Each node in the binary tree is marked by the majority label of the node. The root point is recursively split until it fulfills either of the following two conditions: the labels in a node are all the same, or the number of samples in a node is equal to 1. In Figure 1, we can see that the 20^{th}, 28^{th}, and 29^{th} points are more likely to be noisy samples as they are more likely to be surrounded by heterogeneous points. When traversing upward from a leaf node to the root or to a node with the same label, the *NI* value of each sample in the leaf node is

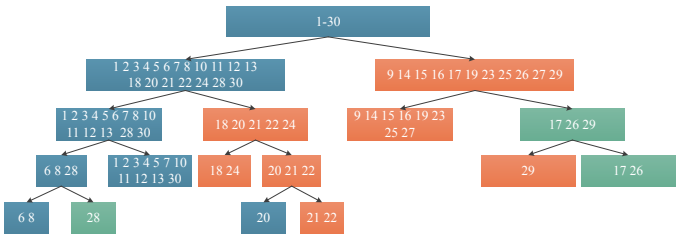


Fig. 1. Structure of the complete random tree.

determined by counting the number of successively different labels between the leaf node and its predecessor nodes. For instance, the *NI* value of the 28^{th} point is 3 as we traverse upward towards the root, the *NI* value of the 20^{th} point is 2, and the *NI* value of the 29^{th} point is 1 because we have traversed upwards to a node with the same label as that of the leaf node. The *NI* values of the remaining samples are zero as their labels are identical to the labels of their parent nodes.

Definition 7: Multiclass label noise filtering.

Let $D \in R^d$ be a data set consisting of m classes of samples, represented by D_1, D_2, D_3, ..., D_m. A multiclass complete random tree M is constructed on D. For a given *NI_threshold* in D, the points whose *NI* value are larger than the *NI_threshold* are identified as label noise.

In contrast to the definition of label noise in binary classification, the labels of the predecessors of a node are likely to span more than one class in multiclass classification. It is more difficult to remain invariant than it is in binary classification. The process of detecting multiclass label noise also consists of two parts. First, a complete random forest must be constructed, where each tree in the forest is generated by simple random node-splitting. Second, label noise must be detected according to the given threshold. Per Definition 6, the *NI* value for each sample in the leaf nodes can be calculated easily once the tree is constructed. Thus, samples with *NI* values greater than the given *NI_threshold* will be identified as label noise based on Definition 7. The detection results rely on the given *NI_threshold*. In Figure 1, when *NI_threshold* is set to 2, the 28^{th} point and the 20^{th} point will be recognized as label noise; when *NI_threshold* is set to three, only the 28^{th} point will be recognized as label noise. Therefore, we propose two novel methods, a voting-cross-validation method and an adaptive method, to optimize the *NI_threshold* parameter in the following sections.

B. A Novel Voting Cross-Validation Method to Optimize NI_threshold

As the label-noise points in each cross-training set are likely to change, the traditional cross-validation method cannot be used directly to detect label noise. We propose a novel cross-validation method for label noise filtering which incorporates a majority voting mechanism. We call our method the voting cross-validation method. We show an example in Figure 2 with 5-fold cross-validation. The pseudocode for VCV is given in Algorithm 1.

Given a corrupted training data set D'_{tr} and *Max_NI_threshold* (NI^m), we split D'_{tr} into 5 disjoint subsets D'_1, ..., D'_r, ..., D'_5, and let NI^i range from 1, 2, ...,

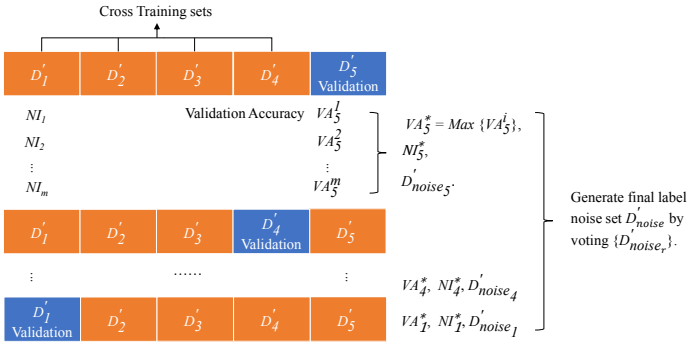


Fig. 2. Example of identifying label-noise samples using the VCV method.

to $[log_2(N)]$, where $i = 1, 2, ..., m$, $[\cdot]$ represents rounding down the bracketed number, and N denotes the number of the samples in D'_{tr}. The VCV method consists of two processes, validation and voting. In the validation process for D'_5, a multiclass complete random forest is constructed. Then, for each NI^i_5, $i = 1, 2, ..., m$, the corresponding validation accuracy using mCRF-learning on D'_5 is notated by VA^i_5. The highest validation accuracy among VA^i_5, $i = 1, 2, ..., m$ is marked VA^*_5, and its corresponding *NI_threshold* and detected noisy samples are respectively notated NI^*_5 and D'_{noise_5}.

In the voting process, if a sample appears in over 50% of $D'_{noise_r}, r = 1, 2, ..., 5$, the sample will be identified as a final noisy sample. As shown in Figure 2, all final noisy samples make up the final label noise set D'_{noise}.

The VCV method is advantageous for label noise filtering learning in particular because the voting mechanism can enhance the reliability of detected label noise, mitigating over-cleaning. In addition, since the distributions of different data sets are not identical, using a fixed *NI_threshold* in mCRF is

Algorithm 1: Detection of label-noise samples using the VCV method.

Input: A corrupted training data set D'_{tr}, splits *R*, *Ntree*, *Max_NI_threshold*.
Output: D'_{noise}, the optimal NI_threshold NI^*.

```
Split D'_tr into R disjoint subsets D'_1, ..., D'_r, ..., D'_R;
for each split D'_r, r=1, 2, ..., R do
    Construct a multiclass complete random forest using the remaining (R-1) splits;
    Calculate the NI value of each sample in the leaf nodes of each tree using Definition 6;
    for each NI_threshold NI^i, i=1, 2, ..., m do
        Detect noisy samples using Definition 7 and mCRF;
        Filter the noisy samples and obtain validation accuracy VA^i_r using a classifier on D'_r;
    end for
    VA^*_r = the highest validation accuracy among VA^i_r, i = 1, 2, ..., m;
    NI^*_r = the corresponding NI_threshold;
    D'_{noise_r} = the corresponding detected noisy samples;
    for each sample x do
        if x appears in over 50% of D'_{noise_r}, r = 1, 2, ..., R then
            x will be identified as a final noisy sample and saved into D'_noise;
        end if
    end for
end for
NI* = Mode_{r=1}^{R} {NI^*_r}; // Mode: the most frequent value.
return D'_noise, NI*.
```

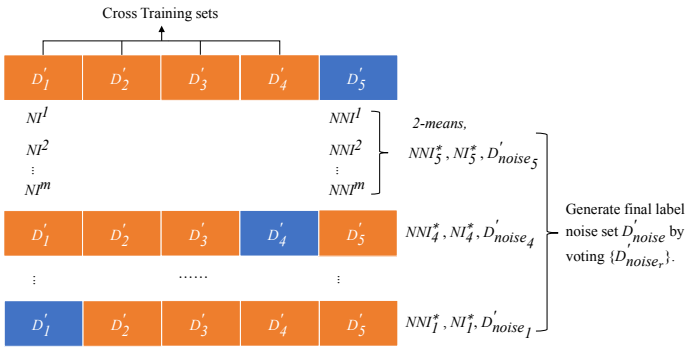


Fig. 3. Example of identifying label-noise samples using our adaptive method.

not reasonable. Assembling a set of *NI_thresholds* during the VCV method is likely to be better than using a fixed value. This is demonstrated in our experimental results.

C. An Adaptive Method to Optimize NI_threshold

Inspired by the VCV method, we now develop an adaptive method to optimize *NI_threshold*. In contrast to the previous optimization of the *NI_threshold* parameter under the guidance of validation accuracy, now *NI_threshold* is dependent on the stable number of identified label-noise samples. The degree of stability is difficult to quantify statistically. However, we note that the number of samples identified as noisy when the *NI_threshold* parameter is smaller than its optimal value is observably different than the number of samples identified as noisy when the parameter is larger than the optimal value. This is due to the different characteristics of noisy and normal samples. Based on this observation, we use 2-means clustering to search for a stable number of samples identified as noisy. Figure 3 shows an example of identifying label-noise samples using our adaptive method. As in the VCV example, we split the corrupted training data set D'_{tr} into 5 disjoint subsets D'_1, ..., D'_r, ..., D'_5. Different cross training sets again result in different constructed forests. Our adaptive method also consists of two steps, clustering and voting. The clustering process for D'_5 begins with the construction of a multiclass complete random forest. Then, for each $NNI^i_5, i = 1, 2, ..., m$, the corresponding label-noise samples are detected using Definition 7 and mCRF. The number of detected label-noise samples $NNI^i_5, i = 1, 2, ..., m$ can be counted instantly for each NI^i_5.

After this, we apply 2-means clustering to the set of numbers $NNI^i_5, i = 1, 2, ..., m$. This set is naturally split into two clusters, C_1 and C_2. Assuming that C_1 is the cluster with a bigger center c_1 and C_2 is the cluster with a smaller center c_2, the maximum value NNI^*_5 in cluster C_2 is selected as the stable number of label-noise samples, and the corresponding *NI_threshold* is selected as the optimal NI^*_5. A one-dimensional vector will be divided into two parts immediately by the 2-means algorithm, greatly enhancing the efficiency of optimizing the *NI_threshold* hyperparameter. Additionally, keeping the R-fold splits and voting mechanism in the optimization process maintains the effectiveness of label-noise detection at a high level. More details are found in Algorithm 2.

Algorithm 2: Detection of label-noise samples using our adaptive method.

Input: A corrupted training data set D'_{tr}, splits *R*, *Ntree*, *Max_NI_threshold*.
Output: D'_{noise}, the optimal NI_threshold NI^*.

```
Split D'_tr into R disjoint subsets D'_1, ..., D'_r, ..., D'_R;
for each split D'_r, r=1, 2, ..., R do
    Construct a multiclass complete random forest using the remaining (R-1) splits;
    Calculate the NI value of each sample in the leaf nodes of each tree using Definition 6;
    for each NI_threshold NI^i, i=1, 2, ..., m do
        Detect noisy samples using Definition 7 and mCRF;
        Count the number NNI^i of identified label-noise samples;
    end for
    Apply 2-means clustering algorithm to split NNI^i, i = 1, 2, ..., m into two clusters C_1 and C_2, where c_1 and c_2 represent the centers of C_1 and C_2 respectively, and c_1 ≥ c_2;
    Set NNI*_r = the maximum of NNI^i_r, i = 1, 2, ..., m, in C_2 (the cluster with the smaller center c_2) as the stable number of identified label noise;
    NI*_r = the corresponding NI_threshold;
    D'_{noise_r} = the corresponding detected noisy samples;
end for
for each sample x do
    if x appears in over 50% of D'_{noise_r}, r = 1, 2, ..., R then
        x will be identified as a final noisy sample;
    end if
end for
NI* = Mode_{r=1}^{R} {NI*_r}; // Mode: the most frequent value.
return D'_noise, NI*.
```

D. Time complexity

Our mCRF noise filtering learning framework consists of two stages. Stage 1 constructs a multiclass random forest and calculates the *NI* value for each sample. Stage 2 searches for an appropriate *NI_threshold* to detect label noise. Assuming that the training set contains N samples which are split into R folds, and the complete random forest contains t trees, the time complexity of Stage 1 is approximate $O(tNlogN)$, as analyzed in [15]. In Stage 2, the time complexity of the VCV method is dominated by the classifier used in the validation process, the time complexity of which we label T. Then the total time complexity of VCV-based mCRF amounts to $O(R * (tNlogN + T))$. Under the guidance of validation accuracy, VCV-based mCRF identifies label noise stably and precisely. However, the validation process can be time-consuming when applied to very large data sets. Our proposed adaptive method remedies this shortcoming. By using a 2-means clustering algorithm on single-dimensional data, the time cost is negligible due to the fast convergence speed of 2-means. Thus, the time complexity of adaptive-based mCRF mainly lies in Stage 1, which is $O(R * (tNlogN))$.

E. Computing v_i in mCRF

In the mCRF method, v_i as seen in (3) is defined as:

$$v_i = \begin{cases} 0, \sum_{r=1}^{R} 1_{NT(F(X_i; Ntree, NI\_threshold^*, w), \hat{Y}_i) \geq 50\% * Ntree} \geq R * 50\% \\ 1, otherwise \end{cases}, \tag{4}$$

where $NT\left(F\left(X_i; Ntree, NI\_threshold^*, w\right)\right)$ denotes the number of trees recognizing X_i as label noise. When the value of $NT(\cdot)$ is greater than 50% of the trees in the forest of one

split, the value of the indicator function $\mathrm{I}(\cdot)$ is equal to 1; when the sum of R indicators is over 50%*R, the instance will be recognized as label noise. By introducing cross-validation and the majority voting mechanism, this identification process becomes more stable and precise.

V. mRD: A Multiclass Version of RD Noise Filtering

In this section, based on Definitions 4 and 5, we develop the relative density filtering learning method from a binary classification method into a multiclass classification method.

A. The Multiclass Relative Density Method

We first use Definition 5 to provide specific definitions of multiclass relative denisity and the relevant k-nearest neighbors for easy reference.

Definition 8: Multiclass k-nearest heterogeneous and homogeneous neighbors.

Let $D'_{tr} = \left\{\left(X_i, \hat{Y}_i\right)\right\}_{i=1}^{n}$ be a corrupted training data set, where D'_{tr} consists of m classes with n samples. Let D'_s represent the s^{th} class, and D'_o represent the remaining (m-1) classes. For all $p \in D'_s$, the k-nearest *homogeneous neighbors* of p are defined as $\mathrm{HONk}(p) = \{q \,|q \in D'_s|\, d(p,q) \leq k - distance(p)\}$, and the k-nearest *heterogeneous neighbors* of p are defined as $HENk(p) = \{q \,|q \in D'_o|\, d(p,q) \leq k - distance(p)\}$.

In a multiclass scenario, the heterogeneous neighbors of a point refer to those neighbor samples that have differing labels, instead of the opposing label as in binary classification. Based on Definition 8, the multiclass relative density is defined as follows.

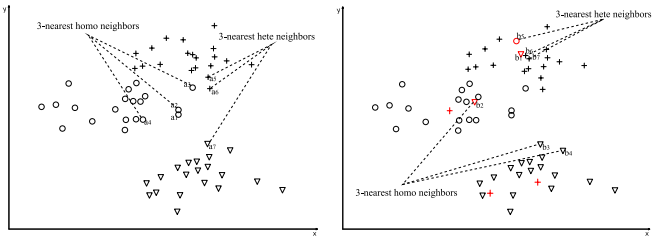
Definition 9: Multiclass relative density.

Let $D'_{tr} = \left\{\left(X_i, \hat{Y}_i\right)\right\}_{i=1}^{n}$ be a corrupted training data set, where D'_{tr} consists of m classes with n samples. Let D'_s represent the s^{th} class, and D'_o represent the remaining (m-1) classes. For all $p \in D'_s$, the *relative density* of p is defined as:

$$\begin{aligned} Relative\ Density(p) &= \frac{Absolute\ density\,(p, D'_o)}{Absolute\ density\,(p, D'_s)} \\ &= \frac{k/\Sigma d(p,o)(o \in HENk(p))}{k/\Sigma d(p,q)(q \in HONk(p))}. \\ &= \frac{\sum d(p,q)(p \in HONk(o))}{\sum d(p,o)(q \in HENk(o))} \end{aligned} \quad (5)$$

For each sample in each class, the $RelativeDensity$ (RD) is calculated by finding the k-nearest homogeneous neighbors in the same class and the k-nearest heterogeneous neighbors in the other m-1 classes, and then computing the RD value using (5). Of significant note, the RD value of a noisy sample is greater than 1; thus, here, the threshold to detect to label-noise samples is set to one.

In Figure 4, there are samples from three classes represented by circle, inverted triangle, and plus symbols, respectively; label-noise samples are colored in red. Figure 4(a) shows the underlying noise-free data set, where each point is close to other points of the same class and far away from points in different classes. Letting k=3, for queried point a_1, we can easily see that a_2, a_3, and a_4 are the *3*-nearest homogeneous neighbors, as they all belong to the circle class. Similarly, a_5, a_6, and a_7 are the *3*-nearest heterogeneous neighbors, and they belong to the triangle or plus classes. Obviously, the sum of distances between a_1 and its *3*-nearest heterogeneous neighbors is greater than the sum of distances between a_1 and its *3*-nearest homogeneous neighbors. Similarly, in Figure 4(b) the *3*-nearest homogeneous neighbors and *3*-nearest heterogeneous neighbors of the label noise sample b_1 are b_2, b_3, and b_4, and b_5, b_6, and b_7, respectively. However, the sum of distances between b_1 and its *3*-nearest heterogeneous neighbors is smaller than the sum of distances between b_1 and its *3*-nearest homogeneous neighbors, indicating that this is likely a noisy sample. We provide the fine details of the algorithm design in Algorithm 3.



(a) Clean data set (b) Corrupted data set

Fig. 4. Illustration of heterogeneous and homogeneous neighbors in clean and corrupted data sets.

Algorithm 3: Detection of label-noise samples using the mRD method.

Input: A corrupted training data set D'_{tr}, number of nearest neighbors k.
Output: D'_{noise}.

```
//1. Calculate the RD value for each point
for each class S do
    for each sample X_i in S do
        Calculate the Euclidean distance d(X_i, X_j) between i^th and the j^th points;
        Find the homogeneous k-nearest neighbors in the same class (X_j ∈ S);
        Find the heterogeneous k-nearest neighbors in the remaining m − 1 classes (X_j ∉ S);
        Calculate the RD value RD_i based on (5);
    end for
end for
//2. Detect the label noise samples
for each sample x do
    if RD_x ≥ 1 then
        X_i will be identified as a label-noise sample and saved into D'_noise;
    end if
end for
return D'_noise.
```

B. Time complexity

The time complexity of our mRD noise filtering learning framework mostly depends on the calculation of the RD value for each sample. Given N samples in the training set, we must compute the Euclidean distances between each query sample with every other sample; the time complexity of mRD is therefore $O(N^2)$.

C. Compute v_i in the mRD

In mRD, we define v_i from (3) by the equation:

$$v_i = \begin{cases} 0, RD\left(X_i\right) < 1 \\ 1, RD\left(X_i\right) \geq 1. \end{cases} \tag{6}$$

This implies that the loss of noisy samples should be set to zero to minimize the objective function. This multiclass filtering framework can be combined with almost all types of classifiers.

VI. EXPERIMENTAL RESULTS

We conduct a series of experiments to empirically evaluate the performance of the proposed algorithms. We first test on synthetic data sets to intuitively demonstrate the performance of the filtering learning frameworks for multiclass classification, showing examples of both mCRF and mRD. We then select the state-of-the-art competitor method, the importance reweighting method (mIW), as our baseline for comparison. mIW leverages a reweighting strategy to minimize the influence of label noise by assigning a low ratio to incorrectly labeled samples in multiclass classification tasks; this ratio is independent of any specific loss function. We also run an experiment in which we incorporate the prevalent Synthetic Minority Oversampling Technique (SMOTE) with our proposed label noise-filtering learning framework, as well incorporating it with the state-of-the-art alternative, in order to alleviate the class imbalance problem in multiclass classification. We then run both SMOTE-enhanced methods on imbalanced multiclass data sets with label noise. Finally, we test the proposed adaptive method on several real-world UCI data sets; in order to verify the proposed framework, we combine it with a selection of traditional classifiers including LR, DT, SVM, Adaboost, LightGBM, and XGboost.

Note that, in mCRF, in order to select the proper *NI_threshold* parameter from the finite set of possible values $\{2, 3, ..., Max\_NI\_Threshold\}$, a validation set of 10% training data is randomly held back from the training set. Instead of fixing the *Max_NI_threshold* at 11 as in [15], we provide a reference value $log2(n)$ as the *Max_NI_threshold*, where n denotes the cardinality of the training data set D'_{tr}.

The labels of the training samples are flipped stochastically according to a noise rate γ, while the test partitions are noise-free, as seen in the methodology of [43]. This allows one to observe how noisy data affect the training process and how the test results are degraded depending on the level of label noise. In the experiments, to balance between the computational overhead and obtaining enough experimental results to draw conclusions, the noise rates γ under consideration are $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$.

A. Synthetic Data

To intuitively show the effectiveness of the two proposed methods, four 2-dimensional data sets are created using the `make_blobs`, `make_moons`, `make_circles`, and `make_classification` functions of the Python library `scikit-learn`. Each data set contains 1000 points and consists of three classes, which we label circle, triangle_down, and plus. The percentage of samples in each class is 20%, 35%, and 45%, respectively. The noisy samples in the training set are generated by flipping the labels according to a given noise rate γ, where $\gamma \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$. Six traditional classifiers—LR, DT, SVM, Adaboost, LightGBM, and XGboost—are chosen as basic classifiers. For fair comparison, the parameters of all classifiers are set to default. For simplicity, only the results of the SVM with a default Gaussian kernel width at the noise rate γ=0.4 are presented in Figures 5 to 8 in this section. Some results with $\gamma \in \{0.1, 0.2, 0.3, 0.4\}$ are presented in Figs. 10 to 101 in Appendix A; other results are uploaded to Github using Git LFS as too many pictures will make it difficult to read the paper. Though omitted, the results of any other desired classifier or noise rate will not change the conclusions drawn. Each figure consists of eight subfigures. The classification hyperplanes are colored in green, yellow, and orange. Both noisy samples in the training sets and misclassified samples in the test sets are colored in red. Test accuracy is given in brackets. In each figure, (a) displays the hyperplane trained with the true noise-free data, which is used as the touchstone to measure the performance of the label noise-filtering framework, and (b) shows the hyperplane trained with the corrupted data with 40 percent of the labels flipped stochastically, where the red-colored points denote the synthetic label noise. It can be seen from Figures 5(b), 6(b), 7(b), and 8(b) that label noise disturbed the distribution of the original noise-free data set, causing the hyperplane to poorly separate the samples into three classes. The accuracies of 0.904, 0.956, 0.956, and 0.916 in Figures 5(f), 6(f), 7(f), and 8(f) are lower than those in Figures 5-8(e) respectively, that is, 0.940, 0.980, 0.992, and 0.956. This indicates that the predictive performance of the classifier was deteriorated in the presence of label noise, making the separable data set nonseparable. To mitigate the influence of label noise, we apply the two proposed methods on the corrupted data set. Figures 5-8(c) show the classification results of the mCRF-SVM, where the noisy samples are detected and filtered by mCRF. Similarly, Figures 5-8(d) illustrate the classification results of the mRD-SVM. The hyperplanes in Figures 5-8(c) and Figures 5-8(d) are quite close to the original hyperplanes, and the corresponding accuracies are also better than the non-filtered results. This indicates that the proposed label noise filtering learning framework is effective in enhancing the generalizability of the classifier. Similar conclusions can also be drawn from Figures 10 to 101 in Appendix A. All accuracies obtained from the four synthetic data sets are summarized in Table V in Appendix A as well.

In summary, Figures 5-8 and Figures 10-101 demonstrate the efficiency of our proposed label noise detection methods and label noise filtering framework in multiclass scenarios. The label noise detection methods can effectively detect and filter label noise before the training process, thus minimizing the influence of label noise and making the resulting hyperplane as similar to the noise-free hyperplane as possible to improve the generalizability of classifiers. Even on a data set with clusters of different and complex shapes, as shown in Figure 8, mCRF still achieves good results.

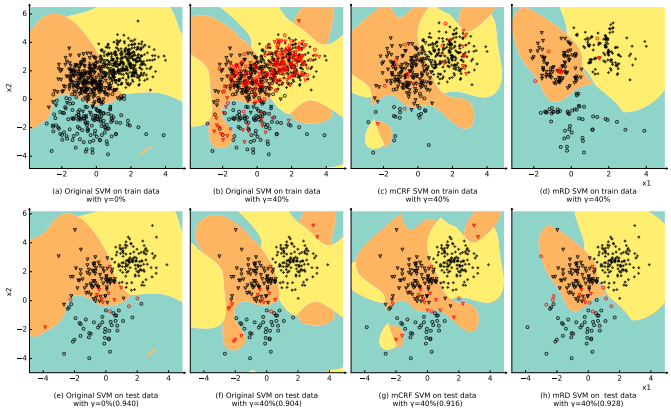


Fig. 5. Hyperplane transformation on blobs data set, $\gamma = 0\%$ and $\gamma = 40\%$.

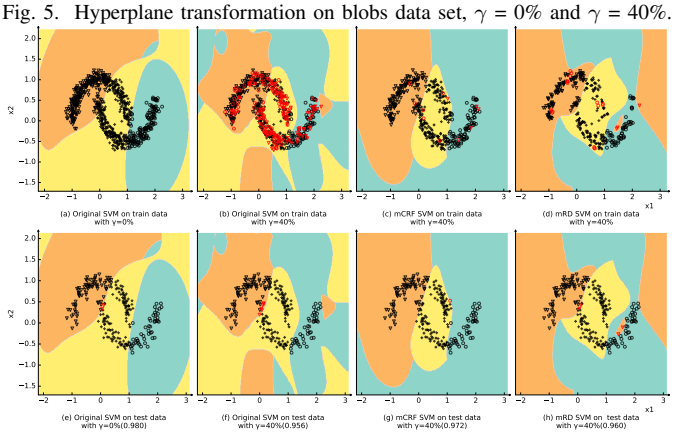


Fig. 6. Hyperplane transformation on moons data set, $\gamma = 0\%$ and $\gamma = 40\%$.

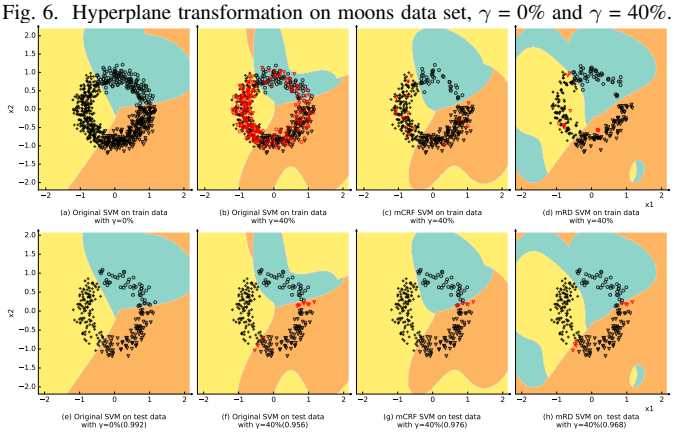


Fig. 7. Hyperplane transformation on circles data set, $\gamma = 0\%$ and $\gamma = 40\%$

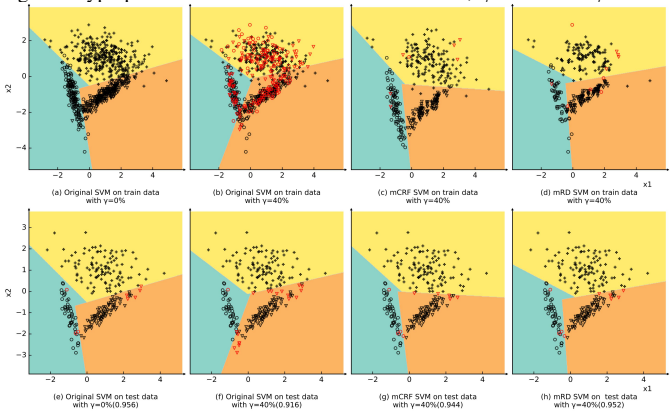


Fig. 8. Hyperplane transformation on artifical data set, $\gamma = 0\%$ and $\gamma = 40\%$

TABLE I
DATA SET INFORMATION

Data	#Features	#Classes	#Instances	Class distribution
iris	5	3	150	[50, 50, 50]
wine	14	3	178	[71, 59, 48]*
seeds	8	3	210	[70, 70, 70]
newthyroid	6	3	215	[150, 35, 30]*
vertebralColumn	7	3	310	[150, 100, 60]*
userknowledge	6	5	403	[129, 122, 102, 26, 24]*
balancescale	5	3	625	[288, 288, 49]*
PhishingData	10	3	1353	[702, 548, 103]*
segmentation	20	7	2310	[330, 330, 330, 330, 330, 330, 330]
satimage	37	6	3188	[788, 674, 597, 443, 391, 295]*
pendigits	17	10	7494	[780, 780, 780, 779, 778, 720, 720, 719, 719, 719]*
shuttle	10	7	43500	[34108, 6748, 2458, 132, 37, 11, 6]*

B. UCI Benchmarks

We conduct our experiments on 12 real data sets from the UCI repository[1] to demonstrate the performance of our proposed multiclass label noise filtering framework combined with various classifiers across different noise rates. Some reference metadata is presented in Table I, where # represents 'number of', and $*$ indicates the data set is imbalanced. We randomly divide each data set into five parts, and use each testing data while the remaining 4 are training data. This process is repeated five times. Thus, 25 testing accuracies are obtained per data set, classifier, and noise rate. The average value of these 25 test accuracies is our final test accuracy; which mitigates the randomness arising from splitting data sets and flipping the labels stochastically. Each accuracy has 3 dimensions—data set, classifier, and noise rate—on which to compare the performance of our proposed methods against our baseline. However, due to paper space limitations, we have further averaged the results across the classifier and data set in order to reduce dimensionality. Only the average accuracies of all data sets for each pair (one noise rate γ, one data set) and average accuracies of all classifiers for each pair (one noise rate γ, one classifier) are listed in Tables II to III, respectively, with the highest two values **in bold** in each row (relative improvement in parentheses). Relative improvement is the average accuracy the label noise filtration method minus the average score of the original method. The actual average results for per data set, classifier, and noise rate across all 25 runs are summarized in Tables VI to LIII in Appendix B.

Here, we investigate the performances of the two proposed methods to optimize *NI_threshold* in mCRF. To differentiate the VCV and adaptive methods, we adopt the respective notation VCV_mCRF and Adp_mCRF in the tables. Table II presents the average accuracies and improvements of the selected noise-filtering algorithms on the UCI data sets with a steadily-increasing noise rate γ. Reading across columns, we see all four noise-filtering methods as well as the baseline classifier report a drop in performance with the increase in noise rate γ for all data sets. However, the performance deterioration is lessened in our proposed algorithms, which manifest

[1] https://archive.ics.uci.edu/ml/index.php

TABLE II
COMPARISON OF AVERAGE ACCURACIES FOR SELECTED ALGORITHMS ON DIFFERENT DATA SETS AT VARYING NOISE RATES γ.

Data	Original	mIW	VCV_mCRF	Adp_mCRF	mRD	γ	Data	Original	mIW	VCV_mCRF	Adp_mCRF	mRD
balancescale	0.8436	0.8374 (-0.0062)	0.8399 (-0.0037)	**0.8627 (0.0191)**	**0.8916 (0.0480)**	0.05	seeds	**0.9106**	0.9005 (-0.0101)	**0.9206 (0.0100)**	**0.9206 (0.0100)**	0.8844 (-0.0262)
	0.8189	0.8141 (-0.0048)	**0.8290 (0.0101)**	0.8040 (-0.0149)	**0.8209 (0.0020)**	0.1		**0.8989**	0.8873 (-0.0116)	**0.8981 (-0.0008)**	0.8571 (-0.0418)	**0.8981 (-0.0008)**
	0.8036	0.8042 (0.0006)	**0.8210 (0.0174)**	**0.8467 (0.0431)**	0.8116 (0.0080)	0.15		0.8603	0.8889 (0.0286)	**0.9101 (0.0498)**	**0.9048 (0.0445)**	0.8857 (0.0254)
	0.7811	0.7847 (0.0036)	**0.7949 (0.0138)**	0.7827 (0.0016)	**0.8111 (0.0300)**	0.2		0.8495	0.8505 (0.0010)	**0.8841 (0.0346)**	**0.9008 (0.0513)**	0.8479 (-0.0016)
	0.7768	0.7825 (0.0057)	**0.7877 (0.0109)**	**0.8120 (0.0352)**	0.7511 (-0.0257)	0.25		0.8153	0.8561 (0.0408)	**0.8833 (0.0680)**	0.8730 (0.0577)	**0.8862 (0.0709)**
	0.7543	**0.7649 (0.0106)**	**0.7746 (0.0203)**	0.7387 (-0.0156)	0.7578 (0.0035)	0.3		0.7847	0.8299 (0.0452)	**0.8574 (0.0727)**	**0.9246 (0.1399)**	0.8513 (0.0666)
	0.7329	**0.7563 (0.0234)**	**0.7692 (0.0363)**	0.7253 (-0.0076)	0.7387 (0.0058)	0.35		0.7728	0.8474 (0.0746)	**0.8548 (0.0820)**	**0.9246 (0.1518)**	0.8529 (0.0801)
	0.7140	0.7585 (0.0445)	0.7497 (0.0357)	**0.7867 (0.0727)**	**0.7634 (0.0494)**	0.4		0.7386	0.8331 (0.0945)	0.8235 (0.0849)	**0.8611 (0.1225)**	**0.8508 (0.1122)**
Average	0.7781	0.7878 (0.0097)	**0.7958 (0.0177)**	**0.7948 (0.0167)**	0.7933 (0.0152)		Average	0.8288	0.8617 (0.0329)	**0.8790 (0.0502)**	**0.8958 (0.0670)**	0.8697 (0.0409)
iris	0.9359	0.9052 (-0.0307)	**0.9615 (0.0256)**	**0.9944 (0.0585)**	0.9574 (0.0215)	0.05	segmentation	**0.9172**	0.7680 (-0.1492)	0.8723 (-0.0449)	0.8734 (-0.0438)	**0.8749 (-0.0423)**
	0.9089	0.9074 (-0.0015)	0.9478 (0.0389)	**0.9556 (0.0467)**	**0.9589 (0.0500)**	0.1		**0.9041**	0.7521 (-0.1520)	0.8937 (-0.0104)	**0.9152 (0.0111)**	0.8603 (-0.0438)
	0.8930	0.9096 (0.0166)	**0.9389 (0.0459)**	0.8611 (-0.0319)	**0.9530 (0.0600)**	0.15		**0.8939**	0.7593 (-0.1346)	**0.9134 (0.0195)**	0.8716 (-0.0223)	0.8677 (-0.0262)
	0.8407	0.8904 (0.0497)	0.8815 (0.0408)	**0.9056 (0.0649)**	**0.9556 (0.1149)**	0.2		**0.8803**	0.7549 (-0.1254)	**0.8909 (0.0106)**	0.8745 (-0.0058)	0.8575 (-0.0228)
	0.8367	0.8778 (0.0411)	**0.9396 (0.1029)**	0.9000 (0.0633)	**0.9441 (0.1074)**	0.25		0.8643	0.7514 (-0.1129)	**0.8936 (0.0293)**	**0.8918 (0.0275)**	0.8655 (0.0012)
	0.8119	0.8681 (0.0562)	**0.9267 (0.1148)**	0.8667 (0.0548)	**0.9415 (0.1296)**	0.3		0.8427	0.7715 (-0.0712)	**0.8804 (0.0377)**	**0.9044 (0.0617)**	0.8593 (0.0166)
	0.7770	0.8596 (0.0826)	**0.8937 (0.1167)**	**0.9278 (0.1508)**	0.8907 (0.1137)	0.35		0.8306	0.7522 (-0.0784)	**0.8740 (0.0434)**	**0.8604 (0.0298)**	0.8494 (0.0188)
	0.7384	0.8584 (0.1200)	**0.8853 (0.1469)**	**0.8778 (0.1394)**	0.8602 (0.1218)	0.4		0.8036	0.7603 (-0.0433)	**0.8718 (0.0682)**	**0.8709 (0.0673)**	0.8402 (0.0366)
Average	0.8428	0.8846 (0.0418)	**0.9219 (0.0791)**	0.9111 (0.0683)	**0.9327 (0.0899)**		Average	0.8671	0.7587 (-0.1084)	**0.8863 (0.0192)**	**0.8828 (0.0157)**	0.8594 (-0.0077)
newthyroid	0.9189	0.9258 (0.0069)	**0.9305 (0.0116)**	0.9070 (-0.0119)	**0.9269 (0.0080)**	0.05	shuttle	**0.9696**	0.1128 (-0.8568)	0.9695 (-0.0001)	**0.9791 (0.0095)**	0.9552 (-0.0144)
	0.9080	0.9080 (0.00000)	0.9083 (0.0003)	**0.9186 (0.0106)**	**0.9194 (0.0114)**	0.1		0.9561	0.1219 (-0.8342)	**0.9668 (0.0107)**	**0.9567 (0.0006)**	0.9513 (-0.0048)
	0.8819	0.8997 (0.0178)	**0.9152 (0.0333)**	0.8798 (-0.0021)	**0.9238 (0.0419)**	0.15		0.9468	0.1580 (-0.7888)	**0.9541 (0.0073)**	**0.9658 (0.0190)**	0.9502 (0.0034)
	0.8530	0.8726 (0.0196)	0.8775 (0.0245)	**0.8953 (0.0423)**	**0.9165 (0.0635)**	0.2		0.9348	0.1275 (-0.8073)	0.9612 (0.0264)	**0.9726 (0.0378)**	**0.9644 (0.0296)**
	0.7848	0.8605 (0.0757)	0.8928 (0.1080)	**0.9264 (0.1416)**	**0.8995 (0.1147)**	0.25		0.9200	0.1302 (-0.7898)	**0.9586 (0.0386)**	**0.9634 (0.0434)**	0.9511 (0.0311)
	0.8165	0.8390 (0.0225)	0.8742 (0.0577)	**0.8953 (0.0788)**	**0.8773 (0.0608)**	0.3		0.9117	0.0745 (-0.8372)	0.9591 (0.0474)	**0.9576 (0.0459)**	**0.9650 (0.0533)**
	0.7646	0.8276 (0.0630)	0.8297 (0.0651)	**0.8333 (0.0687)**	**0.8599 (0.0953)**	0.35		0.8969	0.1113 (-0.7856)	**0.9655 (0.0686)**	**0.9682 (0.0713)**	0.9572 (0.0603)
	0.7359	0.8178 (0.0819)	0.8394 (0.1035)	**0.8566 (0.1207)**	**0.8725 (0.1366)**	0.4		0.8867	0.1186 (-0.7681)	**0.9628 (0.0761)**	**0.9574 (0.0707)**	0.9493 (0.0626)
Average	0.8329	0.8689 (0.0360)	0.8834 (0.0505)	**0.8890 (0.0561)**	**0.8995 (0.0666)**		Average	0.9278	0.1194 (-0.8084)	**0.9622 (0.0344)**	**0.9651 (0.0373)**	0.9555 (0.0277)
pendigits	**0.9304**	0.9171 (-0.0133)	0.8856 (-0.0448)	**0.9184 (-0.0120)**	0.8656 (-0.0648)	0.05	userknowledge	0.7974	**0.7983 (0.0009)**	0.7966 (-0.0008)	**0.8436 (0.0462)**	0.7852 (-0.0122)
	0.9215	**0.9117 (-0.0098)**	0.8780 (-0.0435)	0.8932 (-0.0283)	0.8500 (-0.0715)	0.1		**0.8151**	0.8082 (-0.0069)	0.7968 (-0.0183)	**0.8107 (-0.0044)**	0.7960 (-0.0191)
	0.9158	**0.9104 (-0.0054)**	0.8605 (-0.0553)	0.9067 (-0.0091)	0.8582 (-0.0576)	0.15		0.7970	0.7799 (-0.0171)	0.7932 (-0.0038)	**0.8210 (0.0240)**	**0.8135 (0.0165)**
	0.9034	**0.9075 (0.0041)**	0.8685 (-0.0349)	0.8853 (-0.0181)	0.8523 (-0.0511)	0.2		**0.7762**	0.7686 (-0.0076)	0.7554 (-0.0208)	0.7531 (-0.0231)	**0.7772 (0.0010)**
	0.8929	**0.9100 (0.0171)**	0.8844 (-0.0085)	**0.9134 (0.0205)**	0.8555 (-0.0374)	0.25		0.7322	0.7467 (0.0145)	0.7488 (0.0166)	**0.7984 (0.0662)**	**0.7767 (0.0445)**
	0.8784	**0.8997 (0.0213)**	0.8934 (0.0150)	**0.9075 (0.0291)**	0.8559 (-0.0225)	0.3		0.7227	0.7495 (0.0268)	**0.7643 (0.0416)**	**0.7695 (0.0468)**	0.7575 (0.0348)
	0.8661	**0.8980 (0.0319)**	0.8945 (0.0284)	**0.9144 (0.0483)**	0.8378 (-0.0283)	0.35		0.6925	0.7144 (0.0219)	0.7334 (0.0409)	**0.7346 (0.0421)**	**0.7465 (0.0540)**
	0.8496	**0.8989 (0.0493)**	0.8932 (0.0436)	**0.9207 (0.0711)**	0.8352 (-0.0144)	0.4		0.6608	0.7196 (0.0588)	**0.7299 (0.0691)**	0.6667 (0.0059)	**0.7516 (0.0908)**
Average	0.8948	**0.9067 (0.0119)**	0.8823 (-0.0125)	**0.9074 (0.0126)**	0.8513 (-0.0435)		Average	0.7492	0.7607 (0.0115)	0.7648 (0.0156)	**0.7747 (0.0255)**	**0.7755 (0.0263)**
PhishingData	0.8567	0.8564 (-0.0003)	0.8536 (-0.0031)	**0.8807 (0.0240)**	**0.8574 (0.0007)**	0.05	vertebralColumn	0.7980	**0.8242 (0.0262)**	0.8095 (0.0115)	**0.8575 (0.0595)**	0.7953 (-0.0027)
	0.8394	**0.8510 (0.0116)**	0.8397 (0.0003)	0.8241 (-0.0153)	**0.8560 (0.0166)**	0.1		0.8047	0.8084 (0.0037)	**0.8102 (0.0055)**	**0.8306 (0.0259)**	0.8070 (0.0023)
	0.8247	**0.8435 (0.0188)**	0.8327 (0.0080)	0.7749 (-0.0498)	**0.8491 (0.0244)**	0.15		0.7841	**0.8102 (0.0261)**	0.7996 (0.0155)	0.7823 (-0.0018)	**0.8095 (0.0254)**
	0.8157	**0.8300 (0.0143)**	0.8241 (0.0084)	0.8266 (0.0109)	**0.8433 (0.0276)**	0.2		0.7484	**0.8093 (0.0609)**	**0.7900 (0.0416)**	0.7339 (-0.0145)	0.7844 (0.0360)
	0.7990	**0.8223 (0.0233)**	0.8197 (0.0207)	0.8167 (0.0177)	**0.8282 (0.0292)**	0.25		0.7285	**0.8029 (0.0744)**	0.7916 (0.0631)	0.7661 (0.0376)	**0.7987 (0.0702)**
	0.7825	**0.8135 (0.0310)**	0.8086 (0.0261)	0.8069 (0.0244)	**0.8181 (0.0356)**	0.3		0.7229	**0.7937 (0.0708)**	**0.7858 (0.0629)**	0.7392 (0.0163)	0.7185 (-0.0044)
	0.7731	**0.8003 (0.0272)**	0.8001 (0.0270)	**0.8198 (0.0467)**	0.7956 (0.0225)	0.35		0.6862	**0.7670 (0.0808)**	0.7489 (0.0627)	**0.8118 (0.1256)**	0.7511 (0.0649)
	0.7371	0.7836 (0.0465)	**0.7908 (0.0537)**	**0.8063 (0.0692)**	0.7754 (0.0383)	0.4		0.6535	**0.7734 (0.1199)**	**0.7424 (0.0889)**	0.7097 (0.0562)	0.7286 (0.0751)
Average	0.8035	**0.8251 (0.0216)**	0.8212 (0.0177)	0.8195 (0.0160)	**0.8279 (0.0244)**		Average	0.7408	**0.7986 (0.0578)**	**0.7848 (0.0440)**	0.7789 (0.0381)	0.7741 (0.0333)
satimage	**0.8567**	0.8562 (-0.0005)	0.8205 (-0.0362)	**0.8676 (0.0109)**	0.8218 (-0.0349)	0.05	wine	0.9485	0.9475 (-0.0010)	**0.9537 (0.0052)**	**0.9583 (0.0098)**	0.8738 (-0.0747)
	0.8463	**0.8496 (0.0033)**	**0.8565 (0.0102)**	0.8346 (-0.0117)	0.8102 (-0.0361)	0.1		0.9210	0.9172 (-0.0038)	**0.9470 (0.0260)**	**0.9583 (0.0373)**	0.8514 (-0.0696)
	0.8354	**0.8446 (0.0092)**	0.8391 (0.0037)	**0.8715 (0.0361)**	0.8188 (-0.0166)	0.15		0.8942	**0.9184 (0.0242)**	0.9031 (0.0089)	**0.9537 (0.0595)**	0.8445 (-0.0497)
	0.8169	**0.8341 (0.0172)**	**0.8457 (0.0288)**	0.8075 (-0.0094)	0.8224 (0.0055)	0.2		0.8712	**0.9264 (0.0552)**	**0.9164 (0.0452)**	0.8796 (0.0084)	0.8209 (-0.0503)
	0.8027	**0.8272 (0.0245)**	**0.8103 (0.0076)**	0.7962 (-0.0065)	0.8055 (0.0028)	0.25		0.8448	0.8826 (0.0378)	**0.8864 (0.0416)**	**0.9120 (0.0672)**	0.8162 (-0.0286)
	0.7934	0.8147 (0.0213)	**0.8418 (0.0484)**	**0.8375 (0.0441)**	0.8117 (0.0183)	0.3		0.7956	**0.8713 (0.0757)**	0.8656 (0.0700)	**0.8704 (0.0748)**	0.7953 (-0.0003)
	0.7743	0.8058 (0.0315)	**0.8367 (0.0624)**	0.7986 (0.0243)	**0.8110 (0.0367)**	0.35		0.8041	**0.8647 (0.0606)**	**0.8684 (0.0643)**	0.8519 (0.0478)	0.7749 (-0.0292)
	0.7570	0.7948 (0.0378)	**0.8178 (0.0608)**	**0.8258 (0.0688)**	0.8133 (0.0563)	0.4		0.7501	**0.8747 (0.1246)**	**0.8223 (0.0722)**	0.7546 (0.0045)	0.7350 (-0.0151)
Average	0.8103	0.8284 (0.0181)	**0.8336 (0.0233)**	**0.8299 (0.0196)**	0.8143 (0.0040)		Average	0.8537	**0.9003 (0.0466)**	**0.8954 (0.0417)**	0.8924 (0.0387)	0.8140 (-0.0397)

a distinct advantage over the baselines. When γ increases to a high level—30% to 40% noise—the advantages of noise-filtering often become more significant. This demonstrates the effectiveness and necessity of label noise filtration methods.

From an average perspective, VCV_mCRF performs best on balancescale, satimage, and segmentation with an average improvement of 1.77%, 2.33% and 1.92%, respectively; Adp_mCRF performs best on pendigits, seeds, and shuttle with an average improvement of 1.26%, 6.70%, and 3.73%, respectively. From the research metadata, we see that VCV_mCRF and Adp_mCRF are better suited to midsize to larger data sets. For smaller data sets, including iris, newthyroid, userknowledge, wine, and vertebral, mCRF-based methods have difficulty gathering enough information to detect label noise in the multiclass complete random forest generated by the small amount of data available. In contrast, mIW resorts to KLIEP to estimate the conditional probability $P_{D_\gamma}(Y|X)$, which is effective on midsize data sets [43]. It achieves the best performance on vertebralColumn and wine with the improvements of 5.78% and 4.66% for each data set. However, on the data set shuttle, mIW falls into invalidation as the minority sample of the extremely imbalanced data could not provide enough information to accurately estimate the probability. Because it relies on the Euclidean distances of query samples from homogeneous and heterogeneous neighbors to detect label noise, mRD usually performs well on uniformly distributed data sets [13]. mRD performs best on the data sets iris, newthyroid, PhishingData, and userknowledge with the highest average improvements of 8.99%, 6.66%, 2.44%, and 2.63%, respectively. However, mRD depends on data distribution more than mCRF-based methods, leading to weaker generalizability. This can be observed from trends in bold for each method in Table II. The overall statistics can be summarized by the number of times a method achieves the bold number: original classifier, 14; mIW, 36; VCV_mCRF, 59; Adp_mCRF, 68; and mRD, 41. This indicates that mCRF-based methods outperform other comparable methods.

We compare the proposed methods from Section IV for optimizing *NI_threshold* in mCRF with particular attention to three details: classification accuracy, value of *NI_threshold*, and time cost. We denote the *NI_threshold* value selected under the guidance of the cross-validation result as *vNI_threshold* and that selected by the adaptive method as *aNI_threshold*. From Table II, we can see that Adp_mCRF is highly competitive with VCV_mCRF as a filtering learning framework in many cases. Adp_mCRF even outperforms VCV_mCRF on the data sets pendigits, seeds, shuttle, and userknowledge with improvements of 2.51%, 1.68%, 0.29%, and 0.99%

over the VCV_mCRF method. To facilitate the comparison of the different noise intensity values searched by the two methods, we depict the values in line charts in Figures 103 to 114 in Appendix B. In the each line chart, blue square points represent *vNI_threshold*, red triangle points represent *aNI_threshold*, and green stars represent an overlap between *vNI_threshold* and *aNI_threshold*. Analyzing this data, we find that 80.03% of the differences between thresholds are within 3 intervals. We also find that *vNI_threshold* and *aNI_threshold* generate differences in accuracies that are smaller than 5% about 72.92% of the time, and 90.63% of the time generate less than 10% difference in accuracy.

Time efficiency is a significant advantage of the adaptive method. Because of the fast convergence speed of 2-means clustering, one-dimensional data will converge to two clusters almost instantly. Hence, the *aNI_threshold* value corresponding to the maximum number in the smaller cluster will be readily obtained. In contrast, the search for the *vNI_threshold* value is dominated by many median classifiers used in the validation process. Thus, the time cost may fluctuate depending on which classifier is used. We use bar graphs to show the amount of time require to search for the proper *NI_thresholds* in Figure 102 in Appendix B. Figure 102 consists of 12 sub-figures, and each sub-figure corresponds to a data set. The green and orange bars respectively indicate the validation time and the 2-means clustering time. The average time cost across all data sets and classifiers for the two methods are 187.85s and 0.39s, respectively; that is, the time VCV_mCRF searches is 484 times longer than Adp_mCRF searches. We can also observe that the time cost of VCV_mCRF largely depends on the size of the data set and the selected classifier. On small and midsize data sets, the time overhead of both methods and all classifiers is tolerable. On large data sets, such as shuttle, SVM is time-consuming. In summary, we find that Adp_mCRF can achieve approximately the same excellent performance as VCV_mCRF, but with a lower time cost.

Table III lists the average accuracies and improvements of the selected multiclass classifiers combined with different noise-filtration methods at varying noise rates. These average scores are obtained across eleven UCI data sets; we exclude shuttle, because the extreme imbalance of this data set results in very poor performance in the baseline mIW method, resulting the lowest average mIW score and completely swamping any benefit from mitigating label noise. We return to examine imbalanced data sets in more detail later. The DT classifier appears to profit the most from our noise-filtering learning framework, with improvements to average scores reaching 14.03%, 15.00%, and 15.97% for VCV_mCRF, Adp_mCRF, and mRD, respectively. The largest average score improvement of 25.56% was also achieved using DT at $\gamma = 0.4$ in the mRD learning framework. We note poor performance of all four methods in combination with Adaboost, and investigate the issue on each data set at a different noise rate γ in Tables VI to LIII. Three data sets, pendigits, segmentation, and shuttle, suffer particularly sharp decreases in accuracy, leading to the overall low averages reported for our filtration methods in combination with Adaboost. Excepting these data sets, mIW-Adaboost performs best among the four methods, attaining average scores of 82.27% across the nine data sets, with 1.35% improvement over unmodified Adaboost. We observe that except for Adaboost, all other classifiers benefit more from our filtering learning frameworks. LightGBM benefits most with the highest average enhancement of 4.03% using the VCV_mCRF learning framework. LR and SVM attain maximum average enhancements of 2.04% and 1.09% using the Adp_mCRF learning framework. DT and Xgboost achieve maximum average improvements of 15.97% and 5.90% using the mRD learning framework

TABLE III
COMPARISON OF AVERAGE ACCURACIES FOR SELECTED MULTICLASS CLASSIFICATION ALGORITHMS AT VARYING NOISE RATES γ.

Clf	γ	Original	mIW	VCV_mCRF	Adp_mCRF	mRD
Adaboost	0.05	**0.8285**	0.7970 (-0.0315)	0.7561 (-0.0724)	**0.8397 (0.0112)**	0.7112 (-0.1173)
	0.1	**0.8522**	**0.8046 (-0.0476)**	0.7742 (-0.0780)	0.7861 (-0.0661)	0.7061 (-0.1461)
	0.15	**0.8426**	**0.8182 (-0.0244)**	0.7769 (-0.0657)	0.7435 (-0.0991)	0.7322 (-0.1104)
	0.2	**0.8290**	**0.7910 (-0.0380)**	0.7149 (-0.1141)	0.6905 (-0.1385)	0.6713 (-0.1577)
	0.25	**0.8008**	**0.8030 (0.0022)**	0.7493 (-0.0515)	0.7693 (-0.0315)	0.7167 (-0.0841)
	0.3	**0.8083**	**0.7963 (-0.0120)**	0.7653 (-0.0430)	0.7256 (-0.0827)	0.6644 (-0.1439)
	0.35	**0.7921**	**0.8025 (0.0104)**	0.7429 (-0.0492)	0.7117 (-0.0804)	0.6380 (-0.1541)
	0.4	**0.7788**	**0.8077 (0.0289)**	0.7360 (-0.0428)	0.7182 (-0.0606)	0.6864 (-0.0924)
Average		**0.8165**	**0.8025 (-0.0140)**	0.7520 (-0.0645)	0.7481 (-0.0684)	0.6908 (-0.1257)
DT	0.05	0.8474	0.8550 (0.0076)	0.8817 (0.0343)	**0.9020 (0.0546)**	**0.8846 (0.0372)**
	0.1	0.7979	0.7914 (-0.0065)	0.8620 (0.0641)	**0.8719 (0.0740)**	**0.8809 (0.0830)**
	0.15	0.7624	0.7777 (0.0153)	**0.8518 (0.0894)**	0.8479 (0.0855)	**0.8693 (0.1069)**
	0.2	0.7032	0.7516 (0.0484)	**0.8391 (0.1359)**	0.8219 (0.1187)	**0.8665 (0.1633)**
	0.25	0.6552	0.7000 (0.0448)	0.8289 (0.1737)	**0.8397 (0.1845)**	**0.8426 (0.1874)**
	0.3	0.6348	0.6716 (0.0368)	0.8193 (0.1845)	**0.8235 (0.1887)**	**0.8328 (0.1980)**
	0.35	0.5805	0.6335 (0.0530)	0.7970 (0.2165)	**0.8330 (0.2525)**	**0.8268 (0.2463)**
	0.4	0.5455	0.6253 (0.0798)	0.7701 (0.2246)	**0.7869 (0.2414)**	**0.8011 (0.2556)**
Average		0.6909	0.7258 (0.0349)	0.8312 (0.1403)	**0.8409 (0.1500)**	**0.8506 (0.1597)**
LightGBM	0.05	**0.9164**	0.8999 (-0.0165)	**0.9238 (0.0074)**	0.9077 (-0.0087)	0.9069 (-0.0095)
	0.1	0.8981	**0.9000 (0.0019)**	**0.9100 (0.0119)**	0.8890 (-0.0091)	0.8981 (0.0000)
	0.15	0.8781	0.8922 (0.0141)	**0.8983 (0.0202)**	**0.8934 (0.0153)**	0.8929 (0.0148)
	0.2	0.8603	0.8839 (0.0236)	**0.8900 (0.0297)**	0.8772 (0.0169)	**0.8873 (0.0270)**
	0.25	0.8434	0.8753 (0.0319)	**0.8834 (0.0400)**	**0.8825 (0.0391)**	0.8691 (0.0257)
	0.3	0.8096	0.8632 (0.0536)	0.8605 (0.0509)	**0.8701 (0.0605)**	**0.8670 (0.0574)**
	0.35	0.7928	0.8579 (0.0651)	0.8517 (0.0589)	**0.8730 (0.0802)**	**0.8669 (0.0741)**
	0.4	0.7436	0.8342 (0.0906)	**0.8472 (0.1036)**	**0.8465 (0.1029)**	0.8329 (0.0893)
Average		0.8428	0.8758 (0.0330)	**0.8831 (0.0403)**	**0.8799 (0.0371)**	0.8776 (0.0348)
LR	0.05	0.8814	0.8734 (-0.0080)	0.8828 (0.0014)	**0.8997 (0.0183)**	**0.8830 (0.0016)**
	0.1	0.8855	0.8851 (-0.0004)	**0.8897 (0.0042)**	**0.8927 (0.0072)**	0.8719 (-0.0136)
	0.15	0.8641	0.8712 (0.0071)	**0.8761 (0.0120)**	**0.8854 (0.0213)**	0.8646 (0.0005)
	0.2	0.8554	**0.8678 (0.0124)**	0.8661 (0.0107)	**0.8763 (0.0209)**	0.8658 (0.0104)
	0.25	0.8547	**0.8699 (0.0152)**	0.8688 (0.0141)	**0.8791 (0.0244)**	0.8565 (0.0018)
	0.3	0.8426	**0.8654 (0.0228)**	0.8600 (0.0174)	**0.8732 (0.0306)**	0.8430 (0.0004)
	0.35	0.8421	**0.8642 (0.0221)**	0.8478 (0.0057)	**0.8520 (0.0099)**	0.8352 (-0.0069)
	0.4	0.8248	**0.8720 (0.0472)**	0.8357 (0.0109)	**0.8551 (0.0303)**	0.8192 (-0.0056)
Average		0.8563	**0.8711 (0.0148)**	0.8659 (0.0096)	**0.8767 (0.0204)**	0.8549 (-0.0014)
SVM	0.05	**0.9121**	0.8748 (-0.0373)	0.9040 (-0.0081)	**0.9196 (0.0075)**	0.9021 (-0.0100)
	0.1	**0.9035**	0.8679 (-0.0356)	0.8993 (-0.0042)	**0.9041 (0.0006)**	0.8859 (-0.0176)
	0.15	0.8979	0.8680 (-0.0299)	**0.8985 (0.0006)**	**0.9059 (0.0080)**	0.8863 (-0.0116)
	0.2	**0.8950**	0.8777 (-0.0173)	0.8934 (-0.0016)	**0.8957 (0.0007)**	0.8806 (-0.0144)
	0.25	**0.8790**	0.8668 (-0.0122)	**0.8917 (0.0127)**	0.8728 (-0.0062)	0.8750 (-0.0040)
	0.3	0.8775	0.8705 (-0.0070)	**0.8932 (0.0157)**	**0.8945 (0.0170)**	0.8661 (-0.0114)
	0.35	0.8581	0.8552 (-0.0029)	**0.8746 (0.0165)**	**0.8903 (0.0322)**	0.8366 (-0.0215)
	0.4	0.8440	**0.8734 (0.0294)**	0.8628 (0.0188)	**0.8719 (0.0279)**	0.8377 (-0.0063)
Average		0.8834	0.8693 (-0.0141)	**0.8897 (0.0063)**	**0.8943 (0.0109)**	0.8713 (-0.0121)
xgboost	0.05	**0.9126**	0.9017 (-0.0109)	0.9122 (-0.0004)	**0.9227 (0.0101)**	0.9125 (-0.0001)
	0.1	0.8918	0.8865 (-0.0053)	**0.9039 (0.0121)**	0.8937 (0.0019)	**0.8996 (0.0078)**
	0.15	0.8734	0.8829 (0.0095)	**0.8948 (0.0214)**	0.8915 (0.0181)	**0.9012 (0.0278)**
	0.2	0.8406	0.8622 (0.0216)	**0.8852 (0.0446)**	0.8810 (0.0404)	**0.8953 (0.0547)**
	0.25	0.8093	0.8594 (0.0501)	0.8714 (0.0621)	**0.8873 (0.0780)**	**0.8731 (0.0638)**
	0.3	0.7757	0.8507 (0.0750)	0.8595 (0.0838)	**0.8644 (0.0887)**	**0.8598 (0.0841)**
	0.35	0.7567	0.8375 (0.0808)	0.8515 (0.0948)	**0.8596 (0.1029)**	**0.8557 (0.0990)**
	0.4	0.7025	0.8271 (0.1246)	**0.8388 (0.1363)**	0.7959 (0.0934)	**0.8370 (0.1345)**
Average		0.8203	0.8635 (0.0432)	**0.8772 (0.0569)**	0.8745 (0.0542)	**0.8793 (0.0590)**
All_Average		0.8184	0.8347 (0.0163)	**0.8498 (0.0314)**	**0.8524 (0.0340)**	0.8374 (0.0190)

Table III also reflects that the sensitivity to label noise changes based on classifier. As γ ranges from 5% to 40%, the average accuracy decreases from 84.74% to 54.55%. The greatest overall decrement, 30.19%, is observed for the DT classifier, indicating that DT is the most sensitive to label noise. Even a small amount of label noise in the data set will have a disproportionately severe impact on classification performance. This implies that DT retains a large margin of improvement for predictive performance in the presence of label noise. mRD attains 25.56% average improvement and Adp_mCRF attains 25.25% average improvement over unmodified DT, the highest observed. In contrast, the unmodified classifiers LR and SVM show some resistance to label noise.

TABLE IV
COMPARISON OF AVERAGE $macro{-}F1$ ON IMBALANCED DATA SETS AT VARYING NOISE RATES γ USING LR.

Data	γ	Original	IW	mCRF	S-Original	S-IW	S-mCRF
balancescale	0.05	0.6051	0.6051	0.6040	**0.7602 (0.1551)**	**0.7602 (0.1551)**	**0.7549 (0.1509)**
	0.1	0.6088	0.6088	0.6068	**0.7026 (0.0938)**	**0.7026 (0.0938)**	**0.7008 (0.0940)**
	0.15	0.6019	0.6030	0.6030	**0.6970 (0.0951)**	0.6965 (0.0935)	**0.6969 (0.0939)**
	0.2	0.6071	0.6071	0.6071	**0.6779 (0.0708)**	**0.6779 (0.0708)**	**0.6668 (0.0597)**
	0.25	0.6062	0.6062	0.6043	**0.6781 (0.0719)**	**0.6781 (0.0719)**	**0.6703 (0.0660)**
	0.3	0.6106	0.6106	0.6117	**0.6548 (0.0442)**	**0.6548 (0.0442)**	**0.6578 (0.0461)**
	0.35	0.6050	0.6050	0.6054	**0.6731 (0.0681)**	**0.6731 (0.0681)**	**0.6808 (0.0754)**
	0.4	0.5973	0.5973	0.5949	**0.6354 (0.0381)**	**0.6354 (0.0381)**	**0.6183 (0.0234)**
Average		0.6053	0.6054	0.6047	**0.6849 (0.0796)**	**0.6848 (0.0794)**	0.6808 (0.0761)
newthyroid	0.05	0.8869	0.8913	**0.9228**	0.8929 (0.0060)	0.8935 (0.0022)	**0.9381 (0.0153)**
	0.1	0.8622	0.8672	0.8574	**0.8826 (0.0204)**	0.8761 (0.0089)	**0.9524 (0.0950)**
	0.15	0.8780	0.8939	**0.9055**	0.8220 (-0.0560)	0.8377 (-0.0562)	**0.9341 (0.0286)**
	0.2	0.8413	0.8757	**0.8799**	0.7940 (-0.0473)	0.8045 (-0.0712)	**0.9298 (0.0499)**
	0.25	0.8020	**0.8472**	**0.8364**	0.7524 (-0.0496)	0.7718 (-0.0754)	0.8359 (-0.0005)
	0.3	0.7848	**0.8131**	0.7827	0.7573 (-0.0275)	0.7917 (-0.0214)	**0.8299 (0.0472)**
	0.35	0.7756	0.8245	**0.8397**	0.7510 (-0.0246)	0.7512 (-0.0733)	**0.8505 (0.0108)**
	0.4	0.7605	0.7789	0.7466	**0.7822 (0.0217)**	0.7427 (-0.0362)	**0.8366 (0.0900)**
Average		0.8239	**0.8490**	0.8464	0.8043 (-0.0196)	0.8086 (-0.0404)	**0.8884 (0.0420)**
pendigits	0.05	0.9371	0.9245	**0.9439**	0.9370 (-0.0001)	0.9250 (0.0005)	**0.9443 (0.0004)**
	0.1	0.9292	0.9219	**0.9426**	0.9296 (0.0004)	0.9142 (-0.0077)	**0.9436 (0.0010)**
	0.15	0.9213	0.9139	**0.9405**	0.9227 (0.0014)	0.9096 (-0.0043)	**0.9424 (0.0019)**
	0.2	0.9118	0.9007	**0.9388**	**0.9125 (0.0007)**	0.9039 (0.0032)	**0.9388 (0)**
	0.25	0.9106	0.8968	**0.9362**	0.9105 (-0.0001)	0.9005 (0.0037)	**0.9347 (-0.0015)**
	0.3	0.9071	0.8934	**0.9346**	0.9076 (0.0005)	0.8950 (0.0016)	**0.9334 (-0.0012)**
	0.35	0.9061	0.8898	**0.9276**	0.9070 (0.0009)	0.8889 (-0.0009)	**0.9306 (0.0030)**
	0.4	0.9005	0.8763	**0.9330**	0.9009 (0.0004)	0.8788 (0.0025)	**0.9307 (-0.0023)**
Average		0.9155	0.9022	**0.9371**	0.9160 (0.0005)	0.9020 (-0.0002)	**0.9373 (0.0002)**
PhishingData	0.05	0.6027	0.6027	0.5752	**0.6573 (0.0546)**	**0.6573 (0.0546)**	**0.6569 (0.0817)**
	0.1	0.5958	0.5958	0.5740	**0.6422 (0.0464)**	**0.6422 (0.0464)**	**0.6332 (0.0592)**
	0.15	0.5937	0.5937	0.5708	**0.6424 (0.0487)**	**0.6424 (0.0487)**	**0.6260 (0.0552)**
	0.2	0.5931	0.5942	0.5706	**0.6426 (0.0495)**	**0.6426 (0.0484)**	**0.6401 (0.0695)**
	0.25	0.5763	0.5763	0.5692	**0.6248 (0.0485)**	**0.6248 (0.0485)**	**0.6185 (0.0493)**
	0.3	0.5749	0.5749	0.5756	**0.5979 (0.0230)**	**0.5979 (0.0230)**	**0.5924 (0.0168)**
	0.35	0.5774	0.5774	0.5724	**0.6027 (0.0253)**	0.6022 (0.0248)	**0.6094 (0.0370)**
	0.4	0.5867	0.5867	0.5709	**0.6218 (0.0351)**	**0.6218 (0.0351)**	**0.6197 (0.0488)**
Average		0.5876	0.5877	0.5723	**0.6290 (0.0414)**	**0.6289 (0.0412)**	0.6245 (0.0522)
satimage	0.05	0.7696	0.7711	0.8072	0.8156 (0.0460)	**0.8160 (0.0449)**	**0.8386 (0.0314)**
	0.1	0.7592	0.7620	0.8112	0.8149 (0.0557)	**0.8156 (0.0536)**	**0.8316 (0.0204)**
	0.15	0.7500	0.7510	0.8004	0.8069 (0.0569)	**0.8070 (0.0560)**	**0.8320 (0.0316)**
	0.2	0.7322	0.7337	0.7999	**0.8016 (0.0694)**	0.8015 (0.0678)	**0.8329 (0.0330)**
	0.25	0.7103	0.7110	**0.7962**	0.7799 (0.0696)	0.7812 (0.0702)	**0.8295 (0.0333)**
	0.3	0.7209	0.7224	**0.7852**	0.7738 (0.0529)	0.7742 (0.0518)	**0.8291 (0.0439)**
	0.35	0.6982	0.6996	**0.7942**	0.7581 (0.0599)	0.7590 (0.0594)	**0.8271 (0.0329)**
	0.4	0.6899	0.6911	**0.7802**	0.7342 (0.0443)	0.7344 (0.0433)	**0.8249 (0.0447)**
Average		0.7288	0.7302	**0.7968**	0.7856 (0.0568)	0.7861 (0.0559)	**0.8307 (0.0339)**
shuttle	0.05	**0.5353**	0.0286	**0.5857**	0.4779 (-0.0574)	0.0012 (-0.0274)	0.4940 (-0.0917)
	0.1	**0.5565**	0.1121	**0.5869**	0.4384 (-0.1181)	0.0870 (-0.0251)	0.5008 (-0.0861)
	0.15	**0.5252**	0.1242	**0.5499**	0.3968 (-0.1284)	0.0825 (-0.0417)	0.5031 (-0.0468)
	0.2	**0.5153**	0.0811	**0.5809**	0.3780 (-0.1373)	0.0811 (0)	0.5076 (-0.0733)
	0.25	**0.5317**	0.1512	**0.5504**	0.3732 (-0.1585)	0.1512 (0)	0.5061 (-0.0443)
	0.3	**0.5083**	0.0712	**0.5521**	0.3469 (-0.1614)	0.0712 (0)	0.5037 (-0.0484)
	0.35	0.5059	0.0942	**0.5960**	0.3470 (-0.1589)	0.1497 (0.0555)	**0.5119 (-0.0841)**
	0.4	0.4870	0.1758	**0.5276**	0.3393 (-0.1477)	0.1758 (0)	**0.5038 (-0.0238)**
Average		**0.5207**	0.1048	**0.5662**	0.3872 (-0.1335)	0.1000 (-0.0048)	0.5039 (-0.0623)
userknowledge	0.05	0.5206	0.5206	0.5148	**0.6478 (0.1272)**	**0.6478 (0.1272)**	**0.6441 (0.1293)**
	0.1	0.5055	0.5055	0.5062	**0.6412 (0.1357)**	**0.6412 (0.1357)**	**0.6497 (0.1435)**
	0.15	0.5051	0.5051	0.5018	**0.5292 (0.0241)**	**0.5292 (0.0241)**	**0.5681 (0.0663)**
	0.2	**0.5172**	**0.5172**	0.5047	0.5035 (-0.0137)	0.5035 (-0.0137)	**0.5343 (0.0296)**
	0.25	0.4797	0.4797	0.4892	**0.5205 (0.0408)**	**0.5205 (0.0408)**	**0.5663 (0.0771)**
	0.3	0.4684	0.4684	**0.4896**	0.4582 (-0.0102)	0.4582 (-0.0102)	**0.4854 (-0.0042)**
	0.35	0.4800	0.4800	**0.4805**	0.4631 (-0.0169)	0.4631 (-0.0169)	**0.4991 (0.0186)**
	0.4	0.4445	0.4445	**0.4741**	0.4113 (-0.0332)	0.4113 (-0.0332)	**0.5097 (0.0356)**
Average		0.4901	0.4901	0.4951	**0.5218 (0.0317)**	**0.5218 (0.0317)**	**0.5571 (0.0620)**
vertebralColumn	0.05	0.8019	**0.8115**	0.7770	0.8000 (-0.0019)	0.7987 (-0.0128)	**0.8136 (0.0366)**
	0.1	0.7957	**0.8149**	0.7641	0.7945 (-0.0012)	**0.8028 (-0.0121)**	0.7827 (0.0186)
	0.15	**0.8140**	**0.8142**	0.7853	0.7794 (-0.0346)	0.7315 (-0.0827)	0.7920 (0.0067)
	0.2	0.7611	**0.7730**	**0.7787**	0.7533 (-0.0078)	0.7643 (-0.0087)	0.7556 (-0.0231)
	0.25	0.7675	0.7573	**0.7736**	0.7516 (-0.0159)	0.7568 (-0.0005)	**0.8346 (0.0610)**
	0.3	**0.7991**	**0.7946**	0.7644	0.7548 (-0.0443)	0.7524 (-0.0422)	0.7386 (-0.0258)
	0.35	0.7244	**0.7294**	0.6703	0.7176 (-0.0068)	0.7281 (-0.0013)	**0.7292 (0.0589)**
	0.4	0.6360	0.6359	**0.6659**	0.6328 (-0.0032)	0.6484 (0.0125)	**0.6776 (0.0117)**
Average		0.7625	**0.7663**	0.7474	0.7480 (-0.0145)	0.7479 (-0.0184)	**0.7655 (0.0181)**
wine	0.05	0.9253	0.9253	0.9092	**0.9298 (0.0045)**	0.9238 (-0.0015)	**0.9338 (0.0246)**
	0.1	**0.9341**	**0.9350**	0.9308	0.9285 (-0.0056)	0.9292 (-0.0058)	0.9205 (-0.0103)
	0.15	0.9250	0.9382	**0.9644**	0.9292 (0.0042)	0.9381 (-0.0001)	**0.9554 (-0.0090)**
	0.2	**0.9220**	**0.9339**	0.9202	0.9010 (-0.0210)	0.9210 (-0.0129)	0.9177 (-0.0025)
	0.25	0.9133	0.9130	0.9133	**0.9233 (0.0100)**	0.9221 (0.0091)	**0.9351 (0.0218)**
	0.3	0.8958	0.9013	**0.9267**	0.8977 (0.0019)	0.9185 (0.0172)	**0.9243 (-0.0024)**
	0.35	**0.9564**	0.9239	**0.9447**	0.9297 (-0.0267)	0.9235 (-0.0004)	0.9215 (-0.0232)
	0.4	**0.9140**	**0.8922**	0.8463	0.8826 (-0.0314)	0.8811 (-0.0111)	0.8825 (0.0362)
Average		**0.9232**	0.9203	0.9195	0.9152 (-0.0080)	0.9197 (-0.0006)	**0.9238 (0.0043)**
All_Average		0.7064	0.6618	0.7206	**0.7102 (0.0038)**	0.6778 (0.0160)	**0.7458 (0.0252)**

As the noise rate increases from 0.05 to 0.4, the average accuracy of LR decreases from 88.14% to 82.48%, with a gap of only 5.66%. The average accuracy of SVM decreases from 91.21% to 84.40% with a gap of 6.81%. After Adp_mCRF is applied, these two classifiers achieve a maximum improvement of 2.04% and 1.09% respectively. LightGBM and Xgboost are excellent ensemble algorithms and achieve the highest two average accuracies at 91.64% and 91.26% when $\gamma = 0.05$. However, when γ increases to 0.4, the performances of these two classifiers drop sharply to 74.36% and 70.25%. As seen in the Table III, our proposed noise-filtering learning framework provides strong support to these two algorithms, enabling these ensemble classifiers to more capably handle a large amount of noisy data.

The results in Tables II and III suggest classifier performance on every data set is easily disturbed by the presence of label noise. Our proposed noise-filtering learning frameworks are demonstrably effective in mitigating the negative influence of label noise. Furthermore, the ensemble mechanism incorporated into mCRF to stably and precisely detect label-noise samples is empirically verified.

C. Imbalanced UCI Data sets Experiments

Class imbalance is ubiquitous and unavoidable in multiclass classification. Synthetic Minority Oversampling Technique (SMOTE) is the most prevalent method of dealing with imbalanced data sets [49]. However, the original SMOTE is susceptible to noise because it blindly interpolates among k nearest neighbors, without distinguishing overlapping class regions from so-called safe areas [50]. The influence of label noise can be effectively mitigated by applying noise filters along with SMOTE [23], [51]. Therefore, we incorporate SMOTE into our proposed filtering methods to address the class imbalance problems that arise in multiclass classification with label noise. Specifically, we use the performance metrics $macro{-}F1$ (that is, the average of the F1 measures calculated for each class [52]) and the accuracy rate of correctly identified minority samples, instead of precision, to evaluate the performances of the proposed methods against our baseline, the state-of-the-art method in imbalanced learning. Because the precision metric in traditional classification is biased in favor of the majority class, it does not accurately capture the details of our multiclass classification results in imbalanced problems. Nine imbalanced data sets, marked by $*$ in Table I, are used. For space, only LR is selected as a baseline classifier, due to its high efficiency and effectiveness in classifying multiclass data sets, and VCV_mCRF is selected as the representative method of our frameworks.

We report the $macro{-}F1$ scores of this experimental study in Table IV. Results of mRD are not included in Table IV, as the combination of SMOTE with the mRD filtering learning framework performs poorly in imbalanced classification. This is a result of the hard threshold 1; this threshold easily fails and leads to many minority examples being identified as label noise [13], resulting in under-fitting during classification. Results of the original classifier (Original), mIW, mCRF, and SMOTE-Original(S-Original), SMOTE-mIW(S-mIW), SMOTE-mCRF(S-mCRF) are provided in Table IV, with the highest value in bold in each row. The enhancement on $macro{-}F1$ score for each pair is given in parentheses.

We observe that the combination of SMOTE with different algorithms exhibits varying levels of improvement on different data sets. The average improvements across different noise rates γ for the original LR, mIW-LR, and mCRF-LR are 0.38%, 1.60%, and 2.52%, respectively. This indicates that

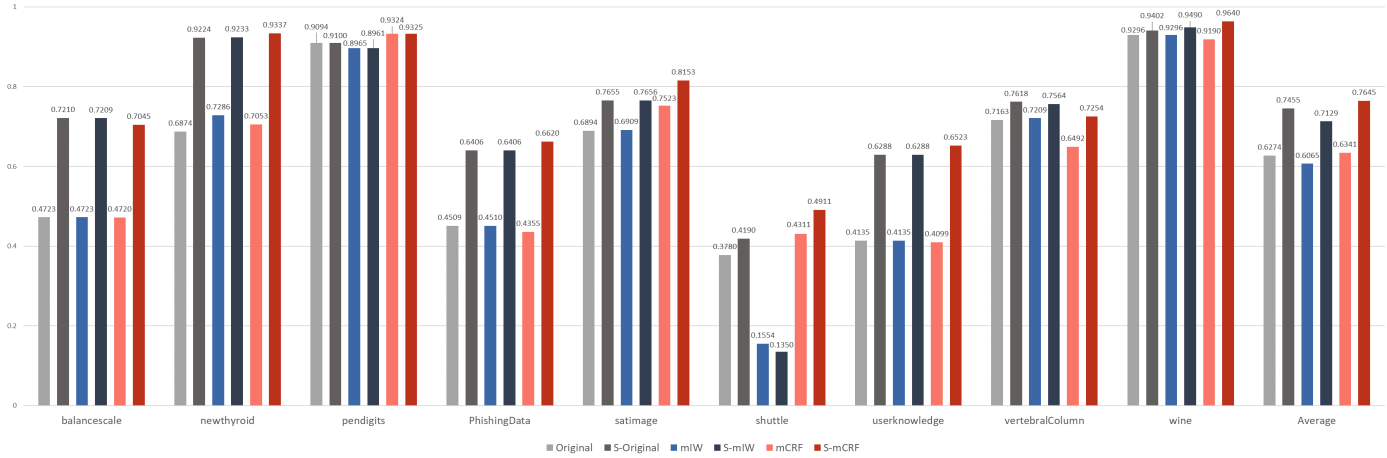


Fig. 9. Comparison of Average Accuracies of Minority Samples, Before and After Applying SMOTE, across Varying Noise Rates γ.

applying SMOTE to a label noise-filtering learning framework to handle imbalanced data will generally lead to improvement. The incorporation of SMOTE shows a high level of improvement to these three algorithms on the data sets balancescale, PhishingData, and satimage. The average improvement across noise rates on these three data sets is 5.93%, 5.88%, and 5.41%, respectively. The LR classifier and mIW-LR both obtain the highest improvement from the application of SMOTE at $\gamma = 0.05$ on the balancescale data set, achieving 15.51% improvement over the unmodified versions. After applying SMOTE on newthyroid and userknowledge, the $macro-F1$ scores of the original LR and mIW-LR increase when the noise rate is at a low level (γ = 0.05 or 0.1), but decrease when the noise rate is at a high level ($\gamma \geq 0.20$). This indicates that, on these two data sets, when γ increases to a high level, noisy samples in the imbalanced data sets will hinder the search for the best hyperplane. In contrast, noise-filtering mCRF-LR demonstrates excellent performance with average improvements of 4.20% and 6.20% on the two data sets. For data sets pendigits and wine, the $macro-F1$ scores change only a little after oversampling the imbalanced data sets. This is a result of the low imbalance ratios of the two data sets producing little difference in the data distribution as well as the classifying results. mIW-LR performs best on the data set vertebralColumn from an average perspective, but for the data set shuttle, which has an extreme imbalance ratio, the $macro-F1$ scores of mIW-LR drop down to about 10%. This is because the mIW method resorts to KLIEP to estimate the conditional probability $P_{D_\gamma}(Y|X)$ [43]. However, in an extremely imbalanced data set, the minority samples can not provide enough information to accurately estimate the probability $P_{D_\gamma}(Y|X)$, resulting in the inaccurate estimation of the reweighing weights for each sample. For data set shuttle, the mCRF-LR algorithm performs best.

Figure 9 visualizes the average accuracies of the minority samples both before and after applying SMOTE to the imbalanced data sets. The tree-pair methods, that is, LR and SMOTE-LR, mIW-LR and SMOTE-mIW-LR, and mCRF-LR and SMOTE-mCRF-LR are colored in grey, blue, and red, respectively. The lighter color in each pair corresponds to the method without applying SMOTE. We observe in Figure 9 that applying SMOTE to imbalanced data sets will improve the accuracy of the minority samples, with average improvements of 11.81%, 10.63%, and 13.04% for each algorithm, respectively. A large improvement is seen on data sets balancescale, newthyroid, PhishingData, satimage, userknowledge, and vertebralColumn. These data sets are moderately imbalanced. The difference before and after applying SMOTE to data sets pendigits and wine is small, due to low imbalance ratios. For shuttle, mCRF-LR demonstrates better performance. As classification accuracy is not the main evaluation index, detailed accuracy comparisons of the minority samples are reported in Table LV.

In summary, the combination of SMOTE with our mCRF filtering learning framework will bring some improvement in multiclass classification with label noise, but for extremely imbalanced data sets with label noise, SMOTE is not the best choice. Therefore, our future efforts will concentrate on developing a new data cleaning algorithm that will be effective on extremely imbalanced data sets.

VII. Conclusion

We have presented a general label noise filtering learning framework for multiclass classification, and developed two extensions, mCRF and mRD, of binary label-noise filtering methods to multiclass classification using that framework. We also propose the voting cross-validation method and an adaptive method for optimizing the *NI_threshold* parameter, resulting in a more stable label noise detection process. In addition, we incorporate the popular Synthetic Minority Oversampling Technique (SMOTE) with our proposed label-noise filtering learning framework to deal with imbalanced noisy data. Experiments on both synthetic and real data sets verify the effectiveness of our proposed methods. Specifically, our mCRF noise filtering learning framework has stable performance with high accuracy, even on imbalanced data sets. Our mRD noise filtering learning framework performs well on small to mid-sized balanced data sets, but becomes less useful on imbalanced data sets due to the hard threshold set at 1 causing an excessive number of minority samples to be misclassified as label noise. Experimental results demonstrate that the VCV method performs better than the adaptive method. The combination of SMOTE with the mCRF noise filtering

learning framework has benefits on moderately imbalanced data sets, but for extremely imbalanced data sets SMOTE is not the best choice. Our results encourage us to continue future work on this concept. Research going forward will concentrate on developing a more effective filtering method by utilizing ensemble mechanisms to detect noisy samples in highly imbalanced data sets.

REFERENCES

[1] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[2] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.

[3] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of Artificial Intelligence Research*, vol. 11, pp. 131–167, 1999.

[4] J. A. SáEz, M. Galar, J. Luengo, and F. Herrera, "Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness," *Information Sciences*, vol. 247, pp. 1–20, 2013.

[5] L. P. Garcia, A. C. de Carvalho, and A. C. Lorena, "Effect of label noise in the complexity of classification problems," *Neurocomputing*, vol. 160, pp. 108–119, 2015.

[6] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[7] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 6, pp. 448–452, 1976.

[8] X. Zhu, X. Wu, and Q. Chen, "Eliminating class noise in large datasets," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 920–927.

[9] J.-w. Sun, F.-y. Zhao, C.-j. Wang, and S.-f. Chen, "Identifying and correcting mislabeled training instances," in *Future Generation Communication and Networking (FGCN 2007)*, vol. 1. IEEE, 2007, pp. 244–250.

[10] C. G. Northcutt, T. Wu, and I. L. Chuang, "Learning with confident examples: Rank pruning for robust classification with noisy labels," *ArXiv Preprint ArXiv:1705.01936*, 2017.

[11] A. Angelova, Y. Abu-Mostafam, and P. Perona, "Pruning training sets for learning of object categories," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 494–501.

[12] S. Xia, Z. Xiong, Y. Luo, L. Dong, and C. Xing, "Relative density based support vector machine," *Neurocomputing*, vol. 149, pp. 1424–1432, 2015.

[13] Y. Liu, S. Xia, H. Yu, Y. Luo, B. Chen, K. Liu, and G. Wang, "Prediction of aluminum electrolysis superheat based on improved relative density noise filter smo," in *2018 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2018, pp. 376–381.

[14] X. Liang, S. Xia, Q. Liu, Y. Liu, B. Chen, and G. Wang, "A multi-granular relative density model for class noise detection," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2651–2656.

[15] S. Xia, G. Wang, Z. Chen, Y. Duan, and Q. Liu, "Complete random forest based class noise filtering learning for improving the generalizability of classifiers," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[16] A. Karmaker and S. Kwek, "A boosting approach to remove class label noise 1," *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 3, pp. 169–177, 2006.

[17] A. Malossini, E. Blanzieri, and R. T. Ng, "Detecting potential labeling errors in microarrays by data perturbation," *Bioinformatics*, vol. 22, no. 17, pp. 2114–2121, 2006.

[18] B. Sluban, D. Gamberger, and N. Lavrač, "Ensemble-based noise detection: noise ranking and visual performance evaluation," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 265–303, 2014.

[19] L. P. F. Garcia, A. C. Lorena, and A. C. Carvalho, "A study on class noise detection and elimination," in *2012 Brazilian Symposium on Neural Networks*. IEEE, 2012, pp. 13–18.

[20] S. Verbaeten and A. Van Assche, "Ensemble methods for noise elimination in classification problems," in *International Workshop on Multiple Classifier Systems*. Springer, 2003, pp. 317–325.

[21] B. Sluban, D. Gamberger, and N. Lavra, "Advances in class noise detection," in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. IOS Press, 2010, pp. 1105–1106.

[22] J. Van Hulse and T. Khoshgoftaar, "Knowledge discovery from imbalanced and noisy data," *Data & Knowledge Engineering*, vol. 68, no. 12, pp. 1513–1542, 2009.

[23] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences*, vol. 291, pp. 184–203, 2015.

[24] J. F. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, and L. I. Kuncheva, "Random balance: ensembles of variable priors classifiers for imbalanced data," *Knowledge-Based Systems*, vol. 85, pp. 96–111, 2015.

[25] X. Wu and X. Zhu, "Mining with noise knowledge: error-aware data mining," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 4, pp. 917–932, 2008.

[26] J. A. SáEz, J. Luengo, and F. Herrera, "Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification," *Pattern Recognition*, vol. 46, no. 1, pp. 355–364, 2013.

[27] P. M. Long and R. A. Servedio, "Random classification noise defeats all convex potential boosters," *Machine Learning*, vol. 78, no. 3, pp. 287–304, 2010.

[28] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems*, 2013, pp. 1196–1204.

[29] C. Scott, "A rate of convergence for mixture proportion estimation, with application to learning from noisy labels," in *Artificial Intelligence and Statistics*, 2015, pp. 838–846.

[30] X. Yu, T. Liu, M. Gong, K. Batmanghelich, and D. Tao, "An efficient and provable approach for mixture proportion estimation using linear independence assumption," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4480–4489.

[31] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[32] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Advances in Neural Information Processing Systems*, 2015, pp. 10–18.

[33] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2016.

[34] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, "Loss factorization, weakly supervised learning and label noise robustness," in *International Conference on Machine Learning*, 2016, pp. 708–717.

[35] D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont, "Robust multi-class gaussian process classification," in *Advances in Neural Information Processing Systems*, 2011, pp. 280–288.

[36] J. Bootkrajang and A. Kabán, "Multi-class classification in the presence of labelling errors." in *ESANN*. Citeseer, 2011, pp. 345–350.

[37] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," *ArXiv Preprint ArXiv:1406.2080*, 2014.

[38] J. A. Sáez, M. Galar, J. Luengo, and F. Herrera, "Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition," *Knowledge and Information Systems*, vol. 38, no. 1, pp. 179–206, 2014.

[39] L. P. Garcia, J. A. Sáez, J. Luengo, A. C. Lorena, A. C. de Carvalho, and F. Herrera, "Using the one-vs-one decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems," *Knowledge-Based Systems*, vol. 90, pp. 153–164, 2015.

[40] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Drcw-ovo: distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems," *Pattern Recognition*, vol. 48, no. 1, pp. 28–42, 2015.

[41] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers," *Pattern Recognition*, vol. 46, no. 12, pp. 3412–3424, 2013.

[42] B. Sun, S. Chen, J. Wang, and H. Chen, "A robust multi-class adaboost algorithm for mislabeled noisy data," *Knowledge-Based Systems*, vol. 102, pp. 87–102, 2016.

[43] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2568–2580, 2018.

[44] L. Liu and Q. Liang, "A high-performing comprehensive learning algorithm for text classification without pre-labeled training set," *Knowledge and Information Systems*, vol. 29, no. 3, pp. 727–738, 2011.
[45] P. Melin, J. Amezcua, F. Valdez, and O. Castillo, "A new neural network model based on the lvq algorithm for multi-class classification of arrhythmias," *Information Sciences*, vol. 279, pp. 483–497, 2014.
[46] G. Kou, Y. Peng, Z. Chen, and Y. Shi, "Multiple criteria mathematical programming for multi-class classification and application in network intrusion detection," *Information Sciences*, vol. 179, no. 4, pp. 371–381, 2009.
[47] R. Y. Toledo, Y. C. Mota, and L. Martínez, "Correcting noisy ratings in collaborative recommender systems," *Knowledge-Based Systems*, vol. 76, pp. 96–108, 2015.
[48] B. Krawczyk, J. A. Sáez, and M. Wozniak, "Tackling label noise with multi-class decomposition using fuzzy one-class support vector machines," pp. 915–922, 2016.
[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "S-mote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.
[50] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," pp. 475–482, 2009.
[51] W. A. Rivera, "Noise reduction a priori synthetic over-sampling for class imbalanced data sets," *Information Sciences*, vol. 408, pp. 146–161, 2017.
[52] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

Shuyin Xia* received his B.S. degree in 2008 and his M.S. degree in 2012, both in computer science and both from the Chongqing University of Technology in China. He received his Ph.D. degree from the College of Computer Science at Chongqing University in China. Since 2015, he has been with the Chongqing University of Posts and Telecommunications in Chongqing, China, where he is currently an associate professor and a Ph.D. supervisor. His research interests include data mining, granular computing, classifiers and label noise detection. He has published more than 30 papers in prestigious journals and conferences, such as IEEE T-PAMI, IEEE T-KDE, T-CYB and IS. He is also the executive deputy director of CQUPT—Chongqing Municipal Public Security Bureau—Qihoo 360 Big Data and Network Security Joint Lab, the director of the Chongqing Artificial Intelligence Association, and an IEEE Member.

Baiyun Chen* received her B.S. degree in information management and systems in 2012 and M.S. degree in management science and engineering in 2015, both from the Zhongnan University of Economics and Law in Wuhan, China. She is currently pursuing a Ph.D. degree in computer science and technology at the Chongqing University of Posts and Telecommunications in Chongqing, China. Her research interests include classifiers, label noise detection, and granular computing, and she is an IEEE member.

Guoyin Wang (M'98SM'03) received a B.E. degree in computer software in 1992, a M.S. degree in computer software in 1994, and a Ph.D. degree in computer organization and architecture in 1996, all from Xi'an Jiaotong University in Xi'an, China. His research interests include data mining, machine learning, rough sets, granular computing, cognitive computing, and so forth. He has been working at the Chongqing University of Posts and Telecommunications in Chongqing, China, where he is currently a Professor and a Ph.D. supervisor, the Director of the Chongqing Key Laboratory of Computational Intelligence, and the Dean of the Graduate School.

Xinbo Gao (M02-SM07) received BEng, MSc, and PhD degrees in signal and information processing from Xidian University in Xian, China, in 1994, 1997, and 1999. From 1997 to 1998, he was a research fellow with the Department of Computer Science at Shizuoka University in Shizuoka, Japan. From 2000 to 2001, he was a post-doctoral research fellow with the Department of Information Engineering at the Chinese University of Hong Kong in Hong Kong. From 2001 to 2020, he was with the School of Electronic Engineering at Xidian University. He is currently the President of the Chongqing University of Posts and Telecommunications. He has published six books and over 200 technical articles in prestigious journals and conferences, including IEEE T-PAMI, T-IP, T-NNLS, T-MI, NIPS, CVPR, ICCV, AAAI and IJCAI.

Elisabeth Giem received two bachelor's degrees, one in pure mathematics and one in music (concentration in performance) from the University of California, Riverside (UCR). She received a master's degree in computational and applied mathematics from Rice University, and a master's degree in pure mathematics from UCR. She joined Zizhong Chen's SuperLab in 2018 as a computer science Ph.D. student. Her research interests include but are not limited to high-performance computing, parallel and distributed systems, big data analytics, computational entomology, and numerical linear algebra algorithms and software.