

Sampling Approach to Resource Light Data Mining

N. Abe, C. Apte, B. Bhattacharjee, K. Goldman, J. Langford, B. Zadrozny*

Abstract

We present a sampling approach to realizing high accuracy predictive data mining with light memory requirement, and describe a host of mining methods derived from our approach. Our approach is an ensemble-based scheme, which by iteratively sampling a series of small data sets from a potentially huge data set, and running a data mining algorithm on them in a piecemeal fashion, significantly lowers the memory requirement. Specifically, we present a number of sampling methods, based on rejection sampling and uncertainty sampling techniques, which not only reduce the amount of data to be used for mining, but achieve state-of-the-art predictive performance overall. We also discuss our efforts in applying the proposed approach to secure federated data mining by embedding them in a cryptographically secure but resource limited processor.

1 Introduction

With increasing success and widening application areas of data mining, the requirements and expectations for mining algorithms are becoming more varied and demanding. Of various requirements that arise in real world applications, the ability to perform under restricted computation resources, such as memory and computation time, is considered most important and has been well studied from various view points [13]. Examples of past work in this general area include various algorithmic approaches to developing efficient mining algorithms (e.g. [9]) and “stream mining” which has received significant attention in the recent years (c.f. [4, 7]). Also, some recent work has applied random sampling to attain comparable predictive performance using less data [5].

In this paper, we present a sampling approach to realizing high accuracy predictive data mining with minimal memory requirement, and describe a host of particular mining methods derived from our approach. Our approach belongs to the so-called ensemble methods, which by iteratively sampling a series of small data sets

from a potentially huge data set, and running a data mining algorithm on them, minimizes the memory requirement. Specifically, we present a number of sampling methods which not only reduce the amount of data to be used in each iteration, but achieve state of the art predictive performance.

Our approach is to be contrasted to earlier work using random sampling [8, 5], in that we use sampling techniques that deliberately alter the sampling distribution.¹ In particular, the sampling methods that we employ in our iterative mining methods consist of two main techniques and combinations thereof. One is the technique of using rejection sampling with *cost proportionate example weighting*, which was proposed and demonstrated to be effective in the so-called cost-sensitive classification learning scenario, in which the goal of a learning algorithm is to minimize the cost of misclassification error [17]. The other is *uncertainty sampling*, a common notion in the literature on query or active learning, which selectively samples examples for which the predictions made by the ensemble hypothesis at the current stage are uncertain.

We discuss three particular instances of our general approach. One is the cost-sensitive learning problem mentioned above, and we describe methods based on cost-proportionate rejection sampling, proposed in [17]. The second is also that of cost-sensitive learning, but a more general setting in which costs may be unknown and may need to be estimated using a regression procedure. For this problem we propose sampling methods that are based on the uncertainty of cost sensitive learning. Finally, we consider a different problem setting of *anomaly detection*, in which the goal of the learning method is not to classify the label of a new instance using labeled training data, but rather to detect new examples that are *anomalous* based on unlabeled training data. For this problem, we propose a method based on uncertainty sampling (or query learning), which is made possible via a novel reduction of anomaly detection to classification.

*N. Abe, C. Apte, B. Bhattacharjee, K. Goldman and B. Zadrozny's address: IBM T. J. Watson Research Center, Yorktown Heights, NY 10598. John Langford's present address: Toyota Technological Institute at Chicago, 427 East 60th Street, Second Floor - Press Building, Chicago, IL 60637.

¹There have been some past work using biased sampling, but the techniques and emphasis therein are quite different from ours, e.g. [11].

2 Cost sensitive learning via sampling with cost-proportionate example weights

2.1 Problem Formulation: Cost-sensitive learning *Cost-sensitive learning* refers to the problem of learning classifiers in which non-uniform costs are associated with different types of misclassification. For many data mining problems of practical importance, it is absolutely necessary to take this aspect into account, if one hopes to obtain any reasonable result. Examples of such applications include, among others, credit rating, fraud detection, churn analysis, and targeted marketing. For fraud detection, for example, frauds are so rare that straightforward application of classification can result in a classifier that always predicts the most common class (non-fraud), but when they occur, frauds are a magnitude more costly and thus it is important to classify them correctly.

A simple and popular formulation of the cost sensitive learning problem is via the use of a cost matrix. A cost matrix, $C(i, j)$, specifies how much cost is incurred when misclassifying a label j as i , and the goal of a cost-sensitive learning method is to minimize the expected cost. The usual classification problem is a special case of this, in which $C(i, j) = 1$ if $i \neq j$ and $C(i, j) = 0$ otherwise, and the goal translates to minimizing the number of misclassifications. This formulation is not applicable in many situations, however, due to misclassification costs that depend on particular instances. This fact was noted by Zadrozny and Elkan [16], who formulated this problem using a more general form of cost function, $C(i, j, x)$, namely one that allows dependence on the instance x .

A useful observation, which was made and exploited in [17] is that, for the binary label case, the above formulation in terms of cost per example, $C(i, j, x)$, can be reduced to a simpler formulation in terms of an importance number per example. This is possible by associating a single number indicating the importance of an example (x, j) , given by $C(1, j, x) - C(0, j, x)$. This conversion allows us to formulate the problem, in which we assume that examples are drawn independently from a distribution D with domain $X \times Y \times C$ where X is the input space to a classifier, Y is a (binary) output space and $C \subset [0, \infty)$ is the importance (extra cost) associated with mislabeling that example. The goal is to learn a classifier $h : X \rightarrow Y$ which minimizes the expected cost,

$$E_{x,y,c \sim D}[c I(h(x) \neq y)]$$

given training data of the form: (x, y, c) , where $I(\cdot)$ is the indicator function that has value 1 in case its argument is true and 0 otherwise.

A basic folk theorem, which was formalized and

proved in [17] states that if we have examples drawn from the distribution:

$$\hat{D}(x, y, c) \equiv \frac{c}{E_{x,y,c \sim D}[c]} D(x, y, c)$$

then optimal error rate classifiers for \hat{D} are optimal cost minimizers for data drawn from D .

2.2 Proposed Methods

2.2.1 Cost-proportionate rejection sampling

The “folk” theorem mentioned above suggests an obvious method of converting from one distribution of examples to another to obtain a cost-sensitive learner by re-weighting the example distribution. It turns out, however, straightforward sampling methods such as re-sampling with replacement do not work well in this case. We make use of a sampling scheme called rejection sampling [14] which allows us to draw samples independently from the distribution \hat{D} , given samples drawn independently from D . In rejection sampling, samples from \hat{D} are drawn by first drawing samples from D , and then keeping the sample with probability proportional to \hat{D}/D . Here, we have $\hat{D}/D \propto c$, so we accept an example with probability c/Z , where Z is some constant chosen so that $\max_{(x,y,c) \in S} c \leq Z$, leading to the name *cost-proportionate rejection sampling*. Rejection sampling results in a set S' which is generally smaller than S . Furthermore, because inclusion of a sample in S' is independent of other samples, and the samples in S are drawn independently, we know that the samples in S' are distributed independently according to \hat{D} .

Using cost-proportionate rejection sampling to create a set S' and then using a learning algorithm $A(S')$ is guaranteed to produce an approximately cost-minimizing classifier, as long as the learning algorithm A achieves approximate minimization of classification error.

2.2.2 Cost-proportionate rejection sampling with aggregation (costing)

From the same original training sample, different runs of cost-proportionate rejection sampling will produce different training samples. Furthermore, the fact that rejection sampling produces very small samples means that the computational time required for learning a classifier is generally much smaller.

We can take advantage of these properties to devise an ensemble learning algorithm based on repeatedly performing rejection sampling from S to produce multiple sample sets S'_1, \dots, S'_m , and then learning a classifier for each set. The output classifier is the average over all learned classifiers. We call this technique “costing”.

Costing(Learner A , Sample Set S , count t , normalization constant Z)

1. **For** $i = 1$ **to** t **do**
 - (a) S' = rejection sample from S with probability c/Z .
 - (b) **Let** $h_i \equiv A(S')$
2. **Output** $h(x) = \text{sign}\left(\sum_{i=1}^t h_i(x)\right)$

Figure 1: The ‘‘Costing’’ algorithm

Note that despite the extra computational cost of averaging, the overall computational time of costing is generally much smaller than for a learning algorithm using sample set S (with or without weights). In particular, this is the case if the component learning algorithm being employed has running time that is superlinear in the number of examples, which is generally the case.

2.3 Empirical evaluation

2.3.1 The datasets used We show empirical results using two real-world datasets from the direct marketing domain. This first data set we use is the well-known and challenging dataset from the KDD-98 competition [3]. The dataset contains data on people who have made donations in the past to a particular charity. The decision-making task is to choose which donors to mail solicitations for donation, with the goal of maximizing the total profit obtained in the mailing campaign. Recall that the importance of each example is the absolute difference in profit between mailing and not mailing an individual. For positive examples (respondents), the importance varies from \$0.32 to \$199.32 (the donation amounts minus the mailing cost of \$0.68), and for negative examples (non-respondents), it is always the mailing cost, \$0.68. The dataset is divided in a fixed way into a training set and a test set, each of size approximately 96000. The second data set, the DMEF-2 dataset [1], contains customer buying history for 96551 customers of a nationally known catalog. The decision-making task is to choose which customers should receive a new catalog so as to maximize the total profit. Information on the cost of mailing a catalog is not available, so we fixed it at \$2. The purchase amount for customers who respond varies from \$3 to \$6247, so the importance varies from \$1 to \$6246 for positive examples. For negative examples, it is fixed at \$2. We divided the data set in half to create a training set and a test set.

2.3.2 Experimental results Table 1 shows the results of applying resampling with replacement to implement the weighted sampling to the KDD-98 and DMEF-2 data using resampled training sets of different sizes. (A range of base learning algorithms were used: Naive Bayes (NB), Boosted Naive Bayes (BNB), C4.5, and Support Vector Machines (SVM).) For each size, we repeat the experiments 10 times with different resampled sets to get mean and standard error (in parentheses). The training set profits are on the original training set from which we draw the resampled sets.

These results here confirm that straightforward application of resampling to implement the weighted sampling can result in very poor performance, as we remarked earlier. The poor performance of resampling is essentially due to overfitting. When there are large differences in the magnitude of importance weights, it is typical for an example to be picked twice (or more). Examples which appear multiple times in the training set of a learning algorithm can defeat complexity control mechanisms built into learning algorithms. In Table 1, we see that as we increase the resampled training set size and, as a consequence, the number of duplicate examples in the training set, the more overfitting we see particularly for C4.5.

Table 2 shows the results of applying costing on the KDD-98 and DMEF-2 datasets, with the same base learners as before, and $Z = 200$ or $Z = 6247$, respectively. (We also show how the performance progresses as a function of the number of sampling iterations in Figure 6 in Appendix.) In the KDD-98 case, each resampled set has only about 600 examples, because the importance of the examples varies from 0.68 to 199.32 and there are few ‘‘important’’ examples. With $t = 200$, the C4.5 version yields profits around 15000 dollars, which is exceptional performance for this dataset. In the DMEF-2 case, each set has only about 35 examples, because the importances vary even more widely (from 2 to 6246) and there are even fewer examples with a large importance than in the KDD-98 case.

These results are particularly remarkable from the point of view of reduction in memory requirement: only hundreds of examples need to be processed in each iteration, as compared to the original data size of approximately 100 thousands. Even with this drastic reduction in memory requirement, costing manages to achieve state-of-the-art predictive performance, at least in the application domain of targeted marketing used for our empirical evaluation.

KDD-98:						
	1000		10000		100000	
	Training	Test	Training	Test	Training	Test
NB	11251 (330)	10850 (325)	12811 (155)	11993 (185)	12531 (242)	12026 (256)
BNB	11658 (311)	11276 (383)	13838 (65)	12886 (212)	14107 (152)	13135 (159)
C4.5	11124 (255)	9548 (331)	22083 (271)	7599 (310)	40704 (152)	2259 (107)
SVM	10320 (372)	10131 (281)	11228 (182)	11015 (161)	13565 (129)	12808 (220)

DMEF-2:						
	1000		10000		100000	
	Training	Test	Training	Test	Training	Test
NB	33298 (495)	34264 (419)	32742 (793)	33956 (798)	33511 (475)	34506 (405)
BNB	33902 (558)	30304 (660)	34802 (806)	31342 (772)	34505 (822)	31889 (733)
C4.5	37905 (1467)	24011 (1931)	67960 (763)	9188 (458)	72574 (1205)	3149 (519)
SVM	28837 (1029)	30177 (1196)	31263 (1121)	32585 (891)	34309 (719)	33674 (600)

Table 1: Profits on the KDD-98 and DMEF-2 datasets using resampling.

KDD-98:			
	1	100	200
NB	11667 (192)	13111 (102)	13163 (68)
BNB	11377 (263)	14829 (92)	14714 (62)
C4.5	9628 (511)	14935 (102)	15016 (61)
SVM	10041 (393)	13075 (41)	13152 (56)

DMEF-2:			
	1	100	200
NB	26287 (3444)	37627 (335)	37629 (139)
BNB	24402 (2839)	37376 (393)	37891 (364)
C4.5	27089 (3425)	36992 (374)	37500 (307)
SVM	21712 (3487)	33584 (1215)	35290 (849)

Table 2: Test set profits on the KDD-98 and DMEF-2 datasets using costing.

3 General cost sensitive learning via query learning

3.1 Problem Formulation: General cost-sensitive learning As in the previous section, we assume that a general cost function of the form $C(i, j, x)$ is given, and that the goal of a cost-sensitive learning method is to minimize the expected cost of its classification. Given this general formulation, a natural approach is to estimate the expected cost, and then output a label that minimizes the expected cost, that is, output i^* where

$$(3.1) \quad i^* = \arg \min_i \sum_j P(j|x)C(i, j, x).$$

If we now let $g(i, x)$ denote the expected cost function, namely $\sum_j P(j|x)C(i, j, x)$, the problem can be decomposed into two parts:

1. Apply a regression method to estimate $g(i, x)$.

2. Translate g into f such that

$$(3.2) \quad f(x) \simeq \arg \min_i g(i, x).$$

When labeled examples are so abundant that one cannot efficiently make use of all the data, applying query learning to perform intelligent filtering of data used for mining becomes a realistic alternative. Following this line of reasoning, Mamitsuka and Abe [10] introduced the idea of applying query learning to scaling up data mining algorithms, in the context of usual classification problems.² Here we extend this approach to the context of general cost sensitive learning.

Recall that the formulation of cost-sensitive learning we consider involves two parts: (1) regression and (2) classification, with the final goal of learning being the latter. This suggests the non-standard idea of applying uncertainty sampling for regression (1), based on uncertainty of classification (2). A very general schema of this is presented in Figure 2.

We propose particular methods in this general schema for the general cost-sensitive learning considered here. Two different methods are derived depending on the type of uncertainty measure employed for the uncertainty sampling, namely measures depending on (1) strictly \hat{f} , or (2) \hat{g} with the goal of estimating f .

3.2 Proposed Methods

3.2.1 Cost sensitive query learning with classification uncertainty

The first method we consider is

²Prior to their proposal, there have been a number of related methodologies known that used selection of sample based on progressively built models (c.f. [13]), but theirs was the first that applied the idea of query learning to this area.

General Cost-Sensitive Query Learning

1. Initialize for $k = 0$
 - 1.1. Obtain sub-sample S_0 via random sampling from S .
 - 1.2. Apply a regression method to S_0 to estimate \hat{g}_0 .
 - 1.3. Translate \hat{g}_0 into \hat{F}_0 .
2. For iteration $k = 1$ to T
 - 2.1. Obtain sub-sample S_k via uncertainty sampling using estimation uncertainty of \hat{f}_{k-1} .
 - 2.2. Apply a regression method on S_k to estimate \hat{g}_k .
 - 2.3. Define \hat{f}_k in terms of $\{\hat{g}_l | l = 0, \dots, k-1\}$.
3. Output \hat{f}_T as the final hypothesis.

Figure 2: General schema for cost-sensitive query learning.

CSQ-CU

Input Parameters:

Sample: S of size N ; Base regression method: R ;

Number of iterations: T ; Sample size for each iteration: $m (\ll N)$.

1. Initialize for $k = 0$
 - 1.1. Obtain S_0 of size m via random sampling from S .
 - 1.2. Apply R to S_0 to obtain \hat{g}_0 .
 - 1.3. Define $\hat{f}_0(x) = \arg \min_i \hat{g}_0(i, x)$
2. For iteration $k = 1$ to T
 - 2.1. For $l = 1$ to m
 - 2.1.1. Sample next example (i, x) from S
 - 2.1.2. Determine classification uncertainty $U(x)$ of x by:

$$M(x) = \text{margin}(\{\hat{f}_0, \dots, \hat{f}_{k-1}\}, x)$$
 - 2.1.3. Calculate acceptance probability $P(x)$ of x by:

$$P(x) = \text{gauss}(k/2, \sqrt{k}/2, (k + M(x))/2)$$
 - 2.1.4. Accept (i, x) to S_k with probability $P(x)$
 - 2.1.5. If accepted, obtain label (cost) l of (i, x) and add $\langle (i, x), l \rangle$ to S_k .
 - 2.2. Apply R on S_k to obtain \hat{g}_k .
Define $\hat{f}_k(x) = \arg \min_i \sum_{l=0, \dots, k} \hat{g}_l(i, x)$
3. Output final hypothesis $\hat{f}_T(x)$

Figure 3: Cost sensitive query learning with class uncertainty (CSQ-CU)

based on the uncertainty of classification. We use the notion of *margin* popular in the theory of classification. The margin defined on an example by an ensemble of classifiers is the difference between the number of classifiers voting for the most popular label and that for the second most popular label. In the simplest case involving two labels, this is just the difference between the number of classifiers voting for one label and the other. We translate the margin of an example to a sampling (acceptance) probability with the following reasoning. Consider for simplicity, the binary classification problem. Suppose that n classifiers are chosen randomly, each having probability one half of voting one way or another. Then the probability of obtaining k more votes for one label than the other is given by the binomial distribution for $(n, (n+k)/2)$ and can be approximated by a gaussian. Given that the observed difference is indeed k for a certain example, then by the Bayes theorem, the likelihood of the hypothesis that, for that example, the two labels are equally likely to be predicted by the *rest* of the classifiers, is proportional to the above probability. Assume further that an inclusion of any example is equally likely to contribute to the correct labeling for that example, by the classifier obtained in the current iteration. Then, for minimizing the classification error, it is optimal to include an example with acceptance probability proportional to this probability. The first method we propose, CSQ-CU in Figure 3, is based on this intuitive idea. In the pseudo-code given in Figure 3, note that we use $\text{margin}(\{\hat{f}_0, \dots, \hat{f}_{k-1}\}, x)$ to denote the margin defined on x as described above, and $\text{gauss}(\mu, \sigma, Z)$ to denote $1 - \int_{-\infty}^Z \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} dx$.

3.2.2 Cost sensitive query learning methods using confidence intervals

The second method we consider is based on the confidence associated with regression (1) for the purpose of making classification (2). Using an ensemble of regression functions, one can determine the variance of predicted values for an example, and hence the standard error of the mean for it. In the absence of an associated classification problem, it is unclear how this variance information should be translated into a probability value. In our current setting with an associated decision (classification) problem, however, there is a natural way to do so, using the idea of confidence interval. That is, by dividing the standard error of the regression estimates for a given example by the estimated difference between the cost of one class v.s. the other for that example, the so-called Z -value of the estimates can be obtained. This can then be transformed via gaussian to the probability that the decision as it stands now could be reversed. By a similar argument as in the case of classification

uncertainty, we see that the same example should be accepted with probability proportional to this probability. Our second method is based on the above idea, and its pseudo-code is exhibited in Figure 4. Note in the pseudo-code that we use $\text{conf}(\{\hat{g}_0, \dots, \hat{g}_{k-1}\}, x)$ to denote the Z -value as explained above, i.e. $Z(x) = \frac{\bar{g}(x)}{SE(x)}$, where $\bar{g}(x)$ denotes the average difference in predicted costs for x , i.e. average of $\hat{g}_k(x) = \hat{g}_k(0, x) - \hat{g}_k(1, x)$, $SE(x)$ denotes the standard error for this prediction, i.e. $SE(x) = \sqrt{\frac{\text{var}(\hat{g}_k, x)}{k}}$.

3.3 Empirical Evaluation We used the same two datasets, the KDD-98 and DMEF-2 datasets, as in the previous section for our evaluation. We compare the profits obtained by the two proposed methods, CSQ-CU and CSQ-CI, and those of the two obvious alternatives, namely using the base regression algorithm³ with no sampling and bagging (random sampling), both using approximately the same amount of computation time.

Table 3 gives the results of this comparison: the no sampling method was run with 30 thousand data, and the 3 sampling methods were run on 10 subsamples each of size 3 thousands. Again, each method was run over 15 randomized runs, and the average and standard error were calculated. (We also plot the profits obtained by various methods as a function of the number of sampling iterations in Figure 7 and Figure 8 in Appendix.) In total, they consumed roughly equivalent computation time – the no sampling method took about 17 and a half minutes on an IBM RS/6000 Model F80 (CPU Clock: 450MHz), whereas the sampling methods took roughly 15 minutes on the same machine. These results show that the proposed methods achieve significantly higher profits per a given computation time, as compared to both base regression method and bagging. It is especially interesting to note that the gain in predictive performance achieved by the proposed methods over No Sampling are much higher than that achieved by bagging. Bagging is known to be quite competitive in the literature, so the improvement over bagging is significant.

4 Anomaly Detection via Query Learning

4.1 Problem Formulation: Reduction of Unsupervised Learning to Classification

We begin by presenting a non-standard model of unsupervised learning. Assume that the data are drawn from some probability distribution U on an instance space X . The goal in this model is to choose a “good” partition π of the

³As the base regression method, we use the multivariate linear regression tree method implemented in the IBM ProbETM data mining engine [12, 2].

CSQ-CI

Input Parameters:

Sample: S of size N ; Base regression method: R ;

Number of iterations: T ; Sample size for each iteration: $m(\ll N)$.

1. Initialize for $k = 0$
 - 1.1. Obtain S_0 of size m via random sampling from S .
 - 1.2. Apply R to S_0 to obtain \hat{g}_0 .
 - 1.3. Define $\hat{f}_0(x) = \operatorname{argmin}_i \hat{g}_0(i, x)$
 2. For iteration $k = 1$ to T
 - 2.1. For $l = 1$ to m
 - 2.1.1. Sample next example (i, x) from S
 - 2.1.2. Determine confidence interval $Z(x)$ of x by:
 $Z(x) = \operatorname{conf}(\{\hat{g}_0, \dots, \hat{g}_{k-1}\}, x)$
 - 2.1.3. Calculate acceptance probability $P(x)$ of x by:
 $P(x) = \operatorname{gauss}(0, 1, Z(x))$
 - 2.1.4. Accept (i, x) to S_k with probability $P(x)$
 - 2.1.5. If accepted, obtain label (cost) l of (i, x) and add $\langle (i, x), l \rangle$ to S_k .
 - 2.2. Apply R on S_k to obtain \hat{g}_k .
- Define $\hat{f}_k(x) = \operatorname{argmin}_i \sum_{l=0, \dots, k} \hat{g}_l(i, x)$

3. Output final hypothesis $\hat{f}_T(x)$

Figure 4: Cost sensitive query learning with confidence intervals (CSQ-CI)

Method	KDD98	DMEF2
No Sampling(ProbE)	12379	33360 (1093)
Random(Bagging)	12412 (176)	33531 (1155)
CSQ-CI	12679 (209)	35341 (775)
CSQ-CU	13004 (142)	36440 (476)

Table 3: Profits (standard error) on the KDD-98 and DMEF2 datasets obtained by various methods using 30 thousand examples in total.

space X . The partition π divides the space X into two subspaces which we call, π and $X - \pi$. A “good” partition π contains “most” of the points while minimizing the size of π . We then define the notion of error for this form of unsupervised learning as follows.

$$e_{U,P}(\pi) = \frac{1}{2} \left(\Pr_{x \sim U}(x \notin \pi) + \Pr_{x \sim P}(x \in \pi) \right)$$

Here we used P to denote the “background” distribution over X , such as the uniform distribution for a finite X . Note that the minimization of the first term attempts to include the set of likely events, while the second term forces exclusion of the unlikely events. Intuitively, the goal here is to find a “small” (w.r.t. P) set π which contains “most” of the data (w.r.t U).

Next, we review a *robust* model of supervised learning (classification), in which no assumption is made

about the target. We assume a distribution D over the input space X and the binary output space $Y = \{0, 1\}$. The goal is to find a classifier $h : X \rightarrow Y$ with a small true error rate:

$$e_D(h) \equiv \Pr_{x,y \sim D}(h(x) \neq y)$$

We can directly connect these two models by the following *reduction*. Recall that, for our formulation of unsupervised learning, we wish to solve the following classification problem: decide whether a point is drawn from an underlying distribution of P or a distribution U . Now define a distribution $D(x, y)$ by the following program:

1. Flip a coin with bias 0.5.
2. If “heads”
 - (a) then draw $x \sim U$ and return $(x, 1)$
 - (b) else draw $x \sim P$ and return $(x, 0)$

It is easy to verify that the following equivalence holds.

$$e_D(c_\pi) = e_{U,P}(\pi)$$

where we let c_π denote the classifier corresponding to the partition π , namely: $c_\pi(x) = 1 \Leftrightarrow x \in \pi$

Q-bad(Learner A , Samples S_{real} , S_{syn} , count t , threshold θ)

1. **Let** $S = \{(x, 0) | x \in S_{real}\} \cup \{(x, 1) | x \in S_{syn}\}$.
2. **For** $i = 1$ **to** t **do**
 - (a) $M(x) = \text{margin}(\{h_0, \dots, h_{i-1}\}, x)$.
 - (b) $S' =$ **rejection sample from S with sampling probability:**

$$\text{gauss}(i/2, \sqrt{i}/2, (i + M(x))/2)$$

- (c) **Let** $h_i \equiv A(S')$

3. **Output** $h(x) = \text{sign}(\sum_{i=1}^t h_i(x) - t\theta)$

Figure 5: Anomaly detection method using query learning

4.2 Anomaly Detection via Query Learning:

Q-bad Simple though it may seem, the relationship established above is useful, since in some sense it allows us to transfer much of what is known about classification to unsupervised learning, including theory and specific algorithms. Here, we propose to apply the idea of query learning, which is a notion that normally makes sense only for supervised learning (classification), to anomaly detection. This gives rise to the simple procedure, which we call Q-bad (Query-based anomaly detection), given in Figure 5. Note in the pseudo code that we make use of *margin* and *gauss*, as defined in Section 2.

4.3 Empirical Evaluation

We evaluated the proposed method using the network intrusion data from KDD cup 99, which is available from the UCI data repository (<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>). We used only the data corresponding to the *normal* connections in the training data, since we are using it to evaluate an anomaly detection method. More concretely, we used the approximately 200 thousands normal connections included in the so-called “10 %” data (“kddcup_data_10_percent.gz”). As synthetic data, we generated data with respect to the product distribution of the empirical marginals for each of the attributes. The marginals were estimated using single gaussians for numerical attributes, and empirical frequencies for the nominals.

We compared the performance of the proposed method against the performance of two other methods reported in the literature, one existing anomaly detection method based on density estimation, and one

Method	KDD cup 99 data
Q-bad	56610 (3627)
Parzen Window	62952
Cup winner	72500

Table 5: Costs (standard error) on the KDD-99 data obtained by Q-bad (40 iterations), Parzen Window, and KDD cup 99 winner.

supervised learning method. The anomaly detection we compare against is based on the so-called Parzen Window method, and is documented in [15]. The supervised learning method is the winner of the KDD cup 99 competition (c.f. [6].) We note that the supervised learning method is solving a different problem: (1) it uses labeled data; (2) it is solving a multi-class classification problem with the labels being one of $\{normal, probe, DOS, U2R, R2L\}$. Note that the anomaly detection methods output only two labels, “normal” and “intrusion”. We therefore use two different cost matrices to evaluate the costs for the two types of methods, as shown in Table 4.⁴

Table 5 gives the results of this experimentation. What is shown is the total cost incurred on the entire test data, consisting of about 311 thousand examples, averaged over 10 randomized runs.⁵ The number in parentheses is the standard error. The result for the KDD cup 99 winner is the result of their submitted classifier, hence it is not averaged. No information on the standard error for the Parzen Window method was readily available in the literature. We also exhibit in Figure 9 in Appendix how the costs obtained by Q-bad change as the sampling iterations progress (for the two methods being compared we do not have the corresponding information.)

The Parzen Window method is a non-parametric density estimation method that basically puts a gaussian distribution around each of the training examples. As such it is an instance of a case based estimation method, and is considered to be well suited and competitive for this type of application (i.e. anomaly detection). Note, however, that it requires the storage of all the training data, and hence its requirement on memory is quite heavy. This is to be contrasted with our proposed method, which by virtue of its iterative sampling approach, is extremely memory light.

⁴The costs calculated using the two cost matrices are not identical, but the difference in cost due to the different cost matrices used is negligible compared to the differences between different methods.

⁵We note that in getting this result, the threshold parameter θ to our method was optimized, though not extensively.

(a) Anomaly Detection:

predicted : true	0 (normal)	1 (probe)	2 (DOS)	3 (U2R)	4 (R2L)
0 (normal)	0	1	2	3	4
1 (intrusion)	2	0	0	0	0

(b) Multi-class Classification:

predicted : true	0 (normal)	1 (probe)	2 (DOS)	3 (U2R)	4 (R2L)
0 (normal)	0	1	2	3	4
1 (probe)	1	0	1	2	2
2 (DOS)	2	2	0	2	2
3 (U2R)	2	2	2	0	2
4 (R2L)	2	2	2	2	0

Table 4: Cost matrices for the KDD cup 99 data: (a) anomaly detection and (b) multi-class classification.

It is interesting to see that for this problem, anomaly detection methods based on unsupervised learning out-perform the best classification method using more information. (The cup winner used all of the 5 million labeled data, whereas both of the anomaly detection methods used (subsets of) the normal connection data included in the so-called 10% data.) This is due, in part, to the unique characteristic of this data set that the test data are drawn from a significantly different data distribution than the training data, which is realistic but makes the problem challenging. As is pointed out by earlier publications[15], this in some sense gives a strong argument for employing anomaly detection based methods for realistic detection problems (such as network intrusion and fraud detection).

5 Discussions and Conclusions

We have described a host of learning methods based on sampling that are particularly well suited for applications in resource constrained environments. There are numerous real world problems in which sophisticated data mining functionalities are required with limited computational resources, such as embedding data analytics in mobile computing devices and other special purpose hardware.

At IBM Research we are currently targeting an application of this technology to provide a solution to an important emerging problem of coping with the apparent contradiction between the respective imperatives of security and privacy. On the one hand, in the interest of security and risk management, legislations mandate that data be widely shared across multiple enterprises and analyzed for detection of relevant trends and anomalies in the data. For example, there are legislations requiring banks to analyze customer transaction data for anti-money laundering and other purposes. On the other hand, the privacy protection legislations pro-

hibit, in many cases, the sharing of data for any other purposes.

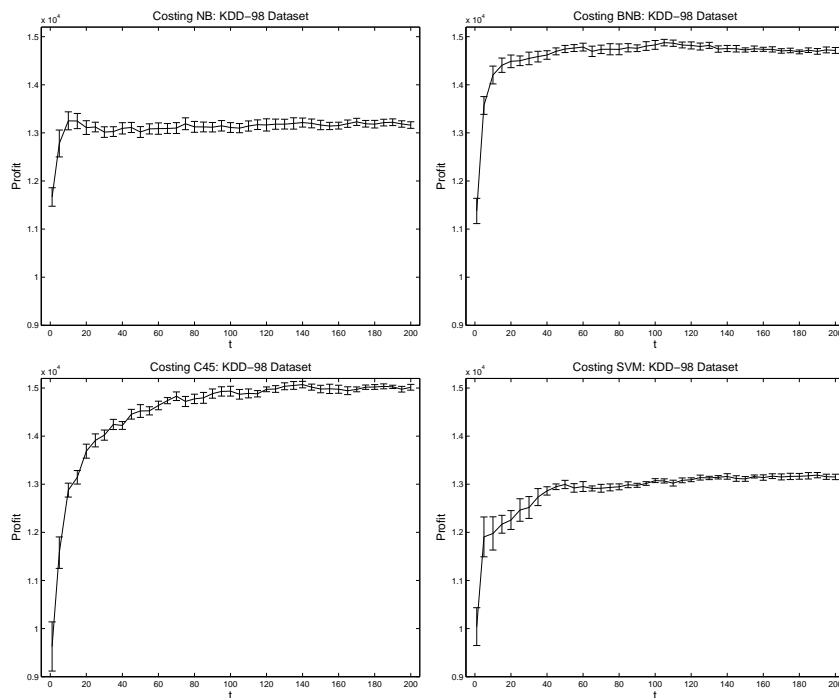
As a solution to this problem, we propose to embed memory light data mining methodologies, such as those described in the present paper, into a cryptographically secure, but resource limited processor. Together with data federation capability made possible by IBM’s light weight database technology, this would allow multiple enterprises, that are generally not permitted to share data, to do so solely for the purpose of detecting particular types of anomalies and for generating alerts. The data will reside within the enterprises themselves and will be processed in a federated set-up with any transmission of the data to the processor being encrypted. Since the data will be decrypted only within the secure confines of the processor, no additional information on the data, other than the alerts themselves, will ever be disclosed to a third party. With this type of strong guarantee, it makes the challenging imperative of meeting both security and privacy mandates a feasible task.

Finally we note that both cost sensitive learning and anomaly detection are extremely important and relevant to the applications in question, such as fraud detection and credit risk assessment. This is the case because in these applications, the cases that need to be detected are rare and much more costly (cost-sensitive scenario), and available training data may or may not contain sufficient labeled data, particularly for the rare and costly cases (anomaly detection scenario). It is our hope that memory light mining methods embedded in a cryptographically secure processor will provide a critical functionality to this important problem in a way that has not been possible with existing technology.

Acknowledgment

We thank Edwin Pednault of IBM Research and Philip Long of Columbia University for helpful discussions.

KDD-98:



DMEF-2:

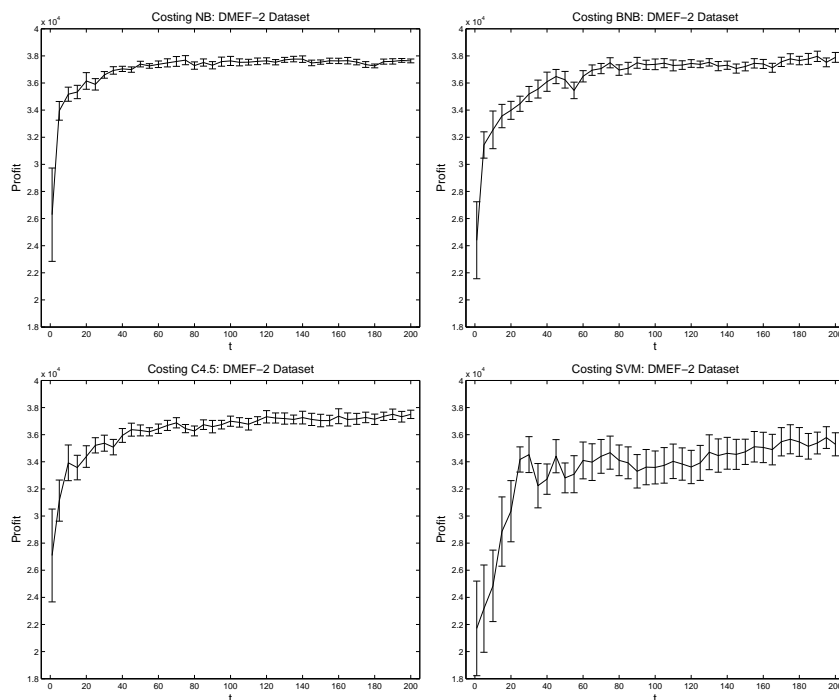
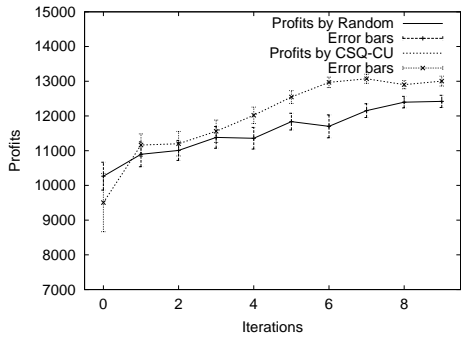
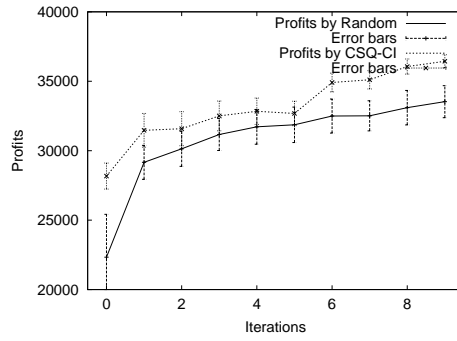


Figure 6: Costing. The graphs shows how the test set profit grows as the number of resampled sets is increased from 1 to 200.

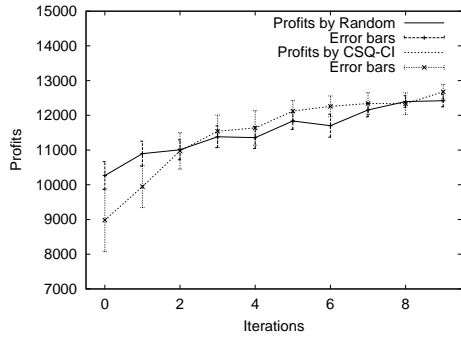


(a) *KDD98*

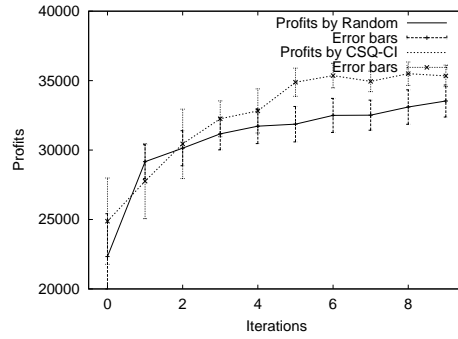


(b) *DMEF2*

Figure 7: Results for CSQ-CU: Profits obtained by CSQ-CU and by Random.



(a) *KDD98*



(b) *DMEF2*

Figure 8: Results for CSQ-CI: Profits obtained by CSQ-CI and by Random.

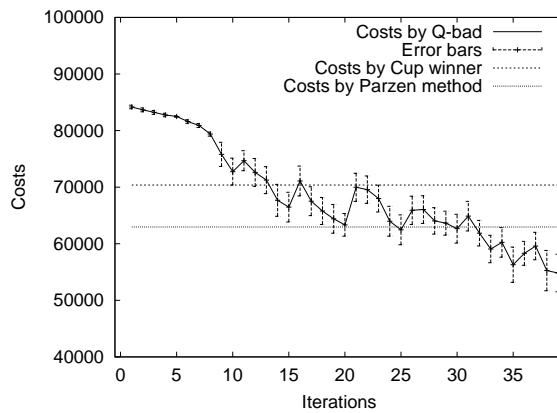


Figure 9: Results for Qbad, Parzen Window method and KDD cup 99 winner on the KDD cup 99 data.

References

- [1] S. Anifantis. The dmef data set library. the direct marketing association, new york, ny., 2002. (<http://www.the-dma.org/dmef/dmefdset.shtml>).
- [2] C. Apte, E. Bibelnieks, R. Natarajan, E. Pednault, F. Tipu, D. Campbell, and B. Nelson. Segmentation-based modeling for advanced targeted marketing. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 408–413. ACM, 2001.
- [3] S. D. Bay. UCI KDD archive. Department of Information and Computer Sciences, University of California, Irvine, 2000. <http://kdd.ics.uci.edu/>.
- [4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Data Mining*, 2000.
- [5] P. Domingos and G. Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- [6] C. Elkan. Results of the kdd'99 classification learning contest. Available at <http://www.cs.ucsd.edu/users/elkan/clresults.html>, 1999.
- [7] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. Tutorial at SIGMOD'02., 2002.
- [8] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 73–84. ACM Press, 1998.
- [9] E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. *IEEE Transactions on Knowledge and Data Engineering*, 12(3), 2000.
- [10] H. Mamitsuka and N. Abe. Efficient mining from large databases by query learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.
- [11] A. Nanopoulos, Y. Manolopoulos, and Y. Theodoridis. An efficient and effective algorithm for density biased sampling. In *Proceedings of the Eleventh International Conference on Information and knowledge management*, pages 398–404, 2002.
- [12] R. Natarajan and E. Pednault. Segmented regression estimators for massive data sets. In *Second SIAM International Conference on Data Mining*, Arlington, Virginia, 2002. to appear.
- [13] F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Knowledge Discovery and Data Mining*, 3:131–169, 1999.
- [14] J. von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards, Applied Mathematics Series*, 12:367–38, 1951.
- [15] D. Y. Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proceedings of the Xisteenth International Conference on Pattern Recognition*, pages 385–388, 2002.
- [16] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 2001.
- [17] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 435–442, 2004.