Proofs of Intrinsic Dimension Estimators from scikit-dimension package

1. id.TwoNN(discard_fraction, dist)

- Intrinsic dimension estimation using the TwoNN algorithm.
- *Facco, E. et al. (2019), Estimating the intrinsic dimension of datasets by a minimal neighborhood information., Nature.*

Results

Let i be a point in the dataset, and consider the list of its first k nearest neighbors; let $r_1, r_2, \ldots, r_k$ be a sorted list of their distances from i. Thus, r_1 is the distance between i and its nearest neighbor, r_2 is the distance with its second nearest neighbor and so on; in this definition we conventionally set $r_0 = 0$.

The volume of the hypersferical shell enclosed between two successive neighbors $l-1$ and l is given by

$$\Delta v_l = \omega_d (r_l^d - r_{l-1}^d), \tag{1}$$

where d is the dimensionality of the space in which the points are embedded and ω_d is the volume of the d-sphere with unitary radius. It can be proved (see SI for a derivation) that, if the density is constant around point i,all the Δv_l are independently drawn from an exponential distribution with rate equal to the density ρ:

$$P(\Delta v_l \in [v,\, v + dv]) = \rho e^{-\rho v} dv. \tag{2}$$

Consider two shells Δv_1 and Δv_2, and let R be the quantity $\frac{\Delta v_i}{\Delta v_j}$; the previous considerations allow us, in the case of constant density, to compute exactly the probability distribution (pdf) of R:

$$\begin{aligned} &P(R \in [\overline{R},\, \overline{R} + d\overline{R}]) \\ &= \int_0^\infty dv_i \int_0^\infty dv_j \rho^2 e^{-\rho(v_i+v_j)} 1_{\left\{\frac{v_j}{v_i} \in [\overline{R}, \overline{R}+d\overline{R}]\right\}} \\ &= d\overline{R} \frac{1}{(1+\overline{R})^2}, \end{aligned}$$

where 1 represents the indicator function. Dividing by $d\overline{R}$ we obtain the pdf for R:

$$g(R) = \frac{1}{(1+R)^2}. \tag{3}$$

The pdf does not depend explicitly on the dimensionality d, which appears only in the definition of R. In order to work with a cdf depending explicitly on d we define quantity $\mu \doteq \frac{r_2}{r_1} \in [1, +\infty)$. R and μ are related by equality

$$R = \mu^d - 1. \tag{4}$$

This equation allows to find an explicit formula for the distribution of μ:

$$f(\mu) = d\mu^{-d-1} 1_{[1,+\infty]}(\mu), \tag{5}$$

while the cumulative distribution (cdf) is obtained by integration:

$$F(\mu) = (1 - \mu^{-d})1_{[1,+\infty]}(\mu). \tag{6}$$

Functions f and F are independent of the local density, but depend explicitly on the intrinsic dimension d.

A Two Nearest Neighbors estimator for intrinsic dimension. The derivation presented above leads to a simple observation: the value of the intrinsic dimension d can be estimated through the following equation

$$\frac{log(1 - F(\mu))}{log(\mu)} = d. \tag{7}$$

Remarkably the density ρ does not appear in this equation, since the cdf F is independent of ρ. This is an innovation with respect to, for instance ref.[14], where the dimension estimation is susceptible to density variations. If we consider the set $S \subset \mathbb{R}^2, S \doteq \{(log(\mu), -log(1 - F(\mu)))\}$, equation 7 claims that in theory S is contained in a straight line $l \doteq \{(x, y) \mid y = d * x\}$ passing through the origin and having slope equal to d. In practice $F(\mu)$ is estimated empirically from a finite number of points; as a consequence, the left term in equation 7 will be different for different data points, and the set S will only lie around l. This line of reasoning naturally suggests an algorithm to estimate the intrinsic dimension of a dataset:

1. Compute the pairwise distances for each point in the dataset $i = 1, \ldots, N$.
2. For each point i find the two shortest distances r_1 and r_2.
3. For each point i compute $\mu_i = \frac{r_2}{r_1}$.
4. Compute the empirical cumulate $F^{emp}(\mu)$ by sorting the values of μ in an ascending order through a permutation σ, then define $F^{emp}(\mu_{\sigma(i)}) \doteq \frac{i}{N}$.
5. Fit the points of the plane given by coordinates $\{(log(\mu_i), -log(1 - F^{emp}(\mu_i)))|i = 1, \ldots, N\}$ with a straight line passing through the origin.

Even if the results above are derived in the case of a uniform distribution of points in equations (5) and (7) there is no dependence on the density ρ; as a consequence from the point of view of the algorithm we can a posteriori relax our hypothesis: we require the dataset to be only *locally* uniform in density, where locally means in the range of the second neighbor. From a theoretical point of view, this condition is satisfied in the limit of N going to infinity. By performing numerical experiments on datasets in which the density is non-uniform we show empirically that even for a finite number of points the estimation is reasonably insensitive to density variations. The requirement of local uniformity only in the range of the second neighbor is an advantage with respect to competing approaches where local uniformity is required at larger distances.

2. id.KNN([k, ps, M, gamma])

- Intrinsic dimension estimation using the kNN algorithm
- *Carter, K.M., et al. (2010), On local intrinsic dimension estimation and its applications., IEEE Trans. on Sig. Proc., 58(2), 650-663.*

A. The k-Nearest Neighbor Algorithm for Dimension Estimation

Let $\boldsymbol{X}_n = \{x_1, \ldots, x_n\}$ be n independent identically distributed (i.i.d.) random vectors with values in a compact subset of $\mathbb{R}^d$. The (1-)nearest neighbor of x_i in $\boldsymbol{X}_n$ is given by

$$\arg \min_{x \in \boldsymbol{X}_n \setminus \{x_i\}} D(x, x_i)$$

where $D(x, x_i)$ is an appropriate distance measure between x and x_i; for the purposes of this paper, let us define $D(x, x_i) = \|x - x_i\|$, the standard Euclidean (L_2) distance. For a general integer $k \geq 1$, the k-nearest neighbor of a point is defined in a similar way. The k-NN graph assigns an edge between each point in $\boldsymbol{X}_n$ and its k-nearest neighbors. Let $\mathcal{N}_{k,i} = \mathcal{N}_{k,i}(\boldsymbol{X}_n)$ be the set of k-nearest neighbors of x_i in $\boldsymbol{X}_n$. The total edge length of the k-NN graph is defined as

$$L_{\gamma,k}(\boldsymbol{X}_n) = \sum_{i=1}^{n} \sum_{y \in \mathcal{N}_{k,i}} D(x, x_i)^{\gamma} \tag{1}$$

where $\gamma > 0$ is a power weighting constant.

For many data sets of interest, the random vectors $\boldsymbol{X}_n$ are constrained to lie on an m-dimensional Riemannian submanifold $\mathcal{M}$ of $\mathbb{R}^d$ $(m < d)$. Under this framework, the asymptotic behavior of (1) is given as

$$L_{\gamma,k}(\boldsymbol{X}_n) = n^{\alpha(m)} c + \epsilon_n \tag{2}$$

where $\alpha(m) = (m - \gamma)/m$, c is a constant with respect to $\alpha(m)$ that depends on the Rényi entropy of the distribution of the manifold and ϵ_n is an error residual [6]. Note that for ease of notation, we will denote $\alpha(m)$ simply as α, except where the explicit expression is desirable (e.g., optimizing over m).

As noisy measurements can lead to inaccurate estimates, the intrinsic dimension $\hat{m}$ should be estimated using a nonlinear least squares solution. By calculating sampled graph lengths over varying values of n, the effect of noise ϵ_n can be diminished. In order to calculate graph lengths for differing sample sizes on the manifold, it is necessary to randomly subsample from the full set $\boldsymbol{X}_n = \{x_1, \ldots, x_n\}$, utilizing the nonoverlapping block bootstrapping method [21]. Specifically, let $\boldsymbol{X}'_n = \{x_{(1)}, \ldots, x_{(n)}\}$ be a spatially or temporally sorted version of $\boldsymbol{X}_n$, and let w be an integer satisfying $w < n/Q$. Define the blocks $\mathcal{B}_i = (x_{((i-1)w+1)}, \ldots, x_{(iw)})$, $i = 1, \ldots, n/w$. As such, we may now redefine $\boldsymbol{X}'_n = \{\mathcal{B}_1, \ldots, \mathcal{B}_{n/w}\}$.

Let $\{p_1, \ldots, p_Q\}$ be Q integers such that $1 \leq p_1 < \ldots, < p_Q \leq n/w$. For each value of $p \in \{p_1, \ldots, p_Q\}$, randomly draw N bootstrap datasets $\boldsymbol{X}_p^j$, $j = 1, \ldots, N$, with replacement, where the p blocks of data points within each $\boldsymbol{X}_p^j$ are chosen from the entire data set $\boldsymbol{X}'_n$ independently. From these samples, define $\boldsymbol{L}_n = \{L_{\gamma,k}(\boldsymbol{X}_p^1), \ldots, L_{\gamma,k}(\boldsymbol{X}_p^N)\}$, where $n = p\,w$.

Since c is dependent on m, it is necessary to solve for the minimum mean squared error, derived from (2), by minimizing over both c and integer values of $m \in \mathbb{Z}$

$$\hat{m} = \arg\min_{m \in \mathbb{Z}} \left\{ \min_c \sum_{i=1}^{Q} \|\boldsymbol{L}_{n_i} - n_i^{\alpha(m)} c\mathbf{1}\|^2 \right\} \qquad (3)$$

where $n_i = p_i\, w$ and $\mathbf{1}$ is the vector of length n_i whose elements are all one. We solve over integer values of m, as we do not consider fractal dimensions for this algorithm. This improves accuracy by constraining the estimation space to discrete values rather than discretizing estimates in a continuous space. One can solve (3) in the following general manner.

```
for  m = 2 to d do
```

1: Calculate $\hat{c}(m)$ from the expansion of (3)

a)$\hat{c} = \min_c \sum_{i=1}^{Q} \|\boldsymbol{L}_{n_i}\|^2 - 2c \sum_{i=1}^{Q} n_i^{\alpha} \boldsymbol{L}_{n_i}^T \mathbf{1} + c^2 \sum_{i=1}^{Q} (n_i^{\alpha})^2 \mathbf{1}^T \mathbf{1}$

$\Rightarrow \hat{c} = \sum_{i=1}^{Q} n_i^{\alpha} \boldsymbol{L}_{n_i}^T 1 / \sum_{i=1}^{Q} (n_i^{\alpha})^2 \mathbf{1}^T \mathbf{1}$

2: Calculate the error $\boldsymbol{\epsilon}(m)$ with m and $\hat{c}$ from step 1)

$\boldsymbol{\epsilon}(m) = \sum_{i=1}^{Q} \|\boldsymbol{L}_{n_i} - \hat{c} n_i^{\alpha(m)} \mathbf{1}\|^2$

```
end.
```

$\hat{m} = \arg\min_i \boldsymbol{\epsilon}(i)$

This nonlinear least squares solution yields the dimension estimate $\hat{m}$ based on the k-NN graphs.

B. The Maximum Likelihood Estimator for Intrinsic Dimension

The MLE method [5] for dimension estimation estimates the intrinsic dimension $\hat{m}$ from a collection of i.i.d. observations $\boldsymbol{X} = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$, $m \leq d$. Similar to the k-NN algorithm for dimension estimation, the MLE method assumes that

close neighbors lie on the same manifold. The estimator proceeds as follows, letting k be a fixed number of nearest neighbors to sample x_i

$$\hat{m}_k(x_i) = \left[\frac{1}{k-2} \sum_{j=1}^{k-1} \log \frac{T_k(x_i)}{T_j(x_i)} \right]^{-1} \tag{4}$$

where $T_k(x_i)$ is the distance from point x_i to its kth nearest neighbor in $\boldsymbol{X}$. The intrinsic dimension for the data set can then be estimated as the average over all observations

$$\hat{m}_k = \frac{1}{n} \sum_{i=1}^{n} \hat{m}_k(x_i).$$

Algorithm 1 Local dimension estimation

Input: Data set $\boldsymbol{X} = \{x_1, \ldots, x_n\}$

1: **for** $i = 1$ to n **do**
2: Initialize cluster $\mathcal{C} = x_i$
3: **for** $k = 1$ to n'
4: Find the kth NN, $x_{k,i}$, of x_i
5: $\mathcal{C} \leftarrow \mathcal{C} \cup x_{k,i}$
6: **end for**
7: $\hat{m}(x_i) = \text{dimension}(\mathcal{C})$
8: **end for**

Output: Local dimension estimates $\hat{m}(x_i)$ for $i = 1, \ldots, n$

C. Local Dimension Estimation

While the MLE method inherently generates local dimension estimates for each sample $\hat{m}(x_i)$, the k-NN algorithm in itself is a global dimension estimator. We are able to adopt it (and any other dimension estimation algorithm) as a local dimension estimator by running the algorithm over a smaller neighborhood about each sample point. Define a set of n samples $\boldsymbol{X} = \{x_1, \ldots, x_n\}$ from the collection of manifolds $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_M\}$ such that each point x_i lies on manifold $\mathcal{M}_j$. Any small sphere or data cluster of samples $\mathcal{C} \subseteq \boldsymbol{X}$ centered at point x_i, with $|\mathcal{C}| = n' \leq n$, will contain samples from $M' \leq M$ distinct manifolds. As $n' \to 1$, all of the points in $\mathcal{C}$ will lie on a single manifold (i.e., $M' \to 1$). Intuitively speaking, as the cluster about point x_i is reduced in size, the local neighborhood defined by said cluster can be viewed as its own data set confined to a single manifold. Hence, we can use a global dimension estimation algorithm on a local subset of the data to estimate the local intrinsic dimension of each sample point. This can be performed as described in Algorithm 1, where "dimension($\mathcal{C}$)" refers to applying any method of dimension estimation to the data cluster $\mathcal{C}$.

One of the keys to local dimension estimation is defining a value of n'. There must be a significant number of samples in order to obtain a proper estimate, but it is also important to keep a small sample size as to (ideally) only include samples that lie on the same manifold. Currently, we arbitrarily choose n' based on the size of the data set. However, a more definitive method of choosing n' is grounds for future work.

We briefly note that our definition of "local" dimension estimation differs from that of the Fukunaga–Olsen algorithm

[1]. Specifically, we aim to find a dimension estimate for each sample point, which accounts for sets consisting of multiple manifolds, while [1] used local subsets of the data to form a global estimate of dimension.

III. Neighborhood Smoothing

For the problem of local dimension estimation, results are often highly variable, where nearby samples from the same manifold may result in different dimension estimates. This issue can be a result of a variety of reasons, such as variability due to random subsampling in the k-NN algorithm, or variability due to the neighborhood size in the MLE method. When constructing a global dimension estimate, this variance is relatively insignificant, as the estimate is constructed as a function of the local estimates. For local dimension estimation, however, this variance is of significant concern, and we propose a variance reduction method known as neighborhood smoothing [13], which improves estimation accuracy.

An initial intuition for manifold learning algorithms is that samples that are "close" tend to lie on the same manifold, which extends to the assumption that they therefore have the same dimension. With this assumption in place, it follows that filtering by majority vote over the dimension estimates of nearby samples should smooth the estimator and reduce variance. This voting strategy is similar to the methods of mode filtering, bagging [22] and learning by rule ensembles [23]. Smoothing simply looks at the distribution of dimension estimates within each sample point's local neighborhood and reassigns each sample a dimension estimate equal to that with the highest probability within its neighborhood. Specifically

$$\hat{m} = \arg\max_{l} \mathrm{P}_{\mathcal{N}_i}[\hat{m} = l] \tag{5}$$

where $\mathrm{P}_{\mathcal{N}_i}$ is the probability over the neighborhood of the current sample $\mathcal{N}_i$. Given a finite number of samples $\{x_1, \ldots, x_n\}$, this may be empirically evaluated as

$$\hat{m}(x_i) = \arg\max_{l} \sum_{x_j \in \mathcal{N}_i} I(\hat{m}(x_j) = l) \tag{6}$$

where $I(\cdot)$ is the standard indicator function. This process may then be iterated until the set converges such that each estimate remains constant. This has the effect of implicitly incorporating the neighbors of each sample's neighbors to some extent, as the dimension estimates within a local region may change through iterations.

Intuitively, neighborhood smoothing is similar to iteratively imposing a k-NN classifier on the local dimension estimates—under the guise that at each iteration, sample x_i is a test sample and all points x_j, $j \neq i$ are appropriately labeled training samples. Similarly to k-NN classification, the key factor to smoothing is defining the neighborhood $\mathcal{N}_i$. If $\mathcal{N}_i$ is too large, oversmoothing will occur. The variance of the dimension estimates will drastically decrease, but there will be a strong bias which will remove the detection of coarsely sampled manifolds. As such, one cannot use a constant region about a point but must adapt that region to the statistics of the sample.

3. id.ESS([ver, d, random_state])

- Intrinsic dimension estimation using the Expected Simplex Skewness algorithm.
- *Johnsson, K., et al. (2015), Low Bias Local Intrinsic Dimension Estimation from Expected Simplex Skewness., IEEE Trans. Pattern Anal. Mach. Intell., 37(1), 196-202.*

The Expected Simplex Skewness (ESS) method takes local data sets and gives intrinsic dimension estimates for them. The method has two closely related versions: ESSa and ESSb. We will start with describing ESSa since it has given ESS its name.

In ESSa we begin by choosing a target dimension d, which is arbitrary except that it has to be lower than the intrinsic dimension that we want to estimate.

Choosing $d = 1$ is our default option but we want to stress that it is a particular case of a more general method. For the local data set we consider simplices with one vertex in the centroid and the other $d+1$ vertices in data points. We use such simplices to estimate the expected value of what we call the simplex skewness measure. The estimation is done by a weighted mean and the result is compared to the true expected simplex skewness for uniformly distributed n-dimensional data for various n.

The simplex skewness measure is the volume of a simplex constructed as above divided by the volume it would have had if the edges incident to the centroid vertex were orthogonal. A very skew simplex has low simplex skewness measure.

As noted in the Introduction, pairs of vectors going from the origin to S^n will more often be close to orthogonal when n increases, and it follows directly that this will also be the case for the edges in the simplex incident to the centroid vertex if the local data set is uniformly distributed in an n-dimensional hyperball. Hence, the expected simplex skewness measure will approach 1 as the dimension increases. Noise and curvature will cause the expected skewness measure to deviate from this, but as we see in experiments the deviation is not very large.

If the target dimension is $d = 1$ the simplex skewness measure is $\sin\theta$, where θ is the angle between the two edges incident to the centroid vertex. The expected simplex skewness measure for uniformly distributed data on the unit ball B^n is then

$$s_n^{(1)} = \frac{1}{V(n)} \int_{B^n} |\sin\theta(x)| \, dV(x) \ , \tag{1}$$

where $V(n) = \pi^{n/2}/\Gamma(n/2+1)$ is the volume of the unit n-ball and $\theta(x)$ is the angle between the line through the origin and $x \in B^n$ and a fixed coordinate axis.

With target dimension $d > 1$ the expected simplex skewness measure for uniformly distributed data on B^n is

$$s_n^{(d)} = \frac{1}{V(n)^d} \int_{B^n \times \cdots \times B^n} |u \wedge \frac{v_1}{|v_1|} \wedge \cdots \wedge \frac{v_d}{|v_d|}| \\ dV(v_1)dV(v_2)\ldots dV(v_d), \quad (2)$$

where u is the unit vector along a reference coordinate axis and $\wedge$ denotes the exterior product.

For the ESSb estimator we consider only target dimension 1 here, but the method can be generalized to higher target dimensions. As in ESSa we assume that

we have a local data set and we consider pairs of vectors going from the centroid to data points. The quantity that we use for dimension estimation is the expected length of the projection of one vector onto the other after the vectors are scaled to unit length. As in ESSa this is estimated by a weighted mean and compared to the expected values for uniformly distributed data. If the angle between two normalized vectors is θ the length of the projection is $\cos\theta$, thus the expected projection for uniformly distributed data in B^n is

$$c_n = \frac{1}{V(n)} \int_{B^n} |\cos\theta(x)|\, dV(x), \quad (3)$$

where $V(n)$ and $\theta(x)$ are defined as above. c_n approaches zero as $n \to \infty$, as was discussed in the Introduction. More precisely, $s_n^{(d)}$ increases monotonically with n and it approaches 1 as $n \to \infty$ and c_n decreases monotonically with n and it approaches 0 as $n \to \infty$. In Appendix A in the Supplemental Material, available online at `http://doi.ieeecomputersociety.org/10.1109/TPAMIxxxxxxx`, we derive closed expressions for $s_n^{(d)}$ and c_n; the resulting expressions are

$$s_n^{(d)} = \frac{\Gamma\left(\frac{n}{2}\right)^{d+1}}{\Gamma\left(\frac{n+1}{2}\right)^d \Gamma\left(\frac{n-d}{2}\right)} \quad \text{and} \quad c_n = \frac{2V(n-1)}{A(n-1)}, \quad (4)$$

where $A(n)$ is the area of the unit n-sphere, i.e. $A(n) = (n+1)V(n+1)$.

In order to reduce the impact of noise and the exact position of the centroid we want points far from the centroid to have higher weights than points that are close to the centroid. Hence we give to each simplex or vector pair a weight that equals the product of the lengths of the edges incident to the centroid or the product of the vectors' lengths respectively. After centering the local data set so that the centroid coincides with the origin, the simplex skewness is computed from the wedge product while the projection length is computed from the dot product. For a local data set X our estimators of the expected simplex skewness with $d = 1$, $s^{(1)}$, and the expected projection length, c, respectively are thus

$$\hat{s}^{(1)} = \frac{\sum_{x,y \in X} |\bar{x} \wedge \bar{y}|}{\sum_{x,y \in X} |\bar{x}||\bar{y}|} \quad \text{and} \quad \hat{c} = \frac{\sum_{x,y \in X} |(\bar{x}, \bar{y})|}{\sum_{x,y \in X} |\bar{x}||\bar{y}|}, \tag{5}$$

where $\bar{x}$ is the vector from the centroid of the local data set to the data point x. The estimator of $s^{(d)}$ for $d > 1$ is an obvious generalization of (5). If the local data set is big, and we have more than 5,000 simplices or vector pairs, we sample 5,000 of them to use for the estimation in order to save computation time, and we find that the precision is still good enough.

Now we only need to define functions that take $\hat{s}^{(d)}$ or $\hat{c}$ and return an estimate of the dimension. Given $\hat{s}^{(d)} = s_n^{(d)}$ or $\hat{c} = c_n$ respectively we know from (4) that the dimension estimate should be n. For $s_n^{(d)} < \hat{s}^{(d)} < s_{n+1}^{(d)}$ or $c_n < \hat{c} < c_{n+1}$ we use linear interpolation to determine the dimension estimate, i.e.

$$\hat{n} = n + \frac{\hat{s}^{(d)} - s_n^{(d)}}{s_{n+1}^{(d)} - s_n^{(d)}} \quad \text{or} \quad \hat{n} = n + \frac{\hat{c} - c_n}{c_{n+1} - c_n}.$$

Given a local data set with N data points and extrinsic dimension D, the time complexity of ESSa is $O(N^{d+1}D(d+1)^2)$ since we get $O(N^{d+1})$ simplices for which we need to compute the volume through $\mathrm{vol}(S) = \sqrt{\det(SS^T)}/d!$, where S is a $(d+1) \times D$-matrix containing the coordinates of the vertices away from the origin. The time complexity of ESSb is the same with $d = 1$.

4. id.FisherS([conditional_number, …])

- Intrinsic dimension estimation using the Fisher Separability algorithm.

- *Albergante, L., et al. (2019), Estimating the effective dimension of large biological datasets using Fisher separability analysis., 2019 International Joint Conference on Neural Networks, IEEE.*

III. Estimating intrinsic data dimension based on separability properties

In the present work, we will follow the framework and notations on estimating the dimensionality of a data point cloud based on the description provided in the works by A.Gorban, I.Tyukin and their colleagues [7].

We remind the reader that a point $\mathbf{x} \in R^n$ is linearly separable from a finite set $Y \subset R^n$ if there exists a linear functional l such that $l(\mathbf{x}) > l(\mathbf{y})$ for all $y \in Y$. If for any point $\mathbf{x}$ there exists a linear functional separating it from all other data points, then such a data point cloud is called *linearly separable* or 1-convex. The separating functional l may be computed using the linear Support Vector Machine (SVM) algorithms, the Rosenblatt perceptron algorithm, or other comparable methods. However, these computations may be rather costly for large-scale estimates. Hence, it has been suggested to use the simplest non-iterative estimate of the linear functional by Fisher's linear discriminant which is computationally inexpensive, after a well-established standardised pre-processing described below [7].

Let us assume that a dataset X is normalized in the following (standardised) way:

1) centering
2) projecting onto the linear subspace spanned by first k principal components, where k may be relatively large
3) whitening (i.e., applying a linear transformation after which the covariance matrix becomes the identity matrix)
4) normalising each vector to the unit length, which corresponds to the projection onto a unit sphere.

The 4th transformation (projecting on the sphere) is optional for the general framework previously defined, but it is necessary for comparing the data distribution with a unity sphere. Choosing the number of principal components to retain in the 2nd step of the normalisation has the objective of avoiding excessively small eigenvalues of the covariance matrix (strong collinearity in the data). An effective way to estimate k, is by selecting the largest k (in their natural ranking) such that the corresponding eigenvalue λ_k is not smaller that λ_1/C, where C is a predefined threshold. Under most circumstances, $C = 10$ (i.e., the selected eigenvalue is 10 times smaller than the largest one) will result in the most popular linear estimators to work robustly.

After such normalization of X, it is said that a point $\mathbf{x} \in X$ is Fisher-linearly separable from the cloud of points Y with parameter α, if

$$(\mathbf{x}, \mathbf{y}) \leq \alpha(\mathbf{x}, \mathbf{x})$$

for all $\mathbf{y} \in Y$, where $\alpha \in [0, 1)$. If equation (III) is valid for each point $\mathbf{x} \in X$ such that Y is the set of points $\mathbf{y} \neq \mathbf{x}$ then we call the dataset X Fisher-separable with parameter α. In order to quantify deviation from perfect separability, let us introduce $p_\alpha(\mathbf{y})$, a probability that a point $\mathbf{y}$ is separable from all other points. Let us denote $\bar{p}_\alpha(\mathbf{y})$ a mean value of the distribution of $p_\alpha(\mathbf{y})$ over all data points.

Following [7], for the equidistribution on the unit sphere $S^{n-1} \in R^n$, p_α does not depend on the data point thanks to the distribution symmetry, and equals:[1].

[1]In [25], this formula, derived for large n, has $\alpha\sqrt{2\pi(n-1)}$ in the denominator. We empirically verified (see Numerical examples section) that changing the denominator to $\alpha\sqrt{2\pi n}$ makes this formula applicable for low dimensions, and the two expressions are very close for large n, since $n/(n-1) \to 1$. In [7] this formula contains a misprint (personal communication with the authors of [7]).

$$p_\alpha = \bar{p}_\alpha = \frac{(1-\alpha^2)^{\frac{n-1}{2}}}{\alpha\sqrt{2\pi n}} \tag{1}$$

Therefore, the distribution of p_α for a uniform sampling from an $n-$sphere is a delta function centered in $\bar{p}_\alpha$. The effective dimension of a data set can be evaluated by comparing $\bar{p}_\alpha$ for this data set to the value of $\bar{p}_\alpha$ for the equidistributions on a ball, a sphere, or the Gaussian distribution. Comparison to the sphere is convenient thanks to having an explicit formula (1). In order to use this formula, one should project data points on a unit sphere. If $\bar{p}_\alpha$ can be empirically estimated for a given α, then the effective dimension can be estimated by solving (1) with respect to n:

$$n_\alpha = \frac{W(\frac{-\ln(1-\alpha^2)}{2\pi\bar{p}_\alpha^2\alpha^2(1-\alpha^2)})}{-\ln(1-\alpha^2)} \tag{2}$$

where $W(x)$ is the Lambert function. The self-contained description of the algorithm for computing n_α is provided below (Algorithm 1).

Based on the above definitions, the fine-grained lumping of the data point cloud can be identified by two interesting features: the histogram of empirical p_α distribution (probabilities of individual point non-separability) and the profile of intrinsic dimensions n_α (2) for a range of α values (e.g., $\alpha \in [0.6, ..., 1.0]$).

Algorithm 1 Computing data point cloud effective dimension from Fisher-separability with parameter α

For a given data matrix X
Center the data by columns $X \leftarrow X - \bar{X}$
Apply PCA: $[V, U, S] = PCA(X)$,
where U *are projections onto principal vectors* V,
and S *are explained variances*
Select the number of components:
$k = max\{i : S(1)/S(i) < C\}$
For columns of U, u_i, *apply data whitening:*
$u_i \leftarrow u_i/\sigma(u_i), i = 1...k$
Project the data vectors onto a unit sphere:
$u_i \leftarrow u_i/||u_i||, i = 1...k$
Compute the Gram matrix $G = UU^T$
Normalize the Gram matrix by the diagonal elements:
$G_{ji} \leftarrow G_{ji}/G_{ii}$
Set to zero diagonal elements of G: $G_{ii} = 0$
For each row of G, compute the number of elements exceeding α: $v_j = \#G_{ji} > \alpha, i, j = 1...N$
Compute empirical unseparability probability distribution:
$p^j_\alpha = v_j/N$
Compute empirical mean of p_α :$\bar{p}_\alpha = \frac{1}{N}\sum_{i=1...N} p^j_\alpha$
Compute intrinsic dimension n_α *from the formula (2)*

5. id.CorrInt([k1, k2, DM])

- Intrinsic dimension estimation using the Correlation Dimension.
- *Grassberger, P. and Procaccia, I. (1983), Measuring the strangeness of strange attractors., Physica.*

https://courses.physics.ucsd.edu/2017/Spring/physics221a/Procaccia_Strangeness_of_Attractors.pdf