IDENTIFYING PCB CONTAMINATED TRANSFORMERS THROUGH ACTIVE LEARNING

by

Yin Chu Yeh
B.Sc., University of British Columbia, 2010

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2012

APPROVAL

Name: Yin Chu Yeh

Degree: Master of Science

Title of Thesis: Identifying PCB Contaminated Transformers through Active Learning

Examining Committee: Dr. Oliver Schulte
Chair

Dr. Ke Wang, Senior Supervisor

Dr. Martin Ester, Supervisor

Dr. Jian Pei, Internal Examiner

Date Approved: August 27, 2012

Partial Copyright Licence

Abstract

Exposure to polychlorinated biphenyals (PCBs) is hazardous to human health. The United Nations Environment Programme has decreed that nations, including Canada and the US, must eliminate PCB contaminated utility equipment such as transformers by 2025. Sampling, which imposes a non-trivial expenditure, is required to confirm the PCB content of a transformer. For the first time, we apply an iterative machine learning technique known as active learning to construct a PCB transformer identification model that aims to minimize the number of transformers sampled and thus reduce the total cost. In this thesis, we propose a dynamic sampling size algorithm to address two key issues in active learning: the sampling size per iteration and the stopping criterion. The proposed algorithm is evaluated using the real world datasets from BC Hydro in Canada.

Acknowledgments

Most of all, I would like to thank my senior supervisor, Dr. Ke Wang, for his invaluable guidance and time devoted to helping me complete this thesis. I would like to express my gratitude to Dr. Wenyuan Li and Dr. Adriel Lau from BC Hydro for their generous support and suggestions.

I would also like to thank Dr. Martin Ester, Dr. Jian Pei, and Dr. Oliver Schulte for generously allocating their time to help me fulfill my graduate requirements.

To my former and current labmates: Chao Han, Peng Wang, Bo Hu, Zhihui Guo, Zhensong Qian, Yongmin Yan, Ryan Shea, Samaneh Moghaddam, Moshen Jamali, Elaheh Kamaliha and Iman Sarrafi, thank you for your constant support and encouragements, and for making graduate studies an endearing experience.

Last but very importantly, I would like to give a big, special thanks to my family and David for their endless love, support and encouragement!

Contents

Approval ii

Abstract iii

Acknowledgments iv

Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

1.1 Problem Background 1

1.2 Problem Requirement 2

1.3 Contributions 3

2 Related Work 6

3 Active Learning 10

3.1 Sampling and Classification 11

3.2 Batch Size and Stopping Criterion 12

4 Cost Model 16

4.1 True Cost 17

4.2 Estimated Cost 19

5 Our Focus **20**
5.1 Learning with the Right Batch Size 20
5.2 Stopping at the Minimal Costing Iteration 21

6 GDB And Stopping Criterion **23**
6.1 Geometric Dynamic Batch Size (An Approach For ***BatchSize***) 23
6.2 Stopping criterion (An Approach For ***SC***) 26

7 Experiment **27**
7.1 Setup 27
7.2 Replace Missing Values 30
7.2.1 Effectiveness of the Estimated Cost 30
7.2.2 Cost Reduction 32
7.2.3 Accuracy 34
7.2.4 Summary 37
7.3 Remove Records With Missing Values 38
7.3.1 Cost Reduction 38
7.3.2 Accuracy 39
7.3.3 Summary 41

8 Conclusions **42**

Bibliography **44**

List of Tables

4.1 Notations at Iteration i 17
4.2 Cost Matrix C 17

7.1 BC Hydro Bushing Data Attributes 28
7.2 Experiment Settings 30
7.3 Replace Missing Values - % of True Cost Reduction for Tests Described in Table 7.2 33
7.4 Replace Missing Values - $|L|$, Recall (R) and Precision P for Tests Described in Table 7.2 36
7.5 Remove Records with Missing Values: % of True Cost Reduction for Tests Described in Table 7.2 39
7.6 Remove Records with Missing Values: $|L|$, Recall (R) and Precision P for Tests Described in Table 7.2 40

List of Figures

4.1 True Cost Learning Curve. 18

5.1 AL batch size performance comparison. 21
5.2 True Cost Learning Curve with $b = 5\%$. 22

7.1 True and Estimated Cost Curves (Test 11) 31

Chapter 1

Introduction

Polychlorinated biphenals (PCBs) based dielectric fluid were widely used for heat insulation purposes in electrical equipment such as transformers for many years up until the 1980s. It has been shown that the exposure to PCB is hazardous to human health and environments [2]. However, many PCB containing transformers are still being used, as the average lifespan of a power transformer is 50 years or even longer. The United Nations Environment Programme (UNEP) has decreed that nations, including Canada and the US, must eliminate all PCB containing equipment by 2025 [25, 1]. This policy imposes an immense challenge for power companies across the world. The goal of this thesis is to provide a strategy to identify PCB contaminated transformers, which aims to jointly minimize the costs spent on verifying the content of transformers and the consequences of unidentified PCB transformers.

1.1 Problem Background

High voltage power transformers are used by power companies to distribute and transfer electrical energy from an incoming voltage to different outgoing voltage levels. During the transformation, energy is lost in the form of heat due to resistance. To avoid the transformers from overheating, PCB dielectric fluids were often embedded in transformers for cooling and insulation purposes up until the 1980s [25]. Unfortunately, the exposure to PCBs is hazardous to human health and environments [2]: PCBs can lead to kidney failure and failure of other human organs, produce headaches and sickness, cause chlor-acne if absorbed through the skin, and cause cancer in animals. PCBs also do not readily break down in the environment and thus may remain there for a very long period of time.

The main challenge to PCB transformer removal is that though many transformers have been labeled to contain PCBs, transformers that were sold as non-PCB may still contain traces of PCBs due to cross-contamination. This is because facilities used for filling transformers often employ both PCB and non-PCB oil mixtures. In some European countries, the cross-contamination rate can be as high as 45 percent [25]. On the other hand, it is financially impossible to simply replace all potentially contaminated transformers, as a transformer could cost upwards of a million dollars.

A more cost-effective solution is to identify the PCB contaminated transformers and replace the contaminated oil mixture or parts. Due to the cross-contamination mentioned above, it is necessary to sample oil mixtures of transformers for PCB verification. However, most of the later designs have the PCB fluids placed in a hermetically sealed structure called a bushing with no external access (e.g., drainage valve) because of their stable and not easily degradable characteristics. Thus, oil sampling requires breaking the sealed structure, which usually costs approximately 10% (or even more) of the total cost of a transformer. Other costs associated with sampling include shutting down power transmission of involved areas, replacing the current transformers with substitutes, and any risk factors involved in sampling. For these reasons, the size sampled should be kept at a minimum.

A transformer has multiple bushings, which may or may not have the same PCB content. Therefore, more often the PCB identification is done at the bushing level. The PCB identification problem also applies to other equipment that potentially contains PCB contaminated oil, such as capacitors. For ease of exposition, we shall consider transformer in our discussion, but the same discussion can be applied to other equipment with the similar problem.

In the remainder of this thesis, the term sampling is used to represent the physical procedure required to obtain the PCB content of a transformer. The term labeled transformer is used to present that a transformer has been sampled and verified as either PCB contaminated (positive (+)) or non-PCB contaminated (negative (−)), whereas the term unlabeled transformer will be used to represent transformers whose true PCB status is unknown.

1.2 Problem Requirement

Given a set of transformers, most or all of which are unlabeled, we want to predict the label of all unlabeled transformers through a sampling process. Since sampling the PCB

status of all transformers is very costly, we are interested in a solution where we only need to sample a subset of unlabeled transformers and use the labeled transformers to build a classifier to predict the PCB status of the remaining unlabeled transformers. The goal is to minimize the sum of the cost from sampling described above and the classification cost due to prediction errors described below.

Classification of an unlabeled transformer produces one of two types of errors. A *false positive error* refers to misclassifying a truly non-PCB contaminated transformer as a PCB contaminated transformer, thereby incurring the unnecessary sampling costs due to parts (such as the bushing) being broken and other damages. Similarly, a *false negative error* refers to misclassifying a truly PCB contaminated transformer as a non-PCB contaminated transformer, thereby incurring the consequence of leaving a PCB transformer unidentified, such as putting human health and the environment at risk. Each of these two types of errors incurs some cost and the cost is imbalanced in that the false negative cost is usually significantly larger than the false positive cost. In this thesis, we assume that sampling cost, false positive cost, and false negative cost per transformer can be computed and compared on a single numerical scale.

At the heart of the problem is the trade-off between the cost induced from sampling and the classification costs: sampling more transformers will incur a higher sampling cost, but will provide more PCB information for building a more accurate classifier for remaining unlabeled transformers, which reduces the classification cost. Our task is to develop a computerized strategy for building a classifier so that the total cost of sampling and classification is minimized. To achieve this goal, the key is to perform the oil sampling on a small subset of transformers that are representative of the remaining unlabeled transformers. The challenge is to identify such representative transformers at a small sampling cost.

1.3 Contributions

Though specifications on how to identify a PCB contaminated transformer through testing/sampling of its fluids have been well established [25], to the best of our knowledge, no research has been done on applying machine learning techniques to help identify PCB contaminated transformers. The current approach of identifying PCB contaminated transformers is by testing every transformer possible; however, this process is slow and financially infeasible.

Active learning (AL) is a well known machine learning technique that instead of mass-sampling once to obtain a training set for building the classification model, iteratively samples only a small number of transformers at a time and slowly builds up the labeled training set. The intuition being we can gradually gain knowledge on the pool of transformers so that at every iteration, we can make smarter sampling choices. That is, this gradual intellectual sampling selection can help us reduce the training set size by avoiding sampling instances with overlapping information and concentrating on more questionable and/or indicative transformers.

The main contribution of this thesis is casting the PCB transformer identification as an AL problem and addressing several challenges in the PCB transformer context:

1. Most AL based methods consider the cost in terms of the number of samples and the number of misclassifications. However, as explained above, the PCB identification problem has a novel cost structure defined in terms of sampling cost, false positive error cost, and false negative error cost. We formulated the problem into a cost model to take into account such different costs in the performance evaluation of an active learner.

2. The traditional AL method samples a fixed number of instances per iteration, denoted as the *batch size*, and stops the learning process using ad hoc stopping criteria. The imbalanced penalty for false negative and false positive errors and far fewer PCB transformers than non-PCB transformers lead to less stable performance of the classifier, which makes the choice of batch sampling size and stopping criteria more important. We discuss a dynamic batch size scheme and a complementary stopping criterion to better address these features in the PCB problem.

3. We apply and evaluate various AL stopping criteria using a real life dataset provided by BC Hydro, an electric power utility company in Canada. From this study, we provide a guideline for choosing AL methods and stopping criteria based on the structure of cost models and the distribution of PCB transformers. The results show that the proposed batch size adjustment and stopping criterion produce favourable results for an imbalanced cost model and an imbalanced distribution of PCB transformers.

The rest of this thesis is organized as follows. Chapter 2 performs a study on existing related work and Chapter 3 presents an overview of AL. Chapter 4 generalizes the PCB

transformer identification problem into a cost sensitive learning model. Chapter 5 considers the issues of stopping criterion and batch size, and Chapter 6 proposes a batch size adjustment algorithm and its complementary stopping criterion. Chapter 7 studies the performance of various solutions on real life utility transformer bushing data.

Chapter 2

Related Work

There are three types of AL querying scenarios: pool-based sampling, stream-based sampling, and membership query synthesis (as described by Settles [29]). The pool-based scenario [30, 5, 34, 28, 21, 36] has the active learner iteratively learning and querying from a pool that does not change as the learning process proceeds, i.e., a fixed set of instances. The stream-based sampling [24, 10, 31, 11, 12] sees unlabeled instances one at a time and makes decisions on whether to query or discard the instance one at a time. Once an instance is discarded, it will never be reconsidered for sampling again for stream-based sampling. The membership query synthesis [3, 19, 9] allows the active learner itself to think of a possible example input that it believes to be the most interesting to know the label of, and asks the user to label such an input. For example, in the classical handwritten digit recognition classification problem, the membership query synthesis may construct an image that it may not have seen before, and requests the user to provide the label, i.e., 0-9, of the given image. Our PCB transformer identification problem falls under the pool-based AL category.

To the best of our knowledge, current existing pool-based active learners and stopping criteria utilize a single fixed batch size to learn throughout the whole learning process [15, 17, 18, 38]. This can be inefficient as we will discuss in Section 5.1 the pros and cons of having smaller or larger batch sizes in various stages of an active learning process. A dynamic batch size active learner for stream-based scenarios has been proposed by Chakraborty et al. [6]. Differing from our problem, in a stream-based AL problem, new unseen data continue to arrive and they must determine the number of instances to sample from these newly arrived data. For our problem, all data (i.e., transformers), labeled or not, are available from the beginning; and we iteratively draw a subset of unlabeled data to sample and the

pool of data (labeled and unlabeled as a whole) do not change.

Since the usual goal of AL is to minimize the number of sampled data, this suggests that the iterative sampling process for pool based AL should most likely stop before we exhaustively sample all instances, i.e., transformers. However, it is challenging to determine an appropriate stopping criterion since at any given iteration, it is impossible to know whether it is more beneficial to stop now or to continue to the next iteration as we do not know the labels of the currently unsampled data, and have little idea how they will affect the model at hand.

Various stopping criteria for active learning have been proposed based on observing the confidence of the active learner. The maximum confidence, overall confidence, minimum expected error and selected accuracy based stopping criteria [39, 40, 41] are all examples of the confidence based criterion. The basic idea of these confidence based stopping criteria is to stop the active learner when some quantity (such as entropy, error or accuracy) achieves a certain predefined threshold. A notable drawback of these mentioned confidence based stopping criteria is that they require an appropriate predefined threshold to have desirable performance. In extreme cases, the predefined threshold may never be reached. Moreover, for a sparse dataset, in early iterations when very few or no minority class instances are sampled, the model is likely to achieve the threshold very quickly, since there are very few minority instances and the model may confidently classify all instances, including all the minority classed instances, as the majority class and stop prematurely. This can lead to severe problem if misclassifying a minority class is extremely expensive, such as in our PCB transformer identification problem.

To alleviate the threshold problem, Zhu et al [41] proposed a strategy that dynamically updates the threshold based on the labeling changes as iterations proceed. They suggest that even when the threshold has been reached, it is possible that the learner has not yet stabilized and it is possible that the model may change dramatically in the next iteration. Hence, they proposed to improve the threshold if the previous threshold has been reached and there exists an unlabeled instance $u \in U$ such that the current classification of u disagrees with the previous classification. The active learner will terminate only when the threshold is reached and the model has stabilized (i.e., classification of all unlabeled instances have not changed between the two latest iterations). Though this method effectively alleviates the threshold problem, the danger of stopping the active learner prematurely still remains – especially for extremely imbalanced datasets. It is quite possible that very little or no

minority class instances are seen in early iterations, in which case, the confidence of the learner is likely well above the threshold and the classification decision may not change dramatically as very few minority instances may be sampled in one iteration. The batch size adjustment algorithm we propose in Chapter 6 alleviates this problem by ensuring that the targeted goal has been observed for at least some number of iterations to ensure the desired target is actually achieved and not just noise.

Another confidence based stopping criteria proposed by Vlachos [35] uses an unlabeled test set, that is separate from the pool of instances where the active learner learns and selects from, to calculate the uncertainty and stops the active learner when the confidence based on the test set peaks (i.e., performance begins to degrade). One problem to consider when applying this approach is that not all datasets conform to a peak pattern in its learning progress and may fail to terminate [41].

Similarly, Laws and Schütze [20] proposed to stop the active learning process when the performance (i.e., confidence), of the learner converges and the gradient of the performance curve approaches 0. Differing from the strategy proposed by Vlachos [35], the gradient based stopping criterion does not require the performance curve to have peaked before stopping. However, the key drawback is that to ensure the gradient is reliable and resilient to noise, it requires an appropriate and large enough window size. However, having a large window size to help confirm that the active learner has stabilized can cause the learner to sample more instances than necessary, especially when the batch size is large and non-adjustable.

In addition to the confidence based stopping criteria, Bloodgood and Vijay-Shanker [4] proposed a stability based stopping criteria. Bloodgood and Vijay-Shanker consider stability as a necessary factor to determine when the active learner should terminate. However, it does not utilize other threshold based stopping criteria and instead, makes a stopping decision based on the stability of the learner over some number of consecutive iterations. This stability based stopping criterion also faces the challenge of needing to determine an appropriate window size.

Tomanek et al. [33] proposed a stopping criterion for query-by-committee based active learners. They suggested monitoring the average classification disagreement among different classifiers between iterations and stopping when the number of disagreements is close to zero. In situations where the construction of a classifier is expensive, committee based active learning is less suitable. The advantage of our approach is that we can apply our proposed batch size adjustment algorithm and stopping criterion to various existing probabilistic

active learners and not be confined to query-by-committee based active learners.

Chapter 3

Active Learning

AL is an iterative process where at each iteration, the learner requests for a specific set of unlabeled data to be labeled in an attempt to build a better classifier. At iteration i, let $L(i)$ and $U(i)$ be the labeled transformers and unlabeled transformers. We build a classifier, $M(i)$, using $L(i)$ to classify the transformers in $U(i)$; request for some subset of $U(i)$, denoted by $S(i)$ for PCB sampling; and append $S(i)$ into $L(i)$. The size of the sample set, $|S(i)| = b(i)$, is called the batch size. This interactive sampling process continues until some stopping criterion, $SC(i)$, is satisfied. Algorithm AL shows the general procedure of a pool-based AL algorithm. Note that we assume $L(0)$ is small, but non-empty. Random sampling may be used to provide $L(0)$ if it is initially empty.

Algorithm AL has four major components that require elaboration:

1. SC: tests whether the iterative process should stop
2. $Classify$: classifies unlabeled transformers
3. $Sample$: picks the transformers to sample
4. $BatchSize$: determines the batch size

Research has focused on finding strategies to choose the data to sample ($Sample$) and to classify the unlabeled data by building conventional classifiers such as a support vector machines (SVMs) or decision trees ($Classify$) (see [9]). In this thesis, we focus on determining the batch size ($BatchSize$) and stopping criteria (SC). However, we will now provide the backgrounds of some existing algorithms we employ in our AL implementation for solving the PCB transformer identification problem.

Algorithm AL

Input: Initial set of labeled, $L(0)$, and unlabeled, $U(0)$, data and initial batch size $b(0)$
Output: A classification model

$M(0)$ = Initial classifier trained using $L(0)$
$i = 0$
// iterative procedure
while ***SC***(i) is not satisfied **do**
  // classify $U(i)$
  $UC(i) =$ ***Classify***$(M(i), U(i))$
  // select and sample $b(i)$ instances in $U(i)$
  $S(i) =$ ***Sample***$(b(i), U(i), UC(i))$
  $U(i+1) = U(i) - S(i)$
  $L(i+1) = L(i) \cup S(i)$
  // prepare for the next iteration
  $i = i + 1$
  $M(i)$ = trained classifier using $L(i)$
  $b(i) =$ ***BatchSize***()
end while
return $M(i)$

3.1 Sampling and Classification

In this thesis we use the SVM based AL algorithm proposed by Brinker [5] for selecting a batch of diverse transformers to sample (*Sample* function). SVMs are well established binary classifiers, and [5] selects instances that the classifier is most uncertain about while ensuring the selected transformers are diverse to avoid sampling transformers with highly overlapping information.

Additionally, we apply Bayes optimal prediction strategy to transform the classification result probabilities from the SVM into a cost-sensitive classifier ($Classify$ function). That is, instead of using the label from the SVM classifier directly, we will label the data point to minimize expected cost. Since the goal of our problem is to minimize both sampling and classification cost, it is best to classify the transformer to be the class that has minimum expected cost to lower the expected classification cost. Let the probability for a given transformer u to be of class y be denoted $p(y|u)$. The Bayes optimal prediction for u is the class x that minimizes the conditional risk [13]:

$$R(x|u) = \sum_{y} p(y|u)C(x,y) \tag{3.1}$$

where $C(x, y)$ is the cost matrix that defines the consequence of predicting a transformer of class y as class x, and $R(x|u)$ is the expected risk/cost of predicting that u belongs to class x. The real conditional probability $p(y|u)$ is almost always unknown, so we use the classifier learnt to provide the probability estimate [22]. Although traditional SVM does not provide such posterior probability, [26] and [7] can be incorporated to provide probability estimates for SVM classification. Using Bayes optimal prediction, we set:

$$u\text{'s class} = \underset{x}{\operatorname{argmin}}\, R\left(x|u\right) \tag{3.2}$$

3.2 Batch Size and Stopping Criterion

To the best of our knowledge, all existing active learning algorithms learn with a constant batch size, i.e., the *BatchSize* function is a constant. Though in situations where the number of allowed iterations is unbounded, sampling one instance at a time can be ideal as the learning model is updated the most often. However, in cases such as when updating the model is expensive and can take a while, sampling only one instance per iteration may not be optimal as it may take far too long for the active learning process to be completed within a reasonable time. As mentioned previously, sampling the PCB content of a transformer entails additional costs such as shutting down power transmission of the involved area, which makes the sampling process inefficient if we shut down an area just to sample one transformer. Hence, it is important to identify and have the active learner learn with appropriate batch sizes. In Section 5.1, we will study how batch size may affect the learning process.

Below are five conventional stopping criteria that we will later implement and evaluate in our experiments. Methods 1) – 4) are from [41] and method 5) is from [4]. Note that these works were presented for general AL purpose, not specifically for PCB transformer identification.

1. *Maximum Uncertainty Method (MU)*: The Maximum Uncertainty (MU) based stopping criterion terminates the active learner when the uncertainty measurement, UM, of all unlabeled instances, denoted as U, fall under some threshold θ_{MU}. The following is the MU stopping criterion as defined by Zhu and Hovy:

$$SC_{MU} = \ true \quad if \quad \forall u \in U, UM(u) \leq \theta_{\text{MU}} \tag{3.3}$$

Where entropy is used as the uncertainty measure as shown in Equation 3.4, $p(y|u)$ is the probability of an instance (i.e., transformer), u to be of class y, and Y is the set of all classes (i.e., $+, -$ in our problem).

$$UM(u) = -\sum_{y \in Y} p\left(y|u\right) \log p\left(y|u\right) \tag{3.4}$$

2. *Overall Uncertainty Method (OU)*: The Overall Uncertainty (OU) based stopping criterion terminates the active learner when the average of the uncertainty measurement, UM, over all unlabeled instances fall under some threshold θ_{OU}. The following is the OU strategy as defined by Zhu and Hovy:

$$SC_{OU} = \ true \quad if \quad \frac{\sum_{\forall u \in U} UM(u)}{|U|} \leq \theta_{\mathrm{OU}} \tag{3.5}$$

Where $UM(u)$ is calculated as described in Equation 3.4.

3. *Minimum Expected Error Method (MEE)*: Minimum Expected Error (MEE) is a stopping criterion that terminates the active learner when the expected error falls under some predefined threshold θ_{MEE}. The MEE stopping criterion is defined by Zhu and Hovy as follows:

$$SC_{MEE} = \ true \quad if \quad Error(M) \leq \theta_{\mathrm{MEE}} \tag{3.6}$$

$Error(M)$ [27] is the expected error of classifier M calculated using Equation 3.7. M is constructed using all labeled data available. The active learning process terminates when the SC_{MEE} is satisfied.

$$Error(M) = \frac{1}{|U|} \sum_{u \in U} \left(1 - \max_{y \in Y} p(y|u)\right) \tag{3.7}$$

4. *Selected Accuracy Method (SA)*: In batch mode active learning, where we sample b unlabeled instances in every iteration, the selected accuracy (SA) based stopping strategy utilizes these b newly sampled instances to determine if an active learner should terminate. The basic idea is to make a stopping decision based on how accurately the current classifier can predict the labels of these newly sampled data points. The accuracy of a classifier M on these b selected instances, denoted as $Accuracy_{\mathrm{b}}(M)$, is a function calculated using the sampling feedback of these b instances provided by the experts. The SA strategy is described by Zhu and Hovy as follows:

$$SC_{SA} = \ true \quad if \quad Accuracy_{\mathrm{b}}(M) \geq \theta_{\mathrm{SA}} \tag{3.8}$$

Where θ_{SA} is some predefined threshold, and $Accuracy_{\text{b}}(M)$ is defined as follows with TP_b and TP_b be the true positives and true negatives of instances in b, as classified by M:

$$Accuracy_{\text{b}}(M) = \frac{|TP_b| + |TN_b|}{|b|} \tag{3.9}$$

Since b is the newly sampled set of instances, the true labels of instances in b are available for computing TP_b and TN_b. Note that Zhu et. al [41] did not explicitly define the $Accuracy_{\text{b}}(M)$ function; we use Equation 3.9 as it is the conventional function used to calculate accuracy.

Currently the MU, OU, MEE and SA described all require a predetermined threshold value. Zhu et al. proposed a threshold update strategy to alleviate the problem of needing the user to be able to provide an appropriate threshold value, θ, before the learning process begins. The idea is to use a loose threshold value initially and the algorithm will adjust the threshold of the stopping criterion based on monitoring the stability of the learner. The concept of stability described in [41] is that if there is any classification change to the remaining unlabeled instances during two most recent learning iterations, the learning process is deemed to be unstable. When the threshold value is achieved, but the learning process is not yet stable, the threshold will be revised and the active learning process proceeds. The learning process will only terminate when the threshold value and stability requirement are both satisfied.

5 *Stabilizing Predictions Method (SP)*: The basic idea of the Stability Prediction (SP) based stopping criterion is to terminate the iterative process when the class predictions on a set of instances (labeled or not) called the stop set has stabilized over some m number of iterations, called the window. When the AL process has stabilized, sampling more instances will not likely change nor further improve the model, and should be terminated to avoid additional sampling costs. Stability is measured based on what is called the *agreement*, as described in [4]. Bloodgood and Vijay-Shanker uses the Kappa statistics [8] to measure the *agreement* between two consecutive iterations based on the observed agreement (A_o) on the classification of the stop set, and the agreement expected by chance (A_e) as described in Equation 3.10.

$$agreement = \frac{A_o - A_e}{1 - A_e} \tag{3.10}$$

and A_e is calculated as follows [4]:

$$A_e = \sum_{y \in Y} p(y|M(i)) \cdot P(y|M(i+1)) \tag{3.11}$$

where $p(y|M(j))$ is the probability that classifier $M(j)$ classifies an instance as y. This probability is measured based on the proportion of instances in the stop set that $M(j)$ classifies as y [4]. The active learning process will terminate when the average of the agreements from a window of the m most recent iterations achieves a certain predefined threshold θ_{SP}. In general, three parameters need to be predetermined in order to utilize the SP strategy: a stop set, a threshold agreement θ_{SP} and a window size m.

These stopping criteria are based on some notion of confidence and classification stability of classifiers. The advantage of these stopping criteria is that they can be freely applied to various batch mode learning models such as SVM and decision tree based algorithms. Unlike the conventional confidence and stability based stopping goals, recall the goal of our PCB identification problem is to minimize a notion of cost that integrates the penalties on sampling, false positive errors, and false negative errors. We go into further detail in the next section.

Chapter 4

Cost Model

Conventional metrics such as accuracy and precision in the number of correctly identified transformers cannot accurately evaluate the performance of a PCB identification model as they do not differentiate the consequences incurred from false positive and false negative misclassification. For example, if 99% of the transformers are non-PCB contaminated, by simply classifying all transformers as non-PCB contaminated, we can achieve 99% accuracy in terms of number only. However, the damage cost to environment caused by leaving those 1% contaminated transformers unidentified may still be too high to be borne for utility and society. In some other cases, a less accurate number in percentage may be acceptable if the consequence from misclassification is not high. In this chapter, we present a cost model for evaluating the classifier constructed at each iteration.

We continue using the notations from Chapter 3 and let $TP(i)$, $FP(i)$, $TN(i)$, $FN(i)$ be the numbers of true positive, false positive, true negative and false negative transformers in $U(i)$ as determined by the classifier $M(i)$ respectively. The notations are also summarized in Table 4.1. Note that in a real world situation, these subsets are unknown because the ground truth labels for the transformers in $U(i)$ are unknown. In Section 4.1, we assume this information has been omnisciently available for calculating the true cost. In Section 4.2, we introduce a method of estimating the true cost, assuming that the labels of $U(i)$ are not available.

Table 4.1: Notations at Iteration i

NOTATION	DEFINITION
$M(i)$	classifier built using $L(i)$
$L(i)$	labeled transformers at iteration i
$L_y(i)$	labeled transformers whose actual label is $y \in \{+, -\}$
$U(i)$	unlabeled transformers at iteration i
$U_y(i)$	unlabeled transformers in $U(i)$ that $M(i)$ classifies as class y.
$TP(i)$	true positive transformers in $U(i)$ as determined by the classifier $M(i)$
$FP(i)$	false positive transformers in $U(i)$ as determined by the classifier $M(i)$
$TN(i)$	true negative transformers in $U(i)$ as determined by the classifier $M(i)$
$FN(i)$	false negative transformers in $U(i)$ as determined by the classifier $M(i)$

We summarize the cost matrix in Table 4.2. Each entry $C(x, y)$ in the table represents the cost of classifying a transformer with the (actual) label y as the (predicted) label x. $C(-,-)$ represents the cost of classifying a PCB negative transformer as PCB negative, which is zero. $C(+,-)$ represents the cost from classifying a PCB negative transformer as PCB positive (i.e., the false positive cost), and $C(+,+)$ represents the cost from classifying a PCB positive transformer as PCB positive. When a transformer is classified as positive, there will be the cost of sampling the transformer, denoted by $cost_{\text{Sample}}$. $C(-,+)$ represents the cost of classifying a PCB positive transformer as PCB negative (i.e., the false negative cost). This cost is denoted by $cost_{\text{FN}}$.

Table 4.2: Cost Matrix C

	ACTUAL −	ACTUAL +
PREDICTED −	$C(-,-) = 0$	$C(-,+) = cost_{\text{FN}}$
PREDICTED +	$C(+,-) = cost_{\text{Sample}}$	$C(+,+) = cost_{\text{Sample}}$

4.1 True Cost

The cost of the model built at any iteration i comprises of the classification cost of $U(i)$ and the sampling cost involved from acquiring the labels of the training set $L(i)$.

Classification Costs

Each false negative error incurs cost $cost_{\text{FN}}$. Each true positive or false positive incurs $cost_{\text{Sample}}$. Note $U_+(i) = TP(i) \cup FP(i)$. The classification cost is calculated as follows:

$$\begin{aligned} Cost_{\text{Class}}(i) =& |TN(i)| \cdot C(-,-) + |FN(i)| \cdot C(-,+) \\ &+ |TP(i)| \cdot C(+,+) + |FP(i)| \cdot C(+,-) \\ =& |FN(i)| \cdot cost_{\text{FN}} + |U_+(i)| \cdot cost_{\text{Sample}} \end{aligned} \tag{4.1}$$

Labeling Costs

Every labeled transformer in $L(i)$ contributes to a sampling cost, $cost_{\text{Sample}}$, as the label of a transformer can only be obtained through sampling. Moreover, for each sampling process an overhead cost (e.g., shutting down the power of an area) denoted as $cost_{\text{Iter}}$ is entailed. Hence, the total labeling cost required to build the AL model at iteration i is:

$$Cost_{\text{Labeling}}(i) = |L(i)| \cdot cost_{\text{Sample}} + i \cdot cost_{\text{Iter}} \tag{4.2}$$

assuming $cost_{\text{Iter}}$ is a constant value provided by the user.

Therefore, at any arbitrary iteration i, if we terminate the AL process at iteration i, the cost will be the sum of classification (Equation 4.1) and labeling (Equation 4.2) costs.

$$\begin{aligned} Cost(i) =& Cost_{\text{Class}}(i) + Cost_{\text{Labeling}}(i) \\ =& |FN(i)| \cdot cost_{\text{FN}} + i \cdot cost_{\text{Iter}} + (|L(i)| + |U_+(i)|) \cdot cost_{\text{Sample}} \end{aligned} \tag{4.3}$$

Figure 4.1 shows the performance curve of a sample active learning process calculated using Equation 4.3. The curve is produced using BC Hydro bushing dataset, which we will discuss in detail in Chapter 7.

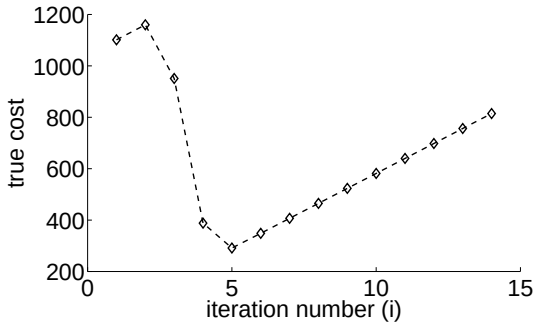


Figure 4.1: True Cost Learning Curve.

The labeled dataset becomes larger as the number of iteration increases. As we can see from Figure 4.1, initially when the labeled set is small (at the low number of iterations), the true cost is high due to high misclassification, but decreases as the model improves from the increasing labeled dataset. However, after a sufficient amount of sampling, the true cost reaches a minimum point and then begins to increase with the number of iterations because the labeling cost increases but the additional samples no longer reduce misclassification. This shows that Equation 4.3 indeed reflects the trade-off between sampling and classification.

4.2 Estimated Cost

The true cost in Equation 4.3 depends on $|FN(i)|$,$|L(i)|$ and $|U_+(i)|$. Unfortunately, $|FN(i)|$ is unknown because the actual labels of the transformers in $U(i)$ are unknown. Note that $|L(i)|$ and $|U_+(i)|$ are known since we know exactly how many transformers the learner has sampled in total (i.e., line 10 from Algorithm AL), and which unlabeled transformers are classified by $M(i)$ as positive regardless of what their actual labels are. Hence, the only unknown term in Equation 4.3 that requires estimation is $|FN(i)|$. To estimate $|FN(i)|$, we consider the probability that a transformer is positive when it is classified as negative by $M(i)$. Let $p_i(+|u)$ be the posterior probability of a transformer $u \in U_-(i)$ being PCB contaminated given by $M(i)$ assuming all $p_i\,(+|u)$ are independent and the classifier can provide the probability estimate. Then the number of false negatives of iteration i is estimated by

$$|FN(i)|_{\text{Est}} = \sum_{u \in U_-(i)} p_i\,(+|u) \tag{4.4}$$

So we estimate the cost in Equation 4.3 as follows:

$$Cost_{\text{Est}}(i) = |FN(i)|_{\text{Est}} \cdot cost_{\text{FN}} + i \cdot cost_{\text{Iter}} + (|L(i)| + |U_+(i)|) \cdot cost_{\text{Sample}} \tag{4.5}$$

Hereafter, we refer to the cost in Equation 4.3 as the true cost, and refer to the cost in Equation 4.5 as the estimated cost.

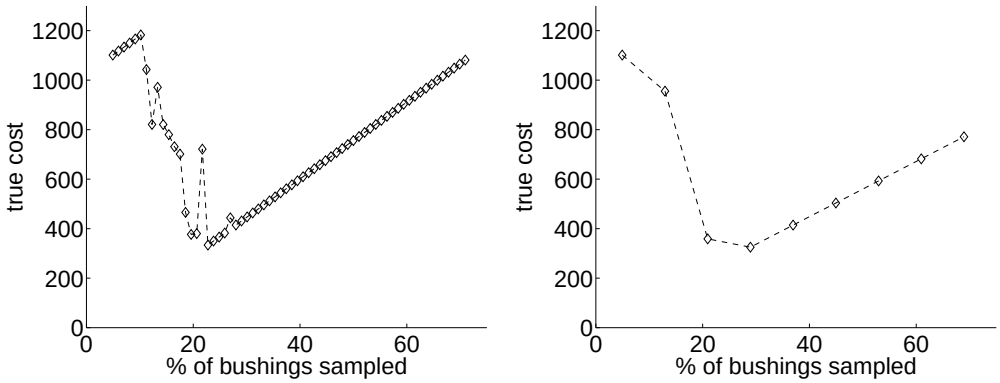
Chapter 5

Our Focus

In this section, we will first study how a batch size may affect the learning process, followed by examining how stopping criteria may greatly affect the learning outcome.

5.1 Learning with the Right Batch Size

The batch size is the number of transformers sampled per iteration. Having a large batch size reduces the overhead of iterations costs, i.e., $cost_{\text{Iter}}$, but risks low quality sampling and over sampling. Figure 5.1 shows the cost curve of two different batch sizes, i.e., sampling 1% and 8% of the entire dataset per iteration. Again, the curves are produced using a real life dataset from BC Hydro, which we will discuss in detail in Chapter 7. The smaller batch size demonstrates dramatic true cost fluctuations between iterations, especially during earlier iterations when the labeled dataset is small. Such fluctuations make it difficult to determine whether a minimum cost has been reached. Having a larger batch size, as shown in Figure 5.1b, can alleviate this initial noise; however, it can cause the active learner to sample more transformers than necessary by a large margin.



(a) Small Batch Size with $b = 1\%$ (b) Large Batch Size with $b = 8\%$

Figure 5.1: AL batch size performance comparison.

Our insight is that in early iterations when the learning process is noisy, having a larger batch size is favorable as it is more resilient to noise. On the other hand, as the active learner samples more transformers and becomes more stable, having a smaller batch size is more favorable to allow the learning process to approach the minimum smoothly and avoid overshooting by a large margin. This observation leads to the development of our dynamic batch size adjustment algorithm that attempts to benefit from both small and large batch sizes.

5.2 Stopping at the Minimal Costing Iteration

The second issue is when to stop the iteration process. To explain this, Figure 5.2 shows a typical trend of the true cost of the classifier calculated by Equation 4.3 (same curve as Figure 4.1, but x-axis shows % of data (bushings) sampled instead of iteration number). Ideally we want to stop the iterative process when the cost reaches the minimum. Practically, however, it is difficult to know when the minimum cost is reached without knowing the result of "future" iterations. We can see the difference between the highest and the lowest costing iterations in Figure 5.2 is quite large. For this reason, a stopping decision should be made carefully as it can potentially make a large difference to the learning outcome.

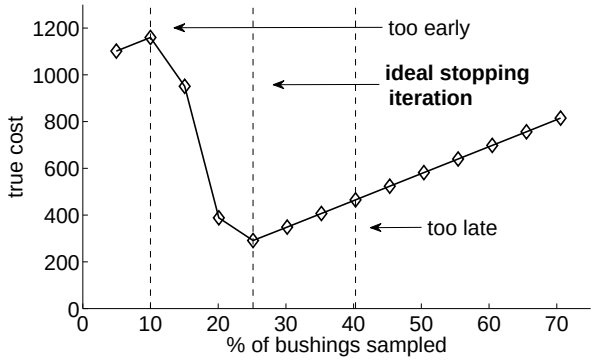


Figure 5.2: True Cost Learning Curve with $b = 5\%$.

As we can see, the choices of batch size and stopping criterion are two key issues for the iterative AL algorithm. These choices become even more important and more difficult due to the imbalanced cost of false positive error and false negative error in our problem. In the next chapter, we propose solutions to these issues.

Chapter 6

GDB And Stopping Criterion

In this chapter, we propose a batch size adjustment algorithm that aims to take advantage from both smaller and larger batch sizes. A complementary stopping criterion is also discussed.

6.1 Geometric Dynamic Batch Size (An Approach For *Batch-Size*)

Our dynamic batch size strategy is based on the following idea. If the estimated cost at the current iteration is lower than that at the previous iteration, the benefit from sampling (i.e., reducing classification error) out-weighs the labeling cost; therefore, we double the batch size to leverage this sampling benefit. A larger batch size also reduces the overhead cost associated with each iteration. If the estimated cost is increasing in comparison to that of the previous iteration, the benefit from sampling does not compensate for its cost, so we reduce the batch size by halve to avoid overshooting the minimum cost by a large margin. In other words, the batch size should be dynamically adjusted according to the trend of the estimated cost of the current classifier.

The above dynamic batch size could become so large that it overshoots the minimum cost by a large margin, or become so small that it takes many iterations before termination (thus a large overhead cost associated with iterations). For this reason, the batch size is restricted to a range $[b_{\min}, b_{\max}]$. Once the maximum batch size $b_{\max}$ is reached, further size doubling is turned off, and once the batch size goes below $b_{\min}$, the AL process terminates

(more on this in the next section). Algorithm $BatchSize_{\text{GDB-naïve}}$ below summarizes the above dynamic batch size adjustment.

Algorithm $BatchSize_{\text{GDB-naïve}}$

Input: $b_{\max}$, $b(0)$, $b(i-1)$, $Cost_{\text{Est}}(i-1)$, $Cost_{\text{Est}}(i)$

Output: $b(i)$

1: **if** $Cost_{\text{Est}}(i) > Cost_{\text{Est}}(i-1)$ **then**
2: **return** $b(i) = b(i-1)/2$
3: **else**
4: **return** $b(i) = \min(b_{\max}, 2b(i-1))$
5: **end if**

One may argue that doubling and halving the batch size can be too stern especially when the cost differences between iterations are very small. Making big adjustments based on small observations may seem like an overreaction. However, assuming $b_{\min}$ and $b_{\max}$ are set so that we have at least few iterations before termination, if at any point we see that the cost is not actually increasing as we expected from the previous iteration, the batch size will be adjusted back up to cope with the noise and similarly in the reverse case. As an extra precaution, we can also add a threshold Δ, to our adjustment policy so that the batch sizes will only be adjusted if the cost difference is greater than Δ.

As we can see in $BatchSize_{\text{GDB-naïve}}$, the batch size at any given iteration depends on the trend of the estimated cost observed previously. Thus, the accuracy of the estimated cost at an earlier iteration $j < i$ can affect how well we adjust the batch size at the current iteration i. If we can improve this accuracy at iteration j, then we could obtain a better batch size $b(i)$ for the current iteration i. This improvement is made possible by the new information that is now available from iteration i, but was not available in iteration j. Below, we consider this improved option for computing $b(i)$.

There are two kinds of new information available at the current iteration i but not available at iteration j. First, it is possible that the true label of some transformers unlabeled in iteration j is known as a result of sampling at iteration i, i.e., $U(j) \cap L(i) \neq \emptyset$. For such transformers we should utilize their confirmed labels to calculate the estimated cost at iteration j, instead of using the probability predicted by $M(j)$, i.e., $p_j(+|u)$, as in Equation 4.4. Second, even if the true label of some unlabeled transformers at iteration j, say u, remains unknown at iteration i, the probability estimate $p_i(+|u)$ given by the classifier

$M(i)$ is preferred to $p_j(+|u)$ given by $M(j)$ because $M(i)$ is based on more labeled data (i.e., $L(j) \subset L(i)$). These observations allow us to modify the estimation of $|FN(j)|$ as follows:

$$|FN(j)|_{\text{Est},i} = |U_-(j) \cap L_+(i)| + \sum_{u \in U_-(j) \cap U(i)} p_i\,(+|u) \tag{6.1}$$

The first term represents the number of false negative errors by $M(j)$ where the true labels are known at iteration i. The second term represents the estimate of false negative errors where the true labels are still unknown at iteration i. Thus, the modified estimated cost at iteration j by using the new information from iteration i is as follows:

$$Cost_{\text{Est},i}(j) = |FN(j)|_{\text{Est},i} \cdot cost_{\text{FN}} + j \cdot cost_{\text{Iter}} + (|L(j)| + |U_+(j)|) \cdot cost_{\text{Sample}} \tag{6.2}$$

To calculate the batch size $b(i)$, we look back and recalculate the estimated cost of each previous iteration j, using Equation 6.2, $j = 0, 1, ..., i$. This improved method is described in Algorithm $BatchSize_{\text{GDB}}$. Note that Equation 4.4 degenerates into Equation 6.1 in the case of $j = i$, as $U_-(i) \cap L_+(i) = \emptyset$ and $U_-(i) \cap U(i) = U_-(i)$. Therefore, $BatchSize_{\text{GDB-naïve}}$ is the special case of $BatchSize_{\text{GDB}}$ when there is no look back. The condition $b(j-1) \geq b_{\min}$ at line 3 will be explained in the next section.

Algorithm $BatchSize_{\text{GDB}}$

Input: $b_{\min}$, $b_{\max}$, $b(0)$, $Cost_{\text{Est,i}}(0)$, ..., $Cost_{\text{Est,i}}(i)$

Output: $b(i)$

1: $b(i) = -1$
2: //Re-estimated costs to determine appropriate $b(i)$
3: **for** $j = 1 \rightarrow i$ **and** $b(j-1) \geq b_{\min}$ **do**
4: **if** $Cost_{\text{Est,i}}(j) > Cost_{\text{Est,i}}(j-1)$ **then**
5: $b(j) = b(j-1)/2$
6: **else**
7: $b(j) = \min(b_{\max}, 2b(j-1))$
8: **end if**
9: **end for**
10: **return** $b(i)$

6.2 Stopping criterion (An Approach For SC)

We propose the stopping criterion, denoted by SC_{GDB}, that halts the active learner if the batch size $b(i)$ of current iteration, i, goes under some minimum threshold $b_{\min}$, i.e.,

$$SC_{\text{GDB}} = b(i) < b_{\min} \tag{6.3}$$

It is easy to see the following properties of this stopping criterion. First, assuming we start the learning process with $b(0) = b_{\max}$, it ensures that once we observe an increase in the cost, we will continue to learn for at least $log_2(\frac{b_{\max}}{b_{\min}})$ more iterations to confirm that the learning process has indeed passed its minimum cost and not just noise before stopping. Second, following the rule of geometric sum [32], $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots = 1$, the proposed stopping criterion ensures that we will take at most $b_{\max} + \frac{b_{\max}}{2} + \frac{b_{\max}}{4} + \frac{b_{\max}}{8} + \ldots = 2b_{\max}$ more samples to stop the active learner, assuming the cost continues to increase. More specifically, since the batch size is bounded by $b_{\min}$, the number of samples required to stop the active learner is therefore $b_{\max} + \frac{b_{\max}}{2} + \frac{b_{\max}}{4} + \frac{b_{\max}}{8} + \ldots + \frac{b_{\max}}{2^n} < 2b_{\max}$, where n must satisfy $\frac{b_{\max}}{2^{n+1}} < b_{\min} \leq \frac{b_{\max}}{2^n}$.

As a suggestion, the range of allowed batch size should be set so that the learning process will allow a few iterations before termination. The user should set the range by taking $cost_{\text{Iter}}$ into consideration. That is when the iterative overhead cost, $cost_{\text{Iter}}$, is large, we should set $[b_{\min}, b_{\max}]$ to be a smaller range to avoid large overhead cost associated with iterations. Whereas when $cost_{\text{Iter}}$ is small, setting $b_{\min}$ to be much smaller than $b_{\max}$ is beneficial to give the active learner more chances to overcome noise.

Now we can explain the condition $b(j-1) \geq b_{\min}$ in $BatchSize_{\text{GDB}}$. This condition states that if the re-computed batch size $b(j)$ of any early iteration $j \leq i$ is less than $b_{\min}$, the for-loop will terminate with $b(i) = -1$, and according to our stopping criterion SC_{GDB}, the AL process will terminate. This is exactly what we want in that the learning process should terminate as quickly as possible once the stopping condition is known to be satisfied at iteration i. Classifier $M(i)$ is returned by the learner since the sampling cost spent on obtaining $L(i)$ cannot be reverted, and a classifier constructed using more labeled data is preferred over $M(j)$.

In the rest of the paper, GDB denotes the algorithm based on the geometric dynamic batch size ($BatchSize_{\text{GDB}}$) and stopping criterion (SC_{GDB}) proposed in this section.

Chapter 7

Experiment

In this chapter, we will evaluate the performance of our proposed GDB algorithm using a real world dataset from BC Hydro in Canada.

7.1 Setup

This dataset contains transformer bushing level information collected from BC Hydro where each record in the dataset represents a unique bushing. It is a labeled dataset provided by the maintenance department of BC Hydro. To simulate an AL process, we hide all the labels initially except for 5% of the dataset, which were labeled and used as the initial training set to start off the AL process. The labels of the remaining 95% data are gradually revealed as the active learner requests for them, and are used to provide the ground truth for evaluation purposes (i.e., calculating true cost using Equation 4.3). The dataset was randomly divided into 20 sections (5% each) and the experiments were repeated 20 times using each of the sections as the initial dataset.

There are more than 1000 bushing records in the dataset, with 11 attributes as described in Table 7.1. More than 90% of the bushings in the dataset were manufactured between 1960 and 1989. The PCB content of these bushings were given as a real numerical value, i.e., we are given the PCB Concentration instead of a simple PCB contaminated or non-PCB contaminated binary value. The regulation on the end of usage of PCB contaminated equipments imposed by Environment Canada states that equipments containing PCB oil mixture with concentration greater than 500 mg/kg must be eliminated by the year 2009, and equipments with PCB concentration less than 500 mg/kg, but greater than 50 mg/kg

have an extension of usage until 2025 [23, 14]. We denote the minimum PCB concentration required to be considered as PCB contaminated as $PCB_{\min}$.

We perform experiments using various PCB levels as $PCB_{\min}$ to see how different threshold values may effect the results of each dataset, and allow BC Hydro to see what to expect when different threshold values are imposed. By varying $PCB_{\min}$, we will also vary the class distribution of PCB and non-PCB contaminated bushings in the dataset. As $PCB_{\min}$ increases, the number of PCB positive bushings in the dataset will decrease since less bushings will make the concentration threshold. This will consequently make the class distribution of the dataset to become even further imbalanced. Unfortunately, for confidentiality reasons, we cannot release the exact size and the data distribution of the dataset.

Table 7.1: BC Hydro Bushing Data Attributes

Attribute	**Type**
Area	Categorical
Equipment Type	Categorical
Equipment ID	Categorical
Bushing Position	Categorical
Manufacturer	Categorical
Model/Type	Categorical
Rated Voltage	Categorical
Rated Current	Numerical
Region	Categorical
Quantity of Liquid	Numerical
PCB Concentration	Numerical

Missing values were observed in the dataset, and two different procedures were taken to deal with the missing values: 1) replace missing values using WEKA Tools ReplaceMissingValues filter [16] where the attributes are filled in by the means and modes of the given dataset, and 2) remove records with missing values from the dataset. Categorical attributes were transformed into multiple binary attributes using WEKA Tools NominalToBinary filter as a necessary procedure for learning with an SVM based active learner. We use LibSVM [7] with its probability estimate option [37] to construct SVM classifiers and to provide posterior probability estimates. Moreover, a linear kernel is used for all our experiments since

in real world situations, we are given very little or no labeled data initially, so parameter tuning for popularly used kernels such as polynomial and Gaussian radial basis kernels is not viable.

We evaluate the proposed GDB algorithm from Chapter 6 and the five conventional stopping criteria, MU, OU, SA, MEE and SP, for fixed batch size AL that were discussed in Section 3.2. The threshold update strategy for MU, OU, SA and MEE described were also incorporated into our implementation. The thresholds of MU, OU and MEE are set to 0.1 initially and are decreased by 0.01 during each update. For SA, the threshold is set to 0.9 and is increased by 0.1 during each update [41]. For SP, the threshold is set to 0.99 and the window size is set to 3. A large portion (50%) of the dataset is randomly selected as the stop set for SP like suggested in [4]. The threshold values for all five compared stopping criteria were selected based on the experiments ran in [41, 4].

A fixed batch size b=5% is used for fixed batch size AL with each of the five compared stopping criteria. For comparison, the same batch size b is used as $b_{\max}$ and $b_{\min} = \frac{b_{\max}}{16}$ in our GDB algorithm.

Twelve different tests were ran using different $cost_{\text{FN}}$ and $PCB_{\min}$, as described in Table 7.2. As discussed, the class distribution of the dataset in the tests shifts from balanced to extremely imbalanced as $PCB_{\min}$ shifts from 50 mg/kg to 500 mg/kg. r is the ratio of negative and positive bushings of the entire given dataset, i.e., $r = \frac{\text{\# of non-PCB bushings}}{\text{\# of PCB bushings}}$. This choice of $cost_{\text{FN}}$ and $cost_{\text{Sample}}$ is based on the assumption that the more imbalanced the dataset is, the more costly it is to misclassify a positive bushing as negative. Note that in a real world situation, r is often unknown, but $cost_{\text{FN}}$ can be provided by the user. We set $cost_{\text{Iter}}$ equal to the cost of sampling 0.5% of the entire dataset.

Table 7.2: Experiment Settings

Test	$cost_{\text{FN}}$	$cost_{\text{Sample}}$	PCB_{min}
1	r	1	50 mg/kg
2	$2r$	1	50 mg/kg
3	r	1	100 mg/kg
4	$2r$	1	100 mg/kg
5	r	1	200 mg/kg
6	$2r$	1	200 mg/kg
7	r	1	300 mg/kg
8	$2r$	1	300 mg/kg
9	r	1	400 mg/kg
10	$2r$	1	400 mg/kg
11	r	1	500 mg/kg
12	$2r$	1	500 mg/kg

7.2 Replace Missing Values

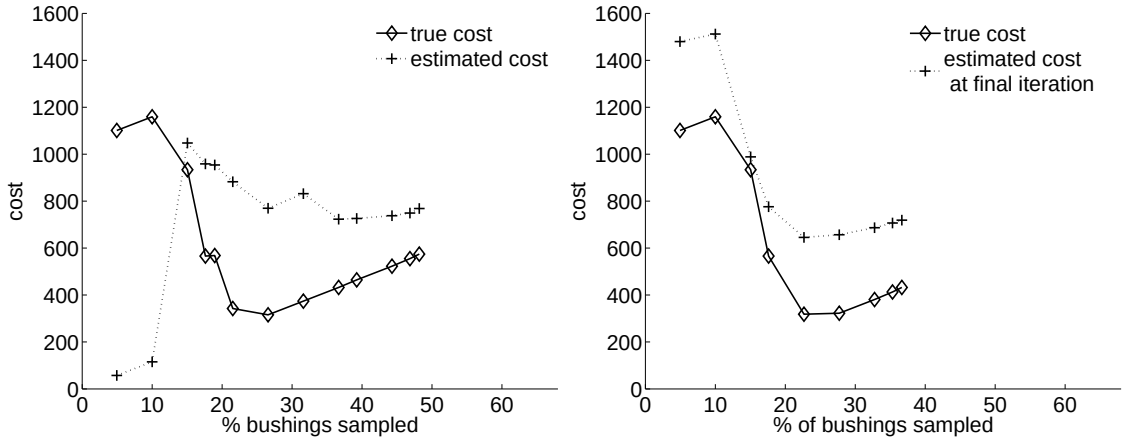
In this first experiment, we use the dataset where missing values are replaced using means and modes. We will first study the effectiveness of the proposed cost estimation methods for Algorithms $BatchSize_{\text{GDB-naïve}}$ and $BatchSize_{\text{GDB}}$. We then evaluate our GDB algorithm and the five existing algorithms discussed in Section 3.2 using cost and accuracy as the metric.

7.2.1 Effectiveness of the Estimated Cost

The purpose of this experiment is to study the effectiveness of the proposed cost estimation methods for both Algorithms $BatchSize_{\text{GDB-naïve}}$ and $BatchSize_{\text{GDB}}$ by examining how closely the estimated cost follows the true cost. Recall from Chapter 4, the true cost is unknown in a real world situation, and hence, the need to estimate the cost for Algorithms $BatchSize_{\text{GDB-naïve}}$ and $BatchSize_{\text{GDB}}$.

From Figure 7.1a we can see that the estimated cost of $BatchSize_{\text{GDB-naïve}}$ is quite noisy in predicting the true cost, especially during early iterations. Consequently, the batch size

was not adjusted appropriately causing the learning process to take longer to terminate than desired. For example, due to noise in the estimated cost, the batch size was not continuously reduced when the true cost was continuously increasing. As a result, though the minimum cost occurs when about 25% of the bushings were sampled, the active learner was not stopped until approximately 50% of the bushings were sampled.



(a) $BatchSize_{\text{GDB-naïve}}$ (b) $BatchSize_{\text{GDB}}$

Figure 7.1: True and Estimated Cost Curves (Test 11)

In comparison to $BatchSize_{\text{GDB-naïve}}$, we can see from Figure 7.1 that $BatchSize_{\text{GDB}}$ allows the active learner to terminate much closer to the minimum costing point. The estimated cost curve in Figure 7.1b is the result of recalculation on the last iteration before the active learner terminates. By comparing the estimated cost curves in Figure 7.1a and 7.1b, we can see that the recalculation of the estimated cost proposed in $BatchSize_{\text{GDB}}$ does improve the quality of the estimated cost of the earlier iterations. The improvement in the estimated cost allows GDB to enhance the batch size adjustment decisions made previously, and in this particular example, reduces the batch size quickly to terminate the active learner close to the minimum costing point as desired. Although the estimated cost calculated by GDB at the last iteration does not always accurately reflect the exact value of the true cost, it provides a good genuine trend of the true cost between iterations, which is what is needed for the adjustment.

Note that the estimated comparison for $BatchSize_{\text{GDB-naïve}}$ and $BatchSize_{\text{GDB}}$ will not be repeated again in Section 7.3 where bushings with missing values were removed from

the dataset. Having another example will just be a repeat of the same discussions of this section.

7.2.2 Cost Reduction

Cost reduction is the goal of AL. One reasonable baseline to measure this cost reduction is to use the cost of sampling all transformer bushings as a baseline reference. Let β be the true cost if we were to sample everything, and α be the true cost of the active learner terminated using a stopping criteria. The cost reduction is calculated as follows:

$$Reduction = \frac{\beta - \alpha}{\beta} \cdot 100\% \tag{7.1}$$

Table 7.3 shows the cost reduction of GDB and the five compared AL strategies on average over 20 runs for the twelve tests described in Table 7.2. In cases when the class distribution is more balanced, i.e., Tests 1-8 where $PCB_{\min} = 50, 100, 200, 300$ mg/kg, by applying AL, we are able to save at least 30% of the costs regardless of which stopping criterion we apply. The performance of OU, MU, SA and MEE give the best results when the class distribution is more balanced, i.e., Tests 1-8; however, as the class distribution becomes more imbalanced, their result becomes less favourable for reasons we will soon discuss. Before that, we first observe that GDB and SP are able to reduce the cost by more than half on all cases except for Test 12 where the class distribution and misclassification cost are both extremely imbalanced (i.e., $PCB_{\min} = 500$ mg/kg and $C_{\text{FN}} = 2r$). However, even in such extreme cases, GDB and SP are still capable of reducing the costs by 28% and 17% respectively.

Table 7.3: Replace Missing Values - % of True Cost Reduction for Tests Described in Table 7.2

Test	**GDB**	**OU**	**MU**	**SA**	**MEE**	SP
1	56%	64%	67%	67%	67%	56%
2	56%	64%	67%	67%	67%	56%
3	66%	74%	77%	77%	77%	65%
4	66%	74%	77%	77%	77%	65%
5	71%	80%	83%	83%	83%	71%
6	71%	80%	83%	83%	83%	71%
7	74%	80%	80%	80%	80%	72%
8	72%	67%	30%	73%	30%	71%
9	70%	37%	1%	41%	1%	65%
10	63%	-17%	-46%	-7%	-46%	61%
11	59%	15%	-4%	15%	-4%	54%
12	28%	-73%	-73%	-61%	-73%	17%

On the other hand, stopping criteria OU, MU, SA and MEE have very poor performance for tests with extremely imbalanced class distributions such as in Tests 9-12. In many cases, their cost reductions are negative, suggesting that it is better off sampling everything than utilizing AL. The reason a negative reduction is possible is due to the false negative misclassification cost being extremely high; by having an inaccurate classifier that leaves PCB bushings unidentified, we will incur a high misclassification cost making the result even more costly than simply sampling everything. The highly imbalanced class distribution (i.e., $PCB_{\text{min}} = 400$ mg/kg and $PCB_{\text{min}} = 500$ mg/kg), makes it less likely to obtain a representative dataset initially. In some cases, very few or no PCB positive bushings are sampled initially. This causes OU, MU, SA and MEE to falsely believe that all bushings can be confidently classified as negative during early iterations and terminate the learning process prematurely. GDB and SP can better overcome this initial noise because when the target (i.e., minimum costing point for GDB and classification stability between iterations for SP) has been observed, they continue learning for a few more iterations before stopping as supported by the results shown in Table 7.3. However, these additional iterations also impose unnecessary sampling cost when the learning curve is smooth, resulting in a worse

performance than OU, MU, SA and MEE as observed for Tests 1 - 7.

Note that Tests 1 & 2, 3 & 4, and 5 & 6 have identical results. This is because when $PCB_{\min}$ = 50 mg/kg, 100 mg/kg, and 200 mg/kg, the class distribution is relatively balanced and it is more likely for us to randomly draw a representative initial labeled dataset than when the class distribution is extremely imbalanced (i.e., $PCB_{\min}$ = 400 mg/kg and 500 mg/kg). As a result, the initial model can already be very accurate with very few or no misclassification. Hence, the difference in $cost_{\text{FN}}$ produces little to no effect on the results. Moreover, since r is a ratio of the negative and positive bushings in the dataset, having a more balanced dataset will equate to a smaller r, which means the difference between r and $2r$ is much smaller than when the dataset is imbalanced.

The effect of the false negative cost can be observed by comparing the results of Tests 7 & 8, 9 & 10 and 11 & 12. As we can see, when $cost_{\text{FN}}$ increases from r to $2r$ (while $cost_{\text{Sample}}$ remains at 1), cost reduction decreases. This is because when the false negative misclassification cost is larger, it also makes the classifier to have a higher tendency of classifying transformer bushings as positive, and hence, increase the sampling cost.

From Table 7.3 we observe that GDB and SP provide more stable results, in the sense that they do not have cases where the stopping criterion fails to provide any improvement. Overall, the results of GDB have shown to be comparable if not better than all the results for SP.

7.2.3 Accuracy

Our last study is to evaluate how accurately the PCB contaminated transformer bushings are identified. We use the standard *recall* (R) and *precision* (P) measures as defined in Equations 7.2 and 7.3. R measures how successful the model is in identifying a PCB positive bushing, and P measures how correct the model is when it classifies a transformer bushing as PCB positive.

$$R = \frac{TP}{TP + FN} \cdot 100\% \tag{7.2}$$

$$P = \frac{TP}{TP + FP} \cdot 100\% \tag{7.3}$$

where TP, FP, and FN are the number of true positive errors, false positive errors, and false negative errors. In the PCB identification problem, having a high R is far more

important than having a high P as the consequence of a false negative is much more severe than false positive.

Table 7.4 shows the recall and precision of the final classifier returned by the AL model when the learning process terminates (evaluated using all 1000+ labeled transformer bushing records provided by BC Hydro) on average over 20 runs. In addition, the average total number of transformer bushings that have been sampled by the AL process before termination, denoted as $|L|$, is also presented in Table 7.4 given as a relative percentage of the entire dataset size. $|L|$ is the labeled dataset that the active learner used to train the final classifier it returns when the learning process terminated.

In congruence with the discussion for Table 7.3, we can observe that in general, OU, MU, SA and MEE have a tendency of terminating the learning process prematurely (i.e., their $|L|$'s are smaller than GDB and SP's $|L|$), resulting in low recalls and precisions in most cases for Tests 9 to 12.

Note that the precision of GDB is also quite low for Test 12 (only 1.3% of the bushings classified as positive are actually positive). This is because for Test 12, the class distribution is very imbalanced, and $C_{\text{FN}} = 2r$ is extremely high, which skews the model to have a higher tendency of classifying more bushings as positive to lower the potential of false negative misclassification.

Table 7.4: Replace Missing Values - $|L|$, Recall (R) and Precision P for Tests Described in Table 7.2

Test	**GDB**			**OU**			**MU**		
	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)
1	19	100	100	10.1	100	100	5.05	100	100
2	19	100	100	10.1	100	100	5.05	100	100
3	19	100	100	10.1	100	100	5.05	100	100
4	19	100	100	10.1	100	100	5.05	100	100
5	19	100	100	10.1	100	100	5.05	100	100
6	19.13	100	100	10.35	100	100	5.05	100	99.52
7	22.79	100	100	11.36	95	75.38	5.05	95	39.43
8	24.3	100	100	11.86	95	31.43	5.05	95	6.84
9	27.08	100	100	11.61	55	10.43	5.05	55	1.43
10	33.28	100	100	13.38	55	4.82	5.05	55	1.34
11	36.46	100	100	13.38	30	16.9	5.05	30	0.57
12	39.68	100	1.3	6.56	30	0.57	5.05	30	0.57

Test	**SA**			**MEE**			**SP**		
	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)
1	5.05	100	100	5.05	100	100	20.19	100	100
2	5.05	100	100	5.05	100	100	20.19	100	100
3	5.05	100	100	5.05	100	100	20.19	100	100
4	5.05	100	100	5.05	100	100	20.19	100	100
5	5.05	100	100	5.05	100	100	20.19	100	100
6	5.05	100	99.52	5.05	100	99.52	20.44	100	100
7	12.62	95	95.22	5.05	95	39.43	25.24	100	100
8	14.89	95	100	5.05	95	6.84	25.74	100	100
9	13.12	55	100	5.05	55	1.43	31.8	100	100
10	16.15	55	100	5.05	55	1.34	35.33	100	100
11	13.88	30	100	5.05	30	0.57	42.15	100	100
12	20.19	30	100	5.05	30	0.57	75.97	100	100

On the other hand, SP has a perfect recall and precision for Test 12. This is because the total number of bushings sampled for SP is quite high (i.e., 75.97% of the entire dataset), and hence, the returned classifier is constructed using more information than GDB's. However, this more accurate classification model is constructed at the cost of sampling almost double the number of bushings sampled by GDB, which in this case (Test 12) is more than the false positive misclassification costs of GDB as indicated by the smaller cost reduction in Table 7.3.

Considering accuracy (i.e. R and P) alone, SP is the most logical choice if we are to select one algorithm to use as it attains 100% accuracy for all tests. However, when considering the cost factor, SP may not always be the optimal choice as its high accuracy is built on the expense of sampling a large number of bushings in comparison to other compared methods. For this reason, we cannot measure the performance of an algorithm based on accuracy alone.

7.2.4 Summary

Based on the results shown in Tables 7.3 and 7.4, we can deduce that for cases when the dataset is expected to have a balanced class distribution, it is recommended for BC Hydro to utilize MU, OU, MEE or SA as the stopping criteria. For a dataset with an imbalanced distribution, it is recommended to apply GDB or SP as the stopping criteria. We can see from the recall values in Table 7.4 that both GDB and SP are capable of identifying all the PCB contaminated bushings in all tests. Although they generally require more sampling, i.e., larger $|L|$, than MU, OU, MEE and SA, this additional sampling is necessary to avoid terminating the learning process prematurely - as discussed earlier. This consequently makes GDB and SP the better choices when working with imbalanced datasets. In particular, GDB can be a better choice as it does not over sample as much as SP, which results in a higher cost reduction than SP.

It is most likely that in a real life situation, the class distribution of the dataset is unknown. In such cases, we may approximate the class distribution from the initial sample set to give us some idea of how to determine the appropriate stopping criteria to follow based on the guideline suggested above.

7.3 Remove Records With Missing Values

We performed a second experiment using the same dataset and ran the tests as described in Table 7.2. However, instead of replacing missing values with means and modes, we removed records with missing values. Only about 75% of the original data remained after the removal process.

7.3.1 Cost Reduction

Table 7.5 shows the results in terms of cost reduction, as defined in Section 7.2.2. For Tests 1 - 6 when the class distribution is more balanced, all compared strategies are capable of reducing the cost by more than half. As expected, the performance of OU, MU, SA, MEE gives the best results for tests with more balanced class distributions, but performs very poorly, e.g., negative cost reductions, for tests with highly imbalanced class distributions and misclassification costs like Tests 7 to 12. Even in extreme cases such as Tests 10 and 12, GDB and SP are still capable of reducing the costs by 48% and 39% for Test 10 and 14% and 22% for Test 12 respectively. This is because OU, MU, SA and MEE have a tendency of terminating the active learner prematurely, and GDB and SP are designed to be more impervious to stopping prematurely as discussed in Section 7.2.2.

Table 7.5: Remove Records with Missing Values: % of True Cost Reduction for Tests Described in Table 7.2

Experiment	GDB	OU	MU	SA	MEE	SP
1	62%	67%	73%	73%	73%	62%
2	62%	67%	73%	73%	73%	62%
3	68%	75%	80%	80%	80%	67%
4	68%	75%	79%	79%	79%	67%
5	74%	81%	81%	84%	81%	72%
6	71%	78%	71%	80%	71%	69%
7	71%	48%	19%	48%	19%	69%
8	67%	7%	-35%	7%	-35%	66%
9	60%	22%	-3%	22%	-3%	58%
10	48%	-52%	-68%	-50%	-68%	39%
11	56%	10%	-4%	13%	-4%	55%
12	14%	-78%	-78%	-69%	-78%	22%

7.3.2 Accuracy

Table 7.6 shows the total sampled size $|L|$, *recall* and *precision* as defined in Section 7.2.3. The results presented in Table 7.6 share similar discussions as Table 7.4. The results of Table 7.6 are consistent with the discussion in the previous section where OU, MU, SA and MEE have a higher tendency of terminating AL prematurely when the class distribution is highly imbalanced as observed in Tests 9 to 12 from their small $|L|$, low R and/or P.

For Tests 10 and 12, the precision for GDB is quite low. This is because the class distribution and misclassification cost for both tests are very imbalanced, causing the classifier to have a higher tendency of classifying a bushing as positive to lower the potential of false negative misclassification. Although the precision of GDB is quite low for Test 10, its cost reduction (from the cost reduction table Table 7.5) still outperforms the cost reduction for SP for the same reasons as Test 12 in Section 7.2.3.

Table 7.6: Remove Records with Missing Values: $|L|$, Recall (R) and Precision P for Tests Described in Table 7.2

Test	**GDB**			**OU**			**MU**		
	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)
1	18.82	100	100	13.75	100	100	5	100	100
2	18.82	100	100	13.75	100	100	5	100	100
3	18.82	100	100	11	100	100	5	100	100
4	18.82	100	100	11.25	100	100	5	100	99.46
5	19.32	100	100	12	100	100	5	95	96.43
6	21.82	100	100	14.75	100	100	5	95	58.65
7	25.32	100	100	12	60	100	5	58.64	4.53
8	28.82	100	100	14	60	100	5	60	3.19
9	35.32	100	100	12	35	100	5	35	0.92
10	45.17	100	22.69	17.25	35	6.11	5	35	0.92
11	39.22	100	100	10.25	25	3.21	5	25	0.66
12	31.14	100	0.84	6.25	25	0.66	5	25	0.66

Test	**SA**			**MEE**			**SP**		
	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)	$\|L\|$ (%)	R (%)	P (%)
1	5	100	100	5	100	100	20	100	100
2	5	100	100	5	100	100	20	100	100
3	5	100	100	5	100	100	20	100	100
4	5	100	99.46	5	100	99.46	20	100	100
5	6.5	100	99.16	5	95	96.43	21	100	100
6	12	100	94.35	5	95	58.65	24.25	100	100
7	12	60	100	5	58.64	4.53	28	100	100
8	14	60	100	5	60	3.19	30.75	100	100
9	12	35	100	5	35	0.92	38	100	100
10	19	35	100	5	35	0.92	55	100	100
11	11.25	25	100	5	25	0.66	41.25	100	100
12	18.5	25	15.82	5	25	0.66	70.5	100	100

7.3.3 Summary

The results observed in Tables 7.5 and 7.6 follow the guidelines discussed in Section 7.2.4 where OU, MU, SA and MEE are the better choices when the class distribution is more balanced as they appear to provide higher cost reduction with high accuracy. However, their performance degrades dramatically when the class distribution becomes highly imbalanced as anticipated. For tests with highly imbalanced class distributions, GDB and SP are again the better choices among the compared strategies.

Chapter 8

Conclusions

The PCB transformer identification problem is a vital concern to power companies across the world as it is required by UNEP to have all PCB transformers removed by 2025. More importantly, the existence of PCBs endangers human health and environments. Because of the enormous expense incurred from sampling the PCB content of a transformer, mass examination of transformers is infeasible.

We proposed to utilize active learning algorithms to help construct a classification model that minimizes the number of instances required to be sampled. We then identified the benefits of various batch sizes and designed a batch size adjustment algorithm, GDB, that exploits the benefits from various batch sizes. A natural stopping criterion was also proposed in this thesis to help stop the active learner near the minimal costing point. We evaluated our GDB algorithm and five other algorithms using the real world dataset from BC Hydro in Canada. The result showed that GDB provides a steady performance and remains comparable to the best performing compared strategy for different cases as a whole.

Though we only used the SVM based active learning algorithm proposed by Brinker [5] in our experiments, it is also simple to apply our GDB algorithm with any other batch mode active learning algorithms that give probabilistic class estimates on the unlabeled instances. We believe that the proposed GDB algorithm and stopping criterion can be applied to other batch mode AL problems to help overcome noise and avoid continuing to sample too many instances when the targeted goal has been reached. In general, GDB can be easily modified to cope with other batch mode active learning problems that consider other quantities, other than cost defined in this thesis, as a performance metric. For example, instead of using cost, we can use confidence as a batch size adjustment metric and decrease, i.e., halve, the batch

size whenever a given confidence threshold is reached and double the batch size when the threshold is not reached.

In addition to the PCB transformer identification problem, we believe the algorithms studied and proposed in this thesis can also be extended to other domains that share a similar problem set up. An example would be disease identification where a profile of potential patients is given, and it is possible to diagnosis the disease through some tests; however, it is unrealistic to ask everyone on the list to take the test. It is obvious that leaving a diseased person unidentified imposes a higher consequence than taking a few unnecessary tests of a healthy person. Testing only one patient at a time is inefficient as the time required to take the patient to come to take the test and wait for the results make it more reasonable to test a batch of patients at a time. For problems like this, we can use the algorithms discussed and proposed in this thesis to help reduce the total number of patients tested and accurately identify the diseased patients from the potential list.

Currently, we assume all the costs such as the sampling cost $cost_{\text{Sample}}$, iterative overhead cost $cost_{\text{Iter}}$, and the misclassification costs are uniform across all transformers in this thesis. However, these assumptions are overly ideal in a real world situation. For example, shutting down the power of an urban area likely imposes a larger loss than a rural area. Future studies can be carried out to create a model that can ease off these constraints to allow the algorithm to be more readily applicable to a real world situation.

Bibliography

[1] Pcb transformer decontamination, standards and protocols, 1995.

[2] Toxicological profile for polychlorinated biphenyls (pcbs), 1997.

[3] D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[4] Michael Bloodgood and K. Vijay-Shanker. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 39–47, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[5] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *In Proceedings of the 20th International Conference on Machine Learning*, pages 59–66. AAAI Press, 2003.

[6] S. Chakraborty, V. Balasubramanian, and S. Panchanathan. Dynamic batch size selection for batch mode active learning in biometrics. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 15 –22, dec. 2010.

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[8] J Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[9] D.A. Cohn, Z. Ghahramani, and M.I. Jordan. Active learning with statistical models. *Arxiv preprint cs/9603104*, 1996.

[10] David Cohn, Richard Ladner, and Alex Waibel. Improving generalization with active learning. In *Machine Learning*, pages 201–221, 1994.

[11] Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. Morgan Kaufmann, 1995.

[12] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in*

Neural Information Processing Systems 20, pages 353–360. MIT Press, Cambridge, MA, 2008.

[13] Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 155–164, New York, NY, USA, 1999. ACM.

[14] Environment Canada. Pcb regulations: An overview, 2011.

[15] Yuhong Guo, Dale Schuurmansdepartment, and Computing Science. Discriminative batch mode active learning.

[16] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[17] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *In Proceedings of the International World Wide Web Conference*, pages 633–642, 2006.

[18] Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *In Proceedings of the 23rd International Conference on Machine Learning*, pages 417–424. Morgan Kaufmann, 2006.

[19] R.D. King, J. Rowland, S.G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L.N. Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[20] Florian Laws and Hinrich Schätze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 465–472, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[21] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[22] Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1622–1623, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[23] Minister of Justice. Pcb regulations, 2012.

[24] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[25] J. Mörner, United Nations Environment Programme, Inter organization Programme for the Sound Management of Chemicals, R. Bos, and M. Fredix. *PCB transformers and capacitors: from management to reclassification and disposal*. United Nations Environment Program, 2002.

[26] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[27] Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

[28] Greg Schohn and David . Less is More: Active Learning with Support Vector Machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

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

[30] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1070–1079, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[31] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. COLT '92, pages 287–294, New York, NY, USA, 1992. ACM.

[32] James Stewart. *Calculus Early Transcendentals*. Brooks/Cole, 2003.

[33] Katrin Tomanek, Joachim Wermter, and Udo Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *In Proc. of EMNLP/CoNLL07*, pages 486–495, 2007.

[34] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[35] Andreas Vlachos. A stopping criterion for active learning. *Computer Speech AND Language*, 22(3):295 – 312, 2008.

[36] Shuo Wang, Jian-Jian Wang, Xiang-Hui Gao, and Xue-Zheng Wang. Pool-based active learning based on incremental decision tree. In *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, volume 1, pages 274 –278, july 2010.

[37] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975–1005, December 2004.

[38] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback.

[39] Jingbo Zhu. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *In Proceedings of ACL*, pages 783–790, 2007.

[40] Jingbo Zhu and Huizhen Wang. Learning a stopping criterion for active learning for word sense disambiguation and text classification, 2008.

[41] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing*, 6(3):1–24, 2010.