Improving Data Mining Utility with Projective Sampling

Mark Last
Ben-Gurion University
Beer-Sheva
Israel
+972-8-6461397
mlast@bgu.ac.il

ABSTRACT

Overall performance of the data mining process depends not just on the value of the induced knowledge but also on various costs of the process itself such as the cost of acquiring and pre-processing training examples, the CPU cost of model induction, and the cost of committed errors. Recently, several progressive sampling strategies for maximizing the overall data mining utility have been proposed. All these strategies are based on repeated acquisitions of additional training examples until a utility decrease is observed. In this paper, we present an alternative, projective sampling strategy, which fits functions to a partial learning curve and a partial run-time curve obtained from a small subset of potentially available data and then uses these projected functions to analytically estimate the optimal training set size. The proposed approach is evaluated on a variety of benchmark datasets using the RapidMiner environment for machine learning and data mining processes. The results show that the learning and run-time curves projected from only several data points can lead to a cheaper data mining process than the common progressive sampling methods.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Performance, Economics, Experimentation.

Keywords

Classification, utility-based data mining, cost-sensitive learning, active learning, data acquisition, optimization.

1. MOTIVATION AND SIGNIFICANCE

The costs of many real-world data mining projects depend on the amount of collected data as well as on the duration of the data collection process. Examples of costly and/or time-consuming data sources include medical records provided over time by a hospital, manufacturing records representing results of a costly engineering experiment, seasonal records stored in an agricultural database, etc. Even when an unlimited amount of raw data can be obtained for free, its preparation for the data mining process may still be labor intensive. The model induction may also require substantial computing resources. All these situations require a careful consideration of the added value of new examples (e.g., in terms of the improved predictive accuracy) vs. the costs and/or the times involved in acquiring those examples.

KDD'09, June 28– July 1, 2009, Paris, France.

According to [21], the ultimate goal of *utility-based data mining* is to maximize the utility of the entire data mining process by taking into account all utility considerations. In the case of a classification task, this means considering the trade-off between an increase in predictive accuracy and the cost of acquiring and processing new data. As shown in [21], for each data set and learner there is an optimal training set size that maximizes the overall utility of the classifier. Given a ratio between the data acquisition cost and the error cost, a cost-effective sample size can be found iteratively by one of progressive sampling schemes presented in [21], which stop after the first observed increase in total cost of the data mining process. A clear disadvantage of the progressive sampling approach, beyond the potential overhead associated with each sampling increment, is that it may overfit some local perturbations in the error rate and thus stop prematurely due to a temporary increase in total cost. According to the results presented in [21], progressive sampling costs may exceed the optimal ones by 20%-200% when the ratio between the error and the training example costs goes below 1,000. For higher cost ratios, progressive sampling usually stays within 10% of the optimum.

In this paper, we present and evaluate an alternative, *projective sampling* strategy, which fits a function to a partial learning curve obtained from a small subset of potentially available data and then uses it to analytically estimate the optimal training set size. This strategy is expected to reduce the overall cost of the data mining process, and thus to increase its utility, due to several reasons. First, an accurate estimation of the learning curve should avoid overfitting the progressive sampling data by successfully identifying the global rather than a local

optimum. Second, the data acquisition effort can ideally be reduced to just two iterations: one for estimating the learning curve and one for obtaining the projected optimal amount of the training data. As we build less models, the overall model induction effort should also decrease, which is important in case of significant CPU costs.

2. PROBLEM STATEMENT

In this paper, we follow the utility framework proposed in [21] by Weiss and Tian who define the total utility of the classification process as a sum of the three terms: Data Cost, Error Cost, and Induction Cost. However, we consider the third term as optional, since in many practical applications the actual run times are short and not associated with any tangible costs. Based on the common model assessment approach (see [7]), we assume the data to be divided into three parts: the *training set* of *n* examples used to induce the model, the *test set* used to estimate the model accuracy, and the *score set S* of future examples to be classified by the model. We also assume the Error Cost to be proportional to the number of errors made when classifying the score data set *S* (under the assumption that the costs of all errors are equal). Without loss of generality, a fixed size of *S* (e.g., 100) can be assumed. Like in [21], we measure the Data Cost exclusively in terms of the *amount* of training data records though in many real-world data mining projects it is not practical to acquire one example at a time and the total number of acquired *samples* should also be taken into account when choosing a sampling strategy. Three unit costs involved include C_{tr} for acquiring each new training example (including all its feature values and a class label), C_{time} for one unit of CPU time, and C_{err} for each misclassified example from the score set. *CPU(n)* stands for the time required to induce a classification model from *n* training examples and the error rate of that model measured on the test set is denoted by *err(n)*. Consequently the total cost of a classifier induced from *n* training examples can be calculated by the following expression:

$$Total\ Cost\ (n) = n \cdot C_{tr} + err(n) \cdot |S| \cdot C_{err} + CPU(n) \cdot C_{time} \quad (1)$$

Since the precise shape of a given learning curve is not known in advance [15], the only way to find the truly optimal training sample *n* minimizing the value of *Total Cost (n)* is to acquire all available examples and then evaluate the testing errors of all potential subsets of this amount. Of course, such an approach is of little practical significance, since it incurs the maximum possible data acquisition cost in the first place. However, as indicated in [15], the optimum sample size can be approximated by a progressive sampling procedure *P*, which requires running a classification algorithm on *k* samples of increasing size n_i each, where $i \in [1, k]$ and $\forall i$: $n_{i+1} > n_i$. The overall cost of such procedure equals to

$$Total\ Cost(P) = n_k \cdot C_{tr} + err(n_k) \cdot |S| \cdot C_{err} + \sum_{i=1}^{k} CPU(n_i) \cdot C_{time} \quad (2)$$

Thus, Total Cost (P) is the sum of the cost of acquiring the largest sample of size nk, the computational cost of inducing k models, and the cost of errors committed by the model induced from nk examples. If the learning curve is "well behaved" [15], i.e. the accuracy is a monotonically non-decreasing function of n, one should retain the model induced from the largest sample of size nk, since it cannot be less accurate than any model induced from a smaller sample.

The projective sampling algorithm presented in this paper aims at defining a heuristic sampling strategy *P** that minimizes the total cost of a classification process:

$$P^* = \arg\min_{P} Total\ Cost(P) \quad (3)$$

Like any other sampling techniques, the proposed heuristic algorithm cannot be guaranteed to find the global optimum of *Total Cost (P)*. Still, it can be useful for data miners if its average cost is *not much higher* than the optimal (minimum) cost and *lower* than the cost of alternative sampling strategies. In addition to direct cost reduction, we are also interested in minimizing the total number of acquired samples *k*, though estimating the acquisition cost of each additional *sample* (as opposed to the cost of each additional *example*) is beyond the scope of this study. Thus, the optimal strategy would be to acquire a single sample that minimizes the total cost. Unlike some other active sampling techniques (see [17]), we assume here that the data miner's choice is restricted to the amount of acquired examples, which are drawn randomly from the entire population.

The rest of this paper is organized as follows. In Section 3, we cover the main existing approaches to utility-based sampling and learning curve approximation. Section 4 goes into the details of the proposed sampling strategy (named "projective sampling"). Section 5 evaluates the proposed approach to projected optimization of classifier utility on ten benchmark datasets using a decision-tree classification algorithm. All experiments are performed using RapidMiner (formerly Yale) environment for machine learning and data mining processes [12]. The paper is concluded by Section 6, which summarizes the results and outlines directions for future research.

3. RELATED WORK

Provost *et al.* [15] propose an efficient progressive sampling schedule aimed at minimizing induction and error costs while ignoring data acquisition costs. Specifically, they are concerned with reducing the total CPU time required to detect the convergence, i.e. the plateau region, of a "well behaved" learning curve. They show, both theoretically and empirically, that the *geometric* sampling schedule is asymptotically optimal in terms of run time for

induction algorithms of polynomial time complexity. They have also found empirically that the run time complexity of a decision tree learner (C4.5) varies approximately between $O(n^{1.2})$ and $O(n^{1.4})$ depending on a specific dataset.

The utility model of [15] is extended by Weiss and Tian [21] who also consider the cost of acquiring training examples. Their classification cost model is shown in Eq. (1) above. Based on this model, the utility of several progressive sampling strategies is evaluated in [21] on 12 datasets using the C4.5 decision-tree algorithm [16]. The datasets used in [21] contain between 1,000 and 581,000 examples. The first sampling strategy evaluated by [21] is *uniform sampling,* which generates 50 equally spaced training samples from the data available for training. Obviously, such strategy may cause a substantial increase in the total CPU time represented by the second factor in Eq. (2) above. Under the second, *geometric* sampling strategy, the sample size doubles with each iteration. Both strategies stop after the first increase in the total cost. According to the experimental results, the uniform strategy generally outperforms the geometric sampling unless the error cost dominates all other costs requiring large amounts of training data.

The authors of [21] also try to approximate the learning curve of one dataset by a very simple function, namely $f(x) = a + bx/(x+1)$, though they do not present any formal method for deriving the a and b coefficients of that equation. They also suggest that the optimal training set size can be estimated from a partial learning curve as an alternative to the progressive sampling strategy, but do not evaluate this approach in their paper.

Sheng and Ling [18] propose a progressive sampling framework that allows a cost-sensitive learner to continuously acquire examples using a uniform sampling schedule as long as the total cost (containing only the data cost and the error cost factors) can be reduced. Contrary to [15] and [21], their data acquisition model assumes that each example feature can be procured separately. They also allow a separate acquisition of an example label. Consequently, the data cost of a labeled example can be calculated as a sum of attribute costs and the labeling cost. They apply a cost-sensitive decision tree algorithm (called *CSDT*), based on C4.5 [16], to a collection of benchmark datasets using two attribute acquisition strategies (complete and partial).

Sheng *et al.* [19] extend the data acquisition framework of [18] by allowing a repeated labeling of the same example by multiple noisy labelers. The labeling cost is assumed to be fixed and generally lower than the cost of acquiring an unlabeled example. In their future work discussion, they indicate that given the gradient of the learning curve, a progressive sampling algorithm could choose between acquiring new examples and repeated labeling of existing ones.

Frey and Fisher [5] have conducted an extensive set of experiments showing that the *power law* is the best fit for modeling the error rates of the C4.5 decision-tree algorithm [16]. The percentage of explained variation (r^2) of the power law was compared to r^2 of linear, logarithmic, and exponential functions across 14 benchmark datasets of relatively small size (having less than 1,000 instances on average). The power law has produced the highest value of r^2 in 12 datasets out of 14 resulting in the best models predicting diminishing returns in the error rate for increasing the amount of training data. In their further experiments, the authors of [5] have also shown that a power law model derived from a relatively small portion of data (15%) can be used to reliably estimate the error rate for decision trees learned on the remaining amount of data as a function of the training set size. Similar results have been obtained in [10] when applying the power law to the learning curves of an oblivious decision tree algorithm (Information Network [11]).

Singh [19] argues that the power law is only second best to the logarithmic regression for a variety of classification algorithms, namely ID3, k-Nearest Neighbors, Support Vector Machines, and Artificial Neural Networks. His results are based on four datasets from the UCI Repository [1] with the number of instances varying between 101 and 1,728. The C4.5 decision-tree algorithm used by Frey and Fisher in [5] was not included in Singh's experiments.

4. PROJECTIVE SAMPLING STRATEGY

4.1 Algorithm Overview

The main idea of the projective sampling strategy presented in this paper is to fit an analytical function (such as the power law) to a partial learning curve induced from a small subset of potentially available data and then use it to estimate the optimal training set size. The projective sampling schedule is outlined in this sub-section, whereas the subsequent sub-sections suggest some functions for approximating the learning curves of classification algorithms and describe the procedure for finding the optimal number of training examples given the equation of a specific learning curve.

The pseudo-code of the projective sampling algorithm for minimizing the cost of a classification procedure is shown in Figure 1. The projective sampling schedule starts with an empty training set. A fixed sampling increment n_δ (number of examples acquired in each sample) is determined in advance based on domain-specific constraints such as the minimum number of production batches in each manufacturing experiment. Each purchased sample adds a data point for the calculation of the learning curve equation, where the cumulative training set size n is considered an independent variable and the error rate $err(n)$ of the model induced from n examples is treated as the dependent variable.

The algorithm needs at least *min_points* number of samples for approximating the learning curve. This number is equal to three data points, which is the minimum amount required to determine the equation of a non-linear curve. Following some data transformations, which are needed to convert all functions into a more convenient, linear form, Pearson's *correlation coefficient* (see [13]) can be calculated for each candidate function using the formula (20) shown in the Appendix (k represents the number of data points). Since in a "well-behaved" learning curve, the error rate should be a non-decreasing function of n, a function with a minimal (closest to -1) negative correlation coefficient should be the best fit for the data. However, the actual data points may be noisy, occasionally resulting in positive correlation coefficients over a limited number of data points. In that case, the algorithm will keep purchasing additional samples until the lowest correlation coefficient becomes negative or the maximum amount of available training examples n_{max} is exceeded.

Once the best functional form for a given learning curve is found, the coefficients of the *err (n)* equation are estimated from the acquired data points using the standard *least squares method* [14]. The formulas (21-23) for computing the regression coefficients are shown in the Appendix. The algorithm uses the same method to estimate the coefficients of the run-time complexity function *CPU (n)*. The *CPU (n)* function having the highest positive correlation coefficient is selected, since the run-time is expected to increase with the size of the training set. The computed regression coefficients of *err (n)* and *CPU (n)* are used to estimate the optimal training set size n^*. If n^* is greater than the size of the current training set, the algorithm purchases the missing amount of examples. When the estimated n^* exceeds the total amount of potentially available training examples n_{max}, n^* is set to n_{max}. Once a classification model *M (n*)* is induced from the dataset of size n^*, the error rate *err (n*)* of *M (n*)* is used to calculate the total cost of the projective sampling strategy P^*.

4.2 Approximating the Learning Curve and the Run-Time Complexity

The projective sampling algorithm presented in the previous sub-section is based on the common assumption (e.g., see [15]) that the error rate *err (n)* is a non-increasing function of the sample size n. Following [5] and [15], we also assume that the slope of the *err (n)* function is monotonically decreasing (exhibiting "diminishing returns"), which implies that the linear model should be excluded from consideration. Still, a variety of functions can meet these requirements.

Input: the sampling increment n_δ; the maximum amount of available training examples n_{max}; the minimal amount of data points required to estimate the learning curve *min_points*; the score set S of examples to be classified by the model; the cost of acquiring each training example C_{tr}; the cost of one unit of CPU time *Ctime*; the cost of each misclassified example from the score set C_{err}; and the set of functions to be used for approximating the learning curve.
Output: the projected equation of the learning curve; the projected optimal training set size n^*; the model *M(n*)* induced from n^* examples; *CPU(n*)* - the CPU time required to induce *M(n*)*; *err(n*)* - the error rate of *M(n*)* measured on the score set S; and *Total Cost (P*)* - the total cost of the projective sampling strategy P^*.

1. $n = 0$ // Initialize the training set size n
2. $i = 0$ // Initialize the number of data points used to compute the equation of the learning curve;
3. Do // Find the best approximation of the learning curve
4. Acquire n_δ examples
5. $n = n + n_\delta$ // Update the training set size n
6. $i = i + 1$ // Increment the number of data points
7. Induce a classification model *M (n)* from n examples
8. Measure the error rate *err(n)* of *M (n)* on the score set S
9. If ($i \geq min\_points$) then
10. *Best_Corr* = 0 // Initialize the best correlation coefficient
11. Use Eq. (20) to calculate the correlation coefficient $Corr_{ij}$ for each function j based on i classification models induced so far
12. *Best_Corr* = $min_j\, Corr_{ij}$ // The error rate must be a non-decreasing function of n
13. *Best_Function* = $arg\, min_j\, Corr_{ij}$
14. End if
15. While ((*Best_Corr* $\geq$ 0) and (n < n_{max}))
16. Estimate the regression coefficients of the selected *Best_Function* using Eq. (21-22)
17. Estimate the correlation coefficients of available *CPU (n)* functions using Eq. (20) and select the function having the highest correlation coefficient
18. Estimate the regression coefficients of the selected *CPU (n)* function using Eq. (21-23)
19. Estimate the optimal training set size n^* by Eq. (16-19) (according to the selected *Best_Function*)
20. If $n^* > n_{max}$ then
21. $n^* = n_{max}$
22. End if
23. Induce the classification model *M (n*)* from n^* examples
24. Measure the error rate *err(n*)* of *M (n*)* on the score set S
25. Compute the total cost of the projective sampling strategy P^* using Eq. (2)

Figure 1 The pseudo-code of the projective sampling algorithm

In this paper, we have performed experiments with the following widely used learning curve equations:

Logarithmic [5]: $err_{Log}(n) = a + b \log n$ (4)
Weiss and Tian [21]: $err_{WT}(n) = a + bn / (n + 1)$ (5)
Power Law [5]: $err_{PL}(n) = a \cdot n^b$ (6)
Exponential [5]: $err_{Exp}(n) = ab^n$ (7)

where a and b are two coefficients, which can be easily calculated from the available data points by applying the *least squares method* [14] to the appropriate transformations of n and *err*, which convert the above functions into the linear form $y = a' + b'x$. The corresponding transformations are shown in Table 1.

Table 1 Learning curve functions

Function	x (independent variable)	y (dependent variable)	a (intercept)	b (slope)
Logarithmic	log$n$	*err (n)*	a'	b'
n / (n + 1)	n / (n + 1)	*err (n)*	a'	b'
Power Law	log$n$	*log (err (n))*	*Exp (a')*	b'
Exponential	n	*log (err (n))*	*Exp (a')*	*Exp (b')*

Based on the run-time results presented in [15], we try to fit the following two equations to the run-time data:

Linear: $CPU_L(n) = d \cdot n$ (8)

Power law: $CPU_{PL}(n) = c \cdot n^d$ (9)

Where c and d are regression coefficients. The intercept of the linear run-time curve in Eq. (8) is assumed to be zero, since we cannot induce a classification model from zero examples.

The statistical expressions used by us for estimating the correlation and regression coefficients from k data points are given in the Appendix.

4.3 Finding the Optimal Number of Training Examples

The optimal training set size n^* is estimated based on the best *err (n)* and *CPU (n)* functions selected by the projective sampling algorithm in Figure 1 above. For the sake of simplicity, we will assume here that the run-time curve is approximated by the linear function shown in Eq. (8) above, since in our experiments, the correlation coefficient of the linear run-time curve was almost always higher than the correlation coefficient of the power law curve represented by Eq. (9). Substituting the learning curve Equations (4-7) and the run-time Equation (8) into Eq. (1) results in the following expressions for the Total Cost as a function of the training set size n, respectively:

$Total\ Cost_{Log}(n) = n \cdot C_{tr} + d \cdot n \cdot C_{time} + |S| \cdot C_{err} \cdot (a + b \log n)$ (10)

$Total\ Cost_{WT}(n) = n \cdot C_{tr} + d \cdot n \cdot C_{time} + |S| \cdot C_{err} \cdot (a + bn / (n+1))$ (11)

$Total\ Cost_{PL}(n) = n \cdot C_{tr} + d \cdot n \cdot C_{time} + |S| \cdot C_{err} \cdot an^b$ (12)

$Total\ Cost_{Exp}(n) = n \cdot C_{tr} + d \cdot n \cdot C_{time} + |S| \cdot C_{err} \cdot ab^n$ (13)

The above equations do not include the run-time cost of approximating the learning and run-time curves (represented by the second factor in Eq. 2), since this cost is incurred *before* we can estimate the optimal training set size. While the data and induction costs in Eqs. (10-13) are non-decreasing functions of n (as long as d is positive), it can be easily shown that the error cost is a non-increasing function of n as long as b is negative in the Logarithmic, Weiss-Tian, and Power Law functions and positive, but smaller than one, in the Exponential function. Consequently, an optimal trade-off between these three costs should exist. To find the optimal training set size n^* that minimizes the above cost function we need to compute the first derivative of Total Cost (n) and then set it to zero:

$$\frac{d}{dn} Total\ Cost(n) = 0 \qquad (14)$$

Substituting Equations (10-13) into Eq. (14) and assuming that the relative cost ratio is $R = C_{err} / C_{tr}$ we obtain the following expressions for the optimal training set size n^*[1]:

$$n_{Log}^* = -\frac{R \cdot |S| \cdot b}{1 + dC_{time}} \qquad (16)$$

$$n_{WT}^* = \sqrt{\frac{-R|S|b}{1 + dC_{time}}} - 1 \qquad (17)$$

$$n_{PL}^* = \left(\frac{-1 - dC_{time}}{|S| \cdot R \cdot a \cdot b} \right)^{\frac{1}{b-1}} \qquad (18)$$

$$n_{Exp}^* = \log_b \left(\frac{-dC_{time} - 1}{R|S| a \ln b} \right) \qquad (19)$$

Using equation-specific restrictions imposed on the values of a and b, one can show that n^* is the *global minimum* of Total Cost (n). For example, in the Power Law function where a can take only non-negative values and b cannot be positive , the expression abn^{b-1} is a monotonic non-decreasing function of n and, consequently, the first derivative of Total Cost is a monotone non-decreasing function of n. This implies that Total Cost (n) is a convex function and thus, according to [1], the point n^*, where $dTotal\ Cost\ (n)\ dn = 0$ is a global minimum. A similar line of argument can be followed for the other three equations above. Equations (16-19) can be easily adapted to the case of negligible induction costs by setting the value of C_{time} to zero.

The effectiveness of the proposed projective sampling procedure based on the optimal training set size estimated by Eqs. (16-19) is evaluated empirically in the next section using a decision-tree classification algorithm and 10 benchmark datasets of variable dimensionality.

5. EMPIRICAL RESULTS

5.1 Design of Experiment

Decision-tree learning is known as one of the most popular classification methods [6]. To ensure maximum

[1] For simplicity, we use natural algorithms, where appropriate

repeatability of the results presented in this paper, we have used a publicly available and free data mining environment called RapidMiner [12] with the learning operator DecisionTree (based on C4.5 [16]). In all our experiments described below, we have used the default settings of RapidMiner operators unless indicated otherwise.

Similarly to [21], the datasets were randomly partitioned into 25%-50% of test examples and 50%-75% of examples potentially available for training using the SplitChain operator. Only in one dataset (Census-Income), the original data and test files were used. For the sake of convenience, the sampling increment n_δ was set to 1% of the maximum possible training set size though in general, as indicated in sub-section 4.1 above, the number of examples acquired in each sample can be determined without knowing the maximum amount of available training examples. Thus, the actual percentage of training examples (out of the examples remained for training) was varied in increments of 1% at a time using the step_fraction parameter of the LearningCurve operator. To increase the statistical significance of our results, the error rate of each percentage is based on averages over 10 random partitions of the training set using a local random seed between 1 and 10. The optimal amount of training examples was estimated using the projective sampling procedure presented in Section 4 above for a set of Cost Ratio (R) and CPU Cost (C_{time}) values.

In our experiments, we have analyzed ten datasets described in Table 2. Most datasets were obtained from the UCI Machine Learning Repository [1] except for Census-Income (UCI KDD Archive [8]) and Physics (KDD Cup 2004 [3]). Four "small" datasets have between 600 and 1000 records, three "medium size" datasets contain several thousands of records, and three "large" datasets have between 32k and 299k records. We believe that it is important to include "small-size" datasets in the sampling experiments, since the problem of expensive data acquisition may be especially acute in datasets, which are inherently small.

The default settings of the DecisionTree operator were modified in accordance to the specific characteristics of some datasets. Thus, Gain Ratio was used as the Attribute Selection Criterion in Adult, Breast Cancer, Census-Income, Chess, Mushroom, Physics, and Vehicle datasets. The Minimum Leaf Size in three "large" datasets (Adult, Census-Income, and Physics) was increased from 2 to 100. It is important to emphasize that the goal of this study is to find the optimal number of training examples for given settings of a classification algorithm rather than optimizing the settings of a given algorithm.

Table 2. Datasets Description

Dataset	Number of Attributes	Total Size	Potentially Training Examples	Test Examples
Adult	14	32,561	16,281	16,280
Breast Cancer	10	699	525	174
Census-Income	41	299,285	199,523	99,762
Chess	36	3,196	2,397	799
German	20	1,000	750	250
Hypo-thyroid	25	3,163	2,372	791
Mush-room	22	8,124	6,093	2,031
Physics	78	50,000	25,000	25,000
Soybean large	35	683	512	171
Vehicle	18	846	635	211

5.2 Modeling the Learning Curves

Projected and actual learning curves of ten datasets are compared in Figure 2. The number of data points used to estimate each projected curve is shown in parentheses. One of the curves is obtained using the projective sampling algorithm presented in Section 4 above and the other one is always based on 20 data points. The projected learning curve equations estimated by the proposed algorithm are shown in Table 3.

Despite their inherent noisiness, the actual learning curves are generally characterized by diminishing returns in the accuracy rate for increasing the training set size. Most of them can be easily divided into the two regions identified by [15], a sloping portion and a plateau, except for the three datasets (Adult, Census Income, and Soybean), which have two distinct plateaus with a relatively sharp accuracy increase between them. When the final plateau starts within the first 20% of the available training data (represented by 20 data points), the projected curve fits the actual one quite well, as in the case of Hypothyroid, Mushroom, and Vehicle datasets. In those datasets, we expect the projected training set size n* based on 20 data points to be very close to the optimal one (e.g., see the Hypothyroid chart in Figure 3) For other datasets, the 20-point projected learning curve usually over-estimates the actual accuracy, since the slope is going to considerably decrease with additional examples. Consequently, such projected curves should also over-estimate the optimal amount of training data (e.g., see the Breast Cancer chart in Figure 3). The predictive bias of curves estimated from a smaller number of data points (3-

14) is less predictable: sometimes they underestimate the true accuracy (e.g., in the Adult dataset) and sometimes they over-estimate it (like in Chess).

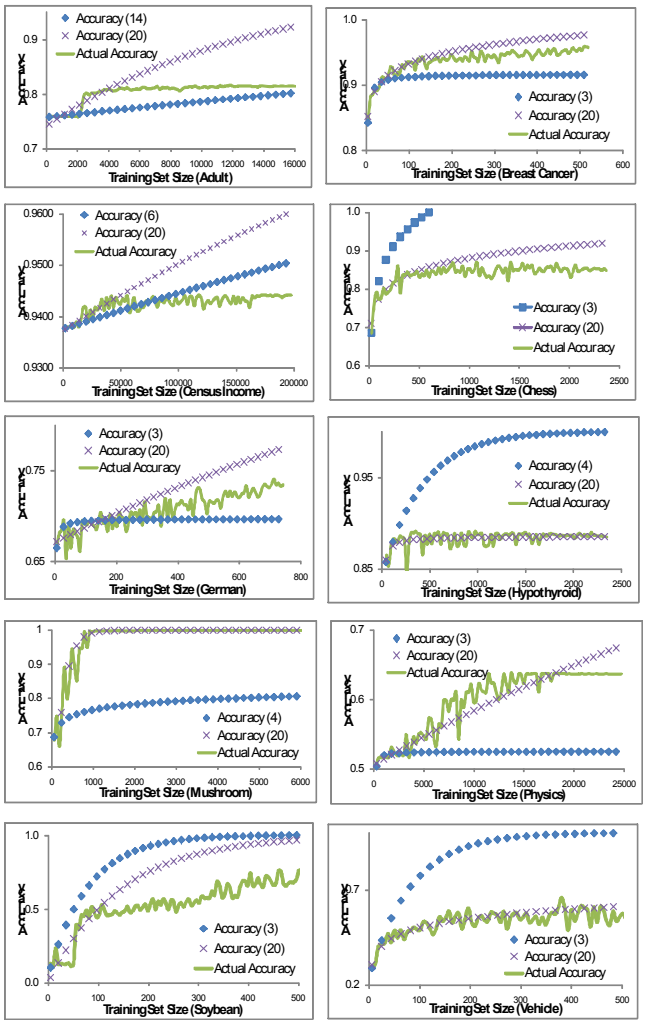

Figure 2 Projected and Actual Learning Curves

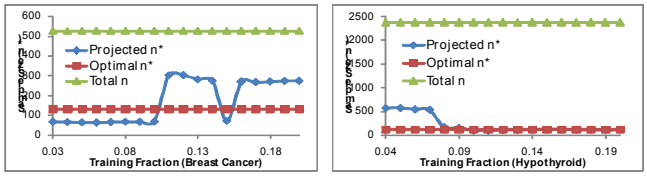

Figure 3 Optimal Sample Size *n

5.3 Projective Sampling Utility

As indicated in Section 2 above, our primary goal is to find a sampling strategy that minimizes the total cost of a classification process for a given dataset. In our experiments, we have compared the utility of the following sampling schedules:

- *Uniform sampling schedule*. Each iteration, the training set is incremented by the sampling increment n_δ equal to 1% of the maximum possible training set size n_{max}. The sampling stops after the first observed increase in the total cost or when n_{max} is reached.
- *Geometric sampling schedule*. The training set size starts with the sampling increment n_δ and doubles each iteration. The stopping conditions are identical to the uniform schedule.
- *Straw man strategy*. All potentially available training data is used for classification.
- *Projective sampling schedule*. This is the algorithm presented in Section 4 above. The sampling increment n_δ is equal to 1% of the maximum possible training set size n_{max}.
- *Optimal strategy*. This is the minimal classification cost that would be incurred if the actual cost of each sampling size could be known in advance. Only sample sizes that are increments of n_δ are considered.

Table 3 Projected Learning Curves

Dataset	Data Points	*Best_ Corr*	Selected Function	Curve Equation *err (n)*
Adult	14	-0.447	Exp	$0.243 \cdot 0.9999^n$
Breast Cancer	3	-0.982	n/n+1	$0.534 - 0.449 \cdot n/(n+1)$
Census-Income	6	-0.654	Exp	$0.062 \cdot 0.999999^n$
Chess	3	-0.999	Log	$0.624 - 0.097 \cdot \log n$
German	3	-0.992	n/n+1	$0.574 - 0.270 \cdot n/(n+1)$
Hypo-thyroid	4	-0.765	Exp	$0.159 \cdot 0.998^n$
Mush-room	4	-0.381	Power	$0.484 \cdot n^{-0.105}$
Physics	3	-0.976	n/n+1	$5.779 - 5.303 \cdot n/(n+1)$
Soybean large	3	-0.775	Exp	$0.953 \cdot 0.988^n$
Vehicle	3	-0.893	Exp	$0.769 \cdot 0.988^n$

5.3.1 Utility without induction cost.

Figure 4 compares the various sampling schedules without considering the induction cost. The left-hand chart shows the average total cost over ten datasets for low Cost Ratios (up to 1,000), whereas the right-hand chart shows the results for high Cost Ratios (between 5,000 and 50,000). When the Cost Ratio is low, the Straw Man strategy is much more expensive than any other schedule. The geometric strategy is the best up to the Cost Ratio of 100, but for higher values of the Cost Ratio, the projective strategy outperforms the progressive schedules though its cost is still significantly higher than the optimal strategy. Only for the highest Cost Ratio of 50,000 the Straw Man strategy becomes slightly cheaper. The two worst strategies for the high values of the Cost Ratio are the uniform and the geometric schedules.

The detailed results of each schedule as a function of the Cost Ratio are shown in Figure 5. The chart explains why the projective strategy outperforms on average both the uniform and the geometric schedules for a wide range of Cost Ratios. In nearly all datasets, the projective strategy performs at least as good as the best one of these two progressive sampling schedules. Thus, in the Adult, Census, Hypothyroid, and Soybean datasets, the projective performance is very close to the uniform sampling, whereas in the remaining datasets, except German and Physics, its performance is as good as the geometric sampling. In the German dataset, the projective strategy outperforms both progressive sampling schedules and its cost is very close to the optimal strategy. In the Physics dataset, the projective strategy outperforms only the uniform schedule.

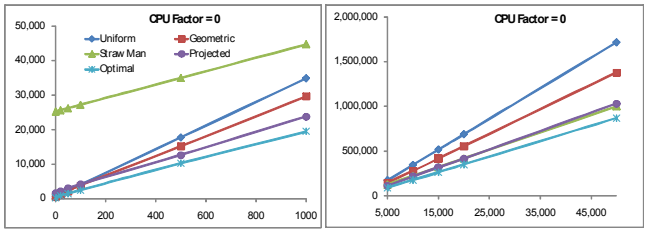

Figure 4 General comparison of sampling schedules without induction cost (*X* Axis –Cost Ratio, *Y* Axis – Average Total Cost)

5.3.2 The effect of induction cost on utility.

Figure 6 shows the average total cost of each sampling schedule over all datasets for the CPU factor C_{time} of 1. For low Cost Ratios (less than 5,000), the geometric and the uniform schedules are much better than the projective strategy. However, the projective strategy still outperforms these two progressive schedules for higher values of the Cost Ratio.

The highest run-time cost used in [21] is 200 per one millisecond of CPU time, though using the CPU factor of 200 in Eq. (2) means that one millisecond of CPU run-time costs the same as acquisition of 200 examples, which seems rather unlikely considering the continuously declining costs of computing power. In any case, as the CPU factor increases up to the value of 10 and higher, the projective sampling becomes less attractive than the uniform and geometric strategies for any Cost Ratio due to the substantial run-time cost of inducing multiple models in the process of estimating the learning curves.

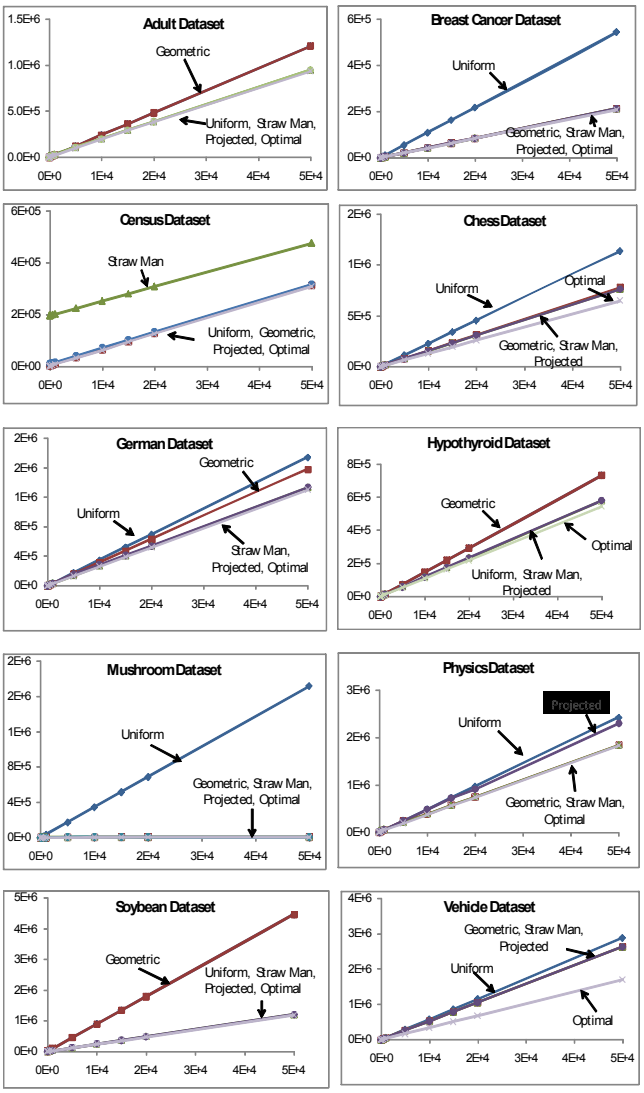

Figure 5 Detailed comparison of sampling schedules without induction cost (*X* Axis –Cost Ratio, *Y* Axis – Total Cost)

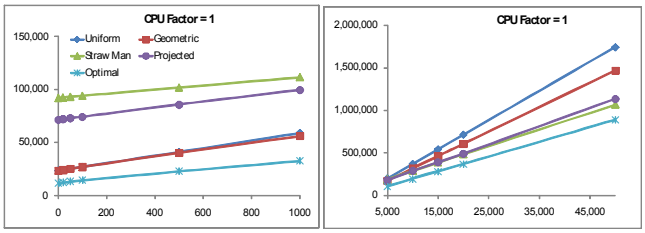

Figure 6 The average total cost for CPU factor = 1 (*X* Axis – Cost Ratio, *Y* Axis – Average Total Cost)

6. CONCLUSIONS

This paper presents a novel, projective sampling strategy, which improves the utility of the classification process by fitting a function to a partial learning curve and then using it to analytically estimate the optimal training set size. The proposed methodology is successfully evaluated on ten benchmark datasets of variable size using a popular

decision-tree classification algorithm. The results show that when the induction costs can be ignored, projective sampling outperforms, on average, the alternative, progressive sampling strategies. Future research may include further optimization of projective sampling schedules, especially under substantial CPU costs, and extension of the proposed sampling strategy to cost – sensitive data mining algorithms. Projecting learning curves for non-random sampling and labeling techniques is another important task.

7. REFERENCES

[1] Asuncion, A. & Newman, D.J. 2007. UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Boyd S. and Vandenberghe L. 2004 Convex Optimization. Cambridge University Press.

[3] Caruana, R., Joachims, T., and Backstrom, L. 2004. KDD-Cup 2004: results and analysis. *SIGKDD Explor. Newsl.* 6, 2 (Dec. 2004), 95-108.

[4] Everitt, B. 2001 Statistics for Psychologists: An Intermediate Course. Lawrence Erlbaum Associates.

[5] Frey L. J. and Fisher, D. H. 1999. Modeling Decision Tree Performance with the Power Law. In Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics, 59-65.

[6] Han J. and Kamber, M. 2006 Data Mining: Concepts and Techniques. Second Edition, Morgan Kaufmann.

[7] Hastie, T., Tibshirani, R., and Friedman, J. 2003 The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Verlag.

[8] Hettich, S. and Bay, S. D. 1999 The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[9] John G. and Langley, P. 1996. Static versus dynamic sampling for data mining. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 367-370.

[10] Last, M. 2007. Predicting and Optimizing Classifier Utility with the Power Law. In Proceedings of the Seventh IEEE international Conference on Data Mining Workshops (October 28 - 31, 2007). ICDMW. IEEE Computer Society, Washington, DC, 219-224.

[11] Last, M. and Maimon, O. 2004. A Compact and Accurate Model for Classification. IEEE Trans. on Knowl. and Data Eng. 16, 2 (Feb. 2004), 203-215.

[12] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. 2006. YALE: rapid prototyping for complex data mining tasks. In Proceedings of the 12th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Philadelphia, PA, USA, August 20 - 23, 2006). KDD '06. ACM, New York, NY, 935-940.

[13] Minium, E.W., Clarke, R.C., and Coladarci, T. 1999. Elements of Statistical Reasoning. New York: John Wiley & Sons, Inc. 2nd Ed.

[14] Montgomery, D.C., Runger, G.C., Hubele, N.F. 2007. Engineering Statistics, John Wiley & Sons, Inc. 4th Edition.

[15] Provost, F., Jensen, D., and Oates, T. 1999. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (San Diego, California, United States, August 15 - 18, 1999). KDD '99. ACM, New York, NY, 23-32.

[16] Quinlan, J. R. 1993 C4.5: Programs for Machine Learning. Morgan Kaufmann.

[17] Saar-Tsechansky M. and Provost, F. 2004. Active Sampling for Class Probability Estimation and Ranking. Machine Learning 54, 2 (Feb. 2004), 153-178.

[18] Sheng, V. S. and Ling, C. X. 2007. Partial example acquisition in cost-sensitive learning. In Proceedings of the 13th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (San Jose, California, USA, August 12 - 15, 2007). KDD '07. ACM, New York, NY, 638-646.

[19] Sheng, V. S., Provost, F., and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA, August 24 - 27, 2008). KDD '08. ACM, New York, NY, 614-622.

[20] Singh, S. 2005 Modeling Performance of Different Classification Methods: Deviation from the Power Law. Project Report, Department of Computer Science, Vanderbilt University, USA (April 2005).

[21] Weiss, G. M. and Tian, Y. 2008. Maximizing classifier utility when there are data acquisition and modeling costs. *Data Min. Knowl. Discov.* 17, 2 (Oct. 2008), 253-282.

8. APPENDIX

Pearson's correlation coefficient [13]:

$$Corr_k = \frac{\sum_{i=1}^{k} x_i \cdot y_i - \frac{\sum_{i=1}^{k} x_i \cdot \sum_{i=1}^{k} y_i}{k}}{\sqrt{\left(\sum_{i=1}^{k} x_i^2 - \frac{(\sum_{i=1}^{k} x_i)^2}{k}\right)\left(\sum_{i=1}^{k} y_i^2 - \frac{(\sum_{i=1}^{k} y_i)^2}{k}\right)}} \quad (20)$$

The least squares estimate of the slope [13]:

$$\hat{b} = \sum_{i=1}^{k} y_i x_i - \frac{\left(\sum_{i=1}^{k} y_i\right)\left(\sum_{i=1}^{k} x_i\right)}{k} \Bigg/ \sum_{i=1}^{k} x_i^2 - \frac{\left(\sum_{i=1}^{k} x_i\right)^2}{k} \quad (21)$$

The least squares estimate of the intercept [13]:

$$\hat{a} = \overline{y} - \hat{b}\overline{x} \quad (22)$$

The slope estimate with zero intercept [4]:

$$\hat{b} = \sum_{i=1}^{k} y_i x_i \Big/ \sum_{i=1}^{k} x_i^2 \quad (23)$$