Efficient Mining from Large Databases by Query Learning

Hiroshi Mamitsuka H-MAMITSUKA@AK.JP.NEC.COM
Naoki Abe N-ABE@AX.JP.NEC.COM
NEC C& C Media Research Laboratories, 4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216-8555 JAPAN

Abstract

We propose a new data mining method that is effective for mining from very large data sets by applying ideas of query learning. In particular, we propose and evaluate a selective sampling method that belongs to the general category of 'uncertainty sampling,' by adopting and extending the 'query by bagging' method, proposed earlier by the authors as a query learning method. We empirically evaluate the effectiveness of the proposed method by comparing its performance against Breiman's Ivotes, a representative sampling method for scaling up inductive algorithms. Our results show that the performance of the proposed method compares favorably against that of Ivotes, both in terms of the predictive accuracy achieved using a fixed amount of computation time, and the final accuracy achieved. This is found to be especially the case when the data size approaches a million, a typical data size encountered in real world data mining applications. We have also examined the effect of noise in the data and found that the advantage of the proposed method becomes more pronounced for larger noise levels. This is further confirmed by our experiments on an actual data set in database marketing.

1. Introduction

With increasing successes of business applications of data mining, machine learning techniques are attracting attention as foundations of data mining technology. One important aspect of data mining which distinguishes it from machine learning, however, is the abundance of data it typically involves. In most real world data mining applications, data sizes reach millions, if not more. A central issue in data mining, therefore, is how to computationally efficiently find effective rules from a very large database. This represents a significant change in mind-set from the usual machine learning practices, in which the goal is to efficiently find a good predictive rule, from as few data as possible.

Provost and Kolluri (1999) give a comprehensive survey of methods for 'scaling up' inductive algorithms, for the purpose of mining from large data sets. Of the approaches surveyed in this article, we are concerned with that of 'data partitioning', and 'sequential multi-subset learning with a model-guided instance selection,' in particular. In this approach, relatively small subsets of data are sequentially sampled, using a model guided instance selection strategy, and the successive models are combined to give the final resulting model. A number of methods that have been proposed to date in the literature belong to this category, including Windowing (Quinlan, 1983), Integrative Windowing (Furnkranz, 1998), boosting (Freund & Schapire, 1997), and Ivotes (Breiman, 1999). One thing that is common among all of these methods is that they employ a sampling method that makes use of the label information in the candidate instances. For example, Ivotes uses a sampling method called 'importance sampling', which chooses examples on which the current hypothesis makes a mistake and (with high probability) discards those on which the current hypothesis predicts correctly. It has been reported, however, that some of these methods do not work well, in the presence of abundant noise in the training data (Catlett, 1991; Provost & Kolluri, 1999).

As a remedy for this problem, we propose to use what is generically known as 'uncertainty sampling', which samples those examples that cannot be reliably predicted at that point. Note that, in uncertainty sampling, the label information in the candidate instances is *not* used in making selections, and thus such a method can be interpreted as a 'query learning method' that queries for the labels of selected instances. The purpose of the present paper is to examine how well this approach works in the current context of efficient mining from large data sets, and

to characterize under what conditions it works better than importance sampling, in particular. The particular sampling method we employ is based on the idea of 'query by bagging' proposed by Abe and Mamitsuka (1998), which was in turn obtained by combining ideas of query by committee (Seung et al., 1992) and 'bagging' (Breiman, 1996). The basic idea is that query points are chosen by picking points on which the predictions made by the hypotheses resulting from applying the component inductive algorithm to sub-samples obtained via re-sampling from the original data set, are most evenly spread. This method is like Breiman's Ivotes, except committee-based uncertainty sampling is used in place of the importance sampling employed in Ivotes.

We empirically evaluated the performance of this method, using a number of different types of data sets. In our first experiments, we used synthetic data sets of size one million each, generated from the 'generator' functions of (Agrawal et al., 1993), used often as benchmark data for evaluating data mining methods. We found that the performance of QbagS was favorable as compared to that of Ivotes, both in terms of the computation time required to reach the same predictive accuracy, and the final accuracy attained.

In order to better understand the conditions under which QbagS performs well, we varied a parameter called the 'perturbation factor,' which controls the noise level of the 'generator' functions. It was found that for larger perturbation factors, the significance level by which QbagS out-performed Ivotes became larger. This result confirms the thesis that uncertainty sampling is more desirable than sampling methods that concentrate on those instances on which prediction errors are made, when the data is noisy. This thesis is further supported by the results of our experimentation on a real world data set, which is inevitably noisy. Specifically, we compared the two methods using a data set in the area of database marketing (internet provider churn data) of size roughly a million. Here we found that the predictive accuracy of QbagS was significantly better than Ivotes.

We also evaluated the performance of these methods on medium sized data sets, using data sets from the Statlog project (Michie et al., 1994), whose sizes are in the range of tens of thousands. On these data sets, we also compared the performance of the original Qbag, in addition to QbagS and Ivotes. It was found that Qbag did the best overall for the medium sized data sets, although the differences in performance were not greatly significant.

2. The Mining/Learning Methods

2.1 Proposed Method

In this section, we describe the mining/learning method we propose and evaluate in this paper, which we call QbagS, standing for 'Query by bagging with a single loop.' This procedure provides a sampling strategy that uses an arbitrary component learning algorithm as a subroutine, and works roughly as follows. (See the pseudocode shown below.) At each iteration, it randomly samples a relatively large number of candidate examples (R, say 10,000) from the database (line 1). It then selects a small enough (D, say 1,000) subset of this set and applies the component learning algorithm to it to obtain a new hypothesis (line 3). When making this selection, it uses the hypotheses from the past iterations to predict the labels of the candidate examples, and then pick those on which the predicted values are split most evenly. More precisely, it calculates the 'margin' of each candidate instance, that is, the difference between the number of votes by the past hypotheses for the most 'popular' label, and that for the second most popular label (line 2). Then, D instances having the least values of margin are selected from the candidates (line 3). The final hypothesis is defined by the majority vote over all the hypotheses obtained in the above process.

Algorithm: Query-by-Bagging:Single (QbagS)
Input: Number of iterations: M
Component learning algorithm: A
Number of candidates at each iteration: R
Number of selected examples at each iteration: D
Candidates at the i-th iteration: C_i
Selected (training) examples at the i-th iteration: S_i
Initialization: 1. Randomly sample initial sample $S_1 = \langle (x_1, y_1), \cdots, (x_D, y_D) \rangle$ from the database.
2. Run A on S_1 and obtain hypothesis h_1.
For $i = 1, ..., M$
1. Randomly sample R examples C_i from database.
2. For all $x \in C_i$, calculate 'margin' $m(x)$ using past hypotheses $h_1, \cdots, h_i$
$m(x) = \max_y |\{t \leq i : h_t(x) = y\}|$
$\quad - \max_{y \neq y_{\max}(x)} |\{t \leq i : h_t(x) = y\}|$
where $y_{\max}(x) = \arg\max_y |\{t \leq i : h_t(x) = y\}|$
3. Select D examples $\langle (x_1^*, y_1^*), \cdots, (x_D^*, y_D^*) \rangle$ from C_i having the smallest $m(x)$ $(x \in C_i)$ and let $S_{i+1} = \langle (x_1^*, y_1^*), \cdots, (x_D^*, y_D^*) \rangle$.
4. Run A on S_{i+1} and obtain hypothesis h_{i+1}.
End For
Output: Output final hypothesis given by:
$h_{fin}(x) = \arg\max_{y \in Y} |\{t \leq M : h_t(x) = y\}|$

Notice that, in QbagS, re-sampling is done directly from the database, and past hypotheses are used to judge what examples to sample next. In fact, this is significantly simplified as compared to the original procedure of query by bagging (Qbag) of Abe and Mamitsuka (1998): As can be seen in the pseudocode presented below,[1] in the original procedure the selected examples are accumulated to form the training data set (line 5). Then at each iteration, re-sampling is done from this set of training data (line 1), and the resulting hypotheses are used to judge what examples to select next, using a committee-based uncertainty sampling (lines 2 to 5). Since re-sampling is done at each iteration, using embedded looping, this procedure is computationally more demanding than the new version QbagS, but may be more data efficient. This is consistent with the fact that the original Qbag was designed as a *data efficient* query learning method, whereas QbagS is meant primarily as a *computationally efficient* method for mining from large databases.

Algorithm: Query-by-Bagging (Qbag)
Input: Number of stages: M
Component learning algorithm: A
Number of re-sampling at each iteration: T
Number of candidates at each iteration: R
Number of selected examples at each iteration: D
Candidates at the i-th iteration: C_i
Selected (training) examples at the i-th iteration: S_i
Initialization:
Randomly sample initial sample
$S_1 = \langle (x_1, y_1), \cdots, (x_D, y_D) \rangle$ from the database.
For $i = 1, ..., M$
1. By re-sampling from S_i with uniform distribution, obtain sub-samples $S'_1, .., S'_T$ (of same size as S_i).
2. Run A on $S'_1, .., S'_T$ to obtain hypotheses $h_1, ..., h_T$.
3. Randomly select R examples C_i from database.
4. For all $x \in C_i$, compute the margin $m(x)$ by:
$m(x) = \max_y |\{t \leq T : h_t(x) = y\}|$
$\quad - \max_{y \neq y_{\max}(x)} |\{t \leq T : h_t(x) = y\}|$
where $y_{\max}(x) = \arg\max_y |\{t \leq T : h_t(x) = y\}|$
5. Select D examples $\langle (x^*_1, y^*_1), \cdots, (x^*_D, y^*_D) \rangle$ from C_i having the smallest values of $m(x)$ ($x \in C_i$)
and update the training data as follows.
$S_{i+1} = append(S_i, \langle (x^*_1, y^*_1), \cdots, (x^*_D, y^*_D) \rangle)$
End For
Output: Output final hypothesis given as follows.
$h_{fin}(x) = \arg\max_{y \in Y} |\{t \leq T : h_t(x) = y\}|$
where h_t ($t = 1, \cdots, T$) are the hypotheses of the final (M-th) stage.

[1] There is one modification from the way Qbag was presented in (Abe & Mamitsuka, 1998): Now it is presented for multi-valued prediction, while the original version was assuming a binary-valued prediction.

2.2 Breiman's Ivotes

We briefly review 'Ivotes' (importance sampling) (Breiman, 1999), with which we compare the performance of our method. Like QbagS, Ivotes takes a sample from the database and applies the component learning algorithm at each iteration, discards the data and keeps just the hypotheses. When sampling, it uses what is called 'importance sampling.' In this sampling method, if the label of an example is wrongly predicted by its current combined hypothesis (out-of-bag prediction), it is automatically chosen. If the prediction on an example is correct, then it is selected with probability $e/(1-e)$, where e is the error probability (measured using a separately reserved test set) of the current hypothesis. Here, the out-of-bag prediction is done by majority vote over those hypotheses trained on subsamples *not* containing the current example. Breiman (1999) claims that this feature contributes greatly to improving the performance of Ivotes, and proposes a particular implementation method for computing out-of-bag predictions: It keeps records of how many times each label of each example in the database has been predicted. We follow this implementation exactly.

3. Empirical Evaluation

We empirically evaluated the performance of the proposed method and that of Ivotes, using a variety of data sets. We used the following three types of data sets.

1. A series of large scale synthetic data sets, generically referred to as Generator in (Agrawal et al., 1993), often used as benchmark data for evaluating data mining methods.
2. A real world data set in the area of database marketing of size roughly a million. In particular, we used data for 'churn' analysis for an internet provider.
3. A number of medium sized (tens of thousands) data sets known as statlog data. They were used in (Breiman, 1999) to evaluate the performance of Ivotes.

In our experiments, we used both C4.5 (Quinlan, 1993) and CART[2] as the component algorithm. Our evaluation was mainly done in terms of the total computation time to achieve a given prediction accuracy (on separate test data), including disk access time. We also compare the 'final accuracy' attained by each of

[2] To be precise, we used a version of CART included in the IND package due to Wray Buntine.

Table 1. Large-scale data summary

Data set	#classes	# disc. atts	# cont. atts	#training samples	#test samples
Generator	2	3	6	800000	200000
Churn	2	29	62	617551 - 617600	154351 - 154400

the methods. By 'final accuracy,' we mean the accuracy level reached for data sizes large enough that the predictive performance appears to be saturating.[3] For the evaluation on the real world data set, we also used the measures of precision and recall, standard performance measures in the field of information retrieval. Note that 'recall' is defined as the probability of correct prediction given that the actual label is 1 (or whatever label of interest), and 'precision' is defined as the probability of correct prediction given that the predicted label is 1.

In most of the experiments (except those using the 'statlog' data), the evaluation was done by five-fold cross validation. That is, we split the data set into five blocks of roughly equal size, and at each trial four out of these five blocks were used as training data, and the last block was reserved as test data. The results (prediction accuracy, learning curves, precision and recall) were then averaged over the five runs. Since the statlog data come with pre-specified test data, the average was taken over five randomized runs. All of our experiments were run on an Alpha server 4100 (466MHz, 512 MB).

3.1 Evaluation on Large Scale Data Sets

It is said in general (Provost & Kolluri, 1999) that large scale data sets are those with sizes in the order of a million. It is, however, difficult to find publically available real world data sets having data sizes in the order of a million.[4] We thus used a series of synthetic data introduced in Agrawal et al. (1993) called Generator. These data sets have been used often as benchmark data in evaluating the scalability issues of data mining methods (Gehrke et al., 1999; Rastogi & Shim, 1998). The properties of these data sets are summarized in Table 1.

[3] To be sure, we performed the 'mean difference significance test' (See Section 3.1 for the definition) for the predictive accuracy attained at the end of each run and that reached 1,000 seconds prior to that point. We found that the significance level was typically around 0.2 and at most 0.5, and thus the difference was insignificant.

[4] The largest data sets we found were those in statlog. The largest ones in UCI ML Repository we could find (mushroom and thyroid0387), for example, contain less than 10,000 data.

Generator contains ten distinct functions for generating the labels, taking a number of different forms. Here we chose five out of the ten (Function 2, 5, 8, 9 and 10) on which the predictive performance of ID3, as reported by Agrawal et al. (1993) was relatively poor. Generator also has a parameter called 'perturbation factor,' which is used to randomly perturb the continuous valued attributes. In our first set of experiments, we chose to set the perturbation factor to be either 0.2 or 0.6. For each of the five functions, we generated a data set of size one million, and performed five-fold cross-validation.[5]

Table 2. Summary of parameter settings in our experiments

Methods	# candidate samples per iteration(R)	# selected samples per iteration(D)
QbagS	30000	3000
Ivotes	-	1000

We note that there is a parameter to be tuned in Ivotes, namely the number of examples to be selected at each iteration. As a test, we observed the predictive performance of Ivotes on one of the five data sets (Function 2), varying this parameter. It was observed that the value of 1,000 was doing about the best, so we set this parameter to be 1,000 in all of our experiments. This is consistent with the observation in the experiments reported in Breiman (1999) that the performance of Ivotes improves up to 800 but appears to be almost saturating. Parameters in the other methods that were used in our experiments are summarized in Table 2. We remark that we did not make extensive effort in optimizing these parameters.

We show part of the results of the above experimentation (with perturbation factor 0.2 and CART) in the form of learning curves in Figure 1. Note that, in these curves, the average predictive accuracy (on test data) is plotted against the total computation time. We also

[5] For Generator and Churn, to be described later, we randomly picked 10,000 out of the test data to evaluate the predictive performance.

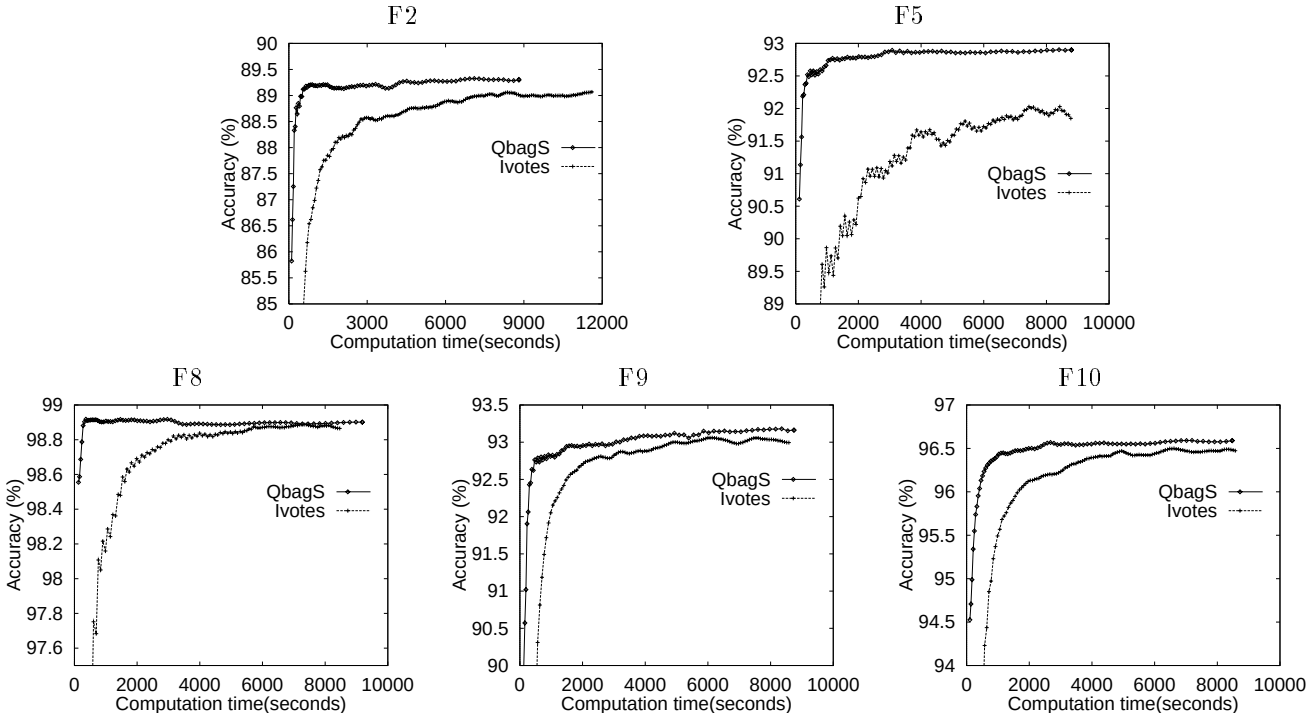

Figure 1. Prediction accuracy of QbagS and Ivotes on Generator, averaged over five runs, plotted as function of computation time.

give, in Table 3, figures that indicate the computation time in seconds (and their ratios as compared to that of QbagS) to reach the level of predictive performance achieved by the method with the lower final accuracy. From these results, it is seen that, in terms of the speed for reaching the same level of accuracy, QbagS is favored over Ivotes in all cases (when using CART). The speed-up factor achieved by these methods over Ivotes, as measured by the amount of computation time they take to reach a given performance level, is anywhere from 2 to 30.

In terms of the 'final prediction accuracy', the results were also favorable for QbagS. These results are also summarized in Table 3, in which the final accuracies reached by the two methods for the five functions and the 'Z' values of the mean difference significance test for the respective cases are exhibited. Here, the Z values are calculated using the following well-known formula (Weiss & Indurkhya, 1998):

$$Z = \frac{acc(A) - acc(B)}{\sqrt{\frac{var(A)}{n_A} - \frac{var(B)}{n_B}}}$$

where we let, in general, $acc(A)$ denote the accuracy estimate for method A, and $var(A)$ the variance of this estimate, and n_A the data size used for this estimate (5 in our case). For example, if Z is greater than 2 then it is more than 95 per cent significant that A achieves higher accuracy than B. For perturbation factor 0.2 and using C4.5 as the component algorithm, QbagS did significantly better than Ivotes in four out of the five cases, and Ivotes did significantly better in the other case. For perturbation factor 0.2 and using CART, QbagS did significantly better than Ivotes in one out of the five cases, slightly (insignificantly) better in three cases, and Ivotes did slightly better in the other case. When perturbation factor is set to 0.6 and when using CART, QbagS did better than Ivotes in all five cases, four out of them being statistically significant. We can see, in these results, that for higher values of perturbation factor, the significance of the difference in the performance of the two methods becomes more pronounced. This is visualized in the two graphs shown in Figure 2; one plots how the Z values of the mean difference significance test vary as the perturbation factor is changed from 0.2 to 0.6 for the five functions, and the other plots how the computation time ratios change.

3.2 Evaluation on a Real World Data Set

To further verify the empirical findings reported in the previous section, we have evaluated these methods using an actual data set in the area of database market-

Table 3. Generator: Final accuracies (%) and computation time to reach target accuracy (and ratios w.r.t. Ivotes), averaged over five runs.

Perturb. factor	Component algorithm	Func.	QbagS		Ivotes		Z
			Final acc.(%)	Comp. time	Final acc.(%)	Comp. time	
0.2	C4.5	F2	**88.46**	**423.1(1.0)**	87.74	3363.5(14.61)	**5.41**
		F5	**92.64**	**230.2(1.0)**	91.23	3687.7(16.01)	**9.37**
		F8	**99.05**	**99.65(1.0)**	98.28	179.8(1.80)	**12.80**
		F9	92.62	5434.8(1.0)	**93.06**	**1656.3(0.30)**	3.12
		F10	**96.72**	**443.1(1.0)**	96.23	5302.5(11.97)	**6.18**
	CART	F2	**89.30**	**498.2(1.0)**	89.06	7255.0(14.56)	1.87
		F5	**92.90**	**186.2(1.0)**	91.91	5405.6(29.03)	**6.20**
		F8	98.90	**320.6(1.0)**	**98.92**	8690.2(27.11)	0.31
		F9	**93.17**	**2514.8(1.0)**	93.00	4747.8(1.89)	0.82
		F10	**96.58**	**1631.7(1.0)**	96.48	5087.8(3.12)	1.37
0.6	CART	F2	**73.09**	**311.2(1.0)**	71.43	7865.8(25.28)	**7.03**
		F5	**85.52**	**258.4(1.0)**	84.33	2304.5(8.92)	**7.68**
		F8	**98.14**	**116.7(1.0)**	97.65	7041.1(60.32)	**3.51**
		F9	**82.93**	**1524.7(1.0)**	82.74	4732.8(3.10)	0.82
		F10	**90.50**	**341.5(1.0)**	89.93	5692.1(16.67)	**2.34**

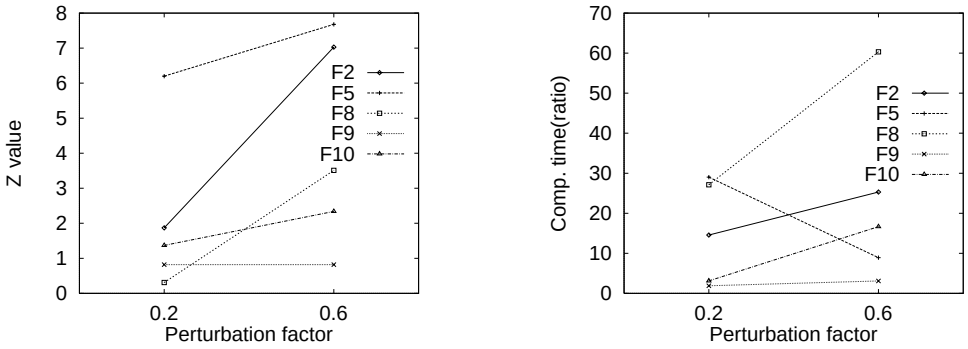

Figure 2. Significance test values Z (Left) and computation time ratios (Right) between QbagS and Ivotes on Generator, plotted as functions of perturbation factor.

ing. More concretely, we have used data in a database containing customer data of a certain internet provider and used them to perform so-called 'churn' analysis, that is, predicting those customers that are likely to quit in the near future. The data we used consisted of three parts: (i) data containing demographic data such as the customers' age, sex, address, etc.; (ii) monthly access data, of the internet and various contents, from the past six monthes; and (iii) the label information of whether the customers quit in the month immediately after the ending month of the monthly data of (ii). The temporal data of (ii) were converted to a number of continuous valued attributes. Overall, there were close to a hundred attributes, a mixture of continuous and discrete valued attributes. There were roughly 800 thousand records in the data that we used. These properties of this data set, which we call Churn are summarized in Table 1. Again, our evaluation was done by five-fold cross-validation.

As before, we plot in Figure 3 the learning curves of the competing methods, plotted against the total computation time. Also, the final predictive accuracy of each method and the speed-up factor measured in terms of the total computation time required to reach a given target accuracy are summarized in Table 4. It is seen clearly in these results that the proposed methods achieve statistically significant improvement in terms of both the final prediction accuracy achieved, and the computation time required to achieve a given accuracy. This tendency is seen to be more pronounced than what we observed for Generator, which is consistent with the fact that the noise level is inevitably high in a real world data set.

Precision-recall curves for QbagS and Ivotes are also shown in Figure 3. The graph shows the precision-recall curves attained after approximately 7,000 seconds of computation time for both methods when us-

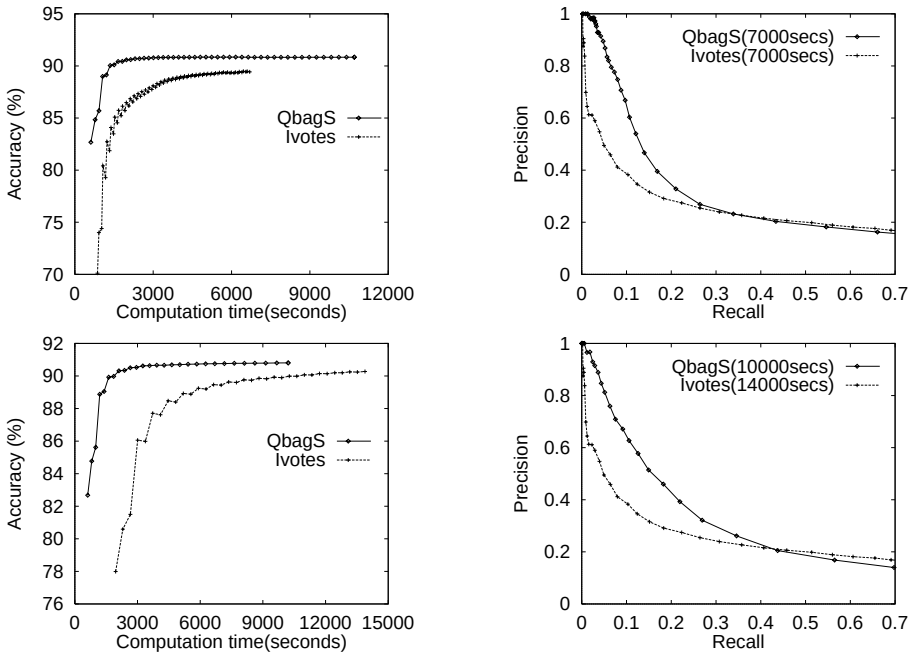

Figure 3. Churn: Learning curves (Left) and precision-recall curves (Right) using C4.5 (Top) and CART (Bottom).

Table 4. Churn: Final accuracies (%) and total computation time in seconds to reach target accuracy (and ratios) averaged over five runs.

Component algorithm	QbagS		Ivotes		Z
	Final accuracy(%)	Computation time	Final accuracy(%)	Computation time	
C4.5	**90.40**	**1207.3(1.0)**	89.43	4946.3(4.10)	**4.87**
CART	**90.80**	**1857.7(1.0)**	90.28	12440.7(6.70)	**2.83**

ing C4.5, and after about 10,000 and 14,000 seconds for QbagS and Ivotes respectively, when using CART. One typical application of churn analysis is to take a certain action to those members that are predicted to be likely to quit soon. Although it all depends on the cost-performance landscape of individual situations, normally a relatively small subset is chosen, for which a reasonably high precision can be achieved. In the current problem, a reasonable point is, say recall 0.1. At this value of recall, it is found that the precision of QbagS is approximately 25 per cent better than that of Ivotes when using C4.5, and 20 per cent better with CART. This represents a sizable difference in business terms.

3.3 Evaluation on Medium Sized Data Sets

We now report on our evaluation using medium sized data sizes. For these experiments, we use the data used in the Statlog project, and described in Michie et al. (1994). We used four data sets from the Statlog data: dna, sat, letter and shuttle, each sized 2000, 4435, 15,000 and 43,500. Note that these four data sets were all used in the empirical evaluation done by Breiman (1999).

We give the results of experimentation on these data sets in Table 5. The figures give the (final) prediction accuracy obtained by each method, this time including the original Qbag and the component algorithm, all averaged over five randomized runs. It is seen that, for data sets of sizes in the range of tens of thousands, QbagS still appears to be favored over Ivotes, but not to the same statistical significance as for the large data sets. For these data sets, in fact, the original Qbag which keeps data from the past iterations, appears to do better than both Ivotes and QbagS. Over all, these results confirm that QbagS is especially suited for large, noisy data sets.

4. Concluding Remarks

We have proposed a sampling method for data mining that is especially effective for efficient mining from

Table 5. Statlog: Average Final Accuracies (%)

Data	Comp. Algo.	QbagS	Qbag	Ivotes
dna	C4.5	**95.20**	94.16	94.03
	CART	**95.30**	95.23	94.27
sat	C4.5	90.28	**90.72**	90.24
	CART	88.93	**91.23**	88.95
letter	C4.5	94.59	**96.11**	89.36
	CART	89.22	**95.40**	88.50
shuttle	C4.5	99.87	**99.99**	99.97
	CART	99.68	99.95	**99.95**

large, noisy data sets. The key property of our method which contributes to this advantage is its sampling strategy that does *not* make use of the label information in the candidate instances. Interesting future work is to characterize more systematically the conditions (noise level and data size) under which the proposed method works well, and better than importance sampling in particular. It would also be interesting to investigate the relationship and possibly combinations between uncertainty sampling and importance sampling and/or boosting, in the context of efficient mining from large data sets.

Acknowledgements

The authors would like to thank Osamu Watanabe and Carlos Domingo for invaluable discussions on the topics of this paper. The authors would also like to thank Dr. S. Doi, Dr. N. Koike and Dr. S. Goto of NEC Corporation for making this research possible. Naoki Abe was supported, in part, by the Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science) 1999 from the Ministry of Education, Science, Sports and Culture, Japan.

References

Abe, N., & Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. *Proceedings of Fifteenth International Conference on Machine Learning* (pp. 1–9). Morgan Kaufmann.

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, *5*, 914–925.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning*, *36*, 85–103.

Catlett, J. (1991). Megainduction: A test flight. *Proceedings of Eighth International Workshop on Machine Learning* (pp. 596–599).

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Furnkranz, J. (1998). Integrative windowing. *Journal of Artificial Intelligence Research*, *8*, 129–164.

Gehrke, J., Ganti, V., Ramakrishnan, R., & Loh, W.-Y. (1999). BOAT – Optimistic decision tree construction. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 169–180). ACM Press.

Michie, D., Spiegelhalter, D., & Taylor, C. (Eds.). (1994). *Machine learning, neural and statistical classification.* London: Ellis Horwood.

Provost, F., & Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Knowledge Discovery and Data Mining*, *3*, 131–169.

Quinlan, J. R. (1983). Learning efficient classification procedures and their applications to chess endgames. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Francisco: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* San Francisco: Morgan Kaufmann.

Rastogi, R., & Shim, K. (1998). Public: A decision tree classifier that integrates building and pruning. *Proceedings of 24th International Conference on Very Large Data Bases* (pp. 404–415). New York: Morgan Kaufmann.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of 5th Annual Workshop on Computational Learning Theory* (pp. 287–294). New York: ACM Press.

Weiss, S. M., & Indurkhya, N. (1998). *Predictive data mining.* San Francisco: Morgan Kaufmann.