Cached Sufficient Statistics for Automated Mining and Discovery from Massive Data Sources

The Auton Lab (http://www.cs.cmu.edu/∼AUTON)
Andrew Moore, awm@cs.cmu.edu, 412-268-7599 (Principal Investigator)
Jeff Schneider, schneide@cs.cmu.edu, 412-268-2339 (Principal Investigator)
Brigham Anderson, brigham@cs.cmu.edu
Scott Davies, scottd@cs.cmu.edu
Paul Komarek, komarek@cs.cmu.edu
Mary Soon Lee, mslee@cs.cmu.edu
Marina Meila, mmp@ai.mit.edu
Remi Munos, munos@cs.cmu.edu
Kary Myers, kary@cs.cmu.edu
Dan Pelleg, dpelleg@cs.cmu.edu

Carnegie Mellon University
Robotics Institute and School of Computer Science
Pittsburgh, PA 15213

July 6, 1999

There many massive databases in industry and science. There are also many ways that decision makers, scientists, and the public need to interact with these data sources. Wide ranging statistics and machine learning algorithms similarly need to query databases, sometimes millions of times for a single inference. With millions or billions of records (e.g. biotechnology databases, inventory management systems, astrophysics sky surveys, corporate sales information, science lab data repositories) this can be intractable using current algorithms.

The Auton lab (at Carnegie Mellon University) and Schenley Park Research Inc. (a start-up company), both jointly run by Andrew Moore and Jeff Schneider, are concerned with the fundamental computer science of making very advanced data analysis techniques computationally feasible for massive datasets.

The computational challenge

How can huge data sources (Gigabytes up to Terabytes) be analyzed automatically? There is no off-the-shelf technology for this. There are devastating computational and statistical difficulties; manual analysis of such data sources is now passing from being simply tedious into a new, fundamentally impossible realm where the data sources are just too large to assimilate by humans. This situation is ironic given the large investment the US has put into gathering scientific data. The only alternative is automated discovery.

It is our thesis that the emerging technology of **cached sufficient statistics** will be critical to developing automated discovery on massive data. A cached sufficient statistics representation is a data structure that summarizes statistical information in a database. For example, human users, or statistical programs, often need to query some quantity (such as a mean or variance) about some subset of the attributes (such as size, position and shape) over some subset of the records. When this happens, we want the cached sufficient statistic representation to intercept the request and, instead of answering it slowly by database accesses over billions of records, answer it immediately.

The interesting technical challenge is: given that there are uncountably many potential statistical queries, how could we hope to precompute them all in advance? And wouldn't it require hopless amounts of memory to pre-store the answer to every possible question just on the off-chance that each question is asked? Happily there are some algebraic tricks such that for an increasingly large class of traditional data mining and statistical operations, it *is* possible to do this. By means of AD-trees (All-dimensions trees) (Moore & Lee, 1998; Anderson & Moore, 1998) we can now do constant-time counting queries for up to hundreds of dimensions. And by means of MRKD-trees (multiresolution k-dimensional trees) (Deng & Moore, 1995; Moore, Schneider, & Deng, 1997) (an extension of KD-trees (Friedman, Bentley, & Finkel, 1977)) we can perform clustering, and a very wide class of non-parametric statistical techniques on enormous data sources several thousand times faster than previous algorithms (Moore, 1999).

We are currently researching extensions to the theory and data structures within projects within CMU. And within SPR we are extending the capabilities of cached sufficient statistics in new ways tailored to commercial application.

The consequence of this ability to do statistics at several orders of magnitude faster are interesting. We can now afford to search amongst millions of potential correlations (or, in statistical terms, among millions of potential models) in the time that would have previously only permitted thousands. Thus, (whilst being careful to avoid overfitting—statistically insignificant correlations) we can be much more aggresive about finding surprising patterns in all kinds of corners of the database.

1 Cached Sufficient Statistics

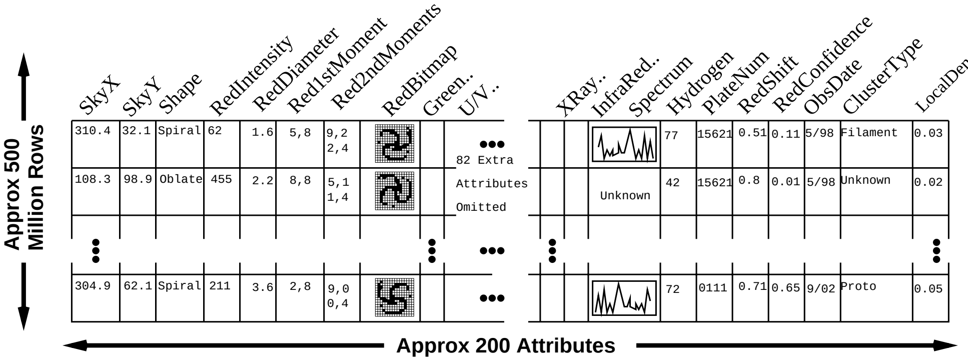
Problem. Consider a huge data matrix (e.g. Figure 1) in which there are millions of rows, each corresponding to one galaxy. There are one or two hundred columns, each specifying a galaxy property. What questions, or tasks, will astrophysicists want to ask of the data?

- A range of *counting queries* such as "how many elliptical galaxies, with a redshift above 0.3, are there?", "what is the mean and variance of ellipticity among radio galaxies within clusters of galaxies compared to outside clusters of galaxies?" or "how does the distribution of galaxy colors observed on May 1st 1999 compare to those seen on June 1st 1999?"

- Simple *visualization queries* such as "give me a smoothed map of all X–ray detected galaxies".

- Sophisticated *statistical queries* which require the clustering, classification, regression and newer probabilistic inference techniques.

Many such questions could be answered easily if there were few records. But with $O(10^9)$ records, even simple individual counts may take hours. Much worse, the visualization and statistics each require at least thousands of these individually time-consuming counts, and so become unthinkable, computationally. We propose a different approach to counting, visualization and statistics based on Cached Sufficient Statistics. The job of a cached sufficient statistic is to pre-compute certain cached information that will allow questions to be answered without a need for scanning the database. The science and technology of cached sufficient statistics is in its infancy and must be extended in many ways for practical use, especially for problems as big as those connected with astrophysics. We must extend the range of possible questions they can service. We must permit them to scale so that the cached sufficient statistics themselves don't require intractable amounts of memory, or time to build, or time to access. And we must make them compatible with the kinds of real statistics and analysis that statisticians and scientists currently can do on small datasets but cannot do on our datasets. It is important to note that this same architecture and software will be applicable for many other domains such as bioinformatics, marketing, manufacturing process monitoring (Atkeson, Moore, & Schaal, 1997).


Approx 500 Million Rows
Approx 200 Attributes

SkyX	SkyY	Shape	RedIntensity	RedDiameter	Red1stMoment	Red2ndMoments	RedBitmap	Green..	U/V..		XRay..	InfraRed..	Spectrum	Hydrogen	PlateNum	RedShift	RedConfidence	ObsDate	ClusterType	LocalDense
310.4	32.1	Spiral	62	1.6	5,8	9,2 2,4			•••	82 Extra Attributes Omitted				77	15621	0.51	0.11	5/98	Filament	0.03
108.3	98.9	Oblate	455	2.2	8,8	5,1 1,4							Unknown	42	15621	0.8	0.01	5/98	Unknown	0.02
⋮								⋮	•••		⋮									⋮
304.9	62.1	Spiral	211	3.6	2,8	9,0 0,4			•••					72	0111	0.71	0.65	9/02	Proto	0.05

Figure 1: A massive datasource, containing hundreds of attributes: some numeric, some categorical, and some exotic (bitmaps and spectra). There are many millions of rows, and there are missing values in some fields of some records.

Current State of the Art. Simple cached sufficient statistics have been in use for decades. The simplest is **binning** (also known as histogramming) in which statistics about single or pairwise attributes are precomputed (Ioannidis & Poosala, 1995). Although often useful, this doesn't help if queries involve three-or-more attributes or are specific to an arbitrarily-chosen subset of the records (e.g. "give me the mean intensity of galaxies that are Spiral, in this part of the sky, and have below average X-ray emissions"). Another simple cached sufficient statistic is to merely use a small sample of the data instead of the full data, and hope that answers to questions about the sample generalize to the full dataset (e.g., (Haas, Naughton, Seshadri, & Stokes, 1995; Ganguly, Gibbons, Matias, & Silberschatz, 1996)). This is an immensely common approach, brought about by necessity, but it loses the ability to make fine distinctions, or notice rare events or anomalies. More advanced technology that could be called Cached Sufficient Statistics include

- Frequent sets (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996; Mannila & Toivonen, 1996). These are often applied to supermarket data analysis, and typically require that all attributes are sparse in order to remain tractable.
- DataCubes (Harinarayan, Rajaraman, & Ullman, 1996) maintain actual and marginal distributions in n attributes. The Data Firehose (John & Lent, 1997) is a similar idea. For example, consider a table with attributes (x-bin, y-bin, galaxy-type), after x and y have been binned. Then, an un-optimized implementation of the Datacube would have the counts of galaxies, for all the possible combinations of the attribute values, including 'don't cares'. This naive representation explodes exponentially with the number of attributes and although ingeneous ordering of the attributes have allowed datacube technology to extend to about

seven joint attributes (Gray, Bosworth, Layman, & Priahesh, 1996), current methods are not applicable for the hundreds of attributes in our galaxy datasets.

- Indexing structures in database management systems such as kd-trees (Bentley, 1980; Moore, 1990), R-trees (Guttman, 1984), Ball-trees (Omohundro, 1987, 1991) or hash-tables, can be used to select a subset of records from the full dataset, which can then be processed. This approach is promising for fast extraction of relevant data and we plan to consider it in our proposed work. Our goal is to extend it, so that it can handle the statistical queries we want to ask in our astrophysics databases.

- **AD-trees:** This new data structure, introduced in (Moore & Lee, 1998), uses the algebra of sufficient statistics and contingency tables to store enough information so that any counting query can be answered in *constant time*, independent of the number of records in the dataset. An important proviso is that this data structure assumes that all attributes are categorical, so any numerical values in the dataset must be quantized into a discrete set of values. This task of being able to answer any question in constant time would be easy if time to build and memory to store the cached sufficient statistic were unlimited; the cached sufficient statistic would be simply a precomputed list of answers to all possible questions. That simple solution is completely intractable in practice because the number of possible questions is, in the *best* case, exponential in the number of attributes. The main innovations in ADtrees allow the memory and build time to be massively reduced (in one case from 10^{38} bytes down to 10^{6} bytes) by storing only very rare surprises, and deducing any other count by a combination of additions and subtractions of surprise values. We have proved that we can still answer any counting query in constant time and that the worst-case and average-case memory and time bounds for an ADtree are exponentially better than those for the naive approach.

  ADtrees empirically give a 50-fold to 5000-fold computational speedup for machine learning methods such as Bayes net structure finders (Moore & Lee, 1998), rule learners (Anderson & Moore, 1998) and feature selectors operating on large datasets of up to 100 attributes.

- **Multiresolution kd-trees** Kd-trees (Bentley, 1980) and R-trees (Guttman, 1984) have long been used as methods for efficiently indexing large databases involving spatially distributed records. They allow the user, or the statistics engine, to efficiently find all the records in some region. This can greatly accelerate the answering of certain statistical queries *provided that the set of records found by the query is small.* In our multiresolution work (Deng & Moore, 1995; Moore et al., 1997), we have extended these data structures so that they can also efficiently answer questions that match large fractions (i.e. possibly millions) of records, and that can answer questions about locally distance weighted combinations of records. We have shown that the costs are asymptotically logarithmic in the dataset size, even if all queries involve a substantial fraction (e.g. 30%) of the data. These include kernel density queries, locally weighted regression, cased-based queries, many forms of mixture-model queries. Multiresolution kd-trees are efficient for anywhere between one to twenty dimensional spaces, with higher dimensionalities possible if the attributes are highly interrelated.

  The augmentation over previous methods includes, at each node, the sufficient statistics to perform a Bayesian regression analysis on all the points inside the node. Without needing to rebuild the tree, it can make fast predictions with arbitrary local weighting functions, arbitrary kernel widths and arbitrary queries. This gives a new, faster, algorithm for exact answers to questions. But, often more more usefully, we can also introduce an approximation that achieves up to a two-orders-of-magnitude extra speedup with negligible accuracy losses.

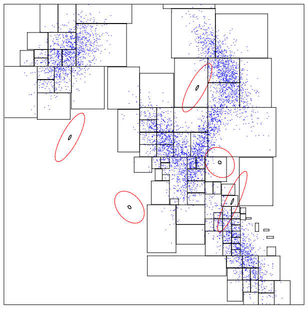


Figure 2: An illustration of the work done by the kd-tree-based clusterer on 100,000 datapoints in two dimensions during one iteration of the EM algorithm. Instead of performing 100,000 center updates on the individual points, it only "visits" the rectangles shown and updates centers according to cached statistics within each rectangle. In this small problem we achieve a 150-fold computational speedup, but since costs are logarithmic in the dataset size, we anticipate much larger speedups on the galaxy data.

(i) Fast, Tractable Clustering. The usual method for finding the parameter estimates – called the EM algorithm – is too slow to be useful for our large data sets. We are developing a new method that uses a new version of MRKD-trees that is many times faster. This preliminary work (started at the same time this proposal was started) has resulting in an algorithm which, for Gaussian mixture models, is many times faster than conventional implementations, yet does exactly the same statistics, not an ad-hoc approximation. Figure 2 shows a simplified example.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*. AAAI Press.

Anderson, B., & Moore, A. W. (1998). AD-trees for fasting counting and rule learning. In *KDD98 Conference (to appear)*.

Atkeson, C. G., Moore, A. W., & Schaal, S. A. (1997). Locally Weighted Learning for Control. *AI Review*, *11*, 75–113.

Bentley, J. L. (1980). Multidimensional Divide and Conquer. *Communications of the ACM*, *23*(4), 214—229.

Deng, K., & Moore, A. W. (1995). Multiresolution Instance-based Learning. In *Proceedings of IJCAI-95*. Morgan Kaufmann.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. on Mathematical Software*, *3*(3), 209–226.

Ganguly, S., Gibbons, P., Matias, Y., & Silberschatz, A. (1996). A sampling algorithm for estimating join size. In *Proc. of ACM SIGMOD* Montreal, Canada. to appear.

Gray, J., Bosworth, A., Layman, A., & Priahesh, H. (1996). Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Proc. 12th International Conference on Data Engineering*, pp. 152–159 New Orleans, Louisiana.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*. Assn for Computing Machinery.

Haas, P. J., Naughton, J. F., Seshadri, S., & Stokes, L. (1995). Sampling-based estimation of the number of distinct values of an attribute. In *Proc. of VLDB*, pp. 311–322 Zurich, Switzerland.

Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing Data Cubes Efficiently. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1996*, pp. 205–216. Assn for Computing Machinery.

Ioannidis, Y. E., & Poosala, V. (1995). Balancing histogram optimality and practicality for query result size estimation. In *ACM SIGMOD*, pp. 233–244 San Jose, CA.

John, G. H., & Lent, B. (1997). SIPping from the data firehose. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.* AAAI Press.

Mannila, H., & Toivonen, H. (1996). Multiple uses of frequent sets and condensed representations. In E. Simoudis and J. Han and U. Fayyad (Ed.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* AAAI Press.

Moore, A. W. (1990). Acquisition of Dynamic Control Knowledge for a Robotic Manipulator. In *Proceedings of the 7th International Conference on Machine Learning.* Morgan Kaufmann.

Moore, A. W. (1999). Very fast mixture-model-based clustering using multiresolution kd-trees. In Kearns, M., & Cohn, D. (Eds.), *Advances in Neural Information Processing Systems 10.* Morgan Kaufmann.

Moore, A. W., Schneider, J., & Deng, K. (1997). Efficient Locally Weighted Polynomial Regression Predictions. In D. Fisher (Ed.), *Proceedings of the 1997 International Machine Learning Conference.* Morgan Kaufmann.

Moore, A. W., & Lee, M. S. (1998). Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Journal of Artificial Intelligence Research*, *8*.

Omohundro, S. M. (1987). Efficient Algorithms with Neural Network Behaviour. *Journal of Complex Systems*, *1*(2), 273–347.

Omohundro, S. M. (1991). Bumptrees for Efficient Function, Constraint, and Classification Learning. In Lippmann, R. P., Moody, J. E., & Touretzky, D. S. (Eds.), *Advances in Neural Information Processing Systems 3.* Morgan Kaufmann.