Estimation of Intrinsic Dimension of Time Series Data

Brian Gauch
Vanderbilt University, TN

November 6, 2017

Abstract

We survey intrinsic dimensionality estimation techniques and dimensionality reduction techinques for time series data. We propose a novel intrinsic dimensionality reduction technique which exploits the temporal information implicit in time series data.

Part I
Motivation

Learning in a high-dimension space is slow, which is where manifold learning comes in. However, depending on how fancy the manifold learning algorithm is, even the manifold learning can be slow! Furthermore, several manifold learning algorithms have as an input parameter the target dimension, which is difficult to know in advance, meaning that you might need to run the manifold learning algorithm over and over again with different parameter values for the target dimension. Therefore, it is desirable to have another step before manifold learning; Estimation of intrinsic dimension.

Finally, even if you don't care about how long it takes to search the parameter space of a manifold learning algorithm, intrinsic dimension estimators can be useful for showing the validity of a low-dimensional model.

Part II
Background

1 Manifold Learning

In manifold learning we start with finite, high-dimensional data, i.e.,

$$\{x_1, x_2, \ldots, x_n\} = X \in \mathbb{R}^D$$

and assume that the data was generated by some lower-dimensional process.

If our assumption that our data was generated by a lower-dimensional process is correct, then our high-dimensional data should fit on a lower-dimensional manifold which is a subset of $\mathbb{R}^D$.

We should therefore use the metric of the manifold instead of metric of the $\mathbb{R}^D$ space it is embedded in.

We therefore learn a projection from this high-dimensional "data space" to a lower-dimensional "latent space", i.e.,

$$f(a \in \mathbb{R}^D) = b \in \mathbb{R}^d.$$

1.1 Inverses

Depending on the problem we are using manifold learning for, we may also want an inverse function f^{-1}. f may not be one-to-one, so in general no such inverse function is guaranteed to exist. In fact, for f to be one-to-one, it would have to be like a space-filling curve, so it would not be smooth, and we would need more bits per dimension in the latent space (for the same total number of bits).

More general notions of inverse functions exist, which suit our purposes. We call a function g the right inverse of f if

$$f(g(x)) = x, \forall x \texttt{ in the domain of } g.$$

We call g the left inverse of f if

$$g(f(x)) = x, \forall x \texttt{ in the domain of } f.$$

So, instead of f^{-1}, let us define the next best thing, a function g

$$g(b \in \mathbb{R}^d) = a \in \mathbb{R}^D$$

which is the right inverse of $f, \forall b \in \mathbb{R}^d$.

g should also be the left inverse of f if the domain of f is restricted to the manifold. However, we will not apply any such restriction.

Because we refer to the application of f as a "projection" onto the latent space, we refer to the application of g as a "reprojection" onto the data space.

Depending on the manifold learning technique used, we may not be able to find g exactly. Call whatever function we actually use to reproject from the latent space to the data space h

$$h(b \in \mathbb{R}^d) = a \in \mathbb{R}^D.$$

$$\texttt{Let } Y = f(X).$$

Depending on the manifold learning technique, f and h may or may not have their domains restricted, so that instead of $\mathbb{R}^D$ and $\mathbb{R}^d$ respectively, they may be, e.g., X and Y respectively.

However, depending on the application, we may need the domains to be unrestricted. For example, we may need f to be unrestricted if we expect more data to appear (perhaps for classification) after the model is built. We may need h to be unrestricted if we plan to learn something (such as cluster means) in the latent space and then reproject our findings back onto the data space.

1.2 Reprojection

Even if a dimensionality reduction technique only explicitly defines

$$f(X) = Y$$

we can easily construct

$$h(Y) = X.$$

Less trivially, we can expand the domains of f and h to $\mathbb{R}^D$ and $\mathbb{R}^d$ respectively. Below we show how to expand the domain of f. The same procedure can be followed to expand the domain of h.

One way to expand the domain of f, which is relatively fast, is to interpolate using k-nearest neighbors (k-NN). Using this method, if we want to find

$$b = f(a \notin X),$$

we first find the k-NN of a, and call it V. We then use some weighting scheme to associate a weight with each element v of the k-NN of a (perhaps $w_v = \frac{1}{\|a-v\|}$). We then normalize the weights W so that they sum to 1. Finally, we say that

$$b = f(a \notin X) = \sum_{v}^{V} w_v \cdot f(v).$$

Therefore, moving forward, we say without loss of generality that f and h have domains of $\mathbb{R}^D$ and $\mathbb{R}^d$ respectively.

1.2.1 Reprojection Error

Recall that $h \approx g$.

Let right reprojection error be defined as follows:

$$RE_R(b \in \mathbb{R}^d) = \|b - f(h(b))\|.$$

Because this essentially measures the error in the right inverse, RE_R tells us how far off h is from g. Depending on the manifold learning technique, we may or may not assume that $RE_R = 0 \forall b \in \mathbb{R}^d$.

Let left reprojection error (usually referred to as simply "reprojection error", because right reprojection error is not often a concern) be defined as follows:

$$RE_L(a \in \mathbb{R}^D) = \|a - h(f(a))\|.$$

Because this essentially measures the error in the left inverse, and g is only the (zero-error) left inverse of f within the domain of the learned manifold, the reprojection error will be nonzero for any data not on the learned manifold. In fact, reprojection error can be considered a metric of distance from a to the manifold.

In some cases we wish to know the reprojection error associated with a latent point. In this case, we say that

$$RE_L(b \in \mathbb{R}^d) = \|h(b) - h(f(h(b)))\|.$$

When we say that a projection of data X into a lower-dimensional space is lossless, what we mean is that $RE_L(x) = 0, \forall x \in X$.

2 Intrinsic Dimension

2.1 Definition of Intrinsic Dimension

Before we continue, we should clear up any confusion surrounding the term "intrinsic dimension" (ID). This term is used within the context of manifold learning, and is, informally, the dimension of the low-dimensional process which we believe generates our data. Therefore, we will not be able to project losslessly onto a latent space of dimension $d <$ `ID`.

Although the term "intrinsic dimension" is useful for expressing our belief that some such number of dimensions exists, we have not found a definition that gives us a good way to actually *find* such a number.

Certainly, if we are able to losslessly project high-dimensional data X onto d dimensions, then we can say after the fact that X has `ID` $\leq d$. Therefore, any manifold learning algorithm can be used to estimate intrinsic dimension, in the sense of providing an upper bound to ID, i.e., `ID` $\leq d$ where $d_{lossless}$ is the lowest dimension that the manifold learning algorithm is able to project onto losslessly. In most cases, finding such a $d_{lossless}$ requires running the manifold learning algorithm multiple times, each time giving it a different target d value as a parameter. when reducing to dimension d. Therefore, this approach is only viable if the underlying manifold learning algorithm is fast and/or does not take a target dimension d as a parameter. Camastra and Staiano refer to these intrinsic dimension estimators as "projection methods", and notably only mentions fast dimensionality reduction algorithms like PCA and MDS.

So, although we can find ID using the same projection we plan to use for dimensionality reduction, we would much prefer to have an estimate of ID `before` any computationally expensive operations like finding such a projection.

Furthermore, while we use a projection method to find an upper bound of ID, it is unclear how to establish a lower bound of ID. If we find a projection which is not lossless, it could either be because we tried to project onto too few dimensions, or it might be because the dimensionality reduction technique we used was inappropriate for the data (e.g., the data has nonlinear relationships, but we are only able to find linear ones).

Perhaps because of these problems, "intrinsic dimension" is commonly equated to one of the following, more well-defined definitions of dimension:

- Hausdorff dimension[1]
- Toplogical dimension, i.e., Lebesgue covering dimension[2]
- Information dimension[3]

Topological dimension is a lower bound on Hausdorff dimension, because the Hausdorff dimension of a fractal is a non-integer. Otherwise, topological dimension and Hausdorff dimension agree where they are both defined. As an example, the Serpinski triangle has topological dimension 1, but Hausdorff dimension $log_2(3) \approx 1.585$.

Information dimension is defined over a probability measure rather than sets, but we can easily construct a probability measure from finite training data by giving each of our n training points probabily $\frac{1}{n}$.

Our training data is a finite subset of $\mathbb{R}^D$, and for each of the above definitions of dimension, the dimension of any finite set is 0. We therefore have to use an approximation of one the three above definitions rather than applying them directly. However, because the differences between the three above definitions are subtle, an approximation of one is an approximation of them all. This may be the reason why there does not seem to be agreement in the literature about the exact definition of "intrinsic dimension".

Camastra and Staiana[4] argue the following, where $\approx$ denotes "approximates":

$$\begin{aligned}\texttt{intrinsic dimension} &= \texttt{Hausdorff dimension}\\ &\leq \texttt{box counting dimension} \geq \texttt{correlation dimension}\end{aligned}$$

Whereas Granata and Carnevale[5] argue the following:

$$\begin{aligned}\texttt{intrinsic dimension} &= \texttt{topological dimension} \approx \texttt{Hausdorff dimension}\\ &\approx \texttt{box counting dimension} \geq \texttt{correlation dimension}\end{aligned}$$

At least, Camastra and Staiano[4] seem to imply that intrinsic dimension is Hausdorff dimension. In any case, the rest of their paper[4] surveys related work on the estimation of Hausdorff dimension and information dimension, rather than ways to directly estimate intrinsic dimension. Estimation of topological dimension is also explored, but seemingly only because topological dimension is a lower bound of Hausdorff dimension.

We choose to define intrinsic dimension to be Hausdorff dimension because this should be more accurate than topological dimension if the data has a fractal nature.

Part III
Estimation of Intrinsic Dimension

3 The Ideal Intrinsic Dimension Estimator

Camastra and Staiano[4] define five criteria of an "ideal ID estimator". An ideal ID estimator should:

1. "Be computationally feasible".
2. Allow "multiscaling". I.e., accept as a parameter the scale we are interested in, so that information on a smaller scale is treated as noise and ignored.
3. "Be robust to ... high dimensionality".
4. "Have a work envelope (or operative range)."
5. "Be accurate, i.e. give an ID estimate close to the underlying manifold dimensionality"

These critera are based on Pestov's[6], with the notable omission of the requirement that an ideal ID estimator should "make no distinction between continuous and discrete objects, and the intrinsic dimension of a discrete sample should be close to that of the underlying manifold"[6]. While the first half of this requirement seems ambiguous, we can interpret the second half as follows: As we keep sampling the same manifold, our estimate of the dimension of that manifold should approach some number, which we can then call the ID of that manifold. I.e.,

$$\exists\ \mathtt{ID}^* \geq 0\ \mathtt{s.t.}\ \lim_{\|X\|\to\infty} \mathtt{ID}(X) = \mathtt{ID}^*.$$

This seems worthwhile to include as a criterion.

A possible alternative to Camastra and Staiano's[4] criterion 2 is to simply require that an ideal ID estimator be robust to noise. This would allow an ideal ID estimator to find the scale of noise itself, instead of accepting scale as a parameter.

More importantly, Camastra and Staiano's[4] criterion 3 seems overly specific about how an ideal ID estimator should present its uncertainty. The idea of a "work envelope" is that the output of sensor, or in this case the output of the ID estimator, is guaranteed to be "reliable" within some subset of its domain. As Camastra and Staiano[4] put it, "The work envelope of an ID estimator is the minimum cardinality that a data set should have so that the estimator gets a reliable estimate."

We propose that instead of necessarily providing a "work envelope", an ideal ID estimator should *in some way* indicate the relationships among:

(a) The number of data points given to the ID estimator as input.

(b) The dimension that the ID estimator returns (it takes more points to characterize a higher-dimensional manifold).

(c) The accuracy of the returned estimate.

When Camastra and Staiano[4] later describe work envelopes, it becomes clear that a work envelope is a function of (a) and (b) above, which returns TRUE if (c) is above some threshold.

Our proposed modification to criterion 3 allows other descriptions of the the relationships among (a), (b), and (c). For example, a generalized work envelope function could take in (a) and (b) and return an upper and lower bound on ID such that the probability that the ID lies in that range is greater than some threshold.

Our criteria are therefore that an ideal ID estimator should:

1. Be computationally feasible.
2. Be robust to noise.
3. Be robust to high dimensionality.
4. Indicate the relationships among:
   (a) The number of data points given to the ID estimator as input.
   (b) The dimension that the ID estimator returns (it takes more points to characterize a higher-dimensional manifold).
   (c) The accuracy of the returned estimate.
5. Be accurate.
6. $$\exists \ \mathtt{ID}^* \geq 0 \ \mathtt{s.t.} \lim_{\|X\|\to\infty} \mathtt{ID}(X) = \mathtt{ID}^*.$$

Part IV
Intrinsic Dimension Estimators

4 Projection Methods

See section 2.1. In short, every dimensionality reduction algorithm can be posed as a manifold learning algorithm, and every manifold learning algorithm can be used as a projection method to find ID, but may be slow. The category of ID estimators which Camastra and Staiano[4] call "Multidimension scaling methods" also falls under the category of projection methods, and simply corresponds to a different category of underlying dimensionality reduction algorithm. Other ID estimators that Camastra and Staiano[4] mentioned that are members of this category include Isomap and Brand's method. In fact, Isomap probably belongs under "Multidimension scaling methods".

5 Fractal-based Methods

Fractal dimension can be a non-integer, and is based on the relationship between the some measure of complexity and the scale at which complexity is measured. Unfortunately, because we are interested in behaviour at small scales, fractal-based methods tend (with exceptions[5]) to require too dense a sampling of the data to be useful.

Mathematical definitions of fractal dimension (meant to be applied to actual fractals which do not have finite points) include Hausdorff dimension, box-counting dimension, and correlation dimension. Hausdorff dimension is hard to estimate[4]. Kégl's algorithm finds box-counting dimension in $O(n^2 D)$ time. There are several algorithms for estimating correlation dimension.

5.1 Correlation Dimension

Correlation dimension is defined as

$$D_{corr} = \lim_{r \to 0} \frac{\ln C(r)}{\ln r},$$

where $C(r)$ can be thought of as the probability of a point x_i being within distance r from another arbitrary point x_j. That is,

$$C(r) = \lim_{k \to \infty} \frac{2}{k(k-1)} \sum_{i=1}^{k} \sum_{j=i+1}^{k} I(\|x_j - x_i\| \leq r)$$

where I is an *indicator function* (i.e., it is 1 if condition holds, 0 otherwise).

When estimating the correlation dimension of finite data, typically one plots values of $C(r)$ and r on a log-log plot, and tries to find the slope of the linear part of the resulting curve near $r = 0$. This is called the *Grassberger-Procaccia algorithm*[7]. Variants of the Grassberger-Procaccia algorithm exist in which we find multiple linear parts of the log-log plot. This corresponds to finding the correlation dimension at different scales, and is especially useful when we want to ignore noise that exists at the smallest scale.

There may be faster ways to calculate $C(r)$ for all values of r, but we note that if we calculate all pairwise distances, put the distances in a list, and sort the list, then finding $C(r)$ for a given r is as simple as counting the number of list elements to the left of the index where r would be inserted. This list takes $O(n^2 D + n^2 \log n)$ time to construct and afterwards finding $C(r)$ for a given r takes $O(\log n)$ time.

5.1.1 Granata and Carnevale

Granata and Carnevale[5] make the following observation: If our data is too sparse we will find few pairs of points which are close together, giving us a poor estimate of $C(r)$ for small r values, and no estimate at all for r values smaller than the distance between the closest pair of points.

Furthermore, it is very likely that our data is sparse due to the high dimension of the data space and the *curse of dimensionality*, which is no doubt why we are estimating the intrinsic dimension and then reducing dimensionality in the first place! This goes against one of Camastra and Staiano's[4] five "ideal ID estimator properties", namely the 3rd, "be robust to ... high dimensionality".

Granata and Carnevale[5] go on to show how to estimate correlation dimension despite this problem, by considering the $C(r)$ values for all values of r instead of only small ones (as they note, the *majority* of point pairs are ignored in the original definition of correlation dimension). In particular, they show that the shape of the curve in the log-log plot seems to depend more on the dimension of the data than it does on the geometry of the manifold, as long as the manifolds have similar local curvature (e.g., hypercube vs hypersphere). This property can be used to compare a manifold to reference manifolds of known dimension, and estimate ID that way.

Finally, Granata and Carnevale[5] define $p(r)$ to be $C(r)$ but using geodesic distances (i.e., distances between nodes in the graph induced by k-NN) instead of Euclidean distances, and show that using $p(r)$ instead of $C(r)$ makes the shape of the log-log plot more regular. This allows comparison between manifolds with dissimilar local curvature, and, critically, allows them to fit a curve to the log-log plot of $p(r)$ and r in order to estimate ID (i.e., $p(r)$ at $r = 0$).

6 Intrinsic Dimension from Size of Minimum Spanning Tree

Costa and Hero [8] had the novel idea of characterizing dimension by the size of a minimum spanning tree. This is a promising idea because a MST seems intuitively to measure the global curvature, and a fractal would have a large

MST. Furthermore, minimum spanning trees are relatively fast to construct, and the size of the minimum spanning tree should depend on all the data, so we do not expect burdensome requirements on sampling density.

Part V
Dimensionality Reduction of Time Series

We now adopt some of the terminology used in the discussion of dynamical systems, in favor of corresponding terms used within the context of manifold learning. The "phase space" of a dynamical process is analagous to the data space, and usually consists of position and momentum variables.

We could assume that we have data during some time interval, in which case we must pick a sampling rate to generate a discrete time series X. However, for simplicity, we could assume that we are given a discrete time series X of length n.

More generally, we assume that our time series data X consists of one or more time series segments (with the indices of the starts and ends of segments explicitly provided).

We avoid the issue of sampling because choosing a sampling rate is not as simple as it might seem [9]. If the sampling rate used to construct X is too high, then we will be biased towards short pairwise distances. Many measures of dimension, such as correlation dimension, are based on pairwise distaces, so this could throw them off.

7 Spatio-temporal Isomap

7.1 Multidimensional Scaling

Many prominent dimensionality reduction algorithms incorporate the calculation of the d largest eigenvalues and their corresponding eigenvectors. There is a great deal of research on the computation of eigenpairs, much of it iterative in nature. The "Lanczos algorithm"[10] is one such heavily used iterative algorithm. In general, eigendecomposition is $O(n^3)$[11].

The central idea of Multidimensional Scaling (MDS) is to minimize the average change in pairwise "dissimilarity" when mapping from the original space to a lower dimensional one. Classical MDS simply minimizes Euclidean distances, but the algorithmically similar Metric MDS is a class of techniques corresponding to different definitions of "dissimilarity" where pairwise dissimilarity forms a metric space. Non-metric MDS also exists for "dissimilarity" matrices which do not correspond to a metric space, although its formulation is somewhat different. The critical step in classical and metric MDS is that after a "dissimilarity" matrix is constructed, we find the d highest eigenvalues of it, and the corresponding eigenvectors. According to Wang[12], "classical MDS [can] be shown to be equivalent to PCA".

7.2 Isomap

Isomap is essentially Metric MDS where the metric is geodesic distance. In particular, a graph is constructed with weighted edges whose weights are equal to the Euclidean distances, but an edge only exists from v to w if w is one of v's k nearest neighbors (k-NN). Recall that k-NN takes $O(Dn\texttt{log}(n))$ time using a k-d tree[13]. All pairwise geodesic distances can then determined graph-theoretically, by e.g. the Floyd-Warshall algorithm[14], in $O(n^3)$ time. We can see that the $O(n^3)$ eigendecomposition corresponding to MDS has no effect on the final complexity, which is $O(n^3 + Dn\texttt{log}(n))$, or simply $O(n^3)$ if $D < n$.

7.3 Spatio-temporal Isomap

As far as we are aware, Spatio-temporal Isomap[15] is the only manifold learning algorithm which incorporates the temporal information implicit in time series data X. Instead of constructing a graph G where an edge exists from v to w iff w is one of v's k nearest neighbors (k-NN), in Spatio-temporal Isomap, we construct G as follows:

First, we add an edge from v to w if v immediately precedes w in a time series.

Next, we use a variant of k-NN to add edges from each node v to its spatial neighbors, like in original Isomap. The variant of k-NN defined by Spatio-temporal Isomap[15] is called KNTN. KNTN will create at most one edge from a node v to each other time series. In particular, it selects the shortest among the possible edges. Furthermore, even if X is only one time series, when finding the KNTN of v, we temporarily segment the data around v using a hyper-spherical window centered on v.

Finally, the authors of Spatio-temporal Isomap[15] turn all of these directed edges into undirected edges (they pose it as treating the relationships as symmetric). This is no doubt to ensure that G is fully connected, because Isomap is only defined on connected components.

8 Dimension of Attractors of Dynamical Systems

8.1 Lyapunov exponents

Loosely speaking, an attractor of a dynamical system is a subset of the phase space which the system tends to approach. An attractor can be characterized by its Lyapunov exponents, which can also be used to estimate the dimension of the attractor. More formally, according to Froehling et al.[16]

> The phase space of a dissipative dynamical system can be divided into regions in which motion is unbounded and regions in which the motion is attracted into compact subsets. These compact subsets are called attractors, the set of all phase space points which asymptotically tend to an attractor is called its basin of attraction. Certain asymptotic properties of a dynamical system's attractor are characterized by the attractor's spectrum of Lyapunov characteristic exponents (LCEs). There are as many characteristic exponents as there are dimensions in the phase space of the dynamical system. The LCEs measure the average rate of exponential convergence of trajectories onto the attractor when negative, and the average rate of exponential divergence of nearby trajectories within the attractor when positive.

Let the Lyapunov exponents be ordered

$$\lambda_1, > \lambda_2 > ... > \lambda_N.$$

Let j be the largest integer such that

$$\lambda_1, +... + \lambda_j > 0.$$

Then the Kaplan-Yorke conjecture[17] states that the dimension of an attractor is

$$d = j + \frac{\sum_{i=1}^{j} \lambda_j}{-\lambda_{j+1}}.$$

This can be called the Lyapunov dimension.

9 Estimation of Intrinsic Dimension of Time Series Data

9.1 Previous Work

Recall that any manifold learning algorithm can be used to estimate intrinsic dimension, in the sense of providing an upper bound to ID, i.e., $\texttt{ID} \leq d_{lossless}$. Therefore, Spatio-temporal Isomap can be considered an estimator of the intrinsic dimension of time series data which exploits the implicit temporal data.

We are not aware of any other previous attempts to estimate the intrinsic dimension of time series data while exploiting the implicit temporal data. Naturally, because this is a special case of intrinsic dimension estimation, any other intrinsic dimension estimators can be applied, but one would imagine that the lack of temporal information would hamper them relative to a intrinsic dimension estimator which takes advantage of temporal data.

9.2 Our Ideas

9.2.1 Using Lyapunov Dimension

We expect the ID of X to be somewhere between the dimension of the attractor and the dimension of the phase space, inclusive. If our dynamical system always starts on the attractor, then the ID of X should be the same as the Lyapunov dimension of the attractor, because all elements of X will remain on the attractor.

If at least one time series in X does not start on the attractor, we expect the ID of the X to be greater than the Lyapunov dimension of the attractor, but the difference may be less than 1.0, because depending on the time scale of our observations and how slowly states in the basin of attraction approach the attractor, behaviour near the attractor may be almost indistinguishable from behaviour on the attractor.

9.2.2 Time as Yet Another Dimension

This approach may not be very interesting, but it is worth mentioning simply because it can be used to extend any existing ID estimation method. If X comes from a single time series, then temporal distance between all pairs of points is defined. If we have them, we can simply augment each data point with the time of the observation. If we do not, then we can augment each data point with its index in X.

If X comes from multiple time series, we may still be able to estimate all pairwise temporal distances by considering each time series in X to be a path graph, and adding edges until we form a connected component. To find a set of edges to add we could, for example, treat each time series as a *node* of another graph, and find the MST of that graph, where the "distance" between two time series is, e.g, the distance of their closest pair of points.

9.2.3 Minimum Spanning Tree

Building off the work of Costa and Hero[8], instead of using a MST in the phase space, we can build a tree by starting with path graphs from our time series and connecting them, as is described above. However, this might introduce a bias where data from one long time series might appear to be more smooth (and therefore low-dimensional) than data from multiple short time series. This is intuitively similar to Spatio-temporal Isomap, because we are augmenting a graph based on nearest neighbors with edges from the temporal ordering.

References

[1] Felix Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79(1):157–179, 1918.

[2] Eduard Čech, Zdeněk Frolík, and Miroslav Katětov. Topological spaces. 1966.

[3] Alfréd Rényi. On the dimension and entropy of probability distributions. *Acta Mathematica Academiae Scientiarum Hungarica*, 10(1-2):193–215, 1959.

[4] Francesco Camastra and Antonino Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

[5] Daniele Granata and Vincenzo Carnevale. Accurate estimation of the intrinsic dimension using graph distances: Unraveling the geometric complexity of datasets. *Scientific reports*, 6:31377, 2016.

[6] Vladimir Pestov. An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21(2):204–213, 2008.

[7] Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. In *The Theory of Chaotic Attractors*, pages 170–189. Springer, 2004.

[8] Jose A Costa and Alfred O Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Processing*, 52(8):2210–2221, 2004.

[9] J-P Eckmann and David Ruelle. Fundamental limitations for estimating dimensions and lyapunov exponents in dynamical systems. *Physica D: Nonlinear Phenomena*, 56(2-3):185–187, 1992.

[10] M Newman and IU Ojalvo. Vibration modes of large structures by an automatic matrix-reductionmethod. *AIAA Journal*, 8(7):1234–1239, 1970.

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 1992.

[12] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2008.

[13] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[14] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962.

[15] Odest Chadwicke Jenkins and Maja J Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM, 2004.

[16] Harold Froehling, James P Crutchfield, Doyne Farmer, Norman H Packard, and Rob Shaw. On determining the dimension of chaotic flows. *Physica D: Nonlinear Phenomena*, 3(3):605–617, 1981.

[17] James L Kaplan and James A Yorke. Chaotic behavior of multidimensional difference equations. In *Functional Differential equations and approximation of fixed points*, pages 204–227. Springer, 1979.