Static Versus Dynamic Sampling for Data Mining

George H. John and **Pat Langley**
Computer Science Department
Stanford University
Stanford, CA 94305-9010
{gjohn,langley}@CS.Stanford.EDU
http://robotics.stanford.edu/~{gjohn,langley}/

Abstract

As data warehouses grow to the point where one hundred gigabytes is considered small, the computational efficiency of data-mining algorithms on large databases becomes increasingly important. Using a *sample* from the database can speed up the data-mining process, but this is only acceptable if it does not reduce the quality of the mined knowledge. To this end, we introduce the "Probably Close Enough" criterion to describe the desired properties of a sample. Sampling usually refers to the use of *static* statistical tests to decide whether a sample is sufficiently similar to the large database, in the absence of any knowledge of the tools the data miner intends to use. We discuss *dynamic* sampling methods, which take into account the mining tool being used and can thus give better samples. We describe dynamic schemes that observe a mining tool's performance on training samples of increasing size and use these results to determine when a sample is sufficiently large. We evaluate these sampling methods on data from the UCI repository and conclude that dynamic sampling is preferable.

Introduction

The current popularity of data mining, data warehousing, and decision support, as well as the tremendous decline in the cost of disk storage, has led to the proliferation of terabyte data warehouses. Mining a database of even a few gigabytes is an arduous task, and requires either advanced parallel hardware and parallelized data-mining algorithms, or the use of sampling to reduce the size of the database to be mined.

Data analysis and mining always take place within the context of solving some problem for a customer (domain expert or data owner), and thus many decisions must be made jointly by the customer and the data miner. The analyst's job is to present the sampling decision in a comprehensible form to the customer.

The PCE (Probably Close Enough) criterion we introduce is a criterion for evaluating sampling methods. If we believe that the performance of our data-mining algorithm on a sample is probably close to what it would be if we ran it on the entire database, then we should be satisfied with the sample. The question is how to quantify *close enough* and *probably*. For simplicity, we consider only accuracy in *supervised* classification methods, although the PCE framework is more general. This focus lets us leverage existing learning theory and make the discussion more concrete by defining *close* in terms of accuracy.

Dynamic sampling refers to the use of knowledge about the behavior of the mining algorithm in order to choose a sample size — its test of whether a sample is suitably representative of a database depends on how the sample will be used. In contrast, a *static* sampling method is ignorant of how a sample will be used, and instead applies some fixed criterion to the sample to determine if it is suitably representative of the original large database. Often, statistical hypothesis tests are used. We compare the static and dynamic approaches along quantitative (sample sizes and accuracy) and qualitative (customer-interface issues) dimensions, and conclude with reasons for preferring dynamic sampling.

Static Sampling Criteria

The aim of static sampling is to determine whether a sample is sufficiently similar to its parent database. The criteria are static in the sense that they are used independently of the following analysis to be performed on the sample. Practitioners of KDD sometimes speak of "statistically valid samples" and this section represents one attempt to make this precise. Our approach tests the hypotheses that each field in the sample comes from the same distribution as the original database.

For categorical fields, a χ^2 hypothesis test is used to test the hypothesis that the sample and the large database come from the same distribution. For numeric fields, a large-sample test (relying on the central limit theorem) is used to test the hypothesis that the sample and the large database have the same mean.

a)

		B		
		T	F	
A	T	50	50	100
	F	50	50	100
		100	100	

b)

		B		
		T	F	
A	T	10	0	10
	F	0	10	10
		10	10	

Table 1: a: Counts of a database containing 50 copies of records <TT>, <TF>, <FT>, <FF>. b: Sample containing 10 copies of records <TT>, <FF>. These both look *the same* to univariate static sampling.

Note that hypothesis tests are usually designed to minimize the probability of falsely claiming that two distributions are different (Casella & Berger 1990). For example, in a 95% level hypothesis test, assuming the two samples do come from the same distribution, there is a 5% chance that the test will incorrectly reject (type I error) the null hypothesis that the distributions are the same. However, for sampling we want to minimize the probability of falsely claiming they are the same (type II error). We used level 5% tests, which liberally reject the null hypothesis, and are thus conservative in claiming that the two distributions are the same. (Directly controlling Type II error is more desirable but complicated and requires extra assumptions.)

Given a sample, static sampling runs the appropriate hypothesis test on each of its fields. If it accepts all of the null hypotheses then it claims that the sample does indeed come from the same distribution as the original database, and it reports the current sample size as sufficient.

There are several shortcomings to the static sampling model. One minor problem is that, when running several hypothesis tests, the probability that at least one hypothesis is wrongly accepted increases with the number of tests. The Bonferroni correction can adjust for this problem. More important, the question "is this sample good enough?" can only be sensibly answered by first asking "what are we going to do with the sample?" Static sampling ignores the data-mining tool that will be used. The tests we describe are only univariate, which is problematic (see Table 1). One could as well run bivariate tests but then there is of course no guarantee that the three-way statistics will be correct. It is also unclear how the setting of the confidence levels will effect sample size and performance, so this is a poor framework to present to a customer.

Dynamic Sampling

Sampling a database is a scary prospect. It involves a decision about a tradeoff that many customers are rightfully hesitant to make. That decision is how much they are willing to give up in accuracy to obtain a decrease in running time of a data mining algorithm. Dynamic sampling and the PCE criterion address this decision directly, rather than indirectly looking at statistical properties of samples independent of how they will be used. Ultimately, the costs of building the model (disk space, cpu time, consultants' fees) must be amortized over the period of use of the model and balanced against the savings that result from its accuracy. This work is a step in that direction.

The PCE Criterion

The Probably Close Enough criterion is a way of evaluating a sampling strategy. The key is that the sampling decision should occur in the context of the data mining algorithm we plan to use. The PCE idea is to think about taking a sample that is probably good enough, meaning that there is only a small chance that the mining algorithm could do better by using the entire database instead. We would like the smallest sample size n such that

$$Pr(\mathtt{acc}(N) - \mathtt{acc}(n) > \epsilon) \leq \delta \ ,$$

where $\mathtt{acc}(n)$ refers to the accuracy of our mining algorithm after seeing a sample of size n, $\mathtt{acc}(N)$ refers to the accuracy after seeing all records in the database, ϵ is a parameter to be specified by a customer describing what "close enough" means, and δ is a parameter describing what "probably" means. PCE is similar to the the Probably Approximately Correct bound in computational learning theory.

Given the above framework, there are several different ways to attempt to design a dynamic sampling strategy to satisfy the criterion. Below we describe methods that rely on general properties of learning algorithms to estimate $\mathtt{acc}(N) - \mathtt{acc}(n)$. But first, in order to test the PCE framework, we must select a learning algorithm.

We chose the naive Bayesian classifier, which has a number of advantages over more sophisticated techniques for data mining, such as methods for decision-tree and rule induction. The algorithm runs in time linear with the number of attributes and training cases, which compares well with the $O(n \log n)$ time for basic decision-tree algorithms and at least $O(n^2)$ for methods that use post-pruning. Also, experimental studies suggest that naive Bayes tends to learn more rapidly, in terms of the number of training cases needed to achieve high accuracy, than most induction algorithms (Langley & Sage 1994). Theoretical analyses (Langley, Iba & Thompson 1992) point to similar conclusions about the naive Bayesian classifier's rate of learning. A third feature is that naive Bayes can be implemented in an incremental manner that is not subject to order effects.

Sampling through Cross Validation

Despite the inherent efficiency of naive Bayes, we would like to reduce its computational complexity even further by incorporating dynamic sampling. There are several problems to solve in deciding whether a sample meets the PCE criterion, but each of them is well-defined in terms of our goal. We examine samples of increasing larger size n, adding a constant number of records to our sample repeatedly until we believe the PCE condition is satisfied.

First, we must estimate $\mathtt{acc}(n_i)$. In our algorithm, we used leave-one-out cross-validation on the sample as our estimate of $\mathtt{acc}(n_i)$. Then we must estimate $\mathtt{acc}(N)$. For a first attempt, we assume that whenever $\mathtt{acc}(n_{i+1}) \leq \mathtt{acc}(n_i)$, the derivative of accuracy with respect to training set size has become non-positive and will remain so for increasing sample sizes. Thus $\mathtt{acc}(N) \leq \mathtt{acc}(n_i)$ and we should accept the sample of size n.

In initial experiments on UCI databases we found that this method for putting a bound on $\mathtt{acc}(N)$ is sensitive to variance in our estimates for $\mathtt{acc}(n_i)$, and often stops too soon. On average, accuracy was reduced about 2% from the accuracy on the full database, while the sample size was always less than 20% of the size of the original database.

Extrapolation of Learning Curves

Perhaps this sensitivity could be overcome by the use of more information to determine $\mathtt{acc}(N)$ rather than just the last two estimated accuracies. One method is to use all available data on the performance of the mining algorithm on varying-sized training sets, and use these to fit a parametric learning curve, an estimate of the algorithm's accuracy as a function of the size of the training sample. Extrapolation of Learning Curves (ELC) can predict the accuracy of the mining algorithm on the full database.

But first, we must estimate $\mathtt{acc}(n)$. In our algorithm, when considering a sample of size n we take K more records from the large database and classify them and measure the resulting accuracy. This is our estimate of $\mathtt{acc}(n)$. Then we must estimate $\mathtt{acc}(N)$. We use the history of sample sizes (for earlier, smaller samples) and measured accuracies to estimate and extrapolate the learning curve.Theoretical work in learning, both computational and psychological, has shown that the power law provides a good fit to learning curve data:

$$\widehat{\mathtt{acc}}(n) = a - bn^{-\alpha} \ .$$

The parameters a, b, α are fit to the observed accuracies using a simple function optimization method. We used Dynamic Hill Climbing (Yuret 1994), which seems to work well.

We know N, the total size of the database. Given $\widehat{\mathtt{acc}}(N)$ as our estimate for the accuracy of our data mining algorithm after seeing all N cases in the database, we can check the difference between this expected value and the current accuracy on our sample of size n, and if the difference is not greater than ϵ, we accept the sample as being representative.

If the difference is greater than ϵ, we reject the sample and add the additional K records (sampled previously, to get an estimate of the accuracy of our model) to our sample, updating the model built by our mining algorithm. For this to be efficient, the mining algorithm must be *incremental*, able to update itself given new data in time proportional to the amount of new data, not the total amount of data it has seen.

Experiments: ELC vs Static Sampling

Preliminary studies with ELC sampling for naive Bayes gave good results relative to the non-sampling version of this algorithm, which encouraged us to carry out a fuller comparison with the static approach to sampling. To this end, we selected 11 databases from the UCI repository that vary in the number of features and in the proportion of continuous to nominal features. Since our goal was to learn accurately from a small sample of a large database, and since the UCI databases are all quite small, we artificially inflated each database by making 100 copies of all records, inserting these into a new database, and shuffling (randomizing their order). Very large real databases also have high redundancy (Moller 1993), so we do believe the results of these experiments will be informative, although real large databases would obviously have been preferable.

For each new inflated database, we shuffled the records randomly and ran five-fold cross-validation — we partitioned it into five disjoint and equal-size parts, and repeatedly trained on four out of the five, while testing on the held-out part. For each training step, we first sampled the database using either no sampling (taking all records), static sampling, or dynamic sampling (ELC). We then recorded the number of samples used and the accuracy on the held-out piece. We repeated this entire procedure five times, getting a total of 25 runs of sampling and 25 estimates of accuracy.

We initialized both sampling algorithms with a sample of size 100. For the static scheme, we used a 5% confidence level test that each field had the same distribution as the large database. For dynamic sampling, we fit the learning curve and checked whether $\widehat{\mathtt{acc}}(N) - \mathtt{acc}(n) < 2\%$. In either case, if the sample

Table 2: Sample size (n) and accuracy for 25 runs.

	Naive	Static		Dynamic	
Data set	Acc.	Acc.	n	Acc.	n
Breast Cancer	95.9	95.9	300	95.9	300
Credit Card	77.7	77.0	500	77.2	1180
German	72.7	63.8	540	71.8	2180
Glass2	61.9	60.0	100	61.9	720
Heart Disease	85.1	83.2	180	85.1	900
Hepatitis	83.8	83.2	100	83.8	540
Horse Colic	76.6	76.1	240	76.6	640
Iris	96.0	96.0	100	96.0	560
Lymphography	69.1	67.1	100	68.5	600
Pima Diabetes	75.3	75.7	420	75.5	1080
Tic-tac-toe	69.7	69.2	620	71.1	620

was ruled insufficient we increased the size by 100 and repeated.

Table 2 shows the results on the 11 inflated databases from the UCI repository. Note that extrapolated learning curve sampling meets the PCE criterion with $\epsilon = .02$: in no case was the accuracy on the entire database more than .9% higher than the extrapolated sample accuracy. Static sampling, while approving much smaller samples, did worse at matching the accuracy on the entire database: in two cases, its accuracy was 1.9% worse than the accuracy on the entire database, and on one domain (German) its accuracy was nearly 10% lower.

Related and Future Work

Perhaps the best examples of dynamic sampling are the peepholing algorithm described by Catlett (1992) and the "races" of Moore & Lee (1994). In both approaches the authors identify decisions that the learning algorithm must make and propose statistical methods for estimating the utility of each choice rather than fully evaluating each.

The form of our parametric learning curve comes from Kohavi (1995), who discusses learning curve extrapolation during model selection. Kadie (1995) proposes a variety of methods for fitting learning curves.

In the future we intend to apply PCE to different data mining tools, such as Utgoff's (1994) incremental tree inducer. The tradeoffs effecting variance in the estimated learning curves should also be addressed.

Conclusion

Data mining is a collaboration between a customer (domain expert) and a data analyst. Decisions about how large of a sample to use (or whether to subsample at all) must be made rationally. The dynamic sampling algorithm proposed offers parameters that relate directly to the performance of the resulting model, rather than to a statistical criterion which is related in some unknown way to the desired performance. Because of the interpretability of the PCE criterion and because of the robust performance of dynamic sampling in our experiments, we recommend the general framework and encourage further study towards its computational realization using fewer or more well-founded approximations.

Acknowledgments

George John was supported by an NSF Graduate Research Fellowship, and Pat Langley was supported in part by Grant No. N00014-94-1-0746 from the Office of Naval Research.

References

Casella, G. & Berger, R. L. (1990), *Statistical Inference*, Wadsworth & Brooks/Cole.

Catlett, J. (1992), Peepholing: Choosing attributes efficiently for megainduction, in *Machine Learning: Proceedings of the Ninth International Workshop*, Morgan Kaufmann, pp. 49–54.

Kadie, C. (1995), SEER: Maximum likelihood regression for learning-speed curves, PhD thesis, Computer Science Department, University of Illinois at Urbana-Champaign.

Kohavi, R. (1995), Wrappers for performance enhancement and oblivious decision graphs, PhD thesis, Computer Science Department, Stanford University, Stanford, CA.

Langley, P. & Sage, S. (1994), Induction of selective Bayesian classifiers, in *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, Seattle, WA, pp. 399–406.

Langley, P., Iba, W. & Thompson, K. (1992), An analysis of Bayesian classifiers, in *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, pp. 223–228.

Moller, M. (1993), "Supervised learning on large redundant training sets", *International Journal of Neural Systems* 4(1), March, 15–25.

Moore, A. & Lee, M. (1994), Efficient algorithms for minimizing cross-validation error, in *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann, pp. 190–198.

Yuret, D. (1994), From genetic algorithms to efficient optimization, Master's thesis, Computer Science Department, Massachusetts Institute of Technology.