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 (≤ 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).

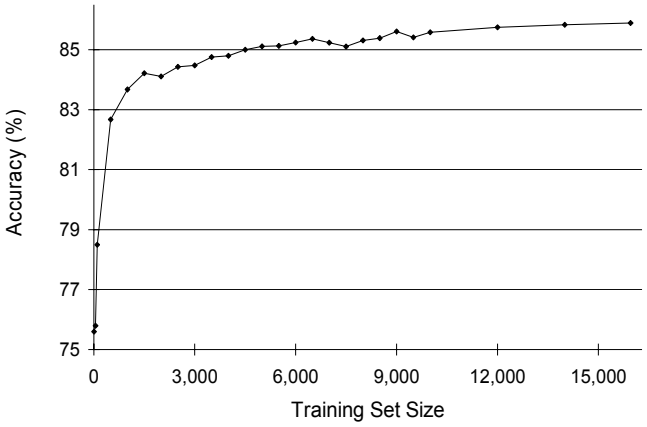

Figure 1: Learning Curve for Adult Data Set

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*).

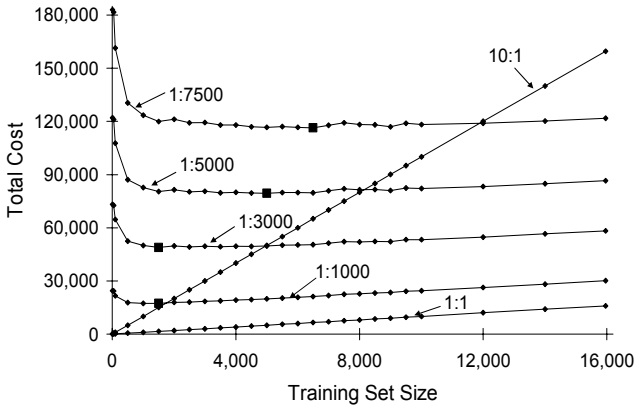

Figure 2: Utility Curves for Adult Data Set

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.

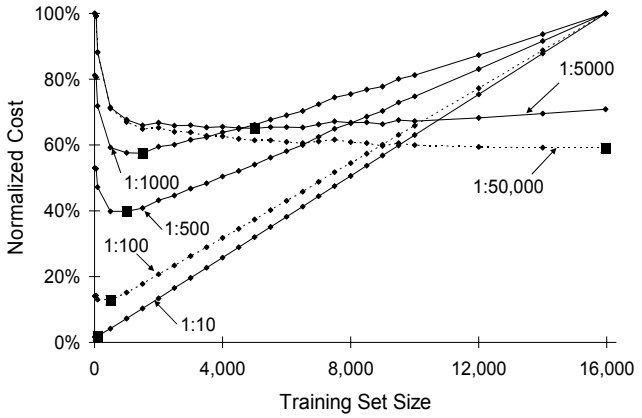

Figure 3: Normalized Utility Curves for Adult Data Set

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.

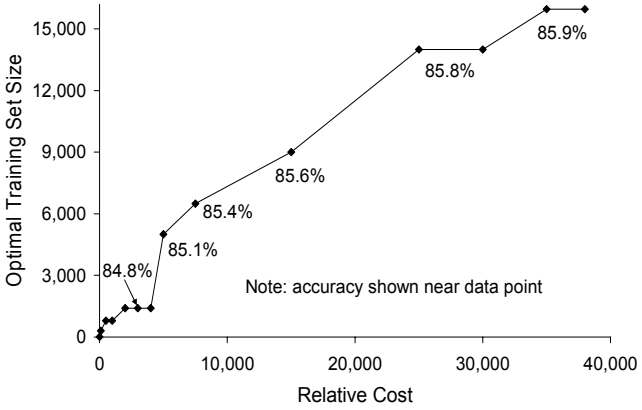

Figure 4: Optimal Training Set Sizes for Adult Data Set

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.

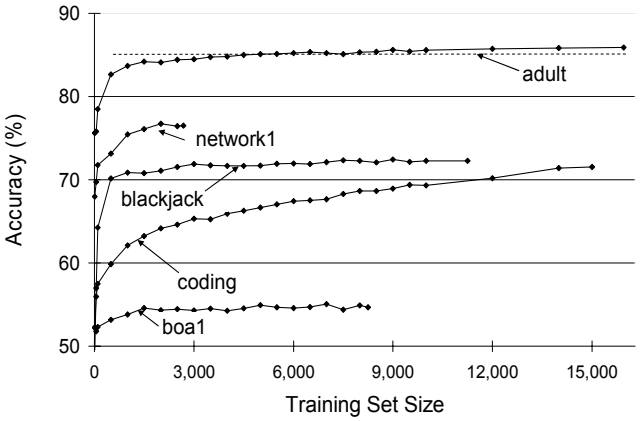

Figure 5: Learning Curves for Large Data Sets

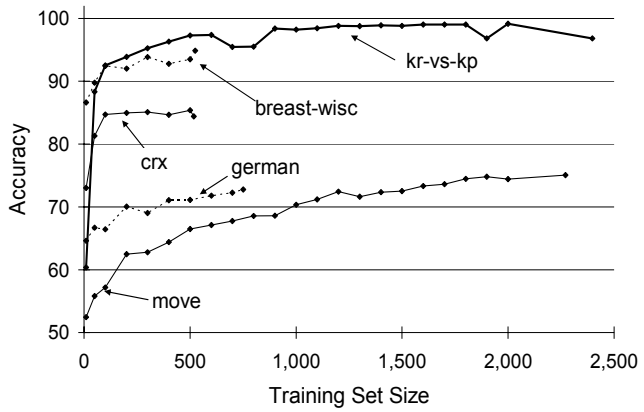

Figure 6: Learning Curves for Small Data Sets

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.

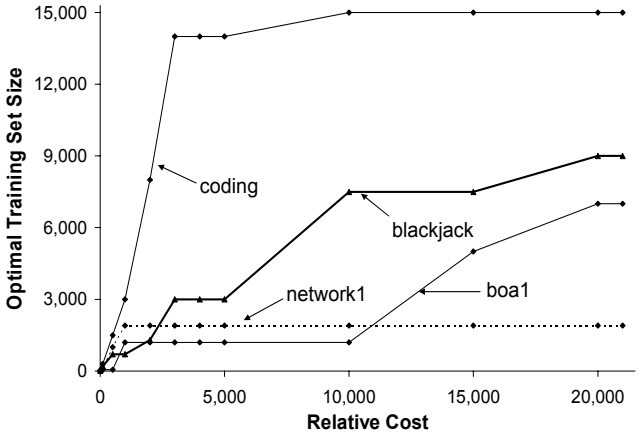

Figure 7: Optimal Training Set Sizes for Large Data Sets

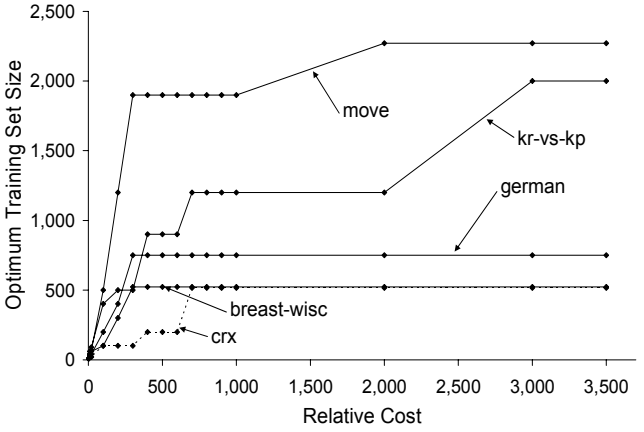

Figure 8: Optimal Training Set Sizes for Small Data Sets

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.

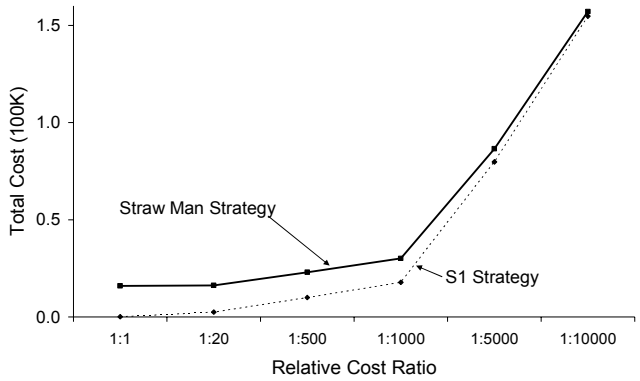

Figure 9: Comparison of S1 and Straw Man Strategies for Adult

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.