When and How to Subsample: Report on the KDD-2001 Panel

Pedro Domingos
Dept. of Computer Science & Engineering
University of Washington
Seattle, WA 98195, U.S.A
pedrod@cs.washington.edu

Introduction

Databases in the terabyte range are now common. In many domains, mining all the data available in reasonable time is already beyond the reach of current systems. Yet the size of databases continues to grow rapidly. Is subsampling unavoidable? Or should it be avoided at all costs? If we subsample, what is the best way to do it? What issues must be taken into account? The KDD-2001 Panel on When and How to Subsample addressed these and related questions, with the twin goals of developing practical guidelines and identifying key research issues. It was chaired by Pedro Domingos (University of Washington), and the participants were Surajit Chaudhuri (Microsoft Research), David Jensen (University of Massachusetts at Amherst), Ronny Kohavi (Blue Martini), and Foster Provost (New York University). Below is each panelist's summary of his position.

Surajit Chaudhuri

Enterprise data is increasingly in relational databases. So, the right issue to discuss is the role of subsampling for data mining on relational databases. I will just use the term sampling as a proxy for subsampling - this is common in the database literature. The first question to ask is what is the objective of sampling and the impact and feasibility of sampling depends much on that answer.

When a data analysis algorithm requires multiple scans over the set of observations, execution of such an algorithm can be very expensive on a large data set. If instead it is possible to use a variant of the algorithm that is capable of using a small uniform random sample, then sampling indeed is very effective. But, this obvious advantage may not be easily realized because of the following reasons. First, sampling could greatly compromise quality and therefore a quantitative model is needed that can precisely define the degree of error with respect to the outcome of the algorithm. Without such a quantitative model, sampling can be a slippery slope. Sadly, this is often missing. Next, if the algorithm requires only a single (or almost a single) pass over the data, then the saving from sampling is much harder to realize. This is because the set of observations that is input to the algorithm is often generated on the fly by combining information from multiple tables in the back end relational database via a SQL query. Therefore, taking a sample of this set of observations requires us to sample the output of the corresponding SQL query. This is no trivial task and can in some situations be no less expensive than evaluating the entire SQL query, especially when an appropriate set of indices is not available in the database (see Chaudhuri et al. paper in SIGMOD-99). Finally, note that sometimes a uniform random sample is not the right kind of sampling; for example, consider the simple task of estimating the total sales of goods in a Wal-Mart that sells Pepsi as well as expensive exercise machines.

David Jensen

Sampling is a central issue in knowledge discovery and data mining. First, it is nearly unavoidable. Few real analyses ever work with a data set that represents the entire population of data, and sampling is one of the simplest methods for scaling up KDD algorithms. Second, sampling is essential for evaluating algorithms. Samples can be used to create learning curves and unbiased estimates of accuracy through cross-validation. Finally, additional research is needed on how to sample in non-independent data sets. While simple random sampling works well for data that are independently distributed, spatial, temporal, and other types of relations among instances can greatly complicate sampling.

Ronny Kohavi

The panel was formed with the tentative name "To subsample or not to subsample," and changed its name to "When and how to subsample," because the panelists agreed that sampling is a tool. When you have a hammer, the question should not be whether to use it or not, but rather when to use. Since sampling, at least theoretically, reduces the accuracy of operations on the subsample, it should be traded off against the advantages it provides, which are reduced resources, and in particularly time. The tradeoff between accuracy and resources is therefore an economic decision. Here are a few extreme examples where economics will dictate that sampling should be used:

- A learning algorithm is used that loads data into memory (RAM). To mine a three terabyte dataset, either a machine with about three terabytes of RAM needs to be purchased, or a new algorithm can be developed that is not limited to RAM. Both of these options may be deemed too expensive and a loss in accuracy due to sampling will be tolerated.
- The running time of a learning algorithm is quadratic, $t = c\,n^2$, where n is the number of records, t is the time in hours, and c is 10^{-9}. The running time for 1,000 records is therefore 3.6 seconds, for 10,000 records it is 6 minutes. However, the running time for 1 million records is 41 days and for 10 million records it is over 11 years. Economics dictate that something must be done and sampling should be considered as one of the alternatives to reduce the running time (others include improving the algorithm, obtaining a faster CPU or multiple CPUs, etc).

- Daily reports take more than 24 hours to compute. In such a case sampling provides an alternative: trade accuracy for timeliness.
- The learning curve looks similar to Figure 1 below, where it appears that Naive-Bayes using 1000 records has asymptoted, outperforming C4.5 on any amount of data tested. Using a dataset of a billion records is unlikely to improve the Naive Bayes model, so economics indicates that in this situation, sampling to a few thousand records suffices.

The question on how to sample should be motivated by similar economic considerations. Sample in a way that provides the best accuracy and uses the least amount of resources. For example, adaptively sample so that "all" decisions are based on large samples. When building a decision tree, for example, always choose an attribute to split based on a minimum sample size. Do not sample at the wrong granularity (e.g., sampling web traffic by pages instead of sessions or visitors).

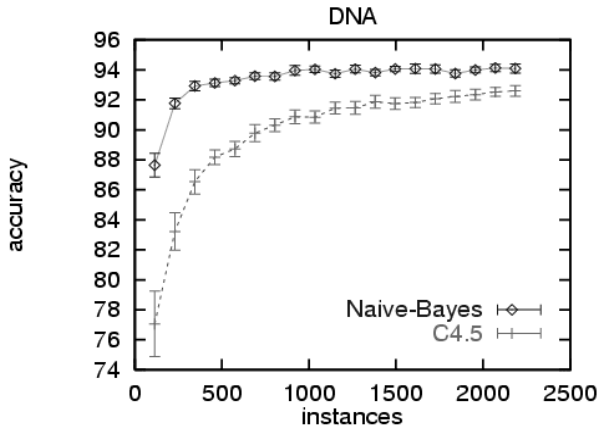

Figure 1

Foster Provost

Subsampling has various uses. For the panel, I concentrated on subsampling when the costs and constraints (e.g., budget, time, memory size) imposed by the problem, the computing environment, etc., do not allow one to mine the entire data set. How to subsample depends on the particular costs and constraints of the problem/environment, and often presents an open research question. Generally, the quality of mined knowledge exhibits decreasing marginal improvements as the sample becomes larger; inductive learning, for example, exhibits the familiar learning curve. However, even with straightforward random sampling it can be difficult to predict *a priori* the size of the sample that is big enough. One option is to take progressively larger samples, and observe the changes in quality - perhaps modeling the relationship between quality and data size. On the other hand, the environment may specify a particular maximum data-set size, due to computational or budgetary limitations (e.g., only *n* data items can fit in main memory; or the budget only allows the cleaning and preprocessing of *n* data items). In such cases, simple random sampling may not be the best alternative. For example, sampling from the classes in a proportion different from the natural proportion can produce better classifiers. Costly data processing also can benefit from "active" sampling, where knowledge learned "so far" helps to guide the selection of subsequent data. Finally, it must be pointed out that conclusions should be drawn cautiously from subsampled data. For example, performance rankings of inductive algorithms observed from subsamples often do not hold on the larger sample, which demands caution even with standard uses of subsampling, such as for evaluation.