Catching Up with the Data: Research Issues in Mining Data Streams

Pedro Domingos **Geoff Hulten**
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98185-2350, U.S.A.
{pedrod,ghulten}@cs.washington.edu

Abstract

In many domains, data now arrives faster than we are able to mine it. To avoid wasting this data, we must switch from the traditional "one-shot" data mining approach to systems that are able to mine continuous, high-volume, open-ended data streams as they arrive. In this paper we identify some desiderata for such systems, report on our recent work realizing them, and outline a number of directions for future research.

1 The Problem

Many (or most) organizations today produce an electronic record of essentially every transaction they are involved in. When the organization is large, this results in millions of records being produced every day. For example, in a single day WalMart records 20 million sales transactions, Google handles 70 million searches, and AT&T produces 275 million call records. Scientific data collection (e.g., by earth sensing satellites or astronomical observatories) routinely produces gigabytes of data per day. Data rates of this level have significant consequences for data mining. For one, a few months' worth of data can easily add up to billions of records, and the entire history of transactions or observations can be in the hundreds of billions. Current algorithms for mining complex models from data (e.g., decision trees, sets of rules) cannot mine even a fraction of this data in useful time. Further, mining a day's worth of data can take more than a day of CPU time, and so data accumulates faster than it can be mined. As a result, despite all our efforts in scaling up mining algorithms, in many areas the fraction of the available data that we are able to mine in useful time is rapidly dwindling towards zero. Overcoming this state of affairs requires a shift in our frame of mind from mining databases to mining data streams. In the traditional data mining process, the data to be mined is assumed to have been loaded into a stable, infrequently-updated database, and mining it can then take weeks or months, after which the results are deployed and a new cycle begins. In a process better suited to mining the high-volume, open-ended

data streams we see today, the data mining system should be continuously on, processing records at the speed they arrive, incorporating them into the model it is building even if it never sees them again. A system capable of doing this needs to meet a number of stringent design criteria:

- It must require small constant time per record, otherwise it will inevitably fall behind the data, sooner or later.
- It must use only a fixed amount of main memory, irrespective of the total number of records it has seen.
- It must be able to build a model using at most one scan of the data, since it may not have time to revisit old records, and the data may not even all be available in secondary storage at a future point in time.
- It must make a usable model available at any point in time, as opposed to only when it is done processing the data, since it may never be done processing.
- Ideally, it should produce a model that is equivalent (or nearly identical) to the one that would be obtained by the corresponding ordinary database mining algorithm, operating without the above constraints.
- When the data-generating phenomenon is changing over time (i.e., when concept drift is present), the model at any time should be up-to-date, but also include all information from the past that has not become outdated.

At first sight, it may seem unlikely that all these constraints can be satisfied simultaneously. However, we have recently developed a decision tree induction system that meets the first five, and are currently extending it to meet the sixth (Domingos & Hulten, 2000). The system is called VFDT, and is able to mine on the order of a billion examples per day using off-the-shelf hardware, while providing strong guarantees that its output is very similar to that of a "batch" decision tree learner with access to unlimited resources. We have also begun to develop VFKM, a version of k-means clustering with similar properties. The goals above, and our experience with VFDT and VFKM, lead us to identify a number of research questions:

- How much data is enough? Even if we have (conceptually) infinite data available, it may be the case that we do not need all of it to obtain the best possible model of the type being mined. Assuming the data-generating process is stationary, is there some point at which we can "turn off" the stream and know that we will not lose predictive performance by ignoring further data?
- If the data-generating process is not stationary, how do we make the trade-off between being up-to-date and not losing past information that is still relevant? In the traditional method of mining a sliding window of data, a large window leads to slow adaptation, but a small one leads to loss of relevant information and overly-simple models. Can we overcome this trade-off?

- Which data mining algorithms are best suited to mining fast data streams? Some algorithms may be harder to adapt to the data-stream setting than others. Can we identify general properties that affect the ease of transformation, and devise general techniques to transform entire classes of algorithms?

We consider each of these in turn.

2 How Much Data Is Enough?

A number of well-known results in statistics provide probabilistic bounds on the difference between the true value of a parameter and its empirical estimate from finite data. For example, consider a real-valued random variable x whose range is R. Suppose we have made n independent observations of this variable, and computed their mean $\bar{x}$. The Hoeffding bound (Hoeffding, 1963) (also known as additive Chernoff bound) states that, with probability at least $1-\delta$, and irrespective of the true distribution of x, the true mean of the variable is within ϵ of $\bar{x}$, where

$$\epsilon = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}}$$

Put another way, this result says that, if we only care about determining x to within ϵ of its true value, and are willing to accept a probability of δ of failing to do so, we only need to gather $n = \frac{1}{2}(R/\epsilon)^2 \log(2/\delta)$ samples of x. More samples (up to infinity) produce in essence an equivalent result. The main thrust of our research has been to "bootstrap" these results, which apply to individual parameters, to similar guarantees on the difference (loss) between the whole complex model mined from finite data and the model that would be obtained from infinite data in infinite time. These results allow us to dynamically decide how many samples from the data stream to use for each decision or parameter, while guaranteeing that the end result will be as close to the "infinite data" case as the user desires. We have so far successfully applied this method to decision tree induction and k-means clustering. Many open issues remain, including obtaining better bounds for these two cases, extending the approach to different algorithms and different types of data mining, and using distribution-specific bounds instead of the generic Hoeffding ones.

3 How Should Mined Models Evolve?

We have recently extended VFDT to handle time-changing phenomena by allowing examples to be forgotten as well as remembered. Forgetting an example involves subtracting it from the sufficient statistics it was previously used to compute. When there is no drift, new examples are statistically equivalent to the old ones and the decision tree does not change, but if there is drift a new best split for a given node may surface. In this case we begin to grow an alternative subtree using the new best split, and replace the old subtree with the new one when the latter becomes more accurate on new data. Replacing the old subtree with the new node right away would produce a result similar to windowing, but at a cost of $O(1)$ per new example, as opposed to $O(w)$, where w is the size of the window. Waiting

until the new subtree becomes more accurate ensures that past information continues to be used for as long as it is useful, and to some degree overcomes the trade-off implicit in the choice of window size. However, for very rapidly changing data the pure windowing method may still produce better results (assuming it has time to compute them before they become outdated, which may not be the case). An open direction of research that we are beginning to pursue is to allow the "equivalent window size" (i.e., the number of time steps that an example is remembered for) to be controlled by an external variable or function that the user believes correlates with the speed of change of the underlying phenomenon. As the speed of change increases the window shrinks, and vice-versa. Further research involves explicitly modeling different types of drift (e.g., cyclical phenomena, or effects of the order in which data is gathered), and identifying optimal model updating and management policies for them. Example weighting (instead of "all or none" windowing) and subsampling methods that approximate it are also relevant areas for research.

4 Which Techniques Are Best Suited to Mining Streams?

We have found that adapting decision tree induction to mining data streams is easier than adapting k-means clustering. The reason is that k-means is more sensitive to the "statistical jitter" that results from a stream of all-new examples being used to do the job that previously was done by repeatedly scanning the same examples. Decision tree induction is an example of a data mining process where once a decision is made it is cast in concrete (once chosen, the test for a node is never changed). In contrast, k-means clustering is an example of a process where the end result (the cluster centroids) can change arbitrarily from step to step, and errors may compound each other. An important direction for research is to systematize these distinctions, uncover further relevant ones, and devise general methods that apply to whole classes of algorithms with similar properties. The high-level approach that we propose consists of three steps:

1. Derive an upper bound on the time complexity of the mining algorithm, as a function of the number of samples used in each step.
2. Derive a upper bound on the relative loss between the finite-data and infinite-data models, as a function of the number of samples used in each step of the finite-data algorithm.
3. Minimize the time bound (via the number of samples used in each step) subject to user-defined limits on the loss.

Where successful, this approach effectively allows us to mine infinite data in finite time, "keeping up" with the data no matter how much of it arrives. The tighter the bounds, the more efficient the resulting algorithm will be. What algorithms this approach can be usefully applied to is an open question. Our immediate plans are to extend it from k-means to learning mixture models via the EM algorithm (of which k-means is a special case), and from there to other applications of EM (e.g., learning Bayesian networks in the presence of missing data) and other iterative-optimization algorithms (e.g., backpropagation). In parallel, we would

like to extend the decision-tree treatment to mining rules, building ensembles of models via methods like boosting, and other sequential decision processes like feature and instance selection.

5 Conclusion

In many domains, the massive data streams available today make it possible to build more intricate (and thus potentially more accurate) models than ever before, but this is precluded by the sheer computational cost of model-building; paradoxically, only the simplest models are mined from these streams, because only they can be mined fast enough. Alternatively, complex methods are applied to small subsets of the data. The result (we suspect) is often wasted data and outdated models. In this paper we outlined some desiderata for data mining systems that able to "keep up" with these massive data streams, and some of the research issues that these desiderata bring up. These include determining how much data is needed to obtain a model that is (nearly) as good as an infinite-data one, finding efficient methods to let a model evolve without losing valuable past information, and determining general properties of database mining algorithms that allow us to efficiently transform them into data-stream mining ones.

References

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 71–80). Boston, MA: ACM Press.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.