Data Min Knowl Disc (2007) 15:3–20
DOI 10.1007/s10618-006-0057-3

On data mining, compression, and Kolmogorov complexity

Christos Faloutsos · Vasileios Megalooikonomou

Received: 19 May 2006 / Accepted: 20 September 2006 / Published online: 27 January 2007
Springer Science+Business Media, LLC 2007

Abstract Will we ever have a theory of data mining analogous to the relational algebra in databases? Why do we have so many clearly different clustering algorithms? Could data mining be automated? We show that the answer to all these questions is negative, because data mining is closely related to compression and Kolmogorov complexity; and the latter is undecidable. Therefore, data mining will always be an art, where our goal will be to find better models (patterns) that fit our datasets as best as possible.

Keywords Data mining · Compression · Kolmogorov complexity · Clustering · Classification · Forecasting · Outliers

1 Introduction

Will we ever have a theory of data mining, in the same way we have the relational algebra and calculus (Codd 1971) (or Ramakrishnan and Gehrke 2002) for databases? Will we ever be able to automate the discovery of patterns, clus-

Responsible editor: Geoffrey Webb

C. Faloutsos (✉)
School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213-3891, USA
e-mail: christos@cs.cmu.edu

V. Megalooikonomou
Department of Computer and Information Sciences, Temple University, 314 Wachman Hall, 1805 N. Broad Street, Philadelphia, PA 19122, USA
e-mail: vasilis@cis.temple.edu

ters, correlations? The first of these questions was raised in several recent KDD panels.

We argue that data mining, in its general case, is equivalent to compression and Kolmogorov complexity, which is undecidable. We will focus on both versions of data mining: supervised learning (classification, regression) and unsupervised learning (pattern discovery, clustering). Additional data mining tasks like forecasting and outlier detection are all closely related to the ones above. For forecasting in a numerical time sequence, we need to do auto-regression and express the future as a function of the past. For outliers in a cloud of n-dimensional data points, once we are able to somehow summarize (that is, compress, or equivalently, describe succinctly) the vast majority of those points we label the remainders as outliers.

Let us start with some examples to illustrate the concepts of Kolmogorov complexity.

Example 1 *(*Outlier) Find the outlier in a cloud of 2-d points, as shown in Fig. 1.

Looking at the linear-linear scatter-plot of Fig. 1(a), we would argue that the point labeled X is the outlier, since it is distant from all the other points. Point X is at (1024, 1024). Figure 1(b) shows the same dataset in log-log scales. Now, point Y seems like the outlier at (17, 17); all the other points are equispaced, because their coordinates are powers of 2: (1, 1), (2, 2), ..., $(2^i, 2^i)$, for $1 \leq i \leq 10$.

Once we are told that almost all the data points are at powers of 2, most people would tend to consider point Y at (17, 17) as the outlier. Why?

Some people may justify their choice of Y, saying that the log-log scatter-plot of Fig. 1(b) reveals more structure than its linear-linear counterpart; since point Y violates this structure, that is the one that is the outlier.

This answer is still qualitative, but it can bring us one step closer to our final destination. How do humans measure ‘structure’ and ‘order’? This is where compression and Kolmogorov complexity help: The log-log scatter-plot is easier to describe: it consists of equi-spaced points, except for Y. Taking logarithms helped us discover a pattern that was not obvious in the linear-linear plot.

A real example is shown in Fig. 2, in which the area is plotted against the population of 235 countries of the world in 2001. Again, in linear-linear scales, we see no major pattern except for a large cluster near the origin and a few countries as outliers, that is, countries with large population or large area, or both: China, India, Russia, Canada, the United States, China, Brazil, Australia. In log-log scales, there is a strong correlation between (log) area and (log) population, which makes sense; in that case the outliers are the densely populated countries: Macau, with 21,606 persons per square mile (p/sqm), Monaco with 16,000 p/sqm, Singapore and Hong Kong with 6,600 p/sqm; and even larger countries, like Bangladesh with 911 p/sqm, Japan with 335 p/sqm. On the other side, outliers are also the sparse ones (Australia, with 2 p/sqm, Mongolia, with 1 p/sqm). The plot also reveals some unexpected outliers: the country with the smallest area is Vatican City (0.44 square miles), and the country with the

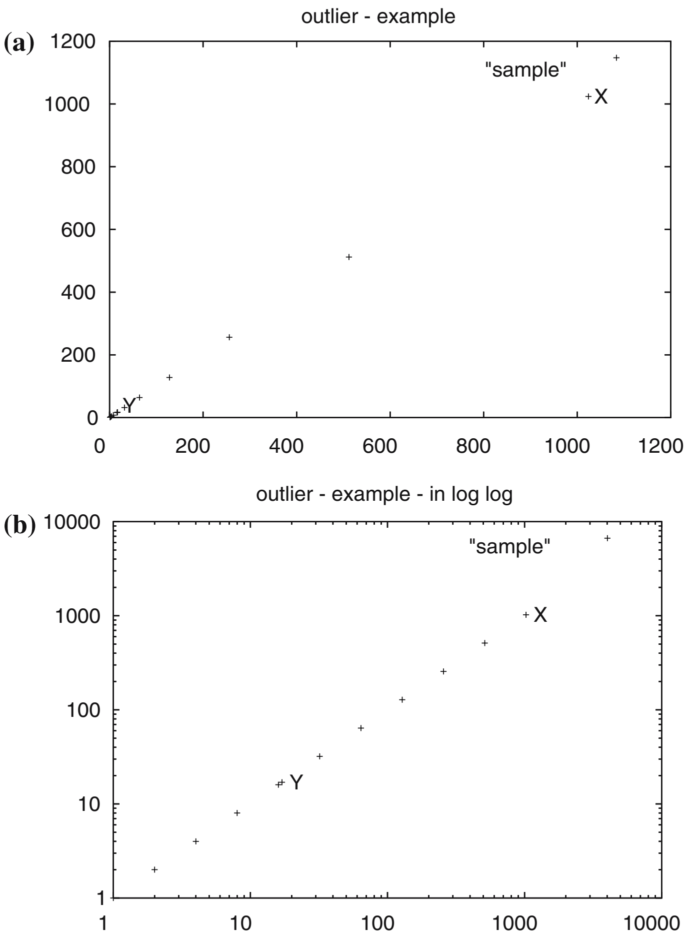


Fig. 1 What is the outlier: (**a**) scatter-plot in linear scales and (**b**) in log-log scales

smallest population is the Pitcairn Islands, with 47 people(!), but with its own web domain name (see www.nic.pn).

As we discuss very soon, almost all data mining operations are closely related to compression, Kolmogorov complexity, and conditional Kolmogorov complexity. Specifically, we have:

- *Unsupervised learning*, such as clustering, would benefit from (lossless) compression. For example, for a cloud of N points in d dimensions, a clustering algorithm would like to find the best-fitting pattern that describes our cloud of points. Typical methods include k-means (which tries to fit k multi-variate Gaussians) and BIRCH (Zhang et al. 1996) (which tries to fit spheres).
- *Outliers:* closely related to unsupervised learning. Following up on the clustering example above, outliers could be considered as the points that belong to singleton clusters. Similarly, points that belong to micro-clusters

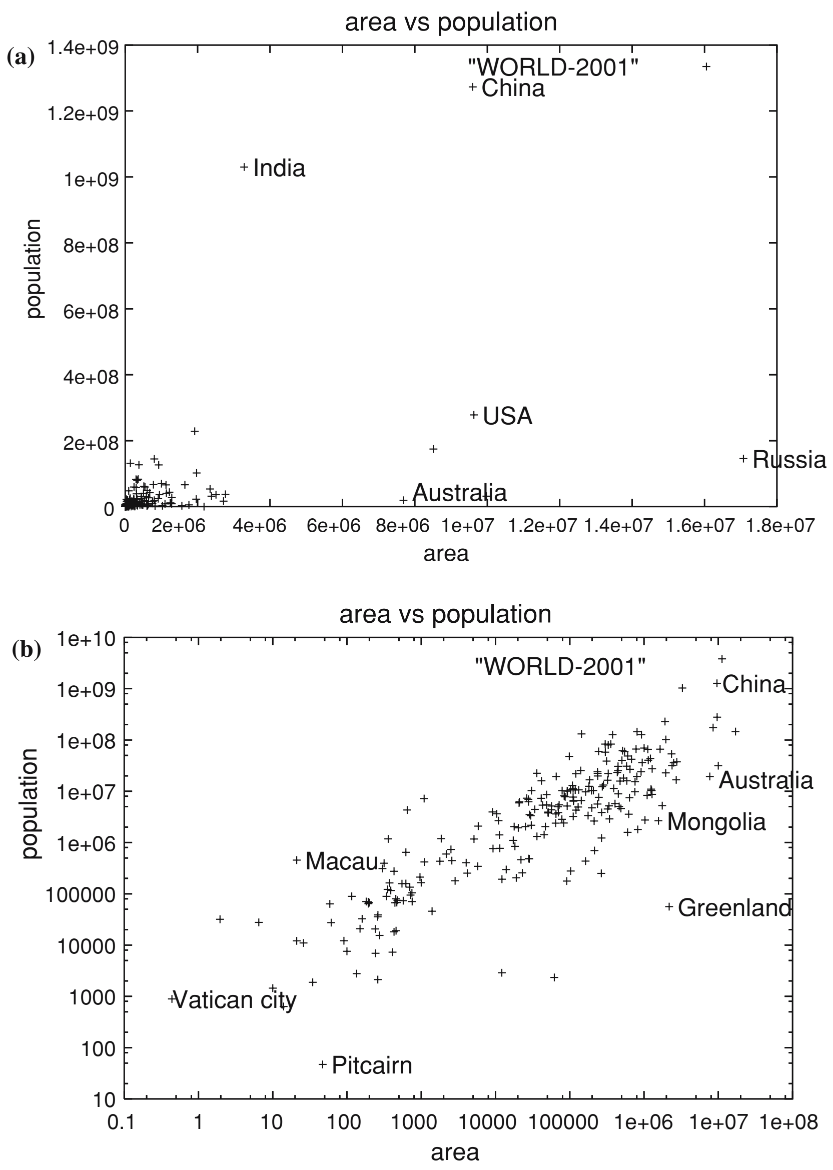


Fig. 2 What is the outlier—area versus population, for all countries (2001): (**a**) scatter-plot in linear scales and (**b**) in log-log scales

(Papadimitriou et al. 2003) are also suspicious. The fact that there is not yet a perfect definition for outliers and micro-clusters is not a surprise; as we argue later, it is also related to the (uncomputable) Kolmogorov complexity.

- *Classification* and in general, *supervised learning*, is closely related to conditional compression. Suppose we are given N patient records, with n attributes each, like body height, body weight, age, gender; and one more

attribute, the class label (e.g., 'healthy' or not). One way of envisioning classification trees is as a means to compress the class labels of the training set given the other n attributes. That is, the tree performs a conditional compression of the class labels given all the other attribute values for all records.
- *Distance function definition*: Given a set of data items (e.g., patient records, or strings, or images, etc.), how can we find a good distance function between a pair of such items? It turns out that this is also related to conditional compression (given the first patient record, what is the minimum number of changes we need to make, to transform it into the second record).

Next, we give a more formal description of the above observations. The rest of this paper is organized as follows: Section 2 gives the main idea and theorems. Section 2.2 lists some case studies where compression and Kolmogorov complexity led to successful data mining concepts and algorithms. Section 3 discusses some related issues. Section 4 gives our "practitioners' guide". Section 5 gives the conclusions.

2 Main idea—Theorems and case studies

All of our arguments are based on the concept of Kolmogorov complexity (see Li and Vitanyi 1997 for a complete treatment). Let us illustrate it with a few examples. We give examples with bit strings for the sake of illustration. Suppose we have the following bit strings, all of the same length, 1 million (1 M) bits long.

- b_1 = '0000. . .000'
- b_2 = '010101. . .0101'.
- b_3: the first million bits of fractional extension of $\sqrt{2}$.
- b_4: the first million bits of fractional extension of π.
- b_5: a binary string corresponding to a sequence of one million tosses with a fair coin where heads are represented by "1" and tails by "0".

Which is more 'complex'? How would we measure complexity? Which string has more information? Clearly, b_1 is simple—just zeros. b_2 is slightly more complex, while b_3 and b_4 are more complex. What about b_5? b_5 can be any of b_1, b_2, b_3, b_4; in fact, any of all possible binary strings. Each such string of million zeros and ones can be generated with equal probability: 2^{-1M}. What if we restrict b_5 to include only binary strings where the number of ones is near to $1M/2$, the number of occurences of blocks "00" is close to $1M/4$, and so on?

Complexity is intuitively measured by the number of English words we need to describe each bit string. We can measure the information of such a bit string in terms of its *description length*. The Kolmogorov complexity, invented by R. J. Solomonoff (1964), A. N. Kolmogorov (1965), and G. J. Chaitin (1969), formalizes this idea of complexity: the complexity of a bit string is the *shortest Universal Turing Machine* (UTM) program that can generate the desired string.

Definition 1 The conditional Kolmogorov complexity $K(x|y)$ of a string x under condition y is equal to $K_U(x|y)$ defined relative to the universal Turing machine U as

$$K_U(x|y) = \min\{|p| : p \in 0, 1^* \quad \textit{and} \quad U(p, y) = x\}$$

where p is a description of x (i.e., a program) and U outputs x when given as input the program p and some extra information y to help generate x (Li and Vitanyi 1997).

Definition 2 The unconditional Kolmogorov complexity of a string x is defined as $K(x) = K(x|\epsilon)$, where ϵ is the empty string ($|\epsilon| = 0$) (Li and Vitanyi 1997).

The definition of Kolmogorov complexity agrees with our intuition: b_1 should need a simple, short UTM program, b_2 should need one slightly more complex, b_3 should need one that, for example, implements Newton's iterative method that computes the square root of a number, and b_4 should need a UTM program that implements, say, Ramanujan's formula for π, up to 1 million bits. This is probably the fastest converging formula for π (Schroeder 1991).

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{n=0}^{\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 396^n}. \tag{1}$$

In fact, π is an example of an infinite sequence that contains O(1) information since the digits of this number can be produced forever by a fixed short program. For b_5, if we impose the restriction we mentioned earlier, an incompressible or Kolmogorov random string (Li and Vitanyi 1997; Megalooikonomou 1997) is produced.

Definition 3 A binary string is incompressible if $K(x) \geq |x|$ where $K(x)$ is the (unconditional) Kolmogorov complexity of x (Li and Vitanyi 1997).

Martin-Löf (1966) has shown that incompressible strings pass all effective statistical tests for randomness, so we will call incompressible strings random strings. However, not all traditionally random strings are Kolmogorov random. π's digits are random according to traditional randomness tests, but quite regular in terms of algorithmic randomness. Moreover, it turns out that almost all strings of a given length n are Kolmogorov random, and thus, incompressible (Li and Vitanyi 1997). The motivation for introducing Kolmogorov complexity is that phenomena with shorter explanations are typically less complex than phenomena with longer explanations.

On the surface, we are done: given a string (and, in general, a dataset represented as a bit string), estimate its Kolmogorov complexity. If the string/dataset obeys some patterns and correlations (like the obvious pattern of b_1, or the subtler pattern of b_4), its Kolmogorov complexity will be low. We could study the UTM that achieves the lowest complexity, and reverse-engineer the patterns it exploited. This would have been great, except that it is impossible.

2.1 Theorems

Theorem 1 (Undecidability of Kolmogorov Complexity (Cover and Thomas 1991) (Sect. 7.7)) *The Kolmogorov complexity of an arbitrary string x, $K(x)$, is undecidable (or non-computable).*

A short argument is the following: To determine $K(x)$ we must execute every program and collect all that stop and compute x, choosing the one that has the smallest size. However, from the halting problem, we cannot decide if a program stops or not. The fact that the function $K()$ is not computable can be formally proved by contradiction.

So, it is not possible to compute $K(x)$ (Cover and Thomas 1991; Li and Vitanyi 1997). We can, however, approximate it. A reasonable upper bound on Kolmogorov complexity is the Lempel-Ziv encoding (Ziv and Lempel 1978).

Let X_i be a stationary, ergodic process over a finite discrete sized alphabet. Let $l_{LZ}(x)$ be the Lempel-Ziv codeword length of a string x of length n, where $x = x_1, x_2, \ldots, x_n$. The following theorem holds:

Theorem 2 (Kolmogorov Complexity and Lempel-Ziv encoding (Cover and Thomas 1991))

$$K(x|n) \le l_{LZ}(x) + c, \ \lim_{n\to\infty}(1/n)l_{LZ}(x) = (1/n)K(x|n)$$

where $K(x|n)$ is the Kolmogorov complexity of string x given n.

The Kolmogorov complexity of a sequence of random variables is also related to its entropy. In general, the expected value of Kolmogorov complexity of a random sequence is close to the Shannon entropy:

Theorem 3 (Kolmogorov Complexity and Entropy (Cover and Thomas 1991) (Sect. 7.3)) *Let the stochastic process $\{X_i\}$ be drawn i.i.d. according to the probability mass function $f(x), x \in \mathcal{X}$, where $\mathcal{X}$ is a finite alphabet. Let $f(x^n) = \prod_{i=1}^{n} f(x_i)$. Then there exists a constant c such that:*

$$H(X) \le \frac{1}{n}\sum_{x^n} f(x^n)K(x^n|n) \le H(X) + \frac{|\mathcal{X}|\log n}{n} + \frac{c}{n}$$

for all n. Thus

$$E\frac{1}{n}K(X^n|n) \to H(X).$$

2.2 Case studies

Here we survey older work on machine learning and data mining, where compression, entropy, and Kolmogorov complexity play an important role. We start our discussion with supervised learning, and continue with unsupervised learning, distance function design and other, more exotic applications such as graph mining and spatial data mining.

2.2.1 Supervised learning—classifiers

The typical way to build a classification tree needs to use a measure of homogeneity of a node in order to decide whether or not we should split it further. The typical measures are the entropy gain, and the Gini index (Mitchell 1997; Quinlan 1993). Notice that the entropy is closely related to Kolmogorov complexity (see Theorem 3), and that the Gini index is a special case of the generalized (or Renyi) entropies (see Schroeder 1991[p. 203]):

$$S_q = \frac{1}{q-1} \log \sum_{i=1}^{M} p_i^q \tag{2}$$

where S_q is the generalized entropy of order q, p_i is the probability of the i-th outcome of the random variable of interest, and M is the number of possible outcomes. Notice that for $q \to 1$ the formula reduces to Shanon's entropy, while for $q = 2$ it is a function of the sum of squares, like the Gini index.

Compression arguments, and specifically MDL (Minimum Description Length), was explicitly used in SLIQ (Mehta et al. 1996); there, a node was split only if it helped the compression of the class labels.

2.2.2 Unsupervised learning and clustering

One of the hard problems in clustering is determining k, the number of clusters. Most methods let the user determine k, or some other threshold, which indirectly decides k. A parameter-free approach to the problem is to penalize model complexity with one of the three methods: Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), or Minimum Description Length (MDL). All of these ideas are related to Kolmogorov complexity. Recent successful clustering methods along these lines include X-means (Pelleg and Moore 2000) and G-means (Hamerly and Elkan 2003).

2.2.3 Distance function design

Distance functions are very important for clustering, similarity searches and case-based reasoning. The question is, how could we design good distance functions? For numerical data (e.g., vectors, or time series), the Euclidean distance seems good, possibly after some normalization, say by the standard deviation.

However, for other data types like strings, trees, graphs, images, and audio, the Euclidean distance is not applicable or not suitable. All of the resulting distance functions essentially attempt to measure the cost of transforming the first item into the second. We can envision it as the conditional Kolmogorov complexity $K(x|y)$. Here, we provide a list of popular distance functions, which operate along these lines:

- Typed text and the string editing distance: The typical distance between two strings s_1 and s_2 is the Damerau (or Damerau-Levenshtein) distance (Damerau 1964), which is the number of insertions, deletions, and substitutions, that are needed to transform the first string into the second. The operations may have weights that depend on the application (e.g., insertions are more expensive in an OCR application; substitution of 'a' with 's' is cheaper, for typed English text on a QWERTY keyboard). The similarity to the conditional Kolmogorov complexity is striking.
- Voice, audio, and the time-warping distance: Here we want to allow for small accelerations and decelerations. The time warping distance between two numerical sequences s_1 and s_2 allows for stutters of s_1 and s_2, so that the resulting sequences match well under the Euclidean distance for example. The stutters may incur no penalty, or merely a small penalty, or they may be allowed under certain conditions. Rabiner and Juang (Rabiner and Juang 1993) provide several variations of this general theme and its applications to voice processing. The time warping distance is also very suitable for biological time series like electro-cardiograms. There is a considerable amount of research on time warping, with recent additions on indexing sequences under the time warping distance (Yi et al. (1998); Keogh (2002)). Again, the point is that the time warping distance is closely related to the conditional Kolmogorov complexity.
- Image comparisons and 'Attributed Relational Graphs' (ARGs): Given two images I_1 and I_2, say of human faces, and assuming that they have been parsed, into, say, 'eyes', 'nose', and 'mouth', a popular distance function is the Eschera-Fu distance (Eschera and Fu 1984). After representing the pieces as a graph, where every node is an object (say, 'left eye', etc), every edge carries information about the relative position of objects (angle, distance) and every node has attributes (area, perimeter, color histogram), the Eschera-Fu distance is defined as the cheapest set of editing operations that need to be done to transform one ARG to the other.
- Trees, tf-idf weighting and cosine similarity: Recently, work has been done (Megalooikonomou et al. 2006) to define the similarity of tree-like structures by using (symbolic) string representations and employing tf-idf weighting and cosine similarity. This has enabled a broadened study of the relation of Kolmogorov randomness (that can be defined for trees using representations that provide 1-1 correspondence with strings), complexity of tree structure and function. In order to apply Kolmogorov complexity arguments to objects that are not strings, typically one needs to encode the objects using strings and the encoding should contain, in some sense, information about the properties under study. For example, by employing Prüfer encoding to obtain a unique characterization string for each tree, some researchers (Megalooikonomou et al. 2006) were able to capture, in the string representation, important information regarding the structure, and in

particular, the branching patterns of the nodes, and use them for tree similarity analysis.
- Symbolic representation of signals and corresponding distance functions: Given a set of time sequences or images, one can represent them by extracting a “vocabulary” of key-subsequences (Megalooikonomou et al. 2005) or key-blocks (Zhu et al. 2002) (employing Vector Quantization, Gersho and Gray 1992), and encoding each time series or image based on the frequency of appearance of each key-subsequence or key-block. The similarities between different time series or images can then be calculated using variations of the histogram model.

All of the above methods use the spirit of conditional Kolmogorov complexity, without explicitly employing it. A formal way of explicitly using the information theoretic concept of (conditional) Kolmogorov complexity to define a sequence distance function was suggested by Li et al. (2001, 2003) followed by (Keogh et al. 2004), that proposed a novel, parameter-free distance function between time sequences. The latter suggested a clever way of approximating the Kolmogorov complexity with the Lempel-Ziv compression length.

2.2.4 Other applications

The MDL and compression view points have been applied to other settings too such as graph partitioning. For example, in the *cross-associations* method (Chakrabarti et al. 2004) the goal was to partition a graph into a “natural” number of communities. The number k of such communities was estimated by compression. In a nut-shell, the idea was to try to compress the adjacency matrix, by re-arranging the columns and rows, and by partitioning the matrix in so many row- and column-groups, so that the resulting rectangles are homogeneous and can be easily compressed.

How about forecasting? Is it also related to compression? The answer is ‘yes’: typical forecasting methods try to find repeating patterns in the past and extrapolate them in the future. For example, ARIMA assumes that there are linear correlations. In fact, ARIMA has been used to compress voice, with great success, under the name of Linear Predictive Coding (LPC) (Rabiner and Juang 1993).

3 Points of argument

There are several subtle points that are related to compression, Kolmogorov complexity, and the use of these for data mining. Here, they are presented in the form of questions and answers.

Question 1 (Lossless, or lossy compression?) *Several methods, like the JPEG image compression, are lossy. Can we extend the above arguments to lossy compression?*

Answer: Yes. Any lossy compression scheme can be turned into lossless if we encode the differences between model and reality. The sum of bytes for the lossy compression, plus the bytes for the deltas, fully agrees with the minimum description length (MDL) approach.

Question 2 *How do we encode integers in a lossless fashion?*

Answer: One scheme is the Elias codes (Elias 1975), or some other form of self-delimiting integer encoding (see Bentley et al. 1986 or Zobel et al. 1992 for more recent applications) A quick reminder is as follows: to encode, for example $(5)_{10}$, we need to encode it in binary $(101)_2$, and also to encode its length, in unary: 000, giving eventually 000 101 (without the middle blank, which is only shown for illustration). Thus $(8)_{10}$ becomes 00001000. In general, the i-th integer needs $O(\log i)$ bits to be encoded, fully agreeing with our intuition that short numbers are easier to describe.

Question 3 *How do we encode floats?*

Answer: We try to avoid them—we turn them to integers (e.g., we turn 1.03 dollars into 103 cents), and encode them as discussed before. This is a *very subtle* point: A randomly chosen real number in the range (0, 1) is incompressible with high probability! Thus, if we try to compress floating point numbers, we may run into paradoxes, since we are probably trying to compress many noise-like digits.

Question 4 (Occam's razor?) *A simple model might not always be the best*[1]*: For example, consider 1 million 2-d points uniformly distributed in the unit square. Assuming that no two of them have the same x value, we can fit a sinusoid through all of them if the frequency is high enough and the amplitude and phase are appropriately chosen. Although it is a simple model (just a sinusoid), it is not suitable for generalization.*

Answer: The compression-based counter-argument is as follows: if the points are indeed random within the unit square, then it is true that we will need only one number for the frequency of the above sinusoid. However, this frequency will be so high, that it will be too expensive to write down its digits! In short, it will have high Kolmogorov complexity. A simpler example would be to try to fit a polynomial; yes, it will fit, and it will describe our points in a single polynomial, but its degree will need to be about one million. Thus, it will be cheaper to just list the one million points.

4 Practitioner's guide—Searching for good models

The results on Kolmogorov complexity are negative, in the sense that we will never be able to automate the process of compression and subsequently of data

[1] We are thankful to Prof. Geoff Webb for bringing up this ingenious argument.

mining. Are there some positive aspects to it? And, in general, what should we do as practitioners, given that perfection is elusive?

We believe that there are two major conclusions from the Kolmogorov complexity viewpoint:

- The first is that the compression viewpoint will help us design parameter-free data mining algorithms for clustering, classification, distance function design, as we listed earlier.
- The second point is that data mining will always be an art, looking for good models—the better the model, the better our compression. Thus, we may never know whether our model is the best (even if it indeed is!), but we will know whether we are the best so far: we simply have to count the number of bytes that our compression-based scheme achieves against the bytes of the competition.

The second point is subtle, and we would like to elaborate on it.

4.1 Searching for good models—Kleiber's law

In a simplified setting, consider points along a line, as in Fig. 3. These points are generated artificially, but the goal is to mimic the behavior of metabolic rate (vertical axis) as a function of the body mass (horizontal axis) for several animal species. Unsurprisingly, the larger the animal, the higher the metabolic rate.

If our only model is a mixture of flat lines, (Fig. 3(a)) we can do a reasonable job compressing it. In fact, this is exactly what several regression systems, like CART (Breiman et al. 1984), will do. However, a careful and trained person would probably try a piece-wise linear approximation, which seems like a better model for this case (Fig. 3(b)): Small animals, say, with mass less than or equal to 30, are well described by a sloping line, while larger animals are well described by a line with smaller slope. Apparently, the metabolic rate slows down for larger animals.

And an even more careful, trained biologist, might try a power law, or equivalently, a linear approximation after taking the logarithms of both axes. It turns out that for the case of mass versus metabolic rate, this is probably the best model (see Fig. 3(c)). The line fits well in log log scales with a slope of 3/4, that is

$$y = x^{3/4} \tag{3}$$

This observation is well established in biology, and is refered to as *Kleiber's law*.

Figure 4 shows some real data, with several species, from (Mackenzie 1999), or http://universe-review.ca/R10-35-metabolic.htm. The law is indeed well obeyed and there is a reasonable explanation for this. Smaller animals dissipate heat in a different rate because their ratio of body volume ($\approx O(r^3)$ to body surface ($\approx O(r^2)$) changes with the body length r, and thus their metabolic

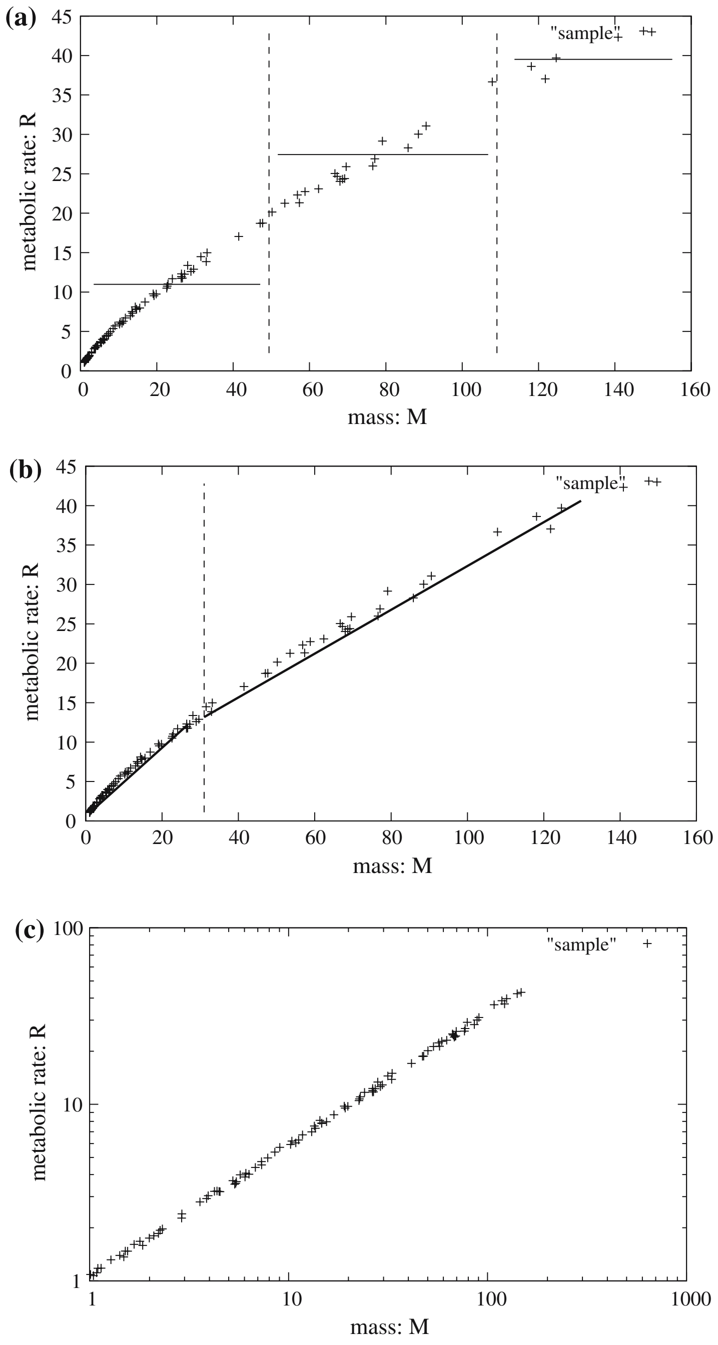
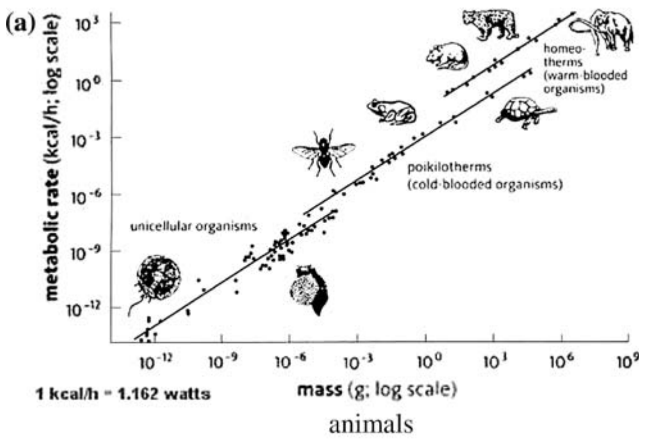


Fig. 3 Importance of good model: (**a**) piece-wise flat approximation, like the typical regression tree, (**b**) piece-wise linear approximation, and (**c**) linear approximation in log-log scales



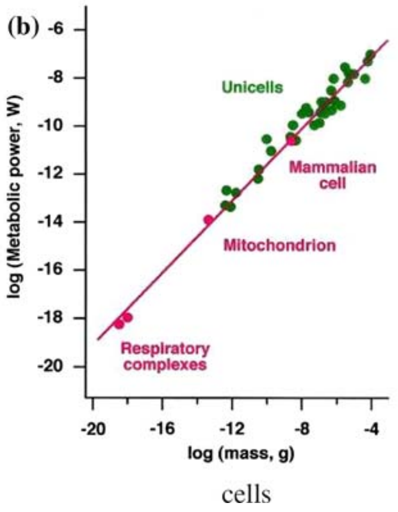


Fig. 4 Kleiber's law: $R = M^{3/4}$. Metabolic rate R versus organism body mass M in log-log scales: (**a**) animal species; (**b**) cells

rate is different. In fact, there are many related laws in biology where the fourth root appears often.

Our point is that *with better models, we can do better compression*. The art is to find these better models without going through an exhaustive search over the space of possible models and parameters for a given dataset.

4.2 Searching for good models - fractals

Looking for better models can be very tricky and subtle. Here we describe an example that is based on fractals and self-similarity. Consider a cloud of points according to the Sierpinski triangle (one of the most famous *fractals*) shown in

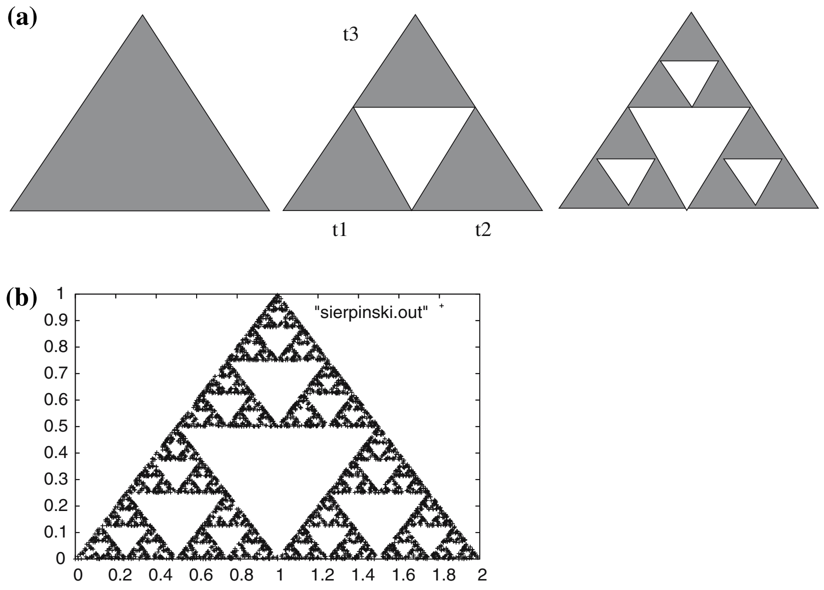


Fig. 5 The Sierpinski triangle: (**a**) its recursive definition and (**b**) some sample points from it

Fig. 5. Figure 5(a) shows its recursive definition, which results in an infinite set of points, and Fig. 5(b) shows a finite sample of 5,000 points of the Sierpinski triangle. Suppose we want to do data mining, to find clusters, and to compress it. How many clusters are there? 3? 9? 5,000? The question is to find a good model, and clearly, Gaussians, spheres, and k-means are not suitable. The reason is that the triangle consists of three miniature replicas of itself, with the replicas having holes in their middles, recursively down to infinitesimal scales. How should we compress it and cluster it?

Intuitively, its Kolmogorov complexity should be low, since we manage to describe it in a few English sentences. Barnsley and his colleagues (Barnsley and Sloan 1988; Barnsley 1988) developed the ingenious method of "*Iterated function systems*", and indeed manage to generate such triangles as well as any other self-similar shapes with a one-page long program [2] and about 20 parameters. The IFS method shows that we only need to describe the three contracting transformations that turn the original Sierpinski triangle into its 3 miniature versions ($t1$, $t2$, $t3$, in Figure 5(a), middle). These three transformations each need six (rational) numbers, and that is enough to describe and construct, the Sierpinski triangle, as well as any finite or infinite sample of its points. The details are too long and involved to describe here (please refer to the original paper (Barnsley and Sloan 1988) or the corresponding book (Barnsley 1988)).

Without getting into much detail, a conjecture is that a cloud of points with higher fractal dimension will be harder to compress (i.e., the model that

[2] For a 'C' version of their program, visit www.cs.cmu.edu/c̃hristos/SRC/ifs.tar

describes it will be more complex) than a cloud of points with lower fractal dimension (Kumaraswamy et al. 2004). In other words, the fractal dimension of a dataset seems related to its Kolmogorov complexity.

5 Conclusions

We argued that several core aspects of Data Mining, like classification, clustering, forecasting, outlier detection, are all essentially related to compression. The negative result is that optimal compression is undecidable, being equivalent to the estimation of Kolmogorov complexity.

However, from the practical point of view, there are also positive aspects and guidelines we can derive from it:

- *Compression for parameter free data mining:* The compression view point (MDL etc) leads to elegant, parameter-free solutions to clustering, graph partitioning, distance-function design, etc.
- *Data Mining will always be an art*, and specifically, the art for looking for better models. We are shooting for optimal (or near-optimal) compression of the given dataset, under a set of assumptions/models; the better the model, the better the compression! The big question is what are good models that we should try to fit: Gaussians, Poisson, Pareto, fractals, power laws, or something else yet to be discovered.

Thus, data mining will never become boring or automated. And the search for better models will always be fun, especially in our time, that we have two wonderful coincidences: (a) unprecedented volumes of real data to look into and (b) unprecedented storage and computing power to try several models.

Acknowledgements Christos Faloutsos is supported by the National Science Foundation under Grants No. IIS-0209107 SENSOR-0329549 EF-0331657 IIS-0326322 IIS-0534205. He is also supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA), a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Additional funding was provided by donations from Intel, NTT and Hewlett-Packard. Vasileios Megalooikonomou is supported by the National Science Foundation under Grant No. IIS-0237921 and by the National Institutes of Health under Grant No. R01 MH68066-03 funded by NIMH, NINDS and NIA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

Barnsley M (1988) Fractals everywhere. Academic Press Inc., San Diego, CA
Barnsley MF, Sloan AD (1988) A better way to compress images. Byte, January 1988. pp 215–223
Bentley JL, Sleator DD, Tarjan RE, Wei VK (1986) A locally adaptive data compression scheme. CACM, 29(4):320–330
Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth Inc., Belmont, CA, USA

Chakrabarti D, Papadimitriou S, Modha DS, Faloutsos C (2004) Fully automatic cross-associations. In: KDD Conference, Seattle, WA, pp 79–88

Codd EF (1971) A database sublanguage founded on the relational calculus. In: SIGFIDET Workshop, pp 35–68

Cover TM, Thomas JA (1991) Elements of information theory. John Wiley and Sons

Damerau FJ (1964) A technique for computer detection and correction of spelling errors. CACM, March 1964. 7(3):171–176

Elias P (1975) Universal codeword sets and representations of integers. IEEE Trans Inf Theory IT-21:194–203

Eschera MA, Fu KS (1984) A graph distance measure for image analysis. IEEE Trans Syst Man Cybern 14:398–407

Gersho A, Gray RM (1992) Vector quantization and signal compression. Kluwer Academic Publishers

Hamerly G, Elkan C (2003) Learning the k in k-means. In: Proceedings of the NIPS

Keogh E (2002) Exact indexing of dynamic time warping. Int. conf. on very large data bases, 406–417

Keogh EJ, Lonardi S, Ratanamahatana C(Ann) (2004) Towards parameter-free data mining. In: KDD Conference, Seattle, WA, pp 206–215

Kumaraswamy K, Megalooikonomou V, Faloutsos C (2004) Fractal dimension and vector quantization. Inform Process Lett 91(3):107–113

Li M, Badger JH, Chen X, Kwong S, Kearney P, Zhang H (2001) An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics, 17(2):149–154

Li M, Chen X, Li X, Ma B, Vitanyi P (2003) The similarity metric. In: Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms, pp 863–872

Li M, Vitanyi P (1997) An Introduction to Kolmogorov complexity and its applications, 2nd edn, Springer Verlag, New York

Mackenzie D (1999) New clues to why size equals destiny. Science, 284(5420):1607–1609; 10.1126/science.284.5420.1607

Megalooikonomou V (1997) Kolmogorov incompressibility method in formal proofs: a critical survey. Technical Report TR CS-97-01, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County

Megalooikonomou V, Kontos D, Danglemaier J, Javadi A, Bakic PA, Maidment ADA (2006) A representation and classification scheme for tree-like structures in medical images: An application on branching pattern analysis of ductal trees in x-ray galactograms. In: Proceedings of the SPIE conference on medical imaging, San Diego, CA, vol 6144, March 2006, pp 497–505

Megalooikonomou V, Wang Q, Li G, Faloutsos C, (2005) A Multiresolution Symbolic Representation of Time Series. In: Proceedings of the 21st IEEE international conference on data engineering (ICDE05), Tokyo, Japan, Apr. 2005, pp 668–679

Mehta M, Agrawal R, and Rissanen J (1996) SLIQ: a fast scalable classifier for data mining. In: Proceedings of the 1996 Int. Conference on Extending Database Technology (EDBT'96), March 1996. Avignon, France, pp 18–32

Mitchell T (1997) Machine Learning. McGraw Hill

Martin-Löf P (1966) The definition of random sequences. Inform Contr 9:602–619

Papadimitriou S, Kitagawa H, Gibbons P, Faloutsos C (2003) LOCI: Fast outlier detection using the local correlation integral. ICDE, March 2003, pp 315–326

Pelleg D, Moore AW (2000) X-means: extending k-means with efficient estimation of the number of clusters. In: ICML '00: Proceedings of the seventeenth international conference on machine learning, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc, pp 727–734

Quinlan J (1993) C4.5: Programs for machine learning. Morgan Kauffman

Rabiner L, Juang B-H (1993) Fundamentals of speech recognition. Prentice Hall

Ramakrishnan R, Gehrke J (2002) Database management systems, 3rd edn. McGraw-Hill

Schroeder M (1991) Fractals, chaos, power laws: minutes from an infinite paradise. W.H. Freeman and Company, New York

Yi B-K, Jagadish HV, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. ICDE 98, Feb. 23–27

Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. ACM SIGMOD, may 1996, pp 103–114

Zhu L, Rao A, Zhang A (2002) Theory of keyblock-based image retrieval. ACM Trans Inform Syst 20(2):224–257

Ziv J, Lempel A (1978) Compression of individual sequences via variable-rate encoding. IEEE Trans Inform Theory IT-24:530–536

Zobel J, Moffat A, Sacks-Davis R (1992) An efficient indexing technique for full-text database systems. VLDB, Aug. 23–27, pp 352–362