Second International Workshop on Utility-Based Data Mining

Held in conjunction with the KDD conference, August 20, 2006

Workshop Chairs

Bianca Zadrozny
Gary Weiss
Maytal Saar-Tsechansky

The Association for Computing Machinery, Inc.
1515 Broadway
New York, New York 10036

ACM ISBN: 1-59593-440-5

Additional copies may be ordered prepaid from:

ACM Order Department	Phone: 1-800-342-6626
P.O. BOX 11405	(U.S.A. and Canada)
Church Street Station	+1-212-626-0500
New York, NY 10286-1405	(All other countries)
	Fax: +1-212-944-1318
	E-mail: acmhelp@acm.org

Printed in the U.S.A.

Preface

Early work in data mining and machine learning did not address the complex circumstances in which models are built and applied. It was assumed that a fixed amount of data was available and only simple objectives such as predictive accuracy were considered. Over time, there has been a growing interest in the data mining and machine learning communities in research addressing economic utility factors related to acquiring data, building models, and applying models. This interest in how utility considerations impact the data mining process led us to introduce the term *utility-based data mining* and to organize the First Workshop on Utility-Based Data Mining, which was held during last year's KDD conference.

The second workshop will again bring together researchers and practitioners who are involved in all aspects of utility-based data mining. Our goal is to continue to promote a study of all aspects of utility-based data mining and examine how utility considerations from different stages of the data mining process interact. We also would like to increase the communication between researchers and data mining practitioners as well as the communication between those in the data mining community and those in related communities, as evidenced by our choice of invited speakers.

We believe the very positive response we have had from both academia and industry during the first workshop indicates the importance of utility-based data mining research and hope that the second workshop will continue to promote a fruitful exchange of ideas to further advance the field.

We would like to thank all the researchers that submitted their recent work, our Program Committee, our panelists, and our invited speakers, Russell Greiner and Michael Littman, for their generous contributions to this workshop.

Bianca Zadrozny
Gary Weiss
Maytal Saar-Tsechansky

Table of Contents

UBDM-2006 Workshop Organizing and Program Committee

Workshop Organizers: Bianca Zadrozny, *Federal Fluminense University, Brazil*
Gary Weiss, *Fordham University, USA*
Maytal Saar-Tsechansky, *University of Texas at Austin, USA*

Program Committee: Naoki Abe, *IBM Research*
Alina Beygelzimer, *IBM Research*
Nitesh Chawla, *University of Notre Dame*
Ian Davidson, *State University of New York at Albany*
Chris Drummond, *National Research Council (Ottawa)*
Wei Fan, *IBM Research*
Tom Fawcett, *Stanford Computational Learning Laboratory*
Howard Hamilton, *University of Regina*
Robert Hilderman, *University of Regina*
Robert Holte, *University of Alberta*
Aleksander Kolcz, *AOL Inc.*
Charles Ling, *University of Western Ontario*
Dragos Margineantu, *Boeing Company*
Prem Melville, *University of Texas at Austin*
Ion Muslea, *Language Weaver, Inc.*
Claudia Perlich, *IBM Research*
Xingquan Zhu, *University of Vermont*

Workshop Program

8:30 – 8:40 **Opening Remarks and Welcome**

8:40 – 9:20 Invited Talk: Budgeted Learning of Probabilistic Classifiers
Russell Greiner

9:20 – 9:40 Maximizing Classifier Utility when Training Data is Costly
Gary Weiss and Ye Tian

9:40 – 10:00 Position Paper: Prediction Games in Infinitely Rich Worlds
Omid Madani

10:00 – 10:30 **Break**

10:30 – 10:50 Efficient Mining of Temporal High Utility Itemsets from Data Streams
Vincent Tseng, Chun-Ron Chu and Tyne Liang

10:50 – 11:10 A Unified Framework for Utility-Based Measures for Mining Itemsets
Hong Yao, Howard Hamilton and Liqiang Geng

11:10 – 11:30 Assessing the Interestingness of Discovered Knowledge Using a Principled Objective Approach
Robert Hilderman

11:30 – 11:50 Utility-Based Anonymization for Privacy Preservation with Less Information Loss
Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi and Ada Wai-Chee Fu

11:50 – 12:00 Discussion

12:00 – 1:30 **Lunch**

1:30 – 2:10 Invited Talk: Reinforcement Learning for Utility-Based Decisions
Michael Littman

2:10 – 2:30 Beyond Classification and Ranking: Constrained Optimization of the ROI
Lian Yan and Patrick Baldasare

2:30 – 2:50 Pricing Based Framework for Benefit Scoring
Nitesh Chawla and Xiangning Li

2:50 – 3:10 Maximum Profit Mining and Its Application in Software Development
Charles Ling, Victor Sheng, Tilmann Bruckhaus and Nazim Madhavji

3:10 – 3:30 Discussion

3:30 – 4:00 **Break**

4:00 – 4:40 Panel: Research Directions and New Applications of Utility-Based Data Mining
Russell Greiner, Michael Littman, Dragos Margineantu, Lian Yan, Gerald Fahner

4:40 – 4:50 **Concluding Remarks**

Budgeted Learning of Probabilistic Classifiers

Russell Greiner
Department of Computing Science
and Alberta Ingenuity Centre for Machine Learning
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
greiner@cs.ualberta.ca

ABSTRACT

Researchers often use clinical trials to collect the data needed to evaluate some hypothesis, or produce a classifier. During this process, they have to pay the cost of performing each test. Many studies will run a comprehensive battery of tests on each subject, for as many subjects as their budget will allow – i.e., "round robin" (RR). We consider a more general model, where the researcher can sequentially decide which single test to perform on which specific individual; again subject to spending only the available funds. Our goal here is to use these funds most effectively, to collect the data that allows us to learn the most accurate classifier.

We first explore the simplified "*coins version*" of this task. After observing that this is NP-hard, we consider a range of heuristic algorithms, both standard and novel, and observe that our "biased robin" approach is both efficient and much more effective than most other approaches, including the standard RR approach. We then apply these ideas to learning a *naive-bayes classifier*, and see similar behavior. Finally, we consider the most realistic model, where *both* the researcher gathering data to build the classifier, and the user (e.g., physician) applying this classifier to an instance (patient) must pay for the features used --- e.g., the researcher has $10,000 to acquire the feature values needed to produce an optimal $30/patient classifier. Again, we see that our novel approaches are almost always much more effective that the standard RR model.

This is joint work with Aloak Kapoor, Dan Lizotte and Omid Madani. See the Budgeted Learning Webpage at http://www.cs.ualberta.ca/~greiner/BudgetedLearning.

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

Reinforcement Learning and Utility-Based Decisions

Michael Littman
Department of Computer Science
Rutgers University
Piscataway, NJ, 08854-8019
littman@cs.rutgers.edu

ABSTRACT

In one model of utility-based data mining (UBDM), the primary concerns are the cost of acquiring data, the computational costs of mining the data, and the benefit of using the mined knowledge. Finding a truly optimal strategy over all these sources of utility is intractable. I will describe some recent trends in the reinforcement learning literature that deal with a set of analogous problems from a PAC perspective and I will attempt to connect these ideas back to the UBDM setting.

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

Maximizing Classifier Utility when Training Data is Costly

Gary M. Weiss and Ye Tian
Department of Computer and Information Science
Fordham University
441 East Fordham Road
Bronx, NY 10458

{gweiss, tian}@cis.fordham.edu

ABSTRACT
Classification is a well-studied problem in machine learning and data mining. Classifier performance was originally gauged almost exclusively using predictive accuracy. However, as work in the field progressed, more sophisticated measures of classifier *utility* that better represented the value of the induced knowledge were introduced. Nonetheless, most work still ignored the cost of acquiring training examples, even though this affects the *overall* utility of a classifier. In this paper we consider the costs of acquiring the training examples in the data mining process; we analyze the impact of the cost of training data on learning, identify the optimal training set size for a given data set, and analyze the performance of several progressive sampling schemes, which, given the cost of the training data, will generate classifiers that come close to maximizing the overall utility.

Categories and Subject Descriptors
I2.6 [Artificial Intelligence]: Learning – *induction.*
H2.8 [Database Management]: Database Applications – *data mining.*

General Terms
Algorithms, Performance, Economics

Keywords
Data mining, machine learning, induction, decision trees, utility-based data mining, cost-sensitive learning, active learning

1. INTRODUCTION
Classification is an important application area for data mining. The quality of a classifier is almost always measured exclusively by its performance on new examples. Originally only simple measures like predictive accuracy were used. However, as the field advanced and more complex problems were addressed, more sophisticated performance measures were introduced—measures that more accurately reflect *how* the classifier will be used in its target environment. Thus, it is now not uncommon for misclassification cost information to be considered when evaluating classifier performance or for profit/loss metrics to be used.

The ultimate goal of utility-based data mining [12] is to consider all utility considerations in the data mining process and maximize the utility of this entire process. In the case of classification, this translates to considering the costs associated with building the classifier and the costs/benefits associated with applying the classifier. As just mentioned, there has been a substantial amount of work in properly measuring the costs and benefits of applying the classifier. However, with the exception of some work from the active learning community, the costs associated with building the classifier are often ignored. This is a mistake, since the cost of building a classifier can be quite substantial. These costs may include the cost of acquiring the training cases, the cost of cleaning and preparing the training data, the cost of labeling the training data and the CPU time and hardware costs associated with building the classification model. These costs are described in more detail in Section 2.

In this paper we focus on the cost of acquiring complete, usable, training cases, where one has no control over which specific training examples can be acquired (this differentiates our work from the work on active learning). Thus, the value/utility of a classifier is the value associated with using the classifier minus the cost of the training data used to build the classifier. With this notion of utility, if classifier A performs only slightly worse than classifier B but is much less costly to generate, classifier A will be considered the better classifier. In Section 3.3 we formalize this notion of total utility so that we can precisely determine when one classifier is "better" than another. The main contribution of this paper is that we analyze the trade-off of between acquiring more training data (and the concomitant increase in predictive accuracy) and the cost of acquiring this data. We show that for each data set and learner there is an optimal training set size that maximizes the overall utility of the classifier. We then propose two progressive sampling schemes and demonstrate that they can be used to generate classifiers with near optimal overall utility.

2. THE COST OF TRAINING DATA
This paper focuses on the cost of training data and how it impacts the overall learning/data mining process. In this section we describe in some detail what we mean by the cost of training data. We then motivate this research by describing three data mining scenarios.

UBDM '06, August 20, 2006, Philadelphia, Pennsylvania, USA.

2.1 What are the Costs of Training Data?

In this paper we are concerned with the cost of acquiring labeled examples that can be used to train a classifier. In this section we describe several of the costs associated with acquiring training data. The primary cost we are concerned with is the cost of acquiring the raw, but labeled, training examples. Specifically, we assume that a data-mining practitioner can request a batch of b examples with some cost c. We assume no restrictions on the value of b, although depending on the domain, there may be restrictions on b (i.e., you may not have total flexibility in specifying the batch size). We also assume that c does not depend on the specific examples, although our analysis is not highly dependent on this assumption. Turney [9], who provides a fairly comprehensive list of costs associated with classifier learning, refers to this cost as the "cost of cases."

Training data often needs to be prepared before it can be used for learning. Data preparation may include cleaning the data, removing outliers and transforming the data into a format suitable for mining. These data preparations steps often have a cost associated with them—especially if some manual effort is required. Thus, even if training cases are free, there still may be a cost for generating *usable* training cases. Thus the research described in this paper is relevant when there are data preparation costs.

In many situations *unlabeled* training cases may be freely available but there may be a cost for labeling them. Turney refers to this as the "cost of teacher" [9]. Our research is also relevant in this situation. However, when there is a cost of teacher one may selectively label examples, which is an example of active learning [1]. Because we assume that the user has no control of the examples that are requested, we do not cover active learning in this paper and hence our work does not apply for situations where active learning is used. Nonetheless, in practice active learning is not always utilized when there is a cost of teacher. Active learning can also be used when there is a cost associated with measuring specific feature values. We do not consider this type of cost. In summary, our work differs from the work on active learning in that we focus on the cost of acquiring complete cases (cost of cases), in which one has little or no control over which specific cases are acquired.

2.2 Motivating Examples

We believe that for many domains there are costs associated with acquiring training data and, just as importantly, one has some *choice* in the number of training examples that can be acquired. In these situations, it is essential to consider the cost of data acquisition if one is to maximize the overall utility of the classifier. Three examples are provided in this section.

Based on past experience in industry, we know that it is quite common to acquire training data from an external vendor, whose business relies on selling information. For example, in order to build classification models to classify businesses, we acquired summary business data from D&B and detailed survey data from Ziff Davis. Companies that sell data typically do not require their customers to buy either "all or nothing." Depending on the company, a customer may be allowed to choose a number of records and pay based on that number or choose from a set of predetermined levels of data coverage and pay based on that level of coverage.

A second example comes from the task of classifying a phone line as belonging to a residential or business customer based on the pattern of phone usage. Information describing every phone call is recorded as a call-detail record, which is generated in real-time and stored sequentially. Because the classification task requires examples to be at the phone-line level, all of the call detail records associated with each phone number must be aggregated. Given that billions of call-detail records are generated every month and because the aggregation step requires sorting all of these records, this data preparation step is very expensive in terms of both disk space and CPU time. Thus, in this domain the training examples were expensive even though the raw data was essentially free.

The third example comes from the domain of game playing. If our goal is to learn something about an opponent so that we can design a game-playing strategy tailored to this opponent, the training data will usually be costly, in terms of time or in money if betting is involved. For example, if you want to learn something about an opponent in poker "you may play only 50 or 100 hands against a given opponent and want to quickly learn how to exploit them" [3].

We hope that the descriptions of the various costs and these two simple examples motivate the need to factor in the training data costs when building a classifier. It is interesting to note, however, that except for the work on active learning, as described in more detail in Section 6, very little research has addressed this issue. Specifically, past research does not address the "cost of cases" at all. This is true even though learning curves, which describe the relationship between training set size and classifier performance, are well known and frequently studied—they are just rarely used in practice.

3. DESCRIPTION OF EXPERIMENTS

The experiments in this paper vary the training set size and then track the accuracy of the induced classifier. This performance information is then combined with cost information in order to determine the overall utility of the classifier. We outline our experimental methodology in Section 3.1, summarize the data sets we analyze in Section 3.2 and discuss how we measure total utility in Section 3.3.

3.1 Experimental Methodology

All of the experiments in this paper use C4.5 [8], a popular decision tree learner that is a descendant of ID3. In order to determine the relationship between training set size and predictive accuracy, training sets are generated with a variety of sizes. The data is partitioned via random sampling as follows. For each experiment, 25% of the data is randomly selected and allocated to the test set, while the remaining 75% of the data is potentially available for training. However, because we want to vary the training set size, for most experiments only a portion of this 75% will be assigned to the training set (the remainder is not used). All results in this paper are based on averages over 20 runs, in order to increase the statistical significance of our results. Because it does not take much CPU time to build the models for any of the data sets analyzed in this paper, the use of multiple runs does not seriously limit our work. In future work we may investigate the use of single runs if the size of the training sets warrants it.

We investigate two simple sampling schedules. Sampling schedule S1 uses the following training set sizes: 10, 50, 100, 500, 1000, 2000, …, 9000, 10000, 12000, 14000, 16000, etc. Specifically, after the first five training set sizes the training set size is incremented by 1,000 until the training set size reaches 10,000, after which the training set size is incremented by 2,000. Our second sampling schedule, S2, starts with a training set size of 50 and then successively doubles the training set size. This geometric sampling scheme is motivated by previous work on progressive sampling, which shows that given certain assumptions (which do not hold in this paper), this schedule is asymptotically optimal [7]. This previous work on progressive sampling is described in Section 6. For sampling schedules S1 and S2, in addition to evaluating the training set sizes just described, the largest possible training set size (i.e., 75% of the data set size) is also evaluated. For some of the plots in our results section, Section 4, some additional training set sizes were evaluated in order to improve the granularity of our results. These additional training set sizes were not used in Section 5, where we discuss a progressive sampling strategy.

3.2 Data Sets

We analyze the ten data sets described in Table 1. In order to improve the presentation of our results, the data sets are partitioned into two groupings based on their relative sizes: large and small. Table 1 also lists the total number of examples in each data set. The data sets were obtained from the UCI Machine Learning Repository [6], except for those marked with an asterisk, which were originally made available from researchers at AT&T (these data sets are available from the author).

Large Data Sets		Small Data Sets	
adult	21,281	kr-vs-kp	3,196
coding*	20,000	move*	3,029
blackjack*	15,000	german	1,000
boa1*	11,000	breast-wisc	699
network1*	3,577	crx	690

Table 1: Description of Data Sets

3.3 Measuring Total Utility

We evaluate the performance of the induced classifiers based on total utility because we need to take the cost of training data into consideration. Thus, our utility metric must take into account the cost of training data (data cost) and the cost of classification errors (error cost). We do not include other possible costs, such as the CPU time cost associated with training the classifier, although we do record these CPU times and comment on their potential impact on total utility. In future work we plan to consider these and other costs in the utility metric. Total cost is defined below in equation 1.

$$\text{Total Cost} = \text{Data Cost} + \text{Error Cost} \quad [1]$$

Before we can understand the error cost term, some background is required. For our experiments, a classifier is built from training data and its accuracy is evaluated using separate test data. The purpose of any classifier is to classify new, unlabeled, examples—not the test set. We thus assume the existence of a score data set, S, which the user will apply the classifier to, over some period of time. The error cost will be based on the number of errors we expect to get when classifying the score data set, which can be estimated as the error rate on the test set multiplied by the size of S, denoted |S|. Thus error cost is directly proportional to |S|. Although we do not know the value of |S| for any of the data sets in this paper, a domain expert should be able to estimate its value, although this may not always be a simple task (e.g., it may depend on how long the classifier is used, how successful it is, etc.).

For each experiment we know the number of training examples, n, and the estimated error rate, e, based on the performance of the classifier on the test set. We assume that there is some fixed cost *Ctr* for acquiring each training example and some fixed cost *Cerr* for each error made on an example in the score data set. *Data cost* will then equal $n \cdot Ctr$ and *error cost* will be estimated as $e \cdot |S| \cdot Cerr$. The total cost for a classifier then is given by equation 2, which is our measure of total utility.

$$\text{Total Cost} = n \cdot Ctr + e \cdot |S| \cdot Cerr \quad [2]$$

With specific domain knowledge we would be able to estimate *Ctr, Cerr,* and $|S|$ and thus calculate total cost. However, in our case we do not know these values. Therefore we need to treat them as variables and analyze a wide range of values in order to properly analyze a data set. The problem with this situation is that three variables make a thorough analysis difficult. However, we can reduce this to two variables by arbitrarily assuming |S| is 100. This does not reduce the generality of our results because we can easily account for other values of |S| via a simple calculation. Namely, error cost is proportional to the product $|S| \cdot Cerr$ so that if we find that |S| is 100,000 instead of 100, we can simply look at the experiment results for *Cerr*/1,000 rather than *Cerr*. In a sense, we are measuring error cost in terms of every 100 score examples and then adjusting for different score set sizes.

Given that we now only need to track *Ctr* and *Cerr*, for analysis purposes we can simplify things further by only tracking the ratio of these two variables. While the real total cost will depend on the actual constants, the optimal training set size, for example, will only depend on the ratio of the costs. So, in our results we simply report the cost ratio, *Ctr:Cerr*, where *Ctr* is typically 1, the unit cost, and $Cerr \geq 1$. Because we want to plot our results using numerical values, we often use the relative cost or relative cost ratio instead, which is simply *Cerr*/*Ctr*. For example, if the cost ratio is 1:100 then the relative cost ratio is 100. Note that in this case we can say that from a utility perspective it is an even trade-off to purchase 100 training examples if it will reduce the number of errors by 1, assuming |S| is 100. We can remove the condition on |S| by stating things in a slightly different manner: purchasing 100 training examples leads to an even trade-off if it results in a 1% reduction in error rate.

As an example, we can compute the total cost associated with one of the experiments reported in this paper, which uses the adult data set. For this particular experiment, n is 1500 and the error rate of the resulting classifier is 15.8%. The cost ratio is 1:1000 and, as discussed, for now we presume the score set will have 100 examples. Using equation 2, the total cost is then:

$$\text{Total cost} = 1500 \cdot 1 + .158 \cdot 100 \cdot 1000 = 1500 + 15800 = 17{,}300$$

One potential issue with the utility measure in equation 2 is that if |S| is sufficiently large then the second term will dominate the first, in which case the cost of acquiring the training data is not important. Will the error cost term always dominate the data cost

term? We do not believe so for several reasons. First, for some domains the cost of acquiring training data is very significant and once the learning curve begins to flatten out it may take tens or hundreds of thousands of training examples to improve accuracy by even a tenth of a percent. In this region, even if |S| is very large, the first term may still play a significant role. It is within that region that we expect our cost model to be most useful. In addition, |S| need not always be extremely large. For example, in the poker example mentioned in Section 2.2 one will not typically play a large number of poker hands against a single opponent. Finally, other work in the field seems to support our intuition that data cost is important. For example, the entire field of active learning is based on the assumption that error cost will *not* totally dominate the data cost—if it did then active learning would be unnecessary.

We conclude this section on measuring total utility by noting that total cost is not the only metric we could have used to measure utility. We could alternatively have factored in a benefit for each correctly classified example and a cost for each incorrectly classified example. However, given the goals of this paper we do not believe that there is much value in also evaluating this performance metric, although we recommend that practitioners use this alternative metric if it makes sense for a specific domain.

4. RESULTS

This section includes our main results. Section 4.1 describes how we use the cost ratio information to analyze the learning curve data generated by our experiments. Section 4.2 presents detailed results for a representative data set and then Section 4.3 provides summary results for the remaining data sets.

4.1 Analyzing the Impact of the Cost Ratio

The basic experiments in this paper involve generating the learning curves for each data set. In order to analyze these results, we need to vary the cost ratios and then see how this impacts the total utility of the classifier. In particular, we want to determine the optimal training set size for any cost ratio and we would like to see how this optimal training set size changes as the cost ratio is varied.

In our analysis we examine a wide range of cost ratios. We cannot focus on the most realistic values since those values are domain specific and we do not have the requisite domain knowledge. Rather, we try to examine a sufficient range of cost ratios so that we hit the "two extremes" and sample some points in between. Specifically, for each data set we strive to analyze cost ratios such that for one of the cost ratios the optimal strategy is to acquire almost no training data ($\leq$ 10 examples) and for another cost ratio the optimal strategy is to acquire all possible training data.

The cost ratio that leads us to acquire all of the training examples may be quite high, such as 1:50,000 (the cost ratio required by the adult data set). This cost ratio, which says that the cost of an error is 50,000 times that of the cost of a training example, may appear to be unrealistic, but that is not necessarily so. For example, if we have a direct marketing campaign where it costs $1 to produce and mail a catalog and the demographic information that is purchased to help build a predictive model is $100 per 10,000 households, one can see that a training example is very cheap relative to the cost of an error ($1 for each household that was mistakenly predicted to make a purchase). Also recall that the errors are per 100 score examples. The cost ratio of 1:50,000 with 100 score examples is equivalent, as described in Section 3.3, to a cost ratio of 1:50 if there are 100,000 examples in the score set. Stated more generally, a cost ratio of 1:50,000 means that purchasing 50,000 training examples leads to an even trade-off if the error rate is reduced by 1%. This certainly seems like it could be a reasonable trade-off.

4.2 Detailed Analysis of the Adult Data Set

Our analysis of the adult data set begins with its learning curve, shown in Figure 1. As is common for learning curves, there is a very steep increase in accuracy at first which then diminishes as more training data becomes available. It is worth noting that the learning curve for this data set never reaches an asymptote, even when there are more than 15,000 training examples. What is particularly interesting is that the learning curve shows a small but steady increase in accuracy for an extended period of time—as the training set size grows from 4,500 to 15,960 training examples the accuracy increases from 85.0% to 85.9%. Also note that the learning curve is not smooth like an idealized learning curve and in a few cases shows a decrease in accuracy as the training set size increases (we expect these statistical aberrations would disappear given an infinite number of runs with random sampling).

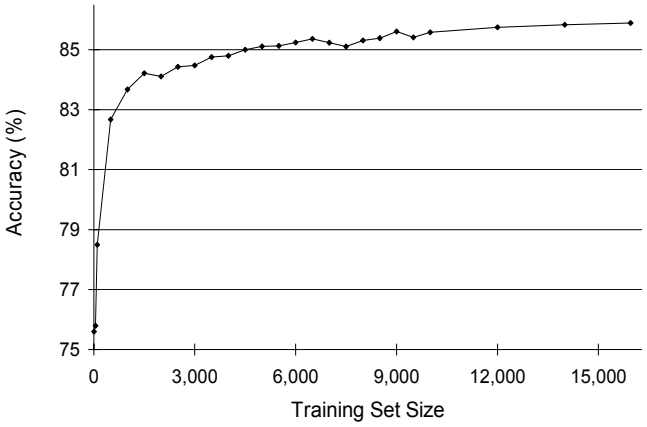


Figure 1: Learning Curve for Adult Data Set

Given the learning curve data it is straightforward to compute the total cost for a variety of cost ratios, by using equation 2 (recall that |S| is fixed at 100). In figure 2 total cost is plotted versus training set size for six different cost ratios (*Ctr:Cerr*).

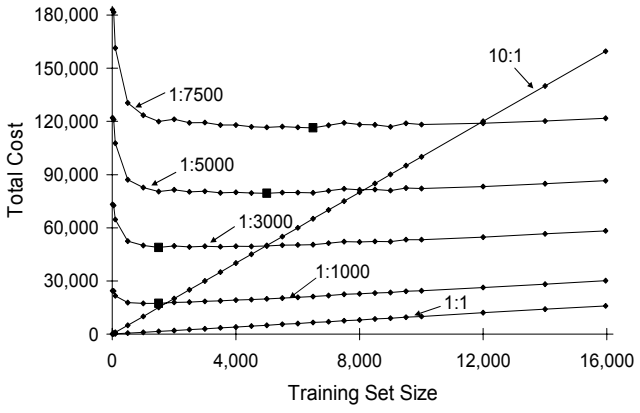


Figure 2: Utility Curves for Adult Data Set

If we look at the 10:1 cost ratio, which places the highest relative cost on the training data (and is the only case where *Ctr*>*Cerr*), we see that the curve is linear. The reason for this is that in this situation *data cost* completely dominates *error cost*, so that *total cost* essentially equals *data cost* and hence is directly proportional to the size of the training set. When the ratio shifts to 1:1, *data cost* still dominates, but the slope is less because the total cost is now much less (approximately one-tenth the cost for 10:1). As the cost ratio continues to shift toward a higher relative cost for errors, the curve becomes non-linear and the minimum total cost (identified for each curve by the large square marker) no longer is at the minimum training set size, but rather shifts towards the larger and larger training set sizes. At a relative cost of 1:7500, the lowest cost is achieved with a training set size of 6,500.

One problem with Figure 2 is that the total cost rises as the cost ratio becomes more skewed, which obscures some of the changes of the curves with lower total cost. To fix this problem we normalize each curve by dividing the total cost by the maximum total cost associated with the curve. The results for normalized cost are shown in Figure 3. This method for representing the results also permits us to examine higher cost ratios and shows us that for a cost ratio of 1:50,000 the optimum strategy is to use all of the training data.

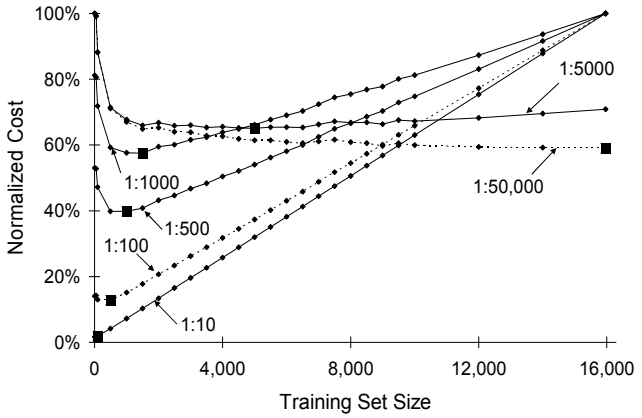


Figure 3: Normalized Utility Curves for Adult Data Set

Figure 3, in conjunction with Figure 1, shows that once the learning curve begins to flatten out, a great increase in the cost ratio is required for it to be worthwhile to use more training data. This is encouraging in that once we get past a certain point, the optimal training set size is not overly sensitive to the exact value of the cost ratio; hence a good *estimate* of this ratio should be adequate. Figure 3 also makes it clear that using all of the potentially available training data is not a good strategy—for most relative cost ratios the total (normalized) cost is much lower for the optimal training set size than when the maximum number of training examples are used.

Figure 4 provides the most highly summarized information concerning the adult data set. It shows, for each relative cost (*Cerr*/*Ctr*), the optimum training set size. The optimum training set size curve can be used by a practitioner to determine the amount of training data to obtain even if the precise cost ratio is not known (in Section 5 we introduce a progressive sampling strategy to find this optimum without first requiring all of the potentially available training data). At a minimum, curves like those in Figure 4 can inform a data mining practitioner of the trade-offs involved. We provide the associated accuracies beneath some of the data points, to help correlate these results with the learning curve results in Figure 1.

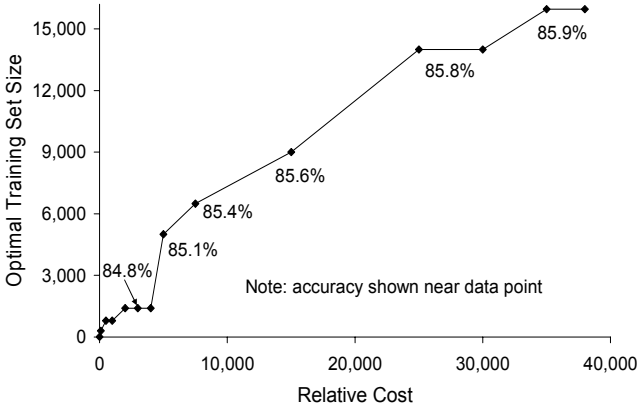


Figure 4: Optimal Training Set Sizes for Adult Data Set

The shape of the optimal training set size curve in Figure 4 deserves some discussion. This curve is not smooth and this is not because we only calculate the optimal training set size for certain relative costs. The curve is not smooth because the learning curve in Figure 1 is not smooth. You may also note that there is a sharp increase in slope when the relative cost increases from 4,000 to 5,000 (between these two values the optimal training set size jumps from 1,400 to 5,000 examples). This is due to the fact that the learning curve in Figure 1 temporarily shows a small *decrease* in accuracy when the training set size increases beyond 1,400 and hence the cost ratio must increase significantly to overcome the "burden" of purchasing training examples which do not increase the accuracy of the classifier.

4.3 Summary Results for all Data Sets

In this section we present summary results for all of the data sets. Figures 5 and 6 show the learning curves for the large and small data sets, respectively. Note that for many of the data sets a plateau is not reached using the available training data. For some, like coding, the performance is still improving relatively rapidly, while for others, like adult, it is improving only slowly.

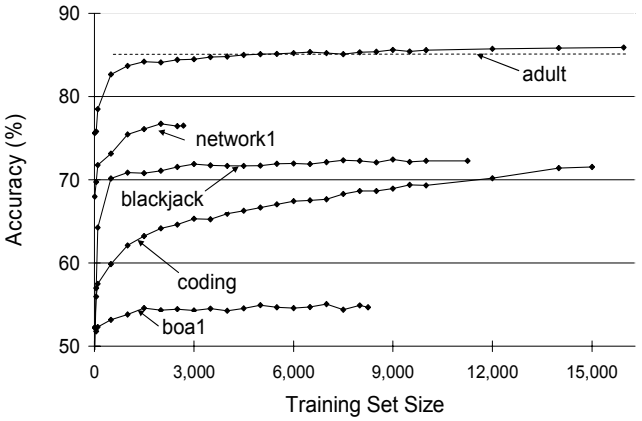


Figure 5: Learning Curves for Large Data Sets

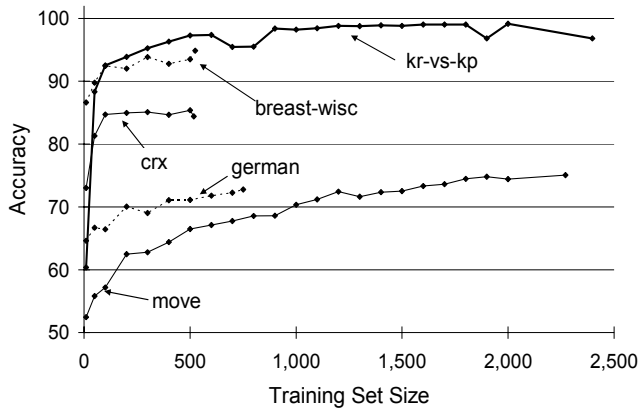


Figure 6: Learning Curves for Small Data Sets

Figures 7 and 8 show the optimal training set sizes for the large and small data sets, respectively (the curve for adult is not provided again). Note that once the relative cost is sufficiently high, all of the potentially available training data will be used and then the optimal training set size curve will flatten out completely.

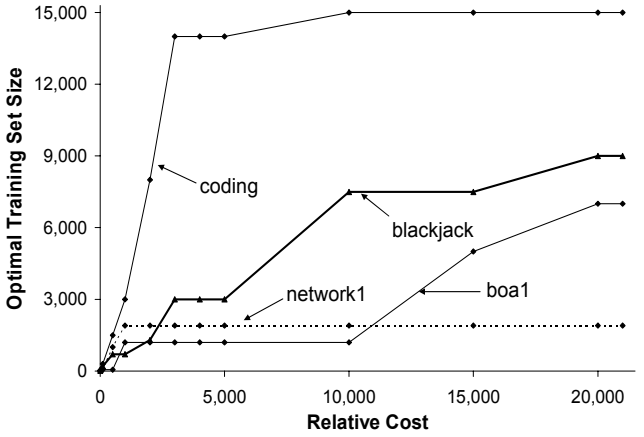


Figure 7: Optimal Training Set Sizes for Large Data Sets

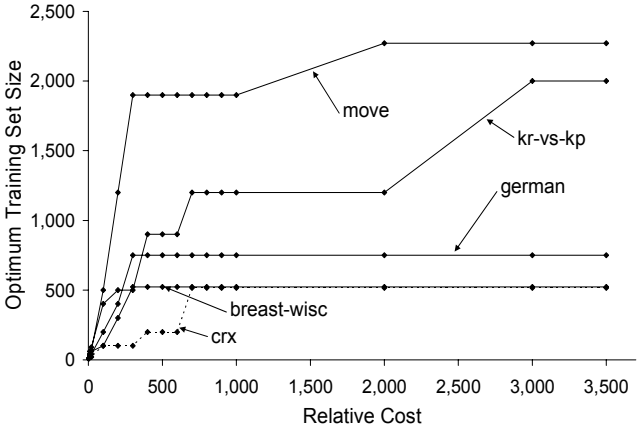


Figure 8: Optimal Training Set Sizes for Small Data Sets

5. Progressive Sampling Strategies

The results in Section 4 demonstrate that one can improve overall classifier utility by properly trading-off the cost of acquiring additional training data with its benefits (i.e., fewer errors). However, to be of practical use, we need a strategy that identifies a good (near-optimal) training set size g without acquiring/using more than g training examples. That is, we must "pay" for training examples "up front", so once we acquire them we will always use them. The strategy used in Section 4 of trying a variety of training set sizes up until the maximum number available makes no sense in this context. What we need is a progressive sampling strategy to identify g while purchasing only g examples.

5.1 Overview of Progressive Sampling

The general outline of a progressive sampling strategy is simple. You begin with some initial amount of training data and then, iteratively, build a classifier, evaluate its performance and acquire additional training data. There are two key decisions faced by such a progressive sampling algorithm: 1) when to terminate the loop and stop acquiring training data, and 2) how many training examples to acquire at each iteration (i.e., the batch size).

In our progressive sampling experiments we use a simple stopping strategy: we stop obtaining more training data after the first observed increase in total cost. Note that this guarantees that we will not achieve the optimal cost because, at minimum, there is one better training set size (i.e., the one observed *before* the increase). That is, once we have acquired additional training data we have incurred the cost associated with purchasing it, and must include this cost we analyzing the performance of the progressive sampling strategy. If the accuracy of the learning curve is non-decreasing, then this stopping condition will lead to a training set size that is very close to optimal. Our results show that the actual learning curves are not always non-decreasing, but this does not usually have a big impact on the results.

The choice of how much additional training data to acquire at each iteration is decided by a sampling schedule. In this paper we only evaluate very simple, non-adaptive sampling schedules, although more sophisticated ones are described later as possible future work. We utilize the sampling strategies S1 and S2 that were described in Section 3.1 as our progressive sampling strategies. With a few exceptions, S1 samples every 1000 examples whereas S2 uses a geometric sampling scheme that starts with 50 examples and then repeatedly doubles the training set size.

5.2 Progressive Sampling Results

This section presents the results of the two progressive sampling strategies, S1 and S2. As described earlier, each strategy terminates after the first observed increase in total cost. Because we want to see how well these strategies perform, we compare them to the "optimal strategy" that always selects the optimal training set size and cost based on S1 (which samples more frequently than S2). We also compare these progressive sampling strategies to our "straw man" strategy, which simply uses all of the available training data.[1] This second comparison quantifies the benefit of considering the training data cost when building a classifier, since without such knowledge a reasonable strategy would be to use all potentially available training data.

[1] There may not always be a maximum "amount of available training data" for a data set, but in many cases there will be (e.g., the number of records describing businesses is limited by the number of businesses). In this paper we assume that the only data available is in the original data set and the maximum amount available for training is equal to 75% of this amount.

5.2.1 Detailed Results for the Adult Data Set

We begin by presenting detailed results for the adult data set. Table 2 presents the results for the progressive sampling strategies S1, S2 and the "optimal" version of S1, Optimal-S1. We report the results for a variety of relative cost ratios. For each cost ratio and strategy, we report the selected training set size, the total cost and the CPU time, in seconds, associated with all of the experiments used to identify that training set size. For example, for a relative cost ratio of 10,000 a training set size of 9,000 yields the optimal cost, which is 152,900. The total CPU time required is 9.15 seconds, which is the time to build all of the classifiers using the sampling schedule, up until the training set size of 10,000.

Relative	Optimal-S1			S1			S2		
Cost Ratio	Size	Cost	CPU	Size	Cost	CPU	Size	Cost	CPU
1	10	34	0.00	50	74	0.00	100	122	0.00
10	10	25	0.00	50	292	0.00	100	319	0.00
20	500	2,233	0.20	50	2,470	0.00	100	538	0.00
200	500	3,966	0.20	1,000	4,266	0.53	800	4,060	0.40
500	500	9,165	0.20	2,000	9,945	1.23	1,600	9,480	0.92
5,000	5,000	79,450	4.17	6,000	79,800	5.27	12,800	83,700	14.84
10,000	9,000	152,900	9.15	7,000	154,700	6.48	12,800	154,600	14.84
15,000	9,000	224,850	9.15	7,000	228,550	6.48	15,960	226,860	20.88
20,000	9,000	296,800	9.15	7,000	302,400	6.48	15,960	297,160	20.88
50,000	15,960	721,460	20.89	7,000	745,500	6.48	15,960	718,960	20.88

Table 2: Progressive Sampling Strategy Comparison for Adult

Table 2 shows us that S1 and S2 are quite effective strategies, since they come relatively close to achieving the optimal cost. The S2 strategy seems to outperform S1, although if more training data were available we would expect S1 to do better—since each strategy stops one iteration after the lowest cost that extra step would be more costly for S2, which geometrically increases the training set size. Our total cost metric does not factor in CPU time, but the results for the adult data set indicate that this is probably okay, since the CPU times are all quite small. However, this might not be true for much larger data sets. We discuss extensions to the total utility metric to factor in the cost of computation in Section 7.

Figure 9 compares the performance of the S1 and Straw Man strategies for cost ratios below 1:10,000. Note that the x-axis is not scaled in this case, in order to make the results easier to read. We see that the straw man strategy of just using all of the training data independent of the relative cost ratio leads to very poor results until a relative cost ratio of about 1:10,000 is reached. This motivates the need and benefit of factoring in the training data cost when building a classifier.

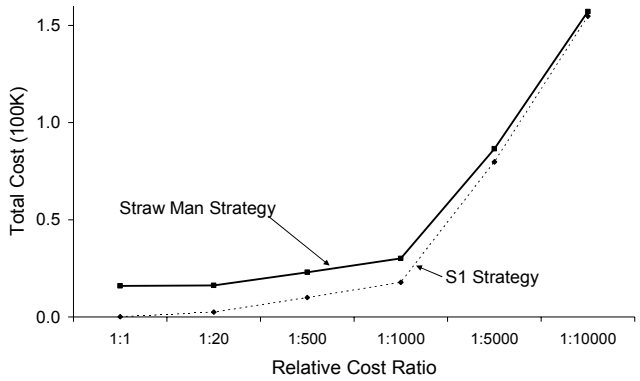


Figure 9: Comparison of S1 and Straw Man Strategies for Adult

5.2.2 Summary Results for the Large Data Sets

A comparison of the performance of the S1 strategy with the S1-optimal strategy, for the five large data sets, is provided in Table 3. The results show that the S1 progressive sampling strategy is quite effective, except for very low relative costs. In these situations the training data is relatively expensive and the stopping criteria, which requires that the sampling strategy go past the optimal training set size, is heavily penalized. An adaptive sampling strategy that reduces the batch size as increases in training set size lead to smaller improvements in total utility would likely reduce the impact of this problem. The results for the S2 strategy are not included in this section due to space considerations and because it is not that different than the S1 sampling strategy given the amount of available training data (it would be very different if the data sets were much larger).

Relative	Increase In Total Cost: S1 vs. S1-optimal				
Cost Ratio	Adult	Blackjack	Boa1	Coding	Network1
1	115.7%	53.2%	70.1%	62.8%	91.0%
20	10.6%	34.6%	5.1%	2.0%	0.7%
500	8.5%	1.0%	1.2%	2.1%	2.7%
1,000	3.2%	2.6%	2.3%	0.6%	3.6%
5,000	0.4%	1.4%	4.7%	0.2%	1.5%
10,000	1.2%	1.1%	5.9%	0.0%	1.3%
15,000	1.6%	1.6%	6.3%	0.0%	1.2%
20,000	1.9%	1.9%	6.5%	0.0%	1.1%
50,000	3.3%	0.7%	6.9%	0.0%	1.0%

Table 3: Optimal vs. S1 for Large Training Sets

Table 4 compares the straw man strategy with the S1 progressive sampling strategy for the five large data sets and shows that, consistent with the results presented in Figure 9 for the adult data set, the straw man strategy performs very poorly when the relative cost ratio is below a certain threshold. After a point they perform similarly and the straw man strategy is even sometimes superior, because the S1 strategy sometimes stops prematurely, due to a temporary decrease in accuracy in the learning curves. If more training data were available, such that the learning curves reached a plateau with only a fraction of this training data, we would then expect the straw man strategy to perform poorly even for these higher cost ratios.

Relative	Increase In Total Cost: Straw Man vs. S1				
Cost Ratio	Adult	Blackjack	Boa1	Coding	Network1
1	21428.5%	12024.0%	8345.6%	15880.9%	3270.9%
20	557.6%	976.5%	802.6%	1539.2%	381.0%
500	131.4%	61.3%	27.9%	43.7%	5.8%
1,000	68.1%	26.1%	11.0%	13.4%	0.0%
5,000	8.4%	2.9%	-2.6%	0%	0%
10,000	1.5%	0.4%	-4.3%	0%	0%
15,000	-0.4%	-0.4%	-4.9%	0%	0%
20,000	-1.4%	-0.8%	-5.2%	0%	0%
50,000	-3.2%	0.1%	-5.7%	0%	0%

Table 4: Straw Man vs. S1 for Large Training Sets

We have not shown the CPU times required by the progressive sampling strategies, except for the adult data set. However, these times are under a minute in every case and hence the cost of computation does not appear to be a significant consideration for these data sets.

6. RELATED WORK

Previous research, to the best of our knowledge, does not directly address the cost of cases that we address in this paper. However, there is a substantial amount of research that studies related costs and issues. We describe these research efforts in this section and comment on how they relate to our work.

Work on progressive sampling has focused on efficiently finding the training set size where the learning curve reaches a plateau [7]. The motivation for that work is to reduce the cost of computation—the training data cost is not taken into account. If we ignore the cost of computation, the past work on progressive sampling is a special case of our work, where the cost of training data is arbitrarily small, but non-zero (if it were zero one would just use all of the available training data even if a plateau is reached). This previous work showed that a geometric sampling scheme, similar to our S2 strategy, is asymptotically optimal, with respect to the cost of computation (not with respect to training data cost). As described shortly, we plan to generalize our work to include the cost of computation, at which point our work will subsume this previous work on progressive sampling.

Weiss and Provost [11] factored in the cost of cases, but only in a limited way. The assumption in that work was that the cost of cases limited the amount of training data and this amount is already specified. The only decision in that work was what class distribution should be used for training in order to maximize classifier performance. That work also used a progressive sampling strategy, although the goal in that case was to identify the optimal class distribution.

In this paper we assume that one has no control over which examples are added to the training data. That is, if there is a cost associated with labeling an example, we cannot selectively choose which examples to label and if there is a cost associated with measuring feature values, we cannot determine which examples to add based on these feature acquisition costs. Thus, we do not consider active learning methods which have been used in other work [1, 4, 10, 13]. Nonetheless, one does not always have the freedom to use active learning methods (e.g., in the scenario where one is purchasing data from an external source). Also, much of the work on active learning assumes a fixed budget [4], in which case the decision is just which examples are best to label or features to measure, and there is no need to determine when to stop acquiring more training data. The closest match to our research from the active learning community is research where the marginal utility of each example is estimated and this is used to determine how many examples to label [5].

One of the contributions of our research is that it shows how the optimal training set size varies based on the relative cost of training examples versus errors. These optimal training set size curves may be useful even if the specific cost ratios are not known. The cost curves of Drummond and Holte [2] are quite analogous to our optimal curves, except that their curves show the optimal performance based on the ratio of the cost of a false positive to a false negative classification error, rather than the cost of a training example versus the cost of an error**.** Both their curves and ours can aid a practitioner who must make decisions about how to generate the best classifier.

7. LIMITATIONS AND FUTURE WORK

The work described in this paper has several limitations and can be extended in many ways. In this section we describe some of the limitations and possible future extensions. We expect to address many of these issues in the near future.

One of the limitations of our work concerns the size of the data sets. Ideally we would have sufficient training data for all of our data sets so that the learning curves would always reach a plateau. If that were the case then additional data would not be of any benefit and then we could completely analyze the behavior of the data set with respect to training set size. Unfortunately, for many of our data sets a plateau is not reached. It would therefore be valuable to analyze much larger data sets, especially those that are complex enough to require a great deal of training data in order for the learning curve to reach a plateau.

Our utility metric considers the cost of data but not the cost of computation (i.e., CPU time). We intend to include the cost of computation in future analyses. However, since both progressive sampling strategies required less than one minute of CPU time when applied to each of the ten data sets, it is important that we first obtain much larger and more complex data sets. In addition, we intend to analyze more sophisticated sampling schedules, including adaptive schedules, where the amount of training data requested in each "batch" varies based on the expected change in total cost (which could be extrapolated based on the changes in total cost for the previous batches). These more sophisticated schemes would be more likely to find the true "optimal" training set size, by reducing the batch size as the marginal utility of adding training data approaches zero. Note that this behavior is the opposite of what happens when the cost of computation is the main cost; the past work on progressive sampling increases the batch size over time since it uses a geometric sampling scheme[7].

8. CONCLUSION

This paper analyzed the impact of training data cost on total classifier utility, where total utility considers the cost of the training data as well as the performance of the classifier on classifying new examples. We introduced a variety of charts to help visualize the relationship between training data cost, the cost of errors and total utility. We also identified the optimal training set size for different data sets and different cost ratios and showed that overall utility can be substantially improved by not using all of the training data that is potentially available. Two simple progressive sampling strategies were also introduced and were shown to be relatively effective in finding the optimal training set size and optimal total utility. Furthermore, one of these progressive sampling strategies was shown to outperform the "straw man" strategy of using all potentially available training data. The research described in this paper fills in a "hole" in the area of Utility-Based Data Mining by considering the cost of training cases in the data mining process.

9. REFERENCES

[1] Cohn, D., Atlas, L., and Ladner, R. Improving Generalization with Active Learning. Machine Learning 15(2): 201-221, May 1994.

[2] Drummond, C. and Holte, R. Explicitly Representing Expected Cost: An Alternative to ROC Representation. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 198-207.

[3] Hoehn, B., Southey, F., Holte, R., and Bulitko, V. Effective Short-Term Opponent Exploitation in Simplified Poker. Proceedings of the Twentieth National Conference on Artificial Intelligence (Pittsburgh, Pennsylvania), AAAI Press, Menlo Park, CA, 783-788, 2005.

[4] Kapoor, A., and Greiner, R. Learning and Classifying under Hard Budgets. Proceedings of the 16th European Conference on Machine Learning (Porto, Portugal), Springer, 170-181, 2005.

[5] Melville, P., Saar-Tsechansky, M., Provost, F. and Mooney, R., Economical Active-feature Value Acquisition through Expected Utility Estimation. Proceedings of the First International Workshop on Utility-Based Data Mining, 10-16, ACM Press, 2005.

[6] Newman, D.J., Hettich, S., Blake, C.L. and Merz, C.J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science. 1998.

[7] Provost, F., Jensen, D., and Oates, T. Efficient Progressive Sampling. Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 23-32, ACM Press 1999.

[8] Quinlan, J. R., C4.5: Programs for Machine Learning. San Mateo, CA. Morgan Kaufmann. 1993.

[9] Turney, P. Types of Cost in Inductive Concept Learning. Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning, 2000.

[10] Veeramachaneni, S., and Avesani, P. Active Sampling for Feature Selection. Proceedings of the Third IEEE International Conference on Data Mining (Melbourne, Florida), IEEE Computer Society, 2003.

[11] Weiss, G. M. and Provost, F. Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. Journal of Artificial Intelligence Research, 19 (2003), 315-354.

[12] Weiss, G. M., Saar-Tsechansky, M. and Zadrozny, B., Proceedings of the First International Workshop on Utility-Based Data Mining (editors), ACM Press, Chicago, IL, 2005.

[13] Zheng, Z., and Padmanabhan, B. On Active Learning for Data Acquisition. Proceedings of the 2002 IEEE International Conference on Data Mining (Maebashi City, Japan, December 9-12, 2002). IEEE Computer Society, 2002, Washington, DC, 562-569.

Prediction Games in Infinitely Rich Worlds

[Extended Abstract]

Omid Madani
madani@yahoo-inc.com
Yahoo! Research
3333 Empire Ave
Burbank, CA 91504

ABSTRACT

In order to familiarize oneself with a rich new world, one needs to learn a great deal. This massive learning includes learning to recognize myriad categories and learning to make use of them for learning other categories. The process of playing *prediction games* may make this massive learning possible. In this paper, we describe prediction games and present a discussion of properties of algorithms and systems that would play them well. The great potential of prediction games is in pointing to ways of achieving powerful large-scale learning without the need for human supervision.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning - Induction

General Terms

Algorithms

1. INTRODUCTION

We introduce the learning process of *playing prediction games* in an *infinitely rich world*. Prediction games may be a step toward making powerful massive unsupervised learning possible. The learned outcome could find tremendous use.

The games are played by a *system* that takes its sequence of inputs from the world and makes learning episodes out of it. Our running example will be *prediction in text*, *i.e.*, playing prediction games in the world of natural language, as available in the online text. One such game is *fill-in-the-blank*, played as follows: the system repeatedly inputs a sentence and hides a portion of it such as a word or phrase. Several components of the system then become active and make predictions about what is missing using the available context. The activated components may then be updated according to the answer, which is available. The system moves on to the next learning opportunity.

UBDM'06 August 20, 2006, Philadelphia, Pennsylvania, USA

The important point is that the amount of training instances is viewed as unbounded. All the learning subprocesses work to improve prediction. The utility is in terms of the operationality of the prediction system: coverage, depth, accuracy, and speed. In the case of prediction in text, the learned system may lead to significant practical improvements in statistical language modeling or complement current techniques [16]. Byproducts of such processes, as we explain, include learning associations and discovering new concepts. We next describe infinitely rich worlds and prediction games in more detail. We motivate a *systems* approach for playing these games and discuss desiderata on solution techniques and some of the challenges we see, such as scalability and prevalence of noise. We briefly describe the work we have begun in this direction. Besides prediction in language, we mention possible games in other worlds. We give the wider context and motivation for prediction games, and situate prediction games with respect to existing learning frameworks.

2. PREDICTION GAMES

Prediction games are proposed to make massive learning possible. By *massive learning*, we mean learning in the order of millions of categories and beyond, and estimating values for variables in the order of billions, and beyond. Prediction games consist of an infinitely rich world and a system that plays the games in that world.

We will use prediction in text as our main example, and in particular the fill-in-the-blank game, as explained in the introduction: every sentence or passage in the online text can serve as a source of one or more learning episodes. In each game the system hides a portion of an input sentence, say a word or a phrase, and sees how well it can predict it, using the context derived from the rest of the sentence and possibly the broader context such as the passage, the page, and so on. For example, in the sentence, "I rode my bike to school", the word "bike" could be covered, and then the question posed by the system to itself is what can replace X in "I rode my X to school". The answer may be in the form of a single phrase, or several candidate phrases ranked (*e.g.*, "bike, vehicle, motor bike, car, horse, table,..") or assigned probabilities. If we constrain the context to the words and sentences occurring before X, we obtain the game of *predict-the-next-word*, which is also the typical problem addressed by statistical language modeling [16, 7].

2.1 The Fundamental Role of Concepts

In describing the world and the system we make use of

the work on categories or concepts[1] in cognitive psychology [14]. An infinitely rich world has a few properties. First, an infinitely rich world is a source of unbounded experience. Ample learning experience is a prereqisite for achieving massive learning. Second, the world is also infinitely rich in that it enjoys infinitely many regularities. These regularities are in the form of the many categories that entities in the world can be grouped into. For us, a category is a very general notion and in our every day world includes groupings of physical things, events, actions, feelings, and so on. Example categories include: "red", "happy", [2] "office chair", "my father", "southern lakes", and so on. In prediction in text, categories are sets of words. From the view point of the system, we may think of concepts (hardwired or learned categories) as an implicit set of vector values (feature values or stimuli) for which membership in the set can be determined by some components of the system at adequate accuracy and efficiency levels.

There is an infinity of ways in which to carve up a rich world into various categories. Learned new categories for the most part should serve a purpose for the system. Here, the main purpose is improving overall prediction ability. Categories serve as great abstractors. Without them, one could not learn from the past, every experience would be brand new, and one would not be able to predict. Categories are fundamental in dealing with a rich world. If they are effectively discovered and recognized, the problems of sparsity in natural language and handling invariance in vision are adequately addressed. Categories are necessary for intelligence.

The regularities in the world are also in the form of the many types of relations that the categories tend to enjoy with one another, such as temporal or spatial co-occurrence, part-of, friends, or generalization and specialization relations (*e.g.*, the category "animal" is a generalization of the category "dog"). Relations help in predicting categories and learning new ones.

Each learning episode contains a few categories that comprise the context (the features) to be used for prediction, and a category to be predicted. Many categories, the raw categories, are hard-wired: their detection in a learning episode is achieved by some component of the system that was part of the system from the outset. Many categories will be constructed (discovered) by the system. A new category can be a grouping of several existing categories, or may be a refinement of an existing category. For example, a category corresponding to the to-be family of verbs may be discovered at some point: it is recognized whenever any of the to-be verbs ("is", "are", "was",..) in their various forms are present. The category "bank" can be refined into the financial-institution sense, the physical building sense (that houses the financial institution), and the side-of-river sense.

Improving prediction performance is the main driver for learning new categories. Other factors include the difficulty of discovering the category and recognizing it. A category has to have some use for the system to spend resources for detecting it. New categories, once learned, are treated the same way as raw categories: they serve as features to define context and they are treated as targets to be predicted, though their detection may take longer time than raw categories.

In text prediction, each possible phrase is treated as a raw category. Detection of these categories may be regarded as hard-wired (NOT learned). Simple features include those corresponding to presence of phrases, and those corresponding to relative position of the phrase with respect to the phrase to be predicted (*e.g.*, a feature can correspond to the word "drink" appearing in two positions before the word to be predicted). More complex features that may be only evaluated part of the time include part of speed tagging and parsing output. Thus, the number of features (learning dimensionality) and classes can be in the millions and beyond. After some learning, the features extracted from the sentence "He flies frequently to X" includes raw features such as the word "flies" appears 3 positions before, as well as more abstract features such as one corresponding to the composite category "traveling-action" appearing before target with confidence 0.6. The system's predictions for X may include the names of a few locations, such as cities or travel destinations, as well as categories or groupings that correspond to travel destinations.

2.2 An Abstraction

We may view prediction games as a table or matrix of instances and target values, $< x_i, y_i >$, just as is common in supervised learning. However the table has infinitely many rows, and the dimensionality and the set of categories to be predicted are unbounded as well. Each row corresponds to an instance or an episode and the last column, the target column to be predicted, corresponds to a category.

In text prediction, each phrase serves as a category, so new phrases will be seen as the online text is processed. More importantly, new categories will be constructed out of old ones and these categories will be equal citizens. Newly constructed categories address the sparsity problem. Efficient discovery of new useful categories is a very important problem. Features are some functions of the categories, such as whether the category is present in the context, or was recently seen, or the location of the category, or conjunctive and relational features involving multiple categories. Therefore, the dimensionality of the problem in practice starts out very high, and furthermore it is growing, as more categories are constructed out of the old ones. The table is highly sparse: only a finite number of features will have a nonzero value in each row. In prediction in text, we do not see a need for negative feature values. Some functionalities may best be achieved if the feature values reflect a measure of confidence or probability.

2.3 Other Games

Other worlds include the world-wide-web, our own every day visual/physical world (*e.g.*, the problem of vision), and the social world and predicting people's actions.

The world-wide-web may be viewed as a superset of the text online. Web pages include text, but are part of a larger structure of a website, and contain additional content or structure that could be utilized, such as tables and pictures, html layout, and links to other pages.

Real-time prediction in a changing visual scene provides

[1] In cognitive psychology, a distinction is made between a category in the external world and the concept of it, as represented internally in the mind [14]. This is an important distinction, but in this paper to simplify the presentation, we don't make a distinction.

[2] The system playing the game is viewed here as part of the world. Therefore, parts of the system may learn to predict the current or subsequent state of other parts.

ample learning episodes. The changes can be due to moving objects, camera movement, and changes in attention. One game can be to predict what will be seen next, based on what has been seen recently. What is learned include temporal or spatial co-occurrence of categories (physical objects) as well as recognizing a whole based on seeing some parts and inferring the remaining parts.

The task of predicting people's actions and reactions, *i.e.*, obtaining a sense of what a person or groups of people may do in various circumstances, for example in social interactions, or in cooperative team work or in competitions, may turn out to be the most challenging of prediction games.

2.4 A Systems Approach

The diversity of categories and relations that could hold between them, and the evolving nature of the world, makes statistical modeling challenging and likely inadequate. Similarly, a single algorithm cannot do the job either. Search in the space of *systems* may be the best strategy for finding effective solutions. A prediction system does not try to model the world, in the sense of building replicas of it internally, nor estimate parameter values for a model within some constrained family of statistical models. It only strives to predict some of the world's aspects well. A system, for example the operating system of a computer, consists of a number of interacting components and is driven by a number of different algorithms responsible for tasks such as paging, process scheduling, and I\O. Similar to an operating system, issues of operationality or maintaining some level of service also holds for a prediction system. The difference from an operating system is that the prediction system learns and adapts and self-organizes extensively, and grows in its functionality over time. Learning is central to its operationality.

2.5 Applications

A powerful predicting system and the byproducts of what is learned should find a number of applications. For the sake of the the system itself, as the system improves at predicting, it does not have to spend as much time and resources into verifying the presence of certain categories such as certain important events. This can have life saving consequences. Based on sounds alone, one may assess the danger level, and quickly finish crossing the street when a fast approaching car is heard. In other situations, partial context is the only information the system has practical access to (*e.g.*, due to occlusion or poor lighting conditions). However, in much of the day to day activity, since the system can predict and classify robustly based on partial context, the system is in general faster and more accurate and is freed to spend its resources to focus on other learning activity.

A prediction system for text may lead to significant practical improvements over current language modeling technology or complement it in ways. Another application is answer to questions of the type "what does the phrase X mean?": find contexts that X appears in by querying the web, hide the phrase, and see what categories (single phrases or sets of phrases) are predicted. Those categories or phrases likely have the same type as X.

3. HOW?

Prediction games involve predicting a large number of categories, and while there is noise, the system has access to an unbounded sequence of learning episodes. The large-scale aspects of prediction games motivate the following desiderata on candidate solutions (systems and algorithms):

- Online algorithm that are time and memory efficient
- Handle large number of features: sample size efficiency
- Handle large numbers of categories (prediction as well as discovery of new categories)
- Robust to imperfections, uncertainty, and variety

Several online linear learning algorithms satisfy some of the above desiderata. The issue of very large numbers of classes appears not to have been addressed adequately before. A basic immediate question is how to efficiently learn and recognize myriad categories? It appears that this problem has found good solutions in nature [17, 10].

A related problem is what may be referred to as the *recall problem*: how to quickly narrow down the possible candidate true positive categories when the system is presented with an instance? An effective solution to the recall problem may play an important part in addressing the challenge of large number of classes. We recently proposed the framework of a *recall system* for this purpose [13]. The ideal recall system, given an instance (a feature vector), quickly outputs a small set of candidate categories (say tens), and does not miss the true categories of the instance. We realized this functionality by an *index* that maps (connects) features to categories. An efficient algorithm for learning the index was given, although superior algorithms may yet to be discovered. The recall system, while high on recall and drastically reducing the number of candidate categories, may still be poor on precision[3]. Online learning algorithms could be used to efficiently learn classifiers for each category. The classifiers corresponding to the retrieved concepts can be applied to the instance to precisely determine the right categories. Alternatively, or in conjunction, learning weighted mappings can make possible adequate ranking of the recalled categories (*i.e.*, without the need for classifier training and application). The recall system may also facilitate other learning activities, such as concept discovery and learning concepts in terms of other learned concepts.

Next, we discuss constraints and desiderata that we see important for any would be candidate architecture and algorithm in effectively playing prediction games.

3.1 Scalability: Information vs. Computation

Scalability is paramount. The system is to operate in an information rich environment, and what is to be learned, an operational predicting system, requires by definition an information hungry process. In some scenarios, for example learning to predict by processing the online text, there is much data available, and the data is relatively static. It is the algorithms' speed that determines how fast it can learn from this abundance. In real-time scenarios such as in vision, the system is bombarded with information, and may

[3]Here, precision and recall are from the point of view of an instance, and not the category. In machine learning literature, recall and precision are often used from the point of view of the category (*e.g.*, recall is the portion of true positive instances that were classified positive for a single category. Our use is similar to information retrieval, were documents are categories here.

have to ignore (drop) much incoming and possibly valuable information due to its processing limits.

There is a trade-off here: on one hand the system can spend much time on the current instance, or it can revisit past instances, perhaps optimize some measure of accuracy over them, or, on the other hand, to spend the computational resources for further exploring the world and acquiring possibly new useful information. A natural question is what determines a good balance and how to achieve it. The ultimate objective, whether the emphasis is competence more on breadth of prediction versus depth, may play a role in the answer. Obtaining insights into the nature of this tradeoff would be very valuable. Recent work in large-scale text mining has raised some of these issues [15].

3.1.1 Learn in an Online Fashion

Batch learning or optimization can provide significant improved accuracy in typical learning problems, when compared to online methods. However, batch techniques are designed inherently with finite data in mind, and prediction games are about unbounded data. Bottou and Le Cun point to both theoretical and empirical advantages of properly designed online algorithms over batch, in terms of accuracy achieved, when training data is abundant [4]. Furthermore, as features are induced (new concepts are discovered), the whole learning activity takes on an evolving nature. While one can imagine subsampling and instance selection (*e.g.*, via some measure of instance utility) to keep things sufficiently small, and using incremental or staged batch learning and optimization, this solution appears complex, and may not be the best utilization of learning time (see also Section 3.3 on code complexity).

A quick calculation demonstrates the considerable potential advantages of light linear time online learning. Consider algorithms A, B and C taking respectively n, $n \log_{10} n$, and n^2 steps to learn from n instances. Then in the time that algorithm B takes to process a million instances, algorithm A can learn from six millions instances, while algorithm C has only processed a 1000 instances. And of course, the advantages of linear time processing grows with increasing n. One has to ask whether the accuracy benefits of a more time consuming algorithm is worth the opportunity loss in learning from more data.

Even within linear time processing settings, one should keep the tradeoffs in mind: imagine a system B taking 10 times longer per instance than a simpler system A as system B uses more sophisticated feature extractors or many more newly discovered concepts or deploys sophisticated inference processes. The lighter system A may still exhibit superiority since it can learn from ten times as many training instances in the same period. More sophistication, for example in improved feature extraction or inference, is definitely needed at some point to extend the reach of the prediction system, but the question is how to determine a good balance. For best results, a system may have to learn to deploy increasingly sophisticated processing only as needed. The field of perceptual leaning in cognitive science has investigated learning of new features for improved recognition as well as learning to speed up task accomplishment [12].

In general, the constraint of a finite memory will also be a great constraining factor, informing the design of systems and algorithms. The significant advantage of online algorithms here is that one need not store the instances: instances (learning episodes) are processed and translated into a feature vector, used for learning, and then discarded (onto the next instance). Most of the memory will be in terms of the categories stored and the connections among them. Thus considerations need to be made regarding for example the number of (nonzero) weights used in the system, and the memory consumption during learning those weights. Research on online computations and mining data streams share some of the objectives and challenges, and insights and algorithmic and analysis techniques developed there may be applicable [8, 3].

Therefore, our view is that the primary solution methods will be online. In order to achieve effective learning from a rich world, a number of online algorithms for various subtasks will have to be devised. These subtasks include:

- online feature selection and reduction
- feature or concept discovery and clustering
- speeding up of feature extraction and classification.

Speed ups in overall concept detection are important so that the system can move on to effectively learning more complex concepts.

3.2 Robustness to Imperfections, Uncertainty, Variety

There are numerous sources of imperfections and uncertainty or noise. In prediction in text, features include phrases or categories discovered, and their relations. But phrases are ambiguous and there can be misspellings, or missing values, or imperfect segmenters, or inaccurate passages. The newly discovered categories are also imperfectly recognized (poor precision or recall). In vision (and other perceptual domains), sensors are noisy and objects may occur in different lighting conditions, or may be partially occluded or be viewed from different angles and distances. There may be considerable variety within the same object class. Therefore, uncertainty and variability may be even more problematic. Prediction games are played for developing robustness in a variety of conditions, to detect/predict categories in many of their guises. Therefore effective solution algorithms will yield a system that is robust in numerous ways. The prediction system should learn from the sum of all its experience, rather than putting too much weight on any single instance. A learning strategy may include ignoring those instances for which the target category to be predicted, the observed outcome, is very uncertain. A similar strategy goes in using feature values. Igoring difficult episodes, at least temporarily, may be an effective partial strategy. Of course, this assumes basically that the system knows (at least for the most part) when it doesn't know: it has access to accurate confidence values. This is also related to a problem we may refer to as the *grounding problem*. The first time a system starts out, how could it be sure of any thing? We assume that the system is endowed with many sensors or feature extractors that work adequately, and they are sufficient to start the games.

3.3 Program Simplicity

In designing such learning systems, low code complexity and uniformity of architecture is an important guideline not to deviate from. Perhaps most of the program complexity and diversity may be concentrated into the preprocessing

and raw feature extraction components. The system will be learning a variety of things to improve overall prediction ability (in breadth, depth, and speed), but we hope that the number of distinct central algorithms utilized will be relatively few. Much of the functionality and diversity in the system should be the result of learning and experience, as opposed to explicit programming. "Keep it simple, as much as possible" should inform the design and search for algorithms.

3.4 Other Aspects

There are a number of other issues that can lead to fruitful research. For example, for the most part, we have assumed that the system will be passive in obtaining its learning episodes. However, increasing coverage (number of categories learned) motivates an active exploration of the world, at least after some period of purely passive observation. In the scenario of learning from the online text on the web, passive learning corresponds to crawling the web randomly, while active crawling can correspond to activities such as looking for specific pages and passages that contain certain phrases, perhaps using a search engine.

Another general area is defining the problems and discovering the algorithms for learning the control mechanisms necessarily for managing the complexity of the system. How are decisions regarding whether to deploy more costly feature extractors or inference algorithms made? How are these decisions streamlined over time? In general, what are the organizational principles and the organizing processes? Much is unclear regarding even the nature of the problems, but we think processes such as prediction games are promising candidates for raising these issues and may inspire useful problem formulations. Prediction games also provide the ample learning experience that appears necessary to achieve effective operational systems.

4. DISCUSSION

Prediction games liberate us from the labeling bottleneck, *i.e.*, explicit human supervision. The world serves as the teacher. Considerations of early learning, in infants and babies, is specially valuable. This stage provides the crucial foundation for learning and development in later years. Understanding how this foundation is developed from a computational point of view, *i.e.*, the nature of the major algorithms and organizing principles at play, is very important. Considering how infants and babies develop, in animals as well as in humans, one may conclude that:

1. There is massive learning taking place during first months and years, and
2. It does not involve (explicit) supervision.

We assume the above two statements are true. This massive learning includes recognizing myriad categories in various conditions (*e.g.*, physical objects, such as faces), learning the dynamics of the physical world, and becoming adept at physical movement. The infant, in the first few months of its birth, may have indeed mined its world very effectively!

There is much that remains unclear regarding the nature of this learning stage, *i.e.*, what problems are being solved and what kind of algorithms are at play. Research on processes similar to prediction games is a promising avenue for pointing to ways on how massive learning of different kinds may take place in the young brain. The outcome of this process may ultimately be a system that has developed a feel or a sense of a world that was once very unfamiliar.

The brain likely implements a variety of algorithms, but it has also been referred to as a prediction machine. Hawkins claims that making predictions is central to all of intelligence [11]. For example, he states that the nature of understanding or knowing may be explained by the ability to predict. He describes at a high level how predictions may be taking place by the brain's circuitry and how the ability may be aquired and developed [11].

Valiant proposes and explores a network or graph model of the neocortex with locally programmable elements [18]. He stresses the importance of paying attention to resource constraints, such as limited connectivity and processing speed, and shows how a number of learning algorithms could be implemented on his model. The field of bounded rationality [9], in studying human behavior, has also emphasized the roles of uncertainty, the issues of resource constraints, and the variety of implicit frequently competing objectives shaping observed human behavior. In playing prediction games, a system has to contend with significant resource constraints and uncertainty. We also expect that achieving some kind of optimization of the overall prediction objective is not the best place to put one's research efforts on. A more useful goal is in understanding what determines satisfactory operationality, *i.e.*, prediction capability. As we have mentioned, prediction ability involves satisfying the desiderata of speed/efficiency as well as accuracy in breadth and depth (how precise the predictions are). Research has found that considerations of category and feature utility can explain certain observed phenomena regarding speed of categorization and category naming in humans [6].

4.1 Relation to Other Learning Frameworks

Prediction games involve unsupervised learning, and in particular the outputs of the components may be interpreted as confidence values or probabilities for random variables (categories). In this sense, they are akin to density estimation or distribution learning. The use of graphical models has made density estimation tasks more efficient, due to the focusing on constraints on the type of relations that the variables may actually have (*e.g.*, the actual dependencies in case of Bayes networks). Graphical models may turn out to be too constraining for learning the myriad categories and the variety of relations between them, which apriori may not be anticipated or programmed. On the other hand, graphical models allow powerful types of inference. The benefits of sophisticated inference versus its computational costs is a subject under the theme of computation versus information: the tradeoff between extensive computation on the current situation, or learning instances accumulated so far, versus foraging for further information via new learning episodes.

Prediction games involve both supervised tasks and techniques, in the sense of learning connections between predictor features and existing categories, as well as unsupervised tasks and techniques, in the sense of forming new categories out of existing ones. The difference with most traditional clustering would be that the grouping or new category building should serve the objective of improving predictions. Another big difference, as raised in Section 3.1.1, is the requirement of scalability. To be most effective, online clustering algorithms need to be devised. Most existing clustering al-

gorithms are too costly and were designed with finite data in mind. Also, the successful clustering techniques may not be based on similarity between instances, but instead may rely on techniques such as observing co-occurrence between existing categories (inside the system, or as observed in the external world). The discovered categories may not necessarily be constrained to form a structure such as a tree or a dag. The major condition is that their benefits (improving prediction) outweigh the costs (memory requirements, time to recognize).

Supervised learning has enjoyed much success, but the issue of obtaining training data has always been a significant bottleneck. This has motivated much research on topics such as query learning, active learning, and semisupervised learning, in order to reduce the need for human/teacher involvement [1, 5]. Researches have even developed social games to motivate humans to label [2]. Playing prediction games involve learning myriad categories and their many relationships. The aim is to learn in the order of billions of variable values, and beyond. We think that the amount of training information required to make sophisticated systems makes classic explicit supervision infeasible.[4]

Prediction games complement reinforcement learning. Learning from prediction games can occur at a larger scale than reinforcement learning, as reinforcement learning requires taking action and attaining (possibly delayed) rewards. In the physical world, actions take time and energy. Prediction games involve mostly observations and information processing. However, from time to time, they also involve information seeking actions such as moving the camera. Acquiring prediction ability serves the goal of familiarization to ones world, and directly or indirectly is geared towards helping the intelligent agent predict and obtain rewards, and in general avoid bad situations and obtain pleasant ones (for example, for a baby, sensing that mom is about to feed her).

For predicting the next word, statistical language modeling uses techniques based on n-gram models [16]. For each possible length-limited history seen so far (sequence of words), these techniques keep counts for each of the different words that occur afterwards. They later use the counts to compute probabilities. The approach proposed here shifts the focus from history to the word to be predicted. It treats each word or phrase as a class to be learned. Statistics on candidate predicting features are kept for each target word rather than the history. In this respect, it is akin to the work of Even-Zohar and Roth [7], who showed that the online classification (discriminative) approach has flexibility in utilizing diverse and sophisticated features, but focused on discriminating between a relatively small number of classes.

Learning tasks such as time series prediction and learning to solve puzzles and to play games such as chess also share similarities, but differ mainly on the aspect of richness of the world and objectives.

5. CONCLUSION: LETS PLAY THE GAME!

In this paper we have been concerned with an abstraction of tasks and processes that would lead to massive learning, and proposed prediction games for that purpose. In the process, we raised a number of issues and potential research directions. At a broad level, a major message of this paper is stressing the importance of thinking about practical massive ongoing learning that can be achieved without supervision.

Prediction games are about having much to learn and plenty to learn from. Domains such as the online text, the web, and vision provide the richness that would enable playing prediction games in infinitely rich worlds.

[4]Note that we are making a distinction between supervised techniques, which will be utilized in playing prediction games, and explict supervision or training signal via a human teacher.

Acknowledgments

Many thanks to Michael Connor, Yann Le Cun, Dennis DeCoste, Mark Gluck, Wiley Greiner, Rosie Jones, Gregory Murphy, and Russ Poldrack for valuable discussions or pointers.

6. REFERENCES

[1] D. Angluin. Comp. learning theory: survey and selected bibl. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 351–369, 1992.

[2] L. V. Anh and L. Dabbish. Labeling images with a computer game. In *ACM CHI 2004*, 2004.

[3] A. Borodin and R. El Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] L. Bottou and Y. L. Cun. Large scale online learning. In *NIPS*, 2003.

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

[6] J. E. Corter and M. A. Gluck. Explaining basic categories: Feature predictability and information. *Psychological Bulletin*, 111(2):291–303, 1992.

[7] Y. Even-Zohar and D. Roth. A classification approach to word prediction. In *Annual meeting of the North Amercian Association of Computational Linguistics (NAACL)*, 2000.

[8] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *SIMOD Record*, 34(2), June 2005.

[9] G. Gigerenzer and R. Selten, editors. *Bounded Rationality*. The MIT Press, 2002.

[10] K. Grill-Spector and N. Kanwisher. Visual recognition, as soon as you know it is there, you know what it is. *Psychological Science*, 16(2):152–160, 2005.

[11] J. Hawkins. *On Intelligence*. Owl Books, 2004.

[12] P. J. Kellman. *Handbook of Experimental Psychology*, chapter 7: Perceptual Learning. NY, Wiley, 2002.

[13] O. Madani and W. Greiner. Learning when concepts abound. Technical report, Yahoo! Research, 2006.

[14] G. L. Murphy. *The Big Book of Concepts*. MIT Press, 2002.

[15] D. Ravichandran, P. Pantel, and E. Hovy. The terascale challenge. In *KDD Workshop on Mining for and from the Semantic Web*, 2004.

[16] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *IEEE*, 88(8), 2000.

[17] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[18] L. G. Valiant. *Circuits of the Mind*. New York: Oxford University Press, 1994.

Efficient Mining of Temporal High Utility Itemsets from Data streams

Vincent S. Tseng
Dept. Computer Science and Information Engineering
National Cheng Kung University, Taiwan, ROC
tsengsm@mail.ncku.edu.tw

Chun-Jung Chu
Dept. of Computer Science
National Chiao Tung University, Taiwan, ROC
cjchu@cis.nctu.edu.tw

Tyne Liang
Dept. of Computer Science
National Chiao Tung University, Taiwan, ROC
tliang@cis.nctu.edu.tw

ABSTRACT
Utility itemsets are considered as the different values of individual items as utilities, and utility mining aims at identifying the itemsets with high utilities. The temporal high utility itemsets are the itemsets with support larger than a pre-specified threshold in current time window of data stream. Discovery of temporal high utility itemsets is an important process for mining interesting patterns like association rules from data streams. In this paper, we propose a novel method, namely *THUI* (Temporal High Utility Itemsets) *-Mine*, for mining temporal high utility itemsets from data streams efficiently and effectively. To our best knowledge, this is the first work on mining temporal high utility itemsets from data streams. The novel contribution of *THUI-Mine* is that it can effectively identify the temporal high utility itemsets by generating fewer temporal high transaction-weighted utilization 2-itemsets such that the execution time can be reduced substantially in mining all high utility itemsets in data streams. In this way, the process of discovering all temporal high utility itemsets under all time windows of data streams can be achieved effectively with limited memory space, less candidate itemsets and CPU I/O time. This meets the critical requirements on time and space efficiency for mining data streams. The experimental results show that *THUI-Mine* can discover the temporal high utility itemsets with higher performance and less candidate itemsets compared to other algorithms under various experimental conditions.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications — data mining.

General Terms
Algorithms, Design

Keywords
utility mining, temporal high utility itemsets, data streams, association rules

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION
The mining of association rules for finding the relationship between data items in large databases is a well studied technique in data mining field with representative methods like *Apriori* [1, 2]. The problem of mining association rules can be decomposed into two steps. The first step involves finding all frequent itemsets (or say large itemsets) in databases. Once the frequent itemsets are found, generating association rules is straightforward and can be accomplished in linear time.

An important research issue extended from the association rules mining is the discovery of temporal association patterns in data streams due to the wide applications on various domains. Temporal data mining can be defined as the activity of looking for interesting correlations or patterns in large sets of temporal data accumulated for other purposes [6]. For a database with a specified transaction window size, we may use the algorithm like Apriori to obtain frequent itemsets from the database. For time-variant data streams, there is a strong demand to develop an efficient and effective method to mine various temporal patterns [11]. However, most methods designed for the traditional databases cannot be directly applied for mining temporal patterns in data streams because of the high complexity.

In many applications, we would like to mine temporal association patterns in data streams for amount of most recent data. That is, in the temporal data mining, one has to not only include new data (i.e., data in the new hour) into, but also remove the old data (i.e., data in the most obsolete hour) from the mining process. Without loss of generality, consider a typical market-

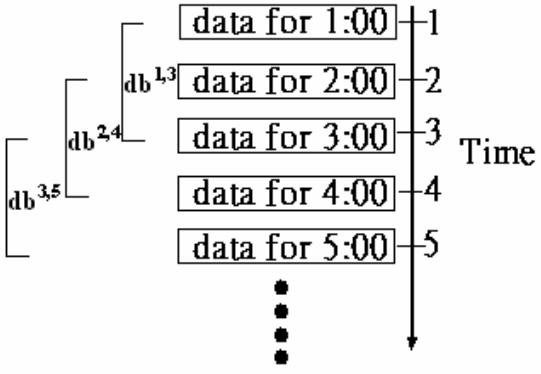


Figure 1. An example of online transaction flows.

basket application as illustrated in Figure 1 has been considered. The transaction flow in such an application is shown in Figure 1 where transaction data purchased by customers as time advances.

In Figure 1, for example, data was accumulated with time passing by. Old data in the past hours becomes useless for reference. People might be most interested in the temporal association patterns in the latest three hours (i.e., $db^{3,5}$) as shown in Figure 1. It can be seen that in such a data stream environment it is intrinsically difficult to conduct the frequent pattern identification due to the constraints of limited time and space. Furthermore, it takes considerable time to find temporal frequent itemsets in different time windows. However, the frequency of an itemset may not be a sufficient indicator of interestingness, because it only reflects the number of transactions in the database that contain the itemset. It does not reveal the *utility* of an itemset, which can be measured in terms of cost, profit, or other expressions of user preference. On the other hand, frequent itemsets may only contribute a small portion of the overall profit, whereas non-frequent itemsets may contribute a large portion of the profit. In reality, a retail business may be interested in identifying its most valuable customers (customers who contribute a major fraction of the profits to the company). Hence, frequency is not sufficient to answer questions, such as whether an itemset is highly profitable, or whether an itemset has a strong impact. Utility mining is thus useful in a wide range of practical applications and was recently studied in [7, 14, 19]. This also motivates our research in developing a new scheme for finding *temporal high utility itemsets (THUI)* from data streams.

Table 1. A transaction database and its utility table.

(a) Transaction table

			TID \ ITEM	A	B	C	D	E	
$db^{1,3}$	$\triangle^-$	P_1	T_1	0	0	26	0	1	
			T_2	0	6	0	1	1	
			T_3	12	0	0	1	0	
	D^-	P_2	T_4	0	1	0	7	0	
			T_5	0	0	12	0	2	
			T_6	1	4	0	0	1	
		P_3	T_7	0	10	0	0	1	
			T_8	1	0	1	3	1	$db^{2,4}$
			T_9	1	1	27	0	0	
	$\triangle^+$	P_4	T_{10}	0	6	2	0	0	
			T_{11}	0	3	0	2	0	
			T_{12}	0	2	1	0	0	

(b) The utility table

ITEM	PROFIT($)(per unit)
A	3
B	10
C	1
D	6
E	5

Recently, a *utility mining* model was defined in [19]. Utility is a measure of how "useful" (i. e. "profitable") an itemset is. The definition of utility of an itemset X, $u(X)$, is the sum of the utilities of X in all the transactions containing X. The goal of utility mining is to identify high utility itemsets which drive a large portion of the total utility. Traditional association rules mining model assumes that the utility of each item is always 1 and the sales quantity is either 0 or 1, thus it is only a special case of utility mining, where the utility or the sales quantity of each item could be any number. If $u(X)$ is greater than a utility threshold, X is a high utility itemset. Otherwise, it is a low utility itemset. Table 1 is an example of utility mining in a transaction database. The number in each transaction in Table 1(a) is the sales volume of each item, and the utility of each item is listed in Table 1(b). For example, $u(\{B, D\}) = (6\times10+1\times6) + (1\times10+7\times6) + (3\times10+2\times6) = 160$. {B, D} is a high utility itemset if the utility threshold is set at 120.

However, a high utility itemset may consist of some low utility items. Another attempt is to adopt the level-wise searching schema that exists in fast algorithms, such as Apriori [3]. However, this algorithm doesn't apply to the *utility mining* model. For example, $u(D) = 84 < 120$, D is a low utility item, but its superset {B, D} is a high utility itemset. If using Apriori to find high utility itemset, all the combinations of all the items must be generated. Moreover, to discover a long pattern, the number of candidates is exorbitantly large. The cost of either computation time or memory is intolerable, regardless of what implementation is applied. The challenge of utility mining is not only in restricting the size of the candidate set but simplifying the computation for calculating the utility. Another challenge of utility mining is how to find temporal high utility itemsets from data streams as time advances.

In this paper, we explore the issue of efficiently mining high utility itemsets in temporal databases like data streams. We propose an algorithm named *THUI-Mine* that can discover temporal high utility itemsets from data streams efficiently and effectively. The underlying idea of *THUI-Mine* algorithm is to integrate the advantages of Two-Phase algorithm [14] and SWF algorithm [12] and augment with the incremental mining techniques for mining temporal high utility itemsets efficiently. The novel contribution of *THUI-Mine* is that it can efficiently identify the utility itemsets in data streams so that the execution time for mining high utility itemsets can be substantially reduced. That is, *THUI-Mine* can discover the temporal high utility itemsets in current time window and also discover the temporal high utility itemsets in the next time window with limited memory space and less computation time by sliding window filter method. In this way, the process of discovering all temporal high utility itemsets under all time windows of data streams can be

achieved effectively under limited memory space, less candidate itemsets and CPU I/O. This meets the critical requirements of time and space efficiency for mining data streams. Through experimental evaluation, *THUI-Mine* is shown to produce fewer candidate itemsets in finding the temporal high utility itemsets, so it outperforms other methods in terms of execution efficiency. Moreover, it also achieves high scalability in dealing with large databases. To our best knowledge, this is the first work on mining temporal high utility itemsets from data streams.

The rest of this paper is organized as follows: Section 2 overviews the related work. Section 3 describes the proposed approach, *THUI-Mine*, for finding the temporal high utility itemsets. In section 4, we describe the experimental results for evaluating the proposed method. The conclusion of the paper is provided in Section 5.

2. RELATED WORK

In association rules mining, Apriori [3], DHP [15], and partition-based ones [13, 16] were proposed to find frequent itemsets. Many important applications have called for the need of incremental mining. This is due to the increasing use of the record-based databases whose data are being continuously added. Many algorithms like FUP [8], FUP_2 [9] and UWEP [4, 5] are proposed to solve incremental database for finding frequent itemsets. The FUP algorithm updates the association rules in a database when new transactions are added to the database. Algorithm FUP is based on the framework of Apriori and is designed to discover the new frequent itemsets iteratively. The idea is to store the counts of all the frequent itemsets found in a previous mining operation. Using these stored counts and examining the newly added transactions, the overall count of these candidate itemsets are then obtained by scanning the original database. An extension to the work in [8] was reported in [9] where the authors propose an algorithm FUP_2 for updating the existing association rules when transactions are added to and deleted from the database. UWEP (Update With Early Pruning) is an efficient incremental algorithm, that counts the original database at most once, and the increment exactly once. In addition the number of candidates generated and counted is minimum.

In recent years, processing data from data streams is a very popular topic in data mining. Many algorithms like FTP-DS [17] and RAM-DS [18] are proposed to process data in data streams. FTP-DS is a regression-based algorithm to mine frequent temporal patterns for data streams. A wavelet-based algorithm, called algorithm RAM-DS, to perform pattern mining tasks for data streams by exploring both temporal and support count granularities.

Some algorithms like SWF [12] and Moment [10] were proposed to find frequent itemsets over a stream sliding window. By partitioning a transaction database into several partitions, algorithm SWF employs a filtering threshold in each partition to deal with the candidate itemset generation. Moment algorithm use the closed enumeration tree (CET), to maintain a dynamically selected set of itemsets over a sliding window.

A formal definition of utility mining and theoretical model was proposed in [19], namely MEU, where the utility is defined as the combination of utility information in each transaction and additional resources. Since this model cannot rely on downward closure property of Apriori to restrict the number of itemsets to be examined, a heuristics is used to predict whether an itemset should be added to the candidate set. However, the prediction usually overestimates, especially at the beginning stages, where the number of candidates approaches the number of all the combinations of items. The examination of all the combinations is impractical, either in computation cost or in memory space cost, whenever the number of items is large or the utility threshold is low. Although this algorithm is not efficient or scalable, it is by far the best to solve this specific problem.

Another algorithm named Two-Phase was proposed in [14], which is based on the definition in [19] and achieves for finding high utility itemsets. It presented a Two-Phase algorithm to prune down the number of candidates and can obtain the complete set of high utility itemsets. In the first phase, a model that applies the "transaction-weighted downward closure property" on the search space to expedite the identification of candidates. In the second phase, one extra database scan is performed to identify the high utility itemsets. However, this algorithm must rescan the whole database when added new transactions from data streams. It need more times on processing I/O and CPU cost for finding high utility itemsets. Hence, Two-Phase algorithm is just only focused on traditional databases and is not suited for data streams.

Although there existed numerous studies on high utility itemsets mining and data stream analysis as described above, there is no algorithm proposed for finding temporal high utility itemsets in data streams. This motivates our exploration on the issue of efficiently mining high utility itemsets in temporal databases like data streams in this research.

3. PROPOSED METHOD: THUI-MINE

The goal of utility mining is to discover all the itemsets whose utility values are beyond a user specified threshold in a transaction database. Utility mining is to find all the high utility itemsets in [19]. An itemset X is a *high utility itemset* if $u(X) \geq \varepsilon$, where $X \subseteq I$ and ε is the minimum utility threshold, otherwise, it is a *low utility itemset*. For example, in Table 1, $u(A, T_8) = 1\times3 = 3$, $u(\{A, C\}, T_8) = u(A, T_8) + u(C, T_8) = 1\times3 + 1\times1 = 4$, and $u(\{A, C\}) = u(\{A, C\}, T_8) + u(\{A, C\}, T_9) = 4 + 30 = 34$. If $\varepsilon = 120$, {A, C} is a low utility itemset. However, if an item is a low utility item, its superset may be a high utility itemset. For example, $u(D) = 84 < 120$, D is a low utility item, but its superset {B, D} is a high utility itemset because of $u(\{B, D\}) = 160 > 120$. Hence, all the combinations of all items should be processed so that it never loses any high utility itemset. But the cost of either computation time or memory is intolerable.

Liu *et al* [14] proposed Two-Phase algorithm for pruning candidate itemsets and simplify the calculation of utility. First, Phase I overestimates some low utility itemsets, but it never underestimates any itemsets. For the example in Table 1, the *transaction utility of transaction Tq*, denoted as $tu(Tq)$, is the sum of the utilities of all items in *Tq* and the *transaction-weighted utilization of an itemset X*, denoted as $twu(X)$, is the sum of the transaction utilities of all the transactions containing X, then $twu(A) = tu(T_3) + tu(T_6) + tu(T_8) + tu(T_9) = 42 + 48 + 27 + 40 = 157$ and $twu(\{D, E\}) = tu(T_2) + tu(T_8) = 71 + 27 = 98$. In fact, $u(A) = u(\{A\}, T_3) + u(\{A\}, T_6) + u(\{A\}, T_8) + u(\{A\}, T_9) = 36 + 3 + 3 + 3 = 45$ and $u(\{D, E\}) = u(\{D, E\}, T_2) + u(\{D, E\}, T_8) = 11 + 23 = 34$. So Phase I overestimates some low utility itemsets, it never

underestimates any itemsets. Table 2 gives the transaction utility for each transaction in Table 1. Second, one extra database scan is performed to filter the overestimated itemsets in phase II. For example, $twu(A) = 157 > 120$ but $u(A) = 45 < 120$. Then item {A} is pruned. Otherwise, it is a high utility itemset. Finally, all of high utility itemsets are discovered by this way.

Table 2. Transaction utility of the transaction database.

TID	Transaction Utility	TID	Transaction Utility
T_1	31	T_7	105
T_2	71	T_8	27
T_3	42	T_9	40
T_4	52	T_{10}	62
T_5	22	T_{11}	42
T_6	48	T_{12}	21

Our algorithm *THUI-Mine* is based on the principle of Two-Phase algorithm [14], and we extend it with the sliding-window-filtering technique and focus on utilizing incremental methods to improve the response time with fewer candidate itemsets and CPU I/O. In essence, by partitioning a transaction database into several partitions from data streams, algorithm *THUI-Mine* employs a filtering threshold in each partition to deal with the transaction-weighted utilization itemsets generation. The cumulative information in the prior phases is selectively carried over toward the generation of transaction-weighted utilization itemsets in the subsequent phases by *THUI-Mine*. In the processing of a partition, a progressive transaction-weighted utilization set of itemsets is generated by *THUI-Mine*. Explicitly, a progressive transaction-weighted utilization set of itemsets is composed of the following two types of transaction-weighted utilization itemsets, i.e., (1) the transaction-weighted utilization itemsets that were carried over from the previous progressive candidate set in the previous phase and remain as transaction-weighted utilization itemsets after the current partition is taken into consideration and (2) the transaction-weighted utilization itemsets that were not in the progressive candidate set in the previous phase but are newly selected after only taking the current data partition into account As such, after the processing of a phase, algorithm *THUI-Mine* outputs a cumulative filter, denoted by CF, which consists of a progressive transaction-weighted utilization set of itemsets, their occurrence counts and the corresponding partial support required. With these design considerations, algorithm *THUI-Mine* is shown to have very good performance for mining temporal high utility itemsets from data streams. In Section 3.1, we give an example for mining temporal high utility itemsets from data stream. The proposed algorithm, *THUI-Mine*, is described in details in Section 3.2.

3.1 An Example for Mining Temporal High Utility Itemsets

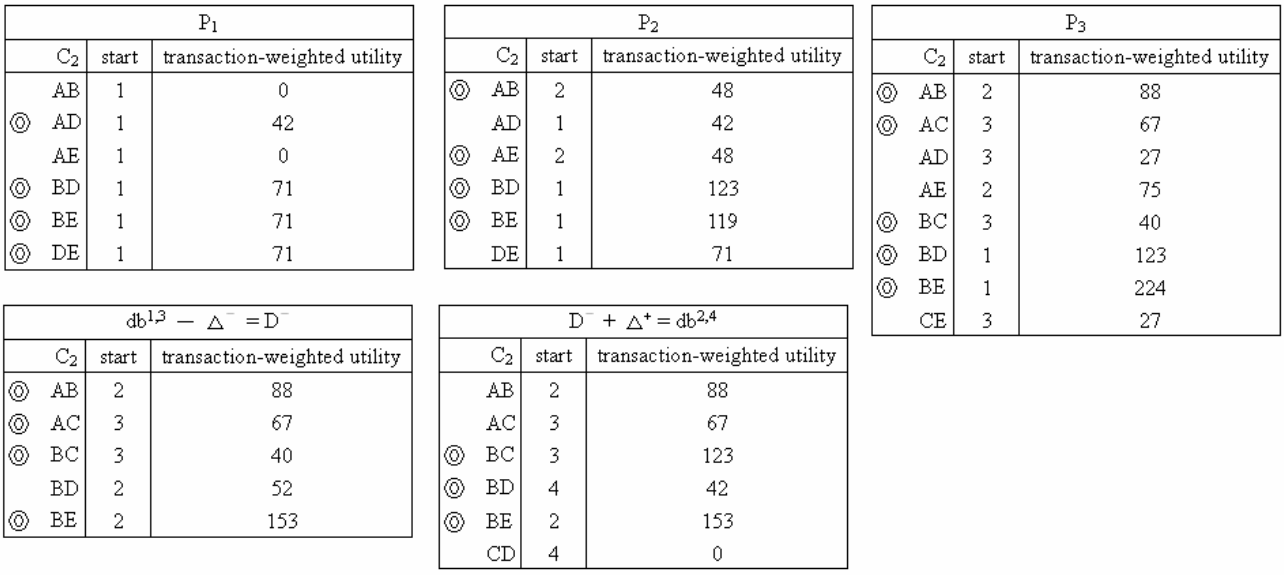
The proposed *THUI-Mine* algorithm can be best understood by the illustrative transaction database in Table 1 and Figure 2 where a scenario of generating high utility itemsets from data streams for mining temporal high utility itemsets is given. We set the utility threshold as 120 with nine transactions. Without loss of generality, the temporal mining problem can be decomposed into two procedures:

1. Preprocessing procedure: This procedure deals with mining on the original transaction database.

2. Incremental procedure: The procedure deals with the update of the high utility itemsets form data streams.

The preprocessing procedure is only utilized for the initial utility mining in the original database, e.g., $db^{1,n}$. For the generation of mining high utility itemsets in $db^{2,n+1}$, $db^{3,n+2}$, $db^{i,j}$, and so on, the incremental procedure is employed. Consider the database in Table 1. Assume that the original transaction database $db^{1,3}$ is segmented into three partitions, i.e., $\{P_1, P_2, P_3\}$, in the pre processing procedure. Each partition is scanned sequentially for the generation of candidate 2-itemsets in the first scan of the database $db^{1,3}$. After scanning the first segment of 3 transactions, i.e., partition P_1, 2-itemsets {AB, AD AE, BD, BE, DE} are generated as shown in Figure 2. In addition, each potential candidate itemset $c \in C_2$ has two attributes: (1) c.start which contains the identity of the starting partition when c was added to C_2, and (2) transaction-weighted utility which is the sum of the transaction utilities of all the transactions containing c since c was added to C_2. Since there are three partitions, the utility threshold of each partition is $120 / 3 = 40$. Such a partial utility threshold is called the filtering threshold in this paper. Itemsets whose transaction-weighted utility are below the filtering threshold are removed. Then, as shown in Figure 2, only {AD, BD, BE, DE}, marked by "◎", remain as temporal high transaction-weighted utilization 2-itemsets whose information is then carried over to the next phase of processing. Similarly, after scanning partition P_2, the temporal high transaction-weighted utilization 2-itemsets are recorded.

From Figure 2, it is noted that since there are also 3 transactions in P_2, the filtering threshold of those itemsets carried out from the previous phase is $40 + 40 = 80$ and that of newly identified candidate itemsets is 40. It can be seen from Figure 2 that we have 4 temporal high transaction-weighted utilization 2-itemsets in C_2 after the processing of partition P_2, and 2 of them are carried from P_1 to P_2 and 2 of them are newly identified in P_2. Finally, partition P_3 is processed by algorithm *THUI-Mine*. The resulting temporal high transaction-weighted utilization 2-itemsets are {AB, AC, BC, BD, BE} as shown in Figure 2. Note that though appearing in the previous phase P_2, itemset {AE} is removed from temporal high transaction-weighted utilization 2-itemsets once P_3 is taken into account since its transaction-weighted utility does not meet the filtering threshold then, i.e., $75 < 120$. However, we do have two new itemset, i.e., AC and BC, which join the C_2 as temporal high transaction-weighted utilization 2-itemsets. Consequently, we have 5 temporal high transaction-weighted utilization 2-itemsets generated by *THUI-Mine*, and 2 of them are carried from P_1 to P_3, 1 of them is carried from P_2 to P_3 and 2 of them are newly identified in P_3. Note that instead of 10 candidate itemsets that would be generated if Two-Phase algorithm were used, only 5 temporal high transaction-weighted utilization 2-itemsets are generated by *THUI-Mine*. After processing P_1 to P_3, those temporal high transaction-



P_1			
	C_2	start	transaction-weighted utility
	AB	1	0
◎	AD	1	42
	AE	1	0
◎	BD	1	71
◎	BE	1	71
◎	DE	1	71

P_2			
	C_2	start	transaction-weighted utility
◎	AB	2	48
	AD	1	42
◎	AE	2	48
◎	BD	1	123
◎	BE	1	119
	DE	1	71

P_3			
	C_2	start	transaction-weighted utility
◎	AB	2	88
◎	AC	3	67
	AD	3	27
	AE	2	75
◎	BC	3	40
◎	BD	1	123
◎	BE	1	224
	CE	3	27

$db^{1,3} - \Delta^- = D^-$			
	C_2	start	transaction-weighted utility
◎	AB	2	88
◎	AC	3	67
◎	BC	3	40
	BD	2	52
◎	BE	2	153

$D^- + \Delta^+ = db^{2,4}$			
	C_2	start	transaction-weighted utility
	AB	2	88
	AC	3	67
◎	BC	3	123
◎	BD	4	42
◎	BE	2	153
	CD	4	0

Figure 2. Temporal high utility itemsets generated from data streams by THUI-Mine.

weighted utilization itemsets in $db^{1,3}$ are {A, B, C, D, E, AB, AC, BC, BD, BE}.

After generating temporal high transaction-weighted utilization 2-itemsets from the first scan of database $db^{1,3}$, we employ the scan reduction technique and use temporal high transaction-weighted utilization 2-itemsets to generate C_k (k = 3, 4, ..., n), where C_n is the candidate last itemsets. It can be verified that temporal high transaction-weighted utilization 2-itemsets generated by *THUI-Mine* can be used to generate the candidate 3-itemsets. Clearly, a C_3 generated from temporal high transaction-weighted utilization 2-itemsets. For example, 3-candidate itemset {ABC} is generated from temporal high transaction-weighted utilization 2-itemsets {AB, AC, BC} in $db^{1,3}$. However, the temporal high transaction-weighted utilization 2-itemsets generated by *THUI-Mine* is very close to the high utility itemsets. Similarly, all C_k can be stored in main memory, and we can find temporal high utility itemsets together when the second scan of the database $db^{1,3}$ is performed. Thus, only two scans of the original database $db^{1,3}$ are required in the preprocessing step. The resulting temporal high utility itemsets are {B} and {BE} because u(B) = 330 >120 and u({B, E}) = 215 > 120 .

One important merit of *THUI-Mine* mainly lies in its incremental procedure. As depicted in Figure 2, the mining database will be moved from $db^{1,3}$ to $db^{2,4}$. Thus, some transactions, i.e., T_1, T_2, and T_3, are deleted from the mining database and other transactions, i.e., T_{10}, T_{11}, and T_{12}, are added. To illustrate more clearly, this incremental step can also be divided into three sub-steps: (1) generating temporal high transaction-weighted utilization 2-itemsets in $D^- = db^{1,3} - \Delta^-$, (2) generating temporal high transaction-weighted utilization 2-itemsets in $db^{2,4} = D^- + \Delta^+$ and (3) scanning the database $db^{2,4}$ only once for the generation of all temporal high utility itemsets. In the first sub-step, $db^{1,3} - \Delta^- = D^-$, we check out the pruned partition P_1, and reduce the value of transaction-weighted utility and set c.start = 2 for those temporal transaction-weighted utilization 2-itemsets where c.start = 1. It can be seen that itemset {BD} were removed. Next, in the second sub-step, we scan the incremental transactions in P_4. The process in $D^- + \Delta^+ = db^{2,4}$ is similar to the operation of scanning partitions, e.g., P_2, in the preprocessing step. The new itemset {BD} join the temporal high transaction-weighted utilization 2-itemsets after the scan of P_4. In the third sub-step, we use temporal high transaction-weighted utilization 2-itemsets to generate C_k as mentioned above. Finally, those temporal high transaction-weighted utilization itemsets in $db^{2,4}$ are {B, C, D, E, BC, BD, BE}. With scanning $db_{2,4}$ only once, *THUI-Mine* obtains temporal high utility itemsets {B, BC, BE} in $db^{2,4}$.

Table 3. Meanings of symbols used.

$db^{i,j}$	Partition_database (D) from P_i to P_j
s	Utility threshold in one partition
$\| P_k \|$	Number of transactions in partition P_k
TUP_k (I)	Trans. in P_k that contain itemset I with transaction utility
UP_k (I)	Trans. in P_k that contain itemset I with utility
$\| db^{1,n},(I) \|$	Trans. No. in $db^{1,n}$ that contain itemset I
$C^{i,j}$	The progressive candidate sets of $db^{i,j}$
$Thtw^{i,j}$	The progressive temporal high transaction-weighted utilization 2-itemsets of $db^{i,j}$
$Thu^{i,j}$	The progressive temporal high utility itemsets of $db^{i,j}$
Δ^-	The deleted portion of an ongoing database
D^-	The unchanged portion of an ongoing database
Δ^+	The added portion of an ongoing database

3.2 THUI-Mine Algorithm

For easier illustration, the meanings of various symbols used are given in Table 3. The preprocessing procedure and the incremental procedure of algorithm *THUI-Mine* are described in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 Preprocessing procedure of THUI-Mine

The preprocessing procedure of Algorithm *THUI-Mine* is shown in Figure 3. Initially, the database $db^{1,n}$ is partitioned into n partitions by executing the preprocessing procedure (in Step 2), and CF, i.e., cumulative filter, is empty (in Step 3). Let $Thtw^{1,n}$ be the set of progressive temporal high transaction-weighted utilization 2-itemsets of $db^{i,j}$. Algorithm *THUI-Mine* only records $Thtw^{1,n}$ which is generated by the preprocessing procedure to be used by the incremental procedure. From Step 4 to Step 16, the algorithm processes one partition at a time for all partitions. When partition P_i is processed, each potential candidate 2-itemset is read and saved to CF. The transaction-weight utility of an itemset I and its starting partition are recorded in I.twu and I.start, respectively. An itemset, whose I.twu ≥ s, will be kept in CF. Next, we select $Thtw^{1,n}$ from I where I∈ CF and keep I.twu in main memory for the subsequent incremental procedure. With employing the scan reduction technique from Step 19 to Step 26, $C_h^{1,n}$ (h ≥ 3) are generated in main memory. After refreshing I.count = 0 where I.twu = 0 where I∈ $Thtw^{1,n}$, we begin the last scan of database for the preprocessing procedure from Step 28 to Step 31. Finally, those itemsets satisfying the constraint that I.u ≧ s×P.count are finally obtained as the temporal high utility itemsets.

```
1. n = Number of partitions;
2. |db^{1,n}| = Σ_{k=1,n} |P_k|;
3. CF = φ;
4. begin for k = 1 to n                    // 1st scan of db^{1,n}
5.       begin for each 2-itemset I ∈ P_k
6.             if ( I ∉ CF )
7.                 I.twu = TUP_k (I);
8.                 I.start = k;
9.                 if (I.twu ≥ s×P.count )     // P.count is number of partitions
10.                      CF = CF ∪ I;
11.            if ( I ∈ CF )
12.                I.twu = I.twu + TUP_k (I);
13.                if (I.twu < s×P.count )
14.                     CF = CF − I;
15.      end
16. end
17. select Thtw^{1,n} from I where I ∈ CF;
18. keep Thtw^{1,n} in main memory;
19. h = 2;
20. begin while (Thtw^{1,n} ≠ φ)       //Database scan reduction
21.    if (h=2)
22.       C_{h+1}^{1,n} = Thtw^{1,n} * Thtw^{1,n} ;
23.    else
24.       C_{h+1}^{1,n} = C_h^{1,n} * C_h^{1,n}
25.    h = h + 1;
26. end
27. refresh I.twu = 0 where I ∈ Thtw^{1,n} ;
28. begin for k = 1 to n                   //2nd scan of db^{1,n}
29.      for each itemset I ∈ Thtw^{1,n} ∪ C_{h+1}^{1,n}
30.           I.u = I.u + UP_k (I);
31. end
32. for each itemset I ∈ Thtw^{1,n} ∪ C_{h+1}^{1,n}
33.      if (I.u ≧ s×P.count )
34.           Thu^{1,n} = Thu^{1,n} ∪ I;
35. end
36. return Thu^{1,n};
```

Figure 3. Preprocessing procedure of THUI-Mine.

```
1. Original database = db^{m,n};
2. New database = db^{i,j} ;
3. Database removed Δ^− = Σ_{k=m,i-1} P_k;
4. Database added Δ^+ = Σ_{k=n+1,j} P_k;
5. D^− = Σ_{k=i,n} P_k;
6. db^{i,j} = db^{m,n} −Δ^− + Δ^+;
7. loading Thtw^{m,n} of db^{m,n} into CF where I ∈ Thtw^{m,n};
8. begin for k = m to i − 1 // one scan of Δ^−
9.      begin for each 2-itemset I ∈ P_k
10.          if ( I ∈ CF and I.start ≤ k )
11.             I.twu = I.twu − TUP_k (I);
12.             I.start = k + 1;
13.             if ( I.twu < s×P.count )
14.             CF = CF − I;
15.     end
16. end
17. begin for k = n +1 to j // one scan of Δ^+
18.     begin for each 2-itemset I ∈ P_k
19.          if ( I ∉ CF )
20.             I.twu = TUP_k (I);
21.             I.start = k;
22.             if (I.twu ≥ s×P.count )
23.                 CF = CF ∪ I;
24.             if ( I ∈ CF )
25.                 I.twu = I.twu + TUP_k (I);
26.                 if (I.twu < s×P.count)
27.                     CF = CF − I;
28.     end
29. end
30. select Thtw^{i,j} from I where I ∈ CF;
31. keep Thtw^{i,j} in main memory;
32. h = 2
33. begin while (Thtw^{i,j} ≠ φ)       //Database scan reduction
34.     if (h=2)
35.        C_{h+1}^{i,j} = Thtw^{i,j} * Thtw^{i,j} ;
36.     else
37.        C_{h+1}^{i,j} = C_h^{i,j} * C_h^{i,j}
38.     h = h + 1;
39. end
40. refresh I.twu = 0 where I ∈ Thtw^{i,j} ;
41. begin for k = i to j                //2nd scan of db^{i,j}
42.     for each itemset I ∈ Thtw^{i,j} ∪ C_{h+1}^{i,j}
43.          I.u = I.u + UP_k (I);
44. end
45. for each itemset I ∈ Thtw^{i,j} ∪ C_{h+1}^{i,j}
46.     if (I.u ≧ s×P.count )
47.          Thu^{i,j} = Thu^{i,j} ∪ I;
48. end
49. return Thu^{i,j};
```

Figure 4. Incremental procedure of THUI-Mine.

3.2.2 *Incremental procedure of THUI-Mine*

As shown in Table 3, D^- indicates the unchanged portion of an ongoing transaction database. The deleted and added portions of an ongoing transaction database are denoted by Δ^- and Δ^+, respectively. It is worth mentioning that the sizes of Δ^+ and Δ^-, i.e., $|\Delta^+|$ and $|\Delta^-|$ respectively, are not required to be the same. The incremental procedure of *THUI-Mine* is devised to maintain temporal high utility itemsets efficiently and effectively. This procedure is shown in Figure 4. As mentioned before, this incremental step can also be divided into three sub-steps: (1) generating temporal high transaction-weighted utilization 2-itemsets in $D^- = db^{1,3} - \Delta^-$, (2) generating temporal high transaction-weighted utilization 2-itemsets in $db^{2,4} = D^- + \Delta^+$ and (3) scanning the database $db^{2,4}$ only once for the generation of all temporal high utility itemsets. Initially, after some update activities, old transactions Δ^- are removed from the database $db^{m,n}$ and new transactions Δ^+ are added (in Step 6). Note that $\Delta^- \subset db^{m,n}$. Denote the updated database as $db^{i,j}$. Note that $db^{i,j} = db^{m,n} - \Delta^- + \Delta^+$. We denote the unchanged transactions by $D^- = db^{m,n} - \Delta^- = db^{i,j} - \Delta+$. After loading $Thtw^{m,n}$ of $db^{m,n}$ into CF where $I \in Thtw^{m,n}$, we start the first sub-step, i.e., generating temporal high transaction-weighted utilization 2-itemsets in $D^- = db^{m,n} - \Delta^-$. This sub-step tries to reverse the cumulative processing which is described in the preprocessing procedure. From Step 8 to Step 16, we prune the occurrences of an itemset I, which appeared before partition P_i, by deleting the value I.twu where $I \in$ CF and I.start < i. Next, from Step 17 to Step 39, similarly to the cumulative processing in Section 3.2.1, the second sub-step generates generating temporal high transaction-weighted utilization 2-itemsets in $db^{i,j} = D^- + \Delta^+$ and employs the scan reduction technique to generate $C_{h+1}^{i,j}$. Finally, to generate temporal high utility itemsets, i.e., $Thu^{i,j}$, in the updated database, we scan $db^{i,j}$ for only once in the incremental procedure to find temporal high utility itemsets. Note that $Thtw^{i,j}$ is kept in main memory for the next generation of incremental mining.

4. EXPERIMENTAL EVALUATION

To evaluate the performance of *THUI-Mine*, we conducted experiments of using synthetic dataset generated via a randomized dataset generator provided by IBM Quest project [3]. However, the IBM Quest data generator only generates the quantity of 0 or 1 for each item in a transaction. In order to fit databases into the scenario of utility mining, we randomly generate the quantity of each item in each transaction, ranging from 1 to 5, as is similar to the model used in [14]. Utility tables are also synthetically created by assigning a utility value to each item randomly, ranging from 1 to 1000. Observed from real world databases that most items are in the low profit range, we generate the utility values using a log normal distribution, as is similar to the model used in [14]. Figure 5 shows the utility value distribution of 1000 items.

The simulation is implemented in C++ and conducted in a machine with 2.4GHz CPU and 1G memory. The main performance metrices used is execution time. We recorded the execution time that *THUI-Mine* spends in finding temporal high utility itemsets. The number of itemsets comparison of *THUI-Mine*, Two-Phase and MEU is presented in Section 4.1. Section 4.2 shows the performance comparison of *THUI-Mine* and Two-Phase. Results on scaleup experiments are presented in Section 4.3.

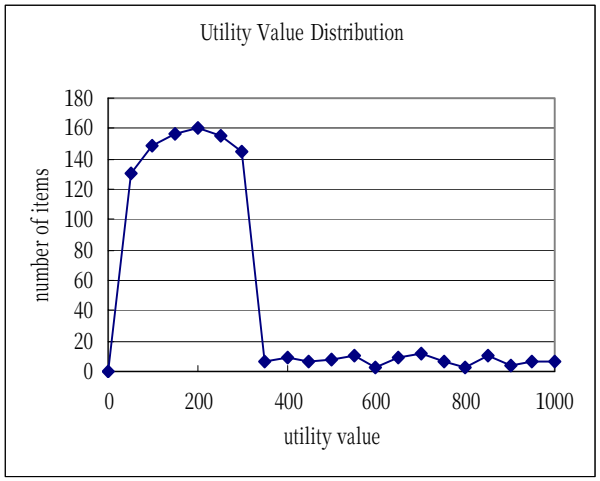


Figure 5. Utility value distribution in utility table.

4.1 Number of Generated Candidates

In this experiment, we compare the average number of candidates generated in the first database scan on the sliding windows and incremental transaction number d10K with different support values for *THUI-Mine*, Two-Phase [14] and MEU [19]. Without loss of generality, we set $|d| = |\Delta^+| = |\Delta^-|$ for simplicity. Thus, by denoting the original database as $db^{1,n}$ and the new mining database as $db^{i,j}$, we have $|db^{i,j}| = |db^{1,n} - \Delta^- + \Delta^+| = |D|$, where $\Delta^- = db^{1,i-1}$ and $\Delta^+ = db^{n+1,j}$. Table 4 and Table 5 show the average number of candidates generated by *THUI-Mine*, Two-Phase and MEU. The number of items is set at 1000, and the minimum utility threshold varies from 0.2% to 1%. The number of candidate itemsets generated by *THUI-Mine* at the first database scan decreases dramatically as the threshold goes up. Especially, when utility threshold is set as 1%, the number of candidate itemsets is 0 in database T10.I6.D100K.d10K where T denotes the average size of the transactions and I the average number of frequent itemsets. However, the number of candidates generated by Two-Phase is still very large and MEU is always 499,500 because it needs to process all combinations of 1000 items. *THUI-Mine* generates much fewer candidates compared to Two-Phase and MEU.

We obtain similar experimental results for different datasets. For example, only 118 candidate itemsets generated by *THUI-Mine*, but 183921 and 499500 candidate itemsets generated by Two-Phase and MEU when the utility threshold is set as 1% in dataset T20.I6.D100K.d10K. In the case of dataset T20.I6.D100K.d10K, more candidates are generated, because each transaction is longer than that in T10.I6.D100K.d10K. In overall, our algorithm *THUI-Mine* can always generate much fewer candidates compared to Two-Phase and MEU for various kinds of databases. Hence, *THUI-Mine* is verified to be very effective in pruning candidate itemsets to find temporal high utility itemsets.

Table 4. The average number of candidate itemsets generated by *THUI-Mine*, Two-Phase and MEU after the first scan on database T10.I6.D100K.d10K.

Databases / Threshold	T10.I6.D100K.d10K		
	THUI-Mine	Two-Phase	MEU
0.2%	3433	361675	499500
0.3%	666	303810	499500
0.4%	161	258840	499500
0.6%	7	182710	499500
0.8%	1	129286	499500
1%	0	91378	499500

Table 5. The average number of candidate itemsets generated by *THUI-Mine*, Two-Phase and MEU after the first scan on database T20.I6.D100K.d10K.

Databases / Threshold	T20.I6.D100K.d10K		
	THUI-Mine	Two-Phase	MEU
0.2%	27357	401856	499500
0.3%	11659	371953	499500
0.4%	5389	337431	499500
0.6%	1364	278631	499500
0.8%	371	229503	499500
1%	118	183921	499500

4.2 Evaluation of Execution Efficiency

In this experiment, we show only the relative performance of Two-phase and *THUI-Mine* since MEU spends much higher execution time and becomes incomparable. Figure 6 and Figure 7 show the execution times for the two algorithms as the minimum utility threshold is decreased from 1% to 0.2%. It is observed that when the minimum utility threshold is high, there are only a limited number of high utility itemsets produced. However, as the minimum utility threshold decreases, the performance difference becomes prominent in that *THUI-Mine* significantly outperforms Two-Phase. As shown in Figure 6 and Figure 7, *THUI-Mine* leads to prominent performance improvement for different average sizes of transaction. Explicitly, *THUI-Mine* is in orders of magnitude faster than Two-Phase, and the margin grows as the minimum utility threshold decreases. We could observe that *THUI-Mine* spends fewer times than Two-Phase with high stability for finding temporal high utility itemsets. This is because Two-Phase algorithm produces more candidate itemsets and needs more database scans to find high utility itemsets than our *THUI-Mine* algorithm. To measure how much execution time could be reduced substantially in using *THUI-Mine* compared to Two-Phase algorithm, we define the Improvement Ratio as follows:

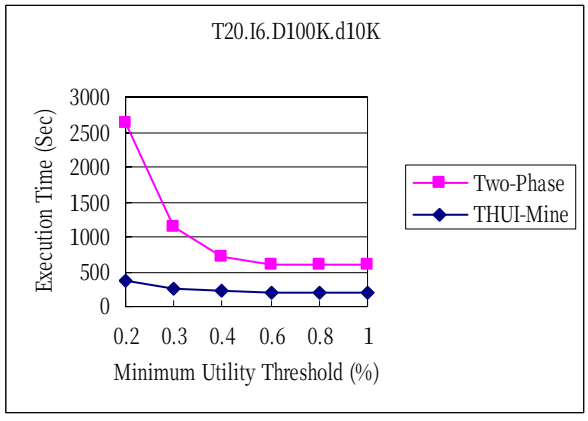


Figure 6. Execution time for Two-Phase and THUI on T20.I6.D100K.d10K.

Improvement Ratio = (execution time of Two-Phase − execution time of THUI-Mine) / (execution time of Two-Phase)

From Figure 6, we get that the Improvement Ratio is about 85.6% with the threshold set as 0.2%. In Figure 7, the average improvement is about 67% with minimum utility threshold varied from 0.2% to 1%. Obviously, *THUI-Mine* reduces substantially the time in finding high utility itemsets. Moreover, the high utility itemsets obtained by Two-Phase are not suitable for applications in data streams since Two-Phase needs more database scans and execution times in finding high utility itemsets by the time change. Hence, *THUI-Mine* meets the requirements of high efficiency in terms of execution time for data stream mining.

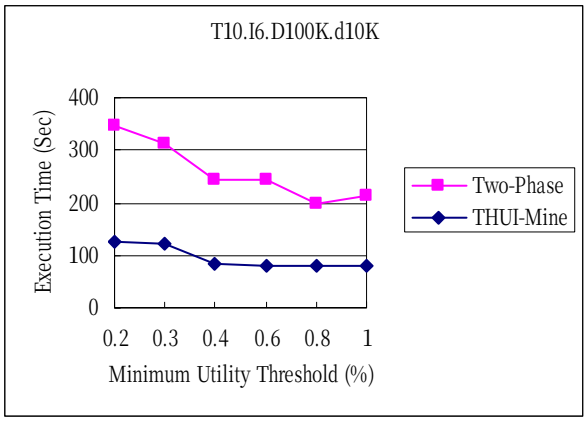


Figure 7. Execution time for Two-Phase and THUI on T10.I6.D100K.d10K.

4.3 Scaleup on Incremental Mining

In this experiment, we investigate the effects of varying incremental transaction size on the execution time of mining results. To further understand the impact of |d| on the relative performance of algorithms *THUI-Mine* and Two-Phase algorithms, we conduct the scaleup experiments which is similar in [12] for both *THUI-Mine* and Two-Phase with minimum support

thresholds being varied as 0.2% and 0.4 %, respectively. Figure 8 shows the experimental results where the value in y-axis corresponds to the ratio of the execution time of *THUI-Mine* to that of Two-Phase. Figure 8 shows the execution-time-ratio for different values of |d|. It can be seen that the execution-time ratio keeps stable with the growth of the incremental transaction number |d| since the size of |d| has little influence on the performance of *THUI-Mine*. Moreover, the execution time ratio of the scaleup experiments with minimum support thresholds varied from 0.6% to 1% keeps still around 0.4%. This implies that the advantage of *THUI-Mine* over Two-Phase is stable and less execution times is taken as the amount of incremental portion increases. This result also indicates that *THUI-Mine* fits for mining data streams with large transaction size.

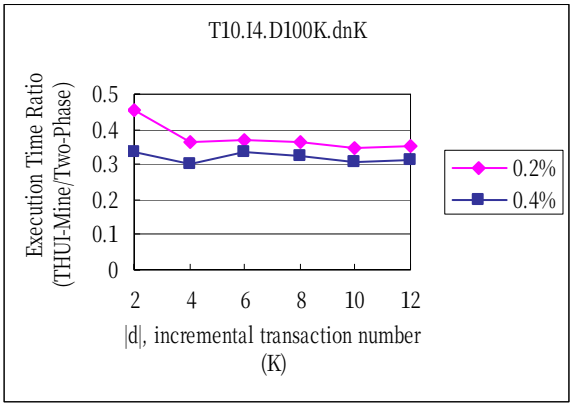


Figure 8. Scaleup performance with the execution time ratio between THUI and Two-Phase.

5. CONCLUSIONS

In this paper, we addressed the problem of discovering temporal high utility itemsets in data streams, i.e., the itemsets that are large than threshold in current time window of data stream. We propose a new approach, namely *THUI-Mine*, which can discover temporal high utility itemsets from data streams efficiently and effectively. The novel contribution of *THUI-Mine* is that it can effectively identify the temporal high utility itemsets with less temporal high transaction-weighted utilization 2-itemsets such that the execution time can be reduced efficiently in mining all high utility itemsets in data streams. In this way, the process of discovering all temporal high utility itemsets under all time windows of data streams can be achieved effectively with limited memory space, less candidate itemsets and CPU I/O. This meets the critical requirements of time and space efficiency for mining data streams.

The experimental results show that *THUI-Mine* can find the temporal high utility itemsets with higher performance by generating less candidate itemsets compared to other algorithms under different experimental conditions. Moreover, it performs scalable in terms of execution time under large databases. Hence, *THUI-Mine* is promising for mining temporal high utility itemsets in data streams. For future work, we would extend the concepts proposed in this work to discover other interesting patterns in data streams like utility item with negative profit.

ACKNOWLEDGMENTS

This research was supported in part by Ministry of Economic Affairs, R.O.C., under grant no. 92-EC-17-A-02-S1-024.

REFERENCES

[1] Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In Proceedings of 1993 ACM SIGMOD Intl. Conf. on Management of Data, pages 207--216, Washington, D. C., May 1993.

[2] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I. Fast discovery of association rules. In book Advances in Knowledge Discovery and Data Mining, pages 307--328. AAAI/MIT Press, 1996.

[3] Agrawal, R., and Srikant, R. Mining Sequential Patterns. *Proceedings of the 11th International Conference on Data Engineering*, pages 3-14, March 1995.

[4] Ayn, N.F., Tansel, A.U., and Arun, E. An efficient algorithm to update large itemsets with early pruning. Technical Report BU-CEIS-9908 Dept of CEIS Bilkent Uniiversity, June 1999.

[5] Ayn, N.F., Tansel, A.U., and Arun, E. An efficient algorithm to update large itemsets with early pruning. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999.

[6] Bettini, C., Wang, X. S., and Jajodia, S. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proc. of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada*, pages 68–78. ACM Press, 1996.

[7] Chan, R., Yang, Q., and Shen, Y. Mining high utility Itemsets. *Proc. of IEEE ICDM*, Florida, 2003.

[8] Cheung, D., Han, J., Ng, V., and Wong, C.Y. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. Proc. of 1996 Int'l Conf. on Data Engineering, pages 106—114, February 1996.

[9] Cheung, D., Lee, S.D., and Kao. B., A General Incremental Technique for Updating Discovered Association Rules. Proc. International Conference On Database Systems For Advanced Applications, April 1997.

[10] Chi, Y., Wang, H., Yu, P. S., and Richard, R. Muntz: Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window Proceedings of the 2004 IEEE International Conference on Data Mining (ICDM'04).

[11] Das, G., Lin, K. I., Mannila, H., Renganathan G., and Smyth, P. Rule Discovery from Time Series. *Proceedings of the 4th ACM SIGKDD*, pages 16—22, August 1998.

[12] Lee, C. H., Lin, C. R., and Chen, M. S. Sliding-window filtering: An efficient algorithm for incremental mining. In *Intl. Conf. on Information and Knowledge Management* (CIKM01), pages 263－270, November 2001.

[13] Lin, J. L., and Dunham. M. H. Mining Association Rules: Anti-Skew Algorithms. Proc. of 1998 Int'l Conf. on Data Engineering, pages 486—493, 1998.

[14] Liu, Y., Liao, W., and Choudhary, A. A Fast High Utility Itemsets Mining Algorithm. In *Proceedings of the Utility-Based Data Mining Workshop*, August 2005.

[15] Park, J. S., Chen, M. S., and Yu. P. S. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. IEEE Transactions on Knowledge and Data Engineering, 9(5):813—825, October 1997.

[16] Savasere, A., Omiecinski, E., and Navathe, S. An Efficient Algorithm for Mining Association Rules in Large Databases. Proc. of the 21th International Conference on Very Large Data Bases, pages 432—444, September 1995.

[17] Teng, W. G., Chen, M. S., and Yu, P. S. A Regression-Based Temporal Pattern Mining Scheme for Data Streams. Proceedings of the 29th International Conference on Very Large Data Bases, pages 93—104, September 2003.

[18] Teng, W. G., Chen, M. S., and Yu, P. S. Resource-Aware Mining with Variable Granularities in Data Streams. SDM 2004.

[19] Yao, H., Hamilton, H. J., and Butz, C. J. A Foundational Approach to Mining Itemset Utilities from Databases. *Proc. of the4th SIAM International Conference on Data Mining*, Florida, USA, 2004.

A Unified Framework for Utility Based Measures for Mining Itemsets *

Hong Yao
Department of Computer Science, University of Regina
Regina, SK, Canada S4S 0A2
yao2hong@cs.uregina.ca

Howard J. Hamilton
Department of Computer Science, University of Regina
Regina, SK, Canada S4S 0A2
hamilton@cs.uregina.ca

Liqiang Geng
Department of Computer Science, University of Regina
Regina, SK, Canada S4S 0A2
gengl@cs.uregina.ca

ABSTRACT

A pattern is of utility to a person if its use by that person contributes to reaching a goal. Utility based measures use the utilities of the patterns to reflect the user's goals. In this paper, we first review utility based measures for itemset mining. Then, we present a unified framework for incorporating several utility based measures into the data mining process by defining a unified utility function. Next, within this framework, we summary the mathematical properties of utility based measures that will allow the time and space costs of the itemset mining algorithm to be reduced.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*

General Terms

Measures

Keywords

Data Mining, Knowledge Discovery, Interestingness Measures, Utility Based Measures, Utility Based Data Mining

1. INTRODUCTION

Data mining can be regarded as an algorithmic process that takes data as input and yields patterns, such as classification rules, itemsets, association rules, or summaries, as output. For example, frequent itemsets can be discovered from market basket data and used to derive association rules for predicting the conditional probability of the purchase of certain items, given the purchase of other items [1, 2, 9]. An *itemset* is a set of items. The goal of frequent itemset mining is to identify all *frequent itemsets*, i.e., itemsets that have at least a specified minimum *support*, which is the percentage of transactions containing the itemset. In this paper, we focus on itemset mining.

Interestingness measures can play an important role in knowledge discovery. These measures are intended for selecting and ranking patterns according to their potential interest to the user. For example, itemset mining is based on the assumption that only itemsets with high support are of interest to users. That is, the support measure uses frequency as an estimate of the utility of a pattern to a user.

Measuring the interestingness of discovered patterns is an active and important area of data mining research. A comprehensive study of twenty-one measures that were originally developed in diverse fields such as statistics, social science, machine learning, and data mining is presented by Tan et al. [19]. Hilderman and Hamilton [8] theoretically and empirically evaluated twelve diversity measures used as heuristic measures of interestingness for ranking summaries generated from dataset. Yao et al. [21] presented a simple and unified framework for the study of quantitative measures associated with rules. Most research on interestingness measures has focused on using a statistical or mathematical method to evaluate the usefulness of rules [10], but such a method is not trivial for a human expert to understand. In general, it is not easy for user to choose one of the measures, because even data mining specialists or practitioners may not be familiar with all available measures.

In practice, the frequency of occurrence may not express the semantics of applications, because the user's interest may be related to other factors, such as cost, profit, or aesthetic value. For example, simply choosing the frequent itemsets does not reflect the impact of any factor except the frequency of the items. The usefulness of the support measure is reduced by problems with the quantity and quality of the mining results. First, a huge number of frequent itemsets that are not interesting to the user are often generated when the minimum support is low. For example, there may be thousands of combinations of products that occur in 1% of the

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

transactions. If too many uninteresting frequent itemsets are found, the user is forced to do additional work to select the rules that are indeed interesting. Second, the quality problem is that support, as defined based on the frequency of itemsets, is not necessarily an adequate measure of a typical user's interest. A sales manager may not be interested in frequent itemsets that do not generate significant profit. In other word, frequent itemsets may only contribute a small portion of the overall profit, whereas non-frequent itemsets may contribute a large portion of the profit [12]. The following example shows that support based itemset mining may lead to some high profit itemsets not being discovered due to their low support.

Example 1. Consider the small transaction dataset shown in Table 1 and the unit profit for the items shown in Table 2. Each value in the transaction dataset indicates the quantity sold of an item. Using Table 1 and 2, the support and profit for all itemsets can be calculated (see Table 3). For example, since for the 10 transactions in Table 1, only two transactions, t_8 and t_9, include both items B and D, the support of the itemset BD is $2/10 = 20\%$. Since t_8 includes one B and one D, and t_9 includes one B and ten Ds, a total of two Bs and eleven Ds appear in transactions containing the itemset BD. Using the Table 2, the profit for each item B is 100 and the profit for each item D is 1. Thus, the profit of the itemsets BD could be considered to be $2 \times 100 + 11 \times 1 = 211$. The profit of the other itemsets in Table 3 can be obtained in a similar fashion. Supposing that the minimum support is 40%, the frequent itemsets in Table 3 are D, A, AD, and C, but the four most profitable itemsets are BD, B, AC, and CD, all of which are infrequent itemsets.

Transaction ID	Item A	Item B	Item C	Item D
t_1	4	0	1	0
t_2	2	0	0	6
t_3	0	0	1	30
t_4	3	0	0	5
t_5	1	0	0	6
t_6	4	0	2	10
t_7	2	0	0	8
t_8	1	1	1	1
t_9	0	1	0	10
t_{10}	5	0	0	9

Table 1: A transaction dataset.

Item Name	Profit ($)
Item A	5
Item B	100
Item C	38
Item D	1

Table 2: The profit table for the items.

In general, a pattern that is of interest to one user may not be of interest to another user, since users have different levels of interest in patterns. The support measure reflects the frequency of combinations of items, but it does not reflect their semantic significance. Thus, a user may incur a high computational cost that is disproportionate to what the user wants and gets [14]. A natural way for interesting measure

Itemsets	Support (%)	Profit ($)
A	**80**	110
B	20	**200**
C	**40**	190
D	**90**	85
AB	10	105
AC	30	**197**
AD	**70**	135
BC	10	138
BD	20	**211**
CD	30	**193**
ABC	10	143
ABD	10	106
ACD	20	150
BCD	10	139
$ABCD$	10	144

Table 3: The support, and the profits of all itemsets.

may allow a user to express his or her concern about the usefulness of results since only the user know his or her information need. That is, to allow data mining to further its impact on real-world applications, it is appropriate to consider user-specified interestingness, which bring more semantics of applications into data mining process and evaluate how user's expectation affect the data mining process.

To make clear the opportunity for a unified framework, we survey measures of interestingness for utility based data mining of itemsets. *Utility based data mining* refers to allowing a user to conveniently express his or her perspectives concerning the usefulness of patterns as utility values and then finding patterns with utility values higher than a threshold [20]. A pattern is of utility to a person if its use by that person contributes to reaching a goal. People may have differing goals concerning the knowledge that can be extracted from a data set. For example, one person may be interested in finding the sales with the most profit in a transaction data set. Another person may be interested in finding the largest increase in gross sales. This kind of interestingness is based on user-defined utility functions in addition to the raw data [4, 5, 6, 11, 13, 16, 20]. In fact, to achieve a user's goal, two types of utilities for items may need to be identified. The *transaction utility* of an item is directly obtained from the information stored in the transaction dataset. For example, the quantity of an item in Table 1 is a kind of transaction utility. The *external utility* of an item is given by the user. It is based on information not available in the transaction dataset. For example, a user's beliefs about the profit associated with items is expressed in Table 2. External utility often reflects user preference and can be represented by a utility table or utility function. By combining a transaction dataset and a utility table (or utility function) together, the discovered patterns will better match a user's expectations than by only considering the transaction dataset itself. To find patterns that conform to a user's interests, in this paper, we present a unified framework to show how utility measures are incorporated into data mining process by defining a unified utility function. Furthermore, three mathematical properties of this unified utility function are identified to allow the time and space costs of the mining algorithms to be reduced.

The remainder of this paper is organized as follows. In Section 2, we survey utility based measures for mining itemsets.

A framework for incorporating these utility measures in the data mining process is presented in Section 3. In Section 4, the mathematical properties of utility based measures are identified. Finally, conclusions are drawn in Section 5.

2. UTILITY BASED MEASURES

Researchers have proposed interestingness measures for various kinds of patterns, analyzed their theoretical properties, evaluated them empirically, and proposed strategies for selecting appropriate measures for particular domains and requirements. In data mining research, most interestingness measures have been proposed for evaluating itemsets and association rules. In this paper, we concentrate on interestingness measures that depend on the utility (usefulness) of the itemsets.

We begin by reviewing pertinent notions used for itemset mining. Adapting from the notation used in the descriptions of other itemset mining approaches [5, 16], we let $I = \{i_1, \ldots, i_p, i_q, \ldots, i_m\}$ be a set of items, where each item is associated with an attribute of a transaction dataset T. Each transaction t_q in T is a subset of I. An itemset S is a subset of I, i.e., $S \subseteq I$. To simplify notation, we sometimes write an itemset $\{i_1, \ldots, i_k\}$ as $i_1 \ldots i_k$; e.g., $ABCD$ represents itemset $\{A, B, C, D\}$. We denote the support value of itemset S as $s(S)$ and the utility value of itemset S as $u(S)$.

Definition 1. The *transaction set of an itemset* S, denoted T_S, is the set of transactions that contain itemset S, i.e., $T_S = \{t_q \mid S \subseteq t_q, t_q \in T\}$.

For instance, consider the transaction dataset shown in Table 1, supposing itemset S is $S = AD$. By definition, $T_S = \{t_2, t_4, t_5, t_6, t_7, t_8, t_{10}\}$.

A *utility based measure* is a measure that takes into consideration not only the statistical aspects of the raw data, but also the utility of the mined patterns. Motivated by the decision theory, Shen et al. stated that the "interestingness of a pattern = probability + utility" [17]. Based on the user's specific objectives and the utility of the mined patterns, utility-based mining approaches may be more useful in real applications, especially in decision making problems.

In this section, we review utility based measures for itemsets. Since we use a unified notation for all methods, some representations differ from those used in the original papers.

The simplest method to incorporate utility is called *weighted itemset mining*, which assigns each item a weight representing its importance [5, 11]. For example, the weights may correspond the profitability of different items; e.g., a computer (item A) may be more important than a phone (item B) in terms of profit. Weights assigned to items are also called horizontal weights [13]. The weights can represent the price or profit of a commodity. In this scenario, two measures are proposed to replace *support*. The first one is called *weighted support*, which is defined as

$$support_w(S) = (\sum_{i_p \in S} w_p)s(S), \quad (1)$$

where w_p denotes the weight of item i_p.

The first factor of the weighted support measure has a bias towards the rules with more items. When the number of the items is large, even if all the weights are small, the total weight may be large. The second measure, *normalized weighted support*, is proposed to reduce this bias and is defined as

$$support_{nw}(S) = \frac{1}{|S|}(\sum_{i_p \in S} w_p)s(S), \quad (2)$$

where $|S|$ is the number of items in the itemset S.

The traditional support measure is a special case of normalized weighted support, because when all weights for items are equal to 1, the normalized weighted support is identical to support. The Weighted Items (WI) approach [5] and the Value Added Mining (VAM) approach [11] use weighted items to capture the semantic significance of itemsets at the item level. Unlike frequent itemset mining, which treats all items uniformly, both of these approaches assume that items in a transaction dataset (columns in the table) have different weights to reflect their importance to the user.

Lu et al. proposed another data model by assigning a weight to each transaction [13]. The weight represents the significance of the transaction in the data set. Weights assigned to transactions are also called vertical weights [13]. For example, the weight can reflect the transaction time, i.e., more recent transactions can be given greater weights. Based on this model, *vertical weighted support* is defined as

$$support_v(S) = \frac{\sum_{t_q \in T_S} w_q}{\sum_{t \in T} w}, \quad (3)$$

where w_q and w denote the vertical weight for transactions t_q and t, respectively.

The mixed weighted model [13] uses both horizontal and vertical weights. In this model, each item is assigned a horizontal weight and each transaction is assigned a vertical weight. *Mixed weighted support* is defined as

$$support_m(S) = support_{nw}(S) \cdot support_v(S). \quad (4)$$

Both $support_v$ and $support_m$ are extensions of the traditional support measure. If all vertical and horizontal weights are set to 1, both $support_v$ and $support_m$ are identical to support.

Objective oriented utility based association (OOA) mining allows a user to set objectives for the mining process [17]. In this method, the attributes are partitioned into two groups, the target attributes and the non-target attributes. A *non-target attribute* (called an *nonobjective attribute* in [17] is only permitted to appear in the antecedents of association

rules. A *target attribute* (called an *objective attribute* in [17]) is only permitted to appear in the consequents of rules. The target attribute-value pairs are assigned utility values. The mining problem is to find frequent itemsets of non-target attributes, such that the utility values of their corresponding target attribute-value pairs are above a threshold. For example, in Table 4 obtained from [17], *Treatment* is a non-target attribute, while *Effectiveness* and *Side-effect* are two target attributes. The goal of the mining problem is to find treatments with high effectiveness and mild side effects. The utility measure is defined as

$$u(S) = \frac{1}{s(S)} \sum_{t_q \in T_S} u(t_q), \tag{5}$$

where S is the non-target itemsets to be mined (the *Treatment* attribute-value pairs in the example) and $u(t_q)$ denotes the utility of transaction t_q. The function $u(t_q)$ is defined as

$$u(t_q) = \sum_{i_p \in C_q} f(i_p), \tag{6}$$

where C_q denotes the set of target items in transaction t_q and $f(i_p)$ is the utility function of item i_p, which denotes the utility associated with i_p. If there is only one target attribute and its weight equals to 1, $\sum_{t_q \in T_S} u(t_q)$ is identical to $s(S)$, and hence $u(S)$ equals to 1.

Continuing the example, we assign the utility values to the target attribute-value pairs shown in Table 5 and accordingly obtain the utility values for the treatments shown in Table 6. For example, *Treatment 5* has the greatest utility value 1.2, and therefore, it best meets the user specified target.

TID	Treatment	Effectiveness	Side-effect
t_1	1	2	4
t_2	2	4	2
t_3	2	4	2
t_4	2	2	3
t_5	2	1	3
t_6	3	4	2
t_7	3	4	2
t_8	3	1	4
t_9	4	5	2
t_{10}	4	4	2
t_{11}	4	4	2
t_{12}	4	3	1
t_{13}	5	4	1
t_{14}	5	4	1
t_{15}	5	1	1
t_{16}	5	3	1

Table 4: A medical dataset.

The approach of Lu et al. [13] and OOA mining approach [6, 17] both capture the semantic significance of itemsets at the transaction level. They assume that transactions in a dataset (rows in the table) have associated utility values that reflect their importance to the user.

Effectiveness			Side-effect		
Value	Meaning	Utility	Value	Meaning	Utility
5	Much better	1	4	Very serious	-0.8
4	Better	0.8	3	Serious	-0.4
3	No effect	0	2	A little	0
2	Worse	-0.8	1	Normal	0.6
1	Much worse	-1			

Table 5: Utility values for *Effectiveness* and *Side-effect*.

Itemset	Utility
Treatment =1	-1.6
Treatment =2	-0.25
Treatment =3	-0.066
Treatment =4	0.8
Treatment =5	1.2

Table 6: Utilities of the items.

Hilderman et al. proposed the Itemset Share framework that takes into account weights on both attributes and attribute-value pairs [7]. The precise impact of the purchase of an itemset can be measured by the itemset *share*, the fraction of some overall numerical value, such as the total value of all items sold. For example, in a transaction data set, the weight on an attribute could represent the price of a commodity, and the weight on an attribute-value pair could represent the quantity of the commodity in a transaction. Based on this model, in the Itemset Share framework, support is generalized. The *count support* for itemset S is defined as

$$count\_sup(S) = \frac{\sum_{t_q \in T_S} \sum_{i_p \in S} w(i_p, t_q)}{\sum_{t \in T} \sum_{i \in I} w(i,t)}, \tag{7}$$

where $w(i_p, t_q)$ denotes the weight of attribute i_p for transaction t_q and $w(i_p, t_q) > 0$.

Similarly, the *amount support* is defined as

$$amount\_sup(S) = \frac{\sum_{t_q \in T_S} \sum_{i_p \in S} w(i_p, t_q) w(i_p)}{\sum_{t \in T} \sum_{i \in I} w(i,t) w(i)}, \tag{8}$$

where $w(i_p)$ is the weight for attribute i_p and $w(i_p) > 0$.

Based on the data model in [7], Yao et al. proposed another utility measure [20], defined as

$$u(S) = \sum_{t_q \in T_S} \sum_{i_p \in S} w(i_p, t_q) w(i_p), \tag{9}$$

where $w(i_p, t_q)$ denotes the utility value of attribute i_p for transaction t_q, $w(i_p)$ denotes the utility value of attribute i_p, $w(i_p, t_q) > 0$ and $w(i_p) > 0$.

This utility function is similar to amount support, except that it represents a utility value, such as the profit in dollars, rather than a fraction of the total weight of all transactions in the data set.

The Itemset Share (IS) approach [4] and the approach of Yao et al. [20] capture the semantic significance of numerical values that are typically associated with the individual items in a transaction dataset (cells in the table).

Table 7 summarizes the utility measures discussed in this section by listing the name of each measure and its data model. The data model describes how the information relevant to the utility is organized in the data set. All these measures are extensions of the support and confidence measures. No single utility measure is suitable for every application, because applications have different objectives and data models. Given a data set, one could choose a utility measure by examining the data models for the utility measures given in Table 7. For example, if one has a data set with weights for each row, then one might choose the vertical weighted support measure. By checking Table 7 carefully, we find that the difference among these models are: (1) different levels of granularity (item level, transaction level, and cell level) are used to specify the semantic significance of itemsets, and (2) different pruning strategies are developed according to the properties of these measure functions. For (1), we present a unified framework for utility base measures that incorporates existing utility based measures into data mining process in Section 3. For (2), we summarize the mathematical properties of the unified framework for utility base measures in Section 4.

3. A UNIFIED FRAMEWORK FOR UTILITY BASED MEASURES

During the knowledge discovery process, utility based measures can be used in three ways, which we call the roles of the utility based measures. Figure 1 shows these three roles. First, measures can be used to prune uninteresting patterns during the data mining process to narrow the search space and thus improve the mining efficiency. For example, a threshold for support can be used to filter out patterns with low support during the mining process and thus improve efficiency [2]. Similarly, a utility threshold can be defined and used for pruning patterns with low utility values [20]. Secondly, measures can be used to rank the patterns according to the order of their interestingness scores. Thirdly, measures can be used during post processing to select the interesting patterns. For example, after the data mining process, we can use the chi-square test to select the rules that have significant correlations [3]. The second and third approaches can also be combined by first filtering the patterns and then ranking them. For the second or third approach, utility based measures need not be incorporated into the data mining algorithm. In this paper, we concentrate on first method since it can improve the mining efficiency by reducing the time and space costs of the mining algorithm.

Now, formal definitions of key terms used in our unified utility framework for utility measures for mining itemsets are presented.

We denote the utility value of an itemset S as $u(S)$, which will be described in more detail shortly.

Definition 2. The *utility constraint* is a constraint of the form $u(S) \geq minutil$.

Definition 3. An itemset S is a *high utility pattern* if $u(S) \geq minutil$, where $minutil$ is the threshold defined by the user. Otherwise, S is a *low utility itemset.*

Based on the utility constraint, the unified utility framework for utility measures is defined as follows.

Definition 4. The *utility based itemset mining problem* is to discover the set H of all high utility itemsets, i.e.,

$$H = \{S \mid S \subseteq I, u(S) \geq minutil\}. \tag{10}$$

For example, consider the itemsets in Table 3. If $u(S)$ is the profit of an itemset S and $minutil = 150$, then $H = \{B, C, AC, BD, CD, ACD\}$.

According to the survey presented in Section 2, $u(S)$ plays a key role in specifying utility based data mining problems. Different utility measures use different formulas for $u(S)$.

Now, we show how to define $u(S)$ in terms of a user defined utility function f. In Example 1, the profit of an itemset reflects a store manager's goal of discovering itemsets producing significant profit (e.g., $minutil = 150$). A user judges BD to be useful, since the profit of itemset BD is greater than $minutil$. We observe that the semantic meaning of profit can be captured by a function $f(x, y)$, where x is the quantity sold of an item and y is the unit profit of an item. The usefulness of an itemset is quantified as the product of x and y, namely, $f(x, y) = x \cdot y$. The value of x can be obtained from the transaction dataset and depends only on the underlying dataset [18]. On the other hand, the value of y is often not available in a transaction dataset and may depend on the user who examines the pattern [18]. Thus, in this case, the significance of an item is measured by two parts. One is the statistical significance of the item measured by parameter x, which is an objective term independent of its intended application. The other part is the semantic significance of the item measured by parameter y, which is a subjective term dependent on the application and the user. As a result, $f(x, y)$ combines objective and subjective measures of an item together. The combination captures the significance of the itemset for this application, which reflects not only the statistical significance but also the semantic significance of the itemset.

Definition 5. The *transaction utility value of an item*, denoted x_{pq}, is the value of an attribute associated with an item i_p in a transaction t_q.

For example, in Table 1, the quantity sold values in the transactions are the transaction utility values. If $i_4 = D$, then $x_{43} = 30$ is the transaction utility value of item D in transaction t_3.

In this paper, we restrict transaction utility variable values to numerical values, because, typically, transaction utility information can be represented in this form.

Measures	Data models	Extension of
Weighted support	Weights for items	Support
Normalized weighted support	Weights for items	Support
Vertical weighted support	Weights for transactions	Support
Mixed weighted support	Weights for both items and transactions	Support
OOA Target and non-target attributes	Weights on transaction for target attributes	Support
Count support	Weights for items and cells in data set	Support
Amount support	Weights for items and cells in data set	Support
Count confidence	Weights for items and cells in data set	Confidence
Amount confidence	Weights for items and cells in data set	Confidence
Yao et al.'s	Weights for items and cells in data set	Support

Table 7: Utility based interestingness measures.

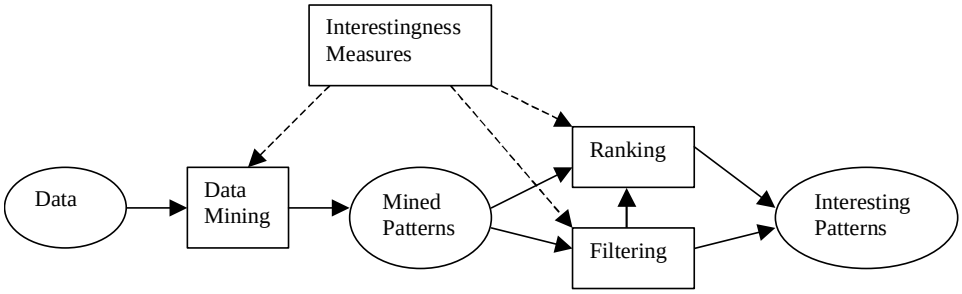


Figure 1: Roles of utility based measures.

Definition 6. The *external utility value of an item*, denoted y_p, is a real number assigned by the user such that for any two items i_p and i_q, y_p is greater than y_q iff the user prefers item i_p to item i_q.

The definition indicates that a external utility value is associated with a specific value in a domain to express user preference. In practice, the value of y_p is assigned by the user according to his interpretation of domain specific knowledge measured by some utility factors, such as cost, profit, or aesthetic value. For example, let $i_1 = A$ and $i_2 = B$. Using the Table 2, we have $y_1 = 5$ and $y_2 = 100$. The inequality $y_2 > y_1$ reveals that the store manager prefers item B to item A, since each item B earns more profit than each item A.

By obtaining the transaction utility value x_{pq} from a transaction dataset and the external utility value y_p from the user, a utility function to express the significance of an itemset can be defined as a two dimensional function $f(x, y)$.

Definition 7. A *utility function* f is a function $f(x, y) : (R, R) \rightarrow R$, where R is the set of real numbers.

Example 2. Consider the transaction dataset in Table 1 and the profit table in Table 2. Let items i_1, i_2, i_3, and i_4 be items A, B, C, and D, respectively. Suppose that the user defines utility function $f(x_{pq}, y_p)$ as $f(x_{pq}, y_p) = x_{pq} \cdot y_p$, where x_{pq} is the quantity sold of an item i_p in a transaction t_q, and y_p is the unit price of the item i_p. Then $f(x_{11}, y_1) = 4 \times 5 = 20$, which indicates that the supermarket earns \$20 by selling four As in transaction t_1. Similarly, $f(x_{21}, y_2) = 0$, $f(x_{31}, y_3) = 1 \times 38 = 38$, and $f(x_{41}, y_4) = 0$.

The utility value of an item is the sum of the values of the utility function for each transaction.

Definition 8. The *utility value of an item i_p in an itemset S*, denoted $l(i_p, S)$, is the sum of the values of the utility function $f(x_{pq}, y_p)$ for each transaction t_q in T_S, i.e.,

$$l(i_p, S) = \sum_{t_q \in T_S} f(x_{pq}, y_p). \tag{11}$$

For example, consider the transaction dataset in Table 1 with the profit table in the Table 2. If $S = ACD$, then $T_S = \{t_6, t_8\}$, thus $l(A, S) = 4 \times 5 + 1 \times 5 = 25$.

The utility value of an itemset is represented by the sum of the utility values of every item in the itemset.

Definition 9. The *utility value of an itemset S*, denoted $u(S)$, is the sum of the utility value of each item in S, i.e.,

$$u(S) = \sum_{i_p \in S} l(i_p, S). \tag{12}$$

By substituting Equation 11 into Equation 12, we obtain

$$u(S) = \sum_{i_p \in S} \sum_{t_q \in T_S} f(x_{pq}, y_p). \tag{13}$$

For example, given $f(x_{pq}, y_p) = x_{pq} \cdot y_p$, for itemset $S = ACD$, we have $T_S = \{t_6, t_8\}$, then $u(S) = l(A, S) + l(C, S) + l(D, S) = 5 \times 5 + 3 \times 38 + 11 \times 1 = 150$.

Equation 13 indicates that user plays an important role in utility based itemset mining process since a user can measure the semantic significance of the itemset by using his own utility function $f(x, y)$. Therefore, an itemset that is of interest to one user, may be of no interest to another user, since users have different levels of interest in itemsets, as expressed by their utility functions. In other word, different itemsets may be discovered for two users according to their interests, as expressed by their utility functions.

Based on the utility formulation of an itemset (Equation (13)), an efficient algorithm, called UMining [20], has been developed to find the high profit itemsets from a dataset.

Now we show that the utility function $f(x, y)$ is a unified utility function. Let c be a constant. Table 8 summarizes the semantic significance of this unified utility function at the item level, the transaction level, and the cell level. Table 9 shows how to use our unified utility function to represent all existing utility based measures described in Section 2. In this framework, by defining deferent $f(x, y)$, several existing utility based measures can be obtained.

4. MATHEMATICAL PROPERTIES OF UTILITY BASED MEASURES

In this section, we analyze the mathematical properties of the utility function $f(x, y)$ to facilitate the design of efficient mining algorithms that will reduce the time and space costs of the mining process.

Three important mathematical properties of utility based measures, namely, the anti-monotone (or monotone) property, the convertible property, and the upper bound property, have been identified and used in existing utility based measures [1, 2, 5, 6, 13, 9, 16, 20].

Definition 10. [15]. A constraint C is *anti-monotone* iff whenever an itemset S violates a constraint C, so does any superset of S. A constraint C is *monotone* iff whenever an itemset S satisfies a constraint C, so does any superset of S.

By definition, the Apriori property [2] that applied to the support measure is a special case of the anti-monotone property that focuses only on the support constraint.

Definition 11. [15]. An itemset $S_1 = i_1 \ldots, i_m$ is a *prefix itemset* of itemset $S_2 = i_1 \ldots, i_n$ if the items in S_1 and S_2 are listed in the same order and $m \leq n$.

For example, suppose we are given an itemset $ABCD$. By Definition 11, itemsets A, AB, and ABC are prefix itemsets of $ABCD$ with respect to the order $\langle A, B, C, D\rangle$

Based on the prefix itemsets of an itemset, the convertible property of the itemset is defined as follows.

Definition 12. [15]. A constraint C is *convertible anti-monotone* w.r.t. an order $\mathcal{O}$ on items if and only if whenever an itemset S satisfies property P, so do any prefix itemsets of S. A constraint C is *convertible monotone* w.r.t. an $\mathcal{O}$ on items if and only if whenever an itemset S violates property P, so do any prefix itemsets of S. A constraint C is *convertible* w.r.t. an order $\mathcal{O}$ if and only if it is convertible anti-monotone or convertible monotone w.r.t. the order $\mathcal{O}$.

The following example shows a constraint that is convertible.

Example 3. Consider the profit table for the items shown in Table 2. Let $avg(S) \geq 30$ be a constraint on the average profit of the itemset S. We have $avg(ABCD) = (5 + 100 + 38 + 1)/4 = 36$. If the items are sorted in unit profit descending order, we get $\langle B, C, A, D\rangle$. The itemset $BCAD$ has BCA, BC, and B as its prefix itemsets w.r.t. the order $\langle B, C, A, D\rangle$. Then we have $avg(BCA) = 47.67$, $avg(BC) = 69$, and $avg(B) = 100$. The average profit of the itemset $BCAD$ is greater than 30, as are the average profits for its all prefix itemsets according to the order $\langle B, C, A, D\rangle$. By definition, the constraint $avg(ABCD) \geq 30$ is convertible anti-monotone w.r.t. the order $\langle B, C, A, D\rangle$. Thus, it is convertible w.r.t. the order $\langle B, C, A, D\rangle$.

Definition 13. [15]. (Prefix monotone functions) Given an order O over a set of items I, a function f: $2^I \rightarrow R$ is a *prefix (monotonically) increasing function* w.r.t. O if and only if for every itemset S and its prefix S' w.r.t. O, $f(S') \leq f(S)$. A function g: $2^I \rightarrow R$ is a *prefix (monotonically) decreasing function* w.r.t. O if and only if for every itemset S and its prefix S' w.r.t. O, $g(S') \geq g(S)$.

Theorem 1. [15]. A constraint $u(S) \geq v$ (resp. $u(S) \leq v$) is convertible anti-monotone (resp., monotone) if and only if u is a prefix decreasing function. Similarly, $u(S) \geq v$ (resp. $u(S) \leq v$) is convertible monotone (resp., anti-monotone) if and only if u is a prefix increasing function.

Before defining an upper bound property for utility based measures, we first introduce some more terminology.

Definition 14. A *$k-$itemset*, denoted as S^k, is an itemset of k distinct items.

Definition 15. The set of all $(k-1)-$itemsets of a k-itemset S^k, denoted L^{k-1}, is the set $\{S^{k-1} \mid S^{k-1} \subset S^k\}$

For the 4-itemset $S^4 = ABCD$, by Definition 15, we have $L^3 = \{ACD, ABD, ABC, BCD\}$.

Definition 16. A *nonnegative utility function* f is a function $f(x, y) : (R, R) \rightarrow R^+$, where R is the set of real numbers, and R^+ is the set of nonnegative real numbers.

A function $f_1(x, y)$ with range $[-n, m]$, where $n, m \geq 0$, can be transformed to a nonnegative function by adding n to

Semantic Significance	**Utility Function** $f(x_{pq}, y_p)$	**Utility Value** $u(S)$
no semantic significance	$\sum_{i_p \in S} f(x_{pq}, y_p) = 1$	$u(S) = s(S)$
semantic significance on item	$\sum_{t_q \in T_S} f(x_{pq}, y_p) = s(S)$	$u(S) = \sum_{i_p \in S} f(i_p) \cdot s(S)$
semantic significance on transaction	$\sum_{i_p \in S} f(x_{pq}, y_p) = c$	$u(S) = c \cdot \sum_{t_q \in T_S} f(t_q)$
semantic significance on cell	$\sum_{i_p \in S} \sum_{t_q \in T_S} f(x_{pq}, y_p) \geq 0$	$u(S) = \sum_{i_p \in S} \sum_{t_q \in T_S} f(x_{pq}, y_p)$

Table 8: Semantic significance of utility function.

Measures	**Unified Utility Function** $f(x_{pq}, y_p)$
Support	$\sum_{i_p \in S} f(x_{pq}, y_p) = 1$
Weighted support	$\sum_{i_p \in S} f(x_{pq}, y_p) = w_p$
Normalized weighted support	$\sum_{i_p \in S} f(x_{pq}, y_p) = w_p/\|S\|$
Vertical weighted support	$\sum_{t_q \in T_S} f(x_{pq}, y_p) = w_q/c$
Mixed weighted support	$f(x_{pq}, y_p) = w_p \cdot w_q/c$
OOA Target and non-target attributes	$\sum_{i_p \in S} f(x_{pq}, y_p) = u_q(S)$
Count support	$f(x_{pq}, y_p) = w(i_p, t_q)/c$
Amount support	$f(x_{pq}, y_p) = w(i_p, t_q) \cdot w(i_p)/c$
Yao et al.'s	$f(x_{pq}, y_p) = w(i_p, t_q) \cdot w(i_p)$

Table 9: Utility based interestingness measures.

all values. Also a nonpositive function $f_2(x, y) \leq 0$ can be transformed to its absolute value, namely $|f_2(x, y)|$ such that $|f_2(x, y)| \geq 0$. Thus, all results obtained for nonnegative utility function can also be applied to function f_1 or f_2. Note that f could be a monotone, a non monotone, a convertible, or non convertible function.

Using Definitions 14-16, an upper bound on the utility value of the itemset S^k can be obtained as follows.

Theorem 2. (Utility Upper Bound Property) [20]. Let $u(S^k)$ be the utility value of a $k-$itemset S^k defined according to Equation(13) based on a nonnegative utility function f. Then the following property holds

$$u(S^k) \leq \frac{\sum_{S^{k-1} \in L^{k-1}} u(S^{k-1})}{k-1} \tag{14}$$

Example 4. For a 4-itemset $S^4 = ABCD$, by Definition 15, we obtain $L^3 = \{ACD, ABD, ABC, BCD\}$. Thus, by Theorem 2, we have

$$u(ABCD) \leq \frac{u(ABC) + u(ACD) + u(ABD) + u(BCD)}{3}.$$

It is important to realize that Theorem 2 indicates that the utility value of itemset S^k is limited by the utilities of all its subset of itemsets of size $(k-1)$.

By exploiting the anti-monotone (or monotone) property, the convertible property, and the upper bound property, efficient algorithms have been developed. More precisely, the Apriori algorithm [2] is based on the anti-monotone property. The FIC^A algorithm suggested by Pei et al. [16] is based on the convertible anti-monotone property, and the FIC^M algorithm suggested by Pei et al. [16] is based on the convertible monotone property. The UMining algorithm [20] is based on the upper bound property. All these algorithms reduced the number of the mined results by exploiting one of the properties of utility based measures.

Now, we consider the mathematical properties of the utility based measures discussed in Section 2. The Weighted Items approach [5] and the Value Added Mining approach [11] reflect the semantic significance of itemsets at the item level by defining different weights on items. Since there is always a decreasing order based on the weights of all items, a prefix monotone function can be defined as $\sum_{i_p \in S} f(i_p)$ for itemset S w.r.t. the descending order of the weights of the items, where $f(i_p)$ is the weight of the item i_p. The Vertical Weighted Support approach [13] and the OOA approach [6] capture the semantic significance of itemsets at the transaction level. Since there is always a decreasing order based on the weights of all transactions, a prefix monotone function is defined as $\sum_{t_q \in T_S} f(t_q)$ for itemset S w.r.t. the descending order on the weights of the transactions, where $f(t_q)$ is the weight of transaction t_q. Thus, the utility functions of the Weighted Items measure, the Value Added Mining, the Vertical Weighted Support measure, and the OOA measure satisfy the convertible property. The Mixed Weighted Support approach [13], the Itemset Share approach [4, 20] and Yao et al. [20] capture the semantic significance of itemsets

at the cell level. Since these three approaches use a nonnegative utility function, by Theorem 2, the utility function of the Mixed Weighted Support measures, the Itemset Share measures (*count_sup amount_sup*), and Yao et al. satisfy the upper bound property. Table 10 summaries the mathematical properties of the utility based measures discussed in Section 2.

Utility Measures	Mathematical Property
Support	anti-monotone property
Weighted support	convertible property
Normalized weighted support	convertible property
Vertical weighted support	convertible property
Mixed weighted support	upper bound property
OOA non-target attributes	convertible property
Count support	upper bound property
Amount support	upper bound property
Yao et al.'s	upper bound property

Table 10: Mathematical properties of utility based measures.

Theorem 3. The mathematical properties of utility measures shown in Table 10 are correct.

Proof: For the support measure, Agrawal et al. [2] showed that it satisfies the anti-monotone property. Now, we prove that Equations (1), (2), (3), and (5) satisfy the convertible property. For Equations(1) and (2), a prefix monotone function can be defined w.r.t. the descending order of the weights of the items. By Theorem 1, they satisfy the convertible property. Similarly, for Equations (3) and (5), a prefix monotone function can be defined w.r.t. the descending order of the weights of the transactions. By Theorem 1, they also satisfy the convertible property. Now we prove that Equations (4), (7), (8) and (9) satisfy upper bound property. For Equations (4), $w_p > 0$. For Equations(7), (8) and (9), $w(i_p, t_q) > 0$ and $w(i_p) > 0$. Thus, Equations (4), (7), (8) and (9) are nonnegative functions. By Theorem 2, they satisfy upper bound property. □

Theorem 3 indicates that we can design an efficient pruning strategy for these utility measures by using the identified mathematical properties. In other words, it is possible to incorporate these properties into the algorithms used for these utility measures.

5. CONCLUSIONS

This paper formalizes all existing utility measures for itemset mining that are known to the authors. We provide three research contributions towards utility based itemset mining in this paper.

First, we formalize the semantic significance of utility measures. Existing utility based measures employ various representations for the semantics significance of applications for the same dataset, which lead to different measures and procedures for determining interestingness. Based on the semantics of applications, we classified the utility based measures into three categories, namely, item level, transaction level, and cell level.

The second contribution is that we defined a unified utility function to represent all existing utility based measures, as shown in Table 9. According to our classification, the transaction utility and the external utility of an itemset is defined, and then a general unified framework was developed to define a unifying view of the utility based measures for itemset mining. That is, existing utility based measures can be represented by this unified utility function.

The third contribution is that the mathematical properties of the utility based measures were identified and analyzed. These properties can facilitate the design of efficient pruning strategies for utility based itemset mining and help current itemest algorithms to reflect the different utilities by using different pruning strategies.

Future research could consider a method for automating the elicitation of different itemset utilities, and then incorporating these different utilities into current itemset mining algorithms [2, 20]. In addition, to make our utility function more practicable, the unified utility function could be extend to a fuzzy utility function for fuzzy utility values.

6. REFERENCES

[1] Agrawal R., Imielinski T., and Swami, A.N. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., 1993, 207–216.

[2] Agrawal R. and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, 1994, 487–499.

[3] Bay, S.D. and Pazzani, M.J. Detecting change in categorical data: Mining contrast sets. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, San Diego, USA, 1999, 302-306.

[4] Barber, B. and Hamilton, H.J. Extracting share frequent itemsets with infrequent subsets. *Data Mining and Knowledge Discovery*, 7(2), 2003, 153–185.

[5] Cai, C.H., Fu, A.W.C., Cheng, C.H., and Kwong, W.W. Mining association rules with weighted items. In *Proceedings of the IEEE International Database Engineering and Applications Symposium*, Cardiff, UK, 1998, 68–77.

[6] Chan, R., Yang, Q., and Shen, Y.D. Mining high utility itemsets. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, Melbourne, Florida, 2003, 19-26.

[7] Hilderman, R.J., Carter, C.L., Hamilton, H.J., and Cercone, N. Mining market basket data using share measures and characterized itemsets. In *Proceedings of the Second Pacific Asia Conference on Knowledge Discovery in Databases*, Melbourne, 1998, 72-86.

[8] Hilderman, R.J. and Hamilton, H.J. Measuring the interestingness of discovered knowledge: A principled approach. *Intelligent Data Analysis*, 7(4), 2003, 347-382.

[9] Mannila, H., Toivonen, H., and Verkamo, A.I. Efficient algorithms for discovering association rules. In *Proceeding of the AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, 1994, 181-192.

[10] Geng, L. and Hamilton, H.J. Interestingness measures for data mining: A survey. *ACM Computing Surveys*. To appear.

[11] Lin, T.Y., Yao, Y.Y., and Louie, E. Value added association rules. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Taipei, Taiwan, 2002, 328-333.

[12] Liu, Y., Liao, W.K., and Choudhary, A. A Fast High Utility Itemsets Mining Algorithm. In *Proceedings of the Workshop on Utility-Based Data Mining in conjunction with the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005)*, Chicago, Illinois, 2005, 90-99.

[13] Lu, S., Hu, H., and Li, F. Mining weighted association rules. *Intelligent Data Analysis*, 5(3), 2001, 211–225.

[14] Ng, R., Lakshmanan, L. V. S., Han, J., and Pang, A. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, 1998, 13-24.

[15] Pei, J. and Han, J. Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, 2000, 350–354.

[16] Pei, J., Han, J., and Lakshmanan, L. V. S. Pushing convertible constraints in frequent itemset mining. *Data Mining and Knowledge Discovery*, 8(3), 2004, 227-252.

[17] Shen, Y. D., Zhang, Z. and Yang, Q. Objective-oriented utility-based association mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, Maebashi City, Japan, December 2002, 426-433.

[18] Silberschatz A. and Tuzhilin, A. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, Montreal, Quebec, Canada, 1996, 275–281.

[19] Tan, P.N., Kumar, V., and Srivastava, J. Selecting the right objective measure for association analysis. *Information Systems*, 29(4), 2004, 293-313.

[20] Yao H. and Hamilton, H.J. Mining itemset utilities from transaction databases, *Data & Knowledge Engineering*. To appear.

[21] Yao Y.Y. and Zhong N. An analysis of quantitative measures associated with rules, Methodologies for Knowledge Discovery and Data Mining. In *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Beijing, China, 1999, 26-28.

[22] Zhang, H., Padmanabhan, B., and Tuzhilin, A. On the discovery of significant statistical quantitative rules. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, Seattle, USA, August 2004, 374-383.

Assessing the Interestingness of Discovered Knowledge Using a Principled Objective Approach

Robert J. Hilderman
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
robert.hilderman@uregina.ca

ABSTRACT

When mining a large database, the number of patterns discovered can easily exceed the capabilities of a human user to identify interesting results. To address this problem, various techniques have been suggested to reduce and/or order the patterns prior to presenting them to the user. In this paper, our focus is on ranking summaries generated from a single dataset, where attributes can be generalized in many different ways and to many levels of granularity according to taxonomic hierarchies. We theoretically and empirically evaluate twelve diversity measures used as heuristic measures of interestingness for ranking summaries generated from databases. The twelve diversity measures have previously been utilized in various disciplines, such as information theory, statistics, ecology, and economics. We describe five principles that any measure must satisfy to be considered useful for ranking summaries. Theoretical results show that the proposed principles define a partial order on the ranked summaries in most cases, and in some cases, define a total order. Theoretical results also show that seven of the twelve diversity measures satisfy all of the five principles. We empirically analyze the rank order of the summaries as determined by each of the twelve measures. These empirical results show that the measures tend to rank the less complex summaries as most interesting. Finally, we demonstrate a technique, based upon our principles, for visualizing the relative interestingness of summaries.

Keywords: data mining, diversity measures, theory of interestingness, statistics and probability, visualization

1. INTRODUCTION

An important problem in the area of data mining is the development of effective measures of interestingness for ranking discovered knowledge. In this paper, we focus on the use of diversity measures as heuristic measures of interestingness for ranking summaries generated from a single dataset, where attributes can be generalized in many different ways and to many levels of granularity according to taxonomic hierarchies. With diversity measures, the problem that we are faced with is essentially one of ranking distributions of populations of objects having some distinguishable characteristics. The problem is common to many disciplines, such as species diversity in ecology, income/consumption inequality in economics, linguistic diversity in geography, market penetration in business, genetic differences in biology, and others. The common theme is that of classifying some quantity of objects into well-defined categories according to the aforementioned distinguishable characteristics.

The question that we ultimately ask when comparing two or more populations is whether one of the categorized populations is more or less diverse than another. And the question is similar, regardless of the discipline in which it is asked. For example, in ecology, we ask whether a sample of individuals from a particular habitat is more diverse than a sample taken from a neighboring or similar habitat. In economics, we ask whether a sample of individuals in a particular region has greater equality of income distribution than a sample of individuals in another region. And in linguistics, we ask whether the possibility for communication in a sample of individuals in a geographic region is more likely than in a sample of individuals from another geographic region. The above situations are all a specific case of the general problem that can be described, as follows. Suppose $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ are two populations of individuals, where x_i and y_j are integers representing the number of individuals classified into X_i and Y_j, respectively. Which of the distributions is more (less) diverse (or depending on the discipline, concentrated or uniform or monopolistic or specialized or dispersed)?

We introduced the use of diversity measures for ranking summaries, as described in the previous paragraph, in [26] and [27], where well-known diversity measures from information theory, statistics, ecology, and economics were proposed as heuristic measures of interestingness. Although diversity measures are frequently used in these various disciplines, their use for ranking the interestingness of summaries was a new application area. An empirical analysis found that highly ranked, concise summaries provided a reasonable starting point for further analysis of discovered knowledge. It was also shown that for selected sample datasets, the order in which some of the measures rank summaries is highly correlated, but the rank ordering can vary substantially when different measures are used. In [28], the

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

notion of a summary was extended to include other well-known forms of knowledge representation, and we showed that these other forms are also amenable to ranking using diversity measures.

We now study twelve diversity measures as heuristic measures of interestingness for ranking summaries in data mining applications, and propose five principles that any measure must satisfy to be considered useful for ranking the interestingness of summaries generated from databases. The five principles provide a foundation for an intuitive understanding of the term "interestingness" when used within this context. We perform a comparative sensitivity analysis of the twelve diversity measures to identify those that satisfy the proposed principles. Since each new measure represents an alternative definition of diversity, the choice of which measure to use may make a difference. That is, when choosing any objective candidate interestingness measure for ranking summaries, determine which of the five principles are satisfied, and then using this knowledge, judge the suitability of the candidate interestingness measure for the intended application. Essentially, this principled approach imposes a subjective bias on the objective measures by suggesting principles that objective measures should satisfy.

The remainder of the paper is organized as follows. In Section 2, we motivate the need for objective measures of interestingness in data mining systems, in general, and the need for principles of interestingness, in particular. In Section 3, we describe the twelve diversity measures empirically evaluated as measures of interestingness in this work. In Section 4, we present the foundation principles for a theory of interestingness for diversity measures used to rank summaries generated from a single dataset. In Section 5, we present experimental results from our evaluation of the twelve diversity measures. In Section 6, we demonstrate the application of the principles to the visualization of the relative interestingness of summaries. We conclude in Section 7 with a summary of our work and suggestions for future research.

2. MOTIVATION

In this section, we describe a data mining example where the task is description by summarization, the representation language is generalized relations, and the method for searching is the Multi-Attribute Generalization algorithm [31]. The problem is described, as follows. Let a *summary* S be a relation defined on the columns $\{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$, where each (A_i, D_i) is an attribute-domain pair. Also, let $\{(A_1, v_{i1}), (A_2, v_{i2}), \ldots, (A_n, v_{in})\}$, $i = 1, 2, \ldots, m$, be a set of m unique tuples, where each (A_j, v_{ij}) is an attribute-value pair and e ach v_{ij} is a value from the domain D_j associated with attribute A_j. One attribute A_k is a derived attribute, called *Count*, whose domain D_k is the set of positive integers, and whose value v_{ik} for each attribute-value pair (A_k, v_{ik}) is equal to the number of tuples which have been aggregated from the base relation (i.e., the unconditioned data present in the original database).

A summary, such as the one shown in Table 1, can be generated from a database, such as the one shown in Table 2, using *domain generalization graphs* (DGGs) [30, 33], such as the one shown in Figure 1. For example, the DGG in Figure 1 is associated with the *Office* attribute in the database of Table 2. In Figure 1, the domain for the *Office* attribute is represented by the *Office* node. Increasingly general descriptions of the domain values are represented by the *City*, *Division*, and *ANY* nodes. A user-defined taxonomy in the form of a table is associated with every arc between the nodes in the DGG and describes a generalization relation from one domain to another in a process called *attribute-oriented generalization* (AOG) [20] (other generalization relations besides table lookups are possible, but we restrict our discussion for the sake of simplicity and clarity). The table associated with the arc between the *Office* and *City* nodes defines the mapping of the domain values of the *Office* node to the domain values of the *City* node (e.g., 1 and 2 map to Vancouver, 3 and 4 map to Los Angeles, and 5 to 7 map to New York). The table associated with the arc between the *City* and *Division* nodes can be described similarly. The table associated with the arc between the *Division* and *ANY* nodes maps all values in the *Division* domain to the special value *ANY*. The summary in Table 1 corresponds to the *Division* node of the *Office* DGG, where the corresponding values in the *Quantity* and *Amount* attributes from Table 2 are also aggregated accordingly.

When there are DGGs associated with multiple attributes, then more complex summaries can be generated (known as *multi-attribute generalization*). For example, a DGG for the *Quantity* attribute is shown in Figure 2, where the generalization space consists of three nodes. The set of all possible combinations of domains from the DGGs associated with the *Office* and *Quantity* attributes defines the generalization space for the many summaries that can be generated from Table 2. Thus, the generalization space consists of the 12 nodes shown in Figure 3 (i.e., 4 nodes in the *Office* DGG ×

Table 1: A sample summary

Office	*Quantity*	*Amount*	*Count*
West	8	$200.00	4
East	11	$275.00	3

Table 2: A sales transaction database

Office	*Quantity*	*Amount*
2	2	$50.00
5	3	$75.00
3	1	$25.00
7	4	$100.00
1	3	$75.00
6	4	$100.00
4	2	$50.00

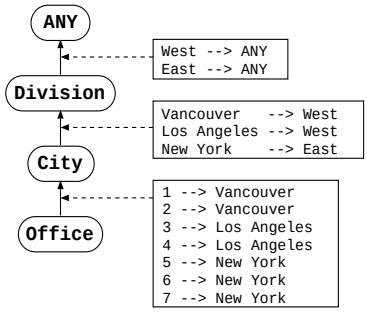


Figure 1: A DGG for the *Office* attribute

3 nodes in the *Quantity* DGG), and each node corresponds to a unique summary. For example, the *Division/Quantity* node corresponds to the summary generated by generalizing the *Office* attribute to the level of the *Division* node in the *Office* DGG, while the *Quantity* attribute remains ungeneralized (this summary is equivalent to the summary in Table 1). Similarly, the *City/Status* node corresponds to the summary shown in Table 3, and is generated by generalizing the *Office* and *Quantity* attributes to the level of the *City* and *Status* nodes, respectively. Naturally, the general technique is applicable to more than two attributes and should now be clear.

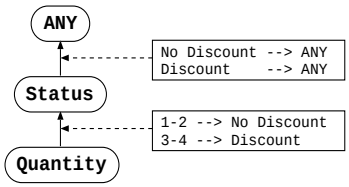


Figure 2: A DGG for the *Quantity* attribute

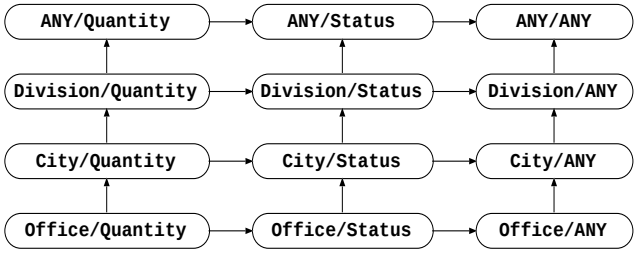


Figure 3: The generalization space defined by the *Office* and *Quantity* DGGs

Table 3: The *City/Status* summary

Office	*Quantity*	*Amount*	*Count*
Los Angeles	No Discount	$75.00	2
New York	Discount	$275.00	3
Vancouver	No Discount	$50.00	1
Vancouver	Discount	$75.00	1

Up to this point, we have only discussed summaries generated from databases using AOG and DGGs. However, alternative methods could be used to guide the generation of summaries. These include Galois lattices [17], conceptual graphs [7], or formal concept analysis [50]. Similarly, summaries could more generally include views generated from databases, characterized/generalized association rules generated from itemsets, or summary tables (i.e., data cubes) generated from data warehouses [28].

3. MEASURING INTERESTINGNESS

The tuples in a summary are unique, and therefore, can be considered to be a population with a structure that can be described by some frequency or probability distribution. Here, we review twelve diversity measures that consider the frequency or probability distribution of the values in the derived *Count* attribute (or some other similar numeric measure attribute) to assign a single real-valued index that represents its interestingness relative to other summaries.

3.1 Background

Diversity is an important concept that has seen extensive use in several different areas of research. However, although diversity is used in many disparate areas, it is widely claimed that diversity is a difficult concept to define [1, 2, 40, 44, 52]. The difficulty in defining diversity arises because it actually encompasses two separate components: the number of *classes* (also referred to in the literature as *richness*, *abundance*, or *density*) and the *proportional distribution* of the population among the classes (also referred to in the literature as *relative abundance*, *heterogeneity*, or *evenness*). Within the context of ranking the interestingness of a summary, the number of classes is simply the number of tuples in the summary; the proportional distribution is simply the actual probability distribution of the classes based upon the values contained in the derived *Count* attribute.

In a typical diversity measure, the two components are combined to characterize the variability of a population by a single value. This concept of a dual-component diversity measure was first introduced in [49]. The diversity measures considered to be most useful, and those most frequently referenced in the literature, are dual-component measures. Yet, despite the widespread acceptance and use of diversity measures, there is no single mathematical definition of diversity which has been widely accepted as the de facto standard and which has been shown to be superior to all others [1, 40, 44]. There is some general agreement, however, that a population is considered to have high diversity when it has many classes and the proportional distribution is fairly even. Similarly, a population is considered to have low diversity when it has few classes and the proportional distribution is uneven. Unfortunately, this leaves considerable room for ambiguity in measuring diversity because a population with few classes and a fairly even proportional distribution could have the same or nearly the same diversity as a population with many classes and an uneven proportional distribution.

Although there are some problems related to a precise and universally accepted definition for diversity, there are numerous research areas where the concept of diversity has been considered useful, such as ecology [1, 8, 9, 41, 42], economics [2, 4, 14, 47], genetics [36], linguistics [18, 37], business [5, 22, 23, 34], epidemiology [35], bibliometrics [45], software engineering [43], and the measurement of scientific productivity [2]. More general treatments attempt to define the concept of diversity and develop a related theory of diversity measurement [44, 52].

Here we apply twelve diversity measures to a new application area, that of ranking the interestingness of summaries generated from databases. They share three important properties. First, each measure depends only on the frequency or probability distribution of the values in the derived *Count* attribute of the summary to which it is being applied. Second, each measure allows a value to be generated with at most one pass through the summary. And third, each measure is independent of any specific units. Utilizing these heuristics for ranking the interestingness of summaries generated from databases is a natural and useful extension for these diversity measures into a new application domain.

3.2 Notation

The variables used to describe the diversity measures are now defined. Let m be the total number of tuples in a summary. Let n_i be the value contained in the derived *Count* attribute for tuple t_i. Let $N = \sum_{i=1}^{m} n_i$ be the total count. Let p be the actual probability distribution of the tuples based upon the values n_i. Let $p_i = n_i/N$ be the actual probability for tuple t_i. Let q be a uniform probability distribution of the tuples. Let $\bar{u} = N/m$ be the count for tuple t_i, $i = 1, 2, \ldots, m$ according to the uniform distribution q. Let $\bar{q} = 1/m$ be the probability for tuple t_i, for all $i = 1, 2, \ldots, m$ according to the uniform distribution q. Let r be the probability distribution obtained by combining the values n_i and $\bar{u}$. Let $r_i = (n_i + \bar{u})/2N$, be the probability for tuples t_i, for all $i = 1, 2, \ldots, m$ according to the distribution r. For example, given the sample summary shown in Table 4, we have $m = 4$, $n_1 = 3$, $n_2 = 2$, $n_3 = 1$, $n_4 = 1$, $N = 7$, $p_1 = 0.429$, $p_2 = 0.286$, $p_3 = 0.143$, $p_4 = 0.143$, $\bar{u} = 1.75$, $\bar{q} = 0.25$, $r_1 = 0.339$, $r_2 = 0.268$, $r_3 = 0.196$, and $r_4 = 0.196$.

Table 4: Another sample summary

Colour	*Shape*	*Count*
red	round	3
green	round	2
red	square	1
blue	square	1

3.3 Diversity Measures

We now describe the twelve diversity measures. Due to space limitations, examples are omitted. The interested reader is encouraged to work examples of each measure based upon the sample summary shown in Table 4.

$\mathbf{I_{Variance}}$: Based upon sample variance from classical statistics, measures the weighted average of the squared deviations of the probabilities p_i from the mean probability $\bar{q}$, where the weight assigned to each squared deviation is $1/(m-1)$. We use sample variance because we assume the summary may not contain all possible combinations of attribute values, meaning we are not observing all of the possible tuples. The *sample variance* is given by

$$I_{Variance} = \frac{\sum_{i=1}^{m}(p_i - \bar{q})^2}{m-1}.$$

$\mathbf{I_{Simpson}}$: A variance-like measure based upon the Simpson index [49], measures the extent to which the counts are distributed over the tuples in a summary, rather than being concentrated in any single one of them. The *concentration* is given by

$$I_{Simpson} = \sum_{i=1}^{m} p_i^2.$$

Let each tuple i be represented by a "commonness value" (i.e., the probability of occurrence p_i). If an individual is drawn at random from the population, the probability that it will belong to tuple i is p_i, and if it does, its commonness value is also p_i. Thus, the expected commonness values for tuple i is p_i^2, and for all tuples $i = 1, \ldots, n$ is $\sum_1^m p_i^2$. Equivalently, this can be viewed as the average commonness value that would be obtained if the experiment of drawing an individual at random were repeated many times.

$\mathbf{I_{Shannon}}$: Based upon a relative entropy measure from information theory (known as the *Shannon index*) [48], measures the average information content in the tuples of a summary. The *average information content*, in bits per tuple, is given by

$$I_{Shannon} = -\sum_{i=1}^{m} p_i \log_2 p_i.$$

Say there are n_i individuals summarized in a tuple i, out of a possible N individuals. The probability of drawing one of the individuals in tuple i is n_i/N, or p_i. The information conveyed by announcing the result of drawing a particular individual in tuple i is $-\log_2 p_i$. The total contribution of these n_i individuals to the overall average information conveyed by announcing the result is $-p_i \log_2 p_i$. Summation over all such cases for all possible individuals is given by $-\sum_{i=1}^{m} p_i \log_2 p_i$.

$\mathbf{I_{McIntosh}}$: Based upon a heterogeneity index from ecology [41], views the counts in a summary as the coordinates of a point in a multidimensional space and measures the modified Euclidean distance from this point to the origin. The *modified Euclidean distance* is given by

$$I_{McIntosh} = \frac{N - \sqrt{\sum_{i=1}^{m} n_i^2}}{N - \sqrt{N}}.$$

The value $\sqrt{\sum_{i=1}^{m} n_i^2}$ is just the Pythagorean Theorem. Since $\sqrt{\sum_{i=1}^{m} n_i^2}$ is a measure of concentration, the N-complement $N - \sqrt{\sum_{i=1}^{m} n_i^2}$ is a measure of diversity. The value $N - \sqrt{N}$ makes it a diversity measure independent of N. The greater the count in a particular class, the further that class will be from the origin. If the count is reduced, or the count is spread more evenly between class, the distance from the origin will be reduced. $I_{McIntosh}$ relates the distance between a class and the origin to the range of possible values as determined by the number of tuples in the original relation.

$\mathbf{I_{Lorenz}}$: Based upon the Lorenz curve from statistics, economics, and social science [51], measures the average value of the Lorenz curve derived from the probabilities p_i associated with the tuples in a summary. The *average value of the Lorenz curve* is given by

$$I_{Lorenz} = \bar{q} \sum_{i=1}^{m} (m - i + 1) p_i.$$

The Lorenz curve is a series of straight lines in a square of unit length, starting from the origin and going successively to points (p_1, q_1), $(p_1 + p_2, q_1 + q_2)$, ... When the p_i's are all equal, the Lorenz curve coincides with the diagonal that cuts the unit square into equal halves. When the p_i's are not all equal, the Lorenz curve is below the diagonal.

$\mathbf{I_{Gini}}$: Based upon the Gini coefficient [51], which is itself defined in terms of the Lorenz curve, measures the ratio of the area between the diagonal (i.e., the line of equality) and the Lorenz curve, and the total area below the diagonal. The *Gini coefficient* is given by

$$I_{Gini} = \frac{\bar{q} \sum_{i=1}^{m} \sum_{j=1}^{m} |p_i - p_j|}{2}.$$

I_{Berger}: Based upon a dominance index from ecology [6], measures the proportional dominance of the tuple in a summary with the highest probability p_i. The *proportional dominance* is given by

$$I_{Berger} = \max(p_i).$$

Say a sample of individuals is taken from some population of species in a particular habitat. The number of individuals taken from each species is assumed to represent the proportional distribution of species in the actual population. I_{Berger} is called a dominance index because the index of diversity that it assigns to the sampled population is simply the proportional distribution of the most dominant species (i.e., the species with the highest proportional distribution).

I_{Schutz}: Based upon an inequality measure from economics and social science [47], measures the relative mean deviation of the actual distribution of the counts in a summary from a uniform distribution of the counts. The *relative mean deviation* is given by

$$I_{Schutz} = \frac{\sum_{i=1}^{m} |p_i - \bar{q}|}{2}.$$

I_{Bray}: Based upon community similarity indices from ecology [8], measures the percentage of similarity between the actual distribution of the counts in a summary and a uniform distribution of the counts. The *percentage of similarity* is given by

$$I_{Bray} = \frac{\sum_{i=1}^{m} \min(n_i, \bar{u})}{N}.$$

$I_{MacArthur}$: Based upon the Shannon index from information theory [39], combines two summaries and then measures the difference between the amount of information contained in the combined distribution and the amount contained in the average of the two original distributions. The *difference*, in bits, is given by

$$I_{MacArthur} = \left(-\sum_{i=1}^{m} r_i \log_2 r_i \right) - \left(\frac{(-\sum_{i=1}^{m} p_i \log_2 p_i) + \log_2 m}{2} \right).$$

I_{Theil}: Based upon a distance measure from information theory [51], measures the distance between the actual distribution of the counts in a summary and a uniform distribution of the counts. The *distance*, in bits, is given by

$$I_{Theil} = \frac{\sum_{i=1}^{m} |p_i \log_2 p_i - \bar{q} \log_2 \bar{q}|}{m\bar{q}}.$$

$I_{Atkinson}$: Based upon a measure of inequality from economics [4], measures the percentage to which the population in a summary would have to be increased to achieve the same level of interestingness if the counts in the summary were uniformly distributed. The *percentage increase* is given by

$$I_{Atkinson} = 1 - \left(\prod_{i=1}^{m} \frac{p_i}{\bar{q}} \right)^{\bar{q}}.$$

Lower values of $I_{Atkinson}$ mean that the distribution of counts in a summary are fairly equal, or near uniform. Higher values mean the distribution is fairly uneven. As an example, say $I_{Atkinson} = 0.105$, as shown in the example below. This value means that if the counts of the tuples were uniformly distributed, then we would need only approximately 90% of the current total count to realize the same level of interestingness.

4. PRINCIPLES OF INTERESTINGNESS

We now describe a theory of interestingness against which the utility of candidate interestingness measures can be assessed. We do this through the mathematical formulation of five principles that we believe must be satisfied by any acceptable diversity measure for ranking the interestingness of summaries generated from databases using our, or a similar, technique. Through the development of these five principles, we have established some basic criteria for the measurement of interestingness within this context which provide the basis for a theoretical foundation in identifying appropriate diversity measures for ranking summaries.

Through the mathematical formulation of the five principles, we study functions f of m variables, $f(n_1, \ldots, n_m)$, where f denotes a general measure of diversity, m and each n_i are as defined in the previous section, and $(n_1, \ldots, n_m)$ is a vector corresponding to the values in the derived *Count* attribute (or numeric measure attribute) for some arbitrary summary whose values are arranged in descending order such that $n_1 \geq \ldots \geq n_m$ (except for discussions regarding I_{Lorenz}, which requires that the values be arranged in ascending order). Since the principles presented here are for ranking the interestingness of summaries generated from a single dataset, we assume that N is fixed. We begin by specifying two fundamental principles.

Minimum Value Principle (P1). Given a vector $(n_1, \ldots, n_m)$, where $n_i = n_j$, for all i, j, $f(n_1, \ldots, n_m)$ attains its minimum value.

P1 specifies that the minimum interestingness should be attained when the tuple counts are all equal (i.e., uniformly distributed). For example, given the vectors $(2, 2)$, $(50, 50, 50)$, and $(1000, 1000, 1000, 1000)$, we require that the index value generated by f be the minimum possible for the respective values of m and N.

Maximum Value Principle (P2). Given a vector $(n_1, \ldots, n_m)$, where $n_1 = N - m + 1$, $n_i = 1$, $i = 2, \ldots, m$, and $N > m$, $f(n_1, \ldots, n_m)$ attains its maximum value.

P2 specifies that the maximum interestingness should be attained when the tuple counts are distributed as unevenly as possible. For example, given the vectors $(3, 1)$, $(148, 1, 1)$, and $(3997, 1, 1, 1)$, where $m = 2, 3,$ and 4, respectively, and $N = 4, 150,$ and 4000, respectively, we require that the index value generated by f be the maximum possible for the respective values of m and N.

The behaviour of a measure relative to satisfying both *P1* and *P2* is significant because it reveals an important characteristic about its fundamental nature as a measure of diversity. A measure of diversity can generally be considered either a *measure of concentration* or a *measure of dispersion*. A measure of concentration can be viewed as the opposite of a measure of dispersion, and we can convert one to the other via simple transformations. For example, if g corre-

sponds to a measure of dispersion, then we can convert it to a measure of concentration f, where $f = \max(g) - g$. Here we only consider measures of concentration. A measure was considered to be a measure of concentration if it satisfied P1 and P2 without transformation. A measure was considered to be a measure of dispersion if it satisfied P1 and P2 following transformation. All measures of dispersion were transformed into measures of concentration prior to our analysis.

Permutation Invariance Principle (P3). Given a vector $(n_1, \ldots, n_m)$ and any permutation $(i_1, \ldots, i_m)$ of $(1, \ldots, m)$, $f(n_1, \ldots, n_m) = f(n_{i_1}, \ldots, n_{i_m})$.

P3 specifies that every permutation of a given distribution of tuple counts should be equally interesting. That is, interestingness is not a labeled property, it is only determined by the distribution of the counts. For example, given the vector $(2, 4, 6)$, we require that $f(2, 4, 6) = f(4, 2, 6) = f(4, 6, 2) = f(2, 6, 4) = f(6, 2, 4) = f(6, 4, 2)$.

Transfer Principle (P4). Given a vector $(n_1, \ldots, n_m)$, $n_i \geq n_j$, $i < j$, and $0 < c \leq n_j$, $f(n_1, \ldots, n_i + c, \ldots, n_j - c, \ldots, n_m) > f(n_1, \ldots, n_i, \ldots, n_j, \ldots, n_m)$.

P4, adopted from [14], specifies that when a strictly positive transfer is made from the count of one tuple to another tuple whose count is greater, then interestingness increases. For example, given the vectors $(10, 7, 5, 4)$ and $(10, 9, 5, 2)$, we require that $f(10, 9, 5, 2) > f(10, 7, 5, 4)$.

Majorization Principle (P5). Given vectors $(n_1, \ldots, n_m)$ and $(n_1^{'}, \ldots, n_m^{'})$, whenever $f(n_1^{'}, \ldots, n_m^{'}) > f(n_1, \ldots, n_m)$, then $(n_1^{'}, \ldots, n_m^{'}) \succ (n_1, \ldots, n_m)$, read $(n_1^{'}, \ldots, n_m^{'})$ *majorizes* $(n_1, \ldots, n_m)$.

The majorization operator, $\succ$, is based upon the Lorenz dominance order. The *Lorenze dominance order* [21] compares vectors with different distributions and says for any two vectors $(n_1, \ldots, n_m)$ and $(n_1^{'}, \ldots, n_m^{'})$, that $(n_1^{'}, \ldots, n_m^{'}) \succ (n_1, \ldots, n_m)$ if the following fours conditions are true:

1. $n_1 \geq \ldots \geq n_m$.
2. $n_1^{'} \geq \ldots \geq n_m^{'}$.
3. $\sum_{i=1}^{j} n_i^{'} \geq \sum_{i=1}^{j} n_i$, for every $j = 1, \ldots, m$.
4. $\sum_{i=1}^{m} n_i^{'} = \sum_{i=1}^{m} n_i$.

An important property of the Lorenz dominance order is that it defines a partial order on the set of all possible vectors, a property useful and important for ranking summaries.

Those measures that satisfy the principles of interestingness are shown in Table 5. In Table 5, the *P1* to *P5* columns describe the proposed principles, and a measure that satisfies a principle is indicated by the *bullet* symbol (i.e., •).

Mathematical proofs were derived for each measure satisfying principles P1 to P5 in Table 5. However, due to space limitations, the proofs are omitted. The interested reader is referred to [29] for the complete proofs.

Table 5: Measures satisfying the proposed principles

Measure	*P1*	*P2*	*P3*	*P4*	*P5*
$I_{Variance}$	•	•	•	•	•
$I_{Simpson}$	•	•	•	•	•
$I_{Shannon}$	•	•	•	•	•
$I_{McIntosh}$	•	•	•	•	•
I_{Lorenz}	•	•		•	•
I_{Gini}	•	•	•	•	•
I_{Berger}	•	•	•		
I_{Schutz}	•	•	•		
I_{Bray}	•	•	•		
$I_{MacArthur}$	•	•	•	•	•
I_{Theil}	•		•		
$I_{Atkinson}$	•	•	•	•	•

5. EXPERIMENTAL RESULTS

A series of experiments were run using *DGG-Interest*, an extension to *DB-Discover*, a research data mining tool developed at the University of Regina [12]. *DB-Discover* generates summaries from databases according to DGGs associated with attributes and the AOG technique described in Section 2. *DGG-Interest* evaluates and ranks the summaries generated using the twelve diversity measures described in Section 3.

Input data for the experiments was supplied by the NSERC Research Awards database, freely available in the public domain, and the Customer Accounts database, a confidential database provided by a commercial research partner in the telecommunications industry. The NSERC Research Awards database contains records of Canadian government funding provided to academic and industrial researchers in the natural sciences and engineering, and has been frequently used in previous data mining research [10, 11, 19, 38]. It consists of 10,000 tuples in six tables describing a total of 22 attributes. The Customer Accounts database has also been frequently used in previous data mining research [13, 24, 32, 25]. It consists of over 8,000,000 tuples in 22 tables describing a total of 56 attributes. The largest table contains over 3,300,000 tuples representing the account activity for over 500,000 customer accounts and over 2,200 products and services. In the discovery tasks run against the NSERC database, from two to four attributes were selected for discovery, and in those run against the Customer Accounts database, from two to five attributes were selected. We refer to the NSERC discovery tasks containing two, three, and four attributes as as *N-2*, *N-3*, and *N-4*, respectively, and the Customer Accounts discovery tasks containing two, three, four, and five attributes as *C-2*, *C-3*, *C-4*, and *C-5*, respectively.

Within the context of discovery tasks that generate summaries, the discovery tasks run against the Customer Accounts database are considered large. For example, the characteristics of the DGG's associated with each attribute are shown in Table 6. In Table 6, the *No. of Paths* column describes the number of unique paths through the DGG, the *No. of Nodes* column describes the number of nodes in the DGG, and the *Avg. Path Length* describes the average path length of the unique paths. The number of summaries to be generated by a discovery task (i.e., the size of the generalization space) is determined by multiplying the values in the *No. of Nodes* column. For example, *C-5* selected attributes *C*, *D*, *E*, *F*, and *G*, generating a generalization space containing 102,816 nodes (i.e., $12 \times 17 \times 8 \times 3 \times 21$). Many of these nodes correspond to summaries that are duplicates (i.e., the count vectors are identical). Dulplicates can either occur by chance, or when the generalization of an attribute to a higher node in the associated DGG does not result in any tuples being aggregated, and this can occur quite frequently.

Since the diversity measures used to rank the vectors cannot differentiate these summaries, they are considered to be of equal interest. Consequently, the number of summaries (i.e., count vectors) actually ranked is considerably less in practice. For example, of the 102,816 summaries generated by *C-5*, there were only 493 unique vectors, but the entire generalization space still needs to be traversed to find them.

Table 6: Characteristics of the DGGs associated with the selected attributes

Attribute	*No. of Paths*	*No. of Nodes*	*Avg. Path Length*
A	5	20	4.0
B	4	17	4.3
C	3	12	4.0
D	4	17	4.3
E	2	8	4.0
F	1	3	3.0
G	5	21	4.2

We now discuss the complexity of the summaries ranked by the various measures. In analyzing a summary, whether it be a two-dimensional spreadsheet or a multi-dimensional data cube, one metric that determines how easily the information it contains may be to understand by a domain expert, is simply its physical size in terms of the number of cells (i.e., where a cell is commonly understood to be the piece of information referenced by a unique combination of labels corresponding to the data items associated with each dimension). So, for this analysis, we define the *complexity* of a summary as simply the product of the number of tuples and the number of non-ANY attributes contained in the summary. One could ask, then, why use diversity measures to rank summaries at all if complexity, as defined, is a suitable metric for comparing summaries? That is, why not simply rank the least complex summaries as most interesting? The answer to this, of course, is that complexity ignores both the number of tuples in a summary and the the proportional distribution of the tuples, while diversity measures do not. However, complexity is still a useful measure because it is easy to understand at an intuitive level, and a good indicator of the amount of information contained in a summary.

Now, in a previous study, domain experts suggested that more information is better than less, provided that the most interesting summaries are not too complex and remain relatively easy to understand [19]. This implies that useful summaries are those that are complex enough to inform, yet not so complex as to overwhelm. That is, the knowledge contained in a summary should be non-trivial, yet understandable with reasonable effort by the domain expert. Consequently, we believe a desirable property of any ranking function be that it tend to rank summaries with low complexity as more interesting. However, although we want to rank summaries with low complexity as more interesting, we do not want to lose the meaning or context of the data by presenting summaries that are either too complex or too simple.

In this experiment, we analyze the measures and evaluate whether they satisfy the complexity guidelines of our domain experts. The relative complexity of summaries ranked by each measure when grouped according to a three-tier scale of relative complexity (i.e., H=High, M=Moderate, L=Low). High, moderate, and low complexity summaries were considered to be the top, middle, and bottom 20%, respectively, of summaries as ranked by each measure. The *N-2*, *N-3*, and *N-4* discovery tasks generated sets containing 22, 70, and 214 summaries, respectively, while the *C-2*, *C-3*, *C-4*, and *C-5* discovery tasks generated sets containing 43, 91, 155, and 493 summaries, respectively. Thus, the complexity of the summaries from the *N-2*, *N-3*, and *N-4* discovery tasks is based upon the top four, 14, and 43 summaries, respectively, while the complexity of the summaries from the *C-2*, *C-3*, *C-4*, and *C-5* discovery tasks is based upon nine, 18, 31, and 97 summaries, respectively.

A graphical comparison of the complexity of the summaries ranked by the twelve measures from the *N-2*, *N-3*, and *N-4* discovery tasks and the *C-2*, *C-3*, *C-4*, and *C-5* discovery tasks is shown in the graphs of Figures 4 and 5, respectively. In Figures 4 and 5, the horizontal and vertical axes describe the measures and the complexity, respectively. Each horizontal row of bars corresponds to the complexity of the most interesting summaries from a particular discovery task. The backmost horizontal row of bars corresponds to the average complexity for a particular measure. Both figures show a maximum complexity on the vertical axes of 60.0, although the complexity of the most interesting summaries ranked by I_{Lorenz}, I_{Schutz}, I_{Bray}, $I_{MacArthur}$, and $I_{Atkinson}$ in *N-4* exceed this value (i.e., 133.6, 289.8, 289.8, 249.5, and 531.1, respectively). When the measures are ordered by complexity, from lowest to highest, they are ordered according to Figure 4, as follows (position in the ordering is shown in parentheses): I_{Max} (1), I_{Total} (2), I_{Gini} (3), $I_{Shannon}$ and $I_{Kullback}$ (4), I_{Theil} (5), $I_{Variance}$ (6), $I_{Simpson}$ and $I_{McIntosh}$ (7), I_{Berger} (8), I_{Lorenz} (9), $I_{MacArthur}$ (10), I_{Schutz} and I_{Bray} (11), and $I_{Atkinson}$ (12). They are ordered according to Figure 5, as follows: I_{Total} (1), I_{Max} (2), I_{Berger} (3), $I_{Variance}$, $I_{Simpson}$, $I_{Shannon}$, $I_{McIntosh}$, and I_{Gini} (4), $I_{Kullback}$ (5), I_{Lorenz} (6), $I_{MacArthur}$ (7), $I_{Atkinson}$ (8), I_{Schutz} and I_{Bray} (9).

6. VISUALIZING INTERESTINGNESS

We now demonstrate the application of the five principles of Section 4 to the ranking of summaries. Here we use the results generated by the *N-3* discovery task described in Section 5 as the basis for the extended example as these are representative of the results obtained for all discovery tasks.

An important implication of P5 is that if $X \succ Y$, then all measures satisfying P5 will order the vectors X and Y in the same way. However, it is important to note that even when two measures order the vectors X and Y in the same way, they may not agree on the extent to which X is more concentrated than Y due to the differing range and distribution of the possible values, as described in Section 5. Consequently, the results we discuss here are valid for all of $I_{Variance}$, $I_{Simpson}$, $I_{Shannon}$, $I_{McIntosh}$, I_{Gini}, $I_{MacArthur}$, and $I_{Atkinson}$.

For this example, we used an extension of *DGG-Interest* to prune the number of summaries generated by *N-3* from 70 down to 27. This extension of *DGG-Interest* utilizes the chi-squared test for independence to consider only those summaries in which the attributes are associated. These summaries and their Lorenz dominance order are shown in Table 7. In Table 7, the *ID* column describes the unique identifiers associated with each of the 27 summaries, the numbered columns describe those summaries that are majorized by the

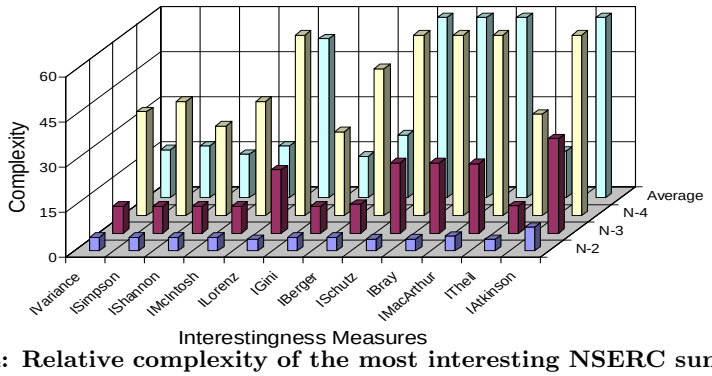


Figure 4: Relative complexity of the most interesting NSERC summaries

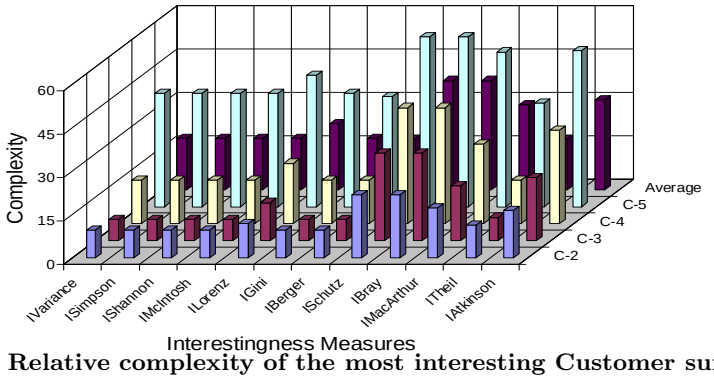


Figure 5: Relative complexity of the most interesting Customer summaries

corresponding summary in the *ID* column, and a summary that is majorized is indicated by the *bullet* symbol (i.e., •). Summaries whose count vectors were identical (i.e., identical number of tuples and identical probability distributions) are grouped together and treated as a single summary for this analysis (because if vector $X = Y$, then $X \succ Y$ and $Y \succ X$, so the vectors are indistinguishable according to the Lorenz dominance order). For example, in the second row, it is shown that summary 8 majorizes 11, 12, 33, 34, 80, 83, and 84 (equivalently $8 \succ \{11, 12, 33, 34, 80, 83, 84\}$). Since we consider majorization to be equivalent to interestingness, then essentially we consider summary 8 to be more interesting than 11, 12, 33, 34, 80, 83, and 84. Summaries 33, 34, and 84 are examples of summaries that do not majorize any other summaries.

Taking advantage of the transitive property of the Lorenz dominance order, we can discover all of the majorization relationships described in Table 7. For example, consider summary 7 in the first row. We see that $7 \succ 8$. Moving to the row beginning with summary 8, we see that $8 \succ 11$. Moving to the row beginning with summary 11, we see that $11 \succ 12$. Moving to the row beginning with summary 12, we see that $12 \succ 84$. Moving to the row beginning with summary 84, we see that 84 does not majorize any other summary. Thus, we can summarize the discovered relationship as the partial order $7 \succ 8 \succ 11 \succ 12 \succ 84$. Note that although we know from the first row that $7 \succ \{8, 11, 12, 84\}$, the first row does not tell us anything about the relationships between 8, 11, 12, and 84. We had to examine the rows corresponding to 8, 11, 12, and 84 to discover these relationships.

Table 7 actually describes 33332 possible partial orders. Using another extension to *DGG-Interest*, 96 rules were generated for consolidating these partial orders into the concise graph of Figure 6. In Figure 6, the majorization relationship of the 27 summaries can be easily determined. The shaded nodes with a bold border indicate summaries that are not majorized by any others, and are start points for traversing the graph. For example, starting at node 17/18, we can follow a path that includes nodes 7, 21/22, 11, and 33/34. Node 33/34 is a shaded node without a bold border, and indicates a stop point (i.e., 33/34 majorizes no other summaries). Similarly, starting at node 17/18, we can follow a path that includes 16, 79, 8, 80, and 33/34. Note that while summary 17/18 majorizes both summaries 7 and 16, there is no path between 16 and 7, so we cannot say anything definitive about the relative interestingness of these two summaries. However, we do know that 17/18 is more interesting than both 16 and 7.

7. CONCLUSION

The use of diversity measures for ranking the interestingness of summaries generated from databases is a new application area. Here we described twelve diversity measures used as heuristic measures of interestingness, and proposed five principles that diversity measures must satisfy to be considered useful for ranking summaries generated from a single dataset. Theoretical results show that seven measures satisfy all of the proposed principles. These include $I_{Variance}$, $I_{Simpson}$, $I_{Shannon}$, $I_{McIntosh}$, I_{Gini}, $I_{MacArthur}$, and $I_{Atkinson}$. The five remaining measures did not perform as well, failing to satisfy at least one of the proposed prin-

Table 7: Summaries and their Lorenz dominance order

ID	7	8	11	12	16	17/18	21/22	27	28	29/30	33/34	52	53/54	57/58	79	80	83	84	99	100	123
7		•	•	•			•		•	•	•			•	•	•	•	•	•	•	
8			•	•							•					•	•	•			
11				•							•						•	•			
12																		•			
16		•	•	•			•		•	•	•	•		•	•	•	•	•	•	•	
17/18	•	•	•	•	•		•		•	•	•	•	•	•	•	•	•	•	•	•	
21/22			•	•							•			•			•	•			
27		•	•	•					•	•	•			•		•	•	•	•	•	
28				•							•							•		•	
29/30				•					•		•						•	•		•	
33/34																					
52		•	•	•					•	•	•			•		•	•	•	•	•	
53/54		•	•	•			•		•	•	•	•		•	•	•	•	•	•	•	
57/58				•							•						•	•			
79		•	•	•							•					•	•	•			
80				•							•						•	•			
83				•														•			
84																					
99				•					•		•						•	•		•	
100											•							•			
123		•	•	•			•		•	•	•	•		•	•	•	•	•	•	•	

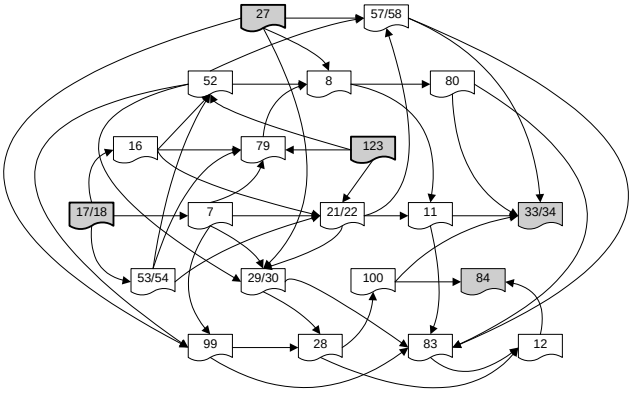


Figure 6: A graph summarizing the Lorenz dominance order

ciples. Experimental results showed that the partial order described by the Lorenze dominance order can be used to generate a graph summarizing the relative interestingness of summaries.

Considerable research remains to be done in the application of diversity measures to the problem of ranking the interestingness of summaries generated from databases. We see two major areas for future research. First, other diversity measures need to be evaluated to determine their suitability for ranking the interestingness of summaries generated from databases. There is certainly no shortage of possible candidates in the literature [45, 46, 16, 3, 15, 15]. And finally, principles of interestingness for comparing summaries generated from different databases need to be developed and evaluated.

8. REFERENCES

[1] R.V. Alatalo. Problems in the measurement of evenness in ecology. *Oikos*, 37(2):199–204, 1981.

[2] P.D. Allison. Measures of inequality. *American Sociological Review*, 43:865–880, 1978.

[3] P.D. Allison. Inequality and scientific productivity. *Social Studies of Science*, 10:163–179, 1980.

[4] A.B. Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2:244–263, 1970.

[5] M. Attaran and M. Zwick. An information theory approach to measuring industrial diversification. *Journal of Economic Studies*, 16:19–30, 1989.

[6] W.H. Berger and F.L. Parker. Diversity of planktonic forminifera in deep-sea sediments. *Science*, 168:1345–1347, 1970.

[7] I. Bournaud and J.-G. Ganascia. Accounting for domain knowledge in the construction of a generalization space. In *Proceedings of the Third International Conference on Conceptual Structures*, pages 446–459. Springer-Verlag, August 1997.

[8] J.R. Bray and J.T. Curtis. An ordination of the upland forest communities of southern Wisconsin. *Ecological Monographs*, 27:325–349, 1957.

[9] L. Bulla. An index of evenness and its associated diversity measure. *Oikos*, 70(1):167–171, 1994.

[10] C.L. Carter and H.J. Hamilton. Fast, incremental generalization and regeneralization for knowledge discovery from databases. In *Proceedings of the 8th Florida Artificial Intelligence Symposium*, pages 319–323, Melbourne, Florida, April 1995.

[11] C.L. Carter and H.J. Hamilton. Performance evaluation of attribute-oriented algorithms for knowledge discovery from databases. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'95)*, pages 486–489, Washington, D.C., November 1995.

[12] C.L. Carter and H.J. Hamilton. Efficient attribute-oriented algorithms for knowledge discovery from large databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):193–208, March/April 1998.

[13] C.L. Carter, H.J. Hamilton, and N. Cercone. Share-based measures for itemsets. In J. Komorowski and J. Zytkow, editors, *Proceedings of the First European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD'97)*, pages 14–24, Trondheim, Norway, June 1997.

[14] H. Dalton. The measurement of the inequality of incomes. *Economic Journal*, 30:348–361, 1920.

[15] L. Egghe and R. Rousseau. Transfer principles and a classification of concentration measures. *Journal of the American Society for Information Science*, 42(7):479–489, 1991.

[16] J. Gaston. *The reward system in British and American science*. Wiley and Sons, 1978.

[17] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[18] J.H. Greenberg. The measurement of linguistic diveristy. *Language*, 32:109–115, 1956.

[19] H.J. Hamilton and D.F. Fudger. Estimating DBLearn's potential for knowledge discovery in databases.

Computational Intelligence, 11(2):280–296, 1995.

[20] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29–40, February 1993.

[21] G. Hardy, J.E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1952.

[22] P.E. Hart. Entropy and other measures of concentration. *Journal of the Royal Statistical Society, Series A*, 134:73–85, 1971.

[23] J.L. Hexter and J.W. Snow. An entropy measure of relative aggregate concentration. *Southern Economic Journal*, 36:239–243, 1970.

[24] R.J. Hilderman, C.L. Carter, H.J. Hamilton, and N. Cercone. Mining association rules from market basket data using share measures and characterized itemsets. *International Journal on Artificial Intelligence Tools*, 7(2):189–220, June 1998.

[25] R.J. Hilderman, C.L. Carter, H.J. Hamilton, and N. Cercone. Mining market basket data using share measures and characterized itemsets. In X. Wu, R. Kotagiri, and K. Korb, editors, *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 159–173, Melbourne, Australia, April 1998.

[26] R.J. Hilderman and H.J. Hamilton. Heuristic measures of interestingness. In J. Zytkow and J. Rauch, editors, *Proceedings of the Third European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 232–241, Prague, Czech Republic, September 1999.

[27] R.J. Hilderman and H.J. Hamilton. Heuristics for ranking the interestingness of discovered knowledge. In N. Zhong and L. Zhou, editors, *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'99)*, pages 204–209, Beijing, China, April 1999.

[28] R.J. Hilderman and H.J. Hamilton. Applying objective interestingness measures in data mining systems. In D.A. Zighed and J. Komorowski, editors, *Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, pages 432–439, Lyon, France, September 2000.

[29] R.J. Hilderman and H.J. Hamilton. Principles for mining summaries: Theorems and proofs. Technical Report CS 00-01, Department of Computer Science, University of Regina, February 2000. Online at `www.cs.uregina.ca/research/Techreport/0001.ps`.

[30] R.J. Hilderman and H.J. Hamilton. *Knowledge Discovery and Measures of Interest*. Kluwer Academic Publishers, 2001.

[31] R.J. Hilderman, H.J. Hamilton, and N. Cercone. Data mining in large databases using domain generalization graphs. *Journal of Intelligent Information Systems*, 13(3):195–234, November 1999.

[32] R.J. Hilderman, H.J. Hamilton, R.J. Kowalchuk, and N. Cercone. Parallel knowledge discovery using domain generalization graphs. In J. Komorowski and J. Zytkow, editors, *Proceedings of the First European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD'97)*, pages 25–35, Trondheim, Norway, June 1997.

[33] R.J. Hilderman, Liangchun Li, and H.J. Hamilton. Visualizing data mining results with domain generalization graphs. In U. Fayyad, G.G. Grinstein, and A. Wierse, editors, *Information Visualization in Data Mining and Knowledge Discovery*, pages 251–270. Morgan Kaufmann Publishers, 2002.

[34] A. Horowitz and I. Horowitz. Entropy Markov processes and competition in the brewing industry. *Journal of Industrial Economics*, 16:196–211, 1968.

[35] J. Iszak. Sensitivity profiles of diveristy indices. *Biometrical Journal*, 38(8):921–930, 1996.

[36] R.C. Lewontin. The apportionment of human diversity. *Evolutionary Biology*, 6:381–398, 1972.

[37] S. Lieberson. An extension of Greenberg's linguistic diversity measures. *Language*, 40:526–531, 1964.

[38] H. Liu, H. Lu, and J. Yao. Identifying relevant databases for multidatabase mining. In X. Wu, R. Kotagiri, and K. Korb, editors, *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 210–221, Melbourne, Australia, April 1998.

[39] R.H. MacArthur. Patterns of species diversity. *Biological Review*, 40:510–533, 1965.

[40] A.E. Magurran. *Ecological diversity and its measurement*. Princeton University Press, 1988.

[41] R.P. McIntosh. An index of diversity and the relation of certain concepts to diveristy. *Ecology*, 48(3):392–404, 1967.

[42] J. Molinari. A calibrated index for the measurement of evenness. *Oikos*, 56(3):319–326, 1989.

[43] D. Partridge and W. Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information and Software Technology*, 39:707–717, 1997.

[44] G.P. Patil and C. Taillie. Diversity as a concept and its measurement. *Journal of the American Statistical Association*, 77(379):548–567, 1982.

[45] A.D. Pratt. A measure of class concentration in bibliometrics. *Journal of the American Society for Information Science*, 28:285–292, 1977.

[46] J.L. Ray and J.D. Singer. Measuring the concentration of power in the international system. *Sociological Methods and Research*, 1:403–437, 1973.

[47] R.R. Schutz. On the measurement of income inequality. *American Economic Review*, 41:107–122, March 1951.

[48] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, 1949.

[49] E.H. Simpson. Measurement of diversity. *Nature*, 163:688, 1949.

[50] G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery in databases using formal concept analysis methods. In J. Zytkow and M. Quafafou, editors, *Proceedings of the Second European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD'98)*, pages 450–458, Nantes, France, September 1998.

[51] H. Theil. *Economics and information theory*. Rand McNally, 1970.

[52] M.L. Weitzman. On diversity. *The Quarterly Journal of Economics*, pages 363–405, May 1992.

Utility-Based Anonymization for Privacy Preservation with Less Information Loss*

Jian Xu[1] Wei Wang[1] Jian Pei[2] Xiaoyuan Wang[1] Baile Shi[1] Ada Wai-Chee Fu[3]
[1]Fudan University, China, {xujian, weiwang1, xy_wang, bshi}@fudan.edu.cn
[2]Simon Fraser University, Canada, jpei@cs.sfu.ca
[3]The Chinese University of Hong Kong, adafu@cse.cuhk.edu.hk

ABSTRACT

Privacy becomes a more and more serious concern in applications involving microdata. Recently, efficient anonymization has attracted much research work. Most of the previous methods use global recoding, which maps the domains of the quasi-identifier attributes to generalized or changed values. However, global recoding may not always achieve effective anonymization in terms of discernability and query answering accuracy using the anonymized data. Moreover, anonymized data is often used for analysis. As well accepted in many analytical applications, different attributes in a data set may have different utility in the analysis. The utility of attributes has not been considered in the previous methods.

In this paper, we study the problem of *utility-based anonymization*. First, we propose a simple framework to specify utility of attributes. The framework covers both numeric and categorical data. Second, we develop two simple yet efficient heuristic local recoding methods for utility-based anonymization. Our extensive performance study using both real data sets and synthetic data sets shows that our methods outperform the state-of-the-art multidimensional global recoding methods in both discernability and query answering accuracy. Furthermore, our utility-based method can boost the quality of analysis using the anonymized data.

*This research was supported by the Shanghai Raising Star Program Grant 05QMX1405, the National Natural Science Foundation of China Grants 69933010 and 60303008, the NSERC Grants 312194-05 and 614067, the NSF Grant IIS-0308001, and the RGC Earmarked Research Grant of HKSAR CUHK 4120/05E. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.
An abridged edition of this paper titled "*Utility-Based Anonymization Using Local Recoding*" appears in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Security, Algorithms, Performance

Keywords

Privacy preservation, data mining, k-anonymity, utility, local recoding

1. INTRODUCTION

Recently, privacy becomes a more and more serious concern in applications involving *microdata*, which refers to data published in its raw, non-aggregated form [17]. One important type of privacy attack is re-identifying individuals by joining multiple public data sources. For example, according to [15], more than 85% of the population of the United States can be uniquely identified using their zipcode, gender, and date of birth.

To protect privacy against this type of attacks, k-anonymity was proposed [12, 15]. A data set is *k-anonymous* ($k \geq 1$) if each record in the data set is indistinguishable from at least $(k-1)$ other records within the same data set. The larger the value of k, the better the privacy is protected.

Since the concept of k-anonymity has been proposed, efficient methods for anonymization has attracted much research work. A few k-anonymization algorithms have been developed. We shall review the related work briefly in Section 2. Generally, to achieve k-anonymity, those methods generalize or suppress the *quasi-identifier attributes*, which are the minimal set of attributes in the table that can be joined with external information to re-identify individual records.

Information loss is an unfortunate consequence of anonymization. In order to make the anonymized data as useful as possible, it is required to reduce the information loss as much as possible. A few models have been proposed to measure the usefulness of anonymized data. For example, the discernability model [4] tries to minimize the number of tuples that are indistinguishable, as long as they satisfy the k-anonymity requirement.

In this paper, we study the problem of k-anonymization and focus on two interesting issues: *anonymization using heuristic local recoding* and *utility-based anonymization*.

1.1 Global and Local Anonymization

Many recent methods (e.g., [4, 8, 9]) use *global recoding*, which maps the domains of the quasi-identifier attributes to generalized or changed values. In other words, the data space is partitioned into a set of (non-overlapping) regions. The anonymization maps all tuples in a region to the same generalized or changed tuple. For example, Figures 1(b) demonstrates a 3-anonymization using global recoding for the table in Figures 1(a), where (age, zipcode) is the quasi-identifier. Tuples $R3$ an $R4$ in Figures 1(a) are identical. They are mapped to the same generalized tuple in global recoding.

In contrast, *local recoding* maps (non-distinct) individual tuple to generalized tuples. For example, Figure 1(c) shows a 3-anonymization using local recoding of the same table in Figures 1(a). The two identical tuples, $R3$ and $R4$, are mapped to different generalized tuples in local recoding. Clearly, global recoding can be regarded as a specific type of local recoding.

Interestingly, from Figure 1, we can observe that *local recoding may achieve a less information loss than global recoding*. In our example, the two generalized tuples in global recoding have the sizes of intervals 8 and 5 in age, and 1 and 0 in zipcode, respectively. In local recoding, the sizes of intervals are 6 and 2 in age, and 1 and 2 in zipcode, respectively. By intuition, smaller the sizes of intervals in the generalized tuples, less information loss in the anonymization.

Can we use local recoding to achieve less information loss in anonymization effectively? Generally, optimal k-anonymity is NP-hard [10, 2]. In this paper, we propose two simple yet efficient heuristic algorithms using local recoding for k-anonymization. Our extensive empirical study on both real data sets and synthetic data sets show that our method outperforms the state-of-the-art global recoding method in both the discernability and the accuracy of query answering.

1.2 Utility-Based Anonymization

Anonymized data is often for analysis and data mining. As well recognized in many data analysis applications, different attributes may have different utility. For example, consider anonymizing a data set about patients for disease analysis. Suppose in order to achieve k-anonymity, we can generalize from a five-digit full zipcode to a four-digit prefix (e.g., from 53712 to 5371$*$). Alternatively, we can also generalize attribute age to age groups (e.g., from 23 to [20, 30]). In many cases, the age information is critical to disease analysis, while the information loss on the accurate location is often acceptable (a four digit prefix in fact still identifies a relatively local region). Thus, the age attribute has more utility than the zipcode attribute, and should be retained as accurately as possible in anonymization.

Can we make the anonymization utility aware? Utility of attributes has not been considered by previous anonymization methods. In this paper, we propose a model for *utility-based anonymization*. We consider both numeric data and categorical data with and without hierarchies. We present a simple method to specify utility of attributes and push them into the heuristic local recoding anonymization methods. Our experimental results show that the utility-based anonymization improves the accuracy in answering targeted queries substantially.

Paper Organization

The rest of the paper is organized as follows. In section 2, we recall the notions related to anonymization, and review the related work. We present our utility specification framework in Section 3. Our heuristic local recoding methods are developed in Section 4. An extensive performance study on both real data sets and synthetic data sets is reported in Section 5. The paper is concluded in Section 6.

2. K-ANONYMITY AND RELATED WORK

Consider a table $T = (A_1, \ldots, A_n)$. A *quasi-identifier* is a minimal set of attributes $(A_{i_1}, \ldots, A_{i_l})$ $(1 \leq i_1 < \cdots < i_l \leq n)$ in T that can be joined with external information to re-identify individual records. In this paper, we assume that the quasi-identifier is specified by the administrator based on the background knowledge. Thus, we focus on how to anonymize T to satisfy the k-anonymity requirement.

Formally, given a parameter k and the quasi-identifier $(A_{i_1}, \ldots, A_{i_l})$, a table T is said *k-anonymous* if for each tuple $t \in T$, there exist at least another $(k-1)$ tuples $t_1, \ldots, t_{k-1}$ such that those k tuples have the same projection on the quasi-identifier, i.e., $t_{(A_{i_1},\ldots,A_{i_l})} = t_{1_{(A_{i_1},\ldots,A_{i_l})}} = \cdots = t_{k-1_{(A_{i_1},\ldots,A_{i_l})}}$. Tuple t and all other tuples indistinguishable from t on the quasi-identifier form an *equivalence class*. We call the class the *group* that t is generalized.

Given a table T with the quasi-identifier and a parameter k, the problem of *k-anonymization* is to compute a view T' that has the same attributes as T such that T' is k-anonymous and T' is as close to T as possible according to some quality metric. We shall discuss the quality metrics soon.

Since the attributes not in the quasi-identifier do not need to be changed, to keep our discussion simple but without loss of generality, hereafter we consider only the attributes in the quasi-identifier. That is, for table $T(A_1, \ldots, A_n)$ in question, we assume $(A_1, \ldots, A_n)$ is the quasi-identifier.

K-anonymization was proposed by Samarati and Sweeney [11, 13, 15, 14]. Generally, data items are recoded in anonymization. Here, we regard suppression as a specific form of recoding that recodes a data item to null value (i.e., unknown).

Two types of recoding can be used [17]: global recoding and local recoding, as described and demonstrated in Section 1.1. Many previous methods use global recoding. In [11, 13], *full-domain generalization*, a specific type of global recoding, was developed, which maps the whole domain of each quasi-identifier attribute to a more general domain in the domain generalization hierachy. Full-domain generalization guarantees that all values of a particular attribute still belong to the same domain after generalization.

To achieve full-domain generalization, two types of partitioning can be applied. First, single-dimensional partitioning [4, 7] divides an attribute into a set of non-overlapping intervals, and each interval will be replaced by a summary value (e.g., the mean, the median, or the range). On the other hand, (strict) multidimensional partitioning [9] divides the domain into a set of non-overlapping multidimensional regions, and each region will be generalized into a summary tuple.

Generally, anonymization is accompanied by information loss. Various models have been proposed to measure the information loss. For example, the *discernability model* [4] assigns to each tuple t a penalty based on the size of the group that t is generalized, i.e., the number of tuples equivalent to t on the quasi-identifier. That is,

$$C_{DM} = \sum_{E \in \text{group-bys on quasi-identifier}} |E|^2.$$

Alternatively, the *normalized average equivalence class size met-*

Row-id	Age	Zipcode
$R1$	24	53712
$R2$	25	53711
$R3$	30	53711
$R4$	30	53711
$R5$	32	53712
$R6$	32	53713

(a) The original table.

Row-id	Age	Zipcode
$R1$	[24-32]	[53712-53713]
$R2$	[25-30]	53711
$R3$	[25-30]	53711
$R4$	[25-30]	53711
$R5$	[24-32]	[53712-53713]
$R6$	[24-32]	[53712-53713]

(b) 3-anonymization by global recoding.

Row-id	Age	Zipcode
$R1$	[24-30]	[53711-53712]
$R2$	[24-30]	[53711-53712]
$R3$	[24-30]	[53711-53712]
$R4$	[30-32]	[53711-53713]
$R5$	[30-32]	[53711-53713]
$R6$	[30-32]	[53711-53713]

(c) 3-anonymization by local recoding.

Figure 1: Global recoding and local recoding. The row-ids are for reference only and are not released with the data. Thus, the row-ids are not part of the quasi-identifier.

ric was given in [9]. The intuition of the metric is to measure how well the partitioning approaches the best case where each tuple is generalized in a group of k indistinguishable tuples. That is,

$$C_{AVG} = \frac{\text{number of tuples in the table}}{\text{number of group-bys on quasi-identifier} \cdot k}.$$

The quality of anonymization can also be evaluated based on its usefulness in data analysis applications, such as classification [6, 16].

The ideal anonymization should minimize the penalty. However, theoretical analysis [2, 10, 9, 3, 1] indicates that the problem of optimal anonymization under many non-trivial quality models is NP-hard. A few approximation methods were developed [3], such as datafly [14], annealing [18], and Mondrian multidimensional k-anonymity [9]. Interestingly, some optimal methods [4, 8] with exponential cost in the worst case were proposed. The experimental results in those studies show that they are feasible and can achieve good performance in practice.

3. UTILITY-BASED ANONYMIZATION

Without loss of generality, in this paper we assume that generalization is used in anonymization. That is, when a tuple is generalized, the ranges of the group of tuples that are generalized are used to represent the generalization, as illustrated in Figure 1. If other representations such as mean or median are used, the definitions can be revised straightforwardly and our methods still work.

3.1 Utility-Based Anonymization: Motivation

In previous methods, the quality metrics, such as the discernability metric and the normalized average equivalence class size metric discussed in Section 2, mainly focus on the size of groups in anonymization. In an anonymized table, when each group of tuples sharing the same projection on the quasi-identifier has k tuples, the penalty metrics are minimized. However, such metrics may not lead to high quality anonymization.

EXAMPLE 1 (QUALITY METRICS). Suppose we want to achieve 2-anonymity for the six tuples shown in Figure 2. (X, Y) is the quasi-identifier. The six tuples can be anonymized in three groups: $\{a, b\}$, $\{c, d\}$, and $\{e, f\}$. In this anonymization scheme, both the discernability metric C_{DM} and the normalized average equivalence class size metric C_{AVG} are minimized.

Let us consider the utility of the anonymized data. Suppose each group is generalized using the range of the tuples in the group. That is, a and b are generalized to $([10, 20], [60, 70])$; c and d are generalized to $([20, 50], [20, 50])$; and e and f are generalized to $([50, 60], [10, 15])$.

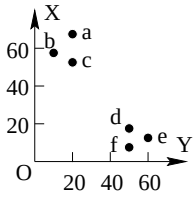


Figure 2: The six tuples in Example 1.

In order to measure how well the generalized tuples approximate the original ones, for each tuple we can use the sum of the interval sizes on all attributes of the generalized tuple to measure the uncertainty of the generalized tuples. That is, $U(a) = U(b) = 10 + 10 = 20$. Similarly, we get $U(c) = U(d) = 60$ and $U(e) = U(f) = 15$. The total uncertainty of the anonymized table is the sum of the uncertainty of all tuples, i.e., $U(T) = \sum_{t \in T} U(t) = 20 + 20 + 60 + 60 + 15 + 15 = 190$. By intuition, the uncertainty reflects the information loss. The less the uncertainty, the less information is lost.

On the other hand, we may anonymize the tuples in two groups: $\{a, b, c\}$ are generalized to $([10, 20], [50, 70])$, and $\{d, e, f\}$ are generalized to $([50, 60], [10, 20])$. In fact, the data set is 3-anonymous, which is better than 2-anonymous in terms of privacy preservation. Moreover, the total uncertainty in this anonymization is 150, lower than the 2-anonymity scheme.

However, this anonymization scheme has a higher penalty than the 2-anonymous scheme in both the discernability metric C_{DM} and the normalized average equivalence class size metric C_{AVG}. In other words, optimizing the quality metrics on group size may not always lead to anonymization that minimizes the information loss. ■

Can we have a quality metric that can measure the utility of the anonymized data? Such a utility-based metric should capture the following two aspects.

- *The information loss caused by the anonymization*. When a record is anonymized, it is generalized in its quasi-identifier. The metric should measure the information loss of the generalization with respect to the original data.
- *The importance of attributes.* As well accepted in data analysis such as aggregate queries, different attributes may have different importance in data analysis. In anonymization, can we introduce less uncertainty to the important attributes? Such utility-aware anonymization may help to improve the quality of analysis afterwards.

3.2 Weighted Certainty Penalty

We introduce the concept of certainty penalty to capture the uncertainty caused by generalization.

3.2.1 Numeric Attributes

First, let us consider the case of numeric attributes. Let T be a table with quasi-identifier $(A_1, \ldots, A_n)$, where all attributes are numeric. Suppose a tuple $t = (x_1, \ldots, x_n)$ is generalized to tuple $t' = ([y_1, z_1], \ldots, [y_n, z_n])$ such that $y_i \leq x_i \leq z_i$ $(1 \leq i \leq n)$. On attribute A_i, the *normalized certainty penalty* is defined as

$$NCP_{A_i}(t) = \frac{z_i - y_i}{|A_i|},$$

where $|A_i| = \max_{t \in T}\{t.A_i\} - \min_{t \in T}\{t.A_i\}$ is the range of all tuples on attribute A_i.

Let each attribute A_i be associated with a weight w_i to reflect its utility in the analysis on the anonymized data. Then, the *weighted certainty penalty* of a tuple is given by

$$NCP(t) = \sum_{i=1}^{n}(w_i \cdot NCP_{A_i}(t)) = \sum_{i=1}^{n}(w_i \cdot \frac{z_i - y_i}{|A_i|}).$$

Clearly, when all weights are set to 1 and all attributes have ranges $[0, 1]$, the weighted certainty penalty is the L_1 norm distance between points $(\max_{t \in G}\{t.A_1\}, \ldots, \max_{t \in G}\{t.A_n\})$ and $(\min_{t \in G}\{t.A_1\}, \ldots, \min_{t \in G}\{t.A_n\})$, where G is the equivalence group that t belongs to.

Our utility-based metric is given by the total weighted certainty penalty on the whole table. That is,

$$NCP(T) = \sum_{t \in T} NCP(t).$$

3.2.2 Categorical Attributes

Distance is often not well defined on categorical attributes, which makes measuring utility on categorical attributes difficult. In some previous methods (e.g., [8, 9]), it is assumed that a total order exists on all values in a categorical attribute. In many applications, such an order may not exist. For example, sorting all zipcodes in their numeric order may not reflect the utility properly. Two regions may be adjacent but their zipcodes may not be consecutive.

More often than not, hierarchies exist in categorical attributes. For example, zipcodes can be organized into hierarchy of regions, cities, counties, and states.

Let $v_1, \ldots, v_l$ be a set of leaf nodes in a hierarchy tree. Let u be the node in the hierarchy on the attribute such that u is an ancestor of $v_1, \ldots, v_l$, and u does not have any descendant that is still an ancestor of $v_1, \ldots, v_l$. u is called the *closest common ancestor* of $v_1, \ldots, v_l$, denoted by $ancestor(v_1, \ldots, v_l)$. The number of leaf nodes that are descendants of u is called the *size* of u, denoted by $size(u)$.

Can we use the hierarchy information to measure the utility on categorical attributes?

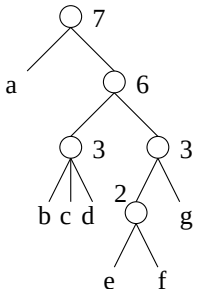


Figure 3: A hierarchy on a categorical attribute.

EXAMPLE 2 (UTILITY ON CATEGORICAL ATTRIBUTES). Consider a categorical attribute of domain $\{a, b, c, d, e, f, g\}$. Suppose a hierarchy exists on the attribute as shown in Figure 3. The values appear in the leaf nodes in the hierarchy tree.

Intuitively, if we generalize tuples having values b and c, the anonymized tuples have good utility on this categorical attributes, since b and c share the same parent in the hierarchy. On the other hand, putting a and f into the same generalized group may have poor utility on the attribute since the common ancestor of a and f is far away from f.

One may wonder whether the shortest distance between u and v in the hierarchy tree can be used as the certainty penalty. Unfortunately, it does not work well. Consider Figure 3 again. Intuitively, generalizing d and e together is better than generalizing a and d together, since the closest common ancestor of d and e is in a hierarchical level lower than the closest common ancestor of a and d. However, the shortest distance between d and e is 5, while the shortest distance between a and d is only 4. If we use the shortest distance as the guide, then merging a and d is better than merging d and e. In other words, the shortest distance may be misleading.

To measure the utility of merging two values x and y into the same generalized group, we can observe that the critical factor is for the closest common ancestor u of x and y, how many other values are also the descendants of u. The smaller the number, the smaller the uncertainty introduced by the generalization. ■

Based on the observation in Example 2, we define the certainty penalty on categorical attributes as follows.

Suppose a tuple t has value v on a categorical attribute A. When it is generalized in anonymization, the value will be replaced by a set of values $\{v_1, \ldots, v_l\}$, where $v_1, \ldots, v_l$ are the values of tuples on the attribute in the same generalized group. We define the *normalized certainty penalty* of t as follows.

$$NCP_A(t) = \frac{size(u)}{|A|},$$

where $|A|$ is the number of distinct values on attribute A. Here, we assume that each leaf node is of the same importance. The definition can be straightforwardly extended by assigning weights to internal nodes to capture the more important leaf nodes and internal hierarchical structures. Limited by space, we omit the details here.

EXAMPLE 3. Let us consider the cases discussed in Example 2 again. Putting a and d together in a group has penalty 1, and putting d and e together in a group has penalty $\frac{6}{7}$ only, which is smaller than the case of a and d. ■

Putting things together, for a table consisting of both numeric and categorical attributes, the total weighted normalized certainty penalty is the sum of the weighted normalized certainty penalty of all tuples. That is,

$$NCP(T) = \sum_{t \in T}\sum_{i=1}^{n}(w_i \cdot NCP_{A_i}(t)),$$

where $NCP_{A_i}(t)$ should be computed according to whether A_i is a numeric or categorical attribute.

Given a table T, a parameter k, the weights of attributes and the hierarchies on categorical attributes, the *problem of optimal utility-based anonymization* is to compute a k-anonymous table T' such that the weighted normalized certainty penalty on T' is minimized.

3.3 Complexity

The previous studies show that the problem of optimal k-anonymity is NP-hard under various quality models. The utility-based model we propose here is a generalization of the suppression model. We have the following results on the complexity.

LEMMA 1 (CATEGORICAL ATTRIBUTES). *Suppose the quasi-identifier has only categorical attributes. The problem of optimal utility-based k-anonymization is NP-hard for $k \geq 2$.*

Proof sketch. We can show that the suppression model used in [2] is a special case of the weighted normalized certainty penalty defined here, where all weights are set to 1 and all hierarchies have only two levels: the detailed values and suppression. The lemma follows from the result in [2]. ∎

Following from the lemma, we have the following result.

THEOREM 1 (COMPLEXITY). *The problem of optimal utility-based anonymization is NP-hard.* ∎

In fact, for a table consisting of only numeric attributes, the problem is still NP-hard. Limited by space, we omit the details here.

4. GREEDY METHODS

In this section, we develop heuristic methods for utility-based anonymization. We propose two greedy algorithms. The first method conducts a bottom-up search, while the second one works top-down.

4.1 The Bottom-Up Method

To maximize the utility of the anonymization of a tuple, we may "cluster" the tuples locally according to the weighted certainty penalty. Those compact clusters having at least k tuples can be generalized. This idea leads to our bottom-up method.

At the beginning, we treat each tuple as an individual group. In each iteration, for each group whose population is less than k, we merge the group with the other group such that the combined group has the smallest weighted certainty penalty. The iteration goes on until every group has at least k tuples. The algorithm is shown in Figure 4.

The bottom-up algorithm is a greedy method. In each round, it merges groups such that the resulted weighted certainty penalty is locally minimized. In one iteration, if one group is merged with multiple groups, it is possible that the group becomes larger than k. In order to avoid over-generalization, if a group has more than $2k$ tuples, then the group should be split. It is guaranteed that in the resulted table, each group has up to $(2k-1)$ tuples.

Please note that, unlike many previous methods that try to minimize the average number of tuples per group, our algorithms try to reduce the weighted certainty penalty, which reflects the utility of the anonymized data. At the same time, they also keep the number of tuples per group small.

Input: a table T, parameter k, weights of attributes, and hierarchies on categorical attributes;
Output: a k-anonymous table T';
Method:

1: Initialization: create a group for each tuple;
2: `WHILE` there exists some group G such that $|G| < k$ `DO` {
3: `FOR` each group G such that $|G| < k$ `DO` {
4: scan all other groups once to find group G' such that $NCP(G \cup G')$ is minimized;
5: merge groups G and G';
}
6: `FOR` each group G such that $|G| \geq 2k$ `DO`
7: split the group into $\lfloor \frac{|G|}{k} \rfloor$ groups such that each group has at least k tuples;
}
8: generalize and output the surviving groups;

Figure 4: The bottom-up algorithm.

EXAMPLE 4 (ADVANTAGES OF THE BOTTOM-UP METHOD). To understand the difference between our method and the previous methods, let us check the case in Figure 2. The bottom-up method generates two groups: $\{a, b, c\}$ and $\{d, e, f\}$, as expected in Example 1. Although it does not minimize the average group size, it optimizes the utility of the anonymized data – the information loss is better than any 2-anonymous scheme in this example. Moreover, as a byproduct, the result is 3-anonymous, which means a stronger protection of privacy. ∎

After the k-th round, the number of tuples in a group is at least 2^k. Therefore, by at most $\lceil \log_2 k \rceil$ iterations, each group has at least k tuples, and thus the generalized groups satisfy the k-anonymity requirement. The complexity of the algorithm is $O(\lceil \log_2 k \rceil |T|^2)$ on table T.

The bottom-up method is a local recoding method. It does not split the domain. Instead, it only searches the tuples. Different groups may have overlapping ranges. Moreover, in the step of splitting, several tuples with the identical quasi-identifier may be split into different groups.

4.2 A Top-Down Approach

The major cost in the bottom-up method is to search for the closest groups (Step 4 in Figure 4). In the bottom-up method, we have to use a two-level loop to conduct the search. We observe, if we can partition the data properly so that the tuples in each partition are local, then the search of the nearest neighbors can be sped up. Motivated by this observation, we develop the top-down approach.

The general idea is as follows. We partition the table iteratively. A set of tuples is partitioned into subsets if each subset is more local. That is, likely they can be further partitioned into smaller groups that reduce the weighted certainty penalty. After the partitioning, we merge the groups that are smaller than k to honor the k-anonymity requirement.

To keep the algorithm simple, we consider binary partitioning. That is, in each round, we partition a set of tuples into two subsets. The algorithm framework is shown in Figure 5.

Now, the problem becomes how we can partition a set of tuples into two subsets so that they are compact and likely lead to small weighted certainty penalty. We adopt the following heuristic. We

Input: a table T, parameter k, weights of attributes, hierarchies on categorical attributes;
Output: a k-anonymous table T';
Method:
1: IF $|T| \leq k$ THEN RETURN;
2: ELSE {
3: partition T into two exclusive subsets T_1 and T_2 such that T_1 and T_2 are more local than T, and either T_1 or T_2 have at least k tuples;
4: IF $|T_1| > k$ THEN recursively partition T_1;
5: IF $|T_2| > k$ THEN recursively partition T_2;
}
6: adjust the groups so that each group has at least k tuples;

Figure 5: The framework of the top-down greedy search method.

form two groups using the two seed tuples that cause the highest certainty penalty if they are put into the same group, and assign the other tuples into the two groups according to the two seed tuples.

Technically, we want to find tuples $u, v \in T$ that maximize $NCP(u, v)$. u and v become the seed tuple of groups G_u and G_v, respectively.

The cost of finding u, v such that $NCP(u, v)$ is maximized is $O(|T|^2)$. To reduce the cost, we propose a heuristic method here. We randomly pick a tuple u_1. By scanning all tuples once, we can find tuple v_1 that maximizes $NCP(u_1, v_1)$. Then, we scan all tuples again, find tuple u_2 that maximizes $NCP(u_2, v_1)$. The iteration goes on a few rounds until $NCP(u, v)$ does not increase substantially. Our experimental results on both the real data sets and the synthetic data sets show that the maximal weighted certainty penalty converges quickly. By up to 3 rounds, we can achieve 97% of the maximal penalty. By up to 6 rounds, we can achieve more than 98.75% of the maximal penalty. In practice, we can choose a small integer as the number of rounds to find the seed tuples.

Once the two seed tuples are determined, two groups G_u and G_v are created. Then, we assign other tuples to the two groups one by one in a random order. For tuple w, the assignment depends on $NCP(G_u, w)$ and $NCP(G_v, w)$, where G_u, G_v are the groups formed so far. Tuple w is assign to the group that leads to lower uncertainty penalty.

If at least one group has k or more tuples, then the partitioning is conducted. The top-down method is recursively applied to those groups having at least k tuples.

We have a postprocessing step to adjust for those groups with less than k tuples. If one group G has less than k tuples, we apply the local greedy adjustment similar to the bottom-up approach. That is, we consider two alternatives. First, we can find a set G' of $(k - |G|)$ tuples in some other group that has more than $(2k - |G|)$ tuples such that $NCP(G \cup G')$ is minimized. Second, we compute the increase of penalty by merging G with the nearest neighbor group of G. By comparing the two penalty measures, we decide whether G' is moved to G or G is combined with its nearest neighbor group. Such adjustments should be done until every group has at least k tuples, i.e., the k-anonymity requirement is satisfied.

In worst case, the partition depth is bounded by $O(|T|)$. In each step of partition, it takes $O(m)$ time cost to partition the m tuples in the current set into two subsets. Thus, the overall partitioning cost is $O(|T|^2)$. After the top-down partitioning, in the worst case, we may have to adjust $\lfloor \frac{|T|}{2k} \rfloor$ groups each having less than k tuples. Thus, the cost of adjustment is $O(|T|^2)$ in the worst case. However, in practice, the number of groups that are smaller than k is much less than the worst case. As shown in our experiments, the top-down method is clearly faster than the bottom-up method.

The top-down method is also a local recoding method, since in the adjustment step, similar to the bottom-up method, two tuples identical in the quasi-identifier may be assigned to two different groups.

5. EXPERIMENTAL RESULTS

To evaluate the two heuristic methods proposed in this paper, we conducted an extensive empirical study using both real data sets and synthetic data sets.

5.1 Settings and Evaluation Criteria

We compare three methods: the mondarian multidimensional k-anonymization method [9], the bottom-up method and the top-down method developed in this paper. According to [9], the mondarian multidimensional k-anonymization method (called MultiDim for short hereafter) is so far the best method in both quality (measured by the discernability penalty) and efficiency. The general idea of the method is a top-down greedy search that is similar to building kd-trees [5]. At each step, it chooses a dimension to split the data set at the median of the dimension. Heuristically, the dimension with the widest normalized range of values is chosen.

We measure the quality of the anonymization using three criteria: the certainty penalty, the discernability penalty, and the error rate in query answering. The certainty penalty proposed in this paper measures the utility of the anonymization. The discernability penalty is a de facto standard measure on anonymization quality used in many previous studies. The error rate measures how effective the anonymized data sets are in query answering.

All our experiments were conducted on a PC with a Pentium P4 2.0 GHz CPU and 512 MB main memory, running Microsoft Windows XP. All the algorithms were implemented by us in Microsoft Visual C++ version 6.0.

5.2 Results on Real Data Set Adults

The Adults census data set from the UC Irvine machine learning repository has become a de facto benchmark for k-anonymization. The data set was configured as described in [4]. The salary class attribute was dropped, and the tuples with missing values were removed. The resulting data set contains $30,162$ tuples.

Since the MultiDim method does not handle hierarchies on categorical attributes but treats a categorical attribute as a discrete numeric attribute, we configured the data set for MultiDim as it was used in [9]. For the bottom-up method and the top-down method proposed in this paper, we used age and education levels as numeric data, and use the other attributes as categorical attributes. We used the two hierarchies in Figure 6 on attributes work-class and marital-status. On other categorical attributes, a simple two-level hierarchy is applied: the values are the leaf nodes and the root is ALL (i.e., suppression). All weights were set to 1.

Figure 7 shows the certainty penalty of the anonymization of the three methods with respect to different k values. As expected, since the bottom-up method and the top-down method focus on the certainty penalty, but the MultiDim method does not, the anonymiza-

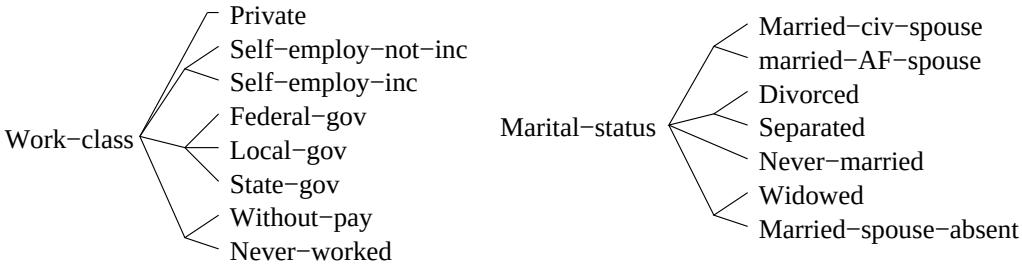


Figure 6: The hierarchies on attributes work-class and marital-status.

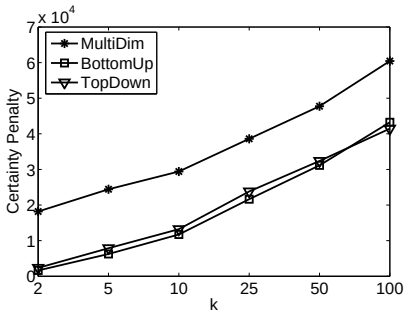


Figure 7: Certainty penalty on data set Adults.

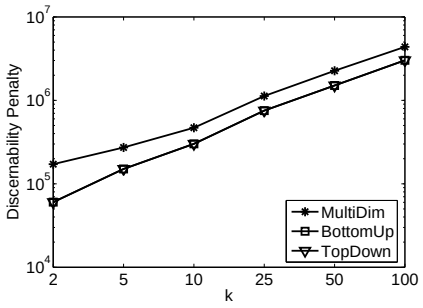


Figure 8: Discernability penalty on data set Adults.

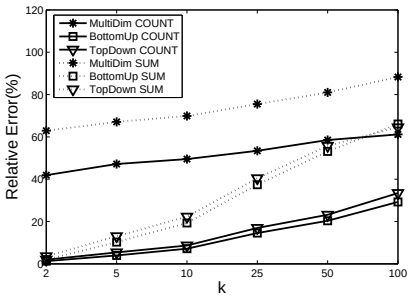


Figure 9: Query answering error rate on data set Adults.

tion generated by the bottom-up method and the top-down method has a clearly lower certainty penalty. The gap is stable, about 2×10^4.

Figure 8 compares the discernability penalty of the anonymization generated by the three methods with respect to different values of k. Interestingly, although the bottom-up and the top-down methods do not explicitly focus on reducing the discernability penalty, they outperform the MultiDim method. Please note that the discernability penalty in the figure is drawn in the logarithmic scale. The results show that optimizing the utility and the reducing the discernability are not conflicting with each other. In fact, the two methods also try to keep the size of groups same when they reduce the certainty penalty. Grouping tuples locally can bring us benefit on reducing both the certainty penalty and the discernability penalty.

Interestingly, the anonymized data sets generated by the bottom-up method and the top-down method are comparable in both the certainty penalty and the discernability. This is not unexpected since the two methods greedily group tuples locally to achieve k-anonymity.

To test the effectiveness of query answering using the anonymized data, we generate workloads using `SUM` and `COUNT` aggregate queries, respectively. Each workload has $1,000$ random queries. Each `COUNT` query involves all the attributes, and each `SUM` query involves all but the age attribute that is used to compute the sum. The ranges of the attributes are selected randomly. For a categorical attribute, a query carries either a random categorical value, or a set of values that are summarized by an internal node in the hierarchy as the range. This is consistent with the settings in [9].

Figure 9 shows the results on two workloads of aggregate functions `COUNT` and `SUM`, respectively, with respect to different k values. Clearly, the bottom-up method and the top-down method outperform the MultiDim method substantially. The results can be explained in two aspects. First, the utility-driven anonymization put tuples that are similar to each other into groups. Thus, the generalized groups often have small ranges, and can answer queries more accurately. Second, our methods handle categorical attributes better than the MultiDim method. The hierarchies are considered in the anonymization. This contributes to the query answering quality strongly.

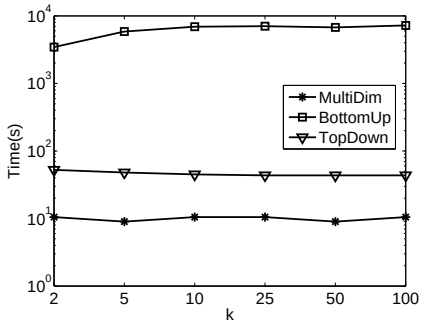


Figure 10: Runtime with respect to k on data set Adults.

Figure 10 shows the runtime of the three methods. As the trade-off, the bottom-up and the top-down methods consumes more runtime than the MultiDim method. The top-down method is about 5-6 times slower than MultiDim, and is much faster than the bottom-up method. The runtime of the three methods is not sensitive to k. The difference in the efficiency can be explained by their complexity. While the MultiDim method has the complexity $O(|T| \log |T|)$, the bottom-up and the top-down methods have complexity $O(|T|^2)$.

5.3 Results on Synthetic Data Sets

To test the performance of the three methods more thoroughly,

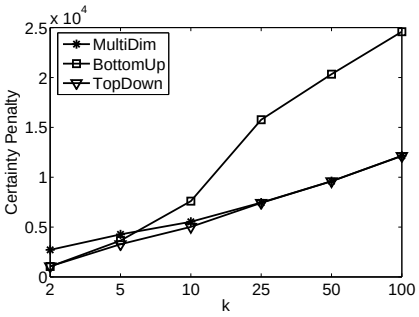


Figure 11: Certainty penalty with respect to k, on synthetic data sets with uniform distribution (dimensionality = 4).

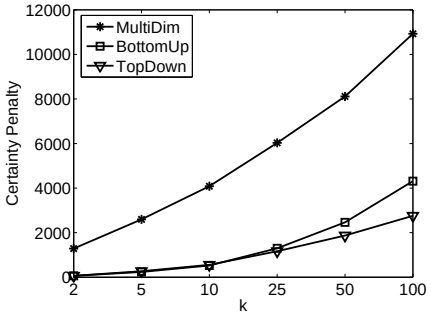


Figure 12: Certainty penalty with respect to k, on synthetic data sets with Gaussian distribution (dimensionality = 4, $\sigma = 1.0$).

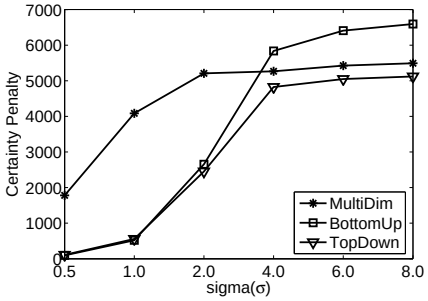


Figure 13: Certainty penalty with respect to σ, on synthetic data sets with Gaussian distribution (dimensionality=4, $k = 10$).

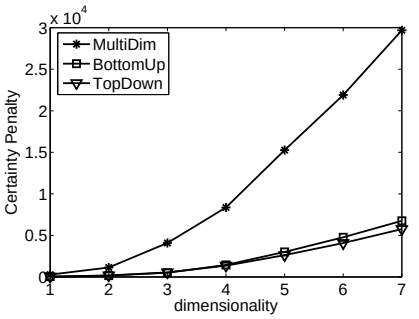


Figure 14: Certainty penalty with respect to dimensionality, on synthetic data sets with Gaussian distribution ($\sigma = 1.0$, $k = 10$).

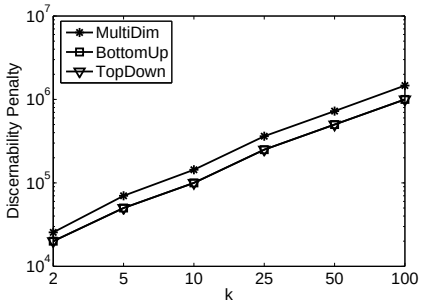


Figure 15: Discernability penalty with respect to k, on synthetic data sets with uniform distribution (dimensionality = 4).

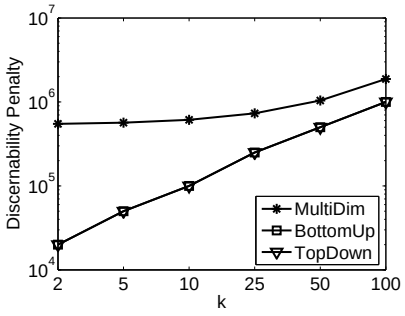


Figure 16: Discernability penalty with respect to k, on synthetic data sets with Gaussian distribution (dimensionality = 4, $\sigma = 1.0$).

we generated synthetic data sets in two types of distributions: uniform distribution and Gaussian distribution. The dimensionality and the number of tuples may vary according to the needs of experiments. By default, a data set has $10,000$ tuples and each attribute is in the domain of integer with range $[1, 16]$. Again, by default the weights are set to 1.

5.3.1 Anonymization Quality

Figures 11 and 12 show the certainty penalty with respect to k on the synthetic data sets with uniformly distribution and Gaussian distribution, respectively. In the uniform distributed data, the MultiDim method and the top-down method are comparable, and the top-down method is better when k is small. The bottom-up method performs poorly. The reason is that with uniform distribution, the kd-tree like construction in the MultiDim method can partition the data set evenly into groups with hyper-rectangle bounding boxes so that each group is balanced and achieves low penalty. The same happens to the top-down method as well. In the bottom up method, the groups formed by merging may be in irregular shape and thus may lead to high certainty penalty.

In data sets with Gaussian distribution, both the top-down method and the bottom-up method work better than the MultiDim method. The advantage is clear. With bias data, local search and local recoding may have good chance to find local clusters that lead to low certainty penalty.

It is interesting to test the certainty penalty with respect to the degree of bias in data. Figure 13 shows the results. The top-down method is consistently the best. When the data is severely biased, the MultiDim method performs poorly. But when the data becomes less biased, the MultiDim method catches up with and even outpeforms the bottom-up method, but is still worse than the top-down method.

Figure 14 shows the certainty penalty with respect to various dimensionality. The top-down method and the bottom-up method are comparable, and the top-down method is slightly better. The MultiDim method has a high certainty penalty in high dimensional data. Please note that, as the dimensionality increases, the certainty penalty generally increases accordingly since each attribute contributes to the certainty penalty. The bottom-up and the top-down methods try to reduce the penalty in the anonymization procedure and thus may achieve good results.

We also test the quality of the anonymization using the discernability penalty measure. Figures 15, 16, 17, and 18 show the results on the cases in Figures 11, 12, 13, and 14, respectively. The results using the discernability penalty measure are consistent with the results reported in [9].

From the results, we can observe that the bottom-up method and the top-down method have similar performance, and achieve less discernability penalty than the MultiDim method in all cases. This is consistent with the results on the real Adults data set.

From this set of experiments, we conclude that the bottom-up and the top-down methods often have similar performance in anonymiza-

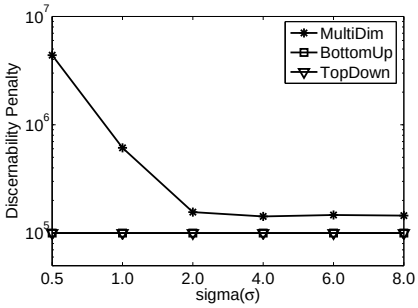


Figure 17: Discernability penalty with respect to σ, on synthetic data sets with Gaussian distribution (dimensionality=4, $k = 10$).

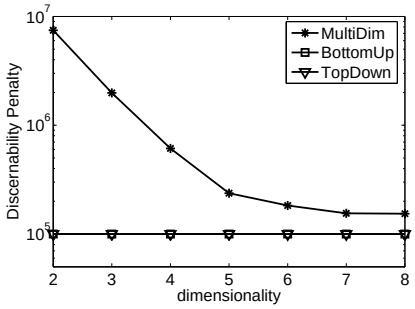


Figure 18: Discernability penalty with respect to dimensionality, on synthetic data sets with Gaussian distribution ($\sigma = 1.0$, $k = 10$).

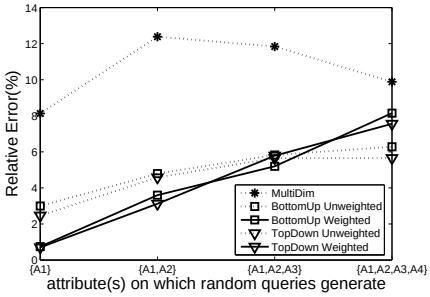


Figure 19: Utility in query answering, on synthetic data sets with uniform distribution (dimensionality=4, $k = 10$).

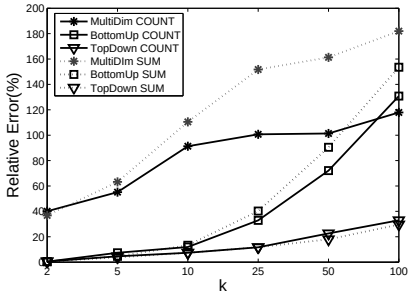


Figure 20: Query answering error rate, on synthetic data sets with Gaussian distribution (dimensionality=4, $\sigma = 1.0$).

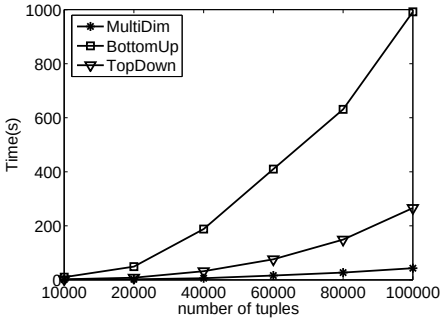


Figure 21: Scalability with respect to database size, on synthetic data sets with uniform distribution (dimensionality=4, $k = 10$).

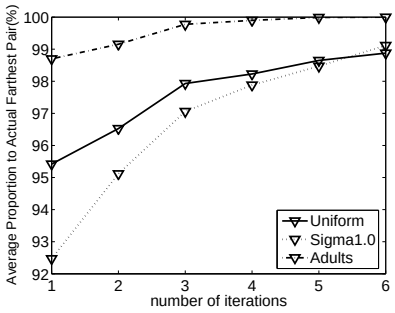


Figure 22: The effectiveness of the seed tuple choice heuristic in the top-down method, on real data set Adults, and synthetic data sets with uniform and Gaussian distribution (dimensionality=4).

tion quality, measured by both the certainty penalty and the discernability. The anonymization quality using those two methods are often better than the MultiDim method.

5.3.2 Utility and Query Answering

To test the utility in query answering, we use a uniformly distributed data set with 4 attributes, and set $k = 10$. We assign weights 8, 4, 2, and 1 to attributes A_1, A_2, A_3, and A_4, respectively. That is, the information loss in attribute A_1 is strongly undesirable.

We generate 4 groups of random queries on attribute combinations A_1, A_1A_2, $A_1A_2A_3$, and $A_1A_2A_3A_4$, respectively. The average error rates of the queries in each group is shown in Figure 19. For comparison, we also conduct the same queries on anonymization that do not consider the weights.

As can be seen, the effect of utility-based anonymization is significant. The anonymization using the weighted top-down or bottom-up methods answers the queries on A_1, A_1A_2, and $A_1A_2A_3$ more accurately than the non-weighted methods. When all attributes are involved in a query, the weighted methods may lose some accuracy as the trade-off.

We also test the average error rates using the anonymized data to answer aggregate queries. Figure 20 shows the results. In this experiment, we assign the default weight 1 to every attribute, and test two aggregate functions `SUM` and `COUNT`. The average error rate is computed from $1,000$ random queries. The methodology is the same as the experiment reported in Figure 9 and the experiments reported in [9].

The results show that both the bottom-up and the top-down methods achieve lower error rate than the MultiDim method when k is not large, since local recoding often groups tuples with small certainty penalty. When k is large, the top-down method has the best performance, and is clearly better than the other two methods.

5.3.3 Efficiency and Scalability

The advantages of the bottom-up and the top-down methods in anonymization quality do not come for free. The trade-off is the longer computation time. Figure 21 shows the results on scalability. The complexity of the MultiDim method is $O(|T| \log |T|)$, lower than that of the bottom-up and the top-down methods. Thus, the MultiDim method is more scalable. However, since anonymization is typically an offline, one-time task, quality can be a more important concern than the runtime. On the other hand, the difference between the top-down method and the MultiDim method is not dramatic. In our experiments, even when the data set scales up to $100,000$ tuples, the runtime of the top-down approach is just

less than 6 times slower than that of the MultiDim method.

The top-down method is substantially faster than the bottom-up method. As analyzed in Section 4, splitting in the top-down method is much faster than merging in the bottom-up method.

A critical step in the top-down method is to choose two seed tuples. We used a heuristic method as described in Section 4. Figure 22 shows the effectiveness of the heuristic. We used a thorough method to compute the pair of tuples of the largest certainty penalty. Then, we used the heuristic method to compute seed tuples that are far away, and compare their certainty penalty with the maximum. As shown, with a small number of iterations, our heuristic gives very good approximation to the maximum. Thus, in our implementation, we conduct 3 iterations to obtain the seed tuples.

Summary

The extensive experiments using both real data sets and synthetic data sets show that, in terms of utility and discernability, the bottom-up method and the top-down method developed in this paper often achieve better anonymization in quality than the MultiDim method, the state-of-the-art approach. The top-down method is better than the bottom-up method.

The trad-off of high anonymization quality is the runtime. The MultiDim method is more efficient. However, the runtime of the top-down method is not far away from that of the MultiDim method in practice. Moreover, for anonymization, the computation time is often a secondary consideration yielding to the quality.

6. CONCLUSIONS

As privacy becomes a more and more serious concern in applications involving microdata, good anonymization is important. In this paper, we showed that global recoding, which is often used in previous methods, may not achieve effective anonymization in terms of discernability and query answering accuracy. Moreover, the utility of attributes has not been considered in the previous methods. Consequently, we study the problem of *utility-based anonymization*. A simple framework was given to specify utility of attributes, and two simple yet efficient heuristic local recoding methods for utility-based anonymization were developed. Our extensive performance study using both real data sets and synthetic data sets shows that our methods outperform the state-of-the-art multidimensional global recoding methods in both discernability and query answering accuracy. Furthermore, our utility-based method can boost the quality of analysis using the anonymized data.

Acknowledgement

We sincerely thank the reviewers for their very careful and constructive comments.

7. REFERENCES

[1] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 901–909. VLDB Endowment, 2005.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, pages 246–258, 2005.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, (2005112001), 2005.

[4] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 217–228, Tokyo, Japan, April 2005.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[6] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 205–216, Washington, DC, USA, 2005. IEEE Computer Society.

[7] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 279–288, New York, NY, USA, 2002. ACM Press.

[8] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[9] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, USA, April 2006. IEEE.

[10] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'04)*, pages 223–228, New York, NY, USA, 2004. ACM Press.

[11] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[12] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the 17th ACM Symposium on the Principle of Database Systems*, Seattle, WA, June 1998.

[13] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*, 1998.

[14] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[15] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[16] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 249–256, 2004.

[17] L. Willenborg and T. deWaal. *Elements of Statistical Disclosure Control*. Lecture Notes in Statistics. Springer Verlag, 2000.

[18] W. E. Winkler. Using simulated annealing for k-anonymity. In *Technical Report Statistics 2002-7, U.S. Census Bureau, Statistical Research Division*, 2002.

Beyond Classification and Ranking: Constrained Optimization of the ROI

Lian Yan
NCO Group, Inc.
507 Prudential Road
Horsham, PA 19044
lian.yan@ncogroup.com

Patrick Baldasare
NCO Group, Inc.
507 Prudential Road
Horsham, PA 19044
patrick.baldasare@ncogroup.com

ABSTRACT

Classification has been commonly used in many data mining projects in the financial service industry. For instance, to predict collectability of accounts receivable, a binary class label is created based on whether a payment is received within a certain period. However, optimization of the classifier does not necessarily lead to maximization of return on investment (ROI), since maximization of the true positive rate is often different from maximization of the collectable amount which determines the ROI under a fixed budget constraint. The typical cost sensitive learning does not solve this problem either since it involves an unknown opportunity cost due to the budget constraint. Learning the ranks of collectable amount would ultimately solve the problem, but it tries to tackle an unnecessarily difficult problem and often results in poorer results for our specific target. We propose a new algorithm that uses gradient descent to directly optimize the related monetary measure under the budget constraint and thus maximizes the ROI. By comparison with several classification, regression, and ranking algorithms, we demonstrate the new algorithm's substantial improvement of the financial impact on our clients in the financial service industry. The proposed algorithm can also be applied to several other areas such as maximizing average returns of stock selection and identifying tax auditing targets of highest values.

Categories and Subject Descriptors

H.4 [**Database Management**]: Database Applications - Data Mining; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology - classifier design and evaluation

General Terms

Algorithms

Keywords

return on investment, neural networks, constrained optimization

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION

Classification has been commonly used in many data mining projects in the financial service industry. We have used a classifier to predict defection of mutual fund accounts for a major US mutual fund company [9], where the positive samples are defined as those accounts with a net redemption amount (redemption minus purchase) of 35% or more of the account balance within a two-month window. We set up a control group for the project to evaluate the model's accuracy. Table 1 shows the real-world evaluation results for the control group over a four-month window, which consists of two levels of defection risk and three segments based on account values.

	Higher risk	
	Def. rate	Avg. net redem.
Segment 1	4.8%	-$5,145
Segment 2	10.5%	$14,494
Segment 3	5.7%	$2,733
	Lower risk	
	Def. rate	Avg. net redem.
Segment 1	1.9%	$2,494
Segment 2	1.6%	$13,864
Segment 3	2.4%	$2,686

Table 1: Defection rate and average net redemption amount for the control group for a US mutual fund company.

We can see that the model was successful at predicting defecting accounts as evidenced by the higher defection rate in the higher risk groups for all the three segments. However, the average net redemption amounts in the higher risk groups were *not* significantly higher than those in the lower risk groups. Especially, for Segment 1, even though the higher risk group had a much higher defection rate than the lower risk group, the negative net redemption amount in the higher risk group indicates a positive net purchase. This model can be used to reduce defection rate, but it would not be the best model used to prevent the highest redemption amount. For a fixed budget, the return on investment (ROI) of the project is determined by the amount of redemptions prevented (rather than by the reduction of defection rate). There are many significant factors other than the model affecting the retained amount, but simply classifying the accounts as defection or non-defection does not enable

the mutual fund company to reach out to those accounts with the highest redemption amount.

As another example, a classifier can be used to predict collectability of delinquent accounts receivable for credit card issuers using credit, demographic, and account data, where a binary class label is created based on whether a payment for an account is received within a certain period since the account was placed into the collection process. Typically, the budget restricts how many accounts can be placed into a specific collection process. While the true positive rate among those accounts in the collection process is a meaningful measure of classification accuracy, maximization of the true positive rate is often different from maximization of the collectable amount for the specific collection process. It is the collection amount rather than the true positive rate that determines the ROI under the fixed budget.

Note that we are always addressing a budget constraint, which determines, among other things, how many mutual fund accounts the customer service team can reach out every month and how many accounts receivable can be placed into a specific collection process. In our applications we represent the budget constraint by the pull rate r which is the percentage of accounts to pull out for a specific intervention/collection process. Let us denote x as the target monetary measure, e.g., collection amount, which directly determines the ROI. Then the goal is to find a function $y(e)$, where e is the independent variables such as credit, demographic, and account data, so that the accounts in the top $r\%$ by y correspond to those in the top $r\%$ by the target x. Thus the problem of maximizing the ROI can be formally defined as

$$\text{Max} \sum_{\text{y(e}_\text{i}\text{)}\in\text{Top r\%}} \text{x}_\text{i}, \tag{1}$$

where $i = 0, 1, \ldots, n-1$, and n is the total number of accounts.

1.1 Related work

One might immediately suggest cost-sensitive learning, e.g., [4], and ranking, e.g., [2], [3], would solve the above problem. For cost-sensitive learning, we have the cost matrix in Table 2. Assuming $c_{00} = 0$ and $c_{11} = 0$, a typical sensitive learning algorithm tries to minimize the cost

$$C = \sum_{i\in P}(1-q_i)c_{01} + \sum_{j\in N} q_j c_{10} \tag{2}$$

over the training set, where P, N are the sets of positive and negative samples, respectively, and q_i, q_j are both posterior probabilities of belonging to the positive class. In our application, the actual positives are those accounts which are in the top $r\%$ of x, and predicted positives are those accounts which have a score y in the top $r\%$. It is straightforward that $c_{01} = x$, since if an actual positive is placed out of the top $r\%$ (a predicted negative), the company will not be able to collect $\$x$ or retain the net redemption of $\$x$. If an actual negative is placed among the top $r\%$, the company will lose the opportunity to reach out to one of the accounts with a *larger* x in the top $r\%$, since the number of accounts to be contacted is pre-determined by the pull rate r. Thus, c_{10} is an opportunity cost that is not a constant and unknown. One might still try to train a classifier with sample weights intuitively based on x. In Sections 3 and 4, we compare our algorithm's results with such a classifier's.

	actual negative	actual positive
predicted negative	c_{00}	c_{01}
predicted positive	c_{10}	c_{11}

Table 2: A cost matrix for cost-sensitive learning.

If we can learn a regression model so that $y(e_i) = x_i$, $i = 0, \ldots, n-1$, or a ranking model so that $y(e_i) > y(e_j)$ for any $(i,j) \in \{(i,j)|x_i > x_j, i,j = 0,\ldots,n-1\}$, $\sum_{y(e_i)\in\text{Top r\%}} x_i$ would be optimized for each r. However, both regression and ranking try to solve an unnecessarily difficult problem, and often lead to poorer results for our specific target at pull rate r. Maximization of $\sum_{y(e_i)\in\text{Top r\%}} x_i$ requires only the correct ranking between the Top $r\%$ and the others. The ranking *within* the Top $r\%$ or the others is not necessary, neither is the estimate of x itself by regression. In Sections 3 and 4, we compare our model with a regression and a ranking model, which uses the algorithm in [2].

We present the new algorithm in the next section, where we also describe the several classification, regression, and ranking algorithms which we compare with in our projects. In Section 3, we use the proposed algorithm to predict collectibility of accounts receivable for delinquent consumer loan accounts from several US financial institutions. In Section 4, the new algorithm is applied to predicting defection of mutual fund accounts for a major US mutual fund company. Finally, we discuss several algorithmic and applied extensions of the proposed algorithm in Section 5.

2. CONSTRAINED OPTIMIZATION OF THE ROI

For a model with $0 \le y \le 1$[1], assume that the specified pull rate r can be achieved at a decision threshold β ($0 < \beta < 1$), i.e., the accounts in the pull are those with an output larger than β. In this case, maximization of $\sum_{y_i\in\text{Top r\%}} x_i$ can be solved by the following constrained optimization over y_i, $i = 0, \ldots, n-1$, and β:

$$\text{Max} \sum_{\text{i=0}}^{\text{n-1}} \text{x}_\text{i} \cdot \text{I(y}_\text{i}, \beta), \tag{3}$$

subject to

$$\frac{\sum_{i=0}^{n-1} I(y_i,\beta)}{n} = r, \tag{4}$$

where

$$I(y_i,\beta) = \begin{cases} 1 & : \quad y_i > \beta \\ 0 & : \quad \text{otherwise} \end{cases} . \tag{5}$$

When the constraint $\frac{\sum_{i=0}^{n-1} I(y_i,\beta)}{n} = r$ is satisfied, the number of accounts with the model output $y_i > \beta$ will be exactly $r\%$ of n. The difficulty here is that $I(y_i,\beta)$ is non-differentiable, and gradient based optimization cannot be used to optimize Eq. 8.

In [8], [9], we demonstrate that the sigmoid function

$$\sigma(y_i,\beta) = \frac{1}{1+e^{-\kappa(y_i-\beta)}}, \tag{6}$$

where $\kappa > 0$, does not provide a good differentiable approximation to $I(y_i,\beta)$ when $-1 \le y_i - \beta \le 1$. Instead, we have

[1]For simplicity, we'll omit the independent variable e and use y_i for $y(e_i)$.

proposed the following differentiable approximation

$$f(y_i, \beta+\gamma) = \begin{cases} (y_i - \beta - \gamma)^p & : \quad y_i > \beta + \gamma \\ 0 & : \quad \text{otherwise} \end{cases}, \qquad (7)$$

where $p > 1$ and $0 \leq \gamma < 1$. A small but positive γ is often helpful for a better generalization performance over the test set. Now Eq. 3 becomes

$$\text{Max} \sum_{\text{i=0}}^{\text{n-1}} \text{x}_\text{i} \cdot \text{f}(\text{y}_\text{i}, \beta+\gamma). \qquad (8)$$

However, $\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{n}$ is not a good approximation to $\frac{\sum_{i=0}^{n-1} I(y_i,\beta)}{n}$, since $f(y_i, \beta)$ is often not close to 1. As in [9], rather than trying to use a differentiable approximation to r, we approximate a related ratio $\frac{r}{1-r}$ by the following differentiable function:

$$\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{\sum_{i=0}^{n-1} g(y_i,\beta)}, \qquad (9)$$

where

$$g(y_i, \beta) = \begin{cases} (\beta - y_i)^p & : \quad y_i < \beta \\ 0 & : \quad \text{otherwise} \end{cases} \qquad (10)$$

with $p > 1$. $g(y_i, \beta)$ is a differentiable approximation to the following step function

$$I_p(y_i, \beta) = \begin{cases} 1 & : \quad y_i < \beta \\ 0 & : \quad \text{otherwise} \end{cases}. \qquad (11)$$

Since the optimization often moves most y_i close to β in the end, we will see that Eq. 9 can provide a close approximation to $\frac{r}{1-r}$.

Now we convert the constrained optimization into an unconstrained optimization problem by minimizing the following Lagrangian:

$$L = -\frac{1}{n}\sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta+\gamma) + \frac{1}{\mu}\left(\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{\sum_{i=0}^{n-1} g(y_i,\beta)} - \frac{r}{1-r}\right)^2. \qquad (12)$$

During the training iterations, μ is gradually decreased until convergence of the constraint $(\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{\sum_{i=0}^{n-1} g(y_i,\beta)} - \frac{r}{1-r})^2$ is achieved. In practice, we have found that mapping x_i in Eq. 12 to a value between -1 and 1 by

$$\theta(x_i) = \frac{1 - e^{-x_i}}{1 + e^{-x_i}} \qquad (13)$$

typically obtains improved results.

In the Appendix, we derive the derivatives for y_i. These derivatives together with the chain rule can then be applied to any parametric model, for which one can optimize the differentiable objective function with respect to the parameters using gradient based methods.[2] In our projects, we apply the proposed algorithms to a typical multilayer perceptron (MLP) network with softmax outputs between 0 and 1, and with a single hidden layer and direct connection between the input and output layers. β can also be optimized with the model parameters, but we have found that fixing β at 0.5 achieves almost the same results over our data sets.

[2]We use the limited memory BFGS method in [6].

2.1 Comparing methods

In Sections 3 and 4, we apply the new algorithm to predicting collectibility of accounts receivable and predicting defection of mutual fund accounts, and compare the results of the proposed algorithm with the following four algorithms'.

- *Classification* An ensemble of MLP classifiers is trained by mean squared error based on the defined class label. Since the class prior is typically low, each individual classifier in the ensemble has a modified prior to compensate for the imbalanced data sets [10].

- *Weighted classification* An MLP classifier is trained by mean squared error based on either the defined class label, e.g., whether the net redemption amount is above or below 35% of the account balance, or the ranks of the training samples. Using the ranks to determine the class label is an intuitive idea: labeling those samples in the top r% of x as positive and the others as negative. When r is the same as the prior of the defined class, these two approaches are the same. During training, the samples are weighted by x or a function of x. To avoid the dominance of those samples with an extreme value of x, we typically use the sigmoid function of x to smooth out the weights.

- *Ranking* Burges et al. propose a ranking algorithm using gradient descent [2]. We apply this algorithm to train an MLP model which ranks x in our applications. The algorithm tries to minimize the cross entropy function

$$\sum_{(i,j)\in S} -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}), \qquad (14)$$

where $S = \{(i,j) | x_i \geq x_j, i, j = 0, \ldots, n-1\}$, and $\bar{P}_{ij}$ is the target probability of $x_i > x_j$. P_{ij} is the model's estimate of $\bar{P}_{ij}$ in the form $P_{ij} = \frac{e^{y_i - y_j}}{1+e^{y_i - y_j}}$. Then the cost function becomes

$$\sum_{(i,j)\in S} -\bar{P}_{ij}(y_i - y_j) + \log(1 + e^{y_i - y_j}). \qquad (15)$$

In our experiments, we choose $\bar{P}_{ij} = 1$ if $x_i > x_j$ and $\bar{P}_{ij} = 0.5$ if $x_i = x_j$.

- *Regression* An MLP regressor is trained by mean squared error against x. We map x to a value between 0 and 1 using the sigmoid function.

3. PREDICTING COLLECTIBILITY OF ACCOUNTS RECEIVABLE

Accounts receivable are unpaid customer invoices, and any other money owed to a company by its customers. From credit card issuers to banks, from local retail stores and service businesses, to the federal, state and local governments, if the business or government unit extends credit, offers payment installment plans, or makes assessments, it has accounts receivable. The collection industry serves an important role in the U.S. economy by recovering billions in revenue from charged-off or delinquent accounts receivable for U.S. companies. By returning this money to U.S. companies, the collection industry saves American families

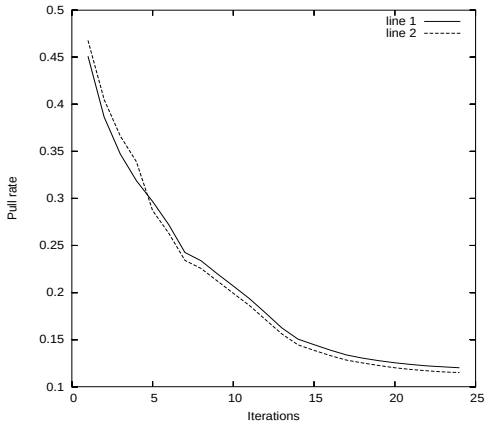


Figure 1: This figure shows convergence of pull rates achieved by the threshold β during the optimization. Line 1 is for the training set, and Line 2 shows the pull rate change over the test set.

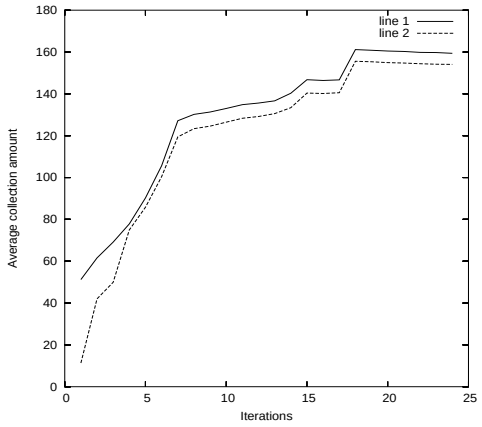


Figure 2: This figure shows the improving average collection amount among the top 11% accounts during the optimization. Line 1 is for the training set, and Line 2 is over the test set.

on average $331 a year in money they otherwise would have spent if businesses raised their prices to cover losses to bad debt [1].

The portfolio of accounts receivable we worked on consists of consumer loan accounts from several US financial service institutions. The portfolio includes several types of accounts in terms of account history. For example, some are the so-called Prime accounts which are newly charged off accounts, and some are Seconds which had already gone through a collection process. Our goal is to develop a generic predictive model which can be used to guide the agents' collection efforts. In particular, we would like to identify a high value segment which consists of 11% of the whole portfolio. The 11% is chosen since the payer rate (percentage of paid accounts in the first six months) is 11%. It is clear that the return on investment is determined by the collection amount from the identified 11% accounts in the segment.

The data set includes 684,600 accounts. We randomly split the data set into a training set and a test set of equal size. In addition to the account history and general demographic information, several hundred data fields from a credit score provider about the account owner are also available. The domain experts guided the feature selection, and 30 data fields are used in the final model.[3] Missing values for continuous variables are simply imputed by the mean with an added binary column indicating missingness for this variable. Most of the data fields are categorical. For categorical variables with missing values, the sets of distinct values are augmented by another value 'missing'. We encode the categorical variable $C = \{c_1, c_2, \ldots, c_k\}$ by replacing c_i with the conditional mean $E(x|c_i)$ and conditional standard deviation $\sigma(x|c_i)$, $i = 1, 2, \ldots, k$.

We set $r = 11\%$ and fix β at 0.5 for the new algorithm, trying to maximize the average collection amount among the top 11% accounts. We choose γ and p in Eq. 12 so that the number of training samples with the model output $y > \beta$ is close to $r\%$ and the average collection amount among the top $r\%$ in the training set is the largest. Here we chose $\gamma = 0.01$ and $p = 2$ [4], and the number of hidden units is 5.[5] In Figures 1 and 2, we show the optimization process along with iterations of μ, which is initialized at 100, and is updated by $\mu_{t+1} = 0.75\mu_t$ during the optimization, where t is the iteration index. Figure 1 shows that the optimization converged when the number of training samples with $y > \beta$ reached 12%, which is quite close to the target pull rate 11%. We can see that the pull rate over the test set, achieved by the same threshold β, is also very close to the target 11%. This demonstrates that Eq. 9 provides a good approximation to $\frac{r}{1-r}$. In Figure 2, with the iterations, a steadily improving average collection amount among the top 11% is observed for both the training and test sets. We rarely observe obvious overfitting, and this justifies the use of the training set to choose γ and p.

Table 3 presents the average collection amount in the top 11% accounts over the test set for five different models. The classification model is an ensemble of 25 MLP networks with a modified class prior between 0.02 and 0.5 [10]. For the model of weighted classification, during training the samples are weighted by $\sigma(x) = \frac{1}{1+e^{-x}}$, which is also the target variable for the regression model. For ranking, most accounts (89%) have a tied collection amount of zero. We can see that the new algorithm is clearly exceeding all other algorithms. Comparing with the classification model, the ROI is improved by 25%. Note that the average collection amount over the whole portfolio is $36 only.

new model	class.	weighted class.	ranking	regress.
$157	$126	$106	$61	$116

Table 3: The average collection amount in the top 11% accounts over the test set for five different models.

[3]While we are still working on several feature selection algorithms trying to reveal more useful data features, up to now we have only achieved marginal improvement by adding more data fields.

[4]In some cases, by choosing different p values in Eq. 8 and Eq. 9, better results over the training set can be achieved.

[5]We have observed that the number of hidden units, varying from 0 to 10, does not have a significant effect on the results over our large data sets. Therefore all the MLP structures in the paper have 5 hidden units.

4. PREDICTING DEFECTION OF MUTUAL FUND ACCOUNTS

Worldwide the mutual fund industry houses 15 trillion US dollars – about 8 trillion from US investors and another 7 trillion from investors in other countries. Today, the US mutual fund industry holds about 18% of all households' financial assets and about 22% of all outstanding US corporate stock [5]. However, in the end of 2003 the industry wide redemption rate stood at 24.2%, implying that the investor base completely turns over in 4 (1/0.242) years. To illustrate the magnitude of redemptions in the mutual fund industry, the Investment Company Institute estimated that in 2003 1.086 trillion new dollars flowed into equity funds but, over the exact same measurement interval, 934 billion (86%) flowed back out [5]. The costs associated with keeping track of this flowing river of money, adding and deleting client information to databases, filing required tax forms with federal, state and local taxing authorities as well as simply cutting checks to redeeming clients is an enormous drain on any funds' expense ratio, not mentioning the revenue drop of fund companies because of the decreased assets under management (AUM) due to redemption. In recent years, more and more mutual fund companies have recognized the importance of early identification of investors at risk of redeeming their assets (i.e., defectors), so that proactive client service and educational programs could be initiated to "plug" the outflow of assets.

We have developed a model to predict account defection for a major US mutual fund company. In order to provide early identification of defectors, there is a two month gap between the end of the independent variable (IV) window and the beginning of the two-month dependent variable (DV) window. For example, at the end of February, we would like to predict which accounts will defect in the time period of May and June. The two-month leading time allows the mutual fund company to act on the predicted potential defectors in March and April. For classification purpose, a defector is defined by the domain experts as an account which had a net redemption amount (redemption minus purchase) of at least 35% of the account balance in a two month window. As the training set, we received about 184,000 accounts, each of which had an account balance of at least $100,000. For training, the IV window is a one-year period ending on May 31, and the DV window is a two-month period of August and September. Based on the definition of defection, the defection rate is below 1% in the two month window. Regardless of the defection definition, the average net redemption amount in the two months over the whole training set was about -$3,000, where the negative sign means that, on average, the account balance had a net increase. We used a forward time-shifted test set of around 434,000 accounts, which had the one-year IV window ending on September 30 and the DV window consisting of December and January.

The data for each account is a mixture of continuous and categorical variables, including basic account information, asset data, transactions, demographic information, benchmark performance data, and customer service records. There are about 2,000 raw data fields, but the final model uses 123 data fields after conducting feature selection and time series transformation [9]. The mutual fund company set $r = 10\%$ based on the predetermined budget. We will discuss the savability issue in the next section. Until then let us assume that the return on investment is primarily dependent on the net redemption amount identified among the top 10%.

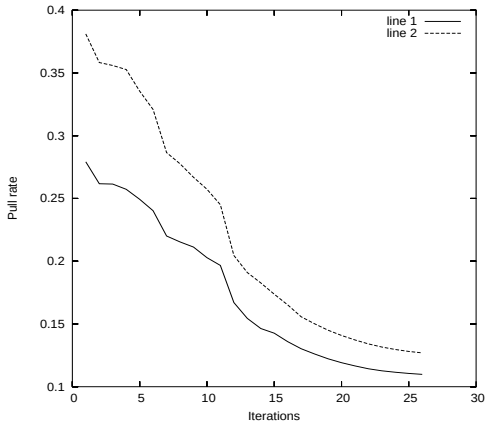


Figure 3: This figure shows convergence of pull rates achieved by the threshold β during the optimization. Line 1 is for the training set, and Line 2 shows the pull rate change over the test set.

Again we fix β at 0.5 for the new algorithm and try to maximize the average net redemption amount in the top 10%. We chose $\gamma = 0$ and $p = 2$ since, with these parameters, the number of training samples with $y > \beta$ is close to 10% and the average collection amount among the top 10% in the training set is the largest. We have seen that these two goals are often quite consistent, i.e., when a set of parameters results in the largest average collection amount, it also brings the number of training samples with $y > \beta$ close to the target pull rate. We show the optimization process along with iterations of μ in Figures 3 and 4. Again we can see that the optimization converged when the number of training samples with $y > \beta$ reached 11%, which is quite close to the target pull rate 10%. Over the test set, the number of samples with $y > \beta$ reached 12%, 2% higher than the target rate. Figure 4 shows a quite large difference of the average net redemption amount between the training and test sets. This is due to the overall net redemption is changed in the test set's DV window, which is four months apart from the training set DV window. The average net redemption amount over all the accounts is now about -$3,400, comparing with -$670 over the training set.

In Table 4, the classification model is an ensemble of 25 MLP networks trained with modified priors based on the defection definition of 35% or more redemption. However, the weighted classification model is trained by class labels based on the ranking, i.e., the samples with the top 10% of net redemption amount are positives and the others are negatives. The training samples are weighted by $\frac{1}{1+e^{-|x|}}$, which gives larger weights to samples with a larger (positive or negative) net redemption amount. The regression model is trained against $\sigma(x) = \frac{1}{1+e^{-x}}$. Note that in Table 4 a negative net redemption amount means a positive net purchase. Though the classification model achieves a 39% true positive rate (the new model has a true positive rate of 14%), it cannot effectively identify those accounts with the highest redemption amount.

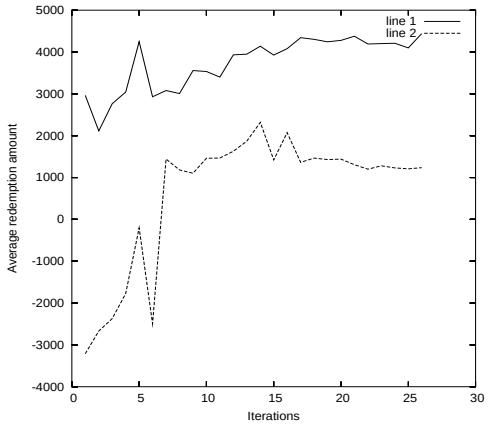


Figure 4: This figure shows the average net redemption amount change among the top 10% accounts during the optimization. Line 1 is for the training set, and Line 2 is over the test set.

new model	class.	weighted class.	ranking	regress.
$1,236	-$6,723	-$18,968	-$9,923	$671

Table 4: The average net redemption amount in the top 10% accounts over the test set for five different models.

5. DISCUSSIONS

We have proposed a new learning algorithm which focuses on maximizing the monetary measure under a fixed budget constraint. The two applications demonstrate the substantial improvement of financial impact by the new algorithm. In this section, we discuss several related practical issues and some algorithmic and application extensions.

- *Savability* There is no doubt that maximizing the collection amount for the top r% accounts also maximizes the ROI of the collection efforts which reach out to a predetermined r%. However, it is arguable that maximizing the net redemption amount in the top r% accounts would maximize the *retained* redemption amount and thus the ROI. Another important factor determining the ROI is the savability of the predicted net redemption amount. It is reasonable to assume that it might be more difficult to retain a substantial redemption amount of a large account, since the redemption rate against the balance might be insignificant and the redemption is just some normal cash flow activity. To model 'savability' directly appears not feasible since 'being savable' is not observable and cannot be defined correctly. However, we have tried to model savability from the other aspect – 'being unsavable', which can be partially defined, i.e., if an account was predicted to defect and was contacted for retention but still became defected, this account was unsavable. Here we implicitly assume retention efforts do not cause originally non-defecting accounts to defect. We have developed such a savability model for a US mutual fund company. However, we have not been able to combine the savability model with scores from the model by the new algorithm in a principled way, since the score is only a ranking indicator rather than a probabilistic estimate of x. An empirical way to consider savability is to increase the predetermined r by a certain percentage, and to exclude those accounts with a savability score below an empirically determined threshold.

- *Valuable false positives* The model trained using the new algorithm does not classify accounts into positive and negative samples defined separately, e.g., by the redemption rate of 35%. We have observed that the model achieving the highest average net redemption amount can have a very low true positive rate based on the defection definition. Some companies will not feel comfortable to see a low true positive rate based on the defection definition given by their domain experts. It would be most desirable to achieve both a high true positive rate *and* a higher average redemption amount among the false positives. We call these false positives valuable false positives since they may have substantial net redemption too. We have tried to simply add an item, which approximates the true positive rate in [9], into Eq. 12 and to minimize the Lagrangian:

$$L = - \frac{1}{n}\sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta+\gamma) - \frac{1}{m}\sum_{i\in P} f(y_i, \beta+\gamma) + \frac{1}{\mu}\left(\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{\sum_{i=0}^{n-1} g(y_i,\beta)} - \frac{r}{1-r}\right)^2, \quad (16)$$

  where $m = |P|$, the number of positive samples. However, this intuitive approach does not appear to work well.

- *No budget constraint* In some cases, there is no fixed budget constraint and r is not predetermined. For example, for the collection industry, the goal might be loosely stated as collecting as much as possible by contacting as less accounts as possible. For this goal one might be tempted to minimize the following Lagrangian:

$$L = -\frac{1}{n}\sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta+\gamma) + \frac{1}{\mu}\frac{\sum_{i=0}^{n-1} f(y_i,\beta)}{\sum_{i=0}^{n-1} g(y_i,\beta)}. \quad (17)$$

  This approach does not work since it always tries to get to a contact rate close to zero. In theory, the maximum profit or ROI is achieved when the marginal collection cost equals to marginal revenue (collection amount). Typically, we can assume the marginal collection cost is a constant. By searching over different r values, for each of which a model needs to be trained by minimizing Eq. 12, the optimum r can be found so that the marginal collection cost is equal to the marginal revenue and the ROI is maximized.

- *Other applications* The new algorithm can also be applied to several other areas. For example, maximizing average returns of stock selection, identifying tax auditing targets of highest values, and identifying fundraising targets with the highest contributions – all these tasks involve a predetermined budget and only concern the related average monetary value in the top r% determined by the budget. Even in the typical customer relationship management area, e.g., churn prediction for wireless service providers [10], since the

ultimate concern is the loss of revenue due to service disconnection, we can apply this algorithm to identify those accounts with the highest revenue losses. It would be interesting to compare this approach to another approach which combines an estimate of customer value with a predicted churn probability [7].

6. ACKNOWLEDGMENTS

The authors thank Bob Ruff, Vladimir Gold, and William Hastings at NCO Group, Inc. for helpful discussions and/or data preparation. This is a slightly revised version of our paper with the same title appearing in the Proceedings of KDD'06. The authors thank the workshop organizers for the invitation of this paper.

7. REFERENCES

[1] Association of Credit and Collection Professionals. Industry Overview. http://www.acainternational.org/about.aspx?cid=320.

[2] C. Burges, T. Shaked, et al. Learning to rank using gradient descent. In *Proc. of the 22nd Intl. Conf. on Machine Learning*, 2005.

[3] R. Caruana, S. Baluja, and T. Mitchell. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. *Advances in Neural Information Processing Systems*, 8, pp. 959-965, 1996.

[4] C. Elkan. The foundations of cost-sensitive learning. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence*, 2001.

[5] Investment Company Institute. 2004 Mutual Fund Factbook. http://www.ici.org/stats/latest/12004_factbook.pdf, May 2004.

[6] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45, pp. 503-528, 1989.

[7] L. Yan and P. Baldasare. Optimizing customer value in a decision theoretic framework. In *Proc. of 2005 Intl. Workshop on CRM: Data Mining Meets Marketing*, New York, 2005.

[8] L. Yan, R. Dodier, M. Mozer, and R. Wolniewicz. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *Proc. of the 20th Intl. Conf. on Machine Learning*, pp. 848-855, 2003.

[9] L. Yan, M. Fassion, P. Baldasare. Enhancing the lift under budget constraints: an application in the mutual fund industry. In *Proc. of the 11th Intl. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 509-515, 2005.

[10] L. Yan, R. Wolniewicz, and R. Dodier. Customer behavior prediction in telecommunications. *IEEE Intelligent Systems*, March/April 2004.

APPENDIX

Derivatives for minimizing Eq. 12

The following partial derivatives with the chain rule together form the basis to minimize Eq. 12 against the model parameters, e.g., the weights of an MLP structure.

Let

$$F = \sum_{i=0}^{n-1} f(y_i, \beta) \tag{18}$$

and

$$G = \sum_{i=0}^{n-1} g(y_i, \beta). \tag{19}$$

For $i \in \{i | f(y_i, \beta+\gamma) \leq 0, i = 0, \ldots, n-1\}$, we have $\frac{\partial L}{\partial y_i} = 0$. For $i \in \{i | f(y_i, \beta+\gamma) > 0, i = 0, \ldots, n-1\}$, we have the following two cases:

- When $f(y_i, \beta) > 0$,

$$\begin{aligned} \frac{\partial L}{\partial y_i} = & - \frac{p}{n} x_i (y_i - \beta - \gamma)^{p-1} \\ & + \frac{2p}{\mu} (\frac{F}{G} - \frac{r}{1-r}) \frac{1}{G} (y_i - \beta)^{p-1}. \end{aligned} \tag{20}$$

- When $g(y_i, \beta) > 0$,

$$\begin{aligned} \frac{\partial L}{\partial y_i} = & - \frac{p}{n} x_i (y_i - \beta - \gamma)^{p-1} \\ & + \frac{2p}{\mu} (\frac{F}{G} - \frac{r}{1-r}) \frac{F}{G^2} (\beta - y_i)^{p-1}. \end{aligned} \tag{21}$$

Pricing Based Framework for Benefit Scoring

Nitesh Chawla
University of Notre Dame
Notre Dame, IN 46556
nchawla@nd.edu

Xiangning Li
University of Notre Dame
Notre Dame, IN 46556
xli3@nd.edu

ABSTRACT

Data mining models often carry the final objective of maximizing profit or minimizing cost. This problem becomes even more profound in financial applications that can have multiple constraints, such as interest rate, score cut-off, and the loan amount to allocate. In this paper, we present a pricing framework for discovering the total profit from a probabilistic model, given a benefit function.

1. INTRODUCTION

An important application of data mining in the financial industry is "scoring" the customers for loans. Credit scoring methods, typically, apply a cut-off paradigm of accepting or declining potential customers. The cut-off score is derived from a model learned on past consumer characteristics, and influences not only the accept or decline decision but also the loan amount and accompanying interest rate. Our goal is to evolve the scoring decision with a pricing scheme that drives the overall utility of the model. The utility is defined in terms of the economic benefit or the profit from approving loan for a customer. We want to able to structure the loan amount and the interest rate for a customer based on the propensity to default.

Our work builds upon the cost-sensitive learning literature [5, 11, 4, 7, 12] and the relevant literature from finance [10]. Stein [10] extends a cut-off score based approach to a pricing approach resulting in a more flexible and profitable model. Using the ROC curve quantities, he formulates the net present value of taking a lending decision and the corresponding benefit of a true negative.

We formulate a pricing framework based on the probability of default assigned by a scoring model and a benefit matrix. The benefit matrix specifies the benefit (positive or negative) from making a prediction. For example, a true negative will result in higher benefit as it is a positive return on the investment (a non-defaulting customer is correctly predicted to be a non-default). The profit from a customer is conditioned on the propensity to default (or not default).

UBDM'06 August 20 2006, Philadelphia, Pennsylvania, USA

Table 1: Benefit Matrix

	Actual Non-default	Actual Default
Predict Non-default	b_{00}	b_{01}
Predict Default	b_{10}	b_{11}

We want to be able to construct a pricing scheme that compensates the risk of the customer, and accordingly proposes an interest rate and a loan amount. Thus, we propose the interest rate and loan amount that can be assigned to an approved customer. We illustrate the workings of the pricing scheme using case dependent benefit functions. We discuss that the calibration of the model and the resulting quality of the estimates is more important than the resulting rank-ordering. We hasten to point out that, while this work serves as a preliminary proof-of-concept, our ongoing goal with the work is to demonstrate the applicability of the proposed approach using multiple simulations. We want to eventually compare multiple models using ROC curves, probability loss functions, and profits in dollars.

2. PRICING SCHEME

The key utility of applying data mining in a business model is the objective of maximizing profit or minimizing cost. The profit is related to the accuracy of the default probability predicted by a model and the case dependent benefit function. A typical benefit matrix can be defined as in Table 1 [5]. The benefit matrix elements reflect the benefits from assigning the loan, as per the corresponding probability of default. Morever, we assume that the benefits can be different for individual customers as they will be conditioned on the loan amount and the interest rate. The benefits should be tunable for the different levels of risk as reflected by the probability of default. This can be achieved by generating a different pricing function. Furthermore, we will assume that a customer, k, is accompanied with a probability of default of P_k. In our empirical analysis, we will assume that $b_{10} = b_{11} = 0$, since there is no benefit or cost from not offering a loan to customer.

Then, the profit from predicting a customer as a non-default (ND) (making the loan offer), can be calculated as:

$$B_{ND} = (1 - P_k)b_{00} + P_k b_{01} \quad (1)$$

and the benefit from predicting the customer as default (declining the application) can be calculated as:

$$B_D = (1 - P_k)b_{10} + P_k b_{11}. \quad (2)$$

To result in a higher profit from issuing the loan than refusing it, we should ideally have $B_{ND} > B_D$. This will form the restricting condition for allocating loan to a customer, and the basis of our pricing scheme. In a rigid competitive environment, it is being able to price and accept the B_D customers, gives the critical advantage.

2.1 Making Non-default Prediction Optimal

An ideal pricing scheme will be governed by the condition that no customer is ever turned down. Rather, a loan product is always priced for the customer, in conjunction with the interest rate, such that the resulting decision leads to a higher profit. That is the loss given default should not exceed the benefit ensuing from granting the loan.

The benefits are obviously example dependent. They are functions of the loan amount (x). The benefits (or losses) for true negatives and false negatives can be defined as a function of x: $b_{00}(x)$ and $b_{01}(x)$. The benefits for false positives and true positives typically are regarded as constants: b_{10} and b_{11}. Since we want $B_{ND} > B_D$, for an optimal non-default prediction, the price $b_{00}(k, x)$ asked for a randomly chosen customer scoring $S(k)$, should be

$$(1 - P_k)b_{00}(k, x) > (1 - P_k)b_{10} + P_k b_{11} - P_k b_{01}(x). \quad (3)$$

The above formulation, arising from equations 1 and 2, compensates for the default risk — the benefit from accepting a customer should be greater than the chance of losing due to the default. Hence, to grant the loan of x for a customer k, the resulting benefit should be:

$$b_{00}(k, x) > \frac{(1 - P_k)b_{10} + P_k b_{11} - P_k b_{01}(x)}{(1 - P_k)}. \quad (4)$$

To do further analysis, we made some simple assumptions: we suppose that the benefit from a non-defaulting customer (true negative) is a function of the loan amount x, the interest rate r and the risk premium $g(x)$ ($b_{00}(x) = g(x) + rx$), particularly we assume that $g(x) = ax$, here a is a very small constant (for example, $a = 0.005$)); while the loss from a false negative is $b01 = cx$, where c is also a constant and $-1 < c < 0$, and a reasonable guess for the value of c could be -0.35 [10]. Then, the interest rate asked for a specific customer can be derived from equation 4 as follows

$$r > \frac{(1 - P_k)b_{10} + P_k b_{11} - P_k cx}{(1 - P_k)x} - a, (r \geq 0) \quad (5)$$

This above equation is a limiting condition for making the loan offer an optimal choice. Figure 1 shows the interest rate asked for the customers to appropriately compensate the default risk that the bank has taken with the assumptions: $b_{10} = b_{11} = 0$, $c = -0.35$, and $a = 0.005$. This result indicates that for any customer we could generate the interest level corresponding to the default rate. But for high default rate customers, we find out that the interest rate is too high to be realistic, so a limit for the interest rate becomes necessary. Now if we define the maximum interest rate r_{max} allowed in the real practices, then the interest rate asked for any customer should satisfying

$$r \leq r_{max}. \quad (6)$$

Note that r_{max} can be fixed prior, depending on the bank practices and the risk appetite. In practice, the following

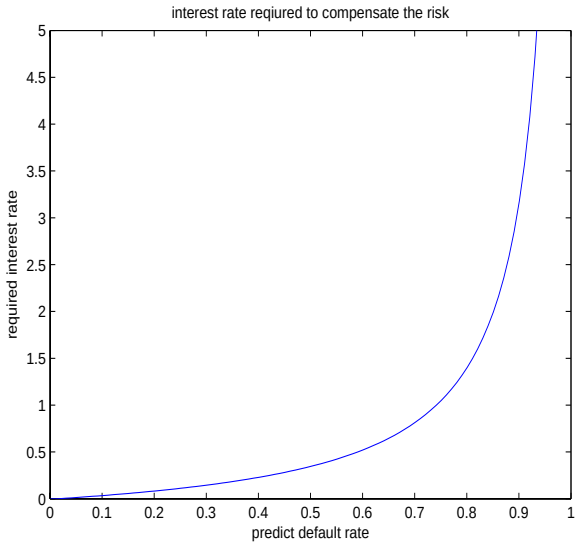


Figure 1: Interest level asked for compensating the default risk ($b_{10} = b_{11} = 0$, $c = -0.35$, and $a = 0.005$).

equation should be satisfiable:

$$r_{max} > \frac{(1 - P_k)b_{10} + P_k b_{11} - P_k cx}{(1 - P_k)x} - a, \quad (7)$$

Rearranging the above equation, we have

$$((a + r_{max})(1 - P_k) + P_k c)x > (1 - P_k)b_{10} + P_k b_{11}. \quad (8)$$

We can now define the range of the loan amount x at different values of P_k, while charging the customer at the possible maximum interest rate r_{max}.

- if $P_k > \frac{r_{max}+a}{r_{max}+a-c}$,

$$x < \frac{(1 - P_k)b_{10} + P_k b_{11}}{(r_{max} + a)(1 - P_k) + P_k c}, \quad (9)$$

- if $P_k < \frac{r_{max}+a}{r_{max}+a-c}$,

$$x > \frac{(1 - P_k)b_{10} + P_k b_{11}}{(r_{max} + a)(1 - P_k) + P_k c}. \quad (10)$$

Based on the above analysis, for a specific customer, we can decide not only the amount of money to loan but also the interest rate. Note that these values are derived using both the probability of default and the benefit matrix.

2.1.1 Making Non-negative Profit

We have discussed conditions to make non-default prediction to be optimal under various circumstances. However, these will not guarantee a positive profit, when $B_D < 0$. To ensure non-negative profit from each customer, $B_{ND} \geq 0$ is also necessary. So the true non-default prediction benefit function $b_{00}(k, x)$ should satisfy

$$(1 - P_K)b_{00}(k, x) \geq -P_k b_{01}(x). \quad (11)$$

From equation 8, we can derive the following conditions: if $P_k = 1$ (see equation 8), we should have $b_{01}(x) \geq 0$, which derives ($x \leq 0$) implying that no money should be loaned;

when $P_k = 0$, the above equation 11 is always true, so always pass the loan at a minimal interest rate; and for $0 < p_k < 1$, the required interest rate should be

$$r(k,x) \geq \frac{-P_k c}{1-P_k} - a. \quad (12)$$

Again if there exists a maximum interest rate r_{max} allowed, then it should satisfy

$$r_{max} \geq r(k,x) \geq \frac{-P_k c}{1-P_k} - a. \quad (13)$$

So, $P_k = \frac{r_{max}+a}{r_{max}+a-c}$ will lead to zero profit with r_{max}. $P_k > \frac{r_{max}+a}{r_{max}+a-c}$ will lead to negative profit even with r_{max}, thus implying that the loan request should be declined. We can gain non-negative profit, only for the following condition: $P_k < \frac{r_{max}+a}{r_{max}+a-c}$, with an interest rate $r > \frac{-P_k c}{1-P_k} - a$.

Combining the results we have now, we get the following conditions for making non-negative profit from each customer:

- For customers with $P_k = 1$, always decline the application.
- For customers with $P_k = 0$, always accept the application.
- For customers with $0 < P_k < 1$, the interest rate should satisfy equation 4 and 11 simultaneously.
  - if $\frac{r_{max}+a}{r_{max}+a-c} < P_k < 1$: results negative profit, decline the application.
  - if $P_k = \frac{r_{max}+a}{r_{max}+a-c}$: zero profit, decline or accept decision can be taken on the basis of profit from resulting underwriting charges, etc.
  - if $0 < P_k < \frac{r_{max}+a}{r_{max}+a-c}$, decide the interest rate by equation 5 and 6 for the loan, or determine the amount of money should be lent by equation 10.

We would like to point out that similar analysis can be easily conducted for any changes in benefit functions, thus becoming a useful tool for loan practices.

3. SIMULATIONS AND RESULTS

At this point, we have only conducted preliminary experiments on publicly available UCI datasets [1] — the covtype, and pima dataset. We randomly divided the dataset into 70% for training and 30% for testing. We used for different classifiers four different models: C4.5 decision trees [9] (J48), NBTree [6], Bagging with J48, and Bagging with NBtree. Our eventual goal is to utilize multiple learning algorithms with a variety of datasets to validate the proposed pricing scheme. We believe that the ROC curve analysis is limiting for pricing schemes, as it does not allow one to evaluate the quality of P_K. Hence, we also use different loss measures that capture the quality of the probability estimates. The actual total profit is a function of:

$$\sum_i (I(y = ND)b_{00}(k_i, x_i) + I(y = D)b_{01}(x_i)) \quad (14)$$

in which k_i is the corresponding percentile for the ith customer, y is the actual class of the instance (k_i, x_i), and $I(\cdot)$ is the indicator function that has value 1 in case the argument is true and 0 otherwise (ND= Non-Default, D=Default).

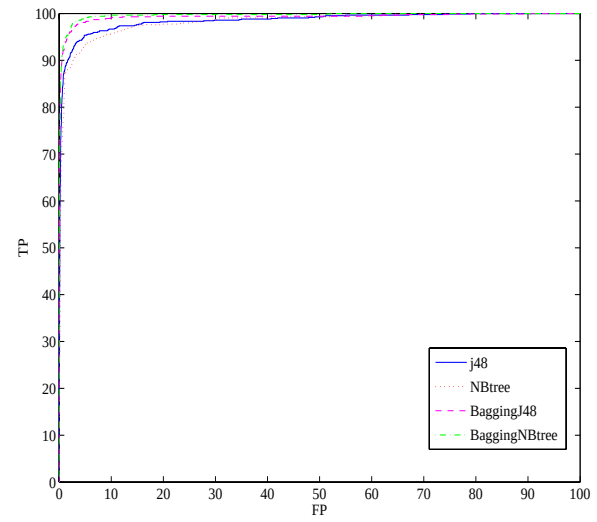


Figure 2: ROC curves for Covtype dataset from J48 (-solid), NBtree (dotted), BaggingJ48(dashed), and BaggingNBtree(dash-dotted)

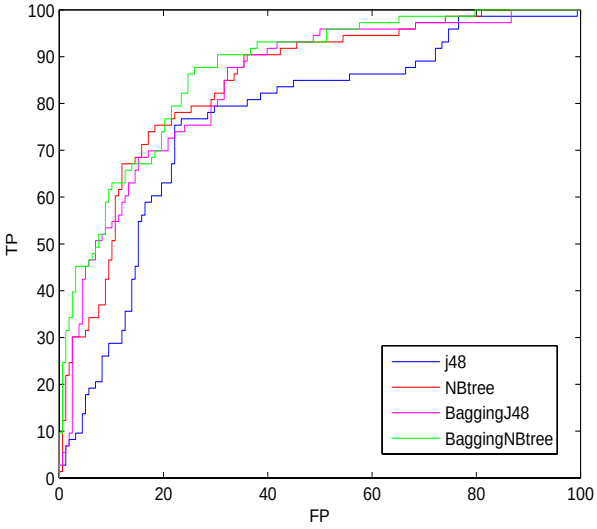


Figure 3: ROC curves for Pima dataset from J48 (-solid), NBtree (dotted), BaggingJ48(dashed), and BaggingNBtree(dash-dotted)

Table 2: Profit from Different Models ($r_{max} = 0.1$)

Dataset and Model	Probability Loss		Profit		
	average NCE	average brier	simulation 1	simulation 2	simulation 3
Covtype			$1.0e+5*$	$1.0e+6*$	$1.0e+5*$
J48	0.0268	0.0295	4.9074	0.5061	4.7703
NBtree	0.0369	0.0369	4.5403	0.4740	4.4527
BagJ48	0.0180	0.0203	6.6351	0.6815	6.7984
BagNBtree	0.0177	0.0204	9.7745	1.0131	9.8566
Pima			$1.0e+4*$		
J48	0.2380	0.3479	3.6396	0.9603	1.6686
NBtree	0.1924	0.2855	3.0532	0.7739	2.0492
BagJ48	0.1992	0.2950	3.2529	1.1815	1.6608
BagNBtree	0.1844	0.2724	4.7752	2.7599	2.9511

Table 3: Profit from Different Models ($r_{max} = 1/3$)

Dataset and Model	Probability Loss		Profit		
	average NCE	average Brier	simulation 1	simulation 2	simulation 3
Covtype			$1.0e+5*$		
J48	0.0268	0.0295	5.1169	5.0477	5.0875
NBtree	0.0369	0.0369	5.9022	5.8350	6.3156
BagJ48	0.0180	0.0203	6.8557	7.0939	7.0119
BagNBtree	0.0177	0.0204	9.7755	9.9511	9.8868
Pima			$1.0e+4*$		
J48	0.2380	0.3479	5.8322	3.0318	3.0332
NBtree	0.1924	0.2855	6.2662	5.2977	2.8935
BagJ48	0.1992	0.2950	7.0204	3.9391	3.0269
BagNBtree	0.1844	0.2724	7.3293	5.5422	4.5443

The profit derived from an individual consumer can potentially be negative, due to bad loaning decisions, but the model ideally should result in aggregated positive profit. The weighted benefit of true negative should overcome the cost of false negative.

To compare the performance from different models, the benefit function b_{11} and b_{10} are assumed to be zero, the loan amount x asked by the customer is generated by *-raylrand(n)* in *matlab*, which generates random numbers with Rayleigh distribution, with $n = 10,000$. But the actual loan amount granted is determined by discussion in previous sections. We ran three different simulations with the random loan amounts generated from the Rayleigh distribution. We also report the losses on the probability estimates using the negative cross entropy (NCE) and brier score measures [3, 2].

$$NCE = -\frac{1}{n}\{(\sum_{i|y=1} \log(p(y=1|x_i)) + \sum_{i|y=0} \log(1-p(y=1|x_i)))\}$$

$$Brier = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2$$

The limit of interest rate is set to be $r_{max} = 1/10$ and $r_{max} = 1/3$. Tables 2 and 3 show the results including the loss measures and overall profit for three different set of simulations and at different maximum interest rates. We notice that the quality of the estimates as indicated by the NCE or Brier score, particularly the NCE, agrees with the overall profit. This is not surprising as the benefit is weighted on the probability. Figures 2 to 3 show the ROC curves. We see that the ROC curves overlap for a large region of the ROC space. However, if we were to fit the ROC convex hull [8], it would potentially capture the models that result in maximal profits. For instance, bagged NBTree model has the highest profit for both covtype and pima.

We would like to point out that ROC curves are certainly limited in their evaluation of the quality of probability estimates. A classifier can achieve a high rank-ordering, but that is not necessarily indicative of the underlying calibration of the model [3]. Moreover, the benefit function is dependent on the quality of probability estimate. A weakly calibrated model can result in less economic utility, as evident from our results too. We advocate the use of appropriate loss measures, indicative of the calibration of the model for evaluating models utilizing a pricing scheme for loan decisions.

The empirical results are limited to be able to draw convincing observations. Nevetheless, there are strong trends that spell out. There is a strong relationship between quality of the probability estimates and the resulting profit from the model. Figure 4 shows the negative cross entropy computed per test example and the corresponding profit for the two dataset. It is clear that as the negative cross entropy increases, indicating poor quality of estimates, the profits decrease.

4. CONCLUSION

We discussed a pricing scheme for loan practice based on credit scoring models (classifiers) and benefit matrix.

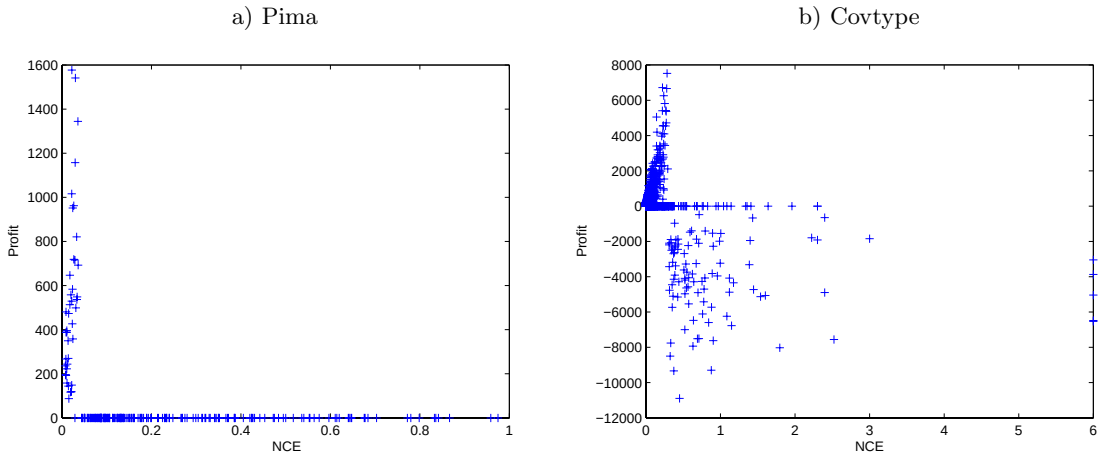


Figure 4: NCE vs Profit Plots

The framework enables pricing the loan for each individual customer conditioned on the loan amount and the corresponding interest rate. The two important conditions for the scheme are: 1. granting loans is more profitable than declining it; 2. ensuring a non-negative profit from loans. The theoretical framework can be encompassed to utilize the different policies as set by the bank. One factor that we did not explicitly consider is the amount of downpayment made by a consumer. That is a potential risk mitigator and can influence the interest rate.

As part of future work, we are investigating applications of the proposed scheme to other domains that require the parameterization of costs and benefits. The parameters are replaceable by the domain specifics. One application that comes to mind is medical informatics. For example, consider the case of diagnosing a patient with a disease. There are obvious benefits and costs associated with that domain. The benefits include correctly diagnosing and predicting the patient with disease, and the losses include the costs from mis-diagnosis.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments on the work. Thanks to Roger Stein for the pointer to his work.

5. REFERENCES

[1] C. Blake and C. Merz. UCI Repository of Machine Learning Databases http://www.ics.uci.edu/∼mlearn/∼MLRepository.html. Department of Information and Computer Sciences, University of California, Irvine, 1998.

[2] G. Blattenberger and F. Lad. Separating the Brier Score into Calibration and Refinement Components: A Graphical Exposition. *The American Statistician*, 39(1):26–32, February 1985.

[3] N. V. Chawla and D. A. Cieslak. Beyond error: Calibration and power of probability estimation decision trees and naive bayes. Technical Report 2006-09, University of Notre Dame, 2006. `www.cse.nd.edu/research/tech_reports`.

[4] P. Domingos. Metacost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, San Diego, CA, 1999. ACM Press.

[5] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 973–978, 2001.

[6] R. Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[7] D. Margineantu. Class probability estimation and cost-sensitive classification. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 270–281, 2000.

[8] F. Provost and T. Fawcett. Robust Classification for Imprecise Environments. *Machine Learning*, 42/3:203–231, 2001.

[9] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

[10] R. M. Stein. The relationship between default prediction and lending profits: Integrating roc analysis and loan pricing. *Journal of Banking and Finance*, 29:1213–1236, 2005.

[11] B. Zadronzy. One-benefit learning: Cost-sensitive learning with restricted cost information. In *Proceedings of the Workshop on Utility-Based Data Mining (UBDM) at the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (UBDM@KDD'05)*, 2005.

[12] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD'01)*, 2001.

Maximum Profit Mining and Its Application in Software Development

Charles X. Ling[1], Victor S. Sheng[1], Tilmann Bruckhaus[2], Nazim H. Madhavji[1]
[1]Department of Computer Science, The University of Western Ontario
London, Ontario N6A 5B7, Canada
{cling, ssheng, madhavji}@csd.uwo.ca
[2] Numetrics Management Systems
20863 Stevens Creek Blvd, Suite 510, Cupertino, CA
tilmannb@numetrics.com

ABSTRACT
While most software defects (i.e., bugs) are corrected and tested as part of the lengthy software development cycle, enterprise software vendors often have to release software products before all reported defects are corrected, due to deadlines and limited resources. A small number of these defects will be escalated by customers and they must be resolved immediately by the software vendors at a very high cost. In this paper, we develop an Escalation Prediction (EP) system that mines historic defect report data and predict the escalation risk of the defects for maximum net profit. More specifically, we first describe a simple and general framework to convert the maximum net profit problem to cost-sensitive learning. We then apply and compare several well-known cost-sensitive learning approaches for EP. Our experiments suggest that the cost-sensitive decision tree is the best method for producing the highest positive net profit and comprehensible results. The EP system has been deployed successfully in the product group of an enterprise software vendor.

(Note: *this paper is accepted by KDD'06*)

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning

General Terms
Algorithms, Management, Performance, Economics

Keywords
Escalation prediction, cost-sensitive learning, data mining

1. INTRODUCTION
Building large enterprise software is generally a highly complex and lengthy process, during which numerous software defects (i.e., bugs) are reported and required to be corrected or fixed. However, often some of reported defects will not be fixed when the software products are released, due to tight deadlines or limited resources [7]. After product release a small number of defects become "escalated" by customers, whose businesses are seriously impacted. Escalations of software defects require software vendors' immediate management attention and senior software engineers' immediate and continuous effort to reduce the business or financial loss to the customers. Therefore, software defect escalations are highly costly to the software vendors, with the associated costs amounting to millions of dollars each year. In addition, software defect escalations result in loss of reputation, satisfaction, loyalty and repeat revenue of customers, incurring extremely high costs in the long run for the enterprise software vendors [2, 3].

Due to time (i.e., deadlines) and resource limitations, enterprise software vendors can only fix a limited number of defects before product release. Thus, they must try to identify which reported defects have a high risk of escalation, which should be fixed at a much lower cost within the product development and testing cycle before product release. However, identifying software defects that are likely to escalate is a complex and difficult task. Software vendors often have in place human-centric processes for evaluating defect reports, but such processes are unreliable and subjective.

In this paper we propose a data-mining solution to predict escalation risks of defects to assist human experts in the review process of software defects. To the best of our knowledge, applying data mining for predicting software defect escalations is novel in software engineering. More specifically, we build an Escalation Prediction (EP) system that learns from history defects data and predicts escalation risk using data mining technology [1, 8, 12]. If the EP system can accurately predict the escalation risk of known defect reports, then many escalations will be prevented. This would save a huge amount of money for the enterprise software vendors [6].

Indeed, the ultimate business goal of EP (and many industrial applications using data mining) is to maximize the "net profit", that is, the difference in the cost before and after introducing the data mining solution, as opposed to the usual data-mining measures such as accuracy, AUC (area under the ROC curve), lift, or recall and precision [14]. However, the net profit is not equivalent to any of these standard machine learning measures

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.

[5], and we have found little previous work that directly optimizes the net profit as the data mining effort.

We first set up a simple framework in which the problem of maximum net profit can be converted to minimum total cost in cost-sensitive learning under certain conditions (see Section 2). We then apply and compare several well-known cost-sensitive learning approaches on a defect report dataset to see how they perform, in terms of maximum net profit (Section 5). The EP system is shown to be able to improve greatly the net profit in software production. The system is currently deployed within the product groups of an enterprise software vendor, and it has quickly become a popular tool for prioritization defects for fixing. EP is one of the successful stories of data mining deployed in industry.

2. MAXIMUM NET PROFIT AND COST-SENSTITIVE LEARNING

In this section we will propose a novel method that converts the maximum net profit problem to cost-sensitive learning.

As discussed in the Section 1, correcting defects after an escalation occurs at the customer site is much more expensive than correcting them before product release. However, deadlines and limited resources usually allow only a small number of defects to be fixed at a lower cost before product release. If we treat escalated defects as positive examples and non-escalated defects as negative examples, then false negative, or *FN*, is when EP predicts negative (non-escalation) but the defect is actually positive (becomes escalated after product release). Thus, the cost of *FN* (correcting an escalated defect) is relatively high. On the other hand, false positive, or *FP*, is when EP predicts positive (escalation defects) but the defect is actually negative (non-escalation). Thus, the cost of *FP* (correcting a non-escalated defect) is relatively low (and is the same as the cost of true positive, *TP*, or fixing an escalated defect). The cost of true negative, *TN*, is zero.

We assume that the cost *FN* is 7 times as the cost of *FP*. Further, we assume that *FN* = $7,000, and *FP* = $1,000 (we did not use the actual numbers here for confidentiality reasons but the actual cost figures are some constant factor of the numbers used here). Then the cost metric can be represented in Table 1. As the cost metric is known, this would seem to be a straightforward cost-sensitive learning problem, in which the weighted misclassification cost is minimized.

Table 1: Cost metric.

	Actual Negative	Actual Positive
Predict Negative	0	7,000
Predict Positive	1,000	1,000

The problem is not that simple. The goal of the EP (Escalation Prediction) system (and many other real-world data mining applications) is to maximize the net profit after data mining is deployed. That is, we want to maximize the gain (or difference) with the data-mining effort compared to the previous, default practice. That is, we want to compare the cost (or profit) after data-mining based EP is deployed to some default policy before EP is deployed.

Let us first establish the formula for the net profit, which is the difference of the total costs before and after deploying EP. Let us first calculate the cost after deploying EP ("gross profit"). If the software vendor follows faithfully the EP's predictions, it will fix all defects predicted positively, and ignore all defects predicted negatively. The cost of correcting all defects predicted positively by EP is thus the multiplication of the number of defects predicated positively and the cost of correcting such a defect; that is, $(tp+fp)\times FP$. (Note that we use small letters, such as *tp*, to represent the number of true positive cases, and capital letters, such as *TP*, to represent the cost of true positive cases. Similar notations are used for other cases.)

After the software is released, the would-be escalated defects predicted negatively by EP (i.e., false negatives) will escalate and must be fixed at a much higher cost of $fn\times FN$. Thus, the total cost after deploying EP is the sum of the two costs described above, plus the cost of the data mining effort (such as the cost of the tool, computer and human cost of using the tool, etc). If we ignore the cost of the data mining effort now, then the total cost of deploying EP is:

$$(tp+fp)\times FP + fn\times FN. \quad (1)$$

Assume that the default policy (before deploying EP) is to ignore all defect reports before software release, and then correct all escalated defects after release. Then using the same notation, the cost of this default policy is simply the cost of correcting all escalated defects. That is:

$$(tp+fn)\times FN. \quad (2)$$

Thus, the net profit is to subtract (1) from (2). That is:

$$Net\ profit = tp\times(FN-FP) - fp\times FP = 6000\times tp - 1000\times fp. \quad (3)$$

On the other hand, if the default policy is to correct all of the defects (which rarely happens due to deadlines and scarce resource), the cost would be $(tp+fp+tn+fn)\times FP$. Subtracting (1) from the cost of the correcting-all policy above, the net profit would be: $1000\times tn - 6000\times fn$. Thus, in general, the net profit varies with the default policy.

We propose a novel and simple approach to convert a formula of maximum net profit (such as (3) above) to cost-sensitive learning under certain conditions. Cost sensitive learning basically attempts to minimize the weighted cost, which can be expressed as:

$$tp\times TP + fp\times FP + tn\times TN + fn\times FN. \quad (4)$$

Therefore, as long as the net profit, such as (3), can be expressed as a linear formula of *tp*, *fp*, *tn*, and *fn*, we can negate its coefficients in the linear formula, and re-assign the cost metric by the negated coefficients. For example, the cost metric for the net profit (3) can be converted and represented in Table 2. In the rest of the paper, we will study EP under this cost metric.

Comparing the two cost metrics in Tables 1 and 2, we can see that one can be converted into the other when the first row is subtracted to the second row [11]. However, even though the optimal prediction for minimal total cost remains the same after

such conversions, the actual value of the total cost (4) would certainly be changed. That is, the software vendors not only want to know what predictions will produce the maximum net profit, but also the actual value of the net profit, as it reflects how much gain it may actually obtain after the data-mining based EP is deployed. Indeed, it is possible that the maximum net profit is negative (see Table 3 later in Section 5), indicating that it may be counter-productive to deploy the data mining system. In addition, as we have omitted the cost of the data mining effort (such as the tool cost, computer and human cost), the software vendor must obtain a positive net profit with a value that is large enough to offset other costs involved and decide if EP is a worthwhile endeavor or not.

To conclude, different default policies often result in different net profit calculations. However, under the condition that the net profit can be represented linearly by *tp*, *fp*, *tn*, and *fn*, we show that a simple and novel method can convert the maximum net profit to a new cost metric. Thus, the maximum net profit problem can now be solved by cost-sensitive learning. The result obtained (actual net profit value) will be useful for the software vendor to decide the overall benefit of deploying the data-mining system.

Table 2: Cost metric converted from (3). This cost metric will be used in the rest of the paper.

	Actual Negative	Actual Positive
Predict Negative	0	0
Predict Positive	1,000	–6,000

3. COST-SENSITIVE LEARNING

Cost-sensitive learning algorithms can be broadly categorized into two types. One is to design a *wrapper* that converts existing cost-insensitive (or cost-blind) base learning algorithms into cost-sensitive. The wrapper method is also called *cost-sensitive meta-learning*. The other is to design cost-sensitive learning algorithms directly [19, 10, 15]. In this section we will discuss briefly several well-known cost-sensitive learning approaches, including a method of our own (CSTree, described later) and several new improvements. Due to space limitation we will not provide full details of each method; instead, only highlights are given. The first four approaches belong to the wrapper method; they apply sampling etc. on cost-blind base algorithms to make them cost-sensitive. The last method is cost-sensitive by nature. These approaches will be compared experimentally in the next section to see which one(s) can reliably obtain the maximum net profit in Escalation Prediction.

The first popular approach, called "Undersampling" in this paper, is to rebalance the training data by sampling the data sets. Previous work (see, e.g., [10]) has indicated that over-sampling the minority class can cause overfitting, and may not be helpful to improve predictive performance. Other research [20] shows that keeping all examples of the rare class, and under-sampling the majority class to be about the same as the rare class performs well measured by AUC. Thus, we will use this simple strategy. That is, we keep all positive examples (escalated defects), and randomly sample without replacement the negative examples such that the two classes are balanced.

However, as the positive examples occupy only about 1% of the original training and test sets (see Section 4), the balanced dataset contains only about 2% of the original training set, which is very small. Therefore, we apply bagging [4] to Undersampling, as bagging has been shown to be effective in improving predictive accuracy and probability estimation measured by AUC. Bagging is also applied to other methods (see later).

As Undersampling itself is not cost-sensitive, the class label produced cannot be used directly in predicting escalation for maximum net profit. As the base learning algorithms used also produce probability estimates for the labels, testing cases can be ranked. To accurately calculate the maximum net profit, the threshold method is used: a separate validation set is used to find the best threshold on the probability estimates to classify the validation cases that would produce the maximum net profit (on the validation set), and this best threshold is then applied to the test set to classify the test cases into positive (if the predicted probability is larger than or equal to the threshold) or negative (if the predicted probability is less than the threshold). The net profit on the test set can then be calculated and obtained.

The second approach, called "Costing", is an advanced sampling which weighs examples according to the cost, producing cost-sensitivity with a provable performance guarantee [22]. The advanced sampling used is rejection sampling, in which each example in the original training set is drawn once, and is accepted into the sample with the probability proportional to its "importance". The size of the re-sampled dataset is smaller than the original set. With bagging, the method is shown to outperform other methods measured by the maximum total profit in two real-world datasets [22].

The third approach, called "Relabeling", relabels the classes of instances by applying the minimum expected cost criterion [9]. MetaCost [9] belongs to this approach. MetaCost uses bagging as the ensemble method.

The forth approach, called "Weighting" [18], induces cost-sensitivity by integrating the instances' weights directly into the classifier building process. It works when the base learners can accept weights directly. This method does not rely on bagging. However, to compare with other cost-sensitive methods, we also apply bagging to this method.

The fifth approach, called "CSTree", is a recently proposed cost-sensitive decision-tree algorithm that uses the total cost directly as a split criterion [15]. That is, instead of minimizing entropy in the tree building process, the split criterion simply minimizes the total misclassification cost given a cost metric. CSTree is, by nature, truly cost-sensitive, and thus, the result of classifying the test cases is used directly in the calculation of the maximum net profit.

Several improvements have been made in the CSTree for the EP application. First of all, compared to [15], the costs for true positive (*TP*) and true negative (*TN*) may not be zero here. Second, *expected* total cost reduction is used during tree building. That is, an attribute may be selected as a root if the expected cost reduction is maximum (among other attributes), and is greater than 0. More specifically, for a given set of cases without split, if C_P (= $tp \times TP + fp \times FP$) is the total cost of labeling a leaf positively, and C_N (= $tn \times TN + fn \times FN$) is the total cost of labeling a leaf negatively, then the probability of a positive outcome is first estimated by the relative cost of C_P and C_N as

$1 - \frac{C_P}{C_P + C_N} = \frac{C_N}{C_P + C_N}$. (The smaller the cost C_P, the larger the probability of being positive, as minimum cost is sought). The expected misclassification cost of being positive is thus: $E_P = \frac{C_N}{C_P + C_N} \times C_P$. Similarly, the probability of a negative outcome is $\frac{C_P}{C_P + C_N}$, and the expected misclassification cost of being negative is: $E_N = \frac{C_P}{C_P + C_N} \times C_N$. Therefore, without splitting, the expected total misclassification cost of a given set of examples is: $E = E_P + E_N = \frac{2 \times C_P \times C_N}{C_P + C_N}$. If an attribute A has k branches, then the expected total misclassification cost after splitting on A is: $E_A = 2 \times \sum_{i=1}^{k} \frac{C_{P_i} \times C_{N_i}}{C_{P_i} + C_{N_i}}$. Thus, ($E$ – E_A) is the expected cost reduction splitting on A. During tree construction, an attribute with the maximum expected cost reduction is chosen, and if the value is greater than 0, then the attribute is chosen to split the training examples; if not, a leaf node is formed.

The last improvement is tree pruning. The original method did not incorporate pruning. We have found that without pruning, the tree overfits the training examples. This means that a larger tree (that fits the training examples better) predicts worse compared to a small tree. We have implemented post pruning similar to the post-pruning in C4.5 [17] in our cost-sensitive decision tree (but our post-pruning is guided by minimum total misclassification cost).

The CSTree has two distinctive advantages. The first one is that there is no need for sampling in the training data. The original training data can be used directly in building the decision tree. That is, CSTree can naturally utilize all information in the training data. Second, CSTree does not rely on bagging for the final outcome. This implies that the final results can be comprehensible by domain experts.

4. THE DATASETS

Our dataset consists of historical defect reports from industry software projects of an enterprise software vendor. Defect reports change over time and so there is an opportunity to learn from multiple different versions of a single defect report. Additionally, the same defect can be reported several times by different parties. Therefore, numerous data records in the dataset may belong to only one defect. Confidentiality of the data only allows us to give a brief description of the data. The data was collected in a period of 2004, and contains a total of about 165,000 records (defect report observations). Note that the class labels of this dataset have all been verified by real escalations of the software products, thus this dataset contains much less noise than the one used in our previous work [16]. The total number of attributes is 53, most of which are numerical attributes (including dates), and a few are nominal (such as product name). The system uses three types of inputs: 1) raw inputs which correspond to field values which are reported by users and stored in the defect tracking database, 2) row-level transformations of raw values, such as concatenating two string-valued raw inputs into a derived new string-valued input, and 3) statistical inputs which are derived from statistical analyses of all rows which fall into a given time period, such as a fiscal quarter.

Statistical inputs were particularly useful for string valued raw inputs because string-valued data is typically difficult to use for machine learning. For example, a raw input for "CUSTOMER_NAME" may have 20,000 possible customer name values. Therefore, the system calculates statistics for string-valued inputs. These statistics include counts (number of defects submitted by customer "abc", and number of escalations raised by customer "abc"), means (number of escalations over number of defects submitted for hardware platform "xyz"), and the probability of an escalation, and so on. Further analysis of these inputs and their utility for prediction is of great interest but beyond the scope of this paper.

The target attribute is binary (escalation or no escalation). The whole dataset is split up into training and test sets according to the defect report date. The training set contains about two thirds of the records (before a certain date), and the test set contains one third of the records (after that date). The dataset is very imbalanced, with slightly less than 1% of the positive examples (escalated defects).

5. COMPARING COST-SENSITIVE LEARNING APPROACHES FOR EP

In this section we compare the five cost-sensitive learning approaches (Undersampling, Costing, Relabeling, Weighting, and CSTree) discussed in Section 3 on the latest dataset we have obtained. Again the latest dataset contains much less noise than the dataset used in our previous work [16], thus the results reported here are different (but the general conclusion is still the same). In addition, more base learning algorithms are included in the current study (see later).

Again the first four approaches are wrappers or meta cost-sensitive learning methods, as they apply to any cost-blind base learning algorithms to make them cost-sensitive. We choose four popular learning algorithms, naïve Bayes, the decision tree algorithm C4.5 [17], decision stump [13], and REPTree [21] as their base learners, due to their popularity and high efficiency. The fifth method (CSTree) is also a decision tree but it is cost-sensitive in nature. It would be interesting to compare various wrapper methods applying to cost-blind C4.5 to the cost-sensitive decision tree (CSTree).

To make the net profit result comparable with different approaches, we use "unit" net profit, defined as the net profit divided by the number of records in the test sets in our comparison. Note that even if a data mining method obtains a unit net profit of $1, the saving to the enterprise software vendors can be quite significant. If we assume that 1 unit represents $50 in reality (a reasonable figure), then with 1,000 software defect reports in a software product to be released, a total saving with EP's prediction would be 1,000×50 = $50,000 for a typical customer. Alternatively, assuming a unit cost of $250 and 50,000 defect reports the total savings amount to $12.5 million.

Bagging is applied to all methods. The maximum unit net profit, when the bagging iteration is set to 1 (no bagging), 10, and 100 are shown in Table 3.

Table 3: Comparing unit net profit using five different cost-sensitive approaches.

Cost-sensitive Approaches		Bagging Iterations		
		1	**10**	**100**
Undersampling	Naïve Bayes	0.00	0.43	1.85
	C4.5	-0.05	0.01	3.52
	REPTree	0.00	6.97	10.80
	DecisionStump	0.00	0.00	0.00
Costing	Naïve Bayes	-6.60	-5.45	-5.26
	C4.5	11.87	13.85	13.99
	REPTree	9.63	11.31	13.34
	DecisionStump	0.00	0.00	0.00
Relabeling (MetaCost)	Naïve Bayes	-16.49	-17.56	-18.01
	C4.5	0.00	0.00	0.00
	REPTree	8.00	12.83	11.15
	DecisionStump	0.00	0.00	0.00
Weighting	Naïve Bayes	-5.69	-6.10	-6.93
	C4.5	9.10	11.38	11.47
	REPTree	12.96	14.16	14.39
	DecisionStump	0.00	0.00	0.00
CSTree	NA	14.72	14.15	14.07

From the results in the table, we can draw several interesting conclusions.

First, in terms of performance of the five different cost-sensitive learning algorithms, we can see that they perform quite differently, even when they use the same base learning algorithms. This means that some trial-and-error is needed to choose the best performing cost-sensitive approach in EP (and other real-world applications). For meta-learning approaches (the first four approaches), different base learners also yield very different performance. In general, C4.5 and REPTree perform better than other base learning algorithms. The fifth method, CSTree, is the best. Decision tree models (such as C4.5 and REPTree) with no bagging (bagging iteration is 1) also produce comprehensible results that can provide insights to the software vendors as to why certain defects are highly likely to escalate.

Second, compared to no bagging (bagging iteration is 1), bagging (with 10 or 100 iterations) generally improves the result when the maximum unit net profit is positive. However, CSTree achieves an excellent result without bagging (with the highest unit net profit of 14.72). Again without the need of bagging, a single tree has the advantage of producing comprehensible results for domain experts to exam the outcome of data mining.

Third, sometimes cost-sensitive methods produce very negative in terms of unit net profit, such as Costing, Relabeling, and Weighting with naïve Bayes. As we discussed in Section 2, often these absolute values of the unit net profit is important for software enterprise vendors to decide if the deployment of data mining is profitable or not, and by how much, compared to the default strategy. Negative net profit, even though it is a maximal value, indicates that it is not worthwhile to deploy the data mining effort. Fortunately in our case, we have found clear winners: the CSTree, Costing, and Weighting, as they have resulted in large positive unit net profit. Deployment of EP with CSTree is encouraging. See Section 6 for details.

To summarize, of all algorithms tested, CSTree performs best for the EP task. In Table 4, we rank the five algorithms according to their highest unit net profit, and the average unit net profit (averaging over the four base learners for the meta cost-sensitive learning methods). Both rankings indicate that CSTree performs the best, followed by Costing or Weighting, and followed by Undersampling or Relabeling.

Table 4: The rankings of the five cost-sensitive methods in comparison. Numbers in parentheses are the unit net profit.

	1 (best)	2	3	4	5
Ranked by highest profit	CSTree (14.72)	Weighting (14.39)	Costing (13.99)	Relabeling (12.83)	Undersampling (10.80)
Ranked by average profit	CSTree (14.31)	Costing (4.72)	Weighting (4.56)	Undersampling (1.96)	Relabeling (-1.67)

Clearly CSTree without bagging is the most appropriate choice for deployment in the software development cycle, due to its superior performance and comprehensibility. We will discuss the deployment of EP in the next section.

6. DEPLOYMENT

Our EP system has been in deployment for many months in the product group of an enterprise software vendor where the dataset comes from. It has been used to make suggestions on current defect reports with high risks of escalation. As the software vendor must wait to see which defect reports are actually escalated after the product is released, and the would-be escalations will not happen after they are accurately predicted by EP and corrected before software release, the final evaluation cannot be obtained until the software has been released and has been in use by customers for some period of time (in the order of one or more quarters).

We have evaluated EP using the defect reports submitted or updated during the most recent three weeks in the test set. Any records corresponding to defect reports which had already been escalated at the time of preparing the data set were also removed. After EP makes its predictions, the results are compared to the actual escalations happened up to date. The EP prediction performance is quite good. The lift chart of EP's prediction is well above the random diagonal line: at top 10 percentile (among top 10% of the most likely predicted escalations), about 70% of the actual escalations are predicted; at 20 percentile about 85%; and at 30 percentile about 90%. These results are of significant business value to the software vendor. In addition, the product group has provided positive feedback on the performance of the EP system, citing that it "catches" defect reports that, after thorough evaluation by specialist, are considered likely candidates for future escalations. For instance, some defect reports have been found to have been assigned a lower than appropriate priority. After a prediction of high escalation risk becomes available such a defect report can be upgraded to a higher priority which will lead to expedited evaluation and resolution.

7. CONCLUSIONS

Enterprise software development is an extremely complex process with hundreds or even thousands of defects that need to be prioritized and resolved, preventing them from escalation by customers and incurring a very high cost to the software vendors. In this paper, we present a successful case for predicting escalation risks mined from known product defect reports. The enterprise software vendors can proactively resolve these defects with the greatest risk of escalation at a much lower cost. This can save software vendors an enormous amount of software maintenance cost. An escalation prediction (EP) system based on data-mining for the maximum net profit has been proposed and tested, and is currently deployed at an enterprise software vendor. Results provide strong evidence that we can indeed make useful predictions about the escalation risk of product defects.

More specifically, we establish a general framework in which the maximum net profit problem can be converted to cost-sensitive learning. We then evaluate and compare five major cost-sensitive learning methods for their effectiveness for EP. We find that the CSTree (with novel improvements) can produce large positive unit net profit, as well as comprehensible results. This is often important for deploying data mining solutions in industry.

In our future work, we plan to continue to improve the effectiveness of the EP system and track its results from the software vendors.

8. REFERENCES

[1] Berry, M.J.A., and Linoff, G. 1997. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons.

[2] Boehm, B.W., and Basili, V. 2001. *Software Defect Reduction Top 10 List*. Computer 34(1): 135-137.

[3] Boehm, B.W. 1981. *Software Engineering Economics*. Prentice-Hall Advances in Computing Science & Technology Series.

[4] Breiman, L. 1996. Bagging Predictors. *Machine Learning* 24(2): 123-140.

[5] Bruckhaus, T. 2006 (forthcoming). The Business Impact of Predictive Analytics. Book chapter in *Knowledge Discovery and Data Mining: Challenges and Realities with Real World Data*. Zhu, Q, and Davidson, I., editors. Idea Group Publishing, Hershey, PA

[6] Bruckhaus, T., Ling, C.X., Madhavji, N.H., and Sheng, S. 2004. Software Escalation Prediction with Data Mining. *Workshop on Predictive Software Models*, A STEP Software Technology & Engineering Practice.

[7] Chulani, S., and Boehm, B.W. 1997. *Modeling Software Defect Introduction*. California Software Symposium, Nov.

[8] Dai, H. (editor). 2003. Proceedings of the International Workshop on Data Mining for Software Engineering and Knowledge Engineering.

[9] Domingos, P. 1999. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 155-164, ACM Press.

[10] Drummond, C., and Holte, R.C. 2003. C4.5, Class Imbalance, and Cost Sensitivity: Why under-sampling beats over-sampling. *Workshop on Learning from Imbalanced Datasets II*.

[11] Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. In *Proceedings of the International Joint Conference of Artificial Intelligence*, 973-978.

[12] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (editors). 1996. *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

[13] Iba, W. and Langley, P. 1992. Induction of One-level Decision Trees. In *Proceedings of the Ninth International Workshop on Machine Learning*, 233-240.

[14] Ling, C.X., and Li, C. 1998. Data Mining for Direct Marketing: Specific Problems and Solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 73-79.

[15] Ling, C.X., Yang, Q., Wang, J., and Zhang, S. 2004. Decision trees with minimal costs. In *Proceedings of International Conference on Machine Learning*.

[16] Ling, C.X., Sheng, S. Bruckhaus, T., and Madhavji, N.H. 2005. Predicting Software Escalations with Maximum ROI. In *Proceedings of International Conference of Data Mining*.

[17] Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[18] Ting, K.M. 2002. An Instance-Weighting Method to Induce Cost-Sensitive Trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659-665.

[19] Turney, P.D. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2:369-409.

[20] Weiss, G., and Provost, F. 2003. Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research* 19: 315-354.

[21] Witten, I.H., and Frank, E. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco.

[22] Zadrozny, B., Langford, J., and Abe, N. 2003. Cost-Sensitive Learning by Cost-Proportionate Example Weighting. In *Proceedings of International Conference of Data Mining*.