



저작자표시-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

February 2018  
Master's Degree Thesis

# Strategies to deal with Misabeled Data

Graduate School of Chosun University

Department of Computer Engineering

Malik Muhammad Ammar

# Strategies to deal with Mislabeled Data

기계학습의 정확도 향상을 위한 레이블 노이즈  
제거 알고리즘

February 23, 2018

Graduate School of Chosun University

Department of Computer Engineering

Malik Muhammad Ammar

# Strategies to deal with Mislabeled Data

Advisor: Prof. Moonsoo Kang, PhD

A thesis submitted in partial fulfillment of the  
requirements for a Master's degree




October 2017

Graduate School of Chosun University

Department of Computer Engineering

Malik Muhammad Ammar

# 마릭 무하마드 아마르의 석사학위논문을 인준함

위원장	조선대학교 교수	<u>양희덕</u>	
위원	조선대학교 교수	<u>강문수</u>	
위원	조선대학교 교수	<u>정호엽</u>	

2017년 11월

조선대학교 대학원

## TABLE OF CONTENTS

ABSTRACT .....	v
한 글 요약.....	vi
I. INTRODUCTION.....	1
A. Motivation.....	1
B. Contributions.....	1
C. Thesis Layout.....	2
II. RELATED WORKS .....	3
A. Algorithmic Level Approaches.....	3
B. Data Level Approaches .....	4
III. CLUSTERING BASED LABEL NOISE CLEANING .....	6
A. Introduction.....	6
B. K-means Clustering .....	6
C. Mixture Models.....	7
D. Hierarchical Clustering.....	8
E. Methodology .....	8
F. Datasets.....	10
G. Experiments and Results .....	11
H. Conclusion .....	14
IV. SIMILARITY BASED CLEANING OF LABEL NOISE.....	15
A. Introduction.....	15
B. Support vector machines (SVMs) for label noise cleaning .....	16
C. Proposed Idea.....	19
D. Datasets.....	20

E.	Experimental Setup .....	21
F.	Data Preprocessing .....	22
G.	Experiments .....	23
1.	Experiment 1.....	23
2.	Experiment 2.....	24
3.	Experiment 3.....	25
H.	Conclusion .....	26
V.	IMPROVED LABEL NOISE FILTER.....	28
A.	Introduction.....	28
B.	Precision and Recall .....	29
C.	Majority Filter .....	31
D.	Consensus Filter .....	32
E.	Potential Problems with MF and CF .....	33
F.	Proposed Approach .....	35
G.	Datasets and Experimental Setup.....	38
H.	Results .....	38
I.	Conclusion .....	40
VI.	CONCLUSIONS.....	42
	BIBLIOGRAPHY .....	44
	ACKNOWLEDGEMENTS .....	46

## LIST OF FIGURES

Figure 1: Illustration of the clustering based label noise removal method .....	9
Figure 2: A Dummy Dataset with original class labels.....	17
Figure 3: Dataset with label noise added.....	17
Figure 4: Noisy dataset after passing through an SVM classifier.....	18
Figure 5: Noise removal performance for difference percentages of SV reviewed.....	27
Figure 6: Illustration of Precision and Recall .....	31
Figure 7: Illustration of Majority / Consensus Filter .....	34
Figure 8: Illustration of Proposed Improved Majority Filter.....	36
Figure 9: Performance comparison of MF, CF and IMF in terms of time taken.	41



## LIST OF TABLES

Table 1: Average noise removal performance. ....	12
Table 2: Average noise removal performance of CBNR and ALNR on ADNI mean cortical thickness data.....	13
Table 3: Level of noise before and after the experiments. ....	13
Table 4: Classification accuracy after label cleaning. ....	14
Table 5: Algorithm of Proposed Approach.....	20
Table 6: Number of Support Vectors captured for MNIST Digit Dataset with and without feature extraction.....	22
Table 7: No. of Support Vectors captured for varying level of label noise.....	23
Table 8: Noise removal performance comparison. ....	25
Table 9: Comparison of number of examples reviewed.....	26
Table 10: Algorithm of majority filter .....	32
Table 11: Algorithm of consensus filter.....	33
Table 12: Algorithm of improved majority filter.....	37
Table 13: Performance comparison for MNIST dataset.. ....	39
Table 14: Performance comparison for UCI dataset.....	39
Table 15: Performance comparison for Wine quality dataset.. ....	40
Table 16: Performance comparison for Wisconsin Breast Cancer dataset.....	40

## ABSTRACT

### Strategies to deal with Mislabeled Data

Malik Muhammad Ammar

Advisor: Prof. Moonsoo Kang, Ph.D.

Department of Computer Engineering

Graduate School of Chosun University

Performance of machine learning classifiers is heavily dependent on labeling quality of datasets. Generally, human supervision is required for the labeling of instances in datasets. This labeling can be erroneous, and detecting such erroneous examples from the dataset is extremely important. In this work we discuss some of the machine learning approaches to deal with the problem of label noise in datasets. The experiments are conducted on some of the widely used datasets in the machine learning community. Firstly, a clustering based technique for relabeling of instances in datasets is studied. Secondly, a similarity based technique that utilizes the concept of Euclidean distance for cleaning of label noise. The instances having similar scores with positive and negative classes are selected for expert review. Lastly, an improved majority filter is proposed. Our experiments show that the improved majority filter is faster as compared to the conventional majority filter. We also compare the performance of proposed method with majority and consensus filter in terms of *precision*, *recall* and  $F_1$ Score.

## 한 글 요약

### 기계학습의 저확도 향상을 위한 레이블 노이즈 제거 알고리즘

마릭 무하마드 아마르

지도 교수: 강문수

컴퓨터공학과

대학원, 조선대학교

머신러닝을 위한 분류기 학습 데이터에서 각 데이터의 클래스가 항상 정확할 수 없기 때문에 기계학습데이터 레이블링에 오류가 포함될 가능성이 높다. 예를 들어 의학 자동 진단 분야에서, 질병의 분류 및 진단에 대한 오류가 포함될 가능성이 항상 존재한다. 기계학습 알고리즘은 입력 데이터의 클래스 레이블링 정확도에 많은 영향을 받기 때문에, 분류기의 성능은 잠재적인 오류들이 포함된 데이터들에 의해 결정이 된다. 본 논문에서는 기계학습 데이터에 오류가 존재할 때, 이 오류를 인지하고, 제거하는 알고리즘을 제시한다. 이러한 오류 데이터들의 대부분은 기계학습에서 사용되는 분류기에 의해 명확하게 구분되지 않는 구간에 대부분 존재한다는 것에 착안하여, 기계학습에 가장 많이 사용되는 SVM 분류기를 기준으로 학습데이터의 유클리안 위치를 이용하여 오류가 포함되었을 가능성이 높은 데이터를 인지하는 방법과, 이와 반대로 SVM 분류기에서 멀리 떨어져, 오류가 발생하지 않았을 가능성이 높은 데이터를 활용하여 오류가 포함되었을 가능성이 높은 데이터를 다시 레이블링하는 두가지 종류의 알고리즘을 제시하였다. 제안된 방법들 여러 가지 종류의 데이터를 이용하여 효율적으로 레이블링 에러를 제거할 수 있다는 것을 검증하였다.

# I. INTRODUCTION

## A. Motivation

The performance of classifiers in supervised learning setting is heavily dependent on accurate labelling of training data. Mislabeled examples in datasets can severely degrade the performance of classifiers. The labelling error in datasets is mainly because mostly human supervision is required for the labelling of datasets. And this labelling can be erroneous. One another possible reason for the presence of mislabeled examples in datasets is that an adversary can try to change the labels of examples in a dataset for his own benefit. Therefore, identification of such examples in the datasets is an important step before training the classifiers.

Noise in explanatory variables i.e. features has been widely studied in literature, but noise in response variables i.e. labels has received relatively less attention. This work focuses on latter category i.e. noise in response variables or labels.

In the literature, there are two widely used options to deal with the problem of label noise. 1) creating algorithms that are inherently robust to the label noise, 2) creating algorithms for correcting or filtering out the potentially mislabeled examples. We focus on the second approach since the first approach does not actually solve the problem of noise in labels. That means the noise is still present in the datasets. The better approach is to target the potentially noisy examples and either remove them from the dataset or correct them.

## B. Contributions

In this work, the problem of noise in class labels is studied. We propose an improved filter for identifying the noisy examples in datasets. The proposed

method utilizes the concept of Support Vector Machines (SVM) for the capturing of noisy examples and later using ensemble learning methods to identify the noisy examples. These identified potentially noisy examples can either be eliminated from the datasets or can be reviewed by an expert.

In addition to this we also discuss some of the other methods and their weaknesses and advantages that were studied during the preparation of this thesis. These include the noise removal using clustering and noise removal using distance measures.

### **C. Thesis Layout**

The thesis is organized as follows. In Chapter II, we discuss the previous work related to label noise. In Chapter III, we discuss a clustering based technique for the removal of label noise in datasets. We also discuss the potential weakness of the technique. In Chapter IV, we discuss a noise removal technique based on distance measure (e.g. Euclidean distance). In Chapter V, we introduce our proposed improved majority filter for the label noise reduction from datasets. Finally, the thesis is concluded in Chapter VI.

## II. RELATED WORKS

The performance of the classifier trained on mislabeled examples is much poorer as compared to the one being trained on accurately labeled examples. Also, the results of the classifiers trained on poorly labeled examples will not be reliable as the confidence of the labels is very low.

Although, the problem of label noise has received relatively less attention as compared to the problem of feature noise, but still several methods and approaches have been proposed to deal with label noise. In this section, we look at some the approaches proposed in the literature to deal with class label noise.

To deal with the problem of class label noise, mainly two kinds of approaches have been proposed in the literature.

- 1) Algorithmic level approach
- 2) Data level approach

A brief overview of both these approaches is given below:

### A. Algorithmic Level Approaches

Algorithmic level approaches mainly rely on creation of algorithms that are inherently robust to the class label noises. The main objective of such approaches is to mitigate the effect of the incorrectly labeled examples. Some of the approaches related to this category are discussed below.

In [1] a label-noise robust extension of the widely used Bayesian logistic regression classifier is proposed and in [2] the same approach is used for the classification of

mislabeled microarrays. In [3] a generalized label noise model, which can resist the negative effects of labeling errors is proposed for classification in presence of labeling or annotation errors.

In [4] a noise tolerant method has been proposed to modify the decision tree classifier to make it robust to label noise. In [5] *noise-tolerant Occam algorithms*, are proposed to build effective robust learning algorithms. They have used the idea of boosting for limiting the sensitivity of classifier to the class label noise.

The main problem with such approaches that although they try to mitigate the effect of noisy examples, but noisy examples are still present in dataset. Therefore, as the level of noise increases the performance of the proposed approaches degrades quickly. Secondly, not all the machine learning algorithms can be modified to be robust to the label noise, hence such approaches are not applicable for all the datasets.

## **B. Data Level Approaches**

Data level approaches mainly rely on either correcting or removing the potential noisy examples from the datasets. Some of such approaches are discussed in the following paragraphs.

In [6], noise in attributes or features as well as noise in class labels is studied and a technique for replacing the erroneous values with appropriate values is proposed. In [7], *Self-Training Correction (STC)* and *Cluster-based Correction (CC)* algorithms are proposed to correct the label noise in datasets. In [8], authors investigate the correction of mislabelled examples in context of crowdsourcing. In [9] and [10] authors have proposed techniques which utilize the idea of support vectors for targeting the noisy examples and later a human expert reviews the potentially noisy examples selected by the algorithms.

The approaches mentioned above try to correct the labels of the examples in datasets and then use all the examples in the datasets for the training process. But in real world scenarios such corrections can be challengeable. Therefore, in literature another method to deal with class label noise is mislabelled filtering. These approaches try to find the potentially noisy examples in the dataset and remove them from the dataset. Hence, rest of examples will be used for training the classifiers.

Ensemble learning methods are widely used for mislabelled filtering. Majority filtering (MF) and consensus filtering (CF) are widely used algorithms proposed in [11] and [12]. In MF if majority of the classifiers incorrectly classify an example then that example is labelled as potentially noisy. Similarly, in CF if all of the classifiers fail to classify an example correctly then that example is marked as potentially noisy.

A disadvantage of filtering methods is that they can make two types of errors. 1) a clean example incorrectly marked as noisy and removed from dataset, 2) a mislabelled example being retained in the training dataset. Type 1 error reduces the number of correctly labelled examples, which can be problematic if the number of examples in dataset is very less and it can also cause overfitting. Whereas, the 2<sup>nd</sup> type of error is harmful because if the mislabelled examples are retained in the dataset then it can negatively affect the performance of a classifier.



### III. CLUSTERING BASED LABEL NOISE CLEANING

#### A. Introduction

The samples from same class in a dataset tend to lie close to each other in form of groups or clusters. Clustering analysis can help find these clusters in datasets. Therefore, by comparing the label of the sample with the labels of the samples in the same cluster, the sample can be tagged as mislabeled or clean. In [17], a technique based on this idea is proposed for removing or relabeling the potentially mislabeled samples from dataset.

In [17], only nearest neighbor based technique is considered, therefore it was interesting point to investigate the performance of clustering techniques like *Kmeans*, *mixture models* and *hierarchical clustering*, which are some of the widely used clustering techniques, using the similar idea. The performance is then compared with a newly proposed approach to address the problem of label noise in [10].

This served as a stepping stone to the research conducted in this thesis and helped us coming up new ideas which are discussed in subsequent chapters

A brief description of the clustering techniques used in experiments is given below.

#### B. *K*-means Clustering

If the given dataset is represented by  $(x_1, \dots, x_n)$ , consisting of  $n$  number of examples, the objective is to group the given dataset into  $K$  number of clusters. A cluster can be considered as a group consisting of data point with less inter-point distances in comparison to the points outside that group. Suppose that the cluster centroids, which represent our present guess for the positions of the centers of the

clusters, denoted by  $(\mu_1, \dots, \mu_k)$ . Also assume  $C$  represents the cluster assignments. Then the  $k$ -means algorithm is as follows:

1. Initialize cluster centroids  $\mu_1, \dots, \mu_k$  randomly.
2. Repeat until convergence: {  
For every  $i$ , set

$$C_i = \arg \max_j \|x_i - u_j\|^2 \quad (1)$$

For every  $j$ , set

$$\mu_j := \frac{\sum_{i=1}^n x_i \forall C_i=j}{|C_i=j|} \quad (2)$$

Where  $|\cdot|$  represents cardinality i.e. no. of elements in the set }.

## C. Mixture Models

Let us consider a mixture model consisting of unsupervised structure of  $K$  clusters, represented by random discrete variable  $C$ . We assume that, as in standard mixture model the data  $(x_1, \dots, x_n)$  are the independent realization of a random vector  $X \in \mathbb{R}^p$  with density function:

$$p(x) = \sum_{i=1}^K P(C = i)p(x|C = i) \quad (3)$$

Where  $P(C = i)$  is the prior probability of  $i$ th cluster and  $p(x|C = i)$  is the corresponding conditional density. Using classical notation of mixture models, the prior probability  $P(C = i)$  can be represented by  $\pi_i$  and in the case of Gaussian mixture model, the conditional density  $p(x|C = i)$  is modelled by a Gaussian density  $\phi$  with mean  $\mu_i$  and covariance  $\Sigma_i$ . So, equation (3) can be written as:

$$p(x) = \sum_{i=1}^K \pi_i \phi(x; \mu_i; \Sigma_i) \quad (4)$$

So, the following Bayes' rule can be used to find the cluster assignments:

$$P(C = i|X = x) = \frac{P(C=i)p(x|C=i)}{p(x)} \quad (5)$$

Expectation Maximization (EM) algorithm proposed in [15] can be used to estimate the parameters  $\mu_i$  and  $\Sigma_i$  for the Gaussian models. The EM algorithm can provide the Maximum Likelihood (ML) estimates using an iterative process.

## D. Hierarchical Clustering

In hierarchical clustering, the data is grouped over a variety of scales by creating a dendrogram. In agglomerative hierarchical clustering, each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. For deciding the clusters that should be combined a method of dissimilarity between sets of observations is required. This can be done by computing distance measures like *euclidean*, *minkowski*, *mahalanobis*, *chebychev* etc. between pair of observations. Hierarchical clustering also requires a linkage criterion that determines the distance between the sets of observations as a function of the pair-wise distances between observations. Some of the commonly used linkage criteria include *complete-linkage*, *average-linkage*, *centroid-linkage*, and *Ward's linkage*.

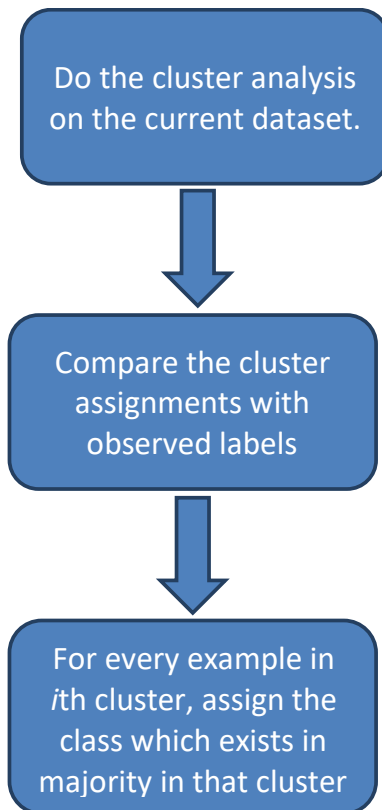
## E. Methodology

Once the cluster assignments are made for all the examples in data, these cluster assignments are then compared with the observed labels of the examples (supervised information). The examples belonging to  $i$ th cluster will be assigned the class or label to which majority of these examples belong in the observed data. The intuition behind this approach is that if less than 50% examples of belonging to the class are mislabeled then majority of the labels are correct always. So, the similar examples will be grouped together in a same cluster. Hence, by assigning the majority class to the examples in a cluster, the potentially mislabeled examples

will be relabeled to the correct class. To mathematically define the process, assume that the observed labels of the examples in data are represented by  $(y_1, \dots, y_n)$ . And the labels assigned after the comparisons are represented by  $(\hat{y}_1, \dots, \hat{y}_n)$ . So, the new label assignments will be according to the following equation:

$$\hat{y}_a = \arg \max y_a \tag{6}$$

where  $a$  represents a vector containing indices of examples belonging to  $i$ th cluster. The proposed method is illustrated in Fig. 1.



**Figure 1: Illustration of the clustering based label noise removal method**

## F. Datasets

To evaluate the performance wine quality dataset, Wisconsin breast cancer dataset and iris flower dataset were obtained from UCI machine learning repository. The wine quality dataset has 1599 examples of *Red Wine* class and 4898 examples of *White Wine* class. Each example is represented by 12-dimensional feature vector representing physicochemical properties of the red and white wines. The 12th feature is the quality score computed using the 11 physicochemical properties. Therefore, in our experiments the 12<sup>th</sup> feature is not used. We randomly selected 500 examples from each class for our experiments. The Wisconsin Breast cancer dataset includes 357 *malignant* class and 212 *benign* class examples. Each example is represented by a 30-dimensional feature vector. For the experiments, 200 examples from each class were selected randomly. To evaluate the performance of the proposed approach on multiclass problems, iris flower dataset was used. Which includes 50 examples from each of three iris species, i.e., *setosa*, *virginica*, and *versicolor*. Each example is represented by four features, i.e. the length and width of the sepals and petals.

At the end, we evaluate the performance of the clustering based approach for a real-life problem i.e. diagnosis of Alzheimer's disease. The data is downloaded from (<https://adni.loni.usc.edu/>)<sup>1</sup>. For our experiments, we have used *the AD Challenge Training Data: Imaging files* available under Test Data in Download Study Data section, on the website. All data is derived from the ADNI1: Complete 1Yr 1.5T standardized MRI collection (<http://adni.loni.usc.edu/methods/mri-analysis/adnistandardized-data/>). All source imaging data consist of 1.5 Tesla T1-

---

<sup>1</sup> Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database ([adni.loni.usc.edu](http://adni.loni.usc.edu)). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: [http://adni.loni.usc.edu/wp-content/uploads/how\\_to\\_apply/ADNI\\_Acknowledgement\\_List.pdf](http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf)

weighted magnetic resonance image (MRI) volumes in the NIfTI (.nii.gz) format from the ADNI1:Complete 1Yr 1.5T Data Collection. All images were processed using three neuroimaging software pipelines: 1) FreeSurfer, 2) Advanced Normalization Tools (ANTs), and 3) Mindboggle. The resulting data consists of tables consist of morphometric (shape) data derived from the images. The complete data consists of 133 *Alzheimer's disease (AD) subjects*, 190 *Healthy Control (CN) subjects*, and 305 *Late Mild Cognitive Impairment (LMCI) subjects*. In this work, FreeSurfer mean cortical thickness data and Mini-Mental State Examination scores were used for AD and CN subjects. Total of 323 subjects were included in the experiments of which 133 were AD and remaining 190 were CN subjects.

## G. Experiments and Results

We ran our experiments by manually adding the label noise in the data. The level of noise was increased from 10% to 40% with the interval of 10%. For each experiment, same proportion of examples were randomly selected, and the labels of these randomly selected examples were flipped. For example, the wine quality dataset has 1000 examples, so at 10% noise level 100 examples will be selected randomly (50 from each class) and their labels will be flipped. We repeated our experiments 30 times to avoid any bias due to selected examples, and average performance of these 30 experiments is reported. For all the clustering algorithms, we assume that number of clusters present in the data is equal to the number of classes in the dataset. For the case of mixture models, we assume that the data comes from mixture of gaussians, and for our experiments we have considered the covariance matrix to be diagonal. For hierarchical clustering, we have used agglomerative hierarchical clustering. The linkage criteria used is *Ward's Method* which uses inner square distance (minimum variance algorithm) and the distance

metric used is *Euclidean distance*. All the experiments are performed using *Statistics and Machine Learning Toolbox* MATLAB.

We refer to the clustering based method as CBNR (Clustering Based Noise Removal). In the tables *CBNR-GMM* indicates that clustering method used is mixture of gaussians, and *CBNR-Kmeans* and *CBNR-Hier* indicate k-means and hierarchical clustering respectively. The results of the experiments are reported in Table 1.

**Table 1: Average noise removal performance. The results are in percentage of noise examples successfully relabeled to correct labels versus total number of noise examples in the dataset.**

Noise Level (%)	CBNR-GMM	CBNR-Kmeans	CBNR-Hier
<b>Wine Quality</b>			
10	<b>96.13</b>	84.47	86.77
20	<b>96.07</b>	84.48	86.85
30	<b>96.19</b>	84.78	86.87
40	<b>96.14</b>	85.15	86.47
<b>Breast Cancer</b>			
10	<b>87.44</b>	79.77	75.78
20	<b>90.11</b>	78.78	76.33
30	<b>89.00</b>	78.26	76.52
40	<b>88.33</b>	79.70	76.33
<b>Iris Flower</b>			
10	<b>91.56</b>	91.33	87.11
20	<b>91.44</b>	88.89	88.11
30	<b>90.22</b>	87.92	86.63
40	<b>89.61</b>	87.89	89.56

Table 1 shows that It can also be seen that the proposed method performs much better at higher noise levels. By on average reducing the noise level by 78.73% when initially 40% of examples were incorrectly labelled.

Finally, the performance is compared with the recently proposed label noise cleaning method [10], called as ALNR by the authors on the ADNI mean cortical thickness data for diagnosis of Alzheimer's disease. Based on the results from Table 1 we only use CBNR-GMM for comparison with ALNR. The results of this experiment are shown in Table 2. In Table 3 the performance of our method in terms of noise present before and after the experiments in data on ADNI data is shown.

**Table 2: Average noise removal performance of CBNR and ALNR on ADNI mean cortical thickness data. The results in percentage of noise examples successfully relabeled to correct labels versus total number of noise examples in the dataset.**

Noise Level (%)	CBNR-GMM	ALNR
10	<b>95.63</b>	93.02
20	<b>92.77</b>	90.72
30	<b>95.74</b>	84.67
40	<b>98.45</b>	78.19

**Table 3: Level of noise before and after the experiments.**

Start Noise Level (%)	CBNR-GMM	ALNR
10	5.79	<b>0.5</b>
20	4.66	<b>1.41</b>
30	4.66	<b>3.01</b>
40	<b>4.66</b>	6.14

As evident from Table 3, that ALNR tends to perform better in terms of ending noise level for low noise levels, and its performance starts to decrease with increase in noise level. An important thing to mention here is that the better performance of ALNR is due to the involvement of the human expert, whereas in our method no human expert is needed. And in Table 4, we show that learning methods like Support Vector Machine (SVM) are robust enough to deal with this small amount of noise remaining in the dataset after cleaning. We used 5-fold cross validation to



test the classification performance after cleaning of dataset. Classification performance of *CBNR-GMM* is shown only.

**Table 4: Classification accuracy after label cleaning.**

Noise Level (%)	Classification Rate (%) <i>CBNR-GMM</i>
10	95.98
20	95.98
30	95.98
40	95.98

## H. Conclusion

We investigated the performance of different clustering techniques to clean the potentially mislabeled examples from the dataset. We have shown that by using unsupervised learning methods for clustering and then by comparing those cluster assignments with the observed labels, most of the label noise can be eliminated from data. Furthermore, it requires no human expert to review the examples.

The clustering based technique has few disadvantages too. For example, as mentioned earlier, that it is prone to change the labels of few correctly labeled examples to the incorrect labels. Another disadvantage is that it can only work for the datasets which are *cluster-able*. That means the technique will not work on datasets which do not have separable classes.

## IV. SIMILARITY BASED CLEANING OF LABEL NOISE

### A. Introduction

In a very recent work R. Ekambaram et. al [10] proposed an active cleaning of label noise approach, in which they show that mislabeled examples tend to occur as support vectors and by reviewing a subset of these support vectors label noise can be cleaned. Their proposed method shows excellent results in terms of cleaning the label noise, but the problem with their method is that it is iterative in nature, which means more than 1 iterations are needed to clean the label noise, and the criteria to review the example is based on the misclassification probability of support vector examples when tested using a classifier created by non-support vector examples. The problem with this approach is that the number of examples that are selected for review in each iteration depends on the level noise, this implies that there should be some prior knowledge about the noise level in the datasets. But as in the many practical scenarios this information may not be given. So, our proposed method not only cleans the dataset in only 1 iteration but also defines a clear strategy to select the examples for expert review. Our review strategy is based on the concept of similarity measure, where the support vector examples are assigned the labels to which they have more similarity and the examples that are similar to both the labels are selected for expert review.

## B. Support vector machines (SVMs) for label noise cleaning

As shown in [10], the dual form of optimization problem created by an SVM is usually solved due to its efficiency for high dimensional features, and due to the fact that solution can easily be obtained using kernel trick. The dual form of optimization problem is formulated as below

$$\begin{aligned} \max_c \quad & \sum_{i=1}^N c_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j c_i c_j K(x_i x_j) \\ \text{s. t. } \quad & c \geq 0 \\ & \sum y_i c_i = 0 \end{aligned} \tag{7}$$

where  $N$  denotes the number of training examples,  $y_i \in [-1,1]$  represents the class labels,  $x_j$  is  $d$ -dimensional example,  $K(\cdot)$  represents the kernel and  $c_i$  is a Lagrange multiplier. Now the training examples that are needed to create the decision boundary are support vector examples, i.e. examples for which  $c_i > 0$ . This fact is useful because very small proportion of training examples will  $c_i > 0$ . So essentially only support vector examples are needed to make predictions on new test examples [13]. This also implies that since support vectors examples are responsible for defining the decision boundary, so the labels of such examples become very important, rest of the labels are irrelevant as they do not participate in decision making process. And most of the time the label noise occurs to the examples which are hard to label in a sense that they are borderline examples. Also, if an outsider wants to affect the decision-making process for his advantage, they will also focus on such examples. Hence to reduce the label noise from datasets we only need to focus on labels of support vector examples. Getting a human expert for validating these examples can help in solving this problem.

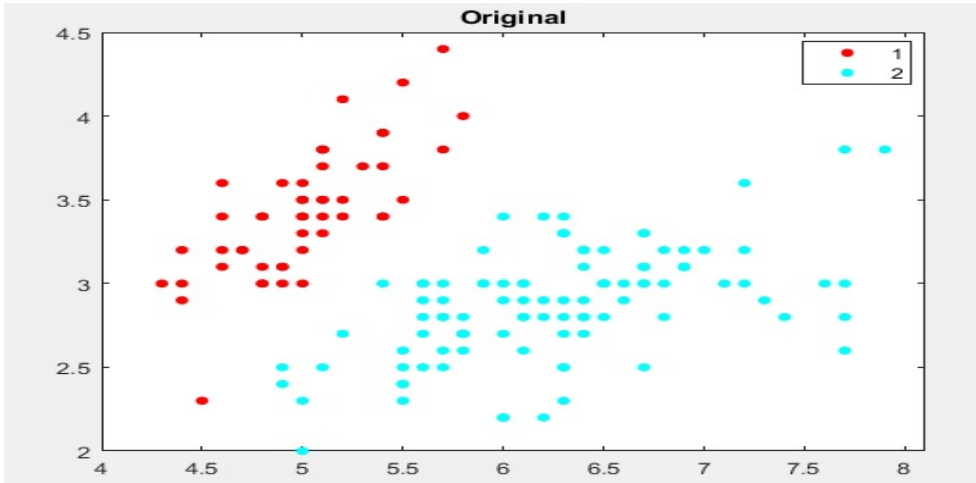


Figure 2: A Dummy Dataset with original class labels

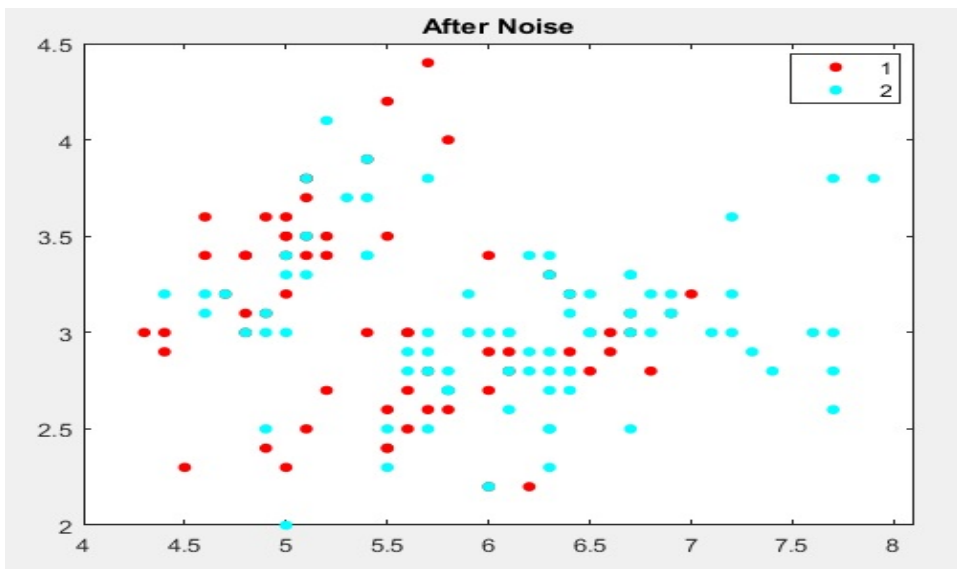
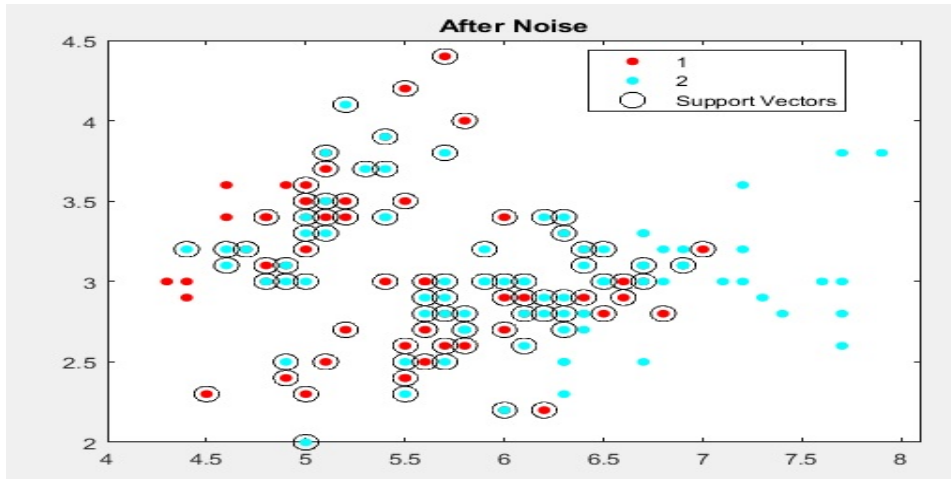


Figure 3: Dataset with label noise added



**Figure 4: Noisy dataset after passing through an SVM classifier**

To better visualize the ability of SVMs to capture the label noise examples, look at the Figures 2,3 and 4. In Fig. 2, we can see a dummy dataset consisting of 2 classes. In Fig. 3, the same dataset is corrupted with label noise. Labels of the 40% of the examples from both the classes are flipped. In Fig. 4, the corrupted dataset is passed through an SVM classifier. We can see that all the noisy examples are captured as support vectors.

Based on this ability of SVMs, in [10] a strategy to select a subset of these support vector examples for an expert review is proposed. The potential problems with their proposed strategy were discussed in introduction section of this chapter.

## C. Proposed Idea

In this section, we propose a new strategy, to select a subset of support vector examples for a human expert to review. The proposed strategy needs only a single iteration and automatically decides the number of examples to review regardless of the level of noise, unlike [10], where the review strategy needs to be reviewed for varying level of noise.

We hypothesized based on our experiments that since most of the examples that are not selected as support vectors are noise free, we can measure the similarity of the support vector examples with the mean representation of positive and negative class of the non-support vector examples. The noisy examples will generally have almost same similarity score for both the classes as they are borderline examples. So, by selecting only such examples for expert review most of the label noise can be removed from the datasets. Our experiments show that selecting half of the support vector examples based on this criterion is enough to remove more than 90% of the label noise from datasets for every noise level. The detailed steps of our proposed method are described in Table 1.

Although various similarity measures like *Euclidean distance*, *Minkowski distance*, *Chebyshev distance*, *Tanimoto distance*, *cosin distance* etc. exist [14], we have used *Euclidean distance* measure here due to its simplicity. The *Euclidean distance* between two points is defined as follows:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (\mathbf{q}_i - \mathbf{p}_i)^2} \quad (8)$$

where  $p = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$  and  $q = (\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n)$ , are two points in Euclidean  $n$ -space, and  $d$  is the distance between two points.

**Table 5: Algorithm of Proposed Approach**

Algorithm
<ol style="list-style-type: none"> <li>1. Create a SVM classifier using all the examples in dataset.</li> <li>2. Separate the SV and NonSV examples from the dataset.</li> <li>3. Create a mean representation of examples belonging to positive and negative class from Non_SV_Set.</li> <li>4. Measure the Euclidean distance of every example in SV_Set with mean representation of positive and negative examples in Non_SV_Set.</li> <li>5. Measure the similarity score by subtracting the score obtained in previous step for positive and negative class for every example.</li> <li>6. Sort the examples based on the distances obtained in previous step in ascending order.</li> <li>7. Select Top Half (i.e. 50%) of the examples for an expert review from the SV_Set based on the sorted scores in step 6.</li> <li>8. For rest of the examples, make the decision in favor of the class with which the distance is lesser.</li> </ol>

## D. Datasets

In addition to the wine quality dataset and Wisconsin breast cancer data used in the previous chapter, here we also use the MNIST handwritten digit recognition dataset and UCI letter recognition dataset. The MNIST handwritten digit recognition dataset consists of 60000 samples obtained from approximately 250 writers, representing digits 0 – 9. Each digit is represented with 784-dimension feature vector. In [15] it is stated that the digits 4,7 and 9 had the highest misclassification rate. So, we randomly selected 1000 samples for each of these digits for our experiments. The UCI letter recognition dataset consists of 20000 samples representing letter A-Z by 16-dimensional feature vector. [10] claimed that digits

$H$ ,  $B$  and  $R$  are most likely to be confused so we randomly selected 500 samples for each of these digits.

## E. Experimental Setup

The Wisconsin breast cancer dataset and the wine quality dataset have only 2 classes but the UCI letter recognition and the MNIST digit dataset have 3 classes each. So, for every experiment these two datasets were divided in two classes. For example, when using MNIST dataset, first time digit 4 was considered as class 1 and half of the examples for digits 7 and 9 were considered as class 2, second time digit 7 was considered as class 1 and half of the examples for digits 4 and 9 were considered as class 2, and the third-time digit 9 was considered as class 1 and half of the examples for digits 4 and 7 were considered as class 2. Same approach was used for UCI letter dataset as well. So, for these two datasets every experiment had to be run thrice.

All the experiments were performed on an Intel® Core™ i5-4590 CPU @ 3.30 GHz machine with 8.00 GB memory, using *Pattern Recognition Toolbox*, MATLAB.

To test the performance of the proposed approach we introduced the noise in datasets by randomly picking the examples and flipping their labels. For example, for 10% noise level on MNIST dataset we randomly select 10% of the class 1 examples (100 examples) and flip their labels to class 2 and similarly we select 10% of the class 2 examples (50 examples for each of the two digits) and flip their labels to class 1. Same procedure was applied for rest of the datasets as well.

To avoid bias due to the random selection of examples for flipping of labels, every experiment was performed 30 times. So, for Wisconsin breast cancer and the wine quality datasets every experiment was performed 30 times and for the MNIST digit



and UCI letter recognition datasets the experiments were performed 90 times (30 repetitions x 3 experiments), and average results of these experiments are reported.

## F. Data Preprocessing

All the features in all the datasets were scaled between -1 and 1. Apart from the MNIST digit dataset, rest of the datasets have relatively very less features, but MNIST dataset consists of 784-dimensional features vectors. Our experiments (Table 6.) show that feature extraction can improve the capturing of noisy examples as support vectors on MNIST dataset. So, for feature extraction we trained an autoencoder with 20 neurons in the hidden layer. So, to find the support vector examples, each example in the dataset was represented with 20-dimensional feature vector rather 784 features per example.

**Table 6: Number of Support Vectors captured for MNIST Digit Dataset with and without feature extraction. FE: Feature Extraction**

Noise (%)	Support Vector Examples (%)		Noisy Support Vector Examples (%)	
	Without FE	With FE	Without FE	With FE
10	40.26	47.23	97.31	99.11
20	57.83	63.97	96.64	99.54
30	71.20	78.14	94.91	99.69
40	79.91	89.94	90.49	99.73
50	89.66	96.96	82.63	96.62

## G. Experiments

### 1. Experiment 1

In our first set of experiments we tried to establish the relationship between the level of noise and the number of support vector examples captured. The results of these experiments are shown in Table 7. It can be seen from the results that as the level of noise is increased the number of

**Table 7: Number of Support Vectors captured for varying level of label noise**

Noise (%)	Support Vector Examples (%)				Noisy Support Vector Examples (%)			
	MNIST	UCI	Wine	Cancer	MNIST	UCI	Wine	Cancer
10	47.23	50.00	31.16	42.22	99.11	98.07	99.97	99.50
20	63.97	65.75	51.58	61.97	99.54	98.13	99.94	99.58
30	78.14	79.22	70.43	77.05	99.69	95.53	99.94	99.50
40	89.84	90.03	86.47	88.81	99.73	98.85	99.98	98.50
50	96.96	95.49	96.92	92.60	96.62	95.49	97.13	92.70

support vector examples also increase drastically. And for all datasets at 50% noise level the more than 90% of the examples were selected as support vectors. It can also be seen that for all datasets approximately 98%-99% of label noise examples were captured inside support vector examples for noise levels of 10%-40%. This shows the importance of reviewing support vectors in cleaning the label noise from datasets. But this also leads to a problem that reviewing such a huge number of examples becomes

tedious job for the reviewer. This problem is addressed in subsequent experiments.

## 2. Experiment 2

As evident from the results of Experiment 1 we need a strategy to select only a subset of the support vector examples that are most likely to be mislabeled. Here we perform the experiments using the strategy proposed in Table 6. But here we select the varying proportions of top most similar support vector examples for review to see the effect of varying the number of examples reviewed with noise cleaning performance. The results of these experiments are shown in Fig. 5.

The results clearly show that reviewing more examples leads to better noise removal performance. But an interesting result that can be observed is that by reviewing 50% of the total support vector examples for all noise levels and for all datasets more than 90% of the label noise present in datasets can be cleaned. This indicates that for lesser noise levels, the number of examples to be reviewed will be lesser because of lesser number of support vector examples, as evident from Table 7. And for higher noise levels more examples need to be reviewed. Since at 50% noise level, half of the examples in the dataset are labeled incorrectly, therefore the probability of every example to be labeled correct is just 0.5. In this case, it makes sense to have an expert validate more than 50% examples in the dataset. The major contribution of this result is that for varying levels of noise, there is no need to define separate strategies as in [9] and [10]. As evident from Fig. 4., reviewing half of the support vector examples is enough and our experiments show that it can produce more or less similar results as compared to the other approaches.

### 3. Experiment 3

In this set of experiments, we compare the noise removal performance of method proposed by us (referred to as SLNR [Similarity based Label Noise Removal]) with the method proposed in [10] (referred to as ALNR [Active Label Noise Removal]). The performance is also compared in terms of number of examples to be reviewed in each experiment.

**Table 8: Noise removal performance comparison. All results are in percentage of number of correctly labeled examples at the end of experiment versus total examples in datasets.**

Noise Level (%)	Noise Removed (%)							
	MNIST		UCI		Wine		Cancer	
	SLNR	ALNR	SLNR	ALNR	SLNR	ALNR	SLNR	ALNR
10	<b>96.39</b>	94.08	<b>93.59</b>	90.48	<b>99.90</b>	99.17	<b>98.78</b>	96.00
20	<b>96.63</b>	94.63	<b>92.71</b>	90.77	<b>99.90</b>	98.77	<b>98.97</b>	95.67
30	<b>96.74</b>	94.69	<b>93.00</b>	90.80	<b>99.90</b>	99.00	<b>98.85</b>	96.00
40	<b>96.31</b>	95.12	<b>93.43</b>	91.02	<b>99.92</b>	99.19	<b>98.35</b>	95.50

It is evident from Table 8, that SLNR performs better than ALNR to relabel the noisy examples for all the datasets at all noise levels. In terms of total number of examples needed for expert review (Table 9), ALNR performs better at lower noise levels (10% - 20%) for all except UCI letter recognition dataset. Whereas for the higher noise levels (30% - 40%) SLNR performs better all datasets except the wine quality dataset. SLNR requires more examples to be reviewed in case of lower noise levels because the criterion for selecting the examples to review in our case is to select the half of the support vector examples regardless of the noise level. So, even though the number of examples to be reviewed for less noise level is more

than ALNR but the review criterion is same for all the noise level which can useful when the noise level is not known.

**Table 9: Comparison of number of examples reviewed. All results are in percentage of number of examples reviewed versus total examples in datasets.**

Noise Level (%)	Examples Reviewed (%)							
	MNIST		UCI		Wine		Cancer	
	SLNR	ALNR	SLNR	ALNR	SLNR	ALNR	SLNR	ALNR
10	23.61	<b>15.82</b>	25.00	<b>19.58</b>	15.58	<b>12.90</b>	20.61	<b>13.77</b>
20	31.99	<b>26.20</b>	26.30	<b>28.83</b>	25.92	<b>22.85</b>	30.98	<b>23.58</b>
30	<b>38.07</b>	38.28	<b>39.61</b>	39.72	<b>35.22</b>	35.63	38.53	<b>34.43</b>
40	<b>44.92</b>	53.44	<b>45.02</b>	48.56	<b>43.24</b>	45.89	<b>44.41</b>	46.45

## H. Conclusion

In this chapter, a novel method to remove label noise from datasets is proposed. The method utilizes the concept of similarity score to assign the classes to the examples with doubtful labels. The examples which are closer to positive class are assigned positive class, whereas the ones nearer to the negative class are assigned negative class. Examples having similar score for positive and negative classes are selected for expert review.

An important contribution of the proposed work is the well-defined criterion for the selecting the examples for expert review. The criterion remains same in presence of varying levels of noise. This can be helpful when trying to clean the datasets quickly when no prior information about possible noise level is available. Also, there is no need for multiple iterations. The proposed method needs just iteration for this purpose.

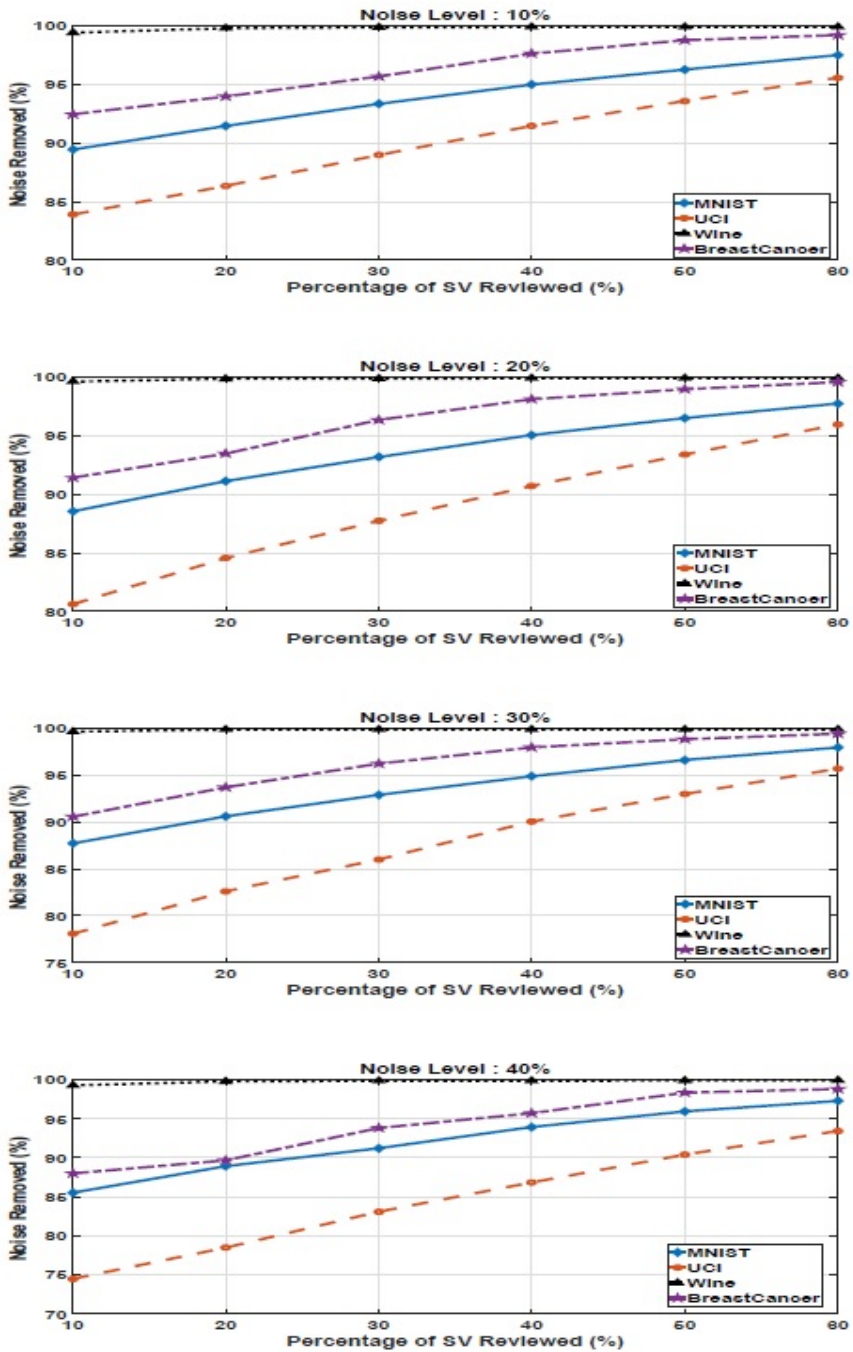


Figure 5: Noise removal performance for difference percentages of SV reviewed

## V. IMPROVED LABEL NOISE FILTER

### A. Introduction

As discussed in the Chapter II, one of the methods to deal with the mislabeled data is mislabeled data filtering. Mislabeled data filtering mainly focuses on identifying and removing mislabeled data before the training stage.

These data filtering methods are prone to make two types of errors: (type1) a correctly labeled sample is regarded as mislabeled and removed from the dataset. This type of error can cause reduction of correctly labeled data from training dataset, this can be harmful for performance of classifier, especially when the number of training samples is small. (type2) an incorrectly labeled data is regarded as correctly labeled and retained in the training set. This type of error can degrade the performance of the classifier because of the mislabeled examples in the training set.

The existing mislabeled data filtering methods do not pay much attention to these two types of errors. They mainly focus on decreasing the error rate or misclassification rate. This means that generally the main objective of such approaches is to create techniques which make lesser number of incorrect predictions. Therefore, the performance measure used in such approaches is classification accuracy on the testing set.

But, a better classification accuracy on testing set does not necessarily mean that the data filtering method used is better at identifying the mislabeled instances from training set and removing them. The better classification on testing set may be due to the inherent characteristic of the dataset, e.g. the classes are very distinct from each other.

Therefore, a better performance measure in this scenario is *Precision* and *Recall*. The details of *Precision* and *Recall* are given in section B of this chapter.

Two of the widely used mislabeled data filtering algorithms are majority filtering (MF) and consensus filtering (CF). These two methods are described in section C and D respectively. After discussion of the two widely used data filtering algorithms, we discuss their potential problems and propose improved mislabel data filtering approach in the subsequent sections.

## B. Precision and Recall

Let us assume a computer program whose task is to retrieve 60 relevant documents from the total of 100 documents. If the program returns only 30 documents, but only 20 of these are the relevant documents and remaining 10 were irrelevant. If we analyze these results we will realize that since out of the 30 retrieved documents 20 were relevant, so they can be labelled as True Positives (TP). Whereas, 10 were irrelevant but since they were retrieved by the program therefore they can be labelled as False Positives (FP). Out of the 70 documents that were not retrieved 30 were irrelevant, hence they can be called as True Negatives (TN). Remaining 40 were relevant documents but they were not retrieved hence can be called False Negatives (FN). The accuracy of a computer program is defined as follows in classification tasks:

$$Accuracy = \frac{TP+TN}{P+N} \quad (9)$$

Where  $P$  and  $N$ , represent number of Positive and Negative examples respectively. Putting values in eq. (9), we get  $[(20 + 30)/ (40 + 60)] = 50$ . Therefore, the accuracy of the program would be 50%.



But, by looking at the accuracy we are unable to tell how the program performed in terms of retrieving the relevant documents and rejecting the irrelevant documents. For this purpose, *Precision* and *Recall* measures are used. *Precision* and *Recall* are defined as follows:

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

$$Recall = \frac{TP}{TP+FN} \quad (11)$$

Since, out of the retrieved documents ( $TP + FP$ ), 20 were relevant i.e.  $TP$ , therefore, the precision of the program would be  $[20 / (20+10)] = 66.67$ . Similarly, out of the 60 relevant documents only 20 were retrieved, therefore the recall of the program would be  $[20/60] = 33.33$ . Therefore, Precision can be seen as measure of *exactness* or *quality*, whereas Recall is a measure of *completeness* or *quantity*.

In classification tasks, precision of 100% for a class  $C$  indicates that every example labeled as belonging to class  $C$  does indeed belong to class  $C$  (but gives no information about number of examples from class  $C$  incorrectly labeled). Whereas, recall of 100% indicates all examples from class  $C$  were labeled as belonging to class  $C$  (but gives no information about number of examples belonging to other class incorrectly labeled as belonging to class  $C$ ).

In Fig. 6., assume that Red documents are the relevant ones and Blue ones are irrelevant. In first case, all the documents are retrieved (including both relevant and irrelevant ones). Therefore, the recall is 100% but precision is very low. In the second case, all the documents retrieved are relevant but number of the documents retrieved is very less. Therefore, precision is 100% but recall is very low. Whereas, in the third case, there is arguably a good compromise between recall and precision.

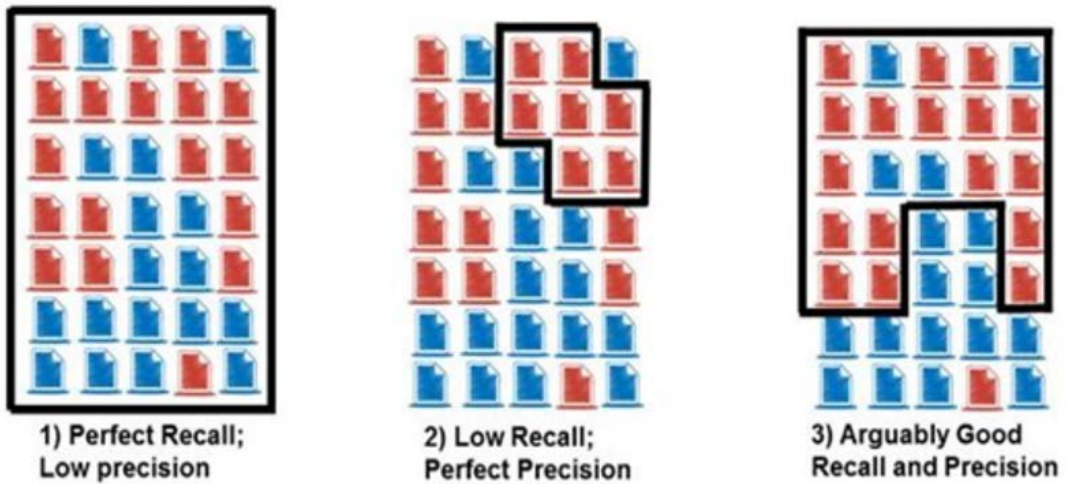


Figure 6: Illustration of Precision and Recall

To find a reasonable compromise between precision and recall, another measure is required which can combine the two.  $F_1Score$ , is one of these measures.  $F_1Score$ , is the harmonic mean of precision and recall. It is defined as follows:

$$F_1Score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (12)$$

### C. Majority Filter

Majority filter (MF) works by tagging an example in the dataset as mislabeled if more than half (majority) of the  $m$  base level classifiers classify it wrongly. The algorithm starts by creating  $n$  equal sized disjoint subsets from the data set. Out of the  $n$  subsets  $n - 1$  are selected for training the  $m$  base-level classifiers and the last subset is used for testing the trained classifiers. An example from the testing set is tagged as mislabeled if majority of the classifiers incorrectly classify it. In the next iteration one of the subsets from training set is used for testing while the previously used subset for testing is now included in the training set, and the process it

repeated. Similarly, the whole process is repeated until all subsets have been used for the testing. The algorithm for MF is shown in Table 10.

**Table 10: Algorithm of majority filter**

Algorithm
$n$ (number of subsets), $y$ (number of examples), 1. Form $n$ disjoint subset using all examples in dataset 2. <b>for</b> $i = 1, \dots, n$ . <b>do</b> 3.     form $i$ th subset for testing = $E_i$ 4.     form $E_t$ such that it contains all subsets except $E_i$ 5. <b>for</b> every $e$ in $E_i$ <b>do</b> 6.         Error $\leftarrow$ 0 7. <b>for</b> $j = 1, \dots, y$ , <b>do</b> 8. $H_j = j$ th classifier trained using all examples in $E_t$ 9.             Test $H_j$ to predict $e$ 10. <b>if</b> $H_j$ incorrectly classifies $e$ 11. <b>then</b> Error = Error + 1 12. <b>end for</b> 13. <b>if</b> Error > $y/2$ 14. <b>then</b> mark $e$ as mislabeled 15. <b>end for</b> 16. <b>end for</b>

## D. Consensus Filter

The Consensus Filter (CF) works similar to the majority filter but the only difference is that in consensus filter if all of the  $m$  base level classifiers fail to

correctly classify an instance only then the instance is tagged as mislabeled. The algorithm for CF is shown in Table 11.

**Table 11: Algorithm of consensus filter**

Algorithm
$n$ (number of subsets), $y$ (number of examples), 1. Form $n$ disjoint subset using all examples in dataset 2. <b>for</b> $i = 1, \dots, n$ . <b>do</b> 3.     form $i$ th subset for testing = $E_i$ 4.     form $E_t$ such that it contains all subsets except $E_i$ 5. <b>for</b> every $e$ in $E_i$ <b>do</b> 6.       Error $\leftarrow$ 0 7. <b>for</b> $j = 1, \dots, y$ , <b>do</b> 8. $H_j = j$ th classifier trained using all examples in $E_t$ 9.             Test $H_j$ to predict $e$ 10. <b>if</b> $H_j$ incorrectly classifies $e$ 11. <b>then</b> Error = Error + 1 12. <b>end for</b> 13. <b>if</b> Error = $y$ 14. <b>then</b> mark $e$ as mislabeled 15. <b>end for</b> 16. <b>end for</b>

## E. Potential Problems with MF and CF

Although MF and CF are the representative algorithms for label noise cleaning problems but there are some potential problems with these. Generally, MF tends to

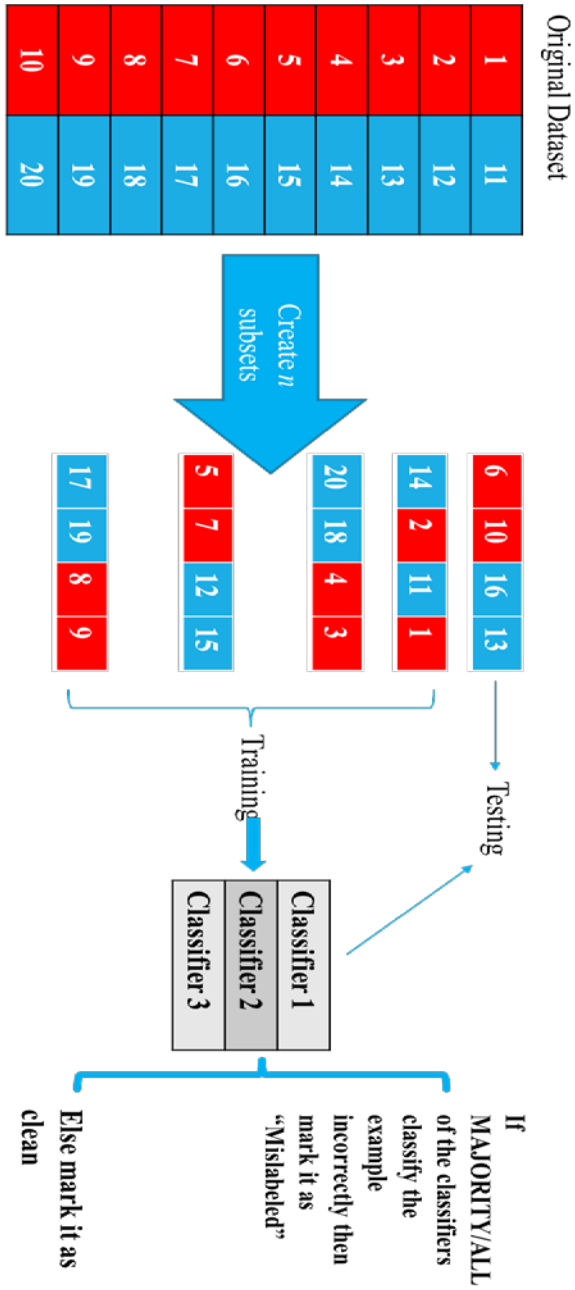


Figure 7: Illustration of Majority / Consensus Filter

perform better than CF. The reason for relatively poor performance of CF is due to the fact that an instance is tagged as mislabeled only when all the classifiers incorrectly classify it i.e. all classifiers have to agree on the same decision. Whereas in MF when majority of the classifiers fail to classify an instance correctly then it is tagged as mislabeled. This implies that consensus of classifiers is not necessary, therefore chances of instance being tagged as mislabeled are greater. In CF chances of a mislabeled example being retained in the dataset are more.

Another potential problem with MF and CF is that it relies on creation of random subsets. It is possible that the training partition contains more noisy examples than testing partition. Also since the intermediate classifiers are trained on noisy examples the testing results of such classifiers may be challenged.

Lastly, since the process must be repeated for all created subsets therefore it is a time-consuming process.

## **F. Proposed Approach**

In this section, we propose an improved majority filter. The proposed method utilizes the ability of support vector machines (SVM) for capturing the potentially mislabeled examples. First an SVM classifier is used to separate potentially mislabeled and clean examples. Then the clean examples are used for training the classifiers and the potentially mislabeled ones are used for testing. If majority of the classifiers incorrectly classify an example, then it is marked as potentially mislabeled and removed from dataset. The algorithm for the proposed improved majority filter is given in Table 12.

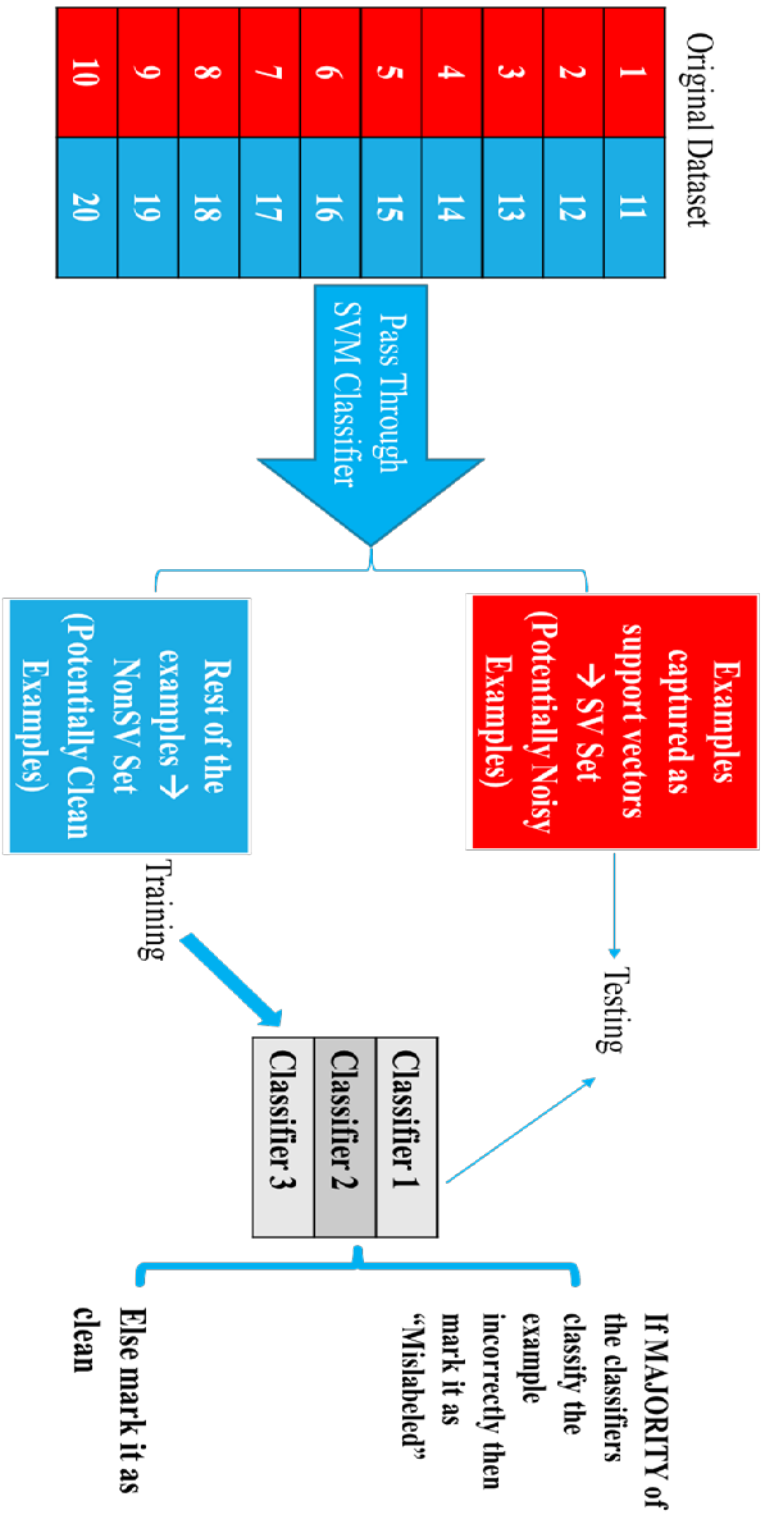


Figure 8: Illustration of Proposed Improved Majority Filter

The main advantage of the proposed approach is that there is no need to create random subsets from the dataset. Therefore, the process has to be done only once as opposed to the conventional majority filter algorithm, for which the process has to be repeated for every subset. Secondly, as discussed in chapter IV, support vector machines have an amazing ability to separate the clean and potentially mislabeled examples. Therefore, majority of the training examples for the classifiers in proposed algorithm are clean examples. So, the results of the classifiers are more reliable as compared the classifiers trained with mislabeled examples.

**Table 12: Algorithm of improved majority filter**

Algorithm
$n$ (number of subsets), $y$ (number of examples), 1. Train an SVM classifier using all examples in dataset and separate SVs and add them to SV_set and add all the remaining examples to Non_SV_set. 2. <b>for</b> $i = 1, \dots, y$ . <b>do</b> 3. $H_i$ = $i$ th classifier trained using all examples in Non_SV_set 4. <b>for</b> every $e$ in SV_set <b>do</b> 5.         Error $\leftarrow$ 0 6.         Test $H_i$ to predict $e$ 7. <b>if</b> $H_j$ incorrectly classifies $e$ 8. <b>then</b> Error = Error + 1 9. <b>end for</b> 10. <b>if</b> Error > $y/2$ 11. <b>then</b> mark $e$ as mislabeled 12. <b>end for</b>



## G. Datasets and Experimental Setup

For experiments in this chapter we have used same datasets and experimental setup as in previous chapter. For further details see section D and E of chapter IV.

We refer to our proposed algorithm as IMF (Improved Majority Filter). For IMF, MF and CF, we have used 3 sub-classifiers. The chosen classifiers are *K-nearest neighbors (KNN) classifier*, *Linear Discriminant Analysis (LDA) classifier*, and *Support Vector Machine (SVM) classifier*. These are some of the widely used machine learning classifiers. For MF and CF, we create 3 random and disjoint subsets.

All the experiments were performed on an Intel® Core™ i5-4590 CPU @ 3.30 GHz machine with 8.00 GB memory, using *Pattern Recognition Toolbox*, MATLAB.

## H. Results

The results of the experiments are illustrated in Table 13, 14, 15 and 16 for MNIST handwritten digit recognition, UCI letter recognition, Wine quality dataset and Wisconsin breast cancer dataset respectively.

From the results it is evident that MF and IMF clearly outperform CF in almost all the cases. It can also be observed that the performance of MF is slightly better than IMF for lower noise levels (10% and 20%), whereas IMF tends to perform better at higher noise levels (30% and 40%). Overall the performance of MF and IMF is almost similar; therefore, the improved performance of IMF is not evident from these results. But the IMF's main advantage is the time taken for the process to be completed. Since, IMF is not dependent on creation of random subsets, and the process need not be repeated for each subset, therefore the process is completed in

lesser time. These results are depicted in Fig. 9. It can be observed from Fig.9 that MF and CF algorithms take almost same time for the process to complete, whereas IMF performs significantly better in terms of time taken for the process to be completed. If  $n$  subsets were created for majority filter algorithm and it takes  $t$  seconds to complete the process then the improved majority filter algorithm approximately takes  $t/n$  seconds to complete the process, with almost similar performance in terms of *precision*, *recall* and  $F_1$ score.

**Table 13: Performance comparison for MNIST dataset. MF: Majority Filter, CF: Consensus Filter, IMF: Improved Majority Filter.**

Noise Level (%)	Precision (%)			Recall (%)			$F_1$ Score (%)		
	MF	CF	IMF	MF	CF	IMF	MF	CF	IMF
10	<b>63.61</b>	10.57	57.1	<b>94.33</b>	10.58	91.53	<b>75.74</b>	10.58	69.94
20	<b>76.01</b>	20.58	72.3	<b>92.52</b>	18.09	90.06	<b>83.39</b>	19.25	80.09
30	79.37	30.71	<b>79.68</b>	89.19	23.92	<b>89.31</b>	83.99	26.88	<b>84.19</b>
40	74.84	39.93	<b>81.6</b>	82.02	24.11	<b>85.89</b>	78.28	30.06	<b>83.68</b>

**Table 14: Performance comparison for UCI dataset. MF: Majority Filter, CF: Consensus Filter, IMF: Improved Majority Filter.**

Noise Level (%)	Precision (%)			Recall (%)			$F_1$ Score (%)		
	MF	CF	IMF	MF	CF	IMF	MF	CF	IMF
10	<b>51.59</b>	11.86	44.81	<b>90.43</b>	11.80	86.37	<b>65.63</b>	11.82	58.98
20	<b>66.60</b>	21.00	60.12	<b>89.18</b>	18.07	84.97	<b>76.25</b>	19.40	70.41
30	<b>73.87</b>	31.04	69.74	<b>87.08</b>	22.77	82.68	<b>79.92</b>	26.26	75.64
40	71.04	40.1	<b>71.78</b>	76.18	24.68	<b>76.29</b>	73.52	30.54	<b>73.96</b>

**Table 15: Performance comparison for Wine quality dataset. MF: Majority Filter, CF: Consensus Filter, IMF: Improved Majority Filter.**

Noise Level (%)	Precision (%)			Recall (%)			$F_1$ Score (%)		
	MF	CF	IMF	MF	CF	IMF	MF	CF	IMF
10	<b>94.94</b>	18.60	91.55	99.00	18.10	<b>99.10</b>	<b>96.93</b>	18.35	95.15
20	<b>97.26</b>	22.98	96.24	<b>99.40</b>	20.75	98.40	<b>98.32</b>	21.81	97.31
30	97.77	30.16	<b>97.63</b>	<b>99.27</b>	24.67	98.63	97.47	27.14	<b>98.13</b>
40	96.07	40.51	<b>97.81</b>	97.47	27.53	<b>98.20</b>	96.77	32.78	<b>98.00</b>

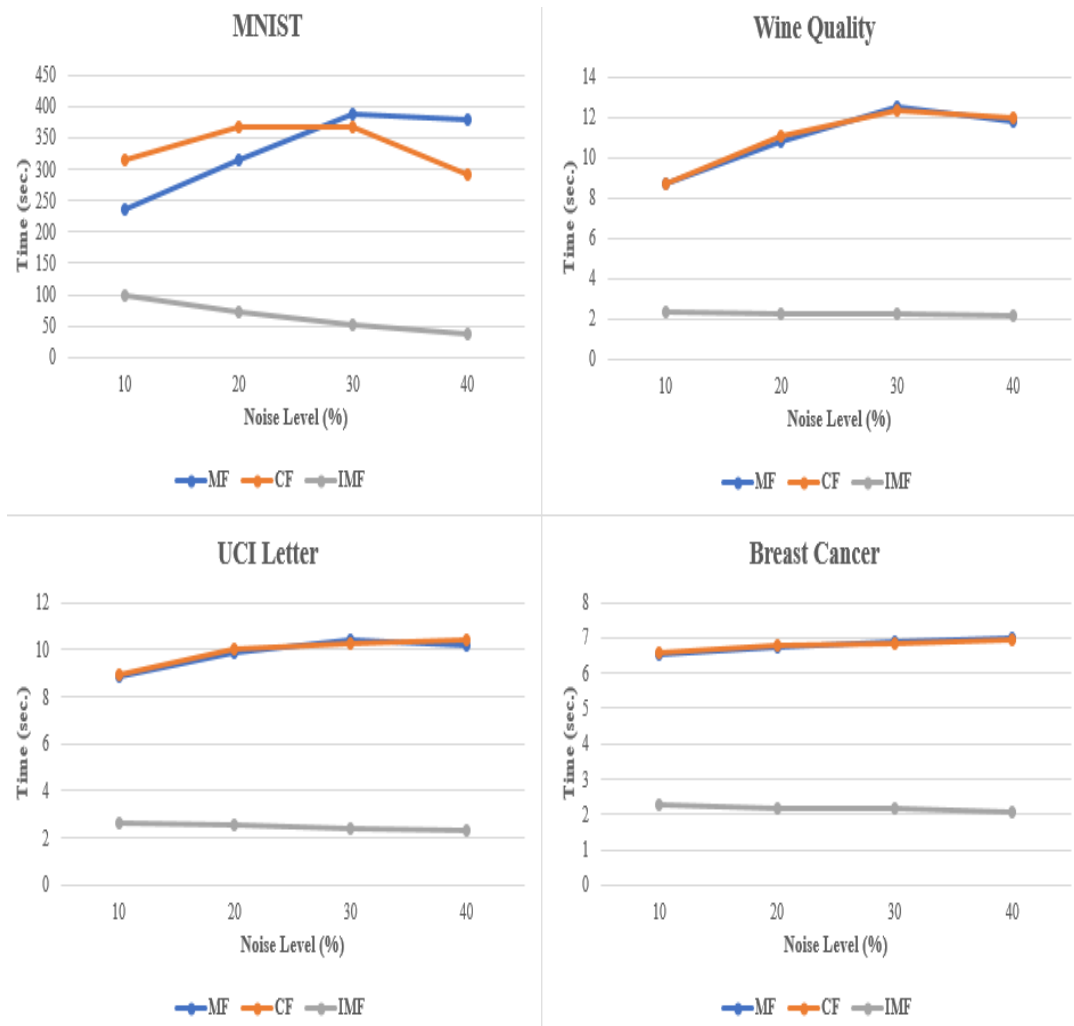
**Table 16: Performance comparison for Wisconsin Breast Cancer dataset. MF: Majority Filter, CF: Consensus Filter, IMF: Improved Majority Filter.**

Noise Level (%)	Precision (%)			Recall (%)			$F_1$ Score (%)		
	MF	CF	IMF	MF	CF	IMF	MF	CF	IMF
10	69.34	11.95	<b>72.91</b>	<b>96.00</b>	13.00	95.25	80.52	12.46	<b>82.60</b>
20	<b>81.67</b>	21.09	80.86	<b>94.88</b>	20.75	91.50	<b>87.78</b>	20.92	85.85
30	84.13	30.84	<b>84.83</b>	90.08	25.33	<b>90.25</b>	87.00	27.82	<b>87.45</b>
40	81.16	41.56	<b>85.95</b>	87.08	25.44	<b>89.31</b>	84.01	31.56	<b>87.60</b>

## I. Conclusion

In this chapter an improved majority filter algorithm for eliminating the potentially mislabeled examples is proposed. The proposed algorithm is significantly faster as compared to the conventional majority filter algorithm. Also, by using support

vector machine classifier’s ability to separate potentially mislabeled and clean examples, the intermediate classifiers are trained using clean examples, unlike the conventional majority filter algorithm where the classifiers are trained using randomly created subsets which may contain mislabeled examples as well. Hence, the results of proposed algorithm are more reliable in this regard.



**Figure 9: Performance comparison of MF, CF and IMF in terms of time taken**

## VI. CONCLUSIONS

The presence of mislabeled instances in datasets can severely affect the performance of machine learning classifiers. Also, an adversary can try to manipulate the dataset by changing the labels of some instances in the database to his/her own benefit. Hence, detection and/or elimination of such instances from datasets becomes extremely necessary.

In this thesis we present some machine learning approaches to deal with this situation. Firstly, we investigated the performance of different clustering techniques for label noise removal. The idea is to first cluster the instances in the dataset and then compare the cluster assignments with the given labels of the instances. If majority of the instances in a cluster belong to a particular class, then all the instances of that cluster are assigned that class. This method has a limitation that it can only perform well in cases where the datasets are easily clusterable.

Secondly, we proposed a similarity based label noise cleaning method. In the proposed method, firstly potentially mislabeled and clean examples are separated using an SVM classifier. Then a mean representation of both the classes is obtained from potentially clean instances (non-support vector examples) and Euclidean distance of each support vector example (potentially mislabeled examples) is measured with both these mean representations. Top half of the instances with similar distances from both the classes are selected for an expert review, while for rest the labels are assigned according to the closeness to a class.

Lastly, we proposed an improved majority filter algorithm. The proposed method utilizes the idea of support vector machines to separate the potentially mislabeled and clean instances. The sub-classifiers are trained using the potentially clean examples obtained after this separation. Hence, eliminating the need of creation of

random subsets, saving a lot of time. And also, the results are more reliable since the classifier were trained using potentially clean examples, rather than the random subsets which may include noisy examples as well.

## BIBLIOGRAPHY

- [1] Bootkrajang, J., & Kabán, A. (2012, September). Label-noise robust logistic regression and its applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 143-158). Springer, Berlin, Heidelberg.
- [2] Bootkrajang, J., & Kabán, A. (2013). Classification of mislabelled microarrays using robust sparse logistic regression. *Bioinformatics*, 29(7), 870-877.
- [3] Bootkrajang, J. (2016). A generalised label noise model for classification in the presence of annotation errors. *Neurocomputing*, 192, 61-71.
- [4] Rousseeuw, P. J., & Leroy, A. M. (2005). *Robust regression and outlier detection* (Vol. 589). John Wiley & Sons.
- [5] Sakakibara, Y. (1993). Noise-tolerant occam algorithms and their applications to learning decision trees. *Machine Learning*, 11(1), 37-62.
- [6] Teng, C. M. (1999). Correcting noisy data. In *Machine Learning*.
- [7] Nicholson, B., Zhang, J., Sheng, V. S., & Wang, Z. (2015, October). Label noise correction methods. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on* (pp. 1-9). IEEE.
- [8] Nicholson, B., Sheng, V. S., & Zhang, J. (2016). Label noise correction and application in crowdsourcing. *Expert Systems with Applications*, 66, 149-162.
- [9] Rebbapragada, U. D. (2010). *Strategic targeting of outliers for expert review* (Doctoral dissertation, Tufts University).

- [10] Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L. O., Goldgof, D. B., & Kasturi, R. (2016). Active cleaning of label noise. *Pattern Recognition*, 51, 463-480.
- [11] Brodley, C. E., & Friedl, M. A. (1996, May). Improving automated land cover mapping by identifying and eliminating mislabeled observations from training data. In *Geoscience and Remote Sensing Symposium, 1996. IGARSS'96. 'Remote Sensing for a Sustainable Future.'*, International (Vol. 2, pp. 1379-1381). IEEE.
- [12] Brodley, C. E., & Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11, 131-167.
- [13] A. Ng, Cs229 lecture notes, part v: Support vector machines, *Technical report*.
- [14] Cha, S. H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2), 1.
- [15] A. Broji, M. Hamidi, F. Mahmoudi, Robust handwritten character recognition with features inspired by visual ventral stream, *Neural Processing Letters* 28 (2), 2008, 97-111.
- [16] Guan, D., Yuan, W., Lee, Y.K., & Lee, S. (2011). Identifying mislabeled training data with the aid of unlabeled data. *Applied Intelligence*, 35(3). 345-358.
- [17] Guan, D., Yuan, W., Lee, Y.K., & Lee, S. (2009). Nearest neighbor editing aided by unlabeled data. *Information Sciences*, 179(13). 2273-2282.