Robust Estimation With Sampling and Approximate Pre-Aggregation

Christopher Jermaine
Department of Computer and Information Sciences and Engineering
University of Florida
Gainesville, FL 32611
cjermain@cise.ufl.edu

Abstract

The majority of data reduction techniques for approximate query processing (such as wavelets, histograms, kernels, and so on) are not usually applicable to categorical data. There has been something of a disconnect between research in this area and the reality of database data; much recent research has focused on approximate query processing over ordered or numerical attributes, but arguably the majority of database attributes are categorical: *country, state, job_title, color, sex, department,* and so on.

This paper considers the problem of approximation of aggregate functions over categorical data, or mixed categorical/numerical data. We propose a method based upon random sampling, called *Approximate Pre-Aggregation* (APA). The biggest drawback of sampling for aggregate function estimating is the sensitivity of sampling to attribute value skew, and APA uses several techniques to overcome this sensitivity. The increase in accuracy using APA compared to "plain vanilla" sampling is dramatic. For `SUM` and `AVG` queries, the relative error for random sampling alone is more than 700% greater than for sampling with APA. Even if stratified sampling techniques are used, the error is still between 28% and 175% greater than for APA.

1 Introduction

Approximate query answering has received much recent research attention from the data management community. The importance of approximation as an area of database research has been greatly enhanced by the emergence of Decision Support Systems (DSS) as a fundamental application area for database technology. Two characteristics specific to such systems make DSS a potential area where approximation methods can be used.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

- First, DSS archive many gigabytes or terabytes of data (often on the order of billions of records), and so exact query answers can be very expensive.
- Second, queries over DSS are usually aggregate queries, and thus lend themselves to numerical approximation.

A great amount of the work in this area has centered around the approximate evaluation of aggregate queries of the form:

```
SELECT AGG_FUNC (TBL.MEAS_ATT)
FROM THE_TABLE TBL
WHERE TBL.SEL_ATT_1
 BETWEEN(N1_LOW AND N1_HI) AND TBL.SEL_ATT_2
 BETWEEN(N2_LOW AND N2_HI) AND ...
```

In most work, it has been assumed that the data model that is queried over is something similar to a multidimensional data cube. In this model, there are *D selection* attributes, any of which can appear in the `WHERE` clause of the query. These attributes partition the data space into a *D*-dimensional cube. In addition to the selection attributes, there is a set of *measure* attributes to which different aggregate functions may be applied in the `SELECT` clause of the query. Each tuple in the database is mapped to a cell in the data cube based on the values of that tuple's selection attributes, and the values of that tuple's different measure attributes are written into the cell. Queries then compute aggregate functions (such as `SUM`, `AVG`, `COUNT`, etc.) over different low-dimensional projections of the measure attributes which are stored in the cube.

A key assumption in much of the work in this area is that the attributes appearing in the `WHERE` clause either are numerical or have been discretized into a set of buckets which have a meaningful ordering. In this case, one can choose from many proposed approximation methodologies. Examples are multidimensional histograms [18], wavelets [20], and kernel estimators [13], among others.

1.1 The Problem: Categorical Attributes

One problem with these approximation techniques is that data often have attributes which are truly categorical. An example of this is the attribute *country* (with possible values *United States*, *Canada*, *France*, etc.). These values are not numerical, and it is difficult to impose an ordering on them that allows for a meaningful decomposition of the data

space (in our example, should Canada be grouped with the United States, or with France?) Such attributes are problematic for any of the numerical or ordinal approximation methods mentioned above, since without an ordering, each possible category must be considered as a separate boolean attribute. The problem is worse if there are several categorical attributes like *seller_country*, *buyer_country*, and *product_class*. In this case, we may have increased our data dimensionality from three to 500 attributes, which is likely far more than any of the mentioned approximation methods can handle. According to Gunopulos et al., it is doubtful that any of these methods can consistently work in more than ten dimensions [13].

1.2 Random Sampling

Thus, handling estimation over categorical attributes (or, most commonly, mixtures of categorical and numerical attributes) is important. Arguably, the only significant subset of current estimators that are suitable for use with both numerical and categorical data is the subset based upon *random sampling*, which handles categorical attributes gracefully since it is unaffected by data dimensionality (as long as the number of sampled tuples remains constant). Sampling is also attractive because it can be done in a single pass over the data, using standard algorithms [19]. However, random sampling has one tremendous drawback. The accuracy of a sample-based estimate for an aggregate function decreases as the variance of the set of values over which the aggregate is computed increases. Combined with the prevalence of attributes in the real world that follow a Zipfian or power distribution, this can make random sampling inaccurate.

The standard statistical method for handling the variance problem is to use *unequal probability* or *stratified* sampling [6]. In stratified sampling, the probability of sampling a record is proportional to the importance of the record. For example, imagine that we want to use a sample of the people who live in the city Metropolis in order to estimate the total net worth of all individuals who live in that city. In stratified sampling, we would make sure that our sample contains more of the richest people in Metropolis, because they are likely to contribute the most to the total net worth.

Still, there are several drawbacks of stratified sampling.

- First, because a stratified sample is targeted towards answering a particular query (or class of queries) we should know something about the workload (at least the attribute(s) that are usually queried, and/or the usual aggregate function(s)). This can be problematic in DSS, where queries are often exploratory in nature, and so past queries might be a poor indicator of the future workload.
- Second, if a stratified sample has been targeted towards answering a particular query, then there will always exist other queries that will suffer as a result. In our Metropolis example, if we use the stratified sample aimed at estimating the total net worth in order to estimate the average net worth of all people younger than 25, we will likely get a very inaccurate result. Why? By skewing our original sample towards wealthy people (who usually tend to be older) we likely have less information about people under 25. Of course, we can always target our sample towards *both* queries, or more generally, a query workload [4][5][9]. However, there will always be some other queries that suffer information loss.
- Third, it is unclear how many types of stratified samples appropriate for answering simple aggregate queries can be guaranteed in a single pass over the data, without any prior knowledge of the data or query distribution.

Given these drawbacks, the question that we address in this paper is as follows:

> *Can we alleviate the effect of variance on the accuracy of random sampling, to produce a method that is suitable for estimation of aggregate functions over categorical or mixed categorical/numerical data in DSS? Furthermore, can this be done in such a way that (1) we still only require one pass through the data; (2) we increase the accuracy of the estimation for* all *aggregate queries; and (3) this is done without prior knowledge of the query workload?*

1.3 Approximate Pre-Aggregation

Our method (called *APA* or *Approximate Pre-Aggregation*) is quite different from stratified sampling. APA uses a true random sample of the data that is not biased towards any query or workload. This sample is combined with a small set of statistics about the data. The statistics can be gathered in one pass, at the same time that the sampling is performed. The statistics can be arbitrarily complex, but a few bytes of information can dramatically increase estimation accuracy.

We now detail a high-level example of APA. Imagine that we have three professors who teach the database course, though not every professor teaches the course every semester. Each semester, a number of students who take the DB class come to see their professor with a concern or a complaint about the grading on a test. This information is recorded over three years as 12 tuples in a database table (shown in Table 1). We sample 50% of the table. The sampled tuples are indicated as unshaded cells in Table 1.

Table 1: Number of complaints over three years.

Prof	Semster	Cmplaints	Prof	Semster	Cmplaints
Adams	Fa 02	3	Smith	Su 01	7
Jones	Fa 02	2	Smith	Sp 01	8
Adams	Sp 02	9	Adams	Fa 00	4
Jones	Sp 02	2	Smith	Fa 00	33
Smith	Sp 02	21	Smith	Su 00	16
Smith	Fa 01	36	Adams	Su 00	3
Jones	Su 01	1	Jones	Su 00	0
Adams	Su 01	2	Jones	Sp 99	1

We then use the sample to answer the following SQL query:

```
SELECT SUM (COMPLAINT)
FROM THE_TABLE
WHERE PROF = 'Smith'
```

The total number of students in the sample who complained to Professor Smith was 36, and the sample constitutes 50%

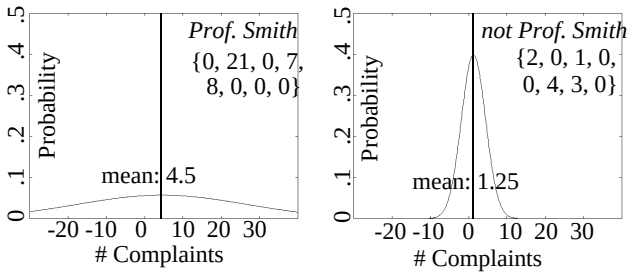

Figure 1: Pdf fitted to the two quadrants of the data space.

of the database tuples. Thus, we estimate that a total of 72 students have come to see Professor Smith with complaints.

As we see from Table 1, the *actual* number of students who came to see Professor Smith was 121 (yielding 40.5% relative error). The problem is the variance in the number of students who complained to Professor Smith each semester. This ranges from a low of 7 to a high of 36, and it happens that our sample missed the two semesters when the greatest number of students complained to Professor Smith.

In APA, we will address sampling's vulnerability to variance by using a bit of additional information about our database. In our example, imagine that we also know the following simple fact:

```
(Sum (COMPLAINTS)) = 148
```

We can then use this information to serve as something of a sanity check on our estimation that will allow us to "undo" the effect of variance on our sample. First, we use the relational selection predicate of the query that we are trying to answer to divide the data space into 2^n quadrants, where n is the number of clauses in the predicate (in our example, there is one such clause: `PROF = 'SMITH'`, and so our data space is divided into two quadrants; one corresponding to `PROF = 'SMITH'` and another corresponding to `PROF` $\neq$ `'SMITH'`). Next, we use the central limit theorem in conjunction with our sample to estimate a probability density function (*pdf*) for each of the 2^n quadrants created by the predicate. These pdfs are shown above in Figure 1.

The pdfs pictured above describe the probability distribution for the expected "real" average value for each of our two quadrants, given the values that we observed in our sample. These pdfs are fairly intuitive. Since there is significant variance in the number of complaints associated with Professor Smith in our sample, we expect to be relatively unsure about the accuracy of our estimation for the number of complaints per tuple for Professor Smith.

The next step is to use our additional information to create a set of constraints on the distributions. In our example, we know that `(SUM (COMPLAINTS)) = 148`. Since the total number of database tuples is 16, this implies that the means of the two distributions shown above in Figure 1 must sum to 148/16 = 9.25.

However, the means of the two distributions do not sum to 9.25 (they sum to 5.75). Thus, there is significant error somewhere, and so the next step is to find the most "likely" way to satisfy all of our constraints and remove the error, given the pdfs that we have derived. To do this, we rely on a method from mathematics and statistics called *Maximum Likelihood Estimation* (MLE). MLE is a general methodology for fitting a set of hidden parameters that explain a set of observed outcomes. In our example, the MLE will have much more freedom to adjust the mean of the distribution associated with Professor Smith, because of the greater variance in the pdf associated with Professor Smith.

After performing the MLE, APA obtains a "re-estimated mean" of 8.52 for Professor Smith, which gives us an estimated total of 136.3 for the total number of students who have complained to Professor Smith (since $8.52 \times 16 = 136.3$). Since the actual number of complaints was 121, we have significantly improved our initial estimate of 72. This improvement is typical for APA.

In this paper, we will subsequently refer to facts of the form `(Sum (COMPLAINTS)) = 148` as "zero-dimensional facts," but in general APA can handle facts of arbitrary dimensionality. An example one-dimensional fact is `(SUM (COMPLAINTS) WHERE PROF = 'JONES') = 6`; and a two-dimensional fact is `(SUM (COMPLAINTS) WHERE PROF = 'JONES' AND SEMESTER = 'SU02') = 0`. As our experiments will show, storing all zero-dimensional facts for a database will require only a few bytes, and can lead to a tremendous increase in accuracy. Storing additional facts can increase accuracy accordingly.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we give a brief description of MLE. In order to phrase APA as a MLE problem, three specific components of the MLE formulation of APA must be defined. These components are described in Section 3. Our method for solving the MLE is described in Section 4. Section 5 details a set of experiments aimed at benchmarking performance of the approximation. Section 6 describes some additional considerations, such as using the method with numerical data. We describe related work in Section 7, and conclude the paper in Section 8.

2 Maximum Likelihood Estimation

We begin with a brief review of the concept of *maximum likelihood estimation*, a methodology from statistics that is of fundamental importance to APA. MLE is a very general (and very powerful) technique used for the discovery of statistical models.

Let x be an observable outcome from a given experiment. In our case, x might be the fact that our random sample predicts that the value of our relational aggregate `SUM` query is 72. Then, let the *probability density function* (*pdf*) with respect to observing outcome x be the function

$$f(x;\theta_1, \theta_2, ..., \theta_k)$$

In this formulation, the parameters $\theta_1, \theta_2, ..., \theta_k$ are the *hidden parameters* that we wish to estimate (they describe the model that we want to discover).

Say that instead of having the answer to just a single experiment, we conduct a set of n different experiments, each of which has outcomes $x_1, x_2, ..., x_n$. In order to "fit" the hidden model parameters to this set of observations, we

maximize the likelihood that our particular model produced the data. This likelihood is given by the function

$$L(x_1, x_2, ..., x_n;\theta_1, \theta_2, ..., \theta_k) = \prod_{i=1}^{n} f(x_i;\theta_1, \theta_2, ..., \theta_k)$$

In order to come up with a set of model parameters to explain the observations, the likelihood function is maximized with respect to all possible values for the parameters $\theta_1, \theta_2, ..., \theta_k$. Since it is often difficult to work with such an unwieldy product as L, the value which is typically maximized is the *loglikelihood*

$$\Lambda = \sum_{i=1}^{n} \log f(x_i;\theta_1, \theta_2, ..., \theta_k)$$

The loglikelihood Λ is the result of applying a logarithmic transformation to L, which renders a simpler expression and does not affect the ordering of the quality of the model parameters.

3 Maximum Likelihood Estimation in APA

The basic idea of APA is simply to find the best (or most likely) explanation for the sample which does not violate any of the known facts about the database, where the facts are things like the counts for individual, categorical values. Thus, we need to come up with a way to pose our problem of approximate aggregation over categorical data as a MLE problem. In this Section, we describe how this is accomplished. As stated previously, there are three specific components of maximum likelihood that we need to describe in the context of APA:

(1) The experimental outcomes $x_1, x_2, ..., x_n$
(2) The model parameters $\theta_1, \theta_2, ..., \theta_k$
(3) The pdf $f(x;\theta_1, \theta_2, ..., \theta_k)$

Once these three components have been defined, we have transformed the problem of estimation of aggregate functions over categorical data into a MLE problem, and we can begin the task of developing a method to solve the problem. The next three subsections define each component for APA.

3.1 Experimental Outcomes

First, we describe how we obtain the "outcomes" $x_1, x_2, ..., x_n$ to postulate APA as a MLE problem. In APA, those outcomes are a set of predictions made by our sample. Thus, we are trying to fit or explain our sample in the context of our model. We begin with an example aggregate query:

```
SELECT SUM (SALARY)
FROM EMPLOYEE
WHERE SEX='M' AND DEPARTMENT=
'ACCOUNTING' AND JOB_TYPE='SUPERVISOR'
```

With a query of this type, we will begin by numbering each of the boolean clauses in the relational selection predicate from 1 to m. In our case, b_1 = (SEX = 'MALE'), b_2 = (DEPARTMENT = 'ACCOUNTING') and b_3 = (JOB_TYPE = 'SUPERVISOR'). We will also consider the negation of each of these clauses: $\overline{b_1}$, $\overline{b_2}$, and $\overline{b_3}$.

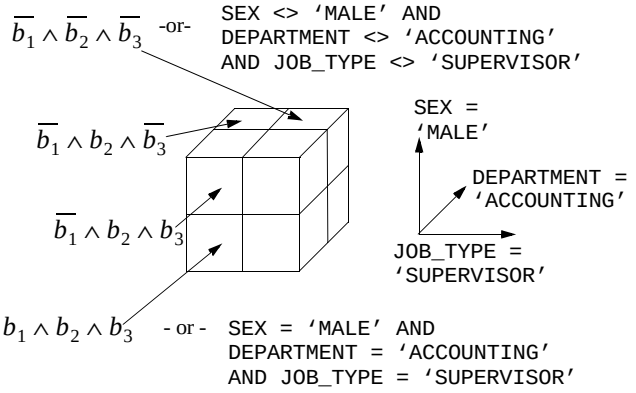

Figure 2: Spatial representation of query predicates.

Given this, let $\overrightarrow{2^m}$ be the list of all "meaningful" conjunctions of these boolean clauses (meaningful in the sense that they do not contain a conjunction of a given predicate with that predicate's negation). In our example, $\overrightarrow{2^m}$ = [$b_1 \wedge b_2 \wedge b_3$, $b_1 \wedge \overline{b_2} \wedge b_3$, $b_1 \wedge b_2 \wedge b_3$, $b_1 \wedge \overline{b_2} \wedge \overline{b_3}$, $\overline{b_1} \wedge b_2 \wedge b_3$, $\overline{b_1} \wedge \overline{b_2} \wedge b_3$, $\overline{b_1} \wedge b_2 \wedge \overline{b_3}$,$\overline{b_1} \wedge \overline{b_2} \wedge \overline{b_3}$]. Note that each of the boolean conditions in this list corresponds to a single cell in the multidimensional data cube defined by b_1, b_2, and b_3 (see Figure 2).

Given this, we can now easily define our set of "experimental" outcomes $x_1, x_2, ..., x_{2^m}$ for the MLE.

Definition 1: The **outcomes** $x_1, x_2, ..., x_{2^m}$ for the MLE in APA are the results of the aggregate function in question with respect to each of the relational selection predicates in $\overrightarrow{2^m}$, estimated using our sample.

For example:

Example 1: We know that our sample-based estimate for SUM(SALARY) over $b_1 \wedge b_2 \wedge b_3$ is \$1.5M (that is, the total salary for male supervisors in the accounting department is \$1.5M). $b_1 \wedge b_2 \wedge b_3$ is the first entry in $\overrightarrow{2^m}$. Thus, \$1.5M is used as the value of x_1.

Example 2: Imagine that the sample-based estimate for SUM(SALARY) over $b_1 \wedge \overline{b_2} \wedge \overline{b_3}$ is \$1.1M (that is, the total salary for male non-supervisors outside of the accounting department is \$1.1M). This is the fourth entry in $\overrightarrow{2^m}$. Thus, x_4 is \$1.1M.

In this way, our random sample is used to estimate the value for each cell in the cube, and these values become the outcomes $x_1, x_2, ..., x_{2^m}$.

3.2 Model Parameters

The second part of the MLE problem that we need to describe is the set of model parameters $\theta_1, \theta_2, ..., \theta_k$ which we will attempt to estimate.

In APA, these parameters are defined to be the APA guess as to the *real* value of the aggregate function in ques-

tion, with respect to each of the cells in the multidimensional data cube defined by b_1, b_2, and b_3. More exactly:

Definition 2: If x_i is the value of cell i predicted by the sample, then the **model parameter** θ_i is the APA maximum likelihood estimate for the correct value for the aggregate function applied to cell i.

Thus, after MLE, the final, approximate answer to the aggregate query will be the value of the parameter θ_1, which corresponds to the predicate $b_1 \wedge b_2 \wedge b_3$.

Note that in the case of APA, we will also effectively have an additional set of *constraints* on the possible values of $\theta_1, \theta_2, ..., \theta_k$. Recall that the basic idea of APA is to find the most likely solution given the results of our sampling as well as a simple set of known facts about the data, like counts of individual attributes. Two examples of these types of facts are:

```
(SUM (SALARY) WHERE SEX='FEMALE' AND
 DEPARTMENT!='ACCOUNTING' AND
 JOB_TYPE!='SUPERVISOR') = $0.4M

(SUM (SALARY) WHERE JOB_TYPE!='SUPERVI-
 SOR') = $2.3M
```

These facts are then easily translated into constraints on $\theta_1, \theta_2, ..., \theta_k$:

Example 3: The relational selection predicate present in the first fact is $b_1 \wedge \overline{b_2} \wedge \overline{b_3}$. This is the 4th predicate in $\overrightarrow{2^m}$. Thus, the first fact is equivalent to the constraint that θ_4 = \$0.4M.

Example 4: The relational selection predicate present in the second fact is $(b_1 \wedge b_2 \wedge \overline{b_3}) \vee (b_1 \wedge \overline{b_2} \wedge \overline{b_3}) \vee (\overline{b_1} \wedge b_2 \wedge \overline{b_3}) \vee (\overline{b_1} \wedge \overline{b_2} \wedge \overline{b_3})$. This is a disjunction of the third, fourth, seventh, and eighth predicates present in $\overrightarrow{2^m}$. Thus, this fact is equivalent to the constraint that $\theta_3 + \theta_4 + \theta_7 + \theta_8$ = \$2.3M.

3.3 The Probability Density Function *f*

The third and final aspect of the MLE that we need to define is the probability density function *f*, which will give us the likelihood that we would see the experimental observations $x_1, x_2, ..., x_{2^m}$, given model parameters $\theta_1, \theta_2, ..., \theta_{2^m}$. Specifically, we need to derive a pdf *f* where $\int_a^b f(x_i;\theta_i)dx_i$ is the probability that a sample would predict that the value for x_i falls in the range *a* to *b*, given that the *real* value for x_i is θ_i. Intuitively, our function will reflect the fact that a value for θ_i which diverges significantly from the value predicted by the sample is a much poorer explanation for the sample than a value for θ_i that is close to the prediction.

To derive this function, we begin with the following set of formulas. Much of our notation and inspiration is borrowed from Hellerstein, Haas and Wang [15]. Assume we attempt to estimate the value for an aggregate of the form

```
SELECT AGG(expression)
FROM THE_TABLE
WHERE (predicate)
```

Assume we have pulled a sample of size *n* from a database of size *db*, and the estimated value of the query based on the sample is *x*. Let t_i denote the *i*th tuple in the sample. Let $v(t_i)$ be the value of AGG(*expression*) predicted by the single tuple t_i. For example,

- if AGG is SUM, then $v(t_i) = t_i.expression \times db$ if $predicate(t_i)$ = true, and 0 otherwise
- if AGG is COUNT, then $v(t_i) = db$ if $predicate(t_i)$ = true and 0 otherwise

Let $T(v) = (n - 1)^{-1} \sum_{i=1}^{n} (v(t_i) - x)^2$. From Hellerstein, Haas and Wang [15], and using the central limit theorem we then have

$$P(|\theta - x| \le \varepsilon) \approx 2\Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{T(v)}}\right) - 1$$

where Φ is the cumulative standard normal distribution. Since this probability relies on the central limit theorem, it is expected to be a good estimate if the *large sample assumption* holds, as it typically will in the type of approximate query answering application that we consider.

However, this probability function is not quite the pdf $f(x;\theta)$ that we require. If $|x - \theta| = \varepsilon$, we need the *density* of the distribution at ε, and not the definite integral of the function evaluated over all values from $-\varepsilon$ to ε, which is what $P(|\theta - x| \le \varepsilon)$ gives us.

Thus, what we really want is

$$f(x;\theta) = \lim_{\varepsilon' \to 0} \frac{P(|\theta - x| \le \varepsilon) - P(|\theta - x| \le \varepsilon + \varepsilon')}{2\varepsilon'} \text{ if } |x - \theta| = \varepsilon$$

Substituting, we have

$$f(x;\theta) \approx \lim_{\varepsilon' \to 0} \frac{\Phi\left(\frac{\varepsilon\sqrt{n}}{\sqrt{T(v)}}\right) - \Phi\left(\frac{(\varepsilon - \varepsilon')\sqrt{n}}{\sqrt{T(v)}}\right)}{\varepsilon'}$$

Since Φ is the cumulative standard normal distribution, the above expression simply becomes $f(x;\theta) \approx g\left(\frac{\varepsilon\sqrt{n}}{\sqrt{T(v)}}\right)$ where *g* is the probability density function for the standard normal distribution. More specifically,

Definition 3: The **probability density function** *f* is defined as follows:

$$f(x;\theta) \approx \frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(\frac{\varepsilon\sqrt{n}}{\sqrt{T(v)}}\right)^2\right) = \frac{1}{2\pi}\exp\left(-\frac{\varepsilon^2 n}{2T(v)}\right)$$

4 Quadratic Programming in APA

We have now described APA as a MLE problem. We next address how we might actually perform the estimation.

There are many ways to attempt to estimate the solution to a maximum likelihood problem. Estimation or approximation is usually necessary because of the inherent intracta-

bility of discovering the most likely model in the general case. The best-known algorithm for such approximation is the *Expectation Maximization* (EM) algorithm [7]. EM begins with an initial guess at a solution and then repeatedly refines the guess until it reaches a locally optimal solution.

However, there are several reasons that we will not use EM to solve our problem. One is that in APA, the search space is very constrained by a set of linear equations that govern the relationship among the model parameters $\theta_1, \theta_2, ..., \theta_{2^m}$ (see Section 3.2, Examples 1 and 2). In most formulations, EM simply seeks to maximize the loglikelihood value Λ (see Section 2). Maximizing Λ in APA without constraints on $\theta_1, \theta_2, ..., \theta_{2^m}$ will be of little use (it will give an answer that is exactly equivalent to the estimate obtained by the sample!). The reason that APA can improve the accuracy of sampling is that it provides a framework to reconcile the error of the sample with a set of known facts.

4.1 Quadratic Programming

As a result of this, we will instead solve the MLE component of APA using a *quadratic programming formulation*. Solving certain constrained MLE problems using quadratic programming is a widely-accepted technique in statistics [3][14][17].

APA can be formulated quite nicely as a quadratic programming problem. Quadratic programming is an extension of linear programming, with the generalization that the objective function to maximize may contain products of two variables, and not simply scalars. Formally, any quadratic programming problem can be stated as:

$$\begin{aligned} \text{maximize} \quad & \theta^T C\theta + d^T\theta \\ \text{subject to} \quad & A\theta = b \\ \text{where} \quad & \theta > l \\ \text{and} \quad & \theta < h \end{aligned}$$

In this formulation, C is a lower triangular matrix, A is a matrix, and θ, b, l, h, and d are vectors. Each of the elements except for θ is given as input into the quadratic programming algorithm; the algorithm then solves for an optimal value of θ so as to maximize the value of the objective function characterized by C and d (which may or may not be possible depending on various characteristics of the input). A concrete example of such a problem would be:

$$\begin{aligned} \text{maximize} \quad & x^2 + y - xz \\ \text{subject to} \quad & x + y = 3,\ x - z = 6 \\ \text{where} \quad & x > 4,\ y > -\infty,\ z > 2 \\ \text{and} \quad & x < 6,\ y < 4,\ z < \infty \end{aligned}$$

A key advantage of using methods like quadratic programming is that a tremendous amount of effort has gone into developing algorithms that work efficiently to solve problems which can be specified within the quadratic programming framework. While solving quadratic programming in general is NP-hard, many algorithms (such as those in the interior point family) work so well, so quickly, and so often that for practical purposes, the problem is often tractable. If we use quadratic programming to perform MLE in APA, we need only find a way to specify the estimation in APA as a quadratic programming problem. In the remainder of this Section, we will focus on the three components of the quadratic programming formulation of APA: the objective function ($\theta^T C\theta + d^T\theta$), the constraints ($A\theta = b$), and the bounds ($l < \theta < h$).

4.2 The Objective Function

The first requirement of the quadratic programming formulation of APA is the development of an objective function that is quadratic with respect to the MLE parameters $\theta_1, \theta_2, ..., \theta_{2^m}$. Fortunately, this will be easy to do in our case. Recall from Section 2 that MLE seeks to maximize the value of the function

$$\Lambda = \sum_{i=1}^{n} \log f(x_i; \theta_1, \theta_2, ..., \theta_k)$$

In the case of APA, this translates to maximizing

$$\Lambda = -\sum_{i=1}^{2^m} \log \frac{1}{2\pi} \exp\left(-\frac{\varepsilon_i^2 n}{2T_i(v)}\right)$$

Since the *log* and the *exp* functions cancel one another out, this expression simplifies greatly. After simplifying and removing multiplicative coefficients and constraints common to each term in the summation (which will not affect the maximization), we have

$$\Lambda = -\sum_{i=1}^{2^m} \frac{\varepsilon_i^2}{T_i(v)}$$

Maximizing this very simple sum is clearly possible within the quadratic programming framework. Since $\varepsilon_i = (\theta_i - x_i)$ and $T_i(v)$ is not dependent on θ_i, we have an expression that is quadratic with respect to each θ_i. Thus, the loglikelihood Λ is readily used as an objective function as part of our quadratic programming formulation of APA.

4.3 Constraints

Next, we need to formulate our constraints on the possible values of $\theta_1, \theta_2, ..., \theta_{2^m}$. The ability of quadratic programming to incorporate constraints makes it ideal for use with APA, since the search space in APA is highly constrained by the set of known facts about the database.

The constraints in the quadratic programming formulation of APA are a straightforward application of the general technique from the examples of Section 3.2. Recall that each target value θ_i is associated with the value of the aggregate function in question, applied to the tuples associated with the boolean, relational selection predicate $\overrightarrow{2^m}[i]$. Each such predicate corresponds to a quadrant in the multi-dimensional data cube created by the boolean predicates present in the aggregate query that we are estimating. The constraints for our quadratic programming formulation are then simply linear sums of the values $\theta_1, \theta_2, ..., \theta_{2^B}$. Each such sum corresponds to a known fact about the database. See Section 3.2 for more detail.

4.4 Bounds

The final components of the quadratic programming formulation of APA that must be defined are the vectors that specify the lower and upper bounds for the estimate. Bounds in quadratic programming perform two important and separate functions:

- First, they specify which answers are unacceptable. In the case of APA, for example, it would be incorrect to estimate that the value of `COUNT(*)` is negative.
- Second, they provide for a quick and accurate solution to the problem by constraining the search space. Since quadratic programming in general is NP-hard, coming up with tight initial bounds that reduce the search space can help lead to an optimal solution.

To produce the bounds for our quadratic programming formulation of APA, we return to the fact (from Section 3.3) that the likelihood function for an estimate θ_i for cell i is based on the normal probability density function g. As is stated in nearly every introductory statistics textbook, 99.7% of the total mass of the normal probability density function is found within three standard deviations of the origin. Because of this, any estimate for θ_i which falls outside of this range is quite unlikely, and it is reasonable to remove all such solutions from consideration by choosing appropriate bounds. We can use this fact to develop reasonable bounds for our estimate of θ_i. From Section 3.3,we have

$$f(x_i;\theta_i) = \frac{1}{2\pi}\exp\left(-\frac{(x_i-\theta_i)^2 n}{2T_i(v)}\right)$$

If we wish to limit our search space to a solution set that comprises 99.7% of the total likelihood of the search space, we simply set the value of this expression to three and solve for the two possible values of θ_i. Thus, our upper and lower bounds on θ_i will be the two solutions to

$$\frac{1}{2\pi}\exp\left(-\frac{(x_i-\theta_i)^2 n}{2T_i(v)}\right) = 3$$

More specifically, after solving for θ

$$l_i = x_i - \sqrt{2T_i(v)\times n^{-1}\times \ln 6\pi} \quad \text{and}$$

$$h_i = x_i + \sqrt{2T_i(v)\times n^{-1}\times \ln 6\pi}$$

5 Experiments

In this Section, we describe a set of qualitative performance results that give a good idea of the kind of accuracy that one can expect when applying APA to the problem of estimating `COUNT`, `SUM`, and `AVG` values over high-dimensional data. We will compare several different flavors of APA against uniform and stratified random sampling, as well as the multidimensional wavelet transform [20]. In this Section, we only give our results over categorical data, because two of the variations on APA that we test cannot be used for numerical or mixed data (see Section 6.1). However, as we discuss in Section 6.1, the variations on APA that can be used with numerical data have no appreciable difference in accuracy on such data, so the results presented here give a good picture of the accuracy of the method.

5.1 Test Data

Our experiments are conducted over eight real, high-dimensional data sets. The eight data sets which we consider are: US Census data, Forest Cover data (from the UCI KDD archive), Water Quality data, River Flow data, Stock Market data, William Shakespeare data (word proximity information), Image Feature Vector data, and Web Page data (keyword frequency information).

Since many of the data attributes in these data sets are not categorical, we transform non-categorical data into categorical data in the following way in order to have a varied testbed. One-half of the numerical data attributes are partitioned into five, equi-width buckets, and those buckets are used as categories. The other half of the data attributes are partitioned into five categories of exponentially decreasing depth (where the depth is the number of tuples that each bucket contains). In this case, the first category has 1/2 of the database tuples, the next has 1/4, the next 1/8, the next 1/16, and the final category has 1/16 as well.

For each data set, we arbitrarily pick (50 + 3) different attributes from the data set in question for use in our experiments (or fewer if the data set is of dimensionality less than 53). The first 50 attributes are transformed into categories as described above and are designated as *selection attributes*. Since each of the 50 query attributes is divided into five categories, there are 250 categories in all, 50 of which are present in each database tuple. Each tuple falls in one cell in a data cube having 5^{50} or 8.8×10^{34} cells in total.

The last three attributes chosen are designated as *measure* attributes. Measure attributes are the target attributes on which aggregate functions are applied during our experiments, whereas query attributes appear only in the relational selection predicate for our test queries. While query attributes are transformed into categories as described above, measure attributes are left in numerical form.

5.2 Test Queries

To test the different methods, we construct a large number of SQL-style aggregate queries over the data sets.

5.2.1 The `WHERE` Clause

For each data set, the `WHERE` clause of each of the test queries is generated as follows. Each query has between one and eight random predicates associated with its relational selection operator. These predicates are formed into a conjunction with a series of boolean `AND` operators. An example of such a three-predicate query is:

```
SELECT SUM (MEASURE_1)
FROM RIVER_FLOW
WHERE CATEGORY_6 = TRUE AND
CATEGORY_48 = FALSE AND
CATEGORY_156 = TRUE
```

The queries are generated so that they have varying expected selectivities, where the expected selectivities are calculated using an attribute value independence assump-

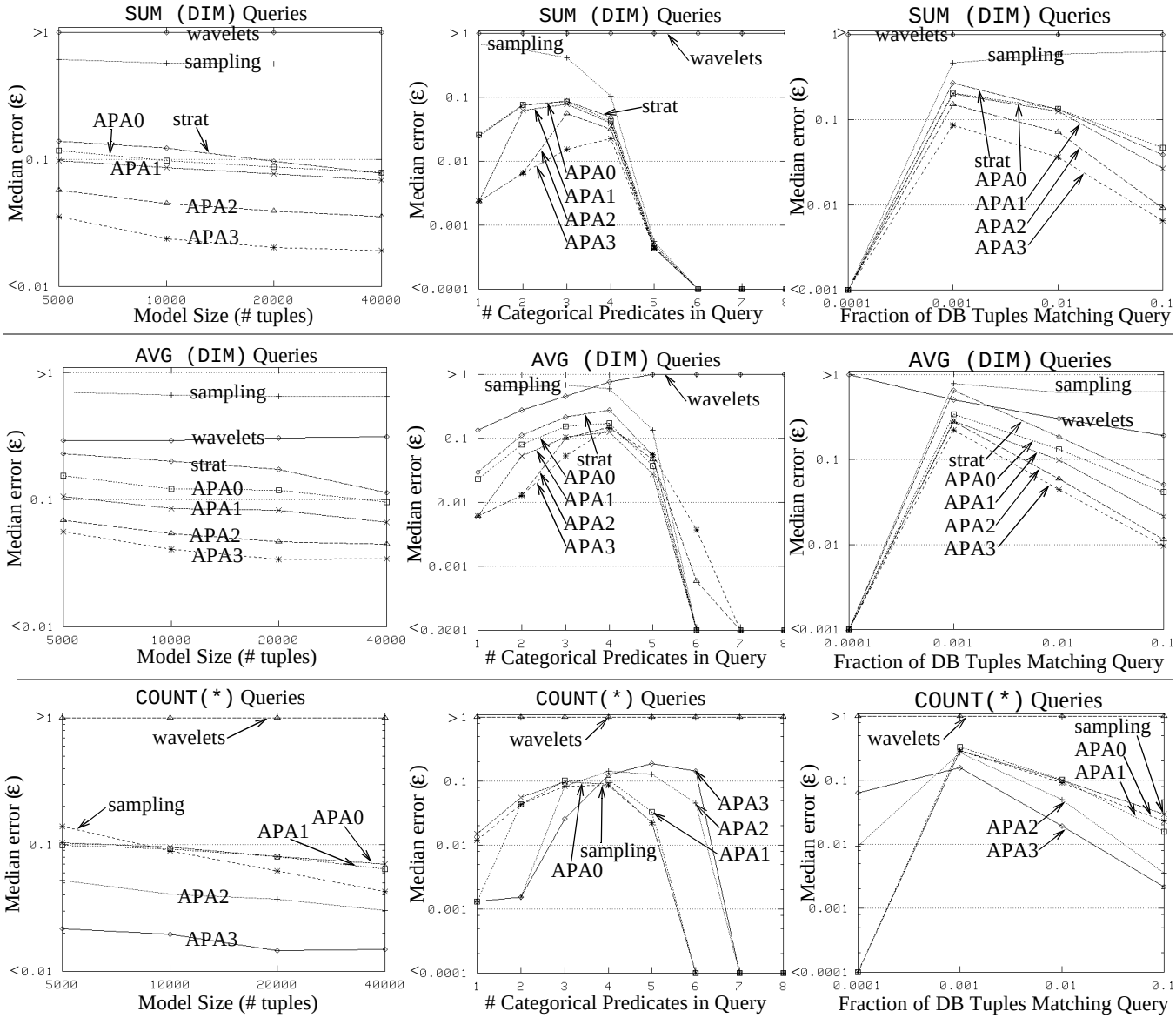

Figure 3: Median error for experiments.

tion. Queries are generated to match roughly between 0.01% to 10% of the database tuples. The joint distribution of query selectivities and number of categorical predicates are shown below.

Table 2: Distribution of test queries.

exp. %DB match query	1 pred	2 pred	3 pred	4 pred	5 pred	6 pred	7 pred	8 pred
10%	4%	4%	4%	4%	4%	4%		
1%		4%	4%	4%	4%	4%	4%	
0.1%		4%	4%	4%	4%	4%	4%	4%
0.01%			4%	4%	4%	4%	4%	4%

5.2.2 The SELECT Clause

Recall that for each data set, three attributes are chosen as the different measure attributes. The measure attributes in our queries have aggregate functions (either SUM or AVG) applied to them by the SELECT clause. Thus the queries themselves return results computed over the measure attributes.

For each data set, we create 7,000 queries. Each of the queries has a WHERE clause generated as described above in Section 5.2.1. One thousand of the queries have the SUM aggregate function applied to the first measure attribute for its respective data set, 1,000 apply SUM to the second measure attribute, and 1,000 apply SUM to the third measure attribute. Likewise, 1,000 of the queries have the AVG aggregate function applied to the first measure attribute, 1,000 apply AVG to the second, and 1,000 apply AVG to the third. Finally, 1,000 COUNT(*) queries are also generated.

Since we generate 7,000 queries for each of the 8 data sets, 56,000 different queries are generated in all.

5.3 Experiments

5.3.1 Approximation Methods Tested

In our experiments, we test six different approximation options over the eight data sets. The six options are:

- *Random Sampling*. We simply perform random sampling without replacement. The random sample is then used to approximate the answer to the various queries.
- *Stratified Sampling*. In this option, we use 3 different unequal probability samples, as well as a single, uniform random sample. Each of the 3 unequal probability samples is tailored to answering one of the three different `SELECT SUM(ATT)` clauses. We use the approximate Neyman allocation technique [6] (also used by Chaudhuri et al. [5]) when performing our stratified sampling. Since a larger number of strata increases accuracy, we maximize the number of strata h by using $h = N$ for a sample of size N. Thus, each strata receives one sample, and strata are chosen to minimize variance. For `COUNT(*)` queries, the stratified sample is equivalent to the random sample. `AVG(ATT)` queries are treated as `SUM(ATT)/ COUNT(*)` queries, which is why the stratified sample must also make use of a uniform random sample. Thus, we note that stratified sampling actually requires several samples.
- *APA0*. This is an implementation of approximate pre-aggregation. For APA0, we store a random sample identical to that used in the approximation based on random sampling. However, for each measure attribute, we also store all "0-dimensional" facts about our data and use them as our constraints as described in Section 4.3. This means that for each measure attribute, we simply know (and can make use of) the fact that (`SUM(MEASURE_ATT_i)` = val_i). In addition, we know that (`COUNT(*)` = val_{cnt}). Thus, APA0 assumes limited information in addition to the sample which can be stored in a few bytes.
- *APA1*. This is identical to APA0, except that for each measure attribute, we also store and use all "1-dimensional" facts. An example of a one-dimensional fact is:

```
(SUM (SALARY) WHERE SEX = 'F') = $1.9M
```

  In this type of fact, there is a single category named in the `WHERE` clause. As a result, with c total categories present in the data, c facts must be stored for each measure attribute. Since c will probably number in the thousands (or less), storing a few thousand such facts introduces only a very small overhead, and is probably very reasonable.
- *APA2*. This is identical to APA1, except that we also store all 2-dimensional facts for each measure attribute. These are facts with a conjunction of two categorical restrictions present in the `WHERE` clause. For example:

```
(SUM (SALARY) WHERE AND SEX = 'F' AND JOB
  = 'SUPERVISOR') = $0.4M
```

  With c categories, there are (c choose 2) such facts to compute and store. Even with 1,000 different categories, there are still fewer than 500,000 such facts, and they can be stored in a few megabytes in total. Since a reasonable sample may be an order of magnitude larger in size, this is probably still a reasonable overhead for many applications.
- *APA3*. Finally, this is APA2 with the addition of all 3-dimensional facts stored and used. Since there are (c choose 3) such facts for each measure attribute, APA3 is probably only reasonable for use with a relatively small number of categories. Still, we test APA3 in order to provide a more informative benchmark.
- *Wavelets*. The sixth option that we test in our benchmarking is a wavelet approximation (similar to Vitter and Wang's [20]). To be fair, wavelets are not meant for categorical data (which is typically very high-dimensional). Still, the comparison is worthwhile because the performance of the wavelet transform in this very high-dimensional domain is probably indicative of the type of performance that other methods for use with relatively low-dimensional, numerical data will exhibit when used with categorical data.

5.3.2 Experiments Run

For each of the six methods described above, we construct the appropriate data structures for each of the eight data sets in order to answer queries over each of the three measure attributes for each data set. Note that the wavelet transform and stratified sampling actually require significantly more space for the same "sized" model than either sampling or the lower-dimensional versions of APA. This is because, for APA and uniform sampling, the same sample can be used for each measure attribute. However, for the other two methods, different models must be used for each measure attribute that it queried over, and a separate model must be constructed to facilitate `COUNT(*)` query evaluation.

For each of the different approximation methods over each of the data sets, we tested four different model sizes: M = 5,000, 10,000, 20,000, and finally M = 40,000.

For all of these different combinations, we ran each of the queries that were generated as described in Section 5.2. For each query, we compare the model's answer to the correct answer obtained from going back over the data. Our computation of the accuracy of the approximation is based on the *relative error in combination with a sanity bound b* [10]. Given an exact answer S and an approximate answer S', this error is defined as:

$$\varepsilon = \frac{|S - S'|}{max(b, S)}$$

The reason for the sanity bound is that without such a bound, relative error effectively treats "small" queries differently from large ones [10]. For example, if the correct answer is 0.2, and the approximate answer is 0.3, the relative error is 50% (this is the same error obtained when the

correct answer is 2 million and the approximate answer is 3 million, which is a more significant error). Adding a sanity bound forces a minimum value for the denominator in the expression for ε. In each case, the sanity bound is set to be 1/10,000 as large as the largest correct aggregate result returned by that class of queries over that data set.

For a set of approximations, we will report the *median value* for ε. The reason that we report the median value (rather than the mean) is that for the partial sum (`SUM(DIM)`) and average queries, errors can vary widely because of tremendous skew of certain attribute values. This can cause individual errors to be so large as to skew the mean error. The mean can be affected to the extent that it often presents a somewhat misleading picture of the accuracy of the approximation. The median value is less susceptible to a few answers which are extraordinarily inaccurate.

In Figure 3, we give the results of our experiments. For each of the three different query types (`SUM`, `COUNT`, and `AVG`), we plot the effect of model size, number of boolean predicates present in the `WHERE` clause of the query, and the query selectivity on the quality of the answer. For the results shown with respect to model size, we disregard queries whose `WHERE` clause accepted fewer than 0.1% of the database tuples, since these queries tended to skew the results a bit (though those queries are present in the plots showing the effects of selectivity on the accuracy of the different approximations).

Plots of the different results follow. Note that the plots showing the median value for ε are in log scale, to more easily show the qualitative performance characteristics of the different methods.

5.4 Discussion

Overall, our experimental results show that for `SUM` and `AVG` queries, there is a dramatic reduction in error and a significant increase in accuracy with the use of APA compared to random sampling. Even use of APA0 and APA1 causes a tremendous reduction in error, despite the very modest amount of additional statistics required. APA0 requires that only the total aggregate value for each measure attribute be stored. This probably amounts to a few dozen to a few hundred bytes for most applications. This storage is negligible when we consider the fact that it is used to augment a random sample which is itself likely many megabytes in size. Even APA1 still requires that we store only the total aggregate value for each measure attribute with respect to each category. This additional storage space is still on the order of kilobytes in size, which again is probably insignificant when compared to the size of a useful sample.

How significant is the reduction in error associated with APA? A couple of examples considering queries we tested which matched more than 0.1% of the database tuples are:

- For a sample of 40,000 tuples, the median error ε for `SUM` queries over all experiments answered through sampling was 0.565; for APA0 it was 0.0792 and for APA1 it was 0.0690. Thus, the median error for `SUM` queries for sampling is 713% greater than for APA0 and 819% greater than for APA1.
- For a sample of 40,000 tuples, the median error for `AVG` queries answered through sampling was 0.651; for APA0 it was 0.096 and for APA1 it was 0.0668. Again, the median error for `AVG` queries answered using a sample is 678% greater than for APA0 and 974% greater than for APA1.
- Stratified sampling can also greatly reduce the estimation error compared to uniform random sampling. However, the error reduction is not nearly as dramatic as it is for APA. For a sample of 40,000 tuples, stratified sampling gave a median error of 0.101 for `SUM` queries over all experiments, and a median error of 0.183 for `AVG` queries. This error is 28% greater than APA0 and 46% greater than APA1 for `SUM` queries. For `AVG` queries, the error for stratified sampling is 91% greater than APA0 and 175% greater than APA1.

Furthermore, we point out that stratified sampling (as implemented) requires several different samples, each for use with different classes of queries. For APA and random sampling, only one sample is used. Thus, a "40,000 tuple" sample for stratified sampling can be substantially larger than a "40,000 tuple" sample for APA or random sampling, especially if many attributes are queried over.

Some other observations from our experiments are:

- Wavelets are generally unsuitable for use in this domain. In almost every case, the error associated with the wavelet approximation showed that it was the poorest option. To be fair, wavelets were not meant to be used with categorical data. But the results still show what is probably a general inapplicability to categorical data of methods that are meant for relatively low-dimensional, numerical data.
- One case where APA0 and APA1 failed to outperform random sampling was for `COUNT(*)` queries. In a sense, this is an expected result. We can predict analytically that the error of random sampling is proportional to the variance of the values that are sampled. Since the values sampled by the `COUNT(*)` aggregate function are all either 0 or 1, a large sample will have a small error. Sampling provides very accurate estimation for `COUNT(*)` queries with a large enough sample size. However, use of APA for `SUM` and `AVG` queries does not preclude the use of plain vanilla sampling for `COUNT(*)` queries, since the same sample can be used both for APA and for sampling.
- For similar reasons, the more selective the query, the better random sampling compares with APA, even for `SUM` and `AVG` queries. Why? As fewer and fewer values match the relational selection predicate, the variance of the sampled values tends towards 0, and sampling becomes more accurate. However, sampling does not compare favorably to APA for `SUM` and `AVG` queries until 0.01% or fewer of the tuples from the database match the relational selection predicate. Since selectivity this great is probably rare in an aggregate query, APA is still greatly superior to random sampling.
- A final observation is that our results indicate that the additional accuracy of APA2 and APA3 would probably not warrant their use. Since APA0 and APA1 already give a median error of well below 10% for `SUM`, `DIM`, and `AVG`

queries with almost no additional overhead, it might be hard to justify the use of APA2 or APA3.

6 Additional Considerations

We now briefly consider some additional issues regarding the use of APA.

6.1 Handling Numerical Data

APA0 and APA1 can easily be extended to handle numerical attributes in the relational selection operator. Imagine that we want to estimate the following using APA1:

```
SELECT SUM (SALARY)
FROM EMPLOYEE
WHERE AGE BETWEEN (30 AND 40) AND
   AND JOB_TYPE = 'SUPERVISOR'
   AND YRS_EMPLOYED BETWEEN (3 AND 5)
```

We would begin by numbering each of the boolean clauses in the relational selection predicate from 1 to *m*, just as in the purely categorical case: b_1 = (`AGE BETWEEN (30 AND 40)`), b_2 = (`JOB_TYPE = 'SUPERVISOR'`) and b_3 = (`YRS_EMPLOYED BETWEEN (3 AND 5)`). Everything in the estimation proceeds normally, until we need to compute the constraints for the MLE. For APA0, the only value that we need to know in order to derive the constraints is the value for `SUM(SALARY)` with respect to the entire database; this does not change for numerical data, and so APA0 is totally unaffected by numerical data.

In APA1, in order to construct the constraints for the MLE in our example, we would need to know the value of `SUM(SALARY)` with respect to each of the boolean clauses b_1, b_2, and b_3. Since b_1 and b_3 are numerical attributes, we would need some sort of histogram on these two attributes in order to use the method. Single-attribute histograms are ubiquitous in relational database systems, and tremendously accurate histograms can quickly and efficiently be constructed using a variety of modern techniques, such as the V-optimal histograms of Jagadish et al. [16]. Thus, we argue that both APA0 and APA1 are practical for numerical or mixed numerical/categorical data. In fact, additional experiments (omitted due to space constraints) have shown no substantial difference in qualitative performance for the methods when used over mixed numerical/quantitative data. For example, for a sample of 40,000 tuples, the median error for `AVG` queries answered through sampling was 0.582; for APA0 it was 0.085 and for APA1 it was 0.079.

On the other hand, APA2 and APA3 are impractical for numerical data. Both APA2 and APA3 would require approximating the joint distributions of numerical and categorical attributes, which is a difficult problem (and storing a large number of multidimensional histograms is prohibitive).

6.2 One-Pass Construction and Maintenance

One advantage of APA is that one-pass construction (and dynamic maintenance) of the statistics required by APA is fairly easy. There exist a wealth of algorithms for dynamically maintaining a sample of a specified size from a database, even if the database is a data stream for which the size is not known beforehand [19]. Thus, APA can be used for one-pass approximation of data streams.

In addition, the summary statistics needed by APA are simple and efficient to maintain dynamically. It is simple to maintain the value of `COUNT` and `SUM` with respect to different categories relating to data insertion and deletion, and if APA is used for numerical data, there exist algorithms for efficiently maintaining one-attribute histograms [12].

6.3 APA Across Foreign Key Joins

As it has been described in this paper, APA is suitable for use in aggregate functions with respect to a relational selection predicate. However, it should easily be possible to extend APA to work across foreign key joins, using a technique for sampling from the results of joins, like the *join synopses* of Acharya et al. [1].

7 Related Work

There has been much recent work in the database community on approximate query processing and selectivity estimation [1][10][13][15][20]. Several papers have dealt specifically with making sampling robust for aggregate estimation [2][4][5][9][21]. Most of these methods rely on stratified sampling of some kind. The work of Wu et al. [21] is an exception, in that they propose uniform sampling from the cumulative distribution function. This approach is useful and well-grounded in statistics, but like wavelets [20], the approach is not meant for categorical attributes.

Of the other sampling-based papers referenced above, all use some variation on stratified sampling. One paper in particular [5] makes use of the Neyman allocation tested in Section 5. Though we have experimentally compared stratified sampling to APA, it is important to point out that stratified sampling is actually quite different from APA. Stratified sampling tailors the sample itself to a particular problem, whereas APA tries to make estimation more accurate, given a particular sample. Interestingly, we see no reason that the general methodology used in APA could not be used along with stratified sampling techniques, or with the concise and counting samples of Gibbons and Matias [11]. In other words, first we could tailor the sample to the problem (as in stratified sampling, or using concise and counting samples), and then use APA-like statistics to increase the sample accuracy further. This is an area for future work.

Maximum likelihood estimation (used by APA) is a standard statistical technique. There are literally thousands of papers from different disciplines on the topic. There is also an extensive literature on methods for solving MLE problems. As discussed in Section 4, the EM algorithm [7] is probably the most widely used. Constrained quadratic programming (used by APA) is also an accepted framework for developing solutions to MLE problems [14][17]; there is general-purpose commercial software for performing such estimation [3]. Many other solution methodologies exist. In our view, the depth of related literature from statistics and other fields is a positive comment on APA; it suggests the underlying principles are sound and well-motivated.

Finally, we mention the work of Faloutsos et al. [8], who considered the problem of recovering data distributions from summary data (where summary information is infor-

mation like the result of aggregate functions such as COUNT and SUM). Since applying aggregate functions to numerical data essentially discretizes the data (similar to building histograms over the data), their work was concerned mostly with "smoothing" the results of aggregate functions to recover the original continuous data distribution. While clearly related in that they are using summary statistics in order to perform estimation, the domain that Faloutsos et al. considered was quite different, and it is not clear how their results could be applied to categorical data or how they could be used to augment sampling as we have done.

8 Conclusions and Future Work

In this paper, we have described Approximate Pre-Aggregation (APA), a framework for using simple summary statistics to greatly increase the accuracy of random sampling for estimation of aggregate queries over categorical or mixed categorical/numerical data. This is important because many previous estimation techniques have largely ignored categorical data. APA is based upon sound, statistical techniques such as maximum likelihood estimation and constrained quadratic programming. It is also suitable for estimation in a streaming environment, since the information used by APA can be collected in a single database scan.

Some work remains to be done with respect to the validation of the ideas presented in the paper. For example, this study has largely sidestepped issues associated with computational efficiency. Performing the actual estimation in APA is relatively efficient (the time required by APA for estimation is typically dominated by the time required to scan the sample in memory), but significant costs can be associated with maintaining the statistics used by APA in the face of data insertion and deletion. Particularly for options like APA2 and APA3, this maintenance might be costly. It would be useful to study under exactly what conditions the extra accuracy obtained by APA2 and APA3 is worthwhile, given these costs. It would also be useful to know what characteristics of queries and data might require the additional accuracy. For example, strong correlations between the selection or functional attributes and the measure attributes, or extremely skewed attribute values might make the additional costs worthwhile.

9 Acknowledgements

Part of this work was completed while the author was a graduate student at the Georgia Institute of Technology. The author wishes to thank his Ph.D. committee members for their support: Ed Omiecinski, Ling Liu, Leo Mark, Sham Navathe, and Renee Miller from the University of Toronto. Special thanks to Jennifer Tomsen and Carlos Ordonez for their help and editorial comments on the paper.

References

[1] S. Acharya, P. Gibbons, V. Poosala, S. Ramaswamy: Join Synopses for Approximate Query Answering. *SIGMOD* 1999: 275-286.

[2] S. Acharya, P. Gibbons, V. Poosala: Congressional Samples for Approximate Answering of Group-By Queries. *SIGMOD* 2000: 487-498.

[3] Aptech Systems, Inc., CML System. http://www.rtime.com/tech/gausslib/s2_cml.html.

[4] S. Chaudhuri, G. Das, M. Datar, R. Motwani, V. Narasayya: Overcoming Limitations of Sampling for Aggregation Queries. *ICDE* 2001: 534-542.

[5] S. Chaudhuri, G. Das, V. Narasayya: A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries. *SIGMOD* 2001: 295-306.

[6] W.G. Cochran: *Sampling Techniques*. John Wiley and Sons, New York, 1977.

[7] A.P. Dempster, N.M. Laird, and D.B. Rubin: Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. of the Royal Stat. Society*, Series B, 39(1): 1-38, 1977.

[8] C. Faloutsos, H. V. Jagadish, N. Sidiropoulos: Recovering Information from Summary Data. *VLDB* 1997: 36-45.

[9] V. Ganti, M.-L. Lee, R. Ramakrishnan: ICICLES: Self-Tuning Samples for Approximate Query Answering. *VLDB* 2000: 176-187.

[10] M.N. Garofalakis, P.B. Gibbons: Wavelet Synopses with Error Guarantees. *SIGMOD* 2002: 476-487.

[11] P.B. Gibbons, Y. Matias: New Sampling-Based Summary Statistics for Improving Approximate Query Answers. *SIGMOD* 1998: 331-342.

[12] P.B. Gibbons, Y. Matias, V. Poosala: Fast Incremental Maintenance of Approximate Histograms. *VLDB* 1997: 466-475.

[13] D. Gunopulos, G. Kollios, V.J. Tsotras, C. Domeniconi: Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes. *SIGMOD* 2000: 463-474.

[14] S.P. Han: A Globally Convergent Method for Nonlinear Programming. *J. of Opt. Theory and App.*, 22:297-309 (1977).

[15] J.M. Hellerstein, P.J. Haas, H. Wang: Online Aggregation. *SIGMOD* 1997: 171-182.

[16] H.V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K.C. Sevcik, T. Suel: Optimal Histograms with Quality Guarantees. *VLDB* 1998: 275-286.

[17] M. Jamshidian, P.M. Bentler: A Modified Newton Method for Constrained Estimation in Covariance Structure Analysis. *Comp. Stat. & Data Analysis*, 15:133-146 (1993).

[18] V. Poosala, Y.E. Ioannidis: Selectivity Estimation Without the Attribute Value Independence Assumption. *VLDB* 1997: 486-495.

[19] J.S. Vitter: Random Sampling with a Reservoir. *TOMS* 11(1): 37-57 (1985).

[20] J.S. Vitter, M. Wang: Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. *SIGMOD* 1999: 193-204.

[21] Y.-L. Wu, D. Agrawal, A.E. Abbadi: Using the Golden Rule of Sampling for Query Estimation. *SIGMOD* 2001: 279-290.