MisDetect: Iterative Mislabel Detection using Early Loss

Anonymous Author(s)

ABSTRACT

Abstract here.

ACM Reference Format:

Anonymous Author(s). 2023. MisDetect: Iterative Mislabel Detection using Early Loss. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (SIGMOD' 24).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data quality is one of the most significant problems in data management, among which mislabels are a common dirty data type that could directly lead to low-quality data analysis and misleading business decisions. Labels are always large-scale and error-prone because they may be crowdsourced from non-experts or collected from web annotations, so it is inevitable to use automatic methods to detect mislabels. However, existing mislabel detection approaches suffer from fair accuracy.

Existing Solutions. Traditional methods [] rely on the data locality to detect mislabels. For example, in order to whether the label of an instance in a dataset is correct, the typical KNN method [] checks its neighbors, and if they have inconsistent labels, the instance tends to be mislabeled, otherwise it is a clean one. This type of methods has low detection accuracy because they just consider the local neighbors of each instance rather than the entire dataset. Therefore, to improve the accuracy, machine learning (ML) models are incorporated to help mislabel detection [] because intuitively, mislabels definitely have impact on the supervised model training. For instances, ensemble-based methods [] involve multiple models to train on the entire dataset and check the consistency of the prediction results from these models for each instance. Cleanlab [] utilizes confident learning to estimate the joint distribution of mislabels and correct labels. This line of methods can capture the entire data distribution of the dataset through ML and avoid the data locality problem. However, the accuracy is still not high because they learn the distribution from both the incorrect and correct labels. In this way the model has already fitted on the dirty data, and thus the learned distribution is not accurate, leading to inaccurate prediction results.

Challenge. As discussed above, to pursue high accuracy of mislabel detection, we have to consider the data distribution of the entire dataset rather than the locality. In terms of using ML to capture the distribution, it is challenging to eliminate the impact of mislabels on the learned distribution. Ideally, if we know the distribution learned from all clean instances, we can easily use it to detect the dirty ones, but unfortunately, we cannot know it in advance.

Based on the challenge, we have the following observation and proposed methods.

Observation. Since it is inappropriate to detect mislabels after training because the model has already fitted the dirty data. We consider whether we can leverage the metadata during (or at the beginning of) training to help detect mislabels. The high level intuition is that this strategy clearly avoids data locality because all instances participate in training, while it can also capture the entire data distribution to a certain degree before the model fits dirty instances.

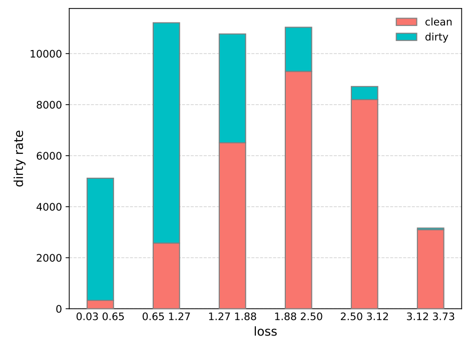

Figure 1: An Example of Early Loss.

Therefore, we observed that the *early loss* can well distinguish mislabeled instances from clean ones. Early loss means the loss of each instance at early training epochs. As shown in Figure 1, the X-axis denotes the range of normalized loss after training the first epoch, and the Y-axis denotes the number of instances falling into the corresponding range. We can see that the mislabeled instances always have large loss, while the clean ones are associated with relatively small loss.

Our proposal. Based on the above observation, generally speaking, we can conclude that mislabeled instances are harder for an ML model to fit than the clean instances especially at early epochs, leading to large loss. Therefore, we propose our MisDetect framework that leverages the early loss to detect mislabeled instances in an ML training set. The key idea is straightforward yet effective that at each early epoch, we sort the instances in the training set according to the loss in descending order. Then we tend to iteratively identify top instances as mislabeled ones and remove them from the training set. We remove because we hope to navigate the model towards a well-performed one, as if it is trained over clean instances. In this way, at the subsequent epochs, the model can better fit clean instances and recognize mislabeled ones.

However, to make the model better, purely removing instances with the largest value is sub-optimal because an instance with a large loss does not imply it has a large impact on the model. Hence, MisDetect has another module to estimate the influence of each instance. A higher influence indicates that if we identify the corresponding instance as mislabeled and remove it, the model will be more affected and quickly lead to a well-performed model trained over clean instances. Besides, we also design a strategy to estimate the rate of mislabeled instances, so as to help judge when the MisDetect stops. What's more, there are likely to exist some outliers in each dataset, which are also hard to be fitted by a model at early epochs with large loss. If they are identified as mislabeled, our precision goes down. Hence, we also propose a module to cope with these outliers.

Contributions. Our contributions are summarized as follows:
(1) We formally model the mislabel detection problem and design an iterative mislabel detection framework `MisDetect` with high accuracy based on machine learning training.
(2) We introduce the early loss to help determine whether an instance is mislabeled, which is very effective.
(3) We propose an influence computation method customized to our problem to further improve the performance of mislabel detection, based on the influence function.
(4) Sufficient experiments on XX datasets over XX baselines demonstrate that `MisDetect` outperforms them up to XX in terms of the F1-score.

2 PRELIMINARY

2.1 Problem Definition

Detection with parameter(gradient?) approximation

The setting of our problems.

Input: A training set which contains mislabeled data.

Output: A subset of the initial training set whose parameters are closest to those of the model trained by removing all the real dirty data in the training set.

Complexity analysis: NP

3 FRAMEWORK

(1)
(2)
(3)

4 INFLUENCE FUNCTION

Theoretical guarantee about the influence function. (add a sample)

5 STOP CONDITION

6 EXPERIMENT

6.1 Experimental Settings

Dataset. We evalute our algorithm on 7 different real-world datasets from a diverse array of domains, whose size varying from the magnitude of 10^3 to 10^6 and the number of class differ from 2 to 100. We first used the dataset form CleanML, which have been performed synthetic mislabel injection with the strategy that flipping 5% of the labels in each class. Then, we also use different kinds of dataset including image and table and dataset with large number of classes to further verify the effectiveness and scalability of our algorithm. The details are listed as follows:

(1) USCensus: This dataset contains 32,561 items about US Census records for adults. Each item has 14 attributes, such as age, education, sex, etc. The classification goal is to predict whether the adult earns more than $50,000.
(2) Marketing: A total of 8993 records are contained in this dataset, which information about household income including education, sex. etc. The goal is to make predictions about whether the annual family income is less than $25,000.
(3) EEG: This dataset consists of 14,980 Electroencephalogram recordings with 14 Electroencephalogram attributes varying from AF3 to, AF4 The classification task is to predict whether the eye-state is closed or open.
(4) CIFAR-10: The dataset is a computer vision data set for universal object recognition, which contains 50000 32 X 32 RGB color pictures, a total of 10 categories. The task is to predict which kind does the picture belong to. Mislabeled data are artificially introduced by flipping labels of 40% for each type of the dataset randomly. **(need explain?)**
(5) CIFAR-100: This dataset is like CIFAR-10, except that it has a total of 100 classes, and each class contains 500 images. The classification task is also to predict the category of a given picture.
(6) CovType: This is also a muti-classification dataset, which contains 7 different forest cover type. With a total of 581012 samples, and each sample consist of 54 attributes, such as Elevation, Wilderness Area, Horizontal Distance To Roadway etc.
(7) Mobile Price Prediction: This is a small tabular dataset with only 2000 records. The task is to predict price range of the mobile on the basis of the information about the mobile, specification like Battery power, 3G enabled , wifi ,Bluetooth, Ram etc.**(need remove?)**

Method. We compare our approach against several competing mislabel detection methods. First, we consider 8 baselines:

1. **K-Nearest Neighbor(KNN):** The method counts the number of inconsistencies between the label of a training instance and the labels of its surrounding neighbors. If there is strong evidence of distinction among the labels, the training instance is marked as mislabeled. This kind of method is called local learning method.

2. **Nearest Centroid Neighborhood (NCN):** This method is also belong to local learning method, which assumes that the labels of mislabeled instances tend to disagree with the labels of other instances in their surrounding neighborhood. The difference between KNN and this method is the approach about how to find the nearest neighbor.

3. **Training Set Debugging Using Trusted Items(DUTI):** DUTI utilizes a small set of additional "trusted items"to help detect incorrectly labeled item, which core is to finds the smallest changes for the labels in training set such that the classifier trained on the changed dtaset classify all the trusted items correctly.

4. **Forgetting Events:** In the process of model training, a sample has been correctly classified by the model. With the update of model parameters, the sample has been incorrectly classified. This process is called a forgetting event of the sample. This method identifies a mislabel sample based on one assumption that noisy sample often experience more forgetting events than normal samples during model training.

5. **Ensemble-based method with consensus filter:** Ensemble-based method assumes that multiple, independent clissifiers often result in conflicting labels about incorrectly labeled training sample.

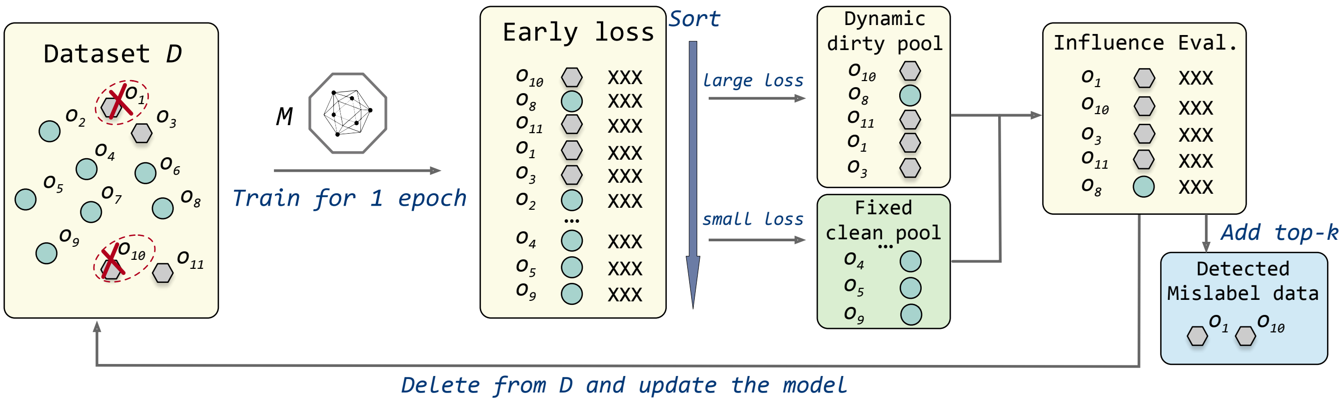

Figure 2: MisDetect Framework

Algorithms that belong to this category vary in terms of how the different classifiers are constructed. Besides, the consensus filter is a strategy which means that a training example could be marked as a mislabel data only if it is misclassified by all the classifiers in the ensemble.

6. **Ensemble-based method with majority vote:** The main idea of this method is tha same as the Ensemble-based method with consensus filter.The only inconsistency between them is that majority vote strategy considers an example to be mislabeled if it disagrees with the majority vote of the classifiers.

7. **Cleanlab:** Cleanlab

8. **Noisy Cross-Validation(NCV):** This method first randomly devide a noisy training set into two halves, then train a neural network for these two half separately. After that, the network which is trained on one half will be applied to another half of the dataset. A sample would be identified as mislabel when its current label is different from its predicted label.

9. **Iterative Noisy Cross-Validation(INCV):** Obviously, this is a iterative method about noisy cross-validation. Apart from selecting mislabel samples, the INCV removes samples that have large categorical cross entropy loss at each iteration.

10. **Self-Ensemble Label Filtering(SELF):** The method

11. **Partition Filter:** This method first partition a noisy dataset into multi-subsets, and then construct a good classifier from each subset. For a given training sample, all classifier will be applied on it, the mislabel sample often have higher probability to have larger misclassified times.

We also consider three variants of our algorithm. The goal is to compare the effectiveness of the cross-validation using both early loss and parameter influence. For all variations, we use the same training paradigms. We consider the following variants:

1. **Without Iteration:** This
2. **Without Parameter influence:**
3. **Without Early Loss**

Hyper-prameter Setting.

Evaluation Metrics. We mainly focus on the effectiveness of our algorithm and other baselines, so we take precision, recall and f1-score as the most important metrics.

6.2 Comparison with Baselines Settings

6.3 Mislabel Ratio Evaluation

6.4 Mislabel Distribution Evaluation

We test the algorithms on two types of label noise: symmetric and class-conditional label noise. In symmetric label noise, each label is randomly flipped to any of the remaining classes with equal probability, whereas for class-conditional noise, label flipping is done in a set of similar classes. For example,

6.5 Model Evaluation

6.6 Stop Condition Evaluation

7 RELATED WORK

8 CONCLUSION

REFERENCES