August 2012

New Paradigms for Active Learning

Ailing Ni
The University of Western Ontario

Supervisor
Charles X. Ling
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

NEW PARADIGMS FOR ACTIVE LEARNING
(Spine title: New Paradigms for Active Learning)
(Thesis format: Monograph)

by

(Eileen) Ai-Ling Ni

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Examiners:

Supervisor:

.....
Dr. Olga Veksler

.....
Dr. Charles. X. Ling

.....
Dr. John Barron

Supervisory Committee:

.....
Dr. Ning Su

.....
Dr. Steve Wang

The thesis by

(Eileen) Ai-Ling Ni

entitled:

New Paradigms for Active Learning

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

.....
Date

.....
Chair of the Thesis Examination Board

Abstract

In traditional active learning, learning algorithms (or learners) mainly focus on the performance of the *final* model built and the total number of queries needed for learning a good model. However, in many real-world applications, active learners have to focus on the learning *process* for achieving *finer* goals, such as minimizing the number of mistakes in predicting unlabeled examples. These learning goals are common and important in real-world applications. For example, in direct marketing, a sales agent (learner) has to focus on the process of selecting customers to approach, and tries to make correct predictions (i.e., fewer mistakes) on the customers who will buy the product.

However, traditional active learning algorithms cannot achieve the finer learning goals due to the different focuses. In this thesis, we study how to control the learning process in active learning such that those goals can be accomplished. According to various learning tasks and goals, we address four new active paradigms as follows.

The first paradigm is *learning actively and conservatively*. Under this paradigm, the learner *actively* selects and predicts the most certain example (thus, *conservatively*) iteratively during the learning process. The goal of this paradigm is to minimize the number of mistakes in predicting unlabeled examples during active learning. Intuitively the conservative strategy is less likely to make mistakes, i.e., more likely to achieve the learning goal. We apply this new learning strategy in an educational software, as well as direct marketing.

The second paradigm is *learning actively and aggressively*. Under this paradigm, unlabeled examples and multiple oracles are available. The learner *actively* selects the best multiple oracles to label the most uncertain example (thus, *aggressively*) iteratively during the learning process. The learning goal is to learn a good model with guaranteed label quality.

The third paradigm is *learning actively with conservative-aggressive tradeoff*. Under this learning paradigm, firstly, unlabeled examples are available and learners are allowed to select examples *actively* to learn. Secondly, to obtain the labels, two actions can be considered: querying oracles and making predictions. Lastly, cost has to be paid for querying oracles or for making wrong predictions. The *tradeoff* between the two actions is necessary for achieving the learning goal: minimizing the total cost for obtaining the labels.

The last paradigm is *learning actively with minimal/maximal effort*. Under this paradigm, the labels of the examples are all provided and learners are allowed to select examples *actively* to learn. The learning goal is to control the learning process by selecting examples actively such that the learning can be accomplished with *minimal effort* or a good model can be built fast with *maximal effort*.

For each of the four learning paradigms, we propose effective learning algorithms accordingly and demonstrate empirically that related learning problems in real applications can be

solved well and the learning goals can be accomplished.

In summary, this thesis focuses on controlling the learning process to achieve fine goals in active learning. According to various real application tasks, we propose four novel learning paradigms, and for each paradigm we propose efficient learning algorithms to solve the learning problems. The experimental results show that our learning algorithms outperform other state-of-the-art learning algorithms.

Keywords: active learning, learning process, minimizing the number of mistakes, guaranteed label quality, labeling cost, learning effort

Acknowledgements

My time at Western has been influenced and guided by a number of people to whom I am deeply indebted. Without their help, friendship and support, this thesis would likely never have been finished.

I would like to thank Dr. Charles Ling, my supervisor, who had the greatest impact in my thesis research in the past four years. He had been a tremendous mentor, collaborator and friend, providing me with invaluable insights about research and academic skills in general. I feel exceedingly privileged to have had his guidance.

I thank the examiners of my thesis proposal for their insights and guidance and the professors who gave me great suggestions and help.

Thanks are also given to members of our Data Mining Lab at Western: Da Kuang, Jun Du, Xiao Li, Yan Luo, David deangelis, Arman Didandeh and Nima Mirbakhsh. They are great colleagues and provided me with great help and a friendly environment to work in.

My deepest gratitude and appreciation is reserved for my parents and brothers and sisters-in-law. Without their constant love, support and encouragement, I would never have been able to produce this thesis. I dedicate this thesis to them.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

In this chapter, we will review some basics of machine learning, supervised learning and traditional active learning first. Then we will state briefly the limitations of the traditional active learning paradigms, and present an overview of our new paradigms of active learning and the contributions of this thesis.

1.1 Machine Learning

Ever since computers were invented, they have been applied to a wide range of tasks by designing and implementing necessary softwares [19, 18]. However, there are many tasks that are difficult or impossible to fulfil by simply programming, as programmers may not be able to anticipate all possible situations and all changes over time or even have no idea on how to program the solution [19, 77]. For example, in speech recognition, it is impossible to map each pronunciation to a word correctly by programming, as the pronunciation of one word varies due to different accents which programmers cannot anticipate. As another example, in detecting credit card fraud, even programmers cannot recognize who is a fraud, thus it is impossible to detect fraud by programming.

Naturally experts have been thinking that if computers can be programmed to **learn** automatically with experience, i.e., making machine learn by itself, the impact would be dramatic. The informal definition of **learning** given by Tom Mitchell [59] is as follows:

> *A computer program is said to* ***learn*** *from experience* E *with respect to some class of tasks* T *and performance measure* P*, if its performance at tasks in* T*, as measured by* P*, improves with experience* E*.*

According to the definition, *machine learning* is to program computers to learn general models from a set of particular examples and optimize their performances.

Generally speaking, the machine learning process has two phases, *learning* and *predicting*, as shown in Figure 1.1. Machine learning researches usually focus on the first phase of learning. As data may be incomplete or come from multiple sources [30], the first step in the phase of learning is preprocessing data, and which is then followed by learning from the data and evaluating the performances of the learned model. In this thesis, we assume that data have

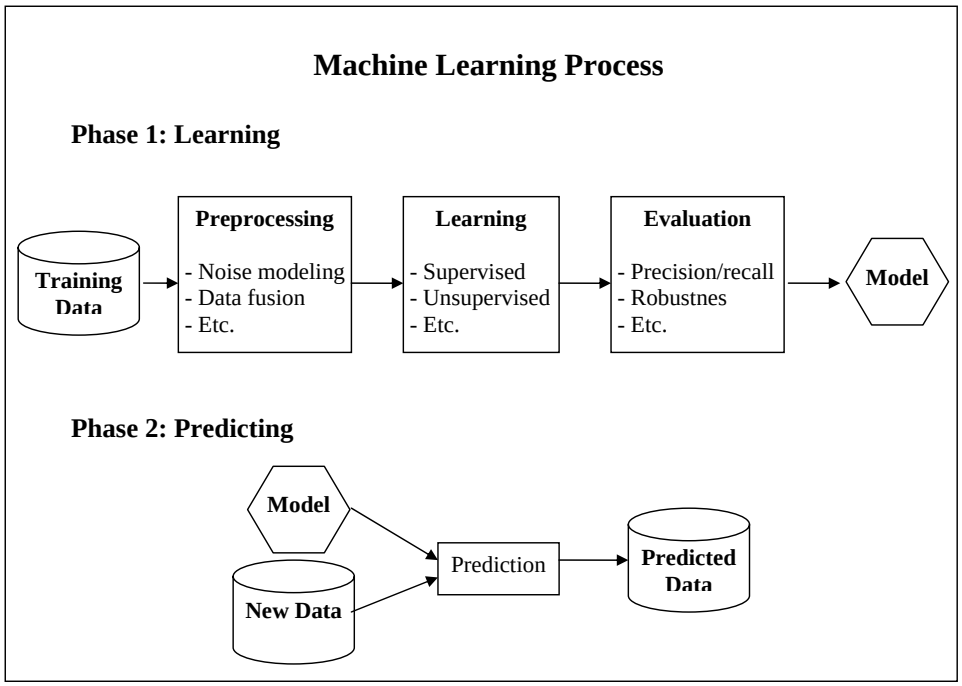


Figure 1.1: Machine learning process.

been preprocessed and we focus on the learning and evaluating steps. The second phase of machine learning is to apply the model learned in the first phase to predicting new data.

Machine learning is clearly meaningful and important, as data are cheap and abundant (data warehouses, data marts), while knowledge (models) is expensive and scarce [11]. According to the important applications on various tasks, machine learning algorithms can be grouped into different types, such as supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning and so on. Among them, supervised learning is an important research area and is useful for numerous applications [48].

1.2 Supervised Learning

Supervised learning is the process of constructing a set of rules, or more generally speaking, creating a model, from examples having both attributes and labels (nominal or numeric). The rules or model is required to achieve minimum error on predicting the labels for future examples drawn independently from the identical distribution (minimum generalization error).

Supervised learning can be illustrated simply with a small set of examples in Table 1.1. The examples have four attributes (*outlook*, *temperature*, *humidity* and *windy*) and one label (*play* = *yes* or *play* = *no*). We can build a model with a classical supervised learning algorithm C4.5 [70](see Figure 1.2 for the model).

The decision tree model in Figure 1.2 can be interpreted with a set of rules as follows.

If outlook = sunny and humidity = high then play = no;
If outlook = sunny and humidity = normal then play = yes;

Table 1.1: A small dataset - weather

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

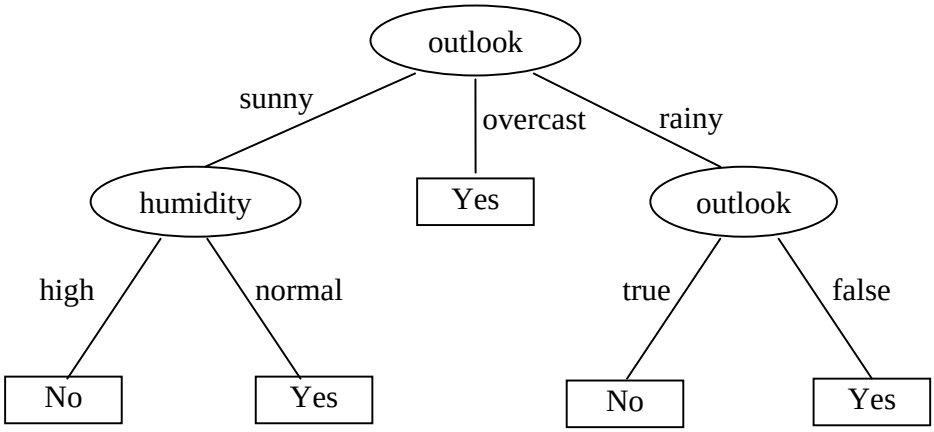


Figure 1.2: One supervised learning model built on the weather data.

If outlook = overcast then play = yes;
If outlook = rainy and windy = true then play = no;
If outlook = rainy and windy = false then play = no.

These rules are meant to be explained in order: the first rule, then if it does not apply then the second, and so on. With the rules, we can predict the label (*play = yes* or *no*) for a future example. For instance, given an example, *outlook = rainy, temperature = normal, humidity = high and windy = true*, we will predict its label as *play = no*.

In the above toy example, we choose to build a decision tree model due to its interpretability. Some other supervised learning algorithms, such as naive Bayes [29], support vector machine (SVM) [98] and neural network [40], can also be applied. Different learning algorithms

have different properties and are suitable for different types of problems. However, in this thesis, we only focus on different learning paradigms and any supervised learning algorithms can easily be used.

1.3 Active Learning

Traditional supervised learning algorithms have been widely and successfully used in many real applications, such as speech recognition, loan applications, webpage categorization and so on [59, 42, 69]. However, to construct a good model, a considerably large amount of labeled data is usually required, which may be difficult to obtain in real applications. For example, in webpage categorization, each webpage in the training set has to be tagged with certain labels, such as economics, politics, entertainment and so on. This tedious work is usually done manually, and costs a considerable amount of time, human resource and money. Under this circumstance, active learning has been proposed and studied in the past decades.

In active learning, unlabeled examples are assumed to be easily available such as the webpages on Internet; while the labels of the examples can only be obtained by paying certain cost, such as by paying an expert to label them. Obviously, we can reduce the cost by minimizing the number of examples to be labeled. Instead of obtaining a complete training data passively by traditional supervised learning, active learning selectively obtains the labels of the examples that are most informative for the current model. In this way the final model may be built with fewer labeled examples. Accordingly the cost for labeling examples can be reduced.

Particularly, in active learning, a learner is usually provided with a small (or even empty) set of training examples, and a model is built accordingly. Then the learner selects the most informative example to query an oracle (an expert) for its label and adds the new labeled example to the training data. By repeating the building-and-querying process, the training set is expanded gradually with informative examples, and the model built from them can be improved quickly (i.e., the generalization error on future examples reduces quickly) compared to the "passive" learning. Consequently the number of labeled examples is reduced, and so isthe labeling cost.

Active learning has been widely and successfully applied to many real applications. In the following subsections, we will introduce the commonly used scenarios, query strategies and variants of active learning.

1.3.1 Active Learning Scenarios

Different scenarios have been considered in the active learning research. Two of them are well-studied. One is membership query synthesis [1], and the other one is pool-based sampling [54].

Under the *membership query synthesis* scenario, usually only a small set of labeled examples is given. Unlabeled examples are all generated synthetically from input space. This scenario is mainly used for theoretical research in active learning. For some real applications, synthetic queries may cause unexpected problems. For example, in handwritten recognition, synthetic symbols for query may not be recognizable and have no meanings.

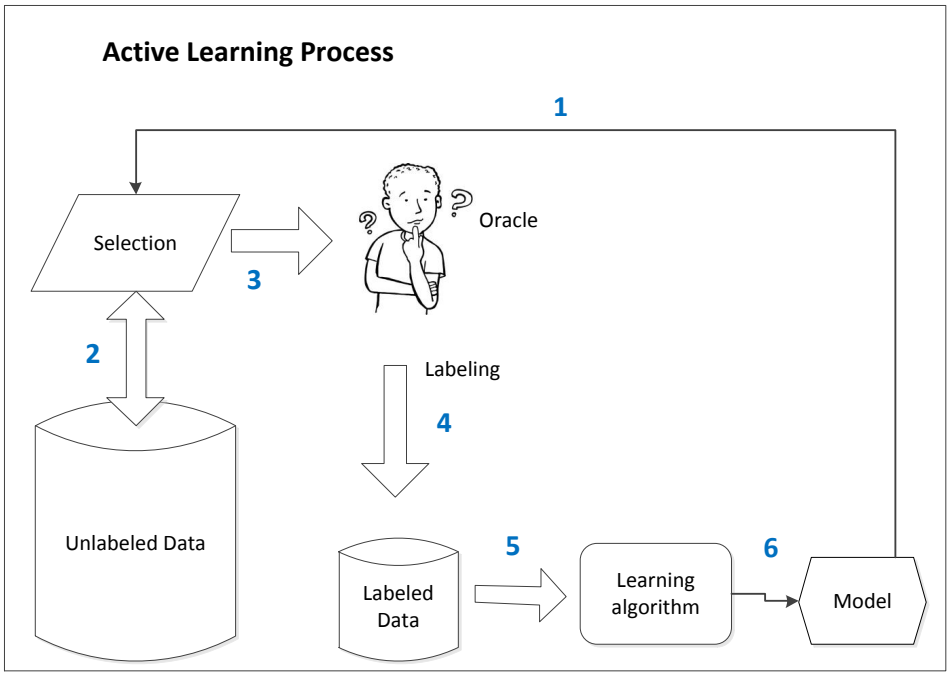


Figure 1.3: Active learning process. The numbers (1-7) indicate the workflow of active learning.

Under the *pool-based sampling* scenario, in addition to a small (or even empty) set of labeled examples, a large set of unlabeled examples (called "pool") is given. Usually the unlabeled examples in the pool are assumed to be collected for free or cheaply, such as the webpages and images from Internet. The active learning algorithms select the most informative example from the pool to query an oracle for its label, and update the current model with all the labeled examples. In this way, a good model is expected to be built with fewer labeled examples. In this thesis, the research is under the pool-based active learning scenario, as it is for solving the problems in real-world applications.

In addition to the two scenarios, *stream-based selective sampling* [14] is also studied in the active learning research. In stream-based selective sampling, unlabeled examples are coming from real data stream (instead of a set of unlabeled examples ("pool") in pool-based sampling). Each unlabeled example can be observed only for once, and learning algorithms have to decide whether to query it or not based on its informativeness.

1.3.2 Active Learning Query Strategies

As mentioned, to reduce the labeled examples needed, active learning algorithms under the pool-based sampling scenario usually select the most informative example from the pool of unlabeled examples to query. How can we evaluate the informativeness of each unlabeled example? Many approaches have been proposed for the evaluation, and here we focus on reviewing three commonly used approaches including *uncertain sampling*, *expected error reduction* and *density-weighted method*.

Uncertain sampling [54] is the simplest and most commonly used query strategy. It selects

the example that is predicted by the current model with the lowest confidence to query an oracle for its label. For a binary classification problem, uncertain sampling selects the example with predictive probability closest to *0.5*. The rationale is straightforward: if the current model can predict the example well, then the example does not have much information to improve the model; otherwise, the example is expected to improve the current model effectively.

Expected error reduction is proposed in [76], and it directly focuses on reducing the generalization error of the model being built. More specifically, for each unlabeled example x_i in a given pool, the learning algorithm estimates its label and builds a model over the combination of x_i and the training examples. Among them, the example that minimizes the generalization error is selected to query an oracle for its label.

The density-weighted method is proposed by Settles and Craven [85]. The rationale behind this method is that, to reduce the generalization error on the entire example space, the examples from the high density space should be predicted with less error rate. Thus, the unlabeled examples from the high density example space should take precedence over other examples to be queried for labels.

1.3.3 Variants of Active Learning

The query strategies mentioned are to select the most informative example to label such that a good model can be built with fewer labeled examples. The success of the strategies are usually subject to certain assumptions, such as noise-free labels given by an oracle, evenly distributed labeling cost and so on. For solving the learning problems in real applications, different variants, such as *active learning with noisy oracles* and *active learning with variable labeling cost*, have been proposed by recent works.

Active learning with noisy oracles is proposed due to the fact that in real applications, noisy labels are ubiquitous. Noise can be introduced to labels by oracles in different ways, such as inaccurate instruments in empirical experiments, distraction or fatigue of experts and so on [83]. The noisy labels affect the active learning performances badly as usually active learning algorithms are noise-prone [3, 97]. To reduce the negative effects, different types of learning strategies have been proposed. One type is focusing on how to select the example that is more informative and less likely to be noisy [3, 25]. The other type is to get rid of the noise in labels by querying multiple oracles [82, 90, 22]. However, label quality still cannot be guaranteed in the previous works. In this thesis, we will study how to remove the noise and guarantee label quality in Chapter 3 with multiple oracles.

Active learning with variable labeling cost tries to solve the problems that the labeling cost is example-dependent or oracle-dependent. For example-dependent labeling cost, two groups of active learning strategies have been proposed. One is that the labeling cost is known before selecting the examples (e.g., [46, 47]). The other group is to handle the problem that the costs for labeling examples are variable and unknown [86]. For oracle-dependent labeling cost, learning strategies have been proposed in [110, 21] to select low-cost combinations of oracles that result in high-accuracy labels of examples. As a result, the total cost in building a good model can be minimized.

More variants of active learning, such as batch-model active learning [13, 44], multi-task active learning [74, 109] and active learning with generalized query [24, 26], have also been proposed. In addition, active learning strategies have also been successfully applied to other

machine learning researches, such as active clustering [43, 7], active transfer learning [91], active feature acquisition and classification [58, 78], active class selection [55] and so on.

1.3.4 Limitations of Traditional Active Learning Paradigms

Active learning has been fairly well-established in theoretical research and various real-world applications. Most of the active learning research has concluded with encouraging results that the number of queries can be reduced by querying oracles actively (e.g., [81, 54, 32, 41]). However, some previous works [80, 39, 86, 4] also show the limitations of active learning on some circumstances, such as alternative query types and so on. This may be due to the many simplified assumptions in previous works [81]. In this thesis we address two types of new and different limitations.

The first limitation is that most traditional active learning paradigms only focus on the number of examples labeled (queries issued) and the prediction accuracy of the final model built. However, they do not study learner's performances during the learning process which in fact is very important. For example, in directed marketing (a process of identifying likely buyers to market products actively), sales agents need to select and approach the "right" customer (the customer who will buy the products). Obviously its final goal is to make fewest mistakes *during* the selection (learning) process, rather than the high prediction accuracy of the final model as in traditional active learning (See Chapter 2 for further discussion). As another example, human often prefers taking minimal effort during the process of learning new knowledge. It is also crucial to minimize the machine learning effort during the learning process as effort is related to energy consumption, system reliability and so on (See Chapter 5 for further discussion). The traditional active learning does not study those goals in the process of learning, as we do in this thesis.

The other limitation is that the label quality in active learning cannot be guaranteed. Most traditional active learning paradigms assume that oracles are always correct. However, noise can be introduced to labels in different ways as mentioned in Section 1.3.3. To rule out the negative effects of the noisy labels, multiple imperfect oracles are used in previous works [82, 90, 22]. By querying multiple oracles for each example, the final label obtained is expected to be more accurate. This multiple-oracle strategy is reasonable and useful in improving label quality. However, there is still no way to guarantee the label quality. We will propose a novel learning paradigm in Chapter 3 to overcome the limitation.

1.4 New Paradigms of Active Learning: an Overview

Due to the limitations of traditional active learning, we propose new learning paradigms for solving real-world problems. According to different settings and goals, four related paradigms are proposed, including *learning actively and conservatively*, *learning actively and aggressively*, *learning actively with conservative-aggressive tradeoff* and *learning actively with minimal/maximal effort*.

1.4.1 Learning Actively and Conservatively

The new paradigm of *learning actively and conservatively* is proposed to study the learning processes that the traditional active learning ignores. More specifically, it focuses on the learning process of selecting unlabeled examples gradually, and tries to reduce the number of mistakes that the learner makes in predicting them. The learning process is important in many real-world applications. Taking the direct marketing as an example again, to make the marketing more efficient, a learner (or an agent) has to focus on the selection process and tries to find and approach the "right" examples (customers) [106]. Accordingly the number of mistakes made during the learning process is reduced.

The proposed paradigm of learning actively and conservatively (see Figure 1.4 for its framework) is applicable for the problems that are of the following settings. First of all, unlabeled examples are provided and learners can actively select examples from them (Steps 1 and 2 in Figure 1.4). This is the same as traditional active learning. In the direct marketing example, all the customers are unlabeled (not knowing who is "buying" and who is "not"), and a sales agent can select the customers to approach. Secondly, the selected unlabeled examples are predicted by the learner itself, and the true label is revealed after each prediction (Steps 4 and 5). In direct marketing, the true label of a customer ("buying" or "not buying") is revealed after a learner (an agent) approaches a predicted buyer. The goal of this paradigm is to minimize the number of mistakes for predicting unlabeled examples during the iterative learning process.

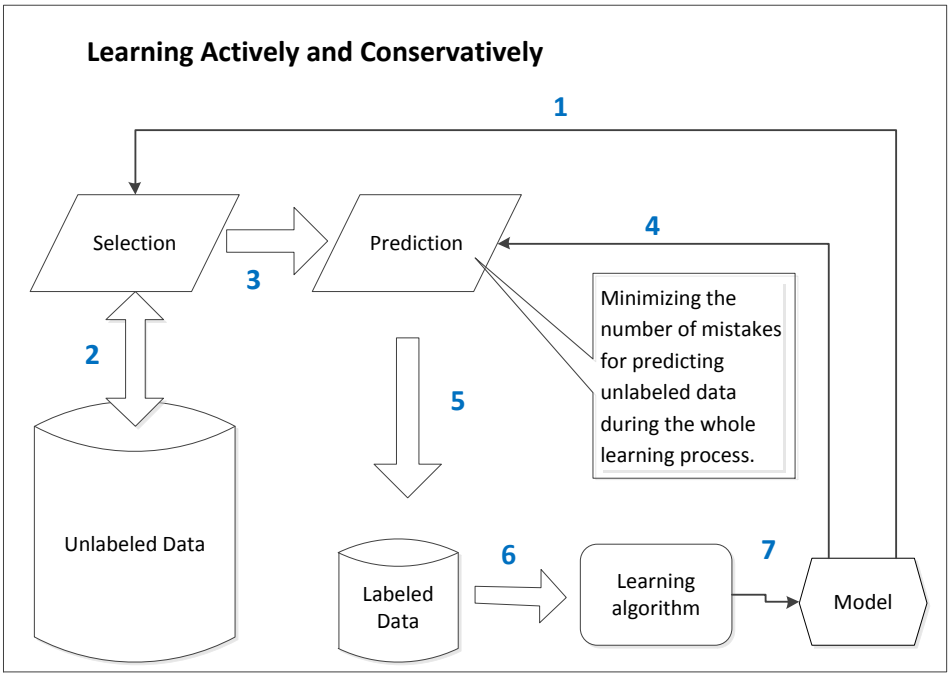


Figure 1.4: The framework of learning actively and conservatively. The learning is an iterative process of selecting examples actively to predict. The goal is to minimize the number of mistakes in predicting the unlabeled examples during the learning process. The numbers (1-7) indicate the workflow of the paradigm in each iteration.

Clearly, the problems under the paradigm of learning actively and conservatively cannot

be solved by the traditional active learning algorithms. To fill the gap, we propose a new and effective algorithm called *Most-Certain Learning* (*MCL)*. The basic idea of MCL is to select the next unlabeled example that can be predicted by the learner with the highest certainty (thus, it is conservative). The rationale behind is that learning is a gradual process, and the uncertain examples can become certain ones with more examples being learned. In this way, the goal of minimizing the number of mistakes during the learning process can be achieved.

In our empirical study, first of all, we apply MCL to an educational game Zoombinis, which is a learning problem under the paradigm of learning actively and conservatively (see Section 2.3.2 for the details), to illustrate the learning process of MCL. Then we also apply MCL to both UCI [2] datasets and one real-world application, direct marketing. The experimental results show that MCL works much better than other learning strategies. Furthermore, we also discover another advantage of MCL: the learning process is much more stable than other learning strategies. This property is often important as it makes the learning behavior more predictable. See Chapter 2 for detailed studies on this learning paradigm.

1.4.2 Learning Actively and Aggressively

To the opposite of the paradigm of learning actively and conservatively, traditional active learning algorithms with uncertain sampling are under the paradigm of *learning actively and aggressively* as they usually select the example that is predicted with the least certainty (i.e., aggressively) to query an oracle. Under this paradigm, the goal is to build a good model with as few labeled examples as possible. However, in traditional active learning, the label quality cannot be guaranteed as oracles are usually imperfect and the noisy labels of examples deteriorate the learning performances badly.

To guarantee the label quality, we use multiple imperfect oracles which are able to return both labels and their confidences in the labels (see Figure 1.5 for the framework). In fact, this circumstance exists commonly in real-life. For example, in paper reviewing, multiple reviewers (i.e., oracles or labelers) are requested to label a paper (as accepted, weak accepted, weak rejected or rejected), and usually the reviewers are required to give not only labels (accept, weak accept, weak reject or reject) for the paper, but also their confidences (high, medium or low) for the labelings. With the labels and confidences given by oracles, the final label (decision) quality can be estimated and guaranteed by querying more oracles if needed (see Steps 4, 5 and 6b in Figure 1.5).

Under this paradigm, we propose a new active learning strategy, called *c-certainty labeling*. C-certainty labeling guarantees the label quality to be greater than or equal to a given threshold c (c is the probability of correct labeling; see Section 3.2). Furthermore, instead of assuming noise level to be example-independent in the previous works, we allow it to be example-dependent. Our learning algorithm selects the best oracles to query for each given example. Thus, fewer queries are required on average for a label to meet the threshold c compared to random selection of oracles. As a result, for a given query budget, a more accurate model can be built with our learning algorithm.

Extensive experiments are conducted on the UCI [2] datasets by generating various types of oracles. The results show that our new algorithm is robust, and performs well for all the types of oracles. The reason is that, under the new learning paradigm, our learning algorithm can guarantee the label quality by selecting the best oracles to query (See Chapter 3 for the

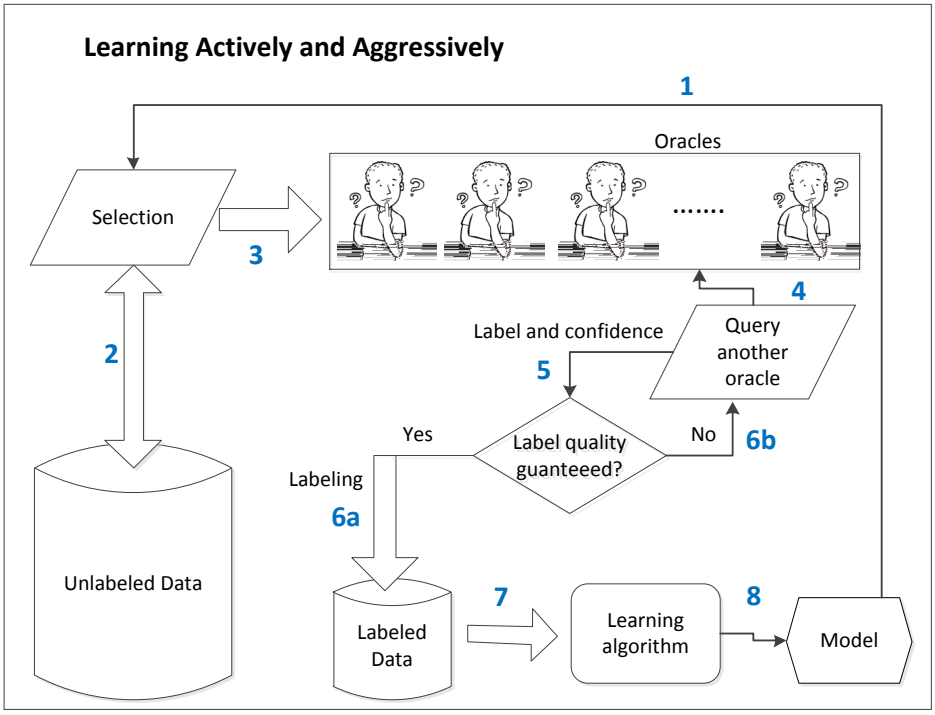


Figure 1.5: The framework for learning actively and aggressively with multiple noisy oracles. The learning is an iterative process of selecting examples actively to query oracles. For each example, the learner queries different oracles repeatedly until the label quality is guaranteed such that a good model can be built through this learning process. The numbers (1-8) indicate the workflow of the paradigm in each iteration.

detailed studies of this learning paradigm).

1.4.3 Learning Actively with Conservative-Aggressive Tradeoff

In the paradigm of learning actively and conservatively, learners always predict unlabeled examples during the learning process. On the other hand, in the paradigm of learning actively and aggressively, learners always query oracles for the labels. However, in many real-world applications, labels can be acquired by either querying oracles or making predictions. For example, when letters are sorted by using OCR (optical character recognition) devices of the post office, if the hand-written postal codes are ambiguous, or too difficult to recognize, they will be passed to the oracles (human) for labeling (i.e., querying oracles). However, if the OCR can predict accurately the hand-written postal codes, the letter will be sorted and mailed to the recipient directly (i.e., predicting directly) even though there is a small chance the prediction is wrong. To handle this type of learning problems, we propose a new paradigm of *learning actively with conservative-aggressive tradeoff* (as shown in Figure 1.6) such that the correct actions (predicting or querying) can be taken during the learning process.

The settings for the paradigm of learning actively with conservative-aggressive tradeoff are as follows. Firstly, the labels of examples are unknown. Secondly, to obtain the labels,

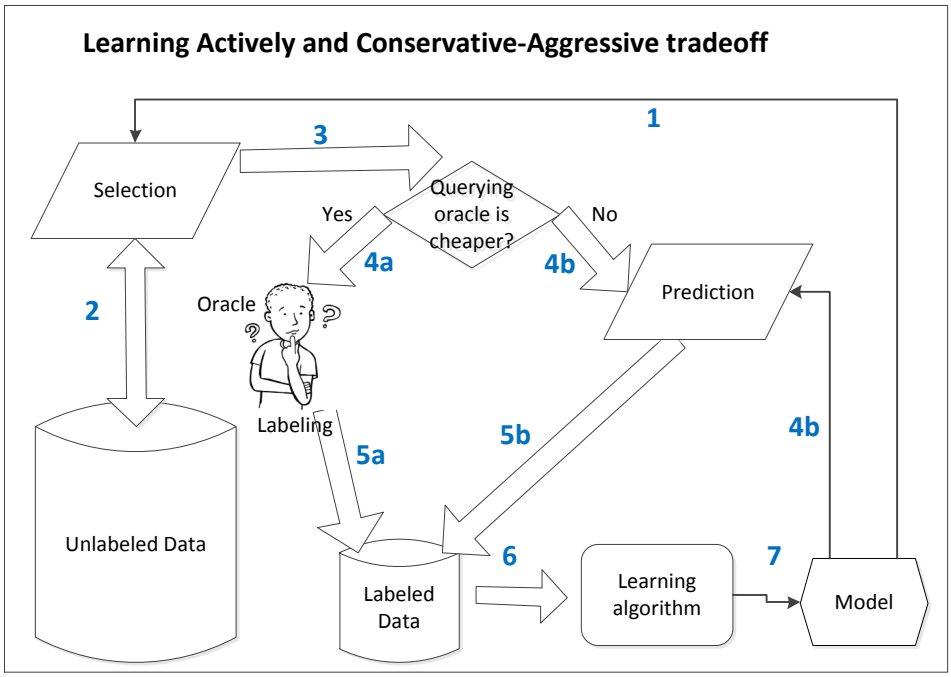


Figure 1.6: The framework of learning actively and conservative-aggressive tradeoff. The learning is an iterative process of actively selecting unlabeled examples to query oracle or for prediction depending on their costs. The numbers (1-7) indicate the workflow of the paradigm in each iteration.

two actions can be considered: querying oracles (e.g., asking human and experts) or making predictions (e.g., predicting mails by OCR directly). Lastly, cost has to be paid for making a wrong prediction or for querying an oracle. To reduce the cost, the choice of actions depends on the expected cost for making a wrong prediction (C_e) and the cost for query an oracle (C_q). For example, if $C_e > C_q$, the learner will choose to get the label by querying an oracle; otherwise, the learner will choose to predict its label directly. The selection of the two actions is optimal (i.e., obtaining the labels with minimal total cost) if the estimation of C_e is correct.

However, the expected cost C_e may not be very accurate particularly in the beginning of the learning process as the model is not good enough. For the example that its expected cost C_e is close to query cost C_q, it is likely that a wrong action will be taken, which may consequently lead to a high cost. On the other hand, wrong actions will unlikely be taken for the example that its C_e is much higher or much lower than C_q.

Considering the two actions and their different costs, we propose a novel learning algorithm, called *Decisive Active Learner* (DAL), which always selects the example likely leading to correct actions and prefers the action that is expected to cost less during the learning process. The examples that will likely lead to wrong actions will be learned during the later stage, as the learner may become more reliable (the probability is more accurate) and the action taken will become more accurate with more example being learned.

An empirical study is conducted on UCI [2] datasets, and the results show that DAL is able to select the correct actions to take during the learning process. Overall the performance of DAL is the best in terms of the total cost compared with other typical learners under different

cost settings. See Chapter 4 for detailed study on this learning paradigm.

1.4.4 Learning Actively with Minimal/Maximal Effort

In addition to the paradigms mentioned to improve the performances of learners on different goals, we also consider the relations between the performance of a learner on achieving learning goals and the *effort* that the learner takes during the learning process. Intuitively the effort can be reflected by the energy consumption of machine learning algorithms. As energy consumption can be affected by many factors such as processor speed, monitor size, we will use the amount of error to be corrected during learning, and the size of the learning models, to approximate the effort.

In fact, the *effort* of active learning can correspond well to the actual effort in human learning, where two circumstances are usually studied. One is learning knowledge with minimal effort. It is supported by an "i+1" education theory which suggests that less effort is required for human learning a small piece of new knowledge ("1") based on a large body of previously learned knowledge ("i")[49]. The other circumstance is that learning can be more efficient by paying maximal effort on correcting mistakes. It also has been studied in psychology and education in the past [105, 67].

The two circumstances of human learning researches are also important in the process of building machine learning modules. More specifically, in machine learning, minimal effort indicates consuming minimal energy during the learning process. Energy consumption controlling is very important[60], and when modules consume more energy they must be aggressively cooled or batteries will die sooner [45]. On the other hand, learning efficiency is also crucial for machine learning problems as tremendous data are collected everyday.

To study this learning effort problem, we propose a new paradigm of *learning actively with minimal/maximal effort*. In this paradigm, the labels of the examples are all provided and learners are allowed to select examples actively to learn. The learning goal is to control the learning process by selecting examples actively such that the learning can be accomplished with minimal effort or good models can be built fast with maximal effort. The framework of the paradigm is shown in Figure 1.7.

For the minimal effort learning, we propose an effective learning algorithm *Simple-to-Complex* (S2C). The basic idea of S2C is to repeatedly select and learn the example that its prediction given by the current learner is the closest to the true label (i.e., the simplest example) during the learning process. The rationale behind this greedy algorithm is that the complex examples will become simpler with more examples being learned. In this way, the learning effort can be minimized for the whole learning process.

On the other hand, for maximal effort learning, we repeatedly chose to learn the example that its prediction is the most different from the true label during the learning process. Due to the largest difference, the example is expected to be the most informative for the learner. In this way, it can improve the learner the most, and the learning is expected to be the most efficient.

The empirical study on UCI [2] datasets shows that our learning algorithms achieve the learning goals quite well. In particular, minimal effort learner does take less effort compared to other typical algorithms but it learns more slowly; on the other hand, maximal effort learner learns faster but with more effort. Thus, both of the learners have their own advantages and the determination on which learner to use depends on the learning goals.

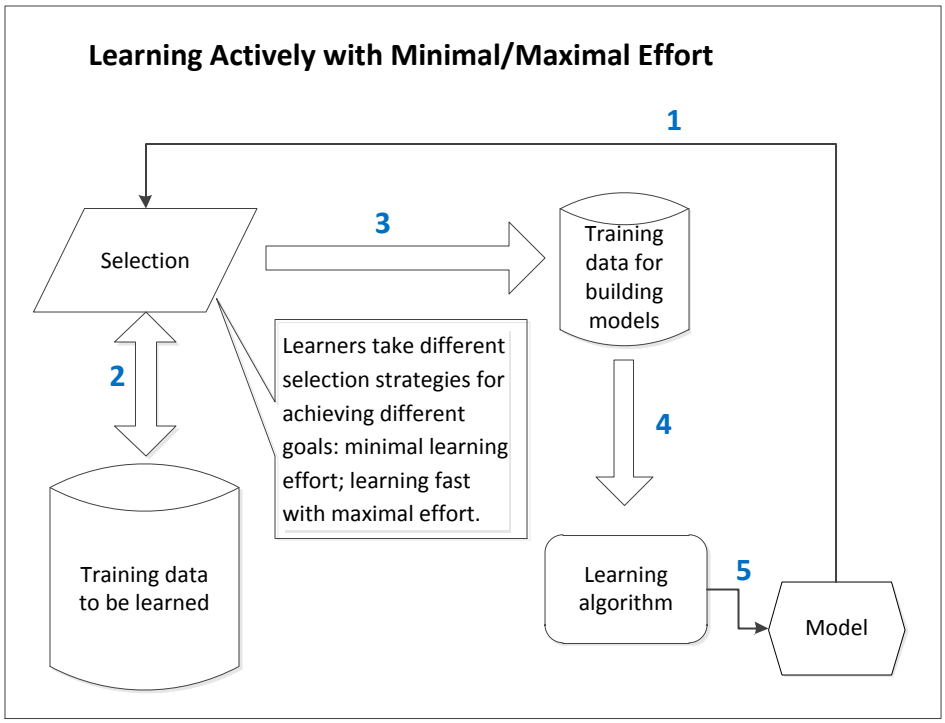


Figure 1.7: The framework of learning actively with minimal/maximal effort. The learning is to control the iterative process of selecting examples actively such that the minimal effort is consumed or model can be learned fast by paying maximal effort. The numbers (1-5) indicate the workflow of the paradigm in each iteration.

1.5 Contributions of the Thesis

Traditional active learning has been extensively studied on learning a better model with fewer labeled examples compared with traditional supervised learning. However, limitations exist when the traditional active learning algorithms are applied to real applications, as mentioned. To fill the gap, we propose four novel active learning paradigms based on the requirements of real applications. The contributions of this thesis can be summarized as follows.

- We propose a new learning paradigm, *learning actively and conservatively* in Chapter 2. Under this paradigm, learners actively select examples to learn, but their behavior of selecting examples is conservative in order to minimize the error in predicting examples. The learning algorithm is called MCL (Section 2.2) which prefers to learn the example that can be predicted with the highest certainty. This work is published in *The Proceedings of International Conference on Artificial Intelligence and Education (ICAIE), 2010* [63]

- Under the paradigm of learning actively and conservatively, we further implement two learning algorithms: MCL-b (Section 2.3) and MCL-1 (Section 2.4). MCL-b is for minimizing the number of examples of both of the classes. The empirical studies are conducted on an educational game and UCI [2] datasets. The results show that the algorithm

proposed does reduce the number of mistakes in predicting unlabeled examples during the learning process. MCL-1 is for retrieving the examples of one class in Chapter 2.4. The experiments are conducted on both UCI datasets and a direct marketing dataset. The results show that our learning algorithm works well on all the datasets, and our learning algorithm does select more customers who will buy products than other learning algorithms. This work is published in *The 7th International conference on Advanced Data Mining and Applications (ADMA'11)* [62].

- The second paradigm is *learning actively and aggressively* in Chapter 3. It selects examples actively and aggressively to learn, which is the same as traditional active learning. However, traditional active learning algorithms cannot guarantee label quality as oracles may not be always correct. We extend this learning paradigm by asking oracles to provide both labels and confidences. With this extension, we guarantee label quality to be higher than a given threshold c. In addition, an efficient learning algorithm is proposed to select the oracles with high labeling confidence to query. Extensively empirical studies are conducted on UCI [2] datasets. This work is published in *The 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining. May, 2012* [64].

- The third paradigm proposed is called *learning actively with conservative-aggressive tradeoff* in Chapter 4. Under this paradigm, to obtain labels, learners are allowed to take two actions: predicting directly by its current model or querying an oracle. The decision on taking which of the two actions depends on the expected cost for predicting and the cost for querying an oracle. To minimize the total cost, we proposed a novel learning algorithm, *Decisive Active Learner* (DAL), which prefers to learn the example that its action is more likely to be correct. Empirical study is conducted extensively on UCI dataset and the results show that DAL does work well. This work is submitted to *The IEEE International Conference on Data Mining (ICDM), 2012*.

- The last paradigm proposed is *learning actively with minimal/maximal effort* in chapter 5. It studies the relations between learning effort and learning efficiency. More specifically, two types of learning circumstances are considered: learning models with minimal effort and learning models fast with maximal effort. Two learning algorithms, S2C and C2S, are proposed for the two circumstances respectively. Experiments are conducted on UCI [2] datasets and the results show that S2C learns models with minimal effort but its learning efficiency is low; while C2S learns models with maximal effort but with high learning efficiency. This work is published in *Advances in Knowledge Discovery and Data Mining, 2010* [61].

Chapter 2

Learning Actively and Conservatively

In traditional active learning, learners only focus on the number of queries issued and the performances of final models learned. However, they ignore the performances of learners during the learning process. In this chapter, we propose a new paradigm of *learning actively and conservatively* which focuses on the process of learners selecting examples iteratively to learn and tries to make as few mistakes as possible in predicting unlabeled examples during the learning process.

The learning process is important for many applications. For example, in direct marketing, a sales agent (learner) has to focus on the process of selecting customers to approach, and tries to make correct predictions (i.e., fewer mistakes) on the customer who will buy the product. As another example, a doctor has to focus on the process of diagnosing each patient and tries to predict correctly. This type of learning problems are under the paradigm of learning actively and conservatively.

In general, the problems under the paradigm proposed have specific settings as mentioned in Section 1.4.1. One is that unlabeled examples are available for learners to select actively. This is the same as traditional active learning. The other one is that the selected example is predicted by the learner and the true label will be revealed after each prediction. In the direct marketing example, the true label of a customer ("buy" or "not buying") will be revealed after an agent predicts him/her as "buying" and approaches. The goal of the learning paradigm is to minimize the number of mistakes in predicting unlabeled examples during the iterative learning process.

Under the paradigm of learning actively and conservatively, we propose a new, practical and effective learning algorithm, *Most-Certain Learning* (*MCL)*. To minimize the number of mistakes, MCL chooses to learn the next example whose label can be predicted by the current learner with the highest certainty. This work of MCL is published in *The Proceedings of International Conference on Artificial Intelligence and Education (ICAIE), 2010* [63].

Furthermore, to satisfy the requirements of real-world applications, we further discuss two types of MCL. One is that the learner must select and predict all of the remaining examples as either positive or negative classes, called *binary-class MCL*, or*MCL-b*. The goal of MCL-b thus is to minimize mistakes of both classes. For example, a doctor usually must see and diagnose all patients as healthy or sick. The other type is that the learner only needs to select and predict one class of examples, called *single-class MCL*, or *MCL-1*. In the direct marketing example, an agent usually only cares about those customers who will likely buy the product

(positive examples), and does not need to predict, approach, and verify customers who will unlikely buy the product (negative examples). This work is published on *The 7th International conference on Advanced Data Mining and Applications (ADMA'11)* [62].

The rest of this chapter is organized as follows. Section 2.1 reviews related works. Section 2.2 describes the framework of the Most-Certain Learning (MCL) strategy. Section 2.3 and Section 2.4 present the details of implementing the two versions (MCL-b and MCL-1) of MCL and present their experimental results respectively. We summarize the work of this chapter in Section 2.5.

2.1 Related Works

The paradigm of learning actively and conservatively is an active learning paradigm in the sense that it allows learners to select examples actively to learn. However, existing active learning algorithms [82, 15] cannot solve the problems under the paradigm as their settings and goals are different. The traditional active learning targets on minimizing the number of examples labeled and the high prediction accuracy of final models; while our paradigm is to minimize the number of mistakes in predicting unlabeled examples during the learning process. Furthermore, in the traditional active learning learners can only get labels by querying oracles; while under our paradigm, examples are predicted by learners and the true labels will be revealed after each prediction.

The learning problems under our paradigm may seem to be similar to agnostic active learning [3, 16] as both of them are related to mistakes in labels. However, the noise in agnostic active learning comes from the oracle who provides the labels, and the true labels are hidden. The goal of agnostic active learning is to improve the sample efficiency. For our learning problem, the mistakes come from the prediction of the current immature learner and the true label will be revealed after the prediction, and the goal is also different.

Another work that is quite similar to our work is self-directed learning [9, 38, 75]. Theoretically it studies the learning problem on simple classes of concepts, such as disjunction, conjunction, k-DNF and so on, to minimize the number of mistakes. However, the learning algorithm proposed must know and keep the set of all target concepts. It chooses the next example that has the greatest difference between the number of concepts that predict it differently (positive vs. negative labels). In a sense, the algorithm chooses the example that can be predicted most certainly. However, the target concept class is often unknown, nor is it feasible to keep all the concepts for a learning algorithm in real-world applications. As far as we know, there is no previous work in designing a practical learning algorithm that works well (i.e., making fewer mistakes) for solving the problems under the paradigm of learning actively and conservatively.

2.2 Most-Certain Learning (MCL) Strategy

As mentioned, the problems under the new paradigm cannot be solved by the traditional active learning algorithms. To fill the gap, we propose a new, practical and effective learning algorithm, *Most-Certain Learning (MCL)*. The basic idea of MCL is to learn the example that can

be predicted by the current learner with the highest certainty first, and leave the examples with low certainty to learn during the later stage. One might think that the total number of mistakes would be the same no matter what order of examples to be selected. Afterall, all unlabeled examples need to be learned, from easy to complex or from complex to easy ones. It is analogous to $1 + 2 + 3 = 3 + 2 + 1$. However, we will show that the learning sequence does matter. The rationale behind is that, with more examples being learned, the prediction accuracy of the learner can be improved, and consequently uncertain examples can become certain. In this way, the goal of minimizing the number of mistakes in predicting unlabeled examples can be achieved. In particular, MCL can be defined formally as follows.

Let $\mathcal{D}_U$ be the unlabeled example set, and C be the concept class over $\mathcal{D}_U$. MCL focuses on the iterative learning process as follows: in each iteration, it chooses a new element (example) $x_i \in \mathcal{D}_U$ that can be predicted by the current learner $\mathcal{L}$ with the highest certainty. It then outputs the label l_i given by $\mathcal{L}$ and in response the true value $c_t(x_i)$ will be revealed, where c_t ($c_t \in C$) denotes the target function. The learner will update its current model with all the labeled examples $\mathcal{D}_T$. The learning process continues until all the elements in $\mathcal{D}_U$ are learned or other stopping criteria are met. Let $m(\mathcal{L}, c_t)$ denote the number of mistakes made by $\mathcal{L}$, i.e., the total times of $l_i \neq c_t(x_i)$. The goal of MCL is to minimize $m(\mathcal{L}, c_t)$. The pseudocode for MCL is shown in Algorithm 1.

Algorithm 1: MCL

Input: Unlabeled Dataset $\mathcal{D}_{\mathcal{U}}$; Training data: $\mathcal{D}_T$; Initial model: $\mathcal{M}_0$
Output: Model: $\mathcal{M}$ and the number of mistakes: m

```
1  begin
2      itr = 0; //the first iteration
3      m = 0;
4      while D_U <> NULL do
5          for each x_i ∈ D_U do
6              Calculate the certainty of x_i;
7          end
8          //Select the most certain example
9          Select the example x_i with the highest certainty;
10         l_i ← label prediction of x_i;
11         c_t(x_i) ← the true label of x_i;
12         if l_i ≠ c_t(x_i) then
13             m = m + 1;
14         end
15         D_T ← D_T + x_i;
16         Update the current model with D_T;
17         D_U ← D_U − x_i;
18         itr + +;
19     end
20     M ← M_itr;
21     Return M and m;
22 end
```

MCL is a wrapper learning algorithm, and can take any classifier that generates delicate probability prediction as its base learner, $\mathcal{L}$. In this paper, bagged decision trees [71] are taken as the base learner for two reasons. First of all, it is easy to obtain an accurate probability of the prediction. The second reason is that a large number of empirical studies in machine learning have shown that bagged decision trees make classification more accurately compared to a single tree [8, 71].

MCL provides a framework for solving problems under the new paradigm. According to the diverse learning problems in real-world applications, we further implement two types of MCL. One type is that learners must select and predict all of the remaining examples as either positive or negative class[1], called *binary-class MCL*, or *MCL-b*. The goal of MCL-b thus is to minimize the number of mistakes on both classes. For example, in the Zoombinis game mentioned in Chapter 1 (for details see Section 2.3.2), a player has to label a Zoombini (a small creature) with either of the two bridges. As another example, a doctor usually must see and diagnose each patient as healthy or sick. The other type is that learners only need to retrieve one class of examples, called *single-class MCL*, or *MCL-1*. In the direct marketing example, an agent usually only cares about those customers who will likely buy the product (positive examples), and does not need to predict, approach, and verify customers who will unlikely buy the product (negative examples).

2.3 MCL-b Learning Algorithm

MCL-b works under the framework of MCL as follows. It selects the most certain example of *both* classes and predicts it with its current model. Then MCL-b updates its current model with the newly labeled example. This predicting-and-updating process iterates until all the examples are labeled.

In this subsection, first of all we will introduce the implementation details of MCL-b. Then we will present how MCL-b works and its performances on an educational game Zoombinis. After that, we will show the experimental results of MCL-b on UCI [2] datasets extensively.

2.3.1 MCL-b Implementation Details

To implement MCL-b, three technical issues deserve extensive discussion. The first issue is the selection of the most certain example. The second issue is the selection of the first example. The last issue is how to select an example when the labeled examples are of the same class.

The first issue, selection of the most certain example, is crucial. The most-certain example is the one that is predicted with the highest probability by the current learner, i.e., the example that satisfies $\arg\max_{x_i \in \mathcal{X}}(\max(prob^+_{x_i}, (1 - prob^+_{x_i})))$. $prob^+_{x_i}$ is the probability of predicting x_i as positive. In bagged decision trees (which is used as our base learner), $prob^+_{x_i}$ is the number of decision trees that vote for positive out of the total number of decision trees.

The second issue, the selection of the first example, can be tricky, as the current model is empty. MCL-b scans the dataset and chooses the example that appears with the highest

[1] This study assumes that the learning problem is binary-class. The multi-class problem can be transformed to binary-class problems.

frequency. If no example appears more than once, MCL-b chooses one example from the dataset randomly.

The last issue is how to choose the next example if labeled examples so far are of the same class, as a model built from the examples of the same class, say positive, will predict all examples as positive. Here, we use the Euclidean distance [92] as heuristic information for selecting the next example. The basic idea is that the example far from a positive (negative) center is likely negative (positive). Assuming that MCL-b has a positive example x_1 with nominal attribute values {*1, 0, 0, 1*}, we consider it as the center of positive, and the point that has the furthest Euclidean distance from x_1 (here the possible furthest Euclidean distance is *4*) as the center of negative. The most certain example is the one that is closest to either of the two centers, and the label of the center will be assigned to the example. For example, if the nearest example from x_1 is y_u, and the Euclidean distance $d_{x_1y_u} = 1$, and the furthest is y_v, and $d_{x_1y_v} = 4$, MCL-b would consider y_v as the most certain example and label it as negative. The reason is that y_v has distance *0* to the negative center, which is closer than the distance of y_u to the positive center. This strategy can be formulated as follows.

$$y = \begin{cases} y_u, & \text{if } \sum_{x_i \in S} d_{x_i y_u} < |S| * m - \sum_{x_i \in S} d_{x_i y_v} \\ y_v, & \text{otherwise} \end{cases} \tag{2.1}$$

where S is the set of labeled examples, $|S|$ is the size of S, m is the number of attributes, $d_{x_iy_u}$ is the Euclidean distance between x_i and y_u, and y_u and y_v are the closest and furthest unlabeled examples to the current labeled examples respectively.

After presenting the implementation techniques, we will show how MCL-b works on a learning problem (an educational game Zoombinis) under the paradigm of learning actively and conservatively. Then we will present the performances of MCL-b in playing the game. Furthermore, extensive empirical studies on UCI [2] datasets will also be conducted.

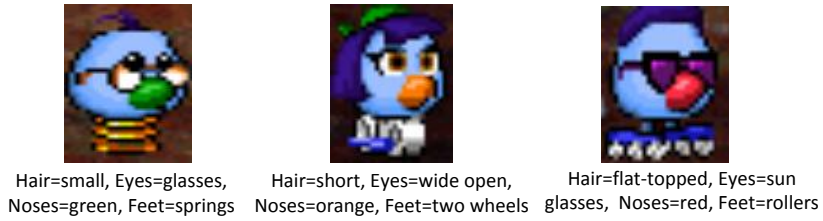
2.3.2 Application in the Educational Game Zoombinis

MCL-b can be applied to many real-world learning problems under the paradigm of learning actively and conservatively, even including some human concept learning problems. In this section, we apply the machine learning algorithm MCL-b to an educational game, Zoombinis, to illustrate how it works and show its performances on this learning problem.

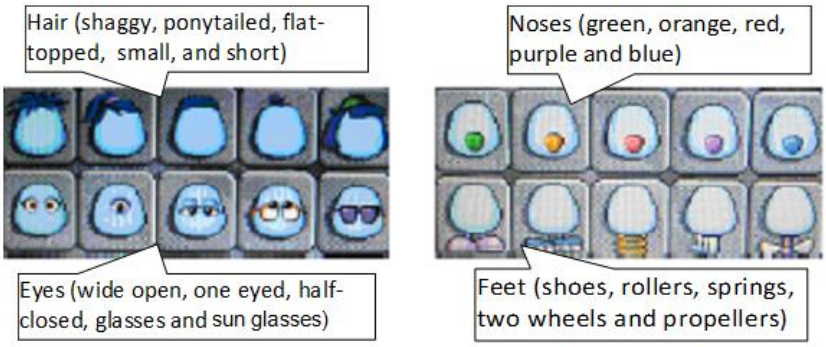
The game Zoombinis is a well-known series of software published by the Learning Company [2]. It aims at children or teenagers, but fun for adults as well. It is based on the "Zoombinis", small blue creatures, which are depicted with varying hairs, eyes, noses and feet. Figure 2.1 shows three Zoombinis and possible feature values. In the Logical Journey of the Zoombinis game, Zoombinis have to search for a new home, and on their journey they encounter a variety of "obstacles" (puzzles), which players must solve. In this experiment, we choose one puzzle, called the Allergic Cliffs, to work on.

In the Allergic Cliffs (Figure 2.2), the player is given 16 randomly-generated Zoombinis. The goal is to take all of the 16 Zoombinis across the cliff to the right side over the two bridges. The two bridges are supported by 6 wooden pegs. Each bridge allows only certain

[2]There are three titles in the series: The Logical Journey of the Zoombinis, Zoombinis: Mountain Rescue, and Zoombinis: Island Odyssey



(a) Three Zoombinis



(b) Zoombinis' features

Figure 2.1: Zoombinis and their features.

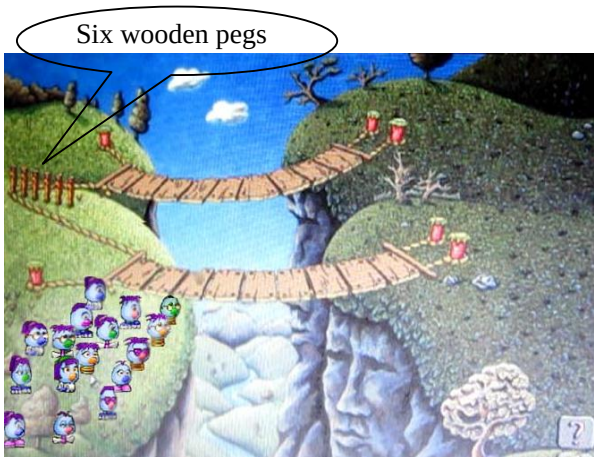


Figure 2.2: Allergic cliffs of Zoombinis game.

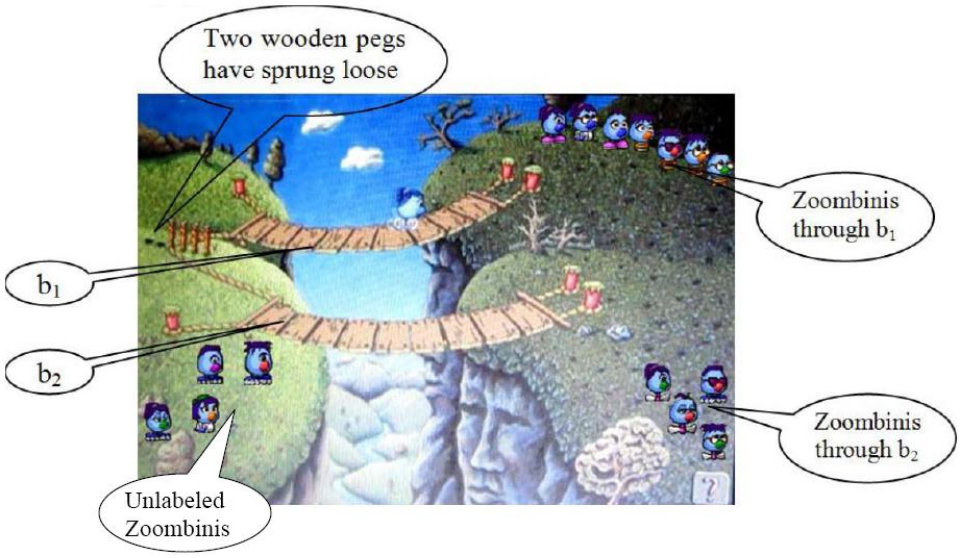


Figure 2.3: Screen shot of playing the Allergic Cliff (Zoombinis at the top right corner crossed bridge b1 (top bridge); those at the bottom right corner crossed bridge b2 (bottom bridge)).

Zoombinis to cross if they have certain combinations of features. The pattern (model) for each bridge is randomly formed and hidden to the player. For example, the top bridge may only allow Zoombinis with red or blue nose to cross (and thus, the bottom bridge allows all other Zoombinis to cross). If the player selects a Zoombini to cross a wrong bridge, the cliff would sneeze, the Zoombini will be sent back to the left side, and a peg will spring loose by the powerful sneeze. Figure 2.3 is a screen shot in taking the Zoombinis in Figure 2.2 across the cliff. We can see that two pegs have sprung loose due to two mistakes. If all six pegs come loose before all of the 16 Zoombinis pass through, the player fails this part of the journey, and must try again with 16 new Zoombinis (and a new hidden concept for the bridge). Thus, to win the game, players are allowed to make at most 5 (including 5) mistakes.

This learning problem of Allergic cliffs is clearly under the paradigm of learning actively and conservatively. First of all, unlabeled examples (the 16 Zoombinis without knowing which bridge to cross) are provided, and a learner (player) is allowed to select examples actively to learn (select any Zoombini to try). The label (which bridge a Zoombini should go) is given after each prediction. The goal of the player is to predict and learn the model of allowable Zoombinis for the bridge, without making more than 5 mistakes in the process. In addition, as each Zoombini has to be labeled with either of the two labels (bridges), MCL-b is suitable for this learning problem.

To study the performances of MCL-b in playing the Zoombinis game, we implement it based on the WEKA [104] source code, and take bagged decision trees as the base learner. Ten groups of Zoombinis data are generated by playing the game 10 times and each group has 16 Zoombinis. All the Zoombinis (example) are encoded with their feature values.

2.3.2.1 A Case Study

We first show how MCL-b works by presenting the detailed process of MCL-b in playing one specific game as follows. Figure 2.4 shows the set of 16 Zoombinis that MCL-b needs to take across the bridges in the Allergic Cliff puzzle. For easy description, we assign each Zoombini an ID number as shown in Figure 2.4, and the attribute values of the first three Zoombinis are shown in Table 2.1.

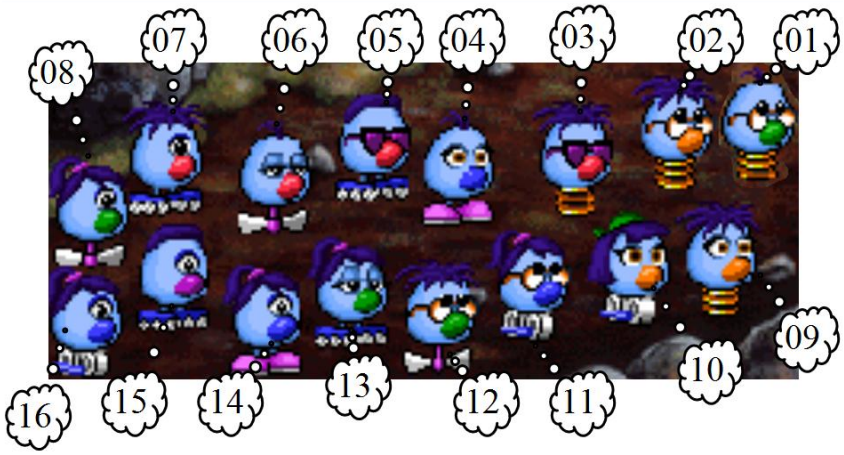


Figure 2.4: A case of the Zoombinis game for a human subject.

Table 2.1: The attribute values of the first three Zoombinis in Figure 2.4 .

Zoombini ID	hair	eyes	nose	feet
01	small	glasses	green	springs
02	shaggy	glasses	orange	springs
03	shaggy	sun glasses	red	springs

As no model exists at the beginning, MCL-b starts learning with Zoombini 01 and selects bridge b1 randomly for it, and finds that it passes b1 safely. One labeled example is obtained. Then, to select the next Zoombini, MCL-b calculates the distances between the labeled example and all unlabeled examples, and finds that Zoombini 05 is totally different from Zoombini 01. Thus, MCL-b predicts Zoombini 05 as going through bridge b2, and the result shows that the prediction is correct. After having two examples, MCL-b builds a model "*if Eyes=glasses then uses b1; otherwise uses b2*". According to this model, MCL-b takes Zoombini 02 (Eyes=glasses) through b1 successfully. Then it takes Zoombini 03 (Eyes=sun glasses) through b2. However, it fails. With new labeled examples, the model is updated to be "*if Feet=spring then use b1; otherwise use b2*". With this model, Zoombini 09 uses b1 safely, and Zoombini 07, 13, 15 use b2 safely. However, Zoombini 11 fails in using b2. The model is updated again and becomes "*if feet = spring then the Zoombini uses b1 (bridge 1); else if nose = blue the Zoombini uses b1; otherwise (feet = not spring and nose = not blue) the Zoombini uses b2*" (see Figure 2.6). According to this model, all the rest Zoombinis go through the bridges safely (the label of each Zoombini are shown in Figure 2.5).

Figure 2.5: Zoombinis going through b1 (upper) and b2 (lower).

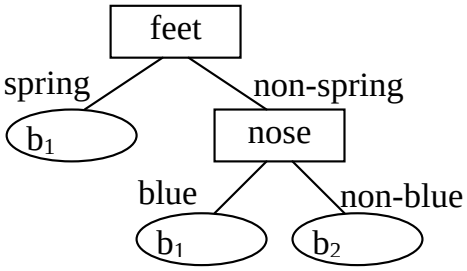


Figure 2.6: Tree model built over the Zoombinis.

2.3.2.2 Comparison of Learning Sequences in the One Case

To compare with MCL-b, we implement another two learning strategies. The first strategy is most-uncertain sampling which is widely and successfully applied in active learning [82, 101], and we call it *most uncertain learning* (*MUL*) in this work. MUL always selects the unlabeled example that is labeled by the current learner with the highest uncertainty. Compared to MCL-b, MUL is competitive and may make fewer mistakes in predicting unlabeled examples, as many active learning works [82, 101] have concluded that it can build a good model quickly and may make fewer mistakes during the later learning stage. The other strategy is *Random*, which builds models by selecting examples randomly, and is used as a baseline for the comparison in the experiment.

For comparison, both MUL and Random also use the bagged decision trees as their base learner and are implemented based on the WEKA source code. The number of mistakes and the learning sequences of Zoombinis (examples) are compared for the specific case in the following.

For comparison, we list the number of mistakes and the learning sequences of MCL-b, MUL and Random in Table 2.2. It shows that MCL-b makes two mistakes for taking all the 16 Zoombinis across the cliff, and MUL makes 6 and Random makes 4 mistakes respectively. That is, MCL-b makes the fewest mistakes, and MUL makes the most mistakes on the same set of Zoombinis.

Table 2.2: The number of mistakes and the learning sequences of MCL-b, MUL and Random.

Learner	No. of mistakes	Sequence of Zoombinis selected
MCL-b	2	01,05,02,03,09,07,13,15,11,04,14,16,06,08,10,12
MUL	6	01,04,07,13,15,08,09,14,16,10,12,02,05,03,06,11
Random	4	01,07,15,04,09,11,05,02,03,12,14,06,10,16,13,08

Why the number of mistakes of the three learning strategies are different? Since the base learners of MCL-b, MUL and Random are the same, i.e., bagged decision trees, the only difference is their learning sequence of examples (Zoombinis), as shown in Table 2.2. It is evident that the learning process is crucial for achieving the learning goal of this problem.

2.3.2.3 Comparison on 10 Sets of Zoombini Data

In this section, we will present the experimental results on all the 10 sets of Zoombini data in terms of the number of mistakes and the learning efficiency. The number of mistakes is crucial in playing the Zoombinis game, and it determines if the learning is successful or not. Learning efficiency is not the goal for solving this problem, but still it allows us to analyze the learning behavior extensively.

To observe the number of mistakes each learning strategy makes, we allow the learning process to continue until all of the 16 Zoombinis have been taken to the right side of the cliff, even if the number of mistakes is already greater than five. The numbers of the mistakes made by MCL-b, MUL and Random on the 10 games of the Allergic Cliff are presented in Figure 2.7. The x axis indicates the 10 games and y axis the number of mistakes. We can see clearly that

for all the 10 games, MCL-b makes no more mistakes than MUL and Random. On average, the number of mistakes made by MCL-b is about 40% less than that by MUL and Random.

If we only consider the number of mistakes less than 6 (otherwise, the game is lost), MCL-b still makes about 15% and 20% less mistakes than MUL and Random respectively. MCL-b wins all the 10 games, but both MUL and random fail in 4 games. The t-test result also confirms that such a difference is significant. (The t-test shows the difference between MUL and Random is not significant). Thus, MCL-b indeed makes the least number of mistakes during the learning process.

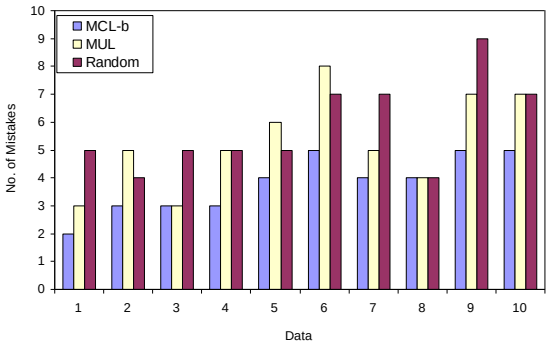


Figure 2.7: The number of mistakes on the 10 sets of Zoombini data.

The experimental result on the number of mistakes agrees with our expectation that MCL-b makes fewer mistakes than the other two learning strategies. The rationale behind is that learning is a gradual process. MCL-b exploits the examples that its current model can predict with high certainty, and this exploitation process makes those uncertain examples become certain gradually. However, MUL, which we think may be a strong competitor of MCL-b in minimizing the number of mistakes, makes much more mistakes than MCL-b does in playing this game. Overall, MUL even makes more mistakes than Random.

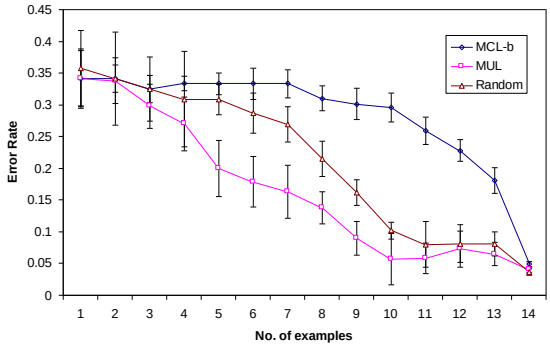


Figure 2.8: The learning efficiency of the three learning strategies.

Is there any advantage of the aggressive learner MUL over the conservative learner? Intuitively, MUL should learn the model more efficiently than MCL-b (even though MUL makes more mistakes than MCL-b). To confirm our intuition, we show the average testing error rate of

the 10 sets of data with error bar during the learning processes of MCL-b, MUL and Random in Figure 2.8. The x axis is the number of examples that the current model is built on, and the y axis is the average prediction error rate on the remaining examples. The smaller the prediction error rate, the better the learned model is. From the figure, we can clearly see that the error rate of MUL decreases most quickly. That is, MUL does learn the model more efficiently (even though it makes more mistakes as shown earlier).

To summarize, the experimental results show that MCL-b does play the educational game well. However, its learning efficiency in building a good model is quite low.

2.3.2.4 Comparison between Human Subjects and Machine Learners

After presenting the performances of the three machine learners in playing the Zoombinis game, it would be interesting to compare the performances of human learners and machine learners. Several graduate students in our lab are invited to solve the same puzzle. We first introduce the game to them (without asking them to use any learning strategy), and let them play it once to become familiar with the game. We tried to find if they are using a strategy similar to our machine learners consciously or unconsciously when they are playing the game.

To compare the differences between human subjects and machine learners, we present the average number of mistakes they make for playing the game on the 10 sets of Zoombinis in Figure 2.9. We can see that, in general, human subjects make a bit more mistakes than MCL-b, and much fewer than MUL.

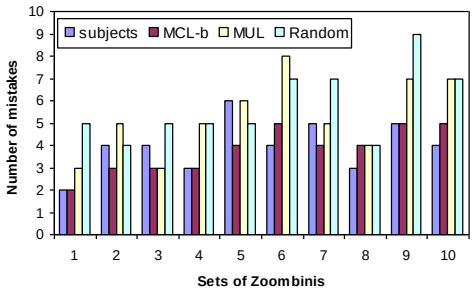


Figure 2.9: Comparison of the number of mistakes.

How could human subjects perform so well? By studying their playing process, we find that the human subjects tend to select the examples that can be predicted well by the current concept. For example, after taking two Zoombinis across the cliff one subject learns that the Zoombinis with red nose go through bridge b1. Then he prefers taking the Zoombinis with red nose to cross b1 as he thinks that it is more likely to be correct. That is, in this case, the subject does seem to use the MCL-b strategy, even though the performance is not as optimally as MCL-b.

2.3.3 Empirical Studies of MCL-b on UCI Datasets

The empirical study of MCL-b has been done on the Zoombinis game as stated and the results have shown that MCL-b works well in reducing the number of mistakes for predicting unla-

Dataset	# of Attribute	Class dist.(0/1)	Dataset	# of Attribute	Class dist.(0/1)
anneal	38	484/214	ecoli	7	193/143
autos_new	25	138/67	glass_new	9	138/76
breast_cancer	10	458/241	heart-h	12	188/106
colic	22	232/136	sonar	60	111/97
diabetes	8	500/268	vote	16	248/187

Table 2.3: Datasets used in the experiments

beled examples during the learning process. However, the dataset for the game is too small (only 16 Zoombinis). To show the performances of MCL-b further, we conduct the experiment on UCI [2] datasets and compare MCL-b with MUL and Random, which have been introduced in Section 2.3.2.2.

In this experiment, MCL-b, MUL and Random are implemented and applied to 10 UCI [2] datasets shown in Figure 2.3. The datasets are commonly used in the supervised learning research, and for each dataset 70% examples are used as training data and 30% as test data. In our experiment the t-test results are of 95% confidence.

To compare the different learning strategies extensively and systematically, three crucial measurements, *the number of mistakes*, *volatility* and *learning efficiency*, are used in this empirical study[3]. The *number of mistakes* is the number of unlabeled examples being predicted wrongly by the learner during the learning process. This measurement would be the most important, as minimizing the number of mistakes is the goal of the learning paradigm. The second measurement *volatility* is the standard deviation of the number of mistakes from running r times (r=5 in our experiment). In particular, $volatility = (\sum_{i=0}^{r}(M(MCL, c_t)_i - \frac{1}{r}\sum_{i=0}^{r} M(MCL, c_t))_i^2)^{1/2}$. That is, volatility indicates the stability of a learning algorithm. The less the volatility, the more predictable the learning behavior is. The last measurement is *learning efficiency*, which indicates how fast a learner learns a good model. A learner with higher learning efficiency requires fewer labeled examples to train a model that has lower error rate in predicting test data.

2.3.3.1 Comparison of Number of Mistakes

To observe the performance of MCL-b in terms of the number of mistakes, we run the learning strategies on each dataset for 5 times, and show the averages number of mistakes on learning 1/4, 2/4, 3/4 and 4/4 of the training data respectively in Figure 2.10. We can see clearly that the mistakes made by MCL-b on learning 1/4 data are much lower than that by Random and MUL. Even though the difference between MCL-b and the other two learning strategies reduces with increasing labeled examples, it still makes fewer mistakes.

To make the comparison clearer, we summarize them further in Table 2.4. The table shows that the average number of mistakes made by MCL-b are 10%, 22% and 19% less than that by Random, and MUL respectively.

[3]Actually the two measurements, the number of mistakes and learning efficiency, have been used in the empirical study on the Zoombinis game. However, volatility has not been used, as the dataset of the game is too small (only 16 examples) and volatility may not work properly.

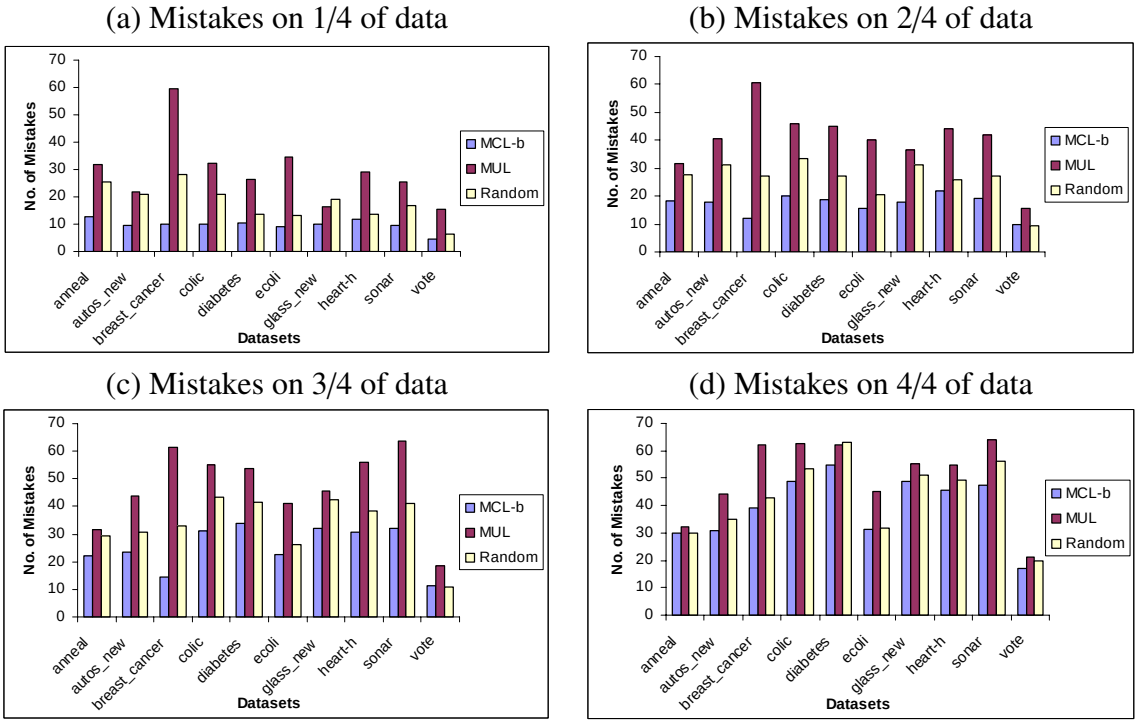


Figure 2.10: The number of mistakes

Table 2.4: The ratio of mistakes between MCL and the others

	MCL/Random	MCL/MUL
1/4 of data	55%	33%
2/4 of data	66%	43%
3/4 of data	75%	54%
4/4 of data	90%	78%

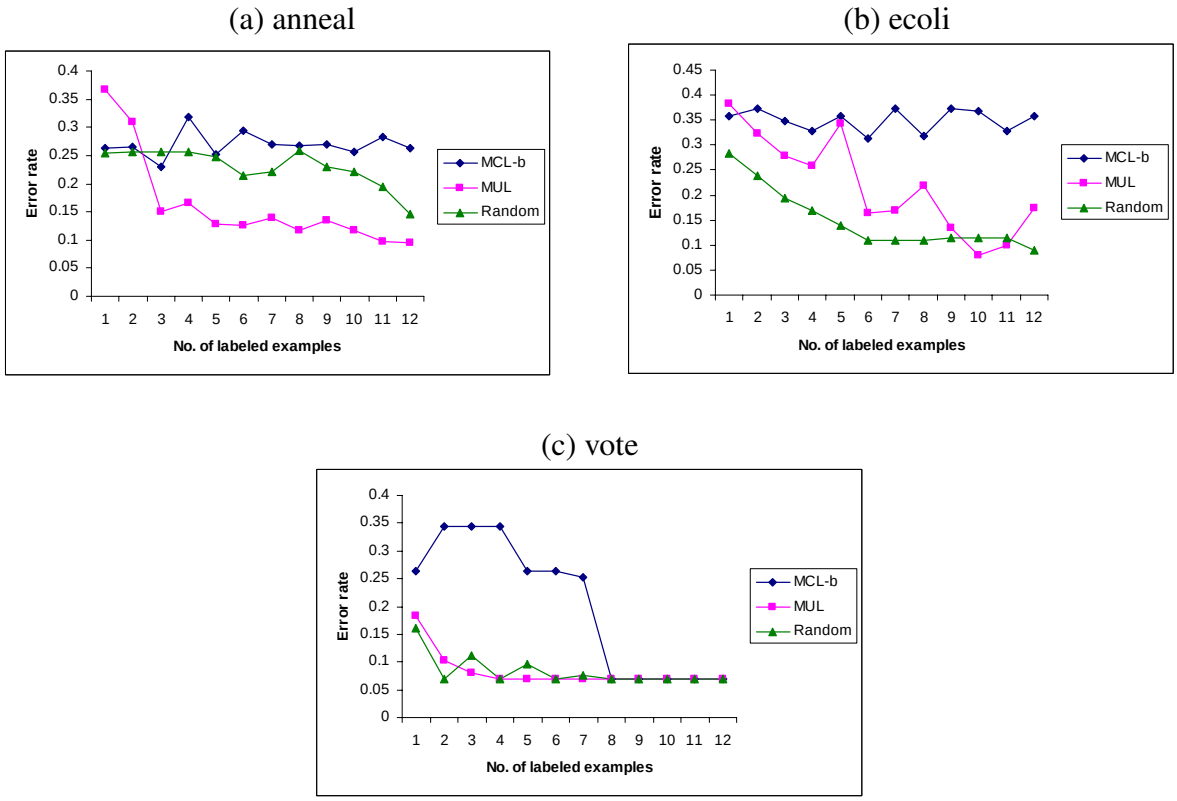


Figure 2.11: Performances of models built on the first 12 examples

The t-test results also show that MCL-b performs better than the other two learning strategies on all the 10 datasets except that it ties with Random on three datasets (anneal, ecoli and vote) after the whole training data are learned. That is, in general MCL-b does reduce the number of mistakes made during the learning process.

Why does the tie between MCL-b and Random happen on the three datasets? The prediction error rate of the models learned over the first 12 examples in Figure 2.11 shows that the model built by Random has already performed well by learning a few examples on the three datasets. The good performance of the model indicates few mistakes in labeling the remaining examples, which leads to similar performance between Random and MCL-b.

However, one might wonder why MUL makes more mistakes than Random and MCL-b, even though we can see that it builds a good model even faster than Random in Figure 2.11. To explain this, we show the number of mistakes made during the learning process on two datesets (due to the similar performances on the 10 datasets), anneal and autos, in Figure 2.12. The x and y axis are about the number of mistakes and the number of examples labeled. It shows that MUL may make many mistakes even after a good model is built, as shown in Figure 2.12. This is because MUL always chooses the most uncertain example to predict. Even for a good model, mistakes still can happen on the most uncertain examples.

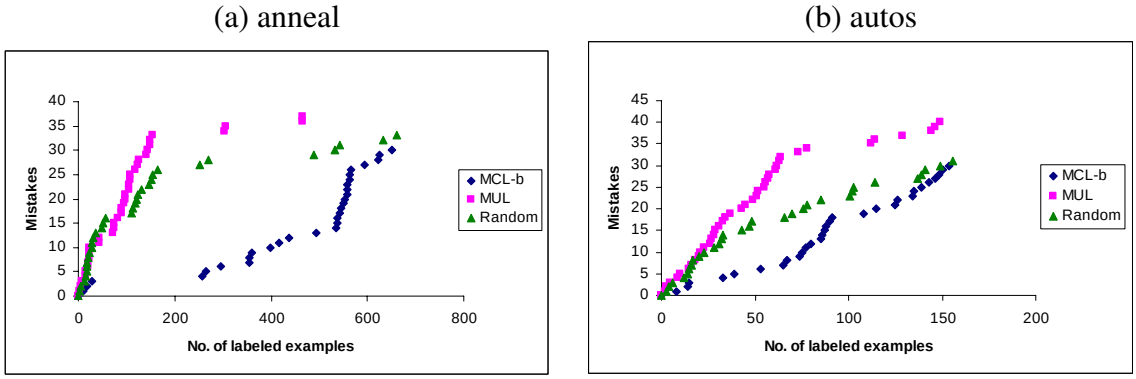


Figure 2.12: Mistakes vs. labeled examples

2.3.3.2 Comparison of Volatility

As mentioned, volatility is the standard deviation of the number of mistakes of different runs. The volatilities of the three strategies are presented in Figure 2.13. It shows clearly that the volatility of MCL-b is significantly less than that of MUL and Random on all the 10 datasets without exception. The reason is that MUL chooses the most-uncertain example to learn in each iteration, which may affect the current learner by a wide margin. On the other hand, MCL-b always selects the most-certain example, which may not be able to improve the current model much, but in a stable manner. The low volatility is important in real applications, as it indicates that the learning behavior is more predictable.

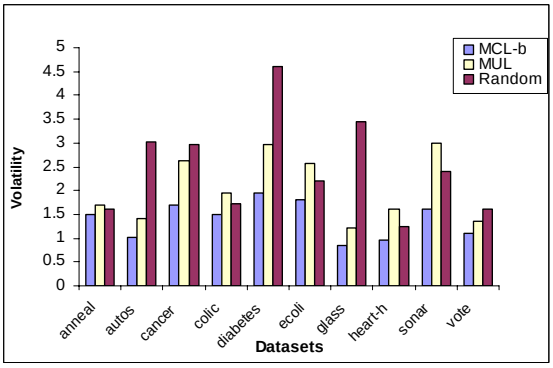


Figure 2.13: Volatility of learners on UCI datasets.

2.3.3.3 Comparison of Learning Efficiency

Learning efficiency indicates how fast a learner learns a model that has low error rate in predicting test examples. Active learning research has shown that MUL has high learning efficiency. On the other hand, MCL-b is expected to learn slowly. To verify our expectation, we show the predictive error rate of the models on one dataset (due to the similar performance on the 10 datasets), autos, in Figure 2.14. The x and y axis are about the number of examples labeled and the prediction error rate of the current model on test data. Figure 2.14 shows clearly that the error rate of MUL drops dramatically, while the error rate of MCL-b reduces slowly, as expected. This indicates that the most uncertain examples are more informative and can improve the current model efficiently.

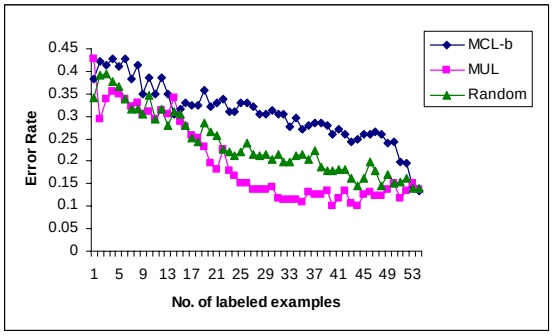


Figure 2.14: Learning efficiency on UCI datasets.

In summary, MCL-b does reduce the mistakes in predicting examples during the learning process. Meanwhile, MCL-b produces more stable results in different runs than MUL and Random. Thus, MCL-b is a better learning strategy for solving the problems under the paradigm of learning actively and conservatively. However, MUL, which is considered to be a strong competitor to MCL-b, achieves the opposite in spite of its great learning efficiency (learning a good model with fewer examples). It confirms our expectation that traditional active learning strategies cannot solve the problems well.

After introducing MCL-b and the results of the empirical studies, we will present the implementation details of MCL-1 and its experimental results on both UCI datasets and a real-world application of direct marketing.

2.4 MCL-1 Learning Strategy

MCL-1 is designed to label (retrieve) examples of one class, such as the customers who will buy the product in direct marketing or the movies one likes to watch in movie selection. In the following subsection, we will introduce the details of implementing MCL-1, and present the experimental results on UCI datasets and one real application dataset respectively.

2.4.1 MCL-1 Implementation Details

Similar to MCL-b, MCL-1 selectively learns the example that can be labeled by the current model with the highest certainty in each iteration, and updates its model with all the labeled examples. However, as MCL-1 only cares about the examples of one class (to ease description, we assume the class is positive), the implementation techniques of MCL-1 are different from MCL-b on three aspects.

Firstly, MCL-1 chooses the most certain *positive* example to label. More specifically, MCL-1 chooses the example that satisfies $\arg\max(prob^+_{x_0}, \cdots, prob^+_{x_i}, \cdots, prob^+_{x_n})$, where $prob^+_{x_i}$ is the probability of labeling an example x_i ($x_i \in \mathcal{X}$) as positive by the current model. For instance, if two examples, x_1 and x_2, are predicted as negative with probability 0.8 and positive with 0.7 respectively, MCL-1 will select x_2 to predict, as it is more certain to be positive.

Secondly, MCL-1 chooses the example that has the shortest Euclidean distance to the *positive* center if all the labeled examples are of the same class. Suppose that we have one labeled example x_1 and two unlabeled y_u, which is the closest to x_1, and y_v, the furthest from x_1. If x_1 is negative, MCL-1 will choose y_v and predict it as positive; otherwise it will choose y_u and predict it as positive.

Lastly, what is the stopping criterion for MCL-1? Ideally a learner should stop learning when all the positive examples are retrieved. However, it is infeasible and impossible to know if the criterion is satisfied before all the examples are labeled. MCL-1 takes a similar but more practical criterion. It calculates the value of the F measure[4] after selecting each most positive example. The F measure can be calculated as follows.

$$Fmeasure = \frac{2 \times precision \times recall}{precision + recall} \tag{2.2}$$

If the value of the F measure reduces in three iterations successively (three-step lookahead strategy), MCL-1 stops learning. The idea is that if three most positive examples selected successively are negative, the positive examples are expected to be quite rare in unlabeled examples with high certainty, and MCL-1 stops learning.

[4] F measure is often used to evaluate the performance of Information Retrieval (IR) system [56, 93].

2.4.2 Empirical Studies of MCL-1 on UCI Datasets

MCL-1 is to retrieve positive examples from a dataset. Accordingly we modify the learning strategies, MUL and Random, so that they are comparable. More specifically, MUL selects the most uncertain positive example. That is, the example that is labeled as positive but with the probability closest to 0.5. Random selects positive examples randomly. That is, it predicts an example if the prediction is positive; otherwise it reselects another one.

Due to the different goals, the measurement for MCL-1 is also different from MCL-b. Firstly, MCL-1 uses the *F measure* to assess the performance of learning strategies, as the number of mistakes may not work well when only one class is cared about. For example, given a dataset including 20 positive examples, learning strategy A retrieves 10 examples as positive with 2 mistakes, and strategy B retrieves 20 examples with 4 mistakes. Strategy A would be better according to the number of mistakes. However, the measurement disagrees with the goal of MCL-1, i.e., retrieving more positive examples. On the other hand, F measure shows that strategy B is better, as the F measure value of B ($\frac{2\times0.8\times0.8}{0.8+0.8} = 0.8$) is greater than that of strategy A ($\frac{2\times0.8\times0.4}{0.8+0.4} = 0.54$). Secondly, similar to MCL-b, *volatility* is also an important measurement in this experiment. The difference is that the volatility here is the standard deviation of the F measure.

As mentioned, MCL-1 stops learning if its F measure decreases on three steps successively. However, this lookahead strategy is not proper for MUL and Random, as their F measure may still increase largely even after many steps of decrease. Thus, for them, the learning process continues until all the examples are learned, and the peak value of F measure is compared with MCL-1.

The 10 UCI [2] datasets used are the same as those introduced in Section 2.3.3. The comparison is based on two aspects, including the F measure and volatility[5].

First of all, we show the F measure of MCL-1, MUL and Random in Figure 2.15 on two datasets (due to similar results on all datasets), autos and sonar. In Figure 2.15, the curve of MCL-1 ends when the F measure reduces successively for three steps (without learning the rest); while the curves of the other two strategies end at the peak values (learned all the rest but without showing them). We can see clearly that the F measure of MCL-1 still is much higher than that of the other two strategies.

Secondly, we present the performance of MCL-1, MUL and Random in terms of the F measure on all the 10 datasets, and the results are presented in Figure 2.16. The x axis is the datasets and y axis is the average F-measure values of five runs on each dataset. It is clear that MCL-1 has much higher F measure value than MUL and Random. More specifically, the F measure value of MCL-1 is about 1.4 to 1.7 times as much as that of MUL and Random. The t-test results also confirm that MCL-1 wins the other two strategies on all the datasets without exception.

Finally, we present the volatility of the three learning strategies in Figure 2.17. The x and y axis indicate datasets and volatility respectively. It shows that MCL-1 is extremely stable on all the 10 datasets. Particularly, the volatilities MCL-1 on 5 datasets are almost zero. However, MUL is extremely volatile, even much more volatile than Random.

[5]We do not measure the learning strategies with learning efficiency as we do on MCL-b, as in this experiment we try to retrieve the examples of one class.

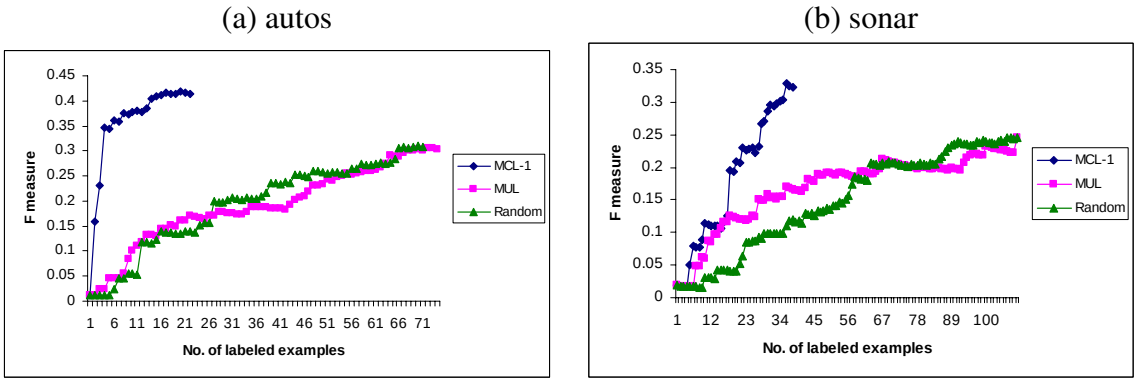


Figure 2.15: The F measure curves of MCL-1 (with three-step lookahead stopping criterion), MUL and Random.

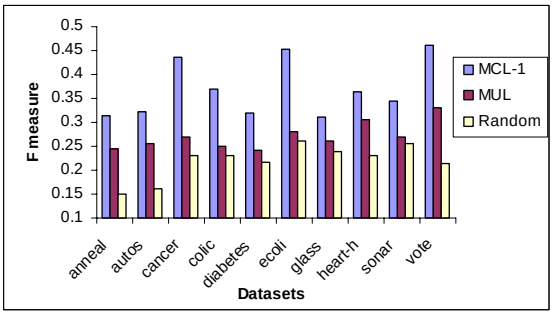


Figure 2.16: Comparison of the F measure on UCI datasets

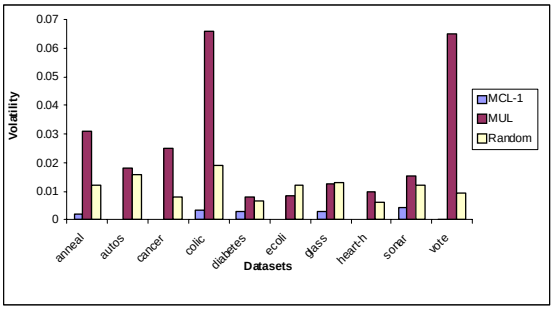


Figure 2.17: Volatility of the F measure on the 10 UCI datasets

The experiment on UCI [2] datasets show that MCL-1 performs better than the other learning strategies in solving the problems under the paradigm of learning actively and conservatively. Furthermore, the conservativeness of MCL-1 in example selection makes the learning process more stable and predictable.

2.4.3 Application in Direct Marketing

In addition to the UCI datasets, we also apply the learning strategy MCL-1 to an up-selling (marketing) application[6]. The up-selling dataset is from KDD-Cup 2009 (the small training set) [66], which has 50,000 examples and each example has 230 attributes. The first 190 attributes are numerical and the rest 40 are categorical, and plenty of missing values are included in the dataset. The class label is binary {+1, -1}, and "+1" indicates successful up-selling.

As in the up-selling application, a sales agent only cares about the customers (examples) of the one class (who will buy more expensive items), MCL-1 is suitable for this learning problem. For comparison, MUL and Random are also applied to the dataset. The experiment results in terms of the F measure are shown in Figure 2.18. The results are consistent with that on UCI datasets. In spite of the huge number of attributes and large number of missing values, MCL-1 still works significantly better than the other two learning strategies. Specifically, the F measure value of MCL-1 is about 35% higher than that of Random, and 25% higher than that of MUL.

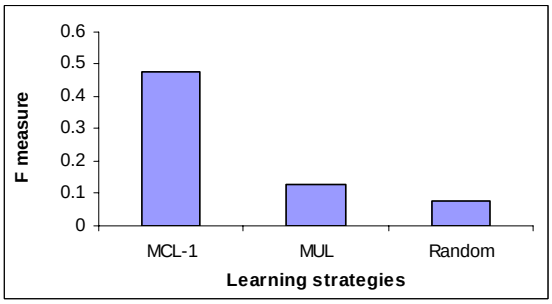


Figure 2.18: The F measure on the marketing data

However, the F measure value of MCL-1 is quite low (only about 0.12). To find out the reason, we check the precision and recall, and find that the recalls for MCL-1 and MUL are quite good, close to 0.8 and 0.7 respectively (0.47 for Random), but the precisions are extremely low, 0.14 for MCL-1, and 0.125 and 0.09 for MUL and Random respectively. The low precision is caused by the extremely imbalance (more than 92% of the examples are negative) of the up-selling dataset.

Note that our research is different from the competition of KDD Cup 2009. The competition is a traditional classification problem and concerns the prediction error rate of the final model and time efficiency. It is not for handling the problems under the learning paradigm proposed in this chapter.

[6] In up-selling, an agent only cares about the customers who probably will purchase more expensive items.

2.5 Summary

In this chapter, we proposed a new paradigm of learning actively and conservatively, which is similar to traditional active learning in terms of selecting examples actively to learn. However, its settings and goal are different from the traditional active learning. Under the paradigm of learning actively and conservatively, the selected unlabeled examples are predicted by the learner itself, and the true label is revealed after each prediction. The goal of the new paradigm is to minimize the number of mistakes for predicting unlabeled data during the iterative learning process, instead of the final models learned and the total number of labels needed in traditional active learning.

To solve the learning problems under the new paradigm, we propose an effective learning algorithm MCL. MCL iteratively selects the most certain examples to learn during the learning process, such that the number of mistakes the learner makes in predicting unlabeled examples is minimized. For diverse real-world applications, we further implement binary-MCL (MCL-b) for the problems that care about the examples of both classes and single-MCL (MCL-1) for the ones that care about only one class.

In the empirical studies, two other learning strategies are implemented for comparing with MCL-b and MCL-1 respectively. For MCL-b, the experiments were conducted on the educational game Zoombinis and 10 UCI datasets. For MCL-1, the experiments were conducted on the same 10 UCI datasets and one real-application dataset. The experimental results show that both MCL-b and MCL-1 achieve their goals successfully on all the different datasets and applications.

Chapter 3

Learning Actively and Aggressively (for Label Quality)

To the opposite of the conservative paradigm in Chapter 2, the paradigm of *learning actively and aggressively* proposed in this chapter is aggressive. It allows learners to actively select the challenging examples to learn. Its goal is to learn a model the fastest, i.e., to build a good model with as few labeled examples as possible. Traditional active learning algorithms with uncertain sampling strategy are under this paradigm as they usually select the example that is predicted with the least certainty (i.e., challenging) to query an oracle and improve the current model efficiently. However, traditional active learning algorithms cannot solve the problems under this paradigm very well, as oracles are usually imperfect and the noisy labels of examples can deteriorate the learning performances badly.

To rule out the negative effects of the noisy labels, querying multiple oracles has been proposed in active learning [82, 90, 22]. This multiple-oracle strategy is reasonable and useful in improving label quality. For example, in paper reviewing, multiple reviewers (i.e., oracles or labelers) are requested to label a paper (as accepted, weak accepted, weak rejected or rejected), so that the final decision (i.e., label) can be as accurate as possible.

However, there is still no way to guarantee the label quality in spite of the improvements obtained in previous works [90, 22, 72]. Furthermore, strong assumptions, such as even distribution of noise [90], and example-independent (fixed) noise level [22], have been made. These assumptions, in the paper reviewing example mentioned above, imply that all the reviewers are at the same level of expertise and each reviewer has the same probability in making mistakes on the papers of different topics.

Obviously, these assumptions may be too strong and not realistic, as it is ubiquitous that label quality (or noise-level) is example-dependent in real-world data. In the paper reviewing example, the quality of a label given by a reviewer should depend heavily on how close the reviewer's research is to the topic of the paper. The closer it is, the higher quality the label has.

In this chapter, to guarantee the label quality given that the noise level is example-dependent, we extend the settings of the aggressive paradigm as follows. Learners are allowed to query multiple imperfect oracles which are able to return both labels and confidences in the labels. As mentioned in Chapter 1, these extended settings exist commonly in real-world. Taking the paper reviewing example again, usually a reviewer is required to provide not only a label (accept, weak accept, weak reject or reject) for a paper, but also his confidence (high, medium or

low) for the labeling. Similar situation also exists in webpages labeling, image labeling and so on. Under the new settings, the quality of final decisions (labels) can be estimated easily.

To guarantee the label quality under the paradigm with extended settings, we propose a new active learning strategy, called *c-certainty learning*. For a given example, the c-certainty learner assesses its label quality every time after obtaining a response from an oracle. If the label quality is higher than or equal to a given threshold c (c is the probability of correct labeling; see later), the example and its label will be added to labeled data; otherwise, more oracles will be queried. In the paper reviewing example, with the labels and confidences given by reviewers (oracles), we can estimate the certainty of the label. If the certainty is too low (e.g., lower than a given c), another reviewer has to be sought to review the paper to improve the label quality.

In addition, as in our work the noise level of each oracle is allowed to be example-dependent, it would be more difficult for a learner to select good oracles to query than the previous work assuming that the noise level is example-independent. We propose an effective learning algorithm that is able to select the *Best Multiple Oracles* to query (called *BMO*) for each given example. With BMO, the number of queries for an example to meet the threshold c is expected to be reduced. This is crucial, as it indicates that, for a given query budget, BMO is expected to obtain more examples with labels of high quality compared to selecting oracles randomly. Consequently more accurate models can be built. The work in this chapter is published in *The 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining. May, 2012* [64].

This chapter is organized as follows. We review related work in Section 3.1. Section 3.2 introduces confidence of labeling, and the calculation of certainty. We present our learning algorithm BMO in Section 3.3 and the experimental results in Section 5.5. We summarize the work of this chapter in Section 3.5.

3.1 Related Works

Labeling each example with multiple oracles has been studied when labeling is not perfect in supervised learning [72, 94, 96]. Some principled probabilistic solutions have been proposed on how to learn and evaluate the multiple-oracle problem. However, as far as we know, none of them can guarantee the label quality to satisfy a given threshold c, which can be guaranteed in our work.

Other recent works related to multiple oracles make some assumptions which may be too strong and unrealistic. One assumption is that the noise of oracles is equally distributed [90]. The other type of assumption is that the noise level of different oracles are different as long as they do not change over time [22, 110]. Their works estimate the noise level of different oracles during the learning process and prefer querying the oracles with low noise levels. However, it is ubiquitous that the quality of an oracle is example-dependent. In this paper, we remove all the assumptions and allow the noise level of oracles to vary among different examples.

Active learning on the data with example-dependent noise level was studied in [25]. However, it focuses on how to choose examples considering the tradeoff between more informative examples and examples with lower noise level. In our work, we focus on how to use multiple oracles to get labels with high quality.

3.2 C-Certainty Labeling Quality

Under the paradigm of learning actively and aggressively, oracles are able to return both labels and the confidences in the labels. For the study of this learning paradigm, we define *confidence* formally first here. *Confidence* for labeling an example x is the probability that the label given by an oracle is the same as the true label of x. We assume that the confidences of oracles on any example are greater than 0.5[1].

According to the labels and confidences given by oracles, we guarantee the label certainty of each example to be greater than or equal to c ($c \in (0.5, 1]$) (called *c-certainty labeling*). That is, a label is valid if its certainty is greater than or equal to c. Otherwise, more queries would be issued to different oracles to improve the label certainty until the threshold c is met.

However, given a label certainty $C(T_P|A^{n-1})$ (lower than c) of an example x, how can we update it after obtaining a new answer from an oracle? Let the set of previous $n-1$ answers be A^{n-1}, and the new answer be A_n in the form of (P, f_n) which indicates that the confidence of labeling x to be positive (P) is f_n. According to Bayes rule, the update of the label certainty $C(T_P|A^n)$ can be calculated with Formula 3.1 (See Appendix A.1 for the details of the derivation).

$$C(T_P|A^n) = \begin{cases} \frac{p(T_P)\times f_n}{p(T_P)\times f_n + p(T_N)\times(1-f_n)}, & \text{if } n = 1 \text{ and } A_n = \{P, f_n\} \\ \frac{C(T_P|A^{n-1})\times f_n}{C(T_P|A^{n-1})\times f_n + (1-C(T_P|A^{n-1}))\times(1-f_n)}, & \text{if } n > 1 \text{ and } A_n = \{P, f_n\}, \end{cases} \tag{3.1}$$

where T_P and T_N are the true positive and negative label respectively. Formula 3.1 can be applied directly when A_n is positive (i.e., $A_n = \{P, f_n\}$); while for a negative answer, we can transform it as $A_n = \{N, f_n\} = \{P, (1-f_n)\}$ such that Formula 3.1 is also applicable. In addition, Formula 3.1 is for calculating the certainty of x to be positive. If $C(T_P|A^n) > 0.5$, the label of x is positive; otherwise, the label is negative and the certainty is $1 - C(T_P|A^n)$. With Formula 3.1, the label certainty of x can be updated easily by querying oracles repeatedly until $max(C(T_P|A^n), 1 - C(T_P|A^n))$ is greater than or equal to c.

However, according to Formula 3.1, the certainty $C(T_P|A^n)$ is not monotonic (See appendix A.2 for the proof). This can be explained intuitively. For example, in paper reviewing, if the labels given by reviewers are alternating between positive and negative, the certainty may not be able to reach the threshold c even many reviewers are requested. That is, it is possible that the certainty dangles around and is always lower than c.

To guarantee that the threshold c can be reached with as few queries as possible, we will propose an effective learning algorithm to select the best oracles to query.

3.3 BMO (Best-Multiple-Oracle) with C-Certainty

As the capability of each oracle varies in labeling different examples, the key issue for improving the querying efficiency is to select the best oracle for *every* given example. However,

[1]This assumption is reasonable, as usually oracles can label examples more correctly than assigning labels randomly.

the varying capability makes it difficult to estimate the performance of an oracle in labeling a given example, and more difficult than the case where the capability of each oracle is evenly-distributed [22, 110].

3.3.1 Selecting the Best Oracle

How can we select the best oracle given that the noise levels are example-dependent? The basic idea is that an oracle can probably label an example x with high confidence if it has labeled x_j confidently and x_j is close to x (See Figure 3.1).

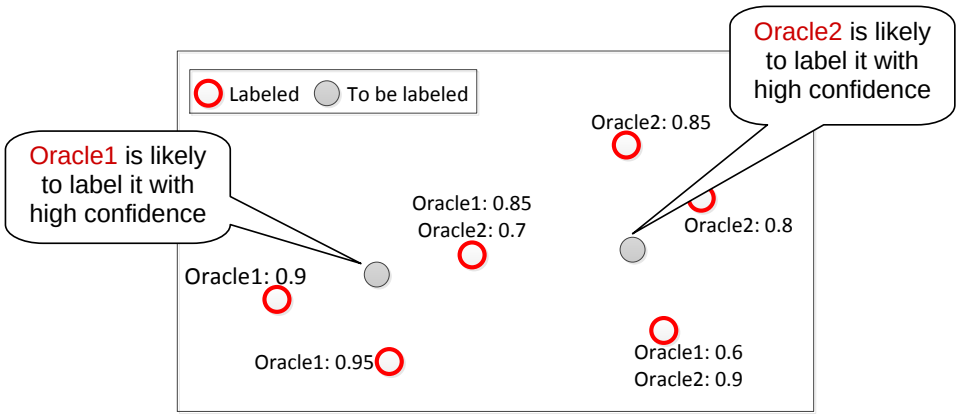


Figure 3.1: The basic idea for selecting the best oracle. The decimals in the figure indicate the labeling confidence.

More specifically, we assume that each of the m oracle candidates $(O_1, \cdots, O_m)$ has labeled a set of examples $\mathcal{E}_i$ $(1 \leq i \leq m)$. E_{k_i} $(1 \leq i \leq m)$ is the set of k ($k = 3$ in our experiment) nearest neighbors of x in $\mathcal{E}_i$ $(1 \leq i \leq m)$. BMO chooses the oracle O_i such that the examples in E_{k_i} are of the highest confidence. The potential confidence for each oracle in labeling x can be calculated with Formula 3.2.

$$P_{c_i} = \frac{\frac{1}{k} \times \sum_{j=1}^{k} f_{x_j}^{o_i}}{1 + \frac{1}{k} \times \sum_{j=1}^{k} |x - x_j|}, \tag{3.2}$$

where $x_j \in E_{k_i}$, $f_{x_j}^{o_i}$ is the confidence of oracle O_i in labeling x_j, and $|x - x_j|$ is the Euclidean distance between x_j and x. The numerator of Formula 3.2 is the average confidence of the k nearest neighbors of x. The last item in the denominator is the average distance, and the 1 is added to prevent the denominator from being zero.

High confidence in labeling x's nearest neighbors indicates that the oracle O_i will likely label x with a high confidence. That is, BMO selects the oracle O_i if $i = \arg\max_i(P_{c_1}, \cdots, P_{c_i}, \cdots, P_{c_m})$. This strategy of selecting the best oracle is reasonable as the confidence distribution (expertise level) of oracles is usually continuous, and does not change abruptly.

3.3.2 Active Learning Process of BMO

BMO is a wrapper learning algorithm, and it treats the strategy of selecting examples to label as a black box. Any existing query strategies in active learning, such as uncertainty sampling [54], expected error reduction [76] and the density-weighted [87] method can be fit in easily.

At a high level, BMO works as follows. For an example x_i ($x_i \in \mathcal{E}_u$) selected by an example-selecting strategy (e.g.,uncertain sampling), BMO selects the best oracle among the ones that have not been queried for x_i yet to query, and updates the label certainty of x_i with Formula 3.1. This process repeats until the certainty meets the threshold c. Then BMO adds x_i into its labeled example set $\mathcal{E}_l$. This labeling process continues until certain stopping criterion is met.

More specifically, as usually active learning starts with a small set of initial data, BMO has two learning phases: exploitation in initial training data and exploration in unlabeled data. As noise in the initial training data may lead the future active learning process to a wrong track, BMO starts with the exploitation phase.

- **Exploitation in initial Data** Given initial training data $\mathcal{E}_I$, two situations may be encountered. One is that only labels of examples are provided but without confidence. BMO takes the leave-one-out strategy to check if more queries are needed for each example. More specifically, for each example $x_i \in \mathcal{E}_I$, BMO builds a classifier with the set of examples except x_i. If the current label of x_i agrees with the prediction of the classifier, BMO treats it as certain enough. Otherwise, BMO will select the best oracle according to Section 3.3.1 to query and update the certainty with Formula 3.1 until the certainty satisfies the given c, and then x_i is added into labeled example set $\mathcal{E}_l$ (See Line 1 to Line 20 in Algorithm 2). The other situation is that the initial training data $\mathcal{E}_I$ have both labels and confidence. This is the same as exploring in unlabeled data which will be introduced in the following except that one oracle has already been queried.

- **Exploration in Unlabeled Data** Given an unlabeled dataset $\mathcal{E}_u$, an example x_i ($x_i \in \mathcal{E}_u$) can be selected by any example-selecting strategy (such as uncertain sampling). Then BMO selects the best oracle O_i ($O_i \in O_1, \cdots, O_m$) according to Section 3.3.1, and posts the query to it and updates the label certainty of x_i with Formula 3.1 (See Line 21 to Line 39 in Algorithm 2). This process repeats until the certainty is greater than or equal to c. BMO updates its labeled example set $\mathcal{E}_l$ by adding in x_i. This process repeats until a given query budget is used up (such as, the predefined query budget, such as $50, 100, \cdots, 500$, is used up in our experiment).

By selecting the best oracle iteratively, BMO can improve the label certainty of a given example to meet the threshold c with only a few queries (See Section 5.5). That is, BMO is expected to get as many examples with guaranteed label quality as possible for a given query budget, and the model built from the examples is expected to have better performance than querying oracles randomly.

It is not trivial that our strategy will outperform other learning strategies, as the number of distinct examples labeled would be much less, though each would have a higher "quality". In the next subsection, we will empirically study the performances of BMO.

Algorithm 2: BMO (Best Multiple Oracles)

Input: Initial training data set $\mathcal{E}_I$; Unlabeled data: $\mathcal{E}_u$; oracles: O; oracles queried: O_q; threshold: c; queries budge: *budget*;

Output: labeled example set $\mathcal{E}_l$

```
1  begin
2      for each x_i ∈ E_I do
3          //Exploitation in labeled data
4          if budget > 0 then
5              E_I = E_i − x_i
6              Build a classifier with E_I ∪ E_l;
7              if label of x_i ≠ prediction given by the classifier then
8                  while certainty < c do
9                      for each O_i ∈ (O − O_q) do
10                         P_{c_i} ← Formula 3.2;
11                     end
12                     O_m ← oracle with maximal P_c;
13                     certainty ← update with Formula 3.1;
14                     O_q ← O_q ∪ O_m;
15                     budget ← budget − 1;
16                 end
17             end
18         end
19         E_l ← E_l ∪ x_i;
20     end
21     while budget > 0 do
22         //Exploration in unlabeled data
23         x_i ← selection with uncertain sampling (x_i ∈ E_u);
24         O_q ← null;
25         certainty ← 0;
26         while certainty < c do
27             for each O_i ∈ (O − O_q) do
28                 P_{c_i} ←Formula 3.2;
29             end
30             O_m ← oracle with maximal P_c;
31             certainty ← update with Formula 3.1;
32             O_q ← O_q ∪ O_m;
33             budget ← budget − 1;
34         end
35         E_l ← E_l ∪ x_i;
36         update the current model;
37     end
38     Return E_l;
39 end
```

3.4 Experiments

In our experiment, to compare with BMO, we implement two other learning strategies. One is *Random selection of Multiple Oracles* (RMO). Rather than selecting the best oracle in BMO, RMO selects oracles randomly to query for a given example and repeats until the label certainty is greater than or equal to c. The other strategy is *Random selection of Single Oracle* (RSO). RSO queries oracle only once for each example without considering c. RSO essentially is the traditional active learning algorithm.

As RSO only queries one oracle for each example, it is expected to have the most labeled examples for a predefined query budget but with the highest noise level. To reduce the negative effect of noisy labels, we *weight* all labeled examples according to their label certainty when building final models. To make all the three strategies comparable, we also use weighting in BMO and RMO. In addition, all the three algorithms take uncertain sampling as the example-selecting strategy and decision tree (J48 in WEKA [104]) as their base learners. The implementation is based on the WEKA source code.

The experiment is conducted on UCI datasets [2] including abolone, anneal, cmc_new, credit, mushroom, spambase and splice, which are commonly used in the supervised learning research. The number of attributes of the 7 datasets varies from 9 to 61 and the number of examples varies from around 1,000 to 8,000. Each dataset is split into 70% for training and 30% for testing. The initial training dataset is of size 10, and with 20% - 30% noise in labels and no confidence is given. In addition, as the number of oracles cannot be infinite in real world, we only use 10 oracles for each dataset. If an example has been presented to all the 10 oracles, the label and the certainty obtained will be taken in directly. The threshold c is predefined to be 0.8 and 0.9 respectively. The experimental results presented are the average of 10 runs, and t-test results are of 95% confidence.

In our previous discussion, we consider the confidence given by an oracle as the true confidence. However, in real life, oracles may overestimate or underestimate themselves intentionally or unintentionally. If the confidence given by an oracle O does not equal the true confidence, we call O *an unfaithful oracle*; otherwise, it is faithful. To observe the robustness of our algorithm, we conduct our empirical studies with both faithful and unfaithful oracles[2] in the following.

3.4.1 Results on Faithful Oracles

As no oracle is provided for the UCI datasets, we build 10 faithful oracles for each dataset. Each oracle is generated as follows. Firstly, we select one example x randomly as an "expertise center" and label it with the highest confidence. Then, to make the oracle faithful, we calculate the Euclidean distance from each of the remaining examples to x, and assign them confidences based on the distances. The further the distance is, the lower confidence the oracle has in labeling the example. Noise is added into labels accordingly. Thus the oracle is faithful.

The confidence of an oracle in labeling examples is supposed to follow certain distribution. We choose three common distributions, linear, normal and mixture models of two normal

[2]Actually it is difficult to model the behaviors of unfaithful oracles with a large confidence deviation. In our experiment, we show that our algorithm works well given unfaithful oracles slightly deviating from the true confidence.

distributions. Linear distribution assumes the confidence reduces linearly as the distance increases. For normal distribution, the reduction of confidence follows the probability density function $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{x^2}{2\sigma^2}) - 0.55$. A mixture model of two normal distributions indicates that the oracle has two "expertise centers" (see Figure 3.2). As mentioned earlier 10 oracles are generated for each dataset, and three of them follow the linear distribution, three the normal distribution and four the mixture models of two normal distributions.

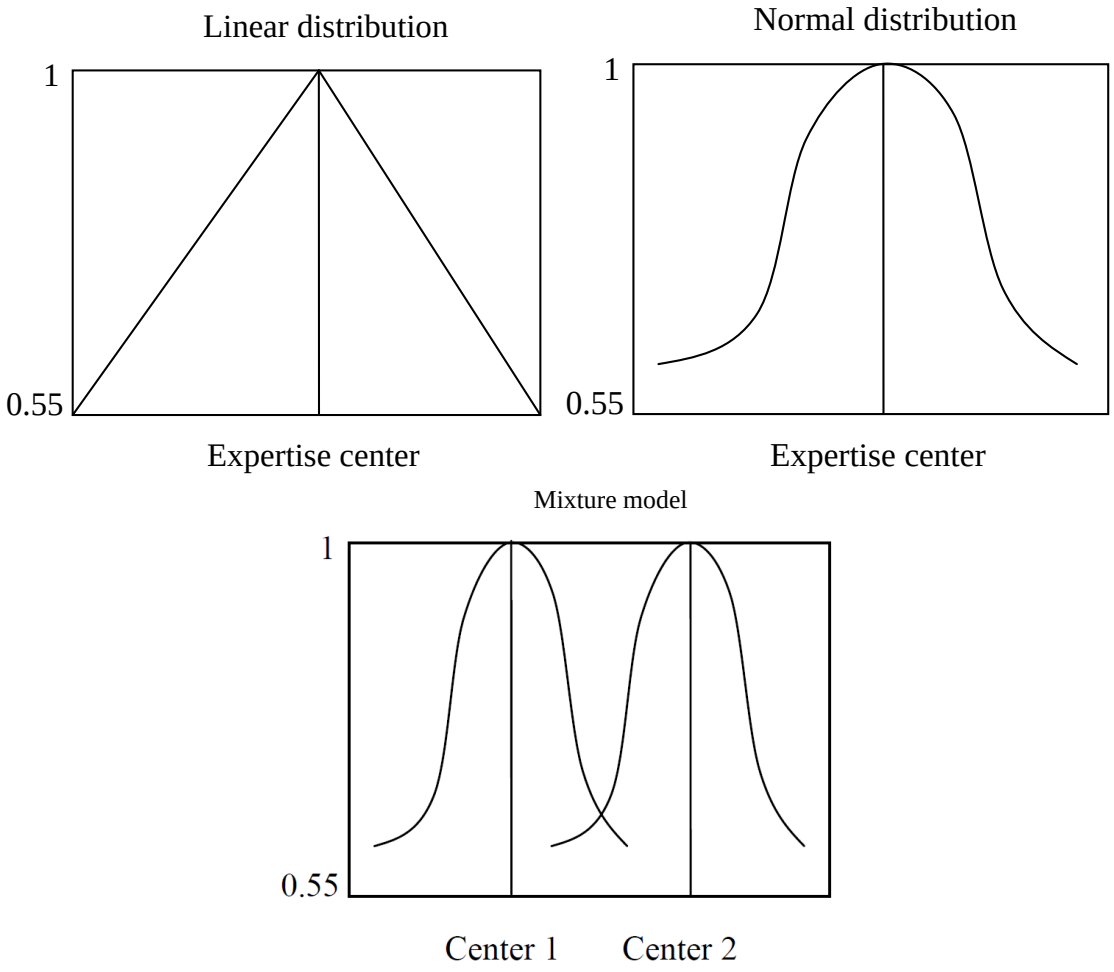


Figure 3.2: Three distributions

Firstly, we present the prediction error rate of the models built with different query budgets. Due to the similar results of different datasets, we only show the details of one dataset (anneal) in Figure 3.3 and a summary of the comparison afterwards. Figure 3.3 shows the prediction error rates of BMO, RMO and RSO for the threshold 0.8 (left) and 0.9 (right) respectively. The x axis indicates the query budgets while the y axis represents the error rate on test data. On one hand, as we expected that, for both thresholds 0.8 and 0.9, the error rate of BMO is much lower than that of RMO and RSO for all different budgets, and the performances of the latter two are similar. On the other hand, the curve of RMO when $c = 0.8$ is not as smooth as the other ones.

Secondly, why are the performances of the three learners different? It can be explained by

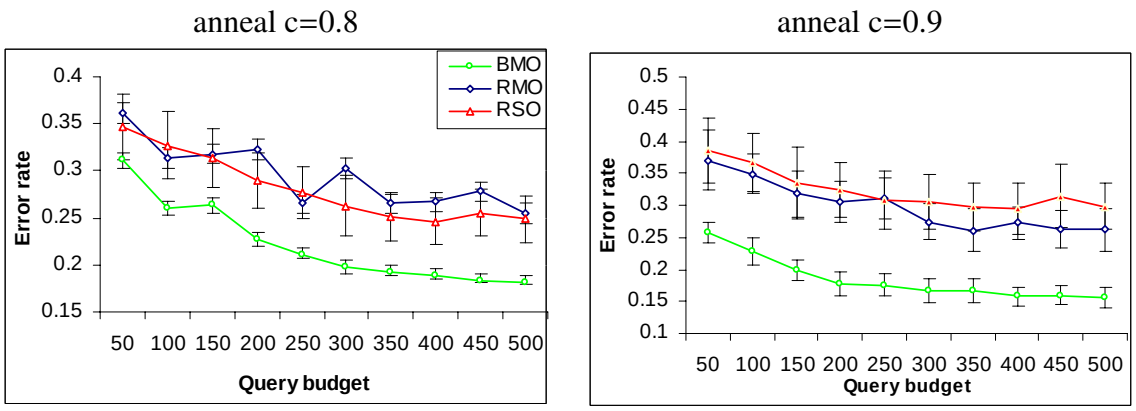


Figure 3.3: Error rate on faithful oracles

two factors, the noise level and the number of examples. Due to similarity, we only show how the two factors affect the performances through one dataset (anneal) when the query budget is 500 in Figure 3.4.

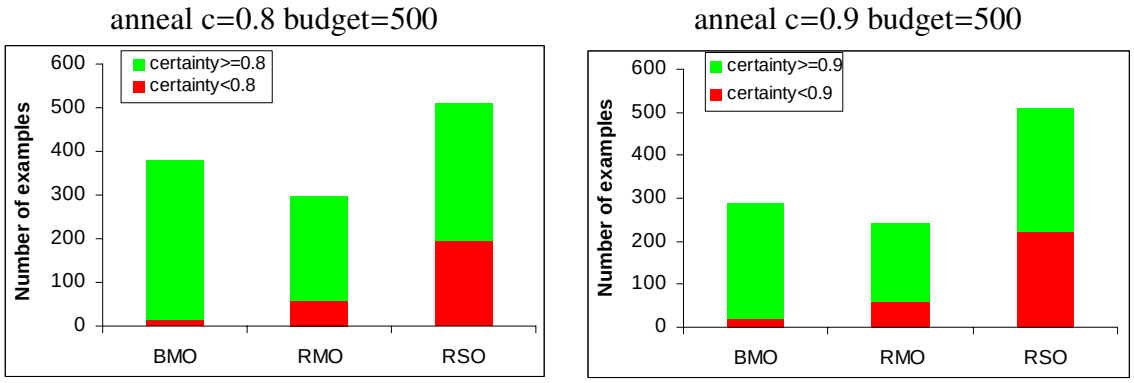


Figure 3.4: The number of examples and label quality of faithful oracles

Figure 3.4 shows that on average BMO only queries about 1.4 ($c = 0.8$) and 1.7 ($c = 0.9$) oracles for each example; while RMO queries more oracles (1.7 and 2.0). That is, BMO obtains more labeled examples than RMO for a given budget. Moreover, the examples labeled by BMO have much higher label certainty than that by RMO[3]. On the other hand, the examples labeled by RSO is much more noisy than BMO (i.e., the red portion is much larger). It is the noise that deteriorates the performance of RSO. Thus, we can see clearly that BMO outperforms the other two strategies because of its guaranteed label quality and the selection of the best oracles to query.

By looking closely into the curves in Figure 3.3, we find that the curve of RMO when $c = 0.8$ is not as smooth as the other ones. The reason is that RMO of $c = 0.8$ has fewer labeled examples when compared to BMO and RSO of $c = 0.8$ and has more noise when compared to

[3]Some of the examples still have certainty lower than c due to the limited oracles in our experiment.

Table 3.1: T-test results on 7 datasets with 10 different budgets

	BMO vs. RMO		BMO vs. RSO		RMO vs. RSO	
	c=0.8	c=0.9	c=0.8	c=0.9	c=0.8	c=0.9
Win	43	51	40	42	0	9
Draw	27	19	30	38	70	51
Lose	0	0	0	0	0	10

that of $c = 0.9$. Fewer examples make the model learnt more sensitive to the quality of each label; while the label quality of $c = 0.8$ is not high enough. Thus, the stability of RMO when $c = 0.8$ is weakened.

Lastly, we will show the t-test results in terms of the error rate on all the seven UCI datasets. As for each dataset 10 different query budgets are considered, the total times of t-test for each group is 70. As shown in Table 3.1, BMO wins RMO 94 times out of 140 ($c = 0.8$ and $c = 0.9$) and wins RSO 86 out of 140 without losing once. It is clear that BMO outperforms RMO and RSO significantly.

In summary, by querying faithful oracles, c-certainty learning is able to guarantee the label quality. Furthermore, the experimental results show that BMO can build a better model than RMO by selecting the best oracles to query. In addition, the results of RSO illustrate that weighting with the label quality may reduce the negative influence of noisy label but still its effect is limited.

3.4.2 Results on Unfaithful Oracles

For various reasons, unfaithful oracles exist widely in real world. To present the robustness of our learning algorithm, we build unfaithful oracles for extensive empirical studies. In this experiment, unfaithful oracles are generated for each dataset by building models over 20% of the examples. More specifically, to generate an unfaithful oracle, we randomly select one example x as an "expertise center", and sample examples around it. The closer an example x_i is to x, the higher the probability it will be sampled with. In this way, the oracle built on the sampled examples can label the examples closer to x with higher confidences. The sampling probability follows exactly the same distribution in Figure 3.2. For each data set, 10 oracles are generated and three follow the linear distribution, three the normal distribution and four the mixture models of two normal distributions.

As sampling rate declines with the increasing distance, the oracle built may fail to give true confidence for the examples that are far from the "center". As a result, the oracle is unfaithful. That is, the oracles are unfaithful due to "insufficient knowledge" rather than "lying" deliberately.

We run BMO, RMO and RSO on the seven UCI datasets and show the prediction error rates and the number of labeled examples on one dataset (anneal) in Figure 3.5 and Figure 3.6 respectively, and a summary on all the datasets afterwards. It is surprising that the performances of BMO on unfaithful oracles are similar to that on faithful oracles. In particular, the error rate of BMO is much less than that of RMO and RSO, and the latter two are similar. The examples labeled by BMO are more than the number labeled by RMO. Furthermore, the label quality of RMO is higher than that of both RMO and RSO, which is also similar to that on faithful

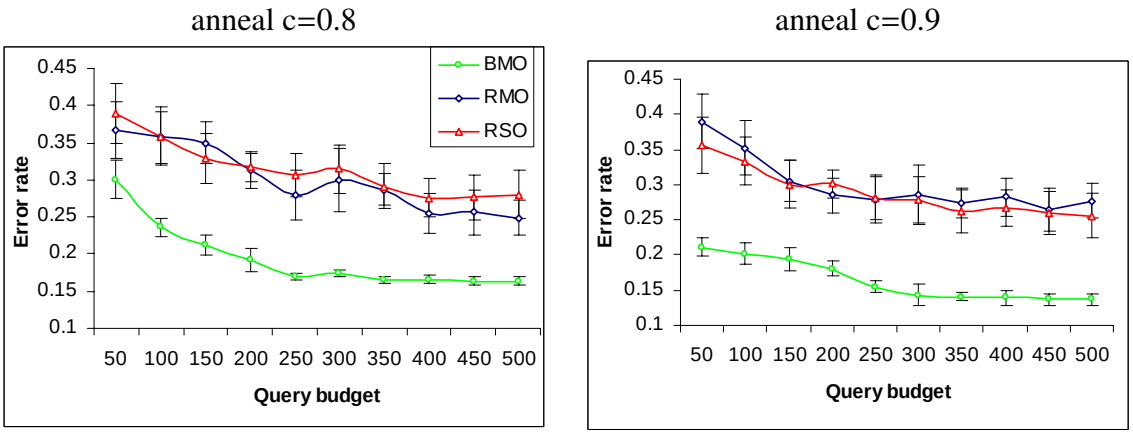


Figure 3.5: Error rate on unfaithful oracles

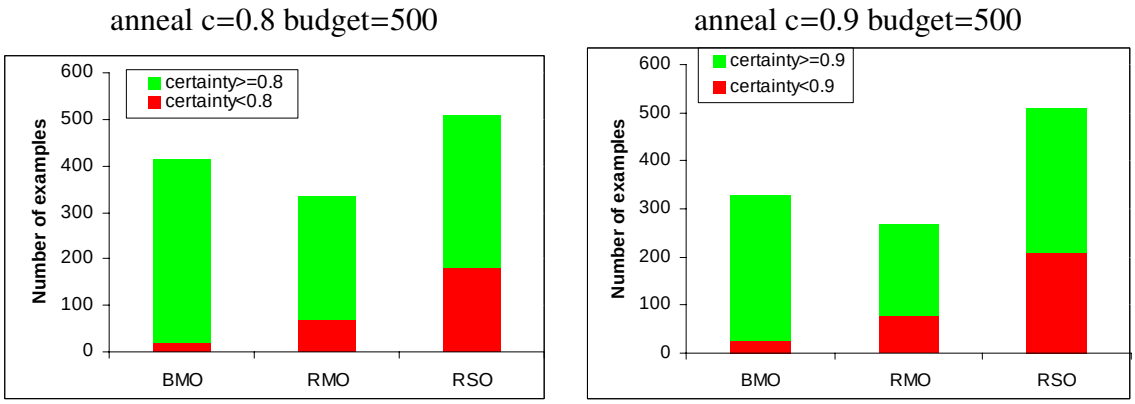


Figure 3.6: The number of examples and label quality of unfaithful oracles

Table 3.2: T Test results for all datasets and budgets on unfaithful oracles.

	BMO vs. RMO		BMO vs. RSO		RMO vs. RSO	
	c=0.8	c=0.9	c=0.8	c=0.9	c=0.8	c=0.9
Win	53	42	53	45	12	0
Draw	6	22	17	15	50	50
Lose	11	8	0	10	8	20

oracles.

The comparison results show clearly that BMO is robust for unfaithful oracles. The reason is that BMO selects the best multiple oracles to query, and it is unlikely that all the best oracles are unfaithful at the same time as our unfaithful oracles do not "lie" deliberately as mentioned.

Table 3.2 shows the t-test results on 10 different query budgets for all the seven UCI datasets. We can see that BMO wins RMO 95 times out of 140 and wins RSO 98 out of 140, which indicates that BMO works significantly better than RMO and RSO under most of the circumstances. However, BMO loses to RMO 19 times and RSO 10 times, which are different from the results on faithful oracles. Thus, even though BMO is robust, still it works slightly worse on unfaithful oracles than on faithful ones.

In summary, BMO is robust for working with unfaithful oracles, even though its performance may be reduced slightly. This property is crucial for the success of BMO in real-world applications, as unfaithful oracles are ubiquitous.

3.5 Summary

Learning actively and aggressively is a paradigm that can be applied to many learning problems in real applications. It allows learners to select examples actively to learn and its goal is to learn a good model from as few labeled examples as possible. Traditional active learning with uncertain sampling is under this paradigm. However, due to the labeling noise from oracles, existing active learning algorithms cannot solve the problems under this paradigm well.

In this chapter, we extend the settings of the learning paradigm based on the fact that usually multiple oracles are available and most of them are able to provide not only labels but also their confidences in the labels. According to the labels and confidences, we proposed c-certainty learning to guarantee the label quality of every example to meet the given threshold c. In addition, we allow the noise level to be example-dependent. To improve the query efficiency, we propose the learning algorithm BMO to select the best oracles to query such that the threshold c can be met with as few queries as possible.

As in real-world applications, oracles can be faithful or unfaithful, we conduct empirical studies on both faithful and unfaithful oracles. The results show that BMO works robustly and outperforms other active learning strategies significantly on both types of oracles, even though its performance can be affected slightly by unfaithful oracles.

Chapter 4

Learning Actively with Conservative-Aggressive Tradeoff

Under the paradigm of learning actively and conservatively presented in Chapter 2, learners always predict unlabeled examples with its model during the learning process. On the other hand, in Chapter 3 learners always query oracles for labels under the paradigm of learning actively and aggressively. The two opposite paradigms use two totally different ways, predicting and querying, respectively to obtain labels and work successfully for solving many real-world learning problems.

However, the two ways for obtaining labels, making predictions and querying oracles, are not mutual exclusive. Actually, they can be used together during the learning process and complement each other in many applications. For example, when letters are sorted by using OCR (optical character recognition) devices of the post office, if the hand-written zip codes are ambiguous or too difficult to recognize, they will be passed to the oracles (human) for labels (recognizing). However, if the OCR can predict accurately the hand-written zip codes, the letter will be sorted and mailed to the recipient directly. If the prediction is not correct, the letter will be returned and redelivered (cost or consequence of the wrong prediction). As another example, a company is training an automatic online advertising system to predict which website will be profitable to place advertisements. If the system is not sure whether it is suitable to advertise on a website, the company can pay an oracle (expert) for the label; while if the system is pretty certain that a website is profitable to advertise, then an advertisement slot can be bought directly on the website. If the advertisement is clicked often, the label of the website is positive; otherwise, it is negative and incurs cost as well.

In both examples above, the two actions, making prediction (e.g., directly mailing the letter and buying an advertisement slot) and querying oracle (e.g., human and expert), contribute to obtaining labels for different circumstances. Meanwhile the acquired labels can be given to the learner (OCR or Ads system) for further training iteratively. The key issue is how to control the iterative learning process and take correct actions (predicting or querying) for each example so that the cost for obtaining labels can be minimized.

In this chapter, we propose a new paradigm of *learning actively with conservative-aggressive tradeoff* for the type of problems discussed above. The settings for the paradigm are as follows. Firstly, learners are allowed to select examples actively to learn and the labels of examples are unknown. Secondly, to obtain the labels, two actions can be considered: making prediction

(e.g., predicting mails by OCR directly) and querying oracle (e.g., asking human and expert). Lastly, cost has to be paid for making wrong predictions or for querying oracles.

The goal for the paradigm is to build a good model by paying minimal cost for obtaining labels (predicting and querying). To achieve this goal, each decision on choosing which action to take has to be optimal during learning processes. Optimal decisions should depend on the expected cost C_e (the production of the cost for misclassifying an example and the probability that the mistake will happen) for making a wrong prediction and the cost for query an oracle C_q. Given an example, if its $C_e > C_q$, the learner will choose to get the label by querying an oracle; otherwise, the learner will choose to make prediction on it directly. The selection of the two actions is optimal (i.e., obtaining the labels with minimal total cost) if the expected cost C_e can always be estimated accurately.

However, the expected cost C_e usually is not accurate particularly in the beginning of the learning process as the model is not good enough. As a result, wrong (nonoptimal) actions are likely to be taken, and consequently high cost incurs. To tackle the problems under the paradigm of learning actively with conservative-aggressive tradeoff, we propose a novel learning algorithm *Decisive Active Learner* (*DAL*). DAL always selects the example likely leading to correct (optimal) actions and prefers the action that is expected to cost less during the learning process. The examples that will likely lead to wrong actions will be learned during the later stage, as the learner may become more reliable (the probability is more accurate) and the action taken will become more accurate with more example being learned. This work is submitted to *The IEEE International Conference on Data Mining (ICDM), 2012*.

The rest of this chapter is organized as follows. In Section 4.1, we will review some related work and discuss the difference with our work. In Section 4.2, we will introduce some preliminary for our problem setting and the new concept of decisive action. Section 4.3 will talk about the proposed algorithm to select actions. In Section 4.4, we will experimentally compare our algorithm with other typical learning algorithms. Summarization of the work in this chapter will be presented in Section 4.5.

4.1 Related Works

The learning algorithm DAL under the paradigm of learning actively with conservative-aggressive tradeoff is bridging between the traditional active learning and classification with rejection. It is also similar to two-oracle setting in active learning. In this section, we will discuss the similarities and differences with them.

As we mentioned in the previous section, traditional active learning has only one option to obtain the true labels, which is to query oracle. Most of the previous works assume that there exists at least one oracle who can provide the labels of the examples. Some previous works [102, 88] assume that there is one perfect oracle who can correctly give all labels. Other works [89, 95, 28]study assumes that the oracle is noisy and may mislabel the examples. Some existing works [23, 107] explore the case where multiple oracles or labelers may contribute to the quality of the labels. We can see in those works querying oracles (human experts or labelers) has been regarded as the only approach to retrieve the true labels.

Under the paradigm proposed in this chapter, we can have two options (actions) to acquire the true labels. Besides querying an oracle, the learner can make predictions directly and the

true label will be revealed from the feedback (success or failure). Thus, we need to consider not only the cost to query an oracle but also the cost of the wrong predictions. The best learning strategy might not be always selecting the most informative (or uncertain) examples as in the traditional active learning. Our goal is to explore the best learning sequence that minimizes the total cost under this new setting.

For the classifiers with the reject option (also known as abstaining classifiers), the learner can reject to make predictions on the uncertain examples. The decision when to reject to make prediction also relies on the ratio between the misclassification cost and the reject cost. However, some related works [33, 31, 51] study how to effectively reduce the misclassification rate without considering the cost ratio. Others [68, 6] do take the misclassification cost and the reject cost into consideration. Du and Ling [27] study this problem by considering different misclassification costs of false positive and false negative, and solving them by using two types of learning algorithms (thresholding and sampling). However, the existing works on classifiers with reject option only study how to minimize the cost given a predictive model, while in DAL we care more about the learning process that builds the predictive model with the minimal total cost.

Under our paradigm, since making predictions can reveal the true labels, it can be regarded as another oracle. However, our paradigm is substantially different from the two-oracle setting [23] in active learning. In our paradigm, the "oracle" (the model for making predictions) is updated with each new labeled example, while the two oracles in the two-oracle setting are static.

4.2 Preliminary

Before presenting the details of DAL learning algorithm, we will formally define the learning problem under the paradigm of learning actively with conservative-aggressive tradeoff, the learning goal and the concepts to be used in detail.

4.2.1 Problem Definition

Given a set of labeled data L, a set of unlabeled data $\mathcal{D}_U$ and a learner M learned from L, M is allowed to select examples from $\mathcal{D}_U$, retrieve the labels from an oracle O and update its model iteratively. Given an example in $\mathcal{D}_U$, its label can be obtained by taking one of the two actions. The first action is to query the oracle O for its label by paying the querying cost C_q. The second action is to make a prediction (positive or negative) on the example. The consequence of the prediction will reveal the true label. If the prediction is wrong, we have to pay the misclassification cost[1] C_m. For each example with a posterior probability produced by the learner, the action to take can be determined. The goal of this learning problem is to find a proper learning sequence for the examples from $\mathcal{D}_U$, such that the total cost is minimized.

[1]In this chapter, we only consider the case that the misclassification costs of false positive and false negative are equal. In addition, the querying cost and misclassification cost are constant for different examples. It can be easily extended to uneven costs of false positive and false negative and the instance-dependent cost.

4.2.2 Choice of Actions

For an example x in $\mathcal{D}_U$, the selection of action depends on the costs for the two possible actions. Given the posterior probability $p(1|x)$ predicted by a learner, the expected misclassification cost C_e will be $P(1|x)\times C_m$ if $p(1|x) \in [0, 0.5]$, or $(1-p(1|x))\times C_m$ if $p(1|x) \in (0.5, 1]$. If the expected misclassification cost C_e is less than the cost of querying an oracle C_q, we should make the prediction directly; otherwise, we should query the oracle for the label.

Similarly, according to the inequality above we can easily derive a probability interval $[\alpha, \beta]$ as shown in Figure 4.1) for given query cost C_q, misclassification cost C_m. The value of α and β can be calculated with Formula 4.1. For the examples with the posterior probability falling in the interval, the learner will choose to learn them by querying an oracle for its label; otherwise, the learner will choose to make predictions on them directly. The interval is, in fact, the same as the rejection interval in the classifiers with reject option [27, 6], where the learner can reject to make prediction on an example when it is not certain about it. However, it does not make predictions during the learning process.

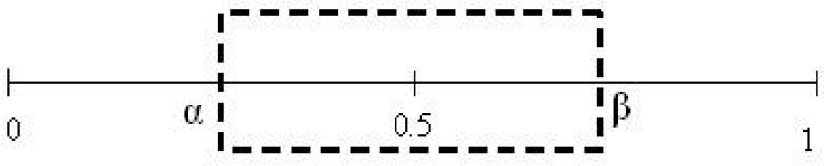


Figure 4.1: Probability interval to query oracle.

$$\begin{cases} \alpha = C_q/C_m \\ \beta = 1 - C_q/C_m. \end{cases} \tag{4.1}$$

In addition, in Formula 4.1 we can see that α and β are symmetric[2] of 0.5 and Figure 4.1 shows that the actions on the two sides of 0.5 are also symmetric. Thus, we can transform Figure 4.1 into Figure 4.2. Instead of $P(1|x)$, the horizontal axis in Figure 4.3 changes to $P(d|x)$. Here, d is the prediction (0 or 1) made by the learner, which depends on the higher probability of $P(0|x)$ and $P(1|x)$. If $P(0|x) > 0.5$, then $d = 0$; otherwise, $d = 1$.

4.2.3 Action Boundary

In Figure 4.2, due to the symmetry, we only have one threshold $\beta = 1 - C_q/C_m$ for $P(d|x)$, instead of two thresholds α and β in Figure 4.1. For examples with $0.5 \le P(d|x) < \beta$ we should take the action of querying oracle, while for examples with $\beta \le P(d|x) \le 1$ we should take the action of making prediction. We call the threshold β *action boundary*. The position of β is not necessarily in the center of the axis, instead it relies on the cost of querying an oracle and the misclassification cost. If an oracle is too expensive to query, the value $1-C_q/C_m$ will be small, and thus β will be closer to 0.5. If the wrong prediction is costly, β will be closer to 1.

[2] α and β are symmetric of 0.5 as in this work we assume that the cost for false positive and false negative are even. This work can be extend to uneven costs easily.

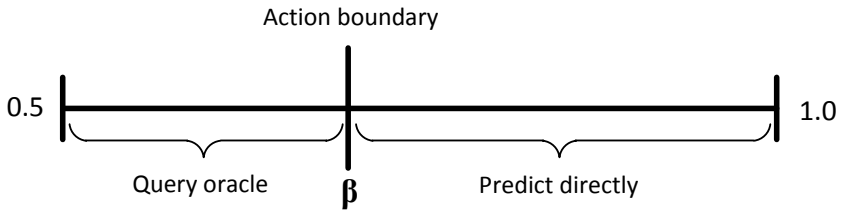


Figure 4.2: Illustration for action boundary, and the horizontal axis represents $P(d|x)$.

The correct choice of the actions depends on the accuracy of the posterior probability $P(d|x)$. Here, the accurate probability means that the probability is perfectly calibrated. Calibrated probability reflects the prediction uncertainty. For example, if the probability is calibrated, then for 100 examples with $P(1|x) = 0.8$, there will be 80 positive examples and 20 negative examples. Two factors will affect the accuracy of $P(d|x)$: calibration of the classifier and data sufficiency.

Different classifiers have different calibration behaviors. As reported previous works [65, 108], many learning algorithms, such as *Naive Bayes*, *Decision Tree* and *SVM*, can output scores in addition to the labels given examples, but their scores are not calibrated. We need some calibration methods to transform them into calibrated probabilities. On the other hand, for some classifiers such as *Bagged Decision Trees* and *Random Forest*, their probabilities are better calibrated. Good calibration of probability is critical for choosing the correct actions.

In addition, insufficient training data may also result in the poor accuracy of the probability, since the learner does not observe enough examples. Thus, at the beginning of the learning process, it is likely that the probability is not so accurate. However, the accuracy of the probability can be improved by learning more labeled examples.

If a classifier can produce perfectly calibrated probability $P(d|x)$, then the action taken on each example in $\mathcal{D}_U$ will always be the correct choice. However, due to the poor calibration and data insufficiency, usually the posterior probability generated by the classifiers may be inaccurate.

4.2.4 Indecisive and Decisive Actions

The inaccuracy of the posterior probability $P(d|x)$ will easily lead to the wrong choice of the actions, particularly in the boundary area close to β as shown in Figure 4.3. For the boundary examples, the learner is not sure which action to take. Therefore, we call those boundary examples *indecisive examples*, and the actions taken on them *indecisive actions*. For the examples far away from the action boundary β (approaching 0.5 or 1), the learner is more certain about which action to take and the actions taken on them will be less likely to be mistaken. Hence, we call those examples *decisive examples*, and their actions *decisive actions*[3].

In fact, there is no clear threshold to distinguish decisive and indecisive actions. In Figure 4.3, we use the darkness to demonstrate the decisiveness of the actions. The darker the color of the area, the less decisive the actions taken in the area. We can see that the decisiveness of the actions gradually decreases from β to the two ends (0.5 or 1).

[3]In this paper, for each selected example an action will be taken, thus we regard taking actions and selecting examples equivalently.

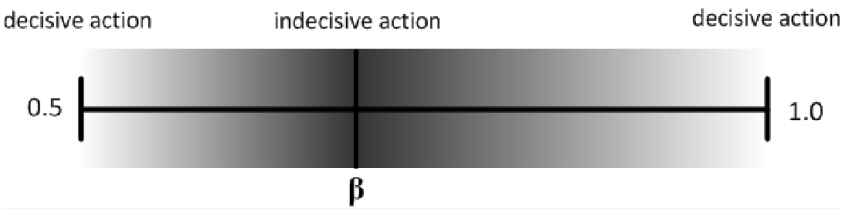


Figure 4.3: Illustration for indecisive action.

Based on the decisiveness of action, in the next section, we will propose a new learning algorithm to minimize the total cost.

4.3 Decisive Active Learner

The key issue to the learning problem defined in Section 4.2 is how to correctly determine the action sequence on the examples in $\mathcal{D}_U$ such that we can minimize the total cost. In the following, we will present a novel learning algorithm *Decisive Active Learning* (DAL) to learn examples from decisive to indecisive ones.

Algorithm 3: DAL

Input: Unlabeled Dataset $\mathcal{D}_\mathcal{U}$; Training data: $\mathcal{D}_T$; Initial model: $\mathcal{M}_0$
Output: Model: $\mathcal{M}$ and the total cost: C

```
1  begin
2      itr = 0; //the first iteration
3      C = 0;
4      while D_U <> NULL do
5          for each x_i ∈ D_U do
6              Calculate the decisiveness of x_i;
7          end
8          //Select the example that is expected to have the most decisive action
9          Select the example x_i with the highest decisiveness;
10         Calculate the cost c_i for labeling x_i; //c_i = C_e or C_q
11         C ← C + c_i;
12         D_T ← D_T + x_i;
13         Update the current model M_itr with D_T;
14         D_U ← D_U − x_i;
15         itr + +;
16     end
17     M ← M_itr;
18     Return M and C;
19 end
```

4.3.1 DAL Algorithm

Decisive Active Learner (*DAL*) is an algorithm on how to determine the learning sequence of examples in unlabeled dataset $\mathcal{D}_U$. The basic idea of DAL is that the decisive examples take precedence to be selected for learning since the actions taken on them are more likely to be correct, and the indecisive examples will be left for learning later (See Algorithm 3 for the psuedocode). As we mentioned in Section 4.2.3, when more examples are observed by the learner, the model built is expected to become more accurate, the indecisive examples may become decisive, and consequently actions will be less likely to be mistaken.

More specifically, the decisive examples that DAL starts with have two different types: examples (close to 0.5) to query oracle and examples (close to 1) to make predictions. Both of the two types of examples are beneficial for the learner. The examples with probabilities close to 0.5 can help the learner achieve high accuracy with few examples. Direct prediction on the examples with probabilities close to 1 makes good use of the current learner and is likely to obtain the true labels without paying any cost. Moreover, those (certain) examples can make the learner more robust. Therefore, in addition to indecisive examples, DAL has to determine the learning sequence of the two types of decisive example such that it can take the advantage of learning them. Specifically, we design the learning sequence of DAL (illustrated in Figure 4.4) as follows.

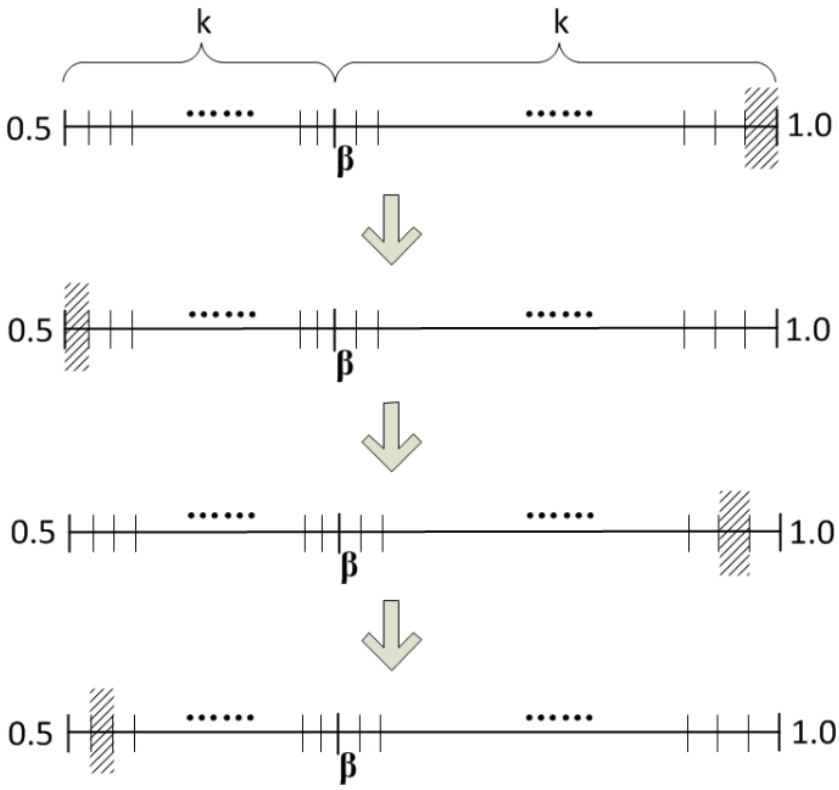


Figure 4.4: Illustration of the learning process for the decisive active learner (DAL). DAL learns the examples in the intervals (shadowed) alternately on the two sides of the action boundary β, gradually approaching β. The actions are taken from the most decisive to the most indecisive.

1. **Splitting probability interval**: Each of the probability ranges $[0.5, \beta)$ and $[\beta, 1.0]$ is split into k ($k = 50$ in our experiment) equal intervals. Each example in the unlabeled set $\mathcal{D}_U$ falls in an interval according to the posterior probability $P(d|x)$ predicted by the current learner. See the intervals in Figure 4.4.

2. **Selecting the starting interval**: The learner chooses the most decisive interval, which is furthest from the action boundary β, as the starting interval for the learning. See the shadowed interval in the top of Figure 4.4.

3. **Learning in an interval**: The current learner checks if there is any example in $\mathcal{D}_U$ located in the current interval. If yes, we select the most decisive example in the current interval, acquire its label by taking the corresponding action and update the learner, and then repeat step 3; otherwise, we proceed to step 4.

4. **Alternating Interval**: If all examples in $\mathcal{D}_U$ are learned, we terminate the learning; otherwise, the learner selects an interval on the other side of β as the next interval, and then go back to step 3 to learn examples in it. See the process indicated by shadowed intervals in Figure 4.4.

4.3.2 Splitting Probability Interval

What is the motivation for the probability intervals? Why not learn the most decisive examples on the two sides of β alternately, instead of alternating the intervals? The problem is that if the probability distribution on the unlabeled set is biased or skewed, the selected examples may not be useful for the learner, and thus the resulting action sequence might not be optimal. Suppose for most of the unlabeled examples the posterior probabilities $P(d|x)$ are close to 1, and few examples are close to 0.5 and even the most uncertain example is pretty far from 0.5 (say $P(d|x) = 0.6$). In this case, if the learner is designed to learn an uncertain example every two iterations, the learner may not benefit much from the uncertain example. However, alternation of the probability intervals can avoid this problem. If there is no unlabeled examples falling into the current interval, the learner will skip the interval, and select an interval on the other side. Thus, our algorithm can guarantee that the most important (beneficial) examples for the current learner will be learned.

4.3.3 Selecting the Starting Interval

From our algorithm, we start learning from the furthest interval from β. We can see the starting interval depends on the position of β. The starting interval is actually critical to the performance of the learner, since the learner can learn a number of examples in a probability interval and we hope the actions taken on those examples are as correct as possible. Thus, we prefer to learn the interval where examples are more decisive. Intuitively, the further the interval is from the action boundary, the more likely the actions taken in it are correct. In Section 4.4.9, our experiment further confirms that this selection method of the starting interval indeed reduces the total cost in the learning process.

4.3.4 Learning in an Interval

Learning all the examples in an interval may take multiple iterations. In each iteration, we only select the most decisive example x in the current interval. Then if $P(d|x) \in [0.5, \beta]$, then we query the oracle for its label; otherwise, we make prediction on x. After the label is revealed, we include it into the training set and update the learner, as well as all the posterior probabilities of examples in $\mathcal{D}_U$. In the next iteration, the learner will select another example in the current interval, based on the updated probabilities. If the interval is empty, the learner will alternate to learn examples in the next interval.

4.3.5 Alternating Intervals

If there is no update on the learner, the probabilities $P(d|x)$ of the examples in $\mathcal{D}_U$ will not change. In this case, we will alternate to learn examples in the next non-empty interval on the other side of β. However, if the learner is updated, the probabilities predicted by the learner may vary, and examples may fall into the previously empty intervals again. Hence, once the learner is updated, we need to restart the alternation to recheck the previous intervals. Suppose we finished learning the examples in the interval [0.50,0.52) and the probability $P(d|x)$ of an example x changed from 0.7 to 0.51 after the learner is updated, which makes the x very informative to the current learner. If we skip this empty interval [0.50,0.52), we will lose the chance to learn a useful example. Therefore, it is reasonable to revisit the previous intervals.

From the algorithm of DAL, it is clear that the learner always learns the most decisive examples and attempts to make as few mistakes on the actions as possible. This feature ensures that DAL achieves a good performance in terms of the total cost.

4.4 Experiment

In this section, we will empirically study the performance of DAL in terms of the total cost. We will compare it with other four typical learners. Besides, we will experimentally explore different factors that affect the performance of DAL.

4.4.1 Datasets

We will evaluate the performance of DAL on 10 UCI [2] datasets [2] with the size ranging from 898 to 32561. The detailed information of the 10 datasets is tabulated in Table 4.1.

4.4.2 Cost ratios

To be more comprehensive, our evaluation will be conducted under different cost ratios between the misclassification cost C_m and the oracle querying cost C_q. We will choose three cost ratios, $C_m/C_q = 2.5$, $C_m/C_q = 4$ and $C_m/C_q = 10$. Since $\beta = 1 - C_q/C_m$, the corresponding action boundary β are 0.6, 0.75 and 0.9. The reason we choose the minimum cost ratio as 2.5 is that we should make sure $C_m/C_q >= 2$; otherwise, for any $P(d|x)$, $C_m \times (1 - P(d|x)) < 2 \times C_q \times 0.5 < C_q$, meaning that making prediction is always better than querying the oracle regardless of $P(d|x)$.

	No.Att	No.Ex	Class dist
abalone	8	4177	2730/1447
adult-census	14	32561	24720/7841
anneal	39	898	214/684
credit-g	20	1000	700/300
diabetes	8	768	500/268
nursery	8	12960	4650/8310
sick	29	3772	3541/231
spambase	57	4601	2788/1813
splice	60	3190	1535/1655
waveform	40	5000	3345/1655

Table 4.1: 10 UCI datasets.

4.4.3 Other Learners

In our experiments, DAL will be compared with other four typical learners. We will give a brief introduction of them in this subsection.

- **Indecisive Active Learner**: The first learner is named indecisive active learner (*IAL*), which takes the opposite learning sequence of DAL. It always selects the most indecisive examples in the learning process. Specifically, in each iteration, IAL always selects the example x from the unlabeled set $\mathcal{D}_U$ such that $p(d|x) = min_{x \in U} |p(d|x) - \beta|$. The process iterates until $\mathcal{D}_U$ becomes empty. It can be expected that the learner will make many mistakes on actions, especially at the beginning of the learning process.

- **Aggressive Learner**: Aggressive learners (AGG) are those that choose the most challenging example (i.e., the example that is least certain by the current learner), to learn in each step. It is similar to *uncertainty sampling* [102] in active learning. It always gives preference to the example that is closest to the decision boundary and queries the oracle for its label. Specifically, in each iteration of the learning process, AGG always selects the example x from the unlabeled set $\mathcal{D}_U$ such that $p(d|x) = min_{x \in U}(p(d|x) - 0.5)$. The process iterates until $\mathcal{D}_U$ becomes empty. Previous works [84] show that aggressive learner (uncertainty sampling) can achieve high accuracy with few labeled examples (low querying cost). However, if the cost of querying the oracle is very expensive, it will be very costly for the learner to further learn from the examples. Besides, starting from few examples, the aggressive learner may not be very robust, and sometimes as more examples are learned the accuracy even decreases.

- **Conservative Learner**: In contrast to the aggressive learner, conservative learner (*CON*) always exploits the data it can predict well and tries to make as few mistakes on the predictions as possible. Thus, it prefers to learn the examples with the posterior probability $P(d|x)$ close to 1. Specifically, in each iteration of the learning process, CON always selects the example x from the unlabeled set $\mathcal{D}_U$ such that $p(d|x) = min_{x \in U}(1 - p(d|x))$. The process iterates until $\mathcal{D}_U$ becomes empty. Conservative learner has two obvious deficiencies. One is that the learning efficiency can be poor, as the ex-

amples selected are homogeneous to the labeled examples. The second deficiency is that if the misclassification cost is expensive, the total cost can be very high.

- **Random Learner**: Random learner (*RND*) does not have any bias in the learning process. Each iteration it randomly selects one of the unlabeled example in $\mathcal{D}_U$. The process iterates until $\mathcal{D}_U$ becomes empty.

In all the learners above, for the selected example x, if $p(d|x) < \beta$, the learner will query the oracle for its label; otherwise, it will make prediction on x. Figure 4.5 illustrates the basic learning process of the five learners.

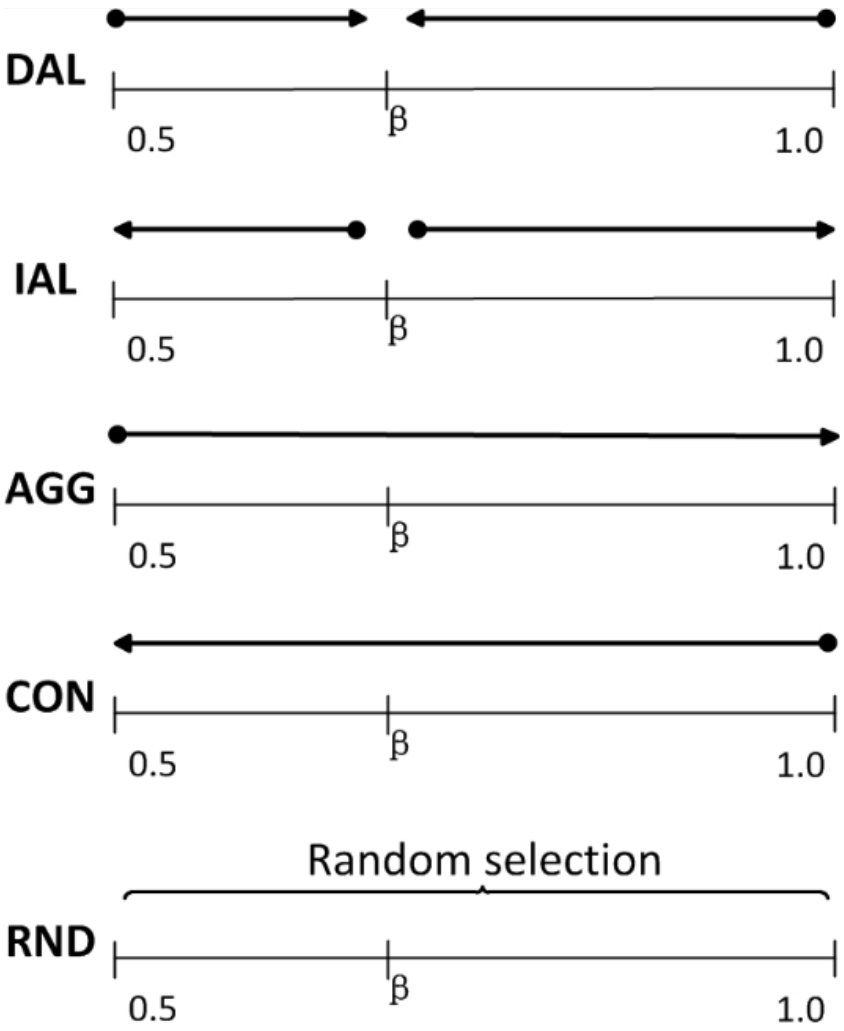


Figure 4.5: Illustration of the learning process for five learners. DAL is the decisive active learner. IAL is the indecisive active learner learner. AGG is the aggressive learner. CON is the conservative learner. RND is the random learner.

4.4.4 Base Classifier

In our experiment, we use *bagged decision trees* as the base classifier for all the five learners. The reason is that bagged decision tree as reported by [65] can output well-calibrated posterior probabilities, which makes the prediction more accurate. In Section 4.4.8, we will utilize another base classifier *Decision Tree* and *Naive Bayes* to further explore the effect of calibration on DAL.

4.4.5 Experimental Setting

For each of the 10 UCI [2] datasets in Table 4.1, we randomly select 100 examples as the labeled set, and use it to train the initial classifier for each of the five learners. The rest of the examples belong to the unlabeled set. For the five learners mentioned in Section 4.4.3, we calculate the total cost (the misclassification cost and the cost of querying oracle) spent in the entire learning process. The less the cost, the better the learner. We run the five learners on the 10 datasets under the three cost ratios (Section 4.4.2) for 10 times. Friedman test and Wilcoxon signed-rank test will be chosen to statistically test the difference of the total cost among the five learners. It should be noted that after learning all the unlabeled examples, the five learners should have the same predictive model, since the model is built on the same set of examples.

4.4.6 Statistical Testing Methods

The total cost in each repeat can be affected by the initial split of the dataset, thus the cost may have large variance in different repeats and even the data itself may not be normally distributed. In this case, the Friedman test can be a reasonable choice for our statistical testing, since it uses the ranks of the data rather than their raw values to calculate the statistic. Friedman test has been widely used to test whether there is a statistically significant difference between a group of values [17, 36]. If significant difference exists in the group, we still need a post-hoc test on different pairs of groups to report their statistical difference. Wilcoxon signed-rank is a commonly used post-hoc test following Friedman test [17, 36], thus we will use it in our experiment.

4.4.7 Comparative Results

Table 4.2 demonstrates the average total costs of the five learners on the 10 UCI datasets under three cost ratios. In order to evaluate the statistical difference, we also calculate the ranking of the five learners based on Wilcoxon signed-rank test. The ranking is calculated by the following steps. For each row in the table, we first sort the five learners by the average cost ascending. Then we compare the learner with the smallest mean to the one with the largest mean. If there is no significant difference, all the learners will be ranked as 1; otherwise, we continue to compare the smallest mean with the second largest mean, until all the learners have been compared or no significant difference is found. In the next round, we will compare the second smallest mean with the largest mean, and repeat the same step. The process iterates until all learners are ranked.

	Cost ratio	DAL	IAL	AGG	CON	RND
abalone	2.5	889(1)	955(3)	1031(4)	891(1)	915(2)
	4	1081(1)	1313(4)	1238(3)	1341(4)	1122(2)
	10	1461(2)	1520(3)	1410(2)	1723(4)	1394(1)
adult-census	2.5	4803(1)	4904(2)	5511(3)	4813(1)	4918(2)
	4	5893(1)	7178.8(4)	6887(3)	7102(4)	6228(2)
	10	8097(1)	9228(3)	9121(2)	9199(2)	8323(1)
anneal	2.5	66(1)	75(2)	88(2)	58(1)	66(1)
	4	102(1)	124(2)	124(2)	101(1)	95(1)
	10	130(1)	210(3)	194(3)	167(2)	153(2)
credit-g	2.5	268(1)	282(2)	308(3)	271(1)	271(1)
	4	331(1)	397(3)	350(2)	396(3)	342(2)
	10	375(1)	400(2)	380(2)	445(3)	372(1)
diabetes	2.5	195(1)	204(2)	211(2)	194(1)	200(1)
	4	234(1)	262(2)	241(1)	274(2)	241(1)
	10	290(1)	294(1)	285(1)	309(1)	287(1)
nursery	2.5	273(1)	276(1)	385(3)	285(2)	282(2)
	4	349(1)	436(3)	497(4)	391(2)	339(1)
	10	610(1)	879(3)	802(2)	764(2)	562(1)
sick	2.5	84(1)	93(1)	93(1)	83(1)	86(1)
	4	122(1)	143(2)	132(1)	137(2)	125(1)
	10	217(1)	258(2)	254(2)	270(2)	210(1)
spambase	2.5	416(3)	364(1)	628(4)	389(2)	413(3)
	4	554(1)	704(3)	891(4)	623(2)	589(1)
	10	753(1)	1436(4)	1415(3)	1335(3)	976(2)
splice	2.5	391(2)	322(1)	625(3)	333(1)	337(1)
	4	557(1)	673(3)	802(4)	535(2)	482(1)
	10	688(1)	1094(3)	1077(3)	1055(3)	798(2)
waveform	2.5	640(1)	624(1)	940(2)	614(1)	616(1)
	4	847(1)	959(2)	1101(3)	971(2)	840(1)
	10	1039(1)	1503(3)	1341(2)	1952(4)	1084(1)
average rank	2.5	1.3	1.6	2.7	1.2	1.5
	4	1	2.8	2.7	2.4	1.3
	10	1.1	2.7	2.2	2.6	1.3
	overall	1.1	2.4	2.5	2.1	1.4

Table 4.2: Statistical comparisons between the five learners in terms of the cost. Each cell shows the average cost and its rank (in the bracket) of a specific learner on a dataset under a cost ratio. The rank is calculated by a statistical test named Wilcoxon signed-rank. In the bottom, the overall average rank and the average rank under the three cost ratios (2.5, 4 and 10) are presented.

In Table 4.2, the rank of each learner is presented in the bracket next to the average total cost. The four rows in the bottom of Table 4.2 present the average rank of the five learners over all the 10 datasets under the cost ratio (2.5, 4 and 10) respectively, as well as the overall average rank over 30 rows in the table. We can see clearly that DAL is top ranked in 27 out of the total 30 comparisons and has the lowest average rank 1.1 over the 30 rows. It is evident that DAL has the overall best performance in terms of the total cost.

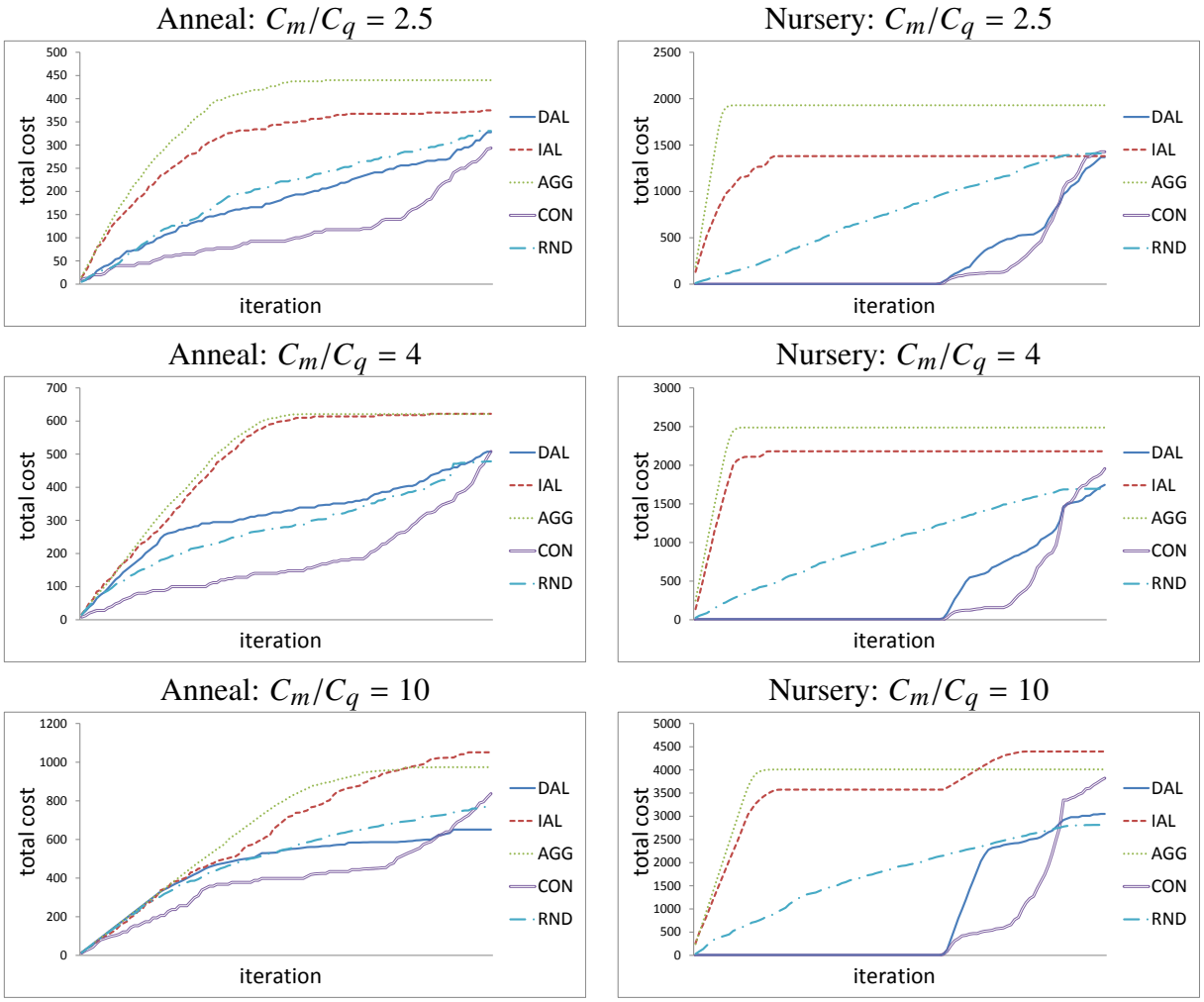


Figure 4.6: Comparison of learning curves in terms of the total cost. Two datasets (anneal and nursery) corresponds to the three rows from left to right respectively.

Itt is also interesting to note that the random learner (RND) has the 1st rank in 20 out of the total 30 comparisons and an overall average rank (1.4) that follows closely DAL (1.1). It seems contradictory to the previous works on active learning, where random learner usually has the poorest performance compared to other active learners. However, in those cases, they do not consider making predictions in the learning process and only consider the cost to query the oracle. In our case, the random selection strategy of RND is likely to unbiasedly take the two actions (making prediction and query oracle), which is similar to the action alternation strategy of DAL. In addition, although some of the examples chosen by RND may be indecisive

(close to β), the random distribution makes most of them relatively decisive. Thus, RND has a reasonable performance in our scenario.

Theoretically, IAL is supposed to have the poorest performance since it always selects the most indecisive actions and is expected to make many mistakes. However, we observe that it is not exactly the case in Table 4.2. Overall, the average rank (2.4) of IAL is slightly better than that (2.5) of AGG. In fact, under the cost ratio 4 and 10, IAL indeed has the lowest rank among the five learners. The slight superiority of IAL to AGG is due to the fact that IAL performs much better than AGG when the cost ratio is 2.5. Under the cost ratio 2.5, β (1-1/2.5=0.6) is relatively close to 0.5. IAL starts learning from the examples with probability around 0.6 while AGG from the examples with probability close to 0.5. Although the examples selected by IAL are not as informative as those selected by AGG, those examples are still useful for the learner. Furthermore, IAL can even save more costs by directly making predictions on the examples to the right side of β, as the expected cost of making prediction on those examples is lower than the cost of querying an oracle ($(1 - P(d|x)) \times C_m < C_q$, where $P(d|x) > \beta$).

From the average rank in Table 4.2, we can also observe that DAL is more likely to have better performance when the cost ratio is high, as its average rank increases when cost ratio becomes higher. It means when the misclassification cost is much higher than the querying cost, it is safer and more desirable to use DAL as the learner.

In order to visualize the learning process, we also plot the learning curves of the five learners on three datasets (anneal and nursery), in Figure 4.6. The Y axis in Figure 4.6 is the average value of the total cost over the 10 repeats. We can see the five learners behave differently in the trend of the learning process.

In most of the cases, the cost of AGG increases sharply in the beginning of the learning but maintains steady in the later stage. It is because it selects the uncertain examples first and spends most of the cost on querying the oracle. Once it begins to make predictions, it becomes very accurate and makes few mistakes, thus the total cost keeps stable in the later stage of learning.

CON takes the opposite trend compared to AGG. At the initial stage, certain examples ($P(d|x)$ close 1) can be predicted well and thus the total cost grows slowly. However, after the certain examples are learned, the poor predictive performance of CON tends to misclassify the challenging examples, which contributes to the sharp increase of its total cost.

Generally, the curves of IAL are slightly under the curves of AGG. However, in some cases, especially when the cost ratio is high (e.g., 10), the total cost of IAL becomes even higher than that of AGG in the later stage. This observation is actually consistent with Table 4.2. IAL starts to select the most indecisive examples, and thus it can be expected that IAL will make many mistakes on the actions. It is the reason that the total cost of IAL increases rapidly in the beginning of the learning process.

The curves of DAL are between the curves of AGG and CON, but it ends up with less total cost in most of the cases. By making predictions on decisive examples, DAL saves more cost compared to AGG. On the other hand, DAL spends extra cost on querying oracle, which leads to its higher cost than CON in the beginning stage.

The curves of RND are basically along the diagonal. As we mentioned before, RND randomly selects the examples, thus it does not bias any of the two actions in the learning process. It is reasonable that RND learns steadily in the whole learning process, without any rapid growth and plateau in the curve of the total cost.

One may notice that, in Figure 4.6, the total costs of DAL are not always the lowest ones, which seems contradictive with the previous results that DAL obtains labels with the lowest cost. The reason is that the lowest cost of DAL is a statistical result. It does not indicate that DAL requires the least labeling cost on every dataset and under every cost ratio.

4.4.8 The Effects of Calibration

As we mentioned in Section 4.2.3, the performance of DAL may be affected by the calibration of the base classifier. In this subsection, we will study the performance of DAL under different base classifiers with varied calibration capabilities. As reported by previous works [65], *decision tree* and *naive bayes* have poor calibrated probabilities. We would like to see how DAL behaves by using them as the base classifier compared to *bagged decision tree*. Table 4.3 tabulates the average costs of DAL on the 10 UCI datasets by using the bagged decision tree, decision tree and naive bayes. The bold value(s) in each row of Table 4.3 mean that no others are significantly better in that row by using Wilcoxon signed-rank test.

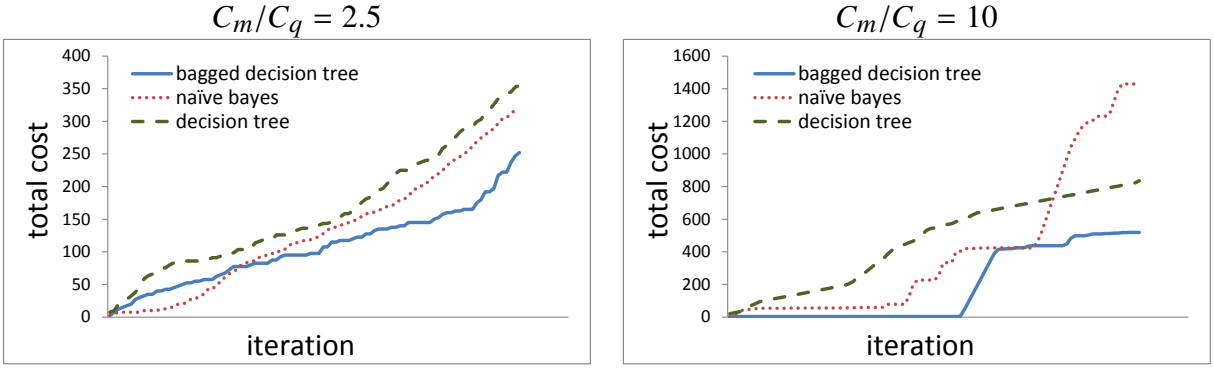


Figure 4.7: Comparison of learning curves of DAL by using different base classifiers on a dataset (credit-g) under two cost ratios (2.5 and 10).

It is evident that by using bagged decision tree as the base classifier, DAL can achieve the lowest cost on all the 10 UCI datasets. We plot their learning curves on a dataset (credit-g) in Figure 4.7 under two different cost ratios (2.5 and 10). In the two subgraphs, the curves of bagged decision tree are generally under the curves of the other two classifiers, especially at the later stage of the learning process. It is due to the difference of calibration of the three classifiers. As more examples are learned in the learning process, the calibrated classifier (bagged decision tree) can produce more accurate probabilities than the uncalibrated classifiers (decision tree and naive bayes). The accurate probabilities can help the learner choose the correct actions for DAL, and thus reduce the total cost in the learning process. Hence, from Table 4.3 and Figure 4.7, we arrive at the same conclusion. That is well-calibrated classifier is crucial for the performance of DAL.

4.4.9 The Effects of Starting Interval

DAL adaptively selects the starting interval based on the cost ratio (from the side further away from β). Does it really have significant effect? What if we select the starting interval in the

Dataset	CRatio	BDT	DT	NB
abalone	2.5	**889**	1540	1877
	4	**1081**	2296	2805
	10	**1461**	4444	5763
adult-census	2.5	**4803**	6560	8088
	4	**5893**	8735	11710
	10	**8097**	17892	21355
anneal	2.5	**66**	61	107
	4	**102**	**113**	146
	10	**130**	**185**	222
credit-g	2.5	**268**	358	306
	4	**331**	510	394
	10	**375**	967	541
diabetes	2.5	**195**	234	223
	4	**234**	345	280
	10	**290**	478	377
nursery	2.5	**273**	406	535
	4	**349**	608	808
	10	**610**	1230	1731
sick	2.5	**84**	112	362
	4	**122**	172	511
	10	**217**	388	795
spambase	2.5	**416**	685	1338
	4	**554**	1084	2142
	10	**753**	2558	5456
splice	2.5	**391**	**420**	**405**
	4	**557**	**576**	**582**
	10	**688**	1257	**865**
waveform	2.5	**640**	1287	1019
	4	**847**	2020	1546
	10	**1039**	5265	3136

Table 4.3: Comparison of the total cost of DAL by using different base classifiers. In the header, CRatio means cost ratio, BDT represents bagged decision tree, DT represents decision tree and NB represents naive bayes. The bold value(s) in each row mean that no others are significantly better in that row by using Wilcoxon signed-rank test.

opposite way? In the following, we will empirically explore this issue. We will modify the original DAL so that it starts from the side closer to β. We name the modified version *ODAL*.

Table 4.4 presents the comparison between DAL and ODAL in terms of the total cost. The bold value(s) in each row mean that no others are significantly better in that row by using Wilcoxon signed-rank test. We can see, statistically, the original DAL achieves the best performance in 29/30 cases, while ODAL only in 23/30 cases. Thus, by choosing starting interval further from β, the learner is more likely to achieve lower cost.

Dataset	CRatio	DAL	ODAL
abalone	2.5	**889**	**919**
	4	**1081**	**1125**
	10	**1461**	**1516**
adult-census	2.5	**4803**	**4788**
	4	**5893**	**5976**
	10	**8097**	8457
anneal	2.5	**66**	**74**
	4	**102**	**106**
	10	**130**	158
credit-g	2.5	**268**	**267**
	4	**331**	369
	10	**375**	**371**
diabetes	2.5	**195**	**195**
	4	**234**	262
	10	**290**	**275**
nursery	2.5	**273**	**248**
	4	**349**	**385**
	10	**610**	723
sick	2.5	**84**	**89**
	4	**122**	**136**
	10	**217**	249
spambase	2.5	**416**	453
	4	**554**	**553**
	10	**753**	**699**
splice	2.5	**391**	**425**
	4	**557**	**530**
	10	**688**	**680**
waveform	2.5	**640**	**775**
	4	**847**	**879**
	10	1039.6	**899**

Table 4.4: Comparison of total cost with different interval-starting strategies. The bold value(s) in each row mean that no others are significantly better in that row by using Wilcoxon signed-rank test.

4.5 Summary

In this chapter, we proposed the paradigm of learning actively with aggressive-conservative tradeoff, under which two actions, making prediction and querying oracle, are available for obtaining labels. The goal of this learning paradigm is to minimize the total cost on the two actions. We first defined what is decisive action and then proposed a novel learning algorithm named decisive active learner (DAL), which always selects the most decisive examples and attempts to make as few mistakes on the actions as possible in the learning process. In the experiments, we demonstrated the outstanding performance of DAL in reducing the total cost,

compared to four other typical learning strategies. Besides, we empirically analyze the sensitivity of DAL to the calibration capability of the classifier, as well as starting point of the learning process. The results showed that well-calibrated classifier can boost the performance of DAL and the selection method of the starting point used in DAL helps to reduce the total cost. The proposed learning algorithm can be applied to various real-world applications, particularly in the scenarios where true labels can be acquired after predictions.

Chapter 5

Learning Actively with Minimal/Maximal Effort

In Chapters 2, 3 and 4, three different new learning paradigms for active learning have been proposed. Under each paradigm, novel active learning algorithms have been proposed to achieve the learning goals. In this section, we propose another goal during the active learning process - the "effort" of learning. The "effort" of active learning can correspond well to the actual effort in human learning (see below), as well as the energy consumption in implementing the actual active learning algorithms. This can be important when the learning algorithms actually run in circuit boards [60, 45]. Algorithms using less energy generate less heat, and use less battery. However, we understand that the actual energy consumption of any algorithm depends on many factors (such as processor speed, the architecture of the chips, the types of transistors, circuit boards and so on). As these factors are related to hardware and their values may vary from machine to machine, it is difficult to measure them. Thus, in this section, we will use the amount of error to be corrected during learning, and the size of the learning models, to approximate the effort, or the energy consumption of the algorithms. This is reasonable as many learning algorithms, such as ANN [40], reduce their training error gradually and iteratively. Less error to be corrected generally indicates fewer learning iterations, i.e., less energy consumption. Some other learning algorithms, such as decision tree algorithms [70], build models gradually. A decision tree of smaller size means splitting nodes fewer times, i.e., less energy consumption, during the learning. See Sections 5.4 for details.

In education research of human learning, an effort has been considered when people learn knowledge. The theory called the "i+1" learning has been proposed [49, 20]. It suggests that human learns a small piece of new knowledge ("1") based on a large body of previously learned knowledge ("i")[49]. For example, infants learn simple words and sentences first, and gradually learn complex ones through repeated exposures from parents and care takers. Adults also learn gradually from simple materials to complex ones through series of class or lectures. While learning with minimal effort, we can expect that learning is slow but more stable. An opposite type of learning is "maximum-effort learning". Researchers have shown that by paying more or maximal effort (in taking the most challenging problems, for example), learning can be faster and more efficient, but less stable [105, 67].

As far as we know, little or no previous work has been done in studying the "effort" in active learning. In this section, we propose a novel paradigm of *learning actively with minimal*

or maximal effort. Under this paradigm, the labels of the examples to be learned are provided, and the learners can select actively which examples to learn in sequence. The goal is to select actively a proper sequence of examples to learn, such that the "effort" (to be defined as the amount of errors to be corrected, or the model size; see Section 5.4) can be minimized, or maximized (i.e., a model can be built fast).

Under the paradigm of learning actively with minimal/maximal effort, we propose two novel learning algorithms to achieve the two goals. For active learning with minimal effort, we propose a simple yet effective algorithm called S2C, which stands for "Simple to Complex". S2C always selects the current simplest example to learn. The "complexity" of an example is defined in terms of the predictive accuracy of the example by the current model. The rationale is clear: if the current model can predict the example accurately and correctly (that is why the label of the example is given), there is virtually no need to learn and update the model. We will show empirically in Section 5.5.1 that indeed, S2C does use much less effort compared to other algorithms. We also show that S2C learns more slowly, and the learning process is very stable (not volatile). See Sections 5.5.2 and 5.5.3 for details.

The S2C algorithm can be modified easily to become C2S (Complex to Simple). Opposite to S2C, C2S always selects the most complex or challenging example to learn in the learning process. We will show empirically in Section 5.5.1 that C2S does use much more effort compared to other algorithms. We also show that C2S learns more quickly but the learning process is unstable (volatile). See Sections 5.5.2 and 5.5.3 for details. The work of S2C and C2S is published in *Advances in Knowledge Discovery and Data Mining, 2010* [61].

The rest of this chapter is organized as follows. In Section 5.1, we will review some related works. The learning algorithm for achieving minimal effort will be described in Section 5.2, and the algorithm for achieving learning fast by taking maximal effort will be in Section 5.3. In Section 5.4, we will introduce how the learning performances are measured. Experimental results are presented in Section 5.5. We summarized the work of this chapter in Section 5.6.

5.1 Related Works

Learning from simple to complex gradually may look similar to incremental learning previously studied [52, 37], but they are very different. Incremental learning builds classification models gradually based on the given data passively. However, the S2C learning algorithm actively selects simple examples first to learn, and then complex ones. S2C is also different from the traditional active learning [5, 15]. The learner in traditional active learning selects the most informative unlabeled example and queries an oracle for its label. Then it rebuilds a completely new model with all the labeled examples. The goal of active learning is to build a good model with as few labeled examples as possible. S2C, on the other hand, is provided with labeled examples, and is to select and learn the examples in the simple-to-complex sequence (no oracle is required) and its goal is to minimize the learning effort[1].

Ferr, et al. proposed delegating classifiers [31], and its main idea is divide-and-conquer. The learner builds the first classifier with all the training examples, and delegates those exam-

[1]Reducing effort means less energy, e.g. power, that the algorithm needs to consume. IBM Research shows that power consumption is a major problem in designing computers to simulate human brain. http://spectrum.ieee.org/computing/hardware/ibm-unveils-a-new-brain-simulator.

ples that cannot be predicted well (complex examples) to build delegating classifiers. However, it is difficult to determine the threshold for complex examples to be delegated. Another potential issue is that the delegated examples probably are not sufficient which may affect the reliability of the delegating classifiers. However, S2C learns one model from the examples in a simple-to-complex manner. Thus S2C has neither threshold nor the difficulty of insufficient examples.

A similar idea to S2C is curriculum learning [10]. It learns by assigning the easier examples with higher weights first and then increasing the weights of the complex examples gradually. The curriculum learning is based on the idea that human or animal learns much better when the examples are organized in gradually more complex order [50]. The difficulty in this method is that the so-called easier or more complex examples are given by human, which may be unreasonable and infeasible. However, S2C is a learner-centric algorithm. It means that it is the learner who decides what examples are simple or complex, which is much more realistic and appropriate.

Lifelong learning [100] addresses the situations in which a learner faces a series of different learning tasks providing the opportunity to transfer knowledge. Recently, a number of the transfer knowledge researches have been done in different applications, such as, Web document classification [34, 79], sentiment classification [12], reinforcement learning [99], etc. The object of transfer learning is to transfer knowledge of other related but different source data to the current learning data, such that a good model can be learned with fewer examples. The simple-to-complex strategy in S2C is similar to transfer learning. The difference is that it transfers knowledge that is learned from simple examples to complex examples, such that complex ones can be learned easier. From this perspective, S2C belongs mostly to the vertical transfer learning; while the traditional transfer learning is more similar to the horizontal transfer learning in the psychology research area [73, 35].

Deep structure learning attempts to learn high level features by the composition of lower level features [10]. It starts training on simple human-crafted features and tries to get abstract high level features. One conceivable method is by training each layer one after the other [57, 53]. It is also very different from S2C, as S2C learns simple examples first and then complex examples.

In addition, the query strategy of uncertain sampling [54] in active learning looks similar to our C2S algorithm as both of them aims at achieving high efficiency in building a good model. However, uncertain sampling selects the example that its posterior probability is close to 0.5; while C2S selects the example that is predicted most differently from its true label.

5.2 S2C Learning

Under the paradigm of learning actively with minimal/maximal effort, we propose two learning algorithms: S2C and C2S (C2S will be presented in Section 5.3). In this section, we will introduce the simple and effective learning algorithm S2C, which actively selects the current simplest example to learn. The "complexity" of an example is defined in terms of the predictive accuracy of the example by the current model. The more accurate an example is predicted, the less complex the example is (See Section 5.2.1 for details). The goal of S2C is to minimize the effort for an active learner to build a model.

Why the learning algorithm S2C is expected to minimize the learning effort? The rationale behind is that learning is a gradual process of accumulating knowledge. Learning simple examples first may make the learning of complex examples easier, thus the whole learning process becomes easier (less effort) and more reliable. (See Section 5.4 for evaluation metrics).

5.2.1 S2C Learning Algorithms

At a high level, S2C is a wrapper learning algorithm, and it can use any classifier that can produce a refined class probability estimation (see later in this section for details) as its base learner. Generally speaking, for each iteration, S2C selects the simplest example (i.e., the example with minimal complexity) based on the current model and updates the model locally with the selected example. The learning process continues until all examples have been learned. The pseudocode for S2C is shown in Algorithm 4.

Algorithm 4: S2C

Input: Dataset $\mathcal{D}$; Training data: $\mathcal{D}_T$; Initial model: $\mathcal{M}_0$
Output: Model: $\mathcal{M}$; Effort: $\mathcal{E}$

```
1  begin
2      itr = 0; //the first iteration
3      E = 0;
4      while D <> NULL do
5          for each x_i ∈ D do
6              Calculate the complexity of x_i;
7          end
8          //Select the simplest example
9          Select the example x_i with minimal complexity;
10         Calculate the effort e_i for learning x_i;
11         E ← E + e_i;
12         Update the current model M_itr with the example x_i;
13         D ← D − x_i;
14         itr + +;
15     end
16     M ← M_itr;
17     Return M and E;
18 end
```

To implement Algorithm 4, there two key issues. The first one is how to calculate the complexity for a given example (Line 5), such that the simplest (least complex) example can be selected correctly. Here we propose an effective measurement which is the difference between the true label and the prediction given by the current model. The smaller the difference, the simpler the example is for the current model. For example, given one positive example x_1 and one negative example x_2, if the prediction given by the current model for x_1 is positive with probability 0.8 and for x_2 negative with probability 0.9, then x_2 will be simpler as its difference (0.1) is less than x_1's (0.2). Thus, a base learner that can generate a refined class

probability estimation is a requirement of S2C, as we mentioned before. Also the measurement is consistent with human intuition about complexity: the less the surprise (i.e., difference on error), the simpler (less complex) it is. The second issue is how to calculate the effort (Line 9), and we will introduce it in Section 5.4.

In addition to the calculation of complexity, two more important issues deserve further explanation. Firstly, how can S2C select the first simplest example before any model is built? Secondly, how can S2C select the simplest example when tie happens?

The first issue of selecting the first simplest example can be tricky, as the current model is empty. We take a simple yet effective strategy as follows. S2C scans over all the training examples[2] to pick up the most frequent example. If no example appears more than once in the training set, then an example from the majority class will be chosen randomly.

The second issue, the tie-breaking strategy, can be crucial, if tie happens often when S2C selects the simplest example. Indeed, for some algorithms, such as C4.5 [70] (the base learner for S2C in this chapter), ties do happen often. This is because C4.5 predicts all examples falling in the same leaf with the same prediction (i.e., same label and same probability estimation). For this type of algorithms, it is necessary to design an effective tie-breaking strategy to improve the performance of S2C further (See Section 5.2.2 for the details of the strategy on C4.5).

In the following, we will present how S2C works when taking the decision tree algorithm C4.5 as its base learner.

5.2.2 S2C with the Decision Tree Algorithm

As we mentioned, S2C can take most of the classifiers as its base learner easily. In this work, we choose one of the most popular classifiers C4.5 as the base learner. Originally, C4.5 builds a global tree in "one shot". However, S2C only provides C4.5 examples one by one in a simple-to-complex manner, and consequently the decision tree model will be updated locally and gradually. In the following, we will introduce how S2C works when taking C4.5 as its base learner specifically.

As discussed, S2C selects the first example (before any model is built) from the majority examples randomly (assume no repeated examples in datasets) for its base learner, C4.5, to build the first tree model - only a one-leaf tree. Then based on the current tree, S2C selects the simplest example which is predicted most correctly by the tree, then updates the tree with the selected example. This selecting-and-updating process iterates until all the training examples are learned.

However, as mentioned, a tie can occur often in decision trees, because a tree returns the label prediction and probability of an example according to the numbers of the positive and negative examples in the node that the example falls in. Thus, two different types of tie may occur when S2C selects the simplest example. One is that those examples that fall into the same leaf node. The other one is that those examples that fall into different nodes but have the same probability estimation. S2C could solve these tie problems by selecting one example randomly. However, to improve the performance further, we design better strategies to judge

[2]We understand that it takes effort to scan over examples for selecting the simplest example. However, comparing to the effort for training models, the scanning effort is much less. Thus, in this chapter we will not consider the scanning effort.

which of the equally simple examples is simpler than the others. According to the types of tie, we propose two effective tie-breaking strategies as follows.

The first strategy is for breaking the tie happening in the same leaf. In this situation, the equally simple examples belong to one class, say positive. The simplest positive example should be far away from any negative examples. Accordingly S2C should choose the example that has the furthest distance to its closest negative example comparing to other equally simple ones. More specifically, S2C calculates the Euclidean distance for each equally simple positive example to all the negative examples. It records the closest distance for each positive example, and selects the example having the greatest distance as shown in Formula 5.1.

$$x_i = \arg\max_i(\min_j(d_{ij})) \quad 0 \le i < n_1; 0 \le j < n_0 \tag{5.1}$$

where, d_{ij} is the Euclidean distance from example x_i to x_j, x_i and x_j belongs to positive and negative respectively, n_1 is the number of the equally simple examples and n_0 is the number of the negative training examples.

The second tie-breaking strategy is for breaking the ties among the examples that fall in different leaves but have the same probability estimation. Two factors can be considered for this circumstance. One is the number of the examples in the leaves where tie happens. Intuitively, the more examples a leaf has, the more reliable the leaf is in terms of probability estimation. For example, if two positive examples fall into two positive leaves, say, A and B, and A has 9 positive examples and 1 negative, and B has 18 positive examples and 2 negative, the predictions for the two examples would be tied. However, B should be preferred over A when breaking this tie because B is more reliable. The other factor is the depth of the leaf in the tree. Intuitively the closer a leaf is to the root, the more preferred the example that falls in the leaf is. For example, if a leaf C is on the first level and D is on the fifth level, and both C and D have, say, 9 positive examples and 1 negative, the example that falls in C should take precedence over the one that falls in D, since only one attribute is needed to predict its label. Accordingly, the preference of an example should be proportional to the first factor and inversely proportional to the second factor. More specifically, we combine the two factors as shown in Formula 5.2 for calculating the "Preference" for each equally simple example.

$$Preference = \frac{1 + \frac{N_1 - N_0}{N_{root}}}{2} * \frac{1}{L_{path}} \tag{5.2}$$

where, N_1 and N_0 are the numbers of the positive and negative examples respectively in the leaf node that x_i falls in, N_{root} is the number of the examples in the root node, and L_{path} is the length from the root to the leaf that x_i falls in.

In Formula 5.2, the value of $\frac{1+\frac{N_1-N_0}{N_{root}}}{2}$ corresponds to the first factor we discussed. It indicates how positive x_i is. If $N_1 = N_0$, the value will be *1/2*; if N_1 is very small, its value will be close to zero (more negative); otherwise, it will be close to 1 (more positive). $1/L_{path}$ corresponds to the second factor. The closer a leaf is to its root, the more preferred the example is. With Formula 5.2, S2C selects the example that has the maximal value of $Preference$.

In addition the tie-breaking strategy, one more issue, how to update the decision tree, deserves further explanation. To minimize the learning effort, the decision tree in S2C is updated

locally and gradually. Particularly, S2C only splits the fringe node if needed when updating the current tree model. That is to say, if an example agrees with the fringe node it enters, S2C will not change the tree structure; otherwise it will split the fringe node according to the traditional C4.5 algorithm. This fringe-node-split strategy is expected to reduce the learning effort effectively.

After presenting the learning algorithm S2C for achieving the goal of minimizing the learning effort, in the following, we will introduce the other algorithm C2S which aims at learning a good model fast by taking maximal effort.

5.3 C2S Learning

C2S is an opposite learning strategy to S2C. At a high level, C2S is different from S2C on two key issues. One is that C2S considers the example that its label prediction is the most different from the true label as the most informative example, and learns it preferentially. The other one is that C2S *rebuilds* its model after selecting one new example, instead of S2C which updates its model locally. The reason is that the goal of C2S is to learn a good model fast by taking maximal effort, and intuitively the model being rebuilt is more optimal and requires more effort comparing to S2C.

More specifically, C2S works in the process as follows. Firstly, it builds its first model with one random majority example. Then C2S selects the example that its label prediction given by the current model is the most different from the true label, i.e., the most complex example, and rebuilds a new model with all the selected examples. C2S works in this way iteratively until all the examples are processed.

However, when C2S selects the most complex example, the two types of ties in S2C can also happen for the same reason. For the first type, i.e., the tie happening among the examples in the same decision tree leaf, we use the Euclidean distance as a heuristic function to break the tie. The basic idea is that the most complex negative (positive) example should be close to positive (negative) examples. Particularly, we modify Formula 5.1 and select the most complex example x_i if $i = \arg\min_i(\min_j(d_{ij}))$, where x_j is the examples having the opposite class to x_i, and d_{ij} is the Euclidean distance between x_i and x_j. For the second type, i.e., the tie happening among the examples in different decision tree leaves, we prefer the example having more accurate posterior probability and closer to the root of the decision tree. That is, Formula 5.2 in Section 5.2.2 can be used directly to select the most complex example for breaking this type of tie.

The selection of the most complex example indicates that C2S is likely to take the most effort to learn the examples. As the complex examples are very informative, the current model is expected to be improved fast. That is, C2S is expected to learn a good model fast by taking maximal effort.

After introducing the two learning algorithm, S2C and C2S, the key issue is how to measure their performances. In the following we will address the measurements used for the comparison among the learning algorithms.

5.4 Measurements

In this subsection, we will discuss how to evaluate the performances of the learning algorithms under the paradigm of learning actively with minimal/maximal effort. Considering the learning goals and the stability of the learning behavior, we present three measurements including *effort*, *learning efficiency* and *volatility*.

The first measurement *effort* in active learning indicates how much work a learner has to do to learn certain examples. Intuitively it corresponds to the energy consumption in implementing the actual active learning algorithms, and the energy consumption depends on many factors, such as processor speed, the architecture of the chips, the types of transistors, circuit boards and so on. It is difficult to calculate the effort directly due to the many factors. In this work, we choose two different approaches, *error-based effort* and *size-based effort*, to reflect the effort indirectly.

- *Error-based effort* is using the prediction rate to approximate the effort. Here we provide the details of how to evaluate the error-based effort for tree-based learning algorithms. For a decision tree, the prediction error of the current tree T_i for a positive example x_i is $(1 - p(1|x_i))$, where $p(1|x_i)$ is calculated by k/n. k is the number of the positive examples and n the total number of the examples in the node that x_i falls in. The error-based effort is essentially the same as the measurement for selecting the simplest example mentioned in Section 5.2.1. If an example is the simplest one for a model, the model will take the least effort to learn it.

- *Size-based effort* reflects the size of the model being built. Usually the larger the size of a model, the more effort the model needs to take. For a decision tree T_i, the size-based effort can be the number of the nodes in T_i. The more nodes in a tree, the more size-based effort is needed to build the tree.

It is reasonable for approximating the learning effort with the two indirect approaches, error-based effort and size-based effort, as many learning algorithms attempt to reduce the training error (such as ANN), or to build small models (such as decision tree algorithms) during the learning. In addition, the two indirect approaches are good and universal to use, because they are independent of hardwares or other factors, and the learning effort can be calculated easily for learning algorithms.

The second measurement, *learning efficiency*, is also very important. It describes the number of examples a learner needs to build a good model, i.e., a model with low predictive error rate. The higher the learning efficiency, the fewer the examples are needed. For example, given two learners A and B, if A needs 100 examples to build a model with 0.1 predictive error rate and B needs 120 examples to reach the same error rate, we say A has higher learning efficiency than B.

The last measurement, *volatility*, is important and it is used to evaluate how stable (volatile) a learner's performance is. It is the average of the standard deviation of the predictive error rate for all of the models built during the learning process. Accordingly it reflects the varying range of the error rate of a learner from different runs. If the error rate (accuracy) of a learner varies greatly from different runs, the volatility will be high. To calculate the volatility, an algorithm has to run k times on one dataset. More specifically, when we apply an algorithm on a dataset

having n examples, we will have n models[3] M_i $(0 < i <= n)$ in total during the process of learning the n examples. For each M_i, we have a prediction error rate, E_i $(0 < i <= n)$. If we run the algorithm on the dataset for k times, and then we will have $n \times k$ models M_{ij} with the predictive error rate E_{ij} $(0 < i <= n, 0 < j <= k)$. Then we calculate the standard deviation S_i $(0 < i <= n)$ of E_{ij} $(0 < j <= k)$. Finally, we get the volatility by averaging all the value of S_i $(0 < i <= n)$.

After presenting the three measurements, effort (error-based and size-based effort), learning efficiency and volatility, we will empirically study the performances of S2C and C2S in the following section.

5.5 Experiments

To compare with S2C and C2S, we also implement a benchmark learning strategy, called *Random*, which randomly selects examples from a given dataset, and rebuilds a new model with all selected examples [4]. This learning process repeats until certain stopping criterion is met. To make the three learning algorithms (S2C, C2S and Random) comparable, they all take C4.5 as the base learner.

The experiments are conducted on 10 UCI [2] datasets including anneal, autos, breast-cancer, colic, diabetes, ecoli, glass, heart-h, sonar and vote, which are commonly used in the supervised learning research area. Originally, autos and glass are of multi-class. However, to compare with the other binary-class datasets, we transform them to binary-class by keeping the majority class and merging all the other classes. The three learning strategies, S2C, Random and C2S are implemented based on the WEKA [104] source code. In the experiments, 10-fold cross validation is used and the t-test results are of 95% confidence.

5.5.1 Comparison of Effort

As discussed in Section 5.4, we are concerned with two kinds of effort, the error-based effort and the size-based effort. The error-based effort on the whole training set is the sum of the error when a learner processes and learns every training example. Similarly, the size-based effort on the whole training set is the sum of the tree nodes built during the whole learning process.

The experiment is conducted on all the 10 datasets to compare the error-based effort among S2C, Random and C2S. The average of error-based effort of the 5 runs on each dataset is shown in the upper part of Table 5.1. On average Random and C2S take about 1.3 times and 2 times as much error-based effort as S2C does respectively. To compare the error-based effort statistically, we also conduct t-test among the three algorithms, and the results show that S2C takes significant less error-based effort than Random on 9 datasets out of 10 and ties with Random on only one dataset. On the other hand, the error-based effort C2S takes is significantly more than S2C and Random on all of the 10 datasets.

[3] For easy description, here we assume that the model will be updated or rebuilt every time when a new example is taken in.

[4] We may also use ID5 [103], an incremental version of C4.5 to update the tree. However, it is shown [103] that ID5 produces identical tree as C4.5; thus we use C4.5 on the current training dataset directly.

	anneal	auto	breast-cancer	colic	diabetes	ecoli	glass	heart-h	sonar	vote
Average Error-based Effort										
S2C	6.9	7.6	10.0	16.0	41.0	7.0	11.6	13.3	12.5	7.1
Rnd	7.1	10.5	12.0	22.0	59.0	8.0	16.0	20.3	16.2	9.0
C2S	23.7	14.5	23.0	30	73.5	19.3	20.1	22.3	21.4	15.9
Average Size-based Effort										
S2C	3357	1198	3141	2841	21157	1478	1470	1792	2711	465
Rnd	14747	2937	12163	7482	49042	4399	4309	4665	3824	1034
C2S	51327	7427	27501	14019	95623	12057	7343	9055	10107	3721

Table 5.1: Comparison of error-based effort and size-based effort. In the table, Rnd stands for Random.

In addition to the error-based effort, we also show the average of size-based effort of the 5 runs on each dataset in the lower part of Table 5.1. On average, Random and C2S take about 2.6 times and 6 times as much size-based effort as S2C does. Furthermore, the t-test results show that statistically S2C takes significantly less effort than Random and C2S on all the 10 datasets without exception; while C2S takes the most size-based effort among the three algorithms. The reason is that S2C updates tree locally instead of Random and C2S rebuild trees. C2C takes even more the size-based effort than Random, because it prefers the examples that are different from the prediction of the current model, and consequently the tree model will have more nodes, i.e., more size-based effort.

The experimental results have shown that S2C does take the least error-based and size-based effort to learn a model comparing to Random and C2S. It is evident that learning simple examples first can reduce the effort for learning the complex examples during the later stage, and consequently the total cost is minimized. Opposite to S2C, the results show that C2S takes the most effort for learning the models. Does the maximal effort result in high learning efficiency?

5.5.2 Comparison of Learning Efficiency

To confirm the expectation that C2S can learn a model with higher efficiency (i.e., learn a good model with fewer examples), we present the average results of learning efficiency in Figure 5.1 (only four datasets are shown due to similar results). The x axis is the number of examples being learned and the y axis is the average predictive error rate of the models built from the corresponding examples. It shows clearly that the error rate of C2S reduces the fastest with increasing the number examples learned. This is because C2S takes the most effort to learn the most complex example, i.e., the example that its prediction is the most different from the current model, and consequently the current model can be improved faster than Random and S2C. To the opposite of C2S, S2C has the lowest learning efficiency in learning the models. This is because it prefers to learn the simplest example which is homogeneous with the current model and cannot improve the model much.

Meanwhile, we notice that the final error rate of S2C is slightly higher than Random and C2S. The reason is that S2C updates its model locally which is not as optimal as the models

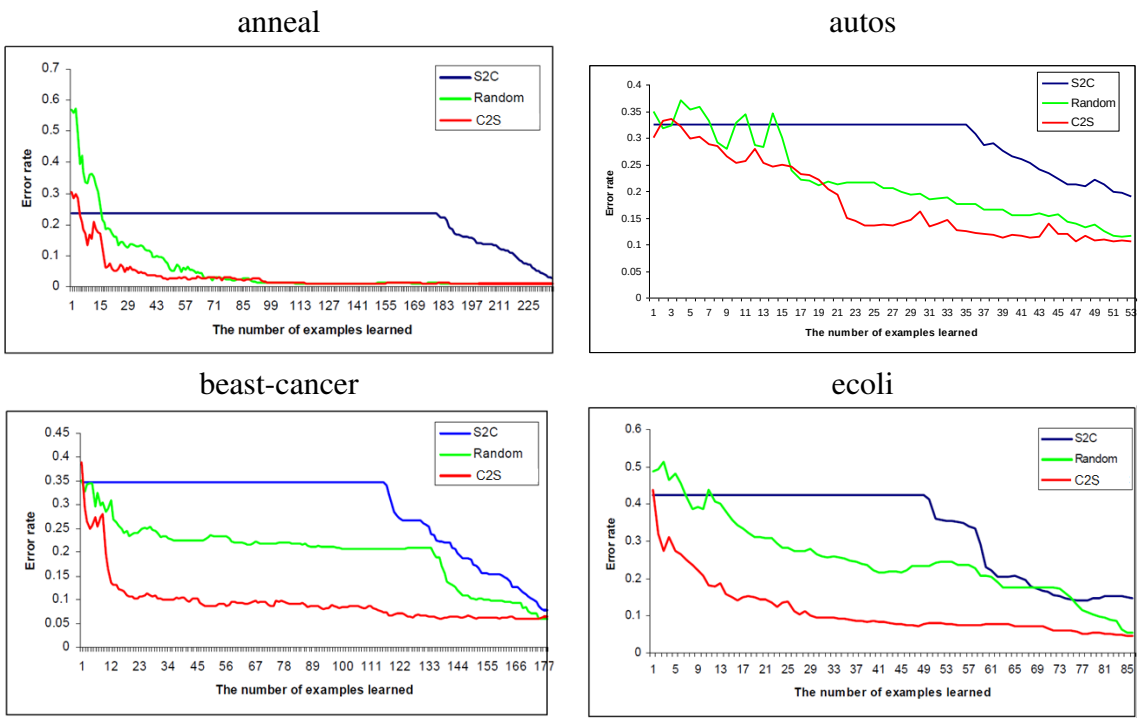


Figure 5.1: Learning efficiency.

that are rebuilt by Random and C2S with all selected examples. Further studies on how to improve the error rate of S2C will be discussed in Section 5.5.4.

5.5.3 Comparison of Volatility

Volatility indicates the stability of the prediction error rate from different runs as mentioned in Section 5.4. To compare the volatility, we run each algorithm on each dataset for 5 times and calculate the standard deviation of the error rate.

We present the volatility of S2C, Random and C2S on each dataset in Table 5.2. The volatility of S2C is almost 0 on all the datasets. Random is much more volatile than S2C, and C2S is about 3 times as volatile as Random. It is clear that S2C is the most stable among the three algorithms. The reason is that, for the same iteration, the simplest examples S2C takes in for different runs may different, but they do not change the current model much. Consequently the predictive error rate keeps relatively stable, i.e., low volatility. On the other hand, the most complex examples selected in C2S may change the current model by a wide range, and the predictive error rate varies accordingly, i.e., high volatility.

In addition, we show the stability of the error rate during the learning process in Figure 5.2. Due to similar results, only the details of the two datasets, colic and heart-h are presented. The x axis indicates the iterations of updating the trees and the y axis is the one-run predictive error rate of the model built in the corresponding iteration. As S2C takes in all the examples of majority class first, the process of building trees actually starts from taking in the first minority example, and consequently its update iterations are much fewer than that of Random and C2S.

	anneal	auto	breast-cancer	colic	diabetes	ecoli	glass	heart-h	sonar	vote
S2C	6E-7	.0	2E-8	3E-6	.0	1E-8	.0	5E-7	4E-6	6E-6
Rnd	.011	.032	.016	.003	.024	.039	.039	.032	.067	.020
C2S	.030	.051	.012	.025	.027	.023	.024	.028	.068	.071

Table 5.2: The comparison of volatility among the three learning algorithms. In the table, Rnd stands for Random.

From Figure 5.2, we can see clearly that S2C decreases more steadily than Random and C2S with an increasing number of training examples. This is also because the simplest example that S2C takes in does not change the model much and results in the steady reduce of error reduction. However, one might notice that the error rate of C2S on one run does not reduces as fast as the average error rate as we show in Section 5.5.2. It is also evident that the volatility of S2C is high, i.e., large variation of S2C on error rate from different runs.

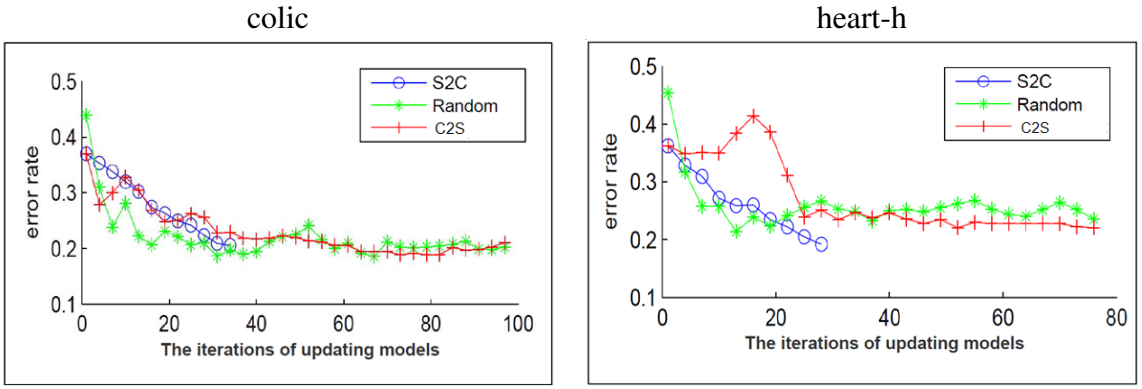


Figure 5.2: The error rate curve vs. iterations of updating models.

In summary, S2C does learn models with the minimal effort but with the lowest learning efficiency; while C2S does learn models with the highest efficiency by taking the maximal effort. In addition, the learning process of S2C is stable; while the process of C2S is volatile.

5.5.4 Reducing the Error Rate of S2C Further

In Section 5.5.2, we have noticed that the predictive error rate of the models built by S2C is slightly higher than C2S and Random. To find out whether the difference is statistically significant or not, we conduct t-test on the 10 datasets, and the results show that S2C is worse than C2S and Random on some datasets. More specifically, S2C ties with Random and C2S on 7 datasets, but loses on 3 datasets. The results are shown in the upper part of Table 5.3. The reason is that S2C only updates the current tree model at its fringe when the new simplest example is selected. It is extremely local and myopic. On the other hand, Random and C2S rebuild "global" trees with all the selected examples, and the final tree is built on the whole dataset. The "global" trees are expected to be optimal.

	anneal	auto	breast-cancer	colic	diabetes	ecoli	glass	heart-h	sonar	vote
S2C vs. Rnd	L	L	T	T	T	L	T	T	T	T
Rnd vs. C2S	T	T	T	T	T	T	T	T	T	T
C2S vs. S2C	W	W	T	T	T	W	T	T	T	T
K = 10										
S2C vs. Rnd	L	T	T	T	T	T	T	T	T	T
Rnd vs. C2S	T	T	T	T	T	T	T	T	T	T
C2S vs. S2C	W	T	T	T	T	T	T	T	T	T

Table 5.3: The t-test results on error rate (L: lose; T: tie; W: win). In the table, Rnd stands for Random.

To avoid the greedy updating strategy of S2C without increasing the effort too much, we design a "mini-review" strategy. The strategy is that after learning K examples, S2C will "review" them and use them together to expand the fringe of the tree. This "mini-review" process is similar to the summarizing process in human learning. The lower part of Table 5.3 shows the experimental results of the 10 datasets when $K=10$. It shows that S2C has almost the same low error rate as the "global" tree of Random and C2S.

In summary, the experimental results confirm our expectation that S2C takes minimal effort to learn a model with lowest learning efficiency; while C2S takes maximal effort to learning a model with the highest efficiency. Furthermore, the learning process of S2C is more stable than C2S. In addition, with our "mini-review" strategy, S2C can work as well as the global algorithm Random and C2S in terms of the predictive error rate. Thus, we can conclude that S2C and C2S have their own advantages and disadvantages, and the determination of using S2C or C2S depends on the learning goal in real applications.

5.6 Summary

In this chapter, to study the learning effort of active learning, we present the paradigm of learning actively with minimal/maximal effort. Under this paradigm, we propose two learning algorithms. One is S2C for achieving the goal of learning a model with minimal effort, and the other one is C2S for the goal of learning a model fast with maximal effort. S2C prefers to learn the simplest examples and leaves the complex example to learn during the later stage. C2S is opposite to S2C and prefers taking maximal effort to learn the most complex example first such that a good model can be built fast. Experimental results show that S2C does take much less effort to learn a model than Random and C2S, and reaches very similar low error rate. On the other hand, C2S can learn a good model fast by taking maximal effort. Furthermore, the learning process of S2C is much more stable (less volatile) than that of Random and C2S. As S2C and C2S have different advantages and disadvantages, they can be applied to different learning problems in real-world applications.

Chapter 6

Conclusions and Future Work

In this chapter, first of all we will summarize the work and the contributions in this thesis. Then we will discuss our future work.

6.1 Conclusion

Traditionally learners in active learning focus on the final model learned and the number of labeled examples needed for learning a good model. In addition to the goals of traditional active learning, we focus on controlling the learning process, i.e., the sequence of examples selected to learn, to achieve four different goals: minimizing the number of mistakes, maximizing the learning efficiency, minimizing the learning effort and maximizing the learning effort. These goals are common and important in machine learning.

To achieve the goals, we addressed four new learning paradigms in this thesis, and under each paradigm we proposed a novel and efficient learning algorithm to solve the related learning problems.

In Chapter 2, we proposed the paradigm of *learning actively and conservatively*, and to solve the learning problems under the paradigm we further proposed the learning algorithm *MCL*. More specifically,

1. The paradigm of learning actively and conservatively allows learners to select unlabeled examples actively to predict their labels and the true label will be revealed after each prediction. Its goal is to minimize the number of mistakes in predicting unlabeled examples during the learning process.

2. To minimize the number of mistakes, MCL is proposed and its basic idea is to repeatedly select the next unlabeled example that can be predicted by the learner with the highest certainty to predict. This work of MCL and its application on Zoombinis game is published on *Proceedings of International Conference on Artificial Intelligence and Education (ICAIE), 2010* [63].

3. According to various real-world applications, we further implemented two types of MCL: MCL-b and MCL-1. In MCL-b, a learner must select and predict all of the remaining examples as either positive or negative classes, and its goal is to minimize the number of mistakes in predicting the examples of both classes.

4. In MCL-1, a learner only needs to select and predict one class of examples. The goal is to retrieve the examples of the class that we care about. The work of MCL-b and MCL-1 is published on *The 7th International conference on Advanced Data Mining and Applications (ADMA'11)* [62].

5. The empirical studies show that MCL-b does reduce the number of mistakes on the educational game Zoombinis and UCI datasets, and MCL-1 also can retrieve more positive examples on UCI datasets and a marketing dataset.

In Chapter 3, we proposed the paradigm of *learning actively and aggressively*, and the learning algorithms *c-certainty learning* and *BMO* to solve the learning problems under this paradigm. More specifically,

1. The paradigm of learning actively and aggressively allows that multiple imperfect oracles are available for returning both labels and confidences, and that the noise level in oracles is example-dependent. Its goal is to learn a good model with fewer queries by guaranteeing label quality.

2. To guarantee the label quality, c-certainty learning is proposed and its learner assesses the quality of the label of an example every time after obtaining a response from an oracle. If the label quality is higher than or equal to a given threshold c, the example and its label will be added to the labeled data; otherwise, more oracles will be queried.

3. Given the noise level of each oracle to be example-dependent, BMO is proposed to select the best oracles to query for each given example. With BMO, fewer queries on average are expected for a label to meet the given threshold c.

4. Empirical studies are conducted on both faithful and unfaithful oracles. The results show that BMO wins other learning strategies in most of the cases for both faithful and unfaithful oracles. The work of c-certainty learning and BMO is published in *The 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining. May, 2012* [64].

In Chapter 4, we proposed the paradigm of *learning actively with conservative-aggressive tradeoff*, and the learning algorithms *DAL* to solve the learning problems under this paradigm. More specifically,

1. Learners under the paradigm of learning actively with conservative-aggressive tradeoff are allowed to obtain the labels through two actions: making prediction and querying an oracle. Certain cost has to be paid for making wrong prediction or for querying an oracle. The goal is to build a good model by paying minimal cost for obtaining labels (predicting and querying).

2. To minimize the cost, the learning algorithm DAL proposed always selects the example leading to correct (optimal) actions and prefers the action that is expected to cost less during the learning process. The examples that will likely lead to wrong actions will be learned during the later stage, as the learner may become more reliable (the probability is more accurate) and the action taken will become more accurate with more example being learned.

3. Empirical studies showed that DAL outperforms other learning strategies significantly. In addition, we showed that well-calibrated classifier can boost the performance of DAL and the selection method of the starting point used in DAL helps to reduce the total cost. The work of DAL is submitted to *The IEEE International Conference on Data Mining (ICDM), 2012*.

In Chapter 5, we proposed the paradigm of *learning actively with minimal/maximal effort*, and the learning algorithms *S2C* and *C2S* to solve the learning problems under this paradigm. More specifically,

1. The paradigm of learning actively with minimal/maximal effort is proposed to study the relations between learning effort and the performance on achieving goals. Under this paradigm, the labels of the examples are all provided and learners are allowed to select examples actively to learn. The goal is to control the learning process by selecting examples actively such that the learning can be accomplished with minimal effort or good models can be built fast with maximal effort.

2. For achieving the goal of learning a good model with minimal effort, we propose the S2C learning algorithm by explicitly applying a human learning theory to supervised machine learning. The basic idea of S2C is to select those examples that are close to the current model's prediction (thus simple) and update the model with them. The complex examples are left to learn during the later stage.

3. For achieving the goal of learning fast with maximal effort, we proposed the C2S learning algorithm. C2S repeatedly chooses to learn the example that its prediction is the most different from the true label during the learning process. Due to the difference, the example is the most informative for the learner. Thus, it is able to improve the learner the most, and the learning is expected to be efficient.

4. The empirical studies show that S2C does take less effort in learning a good model; while C2S learns a model very fast by taking maximal effort. Thus, in real application the selection of the two learning algorithms depends on the learning goal. The work of S2C and C2S is published in *Advances in Knowledge Discovery and Data Mining, 2010* [61].

6.2 Future Work

We have investigated how to control the learning process, i.e., the sequence of selecting examples to learn, for achieving different learning goals. The empirical studies have shown that the proposed learning paradigms and algorithms work well in solving the learning problems. However, there are still some potential to improve our current work.

Firstly, under all the four learning paradigms we proposed, the misclassification cost of false positive and false negative are assumed to be equal. In reality, sometimes it is not the case. In the future, we will adapt our learning paradigm and algorithms so that the learning problems with uneven misclassification costs can also be solved well.

Under the paradigm of learning actively with conservative-aggressive tradeoff, when we make prediction on an example, we assume the true label will be revealed to the learner. In some real-world applications, only positive (or negative) predictions can reveal the true labels. For example, the automatic advertisement system will never get the true label (never know if a website is suitable to advertise) if it does not buy an advertisement slot on a website. The current algorithm DAL may not be applicable directly in the case, since the learner cannot get labels for the examples with probability close to 0. We will study this issue in the future work.

Bibliography

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

[2] A. Asuncion and D. Newman. UCI machine learning repository. URL http://www.ics.uci.edu/ mlearn/mlrepository.html, 2007.

[3] M.F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72. ACM, 2006.

[4] J. Baldridge and A. Palmer. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 296–305. Association for Computational Linguistics, 2009.

[5] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *The Journal of Machine Learning Research*, 5:255–291, 2004.

[6] P.L. Bartlett and M.H. Wegkamp. Classification with a reject option using a hinge loss. *The Journal of Machine Learning Research*, 9:1823–1840, 2008.

[7] S. Basu, A. Banerjee, and R.J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM international conference on data mining*, pages 333–344, 2004.

[8] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Mach. Learn.*, 36(1-2):105–139, 1999.

[9] Shai Ben-David, Eyal Kushilevitz, and Yishay Mansour. Online learning versus offline learning. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, pages 38–52, London, UK, 1995. Springer-Verlag.

[10] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[11] M.J.A. Berry and G.S. Linoff. *Data mining techniques*. Wiley-India, 2009.

[12] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 440, 2007.

[13] K. Brinker. Incorporating diversity in active learning with support vector machines. In *In Proceedings of the 20th International Conference on Machine Learning*, volume 20, page 59, 2003.

[14] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

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

[16] S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. *Advances in neural information processing systems*, 20:2, 2007.

[17] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[18] T. Dietterich. Ensemble methods in machine learning. *Multiple classifier systems*, pages 1–15, 2000.

[19] T.G. Dietterich. Machine-learning research. *AI magazine*, 18(4):97, 1997.

[20] Z. Dong-lin. Krashens Input Hypothesis and English classroom teaching.

[21] P. Donmez and J.G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628. ACM, 2008.

[22] P. Donmez, J.G. Carbonell, and J. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268. ACM, 2009.

[23] Pinar Donmez and Jaime G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 619–628, New York, NY, USA, 2008. ACM.

[24] J. Du and C.X. Ling. Active learning with generalized queries. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 120–128. Ieee, 2009.

[25] J. Du and C.X. Ling. Active learning with human-like noisy oracle. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 797–802. IEEE, 2010.

[26] J. Du and C.X. Ling. Asking generalized queries to domain experts to improve learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(6):812–825, 2010.

[27] J. Du, E.A. Ni, and C.X. Ling. Adapting cost-sensitive learning for reject option. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1865–1868. ACM, 2010.

[28] Jun Du and Charles X. Ling. Active learning with human-like noisy oracle. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 797–802, Washington, DC, USA, 2010. IEEE Computer Society.

[29] C. Elkan. Naive bayesian learning. *Department of Computer Science and Engineering, University of California, San Diego CS97-557*, 1997.

[30] F. Famili, W.M. Shen, R. Weber, E. Simoudis, et al. Data pre-processing and intelligent data analysis. *International Journal on Intelligent Data Analysis*, 1(1), 1997.

[31] C. Ferri, P. Flach, and J. Hernández-Orallo. Delegating classifiers. In *Proceedings of the twenty-first international conference on Machine learning*, page 37. ACM, 2004.

[32] Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.

[33] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern recognition*, 33(12):2099–2101, 2000.

[34] G.P.C. Fung, JX Yu, H. Lu, and PS Yu. Text classification without negative examples revisit. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):6–20, 2006.

[35] R.M. Gagné. *The conditions of learning*. Holt, Rinehart & Winston New York, 1970.

[36] Salvador Garc and Francisco Herrera. An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9:2677–2694, December 2008.

[37] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, 13(4):215–223, 2000.

[38] Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire. Learning binary relations and total orders. *SIAM J. Comput.*, 22(5):1006–1034, 1993.

[39] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. *Advances in Neural Information Processing Systems (NIPS)*, (20):593–600, 2007.

[40] M.T. Hagan, H.B. Demuth, M.H. Beale, and Boulder University of Colorado. *Neural network design*. PWS Pub, 1996.

[41] S. Hanneke. *Theoretical foundations of active learning*. ProQuest, 2009.

[42] G. Heigold, T. Deselaers, R. Schl
"uter, and H. Ney. Modified mmi/mpe: A direct evaluation of the margin in speech recognition. In *Proceedings of the 25th international conference on Machine learning*, pages 384–391. ACM, 2008.

[43] T. Hofmann and J.M. Buhmann. Active data clustering. *Advances in Neural Information Processing Systems*, pages 528–534, 1998.

[44] S.C.H. Hoi, R. Jin, and M.R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*, pages 633–642. ACM, 2006.

[45] C.H. Hsu, W. Feng, and J.S. Archuleta. Towards efficient supercomputing: A quest for the right metric. In *Proceedings of the High-Performance Power-Aware Computing Workshop*. Citeseer, 2005.

[46] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[47] R.D. King, K.E. Whelan, F.M. Jones, P.G.K. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[48] S.B. Kotsiantis, I.D. Zaharakis, and P.E. Pintelas. Supervised machine learning: A review of classification techniques. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 160:3, 2007.

[49] S.D. Krashen. *The input hypothesis: Issues and implications*. Addison-Wesley Longman Ltd, 1985.

[50] K.A. Krueger and P. Dayan. Flexible shaping: how learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

[51] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, and R.P.W. Duin. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, 27(8):908–917, 2006.

[52] S. Lange and G. Grieser. On the power of incremental learning. *Theoretical Computer Science*, 288(2):277–307, 2002.

[53] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

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

[55] R. Lomasky, C. Brodley, M. Aernecke, D. Walt, and M. Friedl. Active class selection. *Machine Learning: ECML 2007*, pages 640–647, 2007.

[56] C.D. Manning, P. Raghavan, and H. Schze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[57] Y. MarcAurelio Ranzato, L. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, 20.

[58] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 483–486. IEEE, 2004.

[59] T.M. Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*.

[60] M. Mohiyuddin, M. Murphy, L. Oliker, J. Shalf, J. Wawrzynek, and S. Williams. A design methodology for domain-optimized power-efficient supercomputing. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12. ACM, 2009.

[61] E. Ni and C. Ling. Supervised learning with minimal effort. *Advances in Knowledge Discovery and Data Mining*, pages 476–487, 2010.

[62] E. Ni and C. Ling. Direct marketing with fewer mistakes. *Advanced Data Mining and Applications*, pages 256–269, 2011.

[63] E. Ni and C. Ling. Self-directed learning. 2011.

[64] Eileen A. Ni and Charles X. Ling. Active learning with c-certainty. In *PAKDD (1)*, pages 231–242, 2012.

[65] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 625–632, New York, NY, USA, 2005. ACM.

[66] French Telecom Company Orange. Kdd cup 2009. URL http://www.kddcup-orange.com/data.php, 2009.

[67] J.M. Pearce. *Animal learning and cognition: An introduction* . Psychology Press, 2008.

[68] T. Pietraszek. On the use of roc analysis for the optimization of abstaining classifiers. *Machine learning*, 68(2):137–169, 2007.

[69] B. Plannerer. An introduction to speech recognition. *March28*, 2005.

[70] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[71] J.R. Quinlan. Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[72] V.C. Raykar, S. Yu, L.H. Zhao, A. Jerebko, C. Florin, G.H. Valadez, L. Bogoni, and L. Moy. Supervised Learning from Multiple Experts: Whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 889–896. ACM, 2009.

[73] N.S. Rebello, L. Cui, A.G. Bennett, D.A. Zollman, and D.J. Ozimek. Transfer of learning in problem solving in the context of mathematics and physics. *Learning to Solve Complex Scientific Problems, Lawrence Earlbaum, Mahwah, NJ*, 2007.

[74] R. Reichart, K. Tomanek, U. Hahn, and A. Rappoport. Multi-task active learning for linguistic annotations. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. Citeseer, 2008.

[75] Ronald L. Rivest and Yiqun Lisa Yin. Being taught can be faster than asking questions. In *COLT '95: Proceedings of the eighth annual conference on Computational learning theory*, pages 144–151, New York, NY, USA, 1995. ACM.

[76] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 441–448, 2001.

[77] S.J. Russell, P. Norvig, J.F. Canny, J.M. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, NJ, 1995.

[78] M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

[79] K. Sarinnapakorn and M. Kubat. Combining Subclassifiers in Text Categorization: A DST-Based Solution and a Case Study. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1638–1651, 2007.

[80] A.I. Schein and L.H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.

[81] B. Settles. From theories to queries: Active learning in practice.

[82] B. Settles. Active Learning Literature Survey. *Machine Learning*, 15(2):201–221, 1994.

[83] B. Settles. Active learning literature survey. *Science*, 10(3):237–304, 1995.

[84] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[85] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[86] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1069–1078, 2008.

[87] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *In Advances in Neural Information Processing Systems (NIPS*. Citeseer, 2008.

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

[89] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM.

[90] V.S. Sheng, F. Provost, and P.G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

[91] X. Shi, W. Fan, and J. Ren. Actively transfer domain knowledge. *Machine Learning and Knowledge Discovery in Databases*, pages 342–357, 2008.

[92] G. Shmueli, N.R. Patel, and P.C. Bruce. *Data mining for business intelligence*. John Wiley & Sons, 2007.

[93] Amit Singhal and Google Inc. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.

[94] R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008.

[95] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[96] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[97] S. Stolbach. Active learning models and noise. 2007.

[98] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[99] M.E. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, page 886. ACM, 2007.

[100] S. Thrun. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

[101] S. Tong. *Active learning: theory and applications*. PhD thesis, Citeseer, 2001.

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

[103] P.E. Utgoff. Improved training via incremental learning. In *Proceedings of the sixth international workshop on Machine learning table of contents*, pages 362–365. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989.

[104] WEKA Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/˜ml/weka.

[105] A.J. Wills. Prediction errors and attention in the presence and absence of feedback. *Current Directions in Psychological Science*, 18(2):95, 2009.

[106] K.W. Wong, S. Zhou, Q. Yang, and J.M.S. Yeung. Mining customer value: from association rules to direct marketing. *Data Mining and Knowledge Discovery*, 11(1):57–79, 2005.

[107] Yan Yan, Romer Rosales, Glenn Fung, and Jennifer Dy. Active learning from crowds. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1161–1168, New York, NY, USA, June 2011. ACM.

[108] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002.

[109] Y. Zhang. Multi-task active learning with output constraints. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[110] Y. Zheng, S. Scott, and K. Deng. Active learning from multiple noisy labelers with varied costs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 639–648. IEEE, 2010.

Appendix A

Appendix

The Formula 3.1 in Chapter 3 is as follows:

$$C(T_P|A^n) = \begin{cases} \frac{p(T_P)\times f_n}{p(T_P)\times f_n + p(T_N)\times(1-f_n)}, & \text{if } n = 1 \text{ and } A_n = \{P, f_n\} \\ \\ \frac{C(T_P|A^{n-1})\times f_n}{C(T_P|A^{n-1})\times f_n + (1-C(T_P|A^{n-1}))\times(1-f_n)}, & \text{if } n > 1 \text{ and } A_n = \{P, f_n\} \end{cases}$$

A.1 Derivation of Formula 3.1

According to Bayes rule, the left of Formula 3.1 can be transformed as follows.

$$\begin{aligned} & C(T_P|A^n) \\ = & \frac{P(A^n|T_P)\times P(T_P)}{P(A^n)} \\ = & \frac{P(A^{n-1}, A_n|T_P)\times P(T_P)}{P(A^n)} \\ = & \frac{P(A^{n-1}|T_P)\times P(T_P)\times P(A_n|T_P)\times P(A^{n-1})}{P(A^{n-1})\times P(A^n)} \\ = & C(T_P|A^{n-1})\times p(A_n|T_P)\times \frac{p(A^{n-1})}{p(A^n)} \end{aligned} \tag{A.1}$$

The last item in Equation A.1 can be further transformed as follows.

$$\begin{aligned}
&\frac{p(A^{n-1})}{p(A^n)} \\
=\ & \frac{p(A^{n-1})}{p(A^n|T_P) \times p(T_P) + p(A^n|T_N) \times p(T_N)} \\
=\ & \frac{p(A^{n-1})}{p(A^{n-1}|T_P) \times p(T_P) \times p(A_n|T_P) + p(A^{n-1}|T_N) \times p(T_N) \times p(A_n|T_N)} \\
=\ & \frac{1}{(C(T_P|A^{n-1})) \times p(A_n|T_P) + C(T_N|A^{n-1}) \times p(A_n|T_N)}
\end{aligned}$$

As $A_n = (P, f_n)$,

$$\begin{aligned}
&C(T_P|A^n) \\
=\ & \frac{C(T_P|A^{n-1}) \times p(A_n|T_P)}{C(T_P|A^{n-1}) \times p(A_n|T_P) + (1 - C(T_P|A^{n-1})) \times p(A_n|T_N)} \\
=\ & \frac{C(T_P|A^{n-1}) \times f_n}{C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)}
\end{aligned}$$

Thus, Formula 3.1 holds.

A.2 The Proof of Non-monotonic of Formula 3.1

Proof:

$$\begin{aligned}
&C(T_P|A^n) - C(T_P|A^{n-1}) \\
=\ & \frac{C(T_P|A^{n-1}) \times f_n}{C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)} - C(T_P|A^{n-1}) \\
=\ & \frac{C(T_P|A^{n-1}) \times f_n - C(T_P|A^{n-1}) \times (C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n))}{C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)} \\
=\ & C(T_P|A^{n-1}) \times \frac{f_n - C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)}{C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)}
\end{aligned} \tag{A.2}$$

As $C(T_P|A^{n-1}) \geq 0$ and $C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n) \geq 0$, the update of confidence is monotonic if $f_n - C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n)$ is guaranteed to be greater than or equal to 0.

$$\begin{aligned}
&f_n - C(T_P|A^{n-1}) \times f_n + (1 - C(T_P|A^{n-1})) \times (1 - f_n) \\
=\ & (2 \times f_n - 1) \times (1 - C(T_P|A^{n-1})
\end{aligned} \tag{A.3}$$

As $1 - C(T_P|A^{n-1}) \geq 0$, if $f_n \geq 0.5$, $C(T_P|A^n) - C(T_P|A^{n-1}) \geq 0$; otherwise, if $f_n < 0.5$, $C(T_P|A^n) - C(T_P|A^{n-1}) < 0$. Thus, Formula 3.1 is not monotonic.

Curriculum Vitae

Name:	(Eileen) Ai-Ling Ni
Degrees:	Guangxi Normal University, China 2003 - 2006 M.Sc. University of Western Ontario, Canada 2008 - 2012 Ph.D.
Honours and Awards:	Second place, Data mining Session at UWO Research in Computer Science, 2012
Related Work Experience:	Teaching Assistant The University of Western Ontario 2008 - 2012 Research Assistant The University of Western Ontario 2008 - 2012

Publications:

1. Eileen A. Ni and Charles X. Ling. Active Learning with c-Certainty. Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2012.

2. Eileen A. Ni and Charles X. Ling. Direct Marketing with Fewer Mistakes. Advanced Data Mining and Applications, 2011.

3. Eileen A. Ni and Charles X. Ling. Learning with Guaranteed Label Quality. The 10th International Conference on Machine Learning and Applications Workshops, 2011.

4. Eileen A. Ni, Charles X. Ling. Supervised Learning with Minimal Effort. Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2010.

5. Jun Du, Eileen. A. Ni and Charles X. Ling. Adapting Cost-sensitive Learning for Reject Option. Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM), 2010.

6. Eileen A. Ni, Charles X. Ling. Self-Directed Learning. Proceedings of International Conference on Artificial Intelligence and Education (ICAIE), 2010.

7. Shengli Sheng, Charles X. Ling, Eileen A. Ni and Shichao Zhang. Test Strategies for Cost-Sensitive Decision Trees. Proceedings of 21st National Conference on Artificial Intelligence (AAAI-06), July 16-20, 2006, Boston, Massachusetts, USA.

8. Eileen A. Ni, Xiaofeng Zhu, Chengqi Zhang. Any-Cost Discovery: Learning Optimal Classification Rules The 18th Australian Joint Conference on Artificial Intelligence (2005) .