H-0239 (H0603-002) March 5, 2006
Computer Science

IBM Research Report

Use Domain Knowledge to Improve Data Mining Performance of Very Large Datasets via Clustering

Uri Shani, Simona Cohen
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Use domain knowledge to improve data mining performance of very large datasets via clustering

By
Uri Shani and Simona Cohen
IBM Haifa Research Lab

Abstract

Data mining is a very computationally intensive task. It is not the same as data query problems where information from a data repository is queried. Data mining involves exhaustive computation to uncover information hidden in the data—information that represents patterns in this data [1]. Therefore, the task is, to a great extent, unlimited. Using statistical analysis methods, data mining tools analyze the data and compute the relationships among the attributes (also called "features") of the data, seeking strong correlations that may be evidence of new and important information [2, 3].

We present methods for using domain knowledge, particularly in the medical domain, to reduce the dataset size for further data mining analysis.

Background

The "raw material" for data mining is instances of data, each represented as a record of an individual (i.e., a person) in a certain population (e.g., patients in a given geographic area). Sometimes such record instances represent "events" for such individuals. Each record is composed of fields representing values of the attributes and features recorded in each record. A common organization of this data is as a table that can be viewed in a spreadsheet program (e.g., MS Excel ©) or a relational database (e.g., IBM DB2 ©). However, data records can also be generalized as objects and stored in other forms such as XML or other non-tabular, but rather hierarchic or other formats. The data may contain errors or missing values and may be comprised of text in which 'fields' and 'attributes' are embedded, and may include nominal or numerical types. Data may be either sparse or very dense.

When data mining tools search for hidden patterns, the problem may very quickly explode in its computational complexity, because all possible combinations among the selected data attributes must be analyzed. When different attributes behave in a similar or totally dissimilar way in the data, this is valuable information, which may be evidence of a newly discovered pattern. Statistically, one can combine similar attributes into a single (synthesized) attribute and thus reduce the data size until all synthesized attributes expose independent behaviors. This reduction helps with further analysis of the data, such as for classification problems, but in our context, this is considered a side-effect.

The high computational complexity of data mining stems from two sources: the number of features and attributes, all of whose combinations must be analyzed, and the number of records in the data instances. We propose methods to reduce this complexity by using domain knowledge to find more dependencies in the data. In particularly, we seek such methods in the medical domain.

Current data mining methods use the technique of reducing the number of records to reduce problem complexity. Such methods use clustering [4], which groups records into classes based on statistical similarity. They then use a representative record of each class further in the analysis, instead of the original records. Assuming the number of classes is smaller than the number of original records, the problem size decreases. Clustering is mostly used for ‘machine learning’, is tuned to a small, predetermined number of clusters, and is usually computationally intensive. An efficient implementation of clustering has been patented [5]. This implementation produces clusters of an initially unknown number, based on a set of parameters that indirectly control that number.

Another method of reducing data size is by random selection of a subset of the records, assuming that the statistical information in the original collection is preserved in the reduced set of this random selection.

This method is quite efficient because there is no expensive processing of data as in cluster analysis. However, this method ignores information in the data and thus is inferior to clustering and to our specific solution.

Our solution involves methods for using domain knowledge to reduce the dataset size for further data mining analysis. We focus particularly on the medical domain, because medical health records have a very large dimension (i.e., a large number of features and attributes as data record fields).

Description

Data mining algorithms, such as that described in [2] above, involve selecting groups of attributes and advancing from small groups to larger groups whose values are analyzed to discover correlations. Domain knowledge can be applied to the attributes (i.e., the names of the columns in the table), along the top row of the table where column names are specified, and with relation to their contents, along the columns of the table.

We suggest using domain knowledge as follows:

1. Organize the combinations of attributes for the analysis phase, so that data is analyzed according to prior knowledge of the value of each combination. Domain knowledge will suggest that analyzing a column of the exam date and age of the patient is less important than the patient's age combined with the level of blood sugar and smoking habits. This is an application of domain knowledge along the attributes.
2. For each column to be analyzed, reduce the variability of column data, based on knowledge of its behavior. For example, based on the normal level range of a certain attribute, one can combine all these values to a new abstract value called "normal". This preprocessing of the data and reduction of the range of values in that field also reduces the histogram for that attribute over the entire collection of instances. A more elaborate preprocessing can apply fuzzy linguistics to the data, so that, for example, age can be considered as one of {YOUNG, ADULT, OLD}. Fuzzy methods here simply use fuzzy membership functions to define the proper ranges of values. An example of using this method for query and visualization purposes is described in [5]. Reducing the number of distinct values in each attribute helps to execute the next step, changing a numeric field to nominal, which requires less computation from the statistical analysis algorithms.
3. Use clustering on the preprocessed data from Step 2, for the collection of attributes produced in Step 1, so that a classification of similar records creates a much smaller group of records than in the original collection. The smaller the range in each attribute, the larger the identified classes and the smaller the number of such classes. Each of these classes is represented in the analysis stage by a single record and a weight. The weight represents an a-priory probability of the representative record for the entire class within this population. If the size of class C is c, and the entire population size is n, than the a-priory probability of C is c/n . The collection of representative records of classes is much smaller than the original population, but also includes the a-priory probability information. The reduced size helps the execution of the combinatorial-complex analysis phase. The a-priory probability factor attached to each representative record is naturally combined into the computation algorithms. If the collection of records displays very poor dependency among the records, so that the number of classes is very large, the clustering process may result in almost no reduction in data size and thus no saving at all in the next phase. If, however, the data displays a strong dependency among the records, the number of classes is small, resulting in a strong data reduction and significant saving in the next phase. Note that the clustering algorithm can be applied off-line before the data mining starts, so it won't add an extra overhead to the data mining algorithm.

   We propose a very simple classification algorithm whose association criterion is exact equivalence among records classified to be in the same class. This can be simply done via sorting and collecting similar records, via insertion into a tree (similarly to what is done in [4], which thus can be considered a generalization of our approach), or via a hash function that counts similar records. While all methods have a worst case complexity of $n\ Log(n)$, the hashing method has the complexity of n and is thus superior. We tested these methods on a large collection of records with large number of fields and obtained very good results showing a strong reduction in the data set size, and a very effective and fast algorithm.

4. Analyze the reduced group of instances of the selected subset of attributes preprocessed in Steps 1,to 3, taking into account that each record now carries a weight to account for the size of the cluster it represents, which is also considered the a-priory probability of the related cluster as described in [4]. This analysis finds which of these attribute collections possesses or belongs to some pattern.

   Once attributes are found to be similar by having a high correlation, they can be eliminated from the collection and replaced with a representative column with a new computed range of values identifying the collection of classes found in the clustering step. This new column can be used for further analysis, but its history of being synthesized from other columns must be kept and used when reporting its relationship to other columns (attributes) in the data. Clearly, applying methods described in Step 2 on synthesized columns is not possible because the domain data will usually not be applicable to them in any of the ways described above.

Note that Steps 2 and 3 can be combined so that for each record, we apply the domain knowledge and then classify it to the appropriate cluster before processing the next record. This combination is the essence of our algorithm, as can be seen in the following diagrams.

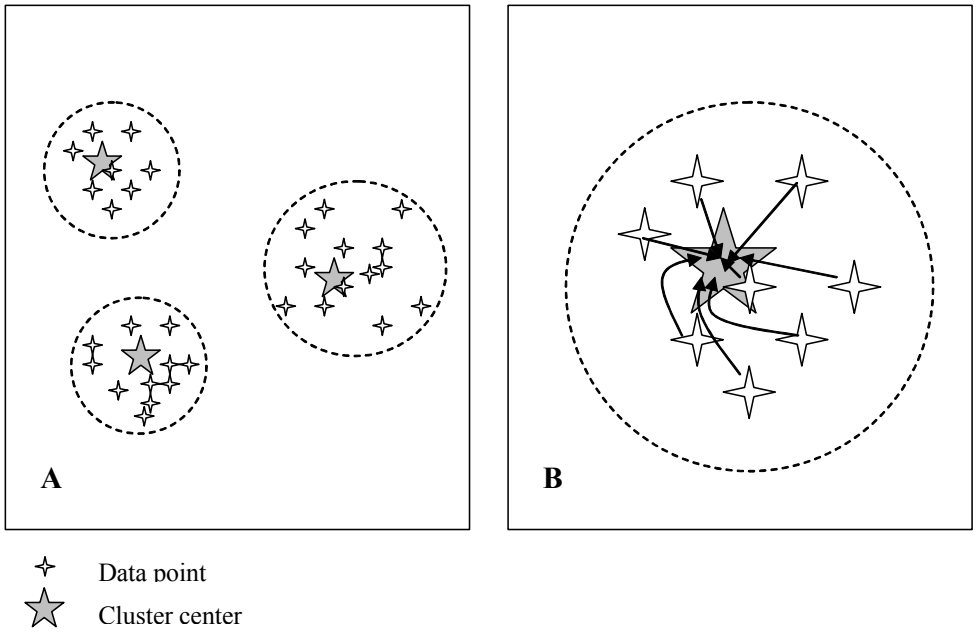

Figure 1

Diagram A in Figure 1 illustrates the analysis of a distribution of data points, resulting in three clusters that are marked with dash-line circles. Each cluster has a centroid point, which is calculated as the average of all members of the cluster (a-al the k-mean clustering algorithm—see the overview in [4]). The centroid points are new points in the problem space and represent the entire cluster of points. The cluster is tagged with a weight that is calculated as the ratio between the cluster size (the number of original points falling into it) and the entire data set size. This process is iterative and slow.

Our approach is demonstrated in diagram B, where only one of the groups of data points (from diagram A) is shown. By applying domain knowledge, we reduce the range of values in each feature, so that the numeric values can become nominal with few different values. We, in fact, map each point from the original numeric domain to a new space of much reduced resolution. Therefore, all the points in this particular cluster map into the same point in the new space of reduced resolution. It is possible that the original points will map into several points in the new space. In the latter case, this clustering process results in more clusters than the statistical process alluded to in diagram A. We now consider each of the points in the new space as representatives of all points mapped into them by this application of domain knowledge—making them representative data points of ‘clusters’. The number of original points mapped into this cluster divided by the total number of points provides a weight factor to the cluster, or its a-priori

probability, which will be used in the follow up data mining steps. The problem of identifying the clusters is now a problem of distinguishing all different points in the new space. There are very efficient solutions to this problem as will be seen later on.

Process

Our process includes several steps, some of which are detailed and some of which are left open to show that many different implementations can be used to perform them. We describe our exact-match clustering method in detail and present the results from a real collection of data records in the medical domain.

The entire process is described in Figure 2:

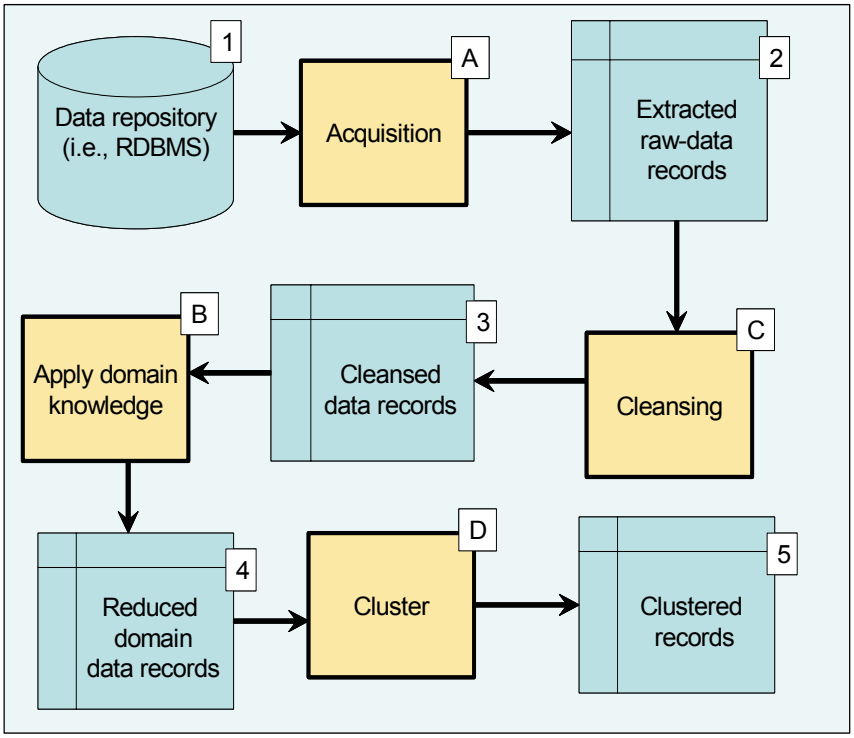

Figure 2: The process

Objects marked with numerals are data sets being processed in the pipeline, starting with a database source (1) through several intermediate formats to a final clustered records format (5). The database source can, for instance, be a relational database management system (RDBMS) such as the IBM DB2 ©. As mentioned in the *Background* section, this source can also be a hierarchical organization of XML objects from which collections of attributes are acquired. Yet, for the sake of simplicity and without lack of generality, we can think of a tabular representation of the data. So, the acquisition process (A) may be an automatic or manual process that performs a query into a database and extracts the data from it into a table format, such as, for example, CSV (Comma Separated Values). The following is an example of such a data format:

```
Alive_or_dead,Age_at_diagnosis,Congestive_heart_failure,Cardiac_arrhyth
mias,Valvular_disease,Pulmonary_circulation_disorder,Peripheral_vascula
r_disorder
1 , 6,-1, 1,-1, 1,-1
-1,72,-1, 1,-1,-1,-1
1 , 9,-1, 1,-1, 1,-1
1 , 9,-1, 1,-1, 1,-1
1 ,19,-1,-1,-1,-1,-1
1 ,48,-1,-1,-1,-1,-1
```

```
1 ,20,-1,-1,-1,-1,-1
1 ,37,-1,-1,-1,-1,-1
1 ,28,-1,-1,-1,-1,-1
1 ,66, 1, 1, 1, 1, 1
-1,58, 1, 1, 1,-1,-1
1 ,22,-1,-1,-1,-1,-1
-1,21, 1,-1,-1,-1,-1
1 , 6, 1,-1, 1,-1,-1
1 ,33,-1,-1,-1,-1,-1
1 , 6,-1, 1,-1,-1,-1
1 ,43,-1,-1,-1,-1,-1
1 ,66,-1,-1,-1,-1,-1
-1,34, 1,-1, 1,-1,-1
```

While the first line represents meta-information or names of columns, successive lines represent data corresponding to the columns' titles. In this case, data for the first column "`Alive_or_dead`" is binary: `1` or `-1`. Data for the second column, "`Age_at_diagnosis`", is numeric and multi-valued. The other columns are binary like the first one. This format can also be viewed in a spreadsheet program such as MS Excel©, as shown in Figure 3:

	A	B	C	D	E	F	G	H
1	Alive_or_d	Year_of_d	Congestive	Cardiac_a	Valvular_d	Pulmonary	Peripheral_vascular_disorder	
2	1	6	-1	1	-1	1	-1	
3	-1	72	-1	1	-1	-1	-1	
4	1	9	-1	1	-1	1	-1	
5	1	9	-1	1	-1	1	-1	
6	1	19	-1	-1	-1	-1	-1	
7	1	48	-1	-1	-1	-1	-1	
8	1	20	-1	-1	-1	-1	-1	
9	1	37	-1	-1	-1	-1	-1	
10	1	28	-1	-1	-1	-1	-1	
11	1	66	1	1	1	1	1	
12	-1	58	1	1	1	-1	-1	
13	1	22	-1	-1	-1	-1	-1	
14	-1	21	1	-1	-1	-1	-1	
15	1	6	1	-1	1	-1	-1	
16	1	33	-1	-1	-1	-1	-1	
17	1	6	-1	1	-1	-1	-1	
18	1	43	-1	-1	-1	-1	-1	
19	1	66	-1	-1	-1	-1	-1	
20	-1	34	1	-1	1	-1	-1	

Figure 3: Data records

The extracted raw data records (2) resulting from the acquisition process (A) is input to the cleansing process (B), which produces cleansed data records (3). In the cleansed data records, all data entries are worked out to a unified standard representation, such as in the above example, and all missing entries receive a proper unified value. The cleansing process is not discussed in this report, but is a necessary preprocessing phase before any data mining can occur. This process can be unified with the next process, applying domain knowledge (C), which produces the reduced domain data records (4), because the domain knowledge can also be used to perform the data cleansing. Domain knowledge is applied to data fields in order to reduce data variability, which is not necessary for the data mining stage. For instance, blood pressure and other medical measurements can be divided into several value ranges and classified as "Low",

"Normal", and "High", or in numerical values: -1, 0, and 1. More divisions can also apply but the end result is a field with far fewer values than measured accurately by the laboratory in the original raw data record. We do not discuss the method of applying this domain knowledge, which can be done automatically or manually. The automatic application of domain knowledge is very sensitive and relies on a program that can clearly parse and apply the proper semantics to the data fields, tying them to ontologies [5], for example, those in which relations among different terms of the domain are defined. The reduced domain data records (4) are input to the clustering process (D), which produces the clustered records (5). The clustered records consist of subsets of the original fields in the input to process (C); those that are relevant to data mining and also based on the domain knowledge. For example, the result of clustering the fields c-d-e-f-g in the spread sheet example above (i.e., the fields "`Congestive_heart_failure`" to "`Peripheral_vascular_disorder`"), which are "normalized" to "`CONGESTTIVE`" and "`PERIPHERAL`" respectively, would appear as in Figure 4 (displayed via a spreadsheet program):

	A	B	C	D	E	F
1	P	CONGESTIVE	CARDIAC_AR	VALVULAR_D	PULMONARY_	PERIPHERAL
2	0.47368422	-1	-1	-1	-1	-1
3	0.10526316	-1	1	-1	-1	1
4	0.15789473	-1	1	-1	1	-1
5	0.05263158	1	-1	-1	-1	-1
6	0.10526316	1	-1	1	-1	-1
7	0.05263158	1	1	1	-1	-1
8	0.05263158	1	1	1	1	1

Figure 4: Reduced domain data records

The result is a new set of records, representing only seven unique value combinations out of the original nineteen records. This provides a 19:7 = 2.7 reduction in problem size, but adds the "P" field for the a-priori probability of each record, which is computed as the ratio between the number of instances in the respective class and the total number of original instances. The respective class consists of all records with same field values. A data mining program analyzing this collection of records will work faster than on the original one.

An important feature of this process is that it is a one-pass process and thus "pipeline-ready". This means that records can be pushed through a pipeline of the A-B-C-D processing units described above without the need to accumulate a file of the entire set of records at the end of each step. Such pipeline architecture also contributes to unifying the cleansing (B) and space reduction through domain knowledge (C) processes, as well as combining the domain knowledge (C) and clustering processes (D).

Results

Figures 5 and 6 present the results of clustering a collection of 228,157 records, performing clustering repeatedly, each time over a larger set of fields, all of which are cleansed and preprocessed to have a binary range of values 1 and -1 as in the example above. In Figure 5, we plot the number of clustered classes logarithmically (base 2) versus the number of fields considered in the clustering process. The line represents a theoretical number for the possible classes, which is simply obtained by multiplying the range size of all considered fields, which in our case is a power of 2. This results in a straight line in the logarithmic scale. However, in reality, the number of classes is bound by the number of instances, which in our case is 228,157, where $\text{Log}_2(228{,}157) = 17.8$—the ----- line in the chart (for data size). Moreover, not all combinations of field values are present, so the number of classes is only a fraction of the total number of instances even for a large number of fields. This is plotted in the line.

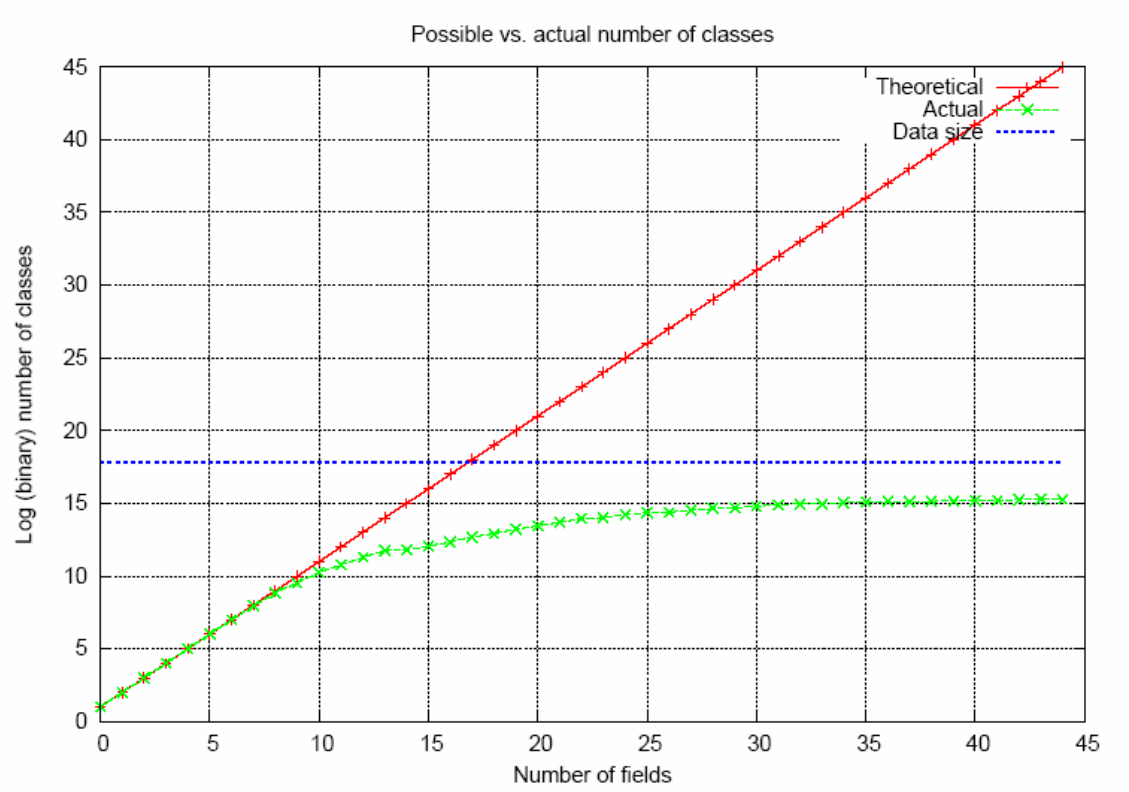

Figure 5

Figure 6 plots the gain in reducing the problem complexity. This is, in fact, the inverse of the graph above, because the problem reduction factor is the size of the original records collection divided by the size of the clustered records collection (computed above). This factor is also plotted in a logarithmic scale (base 2), showing that with few fields (up to 6-7 fields in our large dataset) the reduction is still big: 2^6 - 2^7, and even for a very large number of fields, where theory provides no gain, the actual number of classes gives a factor of 8-10, going down to a factor of 5 for 45 fields—still a useful reduction.

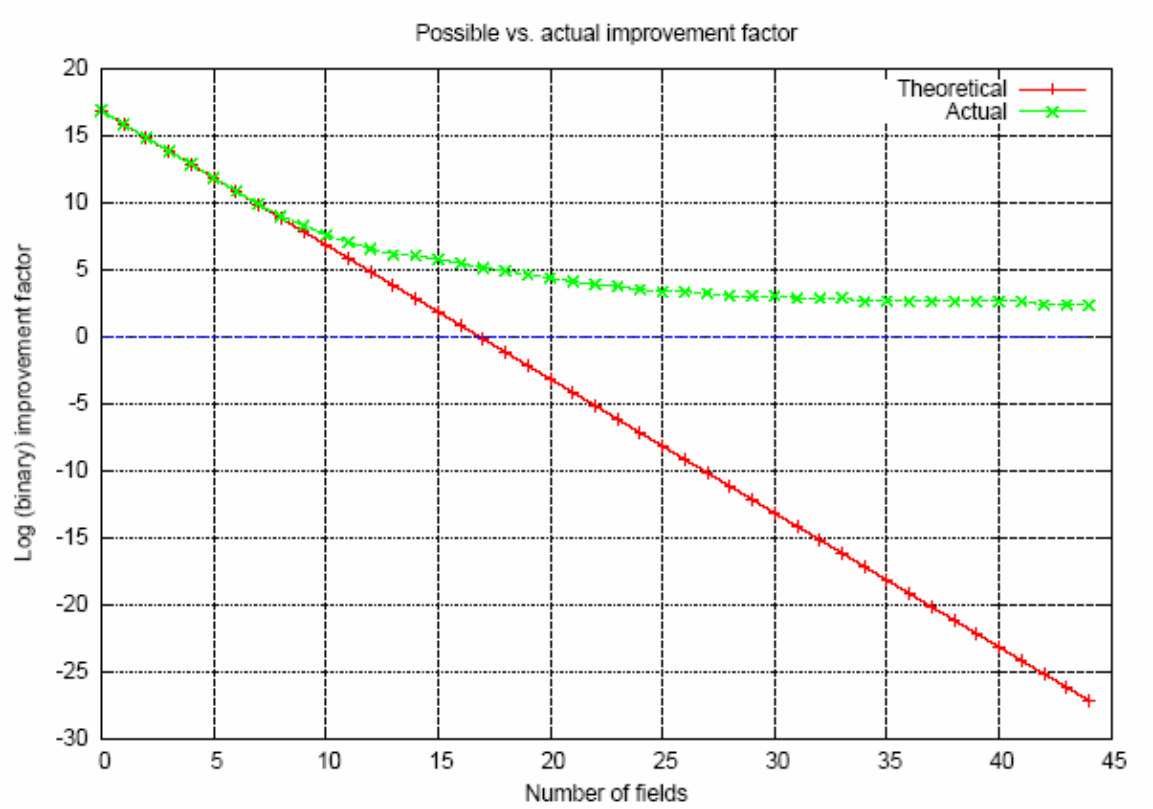

Figure 6

The entire test result plotted above in Figures 5 and 6 is presented in Figure 7 as a spreadsheet table in which we also show the runtime (in milliseconds), when processing on a 1600 MHz Intel Pentium processor PC laptop computer.

	A	B	C	D	E	F	G	H	I
1	#	Attribute name	Cardinalit	Theoretical limit	Classes	Th. factor	ACt. factor	Size	time(ms)
2	1	VALVULAR_D	2	2	2	114078.5	114078	228157	151
3	2	PULMONARY_	2	4	4	57039.25	57039	228157	90
4	3	PERIPHERAL	2	8	8	28519.625	28519	228157	100
5	4	HYPERTENSI	2	16	16	14259.8125	14259	228157	100
6	5	PARALYSIS	2	32	32	7129.9062	7129	228157	120
7	6	OTHER_NEUR	2	64	64	3564.9531	3564	228157	130
8	7	CHRONIC_PU	2	128	127	1782.4766	1796	228157	181
9	8	DIABETES_U	2	256	248	891.2383	919	228157	150
10	9	DIABETES_C	2	512	465	445.61914	490	228157	161
11	10	HYPOTHYROI	2	1024	742	222.80957	307	228157	180
12	11	RENAL_FAIL	2	2048	1222	111.404785	186	228157	201
13	12	LIVER_DISE	2	4096	1772	55.702393	128	228157	200
14	13	PEPTIC_ULC	2	8192	2484	27.851196	91	228157	210
15	14	HIV_AND_AI	2	16384	3420	13.925598	66	228157	230
16	15	LYMPHOMA	2	32768	3585	6.962799	63	228157	230
17	16	METASTATIC	2	65536	4267	3.4813995	53	228157	251
18	17	SOLID_TUMO	2	131072	5190	1.7406998	43	228157	280
19	18	RHEUMATOID	2	262144	6662	0.8703499	34	228157	280
20	19	COAGULOPAT	2	524288	7763	0.43517494	29	228157	290
21	20	OBESITY	2	1048576	9464	0.21758747	24	228157	311
22	21	WEIGHT_LOS	2	2097152	11143	0.108793736	20	228157	331
23	22	FLUID_AND_	2	4194304	13267	0.054396868	17	228157	351
24	23	BLOOD_LOSS	2	8388608	15865	0.027198434	14	228157	481
25	24	DEFICIENCY	2	1.68E+07	16689	0.013599217	13	228157	380
26	25	ALCOHOL_AB	2	3.36E+07	19369	0.006799609	11	228157	400
27	26	DRUG_ABUSE	2	6.71E+07	20810	0.003399804	10	228157	410
28	27	PSYCHOSES	2	1.34E+08	21513	0.001699902	10	228157	551
29	28	DEPRESSION	2	2.68E+08	23626	8.50E-04	9	228157	451
30	29	DIAGNOSTIC	2	5.37E+08	25715	4.25E-04	8	228157	471
31	30	BLOOD_TRAN	2	1.07E+09	26353	2.12E-04	8	228157	481
32	31	PHYSICAL_T	2	2.15E+09	28390	1.06E-04	8	228157	491
33	32	UPPER_GAST	2	4.29E+09	30703	5.31E-05	7	228157	521
34	33	TRACHEOSCO	2	8.59E+09	31944	2.66E-05	7	228157	661
35	34	DIAGNOSTIC	2	1.72E+10	32163	1.33E-05	7	228157	541
36	35	ELECTROCAR	2	3.44E+10	33637	6.64E-06	6	228157	561
37	36	CANCER_CHE	2	6.87E+10	35354	3.32E-06	6	228157	570
38	37	LOBECTOMY_	2	1.37E+11	35546	1.66E-06	6	228157	581
39	38	ENTERAL_AN	2	2.75E+11	35698	8.30E-07	6	228157	591
40	39	RESPIRATOR	2	5.50E+11	36108	4.15E-07	6	228157	591
41	40	HEMODIALYS	2	1.10E+12	36779	2.08E-07	6	228157	591
42	41	MAGNETIC_R	2	2.20E+12	36867	1.04E-07	6	228157	741
43	42	COMPUTERIZ	2	4.40E+12	37817	5.19E-08	6	228157	611
44	43	SKIN_GRAFT	2	8.80E+12	39306	2.59E-08	5	228157	772
45	44	CT_SCAN_CH	2	1.76E+13	39598	1.30E-08	5	228157	641
46	49	ECHOCARDIO	2	3.52E+13	39842	6.48E-09	5	228157	772

Figure 7 Execution results

As can be seen in column G (Act factor), the actual factor is still meaningful (5) even for a large number of fields where the theoretical factor is negligent. For 17-18 fields, which is practically significant for a data mining search, the theoretical factor is about 1 (which is useless), however the actual factors we achieve are 43 - 34, which is more than one order of magnitude better.

The last column "I" presents the run-time statistics, which increase linearly with the number of fields used for clustering. This is a result of the need to compare longer lists of values for our exact clustering algorithm. However, these are all below one second of elapsed time for processing more than 228 thousand records. Current data mining activity works in the order of minutes and hours of elapsed time, so this single second preprocessing, which can provide an order of magnitude of data reduction, is worth its time many, many times over.

Implementation

In the initial sections, we discussed two possible algorithms, but the results described above were achieved by the hashing approach, implemented with the Java [7] Hashtable class. All input records are processed and normalized into a comma-separated string without spaces, of all field values in each record. When inserted into the hash-table, all exactly identical strings are hashed to the same value, so that their number can be computed and associated with their hashing value. When all records have been processed, all hashed strings are retrieved from the hash-table, each representing a class, taking their count as a "weight" of that class, used to compute the a-priory probability of that class by dividing that number by the total number of records.

More pseudo-formally:

```
# create a hash-table
let hashTable be a hash_table;

# classify all records after normalizing them using hashTable
let total := 0;
for each record in input do
      set total := total + 1;
      let N be a normalized representation of this record;
      store N in the hashTable;
      if (N is not already in the hadhTable) do
            associate N with the value 1 in hashTable;
      else do
            retrieve cnt as the associated number with N in hashTable;
            set cnt := cnt +1;
            associate N with cnt in hashTable;
      end-if
end-for

# Now retrieve all classes and assign them with a-priory probabilities
let keys be the list of keys in hashTable;
for each key in keys do:
      let cnt be the number associate with key in hashTable;
      let float p := cnt / total;
      report p and key as a record in output;
end-for
```

We do not describe the implementation of the hash-table because this is well known in the literature. The sorting alternative described above is equivalent to a hierarchical insertion of records into a binary search tree and was found inferior to this method. Note that sorting algorithms may be much less efficient in general and in extreme cases, compared with the hashing method we used. For instance, the Java *Sort.quickSort()* method works for hundreds of seconds to “sort” a trivial list of 60,000 identical elements. It also takes a lot of time when sorting a list of many identical elements, such as in a collection consisting of only a few classes, which represent the most useful situations for our purpose. Contrary to that, hashing works equally efficiently in general for all situations. The patent in [6] uses a hierarchical structure to collect records into clusters and runs in the order of minutes for the same size of collection of records as in our example.

References

1. Watanabe defines pattern as "... the opposite of chaos; it is an entity, vaguely defined, that could be given a name." in W. Watanabe, "*Pattern recognition: Human and mechanical,*" Wiley 1985.
2. B. Robson, "*Clinical and pharmacogenomic data mining: 1. The generalization theory of expected information and application to the development of tools,*" J. of Proteome Res., 203, pp.283-301.
3. B. Robson, "*Clinical and pharmacogenomic data mining: 2. A simple method for the combination of information from associations and multivariances to facilitate analysis, decision and design in clinical research and practice,*" J. of Proteome Res., 203, pp.283-301.
4. Hartigan, J. A., "*Clustering algorithms*," John Wiley, New York, 1975.
5. B. Chandrasekaram, John R. Josephson, and V. Richard Benjamins, "*What are ontologies, and why do we need them?*", IEEE Intelligent Systems, Jan.-Feb. 1999, pp.20-26.
6. Tian Zhang, Raghu Ramakrishnan., Miron Livny, Wisconsin Alumni Research Foundation, Madison, Wis., '*Method and system for data clustering for very large databases,*' US Patent 5,832,182, Nov 3, 1998.
7. David Flanagan, "Java in a Nutshell," O'Reilley, 2002 (4th edition).