Approximations in Database Systems*

Yannis Ioannidis

Dept. of Informatics and Telecommunications, University of Athens, Hellas (Greece)
yannis@di.uoa.gr
http://www.di.uoa.gr/~yannis

Abstract. The need for approximations of information has become very critical in the recent past. From traditional query optimization to newer functionality like user feedback and knowledge discovery, data management systems require quick delivery of approximate data in order to serve their goals. There are several techniques that have been proposed to solve the problem, each with its own strengths and weaknesses. In this paper, we take a look at some of the most important data approximation problems and attempt to put them in a common framework and identify their similarities and differences. We then hint on some open and challenging problems that we believe are worth investigating.

1 Introduction

One of the main selling points of traditional database systems is consistency and accuracy. Given a database, the semantics of a query are precisely defined, and its answer is unique and exact. Approximations have been traditionally used only internally, as part of query optimization, whose goal is not really to find the actual optimal plan to execute a query but one that is close to it, avoiding the really poor ones. Given this approximate goal, database systems obtain approximations of the values of various size-related and cost-related parameters and use them to optimize queries.

In the recent past, however, things have changed and the need for approximations has increased sharply, moving also to the external layers of the database system. The answers to the queries themselves may now be approximate sometimes, as a quick feedback mechanism to the user on what to expect or because an accurate answer is expensive to obtain and not useful [5]. Furthermore, queries are no longer the only form of interaction with data, with on-line analytical processing, data mining, and other knowledge discovery methods playing a prominent role, all requiring essentially the extraction of approximate information from the database [2].

As a result, there has been extensive work on various aspects of the topic, with many interesting results. This work includes: extensive generalizations of earlier

* This is an approximate title as it does not quite represent the text that follows, which touches upon some issues that are beyond approximations or databases as well. The text itself is also an approximation, as it ignores several aspects of the topic, focusing only on personal curiosities.

D. Calvanese et al. (Eds.): ICDT 2003, LNCS 2572, pp. 16–30, 2003.

database techniques for traditional (relational) data; invention of new techniques that approximate more complex types of data, e.g., spatial and temporal data; cross-fertilization with other areas that deal with information approximation, e.g., signal processing, and very successful adaptations of their techniques to the database environment (e.g., wavelets); redefinition of traditional problems of other areas (e.g., pattern recognition, statistics) and invention of scalable solutions that are effective when applied on large databases; and others.

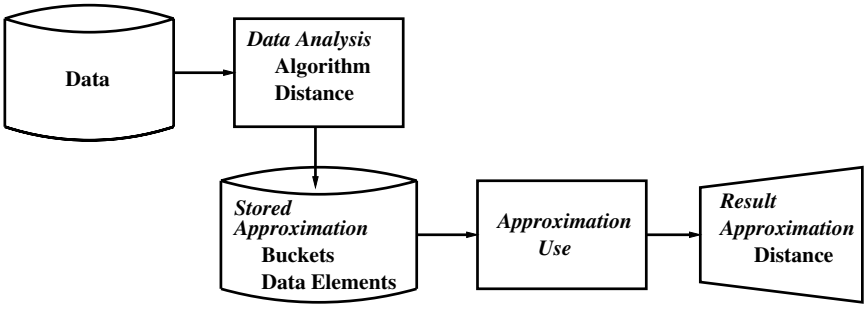

Fig. 1. Generic Flow of Approximation Process

For the purposes of this paper, any process of data approximation, exhibits roughly the flow shown in Figure 1. The nodes in that chain are described below:

1. *Data*: This is the original data that requires approximation.
2. *Data analysis*: This is the overall approach followed to derive an approximation of the incoming data, e.g., histograms or wavelets. Typically, incoming data elements are partitioned into groups of similar elements, called *buckets* (a term we use in this paper), *clusters*, *patterns*, and several other names. The data elements that fall in each bucket are then represented by the same approximate information. There are two aspects of each approximation technique that play a major role in its behavior:
   a) *Algorithm*: This is the algorithm run on the incoming data to generate its approximation.
   b) *Distance*: This is a measure of similarity between data elements that permits data elements that are "close" to be placed in the same bucket.
3. *Stored approximation*: This is the information generated by the approximation algorithm. It approximates the original data and is typically much smaller in size that it. This information is in the form of buckets representing parts of the space from which the original data is drawn. Again, there are two critical aspects of each bucket:
   a) *Bucket definition*: This is the specification of the part of the space that corresponds to a bucket.
   b) *Bucket data elements*: This is the information that is used to approximate the actual data elements that fall in a bucket.

4. *Approximation use*: This is the process by which the appropriate stored approximations are used to generate any approximate information required. Essentially, this corresponds to the actual purpose of the whole data approximation process.
5. *Result approximation*: This is the output of the previous step, i.e., the approximate information required at the end. A key parameter in characterizing this output is the following: that determines
   a) *Distance*: This is a measure of the quality of the generated approximate output, capturing its similarity to the corresponding output that would have been generated by applying a precise procedure on the original data. Usually it is intimately related to the distance parameter used by the data analysis performed earlier.

In this paper, we take a look at some of the most important data approximation processes that have been defined within different scientific areas and put them side by side with some of the most prominent techniques that have been proposed to solve them. We then hint on some open problems that we believe are interesting and challenging, whose solution may have some impact.

We should emphasize at this point that the entire area of data approximation is quite extensive and we make no attempt to be comprehensive or well-balanced by any means. We focus mostly on discrete data as it represents the primary concern of most data applications. Furthermore, we pay special attention to histograms and discuss them in more detail as they represent the main approximation technique used in current database systems. Finally, the choice of open problems is driven mostly by personal observations.

The paper is organized as follows. Section 2 defines data distributions and their characteristics that are useful for the rest of the paper and gives the essentials of histograms. Sections 3 to 8 outline some of the open problems that we consider worth investigating. Finally, Section 9 wraps up our thoughts.

2 Definitions

In the following, we briefly define several concepts on discrete data and present the key characteristics of histogram-based approximation, both of which are central in our discussion.

2.1 Data Distributions

For simplicity, we confine our detailed definitions on 1-dimensional data elements [9]. The *domain* $\mathcal{D}$ of an attribute X in relation R is the set of all possible values of X and the (finite) *value set* $\mathcal{V}$ ($\subseteq \mathcal{D}$) is the set of values of X that are actually present in R. Let $\mathcal{V} = \{ v_i \colon 1 \leq i \leq D \}$, where $v_i < v_j$ when $i < j$. The *spread* s_i of v_i is defined as $s_i = v_{i+1} - v_i$, for $1 \leq i < D$ (with $s_0 = v_1$ and $s_D = 1$)[1].

[1] The above holds for numerical attributes, where difference is defined. A commonly used technique for constructing histograms on non-numerical attributes (e.g., string

The *frequency* f_i of v_i is the number of tuples $t \in R$ with $t.X = v_i$. Finally, the *area* a_i of v_i is equal to $v_i \times f_i$. The *data distribution* of X (in R) is the set of pairs $\mathcal{T} = \{ (v_1, f_1), (v_2, f_2), \ldots, (v_D, f_D) \}$.

The above can be extended to n-dimensional data elements, coming from multiple attributes $X_i, 1 \leq i \leq n$ of R [8]. The *joint frequency* of a combination of n values, one from each attribute, is the number of tuples in R that contain the i-th value in attribute X_i, for all i. The *joint data distribution* $\mathcal{T}_{1,..,n}$ of $X_1, .., X_n$ is the entire set of *(value combination, joint frequency)* pairs.

2.2 Histograms

A *histogram* on an attribute X is constructed by using a *partitioning rule* to partition the data distribution of X into a number of mutually disjoint subsets called *buckets* and approximating the frequencies and values in each bucket in some common fashion. With respect to bucketization, several partitioning rules have been proposed. The most effective ones seem to be V-optimal(V,A) or V-optimal(V,F), whose buckets correspond to non-overlapping regions of X values with the minimum overall variance of the areas (A) or the frequencies (F) of the values grouped together, respectively. The MaxDiff(V,A) and MaxDiff(V,F) histograms are almost as effective but are much simpler to construct [7], and therefore, maybe slightly more popular. Their buckets correspond to non-overlapping regions of X values such that bucket breaks occur where the differences between the areas (A) or the frequencies (F), respectively, of neighboring values are maximized [9].

With respect to the information stored within each bucket to approximate the data elements that fall in it, the most common approach for values is to employ the *uniform spread assumption* [9], under which attribute values are assumed to be placed at equal intervals between the lowest and highest values in the bucket. Likewise, the most effective approach for frequencies is to employ the *uniform frequency assumption*, under which the frequencies in a bucket are approximated by their average.

Histograms can also be constructed on joint data distributions in a similar fashion. The task of identifying the appropriate partitioning of the multi-dimensional space in buckets is harder in this case, but several approaches have been developed that are quite successful in generating effective multi-dimensional histograms, at least for small numbers of dimensions [8,3].

3 Unification of Approximation Problems

In the past, several approximation problems have been defined that appear to have much similarity, yet they have been given different names and are approached using different techniques. Although it is clear that their differences

attributes) is to use a function that converts these vlaues into floating point numbers before constructing a histogram. For example, this technique is used in DB2.

are critical enough that they cannot all be unified completely, even partial unification would be really beneficial and could enrich the set of techniques applied to each of these problems significantly.

To illustrate the issues, we present three such problems and discuss their similarities and differences. One of them comes from the database world: *selectivity estimation*. The others come from the world of statistics and artificial intelligence: *clustering* and *classification*. We present all these techniques for the 2-dimensional case. With respect to the incoming 'Data' in Figure 1, we ignore any common differences in the focus of various problems and take a database perspective, i.e., we assume data elements on discrete numeric dimensions. Each of the other components of Figure 1 is discussed for each problem separately.

Selectivity Estimation

- *Data Analysis*: Partition the joint data distribution $\mathcal{T}_{1,2}$ into buckets, where each bucket contains similar elements. Similarity is defined based on some distance function that takes into account both the values of the two attributes and the value of the frequency.
- *Approximation Storage*: For each bucket, store a very short approximation of the $\mathcal{T}_{1,2}$ elements that fall there, typically by approximating each attribute/dimension and the frequency separately.
- *Approximation Use*: At query time, use the stored aggregate information to approximate the original $\mathcal{T}_{1,2}$ elements of the bucket, so that estimations of the query result size or sometimes the query result itself can be derived.

Clustering

- *Data Analysis*: Partition the joint data distribution $\mathcal{T}_{1,2}$ into buckets, where each bucket contains similar elements. Similarity is defined based on some distance function that takes into account just the values of the two attributes usually. Typically, clustering is used in cases where the frequency of each element is 1, but this does not have to be the case necessarily.
- *Approximation Storage*: For each bucket, store a very short approximation of the pairs of $\mathcal{V}_1 \times \mathcal{V}_2$ that fall there and the general area they occupy. Typically, this is a representative (not necessarily existing) element and a radius around it [10], but for scalable approaches, this may be much richer[2].
- *Approximation Use*: At insertion time, the inserted element is compared against the representative of each cluster and is placed in the most appropriate one (if any).

Classification. In this, next to every element of $\mathcal{V}_1 \times \mathcal{V}_2$ there is another attribute that takes values from a known, finite (typically small) set, the set of classes/categories. The goal is to identify the characteristics of the elements that fall into each class.

- *Data Analysis*: Partition the joint data distribution $\mathcal{T}_{1,2}$ into buckets, where each bucket contains similar elements. Similarity is defined based on some

distance function that takes into account just the values of the two attributes and on equality when it comes to the class attribute. Here again, in classification, the frequency of each element is 1. Even if it is not, however, it is mostly ignored and the effort concentrates on the pairs in $\mathcal{V}_1 \times \mathcal{V}_2$ (actually, usually on only one of the dimensions in each step of the algorithm) and the class attribute.

- *Approximation Storage*: For each bucket, store a very short approximation of the subspace occupied by the pairs of $\mathcal{V}_1 \times \mathcal{V}_2$ that fall there (but not the pairs themselves), typically through specifications of the borders of the subspace, as well as the category they represent.
- *Approximation Use*: At insertion time, the inserted element is compared against the areas of the buckets and is classified into the class associated with the bucket it falls in.

Note that the approximation use is rather distinct for each problem, and this is natural as to a large extent it defines the problem. For example, note that the purpose of selectivity estimation is dealing with queries, whereas the remaining problems deal mostly with updates (what happens when a new element arrives). On the other hand, often clustering may have no explicit use phase (the interest being in identifying clusters in existing information), whereas the other problems do for the most part.

With respect to the other two main components of Figure 1, however, comparing the three problems above and approaching them from a database perspective, we observe some striking similarities. The data analysis performed under each problem is essentially the same, yet the techniques that are used are very different. More or less the same holds for approximation storage. Clearly, some of these differences are to be expected, e.g., the presence of the additional class attribute for classification may have a positive or negative impact on the appropriateness of a specific analysis technique. In general, however, there appears to be no well documented reasoning for many of these differences.

The above raises several interesting questions. Why can't the histogram techniques that have been developed for selectivity estimation be used for clustering or vice versa? What would the impact be of using stored approximations developed for one problem to solve another? Is the different applicability of the techniques due to the differences in the intended use in each approximation problem? In that case, what is the extent of any such dependency and how can it be articulated? In general, given the great variety of techniques that exist for data approximation, it is crucial to obtain an understanding of the advantages and disadvantages of each one, its range of applicability, and in general, their relative characteristics when mutually compared. A comprehensive study needs to be conducted that will include several more techniques than those mentioned here. The "New Jersey Data Reduction Report" [1] has examined many techniques and has produced a preliminary comparison of their applicability to different types of data. It can serve as a good starting point for verification, extrapolation, and further exploration, not only with respect to applicability,

but also precise effectiveness trade-offs, efficiency of the algorithms, and other characteristics.

There are also questions on the nature of the approximation problems as well. Why can't the class attribute in classification be considered as an additional dimension of the data element space and have the problem be considered as clustering? When is this meaningful? Likewise, why can't the frequency in selectivity estimation be considered as another dimension of the joint data distribution and have the problem be considered as traditional clustering? The only distinction of the frequency is that there is a functional dependency from the attributes X_1 and X_2 to the frequency. What exactly does one gain (if anything) by taking advantage of that fact and treating frequency separately?

Elaborating on the last point, consider Figure 2 showing a typical data distribution of a single attribute X of a relation. Assuming any of the established

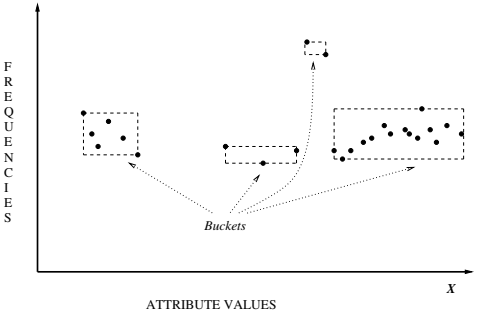

Fig. 2. One-Dimensional Data Distribution

most effective types of histograms mentioned earlier, the data distribution elements will be partitioned so that there is no overlap in the values of X among the buckets and that similar frequencies and distances are placed in each bucket. Assuming further that there is space for four buckets, the buckets shown in Figure 2 are the likely outcome of any histogram construction algorithm.

Let us ignore, for a moment, the fact that the vertical axis represents frequencies and treat it as just a second dimension of the objects. The problem immediately changes to that of clustering; furthermore, applying any of the main clustering techniques that exist in the literature, it is very likely that the resulting clusters would have been the same. It is this relationship and similarity between the two problems that needs to be investigated and capitalized upon.

Such investigation is certainly not trivial and would need to address several general and detailed issues that arise. To mention just one of these issues, in Figure 2, one could argue that typical clustering algorithms applied on the entire 2-dimensional space might not have formed the second bucket. The reason would be that, although the y-values of the points (frequencies) are similar, the x-values are relatively far apart. The reason why histograms would probably form that bucket is that in selectivity estimation, the approximation of the x-axis in each

bucket is through some version of the uniform spread assumption. Hence, it is not proximity of the values themselves that is required for forming a bucket but proximity of their spreads. A hasty conclusion from the above may be that the selectivity estimation problem reduces precisely to clustering if applied on the space where the x-axis represents spreads instead of values. This is not quite true, however, as the spreads of the first points in every bucket are immaterial, so a traditional clustering algorithm would not succeed completely either. What one could do in this case is open for investigation.

As a last point of this section, we would like to note that there may be close relationships between the problems not only at the full-scale level discussed so far, but at a finer level as well. As a simple example, we consider the classification problem and one of its most important classes of algorithms: *decision tree* methods [2]. Operating on a multi-dimensional joint data distribution $\mathcal{T}_{1,..,n}$, the most popular decision tree algorithms (*ordinary binary classification trees*) sequentially split one of the dimensions in two pieces, eventually partitioning the space into hyper-rectangles whose elements belong to a unique class. Ignoring the tree-pruning aspects that these algorithms employ, which are irrelevant to our discussion, this is exactly how, for example, the MHIST approach to multi-dimensional histograms operates [8]. The splitting criteria employed in decision trees methods are different from those associated with MHIST, which are the traditional ones applied for 1-dimensional histogram construction. Could these be appropriate for decisions trees as well (applied on the class attribute)? Would the decision tree construction criteria be appropriate for histograms? Would some of the other histogram approaches be appropriate as a general approach to decision tree construction? This and several other potential relationships need to be worked on for the possible benefit of all problems.

4 Pattern (Bucket) Recognition

In all three problems mentioned above, data analysis is applied on a given set of dimensions. In traditional pattern recognition, however, there is usually an earlier stage where the dimensions are chosen among a great number of possibilities. A critical problem then is which dimensions to choose. There are several techniques that exist that address this problem with varying success depending on the case.

In selectivity estimation, dimension selection does not exist for the most part, as what needs to be approximated is usually known and given. Nevertheless, this issue may arise when we try to transform selectivity estimation into a clustering problem, as mentioned in Section 3, where the actual attribute values may have to be replaced by their corresponding spreads or something else. How does one choose appropriate alternatives? Although for the particular case, this may turn out to be not very hard, in general deriving dimensions in which patterns emerge, not by choosing among alternatives but by transforming them, is a very hard problem, not only for database estimation but for pattern recognition in general.

As an example, consider the 2-dimensional space above (whether the y-axis represents frequencies or not is irrelevant). It is clear to the human eye that

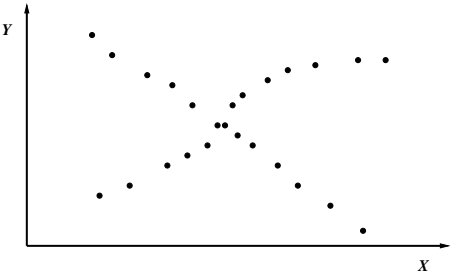

Fig. 3. Hard-to-Discover Patterns

there are two patterns in the figure, one with Y decreasing and one with Y increasing as X increases. Nevertheless, the patterns cross each other in the X-Y space, the elements that belong to each pattern do not satisfy any proximity criteria in that space, so current clustering techniques will most likely fail to identify them. One would have to model the elements in a very specific way, in a different multi-dimensional space, for the elements of each pattern to be close to each other and far from the elements of the other pattern. Understanding the mechanisms of such general pattern recognition is a great challenge that needs investigation. Such an effort should certainly try to address all five Gestalt Rules for clustering: small distance (on which current clustering techniques are quite effective), similarity, completion, continuity (where the example of Figure 3 falls), and symmetry. Some are harder to define than others, but we believe that an investigation in this direction will produce very interesting results.

5 Approximation within a Bucket

5.1 Approximation of Dimension Values and Frequencies

Consider the 1-dimensional data distribution of Figure 2, and the four buckets that have been indicated there. As mentioned in Section 2.2, a histogram makes certain uniformity assumptions within each bucket, most often the *uniform distribution assumption* for frequencies and some version of the *uniform spread assumption* for values. Hence, for the example of Figure 2, the stored information would generate the approximate distribution of Figure 4. Observe that frequencies are treated differently from attribute values: the frequencies in a bucket are assumed constant, whereas the values in a bucket are assumed to follow a linear rule. Hence, buckets need to store more information to approximate the values that fall in them than their frequencies. In principle, however, not all data distributions are served best with such an approach. For example, compare the three distributions of Figure 5. The first two are very similar cases, only that one can be thought of as rotated 90 degrees compared to the other. The third one is a mixture of the two. The current approach of histograms would work very

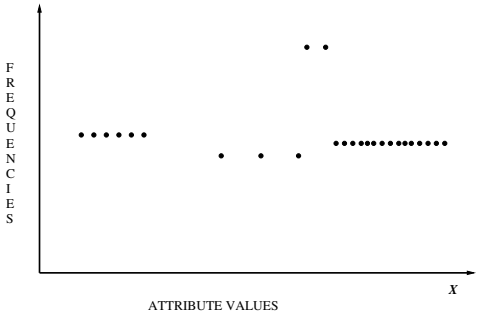

Fig. 4. Approximate One-Dimensional Data Distribution

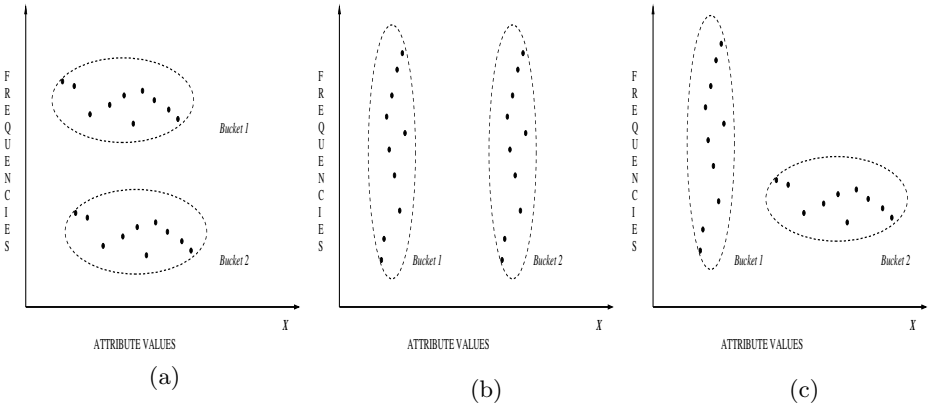

Fig. 5. (a) "Horizontal" Buckets, (b) "Vertical" Buckets, (c) "Mixed" Buckets

well with the left distribution[2], but not with the other two. For that second case, a much better approximation would be to store the average attribute value of a bucket (since they are all roughly the same) and assume a linear function for the frequencies that these roughly-equal values have. Likewise, for the third case, one bucket should be treated like the first case, and the other like the second case.

One may argue that range queries on values would perform poorly if all similar values are represented by their average. However, it is unclear if that would have a negative effect on the average performance if the values in a bucket come from a small range. One may further argue that distributions like those of Figure 5 are not expected to be found in databases, so the problem would never arise in practice. However, the problem remains even if the y-axis does not represent frequencies but a second value dimension, a case which is more

[2] This is assuming that buckets are allowed to overlap in the value domain, which is not typically the case, but this is orthogonal to the focus of this discussion which is approximation within a bucket.

plausible. In that case, current histograms would apply some version of the uniform spread assumption for both dimensions of all buckets, whereas clearly a distinct approach for each dimension in each bucket would have generated much better results.

To increase the accuracy of histogram approximations, there should be no fixed, predefined approximation approach to the value dimensions and the frequencies. Histograms should be flexible enough to use the optimal approximation for each dimension in each bucket, one that would produce the best estimations for the least amount of information. Identifying what that optimal approximation is is a hard problem and requires further investigation. Incidentally, note that this problem is intimately related to the problem of pattern identification mentioned above (Section 4), as identifying a pattern implies identifying a small amount of information that can be used to represent the elements of the pattern without much loss of information.

5.2 Approximation of Multi-dimensional Elements

Consider a 2-dimensional space and suppose that a 2-dimensional histogram uses the uniform spread assumption on each dimension of each bucket to approximate the values that appear there. The original values represent the projections of the actual elements that fall in the bucket, and the uniform spread assumption leads to an approximation of those projections. It does not offer an approximation for the actual placement of the elements in the 2-dimensional space. Current approaches make some gross oversimplifications to the problem, which may be responsible for large estimation errors. For example, if the projection of the elements on dimension X_i generates n_i unique values, $i \in \{1, 2\}$, then often the assumption is made that all elements of the cross-product of the values in the dimension projections are in the bucket. Clearly, this may be far from reality, as the cross-product has $n_1 * n_2$ elements, whereas the same projections might have been generated by as few as $max(n_1, n_2)$ actual elements.

The key question that arises then is how to take into account both the actual number of elements in a 2-dimensional (rectangular) bucket and their projections and place them uniformly in it. There does not seem to be an easy answer to this question. The source of the problem may be that the concept of uniformity is not necessarily appropriately defined in this case. Needless to say, the problem may be even harder in higher dimensions. We believe that any solution to this problem would improve the accuracy of histograms significantly in several cases.

6 Distance Metric

As shown in Figure 1, the concept of distance is central to approximation. It plays a major role in comparing data elements for bucket formation, identifying the appropriate approximate information to be stored, and even comparing results of data processing (on accurate and approximate data) for error measuring. Choosing the appropriate distance metric is crucial.

Clearly, there is no issue with the distance of numeric values, as there is a very natural definition of distance for them based on the difference of their actual values. The same is more or less true for elements in a multi-dimensional space with numeric dimensions, where there are several natural distance definitions (the L-norms) that produce useful results. The situation stops being as straightforward as soon as we move on to other types of elements, especially those with rich structure, e.g., graphs, XML files, and other semi-structured data. To illustrate the problem, we discuss two data types that are problematic: strings and sets (used mostly in the general sense of multisets, i.e., bags).

With respect to strings, a classical approach is to use the *edit distance*, i.e., the number of edit actions (insertions, deletions, replacements, swaps, etc.) that one has to perform in order to reach one string from the other. With respect to approximating string information in a bucket, a trie is usually employed. All these however, are only syntactic approaches to string distance. There are several cases, where a more semantic approach is warranted and might give better results. For example the strings 'city', 'town', and 'village' are very far apart in edit-distance, but are very close semantically. More often than not, the information conveyed by placing them in one bucket and using one of them to represent all three would be far more accurate than any other approach. What forms of semantic distances would make sense, how to maintain such knowledge that would allow for semantic distances to be calculated efficiently, and what the impact of using such distance metrics would be on approximations are questions that require further investigation.

With respect to sets, the problems are even harder. In comparing two sets, one has to overcome value mismatches (different member elements appearing in the two sets) and cardinality mismatches (both at the level of one set being larger than the other and at the level of one member element appearing different number of times in the two sets). The former could be consider as a special case of the latter using zero cardinality for the nonexistent elements, but it carries distinctly different semantics, so it should be treated separately. Traditional measures from other areas (e.g., Hausdorff distance) do not take into account either one of the mismatches above, hence, they are not very appropriate for approximations in databases. Other distances that have been introduced exactly for that purpose (e.g., DIST [6]) still seem to produce unintuitive distances on certain cases, e.g., the distance between $\{1, 1, 1, 2\}$ and $\{1, 2, 2, 2\}$ is equal to 12, the same as the distance between $\{1, 1, 1, 100\}$ and $\{1, 100, 100, 100\}$. Finding distance functions for sets that respect intuition and have useful mathematic properties (e.g., being metric) is quite a challenge, but would be a major step forward in approximating sets in databases.

7 Approximations and Indices

The fact that there is a close relationship between approximate statistics kept in databases, especially histograms, and indices has been recognized in the past in several works [1]. If one considers the root of a B+ tree, the values that

appear in it essentially partition the attribute on which it is built into buckets with the corresponding borders. Each bucket is then further subdivided into smaller buckets by the nodes of the subsequent level of the tree. One can imagine storing the appropriate information next to each bucket specified in a node, hence transforming the node into a histogram, and the entire index into a so called *hierarchical histogram*. This may adversely affect index search performance, of course, as it would reduce the out-degree of the node, possibly making the tree deeper. Nevertheless, although this idea works against the main functionality of an index, its benefits are non-negligible as well, so it has even been incorporated into some systems.

We believe that hierarchical histograms and, in general, the interaction between approximation structures and indices should be investigated further, as there are several interesting issues that remain unexplored as analyzed below. Consider again a B+ tree whose nodes are completely full, and assume for simplicity that it has been built on a foreign key of a relation. In that case, the root of the tree specifies a bucketization of the attribute domain that corresponds to an equi-depth histogram, i.e., each bucket contains an equal number of elements under it. Similarly, any node in the tree specifies an equi-depth bucketization of the range of values in leads to. If the nodes of the tree are not completely full, the equi-depth property does not quite hold, in the worst case one bucket's occupancy being within a factor o 2^h of another's, where h is the depth of the tree. On the average, however, the equi-depth property should hold to a satisfactory level.

The main issue with B+ trees being turned into hierarchical equi-depth histograms is that equi-depth histograms are far from optimal overall on selectivity estimation [9]. Histograms like V-optimal(V,F), V-optimal(V,A), MaxDiff(V,F), and MaxDiff(V,A) are much more effective. What kind of indices would one get if each node represented bucketizations following one of these rules? Clearly, the trees would be unbalanced. This would make traditional search less efficient on the average. On the contrary, other forms of searches would be served more effectively. In particular, in a system that provides approximate answers to queries, the root of such a tree would provide a higher quality answer than the root of the corresponding B+ tree. Furthermore, the system may move in a progressive fashion, traversing the tree as usual and providing a series of answers that are continuously improving in quality, eventually reaching the leaves and the final, accurate result.

Returning to precise query answering, note that typically indices are built assuming all values or ranges of values being equally important. Hence, having a balanced tree becomes crucial. There are often cases, however, where different values have different importance and different frequency in the expected workloads. If this query frequency or some other such parameter is used in conjunction with advanced histogram bucketization rules, some very interesting trees would be generated whose average search performance might be much better than that of the B+ tree.

From the above, it is clear that the interaction between histograms and indices presents opportunities but also several technical challenges that need to be investigated. The trade-off between hierarchical histograms that are balanced trees with equi-depth bucketization and those that are unbalanced with more advanced bucketizations requires special attention. The possibility of some completely new structures that would strike even better trade-offs, combining the best of both worlds, cannot be ruled out either.

8 Original Data and Final Result Approximation

The last set of challenges we would like to mention are related to the two end components of Figure 1. As hinted in the rest of the paper, most current approximation efforts are applicable to data elements with numeric fields, whether alone in a 1-dimensional space or combined in multi-dimensional spaces. Very little has been done on categorical domains. Current practice is to map the categorical values onto a numeric domain and then use its properties. This often presents problems, as categorical values exhibit behavior that is unnatural to them. The earlier discussion on possible distance functions for strings may be relevant in general for categorical domains and their approximation. Even greater challenges are presented in approximating information with more complex structure, e.g., graphs or XML files.

On the other end, most current efforts on selectivity estimation have focused on selection queries. Dealing with joins or other operators or combinations of them has been rare [4] and has not produced completely satisfactory results. Given the importance of more complex queries, these problems should be given some special attention.

9 Conclusions

The importance of approximation in database systems will continue to grow, as increasingly more such functionality will be required. This will come both from demands for higher quality approximations in domains that are already being dealt with in database systems, e.g., better histograms, and from demands for new functionality, e.g., progressive refinement of query answers or advanced data mining applications.

We have identified several problems that we believe they represent nontrivial technical challenges and their solution (or any progress toward a solution) would improve our understanding of how information should be approximated most effectively and could influence directly or indirectly current approaches. These problems are summarized below:

- To what extent can the various approximation problems defined so far be unified under a common model, and what is the effectiveness of techniques developed for one such problem when used for another?

- What are the underlying principles of human pattern recognition and how can they be used to develop techniques for bucketizing data in the most effective way even when that is not a result of proximity in the natural space where the data resides?
- Given a set of elements that fall in a bucket, what information should be stored in the bucket for the most effective approximation of the set, in terms of obtaining the best trade-off between information size and approximation error?
- What are intuitive and mathematically robust distance functions for non-numeric data elements or data elements with complex structure?
- What are the opportunities that arise from combining index technology with approximation technology and how can the trade-off between efficient index searches and high-quality approximations be balanced?
- How can complex data elements and complex operator results be approximated?

It seems that there is a lot of work on approximations in front of us. Hopefully, as a community, we will be able to move forward on some of these problems, and have some fun on the way too!

Acknowledgements. We would like to thank Minos Garofalakis and Raghu Ramakrishnan for their comments on an earlier version of this paper. Their feedback was crucial in making this paper a better approximation of reality!

References

1. Barbará D., et al.: The New Jersey Data Reduction Report. Data Engineering Bulletin **20:4** (1997) 3–45
2. Bradley P., Gehrke J., Ramakrishnan R., Srikant R.: Scaling Mining Algorithms to Large Databases. CACM **45:8** (2002) 38–43
3. Bruno N., Chaudhuri S., Gravano L.: STHoles: A Multidimensional Workload-Aware Histogram. SIGMOD Conference (2001) 294–305
4. Chakrabarti K., Garofalakis M., Rastogi R., Shim K.: Approximate Query Processing Using Wavelets. VLDB Journal **10:2-3** (2001) 199–223
5. Garofalakis M., Gibbons P.: Approximate Query Processing: Taming the Terabytes. VLDB Conference Tutorial (2001)
6. Ioannidis Y., Poosala V.: Histogram-Based Approximation of Set-Valued Query-Answers. VLDB Conference (1999) 174–185
7. Jagadish H. V., et al.: Optimal Histograms with Quality Guarantees. VLDB Conference (1998) 275–286
8. Poosala V., Ioannidis Y.: Selectivity Estimation Without the Attribute Value Independence Assumption. VLDB Conference (1997) 486–495
9. Poosala V., Ioannidis Y., Haas P., Shekita E.: Improved Histograms for Selectivity Estimation of Range Predicates. SIGMOD Conference (1996) 294–305
10. Theodoridis, S.: Pattern Recognition. Encyclopedia of Information Systems, Vol. 3. Elsevier Science (2003) 459–479