

---

## The role of data mining in turning bio-data into Bioinformatics

William Perrizo\*

Department of Computer Science, North Dakota State University, USA;

Email: william.perrizo@ndsu.edu; \* Corresponding author

received January 24, 2007; published online January 31, 2007

### Editorial:

The explosion of high throughput data generation technologies have added tremendous volumes of data to the already huge collections found in digital form. This is particularly true in Biology. In the near future higher throughput technologies will only exacerbate this raw data overload situation. The explosive growth in raw data volume generates the need for new techniques and tools that can intelligently and automatically transform the data into useful information and knowledge. One of the central tenets of all information theories is that “the higher the data volume, the lower the information (and knowledge) level”. This can be referred to as the “Data Overload, Information Underload” (DO/IU) problem. The DO/IU problem exists in most fields, not just in Biology. It has been pointed out by experts in almost all fields that involve voluminous data.

The crux of the DO/IU problem is volume. Data processing tools, which convert voluminous raw data to succinct pieces of information (summaries, relationships, patterns and other “answers”), are needed which can find (data mine) pertinent, accurate information from the raw data and do it in a reasonable amount of time. So the problem is, as it has always been, scalability, of data processing algorithms. Scalability is always cited as one of the main, if not the main, challenge in nearly every major address given by prominent information scientists over the past 50 years. It was the principle motivation for the development of the computer in the first place. Everyone seems to agree on this point.

Scalability comes in at least two varieties, cardinality scalability (too many instances) and dimensionality scalability (too many attributes). The scalability problems can be cast in terms of tabular terminology as too many rows and too many columns. Of course, too many tables can also still be a problem, but one could say that that problem has been solved to a satisfactory extent by the database research community over the past 40 years, evidenced by the fact that most database researchers are now doing data mining. The two problems will be referred to as the curse of cardinality and the curse of dimensionality. The primary solution to date has been sampling.

The primary solution for the curse of cardinality has been to select (randomly?) a representative subset of records (instances or rows), then to analyze or mine that subset.

The tacit assumption is that the information (relationships, patterns, summaries, etc.) found in the subset applies to the full data set as well. Whereas, that tacit (statistical) assumption can (and should always) be justified in many cases (particularly when the answers sought are of a summary nature), it is very difficult to justify in others, e.g., in exception mining. A random subset will almost always miss exceptions, since exceptions are, in some sense, of measure zero, and small random sub-samples intersect measure zero sets with measure zero. Put another way, if the probability is high that sub-sampling will include an exception, then it may be incorrect to call it an exception in the first place.

There is a definite need for a class of full-sample solutions to the curse of cardinality. It is suggested in this editorial, that such solutions should structure the data vertically instead of the ubiquitous horizontal (record-based) structuring. Why? Very roughly, compressed vertical structures do not grow in number as more records are generated. Each of them will grow in size, but with proper compression, they will grow very sub-linearly. Another potentially important characteristic of a good compressed vertical technology is that processing of the structures can be accomplished on the compressed version (and not require decompression first).

Two observations need to be made immediately. First, indexes to horizontal data sets are vertical, so vertical structuring is not new. However, indexes are auxiliary vertical data structures which are created (and maintained) in addition to the horizontal data sets they index. One way to view the vertical solution recommended here is that it replaces the horizontal data set with one universal index, if you will. Second, fully vertical databases have been proposed in the past (e.g., the Bubba project of the 1980's). Possibly, the reason that these vertical database technologies have been over-shadowed by the ubiquitous horizontal relational technologies, is that for database processing, the desired result usually has horizontal structure (the output of a relational query is typically a relation itself). Therefore, to structure the data vertically, process vertically, then have to convert the result to a horizontal structure, may have been too inefficient.

The primary solution for the curse of dimensionality is also to select (non-randomly) a pertinent subset of features

(columns or attributes). This process is often referred to as feature selection (e.g., principal component analysis). It can also involve custom rotation first and then feature selection. In fact, this solution is not in the nature of a work-around (which the sub-sampling of instances is for the curse of cardinality). Provided there IS a reduced subset of features which ARE the pertinent ones for the analysis undertaken (i.e., the subset holds nearly all the information needed), those ARE the features that should be focused upon. However, sometimes that sub-collection of features is still very large (and sometimes all features are pertinent - i.e., hold important information). In these later cases the, so-called, curse of dimensionality may be more appropriately termed the fact of high pertinent dimensionality, which is to say, there may not exist a scalable solution to it.

One of our primary data mining tools in Biology is the fast construction of Nearest Neighbor Sets (NNS) of a sample point. Why are NNSs important? They are important because most predictions and classifications are based on a continuity assumption, namely, if the inputs are close, then the outputs will be close. NNSs provide the mechanism to define close.

For a vertically structured data set, the type of NNS that is most scalable to construct is the, so called, Max NNS (or  $L_\infty$  NNS). Max NNSs are NNSs containing all neighbors within a given radius with respect to the Max distance ( $L_\infty$  distance). However, Max NNS rings do not provide a uniformly near set of neighbors. In fact, as the number of pertinent dimensions increases, the (distance) uniformity of a Max Near Neighbor ring degrades markedly. For example, given a 64-dimensional data set, and a radius,  $r$ , the Max disk about a sample point,  $a$ , of radius  $r$ , contains some boundary points (those on the main diagonals) which are 8 times as far away from the sample as other boundary points (those on the intercepts). So the  $r$ -boundary ring about  $a$  is very non-uniform (relative to standard Euclidean distance, that is). Said another way, the Max disks have spikes. However, there are methods (using Max disk candidate supersets of NNS) which prune down the number of candidates to a set that can be scanned scalably for the uniform or Euclidean NNS of the sample.

Data mining or knowledge discovery in biological databases (KDDDBIO), aims at the discovery of useful patterns from large data volumes. Data mining is becoming much more important as the number of databases and sizes of database grows. A data mining system is considered (linearly) row scalable if, when the number of rows is increased by, e.g., 10 times, it takes no more than 10 times as long to execute the same data mining queries. A data mining system is considered column (linearly) scalable if the data mining execution time increases linearly with the number of columns (or attributes or dimensions).

### What is Vertical Data-mining Technology (VDT)?

In vertical data sets, the data in each table, file or relation is vertically partitioned (projected) into a collection of separate files, one for each column or even one for each bit position of each (numeric) column. Such vertical partitioning requires that the original match-up of values be retained in some way, so that the "horizontal" record information is not lost. In this approach, the horizontal match-up information is retained by maintaining a consistent ordering or tree positioning of the values, relative to one-another. Considering a list to be a 0-dimensional tree, then it is correct to speak in terms of tree-positioning only.

VDT partitions all data tables into individual vertical attribute files, and then, for numeric attribute domains, further into individual bit-position files or other coded bit files. For non-numeric attribute domains, such as categorical attribute domains, VDT either codes them numeric or constructs separate, individual, vertical bitmaps for each category. If the categorical domain is hierarchical, VDT simply uses composite bitmaps to accommodate the higher levels in that concept hierarchy.

The first issue is that data mining almost always expects just one table of data. Although Inductive Program Logicians have attempted to deal with multi-table or multi-relational data directly, it can be argued that these methods have inherent shortcomings. The VDT approach is to combine the multiple tables or relations into one first and then mine the resulting "universal" table. However, any such approach would only exacerbate the curse of cardinality (and to some extent the curse of dimensionality) if applied directly, that is, if applied by first joining the multiple tables into one massively large table and then vertically partitioning it. The VDT approach is to convert the sets of compressed, lossless, vertical, tree structures (Predicate-trees or just P-trees) representing the original multiple tables directly to a set of compressed, lossless, vertical, tree structures (compressed P-trees) representing the universal relation, without ever having to actually join the tables. Since the resulting trees are compressed, this ameliorates the curse of cardinality to a great extent.

As to the curse of dimensionality, except for domain knowledge related and analytical (e.g., Principal Component Analysis) dimension reduction methods, it is the opinion of this author that there is no way to relieve the curse of dimensionality without the loss of information. Thus, again, in some real sense, it is not a curse but a fact.

### What is Data mining?

Data mining, in its most restricted form can be broken down into 3 general methodologies for extracting information and knowledge from data. These (inter-related) methodologies are Association Mining, Classification and

**Clustering.** To have a unified context in which to discuss these three methodologies, assume that the “data” is in one relations,  $R(A_1, \dots, A_n)$  (a universal relation – un-normalized) which can be thought of as a subset of the product of the

attribute domains,  $\times_{i=1}^n D_i$

**Association Mining** is a matter of discovering strong association relationships among the subsets of the columns (in the schema). If these associations are unidirectional, this is called Association Rule Mining (ARM) or antecedent-consequent relationship mining. If the relationships are undirected, this is called Correlation Mining.

**Classification** is a matter of discovering signatures for the individual values in a specified column or attribute (called the class label attribute, which can be composite), from values of the other attributes (called the feature attributes) in a table (called the training table).

**Clustering** is a matter of using some notion of instance similarity to group together training table rows so that within a group (a cluster) there is high similarity and across groups there is low similarity. In Biological Data Mining, it is very common to use clustering to accomplish classification (class discovery). That is, when some (small?) portion of the data is already classified, the entire data set can be clustered based on some similarity notion. Then unclassified samples can be assigned likely classes based upon the preponderance within its cluster. In Biology this is called putative annotation. The so-called BLAST technologies fall in this category.

Given a training table,  $R$ , one can distinguish those attributes which are entity keys,  $K_1, \dots, K_k$ , i.e., each is a (composite?) attribute which minimally, uniquely identifies instances of the entity for all time (not just for the current state). In addition to the key attributes, there are feature attributes for each of the keyed entities. The feature attributes for entity key,  $K_i$ , will be denoted,  $A_{i,1}, \dots, A_{i,n_i}$ . It is assumed that there is one central fact to be analyzed.

The domain of each attribute, whether structural (key) or descriptive (feature), has associated with it a semantic hierarchy (ontology). To model these semantic hierarchies, one use an ontological hierarchy (OH). Moving up the hierarchy is “rolling up” an entire key dimension to the top of its semantic hierarchy where it has one value, the entire domain, and therefore is eliminated along with its features. This is a schema-level rollup in the sense that it can be defined entirely at the schema (intentional) level and need not involve the instance (extensional) level. However, one can partially roll up (or down) any or all key attributes). This is an extension-level rollup on keys.

Finally, one can think of projecting off a feature attribute as a complete (schema-level) rollup of that attribute to eliminate the information it holds completely (and therefore eliminate the need for it completely). One can think of a slice as another example of an aggregation-function-free rollup. Rollups can involve central tendency operators (e.g., mean, median, mode, midrange), true aggregations (e.g., sum, average, min, max, count), and measures of dispersion (e.g., quartile operators, measures of outlier-ness, variance). Each feature attribute can be extension-level rolled up or down within its semantic hierarchy (ontology).

Figure 1 below show three primary biological entities, Genes, Organisms and Experiments (the GEO star Schema and their relationships). Figure 2 below shows the same three primary biological entities, Genes, Organisms and Experiments and all the attendant data sets and their relationships. This is the GEO Constellation (multiple stars). How should this GEO data (and other massive data collections) be structured to facilitate data mining? This presentation advocates vertical structuring.

For several decades and especially with the preeminence of relational database systems, data is almost always formed into horizontal record structures and then processed vertically (vertical scans of files of horizontal records). This makes good sense when the requested result is a set of horizontal records. In knowledge discovery and data mining, interest typically centers on collective properties or predictions that can be expressed very briefly. Therefore, the approaches for scan-based processing of horizontal records are known to be inadequate for some data mining in very large data repositories. For this reason much effort has been focused on sub-sampling and indexing as methods for addressing problems of scalability. However, sub-sampling requires that the sub-sampler know enough about the large dataset in the first place, to sub-sample “representatively” (and it will almost always miss exceptions). Sub-sampling representatively presupposes considerable knowledge about the data. For many large datasets, that knowledge may be inadequate or non-existent.

Index files are vertical structures. That is, they are vertical access paths to sets of horizontal records. Indexing files of horizontal data records does address the scalability problem in many cases, but it does so at the cost of creating and maintaining the index files separate from the data files themselves. A database model in which the data is losslessly, vertically structured and in which the processing is based on horizontal logical operations rather than vertical scans (or index-optimized vertical scans) is proposed. The model is not a set of indexes, but is a collection of representations of dataset itself. The model incorporates inherent data compression and contains information useful in facilitating efficient data mining.

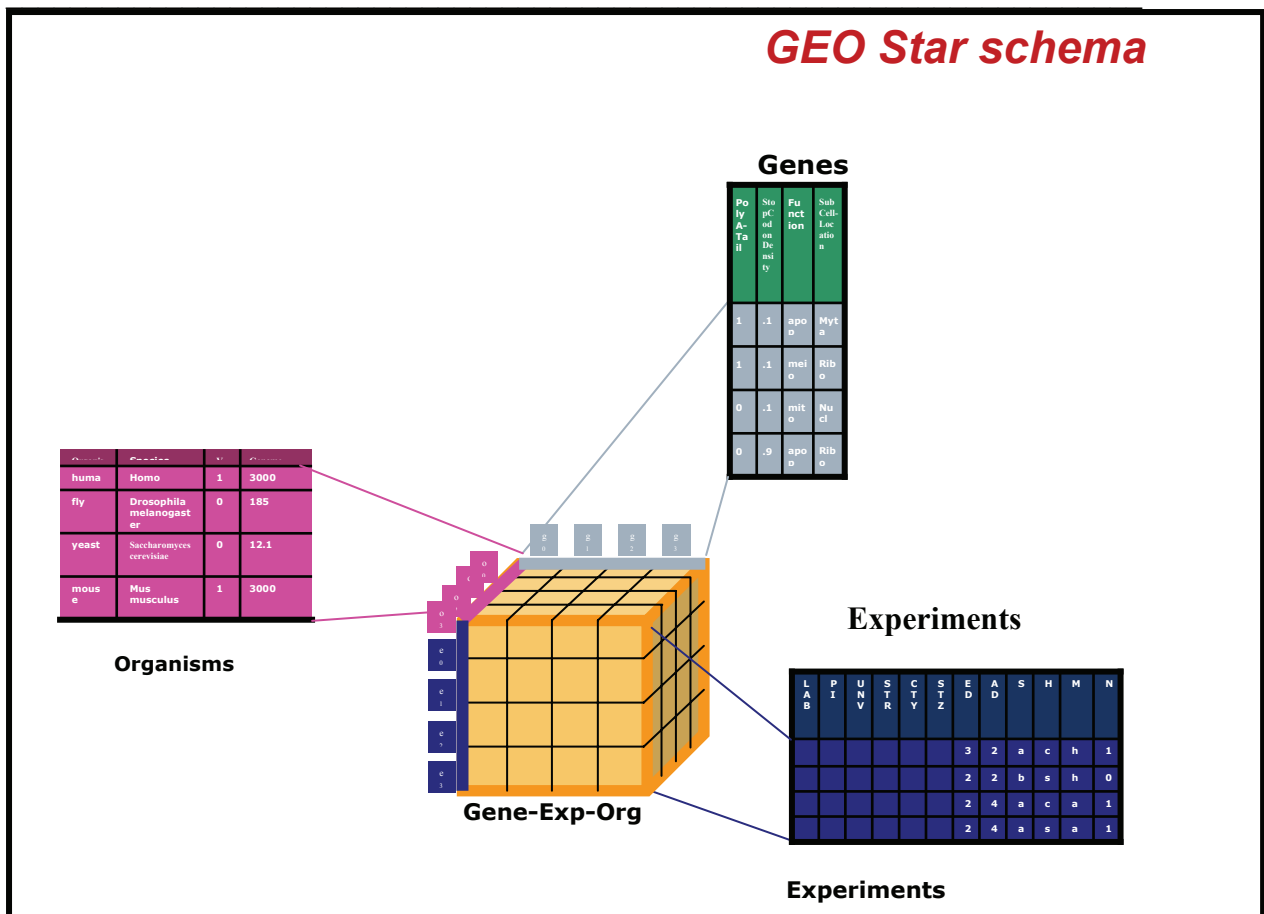


Figure 1: The gene experiment organism Star schema

### Predicate Trees or P-trees from a functional contour point of view

Given  $f: R(A_1, \dots, A_n) \rightarrow Y$  and  $S \subseteq Y$ , the (uncompressed, single node-level) Predicate-tree  $P_{f,S}$  is the bitmap,  $P_{f,S}(x) = 1$  (i.e., true) iff  $f(x) \in S, \forall x \in R$ . Note that  $P_{f,S}$  bitmaps the set,  $F\text{-contour}(\{1\})$  under the function,  $F(x) = 1$  if  $f(x) \in S$  else  $F(x) = 0$ . In Mathematical terms,  $F$  is the Characteristic function of the contour,  $f^{-1}(S)$ .  $F$  can also be viewed as the  $S$ -set containment predicate.  $P_{f,S}$  is called a P-tree for short and is just the existential  $R^*$ -bit map of  $S \subseteq R^* \cdot A_f$ . The Compressed P-tree,  ${}^sP_{f,S}$  is the compression of  $P_{f,S}$  with equi-width leaf size,  $s$ , as follows.

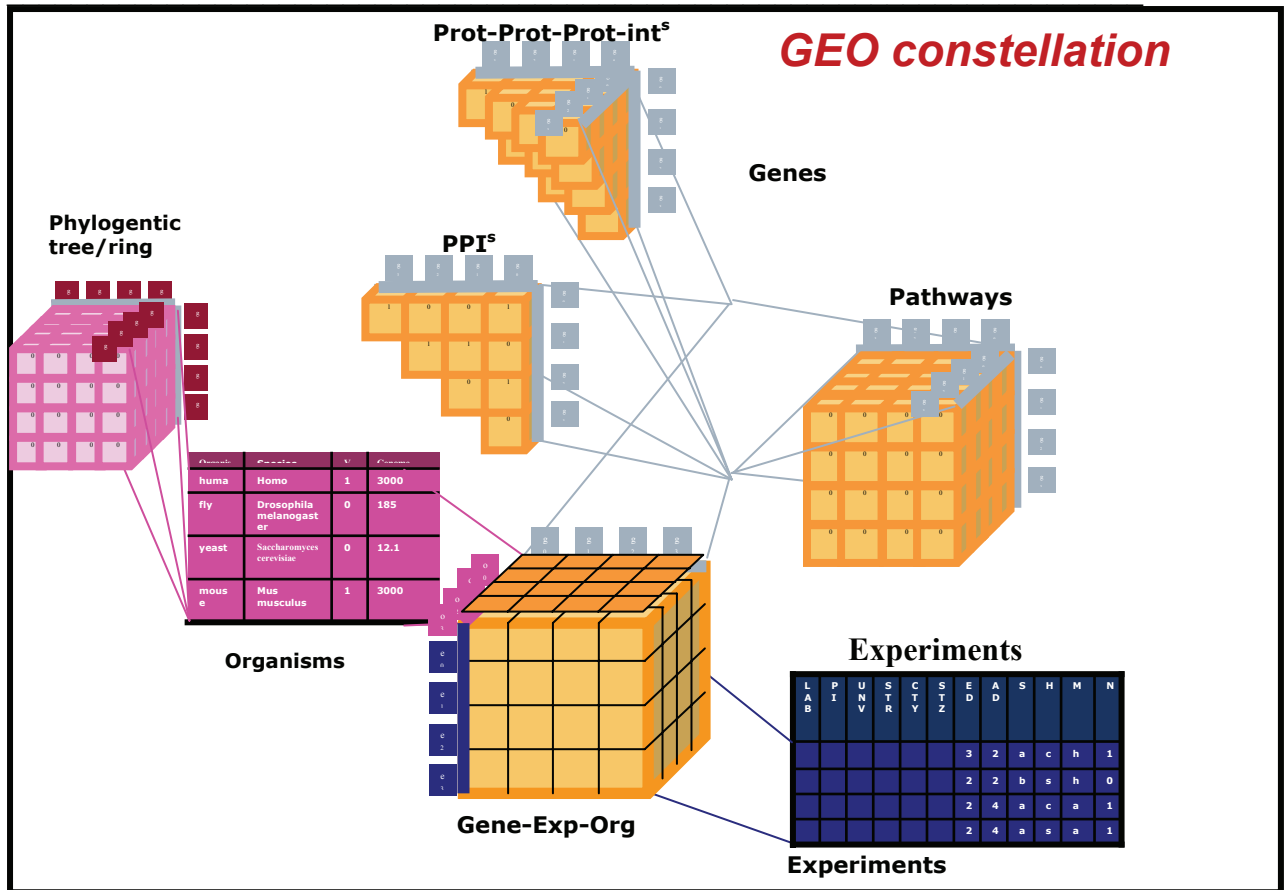
Choose a walk of  $R$ . (which converts  $P_{f,S}$  from a bit map to a bit vector).

Equi-width partition  $P_{f,S}$  along the walk with segment size,  $s$  ( $s$  is called the leaf size. We note that the last leaf segment can be shorter than  $s$ ).

Eliminate and mask to 0, all pure-zero segments (It is called the existential mask, EM, or the NotPure0 mask. It is initiated to all 1 bits and then bits with position numbers corresponding to pure-zero segments are flipped to zero. EM stands for Existential Mask ( $\exists$  a 1 bit)).

Eliminate and mask to 1, all pure-one segments (it is called the universal mask, UM, or the Pure1 mask. It is initiated to all 0 bits and then bits with position numbers corresponding to pure-one segments are flipped to one. UM stands for Universal Mask (universally 1-bits)).

The sense in which this is a tree becomes clear (e.g., UM at its root and the Mixed (not pure zero and not pure one) segments as the second level leaves). There is no need for the EM mask since all the information in it is captured by eliminating the pure zero leaves (even though they are not pure one). Therefore, if the UM bit is zero, and there is no leaf, it is assumed that the leaf is a pure zero segment. Finally, note that the EM tree could just as well have been built. The UM tree, however, and then the P-tree of the complement bit vector is just the EM tree.



**Figure 2:** The gene experiment organism constellation

The compression approach in this technology is a variant of run-length compressing of bit vectors, but with the proviso that all runs are of length the same length,  $s$  (in the EM case, existential aggregation is used and in the UM case universal aggregation is used). The “same length” is not so important (in fact, not at all important) with respect to the consecutive blocks within one bit vector, but is critically important to facilitate fast processing across bit vectors. That is, the common partitioning across all bit vectors is the important issue here.

Since each leaf of the 2-level P-tree,  ${}^sP_{f,s}$  is an uncompressed bit vector of length  $s$  (except that the very

last one may be shorter), recursively, this compression scheme continues (using the same walk) with leafsize= $s_2$  giving a 3-level Universal Predicate tree (P-tree)  ${}^{s_1, s_2}P_{f,s}$ . If  $A_i$  is Real or Binary and  $f_{i,j}(x) \equiv j^{\text{th}}$  bit of  $x_i$  then  $\{ {}^*P_{f_{i,j},\{1\}} \equiv {}^*P_{i,j} \}_{j=b..0}$  are called the basic  ${}^*P$ -trees of  $A_i$ , where  ${}^*$  is the leaf size array,  $s_1..s_k$ . If  $A_i$  is Categorical and  $f_{i,a}(x)=1$  if  $x_i=a$ , else 0, then  $\{ {}^*P_{f_{i,a},\{1\}} \equiv {}^*P_{i,a} \}$  for every  $a \in R[A_i]$  are basic  ${}^*P$ -trees of  $A_i$ .

**Editorial: W. Perrizo**

**Citation: Perrizo, Bioinformatics 1(9): 351-355 (2007)**

**License statement:** This is an open-access article, which permits unrestricted use, distribution, and reproduction in any medium, for non-commercial purposes, provided the original author and source are credited.