A Fast Class Noise Detector with Multi-factor-based Learning

Wanwan Zheng(✉) and Mingzhe Jin

Doshisha University, Kyoto, Japan
cyac1004@mail4.doshisha.ac.jp

Abstract. Noise detection algorithms commonly face the problems of over-cleansing, large computational complexity and long response time. Preserving the original structure of data is critical for any classifier, whereas over-cleansing will adversely affect the quality of data. Besides, the high time complexity remains one of the main defects for most noise detectors, especially those exhibiting an ensemble structure. Moreover, numerous studies reported that ensemble-based techniques outperform other techniques in the accuracy of noisy instances identification. In the present study, a fast class noise detector called FMF (fast class noise detector with multi-factor-based learning) was proposed. FMF, three existing ensemble-based noise detectors and the case with original data were compared on 10 classification tasks. C5.0 acted as the classifiers to access the efficiency of these algorithms. As revealed from the results, the FMF shortened the processing time at least twelve times by three baselines, and it achieved the highest accuracy in most cases.

Keywords: Noise detection · Over-cleansing · High time complexity · Multi-factor-based learning · Fast execution

1 Introduction

Noise detection [1] is a preprocessing technique that can be employed in any given dataset to identify potentially noisy instances. In practice, uncertain or contaminated training sets are commonly conducted. Besides, some studies estimated that even in controlled environments at least 5% of errors exist in a dataset [2,3].

It is suggested that some objects can be impacted by feature or class noise, which offer misleading information and then hinder the learning process of classifiers. Moreover, class noise is potentially more harmful than feature noise, and the reasons are presented below. First, there are numerous features, whereas there is only one label. Second, each feature for learning has different importance, whereas labels always significantly impact learning [4]. Third, the consequences of class noise detection will impact many feature noise detection techniques (e.g. feature selection) directly. Thus, detecting class noise prior to the analysis of

S. Chellappan et al. (Eds.): CSoNet 2020, LNCS 12575, pp. 14–25, 2020.
https://doi.org/10.1007/978-3-030-66046-8_2

polluted data appears to be necessary. In this study, instances that should be misclassified are defined as class noise.

In noise detection field, over-cleansing and the high time complexity are primarily difficult to solve. Preserving the original structure of data is uttermost important for any classifier. Obviously, over-reduction causes information loss, even the original structural failure may occur and then the quality of data would be affected adversely. Moreover, large training data size is crucial for supervised learning tasks. Therefore, though removing noisy instances is considered to be capable of enhancing the accuracy of learning model, the fact is that applying noise detectors reduces the accuracy with great potential, especially for the powerful classifiers. [5] proposed the MCIS (multi-class instance selection) and in the six datasets they used, the accuracies of MCIS are higher than using the entire training data only two times; the enhancements of both are 0.01. [6] proposed MOCHC (multi-objective CHC algorithm for instance selection), which achieves about 50% reduction rates, whereas the accuracies decrease for seven datasets in the applied eight datasets. ElkMSI (evolutionary local k with multi-selection of instance) is superior, as proposed by [7]. In their experiments, a total of 150 datasets were used, and ElkMSI achieves the identical averaged accuracy with the original data.

As described above, in the field of noise detection, one of recurrent problems is the trade-off between the reduction rate and the classification quality. To address this problem, INFFC (an iterative class noise filter based on the fusion of classifiers with noise sensitivity control) was proposed in [8]. INFFC focuses more on the removal of those instances that must be deleted (class noise) instead of the reduction rate. The experiment was based on 25 datasets. ENN (edited nearest neighbor), All k-NN (all k-nearest neighbors), ME (multiedit), CF (classifier filter), EF (ensemble filter), NCNE (nearest centroid neighbor edition), IPF (iterative partitioning filter) seven well-known noise detectors acted as the comparisons. Furthermore, 1-NN and robust classifiers C4.5 and SVM were employed to compute the accuracies. As indicated from the experimental results, INFFC enhances the performance of the other methods as well as the case with the entire training data.

The high time complexity remains one of the main defects for most noise detectors; such problem is scaling up [9]. It is a common way to get better performance at the cost of higher computational complexity. Two representative examples are ensembles and deep neural networks. In machine learning studies, the more powerful algorithms should be capable of learning complex nonlinear relationships. By definition, they are robust to noise, show good balance between high variance and low bias, and meaning predictions vary with the specific data used to train them. This added flexibility and power usually come at the cost of large-scale training data and algorithm which is robust enough to handle the large-scale training data.

Especially in the era of big data, data are increasing in scale and exhibiting critical roles in various fields (e.g., text classification, handwritten recognition and face detection). Furthermore, the quality and size of training data are crucial

for supervised learning tasks. Accordingly, how to build a noise detection algorithm that exhibits high detection performance and fast execution is of notable significance.

This study proposed a fast class noise detector with multi-factor-based learning, which is capable of alleviating the possibility of over-cleansing and performing favorably to three existing ensemble-based techniques and the original data in terms of classification accuracy. Moreover, the fast execution makes it possible to deal with large-scale datasets.

The rest of this article is organized as follows: In Sect. 2, the algorithm is presented to build FMF. In Sect. 3, the characteristics of this method are discussed. In Sect. 4, this method is empirically compared with three existing noise detectors and the original data. Lastly, in Sect. 5, conclusions and future research directions are drawn.

2 Algorithm of FMF

FMF is a three-stage process. Because no method will be efficient for all of data, the first stage determines the proper similarity index for the applied data. The second stage exploits the determined similarity index to make a multi-factor-based learning to yield a noise score for respective instance. In the third stage, a threshold is given to remove the noisy instances. The workflow of FMF is illustrated in Fig. 1.

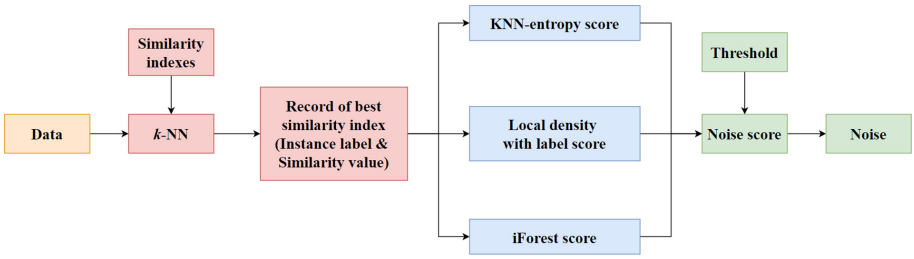

Fig. 1. The workflow of FMF.

2.1 The Determination of the Proper Similarity Index

In this study, KNN was adopted to determine the proper similarity index for its efficiency and fast execution. It learns very quickly because it reads in the training data without much further processing. The computational complexity of KNN is $O(n^2)$, which is significantly smaller than most of the recent methods.

For the similarity indexes, Euclidean distance (Euclidean), Canberra distance (Canberra), Probabilistic symmetric chi-squared measure (Schi), Mahalanobis distance (MD) and Pearson correlation coefficient (Pearson) were selected. To make the selection more accurate, the numeric label was adopted to obtain the

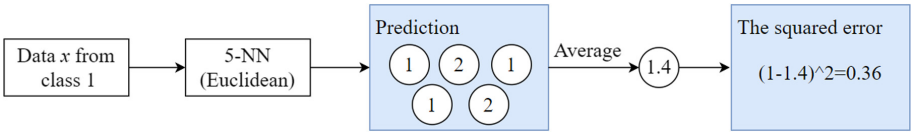

Fig. 2. An example of the squared error computation.

minimum squared error in the case of binary classification. On the other hand, the characteristic label was adopted for multi-class classification. An example of the squared error computation is illustrated in Fig. 2 with Euclidean. The nearest five neighbors of data x with label 1 are 1, 2, 1, 1, 2; thus, the label of data x is predicted as 1.4, and the squared error will be 0.36. Using this method to obtain the prediction squared errors of the rest instances, finally the sum is taken as the squared error of Euclidean. The similarity index that achieves the minimum squared error will be taken as the proper one; its results (for data x_i ($i = 1, 2, 3, \ldots, n$): the labels of the nearest k neighbors, l_j ($j = 1, 2, \ldots, k$) and their similarity values with data x_i, s_j ($j = 1, 2, \ldots, k$)) are transferred to the next stage to process multi-factor-based learning. In this study, k is set to 5.

2.2 Multi-factor-based Learning

For a noise detector, the output can be one of two types, i.e., noise scores and binary labels. This study processed a multi-factor-based learning to output a noise score for respective instance. The output of noise score was taken because noise score retains all of information provided by our algorithm and the possibility of conversion into binary label, besides, it also helps assess the performance of FMF with visualization methods. Furthermore, though classifier is frequently employed in noise detectors, several classifiers are even employed to build an ensemble model, FMF criterion is independent of the classifier(s) in determining the noise score. Thus, the quality of obtained clean data will not be impacted by any classifier.

The noise score s of an instance x is defined as:

$$s(x,k) = weight_k \times ke(x,k) + weight_l \times ldl(x,k) + weight_f \times iForest(x) \quad (1)$$

$$weight_k + weight_l + weight_f = 1 \quad (2)$$

Where k is the number of nearest neighbors. The $ke(x,k)$, $ldl(x,k)$ and $iForest(x)$ represent KNN-entropy score, local density with label score and iForest score, respectively. In this study, $weight_k$, $weight_l$ and $weight_f$ are set to 1/3.

KNN-Entropy Score. The KNN-entropy score (ke) is defined as:

$$ke(x,k) = \begin{cases} 0 & p(x,k) = 0 \\ 1 & p(x,k) = 1 \\ p(x,k) \times (1 - entropy(x,k)) & 0 < p(x,k) < 1 \end{cases} \tag{3}$$

Where $p(x,k)$ is the proportion of the enemies of instance x in the nearest k neighbors. The enemy of x refers to the instance from the other classes. Therefore, the higher $p(x,k)$, the more likely it is noise. The problem is how to determine the value of k. Considering the possibility of data imbalance (in fact, most of real-world data in many domains are reported to be imbalanced), in this study, k is defined as:

$$k = min(I_1, I_2, \ldots, I_n) \tag{4}$$

Where I_n is the number of instances in class n and $n >= 2$. The $entropy(x,k)$ measures the complexity of the k nearest neighbors of x, which takes the same value with:

$$H(X) \triangleq -\sum_{k=1}^{K} p(X=k) \log_e p(X=k) \tag{5}$$

Where X is a random instance label, K is the number of classes, $p(X=k)$ is the proportion of instances from class k.

Therefore, the instances which enter another class completely will get the largest KNN-entropy score, followed by the instances around the decision boundary and normal instances.

A drawback of KNN-entropy score is that it ignores the order of k-nearest neighbors. The local density with label score is designed to alleviate this problem.

Local Density with Label Score. The local density presents the local deviation of density of a given instance with respect to its neighbors. For a given data point x, let $L_k(x)$ be the set of points within the k-nearest neighbor distance of x. Note that $L_k(x)$ will typically contain k points, but may sometimes contains more than k points because of ties in the k-nearest neighbor distance. Then the local density of x is defined as:

$$ld(x,k) = \left(\frac{\sum_{y \in L_k(x)} dist(x,y)}{k} \right)^{-1} \tag{6}$$

Based on the consideration that an outlier should have a substantially lower density than their neighbors, the smaller the local density, the more likely it is outlier. Inspired by the idea of local density, the local density with label score (ldl) is defined as:

$$ldl(x,k) = \begin{cases} 0 & p(x,k) = 0 \\ 1 & p(x,k) = 1 \\ 1 - sigmoid(\frac{ddif(x,h)}{dhit(x,f)}) & 0 < p(x,k) < 1, h + f = k \end{cases} \tag{7}$$

$$dhit(x,h) = \frac{\sum_{y \in H_h(x)} dist(x,y)}{h}, h < k \tag{8}$$

$$ddif(x,f) = \frac{\sum_{y \in F_f(x)} dist(x,y)}{f}, f < k \tag{9}$$

Within the k-nearest neighbor distance of x, $H_h(x)$ is the set of points from the same class with x and the number is h; $F_f(x)$ is the set of points from different class(es) and the number is f. in Eq. (7):

$$\begin{cases} when \quad h = 0 \quad \rightarrow \quad f = k \quad \rightarrow \quad p(x,k) = 1 \quad \rightarrow \quad ldl(x,k) = 1 \\ when \quad h = k \quad \rightarrow \quad f = 0 \quad \rightarrow \quad p(x,k) = 0 \quad \rightarrow \quad ldl(x,k) = 0 \\ when \quad 0 < h < k \quad \rightarrow \quad \frac{ddif(x,k)}{dhit(x,k)} > 0 \quad \rightarrow \quad 0 < ldl(x,k) < 1 \end{cases} \tag{10}$$

iForest Score. iForest utilizes the isolation trees rather than distance or density measures to detect outliers. In an isolation tree, the data is recursively partitioned with axis-parallel cuts at randomly chosen partition points in randomly selected attributes, so as to isolate the instances into nodes with fewer and fewer instances until the points are isolated into singleton nodes containing one instance. In such cases, the tree branches containing outliers are noticeably less deep, because these data points are located in sparse regions. Therefore, the distance of the leaf to the root is used as the outlier score. An iForest is an ensemble combination of a set of isolation trees and the final combination step is performed by averaging the path lengths of the data points in the different trees of the iForest.

Because the iForest is an unsupervised outlier detector, the iForest score is calculated in class unit in this study. The iForest score ranges from 0 to 1, and the higher the score, the higher the potential of being noise.

2.3 Removal of Noisy Instances

As the definition, the final noise score is the average of KNN-entropy score, local density with label score and iForest score, ranging from 0 to 1. However, because iForest score has relatively higher value and lower standard deviation, the $min - max\ normalization$ was taken for three types of score to avoid any one leading the average result.

To detect univariate extreme values, the box-plot (or box and whisker diagram) was used, which is reported to be a particularly useful approach in the context of visualizing noise score. In a box-plot, the statistics of a univariate

distribution are summarized in terms of five quantities, which are the "minimum/maximum" (whisker), the 25th empirical quartile and 75th empirical quartile (box), and the median (line in the middle of box). The distance between the 25th and 75th quartiles is referred to as the $inter-quartile\ range$ (IQR). The "minimum" and "maximum" are enclosed quotations because they are defined in a non-standard way:

$$\text{“}minimum\text{”} = 25th\ empirical\ quartile \quad - \quad range \times IQR \tag{11}$$

$$\text{“}maximum\text{”} = 75th\ empirical\ quartile \quad + \quad range \times IQR \tag{12}$$

In this study, range is set to 1 and the threshold is equal to "maximum". Values more than "maximum" are considered as the extreme cases, which will be removed as noisy instances.

3 Noise Detection with FMF

This section presents a detail discussion about the noise score of FMF with iris dataset. Fig. 3 presents the distribution of noise scores and the noise score of each instance. The minimum and maximum of noise scores were 0 and 0.94, respectively. The reduction rate was about 5.33% ($\approx 100 \times 8/150$) referring to threshold 0.63. According to Fig. 3, the detected noisy instances distributed at the right with higher values and had a distribution different (in location and distributional form) from the other instances.

Figure 4 presents the result of Principal component analysis (PCA) of iris to visualize the relationship between the position and noise score. For each class, a color gradient from blue to red specifics the data points positioned in the center zone, middle zone and border zone. Three points must be remarked:

1. The instances that completely enter another class (e.g., data points with ID 107 and 120) and the outliers (data point with ID 42) got the maximum scores, showing the maximum possibility to be the noise.
2. The data points around the decision boundary (e.g., data points with ID 71, 84, 135) got median scores, showing median possibility to be the noise.
3. The central data points of each class get the minimum scores, showing the maximum possibility to be the normal instances (or representative instances).

4 Comparison Between FMF and the Baselines

In the second experiment, FMF was compared with the case of using the original data, All k-NN, dynamicCF and INFFC, which showed higher accuracy and faster processing in the preliminary experiment. To access the accuracy achieved by the mentioned algorithms, C5.0 was applied. Table 1 lists the datasets. All

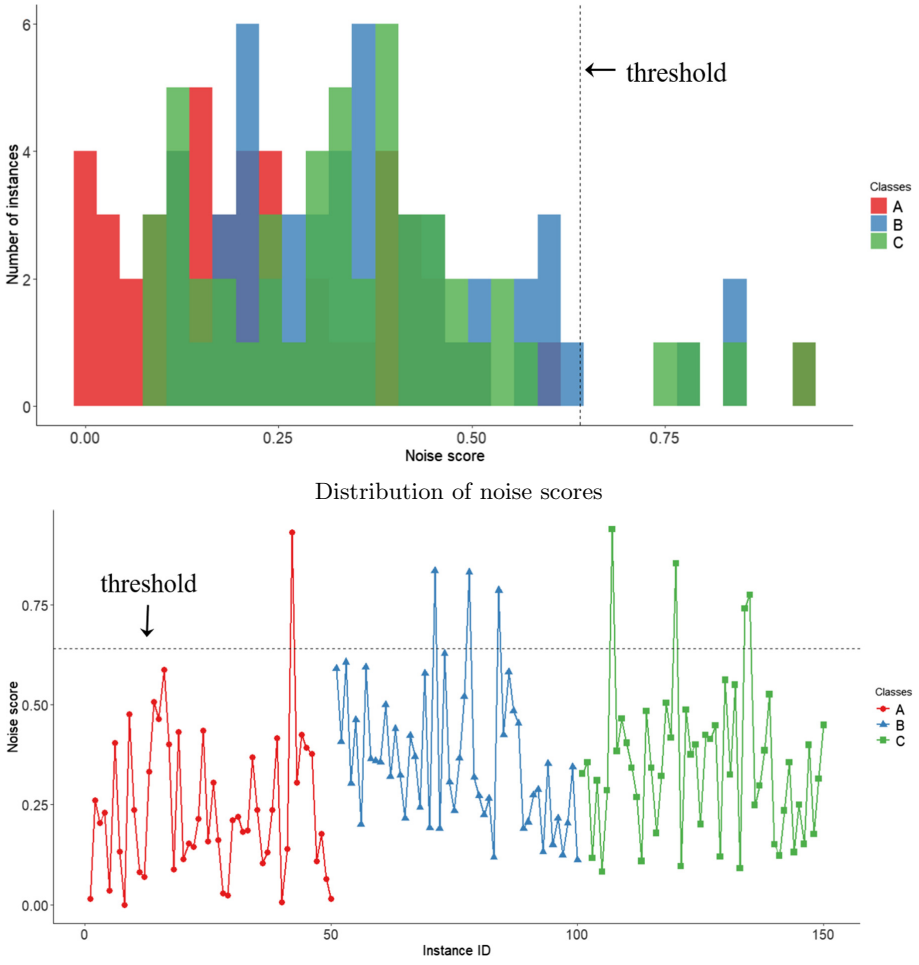

Fig. 3. The distribution of noise scores and the noise score of each instance (iris, the best similarity index: Euclidean).

the datasets were collected from the existing studies of noise detection, including climate data, biological data, physical data, image data as well as text data.

Because noise detection is a multi-objective optimization problem, the classification accuracy of C5.0 (Table 2), the reduction rate (Table 3) and the processing time (Table 4) are shown to verify the hypothesis of the enhanced performance of FMF statistically.

In Table 2, on average, the increases in accuracy were about 3.0%; The largest percentage improvement (PI) was 10% and there were no percentage deteriora-

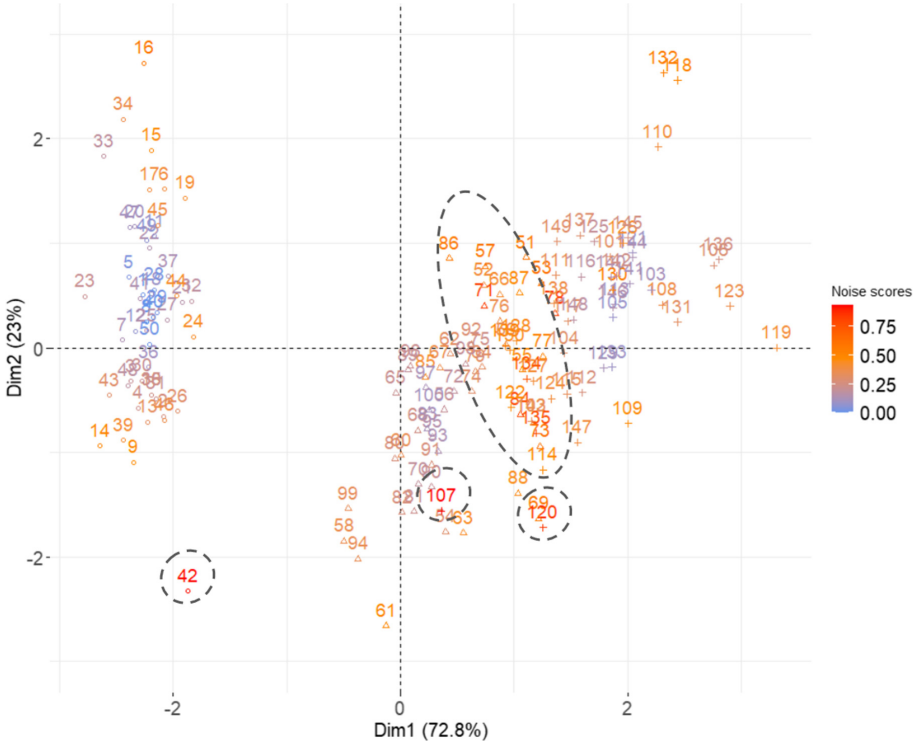

Fig. 4. Principal component analysis (PCA) of iris data.

Table 1. Details of the 10 datasets used in the experiment, including the total number of instances (#instance), the number of attributes (#attribute) and the number of classes (#class).

ID	Dataset	#instance(per class)	#attribute	#class
1	automobile	192(22/23/31/52/64)	14	5
2	heart	267(55/212)	44	2
3	haberman	306(81/225)	3	2
4	SAheart	462(192/270)	9	2
5	fir_c4_500	500(216/284)	25	2
6	climate	540(46/494)	17	2
7	isolet	600(300/300)	546	2
8	halloffame	1340(57/68/1215)	14	3
9	segment	2310(330 × 7)	19	7
10	ozone	2534(160/2374)	72	2

Table 2. Average macro-Fs and ranks for each algorithm in the case of C5.0 works as the classifier. These results are averaged over 10-fold cross-validation. The PI column correspond to the percentage improvement (positive) or percentage deterioration (negative) score, calculated as: $(FMF - All) \times 100$. The highest macro-F of each data is shown in bold.

	All	All k-NN	DynamicCF	INFFC	FMF	PI (%)
automobile	0.75	0.57	0.65	0.70	**0.76**	+1
heart	0.61	0.57	0.57	0.63	**0.67**	+6
haberman	0.52	0.59	0.58	0.52	**0.62**	+10
SAheart	0.55	0.54	**0.57**	0.52	0.55	
fir_c4_500	0.85	0.82	0.82	0.80	**0.87**	+2
climate	0.89	**0.91**	**0.91**	**0.91**	**0.91**	+2
isolet	0.96	0.96	NA	**0.97**	**0.97**	+1
halloffame	0.57	0.60	0.59	0.59	**0.63**	+6
segment	0.96	0.96	0.96	0.96	0.96	
ozone	0.60	0.57	0.52	0.59	**0.64**	+4
Average	0.73	0.71	0.69	0.72	**0.76**	

Table 3. Average reduction rates of all algorithms. These results were averaged over 10-fold cross-validation. The highest reduction rate of each data is shown in bold.

	All k-NN	dynamicCF	INFFC	FMF
automobile	**0.32**	0.25	0.18	0.05
heart	**0.29**	0.20	0.07	0.07
haberman	**0.37**	0.27	0.21	0.04
SAheart	**0.49**	0.41	0.35	0.04
fir_c4_500	**0.37**	0.14	0.17	0.05
climate	0.13	0.09	0.07	**0.20**
isolet	0.01	0.00	0.00	**0.03**
halloffame	**0.09**	0.07	0.05	0.01
segment	**0.04**	0.02	0.01	0.01
ozone	0.07	0.06	0.04	**0.25**
Average	**0.22**	0.15	0.12	0.08

tion (PD) cases; Additionally, among the 10 datasets applied, the win, tie and loss of FMF were 8, 2, 0, respectively.

In Table 3, All k-NN achieved a performance in terms of reduction rate that is better comparable to the performances of the other methods considered in the assessment, which is 22% on average. Followed by dynamicCF (15%), INFFC (12%) and FMF (8%). FMF removed the fewest instances.

Table 4. Average processing time(s) of all algorithms. These results were averaged over 10-fold cross-validation. The fastest processing of each data is shown in bold.

	All k-NN	dynamicCF	INFFC	FMF
automobile	2.92	5.07	29.03	**0.09**
heart	6.68	8.25	29.67	**0.55**
haberman	1.40	2.13	30.55	**0.53**
Saheart	4.02	6.43	297.53	**0.30**
fir_c4_500	10.22	13.03	242.70	**0.48**
climate	10.88	12.07	45.39	**0.42**
isolet	286.56	NA	346.12	**8.97**
halloffame	28.54	25.52	160.37	**1.92**
segment	98.01	80.41	210.36	**5.15**
ozone	227.87	188.08	1293.67	**8.94**
Average	67.71	37.89	268.54	**2.73**

In Table 4, FMF processed significantly faster than the other methods. On average, the processing time was 2.73 s per run, which was about 1/25, 1/12 and 1/90 of the averaged processing times of All k-NN, dynamicCF and INFFC, respectively. Note that one important reason for taking All k-NN, dynamicCF and INFFC as the baselines is because of their faster processing; however, according to the results obtained, FMF was the fastest one.

5 Conclusions

Most of the existing noise detectors are very expensive when the size of the training dataset is large, which was also demonstrated in our preliminary experiment. This study proposed a fast noise detector with fundamentally different structure that makes a multi-factor-based learning rather than the normal simple learning (e.g., density-based learning) or ensemble-based learning with classifiers. Removing the noisy instances which completely enter other classes first and then smoothing the decision boundary allows FMF to achieve a promising result.

According to the results, (1) removing instances identified by FMF for training achieved the highest overall classification accuracy compared with the classifiers trained on the entire training data as well as with noise removed by the other methods. The classification accuracy was improved up to 10%; (2) FMF efficiently speeded up the computation. Compared with the second fast dynamicCF and the slowest INFFC, its speed was about 12 times and 90 times faster, respectively; (3) FMF reached the lowest reduction rate. Compared with All k-NN which achieved the maximum reduction rate, the value of FMF was about its 1/3. However, the low reduction rate also demonstrates the low possibility of over-cleansing.

In the subsequent study, we are interested in detecting the redundant data points with the noise scores.

References

1. Garcia, L., de Carvalho, A., Lorena, A.C.: Effect of label noise in the complexity of classification problems. Neurocomputing **160**, 108–119 (2015)
2. Hu, Z., Li, B., Hu, Y.: Fast sign recognition with weighted hybrid k-nearest neighbors based on holistic features from local feature descriptors. J. Comput. Civ. Eng. **31**(5) (2017). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000673
3. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceeding of 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422 (2008) Location (1999). https://doi.org/10.1109/ICDM.2008.17
4. Sáez, J., Galar, M., Luengo, J., Herrera, F.: Analyzing the presence of noise in multi-class problems: alleviating its influence with the one vs one decomposition. Knowl. Inf. Syst. **38**, 1–28 (2014)
5. Chen, J., Zhang, C., Xue, X., Liu, C.L.: Fast instance selection for speeding up support vector machines. Knowl. Based Syst. **45**, 1–7 (2013)
6. Rathee, S., Ratnoo, S., Ahuja, J.: Instance selection using multi-objective chc evolutionary algorithm. In: Fong, S., Akashe, S., Mahalle, P.N. (eds.) Information and Communication Technology for Competitive Strategies. LNNS, vol. 40, pp. 475–484. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0586-3_48
7. de Haro-García, A., Pérez-Rodríguez, J., García-Pedrajas, N.: Combining three strategies for evolutionary instance selection for instance-based learning. Swarm Evol. Comput. **42**, 160–172 (2018)
8. Sáez, J.A., Galar, M., Luengo, J., Herrera, F.: INFFC: an iterative class noise filter based on the fusion of classifiers with noise sensitivity control. Inf. Fusion **27**, 19–32 (2016)
9. Garcia, L., Lehmann, J., Carvalho, A., Lorena, A.: New label noise injection methods for the evaluation of noise filters. Knowl. Based Syst. **163**, 693–704 (2019)