AN ABSTRACT OF THE DISSERTATION OF

Shubhomoy Das for the degree of Doctor of Philosophy in Computer Science presented on May 18, 2017.

Title: Incorporating User Feedback into Machine Learning Systems

Abstract approved: ______________________________

Weng-Keen Wong

Although machine learning systems are often effective in real-world applications, there are situations in which they can be even better when provided with some degree of end user feedback. This is especially true when the machine learning system needs to customize itself to the end user's preferences, such as in a recommender system, an email classifier or an anomaly detector.

This thesis explores two directions in incorporating end user feedback to machine learning systems. First, I introduce an algorithm that incorporates feature feedback in a semi-supervised text classification setting. Feature feedback goes beyond instance-label feedback by allowing end users to indicate which feature-value combinations are predictive of the class label. In order to incorporate feature feedback in a semi-supervised setting, I develop a Locally Weighted Logistic Regression algorithm that uses a similarity metric combining information from the user's feature feedback and information based on label diffusion on the unlabeled data.

Second, I explore the use of instance-level feedback to anomaly detection algorithms. Anomaly detectors commonly return a list of the top outliers in the data. Although these outliers are statistically unusual, some are uninteresting to a user as the internal statistical model may not necessarily be aligned with the user's semantic notion of an anomaly. I present an algorithm that can increase the number of true anomalies presented to the user if a limited amount of instances are labeled as anomalous or nominal.

Incorporating User Feedback into Machine Learning Systems

by

Shubhomoy Das

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 18, 2017
Commencement June 2017

ACKNOWLEDGEMENTS

I am greatly indebted to my advisor Dr. Weng-Keen Wong who provided me the opportunity to pursue doctoral studies. Under his guidance, I learned how to follow research literature and quickly identify the core ideas. He also allowed me the flexibility to pursue any ideas or approaches of my own interest and that has really broadened my knowledge.

Getting a Statistics masters degree under Dr. Alix Gitelman's guidance has been one of my proudest achievements. I cannot thank her enough for the encouragement and opportunity.

Dr. Tom Dietterich and Dr. Alan Fern are some of the best collaborators anyone could asked for. Apart from enlisting me on their projects, they have been both sources of ideas as well as constructive criticisms.

Dr. Prasad Tadepalli was my very first advisor at Oregon State University. Without his initial support and guidance a research position to pursue doctoral studies would have been out of my reach. I am also thankful that he accepted to serve on my thesis committee.

Dr. Margaret Burnett was one of my earliest collaborators with whom I worked for the first part of my thesis. While working with her, I got an unique opportunity to participate in designing and conducting a user study. It set the tone for my work because I realized how important it is for end users to be able to control machine learning systems.

I am grateful that Dr. John Dilles volunteered to serve as the Graduate Council Representative on my committee. Apart from handling the administrative tasks, he has always been open to accommodate the schedule of my exams despite his own busy schedule.

While pursuing my studies I had the pleasure of working with smart and helpful fellow graduate students and staff members Janardhan Rao Doppa, Jun Yu, Shahed Sorower, Amran Siddiqui, Satpreet Singh, Jed Irvine, and many others with whom I was able to brain-storm ideas and collaborate on projects. Along the way I have also been very fortunate to have taken courses with some amazing professors from both Computer Science and Statistics departments. I am sure very few people get to have such a privilege.

Lastly, I owe a huge debt of gratitude to my parents and my sister who have patiently supported and encouraged me to pursue higher studies.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Motivation 1

1.2 Setup 2

1.2.1 On the nature of feedback 3

1.3 Aspects of Feedback for Machine Learning Algorithms 4

1.3.1 Feedback paradigms 4

1.3.2 Types of feedback 5

1.3.3 Feedback incorporation mechanisms 6

1.4 Thesis overview 7

1.4.1 Feature labeling for semi-supervised learning 7

1.4.2 Instance-level feedback for improving an anomaly detector 8

2 Feature Labeling 9

2.1 Introduction 9

2.2 Related Work 9

2.3 Feature Labeling 12

2.3.1 LWLR 12

2.3.2 LWLR-FL 14

2.3.3 Semi-supervised Learning 16

2.3.4 LWLR-SS-FL 19

2.3.5 Experiment Results 22

2.3.6 Oracle Study 22

2.3.7 End-user Labeling Study 26

2.3.8 Sensitivity Analysis 29

2.4 Discussion 31

2.5 Conclusion 32

3 Feedback to Anomaly Detection algorithms 34

3.1 Introduction 34

3.2 Related Work 35

3.3 Methodology 38

3.3.1 Anomaly Detector 39

3.3.2 The top τ-quantile 39

3.3.3 Accuracy at the top 40

TABLE OF CONTENTS (Continued)

Page

3.3.4 Objective Function 41
3.3.5 Active Anomaly Discovery (AAD) 44

3.4 Results 45
3.4.1 Experimental Setup 45
3.4.2 Query Visualization 50
3.4.3 Contributions of Sub-parts of the Optimization Problem 51
3.4.4 Computational Complexity and Scaling Up 53
3.4.5 Comparison with Rare Category Detection 57
3.4.6 Sensitivity Analysis 59

3.5 Discussion 61

3.6 Conclusion 62

4 Extending Active Anomaly Discovery to other algorithms 64

4.1 Introduction 64

4.2 AAD for Feature Bagging 64

4.3 Applying AAD to a single Isolation Forest 65

4.4 Discussion 69

4.5 Conclusion 72

5 Conclusion and Future Work 73

5.1 Conclusion 73

5.2 Future Work 74
5.2.1 Improvements to Feature Labeling 74
5.2.2 Extensions to Active Anomaly Discovery 75

Bibliography 76

Appendices 85

A Appendix 86

LIST OF FIGURES

Figure Page

1.1 Components of a Machine Learning System with User Feedback (based on a modified version of Figure 1.2 in [4]). The task for the machine learning algorithm $\mathcal{A}$ is to infer a function f which should be as close to the true target function f^* as possible. In addition to not knowing the true target function, the machine learning algorithm is also unaware of the true data distribution $\mathcal{X}$. For its task, $\mathcal{A}$ is provided with a set of training data and the assumption that $f \in \mathcal{H}$, where $\mathcal{H}$ is either a parametric or non-parametric mathematical model. Additionally, if the system supports feedback, it has access to an expert whom it might query. 2

2.1 Example of a situation where LR fails but LWLR succeeds in separating two classes 13

2.2 Exposing structure in data through unlabeled data 18

2.3 Label Diffusion. The circles represent the initial set of labeled training examples. The colors red, blue, and green represent three different classes. Points in grey are unlabeled instances. At each iteration, the labels 'diffuse' from the currently labeled instances to their nearest neighbors. After a number of iterations of label diffusion, the distribution of the labels over the instances becomes stationary. 20

2.4 Clustering in manifold space 21

2.5 Average macro-F1 scores for incorporating 10 Oracle feature labels through semi-supervised feature labeling for the six datasets used in our experiments. 26

2.6 A comparison of the relative benefits of feature labeling and unlabeled data for variants of LWLR-FL. 26

2.7 Results of Semi-supervised Algorithms with User Feature Labels: (Left) incorporating 8 oracle feature labels per class, (Middle) incorporating end-user feature labels only for existing features, (Right) incorporating all end-user feature labels. 29

2.8 Sensitivity of LWLR-SS-FL to hyper-parameters. 30

3.1 Setup for Active Anomaly Discovery. 44

LIST OF FIGURES (Continued)

Figure | Page

3.2 The total number of true anomalies seen vs. the number of queries for all datasets. Total number of queries for the **smaller** datasets (*Abalone*, *Cardiotocography*, *ANN-Thyroid-1v3*, *Covtype-sub*, *KDD-Cup-99-sub*, *Mammography-sub*, *Shuttle-sub* and *Yeast*) is 60. Total number of queries for the **larger** datasets (*Covtype*, *KDD-Cup-99*, *Mammography*, *Shuttle*) is 100. The results were averaged over 10 runs for each algorithm (except SSAD, which is deterministic). The error bars show the 95% confidence intervals. 48

3.3 Low-dimensional visualization of *Abalone* and *ANN-Thyroid-1v3* using t-SNE. Plus signs are anomalies and circles are nominals. A red coloring indicates that a true anomaly point was queried. A green indicates a nominal point was queried. Grey circles correspond to unqueried nominals. To make unqueried anomalies stand out visually, we indicate them with blue plus signs. 52

3.4 Comparison of AAD variants. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 53

3.5 The wall clock computation time at each iteration (averaged across 10 runs). 54

3.6 Comparison of AAD with variants where the constraints are relative to τ-th ranked instance. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 55

3.7 Comparison of AAD with Rare Category Detection Algorithms. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 59

3.8 τ-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying τ while keeping parameters C_A and C_ξ fixed at their default values 100, and 1000 respectively. 60

3.9 C_A-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying C_A while keeping parameters τ and C_ξ fixed at their default values 0.03, and 1000 respectively. 60

3.10 C_ξ-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying C_ξ while keeping parameters τ and C_A fixed at their default values 0.03, and 100 respectively. 61

LIST OF FIGURES (Continued)

Figure Page

4.1 Results when ensemble members were generated using feature bagging. The *y-axis* shows the gain from incorporating feedback with AAD over 60 iterations. The difference is expressed as percentage change from baseline. A gain of 100% would imply that AAD discovered twice the number of anomalies as the baseline. 66

4.2 Random trees in Isolation Forest (IF) for synthetic data. The points in red are true anomalies; points in gray are true nominals. Figure 4.2a shows the leaf node regions for a single tree generated by random IF splits. Figure 4.2b shows the leaf node regions for 10 trees generated by IF; each tree has a different color. Figure 4.2c and Figure 4.2d show the contours of anomaly scores assigned by Isolation Forest with one and ten trees respectively. Deeper red means more anomalous; deeper blue means more nominal. Red circles are true positives among top 20 instances ranked by IF without feedback; green circles are false positives among the top 20. The left sidebar in Figure 4.2c and Figure 4.2d show the ranking of true anomalies (red dots). Ideally, true anomalies should be near the top on this bar. 68

4.3 Incorporating feedback in Isolation Forest (IF) for synthetic data. Figures 4.3a – 4.3e show anomaly score contours in the same way as explained in Figure 4.2. The red curve in Figure 4.3f shows the number of true anomalies discovered when we incorporate feedback; the blue curve in Figure 4.3f shows the number of true anomalies discovered when no feedback was incorporated. 70

4.4 Incorporating feedback in Isolation Forest with AAD. Figures 4.4a–4.4d are the smaller datasets with 60 feedback iterations. Figures 4.4e–4.4h are the larger datasets with 100 feedback iterations. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 71

LIST OF TABLES

Table Page

2.1 Partially labeled document corpus for semi-supervised learning. 17

2.2 The classes of the data sets used in the oracle study, along with the number of instances in each class shown in parentheses. 22

2.3 Results of incorporating 10 oracle features per class through semi-supervised feature labeling for all six datasets. The values in **bold** in each column represent the highest scores for the corresponding dataset. The symbol † denotes values that are significantly greater than all other algorithms at the 0.05 level (Wilcoxon signed-rank test, $p < 0.05$). Results for the baseline SVM and LWLR algorithms are included for reference. 25

2.4 Results of adding 10 oracle features per class for all six datasets for algorithms that are variants of logistic regression. The symbol * denotes values that are significantly greater than the baseline LWLR algorithm at the 0.05 level (Wilcoxon signed-rank test, $p < 0.05$) 27

3.1 Datasets used in our experiments, along with their characteristics. 46

LIST OF ALGORITHMS

Algorithm | Page

1 Semi-supervised Learning with Label Diffusion 19

2 Active Anomaly Discovery (AAD) 45

3 Generating Ensembles by Feature Bagging 65

4 Generating randomized trees in Isolation Forest 67

LIST OF APPENDIX FIGURES

Figure | Page

A.1 Low-dimensional visualization of benchmark datasets using t-SNE. Plus signs are anomalies and circles are nominals. A red coloring indicates that the anomaly or nominal point was queried. Grey circles correspond to unqueried nominals. To make unqueried anomalies stand out visually, we indicate them with blue plus signs. 87

A.2 Comparison of AAD with variants where the constraints are relative to τ-th ranked instance. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 89

A.3 Comparison of AAD with Interleave. The results were averaged over 10 runs. The error bars show the 95% confidence intervals. 90

A.4 Comparison of AAD with MALICE. The results were averaged over 10 runs for AAD. The error bars show the 95% confidence intervals. 92

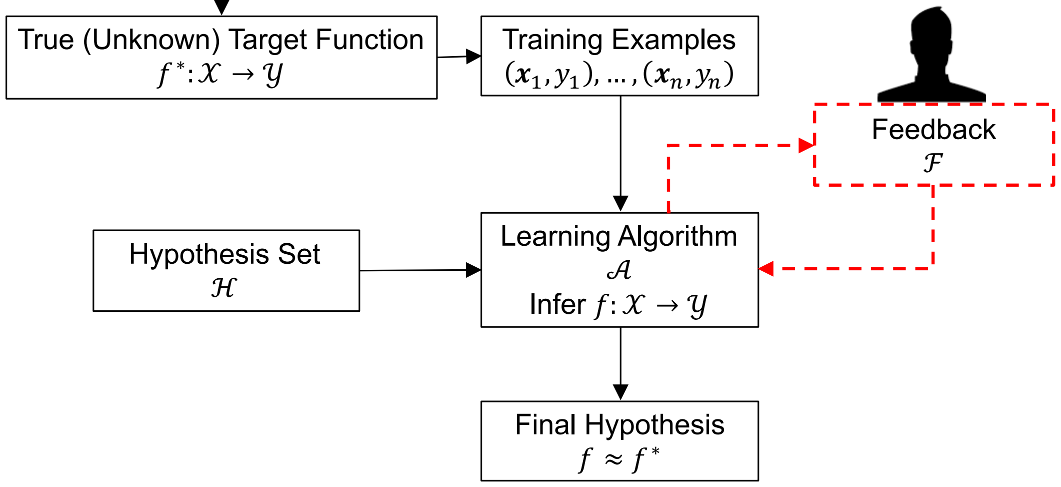
Chapter 1: Introduction

1.1 Motivation

Machine learning has made great strides in recent years, with impressive performance on many real-world problems such as image recognition and speech recognition. Although machine learning systems can be effective when deployed in the real world, they are not perfect. Situations arise when there is a mismatch between the output of the learning algorithm and the end user's expectations. Consider, for instance, an user trying to set up a classifier to sort emails into folders. The end user labels a handful of email messages with their respective folders and then trains the classifier on the labeled training data. The initial classifier often performs poorly due to a limited amount of training data from that particular end user. Over time, as more training data becomes available, the classifier's predictions become more accurate. However, the classifier may still be inaccurate at making fine-grained distinctions between two closely related email folders. In general, mistakes by machine learning algorithms can occur due to a variety of reasons, including insufficient training data, biased training data and inaccurate assumptions made by the underlying model.

One approach for improving machine learning systems is to ask the end user to provide corrective feedback in order to change the learning algorithm. This approach is particularly useful after deployment. In this setting, a machine learning expert, who we refer to as an *algorithm designer*, has completed designing and implementing the machine learning algorithm. The finished system is shipped to an *end user*, who trains the learning system on a training dataset and applies it to a test dataset at performance time. Note that after the learning system is deployed, the algorithm designer is no longer accessible and the end user, who is usually not a machine learning expert, is the main person interacting with the deployed system. This thesis explores new methods for incorporating end user feedback into learning algorithms to improve their performance after deployment.

In this chapter, I first present the algorithmic setup and a general overview of feedback



Data Distribution (Unknown) $\mathcal{X}$
True (Unknown) Target Function $f^*: \mathcal{X} \rightarrow \mathcal{Y}$
Training Examples $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)$
Feedback $\mathcal{F}$
Hypothesis Set $\mathcal{H}$
Learning Algorithm $\mathcal{A}$ Infer $f: \mathcal{X} \rightarrow \mathcal{Y}$
Final Hypothesis $f \approx f^*$

Figure 1.1: Components of a Machine Learning System with User Feedback (based on a modified version of Figure 1.2 in [4]). The task for the machine learning algorithm $\mathcal{A}$ is to infer a function f which should be as close to the true target function f^* as possible. In addition to not knowing the true target function, the machine learning algorithm is also unaware of the true data distribution $\mathcal{X}$. For its task, $\mathcal{A}$ is provided with a set of training data and the assumption that $f \in \mathcal{H}$, where $\mathcal{H}$ is either a parametric or non-parametric mathematical model. Additionally, if the system supports feedback, it has access to an expert whom it might query.

mechanisms in Section 1.2. Next, in Section 1.3, I discuss aspects that are important to the design of feedback and relevant to this thesis. Finally, in Section 1.4, I introduce two particular problems in machine learning where user feedback can be extremely useful after deployment.

1.2 Setup

Figure 1.1 shows the basic components of a machine learning algorithm. When the labels $y_1, \dots, y_n$, are known, the algorithm $\mathcal{A}$ is said to be *supervised*. When the labels are completely unknown, the algorithm is said to be *unsupervised.* When labels are known for only some of the $\mathbf{x}$'s, $\mathcal{A}$ is said to be *semi-supervised.*

The algorithms proposed in this thesis are semi-supervised. Here, the unlabeled data helps get a better estimate of the data distribution. Moreover, if there are systematic biases in the collection of unlabeled data (eg. sample selection bias), it would hurt the performance of the algorithm more rather than help. An expert's explicit feedback such as which features or instances the algorithm should focus on more, and what categories they are more predictive of, might help overcome some of these problems.

1.2.1 On the nature of feedback

Any feedback requested from end users demands their time and imposes a cognitive burden. These are factored into the *cost* of feedback. In domains such as medicine where subject matter experts themselves interact with the machine learning system, the feedback requested might be complex. In other domains such as those where users are primarily citizen scientists or mechanical turkers, feedback necessarily needs to be simple in order to minimize errors. Irrespective of the skill level expected of the user, his/her feedback should ideally have a positive impact on the algorithm performance. However, algorithms vary in how efficiently they utilize the feedback. Therefore, when comparing feedback-based algorithms, we need some metric that measures performance while normalizing for the cost. A commonly used metric is the improvement in performance within a *budget.* Budget usually corresponds to the maximum number of queries the algorithm is allowed to ask a user; e.g., for an algorithm that asks users to label instances or features, the budget would correspond to the maximum number of items (instances or features) *actually labeled* by the user.

Apart from the cost, we need to keep in mind certain other aspects of feedback in machine learning algorithms during design. These are discussed next in Section 1.3. When we consider each of these separately, it helps us break the system down to its logical components and survey the related literature in a systematic manner. While I provide pointers to related work for most of these aspects, the coverage is not meant to be exhaustive. This is intentional; research in this area is large. Nevertheless, I will cover literature that is directly relevant to this thesis in more detail: Section 2.2 presents research related to the first part of this thesis – feature labeling for text classification;

Section 3.2 presents related work for the second part of the thesis – instance feedback for anomaly detection.

1.3 Aspects of Feedback for Machine Learning Algorithms

The design of feedback mechanisms for machine learning systems needs to take into account aspects such as what data is helpful as feedback (Section 1.3.1), in what form it is to be collected (Section 1.3.2), and how the base algorithm needs to be modified in order to incorporate it (Section 1.3.3). These aspects are typically determined by the underlying machine learning algorithm, the domain of application, and the ease with which an end-user would be able to provide the required feedback. An overview of these aspects follows next.

1.3.1 Feedback paradigms

Depending on who initiates the collection of feedback, the user or the algorithm, feedback-based machine learning algorithms may be broadly categorized as follows:

- **Passive (user-solicited feedback)**: A naïve way to incorporate feedback into machine learning algorithms is for users to iteratively add labeled training instances, train the algorithm and examine its output – possibly increasing the amount of training data with each iteration – until the output is satisfactory. Obviously, the downside to this approach is that it requires more manual effort by the users. However, such an approach works well in practice when feedback is of a form that has a large influence over the algorithm's behavior and is easy for users to provide; for example, feature labeling [58, 77] for text classification. Generalized Expectation (GE) [21] and posterior constraints [26] are other ways that allow users to freely express their preferences through statistical expectations. Section 1.3.2 discusses some of the types of feedback relevant to the passive approach. The algorithm presented in Chapter 2 of this thesis, which allows users to label both instances and features, also follows this approach.

- **Active (algorithm-solicited feedback)**: In many machine learning tasks, we might already have a large number of unlabeled data or obtaining such data might be easy. The machine learning algorithm has the option to choose any unlabeled data and query an expert for the true label. This is the ideal setting for *Active Learning* [65, 23] where the goal of the algorithm is to maximize the generalization accuracy while minimizing the number of queries presented to the expert. The large amount of unlabeled data can be used to determine its distributional properties; this in turn suggests which parts of the feature space are most informative. The algorithm improves its efficiency by selecting unlabeled instances from these informative parts for query. As part of this thesis, we discuss an algorithm (AAD) in Chapter 3 that has a setting similar to the one discussed here. However, unlike Active Learning, AAD operates in a transductive setting; it does not try to generalize its learning to unseen test data.

In sequential prediction problems the algorithm's goal is to predict the best action for future observations which are themselves dependent on past predictions. Since the data are not i.i.d, the algorithm cannot make assumptions about the distribution of the training data. *Imitation Learning* [63, 3, 61] is a popular approach in this setting where an expert provides demonstrations of good behavior and then the algorithm tries to emulate the expert. One of the techniques for learning in this domain is *follow-the-leader* [61, 33, 38]. The anomaly detection algorithm (AAD) discussed in Chapter 3 might also be considered as a type of the follow-the-leader algorithm. However, AAD is different from the standard sequence prediction problem because, as already mentioned, it is in a transductive setting where we know the data instances.

1.3.2 Types of feedback

Algorithms differ on the basis of the types of inputs users provide as feedback. Section 2.2 and Section 3.2 discuss in more depth literature that is relevant to this thesis. Here we present only an overview of the commonly encountered feedback types.

- **Instance labels**: This is by far the most commonly encountered feedback in machine learning, especially for active learning. Here the algorithm asks the user to label the most informative instance. In classification settings, this form of feedback helps in removing uncertainty at the class boundaries, thereby creating clear separation between classes. Chapter 3 applies this form of feedback to a non-classification task in which the primary goal is to rerank instances on the basis of anomalousness.

- **Feature labels**: Here the algorithm asks the user to label individual features [58]. In some ways this is a more direct incorporation of domain knowledge than just labeling instances. With instance labeling, the algorithm has to infer correlations between the features and labels, whereas with feature labeling the user directly provides the relationship between a feature-value pair and the class label. We will encounter this type of feedback in Chapter 2.

- **Rule labels**: The expert might try to transfer domain knowledge directly to the algorithm through rules. This approach has been investigated by Rashidi and Cook [59] in an active learning setting. In Rashidi and Cook [59], the algorithm exploits the underlying density distribution to find an informative instance as well as its most similar cases. Then by using a rule induction classifier to infer rules for separating those similar cases from the rest of data, they construct a generic query. This query has the form: $rule \Rightarrow label$. If the user marks the rule with a label, then that label is applied to all unlabeled instances which match the rule. Rule-based feedback is not a part of this thesis.

- **Expected statistics on labeled data**: Some preferences about model expectations may be incorporated into parameter estimation objective functions. GE [49] is one such framework which has been applied in an active learning setting for incorporating feature labels [21]. GE is fairly general and not limited to feature labeling. [49] discusses how a human might *communicate* with the underlying model using GE in terms of *expectations* rather than *parameter values*. Chapter 2 compares GE with other algorithms in a text classification task where both instances and features may be labeled.

1.3.3 Feedback incorporation mechanisms

Once the user has provided feedback or expressed some preference, the algorithm might incorporate it into the model in one of the following ways:

- **Add labeled instances to training data**: This is the most common mechanism for incorporating feedback. The main reason is its simplicity: no other modification is required to the base algorithm apart from retraining after adding the new labeled data.

- **Update feature weights**: This changes the influence each individual feature has on the underlying model. *LWLR-FL* and *LWLR-SS-FL* discussed in Chapter 2 use this method to incorporate feedback.

- **Use as regularizer or prior**: Various forms of regularization can be employed to incorporate user preferences in GE [49, 21] and posterior constraints [26]. Settles [66] incorporates feature labels by changing priors on labeled features.

- **Generate additional training examples that are consistent with feedback**: Section 1.3.2 discusses an algorithm ([59]) where instances matching a rule predicate are selected and labeled as desired by the user. Although this is similar to simply adding labeled instances, the difference is that now multiple instances might be labeled by a single piece of feedback. Moreover, the algorithm might associate weights to these labeled instances based on its confidence since not all instances affected by the feedback are inspected by the user.

1.4 Thesis overview

The two particular types of feedback mechanisms investigated in this thesis are: 1) feature labeling for semi-supervised learning and 2) instance-level feedback for improving an anomaly detector.

1.4.1 Feature labeling for semi-supervised learning

In Chapter 2, I introduce feature labeling as a method of incorporating user feedback. Feature labeling [5, 21, 70, 57] is a method that allows a user to point out which feature-value combinations are predictive of a class label. Raghavan et al. [58, 57] show that users take one fifth of the time to label a feature than they do to label an entire document. Wong et al. [77] propose a supervised feature labeling approach based on adjusting the distance metric of a Locally Weighted Logistic Regression (LWLR). I extend this work with a semi-supervised approach [15] that combines the information from labeled instances in the training set, the feature labels provided by end users and information from the inherent structure found in a pool of unlabeled data. I evaluate this semi-supervised approach using both simulated feature labels as well as feature labels provided by actual end users in a user study.

1.4.2 Instance-level feedback for improving an anomaly detector

Anomaly detection involves detecting unusual data which do not fit the descriptions known to the user. Some of these unusual data might indicate the presence of important processes that the user is unaware of. For instance, an anomalous user activity might point to the malicious intent of exfiltrating sensitive data from a computer network. Anomaly detectors are good at detecting statistical outliers, which include noise, uninteresting outliers, as well as interesting patterns. We refer to the patterns that are of interest to the user, as *anomalies*. Anomalies are not brought to the user's attention in a timely manner if there is a gap between what the detector considers as an outlier, and what the user considers as an anomaly. This could incur a loss for the user. The gap in understanding between the user and the detector can be bridged through instance-level feedback. In Chapter 3, I develop an algorithm called Active Anomaly Discovery [16], for improving the accuracy of an anomaly detector by incorporating instance-level feedback from an end user. A strength of this algorithm is that it can use even one-sided feedback, e.g., when all labeled instances are nominal.

Active Anomaly Discovery (AAD) operates in an interactive data exploration loop to provide instance-level feedback to an anomaly detector. Here we have an iterative feedback cycle in which a user answers queries posed by a learning algorithm. In each iteration the algorithm first determines the most anomalous unlabeled instance, and presents it to the user to label. Once the user responds, the algorithm updates the model and repeats the cycle. The cycle is continued until a budget on the number of queries is exhausted. For this method to be practical, a significant improvement in performance must be achieved with a reasonably small budget of queries, and the feedback should be incorporated in near real-time. As part of my thesis, I also investigate scaling up the proposed anomaly detection algorithm to handle a large volume of data and feedback.

Chapter 2: Feature Labeling

2.1 Introduction

Customizing to the end-user's preferences is challenging, especially when there is limited training data, such as when the application is first deployed. Raghavan et al. [58], and Raghavan and Allan [57] found that labeling a feature with a particular class took humans roughly a fifth of the time as labeling a document, and the benefits of feature labeling were greatest when the training set sizes were small. Motivated by this research, Wong et al. [77] developed an algorithm that incorporates feature labels provided by end-users into Locally Weighted Logistic Regression (*LWLR*). In a user study [77], this algorithm (*LWLR-FL*) outperformed other state-of-the-art feature-labeling algorithms when the training set size was small. As part of this thesis, I present a semi-supervised extension to *LWLR-FL*[1].

2.2 Related Work

In this section I review related work on feature labeling. We divide the approaches for feature labeling into supervised and semi-supervised feature labeling algorithms. Supervised feature labeling algorithms require only a training set of labeled instances. On the other hand, semi-supervised feature labeling requires both a labeled training set as well as a pool of unlabeled data, which is assumed to be relatively easy to obtain.

Two of the SVM-based methods presented by Raghavan and Allan [57] involved supervised feature labeling. Their first method, *Method 1*, scaled features indicated as relevant by the user by a constant a and the rest of the features by a constant d (where $a \geq d$). In their second method, *Method 2*, the user indicated that the j-th feature was relevant for a class label l. For each feature-label pair, Method 2 created a pseudo-document consisting of a value r in index j, zeroes elsewhere and a class label of l. The r parameter controlled the influence of the support vectors of the pseudo-documents on

[1]This work was published in Das et al. [15]

the separating hyperplane.

Another group of supervised feature labeling algorithms are based on multinomial naïve Bayes. The pooling multinomials approach [50] combined parameters from a multinomial naïve Bayes classifier trained on labeled instances and another derived from background knowledge, which in this case were feature labels. This approach, however, was restricted to Boolean class labels. Settles [66] proposed another method based on naïve Bayes in which he changed the priors for labeled features. If a feature was labeled with a class, the corresponding parameter was given a Dirichlet prior of $(1 + \alpha)$, where α was a tunable parameter, while all unlabeled features were given a uniform Dirichlet prior of 1.

The majority of the work in feature labeling has takes a semi-supervised approach. A common strategy employed by several methods is to use the user feedback to label the unlabeled data and then incorporate these soft labels into training. Method 3 of Raghavan and Allan [57], following the approach of Wu and Srihari [80], associated slack variables with the soft labeled data to influence the position of the margin for an SVM. The user co-training algorithm of Stumpf et al. [72] treated the user's feature labels like a classifier and combined this 'classifier' in the co-training framework [8] with naïve Bayes. The Multinomial naïve Bayes (MNB) approach by Settles could be extended to a semi-supervised approach by soft-labeling the unlabeled data and then re-estimating the MNB parameters [24]. Finally, the approach by Liu et al. [43] modified the EM algorithm to incorporate labels produced by representative words for each class selected by the user.

Another common approach to semi-supervised feature labeling is to treat the feature labels as constraints and bias the learned model to respect these constraints. Algorithms falling into this framework used an objective function during learning that consisted of the maximum likelihood of the training data plus an additional term that penalized the model when it failed to satisfy certain constraints. This framework was developed to address a more general class of problems than just feature labeling, and this framework has been used for problems as diverse as multi-view learning [24] and transfer learning [49]. Examples of these approaches include Constraint Driven Learning [11], Generalized Expectation [49, 21], Learning with Measurements [42] and Posterior Regularization [26, 24]. Ganchev et al. [24] described the relationships between these approaches and the subtle differences in the approximations these algorithms employed for inference.

We briefly describe Generalized Expectation (GE) in more detail, since it was specifically applied to feature labeling [21] and we will be using it in our experiments. GE is a framework for incorporating preferences about variable expectations during parameter estimation. We can describe GE as trying to maximize a score function S, between a models expectation of $f(X)$ and a target value $\tilde{f}$, denoted:

$$S(E_\theta[f(X)], \tilde{f}).$$

For instance, in Maximum Likelihood Estimation, the score function is the negative cross entropy and the target value is the empirical distribution on the training data. We can fit our algorithm into the GE framework by using the empirical distribution as the target value and a weighted negative cross entropy as the score function. Our approach incorporates feature labels by changing the score function. This is different from prior work with GE which incorporates feature labels by changing the target value.

Aside from supervised and semi-supervised feature labeling, other work in feature labeling investigated dual supervision [70], which is a term used to describe the process of labeling both instances and features. Raghavan and Allan [57] combined feature labeling with uncertainty sampling for instance labeling in their tandem learning approach. Other dual supervision approaches include a graph-based transduction algorithm [70] and an approach using pooled multinomials [6]. The focus of these last two papers was on active learning for dual supervision, which chooses instances and features for labeling. Our work differs in that it is the end-users, not the active learning algorithm, that choose the features for labeling. Furthermore, we are investigating the effects of labeling only features, not instances, especially with an eye to the initial training period when training data is limited.

Attenberg and Provost [5] investigated feature labeling for budget-sensitive learning under extreme class skew and found that it was a promising alternative for data acquisition. When humans had difficulty finding instances from the minority class, Attenberg et al. suggested that a less costly form of data acquisition would be for humans to describe distinguishing features of the minority class. Our work is not specifically intended for datasets with extreme class skew but for more balanced datasets. Nevertheless, both our work and the work by Attenberg and Provost [5] point to feature labeling being extremely beneficial in either setting.

All of the above methods dealt with labeling existing features. Roth and Small [62] allowed users to create new features by replacing features corresponding to semantically related words with a Semantically Related Word List (SWRL) feature. Their focus, however, was on creating SWRLs to improve classifiers rather than feature labeling. In our user study, we allow end-users to construct new features and label them. Finally, most of the prior work in feature labeling evaluated algorithms under ideal conditions, such as feature labels obtained from an oracle [6, 70]. Some prior work [58, 66] has evaluated feature labeling algorithms using both oracle feature labels and labels obtained from user experiments. However, these experiments were on a small scale with only a handful of users and a subset of these users were knowledgeable about machine learning. In contrast, we perform a large scale study involving 43 participants, all of whom had no background in machine learning or human computer interaction. These non-expert end-users can introduce noisy and inconsistent feature labels. Our study investigates both the use of ideal oracle feature labels and feature labels provided by real world end-users.

2.3 Feature Labeling

The semi-supervised setting for feature labeling incorporates knowledge from three sources: a small labeled training set, the feature labels provided by the end-user and information from the implicit structure of the unlabeled data. We evaluate our semi-supervised algorithm using both oracle feature labels and end-user feature labels from the user study mentioned earlier. Our analysis shows that incorporating unlabeled data during learning sometimes produces worse performance than just using a purely supervised learning approach, both with and without feature labeling. However, adding the information from feature labels consistently improves performance over not including this information, both in the supervised and semi-supervised settings.

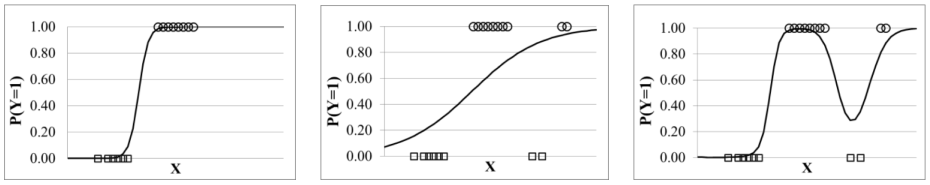
2.3.1 LWLR

Logistic Regression (LR) [28] is a well-known method in statistics for predicting a discrete class label y_i given a data instance $\mathbf{x}_i = (x_i^1, ..., x_i^D)$ with D features; we refer to the d-th feature, without reference to a specific data instance, using the superscript notation i.e. x^d. LR models the conditional probability $P(y_i|\mathbf{x}_i)$ by fitting a logistic function to the training data.

The conditional probability for an M-class problem is:

$$P_\theta(y_i = c_j|\mathbf{x}_i) = \frac{\exp \beta_j^0 + \sum_{d=1}^{D} \beta_j^d x_i^d}{\sum_{m=1}^{M} \exp \beta_m^0 + \sum_{d=1}^{D} \beta_m^d x_i^d} \tag{2.1}$$

In the equation above, the notation c_j refers to the j-th class. The parameters $\theta = (\beta_1, ..., \beta_M)$ are computed by maximizing the conditional log likelihood, which cannot be solved in closed form but must be done numerically.



(a) LR on linearly separable data (b) LR on unseparable data (c) LWLR on unseparable data

Figure 2.1: Example of a situation where LR fails but LWLR succeeds in separating two classes

LR assumes that the parameters θ are the same across all data points. Although this approach works reasonably well when the classes are linearly separable, it fails when the actual decision boundaries are more complex and when the data is noisy [17], which is often the case with real-world data. One solution for dealing with this case is to use Locally Weighted Logistic Regression (*LWLR*) [12, 17], in which the logistic function is fit locally to a small neighborhood around a query point $\mathbf{x}_q$ to be classified. LWLR gives more weight to training points that are "closer" to the query point than those farther away. A common function used to determine the closeness of text documents is cosine similarity. Since we want the distance to increase when a training instance $\mathbf{x}_i$ is less similar to the query instance $\mathbf{x}_q$, we use $\text{cosim}(\mathbf{x}_q, \mathbf{x}_i) = 1 - \cos(\mathbf{x}_q, \mathbf{x}_i)$ as the baseline distance function for *LWLR*. In Section 2.3.1, we will describe how we extend this reweighting of training instances to perform feature labeling.

The log-likelihood of data in *LWLR* is computed with respect to the query instance

$\mathbf{x}_q$ as:

$$l_w(\theta) = \sum_{i=1}^{N} w(\mathbf{x}_q, \mathbf{x}_i) \log(P_\theta(y_i|\mathbf{x}_i)) \tag{2.2}$$

where,

$$w(\mathbf{x}_q, \mathbf{x}_i) = \exp\left(-\frac{f(\mathbf{x}_q, \mathbf{x}_i)^2}{k^2}\right) \tag{2.3}$$

The weight $w(\mathbf{x}_q, \mathbf{x}_i)$ is a kernel function which decays with the distance $f(\mathbf{x}_q, \mathbf{x}_i)$. The parameter k is the kernel width, which controls the size of the neighborhood as the value of k increases. As a consequence of having to fit the logistic function locally to a query point, *LWLR* is considered a lazy algorithm and we now need to train the classifier each time it receives a query point. However, in many cases, we gain a much higher accuracy with this tradeoff in efficiency.

Maximizing $l_w(\theta)$ with respect to the parameters θ cannot be done in closed form. In our experiments, we solve it using *Limited-memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS) [52] for which we need to compute the partial derivative of $l_w(\theta)$ with respect to the β parameters. The partial derivative in Equation 2.4 computes the gradient for the log-likelihood. In this equation, the expression $[y_i = c_j]$ takes the value of 1 if the expression in the brackets is true, and 0 otherwise.

$$\frac{\partial}{\partial \beta_j^d} l_w(\theta) = \sum_{i=1}^{N} w(\mathbf{x}_q, \mathbf{x}_i) \left(x_i^d [y_i = c_j] - \frac{x_i^d}{Z(\mathbf{x}_i)} \exp\left(\beta_j^0 + \sum_{d'=1}^{D} \beta_j^{d'} x_i^{d'}\right)\right) \tag{2.4}$$

where,

$$Z(\mathbf{x}_i) = \sum_{j=1}^{M} \exp\left(\beta_j^0 + \sum_{d'=1}^{D} \beta_j^{d'} x_i^{d'}\right).$$

2.3.2 LWLR-FL

LWLR-FL incorporates feature labeling into *LWLR*. *LWLR* is modified for feature labeling because of its ability to weight training instances differently, rather than its ability to handle non-linear decision boundaries. We include feature labels (provided by the end-user) when we define the local neighborhood surrounding the query point. Training instances that are more similar to the query point according to the feature label information are considered to be closer and hence assigned higher weight. The baseline $\text{cosim}(\mathbf{x}_q, \mathbf{x}_i)$ distance function is modified to incorporate feature labels. The modified distance function between $\mathbf{x}_q$ and $\mathbf{x}_i$ has two distinct components – one based only on their features (satisfied by the baseline distance $\text{cosim}(\mathbf{x}_q, \mathbf{x}_i)$), and the other based on class labels. Since $\mathbf{x}_q$ does not have an associated class label, only the class label of $\mathbf{x}_i$ and the feature label information are used for computing the label similarity.

The label similarity between $\mathbf{x}_q$ and $\mathbf{x}_i$ is based on the difference between the class-relevant and other-class-relevant feature contributions. A *class-relevant* feature is a feature that is labeled with the class label y_i of instance $\mathbf{x}_i$ as specified by the feature labels. The class-relevant feature contribution is the sum of the values of all class-relevant features in $\mathbf{x}_q$, where $\mathbf{x}_q$ is represented as an L2-normalized *term frequency-inverse document frequency* (TF-IDF) vector. Similarly, an *other-class-relevant* feature is a feature that is labeled with a class label other than y_i. The other-class-relevant feature contribution refers to the sum of values of all other-class-relevant features in $\mathbf{x}_q$.

The user feature label matrix $\mathbf{R}$ is now defined as:

$$\mathbf{R}_{[D\times M]} = \begin{bmatrix} r_1(x^1) & \cdots & r_M(x^1) \\ \vdots & \ddots & \vdots \\ r_1(x^D) & \cdots & r_M(x^D) \end{bmatrix}, \tag{2.5}$$

where $r_j(x^d) = 1$ if the d-th feature is labeled to be important for class label c_j and 0 otherwise. The j-th column of $\mathbf{R}$ is denoted by $\mathbf{R}(j)$. Let $\mathbf{U}$ be a $(D \times 1)$ column vector in which the i-th entry is 1 if the i-th feature has been marked important in *any* class. All other entries are zero.

On the basis of the above definitions, the difference between the class-relevant and other-class-relevant feature weights is computed as:

$$\mathbf{R}(y_i)^T\mathbf{x}_q - \left(\frac{(\mathbf{U} - \mathbf{R}(y_i))^T\mathbf{x}_q}{M-1}\right) \tag{2.6}$$

The term $\mathbf{R}(y_i)^T\mathbf{x}_q$ is sum of class-relevant feature values for class y_i and $(\mathbf{U}-\mathbf{R}(y_i))^T\mathbf{x}_q$ is the sum of other-class-relevant feature values. Since we have $(M-1)$ class labels excluding y_i, we divide the other-class-relevant feature contributions by $(M-1)$ to appropriately balance the difference.

We want the distance between similar instances to be smaller. Hence, the label similarity component of the distance function is defined as:

$$1 - \mathbf{R}(y_i)^T \mathbf{x}_q - \left(\frac{(\mathbf{U} - \mathbf{R}(y_i))^T \mathbf{x}_q}{M-1} \right) \tag{2.7}$$

The complete distance function now becomes:

$$f(\mathbf{x}_q, \mathbf{x}_i) = \text{cosim}(\mathbf{x}_q, \mathbf{x}_i) \left[1 - \mathbf{R}(y_i)^T \mathbf{x}_q - \left(\frac{(\mathbf{U} - \mathbf{R}(y_i))^T \mathbf{x}_q}{M-1} \right) \right] \tag{2.8}$$

The above function can be negative in some cases. Hence, a max term in the weight computation is introduced to handle such a scenario.

$$w(\mathbf{x}_q, \mathbf{x}_i) = \exp\left(-\frac{\max(0, f(\mathbf{x}_q, \mathbf{x}_i))^2}{k^2} \right). \tag{2.9}$$

Putting these pieces together, we now have a distance function that incorporates the feature labels into *LWLR*.

The *LWLR-FL* and *LWLR* are lazy algorithms, meaning that they do not perform training until a query is made. This means each query has a computational complexity of $O(n)$, where n is the number of instances in the training set. Although the computational cost can be expensive with a large training set, the *LWLR-FL* algorithm is intended to be applied to small training sets during the initial period when a learning algorithm is first deployed. With small training sets, such as those in our experiments, each query only takes milliseconds on a standard desktop computer, making the *LWLR-FL* algorithm viable in an interactive setting.

2.3.3 Semi-supervised Learning

Before elaborating on how we incorporate unlabeled data into LWLR, we first present an overview of semi-supervised learning and the intuitions behind utilizing unlabeled data.

As mentioned in Section 1.2, semi-supervised learning utilizes data which is partly labeled and partly unlabeled. Unlabeled data is usually easy to obtain and does not

id	label	document
1	*sports*	The Patriots won Superbowl 2017.
2	?	We were watching Monday night football at the bar.
3	?	Only two football teams make it to the Superbowl.
4	*medicine*	He was admitted to the hospital after a massive heart attack.

Table 2.1: Partially labeled document corpus for semi-supervised learning.

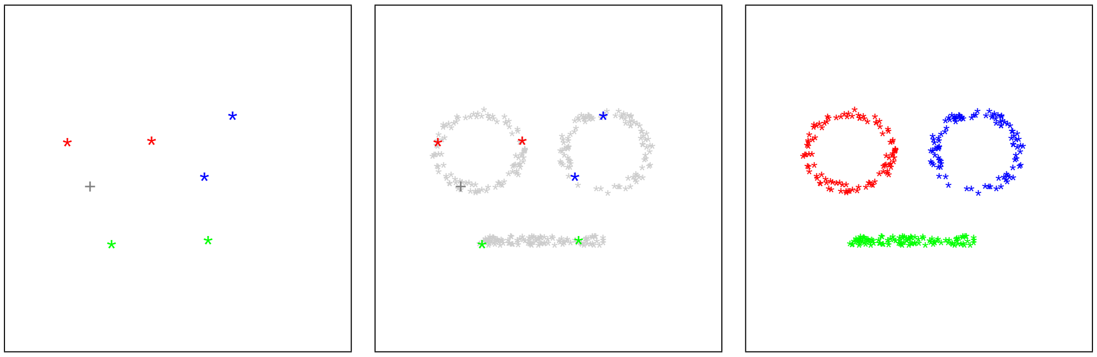
require expensive labeling effort by an expert. Despite the absence of labels, it serves to expose the structure which can be used to build bridges between different parts of the feature space.

For example, assume that we have a corpus with four documents and their corresponding labels as shown in Table 2.1. Here, only documents 1 and 4 are labeled. A purely supervised classifier might be trained on these two documents and then used to predict labels on documents 2 and 3. While the prediction on document 3 might be accurate (as it shares the word 'football' with document 1), prediction on document 2 seems hopeless – it has no words in common with the labeled documents. If, however, the classifier could reason that since document 3 is likely to be labeled *sports* and shares a word ('Superbowl') with document 2, then document 2 should also be assigned the label *sports*. To be able to make this type of inference, the entire corpus – both labeled as well as unlabeled data – should be available during training; and learning under this setting setting is called *semi-supervised*.

Semi-supervised learning distinguishes two different types of prediction tasks. When the goal is to predict labels of unlabeled data which were already available during training, it is called *transductive* semi-supervised learning. When the task is to predict labels for unseen future data, it is called *inductive* semi-supervised learning. The problems we address in this thesis are in the transductive setting.

There are many semi-supervised learning methods [84] such as self-training, probabilistic generative models, co-training, graph-based models, semi-supervised support vector methods, etc. In this chapter we apply a graph-based method that computes pairwise distances in the manifold space through label diffusion over the unlabeled data. We explain this method using an illustration below.

Assume that we only have labeled data as shown in Figure 2.2a. The labeled points are marked as '*'. Additionally, we also have an unlabeled instance marked as '+'.



(a) Small labeled data (b) Labeled data in presence of unlabeled data (c) Ideal labels

Figure 2.2: Exposing structure in data through unlabeled data

When asked to predict the label of the unlabeled instance, there is likely be confusion on whether to label it as green or red. If, on the other hand, we are shown other unlabeled data as in Figure 2.2b, we can see how the labeled instances are part of underlying structures which form separate classes. Those structures make it more likely that the unlabeled instance marked as '+' should be labeled red.

The above example assumes that there are manifold structures in the data that can be used to group instances. If the label for at least one instance in a group is known, it applies to all instances within the group. Correspondingly, the labeling effort is reduced to a large degree. In order to infer the manifold structure, a method commonly used in semi-supervised learning is *label diffusion* [82, 85]. We first describe label diffusion without feature labeling in this section. In the next section, we extend this to the setting where labeled features are available.

All instances (labeled and unlabeled) are represented as points in the feature space. We build a connected graph by joining each instance to its $\#nn$ nearest neighbors with undirected edges. The labeled training instances may now be considered as sources from which the labels propagate to unlabeled instances along the edges. There are several techniques for label propagation – one of which [82] is by an iterative Markov process until a stationary state is reached. At this stationary state, we have at a pairwise affinity matrix $\mathbf{A}$ which takes into account the distance the labels had to diffuse through in manifold space rather than just the Euclidean distances. Let $\mathbf{U}$ be the set of unlabeled instances, $\mathbf{L}$ be the set of labeled instances, and n be the total number of instances ($|\mathbf{U}|+|\mathbf{L}|$). Let $\mathbf{W}_{[n\times n]}$ be an initial affinity matrix, prior to label diffusion, that captures the similarity between two data instances. When $i = j$, $W_{ij} = 0$ otherwise for $i \neq j$, *LWLR-SS* uses the following similarity measure between any two instances $\mathbf{x}_q$ and $\mathbf{x}_i$:

$$W_{qi} = \exp\left(-\frac{||\mathbf{x}_q, \mathbf{x}_i||^2}{k^2}\right) \tag{2.10}$$

Define $\mathbf{D}_{[n\times n]}$ as a diagonal matrix where $D_{ii} = \sum_j W_{ij}$ and let $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. The matrix $\mathbf{A}_{[n\times n]} = (\mathbf{I} - \alpha\mathbf{S})^{-1}$ will contain all pairwise similarities when label diffusion reaches the stationary state. The similarities in matrix $\mathbf{A}$ are in range $[0, 1]$, where 1 means most similar, and 0 means least similar. Here the parameter α controls the rate of label propagation and $\mathbf{I}$ is an identity matrix. We view $\mathbf{A}$ as a distance matrix defined within a transformed space (referred to as the *manifold* space.)

The algorithm for semi-supervised learning is shown in Algorithm 1.

Algorithm 1 Semi-supervised Learning with Label Diffusion

```
1. Setup matrices $\mathbf{W}$, $\mathbf{D}$, $\mathbf{S}$
2. Compute $\mathbf{A} = (\mathbf{I} - \alpha\mathbf{S})^{-1}$
3. Normalize all values of $\mathbf{A}$ into range $[0, 1]$ by dividing row $\mathbf{A}_i$ by the diagonal
   element $A_{ii}$. Since the diagonal elements are self-similarities, these are the highest
   values in the corresponding rows and columns. An IsoMDS plot (Figure 2.4) using
   the computed pairwise distances in $\mathbf{A}$ shows that points which are in the same
   manifold structure cluster together.
4. For each unlabeled instance $\mathbf{x}_q \in \mathbf{U}$, train a classifier with instance weights set to
   normalized $A_{qi}$, where $\mathbf{x}_i \in \mathbf{L}$ and classify $\mathbf{x}_q$.
```

2.3.4 LWLR-SS-FL

Previously, the notion of locality around an unlabeled query instance $\mathbf{x}_q$ (in *LWLR* and *LWLR-FL*) was based on a similarity measure between only labeled instances and $\mathbf{x}_q$. We now extend the similarity measure to include information from other unlabeled instances

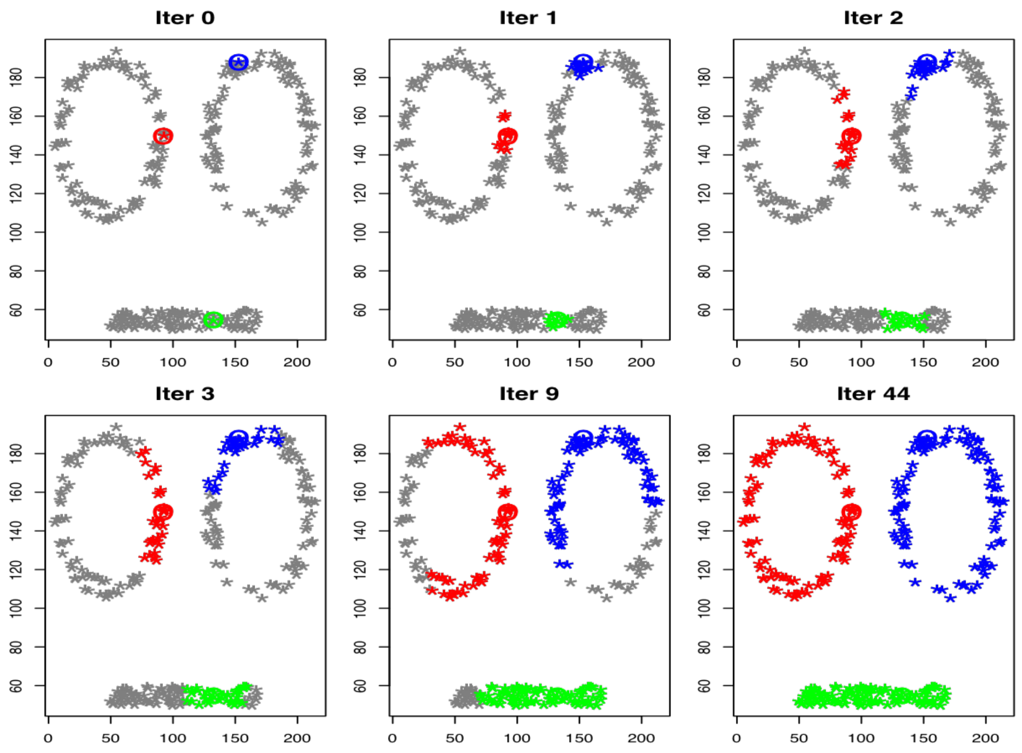


Figure 2.3: Label Diffusion. The circles represent the initial set of labeled training examples. The colors red, blue, and green represent three different classes. Points in grey are unlabeled instances. At each iteration, the labels 'diffuse' from the currently labeled instances to their nearest neighbors. After a number of iterations of label diffusion, the distribution of the labels over the instances becomes stationary.

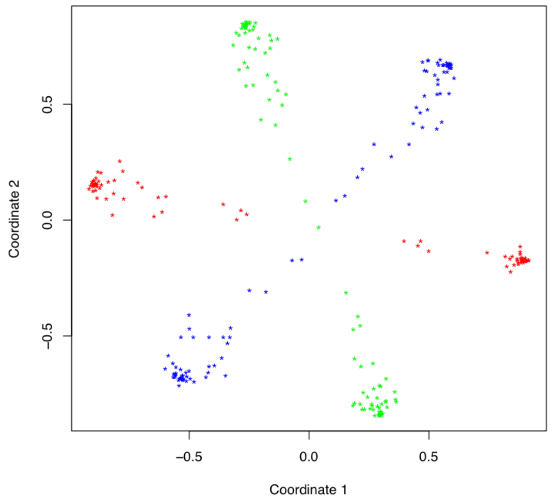


Figure 2.4: Clustering in manifold space

as well using label diffusion. We refer to our semi-supervised algorithm which uses no feature labeling information as *LWLR-SS* and the one using feature labeling information as *LWLR-SS-FL*.

In order to modify Algorithm 1 for *LWLR-SS*, we first define the similarity matrix $\mathbf{W}$ as in Equation 2.11 where W_{qi} is the similarity between any two instances $\mathbf{x}_q$ and $\mathbf{x}_i$. Next, the likelihood function for the query instance $\mathbf{x}_q$ for *LWLR-SS* is modified to be $l_w(\theta) = \sum_{i=1}^{N} A_{qi} \log(P_\theta(y_i|\mathbf{x}_i))$ where we have replaced $w(\mathbf{x}_q, \mathbf{x}_i)$ by A_{qi} in Equation 2.3:

$$W_{qi} = \exp\left(-\frac{\text{cosim}(\mathbf{x}_q, \mathbf{x}_i)^2}{k^2}\right) \tag{2.11}$$

Similarly, for *LWLR-SS-FL* we use the similarity matrix defined in Equation 2.12. Apart from this modification to the similarity measure, the *LWLR-SS-FL* algorithm is identical to the *LWLR-SS* algorithm:

$$W_{qi} = \begin{cases} \exp\left(-\frac{\max(0, f(\mathbf{x}_q, \mathbf{x}_i))^2}{k^2}\right) & \text{when only one of } \mathbf{x}_i \text{ or } \mathbf{x}_q \text{ is labeled,} \\ \exp\left(-\frac{\text{cosim}(\mathbf{x}_q, \mathbf{x}_i)^2}{k^2}\right) & \text{otherwise.} \end{cases} \tag{2.12}$$

Dataset	Number of Instances	Classes Used
20 Newsgroups	2750	*comp.sys.ibm.pc.hardware(953), misc.forsale(759), sci.med(557), and sci.space(481)*
Modapte	1100	*earn(300), acq(300), negative_topic(250), and money-fx(250)*
RCV1	6300	*C15(1260), CCAT(1260), ECAT(1260), GCAT(1260), and MCAT(1260)*
WebKb	3695	*Course(930), faculty(1124), and student(1641)*
Industry Sectors	3311	*basic.materials(950), energy(355), financial(290), healthcare(400), technology(500), transportation(515), and utilities(301)*
Movie Review	2000	*pos(1000) and neg(1000)*

Table 2.2: The classes of the data sets used in the oracle study, along with the number of instances in each class shown in parentheses.

2.3.5 Experiment Results

To evaluate LWLR-SS-FL, we applied it to six real-world text data sets (*20 Newsgroups, Modapte, RCV1, WebKb, Industry Sectors, and Movie Review*) with two kinds of studies. First, to avoid the prohibitive expense of performing a separate user study on each data set, we followed the usual machine learning methodology [57, 70], and simulated end-user feature labeling on multiple data sets using a feature label oracle. Second, we then performed a study with real users on one particular data set to investigate the effectiveness of using feature labels obtained from end-users.

2.3.6 Oracle Study

In our oracle-based experiments, we employed six common text classification datasets: 20 Newsgroups [39], the Modapte split of the Reuters dataset [1], the Reuters Corpus Volume 1 (RCV1) dataset [41], WebKb [14], the Industry Sector dataset [48], and the Movie Review dataset [53]. As a pre-processing step, the text documents were converted into TF-IDF representation and then L2-normalized. We used a vocabulary consisting of unigrams with stopwords removed.

Table 2.2 summarizes the number of instances and the classes used from each of these datasets. Class imbalance, however, can be a problem in some of these datasets. For instance, in WebKb, the smallest class is approximately 1/16th the size of the largest class. Such class imbalance, if unaddressed, would make the classifier focus on improving the accuracy of only those classes which have a large number of instances, and it severely degrades the accuracy of the smaller classes. In order to avoid class imbalance issues, we chose the largest classes with roughly the same number of data instances in each class and avoided classes with an extremely small number of data instances.

For 20 Newsgroups, we chose four classes (Table 2.2) that end-users in our user study could understand easily without the need for specialized knowledge. Since we would also present articles from these newsgroups to end-users in our user study (Section 4.2), we wanted to preserve the topical coherence of articles by choosing articles that fell within a relatively short date range that included a large number of articles from these newsgroups. As a result, we chose 2750 articles from these four newsgroups within the date range April 1, 1993-April 23, 1993.

In order to evaluate the semi-supervised learning algorithms, which assume a pool of unlabeled data instances is available during training, we employed the same 6 datasets as those selected for supervised learning as well as the same oracle feature labels. Furthermore, we applied the same training/validation/test splits, but unlike in supervised learning, we made the unlabeled data instances from the test set available during training. We employed the same oracle feature labels from the previous section and present results when 10 oracle feature labels per class were provided to the semi-supervised algorithms.

We compared LWLR-SS-FL against SVM-M3, GE, and MNB/Priors+EM, which are representative algorithms from the two general strategies employed for semi-supervised feature labeling. For the SVM-based methods, we experimented with different combinations of Method 1 and Method 2 with Method 3, but found that SVM-M3 worked the best. To avoid clutter, our results only show the results from SVM-M3. We chose GE because it was specifically applied for feature labeling in [21] and because the GE code was readily available in the Mallet package [47]. GE has also been shown empirically to perform better than other algorithms (e.g., learning with measurements) with small training data sets [42], which was our particular problem setting.

For LWLR-SS-FL, as suggested in [82], we set the rate parameter α to 0.99. We did not tune the k parameter over the validation set due to the computational expense of tuning LWLR-SS-FL, but set $k = \sqrt{0.08}$ for all datasets. This value was determined empirically and was found to give good results. The number of nearest neighbors was fixed to 100 for all datasets.

We used GE with Schapire distributions as in Druck et al. [21], where the majority class was assigned a weight 0.9. The Gaussian prior in GE was tuned within the range of values from 0.2 to 1.0 at steps of 0.2 using the validation set. The GE objective function can be modified to weight the GE term and the likelihood of the data in order to balance the effects of feature labels with training data. Since the training sets in our experiments were very small, the likelihood term had neglible effect, and we found that using only the GE term produced the best results.

For all SVM methods, including SVM-M3, the parameters C, a, and r were tuned using a validation set. The parameter d was fixed to 1.0. For MNB/Prior and MNB/Prior+EM we tuned the prior α using the validation set. We also tuned the soft-labeling weight for unlabeled instances in MNB/Prior+EM using the validation set.

2.3.6.1 Results of Oracle Study

The results for the Oracle study are shown in Figure 2.5. LWLR-SS-FL outperformed the other algorithms on the 20 Newsgroups, Modapte, and Industry Sectors datasets. GE, however, had the best macro-F1 on the WebKb and Movie Review datasets. For the RCV1 dataset, SVM-M3 had the best macro-F1.

Past work in semi-supervised learning (e.g. [7]) has indicated that the label diffusion approach to semi-supervised learning was successful if the data had a smooth underlying manifold structure in which instances from the same class were close enough to each other to allow labels from labeled training instances to propagate to unlabeled instances that were of the same class. Label diffusion performed poorly if the “islands” of data instances from one class fell in between “islands” of data instances from another class on the manifold structure. We believe that the poor behavior of LWLR-SS-FL on the WebKb and Movie Review datasets was due to these datasets having an underlying structure that did not satisfy this particular assumption of label diffusion.

Figure 2.6 compares the relative benefits of augmenting learning from labeled training instances using feature labeling, unlabeled data, and a combination of the two. In general, information from feature labels helped learning more than information from unlabeled data. In fact, in four datasets (Modapte, RCV1, WebKb, and Movie), the unlabeled data caused the performance of LWLR-SS to degrade below that of the baseline supervised learning LWLR algorithm. Surprisingly, combining feature labeling with semi-supervised learning overcame this deficit, resulting in better performance than the baseline. Semi-supervised feature labeling, however, did not necessarily outperform supervised feature labeling. On three datasets (RCV1, WebKb, Movie Review), LWLR-SS-FL performed worse than LWLR-FL. Overall, however, incorporating information from feature labels always improved performance in both the supervised and semi-supervised settings, as one can see by the improvement of LWLR-FL over LWLR and the improvement of LWLR-SS-FL over LWLR-SS.

Macro F1, Adding 10 Oracle Feature Labels per class (Semi-supervised)						
Algorithm	20 Newsgroups	Modapte	RCV1	Industry Sectors	WebKb	Movie Review
SVM	0.635	0.747	0.559	0.391	0.715	0.597
MNB	0.643	0.665	0.540	0.386	0.604	0.583
LWLR	0.652	0.774	0.554	0.396	0.670	0.628
LWLR-SS-FL	**0.900**†	**0.844**†	0.643	**0.703**†	0.745	0.708
SVM-M3	0.809	0.827	**0.649**	0.659	0.851	0.758
GE	0.823	0.766	0.556	0.663	**0.876**†	**0.782**†
MNB/Prior+EM	0.806	0.431	0.535	0.460	0.601	0.744

Table 2.3: Results of incorporating 10 oracle features per class through semi-supervised feature labeling for all six datasets. The values in **bold** in each column represent the highest scores for the corresponding dataset. The symbol † denotes values that are significantly greater than all other algorithms at the 0.05 level (Wilcoxon signed-rank test, $p < 0.05$). Results for the baseline SVM and LWLR algorithms are included for reference.

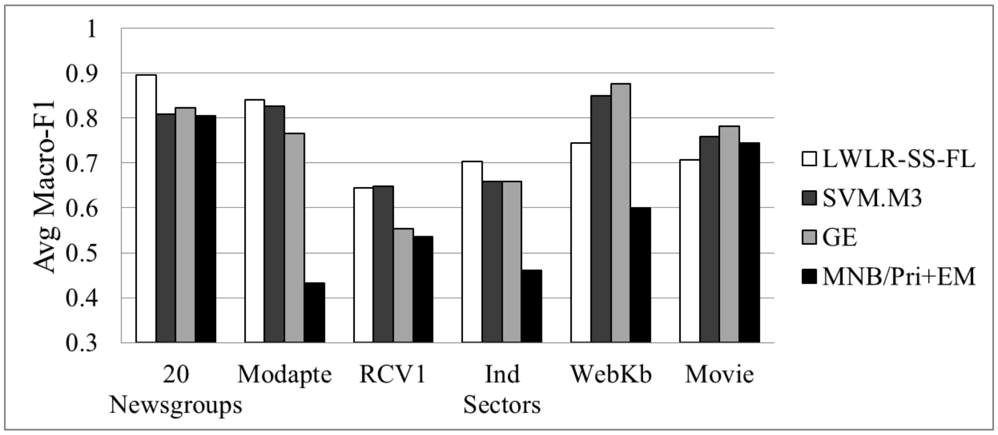


Figure 2.5: Average macro-F1 scores for incorporating 10 Oracle feature labels through semi-supervised feature labeling for the six datasets used in our experiments.

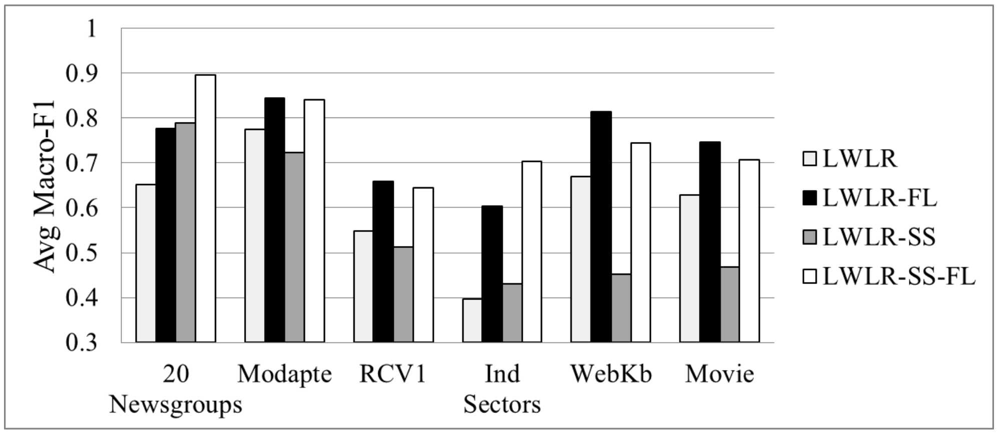


Figure 2.6: A comparison of the relative benefits of feature labeling and unlabeled data for variants of LWLR-FL.

2.3.7 End-user Labeling Study

The strength of oracle studies is the ability to evaluate a variety of data sets, but their weakness is that they may not be realistic as to the choices real users might make. Therefore, for our second experiment, we conducted a user study to harvest feature

Macro F1, Adding 10 Oracle Feature Labels per class (Semi-supervised)						
Algorithm	20 Newsgroups	Modapte	RCV1	Industry Sectors	WebKb	Movie Review
LR	0.623	0.773	0.582	0.396	0.712	0.586
LWLR	0.652	0.774	0.554	0.396	0.670	0.628
LWLR-FL	0.777*	**0.844***	**0.660***	0.603*	**0.815***	**0.747***
LWLR-SS	0.789*	0.723	0.512	0.430*	0.452	0.468
LWLR-SS-FL	**0.900***	0.840*	0.644*	**0.703***	0.745*	0.691*

Table 2.4: Results of adding 10 oracle features per class for all six datasets for algorithms that are variants of logistic regression. The symbol * denotes values that are significantly greater than the baseline LWLR algorithm at the 0.05 level (Wilcoxon signed-rank test, $p < 0.05$)

labels from actual end-users on the same 20 Newsgroups classes as used in Section 2.3.6. We then used the users' data to compare the performance of the same algorithms as in our oracle study, but with smaller validation sets of size 24 (six instances for each class) to simulate a realistic scenario in which users were able to label only a limited amount of training instances for both a training and a validation set.

A starting point for the experiment's design was the user study by Raghavan et al. [58]. However, an important difference was that we chose to remove constraints on the features users were allowed to pick. Specifically, rather than having users select features from a pre-computed list, we allowed them to identify features by freely highlighting text directly in the documents. This gave the participants complete freedom to choose any features that they thought were predictive. Consequently, not only were the users allowed to select existing features in the algorithms representation, but also to create and label new features, such as through combinations of words or punctuation.

The user study had 43 participants: 24 males and 19 females. For the main experiment, the participants were shown 24 previously labeled documents in four topics: Computers, Things For Sale, Medicine, and Outer Space (corresponding to the four newsgroups comp.sys.ibm.pc.hardware, misc.forsale, sci.med, and sci.space, respectively). Each of the four topics had six documents assigned to it, which were randomly selected from a pool of 200 training instances. The order of the documents was randomized for each participant. Participants were asked to teach the machine "suggestions" by identifying features that they believed would help it label future documents. Within a time limit of twelve minutes, participants were asked to provide at least two suggestions per topic, with an emphasis placed on selecting the best features for each newsgroup.

We used participant-provided feature labels instead of the oracle feature labels to compare the performance of LWLR-FL and LWLR-SS-FL against the other supervised and semi-supervised methods described in Section 2.3.2 and 2.3.4. Participants could label features by highlighting any text – they did not have to know whether their feature existed before (recall that we used a vocabulary of unigrams with stopwords removed for the original representation). If a participant created a new feature, we added it to the document representation used for that participant's data and created a corresponding feature label for it. Using these data, we analyzed two variants of this experiment: one variant used participants' labels on existing features only, and the other used all features that participants provided.

2.3.7.1 Results of End-user Labeling Study

Figure 2.7 summarizes the results from this experiment. We also plot the performance of the LWLR and SVM baselines as a reference. Results with 8 oracle feature labels per class are shown in the leftmost group. We divide the results into the "existing" features group, in which feature labels were only permitted on existing features, and the "all" features group, in which new features that participants created were added to the data representation and then labeled. Results for "existing" and "all" features are shown in the middle and rightmost groups respectively.

For SVM-M3, end-user feature labels degraded the performance of the algorithm below the SVM baseline in this semi-supervised setting, while LWLR was more robust. GE performed the worst out of all the other algorithms with user feature labels, indicating that it was very sensitive to the quality of the feature labels. The poor performance of GE was due to the lower quality user feature labels being the only source of supervision for GE's learning process. Furthermore, in past work, GE performed very well when users were guided to provide feedback on features selected by topic models or by active learning [21], unlike in our setup where the users had no guidance at all as to which features to label. Both LWLR-SS-FL and MNB/Priors+EM improved upon their respective baseline algorithms with user feature labels. LWLR-SS-FL significantly outperformed other algorithms on the "all" feature cases (Wilcoxon signed-rank test, $p < 0.05$).

When compared against the gains from supervised feature labeling (Figure 2.7), SVM-M3 was more sensitive to lower quality feature labels from participants than its supervised learning counterpart (SVM-M1M2); SVM-M1M2 performed better than SVM-M3 and resulted in an improvement over the SVM baseline. Similarly, MNB/Priors+EM performed slightly worse than its supervised learning counterpart (MNB/Priors). Unlike the former two algorithms, LWLR-SS-FL outperformed its supervised learning counterpart (LWLR-FL) in all cases.

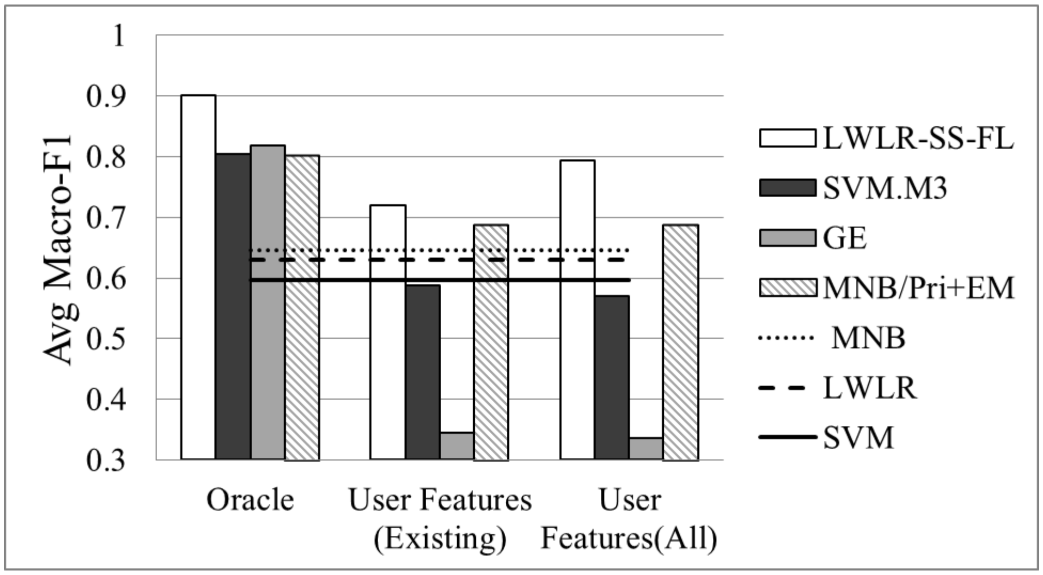
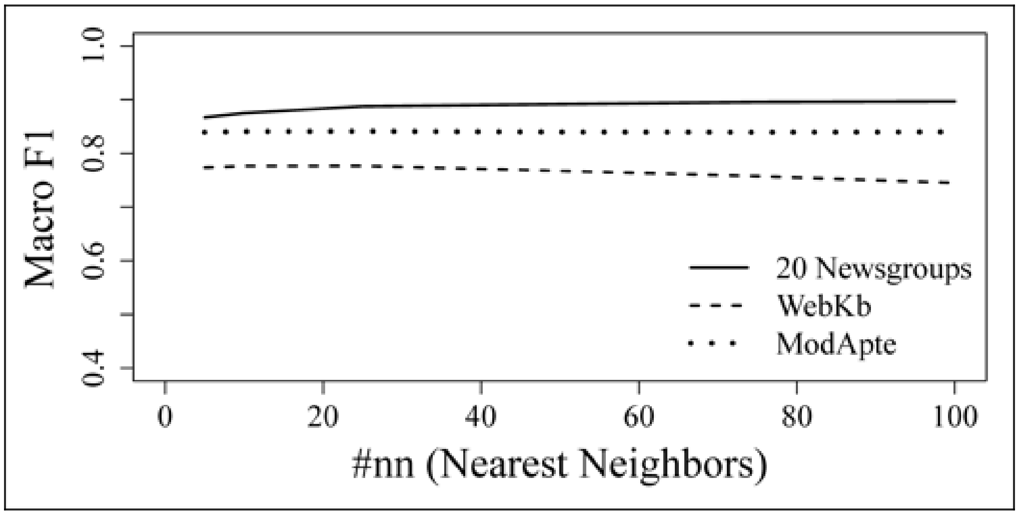


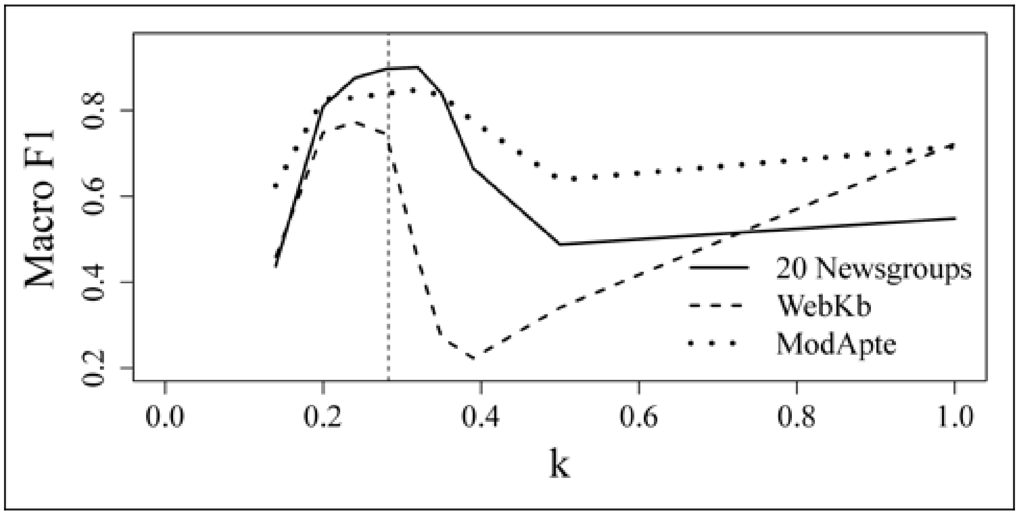
Figure 2.7: Results of Semi-supervised Algorithms with User Feature Labels: (Left) incorporating 8 oracle feature labels per class, (Middle) incorporating end-user feature labels only for existing features, (Right) incorporating all end-user feature labels.

2.3.8 Sensitivity Analysis

LWLR-SS-FL introduces two new parameters – the label diffusion kernel width (k), and the number of nearest neighbors ($\#nn$). In our experiments, we set k to $\sqrt{0.08}$ and $\#nn$ to 100 for all datasets in LWLR-SS and LWLR-SS-FL. Here, we present the sensitivity plots for k and $\#nn$ on three datasets – 20 Newsgroups, ModApte, and WebKB. In the sensitivity analysis of parameter k, we kept $\#nn$ constant at 100 and varied k in the



(a) number of nearest neighbors $\#nn$.



(b) diffusion kernel width k

Figure 2.8: Sensitivity of LWLR-SS-FL to hyper-parameters.

range [0-1]. In the sensitivity analysis of parameter $\#nn$, we kept k constant at $\sqrt{0.08}$ and varied $\#nn$ in the range [5-100].

Figure 2.8a shows that the algorithm was not very sensitive to the number of nearest neighbors ($\#nn$). However, the algorithm was more sensitive to k, as can be seen in Figure 2.8a. Since we used TFIDF-L2 normalization, a kernel of width 1.0 spanned across all instances, and therefore, a "global" fit. We can see that these datasets had similar localized regions where label diffusion helped. The performance of the algorithm on 20 Newsgroups hardly improved beyond $k = 0.5$, which might suggest that instances in this dataset formed very few compact clusters in the feature space and the seed (labeled training) instances managed to cover most of them. In WebKB on the other hand, performance degraded untill $k = 0.4$ and then recovered as k increased to 1 and beyond. This suggests that the WebKB categories might have formed a large number of small clusters that were intermingled and the seed instances had not been able to cover all small clusters. ModApte showed characteristics similar to WebKB, but managed to avoid the steep degradation in performance as observed in WebKB at around $k = 0.4$. This could have been because ModApte had fewer instances than WebKb and hence was sparser in the feature space.

2.4 Discussion

With oracle feature labels, the semi-supervised feature labeling algorithms produced a dramatic increase over their respective baselines, which was similar to results reported in previous work [57, 21]. However, with the lower quality features that came from real users, some semi-supervised feature labeling algorithms performed worse than algorithms that ignore the feature labels (e.g., SVM-M3 performed worse than its SVM baseline). Past work in semi-supervised learning [83, 84] has shown that semi-supervised learning does not always produce an improvement in performance over supervised learning. In his survey on semi-supervised learning [83], Zhu pointed out that a mismatch between model assumptions and the problem structure could produce worse performance than supervised learning, but "detecting this mismatch in advance is hard and remains an open problem".

In semi-supervised feature labeling, the oracle feature labels were generated using the entire labeled data set and thus fit the structure of the data. On the other hand, end-user feature labels could exacerbate the mismatch between the model assumptions and the problem structure. Much of the past work on semi-supervised feature labeling had evaluated algorithms against oracle feature labels. Although oracle feature labels provide an informative "upper bound", for semi-supervised feature labeling, it is also extremely important to evaluate against lower quality feature labels from real users, which can cause dramatically different behavior for these algorithms.

LWLR-SS-FL is suitable for small datasets because it needs to be trained separately for each test instance. Apart from this, there are two other issues with the LWLR-SS-FL algorithm that need to be addressed in future work. First, the LWLR-SS-FL algorithm is more sensitive to the value of k than LWLR-FL. In our experiments, we set $k = \sqrt{0.08}$ for all the datasets. Although this value worked well, we plan to investigate on how to make the algorithm more robust to parameter settings of k. Second, the LWLR-SS-FL algorithm involves a matrix inversion, where the number of rows in the matrix is the total number of labeled and unlabeled instances. A naïve implementation of matrix inversion does not scale well to large datasets, since the complexity is $O(n^3)$ (where n is the number of training and test instances). This is a roadblock to applying LWLR-SS-FL to large datasets in an interactive setting, which is an important requirement of feature labeling [66]. However, we have two advantages here over the general case of matrix inversion, which we plan to leverage in future work. First, our matrix is sparse and symmetric, thus reducing the actual number of computations. Second, we only need the rows in the inverted matrix that correspond to the training instances, which can also significantly reduce the computational complexity.

Overall, semi-supervised feature labeling algorithms can produce large improvements in performance if feature labels cause the resulting model to match the problem structure and if the algorithm's key parameters are set correctly. This is precisely the case with oracle feature labels, which consistently produced improvements over the baseline algorithms. In the case of lower quality feature labels from users, our results showed that LWLR-SS-FL was more robust to lower quality feature labels than other algorithms, but it must be more computationally efficient to be applicable in an interactive setting.

2.5 Conclusion

When deployed, machine learning systems need to customize themselves to an end-user's preferences quickly in order to be useful. However, training data is often limited, especially during the inital stages. As an alternative, we introduced semi-supervised feature labeling, which allows the user to provide corrective feedback by indicating which feature-value pairs are predictive of the class label. The semi-supervised nature of the algorithm allows it to leverage both the labeled data and the existing unlabeled data. We developed a new algorithm, LWLR-SS-FL, which extends LWLR-FL to a semi-supervised setting for text classification. We showed that it performed well when provided with a small amount of high quality labeled features and instances. We compared its performance to other semi-supervised feature labeling algorithms and also analyzed its sensitivity to hyper-parameters.

Chapter 3: Feedback to Anomaly Detection algorithms

3.1 Introduction

High accuracy anomaly detection algorithms have the potential to solve difficult problems in many application domains, including insider threat detection [64], biosurveillance [78], computer security [27] and data cleaning [18]. The goal of anomaly detection is to identify anomalies, which are unusual data instances of interest. More precisely, we define an *anomaly* to be a data instance that is generated by a process that is different from the process generating the "normal" data instances, which we refer to as the *nominal instances*. A dataset typically contains a small percentage of anomalies, such as 1% of the data instances or even less.

Since known anomalies are scarce, most anomaly detection algorithms (e.g. [36, 10, 44, 55]) are applied in an unsupervised setting in which an unlabeled data set is used to build a model of nominal data instances, even though the data may be contaminated by a small percentage of anomalies. The top B outlier instances under the nominal data model are then identified as anomaly candidates. We use the symbol B to refer to a *budget* as B is often determined by some combination of the expected anomaly rate and the amount of human resources available to investigate the candidate anomalies. This approach, however, usually leads to high false positive and false negative rates.

One cause of poor performance is that not all outliers are anomalies and not all anomalies are outliers. If the distribution of nominal data points has heavy tails, then many outliers will be nominal. Conversely, in adversarial situations, the adversary is trying to mimic the nominal data points, so the anomalies will be buried in regions of high nominal density. There is no statistical solution to these problems—the designer of the anomaly detection system must choose features to mitigate these problems. A second cause of poor performance is that it is very difficult to perform any kind of feature selection in unsupervised anomaly detection. As most anomaly detection applications involve high-dimensional feature spaces, this leads inexorably to poor anomaly detection performance. In this paper, we describe a method for incorporating expert feedback to adjust the anomaly detector so that it puts more weight on relevant regions of the feature space and ignores regions that do not correspond to the expert's semantic understanding of the anomalies.

We consider the following interactive data exploration loop. Initially, the anomaly detector is applied to an unlabeled dataset. Then, the anomaly detector presents a data instance to the expert analyst, and asks the analyst to label it as anomalous or nominal. The analyst labels the instance, and the anomaly detector updates its model with the newly acquired instance label. The process continues with the next iteration presenting another data instance to the analyst. This process iterates until a budget B on the total number of instances presented to the analyst is spent. Our goal is to maximize the number of true anomalies presented to the analyst. Although the budget B will vary depending on the cost of investigating each potential anomaly, we expect $B \leq 100$ for most applications.

3.2 Related Work

We call our approach *Active Anomaly Discovery* (AAD)[1] to differentiate it from work in the closely-related areas of active learning [65], rare category detection [54], and learning to rank [45]. We now describe the differences between AAD and these three areas.

Active learning involves selecting the most informative instances to be queried (i.e. labelled) so that a classifier trained on this strategically-chosen set of instances can maximize its predictive performance on an unseen test set. Various criteria for selecting instances have been explored, including uncertainty sampling [73], query-by-committee [68], variance reduction [13], and expected model change [67] (see [65] for a more thorough survey). AAD differs from active learning in terms of its objective; thus, rather than maximizing the predictive performance of the classifier on an unseen test set, AAD's objective is to maximize the total number of true anomalies presented to the analyst during the interactive labeling process. AAD does not differentiate between a training and a test set. Instead, AAD works in a transductive setting in which it is applied to a single dataset containing mostly unlabeled data and a small amount of labeled data.

[1]An early version of this work appeared in [16]. The current work includes a more comprehensive set of results for AAD, introduces approximations to speed up the AAD algorithm and it includes results of applying AAD to other ensemble anomaly detection algorithms.

Active learning has been previously applied to anomaly detection. A common approach is to combine the outputs from supervised and unsupervised techniques. For instance, the ALADIN algorithm [71] uses a classifier to temporarily label unlabeled data instances. Given these temporary labels plus the labeled data, a model is then trained for each class. During active learning, the total number of queries from the ALADIN algorithm is evenly distributed among the classes, with half of the queries for uncertain instances and the other half for anomalous instances. Similarly, the AI^2 algorithm [76] builds a query strategy based on the output of an unsupervised model and the output of a supervised model. In each round, k queries are selected, where half of the queries are the most anomalous instances as determined by the unsupervised model and the other half of the queries are the most anomalous instances as determined by the supervised model. The drawback to these approaches which use supervised learning techniques is that they become very unstable if the class imbalance becomes extremely skewed due to the scarcity of anomalies. In the extreme case, no anomalies have been identified in the training data and the supervised learning algorithms cannot be trained.

Another active learning strategy combines uncertainty sampling with querying the most likely anomalies [51]. Görnitz et al. [2013] also employ a combination strategy but in their work, the querying strategy blends uncertainty sampling with sampling instances from potentially anomalous clusters. We compare against this hybrid querying strategy in our experiments. We emphasize that none of the active learning techniques have the goal of maximizing the number of anomalies presented to the labeler as the objective is to maximize predictive performance on an unseen test set.

Rare category detection (RCD) [54] addresses a slightly different task. RCD is intended to be applied to data in which the anomalies form small but dense clusters that correspond to new categories of items. Given a fixed budget of queries to the analyst, the goal of RCD is to identify a representative instance from as many distinct categories in the data as possible. Query strategies for discovering rare categories include querying instances with greatest density differential [29, 31], with furthest distance from a cluster center [54], or querying according to a multi-scale clustering structure discovered by hierarchical mean shift [75]. We can summarize the difference between RCD and AAD as follows: RCD aims for diversity of anomalies presented to the analyst while AAD aims to simply present anomalies, regardless of whether they are from diverse categories or not. RCD also differs from AAD in terms of the type of feedback the analyst provides.

With RCD, the analyst identifies a data instance as belonging to one of K existing classes or to a new, previously unseen class. AAD places a lighter cognitive burden on the analyst by only requiring instances to be labeled as *anomaly* or *nominal* rather than placing them into K distinct classes. Furthermore, if the anomalies do not form K small, compact clusters in feature space or if the analyst cannot easily classify anomalies into these K categories, then AAD is a more appropriate approach to use for discovering anomalies than RCD. In Section 3.4.5, we perform an empirical comparison of applying RCD algorithms to the AAD problem.

Our work is also closely related to the area of Learning to Rank (LTR) [45], which is an active area of research in document retrieval. In LTR, the goal is to learn a ranking function that can return a total (or partial) order over a set of instances. One strategy for LTR is based on learning from pairwise preferences [32, 19], which is a technique that we adapt to our approach. The majority of LTR methods require a large amount of labeled data (i.e. pairwise preferences) to be effective. To reduce the labeling effort needed to produce pairwise preferences, several authors have proposed active learning (AL) in the context of LTR [81, 46, 19, 20, 56]. Recently, loss functions for maximizing accuracy at the top of the ranked list [9, 34] have been proposed, and we use these loss functions in our work.

On the surface, anomaly detection can be viewed as a ranking problem with the goal of ranking anomalies higher than nominals. However, there are fundamental differences. In LTR, the goal is to rank items on the basis of preference. In contrast, anomaly detection distinguishes between anomalies and nominals. The objective of anomaly detection is to rank anomalies higher than nominals. In our work, only ranks between anomalies and nominals are important. Within the anomalies the ranks are irrelevant, as are ranks within the nominals. There are situations in which some anomalies may be ranked higher than others because of their importance. Although we do not deal with this setting, our algorithm can be extended to handle it. Furthermore, in our setting, we begin with fully unlabeled data, making it impossible to apply LTR. As the AAD algorithm progresses, a small number of labeled instances are available as well as a large amount of unlabeled instances. Of these labeled instances, often a small number of them are anomalies. In the extreme case, there are no labeled anomalies. This class imbalance limits the availability of pairwise rankings between anomalies and nominals. Thus, it is difficult to apply LTR in a straightforward manner.

3.3 Methodology

In our setting, we are given a dataset $\mathbf{D}$ with n instances, $\mathbf{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$. Each instance i is a tuple $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a d-dimensional real vector and $y_i \in \{\mathit{anomaly}, \mathit{nominal}\}$ is the (hidden) class label. We assume that there is a large class imbalance, with $y_i = \mathit{nominal}$ comprising the overwhelming majority of the data.

The anomaly detector is first trained on the unlabeled data. Then, a subset of the class labels is progressively revealed to the analyst, who provides the class labels during the interactive feedback loop. In the first iteration, the model chooses a data instance $\mathbf{x}_1^q$ to present to the analyst, and the analyst provides a class label y_1^q for the queried instance. This process continues until a total of B instances have been queried. We represent the sequence of queries as $\mathbf{X}_B^q = (\mathbf{x}_1^q, \mathbf{x}_2^q, \ldots, \mathbf{x}_B^q)$. The class labels associated with these instances are provided as feedback by the analyst. We denote the analyst feedback as $\mathbf{F} = ((\mathbf{x}_1^q, y_1^q), (\mathbf{x}_2^q, y_2^q), \ldots, (\mathbf{x}_B^q, y_B^q))$.

The goal of our work is to maximize the number of true anomalies seen by the analyst over the B total instances presented:

$$\arg\max_{\mathbf{X}_B^q} |\{(\mathbf{x}_i^q, y_i^q) \in \mathbf{F} : y_i^q = \mathit{anomaly}\}|. \tag{3.1}$$

Equation 3.1 cannot be computed, because we do not know the true class labels. As an alternative, we greedily select instances to query that have the highest probability of being an anomaly under our model. Even though our approach is greedy, we show that it is very effective in our empirical results. To achieve the goal, our model maintains a ranked list of data instances where the rank is determined by the anomaly score computed by the anomaly detector.

In the subsections to follow, we first describe the anomaly detector in Section 3.3.1. Then, in Section 3.3.2, we define what it means to be at the top of the ranked list of anomalies, and we show how to maximize accuracy at the top in Section 3.3.3. In Section 3.3.4, we define our objective function for incorporating expert feedback into the anomaly detector. Finally, we describe the active anomaly discovery step in Section 3.3.5.

3.3.1 Anomaly Detector

For our anomaly detector, we employ the *Loda* algorithm [55], which computes an ensemble $\mathbf{P} = \{\mathbf{p}_m\}_{m=1}^{M}$ of M one-dimensional histogram density estimators computed from sparse random projections. Each projection $\mathbf{p}_m$ is defined by a sparse d-dimensional random vector $\boldsymbol{\beta}_m$, with $1/\sqrt{d}$ randomly chosen non-zero components; each value of these non-zero components is drawn from a standard normal distribution. Loda constructs a one-dimensional density estimator f_m by projecting each data point onto the real line according to $\mathbf{p}_m(\mathbf{x}_i) = \boldsymbol{\beta}_m^{\top}\mathbf{x}_i$ and forming a histogram density estimator f_m. The anomaly score assigned to point $\mathbf{x}_i$ is the mean negative log density (mean surprise):

$$\text{score}(\mathbf{x}_i)^{Loda} = \frac{1}{M}\sum_{m=1}^{M} -\log(f_m(\mathbf{x}_i)) \tag{3.2}$$

We can view Loda as a representation transformation that converts a data point $\mathbf{x}_i$ in the d-dimensional feature space into a log probability vector in M-dimensional real space. Denote the latter as $\mathbf{z}_i = [-\log(f_1(\mathbf{x}_i)), ..., -\log(f_M(\mathbf{x}_i))]^T$, where $\mathbf{z}_i \in \mathbb{R}^M$. The negative-log-pdf for the entire dataset will be represented by $\mathbf{H} = [\mathbf{z}_1, ..., \mathbf{z}_n]^{\top}$. With this notation and defining $\mathbf{w}_U = [\frac{1}{M}, ..., \frac{1}{M}]^T \in \mathbb{R}^M$, we can write Equation 3.2 in a more compact form: $\text{score}(\mathbf{x}_i)^{Loda} = \mathbf{w}_U \cdot \mathbf{z}_i$.

Loda gives equal weight to all projections, and since these projections are selected at random, it is not guaranteed that every projection is good at isolating anomalies by itself. Once the projections have been computed by Loda and fixed, we propose to integrate analyst feedback by learning a better weight vector $\mathbf{w}$ that assigns higher weights to the more useful projections and lower weights to the less useful ones.

Our approach is not restricted to Loda. Other ensemble methods based on Feature Bagging (e.g. [40]) can also be employed, and we explore this further in Section 4.2. More generally, this method can be applied to any ensemble method that combines the scores of different anomaly detectors.

3.3.2 The top τ-quantile

Internally, our model maintains a list of data instances ranked by the anomaly score produced by the anomaly detector. We wish to keep the labeled anomalies at the top of the ranked list. We define the top of the list to be the top τ-quantile. The top τ-quantile value of a function $h : \mathcal{X} \rightarrow \mathbb{R}$ is defined as the value $q_\tau : P_x(h(x) > q_\tau) = \tau$.

For any $\tau \in [0, 1]$, let ρ_τ be defined as:

$$\forall u \in \mathbb{R},\ \rho_\tau(u) = -\tau(u)_- + (1-\tau)(u)_+$$

where,

$$(u)_+ = \max(u, 0), \text{ and } (u)_- = \min(u, 0)$$

The τ quantile value q_τ of a sample of real numbers $\{u_1, ..., u_n\}$ can be computed as $q_\tau = \arg\min_{u \in \mathbb{R}} \sum_{i=1}^{n} \rho_\tau(u_i - u)$ [37]. An alternative (and natural) way to compute the top τ-quantile value is to sort all values in descending order and return the value that is $\tau \cdot n$ from the top of the sorted list.

3.3.3 Accuracy at the top

We want the scores of all labeled anomalies to be higher than q_τ and the scores of all labeled nominals to be below q_τ. When this property is violated on a specific data instance $(\mathbf{z}_i, y_i)$, we incur the loss shown in Equation 3.3, where $\mathbf{w}$ is a vector of weights. Equation 3.3 is based on the surrogate empirical loss function defined in the Accuracy at the Top (AATP) approach [9].

$$\ell(q_\tau, \mathbf{w}; \mathbf{z}_i, y_i) = \begin{cases} 0 & \mathbf{w} \cdot \mathbf{z}_i \geq q_\tau \text{ and } y_i = \text{`}anomaly' \\ 0 & \mathbf{w} \cdot \mathbf{z}_i < q_\tau \text{ and } y_i = \text{`}nominal' \\ (q_\tau - \mathbf{w} \cdot \mathbf{z}_i) & \mathbf{w} \cdot \mathbf{z}_i < q_\tau \text{ and } y_i = \text{`}anomaly' \\ (\mathbf{w} \cdot \mathbf{z}_i - q_\tau) & \mathbf{w} \cdot \mathbf{z}_i \geq q_\tau \text{ and } y_i = \text{`}nominal' \end{cases} \tag{3.3}$$

The *AATP* approach in [9] was presented in a supervised learning setting, meaning that it expects all pairwise preferences between relevant and irrelevant items to be available during training. In contrast, all pairwise preferences between anomalies and nominals are not available in our initial dataset, and we typically only obtain a small subset of these pairwise preferences when instance labels are revealed. As a result, the loss in Equation 3.3 needs to be computed by summing up the loss over the instances in the labeled set $\mathbf{H}_F$, i.e., the set of instances for which the analyst has already provided feedback. The weights $\mathbf{w}$ that minimize the overall loss can be computed as:

$$\begin{aligned} \mathbf{w} &= \arg\min_{\mathbf{w}} \left(\sum_{\mathbf{z}_i \in \mathbf{H}_F} \ell(q_\tau, \mathbf{w}; (\mathbf{z}_i, y_i)) \right) \\ \text{s.t.,} & \\ q_\tau &= \arg\min_{u} \sum_{\mathbf{z}_i \in \mathbf{H}} \rho_\tau(\mathbf{w} \cdot \mathbf{z}_i - u) \end{aligned} \tag{3.4}$$

Equation 3.4 is not convex because the equality constraint is not affine [9]. This makes joint inference of $\mathbf{w}$ and q_τ hard. As an alternative, we solve for $\mathbf{w}$ and q_τ separately. Let $\mathbf{w}^{(t-1)}$ be the value of $\mathbf{w}$ from iteration $(t-1)$. The steps are as follows:

1. Solve $q_\tau = \hat{q}_\tau(\mathbf{w}^{(t-1)})$ using the (fixed) value of $\mathbf{w}^{(t-1)}$. We compute $\hat{q}_\tau(\mathbf{w}^{(t-1)})$ by ranking the anomaly scores given $\mathbf{w}^{(t-1)}$.

2. Compute $\mathbf{w}^{(t)}$ using Equation 3.4 keeping $q_\tau = \hat{q}_\tau(\mathbf{w}^{(t-1)})$ fixed.

Under this approach, there is no guarantee that $\hat{q}_\tau(\mathbf{w}^{(t-1)})$ still remains the τ-th quantile score under the new value $\mathbf{w}^{(t)}$. Nevertheless, this approximation was performed in [9] and shown to produce reasonable results.

3.3.4 Objective Function

A straightforward application of the AATP approach produces the objective function in Equation 3.4. Even with the approximation for $\hat{q}_\tau(\mathbf{w})$, however, Equation 3.4 performs poorly at discovering true anomalies due to two main issues. First, since anomalies are rare, the feedback set $\mathbf{H}_F$ will typically have many more nominals than anomalies. However, the anomalies are more informative and we need to give them higher importance. Second, we want the labeled anomalies to be ranked higher than the labeled nominals. The objective function in Equation 3.4 does not enforce this ranking. Instead, it tries to place labeled anomalies above the τ-th quantile and labeled nominals below the τ-th quantile in such a way that the loss is minimized. It is possible that at the minimum value, there are anomalies below the τ-th quantile or nominals above the τ-th quantile. These misplacements can occur if the parameter τ, which is supposed to correspond to the true fraction of anomalies in the data, is set to be too large or too small. Getting the value of τ to be exactly the fraction of anomalies in the data is difficult to do and one needs to make the algorithm robust to this misspecification.

Our overall objective function is shown in Equation 3.5, and we discuss four necessary modifications below. These modifications all contribute gains to the performance and we will provide a detailed analysis in Section 3.4.3.

$$\begin{aligned}
\mathbf{w}^{(t)} = &\arg\min_{\mathbf{w},\xi} \frac{C_A}{|\mathbf{H}_A|}\left(\sum_{\mathbf{z}_i\in\mathbf{H}_A} \ell(\hat{q}_\tau(\mathbf{w}^{(t-1)}),\mathbf{w};(\mathbf{z}_i,y_i))\right)\\
&+\frac{1}{|\mathbf{H}_N|}\left(\sum_{\mathbf{z}_i\in\mathbf{H}_N} \ell(\hat{q}_\tau(\mathbf{w}^{(t-1)}),\mathbf{w};(\mathbf{z}_i,y_i))\right)\\
&+\|\mathbf{w}-\mathbf{w}_p\|^2 + C_\xi\sum_{i,j}\xi_{ij}^2 \qquad (3.5)\\
\text{s.t.,}&\\
&(\mathbf{z}_i-\mathbf{z}_j)\cdot\mathbf{w}+\xi_{ij}\geq 0 \quad \forall \mathbf{z}_i,\mathbf{z}_j : \mathbf{z}_i\in\mathbf{H}'_A, \mathbf{z}_j\in\mathbf{H}_N,\\
&\xi_{ij}\geq 0
\end{aligned}$$

where, $\mathbf{w}_p = \frac{\mathbf{w}_U}{\|\mathbf{w}_U\|} = [\frac{1}{\sqrt{m}},\ldots,\frac{1}{\sqrt{m}}]^T$, $\hat{q}_\tau(\mathbf{w}^{(t-1)})$ is computed by ranking anomaly scores using $\mathbf{w}^{(t-1)}$ and,

$$\mathbf{H}'_A = \begin{cases} \{\mathbf{z}_\tau\} & \text{when } \mathbf{H}_A = \emptyset \\ \mathbf{H}_A & \text{when } \mathbf{H}_A \neq \emptyset \end{cases}$$

($\mathbf{z}_\tau$ is the transformed τ-th quantile *proxy* anomaly instance $\mathbf{x}_\tau$)

The modifications to Equation 3.4 are as follows. First, we divide the labeled dataset $\mathbf{H}_F$ into the set of labeled anomalies $\mathbf{H}_A$ and the set of labeled nominals $\mathbf{H}_N$. We then introduce a weight C_A that causes the loss for anomalies in $\mathbf{H}_A$ to be higher than that associated with nominals. Typically, C_A needs to be larger than 1, with higher values placing more emphasis on the anomalies. In Section 3.4.6, we perform a sensitivity analysis on C_A and our results indicate that the performance is not overly sensitive to C_A as long as it is bigger than 1.

Second, we use soft pairwise constraints [32] (i.e. $(\mathbf{z}_i - \mathbf{z}_j) \cdot \mathbf{w} + \xi_{ij} \geq 0$) to encourage labeled anomalies to have higher scores than labeled nominals. It might not be feasible to satisfy all such constraints; therefore we introduce *slack variables*[2] ξ_{ij} in the final objective (Equation 3.5). At first, it might appear that the soft constraints are redundant when we already have the hinge-loss in the objective. However, these pairwise constraints are necessary when τ is incorrectly set to be smaller than the true fraction of anomalies in the data, in which case anomalies end up below the τ-th quantile, or when τ is larger than the true fraction of anomalies, resulting in nominals above the τ-th quantile. A possible shortcoming is that we are potentially introducing a large number of constraints which will make the optimization expensive. In practice, we assume that the number of instances labeled by an user is limited by a small budget (e.g., < 100 queries). When the number of true anomalies found is small, this optimization can be solved in a reasonable amount of running time on most modern computers, as we demonstrate in Section 3.4.4. However, if the number of true anomalies found is high, a large number of pairwise constraints will be added. In Section 3.4.4, we discuss further optimizations that can speed up the running time

Third, the proximal factor $\|\mathbf{w} - \mathbf{w}_p\|^2$ avoids the degenerate solution of $\mathbf{w} = 0$ and regularizes the learned weights toward the uniform weights of Loda, which has been shown to perform well as an anomaly detector [55, 22]. As we show in Section 3.4.3, without this term, the performance degrades because the algorithm will severely overfit the small amount of labeled data.

Finally, since anomalies are very few in number, it is very likely that at least initially, all instances labeled by the analyst are nominal. In this case, the number of pairwise constraints reduces to zero because there are no labeled anomalies. To overcome this problem, we introduce $\mathbf{z}_\tau$ as a *proxy* anomaly. The vector $\mathbf{z}_\tau$ is computed from $\mathbf{x}_\tau$, which is the τ-th ranked instance producing quantile value $\hat{q}_\tau(\mathbf{w}^{(t-1)})$. Once $\mathbf{x}_\tau$ has been identified, we compute $\mathbf{z}_\tau = [-\log(f_1(\mathbf{x}_\tau)), ..., -\log(f_M(\mathbf{x}_\tau))]^T$, which is the transformed

[2]There is a slight change from the formulation in [16] as we square the slack variable term. We found that this change helps the optimization by making the objective function more convex and thus yields slightly better results.

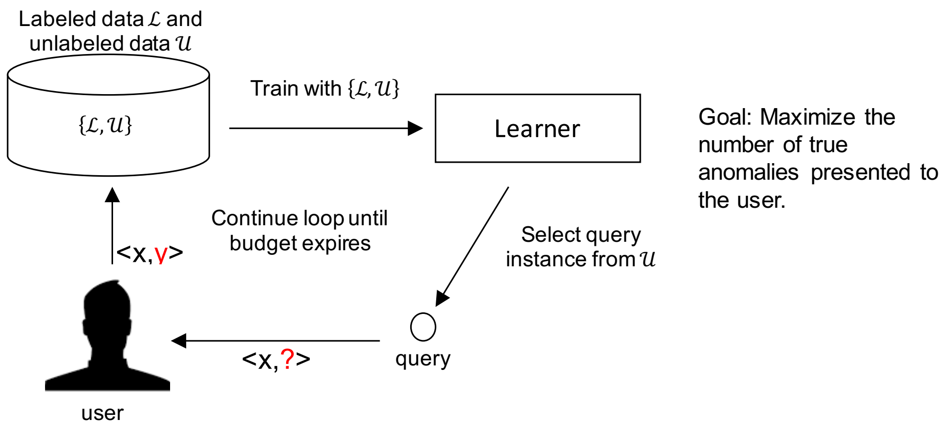

Labeled data $\mathcal{L}$ and unlabeled data $\mathcal{U}$
$\{\mathcal{L},\mathcal{U}\}$
Train with $\{\mathcal{L},\mathcal{U}\}$
Learner
Goal: Maximize the number of true anomalies presented to the user.
Continue loop until budget expires
<x,y>
Select query instance from $\mathcal{U}$
query
<x,?>
user

Figure 3.1: Setup for Active Anomaly Discovery.

version of $\mathbf{x}_\tau$. We then generate pairwise constraints enforcing all labeled nominals to have lower score than the proxy anomaly. This choice is guided by our assumption that labeled nominal instances should be ranked below the top τ-quantile. Note that the constraint introduced by the proxy anomaly uses the weights $\mathbf{w}$ from the current iteration, which is different from the quantile loss function which uses the weights $\mathbf{w}^{(t-1)}$ from the previous iteration. This difference is subtle but important; it provides additional information to the optimization that leads to a gain in performance. The proxy anomaly only exists as a constraint in the optimization until the first labeled anomaly is found. The proxy anomaly ensures that the known nominals, and instances that are similar to them, are forced lower in the ranking due to pair-wise constraints while other potential anomalous instances rise to the top.

3.3.5 Active Anomaly Discovery (AAD)

Algorithm 3.3.5 describes the active anomaly discovery loop for our approach. The setup is illustrated in Figure 3.1. Note that we ask the analyst to label the top ranked instance that has not already been labeled. The updated weights at Line 16 of the algorithm are computed using a standard convex optimization package (*Scipy* in our Python implementation) which solves the objective in Equation 3.5.

Algorithm 2 Active Anomaly Discovery (AAD)

```
    **Input:** Raw data $\mathbf{D}$, negative-log-pdfs $\mathbf{H}$, budget $B$
    Initialize the weights $\mathbf{w}^{(0)} = \{\frac{1}{\sqrt{m}}, ..., \frac{1}{\sqrt{m}}\}$
    Set $t = 0$
    Set $\mathbf{H}_A = \mathbf{H}_N = \emptyset$
    while $t \leq B$ do
      $t = t + 1$
      Set $\mathbf{a} = \mathbf{H} \cdot \mathbf{w}$ (i.e., $\mathbf{a}$ is the vector of anomaly scores)
      Let $\mathbf{x}_i$ = instance with highest anomaly score (where $i = \arg\max_i(a_i)$)
      Let $\mathbf{z}_i$ = negative log probability vector corresponding to $\mathbf{x}_i$
      Get feedback $\{$'anomaly'/'nominal'$\}$ on $\mathbf{x}_i$
      if $\mathbf{x}_i$ is anomaly then
        $\mathbf{H}_A = \{\mathbf{z}_i\} \cup \mathbf{H}_A$
      else
        $\mathbf{H}_N = \{\mathbf{z}_i\} \cup \mathbf{H}_N$
      end if
16:   $\mathbf{w}^{(t)}$ = compute new weights; normalize $\|\mathbf{w}^{(t)}\| = 1$
    end while
```

3.4 Results

3.4.1 Experimental Setup

Since our objective is to maximize the number of true anomalies presented to the end user, we plot the total number of true anomalies discovered against the number of queries presented to the user. We call this plot an *anomaly discovery curve* and in an ideal result, this curve climbs as quickly as possible. In our experiments we assume that the user is an *oracle* that labels instances correctly. We compare the results of our proposed algorithm against four other algorithms:

1. **Baseline:** For the baseline, we present instances in decreasing order of anomaly score computed with the original Loda algorithm (i.e. with uniform weights). This baseline captures the performance of an unsupervised anomaly detector that does not incorporate expert feedback.

2. **Semi-supervised Anomaly Detector (SSAD):** The algorithm proposed by Görnitz et al. [25] encodes labeled anomalies and nominals as constraints and does

Table 3.1: Datasets used in our experiments, along with their characteristics.

Dataset	**Nominal Class**	**Anomaly Class**	**Total**	**Dims**	**#Anomalies (%)**
Abalone	8, 9, 10	3, 21	1920	9	29 (1.5%)
ANN-Thyroid-1v3	3	1	3251	21	73 (2.25%)
Cardiotocography	1 (Normal)	3 (Pathological)	1700	22	45 (2.65%)
Covtype	2	4	286048	54	2747 (0.9%)
Covtype-sub	2	4	2000	54	19 (0.95%)
KDD-Cup-99	*'normal'*	*'u2r', 'probe'*	63009	91	2416 (3.83%)
KDD-Cup-99-sub	*'normal'*	*'u2r', 'probe'*	2000	91	77 (3.85%)
Mammography	-1	+1	11183	6	260 (2.32%)
Mammography-sub	-1	+1	2000	6	46 (2.3%)
Shuttle	1	2, 3, 5, 6, 7	12345	9	867 (7.02%)
Shuttle-sub	1	2, 3, 5, 6, 7	2000	9	140 (7.0%)
Yeast	*CYT*, *NUC*, *MIT*	*ERL*, *POX*, *VAC*	1191	8	55 (4.6%)

not need any labeled data to initialize. The best performing query strategy for SSAD was demonstrated to be the combination strategy called *margin and cluster* [25]. In our experiments we refer to this version of SSAD as SSAD-M+C. For comparison, we also include in our experiments the query strategy that selects the top-ranked anomaly instance for feedback (SSAD-Top).

3. **AI2:** AI^2 [76] is a system that incorporates analyst feedback for detecting malicious attacks on information systems. It is comprised of an ensemble of unsupervised outlier detection methods and a supervised learning algorithm. The technique

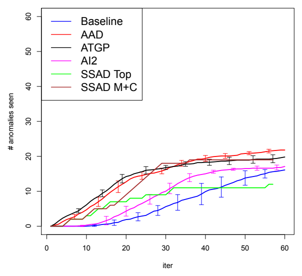
is fairly general and can be applied to most anomaly detection problems. In our implementation of AI^2, we use Loda's random projections as the unsupervised ensemble members, and for the supervised algorithm we use L1-regularized Logistic Regression. In our experiments, we use a batch size of 2, meaning AI^2 queries the user with one instance suggested by the ensemble of unsupervised algorithms and an instance suggested by the supervised algorithm (if such an instance is available).

4. **ATGP:** Grill and Pevný [27] proposed a supervised algorithm that is also based on maximizing accuracy at the top. Unlike our approach, ATGP does not add pairwise constraints and it uses a simple gradient approach for inferring the detector weights. Although this algorithm was presented in a fully supervised setting, it could easily be adapted for active anomaly discovery.

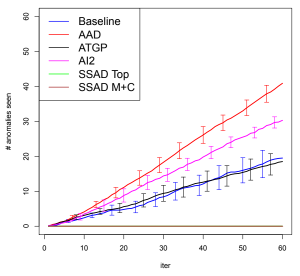
We also investigated a querying strategy for AAD that resembles uncertainty sampling in that it queries unlabeled instances near the "margin". In our setting, the margin would be the current instance $\hat{q}_\tau(\mathbf{w}^{(t)})$ representing the top τ-th quantile. We found that this querying strategy produced very poor results, so we do not include these results to reduce clutter on the graphs.

In our experiments, we used the *Mammography* [79] dataset as well as seven datasets from the UCI repository [2]: *Abalone*, *Cardiotocography*, *Thyroid (ANN-Thyroid)*, *Forest Cover (Covtype)*, *KDD-Cup-99*, *Shuttle* and *Yeast*. The number of true anomalies and true nominals in each dataset are shown in Table 3.4. We implemented our own variant of AI^2 as mentioned above. Our implementation of all algorithms and the datasets in our experiments are available online[3]. For SSAD we used the code made available from the authors[4]. This implementation requires an $n \times n$ kernel matrix and therefore does not scale to large datasets. Therefore, for the larger datasets (*Covtype*, *KDD-Cup-99*, *Mammography*, *Shuttle*), we also include results from a smaller version of the original dataset which was created by sub-sampling 2000 data instances and keeping the ratio of anomalies to nominals the same as in the original. These sub-sampled datasets are named **-sub* (Table 3.4). For the *Cardiotocography* dataset, we retained all instances from the *nominal* class as in the original dataset, but down-sampled the *anomaly* instances so

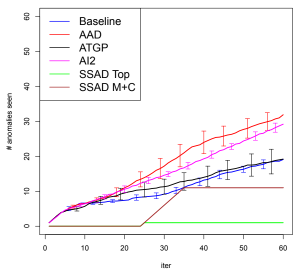
[3]https://github.com/shubhomoydas/aad.git
[4]https://github.com/nicococo/tilitools



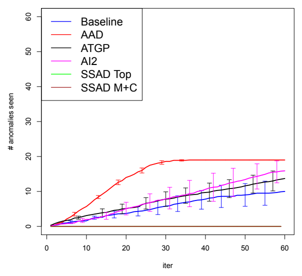
(a) Abalone



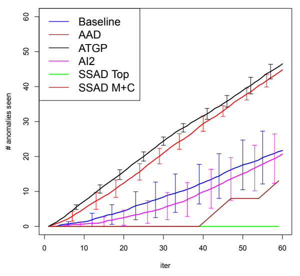
(b) ANN-Thyroid-1v3



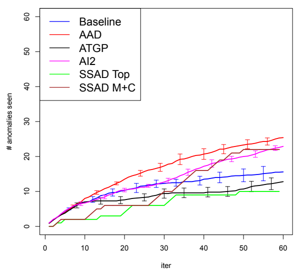
(c) Cardiotocography



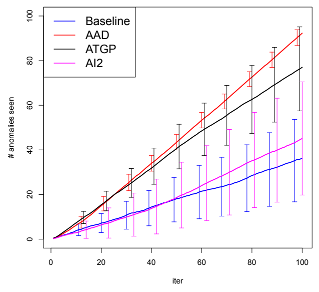
(d) Covtype-sub



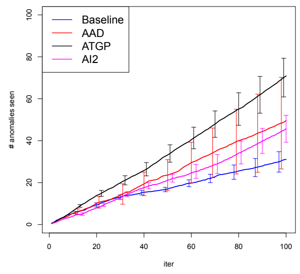
(e) KDD-Cup-99-sub



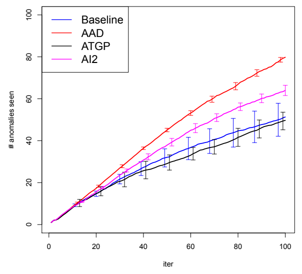
(f) Mammography-sub



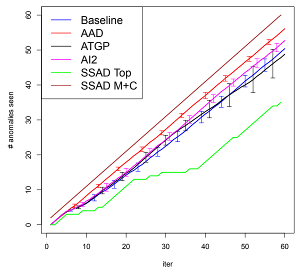
(g) Covtype



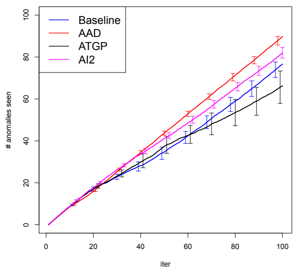
(h) KDD-Cup-99



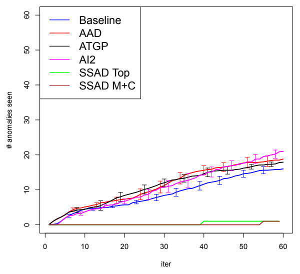
(i) Mammography



(j) Shuttle-sub



(k) Shuttle



(l) Yeast

Figure 3.2: The total number of true anomalies seen vs. the number of queries for all datasets. Total number of queries for the **smaller** datasets (*Abalone*, *Cardiotocography*, *ANN-Thyroid-1v3*, *Covtype-sub*, *KDD-Cup-99-sub*, *Mammography-sub*, *Shuttle-sub* and *Yeast*) is 60. Total number of queries for the **larger** datasets (*Covtype*, *KDD-Cup-99*, *Mammography*, *Shuttle*) is 100. The results were averaged over 10 runs for each algorithm (except SSAD, which is deterministic). The error bars show the 95% confidence intervals.

that they represent only around 2% of the total data. The rest of the datasets were used in their entirety by all algorithms. SSAD is deterministic and was therefore run once per dataset. Since AAD and AI^2 are randomized, their results were averaged across 10 runs for each dataset. The 95% confidence intervals from these 10 runs are plotted on the curves as error bars. In Table 3.4, the number of dimensions of each dataset (under the 'Dims' column) includes the additional dimensions added after performing the standard transformation of multi-class categorical variables into multiple binary variables using dummy coding.

For AAD, we set the parameters $\tau = 0.03$, $C_A = 100$, and $C_\xi = 1000$ by default for all datasets. AAD is not sensitive to τ over a wide range of values $[0.01, 0.10]$, or to C_A. It generally performs well for $C_\xi > 1$. We include results on a sensitivity analysis in Section 3.4.6.

SSAD [25] relies on parameters C_u and C_n. The authors recommend a grid search in $[0.01, 100]$ for these parameters. We ran *SSAD* with values $\in \{0.01, 0.1, 1.0, 10.0, 100.0\}$ for each of these parameters; we gave SSAD an advantage by reporting the C_u and C_n parameter values producing the best results, chosen *after* running the experiments. We fixed $\kappa = 1$ as was set by Görnitz et al. [25]. For all SSAD experiments, we used the *RBF* kernel. The *margin and cluster* strategy to query instances for SSAD requires checking labels of some number of nearest neighbors. We have set this parameter to 10 for all datasets.

Figure 3.2 depicts the performance of all algorithms on the datasets. The results on the full version of the larger datasets do not include SSAD since it was too computationally expensive to run. These results show that AAD is consistently one of the best performers, with substantial improvements in the number of anomalies detected compared to other algorithms on the *ANN-Thyroid-1v3*, *Covtype-sub*, *Shuttle* and *Mammography* datasets. The AAD algorithm's anomaly discovery curve is clearly higher than that of all other algorithms on 6 of the 13 datasets. The AAD algorithm can be considered tied with the top algorithms on the *Abalone* and *Yeast* datasets. On *Shuttle-sub*, *KDD-Cup-99-sub* and *KDD-Cup-99* datasets, the AAD algorithm is outperformed by the SSAD-M+C and ATGP algorithms.

Although AAD and ATGP are similar, ATGP performs worse than AAD on most datasets. Our hypothesis is that even though both AAD and ATGP performed non-convex optimization, and are therefore prone to local optima, the pairwise constraints of AAD likely restrict it to better parts of the parameter space.

Shuttle is the easiest dataset; it has the highest percentage of anomalies discovered. For this dataset, almost all instances ranked at the top by the baseline Loda are true anomalies. Therefore, querying the top ranked instance is very effective for AAD and AI^2. Surprisingly, SSAD-Top, which also queries instances ranked at the top by SSAD, discovers fewer anomalies than SSAD-M+C. We hypothesize that this might be due to the presence of clustered anomalies in the *Shuttle* dataset, which SSAD-M+C is able to exploit more than AAD.

Although SSAD-M+C performs best on *Shuttle-sub*, its performance varies widely across datasets. On *ANN-Thyroid-1v3*, *Cardiotocography*, *KDD-Cup-99-sub* and *Covtype-sub*, it performed very poorly.

AI^2 comes in second place to AAD on a number of datasets (*ANN-Thyroid-1v3*, *Cardiotocography*, *Covtype-sub*, *Mammography* and *Shuttle*). AI^2 performs poorly on the full *Covtype* dataset, which is nearly 100 times larger than *Covtype-sub*. The unsupervised and supervised parts of AI^2 struggle initially to find true anomalies in this much larger dataset, which results in poor performance for the relatively small budget of 100. Similarly, we hypothesize that the poor performance of AI^2 on *KDD-Cup-99* and *KDD-Cup-99-sub* is due to the poor performance of its supervised learning part.

Unlike its competitors, AAD is consistently among the best performing algorithms across all datasets, and it always performs better than its baseline.

3.4.2 Query Visualization

In order to gain a deeper understanding of AAD, we provide visualizations of our datasets, along with the queries by AAD. There are many possible techniques for visualizing high dimensional datasets and discussing the pros and cons of each approach is beyond the scope of this paper. Our intent is to provide some intuition as to how AAD works. For this purpose, we use t-SNE [74], which is a popular dimensionality reduction technique, to produce a 2D representation of the datasets in our experiments. Figure 3.3 shows a 2D representation of the *Abalone* and *ANN-Thyroid-1vs3* datasets produced by t-SNE, along with the queries made by AAD and by the baseline algorithm which does not incorporate expert feedback. Visualizations of all the small datasets can be found in Appendix A.1.

As shown in Figure 3.3 and in Appendix A.1, some datasets have fairly clustered anomalies (e.g. *Covtype-sub*, *KDD-Cup-99-sub* amd *Shuttle-sub*). The remaining datasets have a mixture of both clustered and scattered anomalies (e.g. *Abalone*, *ANN-Thyroid-1vs3*, *Cardiotocography*, *Mammography* and *Yeast*).

Without expert feedback, the baseline tends to discover nominals at the same locations, shown visually as a cluster of green circles (e.g. around (0,0) in *Abalone*) in Figure 3.3. In contrast, when expert feedback is incorporated by AAD, more anomalies are discovered. For example, there are fewer blue plus signs (e.g. around $(15, -80)$ in *ANN-Thyroid-1vs3*). The AAD algorithm is also querying instances at different locations as we can see by noting that there are fewer clusters of green circles (e.g. around $(30, -50)$ in *Abalone*). Overall, AAD appears to avoid unpromising regions of the feature space and hone in on fruitful regions.

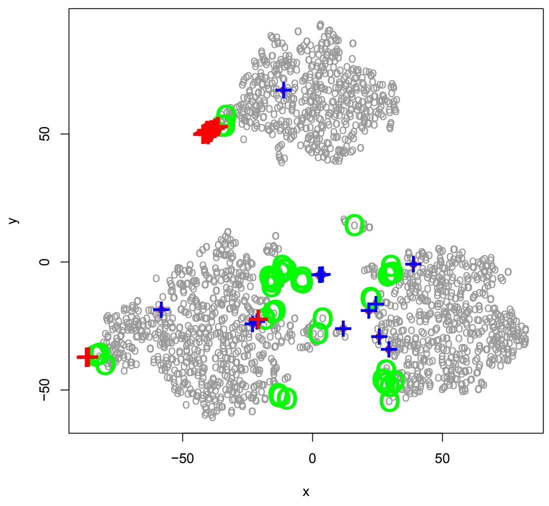
3.4.3 Contributions of Sub-parts of the Optimization Problem

In this section, we discuss the importance of the various parts of the AAD optimization problem. In Equation 3.3.4, the optimization problem has three main parts:

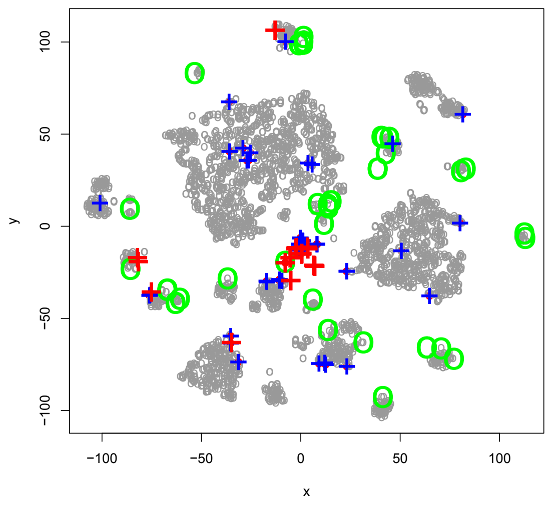
1. **P1** (AATP hinge loss): $\frac{C_A}{|\mathbf{H}_A|}(...) + \frac{1}{|\mathbf{H}_N|}(...)$
2. **P2** (Proximal factor (PF)): $\|\mathbf{w} - \mathbf{w}_p\|^2$
3. **P3** (Pairwise constraints between anomalies and nominals): $(\mathbf{z}_i - \mathbf{z}_j) \cdot \mathbf{w} + \xi_{ij} \geq 0$. P3 also includes the slack variable penalty term in the objective function.

The motivation behind each part has been explained in Section 3.3. In this section, we show the importance of each part to the performance of AAD. To help understand the importance of each part, we created six variants of AAD, as explained below, and compared their performance on the benchmark datasets.

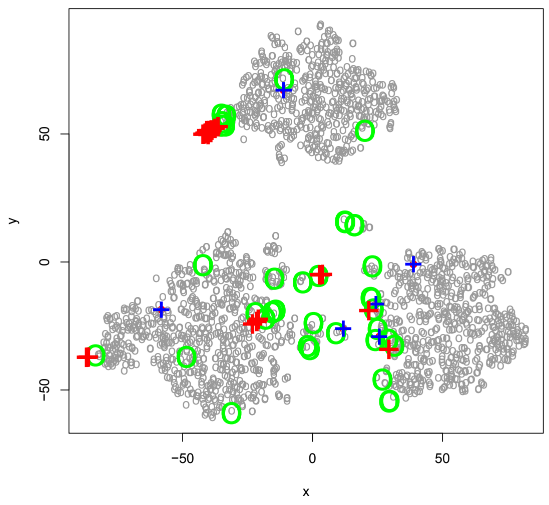
1. **V1** (*AAD*): This is the unmodified AAD algorithm.
2. **V2** (*AAD with no constraints*): Part P3 is dropped from the objective.
3. **V3** (*AAD with no AATP hinge loss*): Part P1 is dropped; in this case the AAD objective becomes similar to an SVM with the proximal factor on the weights.



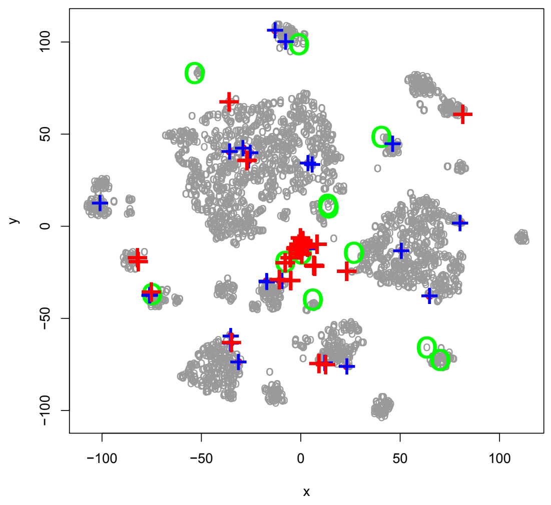
(a) Abalone Baseline



(b) ANN-Thyroid-1v3 Baseline

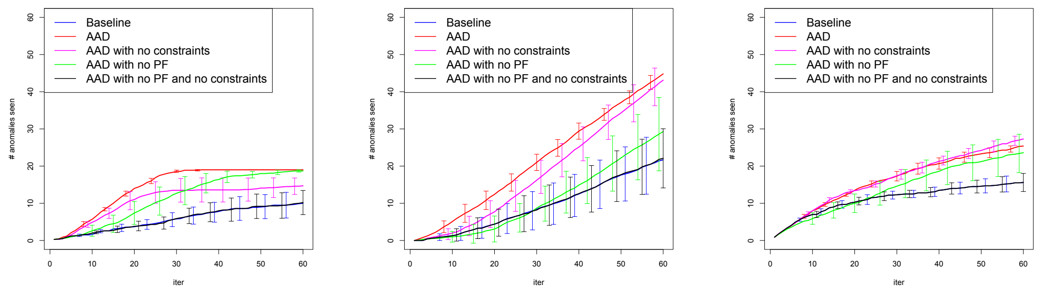


(c) Abalone AAD



(d) ANN-Thyroid-1v3 AAD

Figure 3.3: Low-dimensional visualization of *Abalone* and *ANN-Thyroid-1v3* using t-SNE. Plus signs are anomalies and circles are nominals. A red coloring indicates that a true anomaly point was queried. A green indicates a nominal point was queried. Grey circles correspond to unqueried nominals. To make unqueried anomalies stand out visually, we indicate them with blue plus signs.



(a) Covtype-sub (b) KDD-Cup-99-sub (c) Mammography-sub

Figure 3.4: Comparison of AAD variants. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

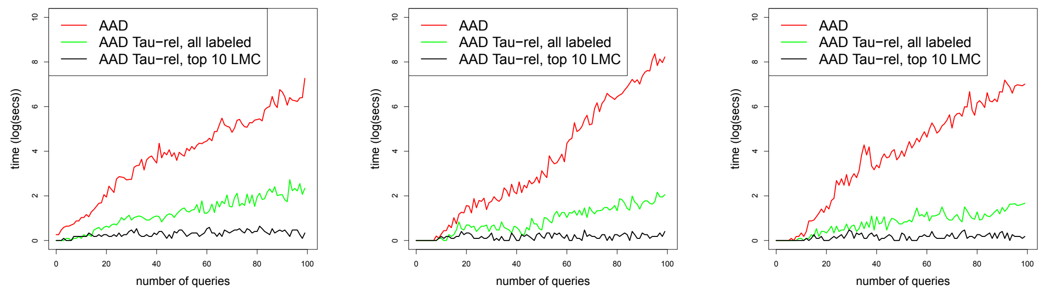
4. **V4** (*AAD with no proximal factor*): Part P2 is dropped from the objective.

5. **V5** (*AAD with no constraints and no proximal factor*): Parts P2 and P3 are dropped from the objective.

Figure 3.4 compares the variants for three representative datasets (*Covtype-sub*, *KDD-Cup-99-sub* and *Mammography-sub*). Since V3 performed poorly across most datasets, we have left it out to reduce clutter. We observe that in almost all cases, the various parts of the objective contribute positively towards the performance. Without any constraint or proximal factor, the algorithm performs almost identically to the baseline. We get a significant gain in performance when we add the proximal factor (P1) without adding any constraints. Adding constraints in addition to the proximal factor improves the performance somewhat; the constraints by themselves do not provide as much improvement as does the proximal factor.

3.4.4 Computational Complexity and Scaling Up

The computational complexity of AAD depends mainly on the number of pairwise constraints. The implementation of AAD in the *R* programming language employs an interior-point method to solve the constrained optimization problem in Equation 3.5. According to [35], this method has a worst-case time complexity of $O(m^{3.5}d^2)$ where m is the number of constraints and d is the number of dimensions of the input data.



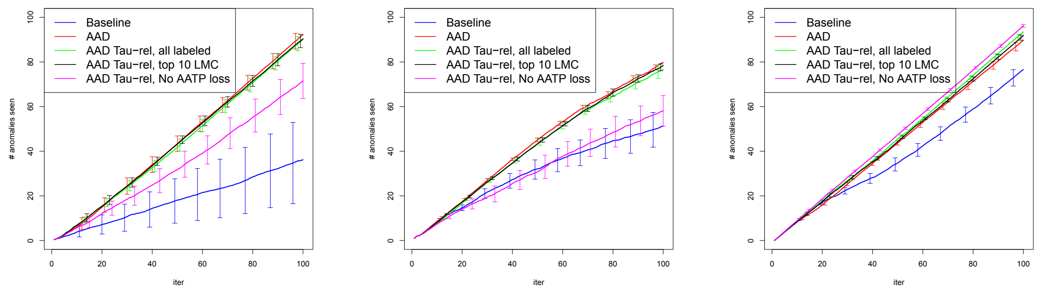
(a) Covtype (b) Mammography (c) Shuttle

Figure 3.5: The wall clock computation time at each iteration (averaged across 10 runs).

Furthermore, after the i-th feedback, the number of labeled instances is i; this implies $m = O(i^2)$ and the overall worst-case complexity in terms of the number of labeled examples is $O(i^7 d^2)$. In practice, the solution is found much faster than the worst-case complexity indicates. Furthermore, we expect the number of labeled examples to be low (less than about 100), since it involves human effort to label these examples.

The red lines in Figure 3.5 shows the average running time in seconds per number of queries for AAD over the *Covtype*, *Mammography*, and *Shuttle* datasets. Since at least one new pairwise constraint is added with every instance labeled by the expert, the computational cost increases with each consecutive iteration. Note that the factor 'd' in the complexity $(m^{3.5}d^2)$ depends on the number of Loda projections and not on the number of dimensions in the original feature space. The number of Loda projections is similar across the three datasets in Figure 3.5. The timing depends on the difficulty of finding the true anomalies. If the true anomalies are difficult to find, there will be fewer pairwise constraints added and the time taken by AAD will be lower. On the other hand, if the anomaly and nominal instances are being discovered in equal proportions, more pairwise constraints will be added, and the time will increase cubically as a function of the number of constraints.

The computation time for these three datasets on a current desktop is on the order of 2-3 minutes when the number of queries is near 60. This running time may be acceptable for anomaly detection domains that require a substantial amount of time for an expert to investigate if a potential anomaly is a false positive or a true anomaly. However, in some



(a) Covtype (b) Mammography (c) Shuttle

Figure 3.6: Comparison of AAD with variants where the constraints are relative to τ-th ranked instance. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

datasets, there might be a large number of true anomalies discovered, thus incurring a significant running time. To speed things up, we present several variants of AAD based on modifying the constraints added to the optimization. These variants are described below:

1. **T1** (*AAD*): Unmodified AAD.

2. **T2** (*AAD Tau-rel, all labeled*): The objective has parts P1 and P2 (defined in Section 3.4.3) and, instead of being based on each anomaly/nominal pair, only pairwise constraints involving the τ-th ranked instance $\mathbf{x}_\tau^{(t-1)}$ (the instance at τ-th rank could change with each feedback) are included. If $\mathbf{z}_\tau$ is the transformed vector for $\mathbf{x}_\tau^{(t-1)}$, then the constraints added are:

$$
\begin{aligned}
&(\mathbf{z}_i - \mathbf{z}_\tau) \cdot \mathbf{w} + \xi_{Ai} \geq 0 \quad \forall \mathbf{z}_i : \mathbf{z}_i \in \mathbf{H}_A \\
&(\mathbf{z}_\tau - \mathbf{z}_j) \cdot \mathbf{w} + \xi_{Nj} \geq 0 \quad \forall \mathbf{z}_j : \mathbf{z}_j \in \mathbf{H}_N, \\
&\xi_{Ai}, \xi_{Nj} \geq 0
\end{aligned}
$$

   The number of constraints added by this variant is linear in the number of labeled examples as opposed to quadratic (as in P3.)

3. **T3** (*AAD Tau-rel, top 10 largest margin constraints (LMC)*): This is similar to T2 above, but instead of adding constraints corresponding to all labeled instances, we

select only the top 10 *largest margin constraints* corresponding to instances from the labeled anomaly set $\mathbf{H}_A$ and the top 10 corresponding to instances from the labeled nominal set $\mathbf{H}_N$. For labeled anomalies, those that have the *largest margin constraint* are the anomalies which have the largest difference $(\mathbf{z}_\tau^{(t-1)} - \mathbf{z}_i) \cdot \mathbf{w}^{(t-1)}$. Similarly, for labeled nominals, those that have the *largest margin constraint* are the nominals which have the largest difference $(\mathbf{z}_j - \mathbf{z}_\tau^{(t-1)}) \cdot \mathbf{w}^{(t-1)}$. This variant results in a constant number of constraints in each iteration.

4. **T4** (*AAD Tau-rel, No AATP loss*): The objective has part P2 and constraints similar to T2 but without the AATP loss.

Figure 3.5 shows the running time for these variants and Figure 3.6 illustrates the performance of these variants on the *Covtype*, *Mammography*, and *Shuttle* datasets. A full set of results over all the datasets in our experiments is included in Appendix A.2. These results show that variants T2 and T3 exhibit anomaly detection performance similar to unmodified AAD (T1) even though they use a much smaller set of constraints. For the majority of datasets, T2 and T3 outperform T4, indicating that having both the AATP loss and the constraints improve performance. Variant T4, which removes the AATP loss but includes the constraints, tends to perform poorly though for a small number of datasets (e.g. *Shuttle*), the variants (T1-T4) have very similar results.

To understand why both the AATP loss and the constraints are needed, recall that we use $\hat{q}_\tau(\mathbf{w}^{(t-1)})$ to estimate the current quantile q_τ in order to make the optimization easier to compute. The AATP loss (Equation 3.3) helps because it encourages the new weights to cause the labeled anomalies to score higher than q_τ and labeled nominals to score lower. The important point here is that approximating q_τ with $\hat{q}_\tau(\mathbf{w}^{(t-1)})$ uses the previous weights $\mathbf{w}^{(t-1)}$. In contrast, the pairwise constraints use the *current* weights $\mathbf{w}$ in $\mathbf{w} \cdot \mathbf{z}_\tau^{(t-1)}$, which is the dot product of the current weights with the transformed representation of the feature vector $\mathbf{x}_\tau^{(t-1)}$ associated with $\hat{q}_\tau(\mathbf{w}^{(t-1)})$. Thus, the pairwise constraints unpack the transformed features of the previous τ-th quantile value and use more recent information in the form of the current weights.

However, without the AATP loss in T4, the weights produced by the constraints can change significantly from one iteration to another. These big changes occur because AAD operates in a transductive setting where there is a small amount of labeled data (which determines the constraints) and a large amount of unlabeled data (which determines the τ-th quantile). The T4 formulation, with just the pairwise constraints, has many possible assignment of values for the weights that satisfy the constraints. Adding the AATP loss further constrains the weights by loosely tying the current weights to the previous weights that were used to compute $\hat{q}_\tau(\mathbf{w}^{(t-1)})$. Since the initial q_τ value is computed using the uniform weights from Loda and is thus a fairly good estimate of q_τ, the loose tying of q_τ between iterations guides the optimization to a better region of the weight space. Thus, these two parts – the AATP loss and the pairwise constraints – are complementary and contribute differently towards satisfying the objective.

3.4.5 Comparison with Rare Category Detection

As mentioned in Section 3.2, our interactive anomaly discovery problem is closely related to the problem of *rare category detection* (RCD). Recall that RCD emphasizes diversity in the points discovered; the goal of RCD is to discover a representative from as many categories in the data as possible, given a fixed budget of queries to the analyst. In contrast, the goal of AAD is to find as many anomalies as possible within a fixed analyst budget.

A natural question to ask is whether an RCD algorithm can be used to solve the AAD problem. To answer this question, we evaluate an RCD algorithm using the anomaly discovery curve metric. However, it is difficult to compare RCD versus AAD directly, because RCD assumes that the analyst can identify an anomaly as belonging to one of K different categories or belonging to a previously-unseen category. On the other hand, the AAD problem has no notion of categories of anomalies as the analyst feedback is binary (*n*ominal or *a*nomaly). This form of binary feedback can be more suitable if the analyst finds it difficult to group anomalies into K coherent categories or if the actual anomalies do not only form small, dense clusters but also include anomalies that are scattered throughout the feature space.

In our experiments, we evaluate three settings for feedback with RCD:

1. **All anomalies in one class:** In this setting, the RCD feedback is binary, essentially meaning that there is a single *nominal* category and a single *anomaly* category which contains all the anomalies.

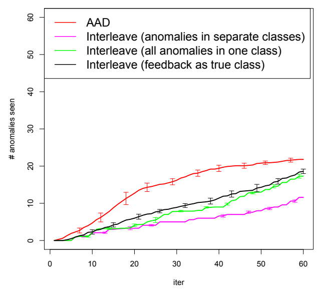
2. **Feedback as true class:** The anomaly class in our experiments consists of one

or more (minority) classes (e.g. in the *Shuttle* dataset, there are five classes that make up the anomaly class). In this setting, the simulated feedback provides the true class label, which correctly identifies the minority class from the original dataset that constitutes the anomaly class in our experiments. For datasets where the nominal class consists of more than one class from the original dataset, the simulated feedback also provides the true class label from the original dataset. This second setting is intended to correspond to the case when the analyst can correctly group the anomalies into coherent categories.

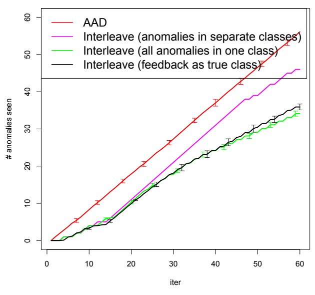
3. **Anomalies in separate classes:** In this final setting, there is a single *nominal* category and the analyst defines a new category for each anomaly that is discovered.

We compare AAD against the density-based **Interleave** [54] RCD algorithm. Interleave models the density using a Gaussian Mixture Model (GMM), and each instance is assigned to a single mixture component, i.e., one that 'owns' it to the highest degree. Each component corresponds to a separate 'category'. Interleave cycles through each component in a round robin fashion and adds the instances which are least owned by each component to a priority queue. These instances are presented to the user in a batch for feedback, ordered by their degree of ownership by a component. In addition to the components of the GMM, the Interleave algorithm also employs a uniform "background" mixture component. In our comparison with AAD, we follow the querying strategy for Interleave implemented by Pelleg et al. [54]. In this setup, we select a batch of two instances for querying: (1) the instance that is least owned across all mixture components and, (2) one instance from the background component.

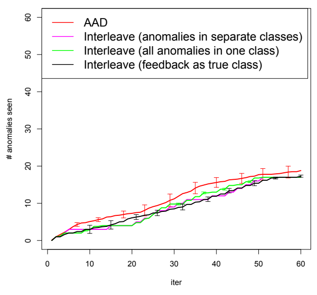
We present results on the *Abalone*, *Shuttle* and *Yeast* datasets in Figure 3.7. A more extensive set of results can be found in Appendix A.3. We picked these three datasets due to their varied characteristics, which can be seen in their visualizations in Appendix A.1. In the *Abalone* dataset, the nominal points form three distinct clusters. One nominal cluster has a small dense cluster of anomalies at its bottom left edge. Another cluster has anomalies along its leftmost edge. The third cluster has anomalies scattered throughout. The *Shuttle-sub* dataset has many small dense "islands" of nominal points; there is one dense anomaly cluster and a few anomalies scattered throughout the nominal clusters. Finally, the *Yeast* dataset is the most difficult of the three, as there are two dense clusters of anomalies but also a large number of singleton anomalies scattered throughout a very



(a) Abalone



(b) Shuttle-sub



(c) Yeast

Figure 3.7: Comparison of AAD with Rare Category Detection Algorithms. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

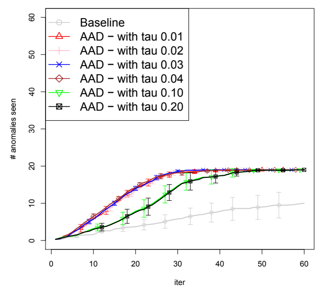
diffuse cloud of nominal points.

On these three very different datasets, AAD is able to find more true anomalies than the Interleave variants. These results indicate that when the data consist of both small, dense clusters of anomalies and also scattered singleton anomalies, AAD outperforms RCD at finding true anomalies. When scattered anomalies are present, the goal of increasing the diversity of anomalies discovered tends to throw off RCD as it queries instances in unpromising regions. In contrast, we noticed that AAD is able to down-weight regions dominated by nominal points and thus hone in on more promising regions of the feature space.

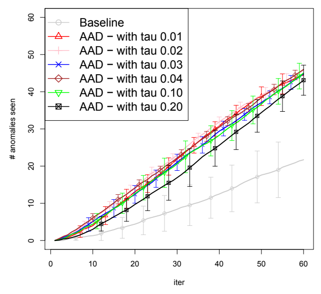
In Appendix A.3, we also include results from MALICE [30], which is another RCD algorithm. MALICE employs a local density differential. The trends are similar to those seen with Interleave as AAD discovers more true anomalies than MALICE.

3.4.6 Sensitivity Analysis

AAD relies on three parameters that need to be set before running the algorithm. These parameters are the quantile τ, the penalty factor C_A (for loss incurred when a labeled anomaly gets a score less than q_τ), and the penalty factor C_ξ (for the slack variables). In all our experiments we set $\tau = 0.03$, $C_A = 100$ and $C_\xi = 1000$. Figures 3.8, 3.9, and 3.10 show the performance of AAD on *Covtype-sub* and *KDDCup-99-sub* when one of these parameters is varied while the other two are kept fixed at their default values.

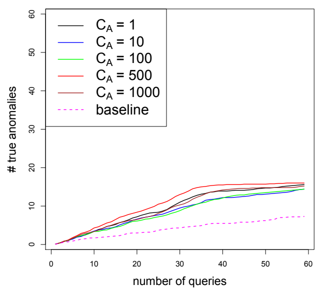


(a) Covtype-sub (τ)

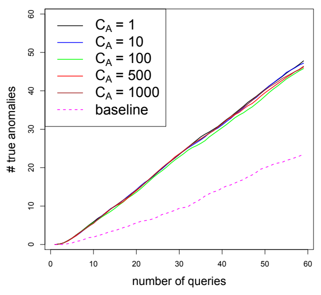


(b) KDD-Cup-99-sub (τ)

Figure 3.8: τ-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying τ while keeping parameters C_A and C_ξ fixed at their default values 100, and 1000 respectively.



(a) Covtype-sub (C_A)

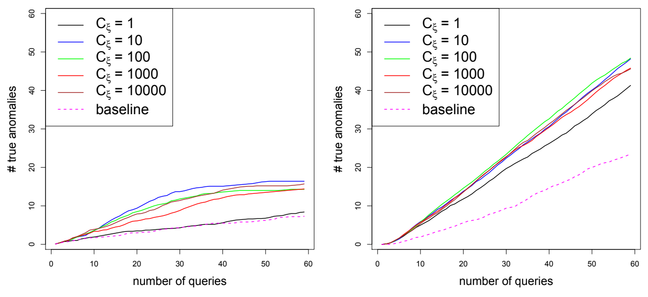


(b) KDD-Cup-99-sub (C_A)

Figure 3.9: C_A-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying C_A while keeping parameters τ and C_ξ fixed at their default values 0.03, and 1000 respectively.

The number of total queries in all these plots is set to be 60.

These plots show that the performance of AAD is generally not overly sensitive to τ (provided it is small enough) and C_A. The variance displayed by *Covtype-sub* for C_A and C_ξ reflects the overall higher variance of AAD on this dataset (see the large error bars on the red line in Figure 3.2d). The most sensitive parameter is C_ξ. Figures 3.10a and 3.10b suggest that C_ξ must be at least greater than 1 in order for the algorithm to perform well.



(a) Covtype-sub (C_ξ) (b) KDD-Cup-99-sub (C_ξ)

Figure 3.10: C_ξ-sensitivity analysis of Covtype-sub and KDD-Cup-99-sub. Performance of AAD while varying C_ξ while keeping parameters τ and C_A fixed at their default values 0.03, and 100 respectively.

3.5 Discussion

AAD is based on the concept that scores assigned to labeled anomalies should be in the top τ-quantile of all scores while scores assigned to nominals should be below that same quantile. This is simple and intuitive, and yet quite powerful as demonstrated in our experiments. It makes efficient use of analyst feedback to fine-tune the behavior of a reasonably good unsupervised anomaly detector.

In many anomaly detection settings, particularly in fraud and threat detection, the limiting resource is the time that the analyst has available to examine the candidate anomalies generated by an anomaly detection system. Global metrics such as AUC, APR, and Lift—although they are commonly reported in anomaly detection research papers—ignore this analyst cost. In this paper, we have instead assumed that the analyst can dedicate a fixed amount of effort (enough to examine k candidates), and the relevant metric is Precision at top k. Even more useful is to examine the precision as a function of k, which gives an "anytime" view of the process, as we have plotted in Figure 3.2.

A critical problem with data exploration for anomaly detection is the *cold start* problem – during the initial stages, few or even no true anomalies are often found. The scarcity of true anomalies causes problems when supervised learning algorithms are employed [64, 76] as they require an initial labeled set of anomalies and nominals. When few labeled anomalies have been discovered, there is a severe class imbalance, which causes the supervised techniques to be unstable and produce inaccurate results. In the extreme case when there are no true anomalies found, supervised techniques fail completely. In contrast, AAD can usefully incorporate expert feedback even when that feedback only consists of labels for nominal instances. This behavior is due to the fact that AAD relies on an internal ranking model and it attempts to push nominal instances below the top τ-th quantile.

There are three potential improvements to AAD that we will investigate in future work. First, we would like to solve for the values of q_τ and $\mathbf{w}$ simultaneously rather than approximating them by solving them separately. Second, our current setting only queries one instance at a time during each iteration. We intend to generalize this to querying k instances per iteration. This would be useful, for instance, when there is a group of k analysts available and working concurrently. In this minibatch setting, it is important to not only query informative instances but also to diversify the instances selected in the batch. Finally, we plan to investigate if there are more effective querying strategies aside from querying the top ranked instance.

For convenient data exploration, AAD might be combined with a low-dimensional visualization tool like t-SNE. The visualization would hint at the parts of data that contain true anomalies or that seem unusual but have not yet been queried. The expert could then use these hints to either query more instances similar to known anomalies or to diversify the query locations.

3.6 Conclusion

We introduced the AAD algorithm, which attempts to maximize the number of anomalies presented to an analyst during an interactive data exploration loop. In each iteration of this loop, the analyst can provide feedback that tells the algorithm whether the presented instance is an anomaly or a nominal point. AAD is based on a weighted ensemble of anomaly detectors in which each ensemble member reports an anomaly score based on a subspace of the original feature space. This ensemble approach is general enough to include a variety of approaches such as Loda [55] and Feature Bagging [40]. AAD works by maintaining an internal ranking of data instances based on the overall anomaly score produced by the ensemble. When AAD receives labeled instances, it introduces constraints to adjust the weights of its ensemble members such that false positives are pushed lower in its internal ranking model while other potential true anomalies increase in rank with each labeled instance provided by the expert. Even when no true anomalies have been identified, AAD is still capable of incorporating nominal-only feedback and improving its performance. In experiments comparing AAD to three other algorithms for incorporating experiment feedback, the results showed that AAD was usually the best (or nearly the best) method. Although AAD can be computationally expensive due to a large number of constraints being added to its optimization, we present efficient approximations that add a constant number of constraints and do not significantly degrade the quality of the solutions found.

Chapter 4: Extending Active Anomaly Discovery to other algorithms

4.1 Introduction

Inspired by the success of AAD with Loda, we sought ways to extend AAD to work with other anomaly detection algorithms. One hypothesis for the success of AAD with Loda is that AAD is performing a form of indirect feature selection by reweighting anomaly detectors that each involve only a small portion of the original features. This hypothesis suggests that we might extend AAD by applying it to an ensemble in which each anomaly detector is trained using only a subset of the features. This can be achieved by a technique known as "feature bagging" [40]. In Section 4.2, we show that AAD can improve the anomaly detection performance of an ensemble created by feature bagging. Next, in Section 4.3 we extend AAD to another algorithm, Isolation Forest [44].

4.2 AAD for Feature Bagging

Each member in a feature bagging ensemble focuses on a particular subspace of the original feature space. Algorithm 3 illustrates how such an ensemble is created. Here we have assumed that the original dataset has F features and the ensemble contains M members. The ensemble produces a set of anomaly scores $\mathcal{O}$ that can be uniformly weighted in the no feedback setting or can be reweighted via AAD.

For our experiments, we set $M = 50$. We evaluated three algorithms for $\mathcal{A}$: Loda, LOF, and Isolation Forest. Figure 4.1 shows that incorporating feedback over 60 iterations with AAD provides substantial gains in the majority of the datasets for all three anomaly detection algorithms. LOF becomes confounded when anomalies are clustered – as is the case in most of our datasets. Therefore, LOF's performance is quite poor in general. The gain in performance for *ANN-Thyroid-1v3* with LOF in Figure 4.1 is misleading: the actual number of true anomalies discovered by LOF without feedback was close to zero. As a result, when the small increase in detected anomalies (with

Algorithm 3 Generating Ensembles by Feature Bagging

```
**Input:** Raw data $\mathbf{D}$, number of members in ensemble $M$
Set $F$ = number of features in $\mathbf{D}$
Set $\mathbf{O} = \emptyset$
**for** $i = 1..M$ **do**
    Sample a number $k$ uniformly at random ($1 \leq k \leq \frac{F}{2}$)
    Sample $k$ features $\{f_1, ..., f_k\}$ at random (without replacement)
    Compute anomaly scores $\mathbf{o}_i$ using features $\{f_1, ..., f_k\}$ with an anomaly detection
    algorithm $\mathcal{A}$
    Set $\mathbf{O} = \mathbf{O} \cup \mathbf{o}_i$
**end for**
**return** $\mathbf{O}$
```

feedback) is divided by the average baseline value, it results in a high gain in percentage (> 650). *Shuttle-sub* has zero gain for Isolation Forest and Loda because it happens to be an easy dataset where the baseline algorithm itself identifies one anomaly (on average) in each step. Thus, for *Shuttle-sub*, there was no room for improvement by incorporating feedback.

4.3 Applying AAD to a single Isolation Forest

In this section, we apply AAD to incorporate user feedback into Isolation Forest (IF) [44]. We selected IF because it has been shown to be competitive with other anomaly detectors [44, 22]. The IF model comprises of a set of trees generated in a randomized manner as outlined in Algorithm 4. IF assigns an anomaly score to each leaf node as a function of the length of the path from the root node to the leaf. IF works on the idea that anomalous instances are few, and they are well-separated from clusters of nominal instances in the feature space. Because of this, anomalous instances very quickly reach leaf nodes through random partitioning. On the other hand, nominal instances, which form dense clusters, require many more splits to finally reach leaf nodes. Therefore, the length of the path traversed by an instance from the root node to the leaf is shorter (on average) for anomalous instances than it is for nominal instances.

Figure 4.2 illustrates how IF works on a synthetic dataset. The size of sub-sample (N) has been fixed to 256 in all the plots. A single tree is not very informative for this case (Figure 4.2c), and therefore the ranking of true anomalies is very noisy (sidebar in

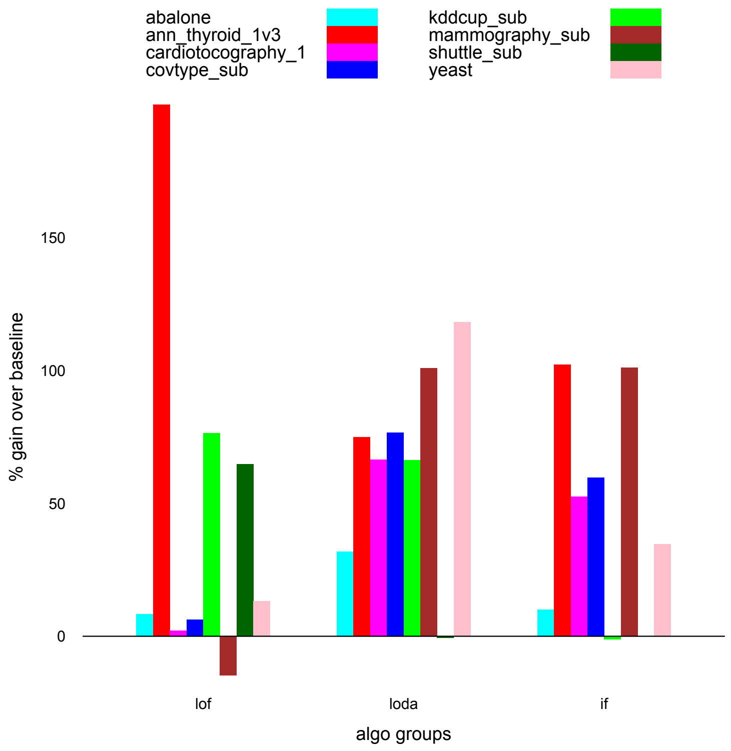


Figure 4.1: Results when ensemble members were generated using feature bagging. The *y-axis* shows the gain from incorporating feedback with AAD over 60 iterations. The difference is expressed as percentage change from baseline. A gain of 100% would imply that AAD discovered twice the number of anomalies as the baseline.

Figure 4.2c). However, as we increase the number of trees (Figure 4.2d), the algorithm's performance improves and it ranks more true anomalies near the top. This is because it now has a better estimate of the number of splits required to reach each region in the feature space. So far, the algorithm is purely unsupervised. Hence, it could potentially misjudge the relevance of outliers and report uninteresting instances as anomalies. We will now incorporate feedback into IF so that we can avoid this problem.

In order to make IF compatible with AAD, we will first treat the structure of the IF model as an *ensemble*. Let the IF model be comprised of t trees denoted by $\mathbf{T} = \{T_1, ..., T_t\}$. It is immediately obvious that IF is an ensemble of *trees*. However, this ensemble is unsuitable for AAD, because each tree is random, discrete, and global. In

Algorithm 4 Generating randomized trees in Isolation Forest

```
Input: $\mathbf{D}$, sub-sample size: $N$, numer of trees: $t$
$\mathbf{T} = \emptyset$
for $i = 1...t$ do
  Let $S_i$ = a sub-sample of $N$ instances from $\mathbf{D}$
  Build tree $T_i$ as follows, by starting with all instances in $S_i$ at the root node:
      Let $U \subseteq S_i$ be the set of instances at the current node
      if $|U| == 1$ then
        return
      else
        Let $f$ be a feature sampled at random
        Let $f_{min}$ = minimum value of $f$ across all instances in $U$
        Let $f_{max}$ = maximum value of $f$ across all instances in $U$
        Let $p_f$ = value selected uniformly at random from $[f_{min}, f_{max}]$
        Partition $U$ into two parts on the basis of $p_f$ and recurse on both partitions
      end if
  $\mathbf{T} = \mathbf{T} \cup T_i$
end for
```

such cases, it is hard to focus the feedback on specific parts of the data. Instead, we will treat IF as an ensemble of *nodes*.

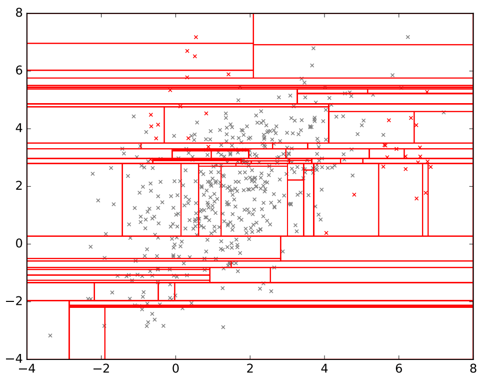
Let each node ν define a region r_ν with an associated score $s_\nu = 1/p_\nu$, where p_ν is the length of the path from the root to the node ν. Next, let $\mathbf{x}$ be an instance, and ν_x^k be the set of nodes that $\mathbf{x}$ visits on its way to a leaf in tree T_k (except the root). The anomaly score for $\mathbf{x}$ (without feedback) is then computed as[1]:

$$score(\mathbf{x}) = \sum_{T_k \in \mathbf{T}} \frac{1}{|\nu_x^k|} \sum_{\nu \in \nu_x^k} s_\nu \tag{4.1}$$

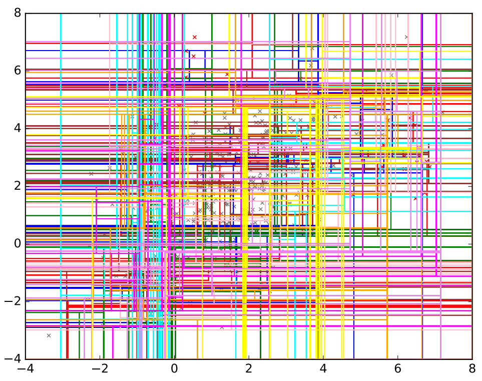
Figure 4.2 shows the anomaly score contours on synthetic data using Equation 4.1.

Now, each node ν in IF — each leaf node as well as each intermediate node (except the root) — can be considered to define a 'feature'. This feature is either zero for an

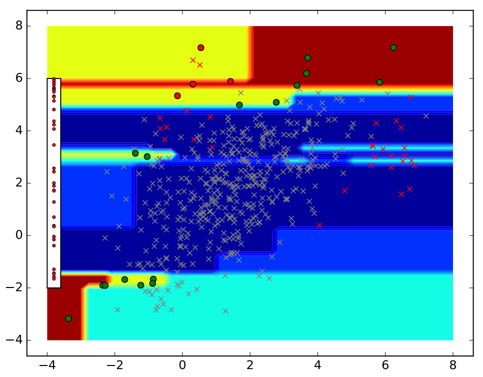
[1]Although the score in Equation 4.1 is different from the score computation in Liu et al. [44], both had similar anomaly detection performance in our experiments. The anomaly score for an instance $\mathbf{x}$ in Liu et al. [44] is computed as $2^{-\frac{E(h(\mathbf{x}))}{c(n)}}$, where $E(h(\mathbf{x}))$ is the expected path length, n is the number of training instances, and $c(n)$ is a normalization term. We compute the score with Equation 4.1 because it allows us to combine the scores linearly across all the nodes in the Isolation Forest.



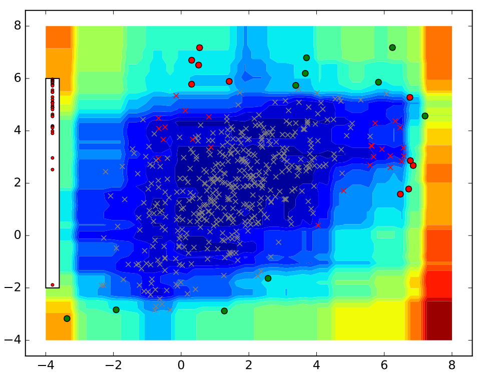
(a) 1 Tree



(b) 10 Trees



(c) Contours with 1 Tree



(d) Contours with 10 Trees

Figure 4.2: Random trees in Isolation Forest (IF) for synthetic data. The points in red are true anomalies; points in gray are true nominals. Figure 4.2a shows the leaf node regions for a single tree generated by random IF splits. Figure 4.2b shows the leaf node regions for 10 trees generated by IF; each tree has a different color. Figure 4.2c and Figure 4.2d show the contours of anomaly scores assigned by Isolation Forest with one and ten trees respectively. Deeper red means more anomalous; deeper blue means more nominal. Red circles are true positives among top 20 instances ranked by IF without feedback; green circles are false positives among the top 20. The left sidebar in Figure 4.2c and Figure 4.2d show the ranking of true anomalies (red dots). Ideally, true anomalies should be near the top on this bar.

instance (i.e., the instance does not belong to r_ν), or it is non-zero (the instance belongs to r_ν). Since the location of an instance in the feature space is bounded by more than one node (across all trees), the instance would have more than one non-zero feature. Let ν^k be set of all nodes (except root) in tree T_k. We define a function $f(\cdot)$ that transforms an instance $\mathbf{x}$ in the original feature space to an instance $\mathbf{z}$ in the space defined by the IF nodes as follows.

Let z_{f_ν} be the feature of $\mathbf{z}$ that corresponds to a node $\nu \in \bigcup_{T_k \in \mathbf{T}} \nu^k$. Then,

$$f(\mathbf{x}) = \mathbf{z} \quad \text{s.t.} \quad \forall T_k \in \mathbf{T}, \quad \forall \nu \in \nu_x^k : \quad z_{f_\nu} = \frac{1}{|\nu_x^k|} s_\nu \tag{4.2}$$

The dimensionality of $\mathbf{z}$ is $d' = |\bigcup_{T_k \in \mathbf{T}} \nu^k|$, which corresponds to the total number of nodes in the IF model. This could be very high, typically in tens of thousands. This is, however, not a problem in practice, because most features in $\mathbf{z}$ are zero. We now introduce the weight vector $\mathbf{w}$, which has dimensionality d'. When the weights are uniform, the anomaly score in Equation 4.1 can equivalently be written as:

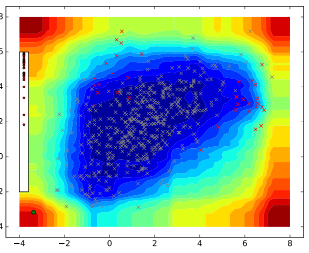
$$\begin{aligned} score(\mathbf{x}) &= f(\mathbf{x}) \cdot \mathbf{w} \\ &= \mathbf{z} \cdot \mathbf{w}. \end{aligned} \tag{4.3}$$

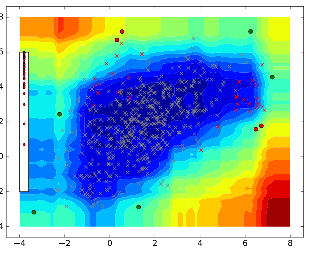
It is clear from Equation 4.3 that we can now use AAD to incorporate feedback by learning new weights $\mathbf{w}$. Figure 4.3 shows the result of incorporating feedback on synthetic data. As the algorithm receives feedback, it alters the contours of the anomaly scores and focuses on the more relevant parts of the feature space. In all experiments we have set the number of trees $t = 100$ and the sub-sample size $N = 256$.

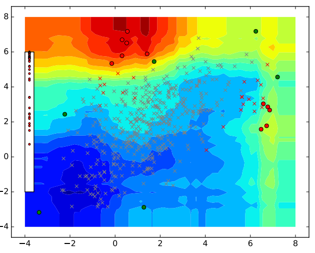
Figure 4.4 shows the result of incorporating feedback in IF on eight real-world datasets (Table 3.4). AAD does not hurt the performance of IF, and in most cases increases the number of true anomalies discovered.

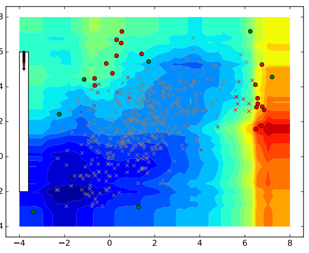
4.4 Discussion

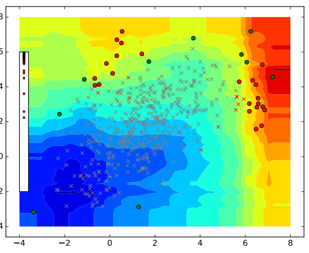
As demonstrated by our experiments, AAD works well when each ensemble member focuses on a particular subspace in a high-dimensional input feature space, as was shown

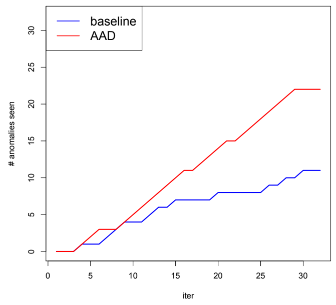

(a) Initial


(b) 8 Iterations


(c) 16 Iterations


(d) 24 Iterations


(e) 32 Iterations

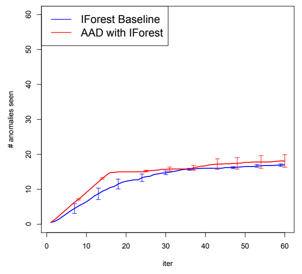


(f) Anomalies discovered

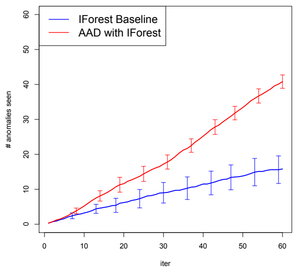
Figure 4.3: Incorporating feedback in Isolation Forest (IF) for synthetic data. Figures 4.3a – 4.3e show anomaly score contours in the same way as explained in Figure 4.2. The red curve in Figure 4.3f shows the number of true anomalies discovered when we incorporate feedback; the blue curve in Figure 4.3f shows the number of true anomalies discovered when no feedback was incorporated.

in Section 4.2 for both Loda ensemble and feature-bagging ensembles. An open question is whether AAD could be successfully applied to other types of ensemble anomaly detectors, such as an ensemble of heterogeneous detectors where the final score is a linear combination of scores from the individual members. This heterogeneous detector setup is common in real-world systems (e.g. [64, 60, 76, 27]).

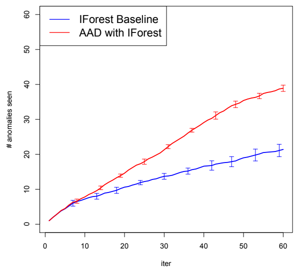
In Isolation Forest, the intermediate nodes generally describe larger regions in the feature space than the leaf nodes. Hence, the features corresponding to intermediate nodes are more 'global' than features corresponding to the leaf nodes. This was the primary reason for including the intermediate nodes in IF when adapting it for AAD; the original IF algorithm [44] only considers the leaf nodes. Our experiments show that



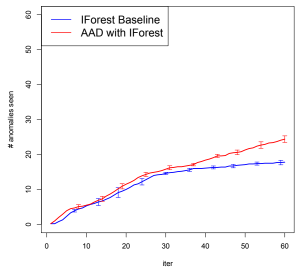
(a) Abalone



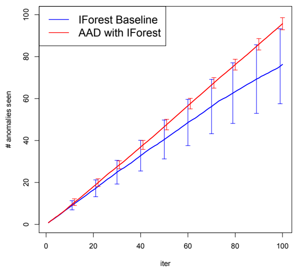
(b) ANN-Thyroid-1v3



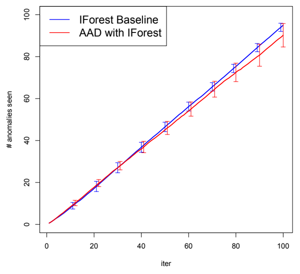
(c) Cardiotocography



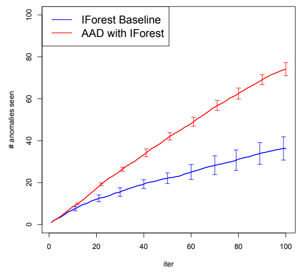
(d) Yeast



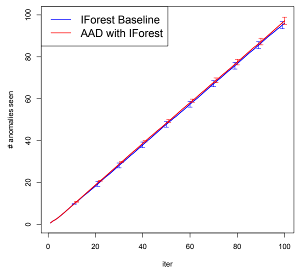
(e) Covtype



(f) KDD-Cup-99



(g) Mammography



(h) Shuttle

Figure 4.4: Incorporating feedback in Isolation Forest with AAD. Figures 4.4a–4.4d are the smaller datasets with 60 feedback iterations. Figures 4.4e–4.4h are the larger datasets with 100 feedback iterations. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

including global features results in better performance than using only the local features. We hypothesize that since global features are shared across more instances, they offer better opportunities for propagating information gathered through feedback. We note that all projection vectors in Loda are global as well. It is likely that this contributes favorably towards the performance we saw in Figure 3.2.

4.5 Conclusion

We showed that AAD can be used to improve the performance of feature bagging ensembles. In some cases it detects twice the number of anomalies as the baseline ensemble. Moreover, when AAD was adapted to be used with Isolation Forest, it discovered more anomalies with feedback than the baseline Isolation Forest with no feedback. These extensions encourage further research on adapting AAD to other algorithms.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

We first discussed in Chapter 1 the importance of feedback in machine learning algorithms. This was followed by an overview of various aspects related to the design of feedback.

Next, in Chapter 2 we developed a semi-supervised algorithm LWLR-SS-FL that incorporates feature labels provided by end users for text classification. It computes a pairwise similarity metric between instances over the underlying manifold structure using label diffusion. We showed that LWLR-SS-FL performs well when the quality of labeled features is high. It is also more robust to poor quality features when compared with other semi-supervised feature labeling algorithms.

In Chapter 3 we presented a novel algorithm called AAD for incorporating instance labels in anomaly detection. We showed that with expert feedback, we can use AAD to present more true anomalies to users than without expert feedback. Among the compared algorithms, AAD performed best on most benchmark datasets. We then suggested modifications to AAD which improve its computational efficiency and make it suitable for use in an interactive cycle by end users in real-time. We also applied AAD to an ensemble of anomaly detectors created using feature bagging and found that it is able to improve detection accuracy over baseline on most datasets. Finally, we showed that active learning algorithms for rare category detection perform poorly in comparison with active learning algorithms for anomaly detection.

Overall, we have shown that performance of deployed machine learning algorithms can be improved by end-users even when the original algorithm designers are no longer available. This facility will become more important as machine learning algorithms become pervasive in many different activities of our lives.

5.2 Future Work

5.2.1 Improvements to Feature Labeling

The current implementation of LWLR-SS and LWLR-SS-FL (Chapter 2) inverts a large matrix of pairwise distances in order to get the corresponding similarities. This has $O(n^3)$ complexity and as a result scales poorly as the number of instances (labeled and unlabeled) increases. Instead, we might first subsample the data, and compute pairwise similarities (in manifold space) by matrix inversion. Next, with the resulting smaller set of similarities, we would learn regression models for computing pairwise similarities locally at each sub-sampled point. These learned regression models would then compute similarities required for classification of out-of-sample points. This approach lowers the computation cost of matrix inversion and also extends the algorithm to a non-transductive setting. I intend to investigate this approach because with increasing volumes of data, scalability is critical in most applications.

An important cause of failure in semi-supervised learning with label diffusion is the presence of noisy data in between the clusters [84]. Labels 'leak' into incorrect clusters through these noisy data points. This is likely to have been the reason for poor performance on some of the datasets. We may be able to avoid this problem by incorporating regularization that enforces smoothness assumptions during the diffusion process [82]. This regularization was avoided in the current work because it requires tuning hyper-parameters and it makes training more expensive. Future work on LWLR-SS and its variants would need to incorporate regularization in conjunction with lowering the computational cost.

The basic idea behind LWLR-SS is to first reduce dimensionality of the data assuming there exists a latent manifold structure. Coordinates of instances in the new dimensions would then help establish better pairwise similarities. With this in mind, any dimensionality reduction technique might be used (e.g., t-SNE [74]) assuming that it preserves most information pertaining to relative distances between instances.

5.2.2 Extensions to Active Anomaly Discovery

5.2.2.1 Extension to Single Anomaly Detector

Siddiqui et. al. [69] present RPAD, a simple yet effective algorithm for detecting anomalies. RPAD assumes that there is an underlying set of patterns to which instances belong. Each pattern is associated with an anomaly score and one instance might belong to more than one pattern. The anomaly score for an instance is the average score across all patterns it belongs to. This setup is compatible with AAD. Since the patterns are usually mined in an unsupervised manner, their anomaly scores might not fit user expectations. AAD might allow users to re-weight the patterns such that the anomalous patterns get higher scores than nominal ones. Isolation Forest [44] is a special pattern space where the 'patterns' are defined by the volumes in feature space corresponding to leaf and intermediate nodes. The fraction of instances from a training set that belongs to a node is the anomaly score for that node under RPAD [69]. The extension to Isolation Forest in Section 4.3 was motivated by RPAD. Future work will improve upon this and also explore using AAD to improve performance of RPAD with other pattern spaces.

5.2.2.2 Developing new query strategies

Many query strategies commonly employed in active learning for classification settings [65, 23] are also applicable to anomaly detection. Strategies for discovering rare categories generally focus on determining cluster boundaries (using uncertainty sampling) whereas those for discovering anomalies query the most unlikely instances under the current model. Various strategies have been proposed [54, 25, 71] that try to query both the novel categories as well as the anomalies. In the current implementation of our algorithm, we only query the top ranked unseen instance in each iteration. Other query strategies could be explored in future, such as, query items most dissimilar to labeled nominals and items most similar to labeled anomalies, query by committee (QBC), query instances that have highest expected gradient length, query instances that lead to maximum variance reduction in the model, etc.

It has never been investigated whether a long-term sequential strategy might be more efficient for anomaly detection. I hypothesize that a sequential query strategy is useful when there are multiple types of anomalies. Some anomalies appear as standalone instances in the feature space, similar to noisy outliers. Others, which are generated by structured processes, tend to form clusters. Querying just the standalone outliers risks wasting too much budget on noise. At the same time, querying all members of an outlier cluster before moving on to others misses out on diversity. In this case we need to design a sequential query strategy that maximizes both the total number of detected anomalies as well as their distinct types within a restricted budget.

Bibliography

[1] Reuters-21578 text categorization test collection. `http://kdd.ics.uci.edu/databases/reuters21578`, 1997.

[2] UC Irvine Machine Learning Repository. `http://http://archive.ics.uci.edu/ml/`, 2007.

[3] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[4] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012.

[5] Josh Attenberg and Foster Provost. Guided feature labeling for budget-sensitive learning under extreme class imbalance. In *BL-ICML: Workshop on Budgeted Learning*, 2010.

[6] Josh Attenberg, Prem Melville, and Foster Provost. A unified approach to active dual supervision for labeling features and examples. In *In European conference on Machine learning and knowledge discovery in databases*, pages 40–55, 2010.

[7] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of highly variable functions for local kernel machines. In *In Advances in Neural Information Processing Systems 18*, page 2006. MIT Press, 2006.

[8] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT, pages 92–100, 1998.

[9] Stephen Boyd, Mehryar Mohri, Corinna Cortes, and Ana Radovanovic. Accuracy at the top. In *NIPS*, 2012.

[10] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104. ACM Press, 2000.

[11] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, pages 280–287, 2007.

[12] William S. Cleveland and Susan J. Devlin. Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting. *Journal of the American Statistical Association*, 83(403):596–610, September 1988.

[13] David A. Cohn. Neural network exploration using optimal experiment design. In *Advances in Neural Information Processing Systems 6*, pages 679–686, 1994.

[14] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI*, 1998.

[15] Shubhomoy Das, Travis Moore, Weng-Keen Wong, Simone Stumpf, Ian Oberst, Kevin McIntosh, and Margaret Burnett. End-user feature labeling: Supervised and semi-supervised approaches based on locally-weighted logistic regression. *Artificial Intelligence*, 204:56–74, 2013.

[16] Shubhomoy Das, Weng-Keen Wong, Thomas G. Dietterich, Alan Fern, and Andrew Emmott. Incorporating expert feedback into active anomaly discovery. In *Proceedings of the IEEE International Conference on Data Mining*, pages 853–858, 2016.

[17] Kan Deng, Scott E. Fahlman, and Christopher G. Atkeson Georgia. Omega; on-line memory-based general purpose system classifier. Technical report, 1998.

[18] Ethan Dereszynski and Thomas Dietterich. Probabilistic models for anomaly detection in remote sensor data streams. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 75–82, 2007.

[19] Pinar Donmez and Jaime G. Carbonell. Optimizing estimated loss reduction for active sampling in rank learning. In *Proceedings of the 25th international conference on Machine learning*, pages 248–255. ACM, 2008.

[20] Pinar Donmez and Jaime G. Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. In *Advances in Information Retrieval*, pages 78–89. Springer, 2009.

[21] Gregory Druck, Gideon Mann, and Andrew Mccallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602, 2008.

[22] Andrew Emmott, Shubhomoy Das, Thomas G. Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. *CoRR*, abs/1503.01158, 2015. URL `http://arxiv.org/abs/1503.01158`.

[23] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, 2012.

[24] Kuzman Ganchev, Jo ao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 11:2001–2049, August 2010.

[25] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection, 2013.

[26] Joao V. Graca, Lf Inesc-id, Kuzman Ganchev, Ben Taskar, Joo V. Graa, L F Inesc-id, Kuzman Ganchev, and Ben Taskar. Expectation maximization and posterior constraints. In *In Advances in NIPS*, pages 569–576, 2007.

[27] Martin Grill and Tomáš Pevný. Learning Combination of Anomaly Detectors for Security Domain. *Computer Networks*, 0:1–9, 2016.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[29] Jingrui He and Jaime Carbonell. Nearest-neighbor-based active learning for rare category detection. In *Advances in Neural Information Processing Systems 20*, pages 633–640. MIT Press, Cambridge, MA, 2008.

[30] Jingrui He and Jaime Carbonell. Rare class discovery based on active learning. In *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics*, 2008.

[31] Jingrui He, Yan Liu, and Richard Lawrence. Graph-based rare category detection. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, pages 833–838. 2008.

[32] Thorsten Joachims. Optimizing search engines using clickthrough data, 2002.

[33] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291 – 307, 2005.

[34] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. 9:189–198, 2015.

[35] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[36] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 392–403. Morgan Kaufmann, 1998.

[37] Roger Koenker. *Quantile Regression (Econometric Society Monographs)*. Cambridge University Press, 2005.

[38] Dima Kuzmin and Manfred K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 684–686, 2005.

[39] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[40] Aleksander Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on K nowledge Discovery in Data Mining*, pages 157–166, New York, NY, 2005. ACM.

[41] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal Of Machine Learning Research*, 5:361–397, 2004.

[42] Percy Liang, Michael I. Jordan, and Dan Klein. Learning from measurements in exponential families. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML, pages 641–648, 2009.

[43] Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Text classification by labeling words. In *AAAI*, volume 4, pages 425–430, 2004.

[44] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, page 413422, 2008.

[45] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[46] Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle Tseng. Active learning for ranking through expected loss optimization. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2010.

[47] Andrew McCallum. MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`, 2002.

[48] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, 1998.

[49] Andrew McCallum, Gideon Mann, and Gregory Druck. Generalized expectation criteria. *Computer science technical note, University of Massachusetts, Amherst, MA*, 2007.

[50] Prem Melville, Wojciech Gryc, Wolfson Bldg, Parks Rd, and Richard D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *In KDD*, pages 1275–1284, 2009.

[51] Nir Nissim, Aviad Cohen, Robert Moskovitch, Assaf Shabtai, Mattan Edry, Oren Bar-Ad, and Yuval Elovici. Alpd: Active learning framework for enhancing the detection of malicious pdf files. In *Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 91–98, 2014.

[52] Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[53] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, 2004.

[54] Dan Pelleg and Andrew W. Moore. Active learning for anomaly and rare-category detection. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2004.

[55] Tomáš Pevny. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 2015.

[56] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008.

[57] Hema Raghavan and James Allan. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proc SIGIR, ACM*, pages 79–86, 2007.

[58] Hema Raghavan, Omid Madani, Rosie Jones, and Isabelle Guyon. Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686, December 2006.

[59] Parisa Rashidi and Diane J. Cook. Ask me better questions: Active learning queries based on rule induction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 904–912. ACM, 2011.

[60] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles. *ACM Trans. Knowl. Discov. Data*, 10(4):42:1–42:33, 2016.

[61] Stphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS*, 2011.

[62] Dan Roth and Kevin Small. Interactive feature space construction using semantic information. In *CoNLL*, 2009.

[63] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233 – 242, 1999.

[64] Ted E. Senator, Henry G. Goldberg, Alex Memory, William T. Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, David A. Bader, Edmond Chow, Irfan Essa, Joshua Jones, Vinay Bettadapura, Duen Horng Chau, Oded Green, Oguz Kaya, Anita Zakrzewska, Erica Briscoe, Rudolph IV L. Mappus, Robert McColl, Lora Weiss, Thomas G. Dietterich, Alan Fern, Weng-Keen Wong, Shubhomoy Das, Andrew Emmott, Jed Irvine, Jay-Yoon Lee, Danai Koutra, Christos Faloutsos, Daniel Corkill, Lisa Friedland, Amanda Gentzel, and David Jensen. Detecting insider threats in a real corporate database of computer usage activity. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1393–1401. ACM, 2013.

[65] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[66] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1467–1478, 2011.

[67] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1069–1078, 2008.

[68] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 1289–1296, 1992.

[69] Md Amran Siddiqui, Alan Fern, Thomas Dietterich, and Shubhomoy Das. Finite sample complexity of rare pattern anomaly detection. In *Conference on Uncertainty in Artificial Intelligence*, UAI, 2016.

[70] Vikas Sindhwani, Prem Melville, and Richard D. Lawrence. Uncertainty sampling and transductive experimental design for active dual supervision. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*, page 120, 2009.

[71] Jack W. Stokes, John C. Platt, Joseph Kravis, and Michael Shilman. Aladin: Active learning of anomalies to detect intrusions. *Technique Report. Microsoft Network Security Redmond, WA*, 98052, 2008.

[72] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret M. Burnett, Thomas G. Dietterich, Erin Sullivan, and Jonathan L. Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *Int. J. Hum.-Comput. Stud.*, 67:639–662, 2009.

[73] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 999–1006, 2000.

[74] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne, 2008.

[75] Pavan Vatturi and Weng-Keen Wong. Category detection using hierarchical mean shift. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–856, 2009.

[76] Kalyan Veeramachaneni, Ignacio Arnaldo, Alfredo Cuesta-Infante, Vamsi Korrapati, Costas Bassias, and Ke Li. Ai2: Training a big data machine to defend. *IEEE International Conference on Big Data Security*, 2016.

[77] W-K Wong, Ian Oberst, Shubhomoy Das, Travis Moore, Simone Stumpf, Kevin Mcintosh, and Margaret Burnett. End-user feature labeling: A locally-weighted regression approach. In *In Proc. IUI, ACM*, pages 115–124, 2011.

[78] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. What's strange about recent events (wsare): An algorithm for the early detection of disease outbreaks. *J. Mach. Learn. Res.*, 6:1961–1998, 2005.

[79] Kevin S. Woods, Christopher C. Doss, Kevin W. Bowyer, Jeffrey L. Solka, Carey E. Priebe, and W. Philip Kegelmeyer. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(06):1417–1436, 1993.

[80] Xiaoyun Wu and Rohini Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 326–333, 2004.

[81] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Advances in Information Retrieval*, volume 4425 of *Lecture Notes in Computer Science*, chapter 24, pages 246–257. Springer Berlin / Heidelberg, 2007.

[82] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schlkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2004.

[83] Xiaojin Zhu. Semi-supervised learning literature survey, 2006.

[84] Xiaojin Zhu and Andrew B. Goldberg. Introduction to semi-supervised learning, 2009.

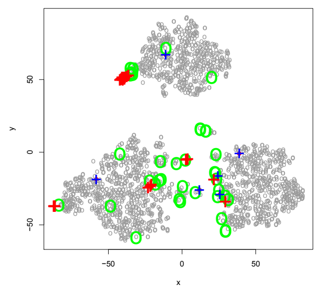
[85] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003.

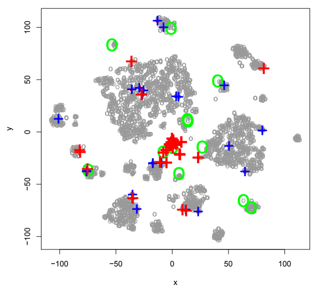
APPENDICES

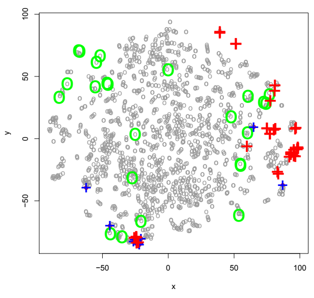
Appendix A: Appendix

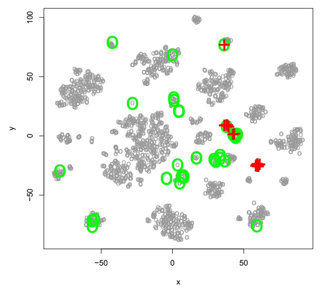
A.1 Dataset Visualization with t-SNE

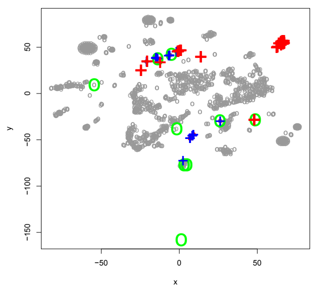
Figure A.1 provides two-dimensional visualizations of the eight small datasets used in our experiments. These visualizations are produced using the t-SNE algorithm [74]. The purpose of these visualizations is to give the reader an illustration of the locations of the anomalies relative to the nominal points and also to show which instances are queried by AAD.

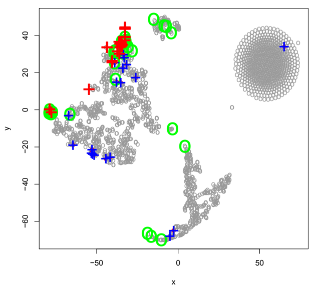

(a) Abalone

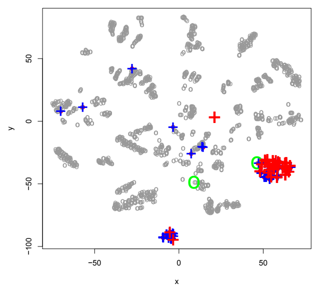

(b) ANN-Thyroid-1v3


(c) Cardiotocography


(d) Covtype-sub


(e) KDD-Cup-99-sub


(f) Mammography-sub


(g) Shuttle-sub

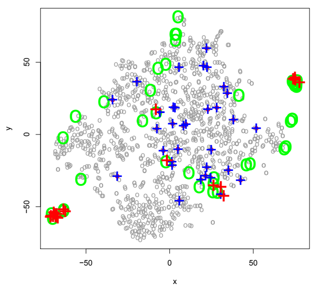

(h) Yeast

Figure A.1: Low-dimensional visualization of benchmark datasets using t-SNE. Plus signs are anomalies and circles are nominals. A red coloring indicates that the anomaly or nominal point was queried. Grey circles correspond to unqueried nominals. To make unqueried anomalies stand out visually, we indicate them with blue plus signs.

A.2 Performance with Constraints Relative to τ-th Ranked Instance

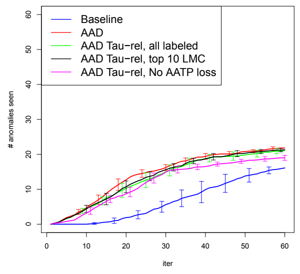
Figure A.2 illustrates the performance of the different approximations to the AAD algorithm on the twelve datasets used in our experiments; for completeness, we include results from the standard AAD algorithm and the LODA baseline. We refer to the three AAD approximations in Figure A.2 as Tau-rel approximations because they rank the anomalies and nominals relative to the τ-th ranked instance as described in Section 3.4.4. The three AAD approximations are:

1. *AAD Tau-rel, all labeled*: This approximation adds constraints that rank the labeled anomalies above the τ-th ranked instance and all the labeled nominals below the τ-th ranked instance. The number of constraints added by this approximation would be linear in the number of labeled examples as opposed to quadratic when all pairwise constraints are added.

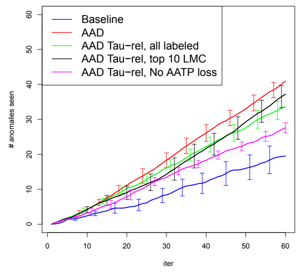
2. *AAD Tau-rel, top 10 largest margin constraints (LMC)*: Instead of adding all the constraints generated by labeled anomalies and nominals, this variant only adds the top 10 largest margin constraints for anomalies and the top 10 largest margin constraints for nominals. This variant results in a constant number of constraints in each iteration.

3. *AAD Tau-rel, No AATP loss*: The objective has part P2 and constraints similar to T2 but without the AATP loss.

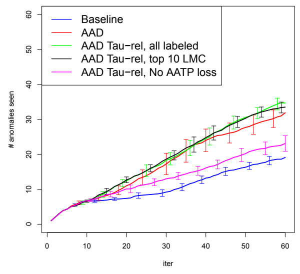
As Figure A.2 illustrates, there is very little degradation in performance, even with *AAD Tau-rel top 10 LMC*, but as we showed in Figure 3.5, there is a substantial gain in performance due to a reduced set of constraints.



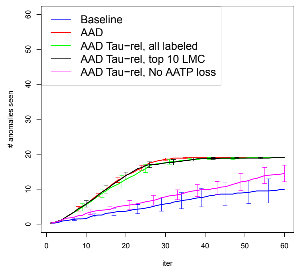
(a) Abalone



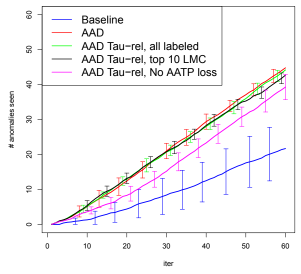
(b) ANN-Thyroid-1v3



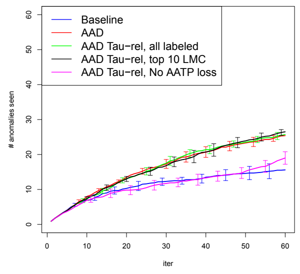
(c) Cardiotocography



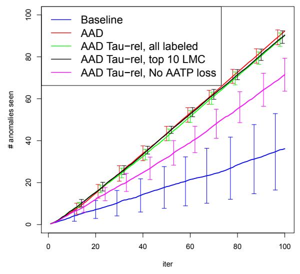
(d) Covtype-sub



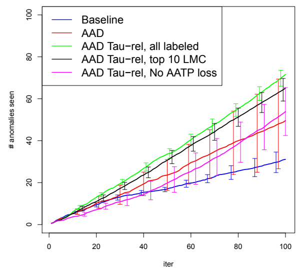
(e) KDD-Cup-99-sub



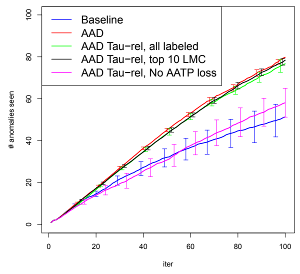
(f) Mammography-sub



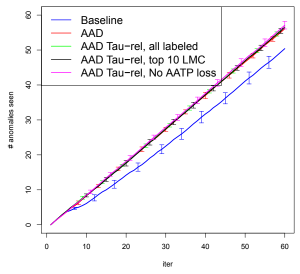
(g) Covtype



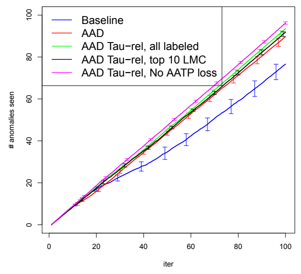
(h) KDD-Cup-99



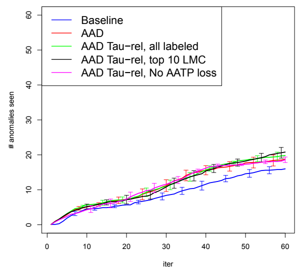
(i) Mammography



(j) Shuttle-sub



(k) Shuttle

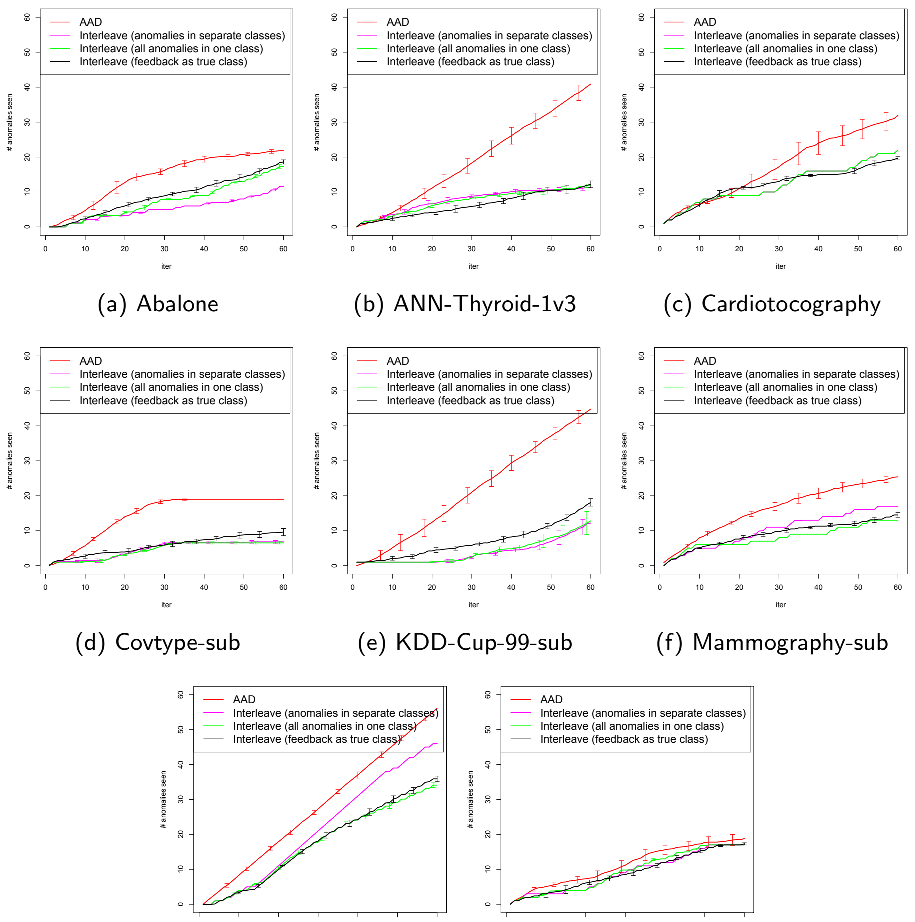


(l) Yeast

Figure A.2: Comparison of AAD with variants where the constraints are relative to τ-th ranked instance. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

A.3 Anomaly Detection with Rare Category Detection Algorithms

Figure A.3 shows the complete set of results of comparing Interleave to AAD on the eight small datasets. As in Section 3.4.5, we investigated the three instance labeling schemes for Interleave: “All anomalies in one class”, “Feedback as true class” and “Anomalies in separate classes”.



(a) Abalone (b) ANN-Thyroid-1v3 (c) Cardiotocography

(d) Covtype-sub (e) KDD-Cup-99-sub (f) Mammography-sub

(g) Shuttle-sub (h) Yeast

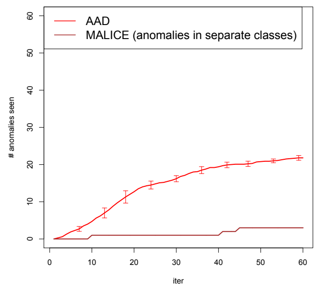
Figure A.3: Comparison of AAD with Interleave. The results were averaged over 10 runs. The error bars show the 95% confidence intervals.

We also include a comparison of AAD against *MALICE* [30], which is a distance-based RCD algorithm that is different from Interleave. *MALICE* computes the local density around each instance, assuming it belongs to class i, based on the minimum K_i-th nearest-neighbor distance where K_i is the estimated number of instances in class i. The anomaly score then corresponds to the change in local density. Instances with the highest anomaly scores are candidates for querying. We compared the performance of MALICE with that of its variant ALICE [30] and found that MALICE performs either better or at par with ALICE in most cases. As a result, we only present results for MALICE. We used $K = 5$ for all datasets.

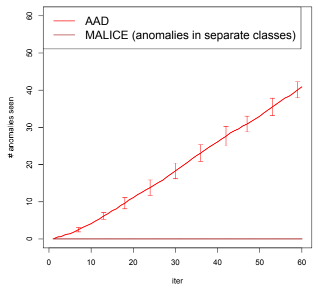
MALICE, by design, requires an estimate of the number of classes (and their respective proportions) in the dataset. It stops querying for new labels and exits as soon as one example from each class has been discovered. When the true number of rare classes is small, such as in the "anomalies as one class" and "feedback as true class" labeling schemes, *MALICE* often terminates before discovering a significant number of true anomalies. This behavior puts MALICE at a substantial disadvantage under these two labeling schemes as it produces an extremely short anomaly discovery curve. As a result, we only run MALICE in the setting where each anomaly is considered as a distinct class.

Figure A.4 compares AAD versus *MALICE* on the eight small datasets. *MALICE* performs very poorly on these datasets and AAD clearly outperforms it when it comes to presenting true anomalies to the analyst.

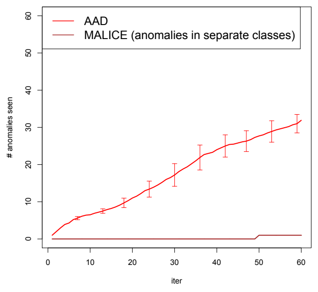
A.3.1 Each anomaly as a separate class



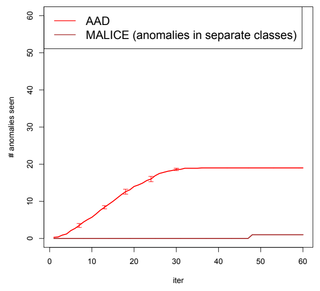
(a) Abalone



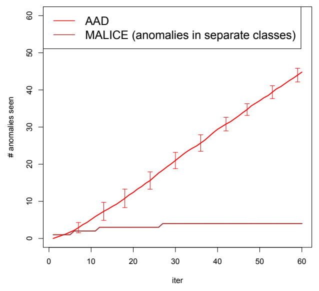
(b) ANN-Thyroid-1v3



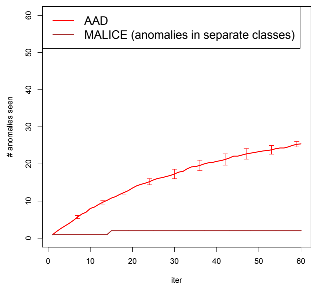
(c) Cardiotocography



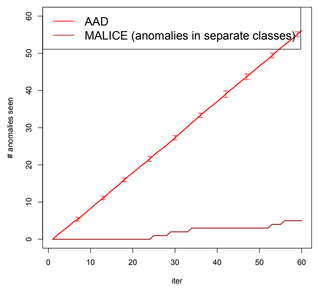
(d) Covtype-sub



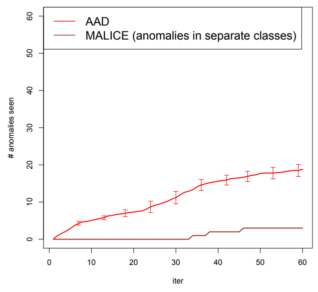
(e) KDD-Cup-99-sub



(f) Mammography-sub



(g) Shuttle-sub



(h) Yeast

Figure A.4: Comparison of AAD with MALICE. The results were averaged over 10 runs for AAD. The error bars show the 95% confidence intervals.