Active Cleaning of Label Noise

Rajmadhan Ekambaram[a,*], Sergiy Fefilatyev[a], Matthew Shreve[a], Kurt Kramer[a], Lawrence O. Hall[a], Dmitry B. Goldgof[a], Rangachar Kasturi[a]

[a]*Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620-5399, USA*

Abstract

Mislabeled examples in the training data can severely affect the performance of supervised classifiers. In this paper, we present an approach to remove any mislabeled examples in the dataset by selecting suspicious examples as targets for inspection. We show that the large margin and soft margin principles used in support vector machines (SVM) have the characteristic of capturing the mislabeled examples as support vectors. Experimental results on two character recognition datasets show that one-class and two-class SVMs are able to capture around 85% and 99% of label noise examples, respectively, as their support vectors. We propose another new method that iteratively builds two-class SVM classifiers on the non-support vector examples from the training data followed by an expert manually verifying the support vectors based on their classification score to identify any mislabeled examples. We show that this method reduces the number of examples to be reviewed, as well as the parameter independence of this method, through experimental results on four data sets. So, by (re-)examining the labels of the selective support vectors, most noise can be removed. This can be quite advantageous when rapidly building a labeled data set.

Keywords: Support Vectors, Label noise, Mislabeled examples

*Corresponding author

Email addresses: rajmadhan@mail.usf.edu (Rajmadhan Ekambaram), sfefilatyev@gmail.com (Sergiy Fefilatyev), mshreve@mail.usf.edu (Matthew Shreve), kurtkramer@gmail.com (Kurt Kramer), hall@cse.usf.edu (Lawrence O. Hall), goldgof@cse.usf.edu (Dmitry B. Goldgof), r1k@cse.usf.edu (Rangachar Kasturi)

1. Introduction

Mislabeled examples in the training data perturb the learning process and are likely to have an adverse effect on the accuracy of a classifier. Label noise can best be examined while an expert is available to label the data. In this paper, we present a procedure for correcting training data that contains label noise. In particular, we investigate finding mislabeled examples using support vector machines (SVM) [1, 2, 3]. This work was motivated by a search for oil-droplet particles in images from underwater platform in the aftermath of Deepwater Horizon Oil Spill. In the search for underwater oil-droplets a new class (actually suspected fish eggs) was found, but because it was a new class, examples were mislabeled. In this case, it was very important to find all "oil droplets". The presence of mislabeled examples in the training data is a critical problem and several approaches have been proposed in the literature [4, 5, 6, 7, 8, 9, 10, 11, 12, 13] to address it. No approach, to our knowledge, focuses solely on the support vectors of a SVM classifier to address this problem. In our previous work [14], we hypothesized that if examples in the training data were erroneously labeled they will tend to be on the margin and get chosen as support vectors of the SVM classifier. In this work, we extend the approach to reduce the number of examples to be reviewed and provide extensive experimental results to demonstrate the validity of the hypothesis. We note that our work is not limited to images. It is also the case that we ignore noise in images, which has been dealt with in many places [15].

We did two sets of experiments to remove the label noise examples. The first set of experiments showed that around 85% and 99% of the label noise examples were selected as support vectors of one-class SVM (OCSVM) and two-class SVM (TCSVM) respectively. In these experiments we also found that large numbers of training examples (around 55% for OCSVM and between 42% and 46% for TCSVM) were selected as support vectors. This leads to reviewing more than 40% of the examples to remove 10% of noise examples. Motivated by the results shown in [6], we rank ordered the support vector of TCSVM

examples based on their class probability. This method showed that most of the label noise examples have low probability for the class to which they are assigned. But we found three problems with this approach: 1) dependency on classifier parameters, 2) the need for the selection of the number of examples to review in each batch, and 3) the need for a threshold to stop the review process. To overcome these problems we have developed a new method and applied it in a second set of experiments. This new method assumes that all the label noise examples are selected as support vectors of a TCSVM, and builds another noise free classifier, which is used to select the potential noise examples in the support vector examples selected in the first step. This leads to a significantly reduced number of examples to be reviewed to remove the label noise examples.

This paper shows that to correct label noise it is enough to review a subset of the support vectors of a trained TCSVM classifier. We re-labeled the noise examples in the support vectors with the help of a human expert. The validity of this approach is demonstrated on four datasets (UCI letter recognition, MNIST digit dataset, Wine quality dataset [16], and Wisconsin Breast Cancer dataset) that contain artificially introduced label-noise. The experimental results show that up to 99%, as shown in Table 6, of the incorrectly assigned labels in the training set are selected as support vectors of an SVM classifier. Using our proposed approach the number of examples to be reviewed can be drastically reduced. The paper is organized as follows. A discussion of previous work related to label noise error is presented in Section 2. The intuition behind our work and the algorithm are explained in Section 3. A detailed description of the experiments and a performance comparison with the probabilistic based method proposed in [6] are presented in Section 4. Section 5 contains our conclusions.

2. Related Work

There are many different approaches to identify and remove mislabeled (label noise) examples that have been explored in the literature. The intuition behind a few of the methods are closely related to our work, i.e., in targeting the

important examples, but differ in the criterion used to define importance. The criterion used is information gain in the work by Guyon et al. [4], distance to the separating hyperplane in the work by Rebbapragada et al. [5], and probability in the work by Rebbapragada [6], and Brodley et al. [17]. In the work by Guyon et al. [4], a method was proposed to select or reduce the number of examples instead of using all the examples for training the classifiers. The examples were manually verified after being put in decreasing order by an information gain criteria to find the most important and potentially mislabeled examples. The examples which produced more information gain were more useful to the classifier, as well as more suspicious. The main idea of this method is similar to our approach. The examples were reviewed based on the information gain criteria and in our approach the criteria is implicitly defined by the large margin principle. We differ from [4] in classifier(s), how we rank examples, the strict use of human in the loop and analysis of the number of trials to remove examples and what percentage of mislabels can be found for removal. In the work by Rebbapragada et al. [5], examples were selected for labeling in an active learning framework using an SVM classifier. The unlabeled examples which lie close to the separating hyperplane were selected for labeling. The intuition of this method is very close in principle to our method, but we are different in the following: our examples are labeled and we only examine the support vector examples. The examples selected for labeling in [5] may or may not become a support vector and online training for large datasets is time consuming. The method of Rebbapragada [6] and Brodley et al. [17] have similarities to our proposed approach. They classified the training data from the classifier created using SMO in Weka [18] and generated a probability with the classification [19]. Then the examples which received low probability were verified by the labeler. The examples are not necessarily support vectors and depending on where the probability threshold for reviewing examples lies, some support vectors on the wrong side of the boundary may be ignored. We compare with this work below.

A few more methods are related to our work, but their approach is different. In the work by Gamberger et al. [7], a complexity measure was defined for the

classifier and a weight was assigned to each example. The method is iterative and in each round of the iteration the example with the highest weight is selected. The selected example is examined for label noise, if its weight is greater than the threshold. Our method is also iterative but the number of rounds is independent of the number of noise examples and also does not require any threshold. In the method of Brodley and Friedl [8], an automatic noise removal technique that also removes good examples was introduced. It increases the classifier accuracy, but may miss a number of mislabels which is problematic if there is a small class of interest. In the method of Zhu et al. [9], a rule based method was proposed to distinguish exceptions and mislabeled examples. The intuition behind the method in [9] is similar to the method in [8], but it can be applied for distributed, large scale datasets. The dataset was divided into subsets and rules were generated for all the subsets. Examples in each subset were classified by the rules generated from all the subsets. The assumption is that the mislabeled examples were misclassified by more rules than exceptions. We do not consider exceptions in our method, but our method can be applied independently in each location of a distributed large scale dataset as long as a sufficient number of positive and negative examples is present in each location. The method of Muhlenbach et al. [10] used geometrical structure to find the mislabeled examples. The Relative Neighborhood graph of the Toussaint method was used to construct a graph. An example is considered as bad or doubtful if its proportion of connections with examples of the same class in the graph is smaller than the global proportion of the examples belonging to its class. This method is closely related to our method, because in both methods examples which are closest to examples from other classes are suspected, but the geometry considered in this method is local whereas in our method the global position of all examples are considered at the same time. A kernel based method was proposed by Valizadegan and Tan [11] for this problem. In this method, a weighted k nearest neighbors (kNN) approach was extended to a quadratic optimization problem. The expression to be optimized depends only on the similarity between the examples and hence can also be solved by projecting the

attributes into higher dimensions with the help of a kernel. The examples whose labels were switched to maximize the optimization expression were considered mislabeled. This method is similar to our method in using the optimization function, but the objective of the optimization function is different. In the work by Rebbapragada and Brodley [12] and Rebbapragada et al. [13], examples are clustered pair wise and a confidence is assigned to each example using the Pair Wise Expectation Maximization (PWEM) method. The classifiers which take a confidence value as input instead of labels can make use of this information. A confidence measure can also be calculated using our method, but the criterion used is different.

The other approach to solve this problem is to mitigate the effect of the label noise examples on the classifier. In the Adaboost learning algorithm, the weights of the misclassified instances are increased and weights of correctly classified instances are decreased. This will create a group of base classifiers which correctly predict the examples that have large weights. The work of Ratsch et al. [20] and Dietterich [21] show that AdaBoost tends to overfit in the presence of mislabeled examples. In order to avoid building base classifiers for noisy examples, a method was proposed by Cao et al. [22] to reduce the weights of the noisy examples using kNN and Expectation Maximization methods. In the work of Biggio et al. [23], Stempfel and Ralaivola [24] and Niaf et al. [25], the SVM problem formulation was modified to handle the label noise problem. In the work of Biggio et al. [23] the optimal decision surface was obtained in the presence of label noise by correcting the kernel matrix of the SVM. The correction reduces the influence of any single data point in obtaining the separating hyperplane. The method in [24] assumes that noise free slack variables can be estimated from the noisy data and the mean of the newly defined non-convex objective function was the noise-free SVM objective function. The method in [25] estimates the probability of each data point belonging to the prescribed class. These probabilities were then used to adjust a slack variable that gives some flexibility to the hard constraints given in the initial optimization problem using a standard SVM. In their experiments, the probabilities were generated

using Platt's scaling algorithm [19] and a function to measure the distance to the boundary. These methods handle noise and create classifiers in a single step, but our method is strictly a preprocessing step to remove the label noise examples before creating any classifier with the training data.

The basic principle of creating a large margin between data from two classes was extended to data from only one class in the work of Schlkopf et al. [26]. The method proposed in the work of Schlkopf et al. [26] is referred as OCSVM. OCSVM finds a small region that encloses most of the data, and the examples that fall outside this region are considered outliers. In the work of Lukashevich et al. [27], Das et al. [28] and Mourao-Miranda et al. [29], OCSVM was used for outlier detection. The method in Lukashevich et al. [27] used OCSVM to detect outliers in image training sets. The method in [28] used OCSVM to remove the outliers in sensor data in a system where the data was distributed across different sites. The method in [29] successfully applied OCSVM on the patterns of fMRI response to find depressed patients. The patterns of the depressed patients were classified as outliers and separated from the normal patients using OCSVM. We considered the mislabeled examples as outliers for the labeled class data, and tested the performance of OCSVM in classifying the label noise examples as outliers.

Comparing to a probabilistic approach. In the dissertation work by [6] several methods (SMO, Naive Bayes, Logistic Regression, Nearest neighbor, Bagging and Boosing) were compared for Iterative Correction of Class Noise (ICCN). The result shows that SMO is one of the best performing confidence based methods and is close in principle to our method, so we choose to compare our method with the SMO confidence based method. The idea is to review the examples in batches and have the reviewer choose the stopping criteria. However, in their experiments they stop when the total number of reviewed examples is equal to the known number of label noise examples present in the dataset. We tested this method on the four datasets (UCI character recognition, MNIST digit recognition, Wine quality, and Wisconsin Breast cancer) following the same

experimental set up used to test our method, which is explained in Section 4. The important difference between our approach and [6] is that we claim label noise examples have a high probability of being selected as support vectors and review only the subset consisting of the support vectors. The method in [6] reviews all the examples (does not differentiate support vectors and non-support vector examples) based on their probability of classification. The other differences are as follows: (a) we select the examples based on a two stage process (b) there is no threshold on the number of examples to be reviewed in a batch and (c) no stopping criteria is required. Based on the experimental results we show that our method produces consistent results for different parameter selection methods. Stopping criteria is an important parameter especially when we don't know the amount of noise in the data, and our method does not require this parameter.

3. Algorithm

Our algorithm exploits the theory behind support vector machines. The dual of the optimization problem created by an SVM is typically solved because it is efficient for high dimensional features and kernel trick can easily be applied to the solution [1]. The SMO-type solver [30, 31], is a computationally efficient way to find the boundary for a training set using an SVM. The dual formulation is

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \alpha_i \alpha_j, \tag{1}$$

where N is the number of training examples, the $y_i \in [-1, 1]$ are the class labels, $\boldsymbol{x}_j$ is a d dimensional example, $K(\cdot)$ is the kernel and α_i is a Lagrange multiplier. Equation (1) is subject to two constraints

$$\alpha_i \geq 0, \forall i, \tag{2}$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0. \tag{3}$$

Now it turns out that $\alpha_i = 0$ for examples that are not needed for the decision boundary. So, only support vectors $\alpha_i > 0$ are used to create the decision boundary. This means two things in this work. First, we only need to look at the labels of support vectors. The other labels are irrelevant in the sense that they do not affect the decision made on test examples. Second, when people find an example difficult to label, one which they are likely to mislabel, it is likely to be a border example near examples that make up the support vectors and be a support vector itself. Also, if an adversary wants to affect decisions by changing labels they must focus on the support vectors.

Another argument for the observation that label noise examples become support vectors is supported by the optimization procedure for SVM parameters [32]. It is reasonable to assume that the mislabeled examples are mixed in with the correctly labeled examples. In such cases, the optimization process of SVMs creates a hyperplane which carves a precise boundary to separate the examples from two classes. These hyperplanes include the mislabeled examples as support vectors. Hence, by validating the support vectors using an expert's knowledge, mislabeled examples can be removed. The process can be iteratively applied to potentially remove all label-noise examples. The algorithm is described in Table 1.

1. Mark all the training examples as not verified
2. Train an SVM classifier using the training examples
3. Have an expert validate all the support vectors marked as not verified:
   (a) Change the labels of the mislabeled examples in the support vectors
   (b) Mark all the support vector examples as verified
4. Repeat steps 2 and 3 until no label error is found

Table 1: Algorithm to verify the hypothesis that the label noise examples are captured in the support vectors.

It can be observed from the experimental results that a classifier with label noise examples has a large number of support vector examples. Reviewing all

1. Create an empty set: SV_set
2. Create an empty set: Non_SV_set
3. Create Classifier A using all the examples in the dataset
4. Separate the SVs of Classifier A from the dataset and add them to SV_set and add all the remaining examples to Non_SV_Set
5. Create Classifier B using the examples in Non_SV_Set
6. Test the examples in SV_Set using the Classifier B and rank the mis-classified examples based on their probability of classification
7. Have an expert validate the previous unseen examples in SV_Set based on the ranking obtained in Step 6
8. Repeat the Steps 2 to 7 until no label noise example is found in Step 7

Table 2: The proposed algorithm to efficiently target the label noise examples in the support vectors.

the support vector examples to find the label noise examples is tedious. So, we have developed a new method which efficiently targets the label noise examples in the support vectors of the TCSVM. If most of the label noise examples are selected as support vectors then it is possible to create a noise free classifier using the non-support vector examples. Though the classifier created using only these non-support vector examples might not perform the best on test data, we show by experiments that it can be used to target the label noise examples. The idea is to measure the distance to the boundary, created by a presumably noiseless model, of the support vector examples and use those with low probability in a class, which are, typically, on the wrong side of the decision boundary, as top candidates for relabeling. We show this reduces the number of examples to examine in Section 4. The detailed steps of this method are given in Table 2.

4. Experiments

We report results for two sets of experiments in this section. The first set of experiments is to show that label noise examples have high probability of being selected as support vectors. To show this we tested the performance of OCSVM,

TCSVM and their combination. The experiments were performed as described in the algorithm in Table 1. In the combination experiment, the support vectors of the OCSVM and the TCSVM are combined at each round until the support vectors of both the classifiers are free of label noise examples. The second set of experiments is to show the performance and the parameter independence of our new method in selecting the subset of label noise examples in the support vectors. The experiments for the new method were performed as shown in the algorithm in Table 2. We also compared the performance of the new method with the method in [6]. We refer to our novel method as ALNR (Active Label Noise Removal) and the method in [6] as ICCN_SMO.

We did experiments with four different datasets widely used in the machine learning community: the UCI Letter recognition dataset, the MNIST digit dataset, wine quality dataset [16], and Wisconsin Breast cancer dataset. The UCI letter recognition dataset has around 700 examples for each letter (A-Z) and each example is represented by a 16 dimensional feature vector. The MNIST digit recognition dataset has around 6000 examples for each digit (0-9) and each example is represented by a 784 dimensional feature vector. We performed some exploratory experiments and selected 3 letters (H, B and R) from the UCI letter recognition dataset which are the most likely to be confused. In the work by [33], it was stated that the digits 4, 7 and 9 in the MNIST digits recognition dataset had the most confusion among them, so these three digits were selected. We performed the first set of experiments with these three selected letters and digits from the UCI and MNIST datasets, respectively. The wine quality dataset has around 1100 examples for the red wine class and 3150 examples for the white wine class and each example is represented by a 12 dimensional feature vector. The Wisconsin Breast cancer dataset has 212 examples for the malignant class and 357 examples for the benign class and each example is represented by a 30 dimensional feature vector. The second set of experiments were performed with all four datasets. The experiments were done using scikit-learn python machine learning library ([34]) which uses the LIBSVM library [35] for SVM classification.

UCI Letter Recognition Dataset								
Experiment #	Class X				Class Y			
	Letter	# CLE	# MLE	# TE	Letter	# CLE	# MLE	# TE
1	H	450	50	100	B	225	25	50
					R	225	25	50
2	B	450	50	100	R	225	25	50
					H	225	25	50
3	R	450	50	100	H	225	25	50
					B	225	25	50
MNIST Digit Recognition Dataset								
Experiment #	Class X				Class Y			
	Digit	# CLE	# MLE	# TE	Digit	# CLE	# MLE	# TE
4	4	900	100	500	7	450	50	250
					9	450	50	250
5	7	900	100	500	9	450	50	250
					4	450	50	250
6	9	900	100	500	4	450	50	250
					7	450	50	250
Wine Quality Dataset								
Experiment #	Class X				Class Y			
	Wine Type	# CLE	# MLE	# TE	Wine Type	# CLE	# MLE	# TE
7	Red	450	50	200	White	450	50	200
Wisconsin Breast Cancer Dataset								
Experiment #	Class X				Class Y			
	Type	# CLE	# MLE	# TE	Type	# CLE	# MLE	# TE
8	Malignant	90	10	30	Benign	90	10	30

Table 3: The number of examples used in each of the experiments at 10% noise level. CLE - correctly labeled examples, MLE - mislabeled examples, TE - test examples. The number of examples correspond to the letter or digit or wine type in the same row under the same class. The mislabeled examples in Class X are labeled as Class Y and vice-versa.

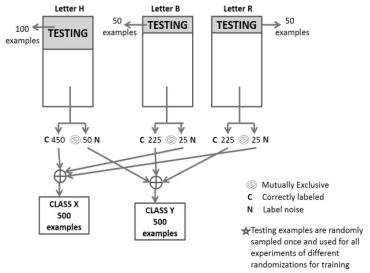

Figure 1: The sampling process of examples for an experiment

In each experiment the dataset was divided into two classes: X and Y. For example, in the first experiment using the UCI letter recognition dataset letter H was considered as class X and letters B and R were considered as class Y. In the second experiment the letter B was considered as class X and the letters H and R were considered as class Y. In the third experiment the letter R was considered as class X and the letters H and B were considered as class Y. For OCSVM experiments only the class X examples were used. The testing examples to evaluate the classifier performance were sampled first from each class. The examples to test our algorithm were sampled from the rest of the examples in the dataset as follows: randomly sample 500 examples from class X and relabel 50 of them as class Y, randomly sample 250 examples from each letter in class Y and relabel 25 of them from each letter to class X. An example sampling process at noise level of 10% is shown in Figure 1. The dataset partition for each experiment at noise level of 10% is captured in Table 3. The number of correctly labeled and mislabeled examples were changed proportionately at different noise levels.

The same procedure was applied in testing the MNIST dataset, but the number of examples used was different. With a large number of examples available for each class in the MNIST dataset, we used 1000 examples for both classes. Class X had 900 correctly labeled examples and 100 noise examples (50 from each digit in class Y). Class Y had 900 correctly labeled examples (450 from each digit) and 100 noise examples from the digit in class X. The wine quality dataset has only 2 classes: red and white wines. Class X is formed from 450 correctly labeled red wine examples and 50 incorrectly labeled white wine examples, and Class Y is formed from 450 correctly labeled white wine examples and 50 incorrectly labeled red wine examples. The Wisconsin Breast cancer dataset has only 2 classes: malignant and benign cells. Class X is formed from 90 correctly labeled malignant cell examples and 10 incorrectly labeled benign cell examples, and Class Y is formed from 90 correctly labeled benign cell examples and 10 incorrectly labeled malignant cell examples. In order to avoid bias from the examples chosen in any one experiment we repeated each experiment in Table

3, 30 times with different randomly sampled examples. All the reported results for the first set of experiments are the average of the 180 experiments (90 each for UCI Letter and MNIST Digit recognition datasets) and the results for the second set of experiments are the average of the 240 experiments (90 each for UCI Letter and MNIST Digit recognition datasets, 30 for Wine Quality dataset and 30 for Breast cancer dataset).

In ICCN_SMO the examples are reviewed in batches and the review is stopped when the number of reviewed examples is equal to the amount of label noise examples in the dataset. The number of examples to be reviewed in a batch was arbitrarily set to 20. In our implementation of ICCN_SMO some changes were made to the experimental setup to make a fair comparison. The number of examples to be reviewed in a batch was varied between datasets. We choose 20 examples for the Breast cancer dataset, 30 examples for the UCI and Wine Quality datasets and 50 examples for the MNIST dataset. These numbers were chosen in proportion to the number of examples in the dataset. Also, the review process was extended to between 20 and 25% more examples than the amount of noise in the dataset. For both methods the criteria for review is based on probability.

The feature values of the data were scaled between -1 and 1 and classifiers were built using linear and RBF kernels. Parameter selection was done independently using 5-fold cross validation for each random choice of training data. The range of the RBF kernel parameter "γ" was varied in multiples of 5 from 0.1/(number of features) to 10/(number of features). In addition, two other "γ" values 0.01/(number of features) and 0.05/(number of features) were tested. The range of the SVM cost parameter "C" was also varied between 1 and 25 in steps of 3.

We first discuss the results for the first set of experiments on the UCI Letter and MNIST character recognition datasets. The overall percentage of label noise examples selected as support vectors on the UCI and MNIST datasets in selecting the label noise examples as support vectors over 30 experiments at the 10% noise level is 85.75% and 85.79% for OCSVM with the linear and RBF

kernels respectively and 99.55% for TCSVM with both the linear and RBF kernels. The detailed results for one of the experiments using OCSVM and TCSVM are shown in Tables 4 and 5, respectively, and the overall performance is shown in Table 6. It was observed that the majority of the noise examples were removed in the 1st round of iterations and very few noise examples were removed in the subsequent rounds, in all experiments. It is clear that up-to 45% of the examples can be support vectors when 10% of the examples have incorrect noisy labels noise examples in the dataset as shown in Table 6. Generally, more complex boundaries will entail more support vectors. The number to be looked at may not scale well as the training set becomes large in some cases.

We also performed another experiment in which the support vectors of both one-class and two-class classifiers (only class X support vectors) at each iteration were added together and examined for the presence of label noise examples. For a linear kernel, this resulted in an overall improvement in finding mislabeled examples of around 1.5% and for the RBF kernel the improvement was only around 0.1%. The results of this experiment are shown in Table 6. The performance of OCSVM in selecting the label noise examples as support vectors for different values of "μ" is shown in Table 7. Again, we see that the number of support vectors can be a significant percentage of the total number of examples which might be problamatic for large data sets, if the number of support vectors scales linearly with training set size.

Figure 2: Example misclassification results. The images on the left and right are labeled as 4 and 9 respectively in the dataset. The image on the left is correctly identified as a mislabeled example, whereas the image on the right is incorrectly identified as a correctly labeled example.

TCSVM using the RBF kernel failed to find 15 mislabeled examples in total over 90 (3 experiments * 30 repetitions) MNIST dataset experiments. Two examples missed by the RBF kernel are shown in Figure 2. The image on the

Iteration #	Cumulative # SV reviewed	Cumulative # Label noise examples removed	RBF Kernel parameter (γ)	# SV in the iteration
1	503	79	0.0014	503
2	546	87	0.0005	465
3	550	89	0.0005	460
4	552	90	0.0005	460
5	553	90	0.001	458

Table 4: The result of a single run of experiment 4 with an OCSVM classifier on the MNIST data at the 10% noise level. This table shows the iteration number, the cumulative number of support vectors to be reviewed until that iteration, the cumulative number of label noise examples selected as support vectors until that iteration, the kernel parameters used for that iteration and the number of support vectors selected in that iteration by the OCSVM classifier. The parameter "μ" was set to 0.5.

# Iteration	Cumulative # SV reviewed	Cumulative # Label noise examples removed	Parameter " C"	RBF Kernel parameter (γ)	Training accuracy in %
1	841	99	25	0.001	88.8
2	848	100	22	0.005	98.95
3	849	100	25	0.005	98.75

Table 5: The result of a single run of experiment 4 with a TCSVM classifier on the MNIST data at 10% noise level. This table shows the iteration number, the cumulative number of support vectors to be reviewed after that iteration, the cumulative number of label noise examples selected as support vectors until that iteration, the kernel parameters used for that iteration and the training accuracy of the classifier using that kernel parameter in that iteration. In this case all noise examples were removed.

Dataset	Linear Kernel					
	OCSVM		TCSVM		Combined	
	% outliers	% noise removed	% support vectors	% noise removed	% support vectors	% noise removed
MNIST	55.05	89.46	42.91	98.23	57.26	99.67
UCI	55.02	78.33	48.80	97.92	53.67	99.31
Overall	55.04	85.75	44.87	98.13	56.06	99.55

Dataset	RBF Kernel					
	OCSVM		TCSVM		Combined	
	% outliers	% noise removed	% support vectors	% noise removed	% support vectors	% noise removed
MNIST	55.23	91.21	45.56	99.85	40.59	99.95
UCI	54.93	74.95	42.80	99.78	33.69	99.95
Overall	55.13	85.79	44.64	99.83	38.29	99.95

Table 6: The average performance over 180 experiments on both the MNIST and UCI data sets and the overall performance at 10% noise level. For OCSVM these results were obtained when using the value 0.5 for parameter "μ"

"μ"	MNIST		UCI	
	% outliers	% noise removed	% outliers	% noise removed
0.3	36.19	77.17	34.69	53.86
0.4	45.80	85.4	44.88	64.15
0.5	55.23	91.21	54.93	74.95
0.6	64.44	94.92	64.14	80.95
0.7	73.43	97.51	73.29	87.15
0.8	82.44	99.17	82.39	93.11

Table 7: The average performance of OCSVM with RBF kernel for different "μ" values over 180 experiments on both the MNIST and UCI data set at 10% noise level.

Extensive parameter selection experiment				
% Noise level	Linear Kernel		RBF Kernel	
	% examples reviewed	% noise removed	% examples reviewed	% noise removed
10	16.40	93.56	13.34	95.84
20	26.40	93.92	23.47	96.01
30	37.26	93.99	34.08	95.69
40	50.64	94.32	48.20	95.72
50	70.03	94.89	71.11	96.22

Table 8: The average performance of ALNR in selecting the label noise examples for labeling over 240 experiments on all the data sets for the extensive parameter selection experiment.

Random parameter selection experiment				
% Noise level	Linear Kernel		RBF Kernel	
	% examples reviewed	% noise removed	% examples reviewed	% noise removed
10	16.94	93.76	18.84	96.06
20	27.15	94.21	29.35	96.30
30	38.12	94.10	40.09	96.30
40	51.16	94.31	51.89	96.35
50	70.21	95.03	73.78	96.67
Default parameter selection experiment				
% Noise level	Linear Kernel		RBF Kernel	
	% examples reviewed	% noise removed	% examples reviewed	% noise removed
10	16.40	93.41	16.37	92.76
20	26.34	93.81	25.82	92.85
30	37.11	93.90	35.36	92.74
40	50.28	94.23	46.70	92.67
50	70.05	94.85	70.17	91.46

Table 9: The average performance of ALNR in selecting the label noise examples for labeling over 240 experiments on all the data sets for the Random and Default parameter selection experiments.

left is mislabeled as a 4 in the dataset and its correct label is 9. By looking at this image we believe that it is a reasonable miss by our method, since the digit is a bit ambiguous. The image on the right is mislabeled as 9 in the dataset and its correct label is 4. Though it appears clear to us from the image that the digit is a 4, our method failed to identify it as mislabeled.

We now discuss the results for the second set of experiments on all four datasets. For the second set of experiments the total number of examples were kept the same but the noise level was varied from 10% to 50%. For example, for the MNIST digit recognition dataset the number of correctly labeled examples in Class X was 700 and incorrectly labeled examples was 300 at a noise level of 30%. In addition to finding the performance in removing the label noise examples, we also report the accuracy of the classifier while cleaning the dataset. When the examples were reviewed and re-labeled, intermediate classifiers were built using the new labels of the examples. The parameter estimation for these intermediate classifiers was done following the procedure explained earlier. The performance of the intermediate classifiers was estimated based on the accuracy of classification on the test set examples. The same test examples were used in all the 30 repetitions of each experiment and the average performance is reported. Classification performance was estimated with an RBF kernel classifier, and its "C", and "*gamma*" are set to 1 and 1/(number of features) respectively. Estimating the performance after reviewing every example is computationally intensive, so performance was estimated at regular intervals of about 1/10 of the amount of noise in the data. For example, in one of the UCI experiments with 30% label noise, performance was estimated after reviewing every 30 examples, whereas for the MNIST experiment with 30% label noise, performance was estimated after reviewing every 60 examples. We also tested two non-character datasets (wine quality dataset and Wisconsin Breast cancer dataset) for this experiment. The cumulative results of this extensive parameter selection method over all the datasets at different noise levels is shown in Table 8.

We tested the parameter dependence of ALNR in two ways: with random parameters and with default parameters. In each round of the random param-

eter experiments random values for "C" and "$gamma$" were uniformly chosen from the range of values mentioned earlier for both the linear and RBF kernels. In the default parameter experiments values for "C" and "$gamma$" were set to 1 and 1/(number of features) respectively. The cumulative results of these two experiments over all the datasets at different noise levels are shown in Table 9. The detailed results of each experiment are shown in Tables 10 and 11 and in Figures 3 to 10. We refer to the extensive parameter selection method as 'Regular', the random parameter selection method as 'Random' and the default parameter selection method as 'Default' in all tables and figures.

The values in the Tables 8, 9, 10 and 11 were obtained by averaging the final results (i.e, when each of the experiment completes) of all the experiments. Ideally each point in the graph in the Figures 3 to 10 should be the average of all the experiments, but the number of examples reviewed in each of the experiments were different. So if a value was not available for averaging for an experiment, its final result was used to get the contribution of that experiment. For example, in one experiment on MNIST dataset with linear kernel with 30% label noise examples, 95.8% of the label noise examples were removed by reviewing 36.9% of the examples. To calculate the average noise removal performance after reviewing 39% of examples, the value 95.8% was used for this experiment. A similar procedure was followed for computing the average accuracy of the classifiers. This was done to reduce bias from any of the experiments if the number of experiments available to calculate the average is small. Due to this small difference in the calculation of the performance values between the Tables 8, 9, 10 and the graphs in the Figures 3 to 8, the last point in each of the graph might not exactly equal to the values in the Tables. Due to the experimental setup this difference is unavoidable. At 50% noise level, only around 55% of the ICCN_SMO experiments reviewed up to 60% of examples, in contrast around 96% of ALNR experiments reviewed up to 60% of examples. Due to large variation in the results of the ICCN_SMO experiments the average results beyond 60% of reviewed examples might be biased by the results of few experiments. For this reason we are not comparing the performance of these two methods at

UCI Letter Recognition Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	90.48	78.18	90.91	78.07	90.04	77.84	95.09	93.14	94.50	89.54	88.02	80.71
20	90.77	86.92	91.44	86.88	90.50	86.79	95.39	94.55	94.87	91.33	88.38	88.07
30	90.80	90.98	91.40	91.02	90.53	90.97	94.39	95.42	94.56	93.34	87.98	91.58
40	91.02	93.20	90.94	93.24	90.74	93.25	93.80	95.87	94.69	91.88	87.76	93.65
50	92.09	39.42	92.25	38.17	91.98	35.48	92.98	55.96	93.74	46.80	82.08	34.26
MNIST Digit Recognition Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	94.08	70.82	94.25	59.01	94.07	71.38	95.75	86.69	96.36	78.16	94.12	93.88
20	94.63	77.65	94.75	68.85	94.59	78.60	95.91	90.47	96.62	85.33	94.10	96.65
30	94.69	81.55	94.55	74.64	94.66	82.49	95.80	86.68	96.72	87.86	94.09	97.84
40	95.12	75.57	95.14	70.58	95.12	81.54	96.15	81.91	96.79	87.84	94.32	98.58
50	95.49	67.90	95.68	65.56	95.49	72.39	97.33	43.45	97.70	53.05	95.90	35.22
Wine Quality Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	99.17	99.37	99.23	99.33	99.17	99.47	99.00	98.73	99.10	98.30	98.93	99.37
20	98.77	99.33	98.87	99.28	98.75	99.30	98.72	99.13	98.78	99.22	98.62	99.27
30	99.00	99.46	98.92	99.48	98.91	99.48	98.91	99.54	98.99	99.51	98.69	99.47
40	99.19	99.64	99.27	99.64	99.17	99.64	99.03	96.35	99.24	96.60	98.99	99.64
50	99.30	32.12	99.29	48.15	99.30	31.80	99.28	51.01	99.41	45.89	95.91	34.10
Wisconsin Breast cancer Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	96.00	91.33	94.50	88.17	95.33	94.17	94.00	91.00	95.00	86.83	92.33	93.67
20	95.67	95.08	96.08	93.92	95.58	97.17	94.83	94.00	96.00	93.50	93.83	96.42
30	96.00	94.61	97.06	93.17	96.50	97.17	95.61	96.17	96.28	92.78	93.83	98.28
40	95.50	83.54	95.21	79.92	95.25	83.96	94.96	85.33	93.88	85.12	85.12	92.42
50	96.97	62.07	95.77	61.73	96.63	51.43	96.13	44.80	96.07	57.93	77.03	40.00

Table 10: Average noise removal performance of ALNR and ICCN_SMO on all the datasets. The performance is the average over 90 experiments on the UCI Letter and MNIST Digits datasets, and 30 experiments on the Wine Quality and Breast cancer datasets. Regular, Random and Default refer to the extensive, random and default parameter selection experiments respectively. All the results are in percentage of noise examples reviewed versus all examples reviewed.

UCI Letter Recognition Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	19.58	12.00	19.78	12.00	19.59	12.00	14.02	12.00	23.32	12.00	20.42	12.00
20	28.83	24.00	29.09	24.00	28.74	24.00	24.57	24.00	34.02	23.43	29.74	24.00
30	37.62	36.00	37.87	36.00	37.59	36.00	35.76	36.00	44.20	35.27	38.93	36.00
40	48.56	48.00	48.53	48.00	48.49	48.00	49.66	48.00	56.27	46.13	50.84	48.00
50	71.20	39.93	71.37	38.73	71.20	35.83	69.26	41.37	74.56	37.40	66.76	32.10
MNIST Digit Recognition Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	15.82	12.50	16.64	12.50	15.81	12.50	13.35	12.47	17.52	12.44	15.25	12.50
20	26.20	25.00	27.36	25.00	26.15	25.00	23.33	24.92	28.44	25.00	24.71	25.00
30	38.28	37.50	39.60	37.44	38.05	37.50	33.36	37.42	39.63	37.47	34.39	37.50
40	53.44	47.36	54.27	46.89	52.89	48.44	48.47	49.94	51.08	49.47	45.32	50.00
50	69.10	56.50	69.46	55.83	69.33	58.58	72.03	43.28	73.68	44.47	72.08	37.31
Wine Quality Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	10.90	12.00	10.91	12.00	10.91	12.00	11.02	12.00	14.11	12.00	11.04	12.00
20	20.85	23.70	20.83	23.80	20.83	23.70	20.88	24.00	21.35	23.90	21.13	23.80
30	30.63	35.80	30.65	35.80	30.59	35.80	32.89	35.80	30.97	35.80	30.74	35.90
40	40.89	47.00	40.80	47.10	40.91	47.10	41.50	45.90	43.28	46.10	41.17	47.20
50	72.08	23.90	71.46	32.10	71.20	22.70	71.41	34.30	72.29	31.50	70.85	24.80
Wisconsin Breast cancer Dataset												
Noise Level %	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
10	13.77	12.50	13.47	12.50	13.65	12.50	14.58	12.50	15.03	12.50	15.70	12.50
20	23.58	25.00	23.42	25.00	23.65	25.00	24.02	25.00	26.73	25.00	23.92	25.00
30	34.43	37.50	35.12	37.50	34.45	37.50	36.35	37.50	38.00	36.58	33.88	37.50
40	46.45	47.67	49.07	48.00	45.48	45.00	51.87	46.00	53.43	46.00	53.52	47.33
50	70.32	52.42	69.20	54.33	68.75	50.00	69.38	39.50	72.77	49.75	60.58	40.58

Table 11: Average examples reviewed for ALNR and ICCN_SMO on all the datasets. The numbers shown are the average over 90 experiments on the UCI Letter and MNIST Digits datasets and 30 experiments on the Wine Quality and Breast cancer datasets. Regular, Random and Default refer to the extensive, random and default parameter selection experiments respectively. All the numbers are in percentage of the total number of examples reviewed versus the total number of examples in the dataset.

Dataset	Kernel: Linear						Kernel: RBF					
	Regular		Random		Default		Regular		Random		Default	
	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO	ALNR	ICCN_SMO
UCI-Letters	7.25	13.33	6.95	12.99	7.48	13.23	8.31	13.45	7.08	12.68	8.01	12.85
MNIST-Digits	11.76	17.89	11.79	18.20	12.50	17.77	7.00	16.80	6.75	16.23	7.98	16.89
Wine quality	4.15	11.87	4.03	11.78	4.31	12.57	4.55	12.67	3.92	11.97	4.42	12.44
Breast Cancer	5.22	4.88	5.03	4.75	5.38	4.96	4.33	4.47	3.81	4.52	4.67	4.75

Table 12: Average number of batches required for reviewing the datasets by ALNR and ICCN_SMO. The numbers shown are the average over all the experiments at all the noise levels for each dataset.

the 50% noise level, but the performance results and graphs are included for completeness.

Table 8 shows that ALNR with RBF kernel removes more than 95% of the label noise examples by reviewing around 8% more examples than the amount of noise in the data. The linear kernel results in reviewing around 3% more examples than the RBF kernel, but the amount of noise removed is 2% less. From these experimental results it appears that RBF kernel is superior to the linear kernel for removing the label noise examples. Comparing Tables 8 and 9, it can be observed that the noise removal performance of extensive and random parameter selection experiments are similar, but around 5% fewer examples need to be reviewed for the extensive parameter selection experiments with RBF kernel. The noise removal performance of default parameter selection experiments is around 1% and 3% less than the extensive parameter selection experiments with the linear and RBF kernels respectively.

From Figures 3 to 10 it can be observed that ICCN_SMO appears to target examples that improve the performance of the algorithm better than the examples targeted by ALNR at the 40% noise level in the UCI and Breast cancer datasets. In contrast ALNR targets examples that improves the performance of the algorithm better than the examples targeted by ICCN_SMO at 40% noise level in the Wine Quality dataset. The noise removal performance of the ALNR is better than ICCN_SMO in the MNIST Digit recognition dataset with a Linear kernel. MNIST is a high dimensional dataset compared to the UCI Letter recognition, Wine Quality and the Breast cancer datasets. Table 10 shows that ANLR removes more noise than ICCN_SMO for UCI, MNIST and Breast cancer datasets except at the 40% noise level for the UCI dataset and for the Breast cancer dataset with the RBF kernel. Table 11 shows that the average difference in the number of reviewed examples between ALNR and ICCN_SMO is less than 3% except at the 10% noise level for UCI with linear kernel, where the difference is around 7%. For the Wine quality dataset both ALNR and ICCN_SMO removed an equal amount of noise and ALNR requires less examples to be reviewed.

From Table 10, it can be observed that ALNR performance varies around 10% between the Regular, Random and Default parameter selection methods for the UCI dataset with an RBF kernel and for the Breast cancer dataset with RBF kernel at 40% noise. For all other datasets the difference in performance between different parameter selection methods is only around 2%. In comparison, ICCN_SMO performance varies around 10% for the UCI dataset with the RBF kernel and for the MNIST dataset with both the linear and RBF kernel and around 5% for the Breast cancer dataset with RBF kernel. This shows that ALNR is robust to parameter selection, which is a useful criteria for large datasets. In ICCN_SMO examples are reviewed in batches, selecting the number of examples to be reviewed is a parameter and should be known apriori for the dataset. This parameter is not required for ALNR.

The results in Table 12 shows that ALNR requires fewer batches be reviewed except for the Breast cancer dataset with a linear kernel in which the difference is less than one batch. Both methods invoke the SVM solver iteratively to find the support vectors for review, but in each round of the iteration ALNR invokes the SVM solver twice whereas ICCN_SMO invokes it only once. We used the LIBSVM implementation of the SVM solver in our experiments and the worst case computational complexity of this SVM solver is $\mathcal{O}(n^3)$ [36], where n is the number of examples. If "k" is the number of rounds to review the dataset, then $\mathcal{O}(kn^3)$ is the computational complexity of both ALNR and our implementation of ICCN_SMO. The results in Table 12 shows that $k << n$.

5. Conclusions

We proposed a method to remove label noise examples in the training data. The method involves reviewing only a fraction of the training examples which are selected using support vector machines. We experimentally showed that label noise examples in the data are selected as outliers and the support vectors of the OCSVM and TCSVM, respectively. The experimental results show that the performance of TCSVM is superior to OCSVM in selecting the label noise

examples as support vectors. TCSVM outperforms OCSVM in both the number of label noise examples that can be removed (more) and the number of examples to be reviewed (less). The combination of the two approaches produced marginal improvements. The experimental results on the UCI and MNIST character recognition datasets show that TCSVM captures around 99% of label noise examples with a review of around 45% of the labeled examples when the data contains 10% label noise examples. We proposed a new method which reduces the number of examples to be reviewed, and is robust to parameter selection. This method removes more than 95% of the label noise examples by reviewing around 10% more examples than the amount of noise in the data. The average difference in performance of this method between the parameters selected using extensive cross validation method and the default parameter is within 1% for the linear kernel and 3% for the RBF kernel. Future work might focus on evaluating our hypothesis and modifying our approach to label noise in multi-class problems. This work may also be applied and evaluated in situations where an adversary is trying to mislabel examples to modify the classifier for their benefit.

Acknowledgement

We thank UCI Machine learning repository for providing the data for this experiment. Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository (*http://archive.ics.uci.edu/ml*). Irvine, CA: University of California, School of Information and Computer Science.

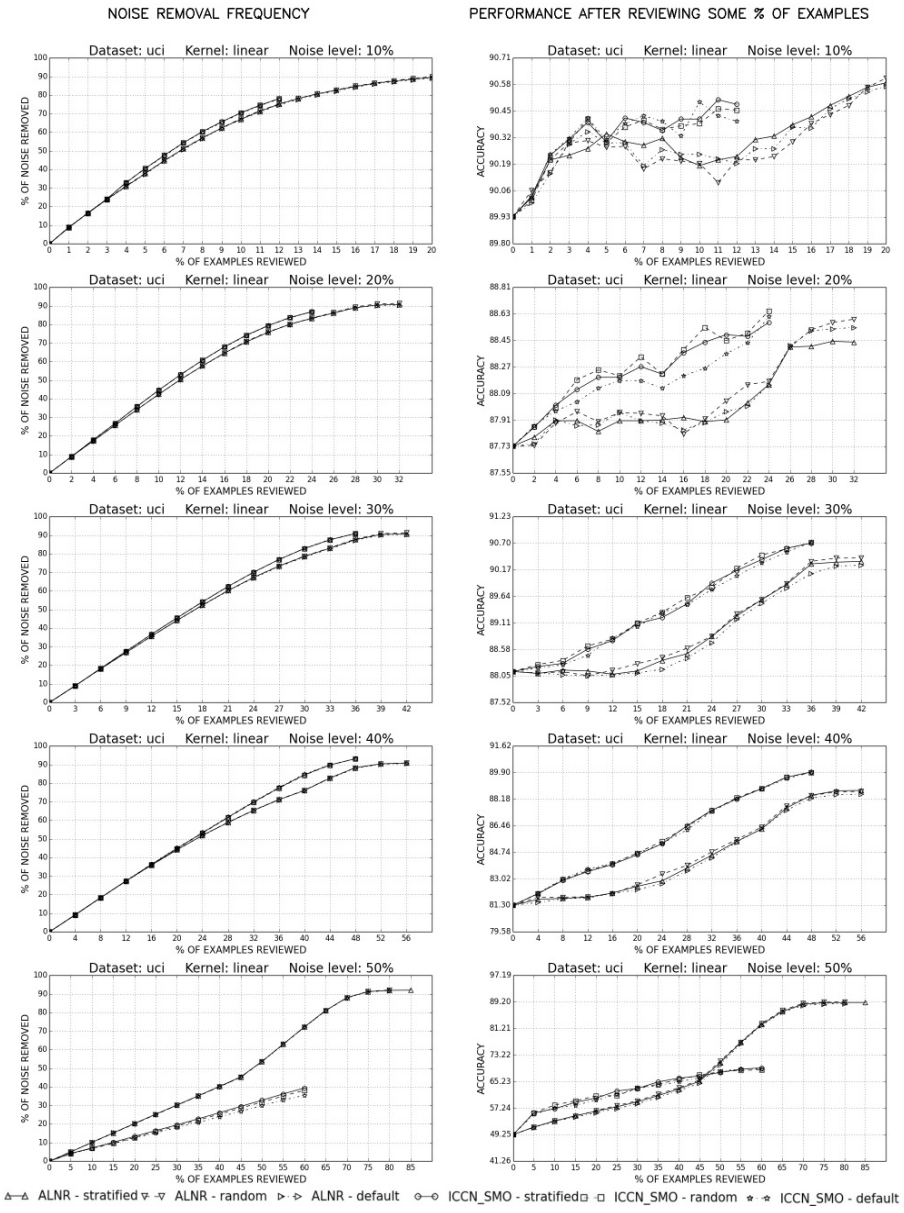

Figure 3: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the UCI Letter recognition dataset using the Linear Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

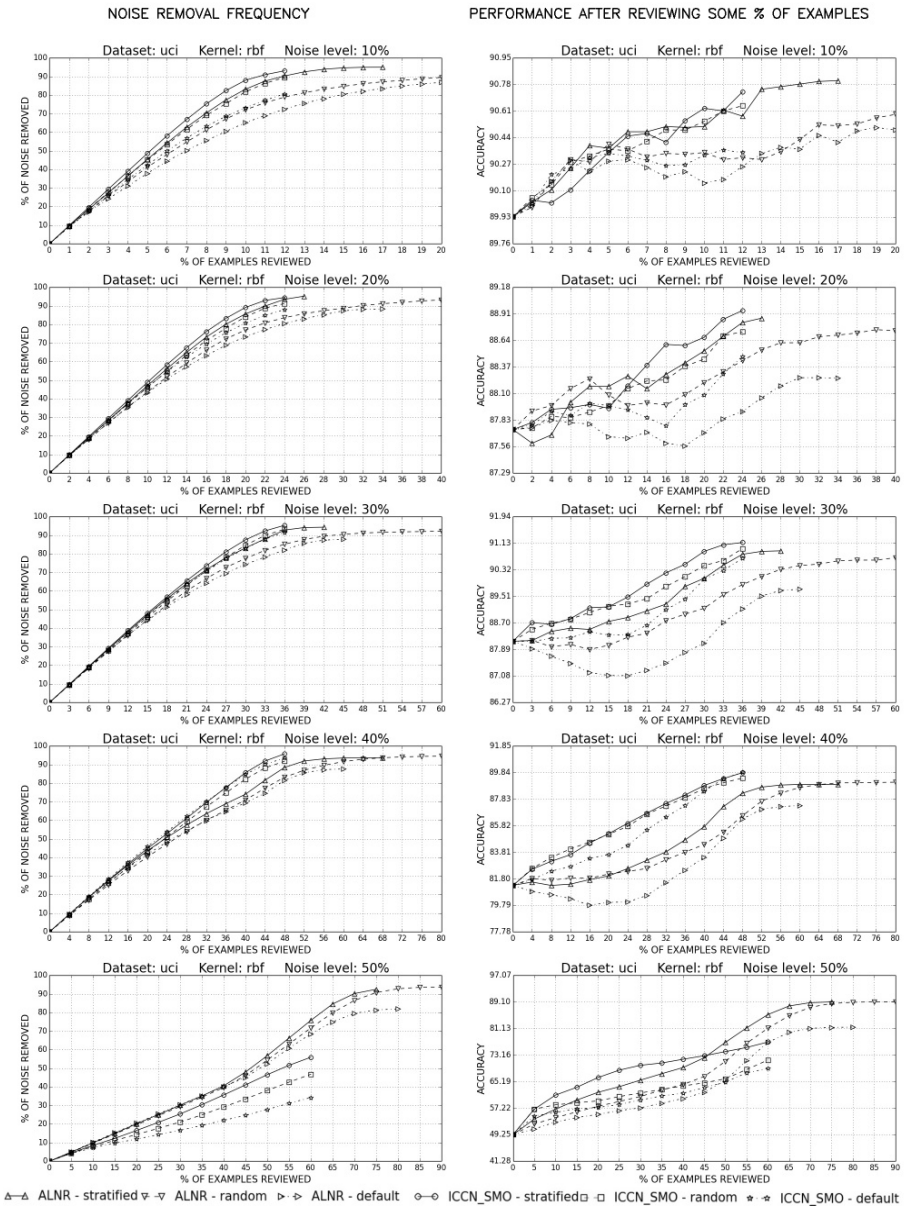

Figure 4: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the UCI Letter recognition dataset using the RBF Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

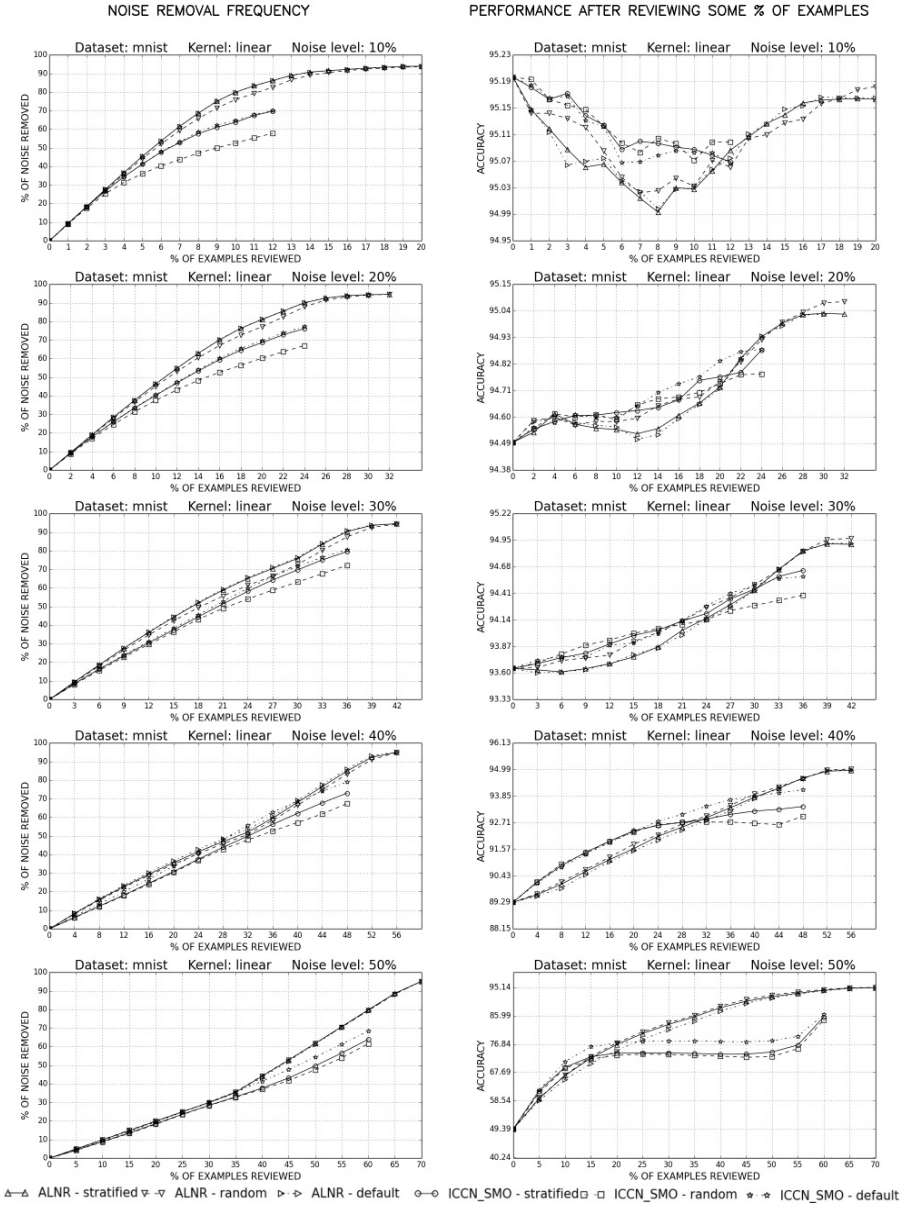

Figure 5: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the MNIST Digit recognition dataset using the Linear Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

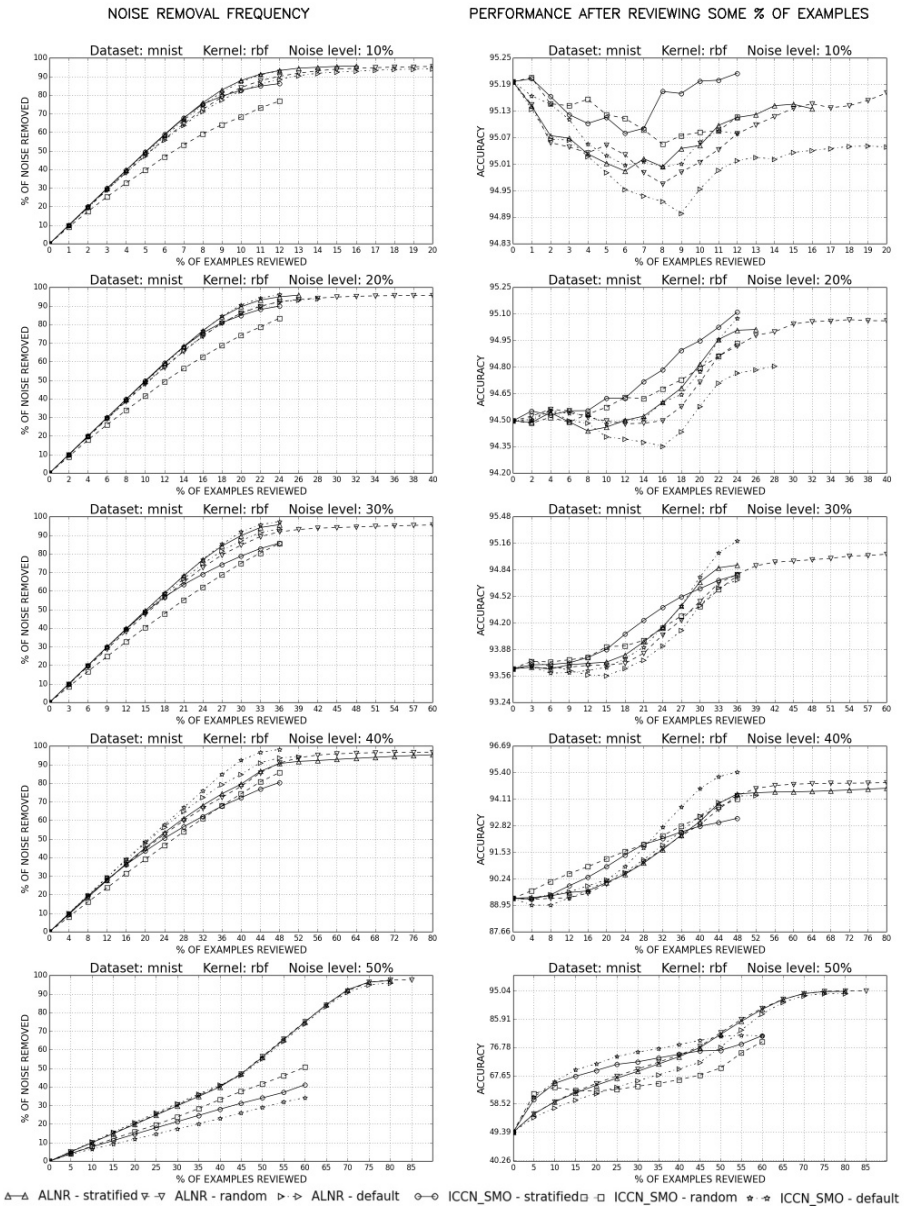

Figure 6: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the MNIST Digit dataset using the RBF Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

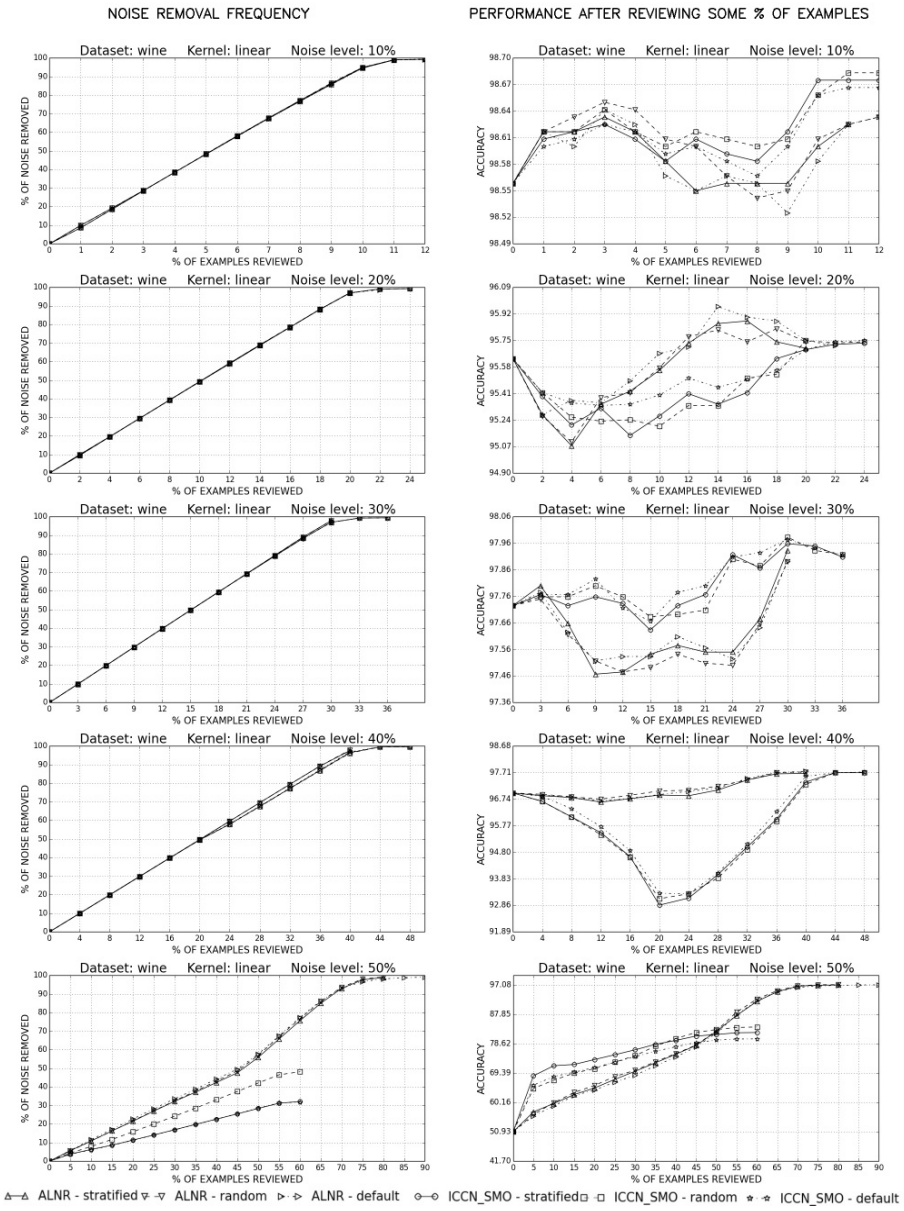

Figure 7: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the Wine Quality dataset using the Linear Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

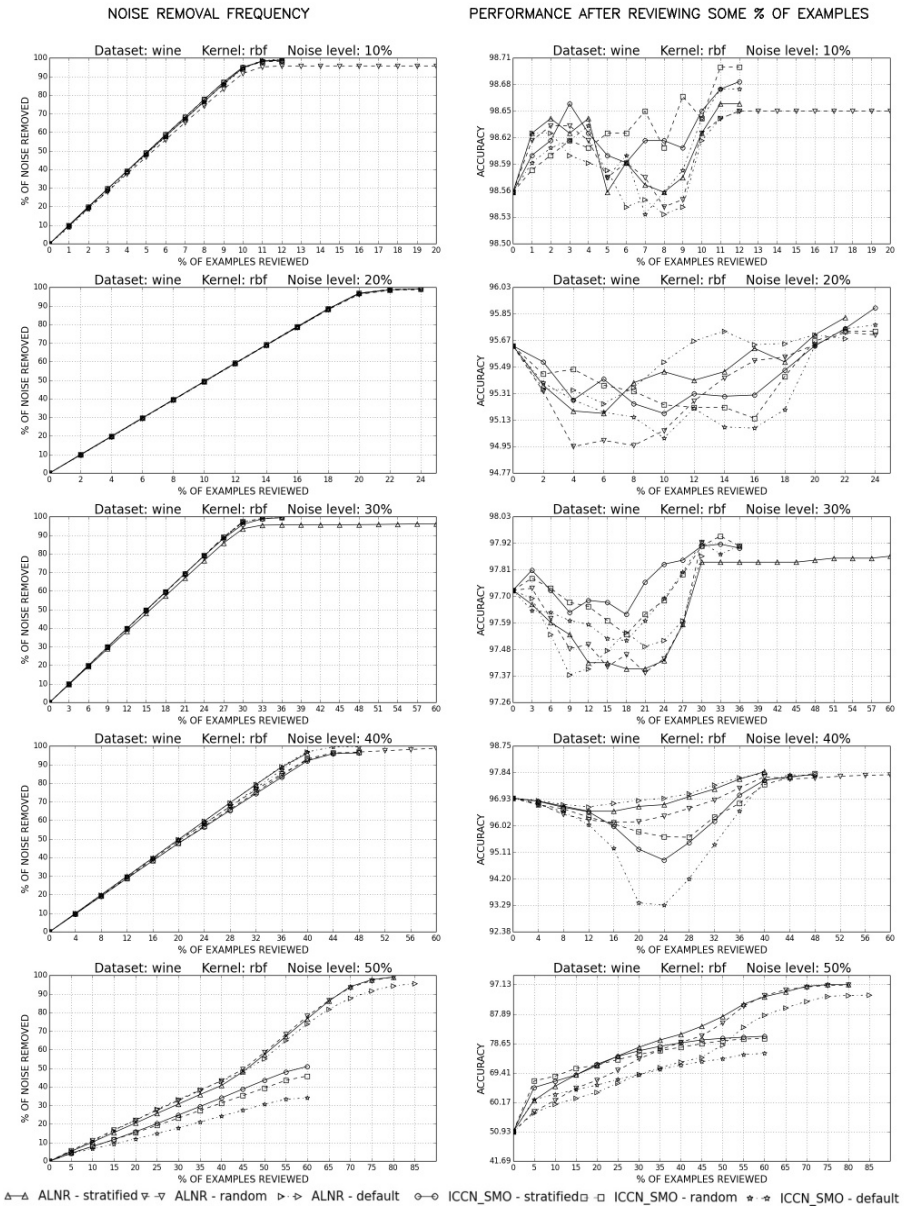

Figure 8: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the Wine Quality dataset using the RBF Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

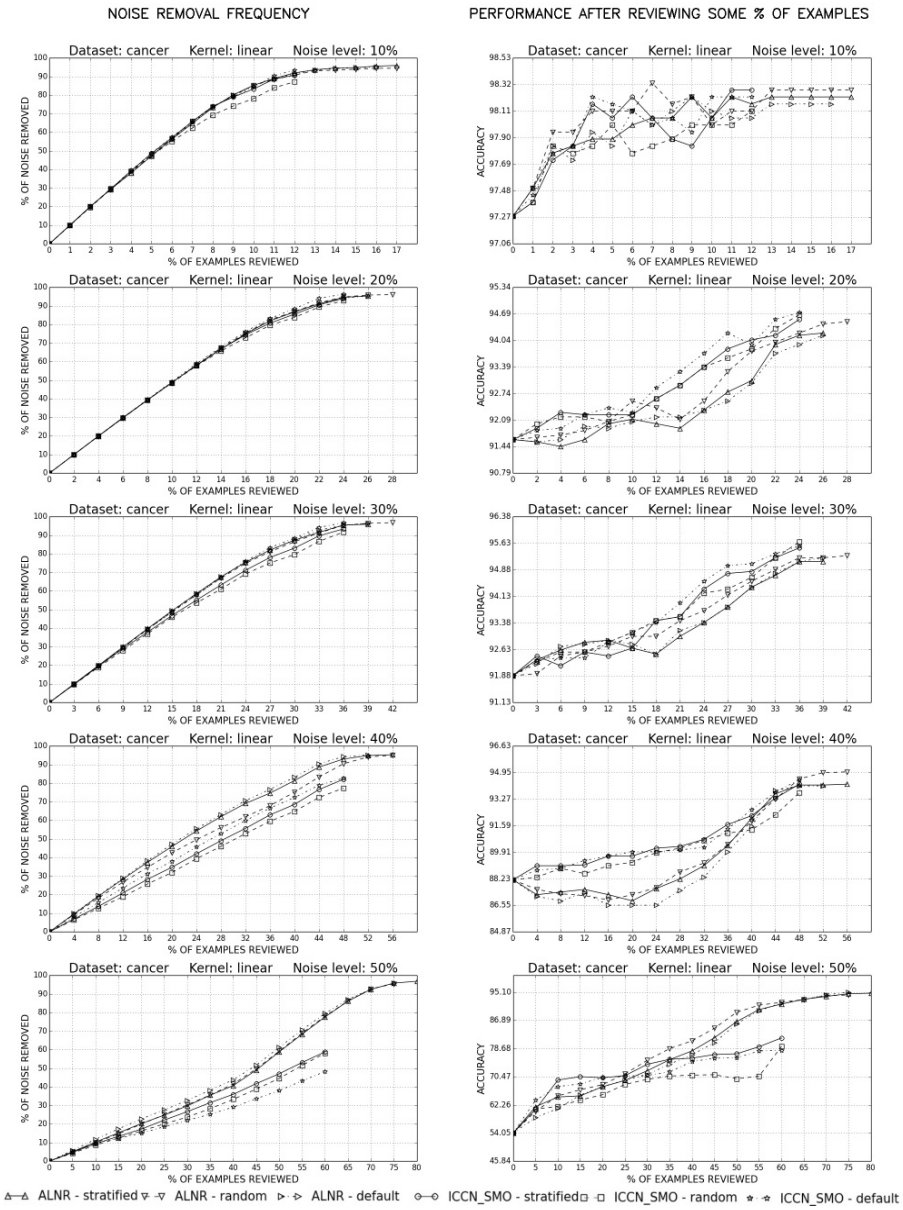

Figure 9: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the Breast cancer dataset using the Linear Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

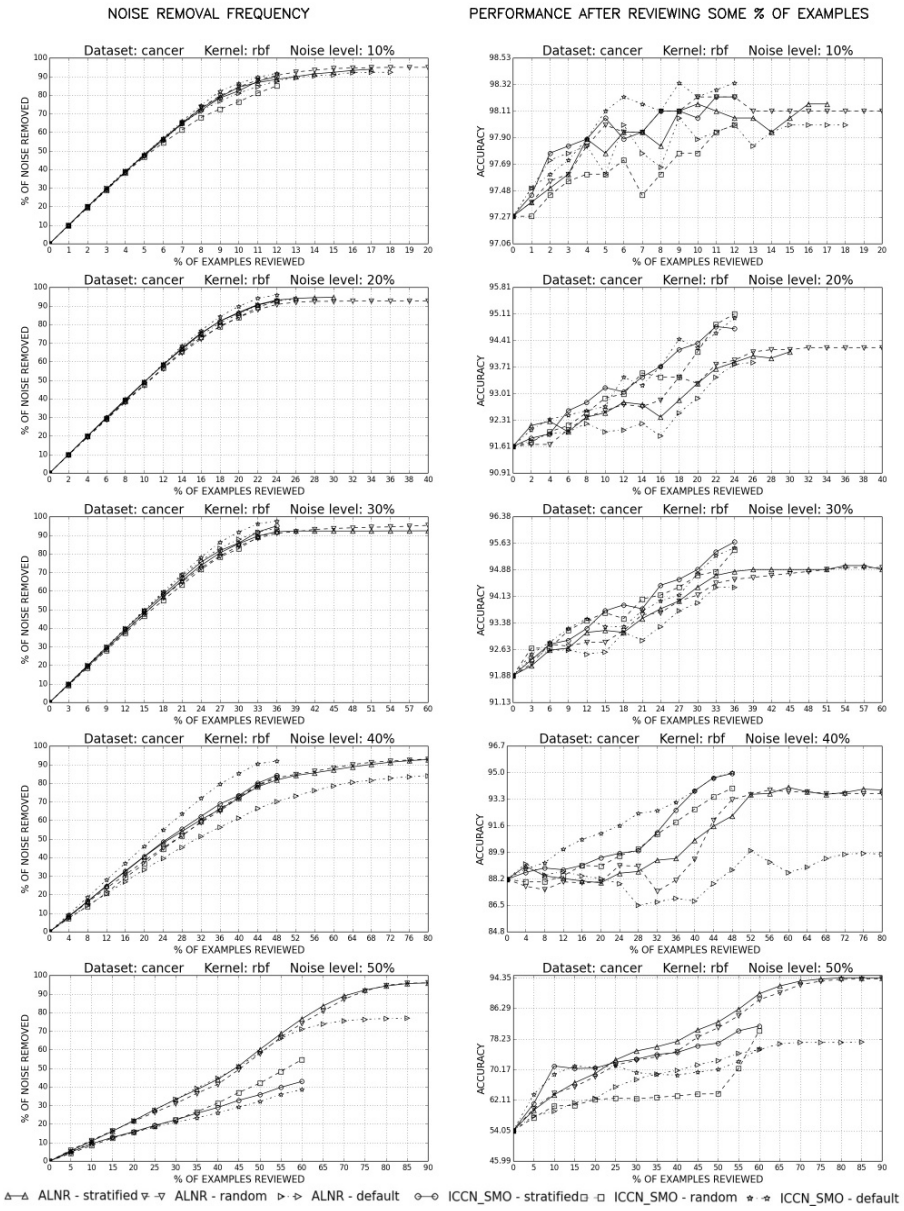

Figure 10: Comparison of ALNR and ICCN_SMO for different parameter selection methods on the Breast cancer dataset using the RBF Kernel SVM. The figures on the left show the noise removal performance at different noise levels and the figures on the right show the accuracy of the classifier on the specified test data after reviewing a fraction of the dataset.

References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

[3] V. Vapnik and S. Kotz. *Estimation of dependences based on empirical data*. Springer, 2006.

[4] I. Guyon, N. Matic, and V. Vapnik. Discovering informative patterns and data cleaning. *Advances in knowledge discovery and data mining*, In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, (Eds.): 181–203, 1996.

[5] U. Rebbapragada, R. Lomasky, C. E. Brodley, , and M. A. Friedl. Generating high-quality training data for automated land-cover mapping. In *International Geoscience and Remote Sensing Symposium*, volume 4. IEEE, 2008.

[6] U. Rebbapragada. *Strategic targeting of outliers for expert review*. PhD thesis, Tufts University, Medford, MA, 2010.

[7] D. Gamberger, N. Lavrac, and S. Dzeroski. Noise detection and elimination in data preprocessing: experiments in medical domains. *Applied Artificial Intelligence*, 14(2):205–223, 2000.

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

[9] X. Zhu, X. Wu, and Q. Chen. Eliminating class noise in large datasets. In *In International Conference on Machine Learning*, volume 3, pages 920–927, 2003.

[10] F. Muhlenbach, S. Lallich, and D. A. Zighed. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1): 89–109, 2004.

[11] H. Valizadegan and P. N. Tan. Kernel based detection of mislabeled training examples. In *In Proceedings of the Seventh SIAM International Conference on Data Mining*, 2007.

[12] U. Rebbapragada and C. E. Brodley. Class noise mitigation through instance weighting. In *In 18th European Conference on Machine Learning*, pages 708–715. Springer, 2007.

[13] U. Rebbapragada, L. Mandrake, K. L. Wagstaff, D. Gleeson, R. Castano, S. Chien, and C. E. Brodley. Improving onboard analysis of hyperion images by filtering mislabeled training data examples. In *Aerospace conference*, pages 1–9. IEEE, 2009.

[14] S. Fefilatyev, M. Shreve, K. Kramer, L. Hall, D. Goldgof, R. Kasturi, and H. Bunke. Label-noise reduction with support vector machines. In *21st International Conference on Pattern Recognition (ICPR)*, pages 3504–3508. IEEE, 2012.

[15] K. N. Le. A mathematical approach to edge detection in hyperbolic-distributed and gaussian-distributed pixel-intensity images using hyperbolic and gaussian masks. *Digital Signal Processing*, 21(1):162–181, 2011.

[16] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, , and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[17] C. E. Brodley, U. Rebbapragada, K. Small, and B. Wallace. Challenges and opportunities in applied machine learning. *AI Magazine*, 33(1):11–24, 2012.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[19] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10 (3):61–74, 1999.

[20] G. Ratsch, T. Onoda, and K. R. Muller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[21] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[22] J. Cao, S. Kwong, and R. Wang. A noise-detection based adaboost algorithm for mislabeled data. *Pattern Recognition*, 45(12):4451–4465, 2012.

[23] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. *Journal of Machine Learning Research-Proceedings Track*, 20:97–112, 2011.

[24] G. Stempfel and L. Ralaivola. Learning svms from sloppily labeled data. In *International Conference on Artificial Neural Networks*, pages 884–893. Springer, 2009.

[25] E. Niaf, R. Flamary, C. Lartizien, and S. Canu. Handling uncertainties in svm classification. In *Statistical Signal Processing Workshop*, pages 757–760. IEEE, 2011.

[26] B. Schlkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[27] H. Lukashevich, S. Nowak, and P. Dunker. Using one-class svm outliers detection for verification of collaboratively tagged image training sets. In

International Conference on Multimedia and Expo, pages 682–685. IEEE, 2009.

[28] K. Das, K. Bhaduri, and P. Votava. Distributed anomaly detection using 1-class svm for vertically partitioned data. *Statistical Analysis and Data Mining*, 4(4):393–406, 2011.

[29] J. Mourao-Miranda, D. R. Hardoon, T. Hahn, A. F. Marquand, S. C. Williams, J. Shawe-Taylor, and M. Brammer.

[30] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methodssupport vector learning*, 3, 1999.

[31] R. E. Fan, P. H. Chen, and C. J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[32] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[33] A. Borji, M. Hamidi, and F. Mahmoudi. Robust handwritten character recognition with features inspired by visual ventral stream. *Neural processing letters*, 28(2):97–111, 2008.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[35] C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[36] L. Bottou and C. J. Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.