Active Learning for One-Class Classification

by

Vincent Barnabé-Lortie

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MCS degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Active learning is a common solution for reducing labeling costs and maximizing the impact of human labeling efforts in binary and multi-class classification settings. However, when we are faced with extreme levels of class imbalance, a situation in which it is not safe to assume that we have a representative sample of the minority class, it has been shown effective to replace the binary classifiers with a one-class classifiers. In such a setting, traditional active learning methods, and many previously proposed in the literature for one-class classifiers, prove to be inappropriate, as they rely on assumptions about the data that no longer stand.

In this thesis, we propose a novel approach to active learning designed for one-class classification. The proposed method does not rely on many of the inappropriate assumptions of its predecessors and leads to more robust classification performance. The gist of this method consists of labeling, in priority, the instances considered to fit the learned class the least by previous iterations of a one-class classification model.

Throughout the thesis, we provide evidence for the merits of our method, then deepen our understanding of these merits by exploring the properties of the method that allow it to outperform the alternatives.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Nathalie Japkowicz, for her continuous support of my research, for her patience, enthusiasm, knowledge and wisdom, for her guidance, and for the countless research opportunities she offered over the last two years.

I would also like to thank many members of the Ottawa-Carleton Institute for Computer Science: Drs. Viktor, Inkpen and Boyd, who taught the courses I took during the first year of my Master's; Dr. Samaan, who supervised the graduate seminar series and provided useful feedback on my seminar there; the attendees of the TAMALE seminar series who also provided feedback, guidance, and ideas for my research; and the graduate director, Dr. Amyot, who offered yet more valuable advice.

Next, I would like to thank two of my advisor's PhD students, Colin Bellinger and Shiven Sharma, for their example and mentorship, and for their help on the projects where we collaborated.

This thesis could not have existed without the motivation and gracious financial support of the Radiation Protection Bureau of Health Canada. I would like to thank, in particular, Drs. Rodney Berg and Kurt Ungar, of the Radiation Protection Bureau, for their collaboration and the research opportunities they offered.

I would like to express my sincere gratitude to the members of my thesis committee for taking time to review this thesis and to provide their insight.

Last but not least, I would like to thank my friends and family for their support. In particular, I am extremely grateful to my partner Jessica, as I could never have gotten through the last two years without her support.

Dedication

For Jessica.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is a common problem, in supervised machine learning, to have a larger sample of data from one class than another. In some cases, however, this imbalance is pushed to the extreme, and it is very difficult to obtain data from the minority class. In such a case, it may not be safe to expect the minority data to form a coherent body, or a representative sample. This can occur in many domains, but has been notably faced in defense and security applications, in problems such as oil spill detection [Kubat et al., 1998], fradulent behavior detection [Fawcett and Provost, 1997] and helicopter gearbox monitoring [Japkowicz et al., 1995].

Fortunately, many solutions have been proposed to deal with imbalanced data. Some of these do so by correcting the imbalance in the dataset; others, by adapting the classifier itself to work with imbalanced classes.

In this thesis, we focus on a third class of methods where data from only one class is used: one-class classification. For future reference, as proposed in [Bellinger et al., 2012], when applying a one-class classifier to a problem with imbalanced data, we will assume that the classifier is trained on the data of the *majority* class. Although it may at first

seem ridiculous to entirely forgo data from the minority class, there is, in fact, a growing body of evidence indicating that this may be a valuable strategy when the data from the minority class is extremely rare. [Japkowicz et al., 2000, Bellinger et al., 2012, Japkowicz et al., 1995, Japkowicz, 2001b, Kubat et al., 1998]

As in all supervised learning, human labeling remains a necessary step for the induction of a one-class classifier. Indeed, from the perspective of the one-class learner, class labels seem useless, as only data from one class is used. However, when developing the system, it is extremely important to use the class labels to ensure that only data of the appropriate class is fed to the one-class learner. In addition, data from both classes is necessary to evaluate a one-class classifier's ability to recognize not only what belongs to the class it was trained on, but also what does not.

The process of data labeling by humans can be costly. It is usually easy to *collect* data, a step which is often automated. Without human effort, however, the collected data remains unlabeled. In order to maximize the benefits gained from the time spent by human experts labeling data, one may decide to label only certain instances expected to be more useful than others, or to label the most important instances first. This strategy is known as *active learning* (or sometimes "query learning", "optimal experimental design") and has often been applied to binary and multi-class classification settings. However, the literature where active learning is applied to one-class classifiers, especially in the presence of extreme class imbalance, is quite sparse. Unfortunately, the techniques which claim to address the problem of active learning for one-class classification too often rely on assumptions which do not stand when data from the minority class is extremely rare.

In Chapter 2, we will describe in more detail this problem, as well as the current state of research into the class imbalance problem, one-class classification, active learning, and the intersections of these areas. In addition, we will provide an example of a domain, gamma-ray anomaly detection, where these different issues surface. Because our work on

this domain was, in large part, the motivation for this thesis, it will be often be used as an example throughout the following chapters.

The main contribution of this thesis is a strategy to perform active learning with one-class classifiers when faced with extreme class imbalance. In short, when minority data is extremely rare, we propose that, once a bedrock set of labeled instances has been established, unlabeled data should be prioritized so that instances that resemble the majority class the least are inspected first by human experts. The advantages of the proposed method are that *a*) it is conceptually simple; *b*) it does not make any assumptions concerning the presence or distribution of minority class data; *c*) it is not tied to any specific algorithm or implementation of one-class classification; and *d*) it naturally fits into the process under which one-class classification is deployed in many applications. The strategy we propose is described in detail in Chapter 3. The remainder of our contributions all relate to this proposed strategy and are described in Chapters 4, 5, and 6:

In Chapter 4, we provide statistically significant experimental evidence for the merits of our method. The results of our experiments suggest that there is definitely worth in carefully selecting data for labeling, and that prioritizing instances according to our strategy produces datasets that lead to higher classification performance in induced models. We also discuss an interesting disparity in how the datasets produced by our method affect the performance of different types of classifiers, and suggest that this disparity may be related to their parametric and non-parametric natures.

In Chapter 5, we experiment with the conditions under which our method can be used and the impact of these conditions on the improvements we can expect from it. In particular, we look at the impact of two factors on the performance gains that result from our active learning method: *a*) the size of the initial, bedrock set of labeled training instances, and *b*) the number of newly labeled instances (picked by active learning selection) added to this training set. We find that the relative performance gains are largest in two situations:

a) when we are forced to be very picky (i.e. we can only afford to label a few instances), and *b*) when the training set is already large, in which case our method appears to find the very rare unlabeled instances that can still contribute something to the training set. In addition, we investigate the advantages of performing active learning incrementally (i.e. re-evaluating the labeling priorities as handfuls of instances are added to the training set) relative to adding many instances in a batch. Here, we find no statistically significant difference. This result, although it brings no guarantee, is reassuring given the higher costs of incremental active learning.

Finally, in Chapter 6, we aim for a better understanding of the properties of our method and some of the indirect effects that may be behind its ability to identify useful instances. In particular, we find that, amongst the instances selected by our method for labeling, small, underrepresented subconcepts of the majority class are much more prevalent than they are in the dataset at large. Since the problem of imbalance in the distribution of a class' subconcepts, the *within-class imbalance problem*, can be a cause of poor classification performance just like imbalance *between* classes [Japkowicz, 2001a], it is possible that this property of our method partly explains why it works. In addition, we also find that, when using one-class support vector machines with our active learning strategy, the instances that are prioritized for labeling tend to be used as support vectors much more frequently than other instances. This suggests that our method tends to identify instances close to the decision boundary, and that it is these instances that allow us to refine this boundary, leading to better classification performance.

Chapter 2

Background and Related Work

In this chapter, we review the literature relevant to this research.

First, we will describe the problem of class imbalance and the solutions that are commonly applied to remedy it.

Second, we will look into one-class learners, with an emphasis on their use as a solution for the class imbalance problem, and, particularly, extreme levels of class imbalance.

Third, we will offer an overview of the existing techniques in the field of Active Learning, and bring special attention to the techniques that have been applied to problems with imbalanced data and situations of one-class classification.

Fourth, we will discuss the methods which can be used to evaluate and compare different techniques when faced with imbalanced data, one-class classifiers, or active learning systems.

Finally, we will provide an example of a domain where the issues described in the first 3 sections have surfaced: gamma-ray anomaly detection. This domain will be used throughout the thesis as an example.

2.1 The Class Imbalance Problem

Class imbalance is a very frequent problem in supervised learning. By class imbalance, we usually mean that there is data of more than one class, and that the data is not distributed uniformly across those classes. Hence, for future reference, when speaking about the distribution of classes, we will often use the term *prior probability* of a class, or, in shorthand, the *class prior*, to denote the probability that a randomly selected instance belongs to that class, assuming we are given no information about that instance. We will also commonly use the terms *majority class* and *minority class* to refer to the classes with, respectively, the largest and smallest priors.

It is very common, in many domains where machine learning is applied, that we are more interested in the less common classes. For example, in medical diagnoses, it is usually true that the majority of the population does not suffer from a given disease. It is important, however, to be able to find the few that do. Similarly, in gamma-ray spectrum anomaly detection, although spectral analyzers may produce samples very frequently, it is very rare that something particularly interesting, i.e. an anomaly, occurs. Once again, however, it is those very rare cases that we are most interested in, and although a system that always judges spectra to be normal may be extremely accurate in this case, it would miss all of the cases about which we care most.

Because the class of interest is so often the minority class, we will, in this thesis, as a convention, and without loss of generality, use *minority class* and *positive class* as synonyms, and *majority class* and *negative class* as synonyms. This will facilitate discussion about the errors made on each class by allowing us to use phrases such as "true positive" and "false negative" without ambiguity.

2.1.1 Levels of Imbalance

As the previous two examples show, the class imbalance for a domain can be very pronounced. This, however, is not always the case, and it is important to make the distinction between levels of imbalance. The imbalance between classes can be very mild (with class priors differing by a factor of no more than 2, for example), somewhat large (with, for example, a ratio of 1:10 between the class priors), large, or extreme (with, for example, a ratio of 1:10,000 between the class priors).

It is important to pay attention to the level of class imbalance, because the adverse effects of the class imbalance problem are much more noticeable when the imbalance is larger. In [Weiss, 2013], using experimental results from [Weiss and Provost, 2003], it is shown that although errors are already more frequent on examples of the minority class than on examples of the majority class at lower levels of imbalance, when the ratio between the priors is larger than 10:1, it is not unheard of to make errors on the minority class 20 times as frequently as on the majority class. In this thesis, the specific problem we tackle is associated with such extreme levels of class imbalance.

2.1.2 Between- and Within-class Imbalances

Although the problem of imbalance typically takes the form of a large difference between the priors of the classes, called between-class imbalance, there can also be imbalance between the sub-concepts that, together, form a class. This second type of imbalance is known as within-class imbalance [Japkowicz, 2001a]. Both imbalances, however, have in common that they involve rare cases. Rare cases are concepts, or sub-concepts, that are relatively rare, and represented by few instances. These rares cases tend to lead to small disjuncts [Jo and Japkowicz, 2004, Weiss, 1995], with which it is typical of classifiers to struggle. In a binary setting, an interesting possible consequence of having rare cases in the *majority* class is

that these would be more easily mistaken for the minority class, affecting our ability to distinguish the two and recognize the minority class without too many false positives.

This distinction between these two types of imbalance is of particular interest to us, because not only does the method we propose apply to situations with large between-class imbalances, but it also appears to diminish the harmful impact of within-class imbalance, and of small majority subconcepts. We explore this question in more detail in Chapter 6.

2.1.3 Common Solutions to the Class Imbalance Problem

There have been many proposed solutions to the class imbalance problem and some great literature reviews on the subject [He and Garcia, 2009]. We will only go over them quickly as this thesis focuses on one particular class of technique.

The first class of methods for handling class imbalance are sampling methods. These include random oversampling, which tends to lead to overfitting; random undersampling, which leads to a loss of potentially valuable information; informed undersampling techniques such as *EasyEnsemble* [Liu et al., 2009], *BalanceCascade* [Liu et al., 2009] and the KNN-based NearMiss family of methods [Mani and Zhang, 2003]; synthetic sampling techniques, the most common of which is SMOTE [Chawla et al., 2002]; adaptive synthetic sampling [Han et al., 2005, He et al., 2008], variations of synthetic sampling (usually SMOTE) which aim to deal with SMOTE's attributed overgeneralization [Wang and Japkowicz, 2004]; cluster-based oversampling [Jo and Japkowicz, 2004]; and many more. The rationale behind sampling methods for imbalanced data is that classifiers tend to perform better on balanced data [Weiss and Provost, 2001, Laurikkala, 2001, Estabrooks et al., 2004].

Second, there are cost-sensitive learning methods, where potentially different costs are assigned to the different types of classification errors a classifier makes on a dataset [Elkan,

2001, Maloof, 2003]. For example, in medical diagnoses, a false negative may be more costly than a false positive. There is empirical evidence that, at least in some imbalanced application domains, cost-sensitive learning can be superior to sampling methods [Zhou and Liu, 2006, McCarthy et al., 2005, Liu and Zhou, 2006]. Cost-sensitive learning can be implemented directly within the classifier: In a decision tree, one can move the decision threshold to account for the class imbalance [Maloof, 2003, Breiman et al., 1984]; in neural networks, Kukar, Kononenko, et al. (1998) [Kukar et al., 1998] offer four alternatives to cost-sensitive learning which apply cost-sensitive modifications to, respectively, the network's probabilistic estimate, its output, its learning rate, and its error minimization function; with support vector machines, there is the SMOTE with Different Costs (SDC) method [Akbani et al., 2004], many ensemble methods [Kang and Cho, 2006, Liu et al., 2006], and methods designed to use asymmetric misclassification costs [Wang and Japkowicz, 2010]. There is also the option to cost-sensitively weigh the instances of a dataset, a strategy often combined with ensemble methods. There have been many proposed variations of the AdaBoost algorithm [Freund et al., 1996, Freund and Schapire, 1997] designed to use unequal misclassification costs; for example, AdaC1, AdaC2 and AdaC3 proposed in [Sun et al., 2007] and AdaCost proposed in [Fan et al., 1999].

Third, one-class classifiers have previously been used to deal with extreme levels of class imbalance. Section 2.2 will cover these methods in detail.

Fourth, and finally, there are active learning methods designed to deal with class imbalance. Those methods are particularly important to this thesis and will be discussed in more detail in section 2.3.2.

2.2 One-Class Classification

One-class classification has been used in the past, particularly in domains where the level of imbalance is very pronounced, i.e. extreme. For example, it was used in typist recognition [Hempstalk et al., 2008], where the goal is to distinguish one typist from every other in the world, and where every other typist can be considered a (rare) subconcept. One-class classifiers were also used in document classification and retrieval [Manevitz and Yousef, 2002]. There is a parallel here between one-class classification and imbalanced or unknown data. As Manevitz and Yousef ([Manevitz and Yousef, 2002], p. 139) put it: "Consider, for example, trying to classify sites of "interest" to a web surfer where the only information available is the history of the user's activities. One can envisage identifying typical positive examples by such tracking, but it would be hard to identify representative negative examples." This is particularly interesting because in many domains where the classes are imbalanced, such as gamma ray anomaly detection, it is not necessarily possible to assemble a representative sample of one of the two classes (for example gamma ray anomalies).

Our research, described in this thesis, focuses on this particular solution to the class imbalance problem - one-class classifiers - for precisely the reason outlined above.

2.2.1 Implementations of One-Class Classification

There are many ways to implement one-class classification. One can, for instance, use density estimation to do so [Duda et al., 2000] by estimating the target class' distribution and setting a threshold on the probability density function, recognizing instances with high densities as belonging to the target class and instances with low densities as being of the *"other"* class, or not being part of the target class. A very simple implementation

of this uses the Mahalanobis distance of an instance to the target class as a measure of *"outlierness"* and sets a threshold on this distance to mark the class' boundary. These are examples of the myriad of ways one can turn what is generally known as methods of outlier detection into one-class classifiers; the common element being that a threshold is used on some measure of difference from the target class.

Alternatively, one can use neural networks to recognize instances belonging to a target class. In [Japkowicz, 2001b], the autoassociator, a variation of the multilayer perceptron where the network is trained to reconstruct, at the output layer, instances fed to it at the input layer, are used in that optic. The reasoning is that the autoassociator will learn to reconstruct instances belonging to the target class, on which it was trained, more effectively than instances not belonging to that class. Hence, the error the autoassociator makes when reconstructing an instance can be used as an indicator of whether or not that instance belongs to the target class.

A third common method of one-class classification is the one-class support vector machine [Schölkopf et al., 2001]. While in the regular binary SVM formulation, the goal is to find a maximum margin hyperplane that separates examples belonging to two classes, in one-class support vector machines, the origin of the kernel space is used as the second class, and the goal hyperplane must separate the target class from this origin.

2.2.2 Using One-Class Classifiers for Binary Classification

As demonstrated in the examples from this section's introduction, it is common that one-class classifiers, although they only learn to recognize one class, are used in binary classification settings. This makes sense when the second class' is defined in contrast to what the first is. For example, gamma ray anomalies are, by definition, what *is not* normal for a gamma ray spectrum. In those situations, the one-class classifier is trained to recognize

one class, and whether or not it recognizes new instances determines whether it belongs to the first, trained, class (if the instance is recognized) or to the second class (if the instance is not recognized).

There has been research into the conditions under which one-class classifiers can reliably be used for binary classification. First, the results of [Japkowicz, 2001b] show that, at least for neural networks (specifically multi-layer perceptrons and autoassociators), the one-class learner tends to outperform the binary learner most in domains that require either *weak* specialization from the learned/target class, or *strong* specialization from the other (non-learned) class.

Second, a result very relevant to this thesis comes from [Bellinger et al., 2012], where the relation between the level of imbalance of a dataset and the relative performance of one-class and binary classifiers is studied. Their results show that the performance of binary classifiers tends to decrease as the imbalance gets more pronounced, supposedly because the binary classifier fails to accurately model the minority class. Conversely, one-class classifiers seem unphased by extreme levels of imbalance, giving them the upper hand over binary classifiers in highly imbalanced situations.

2.3 Active Learning

We now move onto the topic of active learning, on which the proposed method is based. Throughout this section, we will describe the general idea behind active learning, then describe the many variants of active learning that have been proposed throughout the literature. As we will show, these techniques have some flaws that may make them inappropriate when using one-class classifiers, and when faced with extreme levels of class imbalance. This limitation of the existing method is what we aim to address with this

thesis.

An excellent review of the active learning literature has already been published by Burr Settles [Settles, 2009]. In this section, we will cover the relevant material from that literature review, as well as additional publications on active learning from the past few years that deal in more detail with the application of active learning to domains with imbalanced data and to methods of outlier detection and one-class classification.

At the core of supervised learning is the need for labeled data. The labels, in general, are provided by a human expert, and the labeling process can be expensive. The labeling cost, for example, could be the hourly wage of a physicist in charge of labeling gamma ray spectra as "normal" and "anomalous". Because of this, there is usually a tradeoff between the performance of the machine learning system and minimizing the cost of labeling. The goal of active learning, then, is to label fewer instances without losing too much information; to make the most out of the available human labeling effort. We illustrate the process of deploying an active learning system in Figure 2.1: First, a model is trained from an initial set of labeled training data. This model, along with a selection criterion, is used to select instances from a large pool of unlabeled data. The selected instances are then sent off to be labeled by a human expert, sometimes called an *oracle*, before being added to the training set. The process can then be repeated if more data is desired.

The selection criterion used, which is sometimes called the *query strategy*, is the component of the framework that is usually subject to change from one active learning technique to another, and there has been a lot of research into criteria that lead to more beneficial instance selections. The benefits of a selection criterion are judged based on the performance of machine learning models that result from it. The remainder of this section will go over various selection criteria that have been proposed, some of which were designed with imbalanced data or one-class classification in mind.

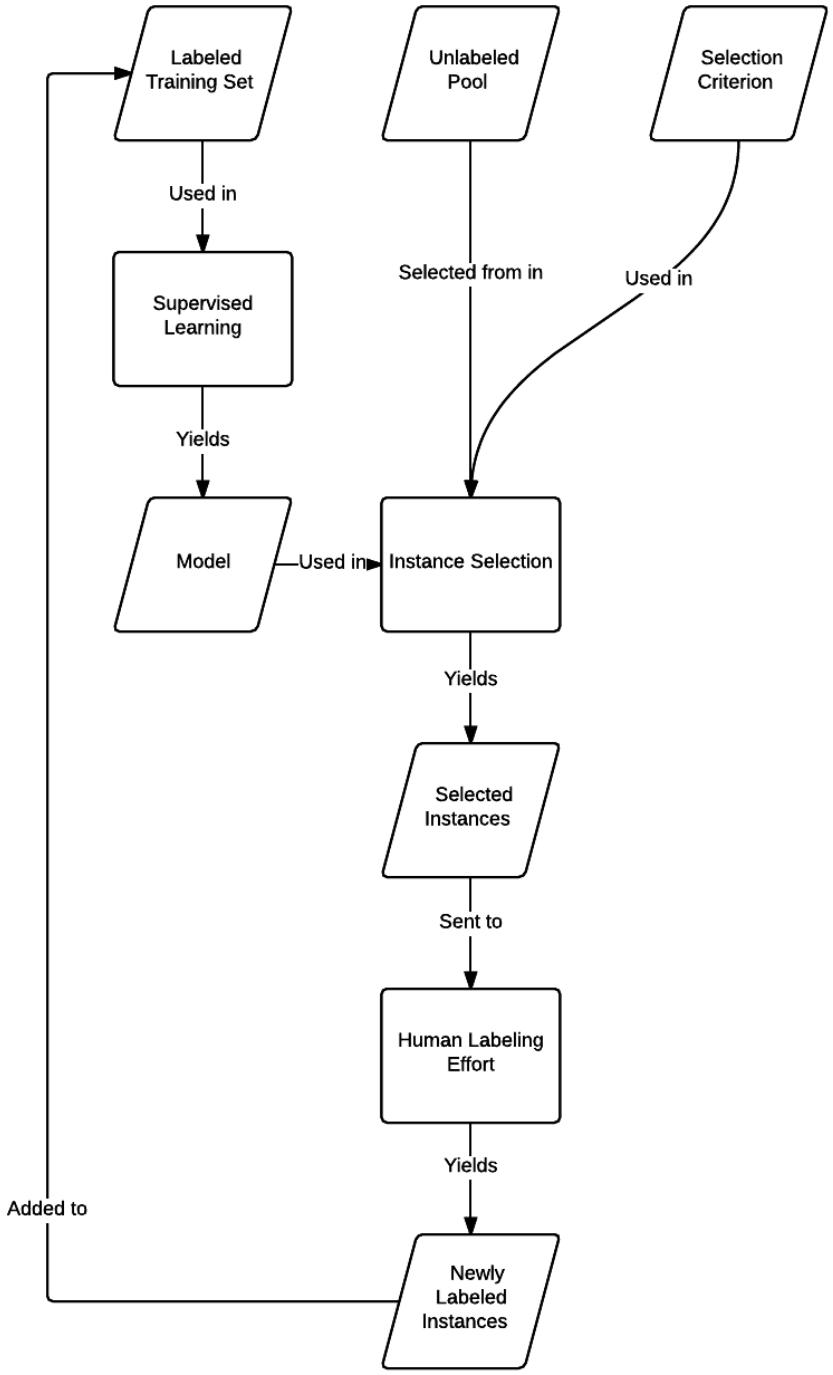


Figure 2.1: Overview of the active learning process

2.3.1 Common Selection Criteria

Uncertainty Sampling

The first selection criterion we will discuss, and perhaps the most common, is uncertainty sampling [Lewis and Gale, 1994]. The name is self-explanatory: This technique consists of selecting instances on which the initial model is least certain when it comes to classification. For example, if the model used outputs posterior probabilities for each class, then we would, under uncertainty sampling, select instances where the posterior probability is close to 1/2 in the binary setting [Lewis and Gale, 1994, Lewis and Catlett, 1994]. In the case of a multiclass domain, we would select the instance for which we are the *least confident*, as [Settles, 2009] puts it, i.e. the instance for which the highest class posterior probability is the lowest.

There are many variations on uncertainty sampling. For instance, margin sampling [Scheffer et al., 2001] takes into account the highest and lowest posterior class probabilities for an instance, and the difference between the two (the margin). The margin sampling criterion selects instances for labeling which have the smallest margin, i.e. where there is the least difference in probability between the most and least probable classes. Another variation uses entropy [Shannon, 1948], calculated over the class posterior probabilities, as a measure of uncertainty.

Although uncertainty sampling exhibits a great synergy with probabilistic classifiers, it can also be used with the non-probabilistic. For example, it has been used with decision trees in [Lewis and Catlett, 1994], with nearest-neighbor methods in [Fujii et al., 1998, Lindenbaum et al., 2004], and with support vector machines in [Tong and Koller, 2002].

The method we propose is *conceptually* similar to uncertainty sampling. The difference,

however, is that we choose, for reasons detailed in Chapter 3, to label first instances about which we are *most certain*, whereas uncertainty sampling does exactly the opposite. The rationale behind this choice rests on properties of one-class classifiers, and of the highly imbalanced domains where we deploy them.

Query-By-Committee

Next, we have the Query-By-Committee selection criterion, proposed in [Seung et al., 1992]. In the Query-By-Committee (QBC) framework, a committee is formed of models all trained on the labeled set, but representing competing hypotheses. The instances on which members of the committee most disagree are then chosen for labeling, settling the disagreements. Query-By-Committee is based on the concept of version spaces [Mitchell, 1982]: The members of the committee should represent different regions of the version space, and labeling the instances on which they disagree should allow us to restrict the size of the version space we must search.

An important issue when using QBC is the question of how the different members of the committee are generated. In the original paper by [Seung et al., 1992], two hypotheses were sampled at random. Since, more advanced methods have been proposed to generate committees. For instance, in [Mamitsuka, 1998], two methods based on ensemble methods are proposed: query-by-bagging, and query-by-boosting.

Once the committee has been generated, the next issue is to measure disagreement between members of the committee. In [Dagan and Engelson, 1995], Dagan and Engelson propose *vote entropy*, yielding a variant of QBC reminiscent of entropy-based uncertainty sampling. Alternatively, in [Nigam and McCallum, 1998], a measure of disagreement is proposed that is based on the average Kullback-Leibler divergence [Kullback and Leibler, 1951] between members of the committee. The Kullback-Leibler divergence is a measure of

the difference between two probability distributions.

There are many more variants on the Query-By-Committee framework, and many more methods of measuring disagreement between the committee members. The literature review by Burr Settles [Settles, 2009] describes them in more detail.

Expected Model Change

Another interesting criterion to be used within the active learning framework is *expected model change*. Following this criterion based on decision-theory, one chooses the instance that, *if it was labeled and used in the training set*, would cause the greatest (expected) change to our current model. This technique requires two pieces of information:

- Class posterior probabilities for each unlabeled instance
- A measurement of the change each instance would cause to the existing model given each possible class label

We know of only one method that implements expected model change: the expected gradient length method (EGL) [Settles et al., 008b]. The EGL method only applies to classifiers with gradient-based training, and the length of the gradient for an instance and class label is used as the measure of model change.

Although expected model change has been empirically shown to be effective, it can be very computationally expensive because of the cost of calculating model change. [Settles et al., 008b, Settles and Craven, 2008].

Once again, there is a conceptual similarity between expected model change and the method we propose. By selecting instances which the current model, trained on the majority class, judges to be most likely to belong to the minority class, we expect to find many

instances which would have surfaced as false positives. If, indeed, these instances turn out to belong to the majority class, then the model will *need to change* to accomodate this new information. Hence, we are selecting instances which we expect to have the largest impact on the existing model, and to cause in it the largest change.

Expected Error Reduction

Yet another decision-theoretic selection criterion, expected error reduction estimates how much generalization error is expected to be reduced by the labeling of an instance. This generalization error can be the 0/1-loss, for example, or the expected log-loss. Expected error reduction was first proposed in [Roy and McCallum, 2001] for text classification using naïve Bayes, but it has since been used with Gaussian random fields [Zhu et al., 2003], logistic regression [Guo and Greiner, 2007], and support vector machines [Moskovitch et al., 2007].

Although the initial formulation used simple loss functions, the general approach of expected error reduction can fortunately be adapted to other performance metrics, such as, for example, the area under the ROC curve.

Unfortunately, expected error reduction is extremely computationally expensive, making it impractical in most scenarios. [Settles, 2009]

Variance Reduction

Inspired by expected error reduction, variance reduction decomposes the expected error into three components (noise, model bias and model variance) and attempts to minimize only the third of these terms. [Settles, 2009]

Although this technique is less computationally expensive than expected error reduction,

it is still more expensive than strategies like uncertainty sampling, and it is also much more restrictive than expected error reduction: While an advantage of expected error reduction was that it could be adapted to other performance metrics such as the AUROC, variance reduction is formulated based on expected loss.

Density-Weighted Methods

The last of the general purpose active learning selection criteria we will look at are density-weighted methods, which aim to select instances that are representative of other instances in the distribution. In the information density framework proposed in [Settles and Craven, 2008], an instance's informativeness, as calculated by some other measure such as uncertainty or QBC (both described above), is multiplied by the instance's average similarity to instances in the labeled set. The rationale is that an outlier, no matter how informative it is, will have a smaller impact on the majority of instances than an instance in a denser area of the domain.

2.3.2 Active Learning for Imbalanced Data

The relevance of active learning, and its ability to identify informative instances rapidly, is particularly evident when faced with heavily imbalanced data. If we were to label data at random, the number of minority class examples we would find could be very small. The question, then, is on how we should select instances when the goal is to obtain a training set that is more balanced than the domain at large.

In [Tomanek and Hahn, 2009], Tomanek and Hahn incorporate class costs to a Query-By-Committee approach. Their method, however, in order to find more instances of the minority class, relies on the assumption that the members of the committee are already

somewhat apt at identifying instances belonging to the minority class, which, in some cases, would require that the problem be solved already.

In [Bloodgood and Vijay-Shanker, 2009], a similar approach is used, but a small random sample is used to estimate the class priors. If the data is very imbalanced, however, it may be difficult to obtain a reliable class prior estimate. In fact, in some cases, the random sample may contain no minority instances at all!

In [Ertekin et al., 2007], Ertekin et al. propose an active learning method for highly imbalanced domains. Although it may be useful if used as a form of undersampling, this technique is not designed for use with unlabeled data, as active learning methods usually do, and does not lead to an expanded training set. Instead, it works towards reducing the size of an available large labeled pool in order to balance it, a technique reminiscent of undersampling, albeit in an informed manner.

Unfortunately, as shown in [Attenberg and Provost, 2010], when the class imbalance in a domain becomes very pronounced, the majority of standard active learning strategies fail to identify minority class examples. Considering that many active learning strategies take density into account [Settles and Craven, 2008, Nigam and McCallum, 1998, Fujii et al., 1998], and try to select instances in dense areas of the dataspace, this is not a very surprising result: If the data is extremely imbalanced, the minority class will be much more sparse than the majority class.

What Attenberg and Provost propose, in [Attenberg and Provost, 2010], is to spend the effort of human experts actively searching for instances of the minority class rather than sending them instances to be labeled. Although their experimental results are very promising, and the reasoning behind their method is sound, it is only applicable to domains where searching is possible, such as web documents. In other domains, such as gamma ray anomaly detection, for instance, their method is unfortunately unusable.

It is also important to note that although the methods mentioned in this section were designed for imbalanced data, they are not appropriate when the classifier used is a one-class learner: Most of these methods have focused on identifying minority instances, but a one-class classifier trained on the majority class would have no use for those instances. Section 2.3.3 will look at active learning methods appropriate for this particular scenario. The method we propose also falls into this category.

2.3.3 Active Learning for Outlier Detection and One-Class Classification

To conclude this section on active learning, we review methods which claim to address the problem of applying active learning to systems of outlier detection and one-class classification. These methods are of particular interest because they are the obvious alternatives to the method proposed in this thesis.

First, in [Abe et al., 2006], the outlier detection problem is reduced to a binary classification problem by artificially generating outliers to form the second class. Then, active learning is applied to the resulting dataset. Our work differs from theirs in that we do not attempt to learn the minority class, or assume that we have any information as to its distribution. In addition, we take the opposite direction: Instead of turning a one-class problem into a binary problem, as they do, we perform binary classification by using a one-class learner. The rationale behind this choice is that, once again, the data we have for the minority class is too sparse, and cannot be assumed to be representative of the class at large.

Second, in [Görnitz et al., 2009], an approach based on support vector data descriptions [Tax and Duin, 2004] is proposed: It selects instances close to the surface of the SVDD sphere as well as instances in unknown areas of the feature space. Although somewhat

similar to our method, this method is limited to a single one-class classifier algorithm, the SVDD, whereas ours can be used with a wide variety of one-class classifiers.

Third, in [He et al., 2006], instances are selected based on their relevance to the learned class and the likelihood that they belong to that class. What we propose is, conceptually, the opposite, as we believe that instances that are very similar to what we already have in the training set may lack informativeness.

Fourth, in [Ghasemi et al., 2011b], Ghasemi et al. use margin sampling, a method we described in section 2.3.1. However, their method requires likelihood estimates for the minority class and calculates these estimates by relying on the assumption that the pool of unlabeled data contains *some* minority data. In a domain where the class imbalance is extreme, however, this assumption may not stand: The minority class may be so rare that it never appears in the unlabeled pool. Our work differs in that we do not make any assumptions concerning the presence of minority class data in the unlabeled pool and attempt instead to make the best out of what we would have if there were, in fact, no such data.

Finally, in [Ghasemi et al., 2011a], Ghasemi et al. use kernel density estimation to find instances in the unlabeled pool which fit the target class well. This approach is similar to that of [He et al., 2006] in that it looks actively for instances that are similar to those that are already available in the labeled set. We believe that this may cause them to miss instances belonging to small disjuncts of the majority class, as the density around those instances will be low. Conversely, we hope that the method we propose can address this problem.

2.4 Evaluating and Comparing Methods

Because we will, in this thesis, propose a novel method of performing active learning for one-class classifiers, and compare that method to existing alternatives, it is important to determine how this comparison should be made. This chapter addresses method of evaluation for classification algorithms, with an emphasis on methods that are appropriate for domains exhibiting a class imbalance. It is heavily based on the book "Evaluating learning algorithms: a classification perspective" by Japkowicz and Shah [Japkowicz and Shah, 2011] as well as the "Assessment Metrics for Imbalanced Learning" chapter from the book "Imbalanced Learning - Foundations, Algorithms, and Applications" [Japkowicz, 2013]. First, we will look at performance measures that can be used to compare algorithms, given a training and test set. Second, we will look at methods of error estimation, which guide how we can use available data to produce reliable estimates of performance. Finally, we will discuss statistical tests that can be used to establish the statistical significance of experimental results.

2.4.1 Measuring Performance

Many performance metrics are based on the confusion matrix, which details how many instances of each class were classified as that class, or any other class. In the binary case, we obtain the confusion matrix shown in Table 2.1.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: The confusion matrix for binary classification problems

From the confusion matrix, we can derive many of a family of performance metrics which some authors call *threshold metrics* [Ferri et al., 2009, Caruana and Niculescu-Mizil, 2004].

For example, equation 2.1 gives the equation for accuracy, a simple but often inappropriate metric.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

The most obvious issue with accuracy, in a highly imbalanced learning setting, is that a classifier that always predicts the majority class label could obtain a very good accuracy score, yet it would still be completely useless. Possible alternatives are sensitivity/specificity and precision/recall, two pairs of metrics that aim to better characterize the performance on both classes.

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.4}$$

These metrics offer more precise information, but have one major disadvantage: They come in pairs of measurements, rather than a single score, making it more difficult to compare alternatives and to perform statistical testing. To remedy this issue, many metrics have been proposed which combine the sensitivity/recall, specificity, and precision scores into composite scores. For example, [Kubat et al., 1998] proposed to use the geometric mean of the sensitivity and specificity, or the geometric mean of precision and recall. The former version places equal importance to each of the two classes, while the second is

designed around the positive class. Other combination metrics include the F-measure, which combines precision and recall using a parameter α which determines the relative importance of precision and recall; the macro-averaged accuracy [Ferri et al., 2009], which is simply the arithmetic mean of the sensitivity and specificity; the mean-class-weighted accuracy [Cohen et al., 2006], a weighted arithmetic average of the sensitivity and specificity; and many more [Japkowicz, 2013].

A disadvantage of the metrics discussed above, however, is that they rely on the assumption that the distribution of classes in the test set is representative of what the classifier will encounter when in operation, and do not inform us on what would happen under different class distributions [Japkowicz, 2013]. In addition, many classifiers allow the user to set a threshold, allowing control over the sensitivity of the classifier, at the cost of specificity, and vice versa. For example, if using the reconstruction error of an autoassociator to judge how well an instance fits a target class, as in [Japkowicz et al., 1995] and [Japkowicz, 2001b], one can vary the threshold on the reconstruction error at which an instance is determined to belong to the target (positive) class. If the threshold is lowered, then more instances will be classified as belonging to that class, which will bring about more true positives, but also possibly more false positives.

The common alternative to threshold metrics is what [Ferri et al., 2009, Caruana and Niculescu-Mizil, 2004] call ranking methods and metrics. The first, and probably most popular, method of this family is the ROC (Receiver Operating Characteristics) Analysis, which consists of plotting the relation between the true positive rate (TPR) and false positive rate (FPR) of a classifier as its decision threshold is moved to allow more or less positive classifications. An example of a ROC curve is shown in Figure 2.2.

The figure demonstrates how ROC curves allow us to see the conditions under which different alternatives perform best. In the figure, the classifier in blue catches more true positives when the threshold is strict and few instances are classified as positive (low end of

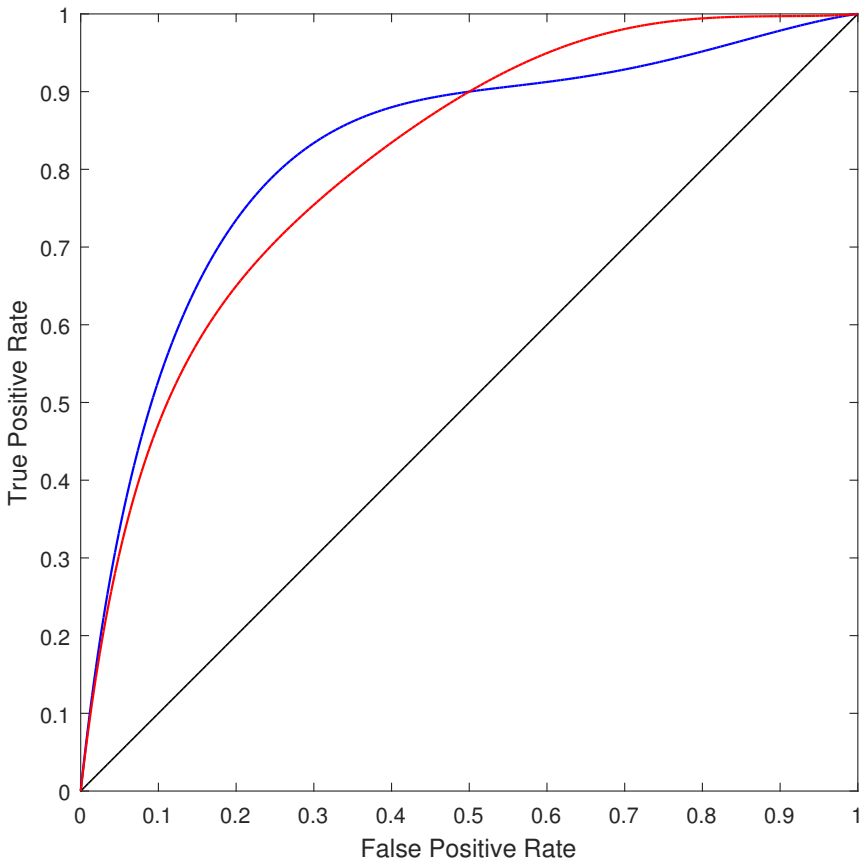


Figure 2.2: Example of a ROC Curve

the FPR axis), but has trouble catching the more difficult examples of the positive class without also incurring many false positives. The classifier in red, on the other hand, needs a more lenient threshold to effectively find true positives, but performs better than the blue classifier in situations where false positives are more tolerable than false negatives.

If a classifier's curve in ROC space is always above another classifier's curve, then the first classifier dominates the second, and is the better alternative. In some cases, however, such as the one in the figure, no classifier dominates across the board, and the classifier to choose depends on the costs of the different mistakes, and the level of imbalance.

Relatedly, it is fairly common to use the area under the ROC curve of a classifier (AUROC) as a performance metric. This measure corresponds to the probability that, given a random positive instance and a random negative instance, the classifier will judge

the positive instance to be "more positive" than the negative instance (i.e. more likely to belong to the positive class). An AUROC of 1 corresponds to a perfect classifier that identifies all positive instances without making a single false positive prediction. A classifier that outputs predictions at random, on the other hand, would obtain an AUROC score of 0.5. Interestingly, a binary classifier that obtains an AUROC below 0.5 can be converted into a stronger classifier by taking the opposite of its predictions.

Although the AUROC has the advantage of not being biased towards any of the classes, in practical applications, it may be misleading: When calculating the area under the ROC curve, we lose information about any potential intersections between the curves of different classifiers. The AUROC, as a general score, tells us nothing about the cost situations under which different alternatives perform best. Hence, a classifier achieving a higher AUROC score than another does not necessarily mean that the first classifier will be preferable in every application. The ROC curves themselves can provide us with this information. However, another ranking method, cost curves, may be preferable when the costs associated to errors and the class distribution are known, due to their readability. In cost-curves, the expected cost of misclassifications is plotted as a function of the class distribution. By using cost-curves, it is therefore much easier to tell at which levels of class imbalance a classifier becomes preferable to another. However, when evaluating classifiers in general, without any specific application in mind, or when the misclassification costs are unknown, cost-curves lose their utility, and ROC curves are preferable.

Finally, in a manner similar to the ROC curves, Precision-Recall curves plot the precision of a classifier as a function of its recall. Here, when a classifier has a high recall score, it is usually less precise, as it is admitting instances into the positive class predictions with more lenience. On the other hand, when a classifier is very strict as to which instances it classifies as positive, then its recall is lower, but its precision is usually higher. The precision recall curves indicate how this relation between recall and precision evolves for a given classifier.

Some authors claim that precision-recall curves may be more appropriate than ROC curves when the class imbalance is very pronounced [Davis and Goadrich, 2006].

In our experiments, we will use ROC curves, and the area underneath, to evaluate classifier performance. We choose to use ROC curves not only because of the advantages listed above, but also because of their popularity in the literature. In addition, we judge that the main disadvantage of using the AUROC, namely that it does not provide information about the intersections between ROC curves, is of lesser importance to us, as we are mostly interested in what happens on average *across the board*.

2.4.2 Error Estimation

There are many ways to estimate the performance and error of a classifier beyond simply dividing the data into a training and a test set, training on the former and measuring performance on the latter. We review a few of them described in [Japkowicz and Shah, 2011] and that are commonly used in the evaluation of learning algorithms.

The first technique is cross-validation. When using cross-validation, the dataset is divided in k folds (we would then call it k-fold cross-validation), i.e. subsets of the dataset each containing the same number of instances. Then, at each of k iterations, one of the folds (a different one every time, or simply the k^{th} fold) is used as the test set, and the other $k-1$ are combined to form a training set. A model is induced using the training set and evaluated on the test set. The results are recorded, and we move onto the next iteration. Table 2.2 gives an example of how the folds would be used at each of the 5 iterations of 5-fold cross-validation.

When $k = n$, the number of instances in the dataset, the test set at each iteration is composed of a single instance, and the training set of every other available instance. This

		Use for each fold				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Iteration #	1	Test	Train	Train	Train	Train
	2	Train	Test	Train	Train	Train
	3	Train	Train	Test	Train	Train
	4	Train	Train	Train	Test	Train
	5	Train	Train	Train	Train	Test

Table 2.2: Folds in 5-fold cross-validation and their use in each iteration

is called *leave-one-out* cross-validation, and is analogous to the jackknife method from statistics.

Because the training set is smaller in cross-validation than if we were using the entire dataset, it may underestimate the performance we would get if we trained a classifier using the entire dataset. For this reason, higher values of k may lead to a smaller bias in the results obtained by cross-validation. In addition, if we are averaging the results obtained over the k folds, then having more folds may reduce the variation in our results. A possible source of variation in cross-validation is the random number generator used to divide the dataset into folds. As a thought experiment, imagine what would happen if one of our classes had 5 subconcepts, and by some awful luck each of the subconcepts was put into its own fold, separate from the others. The classifier would then always be tested on instances of a subconcept it was not trained on. This example is obviously an exageration, but it shows how the randomness in the folds may affect the results. An alternative to increasing k in order to remedy this problem is to perform $r \times k$-fold cross-validation, where k-fold cross-validation is performed r times, each time with a different assignment of instances to folds.

Another consideration when creating the folds is the distribution of classes across the folds. In *stratified* cross-validation, the class priors are kept constant across the folds, which is generally preferred.

An alternative to cross-validation is to use a bootstrap, sampling with replacement from

the dataset to form a new dataset, and dividing this dataset into training and test sets. This process is repeated many times (usually hundreds, at least), and the average result is reported. The bootstrap results may be biased, however, because *a*) some instances may be repeated in the training and test sets, and *b*) some instances may be repeated in the training set, leading to less variety. Some variants of the bootstrap have been proposed to correct these biases, such as the $\epsilon 0$ bootstrap, and the .632 bootstrap [Japkowicz and Shah, 2011].

In our experiments, we will use, in general, 10-fold cross-validation. It is, by far, the most popular candidate in the machine learning literature. In Chapter 6, however, we will use 10x10-fold cross-validation instead. This is because the experiments in Chapter 6 deal with small subconcepts, and, as described above, the randomness in fold partitioning can lead to very lucky and unlucky results on such subconcepts. Using 10 iterations of cross-validation helps to average out the impact of this "luck".

2.4.3 Statistical Tests

The statistics literature offers a vast panoply of statistical tests to choose from, and a few are particularly useful when comparing learning algorithms. The first important concern when choosing a statistical test is whether the samples we have measured for each of the alternatives are paired or not. Typically, when comparing learning algorithms, each algorithm will be applied to the same dataset, or the same fold of the dataset during cross-validation, producing paired samples, making it important to select a statistical test that uses paired samples.

The first, and most obvious, possibility is the paired t-test, which can be used when comparing exactly two alternatives. To perform the paired t-test between samples A and B, indexed by i, we first calculate the pair differences D_i between A and B, then use these

differences D_i in a one-sample t-test, following equation 2.5. The t statistic from that equation follows a Student's t distribution of degree of freedom $n - 1$.

$$\mathrm{t} = \frac{\overline{X}_D}{s_D/\sqrt{n}} \tag{2.5}$$

The t-test, however, relies on a hypothesis that must be taken into account: It assumes that the samples for each compared condition follow a normal distribution. This means that the t-test may be appropriate to compare measurements taken over a series of folds of cross-validation on a single domain, but, unfortunately, the performance of a classifier over a set of *multiple* domains may not satisfy the normal distribution requirement.

In the case where we cannot safely assume that samples are normally distributed, it may be safer to use a non-parametric test. The Wilcoxon rank-sum test, or Mann-Whitney U test [Mann and Whitney, 1947] is a common non-parametric alternative to the t-test. When using the Wilcoxon rank-sum test, a rank is assigned to each observation, with 1 assigned to the smallest value. The test statistic, U, is the minimum of U_1 and U_2 where U_i is defined by equation 2.6, R_i is the sum of the ranks of sample i, and n_i is the size of sample i.

$$U_i = R_i - \frac{n_i \times (n_i + 1)}{2} \tag{2.6}$$

An even simpler non-parametric, and paired, statistical test is the sign test. The sign test is easy to use, but may lack statistical power [Corder and Foreman, 2014]. The general idea behind the sign test is to treat a classifier beating a second classifier as a having probability 0.5 under the null hypothesis. If the null hypothesis is true, then, the number of times W the first classifier beats the second over n paired observations should follow a binomial distribution $W \sim b(n, 0.5)$. To use the sign test, one can compare the probability

of obtaining results as extreme as theirs under this hypothesis to a given significance threshold, and declare their results statistically significant if the threshold is met.

The last case we will cover is that when *more than one* alternative is compared (e.g. multiple competing learning algorithms). In this case, one may choose to use the repeated measures ANOVA if the results can be expected to follow a normal distribution. The repeated measures ANOVA is performed by testing a statistic F equal to the ratio between the mean squared variability *between* groups (or treatments) and the mean squared variability *within* groups. Because of the reliance on normally distributed results, however, we will not use the repeated measures ANOVA, but will use the non-parametric alternative instead: the Friedman test. To apply the Friedman test [Japkowicz and Shah, 2011], given observations x_ij where $1 \leq i \leq n$ corresponds to a row, m a tuple of observations, or a block, and $1 \leq j \leq k$ corresponds to a column, a treatment, or a learning algorithm, we must first compute the ranks r_{ij} in each block, where ties are replaced by the average of the ranks that would've been assigned if there had been no tie. Then, we compute the average rank $\overline{r}_j$ for each column, following equation 2.7; and the average rank overall, following equation 2.8.

$$\overline{r}_j = \frac{1}{n} \sum_{i=1}^{n} r_{ij} \tag{2.7}$$

$$\overline{r} = \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \tag{2.8}$$

Those are used in equations 2.9, 2.10 and 2.11, leading to the Q statistic, which should follow a χ^2 distribution with $k-1$ degrees of freedom when n is large ($n > 15$).

$$SS_t = n \sum_{j=1}^{k} (\overline{r}_j - \overline{r})^2 \tag{2.9}$$

$$SS_e = \frac{1}{n(k-1)} \sum_{i=1}^{n} \sum_{j=1}^{k} (r_{ij} - \overline{r})^2 \tag{2.10}$$

$$Q = \frac{SS_t}{SS_e} \tag{2.11}$$

The Friedman test will tell us if there is a difference *somewhere* between the multiple compared alternatives. To identify the pairs of treatments that differ significantly, however, one must use a post-hoc test such as the Nemenyi test [Nemenyi, 1962]. To apply the Nemenyi test, we compute q following equation 2.12.

$$q = \frac{\overline{r}_{j_1} - \overline{r}_{j_2}}{\sqrt{\frac{k(k+1)}{6n}}} \tag{2.12}$$

We reject the null hypothesis if $q > q_\alpha/\sqrt{2}$.q_α is the q-value for the Tukey test [Tukey, 1949] divided by $\sqrt{2}$, where the degree of freedom for the Tukey test is $(n-1)(k-1)$.

There are more statistical tests, such as McNemar's test [McNemar, 1947], which may be applicable in some situations. Those outlined above give an overview of the different situations under which one may decide to use a statistical test.

To conclude, once again, because in most of our experiments we compare multiple alternative methods against one another in a series of conditions, we will generally use Friedman's test, along with the Nemenyi post-hoc test, to assess the statistical significance of our results.

2.5 Example: Gamma-Ray Anomaly Detection

Before we move on to the details of the proposed method, we will discuss an application of the techniques described above. The domain in question, gamma-ray anomaly detection, was the initial motivation for the research that became this thesis.

As part of a research contract with the Radiation Protection Bureau of Health Canada, we were tasked with developing a system to automate the detection of anomalies in gamma-ray spectra. The spectra, sampled from sodium iodide detectors, consist of energy level measurements at hundreds of different frequency bands.

Due to constant levels of background radiation in the atmosphere, these spectra tend towards a particular shape which domain experts have learned to recognize. If, however, an abnormal source of radiation appeared near the detector, its impact should be visible on the sampled spectra, and experts should be able to recognize it.

Hence, we divide the spectrum domain into two classes: "background" spectra, which show no signs of abnormal levels of radiation, and "anomalies", which do. For obvious reasons, however, the amount of data available in the two classes is extremely imbalanced: While a detector may never see a single anomaly, it will produce hundreds, if not thousands, of background samples every day. It is also important to consider that any set of anomalies we could amass could hardly be considered representative of future anomalies: Gamma-ray anomalies can take a very large variety of shapes, depending on many factors such as the radioactive elements at their source, the environmental conditions, the location and intensity of the source relative to the detector, etc.

For these reasons, one-class classifiers appear to be a sensible choice in this domain. There remains, however, the problem of labeling data: While we expect most of the spectra produced by detectors to be background, we certainly don't want to mistakenly include an

anomaly in the background data training set.

In this specific application, labeling is particularly costly: The Radiation Protection Bureau may deploy many detectors across Canada, each of which may have slightly different hardware, and will operate in different environmental conditions. Because of this, the training set built from data from one detector may not be appropriate to build a model for another detector, and it is necessary to label data from (and for) each detector individually.

Active Learning offers the alluring prospect of potentially obtaining better results with fewer labeled instances. The goal of our research with Health Canada, and the object of this thesis, is to find a way to adapt the idea of active learning to one-class classifiers.

2.6 Chapter Overview

This chapter covered this thesis' background and described the literature that relates to our topic. We started by discussing the three areas of research at the intersection of which lies our work: the class imbalance problem, one-class learners, and active learning. We also provided an overview of the methods that are standard in the field for evaluating and comparing learning algorithms. These methods will be used in our experiments, in Chapters 4, 5, and 6. Finally, we described the problem of gamma-ray anomaly detection, where our work can be applied, and explained how the challenges associated with this problem motivate our work.

In the next chapter, we will describe our proposed solution to this problem.

Chapter 3

Proposed Method

In section 2.3, it was explained that the component of the active learning process which is usually subject to change from one technique to another is the selection criterion. As mentioned in the introduction of this thesis, our main contribution is a new method to perform active learning when one-class classifiers are used, and at the core of this new method is a selection criterion, which Section 3.1 will describe. Later, in Section 3.2, we offer examples of implementations of this selection criterion with various one-class classifiers. Our method, however, is not limited to these implementations and can be adapted for nearly all one-class classifiers.

3.1 A New Selection Criterion

While choosing the selection criterion at the heart of our method, we were faced with five constraints, properties of the problem at hand that would guide the type of selection criterion that could be effective:

First, if we choose to use one-class classifiers, then we must accept that instances which

come out of the labeling process bearing the label of the second class, the class we do not use during the learning process, cannot be used and will not have any impact on the performance of our system. Once again, as explained in section 2.2, it may appear as a terrible idea to simply discard instances of one of the classes, the minority class, when the information in those instances could be used to build a potentially stronger classifier. However, we maintain that position given that *a*) in some domains, such as gamma ray anomaly detection, for example, we can never truly assume that our minority class sample is representative of future minority examples, and that *b*) in such domains, the minority examples we do have help us only marginally on future cases, if not harm us (if for example the future minority examples are very unlike the ones we have).

Second, we would like a selection criterion which has the potential to work with more than one type of one-class classifier, as no implementation of one-class classification is perfect, and every domain may warrant a different choice of classifier. Hence, the ideal active learning method should not rely on any specific implementation.

Third, we are concerned with domains where there is a highly pronounced class imbalance, as these are typically the domains where one-class classification is applied. Because of this, we must choose our selection criterion knowing that however we prioritze the instances, the majority of the instances we find are likely to belong to the majority class nonetheless.

Fourth, we remember that the goal of an active learning process is to establish a dataset which can be used to build a robust classifier, which can preferably perform well not only in the normal case, but also when faced with rare occurences. Our active learning selection criterion should therefore aim to build a training set where all (or, at least, as many as possible) of the subconcepts of the majority class are well represented, despite how small they would be in a randomly sampled dataset. Indeed, it was mentioned in section 2.1.2 that even within-class imbalance in the majority class can be an infamous source of errors. A method which, either directly or as a side-effect, causes the underrepresented subconcepts

of the majority class to be selected more often would therefore be preferable.

Fifth, we draw inspiration from the expected model change [Settles et al., 008b] paradigm and aim for a selection criterion that leads to the labeling of instances that bring about (constructive) changes in the resulting classification models. Indeed, labeling instances just to have a larger dataset would be ridiculous.

These guidelines are summarized in the question which leads to our selection criterion: "Which instances from the unlabeled pool would have the largest impact on our dataset if they were to receive the majority class label?".

The answer we offer to that question is that we should prioritize instances that *look* like they might belong to the minority class, given that *a*) it is very unlikely to actually be the case, *b*) if they are majority instances, then the classifier would likely have made a mistake classifying them as minority, and *c*) by verifying that these instances are indeed majority instances and adding them to the training set, we can cause the model to change and avoid making similar mistakes in the future.

We propose the following criterion:

Given *a*) a one-class classifier, represented by the function $f(x)$ which, for an instance x, provides a score indicating the classifier's confidence that the instance belongs to the majority class (the class on which the classifier was trained), and *b*) $X = \{x_1, x_2, ..., x_n\}$ a set of unlabeled examples from the domain, then let $u_1, u_2, ..., u_n$ be an ordering of Xsuch that $i \leq j \iff f(u_i) \leq f(u_j)$ (each x_i is represented by exactly one u_j, but in this particular order). The selection criterion we propose is that the next instance to be labeled should be u_1, the first instance in this ordering, and that if we are to label a batch of m instances before retraining the model, then that batch should contain the instances $\{u_1, u_2, ..., u_m\}$.

There is an interesting parallel to make between the criterion we propose and an existing active learning method. We have already made the reference to the expected model change approach, but there is similarity between our method and uncertainty sampling. In uncertainty sampling, we choose the instances which the current classifier is most *uncertain* about. In our method, on the other hand, we choose the instances about which our classifier is most *certain*, certain that they belong to the minority class. The nuance, and rationale behind our method, is that it relies on the fact that the dataset is very highly imbalanced and that it is more likely that it is making a mistake than that the instance really belongs to the minority class.

Another interesting property of the method we propose is that its integration into a one-class classifier active learning deployment process comes naturally: If a one-class classifier has already been trained and is in operation, outputting predictions, and that, as is usually the case, the minority class predictions it makes are inspected thoroughly by domain experts, then these experts must be flagging many of these instances as false positives. Taking these false positives, the mistakes made by the system, and feeding them back into the training set of the one-class classifier, without doing anything with the instances the established system judges to be of the majority class, is, in essence, an implementation of the active learning method we propose.

3.2 Specific Implementations

With the proposed selection criterion in mind, all that remains is to devise implementations of our methods for different one-class classifiers. In order to do this, we need to find a way to obtain scores from the classifier that meet the criteria for the $f(x)$ function described above. We will see, in subsections 3.2.1, 3.2.2, and 3.2.3 how to do this with three one-class classifiers: the one-class SVM, a classifier based on the Mahalanobis distance, and, finally,

a classifier based on the distance to the k nearest neighbors.

It is important to note that, although we provide here a handful of implementations, our method could be adapted to work with virtually any one-class classifier. Nearly all one-class classifiers output some form of confidence measure, some sort of distance, or probability, that indicates how well an instance appears to fit the learned class. To adapt our method to such classifiers, it suffices to use this metric as the function $f(x)$ in our method's description. In fact, this is precisely how the implementations that follow were designed.

3.2.1 One-Class Support Vector Machines

As explained in section 2.2, one-class support vector machines find a hyperplane that separates, with the largest margin possible, the data from the target class (the majority class) and the origin of the kernel space.

One possibility to obtain our function $f(x)$ from the one-class SVM is to vary the ν parameter, which controls how tightly the hyperplane wraps around the target class data. With an extremely small value of ν, we can be very selective as to what makes it in the majority class predictions, and make this selection more lenient by increasing ν. There are problems with this approach, however: *a*) it is not necessarily possible to find a ν value that places the decision boundary exactly between two instances, which means that we can't necessarily get the full ordering and *b*) changing ν can (and most likely will) impact the support vectors that are used, which means that if one instance is classified as majority and another as minority under one value of ν, then nothing guarantees that they won't switch places under another..

Another possibility is to use the distance of each instance to the hyperplane, a value

which is usually outputted along with the predictions. Positive distances correspond to values on the target class' side of the hyperplane and negative distances correspond to values on the origin's side, or vice versa depending on the implementation. We can therefore assume that a higher/lower distance indicates that the instance fits the majority class more/less, and use that as the value for $f(x)$. This approach is the one that was used in our experiments.

3.2.2 Mahalanobis Distance Classifier

The Mahalanobis distance classifier fits a normal distribution to the target class' training data, then bases its predictions on the Mahalanobis distance of new instances to that distribution. The Mahalanobis distance [Mahalanobis, 1936] is calculated using equation 3.1

$$d = (x - \mu)\Sigma^{-1}(x - \mu) \tag{3.1}$$

A higher Mahalanobis distance indicates that the value fits the target distribution *less*, and its inverse, or its opposite (to avoid potential divisions by zero) can be used as a measure of confidence that the instance belongs to the majority class. This is precisely what we need for the function $f(x)$.

3.2.3 Distance to K Nearest Neighbors Classifier

In this variant of one-class classification, the level to which an instance fits the majority class is measured by taking the k nearest neighbors of that instance in the training set, calculating the distance between each of these neighbors and the instance in question, and using some aggregate function of these distances as the score, usually the average. An

instance which is far from its nearest neighbors in the training set is determined not to be in the neighborhood of the target class and, therefore, not to fit with that class very well.

Just like with the Mahalanobis distance, we have here a measure of distance on which a threshold is set to determine which instances are majority, and which are minority. To turn this distance into the score $f(x)$, it suffices to take its inverse or opposite.

3.3 Chapter Overview

In this chapter, we presented the method at the core of this thesis, which consists of a novel active learning selection criterion. Through this criterion, once again, we propose that instances that resemble the majority class the most should be labeled first.

The goal of the remaining chapters is to describe our analysis of the proposed method and the experiments performed to investigate its properties.

Chapter 4

Evaluating the Method

The objective of the experiment described in this chapter is to establish with empirical evidence the claimed merits of the method we propose, which was described in Chapter 3.

4.1 Experiment 1

General Evaluation

In this experiment, we assess the impact of the proposed method on classification performance and compare this impact to that of two alternative methods.

4.1.1 Procedure

We follow the same general process as would take place during the deployment of an active learning system:

1. An initial set of labeled instances is used to induce a classification model. Because

we are using a one-class classifier, only the majority class instances are used in this process.

2. The resulting classifier is evaluated on a set of instances held out from the training set.

3. The resulting classifier is used, along with the selection criterion, to select a batch of instances from a large pool of unlabeled instances.

4. The selected instances (called the *augmentation*) are labeled, and the minority class instances are discarded. The majority class instances are incorporated to the initial training set.

5. The augmented training set is used to induce a second classification model.

6. The second classifier is evaluated on the same set of instances that were used when evaluating the initial classifier.

7. The performance of both classifiers is recorded. The first is recorded as the "Initial" result, and the second is recorded under the name of the selection criterion used.

Steps 3 through 6 are executed multiple times, in parallel: once for each of the compared alternatives for selection criteria. These alternatives are described in subsection 4.1.2.

4.1.2 Compared Alternatives

We compare three different selection criteria:

- Random sampling, a naïve approach that allows us to establish how much of our results are due to a larger training set alone

- The selection criterion proposed by Ghasemi et al. in [Ghasemi et al., 2011a], where kernel density estimation is used to select in the unlabeled pool in dense areas of the one-class learner's target class
- The selection criterion proposed in this thesis

In addition, as explained in subsection 4.1.1, the performance of the initial classifier, without any active learning, will be reported.

4.1.3 Evaluation Methodology

Performance is measured in steps 2 and 6 of the procedure (see subsection 4.1.1) using the area under the ROC curve.

In order to obtain a more reliable estimate of performance and reduce the impact of the assignment of instances to the test set, initial training set, and unlabeled pool, we use 10-fold cross-validation and repeat the procedure described in subsection 4.1.1 at each iteration. The dataset is split in 10. At each iteration of cross-validation, one of the folds is used as the test set, 3 are combined to form the initial training set, and the remaining 6 to form the pool of unlabeled instances. Note that we simulate the situation of not having the labels, but keep them around to be returned to the instances selected by the active learning selection criterion, so that we do not need the help of a human expert during the experiment.

The size of the batch of instances selected by the active learning procedure corresponds to one sixth of the instances made available to it. This value was selected to balance the needs *a*) to have a large enough augmentation so that its impact on the training set would be noticeable and *b*) to force the active learning process to be selective, not allowing it to use too many of the instances in the unlabeled pool.

Once 10 performance measurements have been taken for the classifier trained on the initial training set and the 3 compared selection criterion alternatives, Friedman's non-parametric test was used to assess whether there was a significant difference in performance between all 4 four treatments. Given a significant difference, Nemenyi's post-hoc test was used to identify the pairs of treatments between which the difference was important.

4.1.4 Datasets Used

We performed the entire experiment on three different datasets. The first two, *Pen Digits* and the *MAGIC Gamma Telescope* datasets, were taken from the UCI Machine Learning Repository [Bache and Lichman, 2013]. Since these are datasets generally used for multiclass classification, they had to be converted to a binary format: Classes were combined into two groups, only one of which (the larger of the two groups, i.e. the majority class) was used to train the one-class classifier.

The third dataset was given to us by our collaborators at the Radiation Protection Bureau of Health Canada. It consists of 19113 samples from a Sodium iodide detector in the city of Saanich, British Columbia, collected over a period of 7 months. The sampling period was of 15 minutes. Each sample consists of photon counts over 512 energy bins, but only the first 250 were used following advice from domain experts. The photon counts are non-negative integers. Of the 19113 instances, 95 are anomalies and 19018 are normal. On this dataset, it was not possible to use Ghasemi et al.'s active learning method because of the data's dimensionality, which causes sparsity: When applying kernel density estimation to the Saanich dataset, the probability density around points in the dataset was always so small that our MATLAB environment rounded it down to zero.

(a) **Dataset: Pen Digits**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.9484	0.9504	0.9475	0.9391	
2	0.9399	0.9411	0.9399	0.9339	
3	0.9566	0.9576	0.9565	0.9518	
4	0.9394	0.9365	0.9398	0.9344	
5	0.9249	0.9268	0.9249	0.9214	
6	0.9328	0.9337	0.9348	0.9275	
7	0.9286	0.9313	0.9306	0.9289	
8	0.9372	0.9373	0.9383	0.9336	
9	0.9517	0.9531	0.9519	0.9442	
10	0.9485	0.9468	0.9487	0.9370	
Avg.	0.9408	0.9415	0.9413	*0.9352*	-0.0056
S.D.	0.0104	0.0101	0.0099	0.0086	

(b) **Dataset: MAGIC Gamma Telescope**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.8115	0.8123	0.8092	0.8007	
2	0.7998	0.7993	0.8021	0.7943	
3	0.7956	0.7954	0.7961	0.7886	
4	0.8092	0.8078	0.8058	0.7997	
5	0.7909	0.7923	0.7873	0.7854	
6	0.7935	0.7950	0.7951	0.7931	
7	0.7891	0.7913	0.7904	0.7932	
8	0.8078	0.8069	0.8074	0.8011	
9	0.8230	0.8202	0.8193	0.8122	
10	0.8156	0.8143	0.8125	0.8005	
Avg.	0.8036	0.8035	0.8025	*0.7969*	-0.0067
S.D.	0.0115	0.0102	0.0102	0.0076	

(c) **Dataset: Saanich Gamma Spectroscopy**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.7186	0.7196	0.7156	0.7161	
2	0.7076	0.7058	0.7034	0.7052	
3	0.7188	0.7244	0.7195	0.7232	
4	0.7033	0.7078	0.7083	0.7077	
5	0.7114	0.7060	0.7141	0.7106	
6	0.7211	0.7141	0.7175	0.7153	
7	0.7177	0.7118	0.7148	0.7195	
8	0.7196	0.7208	0.7184	0.7198	
9	0.7027	0.7032	0.7046	0.7064	
10	0.7055	0.7084	0.7119	0.7098	
Avg.	0.7126	*0.7122*	0.7128	0.7134	+0.0008
S.D.	0.0073	0.0073	0.0057	0.0063	

Table 4.1: Results of Experiment 1 using the Mahalanobis distance classifier

(a) **Dataset: Pen Digits**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.9913	0.9916	0.9907	0.9950	
2	0.9908	0.9919	0.9902	0.9932	
3	0.9928	0.9936	0.9925	0.9962	
4	0.9854	0.9901	0.9846	0.9931	
5	0.9830	0.9859	0.9826	0.9904	
6	0.9893	0.9900	0.9892	0.9952	
7	0.9846	0.9860	0.9844	0.9886	
8	0.9844	0.9879	0.9838	0.9948	
9	0.9899	0.9921	0.9896	0.9968	
10	0.9847	0.9870	0.9841	0.9918	
Avg.	0.9876	0.9896	*0.9872*	0.9935	+0.0059
S.D.	0.0035	0.0028	0.0036	0.0026	

(b) **Dataset: MAGIC Gamma Telescope**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.7595	0.7650	0.7617	0.7707	
2	0.7653	0.7673	0.7701	0.7745	
3	0.7715	0.7731	0.7701	0.7786	
4	0.7752	0.7744	0.7769	0.7817	
5	0.7645	0.7663	0.7654	0.7713	
6	0.7675	0.7711	0.7706	0.7754	
7	0.7620	0.7700	0.7667	0.7749	
8	0.7788	0.7836	0.7817	0.7872	
9	0.7856	0.7908	0.7889	0.7900	
10	0.7781	0.7790	0.7815	0.7843	
Avg.	*0.7708*	0.7741	0.7734	0.7789	+0.0081
S.D.	0.0085	0.0082	0.0086	0.0067	

(c) **Dataset: Saanich Gamma Spectroscopy**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.7251	0.7240	0.7202	0.7258	
2	0.7168	0.7181	0.7140	0.7174	
3	0.7339	0.7296	0.7251	0.7340	
4	0.7135	0.7037	0.7154	0.7152	
5	0.7070	0.7131	0.7123	0.7100	
6	0.7186	0.7199	0.7206	0.7201	
7	0.7405	0.7268	0.7330	0.7411	
8	0.7223	0.7250	0.7193	0.7229	
9	0.7151	0.7174	0.7217	0.7156	
10	0.7153	0.7139	0.7215	0.7167	
Avg.	0.7208	*0.7191*	0.7203	0.7219	+0.0011
S.D.	0.0101	0.0077	0.0059	0.0095	

Table 4.2: Results of Experiment 1 using the Distance to KNN classifier

(a) **Dataset: Pen Digits**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.9961	0.9958	0.9961	0.9963	
2	0.9954	0.9953	0.9955	0.9949	
3	0.9957	0.9955	0.9956	0.9969	
4	0.9937	0.9942	0.9938	0.9946	
5	0.9883	0.9902	0.9884	0.9916	
6	0.9946	0.9944	0.9947	0.9977	
7	0.9904	0.9915	0.9909	0.9926	
8	0.9901	0.9933	0.9905	0.9967	
9	0.9945	0.9954	0.9945	0.9975	
10	0.9897	0.9916	0.9898	0.9955	
Avg.	*0.9929*	0.9937	0.9930	0.9954	+0.0025
S.D.	0.0029	0.0020	0.0028	0.0021	

(b) **Dataset: MAGIC Gamma Telescope**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.7655	0.7716	0.7700	0.7774	
2	0.7680	0.7739	0.7739	0.7759	
3	0.7812	0.7845	0.7827	0.7862	
4	0.7850	0.7854	0.7852	0.7940	
5	0.7735	0.7773	0.7767	0.7835	
6	0.7755	0.7838	0.7816	0.7870	
7	0.7763	0.7807	0.7808	0.7883	
8	0.7900	0.7941	0.7922	0.8023	
9	0.7984	0.8017	0.8014	0.8046	
10	0.7921	0.7967	0.7943	0.7986	
Avg.	*0.7805*	0.7850	0.7839	0.7898	+0.0093
S.D.	0.0108	0.0099	0.0097	0.0099	

(c) **Dataset: Saanich Gamma Spectroscopy**

Fold	Initial	Rand. Samp.	Ghasemi et al.	Our Method	
1	0.7445	0.7432	0.7526	0.7574	
2	0.7351	0.7455	0.7555	0.7520	
3	0.7494	0.7548	0.7513	0.7635	
4	0.7489	0.7550	0.7433	0.7643	
5	0.7397	0.7415	0.7472	0.7556	
6	0.7430	0.7427	0.7468	0.7501	
7	0.7718	0.7560	0.7591	0.7628	
8	0.7718	0.7742	0.7707	0.7613	
9	0.7735	0.7693	0.7734	0.7660	
10	0.7557	0.7649	0.7566	0.7540	
Avg.	*0.7533*	0.7547	0.7557	0.7587	+0.0054
S.D.	0.0143	0.0117	0.0099	0.0056	

Table 4.3: Results of Experiment 1 using the One-class SVM

4.1.5 Results

Tables 4.1, 4.2 and 4.3 present the results obtained with, respectively, the Mahalanobis distance, the KNN distance and the One-Class SVM, on the three domains. The measurements shown are the area under the ROC curve obtained by varying the threshold at which instances are determined to belong to one class or the other. We report the average measurement over 10 folds of cross-validation, for the classifier trained on the initial training set as well as on the training sets augmented by all 3 selection criteria. In the average measurement row, we underline the best result and italicize the worst. Finally, to the right of the average measurement for our method, we present the difference between the average measurements for our method and the initial training set, to give an idea of the size of our method's effect. The following subsections give an overview of statistical tests on these results.

Mahalanobis distance classifier

As we can see from the results, when using the Mahalanobis distance as the base method, the active learning strategy seems to *hurt* performance rather than help it.

This is confirmed to be statistically significant on the first two domains. The probabilities of obtaining such results under the null hypothesis of the Friedman's test are, respectively, 4.155e-4 and 0.0074 for the pen digits and MAGIC domains. In both cases, Nemenyi's test indicates that the active learning technique is significantly outperformed by the other techniques.

On the Saanich dataset, Friedman's test finds no significant difference between the four treatments.

Distance to KNN classifier

In this case, the active learning technique does seem to lead to better results on average. Its mean performance rank is systematically better than the other two treatments across all three datasets. On the first two, Friedman's test gives us a p-value of 1.380e-6 and 4.709e-5. On both, Nemenyi's test indicates that our active learning method outperforms the Initial and Ghasemi et al. treatments.

On the Saanich dataset, although the trend seems to be the same, with the active learning technique generally coming out on top, the difference is not large enough for statistical significance.

One-Class SVM

Finally, with the one-class SVM, we obtain similar results to the KNN distance. Again, active learning seems to lead to the best results.

Once again, on the first two domains, Friedman's test rejects the null hypothesis. The p-values are 0.0047 and 2.323e-6. On both, Nemenyi's test finds that our method leads to a significant improvement over the Initial and Ghasemi et al. treatments. However, only on the first domain, Pen Digits, is the advantage of our method over random sampling statistically significant according to this test.

4.1.6 Discussion

The disparity between the results obtained with the Mahalanobis distance technique and the other two is an interesting phenomenon. We suspect it may be because of their respective parametric and non-parametric natures.

At the core of the Mahalanobis distance classifier is the assumption of a gaussian distribution. The distance is essentially an indicator of how unlikely an instance is under that distribution. The selection procedure we used essentially oversamples the tails of this gaussian distribution, creating multiple additional modes in the data's distribution. Our hypothesis as to why the Mahalanobis distance classifier performed worse with the active learning is that it is unable to properly handle this artificial multimodal distribution. Another way to see it is through the characteristics of parametric classifiers. With a finite number of parameters, the Mahalanobis model has no choice but to *change* its parameters, to alter its representation of the data it is modeling. Essentially, after the active learning procedure is followed, it is given data with an artificially high variance, because of the added anomalies.

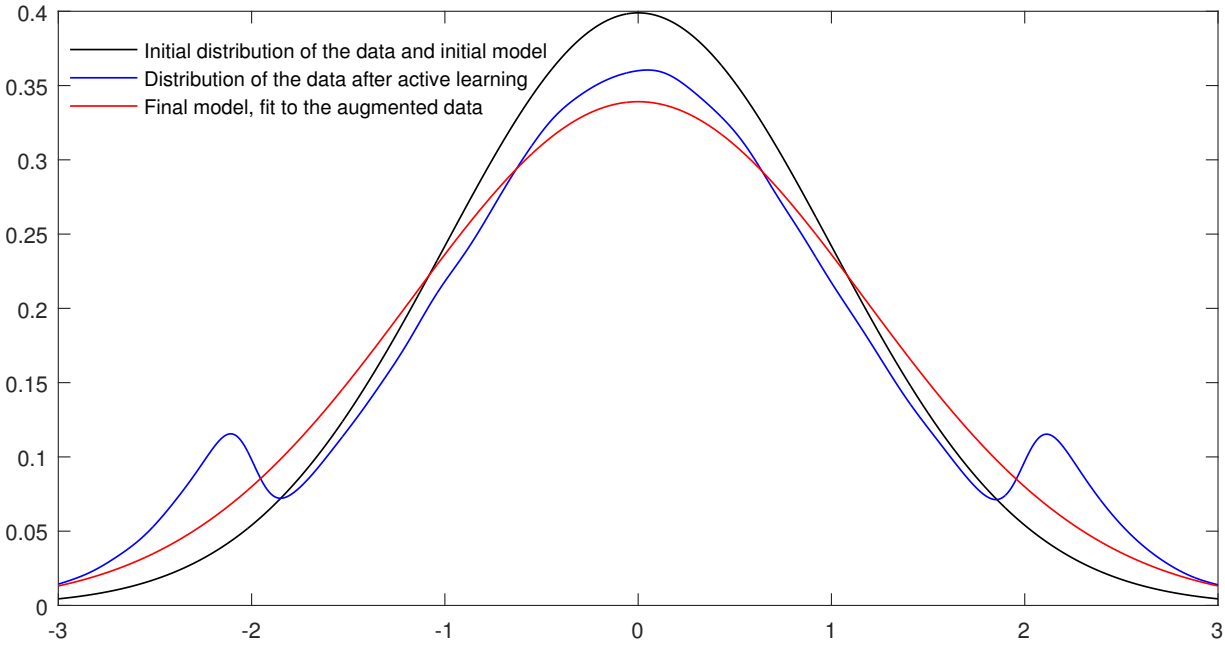


Figure 4.1: Impact of active learning augmentation on data distribution and Mahalanobis distance classifier

An example of how this may occur is shown in Figure 4.1. In the figure, the distribution of the data before active learning, which follows a normal distribution, is plotted in black. In blue, we show the result of oversampling the tails of this distribution, which is what our active learning technique would do in this case. Note that the shape the blue curve takes is

smoothed out because it was drawn using kernel density estimation. Finally, in red, we show the model a Mahalanobis distance classifier would create of the data. This new model is still normal, but with a larger standard deviation caused by the higher proportion of instances away from the mean.

On the other hand, non-parametric classifiers are capable of growing in complexity when given more training data. Rather than *changing* their understanding of the data they learned, they can *add onto it* new subconcepts. The addition of new knowledge does not require them to forget part of what they knew before.

This is particularly true with support vector machines, in this case: These models select a set of instances close to the decision boundary as support vectors. The instances we add through active learning are likely to become candidates for support vectors in the next model, if they are close to the decision boundary. Hence, the active learning process could be providing the SVM with a better selection of support vectors. This is a hypothesis we will investigate further in Chapter 6.

In another vein, we compared our method to Ghasemi et al.'s. Our results suggest that, on these particular domains, and with the two non-parametric one-class classifiers we used, our method is the only one of the two which manages to effect significant performance improvements. This may be because the kernel density based method, by selecting instances in dense areas of the majority class, does not actually provide the distance-to-KNN and One-Class SVM classifiers with useful instances, i.e. instances that will either become nearest neighbours on the border of the class, or support vectors. It is worth mentioning, however, that Ghasemi et al.'s did not, like our method, adversely affect the Mahalanobis distance classifier.

Finally, while we had initially settled on the AUROC as the performance metric of choice to determine our method's merits, we find that it can be difficult to see the concrete

(a) **Mahalanobis Distance**

	TPR		FPR	
Dataset	Before	After	Before	After
Pen Digits	0.8896	0.8800	0.1447	0.1550
MAGIC	0.7127	0.7107	0.2510	0.2592
Saanich	0.5611	0.5600	0.1586	0.1623

(b) **Distance to KNN**

	TPR		FPR	
Dataset	Before	After	Before	After
Pen Digits	0.9603	0.9697	0.0470	0.0302
MAGIC	0.6587	0.6555	0.2520	0.2280
Saanich	0.5863	0.5779	0.2016	0.1779

(c) **One-Class SVM**

	TPR		FPR	
Dataset	Before	After	Before	After
Pen Digits	0.9698	0.9719	0.0370	0.0261
MAGIC	0.6678	0.6811	0.2467	0.2355
Saanich	0.6242	0.6337	0.2254	0.2344

Table 4.4: True and false positive rates before and after our active learning method is used

consequences of an increase in the AUROC in applied scenarios. For this reason, we report, in addition to the AUROC measurements presented in tables 4.1, 4.2 and 4.3, the average true and false positives rates on the different domains; with the different base classifiers; and before (i.e. with the initial training set) and after using our active learning method. To calculate the true and false positive rates, we used the point closest to (0, 1) on each ROC curve (as measured by the Euclidean distance). In addition, we averaged the results over the 10 folds of cross-validation. These results are presented in table 4.4.

What we first notice when looking at the results in table 4.4, is that, while our method inconsistently impacts the classifier's ability to identify true positives, when using the two non-parametric classifiers, it almost always (5 times out of 6) has a positive impact on the false positive rate. In addition, the relative *magnitude* of our method's impact on the true positive rate is much smaller than the magnitude of its impact on *false positives*. Taking, for example, the Distance to KNN classifier on the Pen Digits domain, we notice that

increasing the true positive rate from 0.9603 to 0.9697 means that only $\sim 1\%$ more true positives are caught, but decreasing the false positive rate from 0.0470 to 0.0302 means that the system outputs $\sim 36\%$ fewer false positives. Similarly, on the Saanich domain, with the Distance to KNN classifier, we have reduced the false positive rate from 0.2016 to 0.1779, which translates into $\sim 12\%$ fewer alerts for the domain experts who would, in an applied setting, review all positives signaled by the system.

These results, which highlight the applied consequences of our method, are encouraging. However, it is important to keep in mind that the true and false positive rates tell only a small part of the story (i.e. one point on the ROC curve), and that in any specific application, the true costs of true and false positives should be taken into account to determine the threshold to use and to properly evaluate the method.

4.2 Chapter Overview

In this chapter, we described the experiment performed in order to determine whether or not our method had significant benefits. The results we obtained offer empirical evidence that it does indeed outperform the obvious alternative approaches, and that on more than a single domain.

Although these results are encouraging, more work is required to truly understand how our method works, and the conditions under which it does so best. This is particularly important because the results we obtained, despite being statistically significant, did not have an effect as large as hoped. The next 2 chapters will bring insight into these questions.

Chapter 5

Conditions for Best Performance

Given the support we now have for the merits of our active learning method, we now wish to determine the conditions under which this technique performs best. In this chapter, we will discuss the two experiments we performed to this end. In Experiment 2, we look at the impact of the sizes of the initial training set and of the active learning augmentation on the performance improvement we can expect to see from using active learning. In Experiment 3, we assess whether there is worth in performing active learning incrementally, adding one instance at a time, over adding all newly labeled instances in as a batch.

5.1 Experiment 2

Initial Training Set and Augmentation Sizes

With this experiment, we hope to reveal when it is most profitable to use our method in terms of the number of instances which have already been labeled (the initial training set size) and the number of additional instances which we can afford to label.

5.1.1 Procedure

An important observation from the previous experiments was that active learning did not seem to work very well with the Mahalanobis distance system, while it performed much better with the KNN distance and One-Class SVMs. For this reason, in this section, we narrow down our experiments to one of these three, the KNN distance method. Although the KNN distance method did not obtain the most convincing results in the previous section, it was here selected for practical reasons of computational cost, given the large number of tests performed in this second set of experiments.

In this experiment, we control two factors, the training set size and the ratio between the sizes of the random/active learning augmentation and the training set. The training set size varies exponentially, starting at 10^2 with the exponent increasing by 0.05 until it reaches 10^3. Similarly, the augmentation/training set size ratio runs from 10^{-1} to 10^0 by exponent jumps of 0.05. This gives us augmentations with sizes from 10 (when the training set size is 10^2 and the ratio 10^{-1}) to 10^3 (when the training set size is 10^3 and the ratio is 10^0). All combinations of initial training set size and augmentation size ratio are explored.

5.1.2 Evaluation Methodology

As in the previous experiments, our data is split into 10 folds. The difference is that the set of folds used for training are randomly subsampled to produce the initial training set. In addition, rather than using a sixth of the "held-out" data to augment the training set (either randomly or by the active learning selection procedure), we now have this selection procedure pick a number of instances determined by the chosen augmentation size.

In addition, given the very large number of measurements taken, we only report the average result over the 10 folds.

5.1.3 Datasets Used

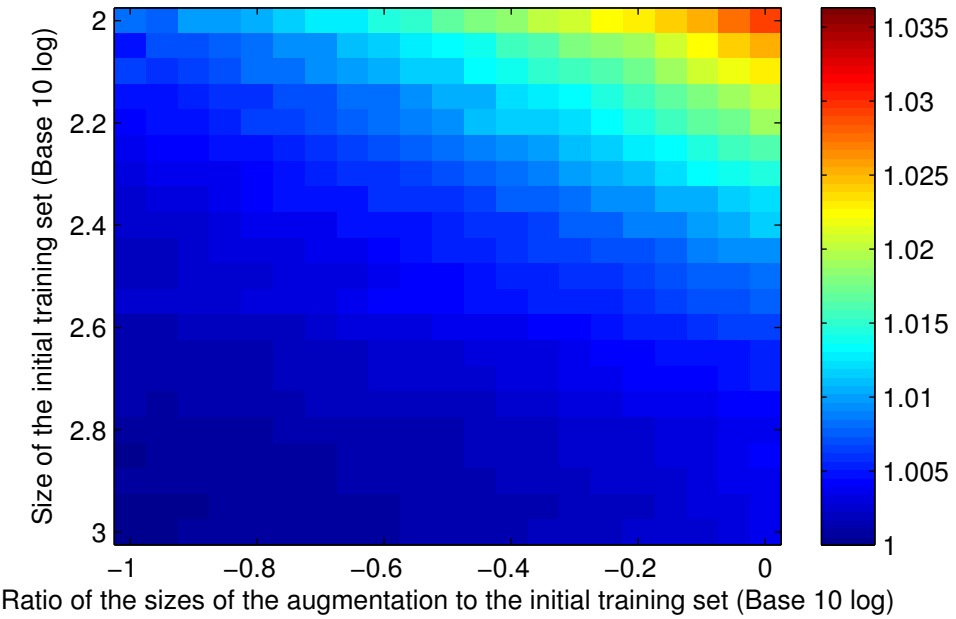
We limit this experiment to the *Pen Digits* dataset, as the low variance in the results obtained when using it may allow us to more easily distinguish general trends in the impact of training set and augmentation sizes.

5.1.4 Results

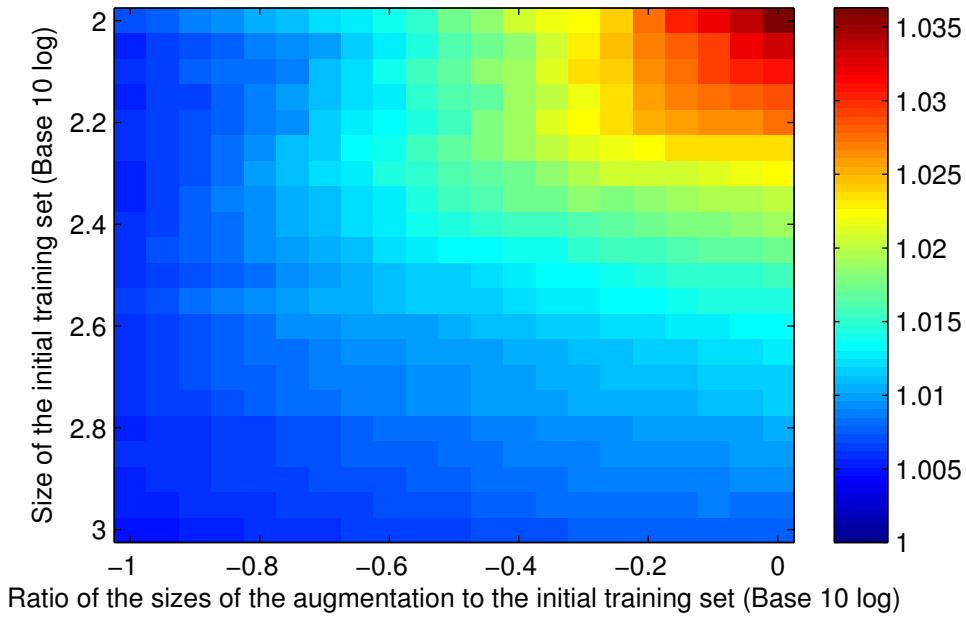
In figures 5.1a and 5.1b, we see the relative performance of the classifier on the augmented and initial training sets when using, respectively, random sampling and the active learning selection procedure to augment the training set. In these figures, and the ones that follow, the horizontal axis represents the base 10 logarithm of the ratio between the size of the augmentations brought by active learning and the size of the initial training set. For example, where the horizontal axis indicates -1, the active learning procedure was allowed to add a number of instances one tenth the size of the initial training set. The vertical axis indicates the size of the initial training set, once again on a logarithmic scale. The color of the heatmap indicates the intensity of the measured value at each point, with colors closer to red corresponding to larger values. In the two heatmaps of figures 5.1a and 5.1b, this is the ratio of the system's performance after/before the active learning augmentation (a value below 1 would correspond to a negative impact).

5.1.5 Discussion

In figures 5.1a and 5.1b, we see the relative performance of the classifier on the augmented and initial training sets when using, respectively, random sampling and the active learning selection procedure to augment the training set. In these figures, and the ones that follow, the horizontal axis represents the base 10 logarithm of the ratio between the size of the



(a) **Random Sampling**



(b) **Our Method**

Figure 5.1: Performance ratio after/before an augmentation of the training set

augmentations brought by active learning and the size of the initial training set. For example, where the horizontal axis indicates -1, the active learning procedure was allowed to add a number of instances one tenth (10^{-1}) of the size of the initial training set. The

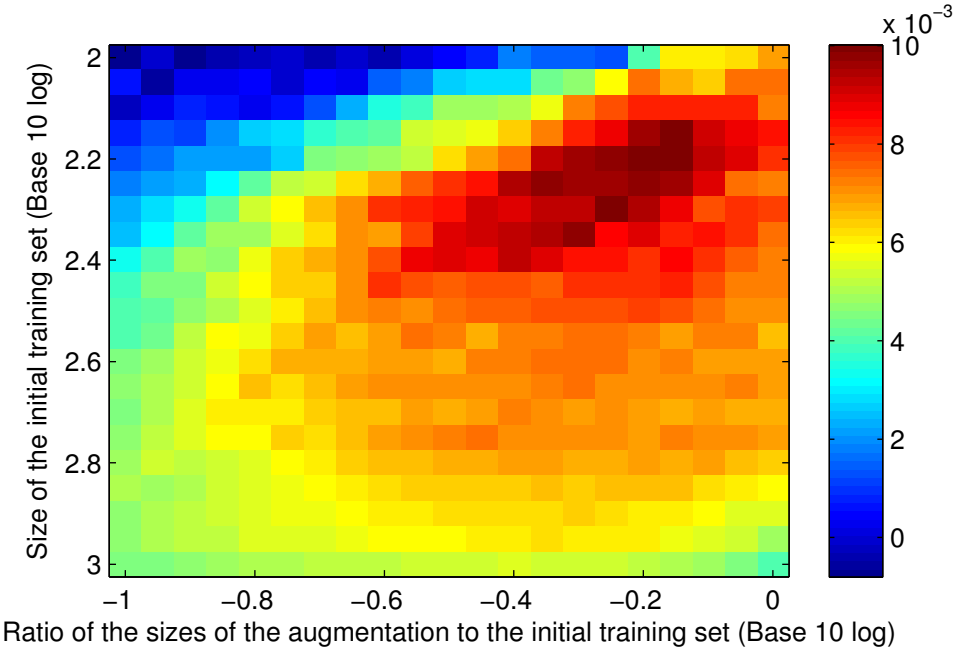


Figure 5.2: Difference in improvement between the active learning and random sampling approaches

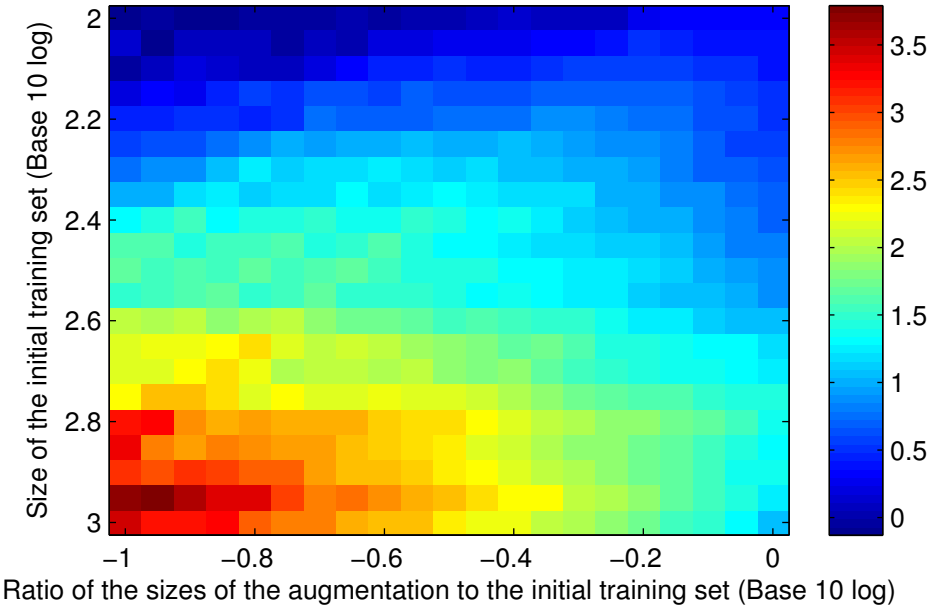


Figure 5.3: Ratio of the improvements brought by the active learning and random sampling approaches

vertical axis indicates the size of the initial training set, once again on a logarithmic scale. The color of the heatmap indicates the intensity of the measured value at each point, with

colors closer to red corresponding to larger values. In the two heatmaps of figures 5.1a and 5.1b, this is the ratio of the system's performance after/before the active learning augmentation (a value below 1 would correspond to a negative impact).

First, we observe that, in both cases, increasing the size of the initial training set while keeping the augmentation size ratio the same brings smaller relative performance improvements. This is not surprising: It is expected that increasing the size of the training set has diminishing returns and that there is less room from improvement when we start with a larger training set.

Conversely, we find that a larger augmentation brings forth larger performance improvements, at a constant initial training set size. For the random augmentation, this is not at all surprising: We always end up with a random sample, but larger is better.

On the other hand, this was not obvious for the active learning augmentation. One might have thought that exclusively selecting seemingly anomalous instances for the augmentations would eventually change the distribution of the training set so much as to adversely affect even the non-parametric method. This, however, does not seem to be the case. The active learning method *systematically* improved the performance.

What is interesting to compare, however, is the shape of the two heatmaps. Given an initial training set size, when using active learning, it seems we can systematically obtain similar levels of performance improvements with a smaller augmentation size. In addition, for every initial training set size, there seems to be a point after which augmentation size increases have little impact on the active learning augmentation, as if it quickly reached its optimum, whereas the random augmentation steadily increases. This suggests that the active learning augmentation rapidly finds the instances most relevant to the problem at hand, after which it is left with irrelevant instances only, which have little additional impact on the performance.

This is particularly easy to see when looking at Figure 5.2, which shows the difference between the previous two heatmaps at each point. The area that is darkest on this heatmap appears to correspond to the general area where, in the active learning augmentation heatmap, the improvements associated with increasing augmentation sizes reach a plateau for a given initial training set size. To the right of this dark area, where the difference between the two augmentations is large, the difference falls back down, as the random augmentation finally identifies the important instances and catches up on the active learning augmentation.

Next we study the improvements of the two types of augmentations as a ratio, rather than an absolute difference. We compute the ratio between the improvements (performance after augmentation - performance before) and draw a heatmap of these ratios, with the heat on a logarithmic scale for ease of visualization. The result is shown in Figure 5.3. Here, warm colors such as shades of red indicate that the active learning augmentation performed many orders of magnitude better than the random augmentation, while a dark blue indicates comparable performance.

This analysis reveals another interesting phenomenon. Although the active learning augmentation gives superior absolute improvements all across the heatmap, the random augmentation is already doing a fairly good job when the initial training set size is small. However, this improvement levels off when the size of the initial training set increases. What we observe in Figure 5.3 is that this is less the case for the active learning augmentation, and that, at large training set sizes, the improvements it effects are orders of magnitude larger than those brought about by random augmentations.

In addition, we see that, mostly at larger training set sizes, the active learning augmentation has a much larger relative advantage when the augmentation is smaller. This is not very surprising: With larger augmentations, the selectivity of the active learning procedure has less of an impact, as it is selecting a large portion of the pool anyway. The instances it

picks are less anomalous, on average. Conversely, with small augmentations, only the most anomalous instances are used, which seemingly leads to a larger impact.

We would like to point out what happens when the initial training set is very small. It seems that at these sizes, the difference between the two types of augmentations is much smaller. This may be because the classifiers still require a base set of relatively normal instances (instances that are not anomalous by any measure) to begin with, and that starting to build a training set exclusively with instances that look anomalous will lead the system to perform poorly in "normal" areas of the data space. This result suggests that perhaps active learning should not always be used from the beginning, and that it might be better to first sample randomly until a critical set of instances has been built, at which point active learning would be used to properly refine the decision boundary. It is not clear how one should determine whether to keep going with random sampling versus active learning, which is an issue that could be explored in future work.

Finally, as the goal of this chapter is to determine when our method works best, we bring attention to a couple of points taken from the data that was used to produce the heatmaps in Figures 5.1b and 5.1a. These points are presented in Table 5.1. Specifically, we show four situations in which our method brought particularly good results relative to what could be obtained using the initial training set (up to a difference of 0.03406 in AUROC), and three situations in which our method brought good results relative to random sampling (up to a difference of 0.01002 in AUROC).

Initial Set	Augmentation	Our Method - Initial	Our Method - Rand. Samp.
100	100	0.03406	
100	90	0.03193	
112	100	0.03117	
112	90	0.03009	
158	71		0.01002
199	56		0.00996
158	63		0.00994

Table 5.1: Points of interest for the performance of the proposed method

In summary, the conclusions we draw from this experiment are that:

- Larger initial training sets lead to smaller performance improvements.
- Larger augmentations lead to larger performance improvements.
- With increasing augmentation sizes, the performance improvements effected by our method initially grow quickly before reaching a plateau, while random sampling grows more slowly to reach this same level much later. This suggests that our method is particularly adept at identifying the relevant instances.
- Our method has the largest relative advantage on random sampling when the initial training set is large, at which point random sampling may have little to no effect.
- Our method has the largest relative advantage on random sampling when the augmentation is small, highlighting its potential when the labeling resources are scarce.
- When the initial training set is very small, the difference between our method and random sampling is smaller, and it may be best to use random sampling first when *very few* instances are labeled already, but to switch to active learning once a bedrock set of labeled instances is available.

5.2 Experiment 3

Comparing Batch and Incremental Augmentations

One may be worried that by adding instances in batches using our active learning method, we are not reaping the full benefits that would come with an incremental process where the instance labeling priorities are reevaluated after each instance is labeled. This experiment

will assess whether or not that is the case by comparing the performance obtained when adding instances incrementally versus in batches.

5.2.1 Procedure

This experiment's procedure is very similar to that of Experiment 1 (described in subsection 4.1.1):

1. An initial set of labeled instances is used to induce a classification model. Because we are using a one-class classifier, only the majority class instances are used in this process.

2. The resulting classifier is evaluated on a set of instances held out from the training set.

3. The resulting classifier is used, along with the selection criterion, to select a *batch* of instances from a large pool of unlabeled instances.

4. The selected instances (called the *augmentation*) are labeled, and the minority class instances are discarded. The majority class instances are incorporated to the initial training set to form the batch training set.

5. The batch training set is used to induce a second classification model.

6. The second classifier is evaluated on the same set of instances that were used when evaluating the initial classifier.

7. A third dataset, the incremental training set, is built incrementally by following these steps until it reaches the same size as the batch training set:

(a) The current version of the model (starting with the classifier built in step 2) is used, along with the selection criterion, to select a *single* instance from a large pool of unlabeled instances.

(b) That instance is labeled and incorporated to the current version of the incremental training set.

(c) The current version of the model is refreshed using the now slightly larger incremental training set.

8. The third (incremental) classifier is evaluated on the same set of instances that were used when evaluating the first and second classifiers.

9. The performances of all three classifiers are recorded. The first is recorded as the "Initial" result, the second as "Batch" and the third as "Incremental".

Steps 3 through 6 are executed multiple times, in parallel: once for each of the compared alternatives for selection criteria. These alternatives are described in subsection 4.1.2.

5.2.2 Evaluation Methodology

We followed the same evaluation methodology for this experiment as we did for Experiment 1. See subsection 4.1.3.

5.2.3 Datasets Used

The *Pen Digits* and *MAGIC Gamma Telescope* datasets were used. We tried running the experiment with the Saanich dataset, but rebuilding the model after adding a single instance made the experiment too computationally expensive.

5.2.4 Results

Tables 5.2a, and 5.2b present the results obtained with, respectively, the *Pen Digits* and *MAGIC Gamma Telescope* datasets. The measurements shown are the area under the ROC curve obtained by varying the threshold at which instances are determined to belong to one class or the other. We report the average measurement over 10 folds of cross-validation, for the classifier trained on the initial training set, on the training set augmented in a single batch, and on the training set augmented. The following subsections give an overview of statistical tests on these results.

These results show no significant difference between the *Batch* and *Incremental* treatments.

5.2.5 Discussion

The results obtained are encouraging: They suggest that, at least on *some* domains, batch active learning, which is computationally much cheaper than the incremental alternative, may not be much worse in terms of the performance benefits it effects, if it is worse at all. In fact, the incremental variant perform marginally better on the *Pen Digits* dataset, but *worse* than batch on the *MAGIC Gamma Telescope* dataset.

This conclusion, however, should be taken with a grain of salt: Although we can, using statistical test, establish a *difference* between two or more treatments, we cannot show that there is no difference between two treatments. Additionally, the results we obtained may be specific to the domains we used, and perhaps incremental active learning may be stronger on other domains.

(a) **Dataset: Pen Digits**

Fold	Initial	Batch	Incremental
1	0.9913	0.9950	0.9950
2	0.9908	0.9932	0.9935
3	0.9928	0.9962	0.9964
4	0.9854	0.9931	0.9931
5	0.9830	0.9904	0.9903
6	0.9893	0.9952	0.9951
7	0.9846	0.9886	0.9885
8	0.9844	0.9948	0.9946
9	0.9899	0.9968	0.9971
10	0.9847	0.9918	0.9918
Avg.	*0.9876*	0.9935	0.9935
S.D.	0.0035	0.0026	0.0027

(b) **Dataset: MAGIC Gamma Telescope**

Fold	Initial	Batch	Incremental
1	0.7595	0.7707	0.7688
2	0.7653	0.7745	0.7723
3	0.7715	0.7786	0.7770
4	0.7752	0.7817	0.7815
5	0.7645	0.7713	0.7713
6	0.7675	0.7754	0.7746
7	0.7620	0.7749	0.7719
8	0.7788	0.7872	0.7860
9	0.7856	0.7900	0.7904
10	0.7781	0.7843	0.7823
Avg.	*0.7708*	0.7789	0.7776
S.D.	0.0085	0.0067	0.0071

Table 5.2: Results of Experiment 3 (AUROC)

5.3 Chapter Overview

In summary, we described, in this chapter, our investigation into the conditions that lead our method to perform at its best. In Experiment 2, we determined that our method brings the largest benefits relative to the naïve alternative of random sampling when *a*) we already have a moderately large set of labeled examples, in which case random sampling usually fails to effectively identify the rare examples in the unlabeled pool that could still contribute to the dataset by the variety they bring and *b*) we can only afford to label a few more examples and need to be very selective, in which case random sampling wastes its time labeling instances of little value whereas our method effectively identifies key instances. In Experiment 3, we assessed the relative benefit of performing active learning incrementally versus in batches. Our results are encouraging as they suggest that not much if any performance gains are lost when adding instances in batches, a process which is typically much cheaper to implement.

Chapter 6

Analyzing the Method's Impact

Before concluding this thesis, there is one more aspect of the proposed method we will investigate: the type of instances that get selected by our method, and their impact on the training set. This will help us understand *why* our method helps us achieve better performance than the alternatives. The first two experiments in this chapter look at subconcepts within the majority class and within-class imbalance: First, we assess whether or not our method helps to rebalance imbalanced subconcepts within the majority class, then compare that effect with what we might get from cluster-based oversampling [Jo and Japkowicz, 2004]. The last experiment of this chapter, and of this thesis, is specific to the one-class SVM implementation: We determine whether or not our method really identifies instances that end up serving as support vectors, a possible explanation for the performance gains.

6.1 Experiment 4

Minority Subconcepts in Augmentations

In this experiment, we assess the impact of small majority subconcepts on classification performance and investigate characteristics of small subconcepts that may give our proposed method an edge in their identification.

6.1.1 Procedure

- Run through the steps below for each of 10 folds of cross-validation:

    1. Use one tenth of the dataset as the test set and the rest as the training set.
    2. In the training set, undersample each of the 7 subconcepts so that they are represented by, respectively, 0, 3, 10, 30, 93, 290 and 900 instances each.
    3. Build a one-class classifier (one-class SVM) on this undersampled training data.
    4. Use the one-class classifier to score/rank each of the instances in the test set by how likely they are to belong to the majority class.
    5. Using the scores of a single of the 7 majority subconcepts in the test set, along with the scores of the minority class' test data, measure the AUROC. This is a measure of the ability of the classifier to distinguish that particular subconcept from the minority class.
    6. Rank the majority class' test instances based on the scores from step 3 and calculate the average rank of each subconcept's instances.
    7. Normalize the average ranks to the $(0, 1]$ range by dividing them by the maximum rank.

8. In this same ranking, count how many instances of each majority subconcept appear in the top 100 instances most likely to belong to the minority class ($1/7^{th}$ of the majority test set).

- Record the average measurements obtained in steps 5, 7, and 8 over the 10 folds.
- Repeat the entire process 10 times, for 10x10-fold cross-validation. This yields 10 measurements for each of the following:
  - average AUROC on each subconcept
  - average rank of instances in each subconcept
  - prevalence of each subconcept in top 100

6.1.2 Datasets Used

We limit this experiment to the *Pen Digits* dataset, as the low variance in the results obtained when using it may allow us to more easily distinguish general trends in the impact of training set and augmentation sizes. In addition, the *Pen Digits* dataset makes it easy for us to undersample subconcepts of the classes we form, allowing us to run this experiment.

From the 10 original classes that form the *Pen Digits* dataset, we merge 7 to form a majority class and 3 to form a minority class. The classes from the original data set form "subconcepts" of each class.

6.1.3 Interpreting the Results

Before the experiments were run, we developed the following guidelines on how the results obtained should be interpreted:

If the classifier does not make more mistakes on small subconcepts, then the AUROCs for all of the subconcepts should be about the same. If they are not, and the AUROCs for distinguishing small subconcepts from the minority class are weaker, then the classifier does indeed make more mistake on instances from subconcepts which it has seen less frequently during training.

If instances in the small subconcepts are, on average, ranked first, then that indicates that the model does tend to think they look more like the minority class than instances from the larger subconcepts, a possible cause for misclassifications.

If instances in the top 100 ranked tend to belong to the small subconcepts in larger proportions than they belong to the larger subconcepts, then our active learning selection criterion will tend to select these instances more frequently, increasing their prevalence in later iterations of the model.

6.1.4 Results

The results of this experiment are described in Table 6.1. In each of the subtables, the columns each correspond to a subconcept, and the column header indicates the size of that subconcept.

First, Subtable 6.1a presents the average AUROCs obtained when attempting to distinguish each subcluster's instances from the minority class. Second, Subtable 6.1b presents the average rank, relative to the maximum rank, where instances belonging to each subconcept placed. Third, Subtable 6.1c presents the average proportion of the top 100 ranked instances (instances which, to the classifier, appear to belong to the minority class) that were of each subconcept.

(a) **AUROC**

	Subcluster Size						
Iteration	0	3	10	30	93	290	900
1	0.2400	0.6911	0.9771	0.9930	0.9905	0.9975	0.9997
2	0.7706	0.6692	0.9353	0.9923	0.9977	0.9988	0.9994
3	0.1916	0.6486	0.8996	0.9589	0.9968	0.9980	0.9990
4	0.8783	0.5204	0.8554	0.9610	0.9908	0.9992	0.9990
5	0.3304	0.8550	0.5209	0.8306	0.9685	0.9884	0.9998
6	0.1727	0.4427	0.6477	0.4629	0.7674	0.9540	0.9730
7	0.3205	0.8146	0.9283	0.9227	0.9076	0.9851	0.9945
8	0.8818	0.6799	0.9537	0.9783	0.9856	0.9765	0.9947
9	0.7096	0.9262	0.9034	0.9582	0.9912	0.9954	0.9917
10	0.8931	0.9451	0.9875	0.9814	0.9949	0.9989	0.9994
Avg.	0.5388	0.7193	0.8609	0.9039	0.9591	0.9892	0.9950

(b) **Average rank (relative to maximum rank)**

	Subcluster Size						
Iteration	0	3	10	30	93	290	900
1	0.0864	0.2292	0.4448	0.6045	0.6006	0.7103	0.8292
2	0.1838	0.1916	0.3493	0.5718	0.7045	0.7036	0.8006
3	0.0765	0.2410	0.4075	0.5070	0.6974	0.7896	0.7860
4	0.2683	0.1172	0.3341	0.5200	0.6298	0.7947	0.8410
5	0.1185	0.4175	0.2372	0.4546	0.6717	0.7383	0.8673
6	0.1472	0.3259	0.4746	0.3473	0.5807	0.7907	0.8386
7	0.1048	0.3140	0.4727	0.5553	0.4878	0.7115	0.8588
8	0.2142	0.1523	0.4228	0.6040	0.6870	0.6254	0.7993
9	0.1116	0.3438	0.3373	0.5078	0.7060	0.7688	0.7297
10	0.1727	0.2411	0.4547	0.4370	0.6207	0.7566	0.8221
Avg.	0.1484	0.2574	0.3935	0.5109	0.6386	0.7390	0.8173

(c) **Prevalence in first 100 (Percentage)**

	Subcluster Size						
Iteration	0	3	10	30	93	290	900
1	0.8200	0.1780	0.0000	0.0020	0.0000	0.0000	0.0000
2	0.3910	0.4900	0.1090	0.0080	0.0020	0.0000	0.0000
3	0.9070	0.0600	0.0290	0.0030	0.0000	0.0010	0.0000
4	0.0700	0.6910	0.1960	0.0380	0.0040	0.0000	0.0010
5	0.6470	0.0430	0.2210	0.0620	0.0250	0.0020	0.0000
6	0.6280	0.1610	0.0400	0.1330	0.0280	0.0070	0.0030
7	0.7180	0.1160	0.0370	0.0580	0.0590	0.0090	0.0030
8	0.2330	0.6150	0.0660	0.0390	0.0150	0.0260	0.0060
9	0.7090	0.0720	0.1450	0.0590	0.0080	0.0020	0.0050
10	0.5290	0.3360	0.0350	0.0810	0.0170	0.0010	0.0010
Avg.	0.5652	0.2762	0.0878	0.0483	0.0158	0.0048	0.0019

Table 6.1: Results of Experiment 4 by Subcluster Size

6.1.5 Discussion

Our interpretation of these results follows the guidelines outlined in subsection 6.1.3. The results support our hypotheses that:

- The one-class classifier tends to commit dramatically more misclassifications on instances belonging to the small subconcepts of the majority class, as shown by low AUROC scores.
- The one-class classifier tends to judge instances belonging to the small subconcepts of the majority class as more likely to belong to the minority class.
- The top 100 ($1/7^{th}$ of the test set) instances by rank belong overwhelmingly often to the small subconcepts, suggesting that, since it is those instances that our active learning selection criterion would pick first, our proposed method may indirectly oversample the small subconcepts.

These results have another interesting consequence: One of the subconcepts in the dataset was *unseen*, in the sense that it was represented by no instances at all in the training set. For obvious reasons, this subconcept was virtually indistinguishable from the minority class to the eyes of the one-class classifier, and appeared very frequently in the top 100 list. Because of this, instances belonging to this unseen subconcept are very likely to be selected by our selection criterion. This suggests that our method may have one key advantage over many other methods: It inherently leads to the discovery of previously unseen subconcepts.

It is in light of this evidence that the relevance of Experiment 5 becomes apparent: If our method leads to an indirect oversampling of the *small* subconcepts of the majority class, then perhaps we should compare it to methods that do this directly.

6.2 Experiment 5

Comparing to Cluster-Based Oversampling

Given the results of Experiment 4, we now compare our proposed method's ability to identify instances belonging to small majority subconcepts to that of a method designed to do just that: cluster-based oversampling [Jo and Japkowicz, 2004].

6.2.1 Procedure

- Run through the steps below for each of 10 folds of cross-validation:

    1. Use one tenth of the dataset as the test set, 3 tenths as a pool of examples from which the active learning technique can pick, and the rest as the training set.
    2. In the training set, undersample each of the 7 subconcepts so that they are represented by, respectively, 0, 3, 7, 17, 45, 116, and 300 instances each.
    3. Build a one-class classifier (one-class SVM) on this undersampled training data.
    4. Evaluate this one-class classifier on the test set and record the AUROC as "Initial".
    5. Use the one-class classifier to score/rank each of the instances in the unlabeled pool by how likely they are to belong to the minority class and select 100 instances to form an active learning augmentation.
        (a) Build a one-class classifier (one-class SVM) on the training set augmented by these 100 instances.
        (b) Evaluate this one-class classifier on the test set and record the AUROC as "Our Method".
    6. Use cluster-based oversampling to add 100 instances.

(a) Build a one-class classifier (one-class SVM) on the training set augmented by these 100 instances.

(b) Evaluate this one-class classifier on the test set and record the AUROC as "Cluster-based oversampling".

- Record the average measurements obtained in steps 4, 5 and 6 over the 10 folds.
- Repeat the entire process 10 times, for 10x10-fold cross-validation. This yields 10 AUROC measurements for each of the three treatments.

6.2.2 Compared Alternatives

We compare two alternatives:

- The method proposed in this thesis, given its supposed ability to find small subconcept data
- Cluster-based oversampling [Jo and Japkowicz, 2004], a method which consists of clustering the data of a class, then oversampling cluster-by-cluster. By sampling more from small clusters, one can lessen the impact of within-class imbalance.

6.2.3 Evaluation Methodology

10x10-fold cross-validation was used to estimate the performance of each of the two methods. The AUROC was used as the performance metric.

6.2.4 Datasets Used

We use the same dataset as in Experiment 4, described in Subsection 6.1.2.

6.2.5 Results

The results of this experiment are described in Table 6.2. In each of the subtables, the columns each correspond to a subconcept, and the column header indicates the size of that subconcept.

Subtable 6.2a corresponds to the results obtained with the initial training set, Subtable 6.2bto the results obtained with our method, and 6.2c to the results obtained when the dataset was balanced using cluster-based oversampling.

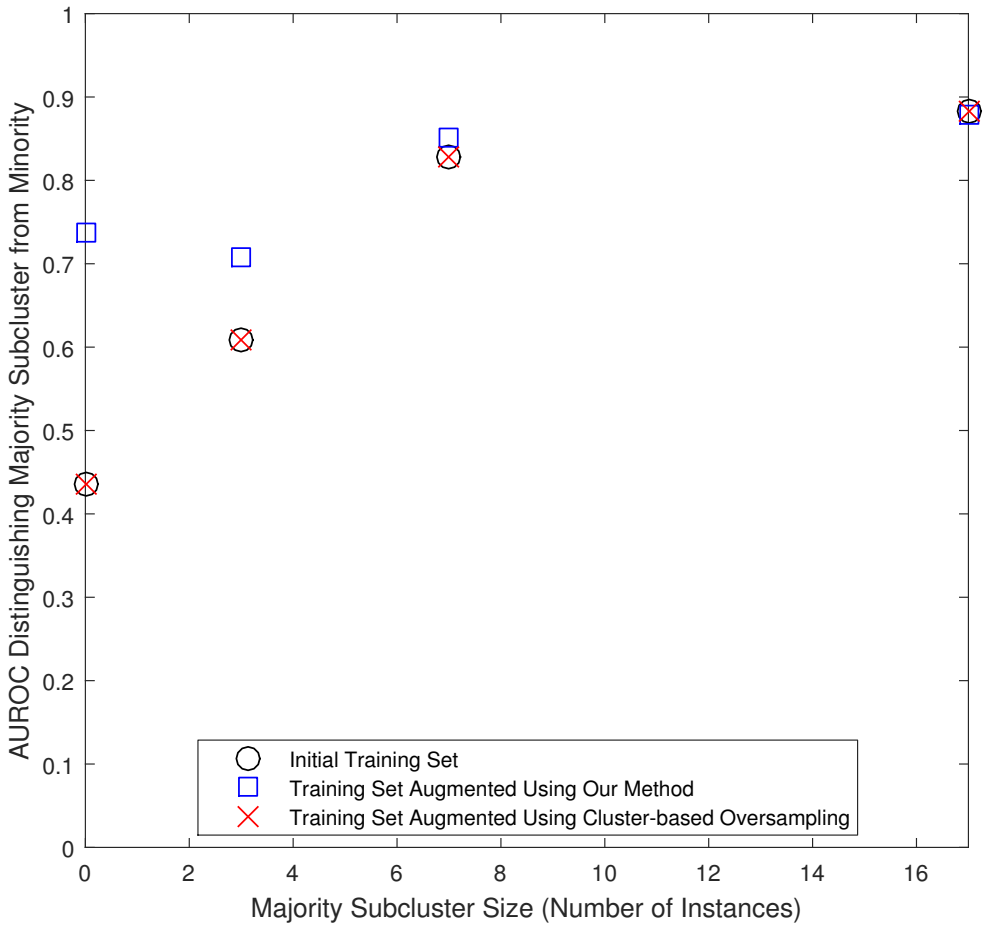


Figure 6.1: AUROCs distinguishing majority subclusters of different sizes from the minority class

The results for the first 4 columns of the 3 subtables are shown in Figure 6.1. We left out the other columns because they show little difference between treatments.

(a) **Initial Training Set**

	Subcluster Size						
Iteration	0	3	7	17	45	116	300
1	0.2627	0.2537	0.7732	0.9628	0.9197	0.9830	0.9830
2	0.1398	0.6792	0.8630	0.5717	0.9166	0.9889	0.9896
3	0.2874	0.2371	0.6832	0.8527	0.8710	0.8722	0.9557
4	0.4622	0.7108	0.9523	0.9266	0.9844	0.9901	0.9850
5	0.8927	0.8001	0.6201	0.8547	0.9751	0.9953	0.9966
6	0.2110	0.2008	0.6858	0.8331	0.8785	0.9107	0.9858
7	0.4191	0.5940	0.9769	0.9082	0.9943	0.9970	0.9952
8	0.7816	0.9906	0.9076	0.9898	0.9932	0.9993	1.0000
9	0.5362	0.8198	0.9025	0.9928	0.9660	0.9710	0.9950
10	0.3680	0.8025	0.9255	0.9400	0.9402	0.9668	0.9961
Avg.	0.4361	0.6088	0.8290	0.8832	0.9439	0.9674	0.9882

(b) **Training Set Augmented Using Our Method**

	Subcluster Size						
Iteration	0	3	7	17	45	116	300
1	0.1590	0.5691	0.7485	0.9593	0.9100	0.9812	0.9812
2	0.6773	0.6655	0.8537	0.5816	0.9118	0.9883	0.9890
3	0.5681	0.4171	0.6829	0.8527	0.8711	0.8717	0.9555
4	0.8894	0.7536	0.9522	0.9283	0.9843	0.9907	0.9850
5	0.8884	0.8418	0.8292	0.9038	0.9756	0.9952	0.9966
6	0.4493	0.5171	0.7133	0.8344	0.8789	0.9142	0.9859
7	0.8669	0.5800	0.9365	0.7926	0.9854	0.9895	0.9903
8	0.9710	0.9789	0.9307	0.9816	0.9872	0.9978	0.9997
9	0.9688	0.9166	0.9298	0.9926	0.9850	0.9723	0.9948
10	0.9387	0.8383	0.9400	0.9625	0.9464	0.9625	0.9958
Avg.	0.7377	0.7078	0.8517	0.8789	0.9436	0.9663	0.9874

(c) **Training Set Augmented Using Cluster-based Oversampling**

	Subcluster Size						
Iteration	0	3	7	17	45	116	300
1	0.2627	0.2537	0.7732	0.9628	0.9196	0.9830	0.9829
2	0.1398	0.6792	0.8630	0.5717	0.9166	0.9888	0.9896
3	0.2875	0.2370	0.6833	0.8529	0.8710	0.8722	0.9558
4	0.4622	0.7107	0.9522	0.9266	0.9844	0.9901	0.9850
5	0.8927	0.8000	0.6200	0.8547	0.9751	0.9953	0.9967
6	0.2108	0.2007	0.6858	0.8328	0.8784	0.9107	0.9858
7	0.4191	0.5940	0.9768	0.9082	0.9943	0.9970	0.9952
8	0.7816	0.9906	0.9076	0.9898	0.9932	0.9993	1.0000
9	0.5364	0.8200	0.9025	0.9928	0.9660	0.9710	0.9950
10	0.3680	0.8024	0.9255	0.9400	0.9401	0.9668	0.9961
Avg.	0.4361	0.6088	0.8290	0.8832	0.9439	0.9674	0.9882

Table 6.2: Results of Experiment 5 (AUROC) by Method Used

6.2.6 Discussion

When reading the results of Table 6.2, it is important to keep in mind that the goal is not to establish whether or not our method works, which was the goal of Chapter 4, but instead to determine *how* it achieves the performance improvements we noticed in our previous experiments by looking at its impact on subclusters of different sizes.

First and foremost, we observe that, as hoped, our method leads to important performance improvements on the small subconcepts of the majority class. This may be an explanation for the performance gains we see at large when using our method. Conversely, we see performance losses on the larger subconcepts. This is not entirely surprising, as we do not expect many instances of large subconcepts to be selected by our method; their prevalence in the dataset should therefore have decreased, affecting performance.

Although the results we obtained on small subclusters are promising for our method, we feel they do not do cluster-based oversampling justice. In the experiment, we limited the number of instances active learning and cluster-based oversampling could add to the training set. This was to analyze the impact of each method on training set *quality* rather than quantity. However, in an applied setting, cluster-based oversampling would allow us to add many more instances than we could ask human experts to label (for active learning). It could be because we limited the number of instances CBO could add to 100 that it did not have a noticeable effect on performance.

We believe, however, that there are reasonable advantages to our method over CBO. First, our method is capable of finding instances belonging to unseen subconcepts. In the experiment, we achieved an AUROC of *0.7377* on these subconcepts after using active learning, whereas that AUROC was at *0.4361* when using the initial training set. Second, when some subconcepts are very small, they may be more difficult to identify for clustering algorithms, which could mistake them for noise. This does not happen with our method, as

there is no notion of clusters in the method itself.

There is one more thing we believe is worth noticing. In Figure 6.1, we see that, with our method, we performed slightly better on the unseen subconcept (0 instances) than on the second smallest subconcept (2 instances) after applying active learning. This may appear surprising, but we believe this is an example of the impact of batch active learning: Our method most likely selected many instances belonging to the 0-instances subconcept, more than the number of instances it selected belonging to the 2-instances subconcept. This lead to a bigger surge in performance on the 0-instances subconcept than on the 2-instances subconcept. If we had performed active learning incrementally, perhaps we would have seen closer performance between these 2 subconcepts, as the first few instances selected would have affected later selections.

6.3 Experiment 6

Prevalence of Support Vectors in Augmentations

In this final experiment, our goal is to determine whether or not instances selected by our active learning method have a higher than usual chance of being used as support vectors. If so, then this would be evidence that our active learning method helps to refine the decision boundary by offering valuable support vector candidates.

6.3.1 Procedure

- Run through the steps below for each of 10 folds of cross-validation:

    1. Use one tenth of the dataset as the test set, 3 tenths as a pool of examples from which the active learning technique can pick, and the rest as the training set.

2. Build a one-class classifier (one-class SVM) on this undersampled training data.
3. Use the one-class classifier to score/rank each of the instances in the unlabeled pool by how likely they are to belong to the minority class and select 100 instances to form an active learning augmentation.
4. Build a second one-class classifier (one-class SVM) on the training set augmented by these 100 instances.
5. Obtain the list of support vectors of this second classifier and count:
   - The number of support vectors which were part of the initial training set
   - The number of support vectors which were part of the augmentation
   - The overall number of support vectors
6. From these values and the total sizes of the initial training set and augmentation, calculate:
   - The fraction of instances in the initial training set that were used as support vectors
   - The fraction of instances in the augmentation that were used as support vectors
   - The fraction of instances at large that were used as support vectors

- Record the average measurements obtained in steps 5 and 6 over the 10 folds.
- Repeat the entire process 10 times, for 10x10-fold cross-validation. This yields 10 values for each of the measurements from steps 5 and 6.

6.3.2 Datasets Used

As in all other experiments in this chapter, we use the *Pen Digits* dataset. However, this time, we do not perform any undersampling on the subconcepts of the majority class. We

(a) **Absolute number of support vectors by type of instance**

Fold	Initial Set	Augmentation	Overall
1	258	65	323
2	199	68	267
3	202	68	270
4	217	70	287
5	184	72	256
6	284	68	352
7	175	56	231
8	177	71	248
9	252	60	312
10	208	77	285
Avg.	215	67	282
Out of	488	100	588

(b) **Fraction of instances that are support vectors by type of instance**

Fold	Initial Set	Augmentation	Overall
1	0.5289	0.6510	0.5497
2	0.4082	0.6810	0.4546
3	0.4145	0.6790	0.4595
4	0.4443	0.6950	0.4869
5	0.3777	0.7220	0.4362
6	0.5811	0.6810	0.5981
7	0.3576	0.5560	0.3913
8	0.3623	0.7060	0.4207
9	0.5156	0.6020	0.5303
10	0.4256	0.7690	0.4840
Avg.	0.4416	0.6742	0.4811

Table 6.3: Proportion of instances that are support vectors by instance type

keep the distribution of the dataset untouched.

6.3.3 Results

The results of this experiment are described in Table 6.3. Subtables 6.3a and 6.3b respectively give the absolute and relative number of instances which became support vectors, divided by type of instance: *a*) instances that were part of the initial training set, *b*) instances that were part of the augmentation, and *c*) instances in general (combining the other two).

6.3.4 Discussion

The experimental results we obtained support our hypothesis that instances selected by our method are likely to become support vectors. Indeed, an instance that was part of the active learning augmentation was, on average, 60% more likely to become a support vector than an instance that was part of the initial training set.

These results are not entirely surprising: Given the performance gains we saw when using our active learning method with one-class support vector machines, it was obvious that the one-class SVM was not reusing the exact same support vectors and that the augmentations were providing a little extra.

It is nevertheless reassuring to see our hypothesis supported by empirical evidence. These results deepen our understanding of what is truly happening behind the scenes when using our active learning method.

6.4 Chapter Overview

This penultimate chapter's goal was to provide insight into the inner workings of our method by looking at the type of instances that are selected by our method and their impact on the resulting classifiers. There were two main directions of enquiry:

- The relation between the instances our method selects and small subconcepts of the majority class
- The relation between the instances our method selects and instances that are used as support vectors by the one-class SVM

Experiments 4 and 5 addressed the first of these two elements. In Experiment 4, we provided evidence that demonstrates that instances belonging to small subconcepts of the majority class are at the source of more misclassifications than instances belonging to large subconcepts. That experiment's results also suggested that instances belonging to small subconcepts of the majority class appear more *like* the minority class to our one-class classifiers and that our selection criterion is likely to select many such instances. With this result in mind, we further inspected our method's capacity to identify instances belonging to small majority subconcepts by comparing it to cluster-based oversampling. Here, our experimental results confirmed that our method inherently leads to a rebalancing of the majority subconcepts, to the extent that such a rebalancing leads to performance gains. Our results also demonstrated how our method may be able to identify instances belonging to previously unseen subconcepts, something that oversampling methods typically cannot do.

Finally, Experiment 6 addressed the nature of support vectors when active learning is used. In this endeavour, we found that, as expected, instances selected by our method tend to be chosen more often than other instances as support vectors, suggesting that our method is particularly apt at finding *useful* instances.

Chapter 7

Conclusion

7.1 Summary

Throughout this thesis, we explored the intersection of three areas of research in machine learning, namely the problem with highly imbalanced data, one-class classification and one-class learners, and active learning.

We noticed that, although there had already been research on *a*) one-class classifiers applied to highly imbalanced data, *b*) active learning for imbalanced data, and *c*) active learning for one-class classifiers,none of the solutions we found in the literature properly addressed the problem we described: How should instances be prioritized for labeling when we are using a one-class learner, learning from the majority class only, and the minority class is particularly rare? Indeed, most of the active learning methods that had been proposed for one-class classifiers in the past made unnecessary assumptions about the prevalence of minority class data, or the representativeness of the available data.

The main contribution of this thesis was to propose a method that aims to fill this gap.

This method is not specific to any implementation of one-class classifiers, and does not rely on finding minority data. Although we offer specific implementations for certain classifiers in Section 3.2, we believe it is the general principle of our method that is most important: label first instances which the latest iteration of a classification model judges to be most likely to belong to the minority class.

The remainder of the thesis consisted of experiments that aimed to provide insight into the method we proposed. First, we established our method's ability to effect performance improvements in classifiers by adding informative instances to their datasets.

Then, in Chapter 5, we analyzed the type of situations in which our method worked best and found that it has the largest advantage, relative to naïvely labeling instances at random, in two situations: *a*) when the dataset is already large, and other methods fail to effectively identify useful instances, and *b*) when labeling costs are high and we can only afford to add a few. We also found that, although in theory incremental active learning, where one instance is added at a time and labeling priorities are recalculated between instances, should lead to better performance than adding many instances in a batch, in our experiments, the difference between batch and incremental active learning was not noticeable, a result that should be encouraging to those worried about the computational costs of active learning.

In Chapter 6, we looked at the type of instances that are selected by our method and what they become. Our results, in this case, suggested that our method may, as a side-effect, discover instances belonging to underrepresented subconcepts of the majority class, or even previously unseen such subconcepts. When compared to a method designed for this specific purpose, cluster-based oversampling, our method appeared to have a much easier time finding instances of the smallest subconcepts, and was the only of the two methods to find instances belonging to the unseen subconcepts.

Finally, we investigated the fate of instances selected by our method. We found that, when using one-class support vector machines, these instances have a probability of being used as support vectors that is much higher than usual, suggesting that our method is particularly apt at finding good candidates for support vectors, a possible explanation for the performance gains it can bring.

7.2 Future Work

Although we had many opportunities to experiment with the proposed method throughout this thesis, the method is still in its infancy. We would like to see more work using our method, reproducing our results and, hopefully, bringing more evidence to support our claims. Ideally, our method's merits should be assessed in more varied conditions such as, for example, domains with lower, or perhaps even higher, levels of imbalance. In particular, we believe it would be interesting to evaluate our method on domains where the other active learning methods discussed in Chapter 2 have proven their worth to determine whether our method does or does not prevail, and why that is.

In fact, while our experimental results have answered many questions, they raise many more. In Chapter 5, we found that our method makes the most difference in very particular conditions: when we either already have quite a lot of labeled instances, in which case our method finds the few remaining instances that can make a difference, and when we can only add a few more, in which case our method makes the best out of limited means and maximizes the benefit of these few added instances. However, we also found that if the initial dataset is very small, then it may not be worth to use our method just yet, as an initial set of "foundation" instances may be necessary to kickstart our method. It would be interesting to investigate the properties of domains that influence the number of instances that are necessary to form this "foundation" set. For instance, it may be that

fewer instances are needed on dense domains than on sparse domains, or on domains with more subconcepts, because the first few instances may be sufficient to represent the core of a dense domain.

More practically, it would be interesting to see if there is a way to determine when we have enough instances in our initial training set to use the proposed method. If it is preferable to first sample at random to build a small "foundation" set of instances, and then to switch to our method, is there any way to tell when we have amassed enough such instances? Perhaps a solution would be to look at a curve of performance improvements by instances added and to switch to our method when the improvements begin to plateau. It is possible, however, that such a solution would not harness the full benefits of our method. It would be interesting to delve deeper into this question and to look at different methods of determining when to use our method versus random sampling, and their impact on the profile of performance improvements.

Similarly, one could investigate methods of determining when it is no longer worth it to continue labeling instances. An obvious contender, given the results of Chapter 5, is to look at the profile of performance improvements by instances added, and to stop labeling when the curve has reached a plateau. This, however, may not work for all domains. It could happen that, in some domains unlike the ones we examined in this thesis, there would be pockets of very valuable instances that our method wouldn't be inclined to find until later. We believe that evaluating our method on more domains, and looking at the performance profiles, might help shed light on this issue and, hopefully, provide insight into methods for determining when labeling should stop.

In Chapter 5, we also looked at incremental active learning, and found encouraging results that suggested that labeling instances in batches may not be much worse than re-evaluating labeling priorities after each instance. However, some of our results from Chapter 6 bring an interesting alternative point of view: When studying the impact of

our method on very small and unseen subconcepts, and adding instances in batches that contained more instances than the smallest of subconcepts, we found that after an iteration of active learning, we had found almost exclusively instances belonging to the unseen subconcepts, which had surpassed the small subconcepts in prevalence. If incremental active learning was used, instead, we could expect to see the unseen subconcept catch up to the smallest subconcept, and then to see them grow "together", at a similar rate. It may be in cases like these that shine the merits of performing active learning incrementally; another question that we have left, for now, unanswered.

In addition, it would be interesting to see more analysis into the types of instances that form a dataset and how varying active learning selection criteria prioritize them. Although our method seems apt at finding instances belonging to small subconcepts, which, when added to the dataset, seem to be beneficial to the resulting classifier, there may be other types of instances that our method systematically misses because of the bias of its selection criterion.

Finally, it could be interesting to analyze the overlap between the instances that are chosen by our method when using different base one-class classifiers (e.g. one-class support vector machines, or the distance to KNN classifier). This may provide insight into the examples that each of these methods tend to misclassify and could help us devise strategies to avoid these mistakes. In addition, this might help us determine the extent to which the instances selected by our method are truly informative about the domain (in which case all classifiers would benefit from their presence in the training set), versus being instances that correct only classifier-specific weaknesses in the dataset, such as areas of the dataset that are commonly missed by support vectors but not by the distance to KNN classifier or vice versa.

References

[Abe et al., 2006] Abe, N., Zadrozny, B., and Langford, J. (2006). Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509. ACM.

[Akbani et al., 2004] Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer.

[Attenberg and Provost, 2010] Attenberg, J. and Provost, F. (2010). Why label when you can search?: alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 423–432. ACM.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository.

[Bellinger et al., 2012] Bellinger, C., Sharma, S., and Japkowicz, N. (2012). One-class versus binary classification: Which and when? In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 102–106. IEEE.

[Bloodgood and Vijay-Shanker, 2009] Bloodgood, M. and Vijay-Shanker, K. (2009). Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 137–140. Association for Computational Linguistics.

[Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

[Caruana and Niculescu-Mizil, 2004] Caruana, R. and Niculescu-Mizil, A. (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357.

[Cohen et al., 2006] Cohen, G., Hilario, M., Sax, H., Hugonnet, S., and Geissbuhler, A. (2006). Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37(1):7–18.

[Corder and Foreman, 2014] Corder, G. W. and Foreman, D. I. (2014). *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons.

[Dagan and Engelson, 1995] Dagan, I. and Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157.

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). Pattern classification (pt. 1).

[Elkan, 2001] Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Citeseer.

[Ertekin et al., 2007] Ertekin, S., Huang, J., Bottou, L., and Giles, L. (2007). Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM.

[Estabrooks et al., 2004] Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36.

[Fan et al., 1999] Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105.

[Fawcett and Provost, 1997] Fawcett, T. and Provost, F. (1997). Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316.

[Ferri et al., 2009] Ferri, C., Hernández-Orallo, J., and Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

[Freund et al., 1996] Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156.

[Fujii et al., 1998] Fujii, A., Tokunaga, T., Inui, K., and Tanaka, H. (1998). Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597.

[Ghasemi et al., 2011a] Ghasemi, A., Manzuri, M. T., Rabiee, H. R., Rohban, M. H., and Haghiri, S. (2011a). Active one-class learning by kernel density estimation. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE.

[Ghasemi et al., 2011b] Ghasemi, A., Rabiee, H. R., Fadaee, M., Manzuri, M. T., and Rohban, M. H. (2011b). Active learning from positive and unlabeled data. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 244–250. IEEE.

[Görnitz et al., 2009] Görnitz, N., Kloft, M., and Brefeld, U. (2009). Active and semi-supervised data domain description. In *Machine Learning and Knowledge Discovery in Databases*, pages 407–422. Springer.

[Guo and Greiner, 2007] Guo, Y. and Greiner, R. (2007). Optimistic active-learning using mutual information. In *IJCAI*, volume 7, pages 823–829.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer.

[He et al., 2008] He, H., Bai, Y., Garcia, E., Li, S., et al. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

[He et al., 2006] He, J., Li, M., Zhang, H.-J., Tong, H., and Zhang, C. (2006). Generalized manifold-ranking-based image retrieval. *Image Processing, IEEE Transactions on*, 15(10):3170–3177.

[Hempstalk et al., 2008] Hempstalk, K., Frank, E., and Witten, I. H. (2008). One-class classification by combining density and class probability estimation. In *Machine Learning and Knowledge Discovery in Databases*, pages 505–519. Springer.

[Japkowicz, 2001a] Japkowicz, N. (2001a). Concept-learning in the presence of between-class and within-class imbalances. In *Advances in Artificial Intelligence*, pages 67–77. Springer.

[Japkowicz, 2001b] Japkowicz, N. (2001b). Supervised versus unsupervised binary-learning by feedforward neural networks. *Machine Learning*, 42(1-2):97–122.

[Japkowicz, 2013] Japkowicz, N. (2013). Assessment metrics for imbalanced learning. In He, H. and Ma, Y., editors, *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 187–206. Wiley-IEEE Press.

[Japkowicz et al., 2000] Japkowicz, N. et al. (2000). Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA.

[Japkowicz et al., 1995] Japkowicz, N., Myers, C., Gluck, M., et al. (1995). A novelty detection approach to classification. In *IJCAI*, pages 518–523.

[Japkowicz and Shah, 2011] Japkowicz, N. and Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

[Jo and Japkowicz, 2004] Jo, T. and Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49.

[Kang and Cho, 2006] Kang, P. and Cho, S. (2006). Eus svms: Ensemble of under-sampled svms for data imbalance problems. In *Neural Information Processing*, pages 837–846. Springer.

[Kubat et al., 1998] Kubat, M., Holte, R. C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215.

[Kukar et al., 1998] Kukar, M., Kononenko, I., et al. (1998). Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449. Citeseer.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, pages 79–86.

[Laurikkala, 2001] Laurikkala, J. (2001). *Improving identification of difficult small classes by balancing class distribution*. Springer.

[Lewis and Catlett, 1994] Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*, pages 148–156.

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

[Lindenbaum et al., 2004] Lindenbaum, M., Markovitch, S., and Rusakov, D. (2004). Selective sampling for nearest neighbor classifiers. *Machine learning*, 54(2):125–152.

[Liu et al., 2009] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(2):539–550.

[Liu and Zhou, 2006] Liu, X.-Y. and Zhou, Z.-H. (2006). The influence of class imbalance on cost-sensitive learning: An empirical study. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 970–974. IEEE.

[Liu et al., 2006] Liu, Y., An, A., and Huang, X. (2006). Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *Advances in Knowledge Discovery and Data Mining*, pages 107–118. Springer.

[Mahalanobis, 1936] Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55.

[Maloof, 2003] Maloof, M. A. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1.

[Mamitsuka, 1998] Mamitsuka, N. A. H. (1998). Query learning strategies using boosting and bagging. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*, page 1. Morgan Kaufmann Pub.

[Manevitz and Yousef, 2002] Manevitz, L. M. and Yousef, M. (2002). One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154.

[Mani and Zhang, 2003] Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of Workshop on Learning from Imbalanced Datasets*.

[Mann and Whitney, 1947] Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

[McCarthy et al., 2005] McCarthy, K., Zabar, B., and Weiss, G. (2005). Does cost-sensitive learning beat sampling for classifying rare classes? In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 69–77. ACM.

[McNemar, 1947] McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artificial intelligence*, 18(2):203–226.

[Moskovitch et al., 2007] Moskovitch, R., Nissim, N., Stopel, D., Feher, C., Englert, R., and Elovici, Y. (2007). Improving the detection of unknown computer worms activity using active learning. In *KI 2007: Advances in Artificial Intelligence*, pages 489–493. Springer.

[Nemenyi, 1962] Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210.

[Nigam and McCallum, 1998] Nigam, K. and McCallum, A. (1998). Employing em in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*.

[Roy and McCallum, 2001] Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

[Scheffer et al., 2001] Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*, pages 309–318. Springer.

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

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

[Settles and Craven, 2008] Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

[Settles et al., 008b] Settles, B., Craven, M., and Ray, S. (2008b). Multiple-instance active learning. *Advances in Neural Information Processing Systems (NIPS)*, 20:1289–1296.

[Seung et al., 1992] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

[Shannon, 1948] Shannon, C. (1948). A mathematical theory of distribution. *Bell Syst Techn*, 27:623.

[Sun et al., 2007] Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

[Tax and Duin, 2004] Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine learning*, 54(1):45–66.

[Tomanek and Hahn, 2009] Tomanek, K. and Hahn, U. (2009). Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the fifth international conference on Knowledge capture*, pages 105–112. ACM.

[Tong and Koller, 2002] Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

[Tukey, 1949] Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114.

[Wang and Japkowicz, 2004] Wang, B. and Japkowicz, N. (2004). Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, page 19.

[Wang and Japkowicz, 2010] Wang, B. X. and Japkowicz, N. (2010). Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, 25(1):1–20.

[Weiss, 1995] Weiss, G. M. (1995). Learning with rare cases and small disjuncts. In *ICML*, pages 558–565.

[Weiss, 2013] Weiss, G. M. (2013). Foundations of imbalanced learning. *H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 13–41.

[Weiss and Provost, 2001] Weiss, G. M. and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. *Rutgers Univ.*

[Weiss and Provost, 2003] Weiss, G. M. and Provost, F. (2003). Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, pages 315–354.

[Zhou and Liu, 2006] Zhou, Z.-H. and Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):63–77.

[Zhu et al., 2003] Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, pages 58–65.