Extending Active Learning for Improved Long-Term Return On Investment of Learning Systems

Ph.D. Thesis

Mohit Kumar

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Jaime G. Carbonell, Co-Chair
Alexander I. Rudnicky, Co-Chair
Burr Settles
Andrew E. Fano, Accenture
Rayid Ghani, University of Chicago

Keywords: Active Learning, Machine Learning, Real-Life Deployments, Human in the Loop, Interactive Systems

Om Shri Guruve Namah

Contents

Chapter 1

Introduction

Decision support problems like error prediction, fraud detection, information filtering, network intrusion detection, surveillance etc are important class of business problems. They are called decision support problems because the system is expected to assist in the human decision making by recommending which transactions to review and also assisting in the detailed review process. Data mining systems for such problems deal with building classifiers that are not working by themselves but are part of a larger interactive system with an expert in the loop (figure 1.1). Characteristics of these domains include skewed class distribution, lots of unlabeled data, concept/feature drift over time, biased sampling of labeled data and expensive domain experts. Such systems are expected to be deployed and to maintain their effectiveness over a long time. Learning in such interactive settings typically involves deploying a classifier to suggest relevant positive examples to human experts for review. The cost and availability of these experts makes labeling and review of these examples expensive. The goal of a machine learning system in these settings is to not only provide immediate benefit to the users, but also to continuously improve its future performance while minimizing the expert labeling/reviewing costs and increasing the overall effectiveness of these experts.

Even though problems like error prediction, fraud detection and surveillance have been researched and there are deployed systems for these problems, most of the work has focused on limited aspects of the overall problem and not understanding and managing all the trade-offs over time between the three factors: exploration (future performance), exploitation (immediate benefit), and labeling/reviewing cost together. Active learning algorithms [Settles, 2012] seek to maximize exploration (often at the expense of short-term exploitation). Traditional supervised learning algorithms maximize short-term exploitation. Cost-sensitive variation of active learning aims to optimize cost and exploration [Kapoor and Greiner, 2005, Margineantu, 2005, Settles, 2012]. Reinforcement learning [Wiering and van Otterlo, 2012] is targeted at optimizing exploration and exploitation. Thus the different threads of research have investigated the pairwise consideration of these factors of cost, exploration and exploitation but not considered them together.

The business goal of deploying these machine learning systems is to maximize the benefit/cost ratio for the system. The metric that corresponds to maximizing the benefit per unit cost is Return on Investment (ROI). In this thesis, we hypothesize that jointly managing the trade-offs between cost, exploration and exploitation leads to better overall return on investment from the machine learning system. We propose a general framework that takes into account these trade-offs. In addition, we design our framework to take into account several domain concerns that

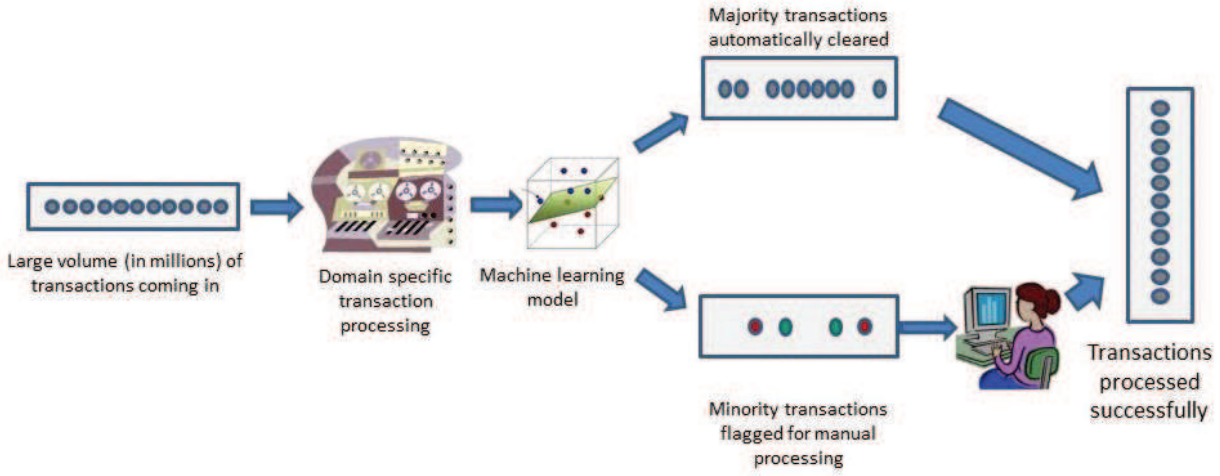


Figure 1.1: Interactive classification setup. A generic pipeline for domains like fraud detection, network intrusion detection, error prediction.

have not been considered previously in literature. Broadly, these concerns can be categorized into dynamically changing exploitation utility of a labeled example, dynamically changing cost of reviewing an example and actively tackling concept drift. We describe these concerns briefly here and explain them in detail in chapter 2.

Exploitation utility of a labeled example can change dynamically depending on the history of the labeled examples. For example, if the target for a machine learning system is root cause analysis for errors in claims processing, it is not valuable to identify/label transactions corresponding to already identified root causes. Thus the value of labeling an example with a root cause is dependent on the history of labeled examples.

The cost of reviewing an example can be reduced based on the order of reviewing the examples. For example, in the health insurance claims error prediction domain [Kumar and Ghani, 2011], the audit time for a claim decreases if similar claims are shown together. This reduction in audit time was achieved because of decreasing the context switching cost. The context switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was an error or a correctly processed claim. Once they had investigated that claim, they were given the next claim that was often completely different (both in terms of the *reason* for flagging and in terms of the procedures or diagnosis) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones.

1.1 Thesis Statement

This thesis extends the active learning paradigm for skewed, interactive classification problems by optimizing multiple objective functions of Cost and Exploitation giving significantly better long-term Return on Investment than existing approaches.

1.2 Hypothesis

1. Jointly managing the trade-offs between cost, exploration and exploitation leads to better overall Return on Investment for the machine learning system.
2. The proposed optimization framework leads to better performance compared to naive/baseline approach for jointly managing the trade-offs.
3. Modeling the dynamically changing reviewing cost leads to better performance compared to ignoring this phenomenon.

1.3 Thesis Contributions

The thesis has the following contributions:

I Define the novel area of interactive classification for practical machine learning systems with skewed class distribution.
   (a) Described the problem setup for the interactive classification for practical machine learning systems.
   (b) Described the characteristics of the problem particularly around dynamically varying cost and exploitation utility as well as changing target concept over time.
   (c) Described the trade-offs/factorization of the problem that motivates the proposed framework.

II Describe the framework extending the active learning framework to take into account differential exploitation utility of a labeled example and dynamically changing cost of reviewing an example.
   (a) Detail the modules that constitute the framework, namely, Cost model, Exploitation model, Exploration model, Utility metric, Evaluation criterion and Optimization algorithm.
   (b) Extend traditional active learning technique based on future error estimate reduction for the Precision@K [Japkowicz and Shah, 2011] metric relevant for the application domains of interest.
   (c) Map the optimization problem for the framework to the Budgeted Max Cover (NP-hard) problem and utilize existing efficient approximation algorithms for solving the problem.

III Evaluate the proposed framework on the real world domains of chemo-informatics (interaction of compounds with HIV Aids virus) and error prediction (health insurance claims error prediction) as well as for information filtering domain (using 20-Newsgroup and Reuters-RCV1 datasets). We provide evidence that:
   (a) the proposed hypothesis of jointly managing the trade-offs of cost, exploration and exploitation leads to better performance does hold true.
   (b) the optimization algorithm proposed by us using Budgeted Max Cover formulation significantly outperforms the naive/baseline approach of managing the trade-off.
   (c) modeling the dynamically changing reviewing cost helps in getting better performance.

IV Demonstrate that traditional active learning strategies are not always successful in handling concept/temporal drift and utilizing active learning for handling drift requires more attention

of the community.

1.4 Organization

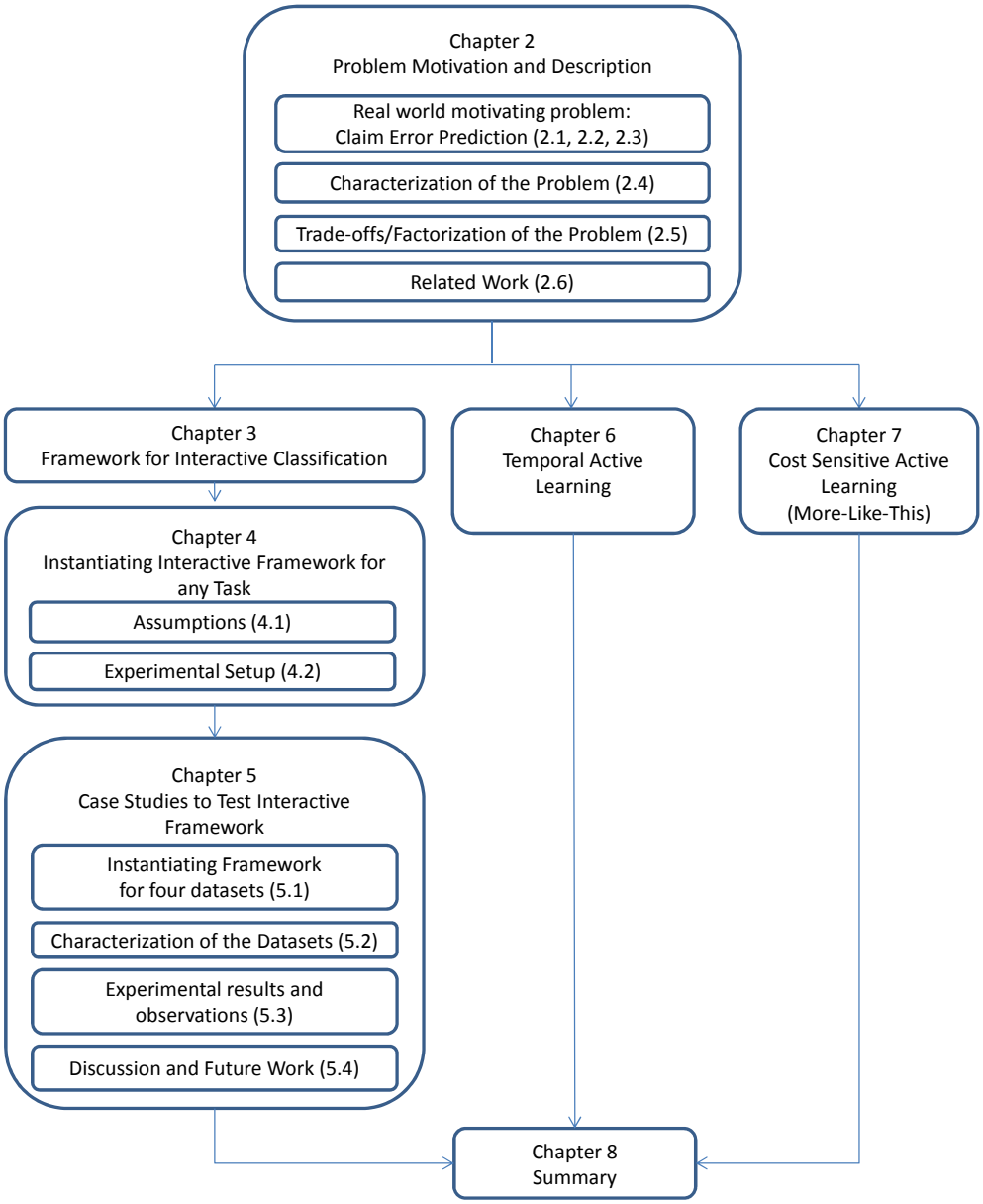


Figure 1.2: Thesis Organization

We start by describing the high level problem domains that motivated the current work in chapter 2. We detail the data and process characterization of the domains of interest. We then present the factorization and trade-offs that are observed in the domains. We present the literature review in section 2.6. In the next chapter 3, we describe the proposed framework based

on the problem description and the factorization described already. We explain how we can map any given task to the proposed framework and mention the details of the experimental setup that we propose to use to evaluate the framework in chapter 4. In chapter 5, we present our experiments and results on three tasks of Information Filtering, Chemo-Informatics and Error Prediction. We mention the details of the temporal active learning setup in the next chapter 6. We describe the various considerations for temporal active learning setup and then present the results for the problem. In the last chapter 7, we mention the details of our work on cost-sensitive exploitation and deploying the machine learning system for Claims Rework domain. We conclude by presenting the summary of the work in chapter 8.

Chapter 2

Problem Motivation and Description

The thesis work is motivated by deploying machine learning systems for efficiently solving real-world decision support problems. The motivating task within the space of decision support problems is health insurance claims error prevention, where the aim is to assist the human auditors in fixing the errors in insurance claims before the payment is made. This problem of incorrect insurance claims payment leads to higher administrative cost for healthcare industry which is a multi-billion dollar business problem globally. We introduce the error prevention task in some detail and describe the importance of such tasks in real life in section 2.1. We highlight the key insights from deploying a machine learning system to solve this business problem in section 2.2.1, whereas the details of the system and the observed results are described later in chapter 7. These insights led to the formulation of the thesis problem which addresses the domain of decision support problems where the machine learning system is deployed as part of a larger interactive system with an expert in the loop, described in section 2.3. We characterize these class of problems in terms of process and data characteristics in section 2.4. We discuss the factorization of such decision support problems and the trade-offs between the different factors in section 2.5.

2.1 Introduction to Health Insurance Claims Error Prediction

Health insurance costs across the world have increased alarmingly in recent years. These costs have been passed down to consumers and employer-sponsored health insurance premiums have increased 131 percent [National Coalition on Health Care, 2010] over the last decade. A large proportion of these increases has been due to increase in the administrative costs of insurance providers. According to a study by Farrell et al. [2008], $91 billion of over-spending on healthcare in the US is related to high administrative and insurance costs in 2006.

The typical process for insured healthcare in the US is that a patient goes to a service provider (medical facility) for the necessary care and the provider files a claim with the patient's health insurance company for the services provided. The insurance company then pays the service provider based on multiple complex factors including eligibility of the patient at time of service, coverage of the procedures in the benefits, and contract status with the provider etc.

Payment errors made by insurance companies while processing claims often result in re-processing of the claim. This extra administrative work to re-process claims is known as *rework* and accounts for a significant portion of the administrative costs and service issues of health plans. In the rest of the thesis, we use claims error prevention and claims rework interchangeably. These

errors have a direct monetary impact in terms of the insurance company paying more or less than what it should have. Anand and Khots [2008] estimates from a large insurance plan covering 6 million members had $400 million in identified overpayment. In our discussions with major insurance companies, we have found that these errors result in loss of revenue of up to $1 billion each year. In addition to the direct monetary impact, there is also an indirect monetary impact since employees need to be hired to *rework* the claim and answer service calls regarding them. According to estimates by an internal Accenture study, 33% of the administrative workforce is directly or indirectly related to rework processing. These statistics make the problem of rework prevention extremely important and valuable to the healthcare industry and motivated the work described in this paper.

There are two industry practices currently prevalent for identifying payment errors: random quality control audits and hypothesis (rule) based queries. Random audits are not very effective at identifying this rework since the majority of claims are correct and most of the effort spent on audits is wasted. In our discussions and research, we found that somewhere between 2% and 5% of the claims audited are rework, making 95% to 98% of the effort spent in random audits a waste. Hypothesis based querying involves domain experts identifying several hypothesis about how rework occurs and instantiates itself in claim data. They create rules which are then turned into SQL queries to find matching claims. Typical systems contain a few thousand rules that are run every day identifying thousands of suspect claims. This results in slightly better precision (or hit rate) than random audits but still require a lot of manual effort in discovering, building, updating, executing and maintaining the hypotheses and rules. These rules are also insurance plan specific and do not generalize well across companies thus making the deployment very time consuming and dependent on domain experts.

We developed a system to help reduce these claims errors using machine learning techniques by predicting claims that will need to be reworked. This Rework Prevention Tool was developed in conjunction with industry experts from Accenture's Claims Administration group who work with most of the large insurance companies in the US. In the next section, we describe the initial system deployment of the tool and the key insights from the deployment that led to the development of the ideas in the thesis.

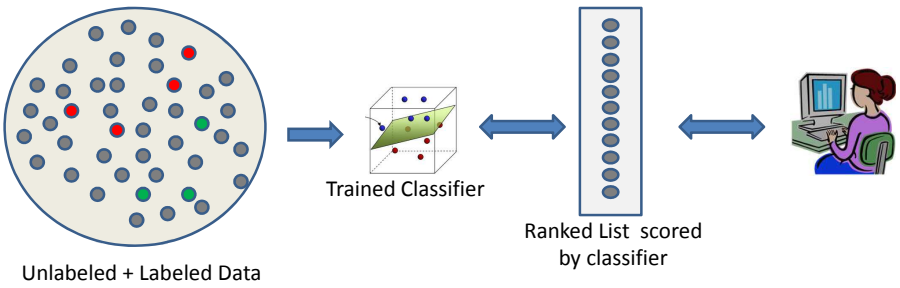


Figure 2.1: Conceptual machine learning system.

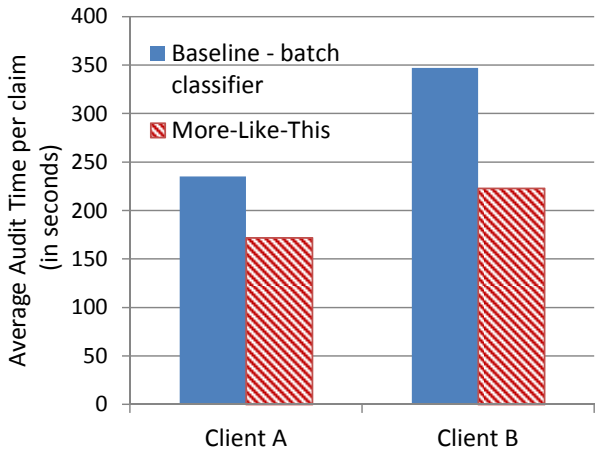


Figure 2.2: Observed average audit times per claim from two pilot deployments with Client A and B.

2.2 Initial Deployment of Machine Learning System for Claims Error Prevention

When we started designing the Rework Prevention tool, we conceptually considered the machine learning system as depicted in figure 2.1. We had lots of unlabelled and some labeled historical data. Using this historical data we wanted to learn a classifier to predict whether a given unlabeled example has payment error or not. Based on the classifier output, we would rank order the unlabeled examples for review by the expert who would give feedback at the cost of auditing time. The feedback can be solicited to verify whether the examples flagged have payment errors (exploitation of the classifier) or it can be used to improve the classifier performance as is done in active learning (exploration for the classifier). Based on this preliminary understanding, we developed an initial system for exploitation of the classifier [Kumar et al., 2010]. When we did a pilot deployment of the system at a large US health insurance company, we got encouraging results but also noticed several problems that affected the efficiency of the auditors when interacting with our rework prediction system. We highlight the key problem that led to the significant insight here and discuss the details of all the problems and observations later in Chapter 7.

Context Switching Costs: We used the classifier score to rank the test examples. The auditors were provided with the ranked list of claims and they started from the top of that list and worked their way down. The key problem they experienced was high switching costs when moving from example to example. This switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was a positive or a negative example. Once they had investigated (and labeled) that claim, they were given the next claim (from the ranked list using the classifier score) that was often completely different (in terms of the *reason* for flagging) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones.

We hypothesized that if we can group similar claims together, it would reduce the *context*

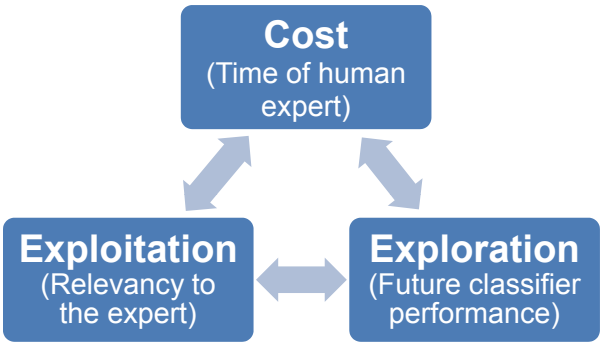


Figure 2.3: Joint consideration of cost, exploration and exploitation

switching and help make the reviewing times shorter. We developed the More-Like-This algorithm [Kumar and Ghani, 2011] based on this hypothesis, which uses the classifier score to select the top ranking examples and then clusters these examples with labeled historical data. We profile the cluster using the labeled historical data and use this profile to rank order the clusters for auditing. The experts audit all the examples within a cluster sequentially thus minimizing the context switch. The algorithm and experimental details are explained in Chapter 7. Key observation around auditing time of claims from two pilots with large US health insurance companies to evaluate the More-Like-This algorithm are shown in figure 2.2. We compare the audit time with the baseline batch classifier where we audit sequentially based on descending classifier score. We observe a significant reduction in audit time of 27% for Client A (from 3 min 55 sec for the baseline system to 2 min 52 sec for More-Like-This strategy) and 36% for Client B (from 5 min 47 sec for the baseline system to 3 min 43 sec for More-Like-This strategy). This observation led us to the following key insight.

2.2.1 Key Insight

Managing the cost of labeling an example (i.e. audit time) is an equally important consideration in addition to exploitation and exploration. We note that by keeping the same level of classifier accuracy, if we just reduce the audit time per example we are able to prevent more claim errors with the same level of auditor resources (budget) as we are able to audit more examples in the same amount of time. This observation leads us to add cost as a consideration along with exploitation and exploration, figure 2.3, for designing the machine learning system. We hypothesize that joint consideration of the three criterion of cost, explore and exploit will lead to better long-term Return on Investment for the machine learning system than existing approaches.

2.3 Thesis Motivation and Problem Definition

Based on the experience of deploying the machine learning system for Claims Error Prevention, we observe that many deployed machine learning systems such as those for fraud detection, network intrusion detection and surveillance, like the Claims error prevention problem, deal with building classifiers (or predictors) that are not deployed in isolation but are part of a larger interactive system (figure 1.1) with an expert in the loop. These applications involve a limited number of labeled examples with a high cost of labeling, and a large number of unlabeled examples, with majority of them being negative (skewed class distribution). Typically, once the models are trained, these systems score a large amount of new data and present the experts with a ranked list of cases to review. The immediate goal of such interactive systems is to help the experts find relevant (positive) examples. The commonly used approaches in such skewed class distribution problems use supervised classification and focus on increasing the classifier accuracy (or related metrics).

2.3.1 Evaluation Criterion

The business goal of deploying these machine learning systems is not just to maximize the performance of the classifier but to maximize the return on investment from such systems over time. The returns from such system is to predict and prevent as much fraud in case of fraud detection, errors in case of error prediction, intrusions in case of intrusion detection as possible. Maximizing the return on investment for such systems is directly related to making the experts more efficient at performing their task because in production or run-time the primary investment in the system is the expert's time for review. The cost of running and maintaining the system is fixed. The return from the system is the benefit received by processing the transaction correctly i.e. benefit from preventing fraudulent transaction, preventing erroneous payment of health insurance claims or preventing network intrusion. Thus to maximize the return on investment, we would want to maximize the benefits and minimize the cost over time.

2.3.2 Problem Domains

The need for an interactive learning system is fairly general across several problem domains in addition to the Claims Error Prevention domain. Fraud Detection systems are very commonly deployed in various domains like healthcare [Fayyad et al., 1996], credit card transactions [Curry et al., 2007]. Network Intrusion detection systems [KDD Cup, 2009] are used to distinguish between "bad" connections, called intrusions or attacks, and "good" normal connections where the learning system is expected to either block the attacks itself or make recommendations to an expert to review and intervene. In video surveillance domain [Collins et al., 1999, Pavlidis et al., 2001], the task is to assist the experts in detecting abnormal activity to allow appropriate intervention. For example, alerting security officers to a burglary in progress, or to a suspicious individual loitering in the parking lot, while options are still open for avoiding the crime. In Information Filtering domain, the system interacts with the user in helping them find the relevant information for them. Often the interesting tasks for information filtering are in domains where the user is an expert, for example an intelligence analyst for military.

2.4 Characterization of the problem

The characterization of the problem can be done along the following two dimensions: Data characteristics and Process characteristics. Data characteristics describe the nature of the data that we expect in these domains whereas the process characteristics determines the processes that are followed in interactive classification problems.

2.4.1 Data Characteristics

The data expected in such interactive domains like fraud detection, network intrusion detection, claims rework detection is expected to have the following characteristics:

1. *Skewed class distribution*: In the domains of fraud, network intrusion, claims rework and other similar domains, the incidences of fraud, intrusion and claim error are far fewer and majority of transactions do not have any problem. The typical percentage of fraudulent cases for credit card transactions is 0.1-0.5% [Ogwueleka, 2011], for network intrusion detection the most relevant Denial of Service category of intrusions are approximately 3-5% [KDD Cup, 2009] of all the network connections and for the domain of claims rework [Kumar et al., 2010] percentage of rework cases are approximately 3-7% of all the claims processed. Also within this small number of positive cases of fraud, intrusion, error etc. there may be further sub-classes. For example in claims rework there are different types of processing errors associated with each claim and in network intrusion there are different types of network intrusions like Denial of Service, Probing etc.
2. *Temporal Drift*: In all the domains it is expected that the target concept and/or data distributions may change over time. In certain domains like fraud and network intrusion, there are adversarial forces at play which keep evolving their strategies in order to prevent detection thus changing the target concept over time. In domains like health claims rework the reasons for error in claim processing may change over time due to factors like changes in contract between insurance company and the medical providers, conversion of claim coding standards from ICD-9 to ICD-10. In the domain of network intrusion, the intrusions may change over time due to various network system updates and intruders finding vulnerabilities within those systems. Also the data distribution may change, for example in the health claims domain due to price increase for the medical procedures over time.
3. *Biased Sampling of labeled data*: The process by which labeled data is acquired in different domains for building the initial models may be biased based on some sampling criterion. For example, in the health claims domain, one source for labeled data is the quality audits that are conducted to verify the effectiveness of the manual adjudication process, where auditors (called claims adjudicators) manually determine the payment amount of a given claim. For selecting the claims for quality audits, certain business rules are used very often based on the amount of the claim, adjudicator identity etc. Thus the primary aim for the quality audits is not to label (unbiased) data examples but to solve a business task of evaluating the adjudicators. Similarly in network intrusion domain, only the incidences that are detected based on currently deployed intrusion detection system are labeled. This is a common scenario in business tasks and introduces the bias in the labeled data.
4. *Lots of unlabeled data*: In these domains the transactions are considered labeled only when there has been a manual verification. For credit card fraud, most of the transaction are never reviewed and verified to be correct or fraudulent. So even though majority of them

are expected to be correct, only a sample of data is considered labeled.

2.4.2 Process Characteristics

Expensive Domain Expert: The annotators/labelers involved in the review task in these interactive domains are expensive, trained and specialized experts. Whereas in quite a few domains, a person may be trained quickly to accomplish the specialized task like part-of-speech tagging, these domains like fraud, network intrusion, claims rework require extensive specialized training. In such cases where the auditor time is not as costly, like Mechanical Turk[1], the system setup cost may be the significant component which is fixed and the runtime cost of reviewing the examples may not be significant. However for the decision support problems in the interactive domains of fraud, network intrusion and claims rework the true business cost of running these systems are the expert's time. Hence the motivation is to optimize the return on the run-time cost of the system which is equivalent to the expert's time investment.

Even though these are expert annotators, factors like inter and intra-annotator variability, providing incorrect labels, failing to provide a label at all or taking non-uniform time for auditing are still a concern. The work related to multiple imperfect annotators [Donmez, 2010], accounting for inter-expert agreements [Shah, 2011] is very relevant in the context of these type of problems. However the focus of the thesis is not to explore such implications in multi-annotator environment.

2.5 Trade-offs/Factorization of the problem

As mentioned in section 2.3.1, the business goal of deploying these machine learning systems is not just to maximize the performance of the classifier but to make the experts more efficient at performing their task over time thus maximizing the return on investment for the system. The natural factorization of the problem is to figure out the factors that influence the cost/investment for auditing an example and the return or benefit received from labeling the example. The run-time cost for reviewing an example is linked to the time it takes for the auditor to review an example[2]. The benefit from labeling an example is two-fold. First is the utility in terms of the relevance (of highly ranked instances) to the experts (*exploitation*). For example, if the fraudulent credit card transaction was $1000, then the immediate benefit in preventing the fraud is a savings of $1000. However, the immediate benefit can be more than just the value of the transaction in terms of saved legal fees in case of lawsuits etc. Thus the exploitation benefit is domain dependent. Second is the future utility or effectiveness (*exploration*) for each example. This is a measure of how much will the classifier improve given the label for this example which is the same measure used in active learning for sample selection. Improving the classifier will lead to more accurate audits (exploitation) in future.

These components, namely, cost, exploitation and exploration are general and although they occur in most domains that fit into an interactive setting, the correct model for each component still needs to be selected for the domain. For example, the annotation cost for an example can be measured in terms of some properties like sentence length for POS tagging [Ringger et al., 2008], seconds of voicemail annotated [Kapoor et al., 2007], number of tokens annotated for a

[1]The Amazon Mechanical Turk [Buhrmester et al., 2011] is a crowdsourcing internet marketplace that enables individuals or businesses (known as Requesters) to co-ordinate the use of individuals to perform tasks online for a monetary payment, usually a few cents, set by the Requesters

[2]In general, other cost factors around the running and maintaining the system are fixed and do not vary across examples.

document annotation task [Tomanek et al., 2007]. Also we have shown in [Kumar and Ghani, 2011] that the labeling time may be reduced for reviewing similar health claims in sequential order as the cognitive load is reduced if the consecutive tasks are similar. For exploitation, the immediate benefit from labeling an example can be different in different domains as well as within a domain may vary based on the business setup. For example, in health claim insurance domain, we can have a fixed/uniform savings for each claim based on the reduction of the administrative processing overhead which is independent of the claim amount. According to this business setup, the savings from preventing error for a claim which has a charged amount of $100 will be the same as a claim with charged amount of $15000. Whereas a more typical business setup will be that the benefit is based on the amount of the claim. For exploration, there have been traditionally three types of approaches: random, density-based and uncertainty-based [Settles, 2012]. Random sampling strategy gives equal importance to all examples, density-based sampling strategy utilizes the distribution and characteristics of the examples to select examples while uncertainty-based strategy uses the classifier's characteristics to determine the examples for which the classifier is most uncertain.

In addition to these factors of cost, exploration and exploitation, we need a *utility metric* to combine these factors and an *optimization algorithm* to maximize the performance against an *evaluation measure*. Utility metric is an estimate of the perceived benefit of an example that is calculated by the system. For example for uncertainty based active learning, where the aim is to select the most uncertain example, the utility metric is inverse of the distance from the decision boundary. In other words, closer an example is to the decision boundary higher its uncertainty of labeling hence higher its utility. The optimization algorithm for uncertainty active learning can be a simple maximizer function that selects the example with highest uncertainty utility. It is important to note that the utility metric is different from the evaluation measure and is intended to be correlated. Evaluation measure in the active learning scenario would be the classifier performance after labeling an example selected based on the utility metric. Thus we would want our utility metric of uncertainty to be correlated with the improvement in classifier performance. Thus utility metric is the system's estimate of the benefit of labeling an example which is optimized using an algorithm to maximize the evaluation measure.

Intuitively thinking through the setups for cost, exploitation and exploration models for various domains, we characterize them into three broad categories: *Uniform, Variable, Markovian*. We give the details below for our proposed formulation of the different models.

2.5.1 Exploitation Model

The exploitation model determines the value of identifying a positive example. For instance, the exploitation value of identifying a fraudulent credit card transaction is the total amount of the transaction. The exploitation value can also include the expected legal fees in case of a lawsuit in the domain of the credit card fraud. Thus, in general, this value is determined by the business task that the system is fulfilling and the business determination of the value for the task. The exploitation model can have the following setups:

1. **Uniform**: The value of each example is same for all positive cases. For example, in claims rework domain [Kumar et al., 2010] if we only account for the administrative processing overhead of fixing a rework claim, then the exploitation value for all rework claims are uniform. This is the typical boolean classification setup. The exploitation model outputs the expected value of the example which is the likelihood of an example being positive.

2. **Variable**: The value of each example depends on the properties of the example. For example, in claims rework domain [Kumar et al., 2010], we can take into account the savings based on the adjustment amount of the claim. For instance, if the correct amount to be paid for the claim was \$100 and the system was erroneously paying \$500 before the review, the adjustment amount is \$500-\$100=\$400 which is the additional savings that the system obtained by preventing this error. Two claims with different paid amounts may have the same adjustment amount, for instance, another claim with paid amount of \$2500 where the correct amount is \$2100, the adjustment is still \$400. Thus we are modeling the adjustment amount. This corresponds to the classic regression setup and the exploitation model outputs a real-value.
3. **Markovian**: The value of each example depends on the history and/or order of examples that have already been labeled, in addition to the factors determined for the *Variable* model. For example, in claims rework domain, once an error with a certain root cause is found, finding additional errors with the same root cause has no value. Similarly, in Information Retrieval, the idea of Maximal Marginal Relevance [Carbonell and Goldstein, 1998] where the incremental relevance of identifying next relevant document is dependant on the examples observed so far.

2.5.2 Exploration Model

The exploration model determines the value of querying for the label of an example to improve future performance. Improving the classifier will lead to more accurate audits (exploitation) in future. Similarly to the exploitation model, the exploration model can have the following setup:

1. **Uniform**: This model assumes that each example has the same contribution towards improving future performance. Although it has been shown theoretically that actively selecting the examples to label improves future performance better than passively/randomly selecting examples by Hanneke [2012], thus ruling out uniform contribution of examples in improving future performance. In practice, there are domains where random sampling strategy gives comparable performance to other known active sampling strategies. For example, in Settles et al. [2008], for the task of extracting contact details from email signature lines, random sampling strategy gives comparable results to any other known sampling strategy. For such domains, a uniform exploration model may be a reasonable practical choice.
2. **Variable**: The value for each example in improving the classifier is different and depends on the properties of the examples (or population) which can be pre-determined. This is applicable in domains where variants of density based active sampling strategies give good performance. For example, in [Ferdowsi et al., 2011], for the KDD cup'1999 network intrusion dataset, a strategy based on sparsity (opposite of density) produced the best performance.
3. **Markovian**: The value for each example in improving the classifier is dependent on the history of examples that have already been labeled, in addition to the factors determined for the *Variable* model. This is applicable in domains where uncertainty based active sampling techniques give good performance as the determination of uncertainty is based on the model learned so far which is in turn dependent on the history of the labeled examples.

In order for it to be comparable with the exploitation model, we need to calculate the exploration utility in the same currency as the exploitation utility. One way to define such exploration utility is by quantifying the future improvement over a fixed period of time in terms of the equivalent

exploitation value. In other words, if by labeling an example the classifier performance is improved by x% then over a period of time this x% improvement has certain exploitation value. We can then calculate the net present value of this future improvement due to exploration and combine it with the current exploitation utility.

2.5.3 Reviewing/Labeling Cost Model

The cost model estimates the cost of getting (or reviewing) the label of an example. The problem of estimating the cost of labeling an example has received some attention recently particularly related to cost-sensitive active learning. Settles et al. [2008] have presented a detailed analysis of annotation cost for four domains: named entity recognition, content based image retrieval, speculative vs. definite distinction for text, and information extraction. There are numerous factors that affect the cost of labeling an example and is an active area of research [Arora et al., 2009]. We propose a general characterization of cost models that helps map the different domain cost models to this framework. The intention with the proposed framework is thus to leverage the given cost models in the various domains (possibly still evolving based on the research) and utilize it in a unified fashion. The cost model can have the following setup:

1. **Uniform**: The cost to label an example is independent of the example and is constant i.e. each example costs the same time to label. For example, in Settles et al. [2008], for the speculative versus definitive language usage distinction task for biomedical abstracts, the hypothesis is that the cost for labeling is uniform.
2. **Variable**: The cost to label an example is dependent on some pre-determined properties of the example (and potentially annotator, GUI etc) like the sentence length for POS tagging task [Ringger et al., 2008] with longer documents taking more time to label. So there is variation in the cost for labeling an example but the cost can be determined as a function of the properties of the example (and potentially annotator, GUI and other factors [Arora et al., 2009]).
3. **Markovian**: The cost to label an example is dependent on the order in which the examples are being labeled, in addition to the factors determined for the *Variable* model. The intuition behind this model is that the cost of labeling an example will be lower if a very similar example was just labeled. Kumar and Ghani [2011] have shown that annotation cost can be reduced if similar claims are shown to the auditors in sequence reducing the cognitive switching costs (more-like-this). Thus the cost of labeling a claim is modified based on the previous claim that has been labeled.

   Another real-world example is in the domain of astro-physics where the objective is to observe certain properties of celestial objects using the Hubble telescope. The cost of executing such experiment is to physically align the telescope towards the particular object which requires burning rocket fuel to maneuver. Thus the cost of conducting an experiment is dependent on the location of the previous object and how much maneuvering needs to be done.

Consequently, the overall approach must aim to optimize multiple criteria including the *cost* of labeling an instance, the utility in terms of the relevance (of highly ranked instances) to the experts (*exploitation*) in conjunction with future utility or effectiveness (*exploration*).

2.6 Related Work

2.6.1 Active Learning

Active learning deals with the problem of proactively selecting examples to be labeled in order to reduce the number of training examples needed to achieve a certain classifier performance. Settles [2009] provides an excellent summary of the area of active learning. Settles [2011a] review and discuss six current directions in active learning research aimed at making it more practical for real-world use. The directions are namely, querying in batches, noisy oracles, variable labeling costs, alternative query types, multi-task active learning, changing (or unknown) model classes. We propose to add to this list the considerations around differential utility of each labeled example, dynamically varying labeling costs based on the order of labeling examples and tackling concept drift.

Since the setup for the proposed framework involves prioritizing the list of examples that are labeled by the auditors, the area of active rank learning [Donmez and Carbonell, 2009, Long et al., 2010], where the aim is to sample example to learn better ranking function is directly relevant. In our current work we do not use this setup as the proposed approaches have been primarily developed/evaluated for web retrieval tasks. However, applying these techniques in our current framework is a very promising future direction for the work.

Recently, significant work has appeared in interactive learning for natural language applications [Settles, 2011b, Small, 2009] where the underlying need is to reduce the necessity of large annotated corpora and feature engineering which are both expensive. However, the notion of exploitation by obtaining the label on example is not considered and may not be defined in some cases.

Given that the problem domains that we are interested in have skewed class distribution, the research work related to rare-category active learning [He and Carbonell, 2007] and imbalanced class distribution [Bloodgood and Shanker, 2009, Zhu, 2007] is also relevant. Another related area is active feature acquisition where the goal is to select the most informative features to obtain during training [Saar-Tsechansky et al., 2009]. Budgeted learning [Guha and Munagala, 2007, Kapoor and Greiner, 2005] aims at learning the most accurate 'active classifier' based on a fixed learning budget for acquiring training data.

Since we aim to come up with a utility of exploration to combine with the exploitation utility, the thread of research in active learning that estimates the future generalization error probability is of interest. Such utility function for estimating future error reduction has been studied by Lindenbaum et al. [2004], Roy and Mccallum [2001], Schein and Ungar [2007] to guide the active learning selection. We utilize the ideas developed in this thread of research for coming up with the exploration utility and then additionally quantify the future improvement's value in order to combine it with exploitation utility. However, a challenge in utilizing the future error reduction strategies is that they have been evaluated in existing literature as a 'relative' measure to rank order the examples for labeling selection. For our approach, we need to use the absolute measure of the utility and transform it so that it can be combined with the exploitation utility.

Cost-sensitive Active Learning

There are two aspects in cost-sensitive active learning that are relevant to the framework. First aspect is whether we are able to determine the cost of labeling an example for any given domain. This is an open and active research area [Arora, 2012, Vijayanarasimhan and Grauman, 2009, Wallace et al., 2010] where the studies have shown that the annotation times vary with instances

and are dependent on multiple factors including intrinsic properties, annotator characteristics etc. However they can also be learnt reasonably well with some observations [Haertel et al., 2008, Settles et al., 2008, Vijayanarasimhan and Grauman, 2009, Wallace et al., 2010].

Second aspect is incorporating the annotation cost in active learning. Settles [2012] have summarized the findings across multiple studies for incorporating the cost information in the query selection algorithm as mixed evidence regarding their effectiveness. They state that it is unclear whether these mixed results are intrinsic, task-specific, or simply a result of differing experimental assumptions. All the studies use the notion of ROI [Haertel et al., 2008] which is the ratio of benefit to cost where benefit is in terms of active learning utility measure like uncertainty. Settles et al. [2008] found that using ROI with predicted costs is not necessarily any better than random sampling for several natural language tasks (this is true even if actual costs are known). However, results from Donmez [2010], Haertel et al. [2008], Wallace et al. [2010] suggest that ROI is sometimes effective, even when using a learned cost model. In our experiments, we have assumed that the true annotation cost is known in order to study the effectiveness of the framework. Also we have extended the notion of benefit to include the exploitation utility in addition to the exploration (active learning) utility.

Temporal active learning

There has been a lot of research in designing learning approaches under temporal drift but not focused on problems that can use the benefits of active learning. Zliobaite [2010] exhaustively reviews these approaches along with the various terminologies. Learner adaptivity approaches discussed there such as adaptive base learners [Hulten et al., 2001], learners with adaptive parametrization [Klinkenberg and Joachims, 2000], adaptive training set formations and fusion rules of the ensembles [Street and Kim, 2001] that are relevant to the current work (see references therein for further details). Hoens et al. [2012] focus on learning with streaming data where there is both concept drift and class imbalance. The authors highlight that this is an under-researched area and applies to many real-world problems. We take special note of this in our work and specifically address problem settings where there is significant class imbalance and show empirical comparison of approaches with different levels of imbalance.

There has been some work in active learning on streaming data with concept drift [Žliobaitė et al., 2011] and without concept drift [Chu et al., 2011, Zhu et al., 2007]. The results from Žliobaitė et al. [2011] show that random sampling performs better than the proposed active strategies and the authors recommend randomization of active sampling strategies. The key difference between streaming data and our focus is that in the streaming data setup, instances are streaming in to the system and a decision needs to be made right away whether to ask for a label or not. The incoming unlabeled data cannot be stored and queried later. This scenario happens in certain real-world problems (web search for example) but is rare in enterprize problems. In most enterprize interactive data mining systems, data needs to be stored anyway for other purposes (auditing for example) and the constraint of making a labeling decision instantaneously is not present. Also, in these problems, the domain experts labeling the data are the expensive components of the process and data storage costs often pale in comparison. For these practical reasons, we consider a setting where the unlabeled pool gets augmented with new data coming in, which is different from the two extreme settings of fixed unlabeled pool and completely stream-based setup with no memory. Chu et al. [2011] also mention that a periodically changing unlabeled pool is a more realistic scenario than the two extremes of static unlabeled pool and online streaming data. There can be multiple settings for the evolving unlabeled pool. *cumulative streaming pool* setting where

new unlabeled examples keep coming in and get added to the streaming pool, thus increasing the unlabeled pool available to the learner. The second setting can be *recent streaming pool* where only the most recent unlabeled examples are available to the learner. In the current work, we only experiment with the *recent streaming pool* setting and leave the *cumulative streaming pool* setting for future work. This corresponds to the *Daily Classification Task* setup recommended by Forman [2006] for studying concept drift.

In learning from data streams with concept drift, the popular approach has been to learn classifiers over different time period and combine them in weighted ensembles [Kolter and Maloof, 2007, Scholz and Klinkenberg, 2007, Wang et al., 2003]. However, the effectiveness of traditional instance selection strategies in the periodically changing unlabeled pool setup is still not well understood. Hence, we incorporate these dimensions in our empirical setup.

2.6.2 Frameworks for solving trade-offs amongst different criterion in learning systems

Zuluaga et al. [2013] have approached the problem of active learning for simultaneously optimizing multiple objectives and coming up with all the pareto-optimal designs. However the primary difference with our proposed approach is that we have a single objective function that we are trying to optimize which is finding maximum number of positive examples over long term. The single objective does have multiple components (cost, exploration and exploitation) that need to be taken into account. However, it is a very interesting future direction to use the pareto-active learning as an intermediate step to obtain the pareto-optimal set and then from the pareto-optimal set figuring out a strategy that aligns to the final objective function of maximizing return on investment. This two-step process would enable using different exploration objectives rather than only generalized error reduction strategies.

Work in the area of reinforcement learning [Sutton and Barto, 1998], particularly around techniques for balancing exploration-exploitation trade-offs is relevant for our framework. The key difference between reinforcement learning with respect to supervised learning is the concept of delayed rewards where the true 'label' or reward is not presented to the learner. Reinforcement learning focuses on having a set of environment states and a set of actions with the goal of understanding the state space and maximizing rewards. Such notion of state space is not present in supervised learning which is the focus of this thesis. However, managing the exploration-exploitation tradeoff through the multi-armed bandit problem [Audibert et al., 2009, Li et al., 2010] formulation in reinforcement learning is relevant. Li et al. [2010] work on contextual bandit problem for personalized news recommendation sequentially selects articles to show the user based on contextual information about the users and the articles. They balance the trade-off between exploring whether an article (particularly new article) is relevant to a user versus showing the most relevant article (exploitation) to the user given the user and article context. The multiple arms in the multi-arm bandit formulation are the news articles which can be shown to multiple users. The mapping of the decision support problems that we have focused in this thesis is not obvious to this formulation as the transactions are not to be reviewed/labeled multiple times. However, this is an interesting direction to explore as the advantage for considering these techniques is that there are theoretical guarantees on the expected regret (performance relative to optimal solution) rates that can be leveraged in the framework.

Another alternate research direction is the importance-weighted active learning by Beygelzimer et al. [2009] where importance weights are used to bias the learning process. The importance of an example can potentially be mapped to the ROI for an example. However, the novel scenario that

we have proposed in this thesis of markovian models for cost, explore and exploit are not easily mapped into this approach.

Chapter 3

Framework for Interactive Classification

3.1 Introduction

Given the factorization of the problem mentioned in the previous section, the hypothesis proposed is that jointly managing the trade-offs between exploitation, exploration and labeling/reviewing cost will lead to better performance of the system over time. We propose an interactive learning framework to manage the tradeoffs over time between the three factors, study the interplay between them, and optimize the net utility of a labeled example resulting from a combination of these three optimization criteria. The proposed framework extends the typical active learning setup to take into account factors related to exploitation and cost in a unified framework.

Related research in this area has focused on managing one or two of these three factors at a time. Active learning algorithms [Settles, 2009] seek to maximize exploration (often at the expense of short-term exploitation). Traditional supervised learning algorithms maximize short-term exploitation. Cost-sensitive variations [Kapoor and Greiner, 2005] of both of these aim to optimize cost and exploration (cost-sensitive active learning [Margineantu, 2005, Settles et al., 2008]) or cost and exploitation (cost sensitive learning [Kumar and Ghani, 2011]). Reinforcement learning is targeted at optimizing exploration and exploitation.

We propose an extension to the current interactive learning framework that encompasses all these related research areas to yield a general framework that takes into account several domain concerns in addition to the absolute measures of performance. Broadly, these concerns can be categorized into differential exploitation utility of a labeled example, dynamically changing cost of reviewing an example and tackling concept drift. Differential exploitation utility corresponds to the variable and markovian exploitation model setup mentioned in the section 2.5.1. Dynamically changing cost corresponds to the markovian cost setup described in section 2.5.3. For tackling concept drift, as described in section 2.4.1, we investigate the existing active sampling techniques for the exploration model which is detailed in chapter 6.

3.2 Interactive Learning Framework

We present a framework for developing practical systems that can be deployed for interactive classification systems. The framework takes into account the cost of labeling or reviewing examples along with the utility of exploration and exploitation as shown in figure 2.3. We envision

that most interactive learning systems can be mapped to this framework and use it to manage the three factors jointly. For instance, traditional classification systems where a classifier is trained offline using historical data and deployed to score transactions for review can be mapped to the setting with Uniform cost, Uniform exploitation and Uniform exploration. It is mapped to uniform cost model because cost is ignored in traditional classification systems hence every example is assumed to have same cost. It is mapped to uniform exploration as exploration utility is not modeled explicitly and hence is implicitly assumed to be same for all examples. It is mapped to uniform exploitation as all the positive examples are considered to have same value.

Classic active learning setup can be mapped to Uniform cost, No exploitation and Variable/Markovian exploration. In classic active learning setup, cost is ignored hence every example is assumed to have same cost. It is mapped to no exploitation as there is no inherent value of an example and the only consideration is to improve the future classifier accuracy. Different active sampling strategies within classic active learning can be mapped to uniform/variable and markovian exploration utility. For example, uncertainty based sampling is a markovian strategy as it uses the information from the classifier trained on historically labeled data to assign the utility for unlabeled examples.

Our framework consists of the following components:

- Exploitation Model
- Exploration Model
- Reviewing/Labeling Cost Model
- Evaluation Metric
- Utility Metric
- Tradeoff optimization algorithm to optimize the utility metric

For any domain, the *Cost function, Exploitation function and Evaluation metric* are specified by the domain, however the user still needs to make an appropriate choice for modeling. The remaining components of the framework: *Exploration model, Utility Metric and Joint Optimization Algorithm* are completely system developer's choices.

The exploration, exploitation and cost model can each have the following three broad variations: *Uniform, Variable, Markovian*, as described in previous chapter 2. *Uniform* means that each example gets the same value from the model. *Variable* means that each example can potentially have a different value that is a function of the example's (or population's) features. *Markovian* means that each example has a variable value which is not only a function of example's (or population's) features but also a function of previous (ordered) set of examples that have already been labeled.

Thus instantiation of the framework for a new domain will involve the following steps:

1. Determine Evaluation Metric
2. Choose Exploitation Model
3. Choose Reviewing/Labeling Cost Model
4. Choose Exploration Model
5. Choose Utility Metric
6. Choose Tradeoff optimization algorithm to optimize the utility metric

3.2.1 Evaluation Metric

The business goal of deploying these machine learning systems is not just to maximize the performance of the classifier but to maximize the return on investment from such systems over time. Maximizing the return on investment for such systems is directly related to making the experts more efficient at performing their task because in production or run-time the primary investment in the system is the expert's time for review. The cost of running and maintaining the system is fixed. The return from the system is the benefit received by processing the transaction correctly i.e. benefit from preventing fraudulent transaction, preventing erroneous payment of health insurance claims or preventing network intrusion. Thus the metric for evaluating the system should be tied to this business goal of finding positive examples with maximum net value. The metric that matches the business needs is **Return on Investment** which is net benefit factored by cost, described by Haertel et al. [2008]. There are two typical business setups for these systems, first is that the expert/auditors are available on demand and even though the maximum allowed budget is specified the system may choose to not use all of it if it expects negative value. Thus the net realized benefit of the system is the exploitation value (total dollars saved in case of fraud) minus the cost of labeling the example. The second setup is fixed cost setup where the budgeted time/cost of experts is incurred irrespective of whether they utilize the time for reviewing or not as they are dedicated to this task only and can't spend their time on other tasks. For example, if there are no credit card fraud transactions for 1 day, the experts still need to be paid their hours even though they did not review any transactions. Thus the Return on Investment metric may not need to include the cost component in this case (as it is fixed) and is simplified to be the cumulative exploitation value of the transactions reviewed. In this work, we consider the latter setup of use-it-or-lose-it budget. Other evaluation metrics which take into account the performance of the final model along with the exploitation value can also be used based on the system needs, for example if the actual model is also considered as the output of the system (in domains, where model is used to generate insights about root cause). We do not consider this metric in the current work.

In addition to the overall evaluation metric for such interactive system as Return on Investment, we also need to specify the metric to evaluate the classifiers within such systems. The operating range relevant for the system is dependent on the number of audits that can be done based on the budget. Thus the classifier performance metric should measure the performance for the top examples corresponding to the audit capacity. Amongst the various evaluation metrics for classifiers such as AUC, F-1 score, Precision, Recall etc [Japkowicz and Shah, 2011], the metric that is most appropriate for these conditions is Precision@Audit percentage, where audit percentage is the expected audit capacity of the system based on the budget. This metric corresponds to the intuition that we need a system which is very accurate for the top scored examples corresponding to the budget in order to efficiently use the budget. This is similar to the motivation for the use of Precision at top 10 document metric [Manning et al., 2008] commonly used in Information Retrieval community as the most relevant results are expected from the search engines on the first page showing top 10 documents. In typical interactive classification domains like claims rework the audit capacity is expected to be 2-5% of all transactions [Kumar and Ghani, 2011].

3.2.2 Choice of Exploitation Model

As described in section 2.5.1, the exploitation model determines the current value of identifying a positive example. Depending on the task and domain, the value can be **uniform** for all positive

examples (the value of each example is same for all positive cases), **variable** (the value of each example depends on the properties of the example), or **markovian** (the value of each example depends on the history of examples that have already been labeled, in addition to the factors determined for the *variable* model). So for the given task, the exploitation model has to be determined based on the context of the task. For example, in credit card fraud detection domain, the business value for preventing a credit card fraud is equal to the value of the transaction. Thus this is a *variable* exploitation model. In claims rework domain, the business value for preventing rework can be determined to be the administrative overhead prevented. Thus this setup will be a *uniform* model.

3.2.3 Choice of Reviewing/Labeling Cost Model

As described in section 2.5.3, the cost model determines the cost of getting (or reviewing) the label of an example. The cost model can be **uniform** (each example costs the same time to label), **variable** (the cost to label an example is dependent on some pre-determined properties of the example (and potentially annotator, GUI etc), or **markovian** (the cost to label an example is dependent on the order in which the examples are being labeled, in addition to the factors determined for the *variable* model). So the choice has to be made by analyzing the given domain which cost model is most appropriate and leverages the current research being done in characterizing the factors for determining the cost [Arora et al., 2009]. For example, in claims rework domain, a reasonable cost model is based on the complexity of auditing the type of claims where the complexity of the claim is pre-determined by domain experts. Thus this is a *variable* cost model as the cost is pre-determined by the property of the example (claim type).

For markovian setup, the cost reduction is based on the similarity of two examples. In certain domains, the domain expert may define business rules to describe similarity between examples. For example in claims domain, the expert can say all air ambulance claims are similar or all cardiac surgeries for stent placement are similar or the similarity can be geography specific that all cardiac surgeries for stent placement in Florida are similar. Such business rules can be easily used with the framework to define the similarity between examples. However we take a generalized approach to evaluate the framework where we use an appropriate distance metric (like cosine, euclidean) over the feature space to define similarity between examples. The similarity threshold can be determined by the domain experts. For our experimentation, we determine it empirically based on some constraints for the optimization algorithm which are discussed in section 5.3.1.

3.2.4 Choice of Exploration Model

The exploration model determines the value of querying for the label of an example to improve future performance of the classifier. Similarly to the exploitation model, the exploration model can be **uniform** (each example has the same contribution towards improving future performance - the random sampling strategy in active learning), **variable** (the value for each example in improving the classifier is different and depends on the properties of the examples (or population) which can be pre-determined), or **markovian** (the value for each example in improving the classifier is dependent on the history of examples that have already been labeled, in addition to the factors determined for the *Variable* model). This is similar to deciding the active sampling technique which typically requires empirical evaluation of various strategies for the domain as there are no 'general' strategies that are known to work in all domains. There has been recent work by [Ferdowsi et al., 2011] in developing 'safe' active learning strategies which extend the hybrid

active sampling approaches [Baram et al., 2004, Donmez et al., 2007] to make an online choice for a safe strategy. However they also do not take into account concept drift and assumes fixed unlabeled pool. So we propose novel active sampling strategies for handling concept drift which also have the property that they should not adversely affect the performance when there is no drift. We introduce our approach and the completed work in chapter 6.

3.2.5 Utility Metric

The utility metric used to optimize system's performance should reflect the evaluation metric specified by the domain. The evaluation metric, as mentioned in section 3.2.1, is Return on Investment with use-it-or-lose-it budget which corresponds to maximizing the cumulative number of positive examples labeled. We hypothesize that a utility metric that takes into account both exploration and exploitation will be correlated with the overall evaluation metric of Return on Investment. Thus the utility metric that we propose is a linear sum of exploitation and exploration utility.

$$OU = \alpha * ETU + (1 - \alpha) * ERU \tag{3.1}$$

where OU is overall utility of an example, ETU is the exploitation utility, ERU is the exploration utility and α is a constant factor to give different importance to the utilities. We can envision more complex/non-linear combination of the two utilities in future, however, a simplistic linear combination of exploration and exploitation, equation (3.1), is intuitive if we can calculate both the utilities in terms of same 'currency' (dollar value). We now present the details of deriving the exploitation and exploration utility in terms of the dollar value.

Exploitation utility

In this work, we instantiate the utility metric for the case of uniform exploitation (i.e. boolean classification). The instantiation of the utility metric for variable exploitation (regression) and markovian (root cause) is left as future work.

Exploitation utility of an example x, is the expected likelihood of the example being positive.

$$ETU = P(f(x) = 1|D) * \tilde{V} \tag{3.2}$$

where $P(f(x) = 1|D)$ is the probability of an unlabeled example being positive given the labeled data D and $\tilde{V}$ is the uniform exploitation value of each example. Uniform exploitation value is the dollar value of a positively labeled example. For example, in Claims rework domain the uniform exploitation value of a claim is \$50 which corresponds to the administrative overhead savings by preventing the rework.

Exploration utility

In order to calculate the joint utility metric we need to convert the exploration model into equivalent terms as the exploitation model so that the total utility can be calculated. One way is to use a utility function that estimates the future generalization error probability and quantify the future improvement's value at present time. Such utility function for estimating future error reduction has been studied by Lindenbaum et al. [2004], Roy and Mccallum [2001], Schein and Ungar [2007] to guide the active learning selection. However, we need to instantiate this utility function for our purposes to get the absolute measure of utility appropriate for the interactive setup. In addition, in order to quantify the future improvement and linearly combine

it with exploitation utility we need to calculate the current value of the future improvement. In other words, if by labeling an example the future performance improves, we want to calculate the value of the future improvement at the current time. We can use the Net Present Value concept commonly used in economics [Nitzan and Bichler, 2009] for this transformation of future improvement to current value. This corresponds to the total exploitation value amortized over a period of time because of this improvement. Thus the exploration utility, for an unlabeled example x is a convolution of two functions:

$$ERU = NPV(U(x)) \tag{3.3}$$

where $U(x)$ is the future error reduction estimate and function NPV converts the value of the utility function $U(x)$ to its Net Present Value. Function $NPV(a)$ is defined as

$$NPV(a) = E * T * \tilde{V} * a \tag{3.4}$$

where E is the Expected number of examples audited per time period, T is the amortization Time period, $\tilde{V}$ is the expected Value of each example and a is any given function.

For evaluating the utility function $U(x)$, we need to first determine the performance metric for the classifier. For the interactive framework, given that the number of audits are fixed dependent on the budget, for boolean classifier evaluation, the classifier performance metric should measure the performance for the top examples corresponding to the audit capacity. Hence the performance metric that is used is the Precision@audit percentage, similar to Kumar and Ghani [2011] (instead of 'accuracy' as used by Lindenbaum et al. [2004], Roy and Mccallum [2001], Schein and Ungar [2007]). For evaluating this Precision@K performance metric, we use Lindenbaum et al. [2004]'s terminology for describing the utility function.

$$\begin{aligned} U(x) = \quad & E_{l\in 0,1}[U(D \cup < x, l >)] = \quad & P(f(x) = 0|D).U(D \cup < x, 0 >) \\ & + \quad & P(f(x) = 1|D).U(D \cup < x, 1 >) \end{aligned} \tag{3.5}$$

where D is the labeled dataset, x is the unlabeled example being evaluated, l is the label {0,1} and $P(f(x) = l|D)$ is the classification probability. Thus equation (3.5) gives the 'expected' utility for adding the unlabeled example x to the training data. We define the utility function as:

$$U(D \cup < x, l >) = Precision@K(D \cup < x, l >) - Precision@K(D) \tag{3.6}$$

In words, equation (3.6) states that the utility function is the increase in the performance for the metric Precision@K that is achieved by adding the example x to the training data. There are multiple ways in which the $Precision@K(D)$ can be evaluated based on the expected error reduction literature for active learning [Baram et al., 2004, Lindenbaum et al., 2004, Roy and Mccallum, 2001]. In our experiments, we estimate Precision@K(D) with the following three variations:

Labeled training history: Measure the performance of the classifier learnt on the labeled data D so far. This method measures training data error (self-error) and has been pointed out by Baram et al. [2004], Schohn and Cohn [2000] to give substantially biased estimates of the classifiers performance due to the biased sample acquired by the active learner.

Unsupervised estimation based on current active pool: We estimate the classifier performance on the unlabeled pool, where the current classifiers posterior class probabilities are taken as proxies

for the true ones [Roy and Mccallum, 2001]. In order to measure the Precision@K metric, we rank order the unlabeled pool examples based on the classifier's probability score and for the top K examples, we calculate the mean probability score.

$$Precision@K(D) = \frac{\sum_{i=1}^{K} P(f(x_i) = 1|D)}{K} \tag{3.7}$$

where $x_{i...k}$ are the top scoring examples based on $P(f(x_i) = 1|D)$. Thus equation (3.7) gives the 'expected' Precision@K based on the current classifier.

Oracle experiment: Using true active pool labels: As a thought experiment to evaluate the methodology and framework, we calculate the exact error reduction by 'assuming' that an oracle tells us the labels of the unlabeled pool examples and the future test examples for calculating the exploration utility. Thus the probability terms in equation (3.5), become either 0 or 1 depending on label and we calculate the exact future performance, assuming knowledge of labels of future test examples, if we add the example to the training data. It is important to note that the oracle information is used for calculating the exploration utility only and not for any other step.

To summarize, the utility metric that we are proposing is a linear combination of the current exploitation utility and future exploration utility. The current exploitation utility is the expected dollar value of a transaction and future exploration utility is defined as the expected increase in dollars saved due to the reduction in future classifier error. In order to calculate the future exploration utility we are proposing to use the generalized error reduction strategy proposed in existing literature and have proposed two novel changes to make them suitable for our purpose. Firstly we have defined the metric in terms of Precision@K metric whereas in literature classifier accuracy is used. Secondly, we have transformed the future error reduction to its present value borrowing a concept from economics called Net Present Value [Nitzan and Bichler, 2009].

3.2.6 Joint Optimization Algorithm

Depending on which of the above models hold for the interactive learning task under consideration and the utility metric that we want to optimize, different optimization algorithms should be considered. The joint optimization algorithm determines how to combine the different factors together and optimize for the desired utility metric. One simple/baseline way of combining the different factors is by rank fusion [Cormack et al., 2009] which is popularly used in Information Retrieval community [Manning et al., 2008] for combining rankings for different search algorithms. In rank fusion, we obtain the ranks based on different factors and then combine the rankings to come up with one joint ranking. However, the rank fusion algorithm is applicable only for cases where the factors are order agnostic and thus is not applicable for combining any 'markovian' setup of cost, exploration or exploitation where the value is dependent on the ordering. We propose a novel method for combining the factors, particularly 'markovian' factors by formulating the optimization as a budgeted maximum set coverage problem which is described below.

Budgeted Max Set Coverage problem

To give the context of the budgeted maximum set coverage problem, we start by describing the general set coverage problems. Max coverage problem is where we are given as input several sets and a number k and the goal is to select at most k of these sets such that the maximum number of elements are covered, i.e. the union of the selected sets has maximal size. The sets

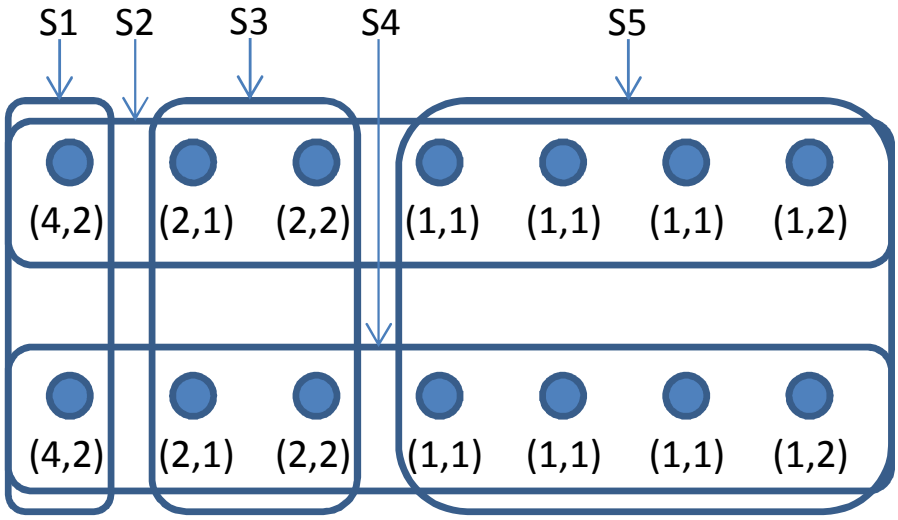


Figure 3.1: Sample elements with their set memberships. There are five sets S1-S5. Each element's weight and cost (w, c) is mentioned below the element.

may have some elements in common. In reference to figure 3.1, if k=2 then the optimal set coverage is with S3 and S4 which covers all the elements. In the weighted version every element e_j has a weight $w(e_j)$. The task is to find a maximum coverage which has maximum weight. In reference to figure 3.1, if k=1 and we consider the weights of each element then the optimal set to select is either S2 or S4 both of which have a total weight of 4+2+2+1+1+1+1=12. In the budgeted maximum coverage version, not only does every element e_j have a weight $w(e_j)$, but also every set S_i has a cost $c(S_i)$. Instead of k that limits the number of sets, in the budgeted version a budget B is given. This budget B limits the weight of the cover that can be chosen. In reference to figure 3.1, if we consider that the cost of the set is a simple aggregation of the element-wise cost then the costs of the different sets are S1:4, S2:10, S3:6, S4:10, S5:10. If the budget given is 14 then the optimal selections would be S1 and S5. Budgeted maximum coverage is an NP-hard problem [Khuller et al., 1999] which has been used in solving several applications related to optimally locating facilities to serve customers, machine job scheduling and circuit design [Hochbaum, 1997].

Why is Budgeted Max Coverage appropriate for solving interactive classification?

For the interactive classification framework, the setup that is neither addressed in existing literature nor is easily solved is the 'markovian' variation for the cost, exploitation and exploration model where the value can depend on the order of labeling examples in addition to the history. One way to approach this order-dependent markovian setup is by defining sets based on the property that determines the markovian effect and thus capturing the markovian effect within sets. In other words, we assume that only the recent ordered history within a set is relevant to determine the markovian cost, exploitation and exploration and hence we are able to calculate it in a deterministic way for the defined set. For example, for the markovian setup for the cost model the labeling cost may get reduced based on the similarity of the examples labeled before. Thus if we cluster the similar examples in a set we can then determine the exact order to label examples in order to minimize cost.

For the other markovian variation for cost, exploitation and exploration where the value is only dependent on the history and not the order of the labeled examples (more appropriate for exploitation and exploration models) budgeted max cover is still appropriate. In such cases, instead of large sets each examples forms its own unique singleton set and the optimization algorithms for solving budgeted max cover allow for calculating the markovian factors efficiently in runtime, more details in section 3.2.6. The non-markovian setups get reduced to the Budgeted Max Cover[1] problem easily and are efficiently solved as well.

Additionally, budgeted max coverage problem also has the notion of utility (weight of an element) and cost of an element which is appropriate for the proposed interactive classification framework. The significance of having the utility and cost concept separately is that they don't have to be in the same 'currency' i.e. we do not have to necessarily specify the cost in terms of dollars or the unit in which the utility is expressed. It can be specified in terms of time, number of experiments or any other appropriate unit in which the budget has been specified. This is practical for a lot of domains as the exact conversion to dollars (or other specified unit) may not be feasible. For example, in Claims Rework domain the auditor time can be specified easily but to convert the time into dollars would require knowing their salary scale, geographic location etc. Also for the markovian utility example of root cause identification the unit for exploitation utility may be the unique number of root causes identified (not dollars). Converting the audit time to such a unit may not be feasible. Thus the Budgeted Max Coverage solution provides the practical flexibility of not requiring the utility and cost to be in the same 'currency' and is appropriate for the interactive classification task.

How is interactive classification mapped to Budgeted Max Coverage problem?

The overall utility value for each example is defined as the sum of the exploitation (current) utility and the exploration (future) utility of an example. This utility value gets mapped to the 'weight' of each example in budgeted max cover formulation. As the cost is defined as a set property in budgeted max cover, we can derive the cost of the set as a function of the cost of each element within the set. While aggregating the cost at set level, we can take into account the markovian property and find the ordering within the set that minimizes the cost as mentioned before. Thus for each set the elements have weights that correspond to the overall utility value and the cost which is the aggregation of the cost of examples within the set.

How are the sets defined?

In our framework, the markovian cost model is dependent on the ordering of the labeled history (markovian exploitation and exploration model are dependent on unordered history), so we define the sets to model the markovian cost. The property that we use for determining the set memberships is 'similarity' of examples as we assume that labeling similar examples in-order reduces the labeling cost. Given a similarity threshold, we determine the similarity matrix for each batch of examples. Using this similarity matrix we find out the maximal cliques[2]. For each maximal clique, we exhaustively create sets of all possible sizes with all members of the clique. Thus we have $O(2^M)$ sets where M is the size of the largest clique. We empirically determine a similarity threshold for the dataset where the practical consideration is to control the size of the maximal clique as the run-time for budgeted max cover is a polynomial function of the total number of

[1]Budgeted max coverage reduces to 0/1 knapsack problem with singleton sets and no markovian effect

[2]We use a matlab implementation of Bron-Kerbosch algorithm [Wildman, 2011] to find the maximal clique

sets. More details about the specific algorithm used and its runtime complexity are mentioned in the next section.

Budgeted Max Coverage algorithm and appropriateness for Interactive Classification framework

Given the sets, weight of each example as the exploration and exploitation utility, cost of the set as a function of the cost of examples in the set and budget L, our framework maps directly to budgeted max coverage problem. Budgeted max cover is an NP-hard problem and Khuller et al. [1999] have proposed an algorithm that achieves an approximation ratio of $1-1/e$. This approximation ratio is also shown to be the best possible approximation ratio unless $NP \subseteq DTIME(n^{O(\log \log n)})$. We do not modify Khuller et al's algorithm but mention it here in order to explain how the different markovian models (exploitation, exploration and cost) can be solved using the same algorithm.

Budgeted max coverage problem is formally defined as follows. A collection of sets $S = \{S_1, S_2, ...S_m\}$ with associated costs $\{c_i\}_{i=1}^m$ is defined over a domain of elements $X = \{x_1, x_2, ...x_n\}$ with associated weights $\{w_i\}_{i=1}^n$. The goal is to find a collection of sets $S' \subseteq S$, such that the total cost of elements in S' does not exceed a given budget L, and the total weight of elements covered by S' is maximized. Some notations used are: let W_i be the total weight of the elements covered by set $S_i, i = 1, ..., m$. Let $G \subseteq S$ be a collection of sets. Let $w(G)$ denote the total weight of the elements covered by G. Let $W_i', i = 1, ...m$, denote the total weight of the elements covered by set S_i, but not covered by any set in G.

Algorithm 1 Khuller et al's Budgeted Max Coverage Approximation Algorithm

1. $H_1 \leftarrow argmax\{w(G)$, such that $G \subseteq S, |G| < k$, and $c(G) \leq L\}$
2. $H_2 \leftarrow \phi$
3. For all $G \subseteq S$, such that $|G| = k$ and $c(G) \leq L$ do
   (a) $U \leftarrow S \backslash G$
   (b) Repeat
      i. select $S_i \in U$ that maximizes $\frac{W_i'}{c_i}$
      ii. if $c(G) + C_i \leq L$ then
         A. $G \leftarrow G \cup S$
      iii. $U \leftarrow U \backslash S_i$
   (c) Until $U = \phi$
   (d) if $w(G) > w(H_2)$ then $H_2 \leftarrow G$
4. if $w(H_1) > w(H_2)$, output H_1, otherwise, output H_2

We now enumerate how we can use Khuller et al's budgeted max coverage algorithm for solving the different models within our framework. For ordered markovian cost, we have already defined intelligent set creation to capture the markovian effect within the sets. For un-ordered markovian exploitation model (for eg root cause analysis), we can embed the calculation in step 3(b)i where we calculate W_i', the total weight of the elements covered by set S_i, but not covered by any set in G. For root cause analysis, we do assume that we have an exploitation model that can help us estimate which examples are similar from root cause perspective.

Similar to exploitation model, we can also embed the markovian exploration model calculation in step 3(b)i. However, one common assumption that is made in active learning for markovian

exploration is to ignore the submodularity consideration for batch learning [Settles, 2011a] i.e. assume that the incremental change by labeling the examples one-by-one within a batch doesn't change the exploration value. In this case we can pre-compute the markovian exploration utility for a batch and don't need to embed it in the algorithm. In our experiments, we also make this choice of ignoring the submodularity consideration. For our experiments, we use a very efficient implementation of Khuller et al's algorithm in Submodular Function Optimization toolkit [Krause, 2010]. The Submodular Function Optimization toolkit can be efficiently used for solving all markovian cases for Cost, Exploration and Exploitation as described above and thus is a very appropriate choice for implementing the framework.

Chapter 4

Instantiating Interactive Framework for any Given Task

4.1 Assumptions for Mapping Any Given Task to the Interactive Framework

In order to instantiate and evaluate our proposed framework and understand its performance, we have made some simplifying assumptions. These simplifying assumptions are not inherent constraints for the framework but rather practical assumptions that allow us to evaluate and characterize the framework empirically. We list out the assumptions below for the expected domain inputs and modeling choices, summarized in table 4.1, some of which we have already described in chapter 3.

4.1.1 Assumptions in domain inputs

As mentioned in section 3.2, the domain specifies the Cost function, Exploitation function and the Evaluation metric. We make the following assumptions:

Cost function

Cost function is given: First assumption is that the exact cost function is given to us. It is a simplifying assumption as determining the cost of labeling an example is an active area of research

		Assumption
Domain Inputs	Cost Function	Cost function is given
		For markovian cost reduction-Simple setup of first order markovian with X% reduction in time
	Evaluation	Use-it-or-loose-it fixed budget for each iteration
Modeling Choices	Cost Model	Cost function is given
	Exploit Model	Probability estimate from exploitation model
	Explore Model	Estimating Error Reduction is feasible and reasonable learning strategy
		Ignoring submodularity for batch learning
	Utility Function	Utility of exploration in terms of NPV is calculable
		Equal weightage to combine explore and exploit utilities for overall utility
	Optimization Algorithm	For Budgeted max cover: utility and cost is well determined

Table 4.1: General assumptions for the thesis

[Arora, 2012]. However, our framework treats this module as input and is intended to assimilate the ideas for estimating the cost. Additionally, for some of the domains like Claims Rework we have true observations for the amount of time taken by auditors to label some of the claims. We have used the observations available to impute the missing timing information for the claims, more details in section 5.1.1.

For markovian cost reduction-simplistic setup of first order markovian with X% reduction in time: Second assumption is that the markovian cost reduction is a simplistic first order markovian process. Thus the cost reduction for a given example is dependent on the similarity with only the last labeled example. Also the cost reduction is a constant factor reduction if the examples are similar. In other words, if the example being labeled is similar to the last example that was labeled, it will have a constant factor reduction in it's cost. Alternately, we could have used the cost reduction factor to be a function of the degree of similarity between the examples but that would have introduced another factor in our experimentation. Thus we chose to use the simpler (constant factor) cost reduction scheme based on thresholding the degree of similarity.

Exploitation function

We do not make any assumptions for the exploitation function. However we have only experimented with the 'uniform' exploitation function in this work. The 'uniform' setup corresponds to the boolean classification task. For the variable and markovian exploitation variations, the major consideration is that we need to define the exploration utility in the same 'currency' as the exploitation function. For example, for variable exploitation model we would need to use active learning for regression techniques [Burbidge et al., 2007, Cohn et al., 1996] to quantify exploration utility. Instantiating the proposed framework for variable and markovian exploitation model is interesting future direction. In the rest of the thesis, all the assumptions and details are mentioned assuming a 'uniform' exploitation function.

Evaluation Metric

Use-it-or-loose-it fixed budget for each iteration: There is one assumption that we have made for evaluating the framework which was already introduced in section 3.2.1. We reiterate the assumption here for completeness. There can be two typical business setups for the interactive learning systems, first is that the expert/auditors are available on demand and even though the maximum allowed budget is specified the system may choose to not use all of it if it expects negative value. Thus the net realized benefit of the system is the exploitation value (total dollars saved in case of fraud) minus the cost of labeling the example. The second setup is fixed cost setup where the budgeted time/cost of experts is incurred irrespective of whether they utilize the time for reviewing or not as they are dedicated to this task only and can't spend their time on other tasks. For example, if there are no credit card fraud transactions for 1 day, the experts still need to be paid their hours even though they did not review any transactions. Thus the Return on Investment metric may not need to include the cost component in this case (as it is fixed) and is simplified to be the cumulative exploitation value of the transactions reviewed. In this work, we assume the latter setup of use-it-or-loose-it budget.

4.1.2 Assumptions in modeling choices for the framework

The modeling choices for the framework are to be made for: Cost model, Exploitation model, Exploration model, Utility Metric and Joint Optimization Algorithm, as described previously in

section 3.2. We make the following assumptions for each of those modeling choices.

Cost model

Cost function is given: We assume that for modeling we are given the true cost function. As mentioned previously that determining the true cost of labeling an example is an open area of research [Arora, 2012], thus we make this simplifying assumption of availability of true cost function to us. In practice, this may be a very challenging assumption to make but we chose to simplify and use the true cost function for our experimentation. It would be a great future direction to relax this assumption and learn a timing model similar to an exploitation model and evaluate its efficacy as well as the impact of inaccurate timing estimates on the overall framework's performance.

Exploit model

Probability estimate from exploitation model: We assume that the exploitation model is able to give us a likelihood estimate for an example to be positive in order for us to calculate the expected exploitation value. This assumption does restrict the choice of algorithms that can be used with the framework but is not overly restrictive as algorithms like logistic regression, SVM, naive bayes are able to give likelihood estimates. However, we have faced some practical challenges in using the SVM algorithm with the framework because of this requirement of likelihood estimates which we have discussed in section 5.4.

Exploration model

There are two assumptions that we have made while making the choice for the exploration models.

Estimating Error Reduction is feasible and reasonable learning strategy: First assumption is that active learning strategy based on estimated future error reduction [Lindenbaum et al., 2004, Roy and Mccallum, 2001, Schein and Ungar, 2007] is a reasonable active learning strategy for real-world tasks. As noted by Settles [2012], certain variants of this general strategy [Guo and Greiner, 2007] have worked reasonably well for several problems in the UCI repository as well as it can be utilized for optimizing any general metric of interest like precision, recall, f-measure. It has also been noted that it is often impractical because it is computationally expensive however Guo and Greiner [2007], Roy and Mccallum [2001] have used sampling and approximate training techniques to make it computationally feasible. Given our particular experimental setup of interactive classification, which is described in the next section 4.2, we have been able to practically execute these strategies.

Ignoring submodularity for batch learning: Our second assumption is based on the fact that the estimated future error reduction strategy is computationally expensive. In order to execute the strategies efficiently, we ignore the submodularity considerations for exploration utility calculation in batch-mode active learning. As mentioned by Settles [2012], Submodularity [Nemhauser et al., 1978] is a property of set functions that intuitively formalizes the idea of 'diminishing' returns.That is, adding some instance x to the set A provides more gain in terms of the utility function than adding x to a larger set A', where $A \subset A'$. Informally, since A' is a superset of A and already contains more data, adding x will not help as much. Formally, a set function s is submodular if it satisfies the property: $s(A \cup x) - s(A) \geq s(A' \cup x) - s(A')$, or, equivalently: $s(A) + s(B) \geq s(A \cup B) + s(A \cap B)$, for any two sets A and B.

It is a common practical assumption made in error and variance reduction batch-mode active learning [Settles, 2012], to ignore the submodularity due to computation cost. Even though it has been shown by Guo and Schuurmans [2008] that simply selecting the 'Q-best' examples to label may be worse than random batch selection for some datasets, most notably 'australian' and 'crx' from UCI repository, there are other datasets like 'glass2' where it is better than proposed batch query by the authors. Whereas for some other datasets like 'vote' there is no difference between the strategies. Thus we chose to ignore the submodularity consideration. However a very interesting future direction is to integrate the exploration utility calculation based on estimated error reduction as a submodular function and utilize it efficiently within the Submodular Function Optimization toolkit [Krause, 2010] that we currently use for the framework.

Utility metric

Utility of exploration in terms of NPV is calculable: For calculating the utility metric, the first assumption is that we are able to calculate the exploration utility in dollars, equation (3.5) and then are able to also calculate its Net Present Value using equation (3.4). The assumption for calculating the exploration utility is linked to being successfully able to instantiate the estimated error reduction strategy [Lindenbaum et al., 2004] for the metric of interest which is Precision@K, equation (3.6). We propose two approximate ways to estimate the Precision@K metric for the dataset using historical labeled data and an unsupervised method estimating the Precision@K using the equation (3.7). Since we have been successfully able to instantiate the exploration utility in our experiments this is a reasonable assumption.

Equal weightage to combine explore and exploit utilities for overall utility: Second assumption that we are making is that the dollar value of exploration and exploitation utilities are combined linearly to get the overall utility, equation (3.1). We can envision more complex/non-linear combination of the two utilities in future, however, a simplistic linear combination of exploration and exploitation is intuitive as we are able to calculate both the utilities in terms of same 'currency' (dollar value) using equations (3.2) and (3.3).

Optimization algorithm

For Budgeted Max Cover: utility and cost is well determined: The assumption that we are making for utilizing the Budgeted Max Coverage optimization algorithm in the framework is that the utility of exploration and exploitation are computable in terms of dollar amounts as well as the cost of labeling an example is available. Currently the framework is not able to utilize learning strategies like uncertainty sampling as there is no mechanism for converting that uncertainty sampling measure into an equivalent dollar amount. So the choice of Budgeted Max Cover optimization algorithm is tied to using the estimated error reduction strategies which can be quantified with a dollar amount as mentioned in the previous subsection.

4.2 Experimental Setup for the Framework

4.2.1 Methodology

Figure 4.1 describes the general process flow for interactive classification. The process is that the system periodically provides the experts with a ranked list of cases to review and verify. Depending on the problems, time is discretized into periods (e.g. days) and of all the new data

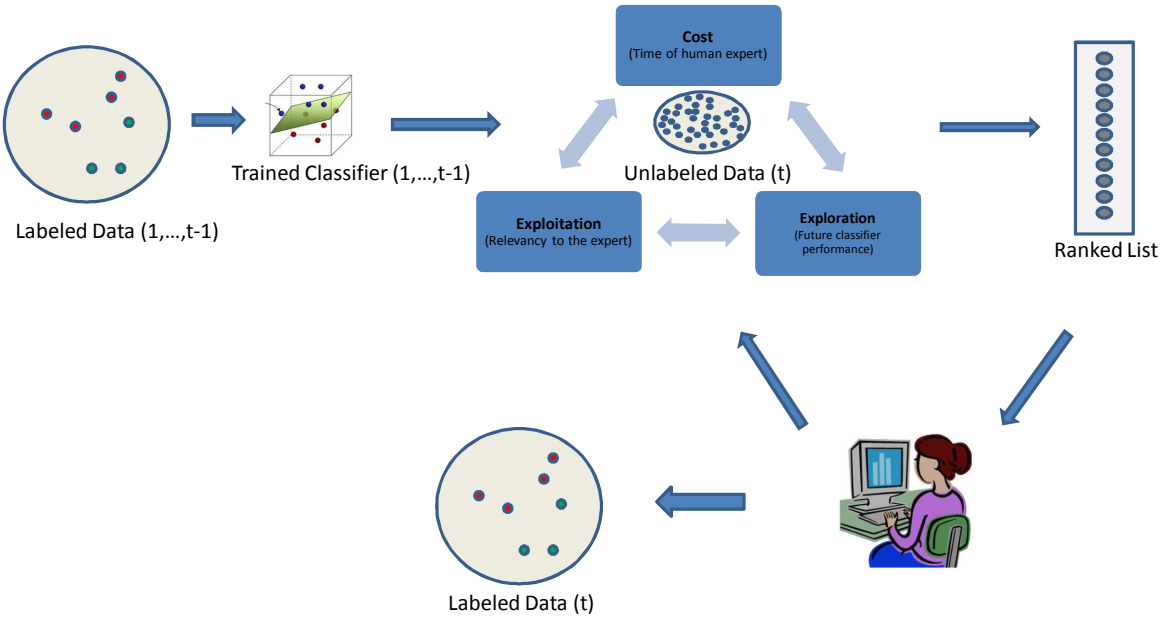


Figure 4.1: Interactive Classification Process Flow

that comes in during that period, a subset of it is labeled based on the system's recommendation. For instance, in information filtering task recommendations can be provided every hour or day or week. The number of examples that the expert can review/label is also typically determined by the domain. For example, for health insurance claims error prediction Kumar and Ghani [2011], the number of audited cases are determined by the daily audit capacity of the insurance company which is close to 2% of all new claims that come in a day. There can be multiple settings for such temporally evolving unlabeled pool of examples. *cumulative streaming pool* setting where new unlabeled examples keep coming in and get added to the streaming pool, thus increasing the unlabeled pool available to the learner. The second setting can be *recent streaming pool* where only the most recent unlabeled examples are available to the learner. *Cumulative streaming pool* setup might be a more appropriate setup for tasks where the value of an example doesn't degrade over time and there is no time-sensitiveness in processing an example for example chemo-informatics task of identifying compounds that interact with AIDS virus. *Recent streaming pool* is more appropriate for tasks where there is time sensitiveness in processing an example for example in information filtering, health insurance claims processing. In the current work, we only experiment with the *recent streaming pool* setting for all the tasks and leave the *cumulative streaming pool* setting for future work. This *recent streaming pool* corresponds to the *Daily Classification Task* setup recommended by Forman [2006] for studying concept drift.

This *recent streaming pool* or *Daily Classification Task* experimental setup is different from the classic active learning setup that is used in existing active learning literature [Settles, 2012]. Most of the active learning work assumes a static unlabeled pool of data from which the examples are selected to be labeled. There has been some work in active learning on streaming data [Chu et al., 2011, Žliobaitė et al., 2011, Zhu et al., 2007]. The key difference between streaming data and our focus is that in the streaming data setup, instances are streaming in to the system and a decision needs to be made right away whether to ask for a label or not. The incoming unlabeled data

	Variation	Parameters
Cost Model	Uniform	
	Variable	
	Markovian	Similarity Threshold per Class
		Cost reduction factor
Exploit Model	Uniform	Parameters for Logistic Regression model
Explore Model	Uniform	
	Variable:Density	
	Variable:Sparsity	
	Markov:Uncertainty	
	Markov:Certainty	
	Markov:Certainty(balanced)	
	Markov:Error Reduction (labeled history)	Evaluation metric of choice
	Markov:Error Reduction (unlabeled activepool)	Evaluation metric of choice
	Markov:Error Reduction (oracle)	Evaluation metric of choice
Utility Metric	Joint Rank based on Individual factor ranks	
	Combined Exploration and Exploitation utility	Exploration utility(NPV Calculation) - Time period to amortize
		Expected number of examples audited (based on budget)
Optimization Algorithm	Rank Fusion	Constant K used in fusion function
	Budgeted Max Cover: Greedy	
	Budgeted Max Cover: Approximation	

Table 4.2: Modeling variations and parameterizations

cannot be stored and queried later. This scenario happens in certain real-world problems (web search for example) but is rare in enterprize problems. In most enterprize interactive data mining systems, data needs to be stored anyway for other purposes (auditing for example) and the constraint of making a labeling decision instantaneously is not present. Also, in these problems, the domain experts labeling the data are the expensive components of the process and data storage costs often pale in comparison. For these practical reasons, we consider the setting where the unlabeled pool gets augmented with new data coming in, which is different from the two extreme settings of fixed unlabeled pool and completely stream-based setup with no memory. Chu et al. [2011] also mention that a periodically changing unlabeled pool is a more realistic scenario than the two extremes of static unlabeled pool and online streaming data.

4.2.2 Modeling Choices and Parameterizations for Experimentation

As described in section 3.2, there are six components of the framework, namely: *Exploitation Model, Exploration Model, Reviewing/Labeling Cost Model, Evaluation Metric, Utility Metric* and *Joint Optimization Algorithm* to optimize the utility metric. The domain specifies the Cost function, Exploitation function and Evaluation metric, however while modeling we may not necessarily have complete knowledge of the cost or exploitation function, thus we need to model cost and exploitation functions. Whereas evaluation metric is usually known and is determined by the business problem. Thus we need to make choices for the following: Exploitation Model,

Exploration Model, Cost Model, Utility Metric and Joint Optimization Algorithm. We mention below the details of the variations, along with their parameterizations, that we experimented for each of these models. They are conceptually summarized in table 4.2.

Cost Variations

As mentioned in section 2.5.3, there are three broad variations of cost model with *uniform, variable or markovian* cost. For all the tasks mentioned in previous section, the domain cost function is given to be markovian. We experiment with all the three cost variations to analyze the effect of modeling markovian cost functions with simpler approximations of uniform and variable cost models. There are no parameters required for the uniform or variable cost model.

For the markovian cost model, there are two parameters. The first parameter is the *similarity threshold* that determines whether two examples are similar enough to enable markovian cost reduction. We need threshold parameter per class as the domains of interest have skewed class distribution. If we parameterize with a single threshold for the entire dataset then the challenge is that the number of sets with more than one positive example are very few. Detailed examples of the similarity threshold is mentioned in section 5.2.3.

The second parameter is the *cost reduction factor* for two similar examples i.e. how much will the cost of auditing an example reduce if it is similar to the previously audited example. We assume that the cost reduction factor is a constant factor independent of the degree of similarity above the threshold. We initially experimented with different cost reduction factors but subsequently fixed this factor to be 0.6 in our experiments. This empirical value was chosen based on the audit time observations in the Health Claims Error Prediction domain [Kumar and Ghani, 2011], where we observed 36% relative reduction in audit time for one of the client pilots. To illustrate this cost reduction scheme using the sample sets from figure 3.1, we note that the cost for set S1 would be 2+0.6*(2)=3.2, for S2,S4: 1+0.6*(1+1+1+2+2+2)=6.4, for S3: 1+0.6*(1+2+2)=4 and for S5=1+0.6*(1+1+1+1+1+2+2)=6.4. In section 5.3.1 we discuss the results of the various cost models. In the future, we can relax this assumption of constant factor reduction and have the reduction factor be dependent on the degree of similarity between examples. Additionally we can consider other complex scheme like exponentially reducing cost based on the markovian order and the similarity. The framework is able to inherently handle all these variations as the set cost calculation is an independent step and does not have any inter-dependence on the optimization algorithm.

Exploit Variations

We experiment with only one exploitation setup of *uniform* value which corresponds to boolean classification as all the tasks that we consider are boolean targets. Under this setup, every positive example has a uniform (fixed) value for example every relevant document is equivalent in Information Filtering task. We have used logistic regression classifier for our experiments. We also tried using an SVM classifier but it wasn't effective because it doesn't inherently produce probability values but rather the SVM score needs to be transformed into a probability. Although there are approaches in literature [Lin et al., 2007, Wu et al., 2004] for estimating it but the interplay of estimating the probability score along with estimating future error reduction active sampling measure didn't work well. More discussion on why we were not able to use SVM algorithm and how can we potentially use it in future is in section 5.4.

We use the liblinear package [Fan et al., 2008] for classification. The parameters for the exploit model are the classic classifier parameters for logistic regression. They are:

- *Choice of solver* (parameter -s in liblinear package): We have used the primal L2-regularized logistic regression (option -s 0 for the liblinear package, version 1.93). We experimented with L1-regularized logistic regression as well as dual formulation of L2-regularized logistic regression but found the primal L2-regularized logistic regression to be faster than dual and better performing than L1-regularized.
- *Misclassification cost* (parameter -c in liblinear package): We have experimented with this parameter with results in chapter 5.
- *Class dependent weight for misclassification cost* (parameter -wi in liblinear package): We have experimented with this parameter with results in chapter 5.
- *Tolerance for stopping criterion* (parameter -e in liblinear package): We have fixed it as 0.001 for all our experiments.
- *Bias term* (parameter -b in liblinear package): We have fixed it as 1 for all our experiments.

Explore Variations

We experiment with nine variations for exploration corresponding to the three broad variations of uniform, variable and markovian models, as described in section 2.5.2.

Uniform The uniform variation corresponds to the *random* sampling strategy.

Variable We use two strategies that uses the dataset density distribution.

- *Density sampling*: We compute the pair-wise Euclidean distance for all the data points in the unlabeled pool, and selects cases with the lowest average distance to all the other examples [Settles, 2009]. It is labeled as 'variable:density' in figures and results.
- *Sparsity Sampling*: This strategy is opposite to the traditional Density sampling. In this sampling strategy, we select cases with the largest average pair-wise Euclidean distance from all the other cases and thus samples from the least dense areas. It is labeled as 'variable:sparsity' in figures and results.

Markovian Markovian Exploration models are based on order-independent labeled historical data.

- *Uncertainty sampling*: This is the classic active sampling strategy of *Uncertainty sampling* which uses the classifier trained on the historical labeled data and selects the cases that the classifier is most uncertain i.e. closest to the decision boundary for Logistic regression or SVMs. It is labeled as 'markov:uncertainty' in figures and results.
- *Certainty sampling*: This strategy is opposite of *Uncertainty sampling* where we sample the most certain examples from both positive and negative class equally i.e. the examples which are farthest from the decision boundary for both classes. It is labeled as 'markov:certainty' in figures and results. Certainty sampling has been shown to be effective in skewed problem domains by Ferdowsi et al. [2011].
- *Classifier sampling*: This strategy selects the most confident positive examples. This corresponds to labeling the examples based on the classifier's recommendations. It is labeled as 'markov:classifier' in figures and results.
- *Estimated Error Reduction*: We estimate the expected reduction in future error and select the examples which maximally reduce the future error in this strategy. There are three

variations of the strategy based on how we estimate the future error. The details of the strategies are mentioned in section 3.2.4 and are summarized below. The input parameter for all the variations is the *evaluation metric* that we are trying to optimize which is Precision@Audit percentage for our case instead of 'accuracy' as used by Lindenbaum et al. [2004], Roy and Mccallum [2001], Schein and Ungar [2007].

- *Estimated Error Reduction:Supervised estimation using labeled history*: We measure the performance of the classifier learnt on the labeled data D so far. This method measures training data error (self-error). It is labeled as 'markov:error reduction-supervised' in figures and results.
- *Estimated Error Reduction:Unsupervised estimation based on unlabeled active pool*: We estimate the classifier performance on the unlabeled pool, where the current classifiers posterior class probabilities are taken as proxies for the true ones, similar to Roy and Mccallum [2001]. The details are mentioned in section 3.2.4. It is labeled as 'markov:error reduction-unsupervised' in the figures and results.
- *Estimated Error Reduction:Oracle*: As a thought experiment to get ceiling performance for estimated error reduction strategy, we assume that an oracle tells us the labels of the unlabeled pool examples and the future test examples for calculating the exploration utility. It is important to note that the oracle information is used for calculating the exploration utility only and not other modeling choices. It is labeled as 'markov:error reduction-oracle' in the figures and results.

Utility metric Variations

We experiment with broadly two variations of the utility metric. First variation is where we use *rank fusion* to combine the rank orderings based on cost, explore and exploit and thus get an implicit utility. It is important to note that rank fusion utility is not defined for markovian cost case as the ranking is not deterministic based on the markovian cost. We use the Reciprocal Rank Fusion technique [Cormack et al., 2009] recently shown to outperform the classical rank fusion techniques like Condorcet [Montague and Aslam, 2002] and others. The exact rank fusion equation from Cormack et al. [2009] is:

$$RRFscore(x \in X) = \sum_{r \in R} \frac{1}{k + r(x)} \tag{4.1}$$

where x is any example belonging to the example collection X, r(x) is the rank given to the example x by ranker r out of all the rankers R (cost, explore and exploit). The parameter for the reciprocal rank fusion technique is the *constant term 'k'* in the equation (4.1). We fix it to 60 as used in the original paper. The reason is that we initially experimented with varying the constant term 'k' in our experiments with oracle knowledge to determine the ceiling performance and selected value that maximized the performance of Rank Fusion setup. However when we used the selected constant term in our modeling experiments it performed worse than using the baseline choice of 60 for the constant term 'k'. Thus we decided to use the recommended value of 60 for the constant term as the authors [Cormack et al., 2009] have also empirically tested the factor across a large number of queries to come up with their final recommendation.

Second variation is where we have an *explicit utility function* as defined in the previous subsection combining the exploitation and exploration utility, equation (3.1). For evaluating equation (3.1), the different parameters are:

- Weight factor α to balance exploration versus exploitation utility.
- Time period to amortize the Net Present Value of exploration utility in equation (3.4).
- Expected number of examples audited per time period, which corresponds to the budget per time iteration, in equation (3.4).
- Choice of estimating the future error reduction in equation (3.6) using either labeled history or unsupervised estimation or true labels.

Optimization Algorithm

We have experimented with three variations of optimization algorithm. The first variation is the simple *rank ordering* based on the fused rank scores utility. There are two variations within the Budgeted Max Coverage optimization setup, namely, *Greedy* approach and *Khuller et al's approximation* algorithm which are detailed in section 3.2.6. There are no parameters for the budgeted max coverage optimization algorithm.

4.2.3 Evaluation

Figure 4.1 describes the general process flow for interactive classification which is analogous to the *Daily Classification Task* introduced by Forman [2006] for studying concept drift. The process is that the system periodically provides the experts with a ranked list of cases to review and verify. Depending on the problems, time is discretized into periods (e.g. days) and of all the new data that comes in during that period, a subset of it is labeled based on the system's recommendation. For instance, in information filtering task recommendations can be provided every hour or day or week. The number of examples that the expert can review/label is also typically determined by the domain. For example, for health insurance claims error prediction Kumar and Ghani [2011], the number of audited cases are determined by the daily audit capacity of the insurance company which is close to 2% of all new claims that come in a day.

As mentioned in section 3.2.5, we use the setup of incurring fixed cost equal to the budget in each time period. Thus the evaluation metric of Return on Investment is simplified to be the exploitation value of the examples labeled cumulatively over time periods. Since the goal of the system is to maximize the long term Return on Investment of the systems we compare the different setups of the framework at the end of all time periods (i.e. after exhausting all the data from the datasets).

For reporting the results, we present them as the percentage of theoretically best possible oracle performance. The theoretical best oracle performance is calculated by assuming that the oracle tells us the labels for all the data as well as the cost of each example. Thus given the budget constraint, we can maximally label as many positive examples as possible [1]. There are two motivations for presenting the results using this relative representation. Firstly, this relative measure allows us to compare the results across the datasets/tasks as the budget/cost for each dataset may be different which means that the total number of examples that are labeled are different for each dataset. The relative measure is bounded between 0 and 1 for all datasets. Secondly, this gives us a sense of how well the proposed framework is performing in absolute

[1]The theoretical oracle ceiling performance calculation is still NP-hard problem mapping to budgeted maximum coverage that requires efficient approximation algorithm as proposed by Khuller et al but is simplified in it's formulation as the absolute utility and cost are known.

terms and quantifies the possible scope for improvement. It gives us an indication of the dataset complexity and the hardness of the problem from a modeling perspective.

$$RelativeROI = \frac{\sum_{x \in D} l_x}{\sum_{x \in D'} l_x} \tag{4.2}$$

where D is the set of examples labeled by any given strategy and D' is the set of examples based on the oracle. $l_x \in 0, 1$ is the label for the example x.

Statistical Comparison Methodology

While comparing the performance across multiple strategies we use the following methodology for statistical comparison. All the experiments are run over 10 randomized data samples. We use two-way analysis of variance (function ANOVA2 in Matlab) omnibus test [Japkowicz and Shah, 2011] where the two sources of variance are the different active strategies being compared and different data samples. We do a post-hoc Fisher's least significant difference test to identify the statistically different strategies while adjusting for multiple comparisons (we use function multcompare in Matlab). If the preliminary test i.e. two-way ANOVA F statistic doesn't show a significant difference ($p < 0.05$) then we fall back to 'Bonferroni' adjustment to compensate for multiple comparisons as Fisher's test is not applicable in such situations. The reason for using Fisher's test, where it is applicable, rather than using Bonferroni correction for all the cases is that it is less conservative in it's adjustment for multiple comparisons. In the rest of the thesis whenever we mention that we statistically compare the strategies, we use the process as described above. In the interest of not repeating the methodology every time, we will refer to this section for the methodology description.

4.2.4 Baseline Approaches

In order to compare the proposed framework with the existing approaches in literature, we consider multiple baseline approaches. These baseline approaches can also be defined in terms of simpler combinations of factors in our framework.

1. Random: The simple baseline approach of randomly selecting examples to label.
2. Classification/Exploitation: The approach of deploying a learnt classifier to label the examples. This approach takes into account only one factor of exploitation utility and ignores the other factors of cost and exploration.
3. Active Learning/Exploration: This is the traditional active learning approach where the examples are labeled based on their exploration utility without taking into account their cost or exploitation utility.
4. Minimizing Cost: Another simple baseline is to label the examples in the reverse order of their cost in order to maximize the number of examples that can be labeled with a given budget.
5. Cost sensitive active learning: Cost-sensitive active learning approach by Settles et al. [2008], Wallace et al. [2010], where they use a notion of ROI to select the next example can be mapped to the setup with variable cost model and BMC-greedy optimization algorithm in our framework. Thus it can be used as another baseline to compare.

Chapter 5

Case Studies to Test the Interactive Framework

5.1 Instantiating the Interactive Classification Framework

We first describe tasks for which we used the interactive classification framework. For each of the tasks we mention the details of the domain inputs in terms of exploitation setup, cost setup and budget. We then describe in detail the choices in the framework in the following subsection.

5.1.1 Task 1: Health claims error prediction

This task deals with the problem of predicting (and reducing) payment errors when processing health insurance claims which belongs to the same class of problems as fraud detection, intrusion detection, and surveillance. The goal is to minimize errors by predicting which claims are likely to have errors, and presenting the highly scored ones to human auditors so that they can be corrected before being finalized and paid.

We use the data from a large US health insurance company which had approximately 40 million claims spanning 12 months. Of these 40 million claims, 150000 had been manually audited and found to be either Error or not. In this set, around 80% claims were Error and the rest Correct. It is important to remember that this distribution does not hold in the operational world since the labeled data has a skewed class distribution. For our experimental setup of *Daily Classification Task* we create the data samples for different time periods. For creation of this dataset, the number of known correct examples is the bottleneck as there are only 30000 correct versus 120000 error claims. We create data samples of 1100 claims per iteration with 5% error(55) and 95% correct(1045) claims. This sample size enables us to have 22 active iterations.

For feature construction, there are four classes of information in each claim: Member information, Provider information, Claim Header, and Claim Line Details. Member and Provider information span the entire claim and provide information about the Member (patient), and the

Dataset	Total Datapoints	Positive Class	Number of Features	Sparseness	Iteration Batch Size	Total Number of Batches
20 Newsgroup	18774	10%	934	1.2%	1000	18
Reuters-RCV1	20000	10%	9007	0.8%	1000	20
HIVA	42687	3.5%	1617	9.1%	2000	21
Claims	150000	5%	12289	0.6%	1100	22

Table 5.1: Summary of datasets

provider (hospital, doctor, or medical facility). The claim header gives information about the entire claim as well. Contract information, Amount billed, Diagnosis codes, Dates of service are some examples of data fields in the claim header. The last source is the Claim Line Details. This gives the details of each line in the claim which is used to itemize the claim for each procedure that was conducted on the patient. For each procedure, the claim line includes the amount billed, the procedure code (CPT), the counter for the procedure (quantity). Since we are currently focusing on predicting the likelihood of the entire claim as being rework, we aggregate claim line details to the overall claim level. For example, we create features using Min, Max, Average and standard deviation functions for each numeric data field in each line. We also create more aggregate features that are specific to procedure codes. We derive some more features that are specific to the claims processing domain. For example, we calculate the time (in days) between the date of service and the date of submission of the claim. The intuition for this feature is to figure out if the claim is valid as there is a time limit within which the claims should be filed with the plan and also to see if there is a correlation between Rework and late/early submission of claims. Since linear classifiers (Logistic Regression/SVM) are not able to handle categorical data we create binary features from the categorical features resulting in around 215,000 features. We used a frequency-based feature pruning technique to remove features present in less than 10 claims which reduces the feature set to 12,000. The data characteristics are summarized in table 5.1 for all tasks.

Domain Inputs: The *exploitation setup* is uniform model i.e. all the claims which are erroneous have equivalent value as we save the same amount of dollars for each claim which reflect the administrative processing overhead for correcting the error. It is a boolean classification task. The *cost model* is markovian i.e. the cost of labeling similar claims is lower as it reduces the cognitive switching as well as provides a line of investigation to the auditors. This was experimentally observed by Kumar and Ghani [2011] as well. We use the observed labeling time of the claims from the user studies as the cost for labeling a claim. We impute missing values for the labeling time of a claim by using the average time to audit for a given error code (each labeled claim is given a detailed error code). The *budget* is determined by the total audit time available for running the experiments. We experiment with budget for labeling 2% of the data in each iteration.

5.1.2 Task 2: Chemo-Informatics

In chemo-informatics task, the aim is to predict chemical activity of molecules. It is a boolean classification problem. The variables represent properties of the molecule inferred from its structure. The problem is therefore to relate structure to activity (a QSAR=quantitative structure-activity relationship problem) to screen new compounds before actually testing them in-vitro(a HTS=high-throughput screening problem). For HIVA, the task is to identify compounds that are active against the AIDS HIV infection. We use the HIVA dataset [Guyon, 2011] which was used in Active Learning Challenge 2011 [Guyon et al., 2011]. There are 42,678 compounds in the dataset with 1503 compounds that interact with the aids virus. Thus the positive class percentage is about 3.5% which is highly skewed.

For the HIVA dataset, the data was made available by the National Cancer Institute (NCI), via the DTP AIDS Antiviral Screen program available at: `http://dtp.nci.nih.gov/docs/aids/aids_data.html`. The DTP AIDS Antiviral Screen has checked these compounds for evidence of anti-HIV activity. Available are screening results and chemical structural data on compounds that are not covered by a confidentiality agreement. The results of the screening tests are evaluated and placed in one of three categories: CA - Confirmed active, CM - Confirmed moderately active, CI - Confirmed

inactive. This was converted into a two-class/boolean classification problem: Inactive (CI) vs. Active (CA or CM). Chemical structural data was converted to structural features by the program ChemTK version 4.1.1, Sage Informatics LLC. 1617 structural features were retained after some pre-processing. Detailed description of the features and the feature selection process can be found in [Guyon, 2011].

For our experimental setup of *Daily Classification Task*, we divide the compounds in time periods by assigning 2000 compounds for each time period and maintaining the positive class skew at 3.5%. Thus in each time period there are 70 positive compounds. For the experiments, we report the results averaged over 10 random data sampling.

Domain Inputs: The *exploitation setup* is uniform model i.e. all the compounds which are active against HIV aids virus are equivalent and it is a boolean classification task. The *cost model* is markovian i.e. we assume that the cost of testing similar compounds is lower as there may be experimentation setup time that can be avoided for similar compounds. We use a surrogate measure for the cost of labeling a compound in the number of structural features present in a compound which indicates the complexity of the compound in the HIVA dataset[1]. The *budget* is determined by the total dollars for running the experiments for labeling the compounds and we approximate it with the average number of compounds that can be labeled in an iteration. We experiment with budget for labeling 3.5% data in each iteration.

5.1.3 Task 3: Information Filtering

In information filtering task [Hanani et al., 2001], the system is expected to recommend a set of relevant objects/articles to the user from a large number of objects/articles. More generally information filtering is used to manage the information overload. Some examples of information filtering systems are email-spam filter, recommender systems. We use two datasets for this task. The data characteristics are summarized in table 5.1.

Dataset 1: 20 Newsgroup dataset [Rennie, 2007]: The 20 Newsgroups data set is a collection of approximately 20,000 news documents, partitioned (nearly) evenly across 20 different newsgroups. The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). The simulated information filtering task is that the user is interested in only a few category of news documents and we need to filter documents from other categories.

There are 18,774 documents with 934 features corresponding to 20 news categories after preprocessing and data clean-up. The features are stopped and stemmed unigram word features. For our experimental setup of *Daily Classification Task*, we sample 50 cases each for the 20 categories, resulting in 1000 documents per time period. This gives us 18 time iterations for 20-newsgroup data. We experiment with positive class percentage as 10% (2 out of 20 categories).

Dataset 2: RCV1 - Reuters dataset [Lewis et al., 2004]: For RCV1 dataset, the dataset is preprocessed as described by Bekkerman and Scholz [2008] where the label hierarchy is reorganized by mapping the data set to the second level of RCV1 topic hierarchy. The documents that

[1]The true cost for labeling the compounds is the cost of Antiviral Screen that is conducted to observe the activity against AID virus. National Cancer Institute has made the in-vitro experiment details available on their site, however the HIVA dataset [Guyon, 2011] cannot be directly linked to the compounds on the website as they have been anonymized in order to conduct the machine learning challenges. Thus we use a surrogate cost function. Even with in-vitro experiments the cost model is markovian as there is usually experiment setup cost for different kinds of screenings that need to be run and the cost can be reduced by running similar screenings together

have labels of the third or fourth level only are mapped to their parent category of the second level. The documents that only have labels of the first level are not mapped onto any category. Further, the multi-labeled instances are removed. Out of the resulting 53 second level topics, we select the top 20 most frequent topics and sample 1000 documents for each topic. Thus we have 19 time iterations for RCV1 data. We experiment with positive class percentage as 10% (2 out of 20 categories).

Domain Inputs: The *exploitation setup* is uniform model i.e. all the relevant articles are equivalent. This corresponds to the traditional boolean classification setup. The *cost setup* is markovian i.e. when the user is shown similar articles in-order, it takes him less time to make a determination of relevance. The cost of labeling an article independently is assumed to be its document length [Ringger et al., 2008] for 20 Newsgroup and vocabulary size for RCV1. The *budget* is specified by the time that the user can devote to the task which is translated linearly into the average number of documents that can be labeled in an iteration. We report the results in the following sections for 5% budget. For all the experiments, we report the results averaged over 10 random data sampling and labeling iterations.

5.2 Characterization of the datasets for different tasks

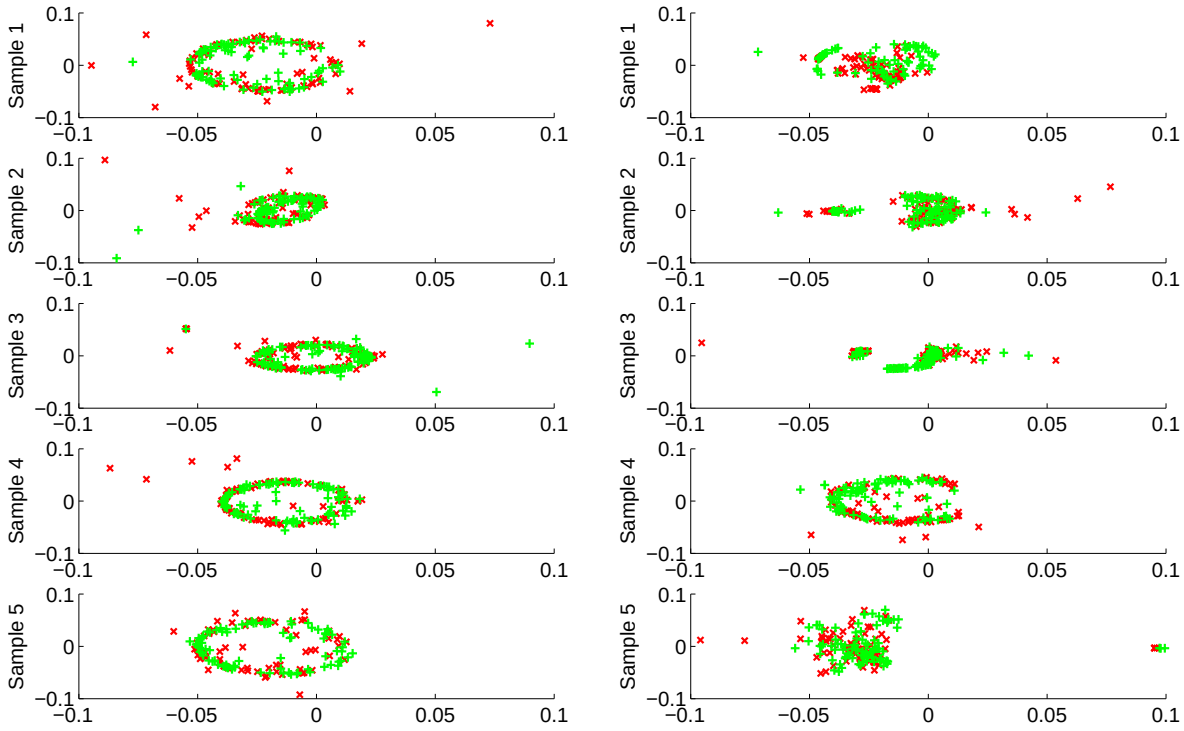
Before experimenting with the framework on the different tasks/datasets, we wanted to characterize the tasks/datasets with two intentions: first, we want to understand the performance of the framework on these tasks i.e. if it works better or worse or doesn't help for a particular task. Secondly, in order to generalize the framework beyond the tasks/datasets that we experimented with we wanted to come up with some generalized observations that can be used to guide the different choices for the framework and help in determining the usefulness of the framework for a future task. Particularly, we are interested in knowing that if we don't know anything about a new task/dataset, is it still a 'safe' choice to use the framework. Are there particular operating ranges in terms of budget that the framework helps more than certain other budget ranges? Are there certain characteristics about the data due to which the framework hurts and why and if it can be detected and overcome?

5.2.1 Visualization of the data

To characterize the datasets we start by initially visualizing the data using Multi-Dimensional Scaling (MDS) to observe the level of similarity of the individual cases in the dataset. An MDS algorithm aims to place each object in N-dimensional space such that the between-object distances are preserved as well as possible. We use N=2 to visualize the data as a scatterplot. We use cosine distance metric for all the datasets as all the datasets/tasks have categorical/boolean features with the exception of some numeric features in Claims dataset. For claims dataset as well since majority of features are sparse boolean features, cosine distance metric is appropriate.[2] We use the Kruskal's normalized stress1 criterion for the optimization using the function 'mdscale' in matlab.

We randomly select 100 positive and 100 negative examples (assuming that we know their labels) from the dataset. Since the datasets/tasks are skewed, ranging from 3.5%-10% in class skewness, a completely random sampling of the data would give biased samples and hence may

[2]For any future task/dataset, appropriate distance metric should be used which should be determined based on the domain.



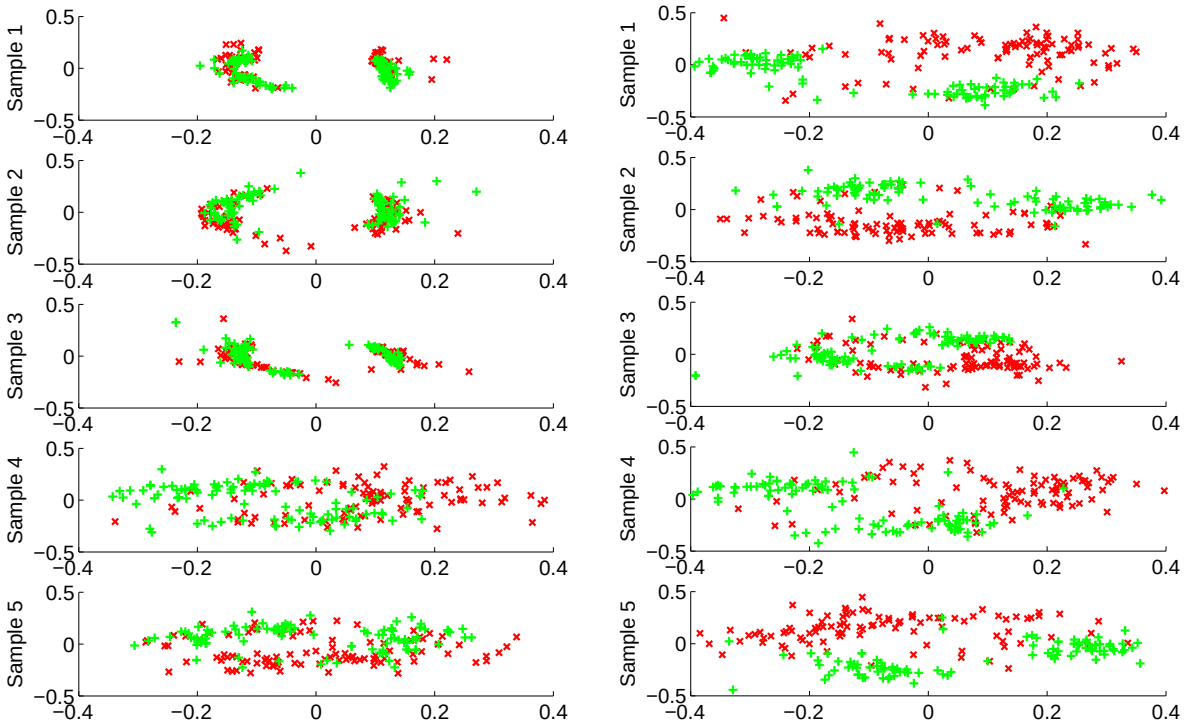
(a) No scaling

(b) Scaled using the feature coefficient

Figure 5.1: Dataset - 20Newsgroup: Multi-dimensional scaling of 200 randomly selected examples (100 positive, 100 negative) with Cosine distance metric. The difference between image (a) and image (b) is the scaling used on features for the cosine distance calculation. (a) No scaling (b) Scaled using the feature coefficient learnt for a logistic regression model using the same data.

not reflect the data characteristics. For these 200 examples we perform MDS and visualize the data in figures 5.1a, 5.2a, 5.3a and 5.4a for 20-Newsgroup, RCV1, HIVA and Claims datasets respectively. We do not do any scaling for this initial visualization and use the features from the dataset for the distance calculation. Figure 5.1a, 5.3a for 20-Newsgroup and HIVA dataset show that the positive and negative datapoints are mostly equidistant from each other and are uniformly distributed across the coordinates. Figure 5.4a shows the data for Claims dataset and we observe uniform distribution for two samples whereas bi-modal distribution with uniformly distributed positive and negative examples for three samples. For the three datasets, 20-newsgroup, HIVA and Claims, this uniform distribution of examples usually means that positive examples are not trivially differentiated from the negative examples. Figure 5.2a for RCV1 dataset shows a bi-modal distribution with both positive and negative datapoints uniformly distributed across the modes for three samples. However, for samples 4 and 5, we notice that there is separation between positive and negative examples. This means that the inter-class distance in smaller than the intra-class distance and hence distinguishing the classes from each other might be fairly easy for this dataset.

In order to visualize the complexity of the dataset, we train a logistic regression model with



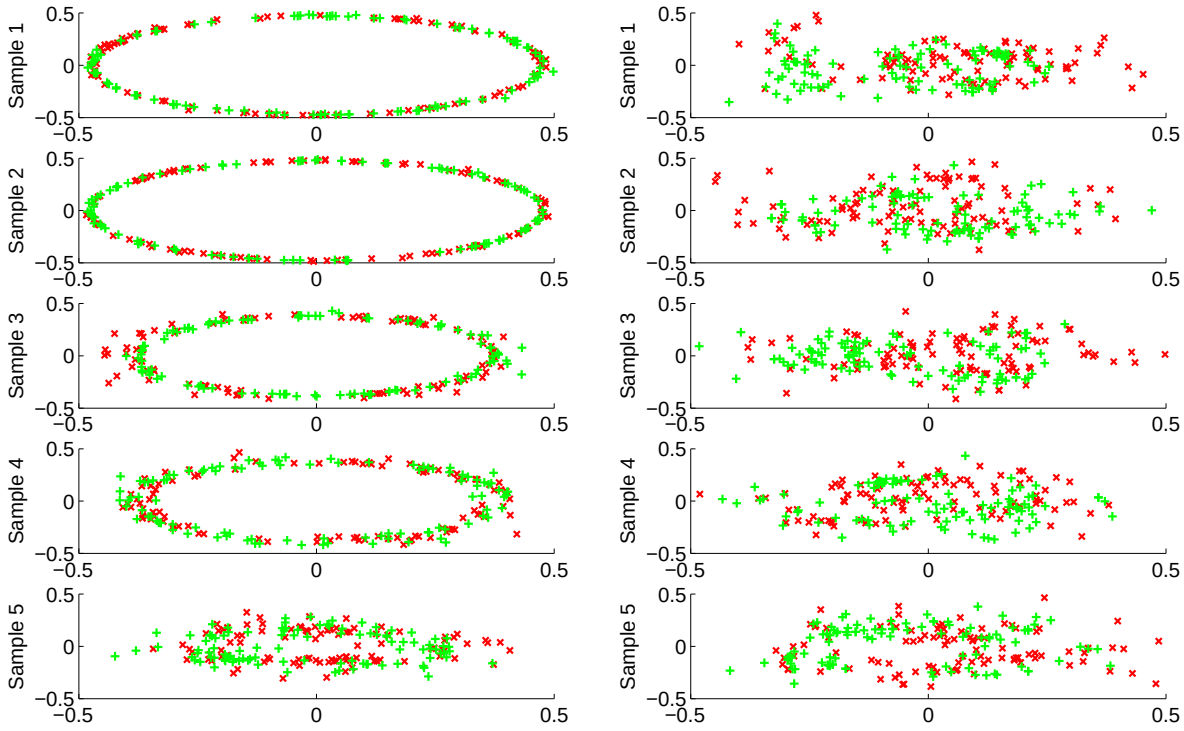
(a) No scaling

(b) Scaled using the feature coefficient

Figure 5.2: Dataset - RCV1: Multi-dimensional scaling of 200 randomly selected examples (100 positive, 100 negative) with Cosine distance metric. The difference between image (a) and image (b) is the scaling used on features for the cosine distance calculation. (a) No scaling (b) Scaled using the feature coefficient learnt for a logistic regression model using the same data.

the 200 datapoints.[3] We modify the features of the examples by multiplying the feature coefficient with the features to obtain the scaled feature vector which gives more weight to relevant features and diminishes the non-relevant features. We again perform MDS with these scaled feature set for the 200 examples for each dataset. We visualize the MDS scaling in figures 5.1b, 5.2b, 5.3b and 5.4b for 20-Newsgroup, RCV1, HIVA and Claims datasets respectively. We observe that for 20-newsgroup, figure 5.1b, and HIVA, figure 5.1b, MDS scaling doesn't change the distribution of the datapoints very much. This indicates that for these two domains a randomly learnt model doesn't have too much distinguishing power between the classes and it is non-trivial to learn a good model. For RCV1 dataset, figure 5.3b, we observe that there is very good separation between the positive and negative examples and it indicates that the problem is relatively easy to model and learn a good classifier with random sampling of datapoints. For Claims dataset, figure 5.4b, the separation between positive and negative examples is not as remarkable as RCV1 dataset but the separation is better than 20-Newsgroup and HIVA dataset. Another observation is that the negative class examples are clustered more compactly together than the positive

[3]We set the logistic regression parameters empirically based on cross-validation performance to maximize AUC, Balanced Accuracy and F-1 score metrics together.



(a) No scaling

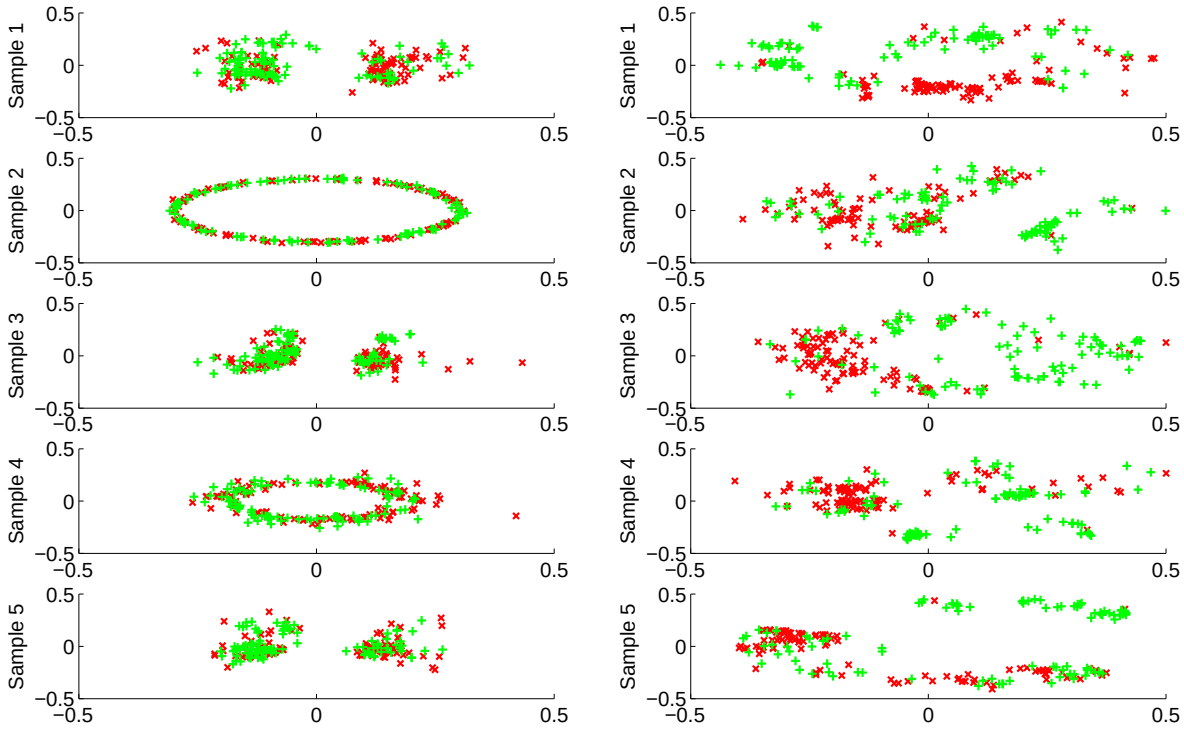
(b) Scaled using the feature coefficient

Figure 5.3: Dataset - HIVA: Multi-dimensional scaling of 200 randomly selected examples (100 positive, 100 negative) with Cosine distance metric. The difference between image (a) and image (b) is the scaling used on features for the cosine distance calculation. (a) No scaling (b) Scaled using the feature coefficient learnt for a logistic regression model using the same data.

examples. This is an interesting observation as it indicates uniformity in the negative examples of the Claims domain which is captured by the learnt model. In our discussions with the Claims domain experts, we found that the process for labeling the negative examples for the domain, which is based on quality audits, is indeed based on rules for deriving the population to be audited hence the negative examples are similar to each other.

Given these observations of the datasets, we can make the following initial hypothesis that for datasets 20-Newsgroup and HIVA intelligent exploration should be valuable to learn a good model and hence the framework should be very helpful. For RCV1 dataset, it is fairly clear that intelligent exploration may not be necessary for learning a good model as it is a relatively easy task. So if the framework can perform in a 'safe' fashion for the domain then it would be helpful. Similarly for Claims dataset, since exploitation is a good exploration strategy, we may not need to take exploration into account explicitly, hence if the framework can again perform in a 'safe' fashion then it would be helpful.

With these initial hypothesis we perform the next step of evaluating the learning curves for the datasets. It is important to note that it may be reasonable to perform initial MDS visualization for a future task with initially labeled data from the first few time periods, however



(a) No scaling

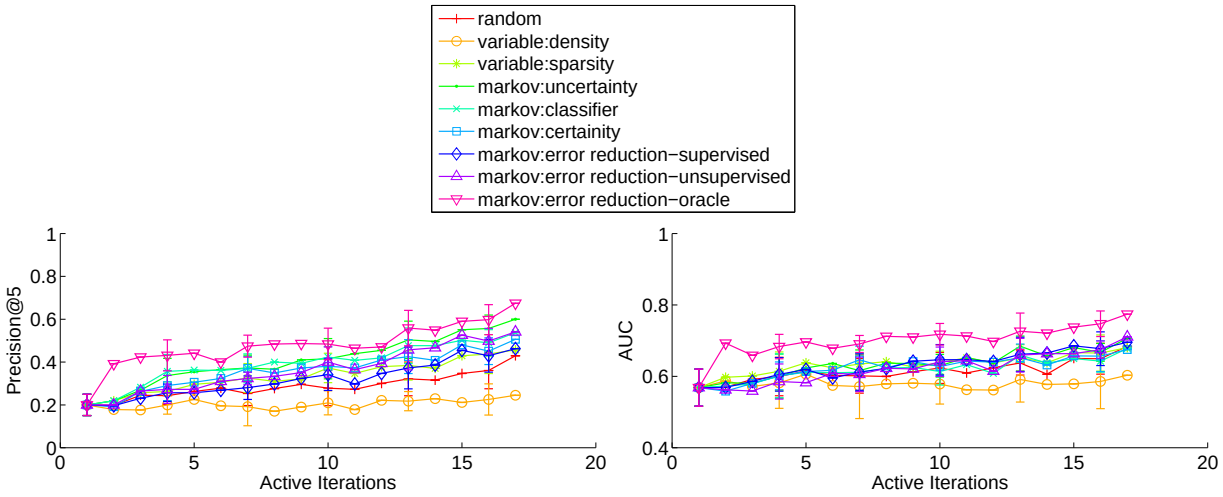
(b) Scaled using the feature coefficient

Figure 5.4: Dataset - Claims: Multi-dimensional scaling of 200 randomly selected examples (100 positive, 100 negative) with Cosine distance metric. The difference between image (a) and image (b) is the scaling used on features for the cosine distance calculation. (a) No scaling (b) Scaled using the feature coefficient learnt for a logistic regression model using the same data.

analyzing the learning curves may not be feasible due to lack of labeled data across time periods for any future domain. The availability of labeled data allows us to perform retrospective analysis like characterizing the learning curve characteristics. It is also important to note a practical consideration that labeling 200 examples with 100 positive and 100 negative may be practically difficult for some domains and an appropriate number of examples should be chosen based on the feasibility for the domain. For example, for Claims domain to get 100 positive examples we will have to completely label (randomly) the first two time periods of data i.e. around 2200 examples to get close to 110 positive examples given that the class skewness is 5%. For HIVA domain, we will have to randomly label 1.5 time periods of examples which is around 3000 examples in order to get 105 positive examples as the class skew is 3.5%. Thus for these two domains a smaller number of examples can be used for doing the analysis.

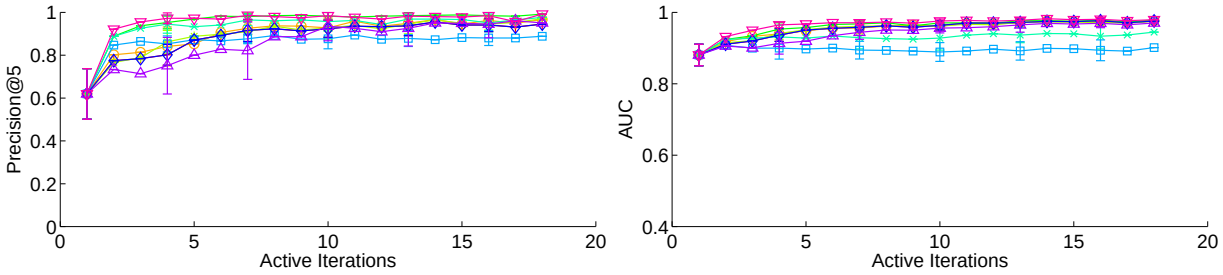
5.2.2 Characterizing learning curves

Before discussing the learning curve characteristics for the datasets, one important consideration to highlight is the difference in our experimental setup compared to the common methodology



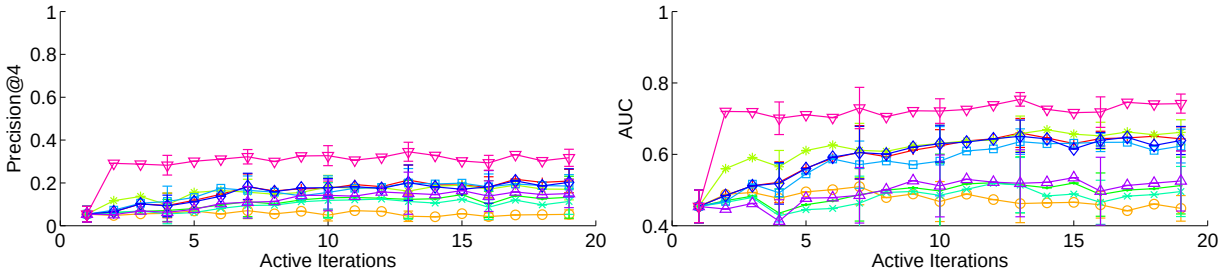
(a) 20-Newsgroup: Metric Precision@5 with fixed batch size of 50 queries per iteration

(b) 20-Newsgroup: Metric AUC with fixed batch size of 50 queries per iteration



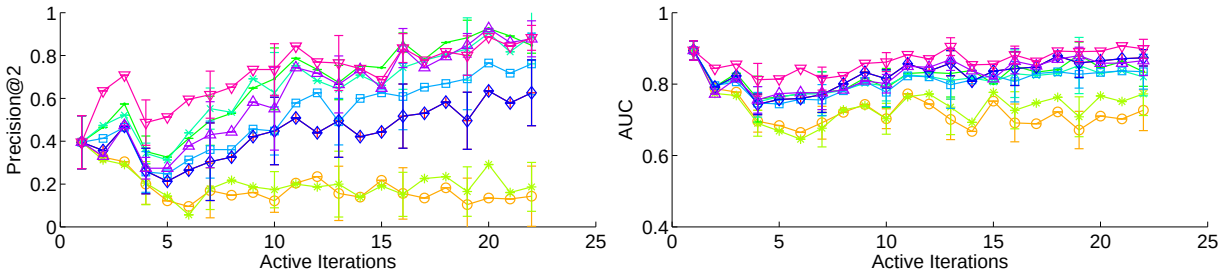
(c) RCV1: Metric Precision@5 with fixed batch size of 50 queries per iteration

(d) RCV1: Metric AUC with fixed batch size of 50 queries per iteration



(e) HIVA: Metric Precision@4 with fixed batch size of 35 queries per iteration

(f) HIVA: Metric AUC with fixed batch size of 35 queries per iteration



(g) Claims: Metric Precision@2 with fixed batch size of 22 queries per iteration

(h) Claims: Metric AUC with fixed batch size of 22 queries per iteration

Figure 5.5: Classic Active Learning Curves for different datasets

	20-Newsgroup		RCV1		HIVA		Claims	
	AUC	Precision@5	AUC	Precision@5	AUC	Precision@4	AUC	Precision@2
Markov:Classifier	-	-	-	-	-	-	-	-
Markov:Error Reduction-Unsupervised	0/1/15	3/0/13	0/10/7	7/0/10	0/0/18	0/0/18	0/2/19	3/0/18
Markov:Error Reduction-Supervised	0/0/16	**13/0/3**	0/13/4	4/0/13	0/16/2	0/5/13	0/3/18	**18/0/3**

Table 5.2: Win/Loss/Tie statistical comparison between 'Markov:Classifier' active sampling strategies and Estimated Error Reduction strategies for AUC and Precision@Audit percentage metrics over the active iterations averaged over 10 random data samples and adjusted for multiple statistical comparisons (4.2.3).

followed in active learning literature [Settles, 2012]. In our setup, after every iteration of labeling the examples based on the active sampling technique, we test the performance of the learnt classifier on next time iteration data. So after every iteration the classifier is tested on a different set of data (from the next time iteration). Whereas in active learning literature the test set typically remains constant throughout the evaluation. The implication of this difference is that in our setup it is not guaranteed to get a monotonically increasing learning curve which is expected in typical active learning setup.[4]

We characterize the learning curve characteristics of the datasets using two metrics. First metric is AUC (Area under ROC curve) which has been shown to be correlated to good rankers to rank positive examples above negative examples [Donmez and Carbonell, 2009]. Second metric that we use is Precision@Audit Percentage which is motivated by the Return on Investment evaluation criterion for the overall system as discussed before in section 3.2.1. Figure 5.5 shows the learning curves for Precision@Audit Percentage and AUC metrics for the four datasets. The scale for all the graphs are same across the datasets with the intention of comparing the behavior across datasets. As observed while visualizing the datasets with MDS, for RCV1 dataset, the performance is really high for all the active learning strategies (including random sampling) with maximum AUC of 0.98 for final iteration. This further supports the argument that it is an easy domain to model. For 20-Newsgroup, figures 5.5(a)(b), and HIVA dataset, figures 5.5(e)(f) we observe that the learning curves are much better with intelligent sampling. Another observation is that HIVA is a really hard domain with the modeling performance relatively lower than other domains. The pattern observed for the Claims dataset is very interesting that there is a large spread in the performance of the different active sampling techniques where the density based strategies ('Variable:Density' and 'Variable:Sparsity') performing worse than random strategy. The maximum AUC across the different datasets for last iteration is: 0.98 - RCV1, 0.79 - 20 Newsgroup, 0.91 - Claims and 0.75 - HIVA for the markovian error reduction strategy based on oracle information.

A significant observation from analyzing the active learning curve characteristics is that the 'Markov:Classifier' strategy which corresponds to labeling top ranked examples based on the classifier confidence is a very good exploration strategy for some skewed class domains. We have observed this phenomenon for multiple datasets in our work Ferdowsi et al. [2011] as well as it has been previously studied in the context of active learning for rare category detection by He and Carbonell [2007]. In rare category detection, [He and Carbonell, 2007] demonstrated that

[4]In practice, even in typical active learning setup the learning curves are seldom monotonically increasing.

once we have labeled a rare positive class example, finding and labeling the nearest neighbors of that example leads to improving the classifier most by learning a more robust model for the rare class. Similarly in Ferdowsi et al. [2011] we found that if the concept was easy to learn like KDD'1999 Cup Network Intrusion detection dataset then classifier based strategy is a good learning strategy as well as for Claims dataset the classifier based strategy works better for Precision at lower audit ranges i.e. classifier based strategy is better if you care about only the top scored examples. Thus in domains where this phenomenon is applicable, we don't need to explicitly explore and are better off saving the resources for exploiting (which implicitly also improves the models by learning better positive class concept).

Before discussing the general patterns observed in the learning curve characteristics, it is important to discuss what patterns are relevant in the context of the interactive framework. We argue that when we observe a learning strategy is not performing well then this pattern is relevant for the framework as we do not expect it to help in conjunction with the framework. However, when a learning strategy performs well it is a good candidate to study in the context of the framework but is still not guaranteed to help as the strategies are combined with other factors and their interrelationships become important and dominant. So a learning strategy that was giving great learning curve performance may in conjunction with the other factors not perform as well. It is not expected that the reverse may happen that a learning strategy that wasn't giving good learning performance becomes helpful in conjunction with the other factors.

So in order to come up with initial hypothesis that whether a learning strategy would be useful in the context of the framework, we compare them against each other. Since the framework with it's Budgeted Max Coverage formulation uses estimated error reduction strategies (markov:error reduction-supervised and markov:error reduction-unsupervised) and classifier strategy ('markov:classifier'), we compare them amongst each other. Table 5.2 shows the comparison of the 'markov:classifier' learning strategy with estimated error reduction strategies. We observe different patterns for each dataset. For 20-Newsgroup, we observe that 'markov:classifier' is mostly tied with the error reduction strategies for AUC metric while better in three iterations over unsupervised error reduction and in thirteen iterations over supervised error reduction for Precision@5 metric. So we estimate that supervised error reduction may not perform well for 20-newsgroup data but other strategies should be good. For RCV1 dataset it is interesting that 'markov:classifier' is better or tied for the Precision@5 metric but is worse or tied for the AUC metric. Since Precision@5 metric is practically more important for the labeling and hence the framework we can say that it is significantly better than error reduction strategies. For HIVA dataset, it is tied to the unsupervised error reduction strategy for both metrics but is significantly worse than supervised error reduction for AUC and Precision@4 metric. Considering the absolute performance level of the 'markov:classifier' learning strategy, it seems unlikely that implicit exploration would help and also that explicit exploration might help for HIVA dataset. For Claims dataset, the 'markov:classifier' strategy is similar or worse than error reduction strategies for AUC metric but for the Precision@2 metric it is significantly better or tied to the error reduction strategies. As with the RCV1 dataset since Precision@2 metric is practically more important in the context of the framework we expect that implicit exploration with the 'markov:classifier' strategy might work better for the claims dataset.

The implication of these observation for our framework is that for certain tasks/datasets the best performing configuration may not need to explicitly model exploration utility. We are calling it implicit exploration using the classifier. The choice of the final learning strategy is still an empirical question that needs to be determined for each task/datasets but the framework

20-Newsgroup		RCV1		HIVA		Claims	
Sim. Thres.	Max Clique Size	Sim. Thres.	Max Clique Size	Sim. Thres.	Max Clique Size	Sim. Thres.	Max Clique Size
0.001	7	0.7	4	5	5	3	10
0.005	15	0.8	13	5.5	7	3.2	12
0.01	19	0.85	19	6	10	3.5	13
0.02	33	0.9	24	7	20	3.7	15
0.03	44	1	53			4	22

Table 5.3: Variation of maximum clique size with different similarity thresholds across datasets.

Dataset	Similarity Threshold	Batch Size	Average Number of Sets per Iteration	Percentage of Sets With More Than 1 Positive Example
20-Newsgroup	0.001	1000	1018	0.008%
20-Newsgroup	0.005	1000	1490	0.26%
RCV1	0.8	1000	2211	0.69%
HIVA	6	2000	6707	0.18%
Claims	3	1100	1232	0.0036%
Claims	3.2	1100	1738	0.0068%

Table 5.4: Variation in the average number of sets and percentage of sets with more one positive example across multiple datasets and feasible similarity thresholds.

can use either explicit or implicit exploration. It is however desirable to come up with learning strategies that can be 'universally' used across different tasks/datasets and is an interesting future direction.

5.2.3 Set properties based on data characteristics

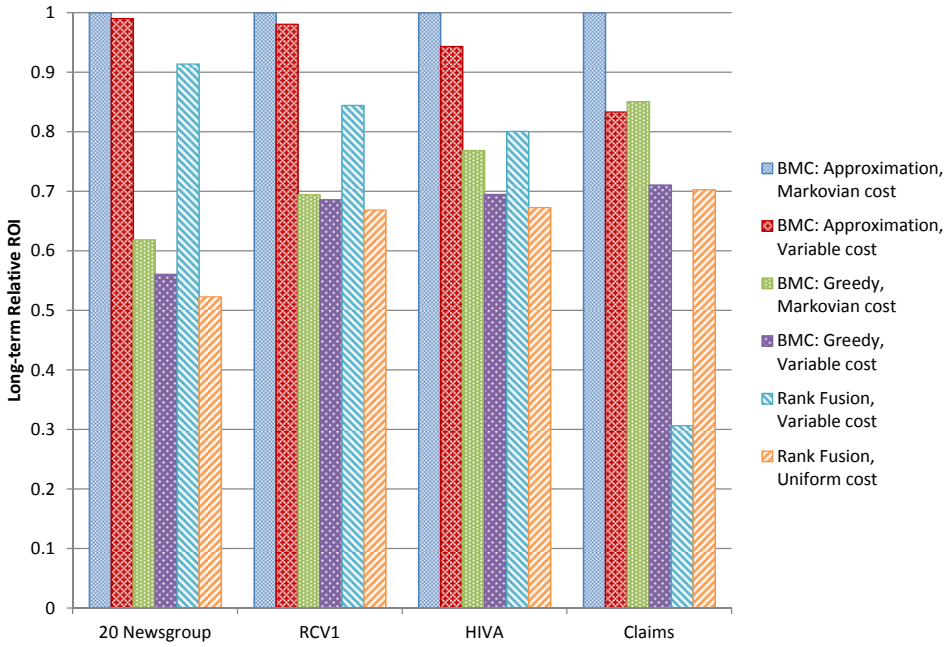
Table 5.3 shows the variation in the size of maximum clique for different similarity thresholds for different datasets. The distance metric that we have used is Euclidean and hence the scale of thresholds vary across datasets due to the size of the feature set. As described in section 3.2.6, the number of sets are $O(2^M)$ where M is the size of maximum clique and the runtime of Budgeted Max Coverage - Approximation algorithm is a polynomial function over the number of sets. Based on the run-time performance across the datasets we fix the maximum feasible clique size to be 18 (giving us the maximum number of sets in the order of $2^{18} = 262144$).

Based on the maximum clique size constraint of 18, we calculate the set memberships for the feasible similarity threshold which are summarized in table 5.4. We observe that the percentage of sets with more than one positive example was much less than respective class skew. This motivates the need to have similarity threshold for each class with higher threshold for the positive class in order to compensate for the class skewness. The advantage of markovian cost is particularly achieved when we audit positive examples together and obtain the cost reduction. However we also want the negative examples to be clustered in sets so that we can audit more examples in the same budget. Thus we need to empirically set the class-dependent thresholds in order to achieve these multiple objectives of having many sets with more than one positive example along with negative claims clustered as well, with the constraint of maximum clique size.

5.3 Results

Cost model	Optimization Algorithm	ROI (Variance)			
		20-Newsgroup	RCV1	HIVA	Claims
Markovian	BMC: Approximation	**1115.9**(±**5.3**)	**1250.5**(±**7.1**)	**734.8**(±**6.7**)	**631.6**(±**4.5**)
Variable	BMC: Approximation	1105.3(±5.4)	1226.9(±5.6)	693.1(±4.4)	526.6(±2.7)
Markovian	BMC: Greedy	690.6(±)	868.6(±15.1)	564.9(±11.7)	537.4(±9)
Variable	BMC: Greedy	625.8(±21.4)	857.9(±12)	510.9(±6)	449.1(±4.1)
Variable	Rank Fusion	1019.8(±5.7)	1056.4(±7.5)	588.5(±5.3)	193.6(±6.7)
Uniform	Rank Fusion	583.5(±22.7)	836.3(±13.8)	494.5(±4.1)	443.9(±5)

(a) Table showing the ROI for different datasets for different configurations. The maximum long-term ROI under all datasets is obtained by using Markovian cost with Budgeted Max Cover: Approximation optimization algorithm which is statistically better (highlighted in bold) than all other configurations at $p < 0.05$ after adjusting for multiple comparisons.



(b) Long-term relative ROI ceiling performance comparison across multiple datasets with oracle information about class labels and cost.

Figure 5.6: Performance with Oracle Knowledge

5.3.1 Instantiating framework with Oracle knowledge

Ceiling Performance

We initially establish the ceiling performance based on the framework assuming oracle knowledge about the class labels and cost. There are two motivations for evaluating the ceiling performance. First is to get the theoretically best possible performance so that other results can be compared against it using equation (4.2) for Long-term relative ROI. Second motivation is to evaluate the contributions of the framework given complete oracle information. Two novel aspects of the

framework that we want to evaluate are a) modeling markovian cost versus variable cost and b) using budgeted max coverage optimization framework with approximation algorithm compared to greedy approach as well as against rank fusion.

Given the details of the domains in section 5.1, we summarize the domain inputs for all the tasks/datasets as follows:

1. Evaluation Metric: Evaluation metric is the Return on Investment which is equivalent to the total number of positive examples labeled assuming use-it-or-loose-it budget.
2. Exploitation function: We are interested in the boolean classification problem for all the tasks, so the exploitation function is uniform exploitation utility for positive examples and no utility for negative examples.
3. Cost function: For all the domains, we are given markovian cost.

In order to instantiate the framework for evaluating ceiling performance we follow the following steps:

1. Determine Evaluation Metric: Based on domain input, we use Return on Investment as the metric. We present the results in terms of Long-term Relative ROI as described in equation (4.2) after determining the ceiling performance.
2. Choose Exploitation Model: Assuming that an oracle is giving us the class labels, it is trivial to assume the exploitation model as the oracle knowledge.
3. Choose Reviewing/Labeling Cost Model: There are two options while modeling the cost, we can experiment with modeling cost as variable or markovian. There are two parameters for markovian cost modeling, namely, similarity threshold per class and cost reduction factor when two examples are similar. We empirically determine the similarity threshold per class based on the practical constraint that the largest clique in any iteration should not be more than 18 (which leads to $O(2^18)$ sets). We set the cost reduction factor to 0.6 based on the real observations from Claims dataset [Kumar and Ghani, 2011].
4. Choose Exploration Model: Since we are assuming oracle knowledge of the class labels, we don't need to explore.
5. Choose Utility Metric: We experiment with both rank fusion utility as well as exploitation utility.
6. Choose Tradeoff optimization algorithm to optimize the utility metric: We use Rank Fusion algorithm as well as the Budgeted Max Coverage formulation with greedy as well as approximation algorithm approach.

Table 5.6a shows the detailed ROI for each configuration for the different tasks/datasets. Figure 5.6b shows the same data as presented in the table 5.6a but the results are represented as Relative ROI with comparison to the best performing strategy in order to contrast the level of performance. We observe that best performance is achieved by using Markovian cost with Budgeted Max Coverage - Approximation optimization algorithm. It is statistically better than all other configurations at $p < 0.05$ using the statistical comparison procedure described before in section 4.2.3.

Markovian vs Variable Cost

In order to analyze the performance difference between markovian and variable cost, we compare them while keeping the same optimization algorithm of Budgeted Max Coverage - Approximation algorithm, figure 5.6b. We observe that markovian cost gives statistically better ROI for all the

Data-set	Positive class sim. threshold	Negative class sim. threshold	Batch size	Largest clique size on avg.	Avg. number of sets per iteration	Percentage of sets with more than one pos example	Avg percentage of pos examples in sets with more than one positive example
20-Newsgroup	0.1	0.005	1000	13	9062	53.6%	60.8%
RCV1	0.9	0.8	1000	12	4961	27%	54.7%
HIVA	9	5	2000	10.6	3610	9.7%	66.5%
Claims	4.8	3	1100	11.2	3502	34.5%	75.9%

Table 5.5: Table showing the counts and characteristics of the sets obtained for markovian cost modeling for different datasets.

datasets. The magnitude of ROI increase from variable to markovian cost ranges from $\sim 1\%$ (20-Newsgroup) to $\sim 20\%$ (Claims). The degree of performance improvement that is obtained by using markovian cost modeling is dependent on the domain characteristics of how similar the examples are to each other. When the positive examples are more similar to each other, it gives greater advantage to audit them together and achieve the cost reduction. We can correlate these results to the data characteristics observed for the Claims data, figure 5.4a, where the positive examples are similar to each other and hence higher improvement $\sim 20\%$. Whereas for 20-Newsgroup data, figure 5.1a, the positive examples are not very similar to each other, thus the cost improvement achieved for markovian cost is only marginal $\sim 1\%$.

In order to further analyze the markovian cost results obtained and to characterize their behavior, we look at the detailed set statistics for the different datasets. The number of sets created is $O(2^M) + b$ where M is the largest clique size and b is the batch size of incoming examples in each active iteration. The largest clique size is in turn affected by the class-dependent similarity thresholds. We empirically determine these thresholds such that the worst case is computationally feasible for the dataset and we maximize the number of sets with more than one positive examples.[5] Thus given this practical constraint of maximum clique size, we want to have as large clique as possible (and correspondingly larger number of sets) because larger cliques would give more markovian cost savings. So the first statistic that we calculate is the average number of sets per active iteration for the dataset. For benefitting from markovian cost setup, we need to have cliques with more than one positive example which correspondingly means that we want sets with more than one positive examples. So the second measure is the percentage of sets that have more than one positive example. If a larger percentage of sets have more than one positive example, it is more likely that we can achieve better markovian cost reductions. Within the sets we want to have mostly positive examples rather than mixed sets with positive and negative examples so that we audit positive examples only. Thus the third measure we calculate is within the sets with more than one positive example what is the percentage of positive examples on average i.e. if a set has 4 positive and 6 negative claims this measure would be $40\%(4/(4+6)$. We want this measure to be as high as possible to achieve maximum markovian cost reduction benefit.

Table 5.5 summarizes the set statistics with the three measures for the different datasets. The distance metric that we have used is Euclidean and hence the scale of thresholds vary across

[5]In practice, determining the class-specific similarity thresholds should be motivated by the domain beyond the practical constraint. In other words, just because we are able to practically execute experiments with certain thresholds doesn't necessarily mean that they are meaningful for the domain.

datasets due to the size of the feature set.[6] The results presented in the table are micro-averaged over 10 test iterations and all active iterations. Before summarizing the data, we removed the extreme outliers based on the number of sets per iteration utilizing the non-parametric test based on Inter-Quartile Ranges [NIST, 2013]. Extreme outlier is defined Q3 + 5*IQ and Q1 - 5*IQ, where Q1 is first quartile (25th percentile), Q3 is third quartile(75th percentile) and IQ is Q3-Q1.

For example, on average we get 3502 sets for Claims dataset. Given that number of sets are $O(2^M) + b$, we get the largest clique size on average as $log_2(3502 - 1100) \sim 11.2$. We measure the other statistics of percentage of sets with more than one positive example (34.5% for Claims) and average percentage of positive claims in sets with more than one positive example (75.9% for Claims). We observe from the table that the Claims dataset has the maximum average percentage of positive claims in sets with more than one positive claims of 75.9% thus giving us the maximum improvement of 20% followed by HIVA which has 66.5% positive claims giving us an improvement of 6%.

Comparing Optimization Algorithms

We compare the performance of the different optimization algorithms in figure 5.6b by analyzing the configurations with variable cost function (varying optimization algorithm: BMC:Approximation, BMC:Greedy and Rank Fusion). By utilizing the same statistical test adjusted for multiple comparison as shown in figure 5.6b we find that BMC:Approximation algorithm is statistically better than BMC:Greedy as well as baseline Rank Fusion method for all four tasks/datasets.

The performance patterns that are observed for the optimization algorithm are similar across all datasets except one surprise observation which is the poor performance of rank fusion algorithm for the Claims data while using variable cost and true class labels. On further analysis we discover that the behavior is not an anomaly but is rather related to the mechanism of the rank fusion algorithm. Even though the rankings based on true class label assign higher and same rank to the positive examples they are fused with rankings based on the cost with equal weightage to both rankings. This results in the observations being dependent on data characteristics and correlation of the class label with the cost characteristics of each domain/task. It leads to the observation that the positive examples in the Claims dataset are not the cheapest examples to label whereas in other domains the positive examples are relatively cheaper hence fusing the class-label based rankings and cost based rankings give good overall ranking for other domains.

Conclusion

From the results above we can conclude that in the presence of complete oracle information, the framework with its novel contributions of ability to model markovian cost and using budgeted max coverage - approximation optimization algorithm performs better than the other alternates of modeling variable or uniform cost and baseline optimization approach of rank fusion. This shows the usefulness and value of the framework albeit with oracle knowledge over baseline approaches. The ceiling performance achieved for different tasks is: 20-Newsgroup - 1115.9, RCV1 - 1250.5, HIVA - 734.8 and Claims - 631.6 for a budget of 50,50,35 and 22 queries per batch respectively. Next we show the usefulness of the framework when we relax the oracle knowledge assumption for the class labels.

[6]We used cosine metric to analyze the data but since cosine metric is not a proper distance metric we chose to use Euclidean metric for set determination.

Dataset	All Experiments	Only BMC:Approximation	Only BMC:Greedy	Only Rank Fusion
20-Newsgroup	0.44^{***}	0.82^{***}	0.74^{***}	0.17
RCV1	0.65^{***}	0.72^{***}	0.75^{***}	0.58^{***}
HIVA	0.47^{***}	0.53^{*}	0.78^{***}	0.31
Claims	0.51^{***}	0.68^{**}	0.61^{**}	0.35^{*}

Table 5.6: Spearman rank correlation coefficients comparing the Long Term Relative ROI with the number of factors used for the different datasets from figure 5.7(a)(c)(e)(g). Higher the correlation coefficient, better performance with more number of factors. '*'s indicate the statistical significance of the correlation. No star means that the correlation is not significant, '*'- significant with $p < 0.05$, '**'- significant with $p < 0.01$, '***'- significant with $p << 0.001$

Dataset	All Experiments	Only BMC:Approximation	Only BMC:Greedy	Only Rank Fusion
20-Newsgroup	0.51^{***}	0.80^{***}	0.35	0.26
RCV1	0.19	0.33	0.32	0.11
HIVA	0.18	0.12	0.54^{*}	-0.14
Claims	0.04	0.16	0.09	-0.21

Table 5.7: Spearman rank correlation coefficients comparing the Long Term Relative ROI with different cost model from figure 5.7(b)(d)(f)(h). Higher the correlation coefficient, better performance with the cost models in the following order: uniform, variable, markovian. '*'s indicate the statistical significance of the correlation. No star means that the correlation is not significant, '*'- significant with $p < 0.05$, '**'- significant with $p < 0.01$, '***'- significant with $p << 0.001$

5.3.2 Does the framework help in improving performance over baseline approaches?

This is the primary question that we want to address where the hypothesis is that jointly optimizing 3 factors is better than optimizing two factors or just one factor. There are five baseline approaches as described in section 4.2.4, which are Random (0 Factors), Classification (1 factor - Exploitation), Active Learning (1 factor - Exploration), Cost Minimization (1 factor - Cost) and Cost-sensitive Exploration (2 factors - Cost and Exploration). We perform the analysis in two stages. In order to analyze the general trend, we initially compare the performance of all configurations while treating the number of factors used as the pseudo-independent variable. After analyzing the general trends, we statistically compare the top performing configurations with a number of baseline approaches to show the value of the proposed framework. Before discussing the results, we start by describing the steps for instantiating the framework as well as the different modeling variations that we experimented with.

In order to instantiate the framework, we need to determine the domain inputs and then make modeling choices in the framework. The domain inputs for cost, exploitation and evaluation metric are the same as defined in the previous section 5.3.1. The modeling steps are as follows:

1. Determine Evaluation Metric: Based on domain input, we use Return on Investment as the metric. We present the results in terms of Long-term Relative ROI as described in equation (4.2).
2. Choose Exploitation Model: For modeling the uniform (boolean) exploitation, we use the classifier's probability output to get the expected exploitation value for an example.
3. Choose Reviewing/Labeling Cost Model: We assume that the true cost for labeling an example is known to us through a domain cost model. There are three variations that we

experiment with, namely uniform, variable and markovian cost. There are two parameters for markovian cost modeling, namely, similarity threshold per class and cost reduction factor when two examples are similar. As outlined in the previous section 5.3.1, we empirically determine the similarity threshold per class based on the practical constraint that the largest clique in any iteration should not be more than 18 (which leads to $O(2^18)$ sets). We empirically set the cost reduction factor to 0.6 based on the real observations from Claims dataset [Kumar and Ghani, 2011].

4. Choose Exploration Model: We experiment with all the variations of exploration model as described in section 4.2.2, namely, uniform (random), variable:density, variable:sparsity, markov:uncertainty, markov:certainty, markov:classifier, markov:error reduction-supervised, markov:error reduction-unsupervised, markov:error reduction-oracle.
5. Choose Utility Metric: We experiment with both rank fusion utility as well as combination of exploration and exploitation utility. It is important to note that rank fusion utility is not defined for markovian cost case as the ranking is not deterministic based on the markovian cost.
6. Choose Tradeoff optimization algorithm to optimize the utility metric: We use Rank Fusion algorithm as well as the Budgeted Max Coverage formulation with greedy and approximation algorithm approach.

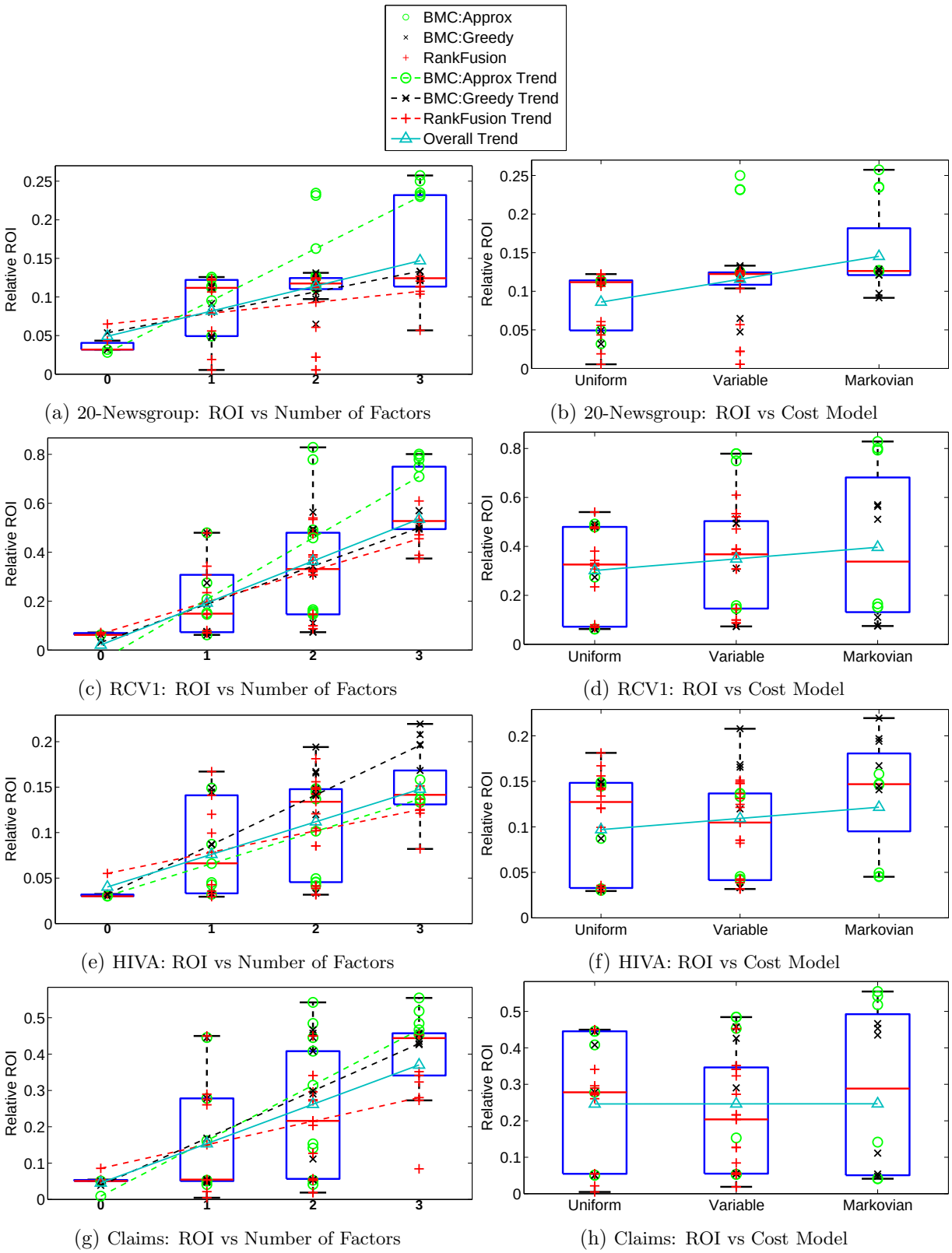
General Trend

Figure 5.7(a)(c)(e)(g) shows the result for the four tasks/datasets for all the variations with the number of factors treated as the pseudo-independent variable. The results are presented as long-term relative ROI. We observe that in general increasing the number of factors increases the long-term relative ROI across all datasets.

Scale of performance across different datasets: We observe the level of performance for the different datasets from figures 5.7(a)(c)(e)(g). We note that for RCV1 dataset, the scale of performance is close to 0.8 of ceiling performance but for HIVA and 20-Newsgroup datasets it is around 0.25. For Claims dataset it is close to 0.5. This indicates the degree of complexity of the different datasets that RCV1 dataset is easiest to model and we are able to achieve good performance close to the ceiling performance. For other datasets, Claims task is relatively easier to model than 20-Newsgroup dataset whereas HIVA dataset is hardest to model.

Rank correlation between performance and number of factors used for optimization: Since the results are across multiple configurations (number of factors is not an independent variable), it is not appropriate to do statistical tests to determine significance based on analysis of variance across the number of factors. However, we can analyze the pattern by calculating the Spearman's rank correlation coefficient between the number of factors and the performance of different configurations. The intuition for doing the Spearman rank correlation is to observe patterns that would indicate that using more factors leads to higher performance i.e. higher the correlation coefficient using larger number of factors performs better.

Table 5.6 shows the Spearman rank correlation coefficients for the four datasets along with their statistical significance. We first compare all the aggregated experiments and then factor the optimization algorithm to compare the correlations. For the aggregated experiments, we find that the correlations are statistically significant for all datasets with varying degree of positive correlation coefficient from 0.44 for 20-Newsgroup to 0.65 for RCV1. For algorithm-wise correlations, we observe that for both BMC:Approximation and BMC:Greedy optimization algorithm



(a) 20-Newsgroup: ROI vs Number of Factors

(b) 20-Newsgroup: ROI vs Cost Model

(c) RCV1: ROI vs Number of Factors

(d) RCV1: ROI vs Cost Model

(e) HIVA: ROI vs Number of Factors

(f) HIVA: ROI vs Cost Model

(g) Claims: ROI vs Number of Factors

(h) Claims: ROI vs Cost Model

Figure 5.7: Comprehensive results across all variations for different datasets. Figures (a)(c)(e)(g) show the variations across different number of factors (cost, explore and exploit) while figures (b)(d)(f)(h) show the same results but comparing across different cost variations of Uniform, Variable and Markovian.

the rank correlation is much higher than the overall comparison for all datasets. For example for the 20-Newsgroup dataset, the rank correlation increases from 0.44 for the overall comparison to 0.82 for BMC:Approximation algorithm. This indicates that the framework with it's Budgeted Max Cover formulation helps more as we have additional factors to trade-off and optimize.

Rank correlation between performance and cost model: We also compare the performance across different cost models in figure 5.7(b)(d)(f)(h). Table 5.7 shows the Spearman rank correlation coefficient between the Long-term Relative ROI and the different cost models: uniform, variable and markovian. We observe that the overall rank correlations are much weaker for RCV1, HIVA and Claims dataset. However for 20-Newsgroup dataset we get statistically significant positive correlations. This indicates that the markovian cost model usually performs better than other cost variations for 20-Newsgroup dataset but for other tasks it is not significantly correlated. Thus further analysis needs to be done to determine which configurations are performing better for the various cost models and how do they compare against each other.

Does it matter to model the Markovian Cost?

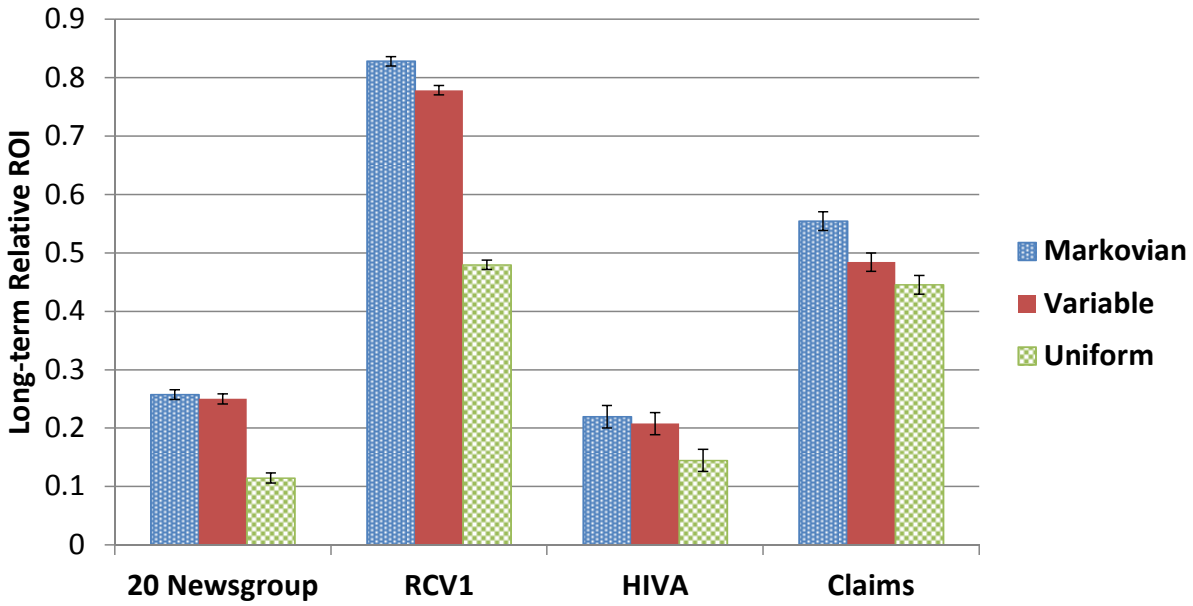


Figure 5.8: Long-term relative ROI ceiling performance comparison across multiple datasets with Markovian, Variable and Uniform cost models. Error bars represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison. If two strategies do not overlap then they are significantly different with $p < 0.05$.

Analyzing the top performing strategy for all the tasks/datasets from 5.7(b)(d)(f)(h), we observe that modeling markovian cost helps consistently across all tasks/datasets. So we can conclude that modeling markovian cost helps consistently across all datasets.

We perform a detailed analysis to compare the degree of improvement of markovian cost modeling over other modeling choices of variable and uniform cost by keeping all other factors constant. We use the best performing configuration for each dataset as the representative configuration to make the comparisons. Figure 5.8 shows the result for the comparison. We observe a similar pattern as observed while comparing the strategies with Oracle knowledge in section 5.3.1 that the degree of improvement based on markovian modeling varies for each dataset. The

improvement over uniform cost modeling, which is the classic assumption made in active learning literature, is statistically significant after adjusting for multiple comparisons (4.2.3) at $p < 0.05$ for all datasets. The relative improvements are 125%, 72.7%, 51.9% and 24.5% for 20-Newsgroup, RCV1, HIVA and Claims datasets respectively. . For comparison with variable cost modeling, we get the following relative improvements: 3%, 6.4%, 5.6% and 14.5% for 20-Newsgroup, RCV1, HIVA and Claims datasets respectively which are statistically significant after adjusting for multiple comparisons for RCV1 and Claims dataset only.

Does the optimization algorithm matter?

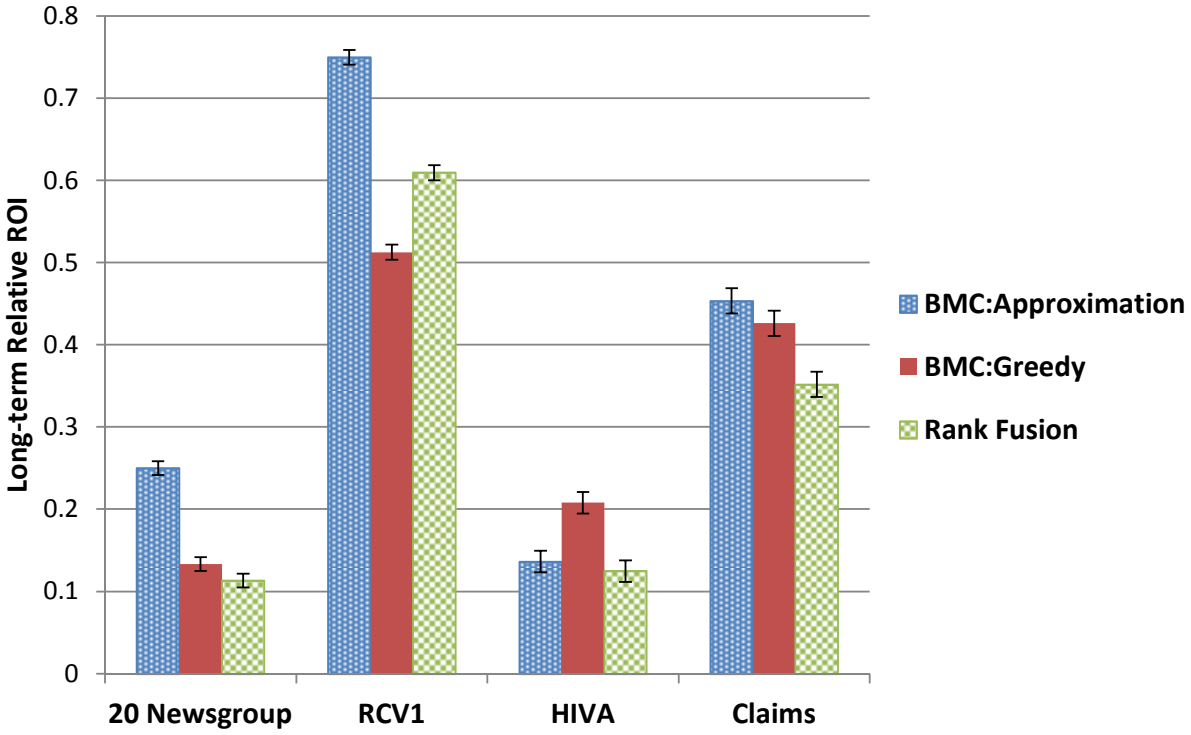


Figure 5.9: Long-term relative ROI performance comparison across multiple datasets with varying optimization algorithm.

We observe from figure 5.7, that the best performing configuration uses the Budgeted Max Coverage formulation for all tasks/datasets. For three datasets, 20-Newsgroup, RCV1 and Claims, the approximation algorithm for Budgeted Max Cover is the best performing algorithm whereas for HIVA greedy algorithm for Budgeted Max Cover gives the best performance. Since we observed that approximation algorithm works best for HIVA dataset with complete Oracle knowledge, figure 5.6b, we have done further analysis. We found evidence that choice of greedy algorithm is better because of two reasons of poor classifier probability estimates and high rank correlation between cost and exploitation. We discuss this conjecture in more detail in section 5.4.1. .

In order to perform further detailed comparison amongst the optimization algorithms, we select the following configurations for each dataset and vary the optimization algorithm. The configuration is cost model: variable (as the baseline optimization algorithm is not applicable for markovian cost model), exploit model: classifier and explore model: markov:error reduction-

unsupervised. We chose this configuration as this is one of the best performing configurations across datasets. Figure 5.9 shows the result of the comparison along with the $p < 0.05$ statistical confidence interval adjusted for multiple comparison. We observe that BMC:Approximation is statistically best strategy for two datasets, 20 Newsgroup and RCV1. BMC:Greedy is statistically best for HIVA dataset. For the Claims dataset, BMC:Approximation and BMC:Greedy are statistically similar to each other..

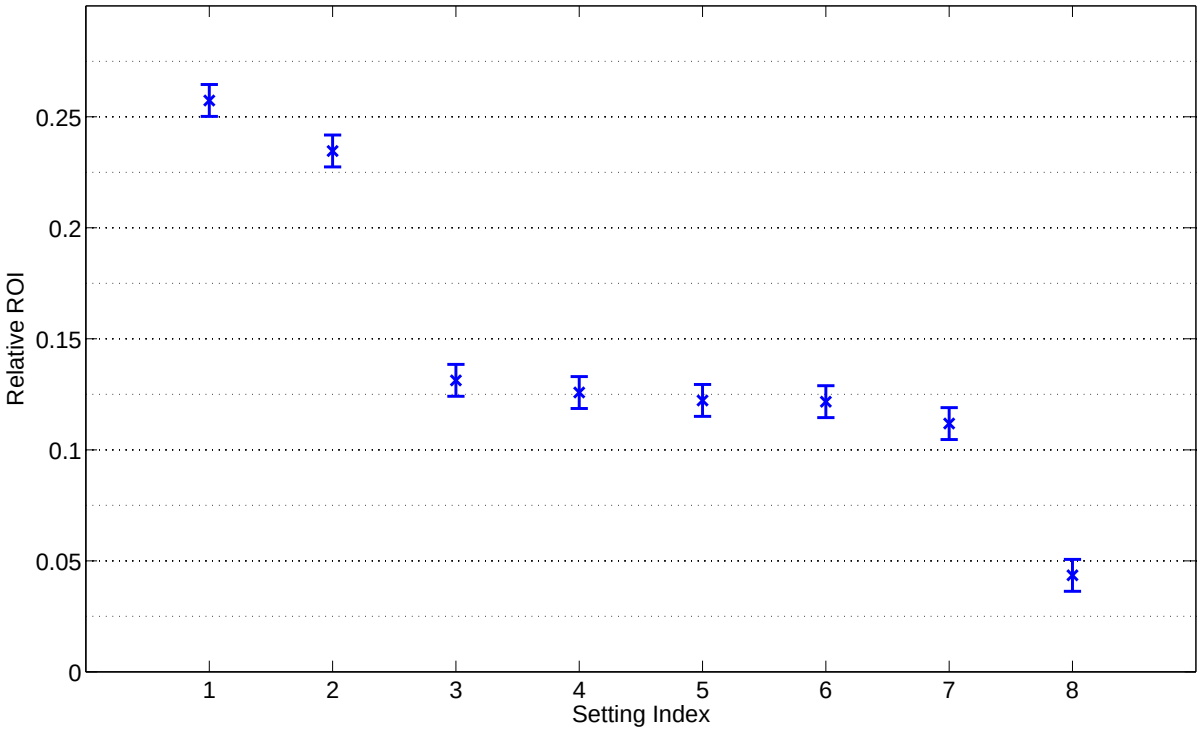
Statistical Comparison

For the second stage of analysis, we statistically compare the top performing configurations with a number of baseline approaches to show the value of the proposed framework. The methodology followed for this comparison is as follows: There are $8(2^3)$ possible setups with different factors activated amongst the three factors of cost, exploration and exploitation. The setup with no factor activated is the random baseline. The setup with only one factor activated for exploit, explore and cost correspond to the following baselines: classification, active learning and minimizing cost respectively. The setup with cost and exploration factor activated corresponds to the cost-sensitive active learning baseline. These baselines were described in detail earlier in section 4.2.4. Each of these 8 setups can have multiple variations amongst them with different choice of cost model, exploration model, exploitation model, utility metric and optimization algorithm. For each of these 8 setups we pick the best performing configuration based on mean performance across test iterations[7]. We then do statistical comparison amongst these best performing configurations across the 8 setups over 10 random trials using the multiple comparison strategy described before in section 4.2.3.

Figures 5.10a, 5.11a, 5.12a and 5.13a show the results for the multiple statistical comparisons of the best performing strategy under each setup. In the figures, error bars represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison. If two strategies do not overlap then they are significantly different with $p < 0.05$. Tables 5.10b, 5.11b, 5.12b and 5.13b show the details of the best performing configuration across the different categories along with their long-term relative ROI performance with variance. In addition to comparing the long-term relative ROI we also statistically compare the relative ROI at each time step across the different configurations and report the Win/Loss/Ties across all the iterations with reference to the best performing strategy. For example, in table 5.10b, comparing the best performing strategy (index 1) with strategy index 2, we observe that it statistically won 12 times and was tied 4 times. We analyze and discuss the results for each task in the following subsections.

Information Filtering: 20-Newsgroup-We observe that the setup optimizing the three factors of Cost, Exploration and Exploitation (index 1 in figure 5.10a) significantly outperforms all other setups including the baseline approaches. The performance is 96% better than the best performing baseline of Cost-sensitive Exploration (index 3 in figure 5.10a). The setup that performs best has the following configuration: cost model: markovian, exploit model: classifier, explore model: markov: error reduction-unsupervised, utility metric: combined exploration and exploitation utility, optimization: BMC - approximation algorithm. We also observe based on the Win/Loss/Tie statistics that the best performing configuration significantly outperforms the other strategies in each active iteration as well.

[7]We empirically observed the variance within each group and since they were similar we chose to select the best performing strategy based on average performance only and didn't take into account the variance. If the variance was not similar then a strategy with lower average performance but smaller variance may be a better choice than the higher average strategy with higher variance

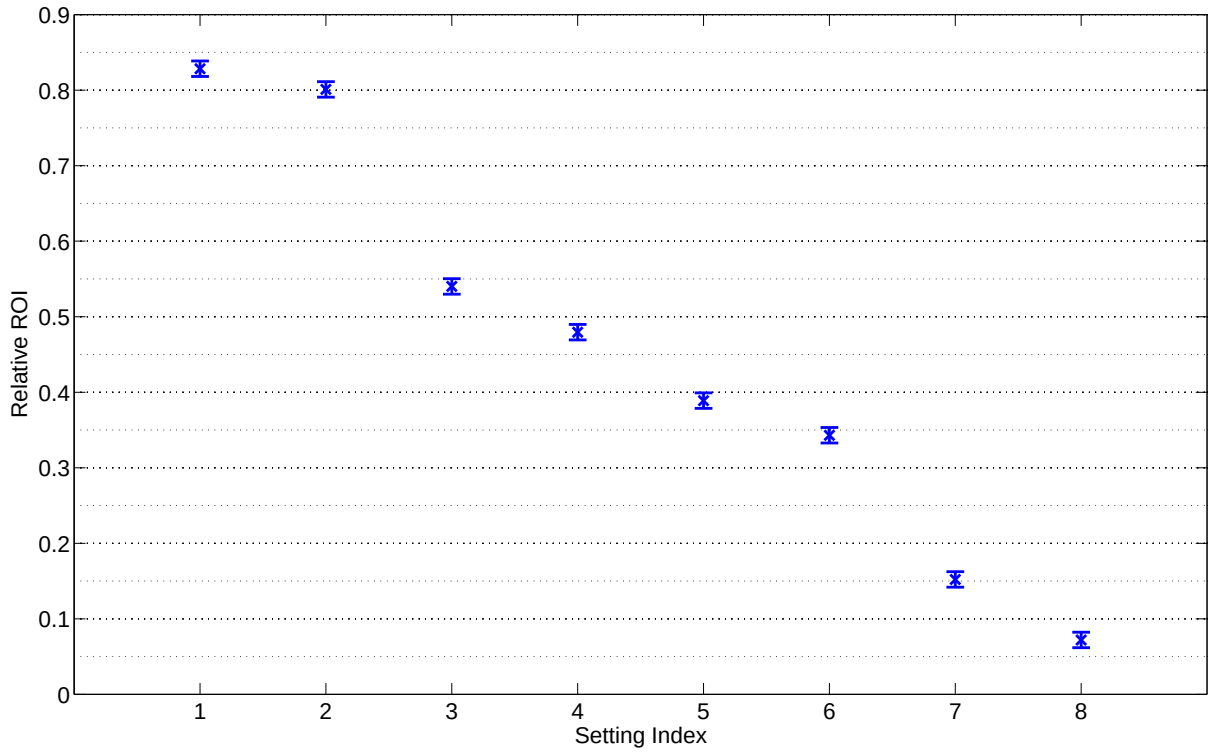


(a) Error bars in the graph represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison. If two strategies do not overlap then they are significantly different with $p < 0.05$.

Setting Index	Number of Factors	Cost model	Explore model	Exploit model	Optimization Algorithm	Long-term Relative ROI(Variance)	Win/ Loss/ Tie
1	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	25.7%(±1.3%)	-
2	2	Markovian	X	Classifier	BMC: Approximation	23.5%(±1.6%)	12/0/4
3	2	Variable	Markov:Error Reduction-Unsupervised	X	BMC: Greedy	13.1%(±1.4%)	16/0/0
4	1	Markovian	X	X	BMC: Approximation	12.6%(±0.5%)	14/0/2
5	1	X	Markov:Uncertainty	X	Rank Fusion	12.2%(±2.1%)	16/0/0
6	2	X	Variable:Density	Classifier	Rank Fusion	12.2%(±2.1%)	16/0/0
7	1	X	X	Classifier	Rank Fusion	11.2%(±3.3%)	16/0/0
8	0	X	X	X	Rank Fusion	4.3%(±0.4%)	16/0/0

(b) Table describing the configurations for the results in (a). It also shows the Win/Loss/Tied statistics for comparing the best performing strategy with other configurations over all iterations. We use similar statistical comparison adjusted for multiple comparisons.

Figure 5.10: 20 newsgroup - Statistical comparison of best performing configurations for each combination of factors

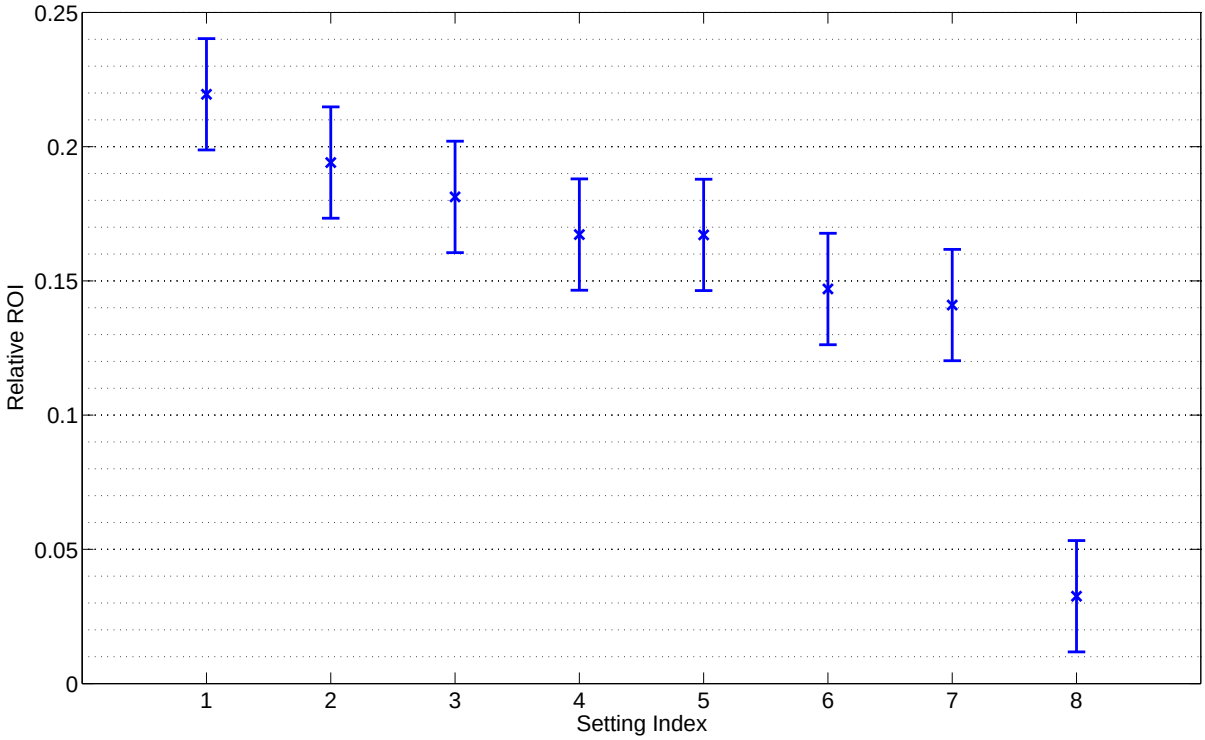


(a) Error bars in the graph represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison. If two strategies do not overlap then they are significantly different with $p < 0.05$.

Setting Index	Number of Factors	Cost model	Explore model	Exploit model	Optimization Algorithm	Long-term Relative ROI(Variance)	Win/ Loss/ Tie
1	2	Markovian	X	Classifier	BMC: Approximation	82.8%(±1.9%)	-
2	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	80.1%(±2.7%)	17/0/0
3	2	X	Markov:Error Reduction-Unsupervised	Classifier	Rank Fusion	54.0%(±2.5%)	17/0/0
4	1	X	X	Classifier	BMC: Greedy	47.9%(±3.5%)	16/0/1
5	2	Variable	Markov:Error Reduction-Unsupervised	X	Rank Fusion	38.9%(±5.9%)	17/0/0
6	1	X	Markov:Uncertainty	X	Rank Fusion	34.3%(±0.8%)	16/0/1
7	1	Markovian	X	X	BMC: Approximation	15.2%(±0.5%)	17/0/0
8	0	X	X	X	Rank Fusion	7.2%(±0.7%)	17/0/0

(b) Table describing the configurations for the results in (a). It also shows the Win/Loss/Tied statistics for comparing the best performing strategy with other configurations over all iterations. We use similar statistical comparison adjusted for multiple comparison with Fisher's least significant difference test.

Figure 5.11: RCV1 - Statistical comparison of best performing configurations for each combination of factors

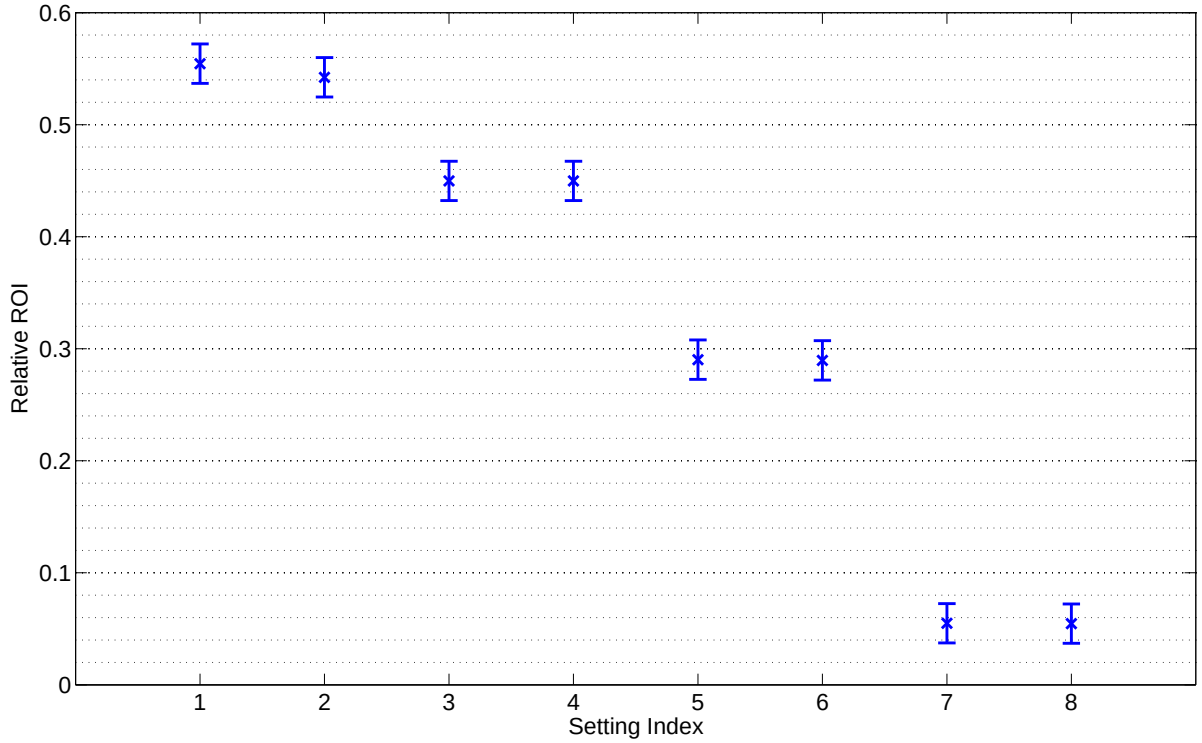


(a) Error bars in the graph represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison with Fisher's least significant difference test. If two strategies do not overlap then they are significantly different with $p < 0.05$.

Setting Index	Number of Factors	Cost model	Explore model	Exploit model	Optimization Algorithm	Long-term Relative ROI(Variance)	Win/ Loss/ Tie
1	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Greedy	22.0%(±6.9%)	-
2	2	Markovian	X	Classifier	BMC: Greedy	19.4%(±7.8%)	0/0/18
3	2	X	Markov:Error Reduction-Supervised	Classifier	Rank Fusion	18.1%(±6.7%)	0/0/18
4	2	Markovian	Markov:Error Reduction-Unsupervised	X	BMC: Greedy	16.7%(±3.0%)	8/0/10
5	1	X	X	Classifier	Rank Fusion	16.7%(±7.9%)	14/0/4
6	1	Markovian	X	X	BMC: Greedy	14.7%(±3.0%)	12/0/6
7	1	X	Markov:Certainty	X	Rank Fusion	14.1%(±2.7%)	15/0/3
8	0	X	X	X	Rank Fusion	3.3%(±0.5%)	18/0/0

(b) Table describing the configurations for the results in (a). It also shows the Win/Loss/Tied statistics for comparing the best performing strategy with other configurations over all iterations. We use similar statistical comparison adjusted for multiple comparisons.

Figure 5.12: HIVA - Statistical comparison of best performing configurations for each combination of factors



(a) Error bars in the graph represent the 95% confidence interval (not variance) for a strategy after adjusting for multiple comparison. If two strategies do not overlap then they are significantly different with $p < 0.05$.

Setting Index	Number of Factors	Cost model	Explore model	Exploit model	Optimization Algorithm	Long-term Relative ROI(Variance)	Win/ Loss/ Tie
1	3	Markovian	Markov:Error Reduction-Supervised	Classifier	BMC: Approximation	55.4%(±8.9%)	-
2	2	Markovian	X	Classifier	BMC: Approximation	54.2%(±8.5%)	0/0/21
3	2	X	Markov:Error Reduction-Supervised	Classifier	Rank Fusion	45.0%(±6.3%)	17/0/4
4	1	X	X	Classifier	Rank Fusion	45.0%(±6.3%)	17/0/4
5	2	Variable	Markov:Error Reduction-Unsupervised	X	BMC: Greedy	29.0%(±5.1%)	21/0/0
6	1	X	Markov:Error Reduction-Unsupervised	X	Rank Fusion	29.0%(±4.3%)	21/0/0
7	1	Variable	X	X	BMC: Greedy	5.5%(±0.8%)	21/0/0
8	0	X	X	X	Rank Fusion	5.5%(±0.8%)	21/0/0

(b) Table describing the configurations for the results in (a). It also shows the Win/Loss/Tied statistics for comparing the best performing strategy with other configurations over all iterations. We use similar statistical comparison adjusted for multiple comparisons.

Figure 5.13: Claims - Statistical comparison of best performing configurations for each combination of factors

The observed results support the hypothesis that for the datasets where explicit exploration is useful and the domain is not very easy to model the framework optimizing the three factors of cost, exploration and exploitation gives significantly better performance than other alternates.

Information Filtering: RCV1-We observe that the setup optimizing two factors of Cost and Exploitation (index 1 in figure 5.11a) significantly outperforms all other setups including the baseline approaches. The performance is 73% better than the best performing baseline of classifier based Exploitation (index 4 in figure 5.11a). The setup that performs best has the following configuration: cost model: markovian, exploit model: classifier, explore model: none, utility metric: combined exploration and exploitation utility, optimization: BMC - approximation algorithm. We also observe based on the Win/Loss/Tie statistics that the best performing configuration significantly outperforms the other strategies in each active iteration as well.

The observed results illustrate that for domains where the concept is easy to model and explicit exploration is not needed, we should not divert any resources for exploration. Instead we should simply use the classifier intelligently taking into account the cost for each example. However based on the framework, if we do divert some resources for exploration (index 2 in figure 5.11a) we still achieve better performance than the best baseline of using the classifier based Exploitation only(index 4 in figure 5.11a).

Chemo-Informatics: HIVA-We observe that the setup optimizing the three factors of Cost, Exploration and Exploitation (index 1 in figure 5.12a) significantly outperforms all the baseline approaches and is statistically comparable to other configurations. The performance is 32% better than the best performing baseline of Cost-sensitive Exploration (index 4 in figure 5.10a). The setup that performs best has the following configuration: cost model: markovian, exploit model: classifier, explore model: markov: error reduction-unsupervised, utility metric: combined exploration and exploitation utility, optimization: BMC - greedy algorithm. We also observe based on the Win/Loss/Tie statistics in table 5.12b that the best performing configuration either significantly outperforms or is equivalent to the baseline strategies in each active iteration.

The target concept in HIVA dataset is fairly hard to model as interpreted based on the initial data analysis (figure 5.3b) and learning curves (figure 5.5(e)(f)). The observed results support the hypothesis that for the datasets where explicit exploration is useful and the domain is not very easy to model the framework optimizing the three factors of cost, exploration and exploitation gives significantly better performance than the baseline approaches. Also it is interesting to note that even though the performance of the best performing strategy using optimizing the three factors is statistically similar to the strategy optimizing two factors of cost and exploitation the performance difference is 13%. For a domain like chemo-informatics where the reward for finding a potentially active compound against HIV aids virus is very high, this relative improvement is practically very important even though statistically it may not be significant. Thus it helps to explicitly explore for such domains.

Error Prediction: Claims-We observe that the setup optimizing the three factors of Cost, Exploration and Exploitation (index 1 in figure 5.13a) significantly outperforms all the baseline approaches. The performance is 23% better than the best performing baseline of classifier based Exploitation (index 4 in figure 5.13a). The setup that performs best has the following configuration: cost model: markovian, exploit model: classifier, explore model: markov: error reduction-supervised, utility metric: combined exploration and exploitation utility, optimization: BMC - approximation algorithm. We also observe based on the Win/Loss/Tie statistics that the best performing configuration significantly outperforms or is equivalent to the baseline strategies in each active iteration as well.

For the claims dataset, we had observed that it was a relatively easier concept to model as the negative class had high internal similarity, figure 5.4b. The classifier based active learning strategy was one of the best performing strategies from figure 5.5(g)(h). Thus the observed result that using the framework without explicit exploration performs really well.

5.3.3 Summary of Results

We summarize all the results observed for the various experiments and analysis as follows:

1. We find that the proposed framework, with its Budgeted Max Coverage optimization formulation and Markovian cost modeling gives significantly better results than all the baseline approaches for all tasks. This is the most important observation that supports the novel concepts introduced in the thesis of markovian cost modeling and formulating the problem as an optimization problem.
2. In domains where explicit exploration is helpful, we achieve significantly better results combining the 3 factors of cost, exploration and exploitation using the framework. Whereas in domains where implicit exploration (exploitation) is a reasonable learning strategy a combination of just 2 factors of cost and exploitation works well as there is no trade-off between exploration and exploitation. This is a very important practical result that highlights the importance of understanding the domain characteristics and their implication on the design choices of diverting resources for exploration or not.
3. Modeling markovian cost helps consistently across all tasks. The degree of improvement over variable cost can vary with the data characteristics as well as the parameter selection of class specific similarity thresholds. So empirical tuning is required to maximize gains based on markovian cost.
4. BMC:Approximation optimization algorithm works well for three datasets of 20-Newsgroup, RCV1 and Claims whereas BMC:Greedy works better for HIVA dataset. However, with complete Oracle knowledge BMC:Approximation works best for all the datasets.

5.4 Discussion and Future Work

5.4.1 Is there a universally 'safe' configuration?

An obvious question to ask is if there is a universally well performing strategy across all datasets and thus is 'safe' to deploy in practice for any new task as the starting strategy which may be refined as we collect and analyze more data about the domain/task. We find that the strategy with Cost: Markovian model, Explore: Markovian - Estimated future error reduction strategy using Unsupervised estimation is the amongst the best performing strategy for all the datasets and hence is recommended as a universally 'safe' strategy. However, the choice of optimization algorithm is domain dependent as BMC:Approximation algorithm works best for Claims, RCV1 and 20 Newsgroup dataset whereas BMC:Greedy works well for the HIVA dataset.

In order to answer this question we compared the three top performing strategies across all datasets/tasks to analyze if there exists a configuration that performs well across datasets. These well performing strategies may be candidates for universally 'safe' strategies. We can then analyze their statistical performance and determine if they are indeed 'safe' strategies to deploy in practice. Figure 5.14 shows the three top performing strategies across all the datasets. We do find two such strategies that are in the top three performing strategy for all tasks/datasets. The strategies

Dataset	Spearman Rank Correlation Coefficient		
	Exploit with Explore	Exploit with Cost	Explore with Cost
20-Newsgroup	0.31	0.18	0.03
RCV1	0.25	-0.13	0.11
HIVA	0.95	0.52	0.50
Claims	-0.24	-0.19	0.09

Table 5.8: Pair-wise Spearman rank correlation between ranking produced by Cost, Exploit and Explore model for the universally 'safe' performing strategy averaged over all active iterations and 10 random trials.

have the following configuration a) Strategy 1 - Cost: Markovian model, Explore: Markovian - Estimated future error reduction strategy using Unsupervised estimation and Exploit: Classifier based estimated likelihood of positive class and b) Strategy 2 - Cost: Markovian model, Explore: Markovian - Estimated future error reduction strategy using Supervised estimation and Exploit: Classifier based estimated likelihood of positive class. The common strategies are highlighted with same background color in the table. The difference between the two strategies is the choice of estimation methodology (supervised or unsupervised) for the future error reduction exploration strategy.

We analyze the statistical performance of the two strategies to make some general inferences. For the first strategy, we observe that for 20-Newsgroup and HIVA datasets, this strategy is the best performing, highlighted in green in figure 5.14. However, it is statistically tied to the next best performing strategy for both the datasets. For RCV1 dataset, the strategy is statistically worse than the best performing strategy while for the Claims dataset it is statistically tied to the best performing strategy. The statistical comparisons indicate that besides RCV1 dataset, this strategy is statistically equivalent or better than other strategies and thus is statistically 'safe' to deploy for these domains. For RCV1 domain, even though it is statistically worse than the best strategy, the absolute performance level of 80.1% for long-term relative ROI with the relative difference of 3.4% with the best strategy makes it a reasonable practical choice for an initial deployment for the Information Filtering task. Hence we recommend this strategy as a universally 'safe' strategy.

For the second strategy, we note that it is the best performing strategy for the Claims dataset and is the third performing for all other datasets, highlighted in red in figure 5.14. In the Claims dataset, it is statistically tied to all other strategies because the variance in performance for the Claims dataset is really high. For HIVA dataset, it is statistically tied to the best performing strategy as the variance is high for the HIVA dataset as well. For 20-Newsgroup and RCV1 dataset, it is statistically worse than the best performing strategies for both the datasets. Given that the strategy is statistically worse for more than one dataset with varying degree of inferior performance, we do not consider it a universally 'safe' strategy.

The choice of optimization algorithm for the recommended 'safe' strategy is BMC: Approximation algorithm for three datasets of 20-Newsgroup, RCV1 and Claims and BMC: Greedy algorithm for HIVA dataset. We also note interestingly that all the top performing strategies for HIVA datasets utilize BMC:greedy optimization algorithm whereas for all other datasets BMC: approximation optimization algorithm is top performing, figure 5.14. Given this observation of different optimization algorithm for different datasets, we do not recommend a universally 'safe' optimization algorithm and instead recommend that it should be empirically determined

	Rank	Number of Factors	Cost model	Explore model	Exploit model	Optimization Algorithm	Long-term Relative ROI(Variance)	Win/ Loss/ Tie
20 Newsgroup	1	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	25.7%(±1.3%)	-
	2	3	Variable	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	25.0%(±1.3%)	0/0/16
	3	3	Markovian	Markov:Error Reduction-Supervised	Classifier	BMC: Approximation	23.5%(±1.5%)	15/0/1
RCV1	1	2	Markovian	X	Classifier	BMC: Approximation	82.8%(±1.9%)	-
	2	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	80.1%(±2.7%)	17/0/0
	3	3	Markovian	Markov:Error Reduction-Supervised	Classifier	BMC: Approximation	79.3%(±1.5%)	11/0/6
HIVA	1	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Greedy	22.0%(±6.9%)	-
	2	3	Variable	Markov:Error Reduction-Unsupervised	Classifier	BMC: Greedy	20.8%(±4.6%)	0/0/18
	3	3	Markovian	Markov:Error Reduction-Supervised	Classifier	BMC: Greedy	19.7%(±8.0%)	0/0/18
Claims	1	3	Markovian	Markov:Error Reduction-Supervised	Classifier	BMC: Approximation	55.4%(±8.9%)	-
	2	2	Markovian	X	Classifier	BMC: Approximation	54.2%(±8.5%)	0/0/21
	3	3	Markovian	Markov:Error Reduction-Unsupervised	Classifier	BMC: Approximation	51.8%(±10.1%)	3/0/18

Figure 5.14: Top three performing strategies wrt Long-term Relative ROI for the different datasets. The strategy with same background color are the common configurations across all datasets/tasks. We compare the strategies with the Win/Loss/Tied statistics with the best performing strategy using statistical comparison test adjusted for multiple comparison for $p < 0.05$, as detailed in section 4.2.3.

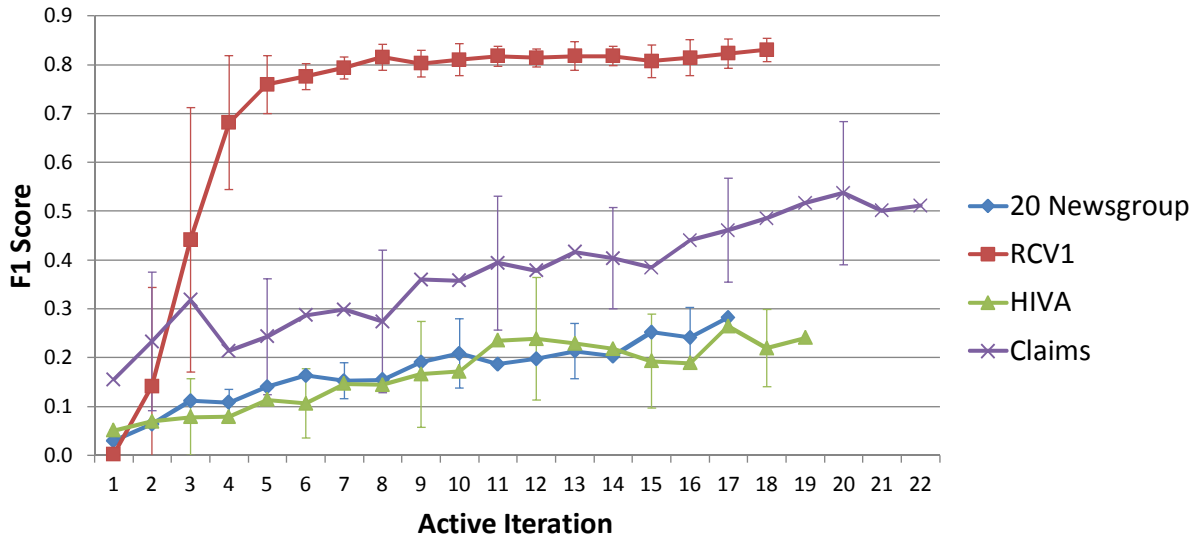


Figure 5.15: F1 Score performance for the 'Safe' strategy with Cost: Markovian model, Explore: Markovian - Error Reduction with Unsupervised estimation and Exploit: Classifier probability score across all datasets. Error bars show the variance across 10 independent trials and are staggered for better readability.

which optimization would work for a given task. In order to characterize the difference between the domains that leads to the different choice of optimization algorithms, we analyze the inter-relationship between the different factors of cost, exploration and exploitation.

We conjecture that the better performance of BMC: greedy strategy for the HIVA dataset is due to a combination of two factors. Firstly, HIVA is a hard domain to model with linear classifier and the classifier probability scores for HIVA dataset are not very accurate as observed with the very low F1-Scores for the performance over active iterations in figure 5.15. We observe that the F1 score for HIVA dataset is less than 0.26 across all active iterations. 20-Newsgroup dataset also has similar low performing F1 score of less than 0.28. Thus the inaccurate probability estimates from classifier leads to poor exploitation as well exploration utility estimates for HIVA and 20-Newsgroup datasets. Secondly, the rank correlation between the exploitation, exploration and the cost model is very high for the HIVA dataset which is not the case for the 20-Newsgroup dataset. Table 5.8 shows the pair-wise rank correlation between the rankings produced by the cost, exploitation and exploration model for the 'safe' performing configuration for all the datasets averaged over all active iterations and 10 random trials[8]. For the HIVA dataset we observe a rank correlation coefficient of 0.95 between exploration and exploitation, 0.52 between exploitation and cost and 0.5 between exploration and cost. The intuitive interpretation of high rank correlation coefficient is that an example that gets high exploitation score from the model also has high exploration score and has a lower cost. Thus the greedy algorithm that selects the examples based on the utility to cost ratio gives good performance as there isn't much trade-off between cost, exploit and explore model. So for any new domain, we would need to empirically characterize if the rank correlation between the factors is high. However it is not enough for only the rank correlation to be high for the Greedy approach to be better than the Khuller et al's Approximation

[8]Since the markovian cost model doesn't produce a static ranking, we use the underlying variable cost model to determine the ranking for the cost model

algorithm as the approximation algorithm selects the better solution between the greedy approach using benefit/cost ratio and a cost agnostic benefit-only approach. In reference to Algorithm 1, H_1 is the cost agnostic benefit-only solution and H_2 is the solution maximizing the benefit/cost ratio. Hence if the benefit estimates are not good then the Approximation algorithm is not guaranteed to make better selection than the Greedy approach whereas if the rank correlation is not high then the greedy solution is not guaranteed to be good. Thus a combination of the two phenomenons of high rank correlation between the factors and poor utility estimates enable the Greedy approach to be better than the Approximation algorithm for HIVA dataset. Since we have observed this phenomenon of BMC:greedy algorithm performing better than BMC:Approximation algorithm for only one dataset in this thesis, we do not make strong generalization claims regarding the characterization of the datasets/tasks that lead to such behavior. It is a very interesting future direction to show more evidence for this conjecture.

As the universally 'safe' strategy contains the novel ideas of markovian cost modeling and estimated future error reduction strategy using unsupervised estimation that have been proposed in the thesis, it validates the thesis and the proposed framework.

5.4.2 Revisiting the submodularity assumption for batch learning

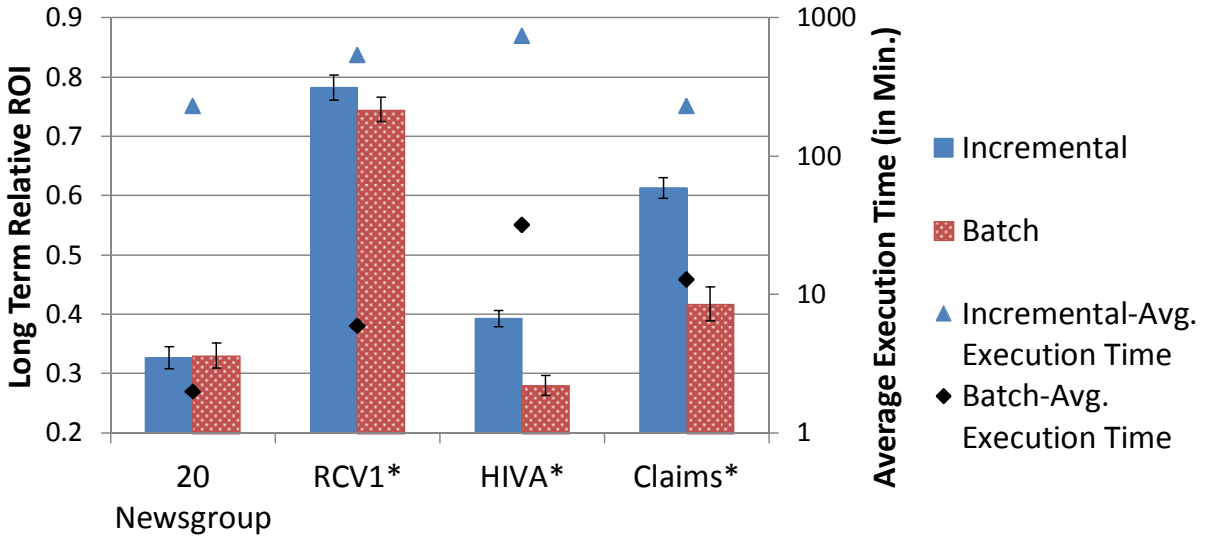


Figure 5.16: Comparing incremental selection with batch selection of examples within the framework. '*' indicates that the difference is statistically significant at $p < 0.05$ for the dataset. Error bars show the variance across 10 independent trials. The cumulative system execution time across all time periods (averaged across 10 independent trials) is represented on the secondary axis with log scale.

We have chosen to make the practical assumption of ignoring the submodularity consideration in active learning due to computation cost, particularly for estimated future error reduction strategy, as mentioned in the section 4.1.2. We find that for 3 out of the 4 datasets, namely Claims, HIVA and RCV1 (figure 5.16), ignoring submodularity is not a good assumption and we achieve significantly better performance by taking into account submodularity compared to ignoring it. For 20-Newsgroup, the performance by taking into account submodularity is very

Dataset	Number of Examples labeled in Batch mode(Percentage positive)	Number of Examples labeled in Incremental mode(Percentage positive)	Overlap percentage of Batch mode selections with Incremental mode selections(Percentage positive)
20 News-group	56(68%)	149(11%)	36%(30%)
RCV1	88(37%)	96(42%)	44%(46%)
HIVA	24(96%)	27(52%)	54%(100%)
Claims	26(8%)	24(50%)	11%(67%)

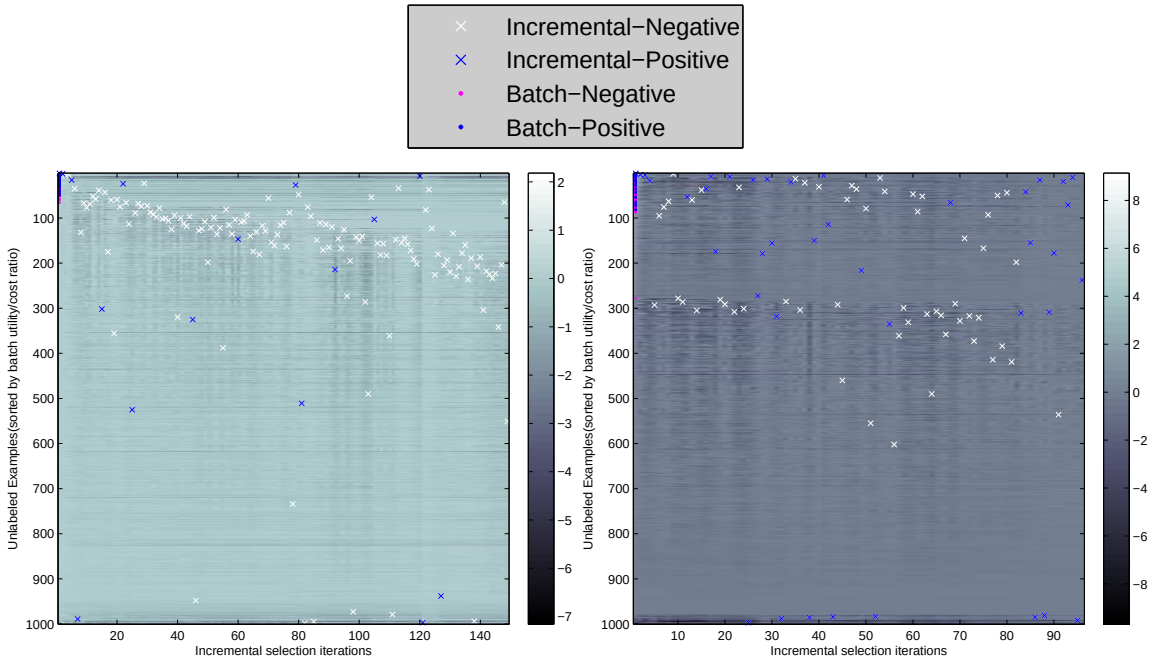
Table 5.9: Summary numbers for the data shown in figure 5.17, comparing batch selection with incremental selection.

similar to the performance by ignoring it. As the average execution time while taking into account submodularity is 2 orders of magnitude higher compared to ignoring submodularity, the practicality of taking submodularity into account is a domain dependent decision to trade-off higher performance with the system's response time.

To recollect, Submodularity [Nemhauser et al., 1978] is a property of set functions that intuitively formalizes the idea of 'diminishing' returns. That is, adding some instance x to the set A provides more gain in terms of the utility function than adding x to a larger set A', where $A \subset A'$. Informally, since A' is a superset of A and already contains more data, adding x will not help as much. By ignoring the submodularity consideration, we evaluate the exploration and exploitation utilities (and markovian cost) of all the unlabeled examples once and choose to label the selected examples in a batch. If we want to take into account the submodularity property then we should label each example incrementally and re-evaluate the utilities (and markovian cost) of every example instead of labeling the examples in a batch[9].

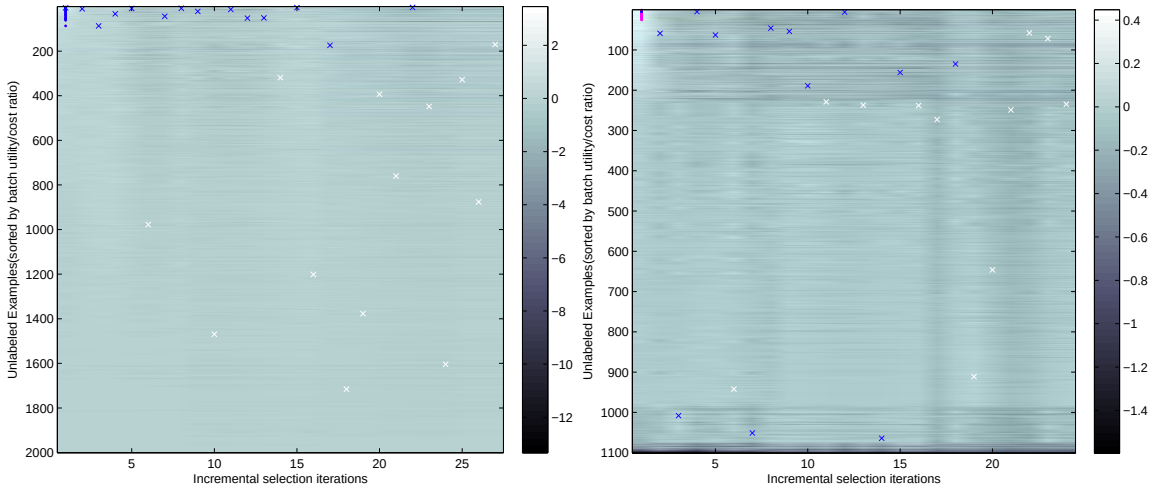
In order to study the effect of submodularity property within the interactive framework, we analyze the change in the utility to cost ratio of the unlabeled examples when we label the examples incrementally and re-evaluate the exploration and exploitation utilities after each iteration. We chose the following configuration to run the analysis: Cost - Variable; Explore: Markovian-Error Reduction-Oracle; Exploit-Classifier. We decided to use the Variable cost model instead of Markovian model due to the practical consideration of runtime for the experiments as the runtime for incremental experiments with markovian cost model is very high. The choice of Oracle based estimated error reduction strategy is to control for additional factors/variations that may be introduced due to other estimation methods for error reduction strategy. The choice for Classifier probability based Exploitation model is obvious to study the effect of submodularity on the exploitation utility. We use the BMC: Approximation algorithm for 20-Newsgroup, RCV1 and Claims dataset and BMC: Greedy algorithm for HIVA dataset based on the analysis of optimization algorithms for the different datasets in the previous section 5.4.1. Also we choose to analyze the results for the second time period of unlabeled data in the *Daily Classification Task* so that the starting point for both batch and incremental selection strategies are the same (in the first time period the examples are always randomly selected for all configurations).

[9]In our experimental setup of *Daily Classification Task*, we get a set of unlabeled examples in each time period. While labeling the subset of examples within each time period, we have a choice of either labeling them in batch or incrementally.



(a) 20-Newsgroup: Number of examples selected in Batch mode:56(Pos:38), Incremental mode:149(Pos:16), Overlap:20(Pos:6)

(b) RCV1: Number of examples selected in Batch mode:88(Pos:33), Incremental mode:96(Pos:40), Overlap:39(Pos:18)



(c) HIVA: Number of examples selected in Batch mode:24(Pos:23), Incremental mode:27(Pos:14), Overlap:13(Pos:13)

(d) Claims: Number of examples selected in Batch mode:26(Pos:2), Incremental mode:24(Pos:12), Overlap:3(Pos:2)

Figure 5.17: Comparison of incremental selection with batch selection for the strategy: Cost: Static; Explore: Markovian-Error Reduction-Oracle; Exploit-Classifier. For each incremental iteration the utility/cost ratio of all the examples in the unlabeled pool are shown in the heatmap to contrast their changing values with each incremental selection. First incremental iteration corresponds to the values for the batch selection. Caption for each dataset lists out the number of examples labeled in batch and incremental mode, and the overlap in their selection. The numbers in parenthesis (Pos:*) indicates the number of positive examples labeled.The color scale of each graph is different in order to highlight the details.

Figure 5.17 shows the comparison of incrementally selecting the examples versus batch selection and the change in the utility (benefit) to cost ratio after each incremental iteration for one random trial for second time period in *Daily Classification Task*. We observe that the benefit/cost ratio of the examples changes across iterations for all datasets, highlighted in the images by the changing color of the row (each example) across iterations. Table 5.9 summarizes the observations of the figure 5.17. We note that the number of examples labeled during batch labeling versus incremental labeling are very different for 20-Newsgroup and RCV1 dataset but not so much for HIVA and Claims dataset. The overlap between the strategies ranges from 11% for Claims dataset to 54% for the HIVA dataset. Also the number of positive examples that are labeled for each strategy are also different between the Batch and Incremental strategies. For example for Claims dataset batch mode finds 8% positive examples whereas in incremental mode we find 50% positive examples whereas for 20-Newsgroup dataset batch mode finds 68% positive examples but incremental mode finds only 11% positive. Thus for the second time period, we observe that there is a significant difference in the performance of batch versus incremental strategies. However since this data is for only one trial and for the second time period in *Daily Classification Task*, we do not draw general inferences about the long-term effect of incremental labeling from this illustration.

In order to assess the impact of the incremental labeling scheme we compare the Long-term Relative ROI for the batch and incremental strategies, presented in figure 5.16. The figure also shows the average aggregate run-time of the experiment(all time-periods in *Daily Classification Task*) in minutes and we observe that the run-time of incremental labeling experiments is 2 orders of magnitude higher than batch labeling. We obtain statistically significant improvements over batch labeling for RCV1, HIVA and Claims dataset, however there is no difference in results for the 20-Newsgroup dataset. The improvement for the incremental labeling as a percentage of the batch labeling performance for RCV1 dataset is: 5%, HIVA dataset is: 40% and Claims dataset is: 47% whereas it is degraded for 20-Newsgroup dataset by $\tilde{1}$%. This observation means that ignoring submodularity for the 20-Newsgroup dataset is a reasonable assumption but for the other datasets/tasks it may not be reasonable. However, the feasibility of incremental labeling depends on the run time of re-evaluating the utilities (and markovian cost) and then executing the optimization algorithm to make it's selection in combination with the domain's time sensitivity for waiting to get the next recommendation to review/label. For the configuration used for analyzing the incremental labeling in figure 5.16, the average incremental run-time per incremental labeling step for the different datasets are - 20-Newsgroup:8.5 seconds, RCV1:23.3 seconds, HIVA: 94.6 seconds, Claims: 26.5 seconds (details about system hardware and software in section 5.4.9). In other words, for HIVA dataset it takes an average of 94.6 seconds to select an example to label incrementally. For the chemo-informatics task (HIVA dataset) 94.6 seconds may be a practical amount of time to wait in order to get the recommendation for which chemical compound to test in-vitro next. However for the information filtering task using RCV1 dataset, waiting 23.3 seconds for next relevant document to be shown to the user may not be practical. Thus the determination of practicality of incrementally labeling examples is dependant on domain's time sensitivity which can be considered as a comparison between the true labeling time versus the system time for making the next recommendation. In other words, if reviewing an example takes days (for example in chemo-informatics domain) then waiting for a few seconds for the system's recommendation is acceptable but if the labeling takes just a few minutes (for example information filtering, health insurance claim error review) then waiting for a few seconds may not be acceptable. For batch mode labeling, the choice of examples to review/label is made offline

and hence doesn't have a system run-time concern.

Improving the system run-time performance for incremental labeling

The system execution time that we obtained utilized a brute force implementation where we reevaluated all the utilities (and markovian cost) for all the unlabeled examples. However the toolkit that we have used for implementing the optimization algorithm, Submodular Function Optimization [Krause, 2010], specializes in fast and efficient implementation for optimizing submodular functions. Thus incorporating the utilities and cost calculations as submodular functions within the toolkit would enable us to improve the run-time of the system in future.

5.4.3 Overlap between learning strategies

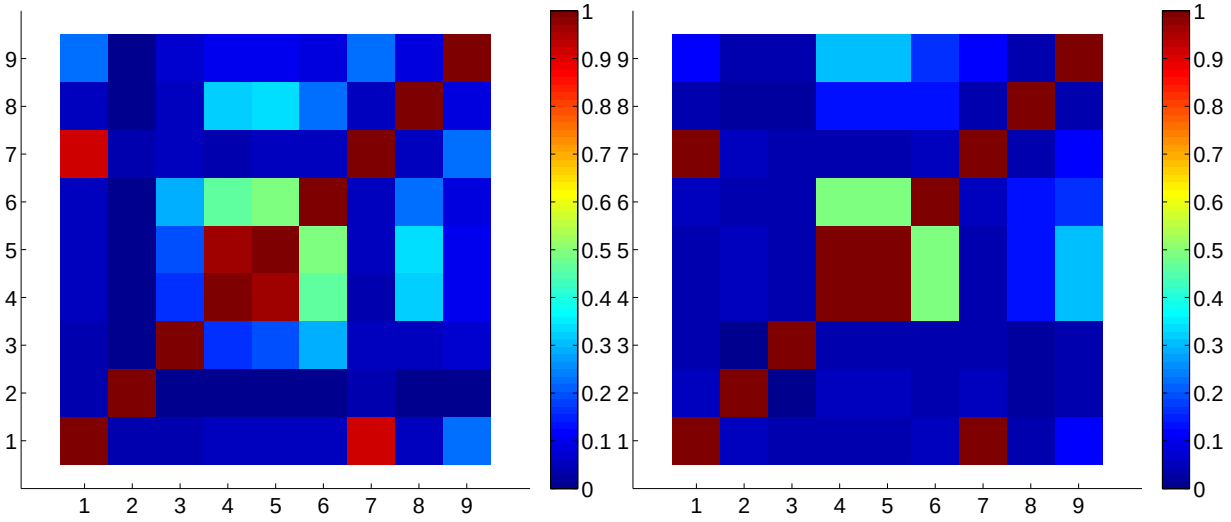
One fundamental question to ask is whether the different learning strategies have different information content and they make different selections. If the information content for the different strategies is the same, then it is not valuable to run multiple experiments across multiple strategies. We find that the information content of the different sampling strategies is different and thus justified comparing them further within the framework. We are particularly interested in the overlap between error prediction strategies ('markov:error reduction-supervised' and 'markov:error reduction-unsupervised') and the classifier ('markov:classifier') strategy as those strategies are relevant for the choices within the framework.

The methodology to observe the overlap is that we compare the choices made by each strategy for the second time period in the *Daily Classification Task*. The reason is that in the first time period examples are randomly selected to be reviewed/labeled, hence all the strategies have the same common model after first iteration. Thus whatever difference get highlighted based on the different strategies can be attributed to the strategy themselves as other factors (including the learnt model so far) are common.

Figure 5.18 shows the mutual overlap between strategies for the four datasets as a heatmap with the color intensity indicating the degree of overlap. We observe that most of the strategies (including estimated error reduction strategies) do not have significant overlap amongst each other, which is a desired result that each strategy has different information content and focuses on different aspects while making example selections. The strategies which have some reasonable overlap are strategies 'Markov:Uncertainty' (index 4), 'Markov:Classifier' (index 5) and 'Markov:Certainty' (index 6). The 50% overlap between 'Markov:Classifier' (5) and 'Markov:Certainty' (6) is expected by design as the 'Markov:Certainty' selects 50% top scored positive examples and 50% top scored negative examples. However the nearly 100% overlap between 'Markov:Uncertainty' (index 4) and 'Markov:Classifier' (index 5) is unexpected as it indicates that the most confident examples are the most uncertain ones as well. This is explained by the fact that we have skewed class domains and in the initial iteration when the classifier is not very accurate the number of examples getting positive scores is relatively small. So most of the examples have near-zero scores, hence the overlap between the strategies as the uncertainty strategy also selects near zero (close to decision boundary) examples.

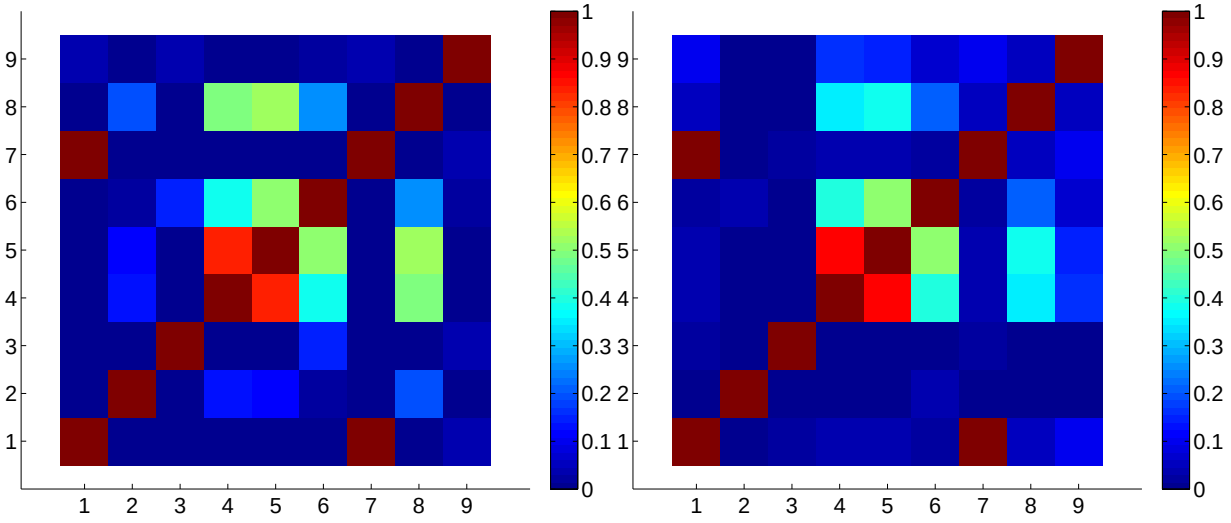
Based on this analysis, we note that the information content of the different sampling strategies are different and thus justified comparing them further within the framework.

Configuration Index	Exploration Strategy
1	Random
2	Variable:Density
3	Variable:Sparsity
4	Markov:Uncertainty
5	Markov:Classifier
6	Markov:Certainty
7	Markov:Error Reduction-Supervised
8	Markov:Error Reduction-Unsupervised
9	Markov:Error Reduction-Oracle



(a) 20-Newsgroup

(b) RCV1



(c) HIVA

(d) Claims

Figure 5.18: Figure showing the overlap between the example selections based on different exploration strategies. It is a symmetric matrix where the observation (i,j) shows the overlap of strategy index i with strategy index j. The values range from 0-1.

5.4.4 Potential limitation of Budgeted Max Coverage optimization algorithm with learnt cost models

When using a potentially noisy cost model (that is learnt from the data) it is likely that the Budgeted Max Cover optimization may not perform better than baseline Rank Fusion approach. The reason is that there is a key difference between the output of BMC and Rank Fusion. BMC gives a 'set' recommendation whereas Rank Fusion gives a complete ordering over the examples. So when using the rank fusion approach we can keep auditing the examples based on the suggested order until we exhaust the budget. Whereas in the optimization approach the algorithm has made its selections based on the noisy models. So the first potential limitation is that the approximation algorithms set selection may be susceptible to noise in the cost models. It is an important future direction to investigate the robustness of the algorithm in the presence of noise.

Secondly the approximation algorithm doesn't give a global ordering (even though it gives a greedy ordering of the elements within the selected set). Therefore the selected set of examples are the only candidate examples to label. Thus in order to optimally use the budget, we would need cost models that in general under-estimate the cost rather than over-estimate. With a cost model that under-estimates the cost we can still use the ordering provided within the selected set to audit examples until the budget gets exhausted. But if the cost model overestimates then we would audit all the examples within the selected set but still not completely utilize the budget. Thus it is also imperative to understand the behavior of the framework with different characterizations of the cost model in terms of under-estimation or over-estimation.

5.4.5 Dealing with class imbalance

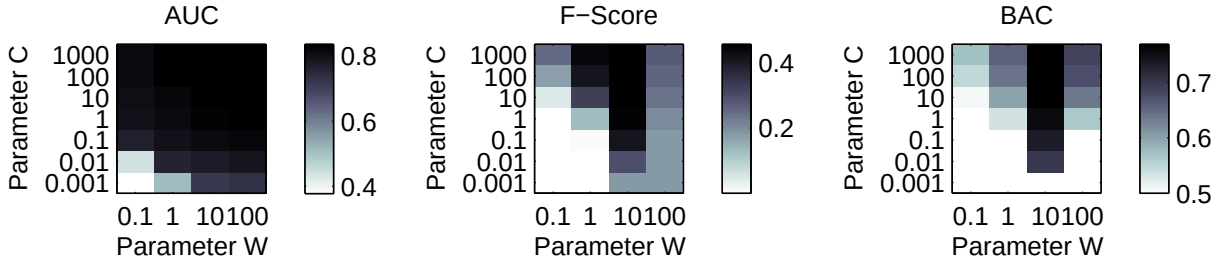
There are three major approaches in machine learning literature [Japkowicz and Stephen, 2002] that are used for dealing with class imbalance and improving classification performance, namely, oversampling minority class, undersampling majority class and modifying misclassification costs per class. The intention with the strategies of oversampling and undersampling is to obtain a balanced training dataset. However in the context of maximizing the long-term return on investment for the learning system which is the focus of this thesis, the end goal is never to obtain a balanced labeled set but rather maximize the number of positive examples. Thus the only strategy to deal with class imbalance that is relevant in the context of the thesis and the interactive framework is to modify the misclassification costs per class.

We have used the liblinear package [Fan et al., 2008] for classification and use logistic regression as the learning algorithm. The package provides parameter for specifying the misclassification cost (parameter -c) as well as class dependent weight parameter for adjusting misclassification cost (parameter -w) for each class. Figure 5.19 shows the performance for three metrics of AUC, F1 Score and Balanced Accuracy (BAC[10]) for different parameter values for C and weight for positive class W with five-fold cross validation using entire labeled data for all datasets. It is interesting to note that there may be trade-offs between different metrics, for example for the HIVA dataset a choice of W parameter of 100 gives good BAC performance but poor F1 score whereas value of 1 gives good F1 Score but poor BAC. However, W parameter value of 10 gives reasonable F1 score and BAC for HIVA dataset. For our experiments we chose the parameters with the guiding principle of keeping the choices simple i.e. if we got similar performance without specifying class specific weight we prefer that choice. We empirically chose the following parame-

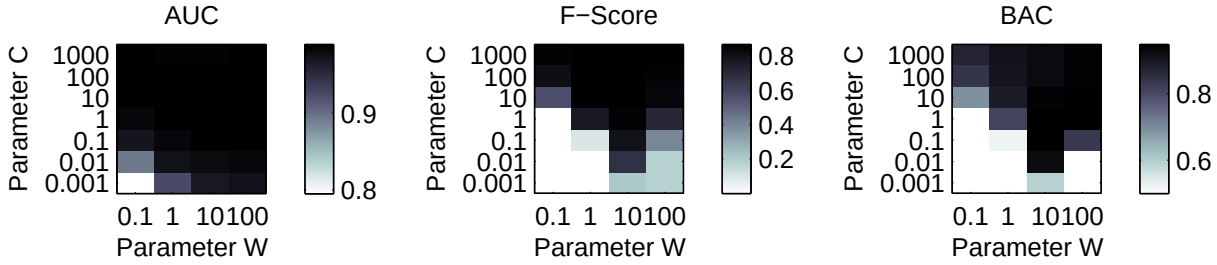
[10]Metric used in skewed class domains [Lichtenwalter and Chawla, 2010] which is the mean accuracy of individual classes

ters for the different datasets: 20 Newsgroup - C:100; RCV1 - C:100; HIVA - C:0.1 W:10; Claims - C:100.

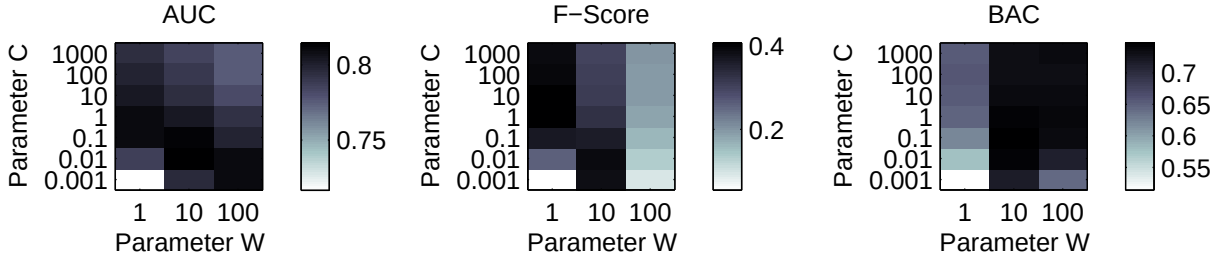
5.4.6 Importance of classifier parameter tuning for error reduction strategies to work



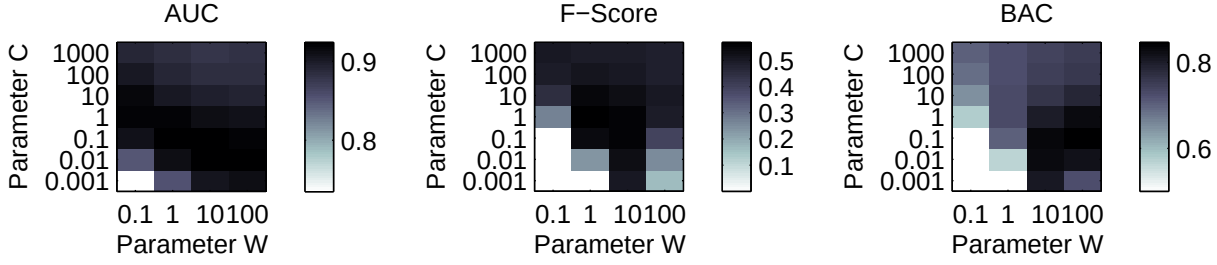
(a) 20-Newsgroup



(b) RCV1



(c) HIVA



(d) Claims

Figure 5.19: Different evaluation metrics: AUC, F-Score and Balanced Accuracy (BAC) for grid search of parameters C (cost for misclassification) and W (adjustment factor for positive class). The color scale for each figure is different in order to highlight the grid patterns.

Example Id	Class Label	Classifier Probability Output
1	+	0.4
2	+	0.35
3	-	0.3
4	-	0.25
5	-	0.2
6	-	0.15
7	-	0.1
8	-	0.05
9	-	0.05
10	-	0.05

Table 5.10: Illustrative test example with classifier probability output and different metrics. Different metrics are: AUC=1, Precision@2=100%, Precision=0, Recall=0, F-Score=0, Accuracy=80%, Balanced Accuracy (BAC)=50%.

An important consideration for the estimated error prediction strategies to be effective is to get accurate probability estimates for evaluating the exploration utility based on equations (3.5) and (3.7) (for unsupervised estimation), repeated here for clarity and context:

$$\begin{aligned} U(x) = \quad & E_{l\in 0,1}[U(D \cup < x, l >)] = \quad P(f(x) = 0|D).U(D \cup < x, 0 >) \\ & + \quad P(f(x) = 1|D).U(D \cup < x, 1 >) \end{aligned} \tag{3.5}$$

$$Precision@K(D) = \frac{\sum_{i=1}^{K} P(f(x_i) = 1|D)}{K} \tag{3.7}$$

where D is the labeled dataset, l is the label $\{0,1\}$ and $P(f(x) = l|D)$ is the classification probability, $x_{i...k}$ are the top scoring examples based on $P(f(x_i) = 1|D)$. To obtain accurate probability estimates we need to have good classifiers. Having accurate classifiers are expected in usual machine learning problems but since we are dealing with skewed classes and using more focused metrics like Precision@K, this is an additional but necessary consideration for the framework. In other words, if the only consideration for the framework was a metric like Precision@K then the approach would be to optimize the classifier as a ranker for a narrow operating range where the absolute scores that the classifier assigns do not matter and the focus would be on relative ranking. However for using the estimated error prediction learning strategies we need to focus on classifier performance in absolute terms thus getting reasonable probability estimates.

To illustrate the difference between optimizing for different metrics, consider a toy scenario in table 5.10 with 10 examples 2 of which are positive and rest are negative. The learnt classifier assigns them probability scores but the scores are not more than 0.5 for any example. Thus the classifier is not predicting any example as being positive. For this toy scenario, we get the following metric values AUC=1, Precision@2=100%, Precision=0, Recall=0, F-Score=0, Accuracy=80%, Balanced Accuracy (BAC)=50%. Thus we see that the two metrics AUC and Precision@2 which are related to the ranking of the classifier rather than the absolute score give perfect result. However the traditional classifier metrics of F-Score and balanced accuracy give very poor results. Thus we need to optimize the traditional classifier metric of F1 Score in addition to the metric directly relevant for the framework of Precision@K. This requires tuning the classifier parameters specifying the misclassification cost for positive and negative class which is the approach selected for dealing with class imbalance as mentioned in previous section.

5.4.7 Using different learning algorithms with the framework

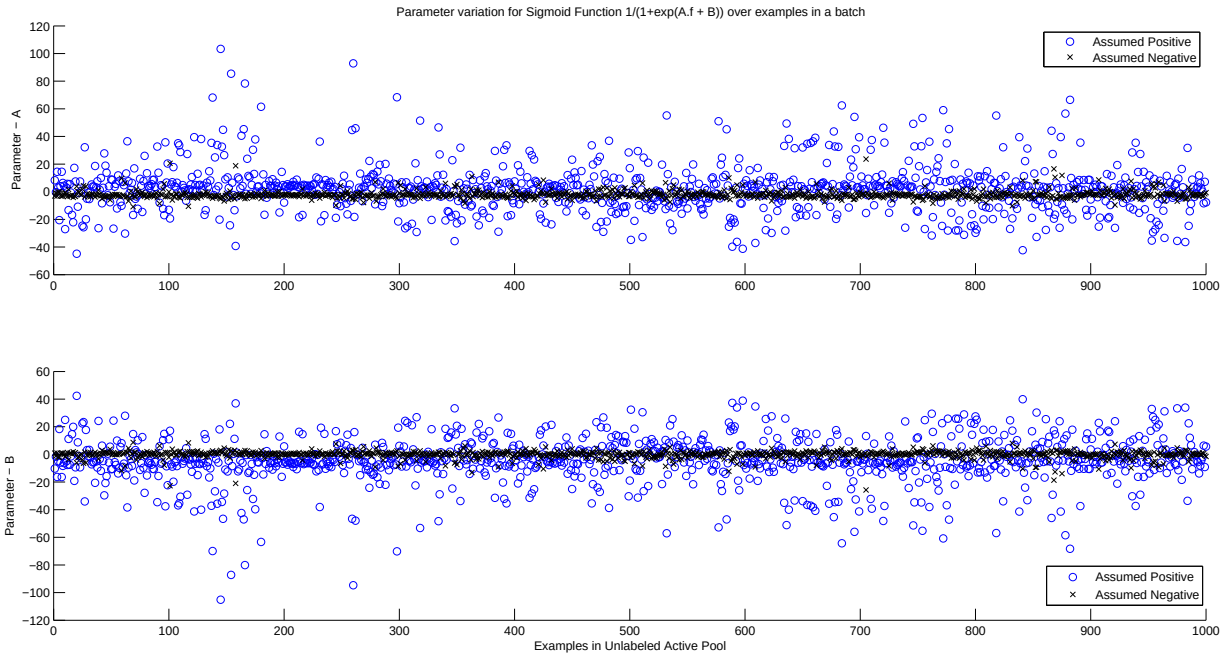


Figure 5.20: Parameter variation for Sigmoid function $\frac{1}{1+exp^{A.\hat{f}+B}}$ over examples in a batch. Each example represented along the x-axis is added to the existing training data as positive as well as negative in order to measure the exploration utility of estimated error reduction strategies based on equation (3.5).

In order to use any learning algorithm with the framework, the requirement is that it should output probability score. We used L2-regularized logistic regression learning algorithm for the framework as it outputs a probability estimate. We used the Liblinear package [Fan et al., 2008] (option -s 0 for version 1.93). However, any algorithm like Naive Bayes that gives probability output would be appropriate to use with the framework. The intention with this thesis is not to come up with the best performing system in practice for a task by choosing the most appropriate learning algorithm but to optimize the performance of any learning algorithm by trading-off the different factors of cost, exploration and exploitation. Thus while deploying a system for solving a business problem, the most appropriate learning algorithm should be chosen and utilized with the framework.

Use of SVM algorithm

We initially started the experimentation with SVM learning algorithm as it is known to be robust and is widely used as the classification algorithm. However SVM doesn't output probability score. SVM output score can be transformed into probability estimate using sigmoid transformation suggested by Platt [1999] as $\frac{1}{1+exp^{A.\hat{f}+B}}$, where A and B are estimated by minimizing the negative log-likelihood function, and $\hat{f}$ are the decision values of training data. We use the LibSVM package for the SVM algorithm implementation which utilizes an improved implementation of Platt's posterior probabilities [Lin et al., 2007].

The challenge in using LibSVM's probability estimate for SVM with the framework was the interaction of learning the parameters A and B in Platt's posterior probability estimate along

with the estimated error reduction exploration strategy. The parameters A and B are learnt by minimizing the negative log-likelihood function over the training data. Whereas for the estimated error reduction strategy we estimate the utility of the example by adding the candidate example as a positive example and then as a negative example based on the equation (3.5), repeated here for clarity and context:

$$\begin{aligned} U(x) = \quad & E_{l\in 0,1}[U(D \cup < x, l >)] = \quad P(f(x) = 0|D).U(D \cup < x, 0 >) \\ & + \quad P(f(x) = 1|D).U(D \cup < x, 1 >) \end{aligned} \tag{3.5}$$

where D is the labeled dataset, x is the unlabeled example being evaluated, l is the label {0,1} and $P(f(x) = l|D)$ is the classification probability. When we estimate the parameters A and B while adding the example as potentially mislabeled, it leads to drastic changes in the parameter estimates. Figure 5.20 shows the variation of the parameter A and B while estimating the error reduction utility for all the examples of the unlabeled pool of batch data for one active iteration for 20-Newsgroup dataset. Each example represented along the x-axis is added to the existing training data as positive as well as negative in order to measure the exploration utility of estimated error reduction strategies based on equation (3.5). As the class skewness is 10%, we observe that since most of the examples are negative, whenever they are assumed to be positive, then the parameters A and B vary a lot whereas when they are assumed to be negative then the estimates are stable. This interplay between estimating error reduction utility and the SVM probability score leads to additional variance in the estimated error reduction strategy. Thus the value of the framework is not realized due to this undesired variance.

One approach to avoid this undesired variance while using SVM algorithm is to use a different implementation of the Platt's estimate where the parameters A and B are learnt from static held out data. This approach requires some experimentation to observe if we are able to get robust estimates of the parameters. One implementation of this approach can be done by modifying the LibSVM software package to take these parameters as input and the parameters are learnt independently. This is an interesting future direction to explore.

5.4.8 Few empirical parameter choices made for the framework

We have made a few empirical choices for certain parameters to reduce the number of variations that we experiment. First choice we made was in selecting the budget for each task/dataset. Based on our experience in deploying the system for Claims error prediction task, we had observed that the audit capacity for the domain was about 1.5% to 2% of all the transactions where the expected positive class percentage was 4-5%. So we kept the budget as 2% for the Claims domain for our experiments with the positive class percentage as 5%. We consider that the audit capacity was roughly 50% of the expected positive class. For the information filtering task we fixed the budget as 5% of all the documents which is 50% of the relevant document percentage of 10%. For the chemo-informatics task (HIVA dataset), we made a slightly different choice to keep the budget as 100% of the expected positive class percentage. This choice was made because of two reasons, firstly the positive class percentage is 3.5% and so expecting the budget to be 3.5% of all transactions is reasonably realistic and at par with the real world observation. Secondly, the reward for finding a positive compound is so high for the chemo-informatics task that it makes logical sense to have a budget which is at least the expected positive class percentage. In general, we expect the relative performance improvement over baselines due to the framework to reduce as we increase the budget because the degree of trade-offs between different factors is reduced.

Second choice that we made was to fix the amortization time period to 1 time period for the exploration utility in equation (3.4), repeated here for clarity and context.

$$NPV(a) = E * T * \tilde{V} * a \tag{3.5}$$

where E is the Expected number of examples audited per time period, T is the amortization Time period, $\tilde{V}$ is the expected Value of each example and a is any given function. By fixing the amortization period T to be 1 time period we are intuitively saying that long term exploration is not that important and the focus is on immediate improvement for the learner. We initially experimented with longer amortization period but found that in the context of the evaluation metric that is relevant for the framework of finding as much exploitation utility as possible in the given budget shorter amortization duration that gives less importance to the exploration utility yields better ROI.

Third choice was in fixing the α weighing factor between exploration and exploitation utility as a constant value of 0.5 thus giving equal weight to both utilities. The reason for making this choice was to reduce the number of experimental variations that we need to run.

5.4.9 Hardware, software setup and experiment run times

The system has been implemented in Matlab®and the experiments are run on two 32-processor Intel®Xeon®E5-2450 @ 2.10GHz machines with 1 Tb RAM running Ubuntu OS in the Carnegie Mellon University Speech cluster. The choice of Matlab and multi-core hardware support allowed us to efficiently parallelize the experiments (using parfor construct in Matlab) and run multiple variations across datasets simultaneously. We have used the liblinear package [Fan et al., 2008] for learning algorithm implementation as it provides state-of-the-art fast implementation of linear SVM and logistic regression algorithm and has a very efficient Matlab interface. For implementing the optimization algorithm we have used the Submodular Function Optimization toolkit [Krause, 2010] which is also written in Matlab. The total system run-time for all the experiments represented in summary figure 5.7 is 30 hours for 20-Newsgroup, 77 hours for RCV1, 236 hours for HIVA and 123 hours for Claims dataset.

.

Chapter 6

Temporal Active Learning

6.1 Introduction

In tasks that we have considered in the thesis (Error Prediction, Information Filtering as well as other similar domains like Fraud Detection, Intrusion Detection, Medical Diagnosis and Video Surveillance), the target concept and data distributions change over time. It has been shown that a static classifier can often result in reduced accuracy in such *temporal drift* scenarios. Temporal drift can happen due to the changing data distribution, the changing nature of the classification problem, or adversarial actions among other reasons.

Active Learning algorithms have traditionally dealt with problems where the target concept is stationary over time. Thus even though our proposed framework inherently learns over time by taking into account the exploration utility, it is not clear how effective are those techniques in dealing with temporal drift scenarios. Since there has not been much work in the area of studying active learning performance in the presence of drift, we start by asking simple questions. Does random sampling work better than more intelligent sample selection strategies? Which sample selection techniques are appropriate for different kinds of temporal drifts? Do these strategies depend on the choice of different evaluation metrics? Can the techniques developed in the temporal drift literature be combined with active learning strategies to give better results? Or, Is there a need for completely new active learning approaches to tackle temporal drift?

We present an empirical comparison of existing active learning strategies in the presence of temporal drift. We show empirical performance of a variety of active learning approaches combined with two approaches developed in temporal drift literature [Zliobaite, 2010] that are designed to deal with temporal drift. The strategies for adapting the classifier to take into account drift are firstly, to assign higher weight to recent instances which makes the misclassification cost of recent examples higher than older examples. Secondly, learning different classifier for separate time period and combining them in an ensemble with weights where the recent time period's classifier get higher weight. Since this is primarily an empirical analysis, we want to generalize the results beyond the specific data sets we used. In order to accomplish that, we parameterize real-world domains using the expected type and amount of drift, the target class distribution, evaluation metric of interest, and the amount of available labeled data. We take two real-world problems, information filtering and network intrusion detection, and generate sub-problems by varying the domain parameters resulting in multiple problem setups with different data characteristics. For each of these problem setups, we perform experiments comparing the performance of several active learning algorithms combined with the approaches to adapt the

classifier for drift.

Our results allow us to understand the difference in performance of various techniques for different domains. In general, we find that some active learning techniques designed for stationary data distributions outperform random sampling strategies under certain characteristics. However, there are several techniques that perform similar to or even worse than random sampling for many of the problem settings tested in our experiments. There is no universally good configuration that performs well across different drift characteristics. To the best of our knowledge, this is the first study that provides an empirical comparison of a wide variety of active learning and temporal drift approaches for a diverse set of real world problem settings. The study provides crucial insights into interrelationships between the performance of active learning strategies, domain characteristics and target evaluation criteria in various temporal drift settings.

6.2 Framework for Empirical Comparison

When we started this work, we spent some time investigating approaches to empirically compare different active learning strategies when applied to problems exhibiting temporal drift. We found that no such framework existed and that empirical comparison was not a trivial exercise. In this section, we present a framework that allows researchers and practitioners to compare the performance of various active learning techniques under temporal drift in a broad range of real-world problems. More specifically, we focus on problem setting where a classifier periodically provides the experts with a ranked list of cases to review and verify. The general setting is analogous to the *Daily Classification Task* introduced by Forman [2006] for studying concept drift. Depending on the problems, time is discretized into periods (e.g. days) and of all the new data that comes in during that period, a subset of it is labeled based on the active sampling strategy. For instance, in information filtering task recommendations can be provided every hour or day or week. The number of examples that the expert can review/label is also typically determined by the domain. For example, for health insurance claims error prediction [Kumar and Ghani, 2011], the number of audited cases are determined by the daily audit capacity of the insurance company which is close to 2% of all new claims that come in a day.

We structure our analysis to first characterize a real-world domain using several parameters that can be determined apriori by a domain expert:

1. *Type of Drift:* Fickle Concept Drift (FCD), Shifting Subclass Distribution (SSD).

2. *Amount of Drift:* We vary the amount differently for each type of drift (described later in this section).

3. *Target Class Distribution:* We specifically deal with skewed class distribution but vary the skewness depending on the domain.

4. *Evaluation Metric of interest:* AUC, Precision@10, and Precision@1.

5. *Cost of Labeled Data:* We incorporate this cost as the amount of data that can be labeled by domain experts in each iteration. For the experiments in this paper, we set this to either 10 or 100 examples for each iteration.

We generate permutations of above parameters to result in 144 different problem settings on two real-world problems, information filtering and intrusion detection. We empirically compare the performance of several algorithms combining active learning strategies with temporal drift motivated strategies:

1.Active Learning (Sample Selection) Strategies: Certainty sampling, uncertainty sampling, density-based sampling, and sparsity-based sampling, random sampling.

2.Learner Adaptation based on historical data: Single model with instance-level weighting, and weighted Ensemble models. We employ three weighting schemes: uniform, linear, and exponential.

Each combination of learning strategies is implemented and applied to all of the 144 problem setups for the two domains. For instance, one learning strategy would result from combining certainty sampling with uniformly weighted ensemble model which can be applied on a problem setup such as that resulting from choosing FCD, High Drift, 10% class distribution, Precision@1, and 10 examples labeled in each iteration.

6.2.1 Domain characterization

Type and Amount of Drift

We experiment with two types of drift scenarios, Fickle Concept Drift (FCD) and Shifting Subclass Drift(SSD) [Forman, 2006]. FCD is defined where an individual case may have different class labels at different times. For example, in information filtering system the user's preference for relevant news articles may change over time. This kind of drift can be characterized by rate of change in user's preference over time. Thus the amount of drift is parameterized by the probability of switching from one class of interest to another (randomly selected) for the next time period. Even though it can be argued that the user interest may not switch randomly and there may be a semantic pattern to it (shifting interest from rec.sports.football to rec.sports.baseball), we chose to use random switching in order to be more general and not to introduce an additional bias factor of semantic pattern. We experiment with drift probabilities of 20%, 50% and 100% (labeled as CD0.2, CD0.5 and CD1.0 respectively in figures) as well as no drift scenario labeled CD0.0.

SSD happens when the positive or negative class comprises of a union of (potentially undiscovered) subclasses, and the distribution of these subclasses shift over time. For instance, in network intrusion detection, certain types of intrusions may show up over time as was described in the KDD Cup'1999 dataset [KDD Cup, 2009]. Consequently, while the feature distribution given a particular subclass may be stationary, the feature distribution of the super-class varies over time, because its mixture of subclasses varies. We parameterize the amount of drift by how frequently do new subclasses appear and old ones disappear. We experiment with two drift amounts: drift occurring every 2nd iteration (labeled as Drift=Low), and drift occurring every iteration (labeled as Drift=High).

Target Class Distribution

Most large-scale enterprise data mining problems exhibit class skewness and the level of skewness varies from domain to domain. We experiment with skewness of 10% and 20% for the information filtering task and 1% and 2% for the network intrusion detection task. Although the natural distribution of intrusion cases is very high for KDD Cup network intrusion dataset, the typical percentage of intrusion cases is expected to be 1 to 2% which is widely used in studies employing this dataset [Eskin et al., 2002].

Evaluation metric of interest

Another important characteristic of real-world problems is the performance metric of interest. The choice of evaluation metric is dependent on the domain and the operating range of interest in

the domain. We choose the following metrics to cover a broad range of domains: Area Under ROC (AUC) curve, Precision@1st percentile and Precision@10th percentile. AUC metric is correlated with the ranking of examples throughout the entire range [Donmez and Carbonell, 2009] and relevant for applications where the performance over all the examples matters. Precision@Kth percentile is a more focused metric that helps distinguish the performance on the top scored cases making it more relevant for skewed classification problems.

Cost of Labeled Data

In typical static active learning the batch size for determining the number of cases labeled per active iteration is determined arbitrarily. However for interactive classification with well defined time periods for getting new unlabeled data, the number of cases to label is an important design choice which is also affected by factors like the budget for labeling. We experiment with representative batch sizes for labeling in a time period with 10 queries (low budget) and 100 queries (high budget - which corresponds to roughly the number of positive examples expected in new unlabeled batch for the two datasets).

6.2.2 Learning Strategies

We use Support Vector Machine (SVM) as the base classifier and employ various learning strategies as described below. We chose to use SVM instead of Logistic Regression, which has been used for evaluating the interactive framework, for two reasons. Firstly, SVMs have been known to be more robust learning algorithm and have been used in previous work related to temporal drift [Klinkenberg and Joachims, 2000]. Secondly, for evaluating the active learning strategies in the presence of drift there is no requirement of probability estimate output from the learning algorithm, hence SVMs are appropriate and we don't need to necessarily use Logistic Regression.

Active Learning (Sample Selection) choices

We experiment with four active sampling strategies and compare it to baseline strategy of random sampling. These include the classical settings of uncertainty and density based sampling and variants of those settings that have been found useful in skewed classification settings by Ferdowsi et al. [2011]. The variant of uncertainty sampling is certainty sampling where we sample the examples with high classifier confidence. For linear SVM classifiers, this is basically the distance from the SVM hyperplane as represented by the SVM score. We sample equally from most confident positive as well as negative examples in order to come up with a balanced training dataset. The variant of density sampling is density outlier sampling or sparsity sampling where we select the examples that are most sparse (least dense). We also evaluate the passive learning setup where all the data is assumed to be labeled.

Learner Adaptation strategies based on historical data

When building classification models from labeled data spanning some period of time, there are multiple ways to use the historical labeled data for the learner to adapt [Zliobaite, 2010]. We focus on two popular strategies: first, that builds separate temporal models from each time window that use only the labeled data from that window and then combine those models using ensemble techniques; and second, that builds a *single model* combining all the data (from all time periods) with instance weighting. *Ensemble models* have been popularly used to handle concept drift in

recent years by Kolter and Maloof [2007], where a classifier is learnt for each time period and then combined in a (weighted) ensemble. However a drawback of ensemble based methods is that they need enough labeled data for each time period to build a reasonably good model. In cases where there is not enough labeled data available for each time period, the ensemble based approach may not be ideal. The alternative strategy to build a *single model* makes the model more robust (in the presence of limited training data) but less adaptive to temporal drift. One way to compensate for that is by weighting the instances differently based on the time period they belong to.

We experiment with three weighting schemes, for both historical models (in an ensemble's case) and examples (in a single model case): exponential, linear and uniform. Exponential weighting scheme gives exponentially decreasing weight to history, linear weighting scheme gives linearly decreasing weight to the history whereas uniform gives equal weight to history.

6.3 Data Generation

In order to generalize our results and conclusions beyond the data sets we initially used, we use those data sets to then generate variations that span the spectrum in terms of the parameters mentioned earlier. All the datasets along with the relevant parameter details are made publicly available at `https://sites.google.com/site/mohitkum/data`. For the experiments, we report the results averaged over 10 random splits.

6.3.1 Fickle Concept Drift

We create FCD datasets based on the 20 Newsgroup and Reuters-RCV1 datasets [Chang and Lin, 2001, Rennie, 2007]. For 20 newsgroup, there are 18,774 documents corresponding to 20 news categories after pre-processing and data clean-up. For RCV1 dataset, the dataset is preprocessed as described by Bekkerman and Scholz [2008] where the label hierarchy is reorganized by mapping the data set to the second level of RCV1 topic hierarchy. The documents that have labels of the third or fourth level only are mapped to their parent category of the second level. The documents that only have labels of the first level are not mapped onto any category. Further, the multi-labeled instances are removed. Out of the resulting 53 second level topics, we select the top 20 most frequent topics and sample 1000 documents for each topic.

For creating datasets with fickle concept drift, for each time period we sample 50 cases each for the 20 categories for both datasets, resulting in 1000 documents per time period. This gives us 18 time iterations for 20-newsgroup data and 19 time iterations for the RCV1 dataset. We experiment with positive class percentage as 10% (2 out of 20 categories) and 20% (4 out of 20 categories). We test with 0%, 20%, 50% and 100% drift probability over each iteration. Figure 6.1 shows a sample iteration where the numeric IDs correspond to newsgroup categories, for example, category id 1 corresponds to alt.atheism. For the experiments, we report the results averaged over 10 random data sampling and labeling iterations for each setting.

6.3.2 Shifting Subclass Distribution

We derive two SSD datasets using 20 Newsgroup and KDD Cup network intrusion detection [KDD Cup, 2009] datasets. The choice of network intrusion dataset for this type of drift was natural as different types of intrusions occur at different times as described in the original KDD

	Time Period																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Positive Category A	16	16	16	16	8	8	8	20	20	20	20	12	12	12	12	12	12	12
Positive Category B	17	17	17	17	17	17	17	17	17	17	17	3	3	6	14	3	3	3

Figure 6.1: A sample iteration of 20-newsgroup data with 10% positive class percentage(2 categories) and 20% drift probability, indicating the positive category id for each time period

Intrusion Name	Time Period																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
warezclient	0	0	0	0	0	0	0	13	12	11	11	9	7	7	13	13	13	13

Figure 6.2: A sample of the number of cases of subclass 'warezclient' included in 18 time period batches for the KDD cup network intrusion dataset

Cup challenge as well. We ignore the training/test split as suggested by the original dataset and instead resample the data according to our strategy to create the temporal dataset.

For each time period, only a subset of the positive classes are present in a batch. We design the sampling scheme such that the first time iteration has only a few subclasses and then new subclasses get added over time while some existing ones are removed. The exact sampling schemes along with the data are shared publicly at `https://sites.google.com/site/mohitkum/data`.

Figure 6.2 shows the number of 'warezclient' intrusion cases included in the batches across 18 time periods for the network intrusion data for *high drift*. There are 40 intrusion subclasses in the dataset, however, we use the 25 most frequent ones. The negative class is predetermined for the network intrusion dataset (subclass: 'normal') and rest of the 24 subclasses are positive (intrusions). We create batches of 8000 datapoints for each time period with positive class varied between 1% and 2%. Sampling for 20-newsgroup is similar where we have 1000 cases in each time period with positive class varied between 10% and 20%. We arbitrarily select the 'talk' newsgroup category as the positive class with talk.politics.guns, talk.politics.mideast, talk.politics.misc and talk.religion.misc subclasses.

6.4 Results

In order to report the results, the performance metric(s) of interest, AUC, Precision@10 and Precision@1 are computed at each time period, and the average is calculated over all time periods similar to Forman [2006]. This performance is averaged over 10 randomized trials with different data samples to come up with a summary evaluation of each learning strategy choice, namely active sampling strategy, type of model (single vs ensemble) and weighting scheme (for historical instances or models) for various drifts (FCD/SSD with varying amount of drift) and domain scenarios (class skewness, cost of labeled data). Thus we get a ranking of 30 learning choices for 144 drift/data/performance metric scenarios. While we could choose to evaluate and report results on other measures or measure with finer granularity over Precision@k, the intent is to cover a reasonable representative range of parameters to observe different trends. We now present the analysis of results on some important criteria.

				CD0.0	CD0.2	CD0.5	CD1.0
20news	Precision@1	pos=10%	Q10	1*	1*	0	0
			Q100	1*	1*	0	0
		pos=20%	Q10	1*	1*	0	0
			Q100	1*	1*	1*	0
	Precision@10	pos=10%	Q10	1*	0	0	1
			Q100	1*	1*	1	0
		pos=20%	Q10	1*	0	0	0
			Q100	1*	1	1	0
	AUC	pos=10%	Q10	-1	0	0	1
			Q100	0	0	1	0
		pos=20%	Q10	0	0	0	0
			Q100	1	1	1	0
RCV1	Precision@1	pos=10%	Q10	1*	0	1*	0
			Q100	1	0	0	0
		pos=20%	Q10	1	0	0	0
			Q100	1	0	0	0
	Precision@10	pos=10%	Q10	0	0	1*	0
			Q100	1	0	0	1*
		pos=20%	Q10	1	0	0	0
			Q100	1	0	0	0
	AUC	pos=10%	Q10	0	0	0	0
			Q100	1	0	0	0
		pos=20%	Q10	0	0	0	0
			Q100	1	0	0	1

Figure 6.3: Summary results for Fickle Concept Drift. '1' indicates that best active learning strategy is statistically better than the best random strategy and '-1' indicates vice versa. '0' indicates that there is no statistical difference between the two strategies.

6.4.1 Usefulness of active learning?

One question that is worth addressing is whether it is even useful to do active sampling for datasets with temporal drift or simply doing random sampling would give the best or comparable to best performance. This is not an unreasonable question to ask since earlier research has shown that random sampling can often outperform active learning strategies under temporal drift [Žliobaitė et al., 2011]. We compare the best performing active sampling technique with the best performing random sampling over the trials by first undertaking a two-way ANOVA omnibus test followed by Bonferroni post-hoc test with ($p < 0.05$) significance level [Japkowicz and Shah, 2011] using the function multcompare in Matlab.

When there is drift, random sampling is not the best choice for nearly 50% of the domain setups. This is an interesting result since it shows that active learning strategies are useful in certain domains but there are a large number of settings where random sampling will perform better.

Figures 6.3 and 6.4 show the statistical significance determination results for FCD and SSD respectively. In the figures, '1' (green colored cell) indicates that best active learning strategy is statistically better than the best random strategy; '0' (orange colored cell) indicates that there is no statistical difference and '-1' (red colored cell) indicates that the best random strategy is statistically better than the best active learning strategy. $*$ indicates that the performance is 10% better for information filtering task (20 newsgroup and RCV1) and 1% for network intrusion detection task (KDD Cup), the significance of which is described in the next section. In general, uncertainty based sampling strategy is better than the rest of the active strategies across most of the data conditions. We describe more detailed results in the following sections.

Effect of type and magnitude of drift on active sampling?: Active sampling is very effective for SSD, figure 6.4 whereas less effective for FCD, figure 6.3, which is considered to be a more difficult drift situation [Forman, 2006]. For FCD, as we increase the drift, active strategies perform similar to random sampling (figure 6.3). The results do vary across datasets where we observe greater performance difference for the 20-newsgroup dataset compared to RCV1. We conjecture the reason for this to be that the categories in 20-newsgroup datasets are related to each other whereas for RCV1 they are very different, making historical labeled data more useful for 20-newsgroup than RCV1, independent of the sampling strategy. Further study is required to quantify the correlation across topics for 20-newsgroup and RCV1 and its effect in the presence of FCD. For SSD, we observe that active learning is very useful under different magnitude of drifts for 20-newsgroup whereas only for Precision@1 for KDD cup dataset. This is intuitively explained by the fact that the subclasses are closely related to each other in 20-newsgroup (same higher level category 'talk') whereas less so in KDD cup dataset.

Does the performance depend on evaluation metrics?: In general, we observe that performance varies more for focused metrics such as Precision@1 whereas for coarser metric such as AUC, the performance is less variable. This shows that if the domain of interest has a narrow operating range, such as many real-world problems with class skewness, the difference in performance of active sampling techniques with random sampling is more noticeable (figures 6.3 and 6.4). Comparing the performance between the worst and best performing sampling strategies gives an indication of the spread of performance and how sensitive the performance is to the choice of sampling strategy. We also compare the worst performing sampling choice with the passive learning setup where we assume that all the data is labeled and available for training which both gives us an upper bound on performance and also gives an indication of the scope of improvement for the different sampling choices and metrics. Figure 6.5 shows the relative difference in percentage

between the passive learning (labeled *all-data* in figure) and the best sampling method (including random) with the worst sampling choice (including random) for FCD for the following data setup for 20-newsgroup dataset: positive class percentage:20%, number of queries per iteration 100 and for the drift scenarios with probabilities:0, 0.2, 0.5 and 1 (labeled CD0.0, CD0.2, CD0.5 and CD1.0 respectively in the figure). The major pattern that is observable is that the difference between the best and the worst strategies is large for Precision@1 and reduces progressively for Precision@10 and AUC. Comparing the relative performance of the best sampling strategy with *all-data* indicates the scope for improvement using any intelligent sampling strategy. For all domain settings, this difference is smallest for Precision@1 and increases for Precision@10 and AUC. The same pattern holds true as drift amount increases from CD0.0 to CD0.2 and CD0.5. However an interesting observation is that when the drift amount is highest (CD1.0) i.e. when in each iteration the positive class is completely changed, the performance of best sampling strategy is better than *all-data* (comparing the CD1.0 observation across 'All-data vs Worst' and 'Sampling vs Worst' columns). This is probably because for *all-data*, the history is not quite relevant in learning new class definition. This shows that the history is not useful when drift is extremely high and it is better to use samples of new data and minimize the use of history in learning.
Are the patterns different for different levels of class skewness?: There is no significant pattern observable with the different class skewness for comparable data setups for FCD or SSD from figure 6.3 and 6.4. For FCD-RCV1 dataset, we observe that there are certain higher skew setups (pos=10%) where active sampling performs statistically better than random sampling but it is hard to generalize based on observations from one dataset.
Are the patterns different for different number of queries per iteration?: In general, for FCD, active sampling with more queries performs statistically better than random compared to less number of queries (10). Whereas for SSD, we observe that for KDD dataset with higher number of queries, random sampling performs significantly better than active sampling for Precision@10 and AUC.

6.4.2 Practical Considerations

In addition to the results of our experiments, we also care about the practical implications of our results and whether the results make it worth implementing more sophisticated systems.
Is the performance difference valuable enough and worth the cost of implementation?: For practical deployment the actual difference in performance achieved is very important in order to justify the value (and added cost of system complexity) of doing active learning. If the difference in performance is not above a certain domain-dependent threshold, the value of spending the additional effort in terms of system development and deployment for active sampling may not be justified. Note that this is not necessarily the same as obtaining a statistically significant difference. The threshold for the justification of effort may vary across applications, for example increasing the precision@1 by 5% (which may or may not be a statistically significant improvement) can be very significant for applications like credit card fraud detection whereas may not be as valuable for information filtering. We choose a threshold of 10% for the information filtering tasks (FCD: 20-newsgroup, RCV1 and SSD: 20-newsgroup) whereas 1% for network intrusion task (SSD: KDD cup) and highlight the results with a '*' in figures 6.3 and 6.4. So a cell has a '*' if the difference between the best active sampling strategy and the best random strategy is more than X%. We observe valuable performance difference for more focused metrics Precision@1 and Precision@10 but do not have this level of performance difference for AUC. However it may be argued that since AUC metric is sensitive towards the performance across the entire operating range, the

threshold for AUC should be lower in order to justify the improvements.
Labeling more examples randomly vs. less examples intelligently: We compare the scenarios where we label 100 queries randomly with scenario where we label 10 queries using active sampling. Figure 6.6 and 6.7 show the results of the statistical significance comparison for FCD and SSD respectively. As most of the cells have '-1' which indicates that labeling more data randomly is statistically better than intelligent sampling. So the general observation is that labeling more data randomly almost always gives better performance than intelligent sampling. One practical implication of this observation is that if the cost of setting up intelligent sampling is high it may be worth spending the resources on sampling more data randomly instead. This may in general be true for non-drift situations as well and may be correlated with sample complexity measures [Settles, 2009] typically used to estimate sample complexity to reach passive learning performance.

6.4.3 Detailed Results

Figure 6.8 and 6.9 shows the heatmap of the difference in performance for each learning choice with the best performing choice for each domain setup for FCD and SSD respectively. The heatmap has separate images for the three different metrics to show the details as the scale of differences is very different for the three metrics. A '0' value in the heatmap indicates the best performing modeling scheme. The larger the value in this heatmap for the modeling strategy the worse it performs. Figures 6.10 and 6.11 show the indexing scheme for all the heatmaps in figures 6.8 and 6.9 respectively. The data in figure 6.10 corresponds to figure 6.8(a) and in figure 6.11 corresponds to figure6.9(a). For example, for figure 6.8(a), row index 3 corresponds to row 3 of figure 6.10, where the data has No Drift; percentage of positive examples is 20% and number of queries per time period is 10. The best performing learning choice is for index 28 which corresponds to an ensemble model with uniform instance weighting scheme and using 'uncertainty' active sampling strategy.
Which active sampling strategy in general performs better?: In general, uncertainty sampling is the best active sampling choice. One difference between FCD and SSD is that density based sampling performs much better for FCD (being the best choice for quite a few cases) whereas it's not one of the best performing strategies for SSD. For SSD, the second best choice for active sampling is certainty based sampling.
Which type of model in general performs better?: For SSD, the best performing models are single models rather than ensembles. For FCD, ensemble models perform better than single models. The intuitive explanation is that since the true concept is not changing for SSD (only the subclass frequency within the broader concept is changing), learning a single model that represents the concept is better. For FCD, where the true concept is changing, learning models for different time periods which represent the concept for that time period helps by possibly learning disjoint concepts which is not possible with a single linear model.
Which weighting scheme in general performs better?: For FCD, the exponential weighting scheme works better than the linear and uniform weighting schemes for both single and ensemble model types. For SSD, a linear weighting scheme works better than the exponential and uniform weighting schemes. This difference in weighting scheme for FCD and SSD seems intuitive because for SSD, history is more useful than for FCD and forgetting the history slowly (linearly) helps for SSD whereas forgetting the history faster (exponentially) helps for FCD.

				Drift=Low	Drift=High
20news	Precision@1	pos=10%	Q10	1*	1*
			Q100	1*	1*
		pos=20%	Q10	1*	1*
			Q100	1*	1*
	Prec@10	pos=10%	Q10	1*	1*
			Q100	1*	1*
		pos=20%	Q10	1*	1*
			Q100	1*	1*
	AUC	pos=10%	Q10	1	1
			Q100	1	1
		pos=20%	Q10	1	1
			Q100	1	1
KDD	Precision@1	pos=1%	Q10	1*	0
			Q100	1*	1*
		pos=2%	Q10	1*	1*
			Q100	1*	1*
	Prec@10	pos=1%	Q10	0	0
			Q100	0	0
		pos=2%	Q10	0	0
			Q100	0	0
	AUC	pos=1%	Q10	0	1*
			Q100	0	0
		pos=2%	Q10	0	0
			Q100	0	0

Figure 6.4: Summary results for Shifting Subclass Drift. '1' indicates that best active learning strategy is statistically better than the best random strategy and '-1' indicates vice versa. '0' indicates that there is no statistical difference between the two strategies.

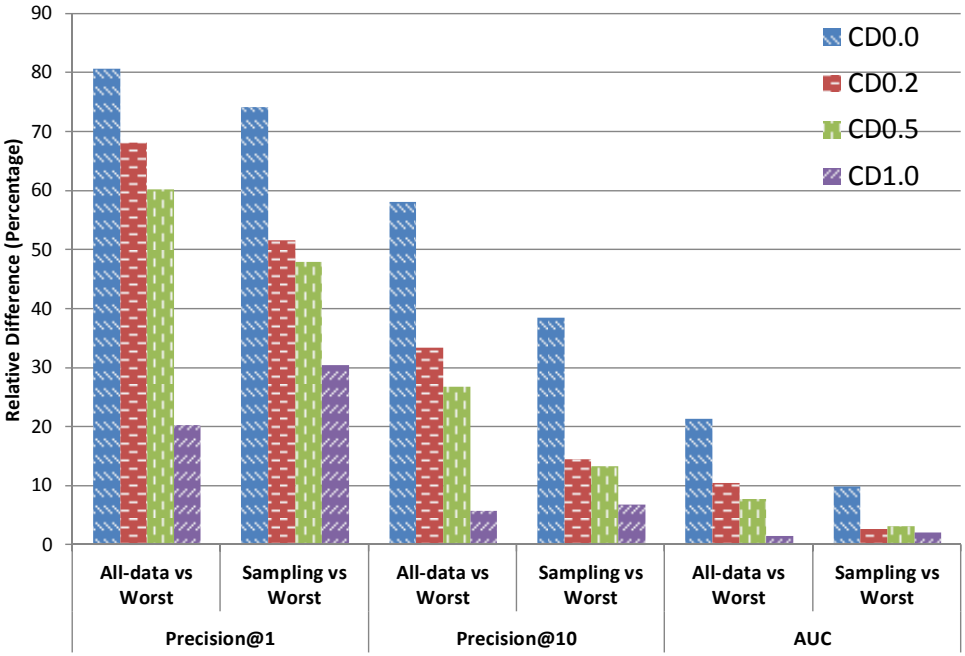


Figure 6.5: Relative difference between passive learning (using all the data) and best active sampling with the worst performance for the sampling techniques indicating the spread of performance

			CD0.0	CD0.2	CD0.5	CD1.0
20news	Precision@1	pos=10%	0	0	-1	0
		pos=20%	0	0	0	0
	Precision@10	pos=10%	-1	-1	-1	0
		pos=20%	-1	-1	-1	0
	AUC	pos=10%	-1	-1	-1	0
		pos=20%	-1	-1	-1	0
RCV1	Precision@1	pos=10%	-1	-1	-1	0
		pos=20%	-1	-1	-1	0
	Precision@10	pos=10%	-1	-1	-1	1
		pos=20%	-1	-1	-1	1
	AUC	pos=10%	-1	-1	-1	1
		pos=20%	-1	-1	-1	1

Figure 6.6: Statistical significance comparison between models with 10 actively sampled examples versus 100 randomly selected samples for FCD. '1'indicates that actively sampling 10 examples is significantly better than randomly sampling 100 examples and '-1' indicates vice versa. '0' indicates no statistical difference.

			Drift=Low	Drift=High
20news	Precision@1	pos=10%	0	0
		pos=20%	0	0
	Precision@10	pos=10%	-1	-1
		pos=20%	-1	-1
	AUC	pos=10%	-1	-1
		pos=20%	-1	-1
KDD	Precision@1	pos=1%	-1	-1
		pos=2%	-1	-1
	Precision@10	pos=1%	-1	0
		pos=2%	-1	-1
	AUC	pos=1%	-1	0
		pos=2%	-1	-1

Figure 6.7: Statistical significance comparison between models with 10 actively sampled examples versus 100 randomly selected samples for SSD. '1'indicates that actively sampling 10 examples is significantly better than randomly sampling 100 examples and '-1' indicates vice versa. '0' indicates no statistical difference.

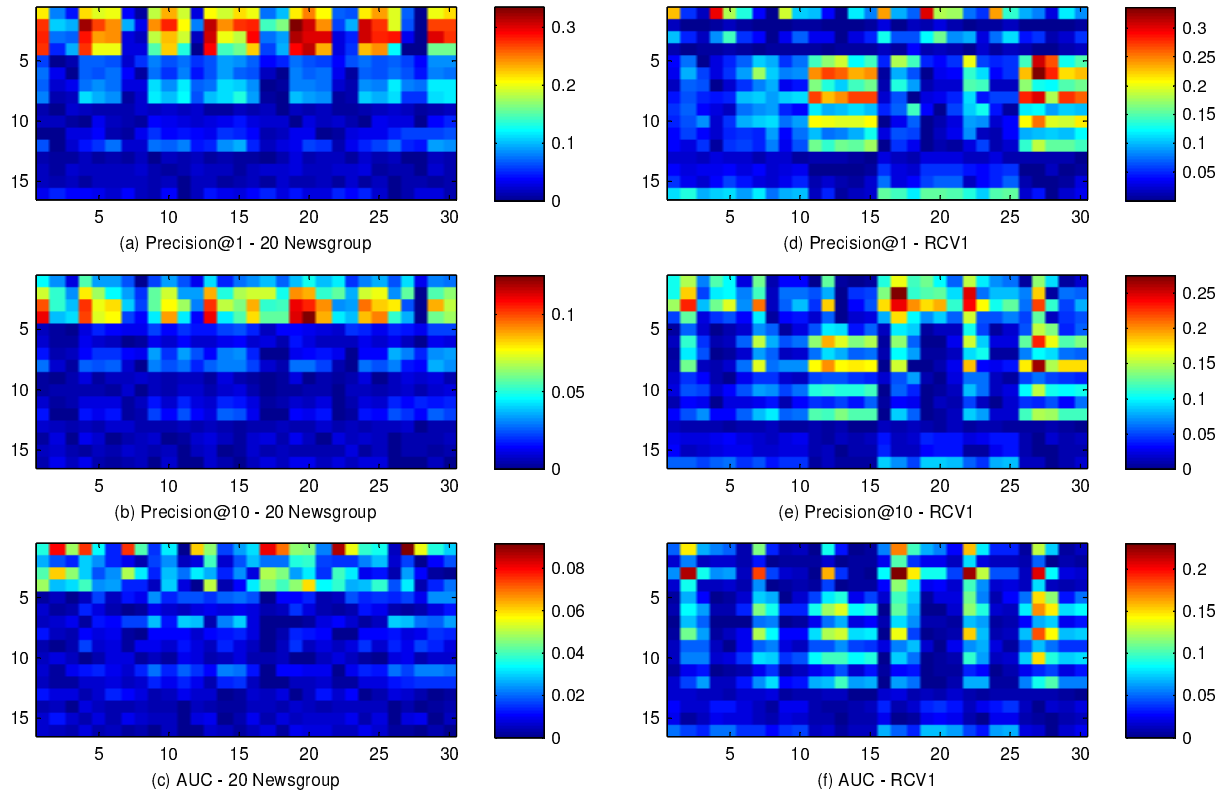


Figure 6.8: Heatmap showing the relative performance of the various experimental setups for the different data settings for FCD. Figure 6.10 shows the indexing scheme for the heatmap. Please note that this is a color image.

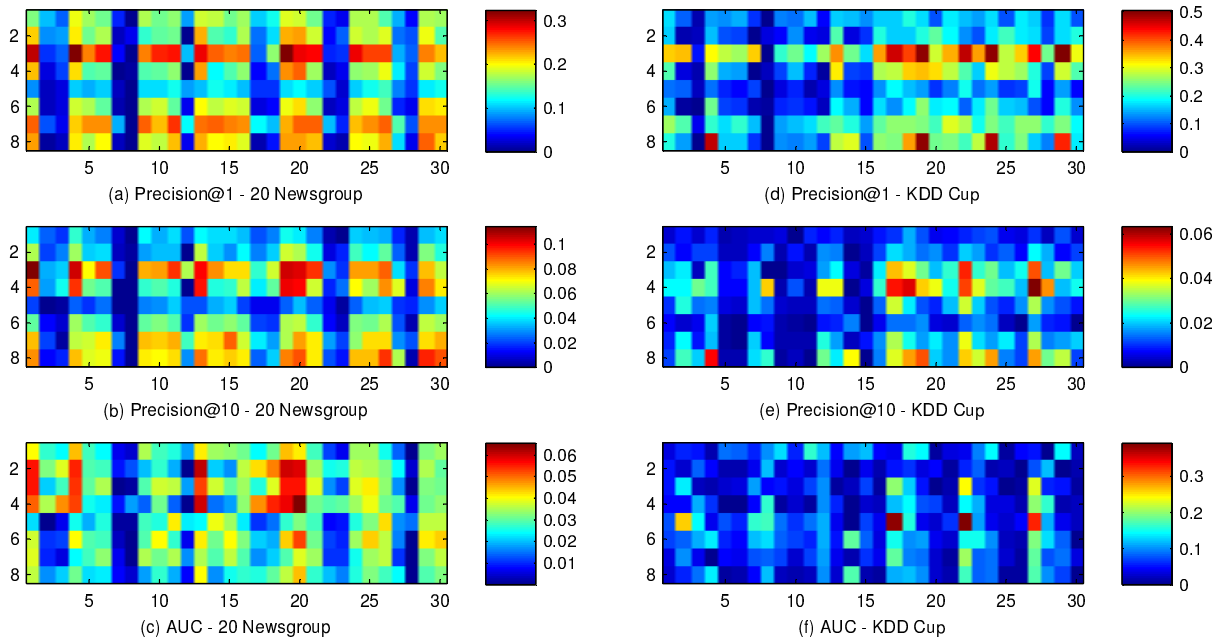


Figure 6.9: Heatmap showing the relative performance of the various experimental setups for the different data settings for SSD. Figure 6.11 shows the indexing scheme for the heatmap. Please note that this is a color image

				model type=single															model type=ensemble													
				weighing scheme=exp					weighing scheme=linear					weighing scheme=uniform					weighing scheme=exp					weighing scheme=linear					weighing scheme=uniform				
				rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl
			Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
No Drift	pos=10%	Queries=10	1	0.23	0.36	0.39	0.22	0.24	0.24	0.40	0.43	0.25	0.21	0.25	0.41	0.23	0.25	0.22	0.24	0.36	0.34	0.23	0.19	0.23	0.37	0.37	0.23	0.20	0.24	0.37	0.44	0.24	0.22
		Queries=100	2	0.31	0.52	0.53	0.31	0.36	0.35	0.57	0.57	0.41	0.37	0.40	0.58	0.29	0.39	0.41	0.30	0.52	0.50	0.28	0.29	0.29	0.56	0.55	0.29	0.31	0.31	0.56	0.59	0.29	0.32
	pos=20%	Queries=10	3	0.32	0.50	0.50	0.28	0.33	0.36	0.51	0.57	0.36	0.32	0.39	0.52	0.29	0.36	0.33	0.32	0.46	0.44	0.27	0.30	0.36	0.54	0.50	0.29	0.33	0.41	0.54	0.60	0.33	0.33
		Queries=100	4	0.46	0.68	0.63	0.45	0.53	0.53	0.70	0.70	0.58	0.51	0.60	0.73	0.44	0.59	0.57	0.51	0.67	0.63	0.44	0.42	0.50	0.70	0.68	0.50	0.48	0.60	0.72	0.73	0.57	0.57
Drift 20%	pos=10%	Queries=10	5	0.13	0.19	0.17	0.12	0.12	0.12	0.16	0.13	0.11	0.12	0.11	0.13	0.12	0.09	0.10	0.12	0.15	0.14	0.11	0.12	0.13	0.15	0.12	0.11	0.12	0.12	0.13	0.15	0.10	0.10
		Queries=100	6	0.20	0.24	0.23	0.19	0.18	0.19	0.24	0.23	0.15	0.18	0.15	0.19	0.19	0.15	0.15	0.18	0.26	0.25	0.18	0.18	0.19	0.24	0.25	0.18	0.18	0.15	0.18	0.19	0.15	0.15
	pos=20%	Queries=10	7	0.21	0.25	0.29	0.21	0.22	0.21	0.25	0.28	0.21	0.23	0.20	0.24	0.22	0.20	0.22	0.22	0.27	0.29	0.21	0.22	0.21	0.25	0.28	0.22	0.22	0.21	0.22	0.24	0.21	0.21
		Queries=100	8	0.30	0.37	0.37	0.27	0.29	0.30	0.38	0.36	0.27	0.29	0.26	0.32	0.29	0.27	0.26	0.29	0.38	0.39	0.28	0.30	0.30	0.38	0.38	0.28	0.29	0.27	0.33	0.33	0.27	0.27
Drift 50%	pos=10%	Queries=10	9	0.12	0.13	0.10	0.11	0.11	0.11	0.11	0.10	0.11	0.10	0.11	0.10	0.12	0.12	0.11	0.11	0.11	0.13	0.11	0.11	0.11	0.12	0.12	0.12	0.11	0.11	0.11	0.12	0.11	0.11
		Queries=100	10	0.17	0.17	0.14	0.14	0.14	0.16	0.17	0.14	0.12	0.14	0.13	0.13	0.14	0.12	0.12	0.13	0.16	0.15	0.14	0.14	0.17	0.16	0.14	0.14	0.14	0.12	0.12	0.11	0.11	0.11
	pos=20%	Queries=10	11	0.23	0.23	0.24	0.25	0.21	0.23	0.23	0.24	0.20	0.21	0.22	0.22	0.22	0.21	0.20	0.22	0.22	0.22	0.24	0.22	0.23	0.22	0.23	0.24	0.21	0.22	0.22	0.24	0.21	0.20
		Queries=100	12	0.24	0.30	0.28	0.26	0.25	0.23	0.28	0.28	0.23	0.23	0.22	0.26	0.25	0.22	0.20	0.25	0.29	0.27	0.27	0.24	0.24	0.29	0.27	0.25	0.24	0.23	0.25	0.25	0.21	0.21
Drift 100%	pos=10%	Queries=10	13	0.09	0.10	0.10	0.09	0.10	0.09	0.10	0.09	0.10	0.09	0.09	0.09	0.10	0.09	0.10	0.10	0.11	0.10	0.08	0.09	0.09	0.09	0.10	0.08	0.09	0.09	0.11	0.10	0.09	0.09
		Queries=100	14	0.09	0.10	0.09	0.10	0.09	0.10	0.10	0.10	0.10	0.10	0.11	0.09	0.09	0.10	0.10	0.09	0.11	0.09	0.10	0.10	0.09	0.10	0.09	0.10	0.10	0.11	0.10	0.10	0.10	0.10
	pos=20%	Queries=10	15	0.20	0.21	0.20	0.20	0.20	0.20	0.20	0.20	0.21	0.21	0.20	0.19	0.20	0.20	0.21	0.20	0.20	0.20	0.19	0.20	0.20	0.20	0.19	0.19	0.21	0.20	0.20	0.22	0.19	0.21
		Queries=100	16	0.18	0.17	0.18	0.18	0.19	0.20	0.18	0.20	0.21	0.18	0.22	0.20	0.18	0.18	0.19	0.18	0.17	0.19	0.17	0.19	0.19	0.17	0.18	0.17	0.19	0.22	0.20	0.19	0.19	0.20

Figure 6.10: Table indicating the indexing scheme for the Heatmaps in figure 6.8. The data shown in the table corresponds to figure 6.8(a) - Precision@1 for 20 Newsgroup dataset. The abbreviated naming convention for the active learning strategies are: rand - random; cer - certainty; uncer - uncertainty; den - density; outl - density outlier;

				weighing scheme=exp					weighing scheme=linear					weighing scheme=uniform					weighing scheme=exp					weighing scheme=linear					weighing scheme=uniform				
				rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl	rand	cer	uncer	den	outl
			Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Drift=Low	pos=10%	Queries=10	1	0.18	0.24	0.27	0.14	0.17	0.19	0.29	0.34	0.16	0.19	0.17	0.25	0.16	0.16	0.19	0.17	0.24	0.21	0.15	0.15	0.18	0.29	0.31	0.16	0.17	0.20	0.25	0.30	0.17	0.17
		Queries=100	2	0.25	0.39	0.38	0.26	0.31	0.29	0.45	0.43	0.33	0.32	0.32	0.47	0.24	0.32	0.33	0.29	0.39	0.36	0.24	0.24	0.28	0.43	0.40	0.28	0.29	0.28	0.40	0.41	0.27	0.32
	pos=20%	Queries=10	3	0.31	0.55	0.54	0.29	0.37	0.33	0.57	0.61	0.37	0.34	0.33	0.51	0.32	0.35	0.37	0.36	0.54	0.44	0.28	0.32	0.33	0.56	0.55	0.32	0.35	0.35	0.51	0.54	0.37	0.39
		Queries=100	4	0.53	0.72	0.69	0.54	0.60	0.60	0.75	0.74	0.60	0.60	0.59	0.75	0.52	0.63	0.61	0.58	0.71	0.68	0.51	0.50	0.58	0.73	0.71	0.59	0.58	0.55	0.63	0.69	0.56	0.59
Drift=High	pos=10%	Queries=10	5	0.16	0.22	0.21	0.13	0.14	0.13	0.22	0.24	0.13	0.13	0.11	0.15	0.14	0.11	0.12	0.14	0.19	0.19	0.13	0.12	0.13	0.18	0.22	0.13	0.12	0.10	0.16	0.19	0.12	0.12
		Queries=100	6	0.25	0.41	0.40	0.24	0.25	0.27	0.41	0.42	0.25	0.24	0.28	0.39	0.22	0.24	0.23	0.24	0.40	0.39	0.23	0.21	0.26	0.40	0.40	0.25	0.23	0.27	0.38	0.39	0.21	0.22
	pos=20%	Queries=10	7	0.29	0.47	0.50	0.32	0.30	0.30	0.44	0.54	0.29	0.32	0.27	0.42	0.29	0.29	0.30	0.30	0.42	0.44	0.31	0.29	0.29	0.44	0.49	0.32	0.30	0.28	0.39	0.47	0.30	0.30
		Queries=100	8	0.48	0.70	0.69	0.50	0.53	0.50	0.69	0.71	0.52	0.53	0.48	0.67	0.48	0.51	0.50	0.52	0.69	0.66	0.48	0.47	0.49	0.69	0.69	0.51	0.51	0.48	0.60	0.67	0.45	0.49

Figure 6.11: Table indicating the indexing scheme for the Heatmaps in figure 6.9. The data shown in the table corresponds to figure 6.9(a) - Precision@1 for 20 Newsgroup dataset. The abbreviated naming convention for the active learning strategies are: rand - random; cer - certainty; uncer - uncertainty; den - density; outl - density outlier;

6.5 Discussion and Future Work

One of the interesting findings from the results is that random sampling is competitive with the best active sampling strategy in many domain settings. Also, there is no universally good strategy thus making the choice of learning strategies dependent on the domain. These results provide crucial insights indicating that the optimal design choices for interactive systems need to consider broader domain parameters rather than adopting a "best practice" strategy. Ideally, we would be interested in a learning approach that performs consistently well if certain domain parameters are known in advance. Furthermore, in real world domains, it may be difficult to know these parameters so the even more valuable learning approach would be one that is robust under *any* domain setting and data characteristics.

6.6 Conclusion

Motivated by the broad question of examining the utility of existing active learning techniques in temporal drift settings, we parameterized real-world domains of information filtering and network intrusion detection, using the expected type and amount of drift, the target class distribution, evaluation metric of interest, and the amount of available labeled data. We generated sub-problems by varying the domain parameters that result in a total of 144 problem setups and compared several active learning algorithms combined with some approaches developed in temporal drift community (single model with instance weighting and weighted ensembles).

To answer the basic question of whether random sampling is better than more intelligent sampling in the presence of drift, we found that active sampling performs statistically better than random sampling in nearly half the problem setups with drift. Although this observation is specific to the choice of metrics and drift, but what seems generalizable is that there is utility of active learning in some drift scenarios and that it is worth further studying active learning's application in temporal drift settings.

We found that the techniques developed in temporal drift literature namely instance weighting and weighted ensembles combined with active learning gave better results. Some intuitive patterns that were observed were: a) ensemble models are better for FCD whereas building a single model (with instance weighting) is better for SSD b) exponential weighting scheme is better for FCD whereas linear weighting scheme is more effective for SSD. We note that there is a need for completely new active learning strategies that are robust to different types of temporal drift and evaluation metrics. We share the data and the sampling scheme for generating FCD and SSD datasets at `https://sites.google.com/site/mohitkum/data`.

Chapter 7

Cost Sensitive Exploitation

7.1 Online Cost-Sensitive Learning

When we did a pilot deployment of the machine learning system for Claims Error Prevention at a large US health insurance company, we got encouraging results but also noticed several problems that affected the efficiency of the auditors when interacting with our rework prediction system..

Context Switching Costs: We used the confidence of the SVM classifier to rank the test examples. The auditors were then provided with the ranked list of claims and started from the top of that list and worked their way down. One problem they experienced was high switching costs when moving from example to example. This switching cost arose from the fact that the auditors spent considerable time (up to 20 minutes) reviewing a claim to figure out whether it was a positive or a negative example. Once they had investigated (and labeled) that claim, they were given the next claim (from the ranked list using the SVM confidence scores) that was often completely different (in terms of the *reason* for flagging) from the previous one. At a later stage, they would be given a claim that was lower down in the ranked list but was scored highly by the classifier for *similar* reasons as an earlier claim. Unfortunately, by that time, they have already gone through enough different claims that they need to redo the investigation and the time it takes them to label it is the same as the earlier ones. We hypothesize that if we can group similar claims together, it would reduce the *context switching* and help make the reviewing times shorter.

Low Precision when reviewing Top 2% scored claims: Based on the class distribution, claim volumes, auditor workload, and audit times, we determined that it's optimal to manually review top 2% of the claims that are scored by our classifier, making precision at 2% the metric we optimize. The scores assigned by the SVM to the top few percent of the claims are fairly similar which results in poor ranking at the top of this list. Also, based on the feedback we got from the auditors, we found that as they went down the list reviewing the claims, they encountered several series of consecutive claims that were all negative examples (and often for similar reasons). We hypothesize that this was because the ranking was purely based on SVM scores and did not have any notion of diversity. Again, grouping similar claims together could help group a lot of these negative examples together and improve the performance of the human auditors.

7.1.1 Interactive Claims Prioritization using Online Cost-Sensitive Learning

The issues described above caused us to think about approaches that could take the results of the classifier and process them in order to improve the overall ROI of the system. That formed the basis of our Claims Prioritization strategy that uses online cost-sensitive learning in order to

maximize the efficiency of the auditors while helping them achieve their goal of finding errors in insurance claims. The efficiency of the auditors is not just a function of the accuracy of the system but also the time it takes them to review a claim. Reducing the time it takes to review an example (claim) is an understudied area which is not often discussed in KDD research literature and is important when dealing with interactive business applications where the experts are expensive and have limited time. We describe our online cost-sensitive learning framework that focuses on reducing the overall time experts spend on labeling (verifying or correcting) examples provided to them by a classifier. We deal with the scenario where a batch classifier has been trained to predict errors in claims and focus on optimizing the interaction between the classifier and the auditors who are consuming the results of this predictor. The goal is to make these auditors more efficient and effective in performing their task as well as eventually improving the system over time.

More-Like-This Strategy

Algorithm 2 Online More-Like-This Algorithm

Require: a labeled set of claims$L$ and an unlabeled set U

1. Train classifier C on L
2. Label U using classifier C
3. Select the top m% scored unlabeled claims U_T (we use m=10 in our case)
4. Cluster the examples $U_T \bigcup L$ into k clusters
5. Rank the k clusters using a cluster-based ranking function (described in the next section)
6. For each cluster in the ranked cluster list
   (a) Rank the claims within the cluster based on their informativeness of the quality of the cluster
   (b) For each claim in the ranked list within the cluster
      i. Query human auditor for the label of the claim
      ii. Move to the next cluster if the evaluation metric (we use precision in our experiments) calculated over claims labeled so far in the cluster falls below threshold P

Our *More-Like-This* (MLT) strategy (Algorithm 2) is motivated by the context switching costs, low precision, and explanation issues described above. The goal of this strategy is to group the scored claims into clusters that contain claims that have been scored highly for *similar* underlying reasons. This allows us to provide the auditors with a cluster of claims that are both highly likely to be *rework* (errors) and can also be audited very quickly with small incremental audit costs. The cost gets reduced because once the auditors have invested the effort in reviewing the first couple of claims in a cluster, the rest of the claims will have similar labels due to similar reasons. At the same time, this technique also allows us to disregard a large number of claims that are similar to each other and are in fact correct claims (negative examples) but have been incorrectly ranked highly by the classifier. This is done by our online strategy where we rank the clusters using a variety of metrics (described later) and give the auditors a few claims from a cluster. If the claims are verified by the auditors as being errors, we continue giving them claims from that cluster. Otherwise, we move on the next cluster thus avoiding a large number of highly scored claims that were in fact negative. This strategy also implicitly provides the auditors with an

explanation for the classification by providing them with other claims that are similar to what they have just labeled. MLT works as a post-processing step on new (unlabeled) claims that have already been scored by the trained classifier.

The intuition behind our MLT approach is to use a supervised classifier like SVM to initially provide a rough partitioning of the scored examples and then use a clustering algorithm to do more finer-grained partitions. The function of the clustering is to take the examples that are scored highly by the classifier and group similar ones together, hopefully creating clusters that correspond to claims that have similar underlying *reasons* for being errors. Our two-step approach of post-ranking (or classification) clustering is also potentially useful in cases where there are different underlying subgroups (or reasons) in the positive class that may be disjoint which could create issues for a binary classifier (unless multiclass labels are available).

In Step 3, we use m=10% for our experiments which selects the top 10% of the claims. Setting m=100% would be equivalent to clustering all the examples without taking into account the classifier scores. Intuitively, we would choose m based on the class priors and the budget available for reviewing the claims.

In Step 4, we vary k (number of clusters) from 200 to 2800 and settle for $U/100$. The higher the number of clusters, the more coherent/similar each group will be which can decrease the audit time and provide better explanations but increases the number of clusters to examine. Another trade-off is that the larger the number of clusters the test examples may be split in different clusters thus not resulting in reduction of context-switch cost. We use Cluto [Karypis, 2002] to perform the clustering.

In Step 5, we experiment with a variety of metrics to rank the clusters described in the next section. These ranking metrics are a function of the number of positive examples (from L in the cluster), the mean score assigned by the classifier to the unlabeled examples U in this cluster, the ratio of the positive to negative labeled examples (from L), the inter-cluster similarity score of the cluster, and the intra-cluster similarity of the cluster.

To rank the claims for review within a cluster (in Step 6a), we use the score assigned by the classifier as the metric in our experiments. This is just one example of a metric that can be used. Other metrics we plan to experiment with include distance from the centroid of the cluster (to select prototypical examples from a cluster first), and sample selection metrics from active learning literature (based on uncertainty sampling for example). In Step 6b(ii), we use precision as the metric and set the threshold to 30%. This is chosen empirically based on the baseline precision scores and the class distribution.

7.2 Experimental Results

7.2.1 Data and Experimental Setup

We present results from two pilots with major US health insurance companies that were conducted using the system. The first pilot was conducted starting in Feb 2010 and the second one started in November 2010. From the first company, we started with approximately 23 million claims spanning 2 years. Of these 23 million claims, 379,000 had been manually labeled through various audit processes. In this labeled data set, around 65% claims were Rework (errors) and the rest were Correct. It is important to remember that this distribution does not hold in the operational world since the labeled data has a skewed class distribution (described in section 4.1). For the second company, we got approximately 40 million claims spanning 12 months. Of these 40 million claims, 150000 had been manually audited and found to be either Rework or not.

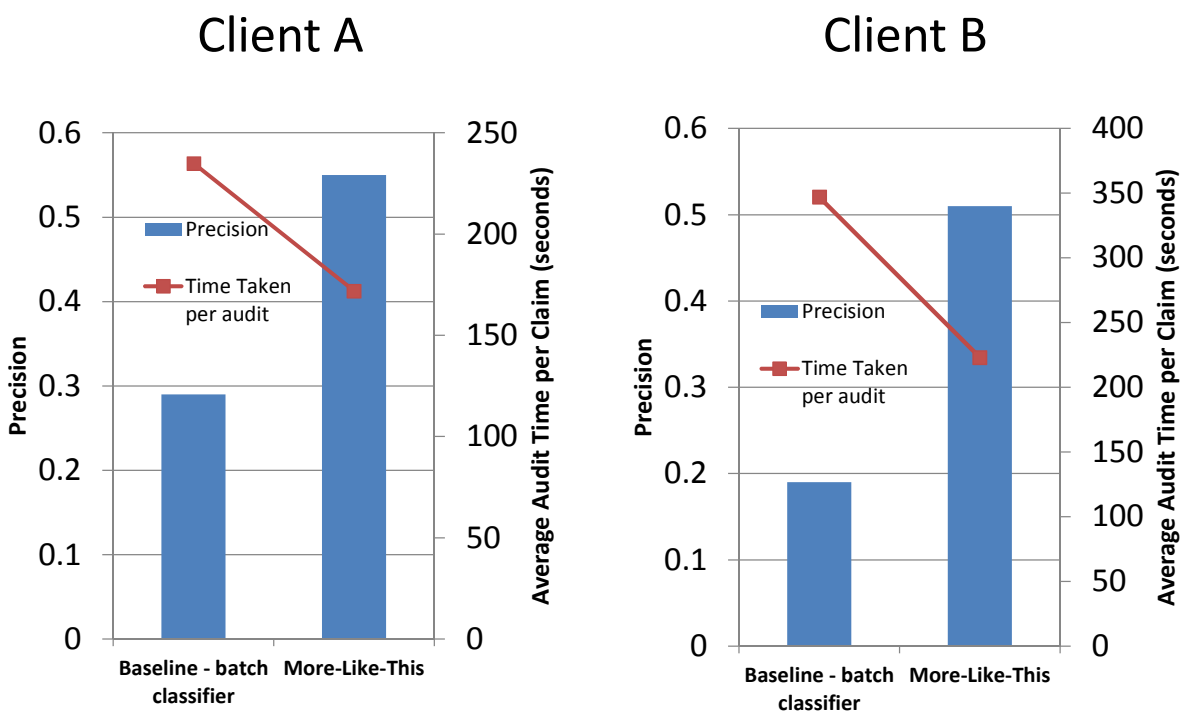


Figure 7.1: Observed precision for the live deployment across two clients of More-Like-This and Baseline system. Secondary axis shows the average audit time per claim.

In this set, around 80% claims were Rework and the rest Correct. Since SVMs are not able to handle categorical data we create binary features from the categorical features resulting in around 238,500 and 215,000 features respectively for the two cases.

We ran two pilots with large US health insurance companies to evaluate the prioritization component described here. The pilots involved a team of auditors from the insurance companies where we gave the auditors a set of claims to audit based on system's recommendations and the different strategies described in the previous section. We will not describe the entire pilot but just a part of the larger evaluation that motivated and verified the More-Like-This strategy. We give detailed results with a second set of offline experiments that were designed to analyze our strategies in more detail.

7.2.2 Live System Deployment

We compared the improvement due to the interactive prioritization strategies in our pilots, figure 7.1. For the first pilot, the baseline system precision was 29% where the auditors audited 200 claims and found 58 claims to be Rework. With More-Like-This strategy, we obtained a hit rate of 55% where the auditors audited 307 claims and found 169 claims to be Rework. Thus we got a 90% improvement over the baseline system. The average audit time per claim was reduced from 3 min 55 sec for the baseline system to 2 min 52 sec for More-Like-This strategy. The reduction in time for auditing as well as the improvement in precision is statistically significant with $p << 0.001$ for Student's T-Test. We also ran further experiments over a period of 3 weeks with two auditors to validate some of our results.

For the second pilot, we obtained a baseline system precision of 19% where the auditors audited 221 claims and found 42 to be Rework. With More-Like-This strategy, we achieved a hit rate of 51% where 1119 claims were audited and 573 were found to be Rework. This represents a 170% relative improvement over the baseline strategy. The average time to audit was reduced from 5 min 47 sec for the baseline system to 3 min 43 sec for More-Like-This strategy (36% relative reduction in time). As observed for the first pilot, the reduction in time for auditing as well as improvement in precision is statistically significant with $p << 0.001$ for Students T-Test.

7.2.3 Offline experiments

Due to the high cost and limited availability of the auditors, we limited the live audits using only the strategies that we thought would work the best. To conduct further in-depth analysis of different strategies, we ran a large number of offline experiments. These experiments were performed 5-fold cross-validation on a claims data from last 5 months with 70,179 claims having 52% positive claims.[1]. Since the distribution of the positive class is skewed in historical data we create the test cases in each fold with 5% positive ratio as follows. We select all the negative examples belonging to the fold and then exclusively sample from the remaining examples so that the test data has 5% positive examples. We keep creating test cases until we exhaust all the positive examples in the fold. Thus each test fold has same negative examples but unique positive examples. In our case, we got 4 folds with 20 test cases and 1 fold with 19 test cases with the desirable 5% positive class ratio.

Metrics

The company we worked with for the experiments reported here, the estimated capacity for auditing the claims is about 760 claims per day (379K claims in 23 months) whereas the volume of claims that need to be processed daily is approximately 46,000 claims per day (23 million claims in 23 months). The audit capacity is approximately 1.5-2% of the total daily volume of the claims which is the case, in general, for most of the insurance companies.[2]

Standard metrics such as accuracy, precision, or recall are useful in comparing different models and approaches, but not enough to minimize the number of claim errors, which is the eventual metric. Based on the audit capacity and discussions with domain experts, we decided that we need a metric that evaluates performance for the top 2-5% scored claims to maximize benefits in real scenarios. Hence, we use Precision (or hit rate) at 2nd Percentile as our metric. This is similar to the evaluation metric, Precision at top 10 documents, popularly used in the Information Retrieval community. In our experimental framework, we do have the flexibility to modify the metric easily and do model selection based on different metrics. In the results reported below, we show precision recall graphs where appropriate, as well as our precision at top 2% metric when comparing different models.

[1]Since we wanted to run multiple detailed experiments, we had to select a smaller dataset for practical runtime considerations

[2]Although the historical data that we have is for post payment auditing, we are using similar numbers to estimate the capacity for pre-payment auditing. Since the audit capacity is typically related to the business value of doing the audit, the capacity may increase in prepayment scenario as it is more valuable to prevent the error from happening than post payment recovery.

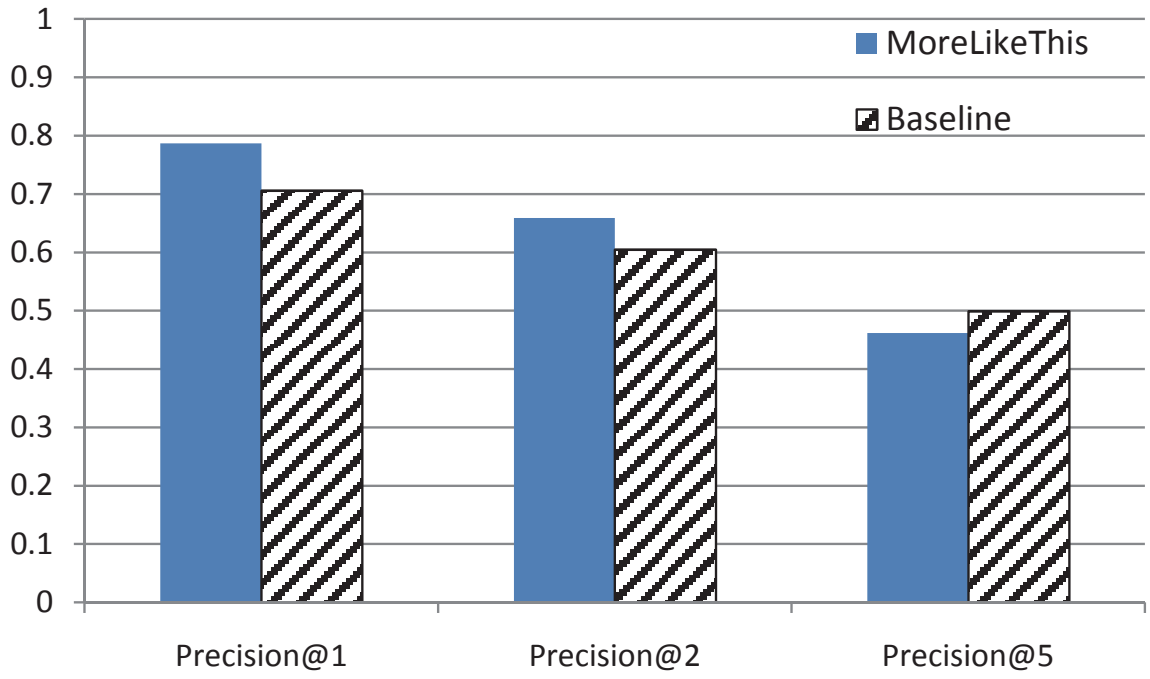


Figure 7.2: More Like This

How well does More-Like-This work?

We use the More-Like-This strategy on top of the baseline system. We experimented with varying the number of clusters (k ranging from 200 to 2800), the clustering algorithms (Repeated Bisections, Repeated Bisections with global optimization and direct k-way clustering), and different cluster ranking metrics. We used Cluto [Karypis, 2002] for the clustering.

The best performance was achieved with a cluster size of 500 with repeated bisection clustering algorithm. The ranking criterion for the clusters was ratio of training Rework to Correct claims in a cluster followed by mean prediction score for test claims. Figure 7.2 shows the performance improvement over the baseline system. We obtain 9% relative improvement for Precision at 2nd percentile, which is the metric that we want to optimize. The improvement is statistically significant with $p << 0.001$ for paired, two-tailed Student's T-test.

Chapter 8

Summary

8.1 Thesis Contributions

In summary, the thesis makes the following contributions:

1. Characterization of the interactive classification problem for practical machine learning systems: We describe the characteristics of the problem of interactive classification and provide a factorization to solve it in terms of cost, exploitation and exploration model. We provide a broad categorization of the various possible cost/exploitaiton/exploration model which can each have three variations, namely, Uniform, Variable and Markovian. Uniform assumes fixed value for each example, Variable assumes that the value for each example is dependent on some properties of the example (and environment like GUI) which can be pre-determined and Markovian assumes that the value for each example is dependent on the (ordered) history of the examples that have already been reviewed in addition to the factors determining the value in the Variable case.
2. Extension of active learning framework to tackle the interactive classification problem: We provide an interactive learning framework that constitutes the choice of cost/exploration/exploitation model, utility metric and optimization algorithm. This framework generalizes the traditional active learning setup to handle differential utility of an example, dynamic cost of labeling an example and temporal drift.
3. Methodology for evaluating interactive classification systems: We provide a novel experimental setup with periodically updating unlabeled pool for evaluating temporal active learning and the interactive framework. We evaluate the framework on three tasks of health insurance claims error prediction, chemo-informatics task of predicting which compound interact with AIDS virus and Information Filtering task using 20 Newsgroup and Reuters-RCV1 datasets.

Bibliography

Abhinav Anand and Dmitriy Khots. A data mining framework for identifying claim overpayments for the health insurance industry. In *Proceedings of INFORMS Workshop on Data Mining and Health Informatics*, 2008.

Shilpa Arora. Minimizing the costs in generalized interactive annotation learning. PhD Thesis CMU-LTI-12-013, Language Technologies Institute, Carnegie Mellon University, 2012.

Shilpa Arora, Eric Nyberg, and Carolyn P. Rosé. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of NAACL-HLT Workshop on Active Learning for Natural Language Processing*, 2009.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, April 2009.

Yoram Baram, Ran El-Yaniv, Kobi Luz, and Manfred Warmuth. Online choice of active learning algorithms. *Journal of Machine Learning*, 2004.

Ron Bekkerman and Martin Scholz. Data weaving: scaling up the state-of-the-art in data clustering. In *Proceedings of CIKM*, 2008.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of ICML*, 2009.

Michael Bloodgood and Vijay Shanker. 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 HLT-NAACL*, 2009.

Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 2011.

Robert Burbidge, Jem J. Rowland, and Ross D. King. Active learning for regression based on query by committee. In *Proceedings of International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, 2007.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR*, 1998.

Chih Chung Chang and Chih Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Wei Chu, Martin Zinkevich, Lihong Li, Achint Thomas, and Belle Tseng. Unbiased online active learning in data streams. In *Proceedings of KDD*, 2011.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. In

American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems, 1999.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of SIGIR*, 2009.

Chris Curry, Robert L. Grossman, David Locke, Steve Vejcik, and Joseph Bugajski. Detecting changes in large data sets of payment card data: a case study. In *Proceedings of KDD*, 2007.

Pinar Donmez. Proactive learning: Towards learning with multiple imperfect predictors. PhD Thesis CMU-LTI-10-002, Language Technologies Institute, Carnegie Mellon University, 2010.

Pinar Donmez and Jaime G Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. In *Proceedings of European Conference on Information Retrieval(ECIR)*. 2009.

Pinar Donmez, Jaime G. Carbonell, and Paul N. Bennett. Dual strategy active learning. In *Proceedings of ECML*, 2007.

Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer, 2002.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Diana Farrell, Eric Jensen, Bob Kocher, Nick Lovegrove, Fareed Melhem, Lenny Mendonca, and Beth Parish. Accounting for the Cost of US Health Care: A New Look on Why Americans Spend More. 2008. URL `http://www.mckinsey.com/insights/health_systems_and_services/accounting_for_the_cost_of_us_h` Date URL checked: 28th Oct 2013.

Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in knowledge discovery and data mining*. AAAI, 1996.

Zahra Ferdowsi, Rayid Ghani, and Mohit Kumar. An online strategy for safe active learning. In *ICML Workshop on Combining Learning Strategies to Reduce Label Cost*, 2011.

George Forman. Tackling concept drift by temporal inductive transfer. In *Proceedings of SIGIR*, 2006.

Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 2007.

Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Proceedings of NIPS*, 2008.

Yuhong Guo and Russ Greiner. Optimistic active learning using mutual information. In *Proceedings of IJCAI*, 2007.

Isabelle Guyon. Datasets of the active learning challenge. *Journal of Machine Learning Research*, 2011. URL `http://jmlr.org/proceedings/papers/v16/supplemental/Datasets_AL_challenge.pdf`. Date URL checked: 28th Oct 2013.

Isabelle Guyon, Gavin C. Cawley, Gideon Dror, and Vincent Lemaire. Results of the active learning challenge. *Journal of Machine Learning Research*, 2011.

R. Haertel, K. Seppi, E. Ringger, and J. Carroll. Return on investment for active learning. In *NIPS Workshop on Cost-Sensitive Learning*, 2008.

Uri Hanani, Bracha Shapira, and Peretz Shoval. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, August 2001.

Steve Hanneke. Activized learning: Transforming passive to active with improved label complexity. *Journal of Machine Learning Research*, 2012.

Jingrui He and Jaime Carbonell. Nearest-Neighbor-Based Active Learning for Rare Category Detection. In *Proceedings of NIPS*, 2007.

Dorit S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1997.

T. Ryan Hoens, Robi Polikar, and Nitesh V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 2012.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of KDD*, 2001.

Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, October 2002.

Aloak Kapoor and Russell Greiner. Learning and classifying under hard budgets. In *Proceedings of ECML*, 2005.

Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision theoretic active learning. In *Proceedings of IJCAI*, 2007.

G. Karypis. Cluto - a clustering toolkit. Computer sciences technical report, University of Minnesota, 2002.

KDD Cup. Kdd cup 1999. `http://www.sigkdd.org/kddcup/index.php`, 2009. Date URL checked: 28th Oct 2013.

Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Information Processing Letter*, 1999.

Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of ICML*, 2000.

J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal Machine Learning Research*, December 2007.

Andrease Krause. SFO: A toolbox for submodular function optimization. *Journal of Machine Learning Research*, 2010.

Mohit Kumar and Rayid Ghani. Interactive learning for efficiently detecting errors in insurance claims. In *Proceedings of KDD*, 2011.

Mohit Kumar, Rayid Ghani, and Zhu-Song Mei. Data mining to predict and prevent errors in health insurance claims processing. In *Proceedings of KDD*, 2010.

David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, G. Dietterich, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of WWW*, 2010.

Ryan N. Lichtenwalter and Nitesh V. Chawla. Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In *Proceedings of Pacific-Asia International Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2010.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, October 2007.

Michael Lindenbaum, Shaul Markovitch, and Dmitry Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, February 2004.

Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle Tseng. Active learning for ranking through expected loss optimization. In *Proceeding of SIGIR*, 2010.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of IJCAI*, 2005.

Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of CIKM*, 2002.

National Coalition on Health Care. Health care facts: Costs, 2010. URL `http://nchc.org/sites/default/files/resources/FactSheet-Cost.pdf`. Date URL checked: 5th Feb 2010.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 1978.

NIST. Nist engineering statistics handbook, 2013. URL `http://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm`. Date URL checked: Oct 28th, 2013.

Jonathan Nitzan and Shimshon Bichler. *Capital as Power: A Study of Order and Creorder*. Routledge, 2009.

Francisca Nonyelum Ogwueleka. Data mining application in credit card fraud detection system. *Journal of Engineering Science and Technology, School of Engineering, Taylors University*, 2011.

I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: From the laboratory to the commercial world. In *Proceedings of the IEEE*, 2001.

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.

Jason Rennie. 20 newsgroups data set, 2007. URL `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`. Date URL checked: 28th Oct 2013.

Eric Ringger, Marc Carmen, Robbie Haertel, Deryle Lonsdale, James Carroll, and Noel Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of LREC*, 2008.

Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of ICML*, 2001.

M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. In *Management Science*, 2009.

Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, October 2007.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of ICML*, 2000.

Martin Scholz and Ralf Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 2007.

B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *NIPS Workshop on Cost-Sensitive Learning*, 2008.

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

Burr Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 1–18. Microtome Publishing, 2011a.

Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of EMNLP*, 2011b.

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

Mohak Shah. Generalized agreement statistics over fixed group of experts. In *Proceedings of ECML/PKDD*, 2011.

Kevin Small. Interactive learning protocols for natural language applications. PhD thesis, University of Illinois at Urbana-Champaign, 2009.

W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of KDD*, 2001.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. A Bradford Book, March 1998.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of EMNLP/CoNLL*, 2007.

Sudheendra Vijayanarasimhan and Kristen Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *Proceedings of CVPR*, 2009.

Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Active learning with evolving streaming data. In *Proceedings of the ECML/PKDD*, 2011.

Byron C. Wallace, Kevin Small, Carla E. Brodley, Joseph Lau, and Thomas A. Trikalinos. Modeling annotation time to reduce workload in comparative effectiveness reviews. In *Proceedings of ACM International Health Informatics Symposium (IHI)*, 2010.

Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of KDD*, 2003.

Marco Wiering and Martijn van Otterlo. *Reinforcement Learning: State-of-the-art*, volume 12. Springer, 2012.

Jeffrey Wildman. Bron-kerbosch maximal clique finding algorithm, 2011. URL http://www.mathworks.in/matlabcentral/fileexchange/30413-bron-kerbosch-maximal-clique-findi Date URL checked: 28th Oct 2013.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, December 2004.

Jingbo Zhu. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of ACL*, 2007.

Xingquan Zhu, Peng Zhang, Xiaodong Lin, and Yong Shi. Active learning from data streams. In *Proceedings of ICDM*, 2007.

Indre Zliobaite. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010.

Marcela Zuluaga, Andreas Krause, Guillaume Sergent, and Markus Püschel. Active learning for multi-criterion optimization. In *Proceedings of ICML*, 2013.