2012

Finding the Boundary of Use to Improve Repair Mechanisms for Supervised Learning Systems

L. Dee Miller
University of Nebraska-Lincoln, lmille@cse.unl.edu

Leen-Kiat Soh
University of Nebraska-Lincoln, lsoh2@unl.edu

Finding the Boundary of Use to Improve Repair Mechanisms for Supervised Learning Systems

L. Dee Miller and Leen-Kiat Soh
Department of Computer Science and Engineering
University of Nebraska—Lincoln
Lincoln, NE, USA
{lmille, lksoh}@cse.unl.edu

Abstract

Numerous "repair" mechanisms have been developed to improve the training data for supervised learning (SL) systems including feature selection, noise correction, and active learning. These repair mechanisms myopically repair instances as long the estimated system performance continues to improve. Such general repair can lead to unnecessary repairs and overfitting from repair which can lower system performance on new instances. We propose a Boundary of Use (BoU) meta-reasoning framework to decide which instances should be repaired. This framework uses a semi-supervised clustering approach to partition the training instances into regions where the SL system does well without repair, regions where it makes some mistakes, and regions where repair is deemed hopeless. Repair is then applied selectively to only mixed regions. We demonstrate that BoU-enhanced versions of repair improve SL system performance on 21 UCI datasets where general repair has varying degrees of unnecessary repair and overfitting.

1. Introduction

Supervised learning (SL) systems learn a function from existing, labeled (i.e., training) instances which can be used to predict the correct labels for new instances (Mitchell, 1997). In the past few decades, there has been considerable work on SL systems including advances in existing algorithms such as decision trees and artificial neural networks (Kotsiantis, 2007). New algorithms have also emerged including support vector machines and boosting (Caruana and Niculescu-Mizil, 2006).

Despite these advances, there are still several weaknesses which all SL systems must contend with to one degree or another. These weaknesses involve assumptions on the training instances. First, they assume that all the features in the instances are relevant to the learned function. However, irrelevant features are common in real-world datasets and can reduce SL system performance (Sayes et al., 2007). Second, they assume that all the labels for the training instances are correct. However, data-entry errors, subjectivity, etc., can all result in errors in the labels (Pechenizkiy et al., 2006) and reduce SL system performance. Third, they assume that the training data includes sufficient instances to learn the function. However, costs associated with the labels can limit the number of training instances (Settles, 2010) and reduce SL system performance (Donmez & Carbonell, 2008).

One widely used approach to address these weaknesses is developing algorithms to "polish" the training instances, and then retrain the learned function using the "repaired" training instances. We collectively refer to these as repair mechanisms in this paper. For example, *feature selection* algorithms detect and remove irrelevant features. Another example is *noise correction* algorithms that identify noisy labels and then remove or replace them. Finally, *active learning* algorithms choose the instances which most improve the SL system.

However, the above repair mechanisms tend to myopically repair training instances as long as the overall estimate of SL system performance (e.g., training accuracy) increases after its learned function is retrained. They never pause to consider which types of instances should be repaired resulting in two sub-problems: unnecessary repairs and overfitting. Both of these contribute to *lower system performance on new instances.*

First, repairs are often applied to *all* instances without singling out those in need of repair. This can lead to unnecessary repairs on some instances where the learned function already does well. When such repairs make significant changes, the function could lose the capability to predict the correct label on similar, new instances. This sub-problem helps explain limitations to repair including lack of robustness in feature selection (Sayes et al., 2008) and in noise correcting models (Li et al., 2007).

Second, repairs are often repeated on instances where the learned function struggles to predict the correct label even with repair. Such repairs to accommodate especially troublesome instances *could increase the chance of overfitting* and lower performance after the learned function is

retrained to fit these repaired instances. This has been observed in repair mechanisms including feature selection (Sayes et al., 2007), noise correction (Li et al., 2007), and active learning (Mesterharm & Pazzani, 2011).

Therefore, we propose a framework called the Boundary of Use (BoU) that is designed specifically to *decide which instances should be repaired in order to enhance the repair mechanisms*. This framework first uses a semi-supervised clustering approach to partition the training instances into three different types of BoU regions: (1) *correct* regions, where the learned function predicts the correct label even without repair, (2) *mixed* regions, where it makes some mistakes and (3) *incorrect* regions, where it struggles to predict the correct label. Then, our approach applies *a single type of repair mechanism (e.g., feature selection) applied separately on* ***only*** *the mixed regions*. This helps avoid unnecessary repairs on instances where the learned function already does well and overfitting on troublesome instances. Further, the BoU maintains separate versions of the function for each region to prevent repairs from affecting other regions.

In the following, we investigate the effectiveness of the BoU framework with two objectives. Objective 1 is to *demonstrate that BoU-enhanced repair mechanisms are better than representatives of successful repair mechanisms*. To this end, we investigate three types of repair mechanisms including an ensemble-based feature selection approach (Sayes et al., 2008), the QcleanNOISE algorithm for reducing label noise without using the SL system (Daza and Acuna, 2007), and density-based active learning for choosing new training instances (Settles, 2010). Objective 2 is to *investigate the impact of BoU-enhanced repair mechanisms on different types of SL systems*. We use three SL systems with very different properties: artificial neural networks (ANNs) (Mitchell, 1997), decision trees (Mitchell, 1997), and support vector machines (SVMs) (Christianini and Shawe-Taylor, 2000).

Note that our present study does not involve AdaBoost (Freund & Schapire, 1995) which is conceptually similar to our BoU framework but with two important differences. First, AdaBoost is generally used by repair mechanisms on *all* the training data, whereas BoU is used to decide which instances should be repaired. Second, AdaBoost generally uses multiple functions trained on the *same* instances, whereas the BoU uses a single function for each region. Since we are interested in demonstrating the impact of selective repair on individual functions rather than general repair on multiple functions we do not use AdaBoost.

The rest of this paper is organized as follows. Section 2 provides background on the repair mechanisms and SL systems used in our study. Section 3 describes the BoU framework and our SSC-based approach. Section 4 discusses the experimental setup and results. Finally, we conclude and outline our future work.

2. Background and Related Work

2.1 Supervised Learning (SL) Systems

We consider three types of SL systems in the experiments below. First, *artificial neural networks* (ANNs) learn a vector of weights on features in the dataset to choose the labels for new instances (Kotsiantis, 2007). ANNs consist of multiple nodes connected to threshold functions or to additional layers of nodes. ANNs are updated iteratively (e.g., using gradient descent) until they correctly predict the labels for the training instances. Second, *decision trees* (for classification) learn a tree data structure to generate the labels for new instances (Kotsiantis, 2007) . The decision tree first selects one feature as the root node and adds an edge for every label value. The decision tree continues to add nodes and edges recursively until all instances have been sorted into groups with similar labels. The leaves are then set to the common label. Third, *support vector machines* (SVMs) learn a hyperplane to separate instances such that those on the same side mostly have the same label (Kotsiantis, 2007). SVMs first use a kernel function to transform all values for the dataset into higher dimensional space where they are linearly separable (Cristianini & Shaw-Taylor, 2000). Then, the SVM attempts to maximize the distance (i.e., margin) between instances with different labels.

2.2 Repair Mechanisms

2.2.1 Feature Selection

Feature selection (FS) mechanisms are designed to detect and remove irrelevant features. FS consists of two basic steps (Liu & Yu, 2005): (1) generate the current subset of relevant features, (2) evaluate the current subset using an evaluation criterion retaining the best subset. FS repeats both steps until some stopping criterion is met and then removes any features not in the best subset. Recent reviews (Liu & Yu, 2005; Sayes et al., 2007) categorize FS methods based on the search strategy and the evaluation criterion. Search strategies are *complete* such as best-first, *sequential* such as hill-climbing, or *random* such as genetic algorithms. Evaluation criteria are based on *filters* or *wrappers*. Filters evaluate the relevant features using only the intrinsic properties of the data whereas wrappers use SL system performance (Liu & Yu, 2005).

We use ensemble-based FS in our experiments which has previously been shown to give improved performance over individual algorithms (e.g., Sayes et al. 2008; Tuv et al. 2009). Our ensemble is based on Sayes et al. (2008) and consists of correlation FS with best-first search, sequential RELIEF, and consistency feature selection with a genetic algorithm.

2.2.2 Noise Correction

Noise correction mechanisms are designed to identify noisy labels and then remove or replace them. Noise correction has been less widely studied than FS, but there are still two general types (Pechenizkiy et al., 2006). First,

noise tolerant correction modifies existing SL systems to accommodate noisy labels (e.g., rule-post pruning for decision trees which remove noisy labels after training). Second, filtering techniques detect and correct noisy labels before the learned function is trained. Li et al. (2007) assume a probabilistic noise model and solve using a kernel fisher method to create a discriminant function in projected space. Daza & Acuna (2007) propose the QcleanNOISE algorithm which identifies noisy labels using *k*-Nearest Neighbor (*k*NN) by first assuming that instances farther from the center potentially have noisy labels. The algorithm then identifies the *k*NNs for those with potentially noisy labels. When neighbors agree on a different label, the instance's label is considered to be noisy.

We use QcleanNOISE algorithm for filtering repair in the experiments below for two reasons. First, this algorithm does not require a separate version for each SL system which could bias the results. Second, this algorithm does not make any assumptions on the distribution of noisy labels (e.g., Gaussian in Li et al. (2007)) which may not hold for real-world datasets.

2.2.3 ACTIVE LEARNING

Active learning (AL) queries the labels for new instances and adds them to the training data. Briefly, this consists of an iterative cycle where repair chooses the most informative instance, obtains the instance's label from an oracle (e.g., human expert), and adds this instance to the training data. Settles (2010) categorize AL based on the sampling and query strategies used. Sampling strategies include sampling instances from a stream or pool of unlabeled instances. Query strategies consider uncertainty in the label, and instance density in feature space (Settles, 2010). AL has been extensively studied in recent years, but most of this work has focused on optimizing limited resources for obtaining labels. Here we are interested in using AL as a repair mechanism to query new instances on demand in order to improve the learned function.

In our experiments, we use the pool-based sampling strategy for AL with both query strategies (Settles, 2010). This AL focuses queries on instances where the learned function is uncertain and instances in close proximity to other instances.

3. Methodology

The BoU is designed as a meta-reasoning framework for SL systems. The purpose of the BoU is to first identify the boundary separating instances where the system's learned function is doing well from those where the function is struggling. Then, the BoU framework refines the learned function on instances outside the boundary while keeping the original function for instances inside. For application to repair mechanisms, such refinements involve applying the repair and retraining the learned function on instances outside the boundary.

The BoU framework identifies the boundary, first, by partitioning the training data into *regions* with similar instances. These regions allow repair mechanisms to focus on making fine-tuned repairs on instances with similar features rather than repairs scattered across the entire dataset. At the same time, these regions contain sufficient instances to make the retrained function generalizable on new instances with similar features. Next, the BoU evaluates the learned function in each region by comparing the predicted and actual labels for the member instances. The BoU flags *an instance's label as* ***correct*** *when the predicted matches the actual*; otherwise, *the BoU flags the label as* ***incorrect***. The BoU uses the distribution of correct and incorrect labels to decide the region's relation to the boundary (cf., Figure 1). This results in three types of regions: (1) those with predominately *correct* labels (Inside the Boundary, i.e., R_1-R_2), (2) predominantly *incorrect* labels (Outside, i.e., R_6-R_7), and (3) those with mixed labels (Border, i.e., R_3-R_5).

Figure 1 gives an example of seven BoU regions (R_1-R_7). Each region contains member instances with correct (clear) or incorrect (grey) labels. In particular, the figure shows R_4 in greater detail with the predicted labels from its function (f_4) along with the actual labels. This figure highlights a *key* difference between our regions and traditional clusters, namely, that a single region is *intended* to have instances with different labels. By identifying a region without seeking for most of its instances to be of the same label, a BoU region is essentially a "compact" cluster of instances as "evaluation candidates" to assess the performance of the learned function in that region. That is, a learned function of a region is evaluated by comparing the actual and predicted labels for members in the region. In turn, this localized evaluation allows meta-reasoning to identify where the learned function is struggling (e.g., in region R_4) and make refinements to it on those regions. For example, meta-reasoning could realign the learned function f_4 on R_4 such that all the +1 instances are on one side of the margin of the function and all the -1 instances are on the other. As a result, this rotation or refinement would lead to better label predictions of new instances that fall into the R_4 region. Without using the BoU regions, it would be difficult to make the retrained function generalizable on new instances with multiple labels. Further, pure clusters could lack sufficient instances to avoid overfitting the retrained function on the members.

As alluded to earlier, BoU-enhanced repair uses these regions to allow for *selective* repair which addresses both unnecessary repairs and overfitting. BoU-enhanced repair addresses unnecessary repairs by repairing instances and retraining the function separately on only mixed regions. In this way, the learned function is left alone on correct regions where it is already doing well. Selective repair can also help reduce overfitting caused by the extensive repairs necessary to get the learned function to work well on incorrect regions. Note that retraining the function

requires mixed regions to keep separate versions of the learned function. After repairs are complete, the label for a new instance is predicted by first assigning it to the region containing the most similar instances and then using the learned function for that region.

In summary, BoU-enhanced repair uses four basic steps for region-based repair: (1) Train the learned function on all the training data; (2) Create regions using the performance results of the learned function on the training data; (3) Apply repair separately to each mixed region, leave incorrect and correct regions untouched; and (4) Retrain the learned function separately on those mixed regions.

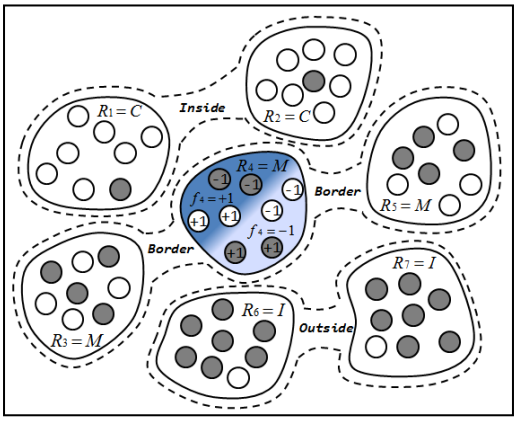

Figure 1. Example BoU Regions and Relationship to Boundary. Regions include member instances and type. Grey circle members have incorrect labels, while clear circles have correct. Types are correct (C), incorrect (I), and mixed (M). Region R_4 also includes both the actual and predicted (f_4) labels.

We use a semi-supervised clustering (SSC) approach for creating the BoU-enhanced repair mechanism. This approach is based on hierarchical clustering algorithms. These algorithms generate a dendrogram with multiple sets of clusters starting with all the instances in a single cluster and splitting these clusters until some stopping condition is reached (i.e., divisive clustering). We use two general guidelines based on SSC to decide when to stop divisive clustering. First, to avoid unnecessary repairs, we stop splitting early when clusters have high purity with predominately correct or incorrect labels (purity stop). The use of these labels is what makes our clustering approach semi-supervised. Second, to avoid overfitting, we stop splitting when clusters lack sufficient coverage based on the percentage of the training data they contain to allow for generalizable learned functions (coverage stop).

At each layer in the dendrogram, clusters are first converted into regions with assigned type (i.e., "correct", "incorrect", or "mixed") based on the results of the learned function on its training data. Then, the repair and retrain steps are performed on the mixed regions. These steps result in a separate version of the learned function for each mixed region. Correct and incorrect regions skip both steps and instead use the learned function from their parent region. Figure 2 provides an example of divisive SSC with the types, repair, and learned function for each region. This figure shows how the regions are split and the learned functions are propagated or revised from one layer to the next. The original, mixed region starts with the learned function on all the training data. After the repair (r_1) and retrain steps (f_1), this region is split into correct, mixed, and incorrect regions. The correct and incorrect regions use the learned function from the parent, while the mixed region goes through its own repair and retrain steps before being split again. Overall, the tight integration of all four steps allows for repairs to be incorporated into the regions and evaluated on subsequent layers. This allows for even more selective repair than repairs applied only after clustering as in Miller (2007).

Some may argue that we are actually introducing overfitting when creating the regions. To address this, we use the above SSC approach to prevent the labels from having too much influence on the regions. Regions are created initially based on feature similarity. The labels are held back and used only to "validate" the repairs based the region types. For example, if the region's type changes from mixed to correct (after repair) then repair is deemed successful and we stop splitting on that region.

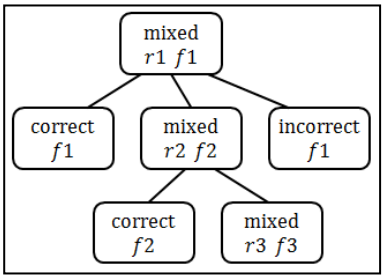

Figure 2. Region Splits from our Divisive SSC along with Type, Repair (r), and Learned Function (f).

We now present the algorithm for our divisive SSC approach with the pseudocode given in Figure 3. This algorithm starts with a single region with all the training instances and its learned function and then runs recursively to create the dendrogram of regions. At the end, the results are one or more regions and their learned functions used for the BoU-enhanced repair.

The program starts with deciding whether repairs are necessary based on the region's type. In Line 1, the type is assigned using Equation (1) below based on the $purity(R_i, f_i)$ for function f_i in region R_i where δ is a threshold to capture the purity requirement of a region:

$$type(R_i, f_i) = \left\{ \begin{matrix} \text{mixed } if\ purity(R_i, f_i) < \delta \\ \text{correct/incorrect } otherwise \end{matrix} \right\} \quad (1)$$

If the region's type is mixed, repair is applied to the region's training data and its function is retrained on the repaired data (lines 2-3). Subsequently, the region's type is updated based on the repairs (line 4). If the region's type is still mixed, the algorithm splits the region into two new regions and attempts to invoke the same clustering

function recursively on them. If a region is not mixed, then there is a purity stop, either because it is not mixed in the first place or that repairs were effective, then the algorithm returns the repaired region and its retrained function. Lastly, the actual splits are done with *k*-Means clustering algorithm (line 5). If both the split regions meet the minimum coverage requirement (line 6), containing a percentage of the training data above the threshold φ, then the algorithm is run recursively on the split regions with the parent's retrained function. Otherwise, there is a coverage stop because there are insufficient instances for training—the algorithm returns the parent region and its function. Note in subsequent recursive calls, correct and incorrect regions will use this function directly, while mixed regions will get their own (lines 2-3).

$R_0 = Region\ for\ training\ data$
$f_0 = Learned\ function\ on\ R_0$
$M = Repair\ mechanism$
$S = Supervised\ learning\ system$
function $BoUKMeans(R_0, f_0)$ ***returns*** R_0' *and* f_0'
(1) ***if*** $type(R_0, f_0) == mixed$ // check purity stop
(2) $R_0' \leftarrow repair(R_0, M)$
(3) $f_0' \leftarrow retrain(R_0', S)$
(4) ***if*** $type(R_0', f_0') == mixed$ // check purity stop
(5) $R_1, R_2 \leftarrow KMeans(R_0', 2)$ // split the region
(6) ***if*** $coverage(R_1) > \varphi$ ***and*** $coverage(R_2) > \varphi$ // check coverage stop
(7) $BoUKMeans(R_1, f_0')$
(8) $BoUKMeans(R_2, f_0')$
(9) ***end if***
(10) ***end if***
(11) ***else***
(12) $R_0' \leftarrow R_0$
(13) $f_0' \leftarrow f_0$
(14) ***end else***

Figure 3: Divisive Semi-Supervised Clustering Algorithm for BoU-Enhanced Repair.

3.1 Integrating Individual Repair Mechanisms

In our approach, we separately used three types of repair mechanisms: feature selection, noise correction, and active learning. Here we discuss how each is integrated into the regions for BoU-enhanced repair in the experiments.

Feature selection (FS) mechanisms are designed to remove features which are irrelevant to the labels. Since there is no "gold-standard", we use an ensemble of three feature selection algorithms similar to Sayes et al. (2008). Each algorithm votes on whether it thinks a feature is relevant. Features considered to be relevant by the majority of these algorithms are kept. Other features are considered to be irrelevant and removed from the training data for *that* region before the learned function is retrained.

Noise correction mechanisms are designed to identify noisy labels and then remove or replace them. We use the QcleanNOISE algorithm which identifies noisy labels using a combination of clustering and *k*-Nearest Neighbor (*k*NN). Instances which are considered to have noisy labels are not used when the learned function is retrained.

Active learning (AL) mechanisms are designed to query the instances which most improve the learned function. We use pool-based AL which starts with the same type of learned function trained on a small subset of the instances in the region (n=5 from Settles (2010)). This repair ranks all the remaining instances in the region with a heuristic which favors instances which are hard for the learned function in close proximity to other instances. AL then iteratively adds the highest ranked instances to the subset until it contains half the instances in the region. The learned function is then retrained using only the instances in the subset. The normal AL version queries half the total instances. This allows for a fair comparison because BoU-enhanced repair queries at most the same number.

4. Implementation and Results

In the experiments below, we investigate both objectives using all three types of repair mechanisms and all three types of SL systems previously discussed. We want to demonstrate that BoU-enhanced repairs are effective on repair mechanisms which "polish" the training data with different goals and on SL systems which generate very different learned functions. For repair mechanisms, we use ensemble-based feature selection (FS; Sayes, et al. 2008), QcleanNOISE noise correction (NC; Daza & Acuna, 2007), and density-based active learning (AL; Settles, 2010). We use Java implementations for all three repair mechanisms based on the original papers. For SL systems, we use artificial neural networks (ANN), support vector machines (SVM), and decision trees (DT). We use the Java implementations for all three from the Weka library with the parameters suggested in Hall et al. (2009).

The BoU-enhanced repair mechanisms use the same type of repair mechanism and SL system plugged into our divisive SSC algorithm (cf., Section 3). We use the following parameters for this algorithm. First, we use δ=0.9 as the purity required for correct or incorrect regions. Second, we use φ=0.3 as the coverage percentage required to continue clustering. Finally, we use k=2 for the number of clusters from the actual split. These values are based on those in Miller (2007).

The experiments below compare the SL system performance for both general repair (on all the instances) and BoU-enhanced repair on 21 benchmark datasets from the UCI machine learning repository. All system performance results are based on the average test accuracy. To better estimate this performance, accuracy is measured over thirty (30) runs using ten-fold cross validation with three different random seeds. These results are then averaged again over either the three SL systems (Objective 1) or the repair mechanisms (Objective 2).

Before we investigate the two objectives, we highlight the results in terms of repair problems and number of regions. Table 1 gives the names, repair problems, and number of regions for all datasets.

First, the repair problems help demonstrate the effectiveness of BoU-enhanced repair on datasets where general repair encounters varying degrees of the unnecessary repair and overfitting problems. The repair problem values in Table 1 are the percentage of the 90 folds (across all three SL systems) where general repair encounters unnecessary repairs (UR; Eq. 2) or overfitting (OF; Eq, 3). Both are measured using the training (trn_acc) and test accuracy ($test\_acc$) for the learned functions with (f_w) and without ($f_{w/o}$) general repair.

$$UR = test\_acc(f_{w/o}) > test\_acc(f_w) \quad (2)$$

$$OF = trn\_acc(f_{w/o}) < trn\_acc(f_w) \textbf{ and } test\_acc(f_{w/o}) > test\_acc(f_w) \quad (3)$$

Second, the number of regions helps demonstrate the effectiveness of BoU-enhanced repair on datasets (1) where it applies repair and retrains the learned function separately on multiple regions and (2) datasets where it decides not to repair and uses the function on the original region (i.e., without splitting the regions). The number of regions in Table 1 is the range across all folds. To facilitate comparison, in Tables 1-3, datasets with multiple regions are on top of the double line while those with a single region are on the bottom.

Table 1: Repair Problems and Range of Regions (Reg.) on all Datasets. Problems are the percentage of 90 folds with unnecessary repair (UR) or overfitting (OF) across all three SL systems.

	UR			OF			
Dataset	FS	NC	AL	FS	NC	AL	**Reg.**
arrhythmia	0.28	0.43	0.45	0.10	0.27	0.18	1-7
blood	0.49	0.31	0.30	0.01	0.31	0.06	1-3
bupa	0.76	0.47	0.38	0.00	0.41	0.12	4-6
contraceptive	0.31	0.84	0.43	0.07	0.84	0.16	1-4
credit	0.41	0.26	0.48	0.00	0.26	0.24	1-5
mammograph	0.00	0.44	0.59	0.00	0.44	0.16	2-3
parkinsons	0.21	0.16	0.32	0.06	0.09	0.28	1-3
pima	0.30	0.53	0.50	0.01	0.53	0.21	4-6
prognostic	0.46	0.31	0.44	0.00	0.29	0.23	1-2
vertebral	0.20	0.49	0.44	0.01	0.47	0.18	1-3
yeast	0.54	0.43	0.44	0.08	0.43	0.20	1-2
diagnostic	0.30	0.28	0.37	0.02	0.26	0.17	1
e-coli	0.02	0.10	0.18	0.02	0.10	0.06	1
car	1.00	0.27	0.53	0.00	0.08	0.24	1
ionosphere	0.31	0.67	0.52	0.03	0.48	0.32	1
monks-1	0.00	0.13	0.46	0.00	0.01	0.21	1
monks-2	0.04	0.13	0.11	0.00	0.04	0.02	1
monks-3	0.70	0.14	0.27	0.00	0.00	0.00	1
sonar	0.46	0.38	0.53	0.06	0.11	0.13	1
spect	0.20	0.49	0.32	0.03	0.49	0.24	1
tic	0.71	0.41	0.52	0.00	0.07	0.19	1

4.1 Objective 1: BoU-Enhanced Repair Mechanisms

Table 2 summarizes the system performance for general repair and BoU-enhanced repair on all datasets. We first consider the datasets above the double line where BoU-enhanced repair applies repair and retrains the learned function separately on multiple regions. We observe that using BoU-enhanced repair mechanisms boosts system performance for all repair mechanisms compared to general repair. BoU-enhanced version improves performance on 7/11 datasets for FS, 5/11 for NC, and 7/11 for AL. These results demonstrate improvements to the repair mechanisms since the results are averaged across SL systems with different learned functions, supporting the effectiveness of BoU-enhanced repair on multiple regions.

Investigating more closely, we cross reference system performance results with the unnecessary repair (UR) and overfitting (OF) repair problems in Table 1. For FS and AL repair, BoU-enhanced repairs are effective on datasets with both extremes. For example, BoU-enhanced repair with FS significantly ($p <= 0.05$) improves system performance for *yeast* which has amongst the highest UR and OF and for *mammograph* with amongst the lowest UR and OF. Likewise, BoU-enhanced repair with AL improves the results on *blood* which has the lowest UR and OF and on *mammograph* and *parkinsons* with, respectively, the highest UR and OF. For NC repair, however, BoU-enhanced repair achieves slightly worse performance on *contraceptive* with the highest UR and OF and on *parkinsons* with the lowest UR and OF. After a closer look, we realize that both of the datasets have imbalanced labels distributions where more than 75% of the instances have one label while less than 25% of the instances have the other. Because of this imbalance, *k*NN NC tends to over-correct by changing the label of the instances in the minority. Conceptually, the BoU should be able to prevent such unnecessary repairs by separating these instances into smaller regions. But, this it is currently restrained by the coverage stop that requires a region to be of adequate size. This problem can be addressed by better preserving the instances with the minority labels—for example, modifying the SSC approach to favor clusters with instances predominantly of the same label.

We next consider the datasets below the double line where BoU-enhanced repair decides *not* to repair and uses the function from the original region. Again, we observe that using BoU-enhanced repair mechanisms boosts system performance for all three types of repair mechanisms compared to general repair. BoU-enhanced version improves performance on 8/10 datasets for FS, 5/10 for NC, and 8/10 for AL. These datasets are relatively simple classification problems for all three SL systems even without repair. Such results show the effectiveness of BoU-enhanced repair deciding when not to repair, supporting the use of our approach as a meta-reasoner.

Again, we cross reference the performance results with the UR and OF problems to provide more evidence. For FS and NC repairs, BoU-enhanced repairs are effective on datasets with both extremes of the UR and OF problems. BoU-enhanced repair with FS significantly ($p <= 0.05$) improve the results on *tic* and *sonar* with, respectively,

the highest UR and OF and on *monks-1* with the lowest UR and OF. Likewise, BoU-enhanced repair with NC improves results on *spect* which has amongst the highest UR and OF and *monks-1* which has amongst the lowest. For AL repair, BoU-enhanced repair achieves significantly higher results for *ionosphere* with the highest UR and OF. However, the results are significantly worse for *monks-2* where instances with the same label are widely dispersed. This label dispersion makes it difficult for AL to benefit from multiple regions because similar instances do not have similar labels. This causes BoU-AL to query an instance using only the instances in the same BoU region that might have different labels. Thus, our approach essentially trained the function using the wrong labels. This is a weakness that we plan to address in the future.

Table 2: System Performance for Repair and BoU-Enhanced Repair on all Datasets. Results are averaged across all three SL systems. Grey cells indicate which have higher system performance while (*) indicates significantly higher performance based on a *t*-test with $p <= 0.05$.

Dataset	FS	BoU-FS	NC	BoU-NC	AL	BoU-AL
arrhythmia	0.75*	0.70	0.70	0.70	0.69	0.71
blood	0.77	0.78	0.77	0.78	0.76	0.77
bupa	0.59	0.61*	0.62	0.64*	0.62	0.64
contraceptive	0.66	0.67	0.67	0.66	0.68*	0.66
credit	0.86	0.85	0.86	0.85	0.83	0.84
mammograph	0.77	0.80*	0.80	0.82*	0.78	0.81*
parkinsons	0.86	0.87	0.88	0.87	0.84	0.85
pima	0.76	0.75	0.75	0.75	0.75	0.74
prognostic	0.76	0.76	0.76	0.77	0.74	0.75
vertebral	0.74	0.83*	0.80	0.81	0.79	0.81*
yeast	0.62	0.65*	0.64	0.64	0.63	0.62
diagnostic	0.94	0.96*	0.96	0.96	0.95	0.96
e-coli	0.99	0.98	0.98	0.98	0.97	0.98
car	0.85	0.98*	0.97	0.98	0.96	0.97
ionosphere	0.87	0.90*	0.87	0.90*	0.85	0.88*
monks-1	0.75	0.98*	0.97	0.98	0.90	0.97*
monks-2	0.66	0.88*	0.87	0.87	0.88*	0.86
monks-3	0.97	1.00*	0.99	1.00	0.97	1.00*
sonar	0.75	0.80*	0.81	0.80	0.75	0.79*
spect	0.80	0.79	0.78	0.79	0.80	0.78
tic	0.74	0.94*	0.94	0.94	0.89	0.91*

4.2 Objective 2: Impact of BoU on SL Systems

Table 3 summarizes the system performance for functions from general repair and BoU-enhanced repair on all datasets. Again, we start with the datasets above the double line where BoU-enhanced repair uses multiple regions. We observe that using BoU-enhanced repair mechanisms boosts system performance for ANNs and SVMs compared to general repair. BoU-enhanced version improves performance on 8/11 datasets for ANNs and 5/11 for SVMs. When averaged across repair mechanisms with different goals, these results demonstrate consistent improvement to the learned functions. Such improvements support the positive impact of BoU-enhanced repair with multiple regions on ANNs and SVMs.

However, the BoU-enhanced repair did not improve the performance for DTs. Indeed, it reduces performance on 5/11 datasets for DTs compared to general repair. We believe that this is because the BoU-based design did not take SL system *post-processing* into account. Specifically, our DTs use rule post-pruning which removes leaves to simplify the learned function and reduce overfitting on the training data. As a result, DTs for larger regions with more leaves can still benefit from post-pruning to reduce overfitting, but DTs for smaller regions have fewer leaves and may be adversely affected by post-pruning. In our study, post-pruning on smaller regions may have simplified them to the extent that they no longer leverage repairs. That is, a tree pruned to a decision stump (with only one decision branching node) cannot leverage noise correction when all the instances are merged together. To demonstrate this, we look at two datasets with multiple regions: *mammograph* and *vertebral*. BoU-enhanced repairs did worse on mammograph, while on vertebral it did significantly better. For mammograph, 91% of the folds contained one or more regions with decision stumps whereas there were only 2% for vertebral. We further ran a quick comparison study by running additional tests using reduced post-pruning (pruning threshold = 0.01 in C4.5) and confirm that BoU-DT-with-reduced pruning indeed did better than DT (with regular pruning) on both datasets (0.83 vs. 0.82 on mammograph and 0.81 vs. 0.79 on vertebral). Therefore, to enjoy the benefits of region-based repair, we suggest the amount of post-pruning be conditioned on the size of the regions.

Table 3: System Performance for Functions Repaired with and without BoU on all Datasets. Results are averaged across all three repair mechanisms. Grey cells indicate which have higher system performance while (*) indicates significantly higher performance based on a *t*-test with $p <= 0.05$.

Dataset	ANN	BoU-ANN	SVM	BoU-SVM	DT	BoU-DT
arrhythmia	0.73	0.76*	0.70*	0.67	0.71*	0.69
blood	0.76	0.78*	0.77	0.77	0.77	0.77
bupa	0.62	0.64*	0.58	0.62*	0.63	0.63
contraceptive	0.67	0.67	0.65	0.65	0.69*	0.67
credit	0.85	0.85	0.85	0.85	0.84	0.85
mammo-graph	0.77	0.82*	0.77	0.80*	0.82	0.81
parkinsons	0.85	0.87*	0.87	0.88	0.85	0.84
pima	0.76	0.75	0.76	0.75	0.74	0.73
prognostic	0.75	0.77*	0.78	0.77	0.74	0.74
vertebral	0.80	0.82*	0.75	0.82*	0.79	0.81*
yeast	0.64	0.64	0.62	0.63	0.64	0.63
diagnostic	0.96	0.96	0.96	0.97	0.93	0.94
e-coli	0.99	0.98	0.97	0.98	0.98	0.98
car	0.90	0.93*	0.94	0.99*	0.94	1.00*
ionosphere	0.85	0.90*	0.87	0.89*	0.87	0.90*
monks-1	0.90	1.00*	0.92	1.00*	0.81	0.93*
monks-2	0.88	1.00*	0.87	0.98*	0.66*	0.63
monks-3	0.99	1.00	0.99	1.00	0.95	1.00*
sonar	0.80	0.83*	0.80	0.83*	0.72	0.71
spect	0.79	0.79	0.79	0.79	0.80	0.80
tic	0.90	0.97*	0.90	0.99*	0.77	0.83*

Finally, we consider datasets below the double line where BoU-enhanced repair decides not to repair. We observe that using BoU-enhanced repair mechanisms boosts system performance for all three types of SL system compared to general repair. BoU-enhanced version improves performance on 7/10 datasets for ANNs, 9/10 for SVMs, and 6/10 for DTs. Again, these results demonstrate consistent improvement to the learned functions. Such results support the positive impact of BoU-enhanced repair deciding when not to repair.

5. Conclusions and Future Work

To summarize, researchers have created repair mechanisms to address weaknesses with SL systems (e.g., irrelevant features). These mechanisms do not consider which instances should be repaired resulting in unnecessary repairs and overfitting which lower system performance. We propose enhanced repair mechanisms designed to decide which instances should be repaired. These repair mechanisms are based on a meta-reasoning framework called Boundary of Use (BoU) which identifies regions where the original learned function is doing well and regions where it is struggling. BoU-enhanced repair uses a divisive semi-supervised clustering (SSC) approach to create the regions, apply repair selectively, and retrain the function on the regions. We have shown that BoU-enhanced repair mechanisms improve system performance on datasets, by either deciding not to repair or identifying a selective set of regions to repair. Further, we have shown that it gives improved performance on datasets with extremes of the unnecessary repair (UR) and overfitting (OF). Overall, BoU-enhanced repair achieves higher system performance than all three types of repair mechanisms and has a positive impact on two (i.e., ANN and SVM) of three types of SL systems.

This paper only scratches the surface for using BoU meta-reasoning to improve repair mechanisms. First, there are still several holdouts in the results we intend to address, namely, adapting the amount of post-pruning to the size of regions for BoU-DT, and exploring ways to improve BoU-AL on datasets with label dispersions. Second, we intend to evaluate other repair mechanisms. We are most interested in data imputation used to fill in missing values in the features. Selective repair should allow for more precision on imputing different types of missing values. Third, we intend to investigate BoU-enhanced repair on more advanced approaches such as AdaBoost. We would like to see whether BoU-enhanced repair can leverage AdaBoost's ability to combine multiple weak functions and whether region-based repair can be used to address overfitting in AdaBoost. Finally, we would like to investigate using multiple repair mechanisms on the same set of regions. Selective repair would choose which repair mechanism would most improve the region. This would allow BoU-enhanced repair to be more effective on real-world datasets with multiple problems.

References

Caruana, R., Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. *ICML*, 161-168, 2006.

Christianini, N., Shawe-Taylor-J. *Support Vector Machines*. Cambridge University Press, 2000.

Daza, L., Acuna, E. An algorithm for detecting noise on supervised classification. *WCECS*, 701-706, 2007.

Donmez, P., Carbonell, J., Paired-sampling in density-sensitive active learning. *International Symposium on Artificial Intelligence and Mathematics*, 2008.

Freund, Y., Schapire, R. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Sys. Sciences*, 119-139, 1997.

Hall, M., et al. The WEKA data mining software: An update. *SIGKDD Explorations*, 10-18, 2009.

Kotsiantis, S. Supervised machine learning: A review of classification techniques. *Informatica*, 249-268, 2007.

Li, Y., et al. Classification in the presence of class noise using a probabilistic kernel Fisher method. *Pattern Recognition*, 3349-3357, 2007.

Liu, H., Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering*, 491-502, 2005.

Mesterharm, C., Pazzani, Active learning using on-line algorithms. *KDD*, 2011.

Miller, L. Genetic algorithm classifier system framework for semi-supervised classification. MS Thesis, CSE, *University of Nebraska—Lincoln*, 2007.

Mitchell, T. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

Pechenizkiy, M., et al. Class noise and supervised learning in medical domains: The effect of feature extraction, *IEEE CBMS*, 708-113, 2006.

Sayes, Y., Inza, I., Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 2507-2517.

Sayes, Y., Abeel, T., Van de Peer, Y. Robust feature selection using ensemble feature selection techniques. *ECML PKDD*, 313-325, 2008.

Settles, B. Active learning literature review. Technical report, Computer Science Department, *University of Wisconsin-Madison*, 2010.

Tuv, E., Borisov, A., Runger, G. Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, 1341-1366, 2009.