CLUSTER-BASED BOUNDARY OF USE FOR SELECTIVE IMPROVEMENT TO SUPERVISED LEARNING

by

Lee Dee Miller

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Leen-Kiat Soh

Lincoln, Nebraska

November, 2014

CLUSTER-BASED BOUNDARY OF USE FOR SELECTIVE IMPROVEMENT TO SUPERVISED LEARNING

Lee Dee Miller, Ph.D.

University of Nebraska, 2014

Advisor: Leen-Kiat Soh

Supervised learning (SL) is an active research area used for data mining in diverse fields such as education and bioinformatics. Algorithms such as feature selection, noise correction, active learning, and boosting are designed to improve SL predictive accuracy. Although often effective, these improvement algorithms still have problems. One common problem is using an improvement algorithm indiscriminately on all the training data can actually reduce predictive accuracy. Another problem is these algorithms require additional running time.

The solution explored in this work is to use improvement algorithms more selectively on the training data. When used selectively, an improvement algorithm is used only on the areas of training data that would most benefit from improvement and not on areas where improvement is unnecessary (or even detrimental).

We propose a new cluster-based selective improvement algorithm called the Boundary of Use (BoU). The BoU starts by using a clustering algorithm to partition the training data into clusters. Then, the BoU decides whether each cluster corresponds to an area where improvement is unnecessary (inside the boundary) or an area that would benefit from improvement (outside). This decision is based on whether the SL system is

doing well or is struggling on member data. Finally, the BoU uses the improvement algorithm selectively on each cluster outside the boundary.

We comprehensively investigate our BoU notion by adapting it to three different application areas resulting in three new selective improvement algorithms: (1) cluster-based boosting (CBB) for boosting, (2) BoU-AL for active learning, and (3) BoU-DP for feature selection and noise correction in data preprocessing. Extensive empirical results on benchmark and real-world datasets demonstrate the effectiveness of these new selective improvement algorithms compared to the same improvement algorithm on all the training data and to existing, selective improvement algorithms.

Ultimately, our BoU work results in five research contributions: (1) an insightful categorization of improvement algorithms based on common problems, (2) a novel, flexible selective improvement algorithm adapted into a suite of new selective improvement algorithms on three different application areas, (3) a novel use of clustering as a means for independently evaluating SL, (4) new work establishing the effectiveness of conventional clustering for selective improvement, and (5) SL applied to educational data mining and survey informatics.

ACKNOWLEDGEMENTS

This research has been supported by a U.S. Department of Education Graduate Assistance in Areas of National Need (GAANN) Fellowship (P200A040150) and three grants from the National Science Foundation (IIS-0632642, DUE-1122956, SBES-1228937). This research has also benefited greatly from the data collected by another National Science Foundation grant (SES-1132015). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Additionally, this research has been partially completed utilizing the Holland Computing Center of the University of Nebraska.

Table of Contents

Chapter 1 Introduction

Supervised learning (SL) is an extremely active research area originally from machine learning, but now also used and further developed in diverse fields including education (Romero et al., 2008) and bioinformatics (Larranaga et al., 2006). SL has also been extensively used for data mining in general (Witten et al., 2011). Lastly, SL has recently been adopted by other fields such as survey informatics (Belli et al., 2013). Because of this diverse usage and development, SL terminology has become fragmented and jargonistic. Therefore, to improve clarity and avoid being encumbered by terminology, a high-level description for SL is provided. To show how SL fits into our work, we provide two running examples for the rest of this chapter: (1) SL for education and (2) SL for survey informatics.

Overall, SL systems are designed to learn a function (model) on ***existing*** data supplied by the user that can ***accurately*** predict (generalize) results on ***new*** data that the function has never seen before. First, existing data corresponds to the data already collected by or available to the user. This existing data contains instances (records) with two properties:

- Each instance has a common set of features (independent variables in statistics). Instances supplied can have the same or different combinations of feature values. In practice, instances often correspond to the same set of tests or measurements conducted on different objects or at different times.
- Each instance has a label (dependent variable) measuring the results that the user is interested in. Labels depend on the feature values, but the exact relationship is *highly complex and unknown to the user*. In practice, labels are often provided by

an external source such as a human expert[1]. These labels can have numeric or nominal values.

Second, new data corresponds to the data that will be collected or data already collected that lacks labels. This new data contains instances with the *same set of features* as the existing data. However, the labels for the new data are **unavailable**[2] and need to be predicted by the SL function. Lastly, function accuracy is measured by comparing the predicted labels with the actual labels provided by the external source. For nominal (unordered) values, the predicted label is correct only when it matches the actual label. For other values, the predicted label is more accurate the closer it is to the actual label.

Of course, we cannot really measure function accuracy on new data since the labels are unavailable. In practice, we evaluate SL systems by breaking the existing data into a ***training set*** used to learn the function and a ***test set*** used to predict the labels. We then compute the function accuracy by comparing the predicted label to the actual label for all the instances in the ***test set***. Since the instances in the test set are not used when the function is learned, they serve as a reasonable approximation for how the function would do on new data. This issue is further discussed in our experimental setup in Chapters 4-8.

In practice, SL systems use different approaches resulting in extremely diverse functions learned on the same existing (training) data (Kotsiantis, 2007; Witten et al., 2011). Examples of these different approaches include (1) using weights on a complex network of interconnected nodes to learn the relationship between feature values and the label (artificial neural networks), (2) using kernel functions to remap the existing data into

[1] The user and human expert may be the same individual. Because the exact relationship is unknown, the labels still come from an external source (e.g., post-hoc statistical analysis).

[2] Due to requirements necessary to obtain the labels (e.g., time required for post-hoc analysis), labels may be unavailable even when the user and human expert is the same individual.

a higher dimensional feature space where instances with different labels are widely separated (support vector machines), and (3) using recursive feature splits to find groups of instances where the instances in a group have the same label (decision trees). Despite these different approaches, SL systems have the following two capabilities in common:

- First, *SL systems learn a function automatically without the need for direct human intervention*[3]. This automation allows a function to be learned quickly on large amounts of existing data (since the user is not required to go through thousands of instances). This also allows SL systems to be used as part of an automatic decision-making process based on the predicted label without the need to wait on the user.
- Second, *SL systems learn a function by discovering relationships or patterns between the features and labels in the existing data*. An example pattern might be that a certain combination of feature values is consistently associated with a particular label. These patterns allow a function to generalize and predict the correct labels on new data. Such patterns also provide the user with novel observations on existing data, which would otherwise be buried underneath a mass of data.

These capabilities have made SL systems very popular and they have been employed in a wide variety of applications (Domingos, 2012) including: (1) disease diagnosis (Pechenizkiy et al., 2006; Delen, 2009; Orru et al., 2012), (2) spam filtering (Wei et al., 2008; Cormack et al., 2011; Caruana & Li, 2012), (3) fraud detection (Hilas & Mastorocostas, 2008; Phua et al., 2005; Travaille, et al., 2011), (4) recommendation systems (McLaren et al., 2010; Deng et al., 2011; Lin et al., 2011), (5) and intelligent

[3] As discussed later, active learning can use human intervention to improve SL.

tutoring systems and learning objects (Roy et al., 2008; Romero et al., 2008; Bernardini & Conati, 2010; Miller & Soh, 2013).

Two example applications of SL systems from previous work are given below. As previously mentioned, these two applications are used as running examples for the rest of this chapter so they will show up again in subsequent sections without being referred to directly in the text.

> **Example 1:** SL for education (Miller & Soh, 2013): *Decision tree used to learn function on student interactions with learning objects* (time spent, number of clicks, etc.). Function predicted whether students would pass or fail test at the end of the learning object. This prediction could be used as intervention to alert students to spend more time on the learning object. Function found relationships interesting to researchers, such as students who spent less time on the test were more likely to pass and female students were more likely to fail.

> **Example 2:** SL for survey informatics (Belli et al., 2013): *Decision tree used to learn function on interview and respondent dialogue sequences involving questions and responses about previous employment history*. Function predicted the occurrence of a desirable respondent behavior. This prediction could guide interviewer on what types of questions to ask. Function found relationships interesting to researchers on the presence and absence of certain behaviors and the sequencing that led to the desirable respondent behavior.

1.1. Supervised Learning Limitations

There has been considerable research showing that different types of SL systems can achieve high predictive accuracy on existing datasets[4]. Indeed, in the diverse work cited above, a predictive accuracy of 80-90% for the SL functions is relatively common on the datasets considered. However, it should be noted that the predictive accuracy for SL systems *varies considerably between datasets* (Demsar, 2006; Caruana & Niculescu-Mizil,

[4] Remember, predictive accuracy on existing datasets is measured using a test set whose labels are unavailable to the function.

2006; Raeder et al., 2010). Namely, the same SL system that learns a function with *perfect predictive accuracy* on one dataset often learns a function whose accuracy is not much better than *random chance* on another dataset—that is, a SL algorithm that works well for one dataset often does not work well for another dataset. Furthermore, there is no clear winner between different types of SL systems. Namely, an artificial neural network may achieve significantly higher accuracy compared to a decision tree on one dataset, while achieving lower accuracy than a decision tree on another dataset.

The discrepancy of results leads to a lack of a one-size-fits-all solution for supervised learning. This lack is extremely frustrating for both conventional users and researchers. Conventional users who employ SL systems as black box solutions, without understanding the existing data, often fail to achieve the desired level of predictive accuracy (Wagstaff, 2012). Researchers, who understand the existing data, are still required to spend copious amounts of time on necessary preparations (Wagstaff, 2012) such as tweaking the existing data (in terms of the features used) to obtain satisfactory results, or even exploring, evaluating, and switching SL systems. In fact, most of the time on a machine learning project is feature engineering the existing data to obtain these satisfactory results (Domingos, 2012).

Researchers have spent considerable time and effort trying to explain the discrepancy in the results for SL systems. After considerable analysis that spans multiple decades, researchers have found several properties that often *reduce the predictive accuracy for SL systems on new data* (Domingos, 2012). In this work, we consider four of these properties that are probably the most commonplace—affecting the largest number of datasets:

- Presence of irrelevant features in the existing data where the features provided are unrelated or tenuously related to the label.
- Presence of noise in the existing data where the values provided are wrong or missing. This is particularly problematic when the labels provided are wrong.
- Lack of sufficient existing data to learn the relationship between the features and labels for all the new data; particularly problematic when the data is highly sparse with relatively few instances compared to the number of features.
- Lack of functions that are complex enough to learn all the relationships and patterns in the existing data; particularly problematic when the data is collected from different sources.

Example 1: SL for education (Miller & Soh, 2013): Student interactions with learning object are *not equally relevant* to whether students pass or fail the test—student responses to surveys before the test were much less relevant than student interactions recorded during the actual test. Student interactions *are highly noisy*—students try and "game the system" to get a good grade without understanding the content. Student interactions are *extremely complex*—students come from diverse backgrounds and there are many ways of going through the learning object.

Example 2: SL for survey informatics (Belli et al., 2013): Interviewer and respondent behaviors may *not be equally relevant*—interviewer laughing behavior is less relevant than interviewer asking question. Behaviors may be *noisy*—behaviors code assigned subjectively. Behavior data is *extremely sparse*—absence of certain behaviors is much more likely than presence of same. Behaviors are *extremely complex*—behaviors based on dialogue sequence between interviewer and respondent where presence of one behavior is conditional on another.

1.2. Improvement Algorithms for Supervised Learning

In the quest for a one-size-fits-all solution, researchers have spent considerable effort developing algorithms to separately address each of the four properties that reduce SL system accuracy on new data. In his seminal work, Domingos (2012) refers to

algorithms used to address the irrelevant features and noise properties as *feature engineering*. We rebrand these algorithms as **improvement algorithms** to include algorithms that address each of the four properties. These algorithms are summarized here and discussed in more detail in Chapter 3.

- **Feature selection** algorithms (Hastie et al., 2008) are designed to detect and remove irrelevant features from the dataset.
- **Noise correction** algorithms (Segata et al., 2010) are designed to identify noisy labels and then remove or replace them.
- **Active learning** algorithms (Settles, 2010) are designed to identify the unlabeled instances in the existing data that best help the SL function learn the relationship between the features and labels. Active learning operates on the assumption that label cost is the limiting factor on the size of the existing data. This is realistic since the features are often collected using the same set of measurements, while the label requires an expensive human expert.
- **Boosting** algorithms (Schapire & Freund, 2012) are designed to combine multiple functions each learned on the existing data (using the same SL system) into a more complex function that can better predict the correct labels. Subsequent functions focus on existing data where the previous function predicted the wrong label. This allows subsequent functions to identify relationships and patterns missed by previous functions.

Each of the above improvement algorithms addresses a single property described previously. The properties addressed by the first two algorithms are obvious: feature

selection addresses the presence of irrelevant features in the existing data, while noise correction addresses the presence of noise in the existing data.

Next, active learning addresses the lack of sufficient existing data using other resources available to the SL system. Active learning leverages *unlabeled* instances that would otherwise be unusable for SL since SL learns the relationship between the features and labels. The idea for active learning is to use other resources, such as a human expert, to provide the labels for those instances. Since resources are limited, active learning queries labels for instances that provides the highest predictive accuracy from the beginning, or further improve predictive accuracy for the SL system.

Finally, boosting algorithms address the lack of complex enough functions property. Boosting does this by combining multiple functions from the same SL system into a single, stronger function. The idea is that each single function is designed to master a specific subset of the existing data—simplifying what needs to be learned. Boosting then combines these functions such that the final decision (on the label) benefits from this individual expertise.

> **Example 1:** SL for education (Miller & Soh, 2013): Use *feature selection* and noise correction to remove student interactions that are irrelevant and noisy. Use *active learning* to decide which learning objects should be redeployed to collect additional test results in order to better predict student learning. Use *boosting* to learn functions on subsets of the student interaction data.

> **Example 2:** SL for survey informatics (Belli et al., 2013): Use *feature selection* and *noise correction* to remove behaviors that are irrelevant and noisy. Use *active learning* to determine which portions of the transcripts should be coded to provide additional existing data. Use *boosting* to learn functions on subsets of the dialogue data.

1.3. Improvement Algorithm Problems and Solution

Previous research has shown that using these improvement algorithms allows SL systems to often achieve higher predictive accuracy on the same dataset than the same SL systems (without improvement). Unfortunately, there are still problems with using these improvement algorithms.

First, when these improvement algorithms are used indiscriminately on all the existing data, they can potentially misuse or even damage the existing data (e.g., using the same method to improve widely separated instances or removing features entirely that are irrelevant to the majority, but not all, instances). This can result in the SL function **overfitting**—a well-known problem referred to as the "bugbear" (Domingos, 2012) of machine learning because it appears repeatedly and is difficult to solve. For improvement algorithms, overfitting occurs when improving the accuracy for the SL system on the existing data actually results in *worse predictive accuracy on the new data*. This happens when the SL system learns distorted patterns that apply to the existing data, but do not apply to the new data. In essence, the SL system tries too hard to be perfect in learning the function to predict existing data such that the function "fits" or describes the existing data so well that it does not generalize to the new data. Previous research has shown that all four improvement algorithms can result in overfitting (Smialowski et al., 2010; Yin & Dong, 2011; Giyon et al., 2012; Saha et al., 2013).

Second, improvement algorithms require additional running time. These algorithms generally require the SL system to learn a function on the existing data multiple times (e.g., to serve as the basis for comparison). This could result in the scenario where a SL system learns the function in the time available, but the addition of an improvement

algorithm prevents the SL system from finishing in time. In any case, the additional running time can result in considerable overhead and even scalability issues when the amount of existing data is extremely large.

The solution explored in this work is to *use improvement algorithms more selectively on existing data*. The selective improvement solution requires two basic steps to identify target areas. First, this solution needs to identify areas in the existing data where the instances would benefit the most from the improvement algorithm—for example, areas where the SL function is struggling to predict the correct label. Second, this solution needs to identify areas in the existing data where the improvement algorithm is *unnecessary*, such as areas where the SL system is already predicting the correct label. Once these target areas are identified, the improvement algorithm is applied selectively only on the areas that would benefit. Additionally, the solution should consider fine-tuning the improvement algorithm separately for each area: using the algorithm aggressively on some areas and conservatively on others. Using selective improvement, we can leverage improvement algorithms to address the properties that reduce SL accuracy while reducing the likelihood that the improvement algorithms will overfit and, in most cases, without increasing the complexity of the solution. In this work, we consider using only a single improvement algorithm at a time. We discuss selective improvement using multiple, different algorithms in Chapter 8.

Example 1: SL for education (Miller & Soh, 2013): *Feature selection* and *noise correction* mistakenly treat existing data from underrepresented students as anomalies and remove features and labels important to those instances. *Active learning* mistakenly query labels for sessions with unusual interactions rather than sessions more interesting to researchers that contain similar interactions, but different test results. *Boosting* mistakenly focuses on existing data where student interactions are not representative of learning the content as these students could have been trying to "game the system".

Example 2: SL for survey informatics (Belli et al., 2013): *Feature selection* and *noise correction* mistakenly treat existing data from dialogues with minority persons as anomalies removing features and labels important to those instances. *Active learning* mistakenly query labels for dialogues containing diverse behaviors while ignoring the less frequent label that is more interesting to researchers (i.e., presence of a response behavior). *Boosting* mistakenly focuses on existing data where dialogues are not representative of the desired behavior because they have been miscoded.

1.4. Cluster-Based Boundary of Use for Selective Improvement

We propose a cluster-based selective improvement algorithm called the Boundary of Use (BoU) summarized here and discussed in more detail in Chapter 3. Our BoU first uses a clustering algorithm to partition the existing data into target areas (i.e., function is struggling or improvement is unnecessary). BoU uses a clustering algorithm because, unlike SL systems, clustering algorithms operate independently of the labels. The purpose of the clustering algorithm is to identify compact areas each containing instances with similar features. Our BoU algorithm then evaluates the instances in each cluster using the labels and the SL system function. For each cluster, BoU decides whether to use the improvement algorithm and how aggressively to use it. Clusters where the SL system is already doing well are considered to be inside the BoU and protected from unnecessary improvement. Finally, BoU uses the improvement algorithm separately on the designated clusters.

Ultimately, the BoU improves the predictive accuracy for SL systems by incorporating selective improvement into the functions. As an example, the BoU could use feature selection selectively on each cluster and then learn a new function on each cluster with a different feature subset. To predict the label for a new instance, the BoU assigns that instance to the nearest cluster and then uses its function to predict the label. As will be shown, this selective improvement addresses both SL limitations on the dataset and problems from improvement algorithms.

> **Example 1:** SL for education (Miller & Soh, 2013): Use *selective feature selection* and *selective noise correction* separately on existing data from underrepresented students to learn relationships and patterns specifically for those students. Use *selective active learning* to query the labels for student sessions with similar interactions, such as, sessions on the same learning object or sessions from students with similar backgrounds. Use *selective boosting* that ignores existing data where students are trying to "game the system" to avoid learning subsequent functions on highly misleading data.

> **Example 2:** SL for survey informatics (Belli et al., 2013): Use *selective feature selection* and *selective noise correction* separately on existing data from dialogues with minority persons to learn relationships and patterns specifically for those persons. Use *selective active learning* to query the labels for dialogues containing similar behaviors to dialogues with the minority label to increase the number of instances with the minority label. Use *selective boosting* that ignores existing data where dialogues have been miscoded to avoid learning subsequent functions on highly misleading data.

The research done to investigate the viability of the proposed BoU are summarized in Figure 1.1. First, as shown in the 2nd tier, the BoU has been applied to three separate application areas. These application areas include all four of the improvement algorithms described above with feature selection and noise correction in the data preprocessing application. The BoU and the application areas are discussed further in Chapter 3. Second, our papers for each application areas are shown in the 3rd tier and discussed further in

Chapters 4-8. The datasets used in the 4th tier are based on previous work and also discussed in Chapters 4-8.

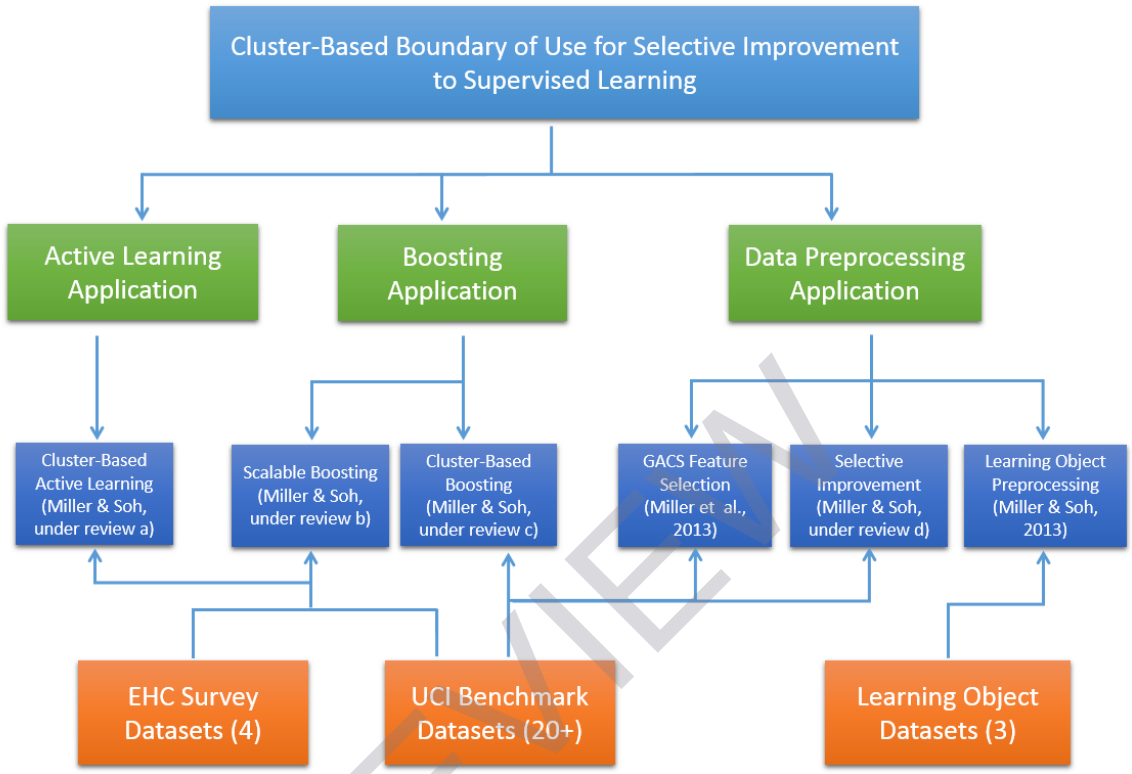

Figure 1.1. Summary of Research. The 2nd tier from the top contains the applications areas for our research discussed in Chapter 3. The 3rd tier contains our papers addressing those application areas discussed in Chapters 4-8. The 4th tier contains the datasets (with number provided in parentheses) used in our papers.

Our research into the viability of the proposed BoU selective improvement has led to four high-level research contributions to machine learning.

- **Categorization of improvement algorithms.** We have expanded the feature engineering notion discussed in Domingos (2012) into the improvement algorithm categorization including four distinct types of algorithms: feature selection, noise correction, active learning, and boosting. These algorithms all share two common characteristics. First, improvement algorithms are designed to make improvements to the SL system by modifying either the existing data or the SL system directly. Second,

improvement algorithms work as an additional meta-reasoning layer "on top" of the SL systems deciding whether to use the SL system with or without improvement. These common characteristics help to explain why algorithms that fit this categorization are subject to the same sorts of problems (e.g., overfitting) as previously discussed. Furthermore, extending feature engineering to include all these algorithms, helps to expand the "toolbox" of algorithms available to users for improving results.

- **Research spanning improvement algorithms.** In the past, copious amounts of research has been done on feature selection, noise correction, active learning, and boosting. However, such research has overwhelmingly been done separately, focusing on a single algorithm type. There are a few a notable examples where research is done spanning two algorithm types, such as boosting and noise correction (Liu & Vemuri, 2011). However, to our knowledge, no previous research has been done spanning all four algorithm types as we do for the BoU. Such research is important for two reasons. First, doing research separately on the algorithm types can be a duplication of effort. These algorithm types use very different approaches, but have common problems as previously discussed. As shown in this work, a solution originally discovered for one algorithm type (feature selection) can be adapted to other types resulting in broad-spectrum improvement. Second, focusing on a single algorithm type limits the amount of improvement that can be done on a dataset. Real-world datasets continue to grow in both complexity and size. These datasets are likely to require complex combinations involving multiple improvement algorithms working together to achieve SL functions with high accuracy.

- **Application of unsupervised learning to supervised learning.** Traditionally, unsupervised learning (clustering) and supervised learning have been separate research areas. The general consensus was that, although clustering is, in itself, an extremely important area in data mining, the lack of any consideration paid by clustering algorithms to the label made it unsuitable for the supervised learning task (finding the relationship between the features and the label). Even when clusters were used for supervised learning, they focused exclusively on finding homogenous clusters each containing members with a single label (Eick et al. 2004). We apply clustering to supervised learning in a completely different and, to our knowledge, novel way. The idea is to use clustering as an independent means of evaluating the existing data and the SL function. Clustering breaks up the existing data into areas containing similar instances without considering the labels. In turn, this allows the clusters to be independently evaluated using the labels and SL function. In other words, clustering is used for selective improvement to identify those areas where the function is struggling that need improvement and areas where improvement is unnecessary.
- **Conventional clustering.** We also show that even conventional (*k*-Means) clustering leads to good results for selective improvement. Further work using more advanced clustering (e.g., cluster ensembles), may provide even better results for selective improvement to supervised learning.
- **Supervised learning on new datasets.** Supervised learning has never been used before on the event history calendar and learning object datasets. These real-world datasets have interesting properties, making them useful for evaluating our BoU in different application areas. At the same time, using SL on these datasets contributes to

interdisciplinary work. In the course of our experiments, SL on these datasets has led to the discovery of new relationships and patterns on the existing data previously unknown to researchers. Furthermore, our selective improvement has allowed us to identify areas containing troublesome data undiscovered by previous methods that operate on all the existing data (e.g., post-hoc statistical analysis).

1.5. Overview

We conclude this chapter with an overview of the remaining chapters in this work. Chapter 2 describes the supervised learning systems used in our experiments. Chapter 3 goes over the design of the BoU cluster-based selective improvement algorithm. Chapters 4-8 provide the experimental setup and results for all the application areas: Chapters 4-5 provides results for boosting, Chapter 6 provides the results for active learning, and Chapters 7-8 provide results for data preprocessing. Chapter 9 explains the previous research that led directly to the initial discovery and refinement of the BoU. We conclude with Chapter 10 that discusses both general conclusions common to all the application areas and specific conclusions found in the results for only a single application area. This chapter also goes over possible future work using selective improvement algorithms.