Computers and Mathematics with Applications 65 (2013) 1471–1482

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Data complexity measured by principal graphs

Andrei Zinovyev [a,b,c,*], Evgeny Mirkes [d,e]

[a] Institut Curie, rue d'Ulm 26, Paris, 75005, France
[b] INSERM U900, Paris, France
[c] Mines ParisTech, Fontainebleau, France
[d] University of Leicester, University Road, Leicester, LE1 7RH, United Kingdom
[e] Krasnoyarsk Institute of Railway Engineering, Krasnoyarsk-28, 660028, Russia

ARTICLE INFO

To our scientific father,
Prof. Alexander N. Gorban,
on his 60th birthday.

Keywords:
Data analysis
Approximation algorithms
Data structures
Data complexity

ABSTRACT

How to measure the complexity of a finite set of vectors embedded in a multidimensional space? This is a non-trivial question which can be approached in many different ways. Here we suggest a set of data complexity measures using universal approximators, principal cubic complexes. Principal cubic complexes generalize the notion of principal manifolds for datasets with non-trivial topologies. The type of the principal cubic complex is determined by its dimension and a grammar of elementary graph transformations. The simplest grammar produces principal trees.

We introduce three natural types of data complexity: (1) geometric (deviation of the data's approximator from some "idealized" configuration, such as deviation from harmonicity); (2) structural (how many elements of a principal graph are needed to approximate the data), and (3) construction complexity (how many applications of elementary graph transformations are needed to construct the principal object starting from the simplest one).

We compute these measures for several simulated and real-life data distributions and show them in the "accuracy–complexity" plots, helping to optimize the accuracy/complexity ratio. We discuss various issues connected with measuring data complexity. Software for computing data complexity measures from principal cubic complexes is provided as well.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

1.1. What is complex data?

Rapid development of computer-based technologies in many areas of science, including physics, molecular biology, environmental research led to appearance of large datasets that are now characterized as "Big Data" [1]. There is a tremendous challenge in how to store, analyze, query and visualize the Big Data. It is frequently said that the problem of the Big Data is not only that it is big but also that it is complex. Hence, it would be useful to define what "complex data" means and be able to measure the complexity. This study is devoted to an attempt to define a way to measure some particular aspects of data complexity, connected to the data's geometry.

When somebody says "I have complex data", this can mean many different things. This can refer to the number of measurements, heterogeneity of measurement types, variety of descriptor types, complexity of descriptors, impossibility or

* Corresponding author at: Institut Curie, rue d'Ulm 26, Paris, 75005, France. Tel.: +33 1 56 24 69 89.
E-mail addresses: andrei.zinovyev@curie.fr, zinovyev@gmail.com (A. Zinovyev).

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.12.009

inability to formalize or abstract the data (like images), etc. Here we are going to deal with only one particular aspect of the complexity: complexity of data point distribution structure in some finite-dimensional space. We assume that a dataset can be represented as a set of vectors in a simple but potentially many-dimensional vectorial space. Formally, the question that we try to answer is: "how complex is the finite distribution of vectors representing data points in R^m space ($m > 1$), accompanied by some simple metrics"?

1.2. Complexity of data as complexity of approximators

There are many ways to approach the question formulated above. For example, describing "gestalt" data clusters on the language of algebraic topology (persistent data homologies) can provide some insights into the complexity of the vector distribution's structure [2,3]. Akaike information criterion (AIC) can be used to select models of data of minimal complexity, using information theory [4].

Here we develop a different approach: we are going to substitute a distribution of data which potentially contains many points by a simpler object which will approximate the data (approximator). Then we will study the complexity of the approximator instead of the complexity of the data itself. By this we believe that our approximator is a good representation of the internal structure of the data, of the data's *gestalt*. A particular point distribution is an implementation of this gestalt, which can be characterized by bigger or smaller scattering of points from it, or by bigger or smaller number of data points. If two datasets have approximators of identical complexity then we postulate that the datasets have identical complexity too. This point of view implies that our measurements of data complexity should in general rather weakly depend on the number of points and the approximation error.

A good approximator always corresponds to a compromise between its accuracy and complexity. In classical data approximation methods, number of centroids in K-means, number of principal components, curvature or length of the principal curve serve as measures of complexity. A good approximator is able to catch the hypothetical intrinsic shape of the data distribution without trying to approximate the data's "noise" (though "one man's noise is another man's signal" [5]). Therefore, limiting approximator complexity is an important aspect of any data approximation strategy. Hence, if we provide (1) a measure of the approximation complexity and (2) a method to limit it, then we can define the complexity of a given data distribution as the complexity of the corresponding *optimal* approximator. As in the case of measuring effective intrinsic data dimension, there might exist a hierarchy of data complexity levels each corresponding to certain approximation "depth".

Determining a trade-off between complexity and accuracy of approximation can be regarded as a particular application of the *Structural risk minimization principle* introduced by Vapnik and Chervonenkis in 1974 [6], if it is understood very generally (initially it was introduced for classification problems). Structural risk minimization principle is a model selection strategy which gives a model minimizing both empirical error and the model complexity (properly measured).

In practical applications, data approximation means a possibility of compressing the data. Less complex data are easier to store: a million points in thousand dimensional space can be simply distributed along a straight line with certain (relatively small) scattering around. Therefore, instead of storing the whole data massif, one can store the approximator structure, accompanied by some *uncertainty* estimates for various parts of it. This might be enough for any practical use of the data. This idea is by no means a new one (going back to the vector quantization [7] and the Minimum description length principle, from probability theory [8]), but its concrete implementations and applications remain open questions: there is yet no standard ways and tools for compressing the vectorial data.

1.3. Principal cubic complexes as universal approximators

A fundamental problem on the way to implement the idea of looking at the data's complexity through its approximation consists in finding rather universal object able to approximate complex data structures and suggest a constructive algorithm to compute it. Gorban and Zinovyev in [9] introduced a good candidate for this role: a principal cubic complex. Exact definition of it is given in the Methods section. In simple words, principal cubic complex is a Cartesian product of graphs (Fig. 1). One-dimensional principal cubic complex is simply a principal graph [10]. The graphs (factors) used to construct the cubic complex are produced by systematic application of some operations from a selected graph grammar. The operations can be scored according to how much they give in terms of optimization of some objective function (such as the elastic energy), and the best operation is applied to transform the graph.

The most trivial graph grammar consists in adding a node without connecting it to other graph nodes. This grammar produces the simplest possible "approximator": a set of principal points [11,10]. The well-known K-means clustering algorithm [12] provides a way to estimate position of this set in dataspace. A bit more complex grammar allows to add a node to one of the terminal nodes of the graph (having only one or zero neighbors). This grammar produces one-dimensional grids which can represent curves. The Cartesian product of simple linear grids gives two-, three- and higher-dimensional grids able to represent hypersurfaces, embedded in the multidimensional space of data. When they approximate data in the sense of mean-squared error and satisfy some regularity (like smoothness) constraints, they are called "principal curves" and "principal manifolds" [13–17]. Putting requirement of linearity on these approximating surfaces corresponds to approximating data by lines and planes, known as Principal Component Analysis, invented by Pearson in 1901 [18].

Next step in increasing the grammar complexity consists in allowing a node to be connected to any other node in the graph, or, be inserted in a middle of an existing edge. This produces an approximator called *the principal tree* which is already

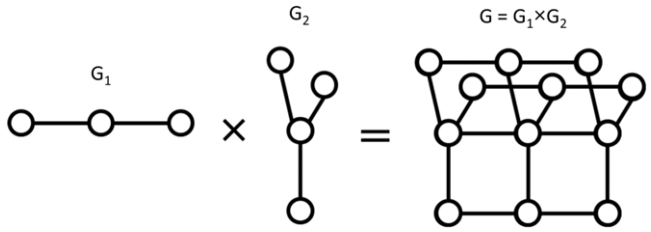

Fig. 1. Simple example of a Cartesian product of two factor-graphs.

able to approximate various non-linear and branching data distributions. In this study we will use this first non-trivial case of graph grammar to estimate the complexity of some artificial and real data distributions.

There exist an infinite number of graph grammars able to produce more complex approximators (though we should define first the notion of the approximator's complexity). Hence, what we mean here by data complexity will drastically depend on the grammar choice: in other words, on how complex is the "language" which we are going to use in order to describe the data.

1.4. Three measures of the approximator's complexity

We introduce three natural measures of the approximator's complexity, similar to the ones suggested in [10]: (1) Geometrical measure, (2) Structural measure, (3) Construction measure.

1.4.1. Geometrical complexity

The geometrical measure of complexity estimates the deviation of the approximating object from some "idealized" configuration. The simplest such ideal configuration is linear: in this case the nodes of the graph are located on some linear surface. Deviation from the linear configuration would mean some degree of *non-linearity*.

However, the notion of non-linearity is applicable only to relatively simple situations. For example, in the case of branching data distributions (see example in Fig. 2(d)), non-linearity is not applicable as a good measure of geometrical complexity. In [9] it was suggested that a good generalization of linearity as "idealized" configuration can be the notion of *harmonicity*. An embedment of a graph into multidimensional space is called harmonic if, in each star of the graph, the position of the central node of the star coincides with the mean of its leaf vectors (see more formal definition in the Methods). A linear grid with equally spaced nodes is evidently harmonic. In order to deal with arbitrary graphs, representing various grids, one has to introduce the notion of pluriharmonicity, when the harmonicity is required only for a subset of stars or for some subsets of leaves in the stars (see Methods).

In our estimations of the geometrical complexity using principal trees we will use the deviation from a harmonic embedment as analogue and generalization of the non-linearity.

1.4.2. Structural complexity

The structural complexity defines how complex is an approximator in terms of its structural elements (number of nodes, edges, stars of various degrees). In general, this index should be a non-decreasing function of these numbers. Contribution of some of the elements (for example, nodes and edges) might be not interesting for measuring the structural complexity and, hence, have zero weight (not present) in the resulting quantitative measure.

1.4.3. Construction complexity

We derive our approximators by the systematic application of the graph grammar operations, in the way which is the most optimal in terms of the objective function. One can introduce a measure of the approximator's complexity in the spirit of Kolmogorov (see, for example, [19]). The complexity of a graph can be defined as a minimum number of elementary graph transformations which were needed to produce it, using given grammar. This measure can be similar to the structural complexity in some implementations but is not equivalent to it.

1.5. Several remarks on measuring the data complexity

Approximating data by complex objects (curves, trees) in our approach is connected to non-linear optimization with all its usual problems of existence of multiple local minima and difficulties in finding the global minimum. We apply several tricks to better deal with these problems like the quadratic form of the energy functional to be minimized at each iteration, or gradual "softening" of the grid [10], but the problem cannot completely disappear, of course. It is manifested already in the case of approximating data by principal points: K-means algorithm can converge to several local minima. Moreover, within the same accuracy, one can approximate the same dataset by different number of centroids. In practice, multiple runs of K-means are needed to choose the most optimal configuration.

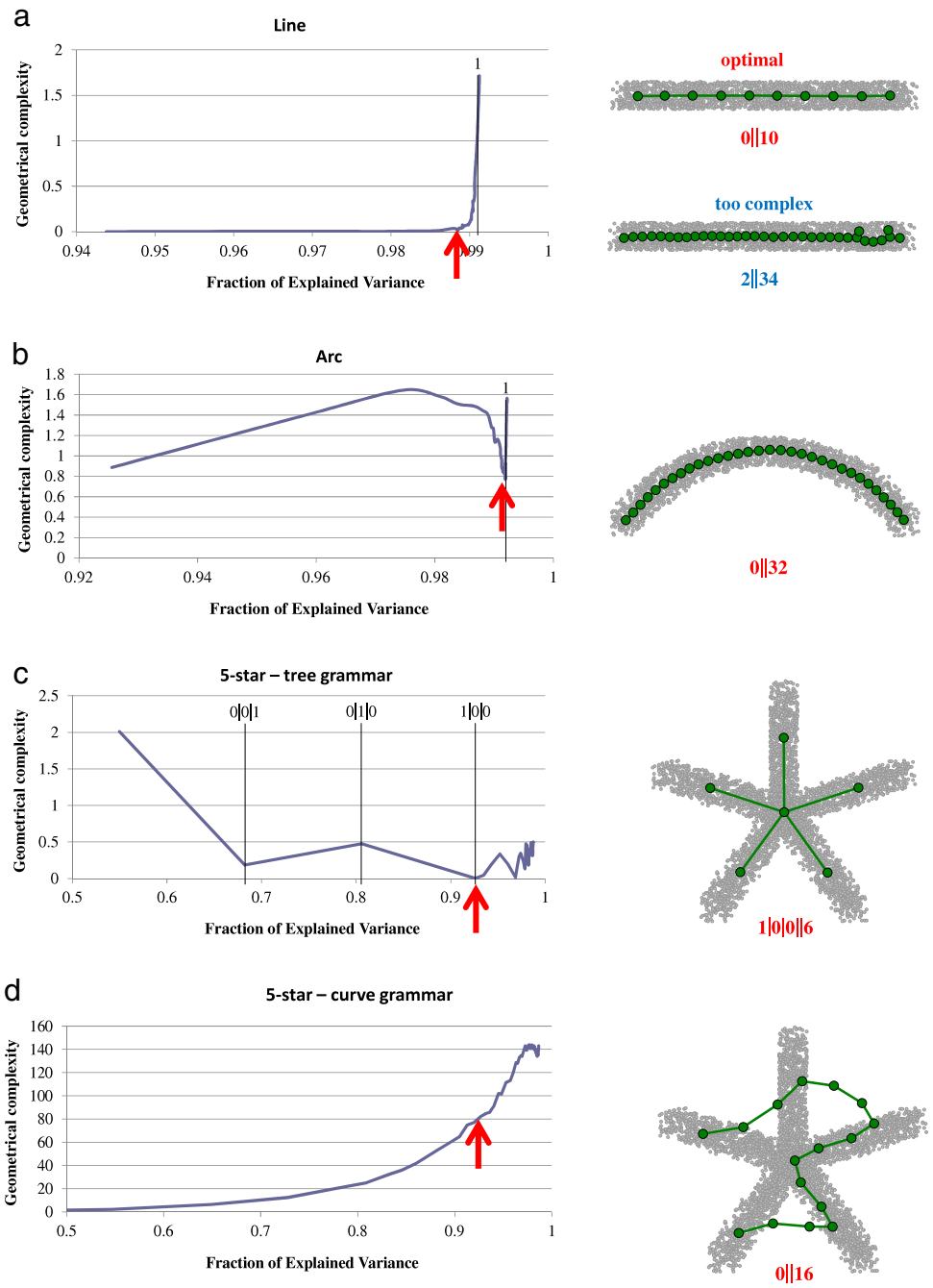

Fig. 2. Examples of the "accuracy–complexity" graphs (on the left) for the test 2D data distributions. The arrows show the accuracy/complexity chosen to construct the principal tree or curve shown on the right together with the data. The barcode shown with the principal tree or curve specifies the structural complexity. Part of the barcode (only the number of stars) is shown on the "accuracy–complexity" plot above the vertical lines visualizing the discrete changes of the structural complexity.

There are several exceptions here. One is approximating data by principal lines and linear manifolds. In this case, if the data does not have degeneracies in the covariation matrix, the quadratic energy functional has a unique global minimum. Another exception concerns a special class of data distributions which can be characterized as "pseudo-linear": when the structure of their underlying "gestalt" can be orthogonally mapped onto a line (see example in Fig. 2(b)). In other words, this corresponds to the situation when the data points can be naturally ordered using projection on a straight line. In this case, finding a close to global minimum is usually easy when starting from this line as an initial approximation. Kernel PCA [20] is another exception: it is able to produce non-linear approximating surfaces with a unique global optimum, but the approximation depends on the kernel form instead.

Non-uniqueness of the optimal approximator is tightly connected to the way of measuring the construction complexity. The graph grammar can contain the operations reducing the graph (for example, removing a node or an edge). The question is: should we take the actual (historical) number of graph grammar applications, or forget the learning history and count the number of steps which would be needed to produce the approximator's structure *de novo*? We leave this question open in this contribution because it appears as not very crucial in the case of principal trees that we use for estimating complexity.

Another remark concerns existence of clusters in data distributions. For the K-means approximator, the number k is the only measure of approximator complexity. For more complex approximators, when the data is separated well into clusters, a relevant approach is to analyze the data complexity inside each cluster separately and then characterize their "global" configuration. Real Big Data, however, is usually organized differently, with no sharp cluster borders and complicated (non-ellipsoidal) cluster shapes. In some cases, "clusters" of data can even overlap but still represent two clearly different gestalts (like two overlapping circular distributions): a case which is very little addressed in the standard clustering methodology. For our purposes we will not separate the data distribution into clusters, assuming that the data distribution represents one relatively compact group of points. However, this question can be addressed by producing sets of unconnected principal objects, such as growing "forest of principal trees" instead of growing one singular principal tree.

Our final remark concerns the dimensionality of data. Is the dimensionality itself an index of data complexity? The answer is not so simple. Of course, for the linear principal manifold, the number of retained components is a natural measure of the approximator's complexity. For more complex ones, first, we have to distinguish the dimensionality of the embedding space and the effective intrinsic dimensionality of data (dimensionality of the gestalt). Only the latter, of course, is in relation with data complexity. Second, higher dimension of data allows more complex patterns of data but not necessarily. Third, dimensionality of some objects (even as simple as principal trees) can be difficult to clearly define. And, moreover, data distributions are very frequently characterized by varying intrinsic dimension, being, for example, one-dimensional in some regions of data space and two- or three-dimensional in other regions (like the standard Iris dataset). We are not going to go deeply into these questions which have already been discussed in the literature [21,22].

2. Materials and methods

2.1. Elastic graphs and their (pluri)harmonicity

In a series of works [14,23,16,9,17,10,24] a metaphor of elastic membrane and plate was used to construct one-, two- and three-dimensional principal manifold approximations of various topologies. Mean squared distance approximation error combined with the elastic energy of the membrane serves as a functional to be optimized. The elastic map algorithm is extremely fast at the optimization step due to the simplest form of the smoothness penalty. The methodology described below is based on these ideas.

Let G be a simple undirected graph with set of vertices V and set of edges E. *k-star* in a graph G is a subgraph with $k+1$ vertices $v_0, v_1, \ldots, v_k \in V$ and k edges $\{(v_0, v_i)|i = 1, \ldots, k\} \in E$. The *rib* is by definition a 2-star.

Suppose that for each $k \geq 2$, a family S_k of k-stars in G has been selected. Then we define an *elastic graph* as a graph with selected families of k-stars S_k and for which for all $E^{(i)} \in E$ and $S_{k(j)} \in S_k$, the corresponding elasticity moduli $\lambda_i \geq 0$ and $\mu_{kj} \geq 0$ are defined.

Primitive elastic graph is an elastic graph in which every non-terminal node (with the number of neighbors more than one) is associated with a k-star formed by *all* neighbors of the node. All k-stars in the primitive elastic graph are selected, i.e. the S_k sets are completely determined by the graph structure.

Let $E^{(i)}(0)$, $E^{(i)}(1)$ denote two vertices of the graph edge $E^{(i)}$ and $S_{k(j)}(0), \ldots, S_{k(j)}(k)$ denote vertices of a k-star $S_{k(j)}$ (where $S_{k(j)}(0)$ is the central vertex, to which all other vertices are connected). Let us consider a map $\varphi : V \to \mathbf{R}^m$ which describes an embedding of the graph into a multidimensional space. The *elastic energy of the graph embedding in the Euclidean space*is defined as

$$U^{\varphi}(G) := U_E^{\varphi}(G) + U_R^{\varphi}(G), \tag{1}$$

$$U_E^{\varphi}(G) := \sum_{E^{(i)}} \lambda_i \left\| \varphi(E^{(i)}(0)) - \varphi(E^{(i)}(1)) \right\|^2, \tag{2}$$

$$U_R^{\varphi}(G) := \sum_{S_k^{(j)}} \mu_{kj} \left\| \varphi(S_k^{(j)}(0)) - \frac{1}{k} \sum_{i=1}^{k} \varphi(S_k^{(j)}(i)) \right\|^2. \tag{3}$$

Let us make two remarks. The values λ_i and μ_{kj} are the coefficients of stretching elasticity of every edge $E^{(i)}$ and of resistance to harmonicity violation for every star $S_k^{(j)}$ (which is an analogue of star "rigidity"). In the simplest case $\lambda_1 = \lambda_2 = \cdots = \lambda_s = \lambda(s)$, $\mu_{k1} = \mu_{k2} = \cdots = \mu_{kr} = \mu(r)$, where s and r are the numbers of edges and stars correspondingly.

$U_E^{\varphi}(G)$ penalizes the total length of the edges and thus provides regularization of distances between node positions. After the graph embedding is computed, λ_i can be put to zero with little effect on the graph configuration.

A map $\varphi : V \to \mathbf{R}^m$ defined on vertices of G is *pluriharmonic* iff for any k-star $S_k^{(j)} \in S_k$ with the central vertex $S_k^{(j)}(0)$ and the neighboring vertices $S_k^{(j)}(i)$, $i = 1 \dots k$, the equality holds:

$$\varphi(S_k^{(j)}(0)) = \frac{1}{k}\sum_{i=1}^{k} \varphi(S_k^{(j)}(i)). \tag{4}$$

Pluriharmonic embedding for primitive graphs is called simply harmonic embedment. For a perfectly pluriharmonic embedding, the last component $U_R^{\varphi}(G)$ of the elastic energy is zero, i.e. minimal.

2.2. Elastic principal graph with given structure

Let us choose an elastic graph, characterized by some structure. *Elastic principal graph* for a dataset X is an elastic graph embedded in $\mathbf{R}^m$ using such a map $\varphi_{opt} : V \to \mathbf{R}^m$ that corresponds to the minimal value of the functional

$$U^{\varphi}(X, G) = MSD(X, G) + U^{\varphi}(G), \tag{5}$$

where the mean squared distance (MSD) from the dataset X to the elastic graph G is calculated as the distance to the finite set of vertices $\{\mathbf{y}^1 = \varphi(v_1), \dots, \mathbf{y}^k = \varphi(v_k)\}$, i.e. for each datapoint the closest node of the graph is determined and the MSD is the mean of all such squared distances.

2.3. Definition of the elastic principal cubic complex

Introducing principal cubic complexes [9] gives a way to use "r-dimensional" graphs (similar to r-dimensional manifolds; see Fig. 1).

Elastic cubic complex K of intrinsic dimension r is a Cartesian product $G_1 \times \dots \times G_r$ of elastic graphs $G_1, \dots, G_r$. We call each G_i a factor of the cubic complex. Each factor is composed of a vertex set V_i: hence, the set of vertices V of the cubic complex is $V = V_1 \times \dots \times V_r$.

Let us select a factor G_i, $i \in 1 \dots r$ and any node in another factor $v_j \in V_j (j \neq i)$. For a chosen set of vertices V_i and a node $v_j \in V_j (j \neq i)$, there is a copy of G_i in G. It is defined by

(1) vertices

$$(v_1, \dots, v_{i-1}, v, v_{i+1}, \dots, v_r) \quad (v \in V_i),$$

(2) edges

$$((v_1, \dots, v_{i-1}, v, v_{i+1}, \dots, v_r), \qquad (v_1, \dots, v_{i-1}, v', v_{i+1}, \dots, v_r)), \quad (v, v') \in E_i,$$

(3) k-stars in the form

$$(v_1, \dots, v_{i-1}, S_k, v_{i+1}, \dots, v_r),$$

where S_k is a k-star in G_i.

For any G_i there are $\prod_{j \neq i} |V_j|$ copies of G_i in G. Sets of edges and k-stars for Cartesian product are unions of that set through all copies of all factors. A map $\varphi : V_1 \times \dots \times V_r \to \mathbf{R}^m$ maps all the copies of factors into $\mathbf{R}^m$ too.

By construction, *the energy of the elastic graph product is the energy sum of all factor copies*.It is, of course, a quadratic functional of φ.

If we approximate multidimensional data by a r-dimensional object, the number of points (or, more generally, elements) in this object grows with r exponentially. This is an obstacle for grammar-based algorithms even for modest r, because for analysis of the rule $A \to B$ applications we should investigate all isomorphic copies of A in G. Introduction of a cubic complex is useful factorization of the principal object which allows to avoid this problem.

2.4. Basic algorithm for optimization of the elastic principal graph with given structure

In the Euclidean space one can apply an Expectation–Maximization (EM) algorithm for computing the optimal embedding map φ of an elastic principal graph for a finite dataset X.

1. Choose some initial position of nodes of the elastic graph $\{\mathbf{y}^1 = \varphi(v_1), \dots, \mathbf{y}^k = \varphi(v_k)\}$, where k is the number of graph nodes $k = |V|$;
2. Calculate two matrices e_{ij}, and s_{ij}, using the following sub-algorithm:
   (a) Initialize the s_{ij} matrix to zero.
   (b) For each k-star $S_k^{(i)}$ with elasticity module μ_{ki}, outer nodes $v_{N1}, \dots, v_{Nk}$ and the central node v_{N0}, the s_{ij} matrix is updated as follows ($1 \leq l, m \leq k$):

$$s_{N_0N_0} \leftarrow s_{N_0N_0} + \mu_{ki}, \qquad s_{N_lN_m} \leftarrow s_{N_lN_m} + \mu_{ki}/k^2$$
$$s_{N_0N_l} \leftarrow s_{N_0N_l} - \mu_{ki}/k, \qquad s_{N_lN_0} \leftarrow s_{N_lN_0} - \mu_{ki}/k.$$

(c) Initialize the e_{ij} matrix to zero.

(d) For each edge $E^{(i)}$ with weight λ_i, one vertex v_{k1} and the other vertex v_{k2}, the e_{jk} matrix is updated as follows:

$$e_{k_1k_1} \leftarrow e_{k_1k_1} + \lambda_i, \qquad e_{k_2k_2} \leftarrow e_{k_2k_2} + \lambda_i$$
$$e_{k_1k_2} \leftarrow e_{k_1k_2} - \lambda_i, \qquad e_{k_2k_1} \leftarrow e_{k_2k_1} - \lambda_i.$$

3. Partition X into subsets K_i, $i = 1 \ldots k$ of data points by their proximity to $\mathbf{y}_k : K_i = \{\mathbf{x} \in X : \mathbf{y}_i = \arg\min_{\mathbf{y}_j \in Y} \mathrm{dist}(\mathbf{x}, \mathbf{y}_j)\}$.
4. Given K_i, calculate matrix $a_{js} = \frac{|K_j|\delta_{js}}{|X|} + e_{js} + s_{js}$, where δ_{js} is Kronecker's symbol.
5. Find new position of $\{\mathbf{y}^1, \ldots, \mathbf{y}^k\}$ by solving the system of linear equations

$$\sum_{s=1}^{k} a_{js}\mathbf{y}^s = \frac{1}{|X|} \cdot \sum_{\mathbf{x}^i \in K_j} \mathbf{x}^i.$$

6. Repeat steps 3–5 until complete or approximate convergence of node positions $\{\mathbf{y}^1, \ldots, \mathbf{y}^k\}$.

As usual, the EM algorithm described above gives only locally optimal solution. One can expect that the number of local minima of the energy function U grows with increasing the 'softness' of the elastic graph (decreasing μ_{kj} parameters). Because of this, in order to obtain a solution closer to the global optimum, the *softening strategy* has been proposed, used in the algorithm for estimating the elastic principal manifold [10].

2.5. Graph grammars

The graph grammars [25] provide a well-developed formalism for the description of elementary transformations. An elastic graph grammar is presented as a set of production (or substitution) rules. Each rule has a form $A \leftarrow B$, where A and B are elastic graphs. When this rule is applied to an elastic graph, a copy of A is removed from the graph together with all its incident edges and is replaced with a copy of B with edges that connect B to the graph.

Let us define *graph grammar* O as a set of graph grammar operations $O = \{o_1, \ldots, o_s\}$. All possible applications of a graph grammar operation o_i to a graph G give a set of transformations of the initial graph $o_i(G) = \{G_1, G_2, \ldots, G_p\}$, where p is the number of all possible applications of o_i to G. Let us also define a sequence of r different graph grammars $\{O^{(1)} = \{o_1^{(1)}, \ldots, o_{s_1}^{(1)}\}, \ldots, O^{(r)} = \{o_1^{(r)}, \ldots, o_{s_r}^{(r)}\}\}$.

Let us choose a grammar of elementary transformations, predefined boundaries of structural complexity $SC_{\max}$, construction complexity $CC_{\max}$ and elasticity coefficients λ_i and μ_{kj}.

Using these ingredients, we can choose the structure of the elastic principal graphs among all possible graph structures that can be obtained by application of the given graph grammar.

Elastic principal graph for a dataset X is such an elastic graph G embedded in the Euclidean space by the map $\varphi : V \to \mathbf{R}^m$ that $\mathrm{SC}(G) \leq SC_{\max}$, $\mathrm{CC}(G) \leq CC_{\max}$, and $U^{\varphi}(X, G) \to \min$ over all possible elastic graphs G embeddings in $\mathbf{R}^m$.

Note that this definition does not define a unique elastic principal graph: for the same dataset, one can have several principal graphs with equal $U^{\varphi}(X, G)$.

2.6. Algorithm for the principal graph construction

1. Initialize the elastic graph G by 2 vertices v_1 and v_2 connected by an edge. The initial map φ is chosen in such a way that $\varphi(v_1)$ and $\varphi(v_2)$ belong to the first principal line in such a way that all the data points are projected onto the principal line segment defined by $\varphi(v_1), \varphi(v_2)$.
2. For all $j = 1 \ldots r$ repeat steps 3–6:
3. Apply all grammar operations from $O^{(j)}$ to G in all possible ways; this gives a collection of candidate graph transformations $\{G_1, G_2, \ldots\}$.
4. Separate $\{G_1, G_2, \ldots\}$ into *permissible* and *forbidden* transformations; permissible transformation G_k is such that $SC(G_k) \leq SC_{\max}$, where $SC_{\max}$ is some predefined structural complexity upper bound.
5. Optimize the embedment φ and calculate the energy functional $U^{\varphi}(X, G)$ of the graph embedment for every permissible candidate transformation *after optimization*, and choose such a graph G_{opt} that gives the minimal value of the energy functional: $G_{opt} = \arg\min_{G_k \in \text{permissible set}} U^{\varphi}(X, G_k)$.
6. Substitute $G \leftarrow G_{opt}$.
7. Repeat steps 2–6 until the set of permissible transformations is empty or the number of operations exceeds a predefined number—the construction complexity.

2.7. Principal trees

Principal tree is an acyclic primitive elastic principal graph.

'*Add a node, bisect an edge*' graph grammar $O^{(grow)}$ applicable for the class of primitive elastic graphs consists of two operations: (1) The transformation "*add a node*" can be applied to any vertex v of G: add a new node z and a new edge (v, z). (2) The transformation "*bisect an edge*" is applicable to any pair of graph vertices v, v' connected by an edge (v, v'): delete edge (v, v'), add a vertex z and two edges, (v, z) and (z, v'). The transformation of the elastic structure (change in the star list) is induced by the change of the tree's edges, because the elastic graph is primitive. Consecutive application of the operations from this grammar generates trees, i.e. graphs without cycles.

Application of the "*add a node*" graph grammar operation should be accompanied by a concrete recipe on how to define the map $\varphi(z)$ (position in the data space) for a new node z. When there is only one node in the graph, there is no evident strategy on how to do this. In practice, we start constructing the graph from two nodes connected by an edge, positioned on the first principal component. Another solution is to use a random node positioning: however, most probably the first iteration will orient it close to the first principal component.

If the graph contains two or more vertices then the harmonicity is used to define the map $\varphi(z)$. For the transformation "*add a node*", the newly formed k-star containing the new node z has to be pluriharmonic, and the formula (4) is used to compute the map $\varphi(z)$. For the "bisect an edge" graph grammar operation, the newly formed rib (a 2-star) with the central vertex z should be pluriharmonic. In this case, the formula to define the map $\varphi(z)$ is very simple: $\varphi(z) = \frac{1}{2}(\varphi(v) + \varphi(v'))$.

'*Remove a leaf, remove an edge*' graph grammar $O^{(shrink)}$ applicable for the class of primitive elastic graphs consists of two operations: (1) The transformation '*remove a leaf*' can be applied to any vertex v of G with connectivity degree equal to 1: remove v and remove the edge (v, v') connecting v to the tree. (2) The transformation '*remove an edge*' is applicable to any pair of graph vertices v, v' connected by an edge (v, v'): delete edge (v, v'), delete vertex v', merge the k-stars for which v and v' are the central nodes and make a new k-star for which v is the central node with a set of neighbors which is the union of the neighbors from the k-stars of v and v'.

Also we should define the structural complexity measure $\mathrm{SC}(G) = \mathrm{SC}(|V|, |E|, |S_2|, \ldots, |S_m|)$. Its concrete form depends on the application field. Here are some simple examples:

1. $\mathrm{SC}(G) = |V|$: i.e., the graph is considered more complex if it has more vertices.
2. $\mathrm{SC}(G) = \begin{cases} |S_3|, & \text{if } |S_3| \le b_{\max} \text{ and } \sum_{k=4}^{m} |S_k| = 0 \\ \infty, & \text{otherwise} \end{cases}$,

i.e., only $b_{\max}$ simple branches (3-stars) are allowed in the principal tree.

Using the sequence $\{O^{(grow)}, O^{(grow)}, O^{(shrink)}\}$ in the above-described algorithm for estimating the elastic principal graph gives an approximation to the principal trees. Introducing the 'tree trimming' grammar $O^{(shrink)}$ allows to produce principal trees closer to the global optimum, trimming excessive tree branching and fusing k-stars separated by small 'bridges'.

2.8. Complexity measures used in the examples

In the examples below, we used principal trees constructed for several artificial and real-life data distributions, using the following forms of the complexity measures.

For measuring the geometrical complexity, we used the last term in the energy function (3), which penalizes deviation from the harmonic tree shape. For measuring complexity here, we put all $\mu_{kj} = 1$. We found out that it is convenient to multiply this term by the number of nodes squared, i.e. we used the following form of the geometrical complexity GC of graph G embedded in the multidimensional space by the map φ:

$$GC^{\varphi}(G) = N_{nodes}^2 U_R^{\varphi}(G). \tag{6}$$

Using the N_{nodes}^2 multiplier makes GC closer to the sum of squared second derivative discrete estimations for the ribs and its analogue for the stars, i.e. $\frac{\sum_{i=1}^{k} \varphi(S_k^{(j)}(i)) - k\varphi(S_k^{(j)}(0))}{\left(\frac{1}{k}\sum_{i=1}^{k} |\varphi(S_k^{(j)}(i)) - \varphi(S_k^{(j)}(0))|\right)^2}$. Adding new nodes when bisecting edges results in decreasing the average length of the graph's edges, and the N_{nodes}^2 multiplier compensates this effect. We have checked numerically the correct scaling of (6) with the increasing number of nodes for several typical growing graphs.

For the structural complexity, below we do not introduce any quantitative measure, but use a symbolic barcoding for showing the number of structural graph elements (nodes, 3-stars, 4-stars, etc.):

$$SC(G) = N_{k-stars}| \cdots |N_{4-stars}|N_{3-stars}||N_{nodes}. \tag{7}$$

For example, "2|6||15" means a principal tree with 15 nodes, 6 stars of order 3 and 2 stars of order 4. We do not show the number of edges and ribs in the barcode because the number of edges in the tree is always $N_{nodes} - 1$. The number of ribs also can be easily computed from the number of nodes and number of k-stars ($k > 2$), and it does not characterize the tree topology, but rather the number of nodes inserted in the tree branches.

The construction complexity of the principal trees which are produced by only applying grammar operations adding one node at a time, equals, evidently, $N_{nodes} - 1$, which makes it a particular case of the structural complexity. This is also true if only the final structure of the principal tree is analyzed forgetting the historical sequence of graph grammar applications. Of course, the construction complexity can be different from the structural complexity in the case of less simple graph

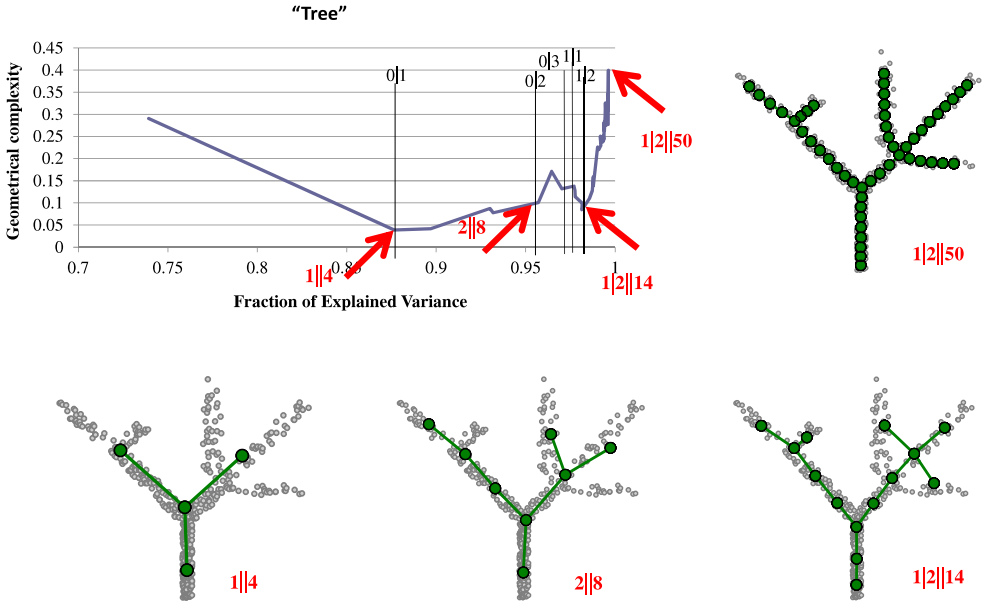

Fig. 3. The "accuracy–complexity" graph for a tree-like 2D data distribution. Several scales of the approximator's complexity are shown. Two of them, corresponding to the structural complexity barcodes 1||4 and 1|2||14 are optimal and approximate the distribution structure at a certain "depth".

grammars. Nevertheless, one can imagine a scenario when the *historical* construction complexity is not trivial for principal trees also. For example, this might be achieved if no trimming operation is applied when the increase of the elastic energy is too big. Then a sequence of graph grammar applications can contain any number of growing and trimming operations (provided that the first is bigger than the second, of course), and the resulting historical construction complexity does not equal $N_{nodes} - 1$. Having all this in mind, we nevertheless do not use the construction complexity explicitly in the examples below.

2.9. Available implementations

Method for constructing elastic principal graphs (including principal curve, principal manifold and principal tree) is implemented in Java language. User-friendly graphical interface for constructing principal manifolds is available at http://bioinfo.curie.fr/projects/vidaexpert. User-friendly graphical interface (Java-applets) for constructing principal trees in 2D is available at http://bioinfo.curie.fr/projects/elmap. The software found applications in microarray data analysis, visualization of genetic texts, visualization of economical and sociological data and other fields [23,26,16,17,24].

3. Results and discussion

3.1. Test examples

Let us first introduce the "accuracy–complexity" plots that we will use to find the optimally complex data's approximator (see examples in Fig. 2). On the abscissa of the plot we show the Fraction of Variance Explained (FVE), i.e. a unity minus ratio between the Mean Squared Error and the total data variance. The Mean Squared Error is measured with respect to the closest distance to the approximator as a polyline, i.e. to its closest node or the closest edge. On the ordinate of the plot we show the geometrical complexity defined by the formula (6). The absolute value of the geometrical complexity is in general not comparable between datasets (because its scale changes with the intrinsic data dimensionality and the spatial data scale). What is informative in the plot is the structure of minima and maxima of the geometrical complexity as well as its behavior when the approximator approaches 100% of explained variance. The changes in structural complexity are shown in the plot by vertical lines labeled by the barcode defined in (7).

To calibrate and understand the behavior of the "accuracy–complexity" graph, we used several simple 2D distributions (Figs. 2 and 3). For example, a simple linear distribution shown in Fig. 2(a) leads to a very simple "accuracy–complexity" plot. The geometrical complexity remains close to zero but drastically grows up close to $FVE \approx 0.99$. At some point, the approximator starts to produce branches which, evidently, approximate some noisy local data structures and, hence, correspond to excessive complexity. In Fig. 2(a), right we show the optimal (corresponding to a not very deep local minimum) principal tree containing 10 nodes, no branching, and the principal tree obtained at 34 nodes, containing two 3-stars (an example of an approximator whose structure is too complex with respect to the data distribution).

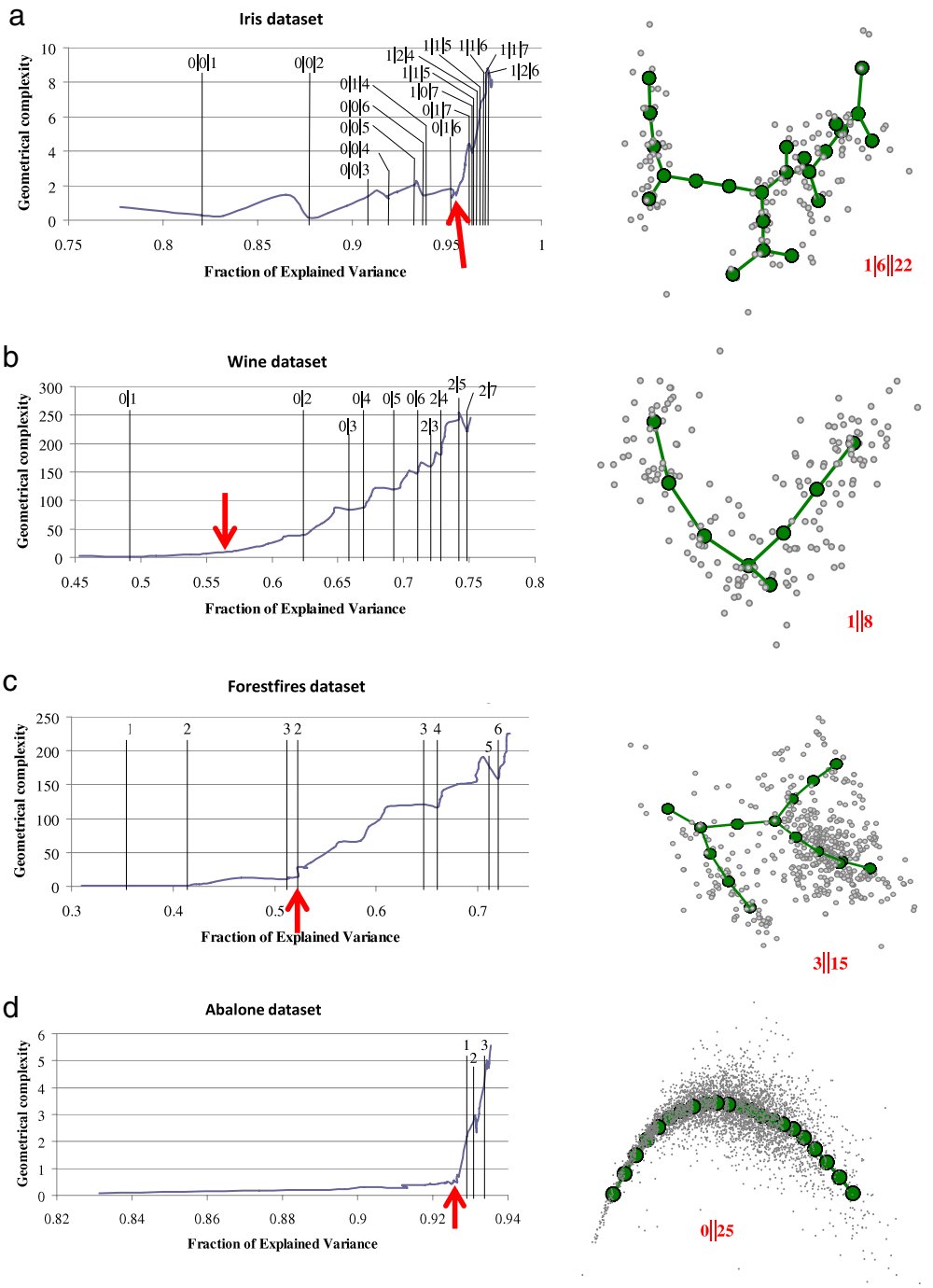

Fig. 4. Examples of the "accuracy–complexity" graphs (on the left) for the real-life data distributions (on the right, shown in projection on the first two principal components). The arrows show the accuracy/complexity chosen to construct the principal tree shown on the right together with the data. The barcode shown together with the data distribution specifies the structural complexity. Part of the barcode (only the number of stars) is shown on the "accuracy–complexity" plot above the vertical lines visualizing the discrete changes of the structural complexity.

Fig. 2(b) gives an example of a pseudo-linear distribution. Its "accuracy–complexity plot" contains a pronounced local minimum at $FVE \approx 0.99$. Further increase of the number of nodes almost does not allow to increase accuracy. Thus, the optimal configuration of the approximator is achieved at 32 nodes and no branching.

A simple example of a branching data distribution is shown in Fig. 2(c) and (d). In Fig. 2(d) we apply a reduced grammar "*Add a node to a terminal node*" which produces the principal curves (not trees). The complexity of the approximator in this case grows exponentially and suddenly saturates at $FVE \approx 0.971$ and 31 nodes. 16 nodes are needed to produce a principal

curve grid approximation for the accuracy $FVE \approx 0.92$. By contrast, the full principal tree grammar "*Add a node to any node, Bisect an edge*" needs only 6 nodes to achieve the same accuracy (Fig. 2(c)), and this corresponds to a local minimum of the complexity. Further increase of the accuracy does not lead to any significant increase of either geometrical or structural complexity. Fig. 2(c) and (d) illustrate dependence of the complexity measures on the grammar chosen: the correct principal tree grammar allows to construct much more optimal approximators.

Fig. 3 represents an interesting non-trivial example of a "tree-like" structure with several complexity scales. The "accuracy–complexity" plot contains two pronounced local minima: at $FVE \approx 0.87$ and $FVE \approx 0.98$. These two minima correspond to two scales of data approximation. The first scale depicts the data structure as a simple 3-star (corresponding to the barcode 1‖4). The second scale corresponds to the "gestalt" formed by the data points: it is a combination of further branching, containing one 4-star and two 3-stars (the barcode is 1|2||14). Further improvement of accuracy (after $FVE \approx 0.98$) is quite expensive in terms of geometrical complexity. The geometrical complexity increases by 4-fold and the number of nodes by 3.5 fold to gain only 1.5% of the total variance explained.

3.2. Examples from UC Irvine machine learning repository

We constructed principal trees for several datasets from the UCI Machine Learning Repository [27] and plotted the "accuracy–complexity" graphs for them (Fig. 4). In this Figure, both changes in the geometrical as well as structural complexity (using vertical lines labeled by the structural complexity barcode) are visualized as a function of the approximator's accuracy.

The plots show significantly different complexity of the datasets which does not necessarily coincides with dimension of the dataset or the number of points in it. For example, the *Abalone* dataset with its 4177 points represents a simple pseudo-linear distribution of points. Approximating it makes sense until the principal tree starts to form branches. When this happens, the complexity estimation abruptly goes up, meaning unnecessary growth of the approximator's complexity. Quite oppositely, the *Iris* dataset (150 data points) shows a non-trivial landscape of complexity, with many local minima and a constant growth of the structural complexity. The *Wine* dataset has a local minimum at only four nodes, forming a 3-star: this corresponds to existence of 3 well-separated ellipsoidally shaped clusters in the dataset. Further improvement of the approximation gradually and exponentially increases the approximator's complexity, and, after it is increased more than tenfold, the principal tree starts to branch further.

Finally, the *forestfires* dataset shows increase of accuracy and complexity in two epochs. During the first epoch, the geometrical complexity practically does not grow. In the second epoch it increases approximately linearly with the accuracy and saturates at ($FVE \approx 0.72$). Interestingly, at the point of the epoch change ($FVE \approx 0.52$), the structural complexity fluctuates from "2‖11" 'to "3‖12", back to "2 ‖ 16" and further to "3‖27", due to the trimming grammar applications (it becomes more advantageous to decrease the structural complexity during 11 iterations).

4. Conclusion

In the conclusion we should, first of all, repeat the principal guiding idea of this study: *good approximator is always characterized by a balance between the approximation accuracy and the approximator's complexity*. We define the data complexity as the complexity of its optimal approximator. Given the type of the approximator, one can estimate its complexity with respect to a dataset by looking at the "accuracy–complexity" plot: the optimal approximator will correspond to such a point where the further increase of accuracy leads to the drastic increase of complexity. Often this corresponds also to a local minimum of the approximator's complexity. Several local minima of the complexity landscape correspond to several "scales" of complexity in the data distribution, just as there exist multiple scales in estimating the intrinsic data dimensionality.

Good and flexible approximators allowing gradual increase of its complexity are principal cubic complexes, which can be constructed using a graph grammar. The simplest graph grammar "*add a node; bisect an edge*" produces principal trees which can be used for measuring the data complexity and choosing an approximator with the most optimal accuracy/complexity ratio, corresponding to the selected complexity scale. We applied this method to several artificial and real-life datasets, and showed that the "accuracy/complexity" plot contains enough information to justify such a choice.

References

[1] C. Lynch, Big data: how do your data grow? Nature 455 (2008) 28–29.
[2] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers 20 (1) (1971) 68–86.
[3] A.J. Zomorodian, Topology for Computing, in: Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2005.
[4] A. Hirotugu, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (6) (1974) 716–723.
[5] S. Blakeslee, Lost on earth: wealth of data found in space, An Edward Ng's quote from the article in New York Times, March, 1990.
[6] V. Vapnik, A. Chervonenkis, Ordered risk minimization I, Automation and Remote Control 35 (1974) 1226–1235.
[7] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Transactions on Communications (1980) 702–710.
[8] J. Rissanen, Modeling by shortest data description, Automatica 14 (1978) 465–471.
[9] A.N. Gorban, N. Sumner, A. Zinovyev, Topological grammars for data approximation, Applied Mathematics Letters 20 (4) (2007) 382–386.
[10] A.N. Gorban, A. Zinovyev, Principal graphs and manifolds, in: E.S. Olivas, J.D.M. Guererro, M.M. Sober, J.R.M. Benedito, A. Lopes (Eds.), Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, Information Science Reference, 2009, pp. 28–59.
[11] B. Flury, Principal points, Biometrika 77 (1990) 33–41.

[12] H. Steinhaus, Sur la division des corps materiels en parties, Bulletin L'Académie Polonaise des Science, C1. III IV (1956) 801–804.
[13] T. Hastie, Principal curves and surfaces, Unpublished doctoral dissertation, Stanford University, CA, 1984.
[14] A.N. Gorban, A.A. Rossiev, Neural network iterative method of principal curves for data with gaps, Journal of Computer and Systems Sciences International 38 (5) (1999) 825–831.
[15] B. Kégl, Principal curves: learning, design, and applications, Ph.D. Thesis, Concordia University, Canada, 1999.
[16] A.N. Gorban, A. Zinovyev, Elastic principal graphs and manifolds and their practical applications, Computing 75 (2005) 359–379.
[17] A.N. Gorban, B. Kégl, D.C. Wunsch, A. Zinovyev (Eds.), Principal Manifolds for Data Visualisation and Dimension Reduction, in: LNCSE, vol. 58, Springer, 2008.
[18] K. Pearson, On lines and planes of closest fit to systems of points in space, Philosophical Magazine 6 (2) (1901) 559–572.
[19] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Problems of Information Transmission 1 (1) (1965) 1–7.
[20] B. Schölkopf, A. Smola, K.-R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.
[21] K. Fukunaga, D. Olsen, An algorithm for finding intrinsic dimensionality of data, IEEE Transactions on Computers 20 (2) (1971) 176–183.
[22] B. Kégl, Intrinsic dimension estimation using packing numbers, in: Advances in Neural Information Processing Systems: NIPS-2002, Vol. 15, MIT Press, Vancouver, CA, pp. 681–688.
[23] A.N. Gorban, A.A. Pitenko, A. Zinovyev, D.C. Wunsch, Vizualization of any data using elastic map method, Smart Engineering System Design 11 (2001) 363–368.
[24] A.N. Gorban, A. Zinovyev, Principal manifolds and graphs in practice: from molecular biology to dynamical systems, International Journal of Neural Systems 20 (2010) 219–232.
[25] M. Nagl, Formal languages of labelled graphs, Computing 16 (1976) 113–137.
[26] A.N. Gorban, A. Zinovyev, D.C. Wunsch, Application of the method of elastic maps in analysis of genetic texts, in: Proceedings of International Joint Conference on Neural Networks, IJCNN-2003, Vol. 3, pp. 1826–1831.
[27] A. Frank, A. Asuncion, UCI machine learning repository (http://archive.ics.uci.edu/ml/), University of California, Irvine, School of Information and Computer Sciences, 2010.