Label Noise Correction Methods

Bryce Nicholson
Computer Science Department
University of Central Arkansas
bnicholson1@uca.edu

Victor S. Sheng
Computer Science Department
University of Central Arkansas
ssheng@uca.edu

Jing Zhang
School of Computer Science and Engineering
Nanjing University of Science and Technology
Xiaolingwei 200, Nanjing 210094, China
jingzhang.cs@gmail.com

Zhiheng Wang
IT Department
China Executive Leadership Academy Pudong
Shanghai 201204, China
zhwang@celap.org.cn

***Abstract*—The important task of correcting label noise is addressed infrequently in literature. The difficulty of developing a robust label correction algorithm leads to this silence concerning label correction. To break the silence, we propose two algorithms to correct label noise. One utilizes self-training to re-label noise, called Self-Training Correction (STC). Another is a clustering-based method, which groups instances together to infer their ground-truth labels, called Cluster-based Correction (CC). We also adapt an algorithm from previous work, a consensus-based method called Polishing that consults with an ensemble of classifiers to change the values of attributes and labels. We simplify Polishing such that it only alters labels of instances, and call it Polishing Labels (PL). We experimentally compare our novel methods with Polishing Labels by examining their improvements on the label qualities, model qualities, and AUC metrics of binary and multi-class data sets, and ultimately conclude that only CC can significantly improve label qualities, model qualities, and AUC metrics consistently. STC and PL can improve these metrics in some cases, but not as reliably. Hence, our Cluster-based Correction method is the best.**

I. Introduction

A main concern in data preprocessing is ensuring that the data in question is clean. In real-world applications, label noise is an unavoidable phenomenon. Briefly put, label noise is the presence of instances whose class values are incorrect. Label noise may be the result of many reasons. Labeling error is one of the main causes. It is unfortunate that label noise is so prevalent, for data practitioners need reliable data for any practical application, such as informational retrieval, statistical analysis, or machine learning. A well-known domain to which dealing with label noise applies is crowdsourcing, in which human workers can be hired to label instances. It is clear that a proportion of instances may be given incorrect labels through crowdsourcing. Although it has been shown that methods of obtaining multiple labels by crowdsourcing and inferring the most-likely label of each instance can greatly reduce label noise [1], the data will still contain some degree of label noise. Thus, it would benefit data mining practitioners if there are more intelligent and effective label noise correction methods available.

Copyright notice: 978-1-4673-8273-1/15/$31.00 2015 IEEE

In this paper, we consider label noise to be a random phenomenon. However, label noise is not always completely random. There exists a phenomenon called *adversarial noise*, which occurs when a malicious agent is permitted to select labels for a given number of instances [2]. Typically, this agent would introduce noise in such a manner as to be particularly difficult to handle.

Although there is a shortage of literature on label noise correction, there is a lot of literature on handling label noise in general. One method of handling label noise is using a noise filter. A noise filter explicitly seeks to eliminate instances from a training set, which are determined by the filter to be mislabeled. Types of noise filters include Multiedit [3], depuration [4], Classification Filter [5], and Iterative Partitioning Filter [6]. Noise filters, despite being a generally effective group of methods for handling noise, can pose the drawback of over-cleansing, or filtering too many instances [7], such that the remaining instances are not enough to train a good classifier. Another group of methods that implicitly handle label noise, using classification methods that are robust to mislabeled data, are bagging, boosting, and random forests [8], Bayesian approaches—e.g., [9]—among many others, which are methods known to perform well even in the presence of noise [7]. Specifically, one work focuses on minimizing the risk incurred from label noise to create an unbiased learning algorithm, assuming that the noise levels are known [10]. However, according to our knowledge, there is only one method to correct noisy labels (belonging to a discrete set)—that is, to identify each mislabeled instance and to correct it according to what a correction method deems its most likely ground-truth label (i.e., its true class) to be. This method in literature is called Polishing. Polishing aims to change attribute values of instances as well as labels [11]. However, it has been observed that the effects of label noise are much more detrimental to classification than that of attribute noise [12] [13]. We seek to tackle this problem first by considering only label noise.

As a side note, there is another method for noise correction [14], but it concerns textual "spectral" labels, while we consider only the discrete case. Because of the lack of literature on discrete label noise correction and the data practitioner's

need to improve data quality, it is necessary for us to conduct foundational research on this topic and to open the doorway for future contributions. In conducting this research, we seek to fulfill that necessity. In Section 2, we begin by discussing two correction methods that we developed(Self-Training Correction and Cluster-based Correction), and another method we adapted to compare with our methods (Polishing Labels). In Section 3, we test these three methods in their ability to improve label quality, model quality, and AUC on binary class data sets and multi-class data sets. In Section 4, we conclude that Cluster-based Correction achieves the best, most consistently positive results.

II. Label Noise Correction Methods

We define three methods to correct label noise in this section, which we use in our experiments in the next section. We first adapt Teng's method such that it only corrects labels, preserving as much of her Polishing algorithm as possible. Then, we will introduce our proposed methods, Self-Training Correction (STC) and Cluster-based Correction (CC).

A. Polishing Labels

As we discussed in Section 1, according to our knowledge, Teng's method seems to be the only noise correction method available in literature. Unfortunately, it also focuses on attribute noise correction, which is not considered in this paper. We remedy this by simplifying the algorithm such that it only focuses on improving the label quality of each data set. It works by taking a single classification algorithm and building ten different models, separating the data set into ten folds on which to build each model. Each instance in the set is then given a *vote*, or a guess as to what its ground-truth label is, from each model, resulting in ten votes per instance. The label that receives the most votes is ascribed to the instance. If there is a tie and the original label is involved in the tie, it is kept. However, if there is a tie and the original label is not involved in the tie, the instance's new label is randomly selected from the labels involved in the tie.

We acknowledge that this implementation is a vast simplification of the author's original algorithm, yet it is justified because the domain of our problem does not include altering attribute values. Since she called this method Polishing, we refer to our adaptation as Polishing Labels (PL).

B. Self-Training Correction

The problem of label noise correction is not unlike that of using a set of labeled data to infer labels of originally unlabeled data, a process known as self-training, described by [15]. Extending the notion to apply to label noise correction is as simple as considering the labeled data in the self-training paradigm as the pool of data remaining after a noise filter has discarded some instances from a data set, which are determined to be noisy. The unlabeled data is therefore analogous to the set of discarded instances, and the task of correcting noisy data becomes easily extendable to the self-training paradigm.

In specific, the Self-Training Correction algorithm first uses a noise-filtering algorithm on a data set to generate a noisy set and a clean set. Then, the algorithm builds a model from the clean set and uses that to calculate the confidence that each of the instances from the noisy set is mislabeled. The noisy instance with the highest calculated likelihood of belonging to some class that is not equal to its current class is relabeled to the class that the classifier determined is the instance's most likely true class. The relabeled instance is then added to the clean set, and the process is repeated until some proportion of noisy instances is relabeled and added to the clean set. Self-Training Correction is flexible in that the user can specify what proportion of noisy instances he wants to correct.

With that in mind, we take inspiration from the self-training algorithm of [15] and develop a noise correction algorithm we call Self-Training Correction (STC).

C. Cluster-based Correction

We propose a novel method for label noise correction, which is based on clustering, a popular unsupervised learning technique [16]. The beauty of clustering in a noisy scenario is that clusters are formed independently of instances' labels—regardless of the noise level of a data set, the same clustering algorithm (with the same parameters) will cluster the data set in exactly the same way. This is completely different from the previous two methods, Polishing Labels and Self-Training Correction. Both Polishing Labels and Self-Training Correction depend on the label quality of the original data set.

Our cluster-based correction method (denoted as CC) executes one or more clustering algorithms on a training set several times, giving the same set of weights to all instances in each cluster based on the distribution of ascribed labels in that cluster and the cluster's size. The weights favor the class where the majority (or plurality) of instances belong to that class. After the data set has been clustered several times, weights obtained from each cluster are summed for each instance, and each instance is given the label corresponding to the maximum weight.

1) One Implementation of Cluster-based Correction: In this paper, we propose a single implementation of CC, acknowledging that CC, instead of embodying just one algorithm, is the framework for a family of algorithms, all of which conform to the above description. Its pseudo-code is shown in Algorithm 1.

The label vector $\mathbf{y}$ contains all the ascribed labels for the instances in the data set X, and the label set Y contains all possible values for the labels in $\mathbf{y}$.

Lines 1 - 4 in Algorithm 1 work to count the distribution of ascribed labels in the data set. This information will be utilized to calculate the label weights for each instance, shown in Algorithm 2. Lines 5 - 14 in Algorithm 1 perform all required clustering algorithms. In our case, we use k-means clustering, varying k from two to roughly half the number of instances in the set (Line 6). This process should yield lots of large and small clusters, which adds to the diversity of our clusters. Line 11 calculates the weights for each instance in each cluster. Finally, Lines 15 - 18 use the totaled weights to perform correction.

The process of calculating weights is detailed in Algorithm 2.

Note that the multiplier (Line 5 in Algorithm 2) works to give more magnitude to the larger clusters, so that the

Algorithm 1 ClusterCorrection

Require: data set X, corresponding label vector **y**, label set Y, number of clusterings a

for $i \leftarrow 1$ to $|X|$ **do**
  $index \leftarrow y_i$
  $LabelTotals_{index} \leftarrow LabelTotals_{index} + 1$
end for
for $i \leftarrow 1$ to a **do**
  $k \leftarrow \frac{i}{a} \times \frac{|X|}{2} + 2$
  $C \leftarrow KMeansCluster(X, k)$
  for $j \leftarrow 1$ to number of clusters $|C|$ **do**
    for all instances **x** in C_j **do**
      $InsWeights_{\mathbf{x}} \leftarrow InsWeights_{\mathbf{x}} +$
      $CalcWeights(C_j, LabelTotals, Y)$
    end for
  end for
end for
for all instances **x** in X **do**
  $LabelGuess \leftarrow argmax(InsWeights_{\mathbf{x}})$
  $y_i \leftarrow LabelGuess$
end for

maximum multiplier, obtained from a cluster of 100 instances, is 2. We do this so that very big clusters do not drown out the smaller ones in significance. $\frac{d_i - u_i}{v_i}$ calculates the difference between the expected proportion u_i (i.e., $\frac{1}{|Y|}$, where $|Y|$ is the number of classes) of instances of class i in a cluster with no useful information and the actual proportion d_i of instances of class i in the cluster, scaled by the proportion v_i of instances of class i in the data set.

Algorithm 2 CalcWeights

Require: Cluster c, data set distribution vector **v**, label set Y
Ensure: Weight vector **w**

d $\leftarrow Distribution(c)$
for all labels y in Y **do**
  $u_y \leftarrow \frac{1}{|Y|}$
end for
$multiplier \leftarrow min(log_{10}(sizeof(c)), 2)$
for $i \leftarrow 1$ to $|Y|$ **do**
  $w_i \leftarrow multiplier \times \frac{d_i - u_i}{v_i}$
end for
return **w**

At first glance, it seems that using only the k-means clustering algorithm may be a short-sighted implementation choice. Can only considering one type of clustering algorithm yield sufficient diversity? While it may be true that more cluster diversity can be obtainable by conferring with multiple types of clustering algorithms, a sufficient degree of diversity can be achieved with k-means alone. This is ensured by k-means's tremendous dependence on its initial centroid generation [12]. Different from previous work, we strive to increase the diversity of our clusters by varying the number of initial centroids k. The effectiveness of considering a range of k-values in k-means clustering is validated in [17], where the authors concluded that this technique "will always provide sensible clustering solutions." Therefore, combining k-means's initial centroid sensitivity with a varying number of initial centroids (k) is a recipe for cluster diversity.

III. Experiments

In this section, we describe our experimental setup and provide our experimental results for comparing the three methods in terms of label quality, model quality, and AUC. In multi-class scenarios, AUC was calculated using a pairwise average AUC metric over each pair of classes.

A. Setup

We use WEKA [18] in both our classification and clustering tasks for the following experiments. We carry out our experiments by taking clean data sets and injecting p noise into them. That is, given a data set X, we select $p \times |X|$ instances at random, assigning them a different label (randomly, when there is more than one alternative, as in multi-class problems). We use nine noise levels p to investigate the performance of the three algorithms. That is,

$$p \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45\}.$$

1) Polishing Labels: As our classifier for PL, we used WEKA's C4.5 decision tree implementation, J48.

2) Self-Training Correction: For our implementation of STC in these experiments, we run Classification Filter on each dataset using J48, and set the number of folds for validation at 10. The STC algorithm itself also uses J48 as its classifier.

3) Cluster-based Correction: Our implementation of CC is mostly detailed in the pseudo-code shown in Algorithms 1 and 2. One extra detail is that we use 200 iterations of WEKA's SimpleKMeans algorithm.

4) Data Sets: To investigate the performance of the three label correction methods on binary and multi-class scenarios respectively, we choose half our data sets to be binary data sets and the other half to be multi-class data sets. Therefore, we will conduct our experiments on 12 binary class data sets and 12 multi-class data sets. Their characteristics are shown in Table 1 and Table 2, respectively. The latter four columns of both Table 1 and Table 2 display the number of categorical (or nominal) attributes, the number of numerical attributes, the number of instances, and the number of classes in each data set, respectively.

Data Set	Cat. Atts.	Num. Atts.	Insts.	Classes
breast-w	0	9	699	2
colic	20	7	368	2
credit-a	9	6	690	2
credit-g	13	7	1000	2
diabetes	0	8	768	2
heart-stat-log	0	13	270	2
hepatitis	13	6	155	2
ionosphere	0	34	351	2
kr-vs-kp	36	0	3196	2
labor	8	8	57	2
mushroom	22	0	8124	2
sonar	0	60	208	2

TABLE I: Experimental binary class data sets.

Data Set	Cat. Atts.	Num. Atts.	Insts.	Classes
autos	10	15	205	7
balance	0	4	625	3
cars	6	0	1728	4
heart-c	7	6	303	5
heart-h	7	6	294	5
iris	0	4	150	3
lymph	15	3	148	4
segment	0	19	2310	7
vehicle	0	18	846	4
vowel	3	10	990	11
waveform	0	40	5000	3
zoo	16	1	101	7

TABLE II: Experimental multi-class data sets.

B. *Experimental Results*

In evaluating the performance of the three label noise correction methods, label accuracy, resulting model quality, and AUC are all used as the measurement in our experiments. Label quality is defined in terms of accuracy (i.e., the proportion of instances whose ascribed label and true label are the same). Model quality is defined as the classification accuracy obtained by training a classifier on the corrected data set and performing cross-validation. In our case, we used WEKA's SMO Support Vector Machine classification algorithm [18], with 10 folds for cross-validation. Finally, AUC represents the Area Under the Curve of the Receiver Operating Characteristic (ROC) curve. AUC reflects how well each class was predicted, so that if a minority class is misclassified frequently, it would not harm label quality as much as it would harm AUC. We will discuss the effectiveness of these three methods in terms of label quality improvement, model quality improvement, and AUC improvement.

The experimental results of the three methods are shown in figures. On a figure, the horizontal axis represents the label noise level p, and the vertical axis represents the respective metric (i.e. accuracy, model quality, or AUC) of the three methods. In order to show the improvement, we have a dashed line to represent the baseline metrics in each figure. For the accuracy figures, each dashed line bisects each figure. For example, the line contains the point $(20, 0.80)$. This signifies that at 20% noise, i.e., $p = 0.2$, the baseline label quality is 0.80. The baselines for model quality and AUC are more variable, however. On each figure, the results for Cluster-based correction is shown in blue, the results for Self-Training Correction is shown in red, and the results for Polishing Labels is shown in green.

The area above the baseline signifies improvement, and the area below the baseline denotes decline. The vertical distance between each point in the curve of each method and the baseline shows how much improvement the method has achieved. The higher the vertical distance, the greater the improvement is.

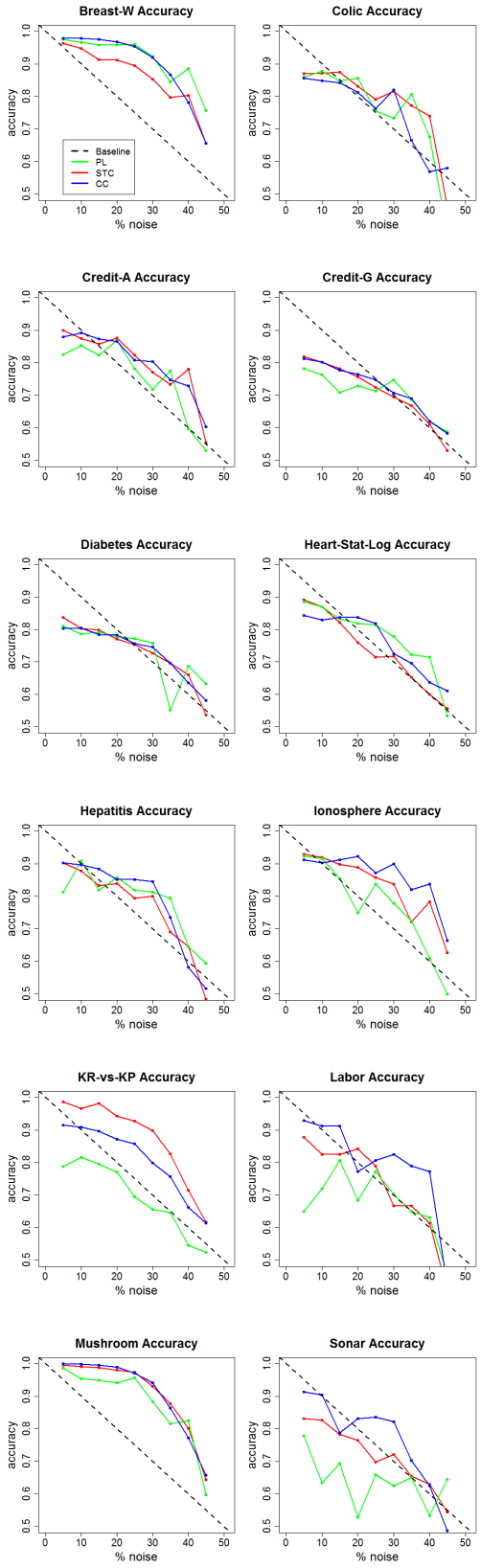

Fig. 1: Accuracy results on binary class data sets.

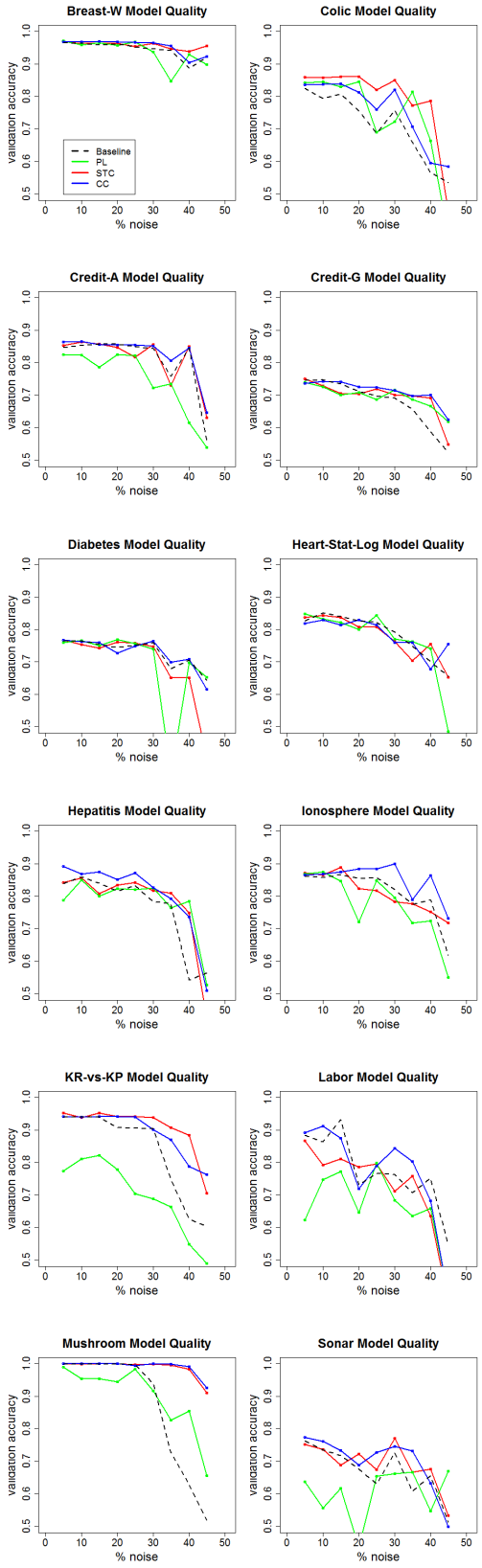

Fig. 2: Model quality results on binary class data sets.

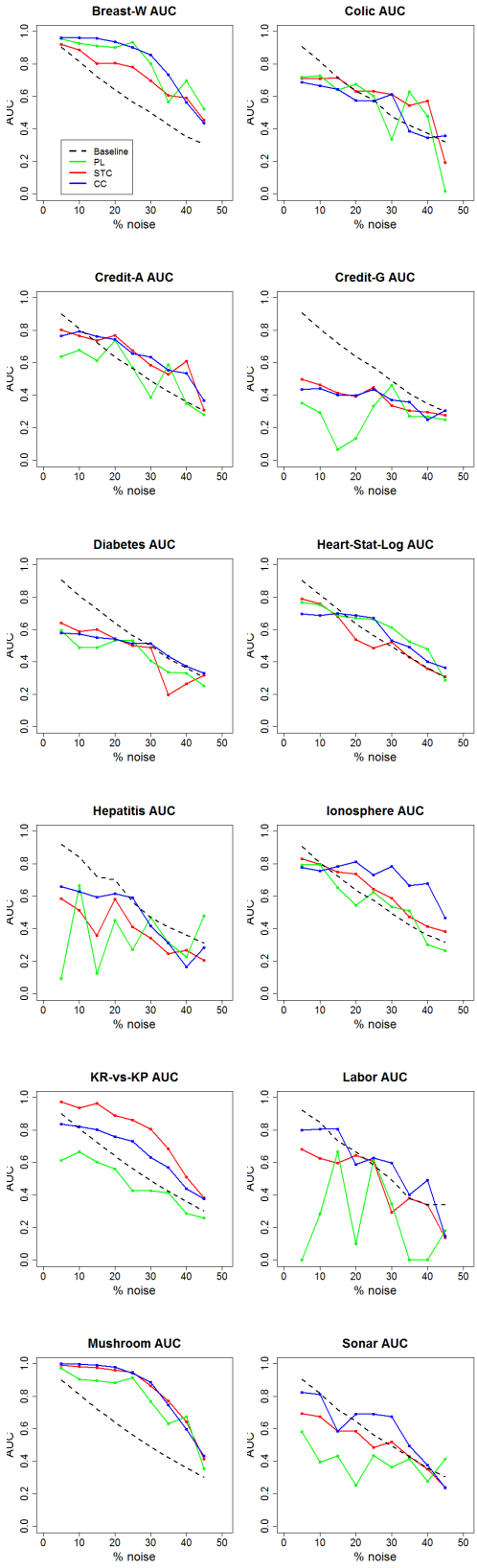

Fig. 3: AUC results on binary class data sets.

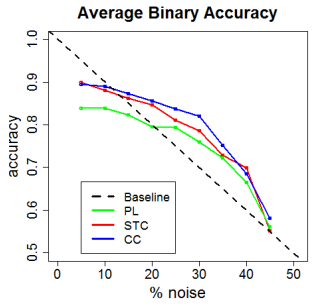

Fig. 4: Average accuracy results on binary class data sets

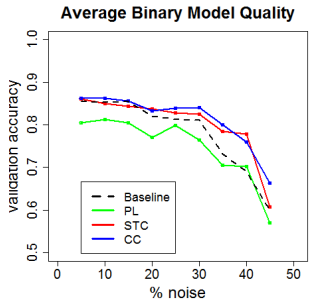

Fig. 5: Average model quality results on binary class data sets

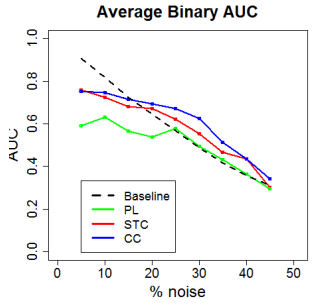

Fig. 6: Average AUC results on binary class data sets

1) Binary Class Data Sets: The influence on accuracy (i.e., label quality) of each label noise correction algorithm is shown in Figure 1 for all binary data sets in our experiments. Figure 1 shows that the three label correction methods improve the label quality on most of data sets. In all the experimental label noise levels in the interval $[0.05, 0.45]$, the three methods significantly improve the label qualities on two data sets (*breast-W* and *mushroom*). On most of the remaining experimental data sets, each of the three methods improves their label qualities after the label noise reaches a certain level. Among the three methods, CC performs the best, followed by STC, in terms of accuracy.

Figure 2 shows the model quality improvement of each correction method on all the binary class data sets. On most data sets, STC and CC improve model quality from the baseline (e.g. *kr-vs-kp*, *mushroom*, *sonar*). As with accuracy, PL performs worst in terms of model quality. We find that PL harms model quality on the majority of the binary data sets (e.g. *credit-a*, *labor*, *kr-vs-kp*). We see again that STC and CC are both very competitive with each other in terms of model quality improvement, while PL performs the worst.

The AUC improvement of each method on each binary class data set is shown in Figure 3. On most of the data sets, we see CC improve AUC from the baseline after the noise reaches a certain threshold (e.g. *credit-a*, *diabetes*, *heart-statlog*, *kr-vs-kp*). On the data sets *mushroom* and *breast-w*, all three methods improve AUC under all noise levels.

To summarize the performance of the three methods better, we have the average label quality improvement results over the 12 binary data sets shown in Figure 4. From Figure 4, we can see that when the label noise level $p > 0.10$, CC improves the label quality. When the label noise level $p > 0.15$, STC can improve the label quality, and when $p > 0.20$, PL can finally improve the label quality. Among the three label correction methods, CC outperforms both STC and PL in the experimental label noise range except at $p = 0.05$ and $p = 0.40$, where STC is only slightly superior. CC dominates both STC and PL on average. PL is only competitive with STC and CC when the noise level $p \geq 0.40$.

The average model quality improvement results on the twelve binary class data sets are shown in Figure 5. CC is the only correction method that does not harm model quality on average for any of the noise levels. For all noise levels, CC greatly outperforms PL, and on most noise levels, CC outperforms STC as well. STC is a close second to CC, though, but PL is clearly the worst. PL almost always harms model quality on average.

Figure 6 displays the average AUC improvements of all three correction methods on the binary class data sets. For the noise levels $p > 0.15$, both STC and CC improve AUC on average, with CC showing superior improvements to STC. For the noise levels $p > 0.20$, PL can leave AUC approximately the same as it was before correction on average, but cannot really improve AUC. To conclude, CC is the superior method in terms of AUC improvement on average.

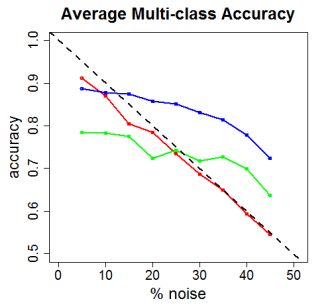

Fig. 10: Average accuracy results on multi-class data sets

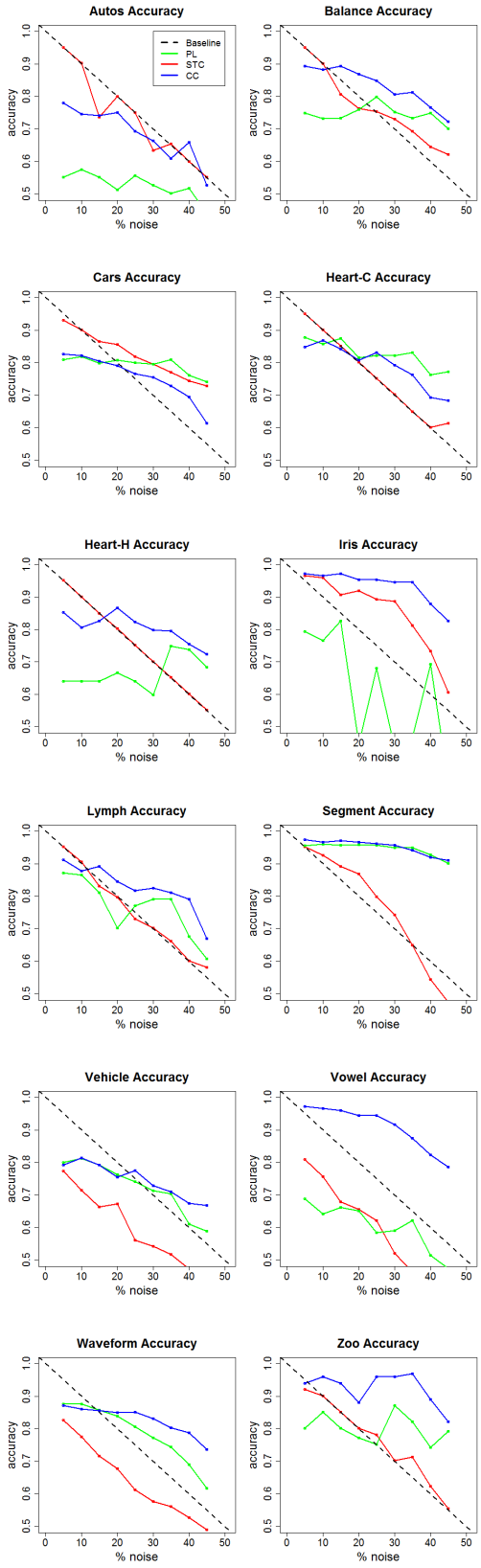

Fig. 7: Accuracy results on multi-class data sets.

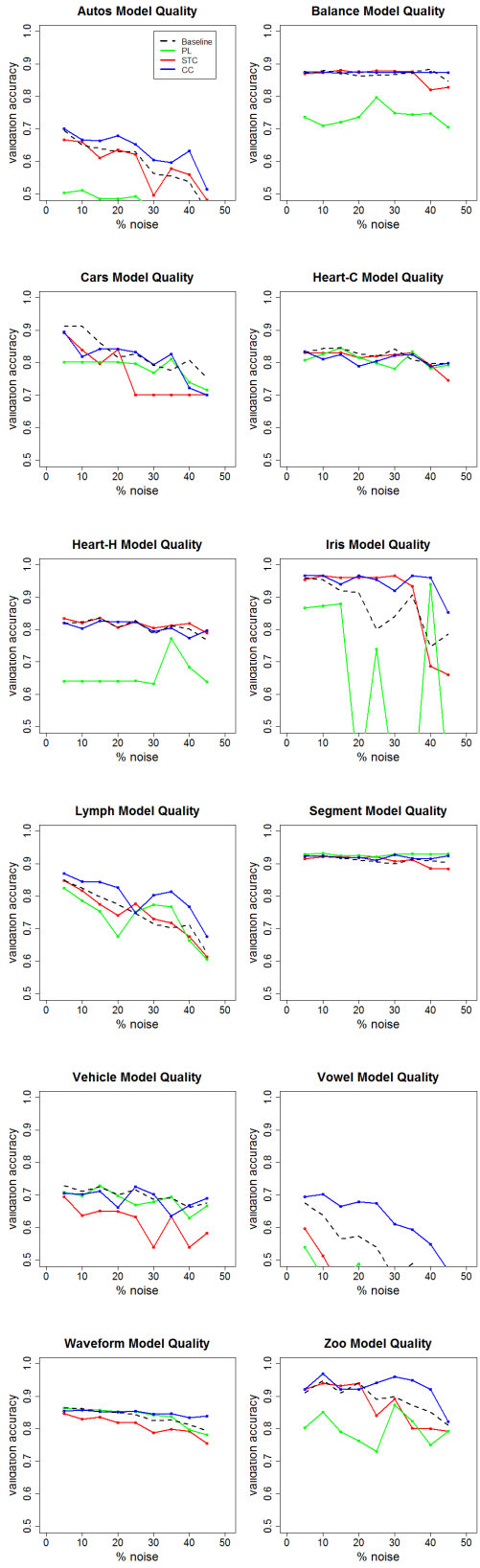

Fig. 8: Model quality results on multi-class data sets.

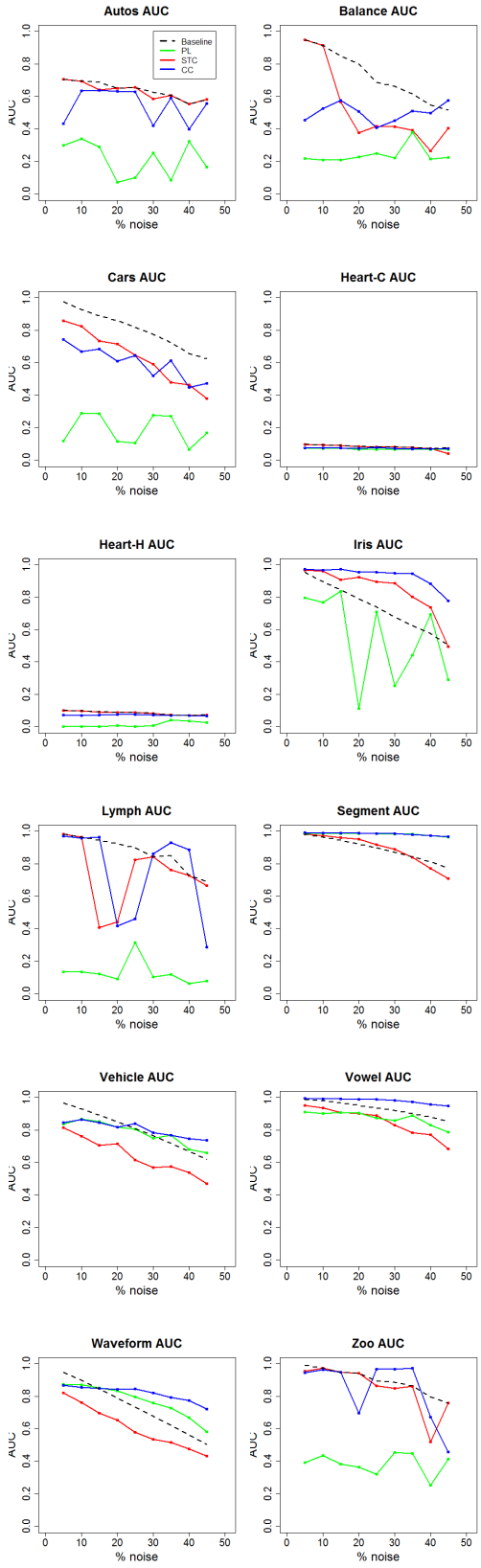

Fig. 9: AUC results on multi-class data sets.

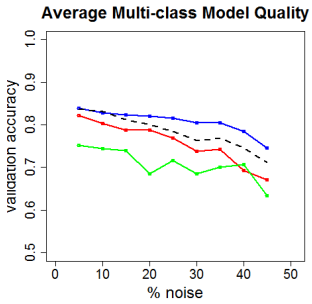

Fig. 11: Average model quality results on multi-class data sets

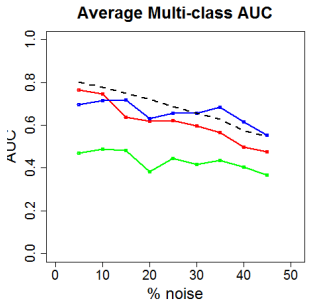

Fig. 12: Average AUC results on multi-class data sets

2) Multi-class Data Sets: The impact of the algorithms on accuracy of the 12 multi-class data sets is shown in Figure 7. The trends discussed for accuracy of the binary class data sets are a bit different for the multi-class data sets: CC performs the best, but it is PL, not STC, that is CC's competitor (in most cases). Both CC and PL perform much better (in terms of label quality) than STC on most of the multi-class data sets. STC does not work well on most of these data sets, except that it improves the label quality on the data sets *iris* and *cars*, and it also slightly improves the label quality on *segment* until the noise level $p \geq 0.35$. This is completely opposite from its performance on the binary data sets, where STC can almost always improve label quality. However, comparing the performance of both CC and PL on the binary data sets vs. the multi-class data sets, the label quality improvement of both CC and PL is much more significant on the multi-class data sets, especially under heavy noise. This observation is much more obvious if we compare their average performance on the binary data sets and on the multi-class data sets.

Figure 8 shows the model quality improvements of each correction method on all twelve multi-class data sets. On the majority of these data sets, CC improves model quality much more than the other two methods (e.g. *zoo*, *lymph*, *autos*). STC generally is second-best in improving model quality, and PL is the worst. On some data sets (e.g. *iris* and *heart-h*), PL's resulting model quality is exceptionally poor.

Figure 9 shows the AUC improvements of the correction methods on the multi-class data sets. CC performs best (e.g. *vowel*, *waveform*, *segment*), STC shows decent performance, and PL is the worst. On most of the data sets, the correction methods can improve AUC from the baseline under some noise

levels, but sadly, there were a couple of data sets (i.e. *cars*, *autos*) where the correction methods harmed AUC under all noise levels.

The average accuracy improvements of all three correction methods over the twelve multi-class data sets are shown in Figure 10. CC is clearly superior to the other correction methods. CC can improve label quality when the noise level $p > 0.10$ on average. PL cannot improve label quality until $p \geq 0.30$ on average, and STC cannot improve label quality at all, on average. Only on the noise level $p = 0.05$ does STC outperform CC. Figure 10 clearly shows the trends discussed above: in the multi-class scenario, on the noise level interval $[0.20, 0.45]$, PL greatly outperforms STC.

Figure 11 shows the average model quality improvements of the correction methods over the multi-class data sets. When the noise level $p \geq 0.10$, CC can improve model quality on average. STC and CC cannot improve model quality at all on average.

Finally, Figure 12 displays the average AUC improvements of the three correction methods over the multi-class data sets. When the noise level $p \geq 0.35$, CC improves AUC on average, while the other two methods cannot improve AUC on any of the noise levels on average.

IV. Conclusion

In this paper, we have set the groundwork for much more research on the topic of label noise correction, a topic that has been addressed much less than it deserves, given the importance of cleanly labeled data for the data practitioner. We have pointed out the lack of a formally defined label noise correction algorithm in literature. We proposed two novel label noise correction methods (CC and STC) and adapted one (PL). Our experimental results showed that all three methods improve the label quality when the label noise reaches a certain level, and that only CC could reliably improve model quality and AUC. STC is second best in all scenarios except in improving label quality on mulit-class data sets, in which case it performs worst. We found that in all other scenarios, PL performs worst out of the three correction methods. Among the three methods, our proposed method CC performs the best on both binary class data sets and multi-class data sets, taking into consideration the three metrics we used for evaluation.

However, we also notice that the three label noise correction methods could not improve the label quality further on half of the experimental data sets when the label noise level is lower than 0.10. When the label noise is very low, it is very challenging to improve the label quality. We will conduct further research on this. As we know, data quality is important for information retrieval, statistics, and learning. An even more challenging quandary is why PL always harms model quality and AUC so greatly on multi-class data sets, while it improves the label quality when the noise level $p > 0.25$. Future research is necessary to discover these answers.

References

[1] V. S. Sheng, F. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 614–622.

[2] P. Auer and N. Cesa-Bianchi, "On-line learning with malicious noise and the closure algorithm," *Annals of mathematics and artificial intelligence 23, no. 1-2*, pp. 83–99, 1998.

[3] J. K. Devijver, "On the edited nearest neighbor rule," in *Proceedings of the Fifth International Conference on Pattern Recognition*, 1980, pp. 72–80.

[4] J. S. Sanchez, R. Barandela, A. I. Marques, R. Alejo, and J. Badenas, "Analysis of new techniques to obtain quality training sets," *Pattern Recognition Letters 24, no. 7*, pp. 1015–1022, 2003.

[5] D. Gamberger, R. Boskovic, N. Lavrac, and C. Groselj, "Experiments with noise filtering in a medical domain," in *Proceedings of the 16th International Conference on Machine Learning*, 1999, pp. 143–151.

[6] T. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," *J. Comput. Sci. Technol. 22*, pp. 387–396, 2007.

[7] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *Neural Networks and Learning Systems, IEEE Transactions on, 25(5)*, pp. 845–869, 2014.

[8] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning 40, no. 2*, pp. 139–157, 2000.

[9] A. Gaba and R. L. Winkler, "Implications of errors in survey data: a bayesian model," *Management Science 38, no. 7*, pp. 913–925, 1992.

[10] 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.

[11] C.-M. Teng, "Correcting noisy data," in *Proceedings of the sixteenth international conference on machine learning*. Morgan Kaufmann Publishers Inc., 1999, pp. 239–248.

[12] R. Xu and D. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on 16, no. 3*, pp. 645–678, 2005.

[13] J. A. Saez, 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 38, no. 1*, pp. 179–206, 2014.

[14] Y. Song, C. Wang, M. Zhang, H. Sun, and Q. Yang, "Spectral label refinement for noisy and missing text labels," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2972–2978.

[15] I. Triguero, J. A. Saez, J. Luengo, S. Garcia, and F. Herrera, "On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification," *Neurocomputing 132*, pp. 30–41, 2014.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR) 31, no. 3*, pp. 264–323, 1999.

[17] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition 36.2*, pp. 451–461, 2003.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter 11, no. 1*, pp. 10–18, 2009.