Similarity-Driven Sampling for Data Mining

Thomas Reinartz

Daimler-Benz AG, Research & Technology, FT3/KL
P.O. Box 2360, 89013 Ulm, Germany
e-mail: reinartz@dbag.ulm.daimlerbenz.com

Abstract. Industrial databases often contain millions of tuples but most data mining algorithms suffer from limited applicability to only small sets of examples. In this paper, we propose to utilize data reduction before data mining to overcome this deficit. We specifically present a novel similarity-driven sampling approach which applies two preparation steps, sorting and stratification, and reuses an improved variant of leader clustering. We experimentally evaluate similarity-driven sampling in comparison to statistical sampling techniques in different classification domains using C4.5 and instance-based learning as data mining algorithms. Experimental results show that similarity-driven sampling often outperforms statistical sampling techniques in terms of error rates using smaller samples.

1 Introduction

Industrial databases often contain millions of tuples but most data mining algorithms suffer from limited applicability to only small sets of examples. In principle, two main alternatives exist. First, scaling up data mining algorithms makes their applications to larger data sets more feasible. Second, data reduction techniques lead to smaller data sets, and if we then apply data mining algorithms to these smaller data sets, data mining becomes more feasible. In this approach, we have to ensure that reduced data sets still contain enough information to reveal appropriate results.

In this paper, we consider data reduction techniques. We specifically present a novel similarity-driven sampling approach which reduces the number of tuples rather than decreasing the number of attributes or values. In the next section, we outline leader sampling as the core approach in similarity-driven sampling and propose two data preparation steps, sorting and stratification, as well as automatic estimation and adaptation strategies for similarity thresholds to overcome negative properties of leader sampling. Thereafter, we experimentally evaluate similarity-driven sampling in comparison to statistical sampling techniques in different classification domains. Finally, we discuss related efforts and outline issues for future work.

2 Similarity-Driven Sampling

Existing efforts towards data reduction for data mining utilize statistical sampling techniques or apply clustering and prototyping approaches (e.g., windowing (Quinlan, 1993) and IBL2 (Aha et al., 1991)). In this paper, we propose to unify clustering and prototyping. The key idea is to generate subsets of sufficiently similar tuples and to select representative prototypes within each subset. Then, we reduce the original data set to the smaller set of prototypes.

2.1 Leader Sampling

The core approach in similarity-driven sampling reuses an improved variant of leader clustering (Hartigan, 1975). Leader clustering simultaneously generates partitions of data and selects leaders within each cluster by a single pass through the data. We propose to follow up leader clustering for data reduction by selecting leaders as representative prototypes without explicitly creating corresponding clusters. The resulting strategy, called leader sampling (LEASAM), works as follows. LEASAM selects the first tuple as the first prototype of the first cluster. For each following tuple, leader sampling considers similarities between this tuple and already selected prototypes. If the similarity between the tuple and the most similar prototype exceeds a pre-defined threshold δ, LEASAM proceeds with the next tuple. Otherwise, leader sampling adds the current tuple to the sample. In this paper, we assume an Euclidian like weighted cumulated similarity measure.

Besides an appropriate similarity measure, leader sampling mainly suffers from the following three drawbacks. First, prototype selection in leader sampling depends on the order of tuples. Second, execution time of leader sampling increases if the set of currently selected prototypes becomes large. Finally, leader sampling relies on appropriate settings of similarity threshold δ. In the following, we define two preparation steps, sorting and stratification, to overcome the first two drawbacks. Then, we develop automatic estimation and adaptation strategies for similarity thresholds.

2.2 Sorting

Sorting ensures a fixed well-defined order of tuples. Therefore, we define the following order relation between tuples which considers attribute values in order of attribute relevance. If the most important attribute value in tuple t is larger than the corresponding value in tuple $\tilde{t}$, then sorting places t after $\tilde{t}$ in the sorted table. If both values coincide, sorting recursively proceeds with the next important attribute in the same way. If two tuples are completely identical, their relation remains the same. For comparisons between continuous attribute values, we use the regular numeric order, whereas for symbolic attributes, we employ the lexicographic order. Note, leader sampling still depends on the new fixed well-defined order of tuples, but now LEASAM always selects the same sample independent of the original order of tuples.

2.3 Stratification

Stratification separates tuples into smaller subsets according to attribute values and again considers values in order of attribute relevance. For continuous attributes, a stratum contains tuples with values in the same interval, whereas for symbolic attributes, a stratum includes tuples with the same value. Note, for continuous attributes, stratification involves discretization approaches to generate intervals (e.g., Dougherty et al., 1995). In the current implementation, we discretize into equal-frequency intervals.

Stratification generates a strata tree. At each level, stratification separates tuples according to single attribute values. Stratification recursively iterates separation according to the next important attribute until a stratum contains less than a pre-defined maximum number of tuples, or no more attributes are available. Stratification returns all leaves of the resulting strata tree.

In terms of leader sampling, stratification is advantageous for two reasons. First, stratification separates tuples into smaller subsets. If we then apply LEASAM within each stratum separately, leader sampling becomes more efficient in terms of running time. Second, each stratum contains locally similar tuples, and the more attributes stratification takes into account, the higher are cumulated similarities within a stratum. Hence, leader sampling is likely to consider the most similar prototypes for each tuple without processing the entire data set.

2.4 Similarity Thresholds

In order to develop estimation and adaptation strategies for similarity thresholds in leader sampling, we relate different δ values and resulting samples to hierarchical clustering. For example, we assume $\delta < \delta' < \delta''$ and data illustrated in figure 1 (black and white bullets). If we apply LEASAM with δ and tuple 1 is the first tuple, leader sampling selects only the first tuple. This prototype represents all tuples within the dotted circle with radius $1 - \delta$. In terms of hierarchical clustering, this prototype represents the root cluster. For threshold δ', leader sampling selects an additional prototype. The final sample contains two tuples, and the corresponding clustering structure refines the initial clustering. If we now increase δ again, leader sampling generates a third sample which again corresponds to a more fine-grained clustering structure.

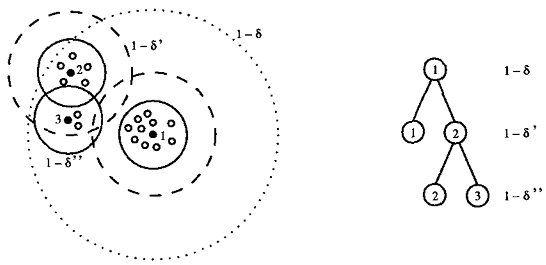

Fig. 1. Increasing similarity threshold values $\delta < \delta' < \delta''$ and resulting samples in leader sampling (left) correspond to refinements in hierarchical clustering (right).

In summary, increasing similarity thresholds correspond to more fine-grained clustering structures. In similarity-driven sampling, we attempt to callibrate δ until prototypes represent subsets of tuples at an appropriate level. Appropriateness of specific levels in hierarchical clustering depends on homogeneity within clusters and heterogeneity between clusters. The general goal is to construct clusterings such that homogeneity within clusters is higher than heterogeneity between clusters. We use minimum similarities between tuples within clusters as a measure of homogeneity and maximum similarities between tuples within different clusters as a measure of heterogeneity.

If we consider each subset of tuples represented by the same prototype in leader sampling as a single cluster, homogeneity depends on similarity threshold δ. High threshold values correspond to high homogeneity. Vice versa, we callibrate δ such that homogeneities within subsets of tuples represented by the same prototype approximately match true homogeneities at the most appropriate level in hierarchical clustering.

Occurring similarities between tuples and their prototypes yield estimates for true homogeneities. Hence, we adapt δ according to observed similarity values in leader sampling. For this reason, we keep minimum, average, and maximum similarities as aggregated information. If similarity-driven sampling adapts δ to the minimum of observed similarities, homogeneity within clusters is low, but heterogeneity between clusters is high. In this case, similarity-driven sampling only selects a few prototypes within a few clusters. If similarity-driven sampling applies the maximum strategy, homogeneity within clusters is high, but heterogeneity between clusters is low. Then, similarity-driven sampling selects more prototypes that represent small homogeneous clusters. The average strategy results in a compromise between homogeneity and heterogeneity.

2.5 Working Sets

In general, it is advisory to adapt similarity threshold values during sampling. If we fix δ, similarity-driven sampling only draws representatives at a single level in the corresponding hierarchical clustering. If we vary δ, it is possible to select representatives at different levels of granularity and to seek for the most appropriate value by moving up and down in the corresponding hierarchical clustering.

In order to reveal appropriate estimations of homogeneities within clusters and dynamically adapt δ, we propose to select intermediate working sets. We apply leader sampling to each working set and use occurring similarities to estimate the current similarity threshold. After each working set, we decide whether to adapt δ to a different value. Since similarity-driven sampling utilizes sorting in advance, we suggest to apply systematic sampling to generate intermediate working sets.

2.6 Algorithm SimSam

Now, we integrate sorting, stratification, and leader sampling with automatic estimation and adaptation of similarity thresholds into an algorithm for similarity-driven sampling (SimSam). SimSam starts sampling with sorting and stratifying the original input data. Within each stratum, similarity-driven sampling utilizes leader sampling on intermediate working sets with varying start positions. As soon as SimSam completes processing a working sample, it compares the last and the current sample size. If sampling does not make progress, i.e., the sample size does not increase, SimSam updates similarity threshold δ according to a pre-specified adaptation strategy. SimSam iterates leader sampling on working sets until either SimSam already processed all tuples in the current stratum, or if similarity-driven sampling does not make progress for a pre-defined number of iterations.

3 Experimental Evaluation

In this section, we experimentally evaluate similarity-driven sampling in comparison to statistical sampling techniques in different classification domains.

3.1 Experimental Procedure

In order to validate benefits of sorting and stratification as well as appropriateness of similarity-driven sampling, we conduct an experimental study on eight different data sets from the UCI repository (Murphy & Aha, 1994). Table 1 outlines characteristics of these data sets and data mining results. For each data set, we first separate the original data set into training and test set. The training set is a random sample which contains approximately 80% of the entire data, and the test set includes remaining tuples. We start data mining by applications of two different classification algorithms to the entire training set. In this study, we compare C4.5 (Quinlan, 1993) and inducer IB in $\mathcal{MLC}$++ (Kohavi et al., 1996). We apply both algorithms with default settings, and for C4.5, we only consider accuracies of pruned decision trees.

Table 1. Data Characteristics

Name	Training Set Size	Test Set Size	Number of Attributes	Error in % C4.5	Error in % IB	Time in Sec. C4.5	Time in Sec. IB
Abalone	3342	835	9	79.4	82.0	10.7	106.5
Balance	500	125	5	30.4	16.0	0.1	3.5
Breast	560	139	11	6.5	5.8	0.2	5.1
Car	1383	345	7	8.7	7.3	0.1	13.0
Credit	552	138	16	16.7	21.0	0.4	6.0
German	800	200	21	26.0	25.5	1.1	22.8
Pima	615	153	9	30.7	30.7	0.6	4.8
Tic-Tac-Toe	767	191	10	10.5	0.0	0.1	6.7

Then, we apply different sampling approaches to each training set and use resulting samples for data mining. We examine six different sampling approaches: Simple random sampling with replacement (R), stratified simple random sampling with replacement (RS), systematic sampling (S), systematic sampling with sorting (SS), leader sampling (L), and similarity-driven sampling with maximum adaptation (SM). First, we apply leader sampling with varying similarity thresholds between $\delta = 0.9$ and $\delta = 0.99$. LEASAM determines its sample size according to the specified threshold. Similarly, SIMSAM chooses as many prototypes as it regards as necessary according to the selected adaptation strategy. For random and systematic sampling, we modify sample sizes within the range of sample sizes of leader and similarity-driven sampling.

For all non-deterministic sampling approaches, we repeat sampling and data mining ten times and present average results. In all experiments, we compute attribute relevance weights according to information gain ratio as if we select the first attribute at top of a decision tree (Quinlan, 1993).

3.2 Experimental Results

Table 2 summarizes experimental results. For each data set, for each sampling approach, and for each data mining algorithm, we show error rate differences in comparison to learning on the entire training set, relative reduced training set size in comparison to the original training set, as well as relative execution time of sampling and data mining in comparison to data mining on the entire training set. Note, each entry in table 2 refers to best results in terms of error rate for samples which contain less than 50% of the original data.

Negative error rate differences indicate that data mining on samples yields more accurate classifiers than learning on the original training set, whereas positive differences depict worse results. For execution time, values below 100 mean that sampling and data mining on resulting samples is faster than data mining on original training sets, whereas values above 100 indicate that sampling and data mining take more time than data mining without reducing the training set.

For each data set and for each data mining algorithm, bold entries refer to best results among different sampling approaches if we consider each aspect separately. For comparisons between sampling approaches, we consider a sampling approach as more appropriate than another sampling approach, if the former yields lower (or equal) error rates on smaller (or equally sized) samples than the latter. If the former approach results in lower error rates but uses larger samples than the latter (or vice versa), we regard these approaches as not comparable.

Detailed analyses of all results (including results not presented here) are beyond the scope of this paper. In summary, we stress the following conclusions:

- Stratification is useful as an enhancement to simple random sampling in some domains. In the majority of domains, both sampling approaches are not comparable.
- Sorting is generally beneficial in combination with systematic sampling. Only in two domains with applications of C4.5, pure systematic sampling outperforms systematic sampling with sorting.

Table 2. Experimental Results

Learning		C4.5						IB					
Data	**Sample**	**R**	**RS**	**S**	**SS**	**L**	**SM**	**R**	**RS**	**S**	**SS**	**L**	**SM**
Abalone	Error Diff.	-0.3	-0.5	-0.6	**-2.9**	4.3	-1.9	-2.3	-2.2	-2.1	**-4.0**	2.9	-3.1
	Size in %	19.7	21.0	**6.3**	20.0	35.2	32.3	37.2	14.2	14.3	**4.2**	35.2	32.3
	Time in %	20.9	24.3	**9.3**	25.3	158.9	45.8	40.2	16.7	15.8	**5.6**	48.2	36.9
Balance	Error Diff.	5.0	4.4	5.4	**2.4**	5.3	**2.4**	13.2	12.6	11.7	**7.2**	19.4	8.8
	Size in %	45.6	43.2	33.4	**11.2**	27.0	19.6	45.6	46.6	**33.4**	**33.4**	42.4	45.6
	Time in %	**200.0**	**200.0**	**200.0**	**200.0**	500.0	300.0	51.4	54.3	57.1	**42.9**	65.7	57.1
Breast	Error Diff.	0.0	0.0	1.5	1.4	**-0.9**	2.1	-0.8	-0.8	0.0	-0.8	-0.3	**-1.5**
	Size in %	40.7	34.6	33.4	33.4	44.3	**24.1**	47.0	33.8	33.4	33.4	37.0	**24.1**
	Time in %	**150.0**	175.0	**150.0**	**150.0**	500.0	250.0	52.9	41.2	39.2	41.2	52.9	**35.3**
Car	Error Diff.	**4.2**	4.6	4.7	5.5	4.5	4.9	8.1	7.2	7.7	7.2	**5.7**	6.0
	Size in %	34.3	35.6	33.3	33.3	34.3	**32.6**	34.3	35.6	33.3	33.3	34.3	**32.6**
	Time in %	**200.0**	**200.0**	**200.0**	300.0	2200.0	500.0	34.6	38.5	**18.5**	40.0	53.8	35.4
Credit	Error Diff.	1.7	1.6	0.3	**-1.5**	0.5	0.0	-1.2	-0.6	-0.3	**-5.8**	1.5	2.2
	Size in %	26.8	48.7	33.3	16.7	47.6	**6.2**	48.6	7.2	33.3	6.0	47.6	**5.3**
	Time in %	**50.0**	100.0	75.0	75.0	350.0	125.0	55.0	**13.3**	40.0	**13.3**	70.0	16.7
German	Error Diff.	4.0	4.1	3.3	**-4.0**	15.9	0.5	5.8	5.4	5.5	**2.5**	12.0	11.5
	Size in %	18.2	33.8	33.2	20.0	30.1	**13.5**	32.2	32.0	33.4	**10.0**	30.1	14.5
	Time in %	**54.5**	81.8	72.7	**54.5**	290.9	127.3	36.0	36.0	42.1	**14.5**	46.9	22.8
Pima	Error Diff.	-2.1	-1.9	-1.0	**-6.5**	4.3	-3.2	-1.2	0.5	0.7	**-1.9**	8.6	-1.3
	Size in %	40.2	41.6	20.0	20.0	40.2	**14.8**	5.5	22.1	4.7	**1.1**	40.2	14.8
	Time in %	50.0	66.7	**33.3**	**33.3**	166.7	66.7	10.4	29.2	10.4	**8.3**	58.3	25.0
Tic-Tac-Toe	Error Diff.	10.9	9.0	10.8	12.5	11.3	**7.8**	6.8	5.2	6.1	5.8	5.0	**2.6**
	Size in %	31.8	43.3	33.4	33.4	**28.8**	41.6	42.6	43.3	33.4	33.4	**28.8**	42.2
	Time in %	**100.0**	200.0	**100.0**	200.0	900.0	500.0	46.3	46.3	**37.3**	**37.3**	43.3	49.3

- If experiments of leader sampling and similarity-driven sampling are comparable, SimSam always produces better results than LeaSam. We also notice that similarity-driven sampling is faster than leader sampling in all domains, except on tic-tac-toe if we apply instance-based learning.
- Although about 50% of experimental results are not comparable if we relate statistical sampling techniques to similarity-driven sampling, we recognize

that SimSam always outperforms simple random sampling as well as systematic sampling in domains where comparisons are reasonable.

All in all, these results indicate that both preparation steps, sorting and stratification, are useful enhancements and that similarity-driven sampling is an appropriate alternative to statistical sampling techniques.

4 Related Work

The origin of all efforts to learn on subsets of tuples is Quinlan's windowing approach in ID3. Wirth and Catlett (1988) showed that costs of using windowing in ID3 are almost always significantly increase running time or do not lead to improvements. Catlett (1991) considered windowing in C4.5 which uses stratification as SimSam but only according to the class attribute. Experimental results indicate that windowing increases accuracy for noisy domains with continuous attributes. In general, effects of windowing are not significant but it slows down induction.

John and Langley (1996) discuss static versus dynamic sampling for data mining. Dynamic sampling uses the PCE (probably close enough) criterion, and a sample meets this criterion if the probability is low that data mining on the entire data set achieves higher accuracy. For naive bayes, their comparison showed only minor differences between static and dynamic sampling.

Toivonen (1996) and Zaki et al. (1997) examine applications of random sampling for finding association rules. They both report that sampling speeds up computation of large itemsets by reducing I/O costs. Zaki et al. only consider computation of large itemsets and demonstrate that up to 80% of true large itemsets can be found on a random sample. Toivonen uses itemsets produced on a sample as hypotheses that are tested on the entire database.

Ester et al. (1995) describe sampling on representatives for data mining with Clarans (Clustering Large Applications based on Randomized Search). They use R^*-trees to determine center points for each data page in a spatial database, and then run their clustering algorithm on the set of center points. In terms of efficiency, focusing on representatives gains a significant speed-up. In terms of effectiveness, their sampling approach is slightly worse than clustering the entire database.

5 Conclusion

In this paper, we introduced similarity-driven sampling for data mining. In summary, similarity-driven sampling often outperforms statistical sampling techniques in terms of error rates using smaller samples if we train C4.5 and instance-based learning on reduced data sets and estimate their accuracy on separate test sets. Systematic sampling with sorting also yields astonishing good results.

We propose several technical enhancements for future work. For example, SimSam's performance will benefit from more sophisticated discretization approaches in stratification. We also plan alternative uses of different similarity measures. Future work also includes efforts to re-implement similarity-driven sampling such that SIMSAM utilizes direct access to databases.

The experimental study also raises more basic research questions. Differences in success between statistical sampling techniques and more intelligent sampling approaches are less significant than generally expected. The best sampling approach depends on both data characteristics and the selected data mining algorithm. The ultimate goal is to provide a set of sampling approaches accompanied by guidelines which context requires which data reduction approach.

References

Aha, D.W., Kibler, D., & Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, p. 37-66.

Catlett, J. (1991). *Megainduction: Machine Learning on Very Large Data-bases.* Ph.D. Thesis, University of Sydney, Australia.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. in: Prieditis, A., & Russell, S. (eds.). *Proceedings of the 12th International Conference on Machine Learning.* July, 9-12, Tahoe City, CA. Menlo Park, CA: Morgan Kaufmann, pp. 194-202.

Ester, M., Kriegel, H.-P., & Xu, X. (1995). Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. in: Egenhofer, M.J., & Herring, J.R. (eds.) *Proceedings of the 4th International Symposium on Spatial Databases.* August, 6-9, Portland, Maine. New York, NY: Springer, pp. 67-82.

Hartigan, J.A. (1975). *Clustering Algorithms.* New York, NY: John Wiley & Sons, Inc.

John, G.H., & Langley, P. (1996). Static Versus Dynamic Sampling for Data Mining. in: Simoudis, E., Han, J., & Fayyad, U. (eds.) *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.* August, 2-4, Portland, Oregon. Menlo Park, CA: AAAI Press, pp. 367-370.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1996). *Data Mining Using $\mathcal{MLC}$++: A Machine Learning Library in C++.* http://robotics.stanford.edu/~ronnyk.

Murphy, P.M., & Aha, D. (1994). *UCI Repository of Machine Learning Databases.* ftp://ics.uci.edu/pub/machine-learning-databases.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Toivonen, H. (1996). Sampling Large Databases for Finding Association Rules. in: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., & Sarda, N.L. (eds.) *Proceedings of the 22nd International Conference on Very Large Databases.* September, 3-6, Mumbai, India. San Mateo, CA: Morgan Kaufmann, pp. 134-145.

Wirth, J., & Catlett, J. (1988). Experiments on the Costs and Benefits of Windowing in ID3. in: Laird, J. (ed.) *Proceedings of the 5th International Conference on Machine Learning.* June, 12-14, University of Michigan, Ann Arbor. San Mateo, CA: Morgan Kaufmann, pp. 87-99.

Zaki, M.J., Parthasarathy, S., Li, W., & Ogihara, M. (1997). Evaluation of Sampling for Data Mining of Association Rules. in: Scheuermann, P. (ed.) *Proceedings of the 7th Workshop on Research Issues in Data Engineering.* April, 7-8, Birmingham, England. Los Alamitos, CA: IEEE Computer Society Press.