SAPIENZA
UNIVERSITÀ DI ROMA

Dottorato di Ricerca in Statistica Metodologica

Tesi di Dottorato XXVIII Ciclo – anno 2015/2016

Dipartimento di Statistica, Probabilità e Statistiche Applicate

Novel Methods for Intrinsic Dimension Estimation and Manifold Learning

Alessandro Lanteri

Contents

List of Figures

List of Tables

Chapter 1

Motivation and Objectives

One of the most challenging problems in modern science is how to deal with the huge amount of data that today's technologies provide. Several difficulties may arise. For instance, the number of samples may be too big and the stream of incoming data may be faster than the algorithm needed to process them. Another common problem is that when data dimension grows also the volume of the space does, leading to a sparsification of the available data. This may cause problems in the statistical analysis since the data needed to support our conclusion often grows exponentially with the dimension. This problem is commonly referred to as the Curse of Dimensionality and it is one of the reasons why high dimensional data can not be analyzed efficiently with traditional methods. Classical methods for dimensionality reduction, like principal component analysis and factor analysis, may fail due to a nonlinear structure of the data. In recent years several methods for nonlinear dimensionality reduction have been proposed. A general way to model high dimensional data set is to represent the observations as noisy samples drawn from a probability distribution μ in $\mathbb{R}^D$. It has been observed that the essential support of μ can be often well approximated by low dimensional sets. These sets can be assumed to be low dimensional manifolds embedded in the ambient dimension D. A manifold is a topologial space which globally may not be Euclidean but in a small neighbor of each point behaves like an Euclidean space. In this setting we call *intrinsic dimension* the dimension of the manifold, which is usually much lower than the ambient dimension D.

Roughly speaking, the intrinsic dimension of a data set can be described as the minimum number of variables needed to represent the data without significant loss of information. In this work we propose different methods aimed at estimate the intrinsic dimension. The first method we present models the neighbors of each point as stochastic processes, in such a way that a closed form likelihood function can

be written. This leads to a closed form maximum likelihood estimator (MLE) for the intrinsic dimension, which has all the good features that a MLE can have. The second method is based on a multiscale singular value decomposition (MSVD) of the data. This method performs singular value decomposition (SVD) on neighbors of increasing size and find an estimate for the intrinsic dimension studying the behavior of the singular values as the radius of the neighbor increases. We also introduce an algorithm to estimate the model parameters when the data are assumed to be sampled around an unknown number of planes with different intrinsic dimensions, embedded in a high dimensional space. This kind of models have many applications in computer vision and patter recognition, where the data can be described by multiple linear structures or need to be clusterized into groups that can be represented by low dimensional hyperplanes. The algorithm relies on both MSVD and spectral clustering, and it is able to estimate the number of planes, their dimension as well as their arrangement in the ambient space. Finally, we propose a novel method for manifold reconstruction based on a multiscale approach, which approximates the manifold from coarse to fine scales with increasing precision. The basic idea is to produce, at a generic scale j, a piecewise linear approximation of the manifold using a collection of low dimensional planes and use those planes to create clusters for the data. At scale $j+1$, each cluster is independently approximated by another collection of low dimensional planes. The process is iterated until the desired precision is achieved. This algorithm is fast because it is highly parallelizable and its computational time is independent from the sample size. Moreover this method automatically constructs a tree structure for the data. This feature can be particularly useful in applications which requires an a priori tree data structure. The aim of the collection of methods proposed in this work is to provide algorithms to learn and estimate the underlying structure of high dimensional dataset.

Part I

Intrinsic Dimension Estimation

Introduction

Modern data sets often consist of noisy samples taking values in a high dimensional Euclidean space yet containing an underlying low-dimensional structure. In this part of the dissertation we deal with the problem of estimating the local dimension of the low-dimensional structure that we call the intrinsic dimension of the data. Assume that we have samples from a set with intrinsic dimension d, in other words that the set is locally well approximated by a d-dimensional plane. Now, let this set be embedded in $\mathbb{R}^D$ with $D > d$. Our goal is to estimate d given the samples which are generally corrupted by noise in D dimensions. Intrinsic dimension estimation is a relevant problem in many applications, including physics, genomics, statistics, finance and machine learning. In these various areas it is commonly needed to analyze high-dimensional data sets, sometimes with limited sample size. Many algorithms have been developed to perform dimensionality reduction in order to obtain representations for high-dimensional data sets. One of the usual frameworks is to map the data from $\mathbb{R}^D$ to $\mathbb{R}^d$ preserving the pairwise distance between points. These low-dimensional representations are necessary in exploratory data analysis, enabling the visualization of very complicated data and possibly uncovering the underlying patterns and structures. However, most of these techniques require to specify the dimension d of the Euclidean space into which the data will be mapped. Values too small of d may result in the loss of significant information, while too large value of d may cause difficulties in revealing the underlying structure of the data. Thus, although an accurate estimation of the intrinsic dimension is extremely useful in data analysis, in practice this task is everything but straightforward. Classical linear methods for intrinsic dimension estimation, such principal component analysis (PCA), may overestimate the dimension when the data structure is non-linear. Nevertheless, if the data are corrupted by noise or the sample size is too small, the underlying structure may be obscured or partially explored, in these cases estimate the intrinsic dimension may be difficult. Thus, to reliably estimate the intrinsic dimension in real-world data sets, it is crucial to develop techniques that are robust to curvature in the data, noise and small sample size. In this part of the dissertation we will present some methods for the intrinsic dimension estimation based on

a modeling approach that leads to maximum likelihood estimators (MLE) for the intrinsic dimension d. We extend these methods using the composite likelihood theory and present some methods for the choice of the tuning parameters. Eventually we present some numerical results to describe the performances of the algorithms we propose.

Chapter 2

Background Theory

2.1 Intrinsic Dimension Estimation

There are several ways to classify intrinsic dimension estimation techniques; however most of them may be divided in two main categories: local and global methods. Roughly speaking, local techniques estimate the dimension in a neighbor of each data points, while global techniques estimate the intrinsic dimension of the entire data set under the assumption that it actually has a unique value. In this section we will briefly introduce different notions of dimension and some estimation methods capable to recover them from data.

2.1.1 Notions of dimensionality

In the literature, there are many different definitions of dimension, and some of them have been used to design appropriate intrinsic dimension estimation methods.

To begin with, consider the definition of the d-dimensional Hausdordff measure of a set $A \subseteq \mathbb{R}^D$, which can be found in David and Semmes (1993):

$$H^d(A) = \lim_{\delta \to 0} \inf_{\substack{\cup_i E_i \supseteq A \\ \texttt{diam}(E_i) \leq \delta}} \left(\sum_i \texttt{diam}(E_i)^d \right).$$

The infimum is taken over all sequences of sets $E_i \subseteq \mathbb{R}^D$, with diameter bounded by δ, whose union covers the set A. If $H^d(A) = C$ for some finite, positive constant, then A has Hausdorff dimension d.

Another option is the so called box-counting dimension which is a simplified

version of the Hausdorff dimension and is defined as

$$d_B = \lim_{r \to 0} \frac{\ln\big(v(r)\big)}{\ln\big(\frac{1}{r}\big)},$$

where $v(r)$ is the minimal number of boxes of size r needed to cover Ω. In practice, the box-counting dimension can only be computed for low-dimensional sets, because the computational complexity is exponential in the dimension. Thus, for high dimensional sets, the box-counting dimension needs to be approximated. This approximation is usually done via the correlation dimension. So, let $x_1, \ldots, x_n$ be a set of n i.i.d. samples from some domain $\Omega \subseteq \mathbb{R}^D$, then the correlation integral is defined as

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

and the corresponding correlation dimension is

$$d_C = \lim_{r \to 0} \frac{\ln\big(C(r)\big)}{\ln(r)}. \tag{2.1}$$

2.1.2 Estimators based on the correlation integral

One way to estimate the correlation dimension in Equation (2.1) is to plug a sufficiently small value of r into the righthand side of the equation. This approach, of course, may be very sensitive to the choice of r. As a consequence, another way to estimate d_C is simply to plot $\ln(r)$ versus $\ln\big(C(r)\big)$ and evaluate the slope of the linear portion in the resulting plot. This approach, introduced in Grassberger and Procaccia (1983), is known as the GP algorithm. However the number of samples to obtain the intrinsic dimension is prohibitive, even for a small intrinsic dimension, since the number of samples needed must be bigger than $10^{\frac{d_C}{2}}$. To tackle this issue, in Camastra and Vinciarelli (2002), a fractal method is proposed which is a modification of the GP algorithm and which is more accurate at small sample sizes. They basically assume that the intrinsic dimension d can be determined from the relationship between the correlation dimension d_C and the sample size n.

2.1.3 Nearest neighbor estimators

Another branch of techniques to estimate the intrinsic dimension d is based on geometrical information contained in neighbors of the data. Most of the algorithms in this category are based on the assumption that the density $f(x_i)$ can be locally

well approximated by

$$\widehat{f}(x) = \frac{k/n}{V_d}, \tag{2.2}$$

where k is the number of nearest neighbors to x, n is the sample size of the of data set and V_d is the volume of the d-dimensional sphere with radius r equal to the distance between x and its k–th neighbor. One of the first works in this direction is Pettis et al. (1979) where the Authors exploit Equation (2.2) to find a function similar to its logarithmic transformation which can be used to isolate and estimate d via optimization. A similar idea is introduced in Sricharan et al. (2010), where the Authors consider a statistic that averages the logarithm of the radii of many k–nearest-neighbors and then again isolate and estimate d via optimization.

In this same category there are also model based methods that tries to render geometric information in a probabilistic fashion. Works like Levina and Bickel (2004), MacKay and Ghahramani (2005) and Gupta and Huang (2012), for example, are based on the assumption that the number of points randomly drawn from a smooth Riemannian manifold with unknown density $f(\cdot)$ that fall in a small enough hypersphere, can be reasonably modeled as a suitable Poisson process. These methods usually lead to closed form maximum likelihood estimators characterized by strong theoretical properties.

2.1.4 Projective estimators

One of the most commonly used method to reduce the dimensionality of a dataset is principal component analysis (PCA) or, equivalently, the singular value decomposition (SVD) of the covariance/correlation matrix. These methods apply an orthogonal transformation to the observations in order to obtain a new set of uncorrelated variables. Dimensional reduction is usually achieved by dropping the (new) variables with an associated small eigenvalue. The main issue with the use of these methods to estimate the intrinsic dimension is the assumption that the embedded manifold is linear. If this assumption does not hold (i.e. the embedded manifold have a non linear structure) the intrinsic dimension may be overestimated due to the curvature of the manifold. This issue can be solved thinking locally under the assumption that we are dealing with a smooth Riemannian manifold with bounded curvature. In this setting a small enough region of the data should behave as a linear manifold and so, if there are enough data points to perform PCA in that neighborhood, the number of non zero eigenvalues we get can be interpreted as an estimate of the (local) intrinsic dimension. One of the first approach that uses local PCA is described in Fukunaga and Olsen (1971) which proposed a non–iterative semi–automatic algorithm subse-

quently improved in Fan et al. (2010) where initially we build a minimal cover of the data set and then perform a local PCA on each subset of the cover. A multiscale approach is proposed by Little et al. (2009b) which estimates the intrinsic dimension carefully studying the behavior of the singular values as functions of the radius of the sub–region where the local SVD is performed.

Chapter 3

Maximum Likelihood Approach

In this chapter we will introduce and describe some model based methods for the estimation of the intrinsic dimension. The use of point processes to model the number of observations that fall in arbitrary sub–region of the ambient space allows us to write a likelihood function that can be subsequently maximized to obtain a well defined estimator in closed form. In this section we will extend the original setup described in Levina and Bickel (2004) by using composite likelihood theory to obtain adjusted global estimators for the intrinsic dimension. In addition, to choose an appropriate neighborhood size, we propose a method based on cross–validation and present some numerical results on simulated datasets to show the performances of our proposals.

Model based approaches are usually strictly linked to nearest neighbor techniques. Let $X_n = \{x_1, \ldots, x_n\}$ be a set of n samples drawn from some unknown density $f(x)$ in $\mathbb{R}^D$. If the true intrinsic dimension of our point cloud is d, then, on a d–dimensional sphere $S_k(x_i)$ centered on x_i with radius $R_k(x_i)$ – which is the distance of x_i from its $k^{\texttt{th}}$ nearest neighbor, we have

$$\frac{k}{n} \approx f(x_i)\, V_d\, R_k\,(x_i)^d,$$

where V_d is the volume of the d–dimensional unit sphere. We can now consider an inhomogeneous point process $N(R, x)$ which counts the number of samples falling into a small d–dimensional sphere of radius R centered at x. This is a binomial process that under appropriate conditions can be approximated by a suitable Poisson process. If we assume that $f(x)$ is approximately constant inside a small enough sphere, then the rate λ of $N(R, x)$ is

$$\lambda(R, x) = f(x)\, V_d\, d\, R^{d-1}$$

and, according to Snyder and Miller (1991), the associated local log–likelihood have the form

$$L\big(d\left(x\right),\theta\left(x\right)\big)=\int_{0}^{R}\log\lambda\left(r,x\right)\mathrm{d}N\left(r,x\right)-\int_{0}^{R}\log\lambda\left(r,x\right)\mathrm{d}r \tag{3.1}$$

where $\theta\left(x\right)=\log f\left(x\right)$. This is an exponential family likelihood, and the maximum likelihood estimators can be found solving the equations

$$\begin{cases}\frac{\partial L}{\partial\theta}=N\left(R,x\right)-e^{\theta}V_{d}R^{d}=0\\ \frac{\partial L}{\partial d}=\left(\frac{1}{d}+\frac{V_{d}^{\prime}}{V_{d}}\right)N\left(R,x\right)+\int_{0}^{R}\log(r)\mathrm{d}N\left(R,x\right)-e^{\theta}V_{d}R^{d}\left(\log\mathrm{R}+\frac{V_{d}^{\prime}}{V_{d}}\right)=0\end{cases}.$$

The solution to those equations lead to closed form estimators

$$\begin{cases}\widehat{d}\left(x\right)=\left[\frac{1}{N(R,x)}\sum_{j=1}^{N(R,x)}\log\frac{R}{R_{j}(x)}\right]^{-1}\\ \widehat{f}\left(x\right)=\frac{N(R,x)}{V_{\widehat{d}(x)}R^{\widehat{d}(x)}}\end{cases}$$

where R_j is the euclidean distance between x and its $j^{\mathtt{th}}$ neighbour. To obtain a global estimate of the intrinsic dimension Levina and Bickel (2004) propose to locally estimate it around all the data points and then simply average the resulting estimates. However this approach leads to a very strong bias for low number of neighbors. A simple correction is proposed in MacKay and Ghahramani (2005) under the working (and usually unrealistic) assumption that the number of observations falling around each of the n points are independent to each other. In this case the MLE for the intrinsic dimension associated to this approximated global likelihood can still be written in closed form as

$$\widehat{d}=\left[\frac{1}{N\left(R,x\right)}\sum_{i=1}^{n}\sum_{j=1}^{N(R,x)}\log\frac{R}{R_{j}\left(x_{i}\right)}\right]^{-1}. \tag{3.2}$$

An alternative formulation for the likelihood and the MLE can be obtained fixing the number of neighbour k instead of the radius R. In this case we get

$$\widehat{d}=\left[\frac{1}{n\left(k-1\right)}\sum_{i=1}^{n}\sum_{j=1}^{k-1}\log\frac{R_{k}\left(x_{i}\right)}{R_{j}\left(x_{i}\right)}\right]^{-1}. \tag{3.3}$$

3.1 Composite Likelihood Approach

The framework proposed by MacKay and Ghahramani (2005) we just described can be seen as a composite likelihood (CL) maximization problem. We compute the log–likelihood in Equation (3.1) for each point; we assume that they are independent, and we then sum them all to obtain a logarithmic CL. Equation (3.2) is then the MLE associated to this simple unweighted CL. In this section, after a brief introduction on the composite likelihood theory, we will compute a correction term to the composite likelihood.

3.1.1 Introduction to Composite Likelihood

Consider a D-dimensional vector random variable Y having pdf $f(y;\theta)$ for some unknown p-dimensional parameter vector $\theta \in \Theta$. Let $\{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$ be a set of conditional or marginal events with associated likelihoods $L_k(\theta; y) \propto f(y \in \mathcal{A}_k; \theta)$. Following Lindsay (1988), a composite likelihood is

$$L_C(\theta; y) = \prod_{k=1}^{K} L_k(\theta; y)^{w_k},$$

where w_k are nonnegative weights to be chosen. If the weights are all equal then they can be ignored. Nevertheless, a careful choice of unequal weights can improve the efficiency of the resulting estimators.

In the case of n independent and identically distributed samples, for D fixed, the use of the composite likelihood is supported by some standard asymptotic results (i.e. $n \to \infty$). These results can be found in Lindsay (1988), Kent (1982), Verbeke (2005) and Varin et al. (2011). One of these results is a central limit theorem for the CL score statistics, which implies that the composite maximum likelihood estimator $\widehat{\theta}_{CL}$ is asymptotically normally distributed:

$$\sqrt{n}\big(\widehat{\theta}_{CL} - \theta\big) \xrightarrow{d} N_p\big(0, G^{-1}(\theta)\big),$$

where $N_p(\mu, \Sigma)$ denotes the p-dimensional normal distribution and $G(\theta)$ is the Godambe information matrix given by

$$G(\theta) = \left[H(\theta)^{-1} J(\theta) (H(\theta)^T)^{-1}\right]^{-1},$$

where $H(\theta)$ is the Fisher information matrix and $J(\theta)$ is the covariance matrix of the score.

Suppose that there is scientific interest in a q-dimensional subvector ψ of the parameter $\theta = (\psi, \tau)$ where τ is a nuisance parameter subvector. Composite likelihood versions of Wald and score statistics for testing the hypothesis $H_0 : \psi = \psi_0$ can be easily constructed and have the usual asymptotic χ^2_q distribution. The Wald statistic is

$$W_e = n\big(\widehat{\psi}_{CL} - \psi_0\big)^T G_{\psi\psi}\big(\widehat{\psi}_{CL} - \psi_0\big),$$

where $G_{\psi\psi}$ is the sub $(q \times q)$ submatrix of the Godambe information related to ψ. The score statistic is

$$W_u = \frac{1}{n} u_\psi\big(\psi_0, \widehat{\tau}_{CL}(\psi_0)\big)^T \tilde{H}^{\psi\psi} \tilde{G}_{\psi\psi} \tilde{H}^{\psi\psi} u_\psi\big(\psi_0, \widehat{\tau}_{CL}(\psi_0)\big),$$

where $H^{\psi\psi}$ is the $(q \times q)$ submatrix of the inverse of $H(\theta)$ related to ψ, and $\tilde{H} = H\big(\psi_0, \hat{\tau}_{CL}(\psi_0)\big)$. Like in ordinary likelihood inference W_e is not invariant to reparametrization and W_u may be numerically unstable.

It may be preferable to use the composite likelihood ratio statistic

$$W = 2\left[\log L_C\big(\widehat{\theta}_{CL}; y\big) - \log L_C\big(\psi_0, \widehat{\tau}_{CL}(\psi_0); y\big)\right],$$

which, however, have the drawback of a non-standard asymptotic distribution

$$W \xrightarrow{d} \sum_{j=1}^{q} \lambda_j Z_j^2,$$

where $Z_1, \ldots, Z_q$ are independent normal variables and $\lambda_1, \ldots, \lambda_q$ are the eigenvalues of the matrix $(H^{\psi\psi})^{-1} G^{\psi\psi}$. To overcome this problem Geys et al. (1999), proposed a correction to likelihood ratio statistic and introduced the statistic $W'(\theta) = W(\theta)/\psi$, with

$$\psi = \frac{1}{d}\sum_{i=1}^{d} \lambda_i\big(\widehat{\theta}\big)_{CL} = \frac{1}{d}\mathrm{tr}\big(H(\widehat{\theta})_{CL}^{-1} - J(\widehat{\theta})_{CL}\big). \tag{3.4}$$

The statistic W', in general, has an approximate χ^2_q distribution (see Varin (2008)). In Pauli et al. (2011) the authors propose to use $\frac{1}{\psi}$ as a particular choice of weight for the composite likelihood in order to alleviates inefficiency due to model misspecification of the composite likelihood that may lead to unreasonable inference. For instance, the variability of the composite maximum likelihood estimator may not be reflected by the shape of the composite likelihood.

3.1.2 Computing the Correction Term

In this section we will compute the composite likelihood for our model and its correction term ψ shown in Equation (3.4). The log–likelihood described in Levina and Bickel (2004) for fixed k is

$$L\big(d\left(x\right), f\left(x\right)\big) = (k-1)\log f\left(x\right) - (k-1)\log d\left(x\right) + (k-1)\log V_d + \\ + \sum_{j=1}^{k-1} \log R_j - f\left(x\right) V_d R_k^d.$$

The composite log–likelihood is then

$$cL(d,f) = \sum_{i=1}^{n} \Bigg[(k-1)\log f\left(x_i\right) - (k-1)\log d + (k-1)\log V_d + \\ \sum_{j=1}^{k-1} \log R_{ij} - f\left(x_i\right) V_d R_{ik}^d \Bigg].$$

The correction ψ is the average of the eigenvalues of the $I^{-1}J$ matrix and can be written

$$\psi = \frac{\operatorname{tr}\left(H^{-1}J\right)}{n+1},$$

where H is the Fisher's information matrix and J is the covariance matrix of the score function. We define the the score function as

$$\mathbf{u} = (u_0, u_1, \ldots, u_n) = \left(\frac{\partial cL\left(d,f\right)}{\partial d}, \frac{\partial cL\left(d,f\right)}{\partial f_1}, \ldots, \frac{\partial cL\left(d,f\right)}{\partial f_n} \right).$$

Lets define every component of J:

$$u_0 = \frac{\partial cL\left(d,f\right)}{\partial d} = \sum_{i=1}^{n} \frac{\partial L\left(d,f_i\right)}{\partial d} = n\frac{k-1}{d} + n\left(k-1\right)\frac{V_d'}{V_d} + \sum_{i=1}^{n}\sum_{j=1}^{k-1} \log R_{ij} - \\ \sum_{i=1}^{n} f_i V_d' R_k^d - \sum_{i=1}^{n} f_i V_d R_{ik}^D \log R_{ik},$$

where V_d' is the first derivative of V_d with respect to d. We note that the only component of u_0 that depends on the data is

$$\sum_{i=1}^{n}\sum_{j=1}^{k-1} \log R_{ij}.$$

Under the usual assumption that the number of points around each data point are independent, we have that the random variables R_{ij} are also independent, so the variance of u_0 is

$$\mathbb{V}ar\left(u_0\right) = \sum_{i=1}^{n} \mathbb{V}ar\left(\sum_{j=1}^{k-1} \log R_{ij}\right).$$

Since it is known from Levina and Bickel (2004) that the random variable $Y = \frac{1}{d}\sum_{i=1}^{k-1} \log \frac{R_k}{R_j}$ follows a Gamma $(k, 1)$ distribution, then

$$\sum_{i=1}^{k-1} \log R_j = \sum_{i=1}^{k-1} \log R_k - d\,Y,$$

and since $\mathbb{V}ar\left(Y\right) = k$, then

$$\sum_{i=1}^{n} \mathbb{V}ar\left(\sum_{j=1}^{k-1} \log R_{ij}\right) = n\,d^2\,\mathbb{V}ar\left(Y\right) = n\,d^2\,k.$$

The generic component u_j for j in $(1, ..., n)$ is

$$u_i = \frac{\partial cL\left(d, f_i\right)}{\partial f_i} = \frac{k-1}{f_i} - V_d R_{ik},$$

which doesn't depend on the data so, for $i \neq i'$ and $i > 0$

$$\mathbb{V}ar\left(u_i\right) = 0,\ \mathbb{C}ov\left(u_i, u_{i'}\right) = 0,$$

and

$$\mathbb{C}ov\left(u_i, u_0\right) = \mathbb{E}\left(u_j u_0\right) - \mathbb{E}\left(u_j\right)\mathbb{E}\left(u_j\right) = 0$$

$$J = \mathbb{V}ar\left(\mathbf{u}\right) = \begin{bmatrix} \mathbb{V}ar\left(u_0\right) & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} n\,d^2\,k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Define the expected Fisher's information matrix, with $l = (0, \ldots, n)$ as

$$H = \begin{bmatrix} H_{00} & \cdots & H_{1l} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ H_{l1} & \cdots & H_{ll} & \cdots & H_{ln'} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ H_{n1} & \cdots & H_{ln} & \cdots & H_{nn} \end{bmatrix},$$

where the first entry is

$$H_{00} = -\frac{\partial^2 cL\left(d, f\right)}{\partial d^2} =$$
$$\sum_{i=1}^{n}\left[\frac{k-1}{d^2} - (k-1)\,\frac{V_d'' V_d - \left(V_d'\right)^2}{V_d^2} + f_i V_d'' R_{ik}^d\right] +$$
$$+ \sum_{i=1}^{n}\left[f_i V_d' t_{ik}^d \log R_{ik} + f_i \log R_{ik} V_d' R_{ik}^d + f_i V_d \log R_{ik}^2 R_{ik}^d\right].$$

Values in the first row and first column (excluding H_{00}) are given by

$$H_{0l} = H_{l0} = -\frac{\partial^2 cL\left(d, f_l\right)}{\partial d \partial f_l} = V_d' R_{lk}^d + V_d' R_{lk}^d \log R_{lk}.$$

Values on the diagonal (excluding H_{00}) are

$$H_{ll} = -\frac{\partial^2 cL\left(d, f_l\right)}{\partial f_l \partial f_l} = \frac{k}{f_l^2}.$$

Finally, all the values not on the first row, first column or diagonal are equal to 0

$$H_{ll'} = -\frac{\partial^2 cL\left(d, f\right)}{\partial f_l \partial f_{l'}} = 0, \quad \forall\, l \neq l'.$$

We notice that H is an arrow shaped matrix

$$H = \begin{bmatrix} H_{00} & H_{10} & \cdots & H_{1l} & \cdots & H_{1n} \\ H_{01} & H_{11} & 0 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & 0 & 0 \\ H_{l1} & 0 & 0 & H_{ll} & 0 & 0 \\ \vdots & 0 & 0 & 0 & \ddots & 0 \\ H_{n1} & 0 & 0 & 0 & 0 & H_{nn} \end{bmatrix}.$$

Since H does not depend on data it can be considered both the observed and the expected Fisher Information matrix. We can also notice that

$$H^{-1}J = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{1n} \\ a_{01} & a_{11} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} n\,d^2\,k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} a_{00}\,n\,d^2\,k & 0 & 0 & 0 \\ a_{01}\,n\,d^2\,k & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}\,n\,d^2\,k & 0 & 0 & 0 \end{bmatrix},$$

then, to compute the the trace of the $H^{-1}J$ matrix we only need the element a_{00} of H^{-1} matrix. Now, since H is an arrow shaped matrix, the element a_{00} can be computed as

$$a_{00} = \left[H_{00} - \sum_{l=1}^{n} \frac{H_{0l}^2}{H_{ll}} \right]^{-1}.$$

The correction term ψ for the composite likelihood is then

$$\psi = \frac{\operatorname{tr}(I^{-1}J)}{n+1} = \frac{a_{00}\, n\, d^2\, k}{n+1},$$

that can be written in its non–simplified form as

$$\begin{aligned}\psi =& \frac{nd^2k}{n+1} \left\{ \sum_{i=1}^{n} \left[\frac{k-1}{d^2} - (k-1)\, \frac{V_d'' V_d - (V_d')^2}{V_d^2} + f_i V_d'' R_{ik}^d + f_i V_d' t_{ik}^d \log R_{ik} + \right.\right. \\ & \left.\left. + f_i \log R_{ik} V_D' R_{ik}^d + f_i V_d \log R_{ik}^2 R_{ik}^d \right] - \sum_{l=1}^{n} \frac{\left(V_d' R_{lk}^d + V_d' R_{lk}^d \log R_{lk} \right)^2}{\frac{k}{f_l^2}} \right\}^{-1}. \end{aligned} \tag{3.5}$$

3.1.3 Numerical Results

In this section we show some numerical results on the performance of the correction term introduced in the last section. The simulation study is organized as follows

1. Draw 200 samples uniformly from a 2–dimensional flat torus embedded in $\mathbb{R}^4$.
2. Randomly pick 50 points and build their neighbors.
3. Estimate the intrinsic dimension of the manifold in three different ways:
   - $d_{\texttt{mle}}$ – Closed form MLE in Equation (3.3).
   - $d_{\texttt{emp}}$ – Evalute on a grid of d values the composite marginal likelihood built using the 50 (local) likelihoods associated to the selected 50 neighborhoods and find the value which empirically maximize the composite likelihood.
   - $d_{\texttt{cor}}$ – Computed as the previous one but the composite likelihood used is corrected as in Section 3.1.2.
4. All the estimators are evaluated on a grid of k values.
5. Repeat 100 times and average the results.

Figure 3.1.3 shows the results of the numerical experiment. We see how, for a good range of k values, the correction term tends to gives estimates closer to the real value of the intrinsic dimension d, compared to the other two methods.

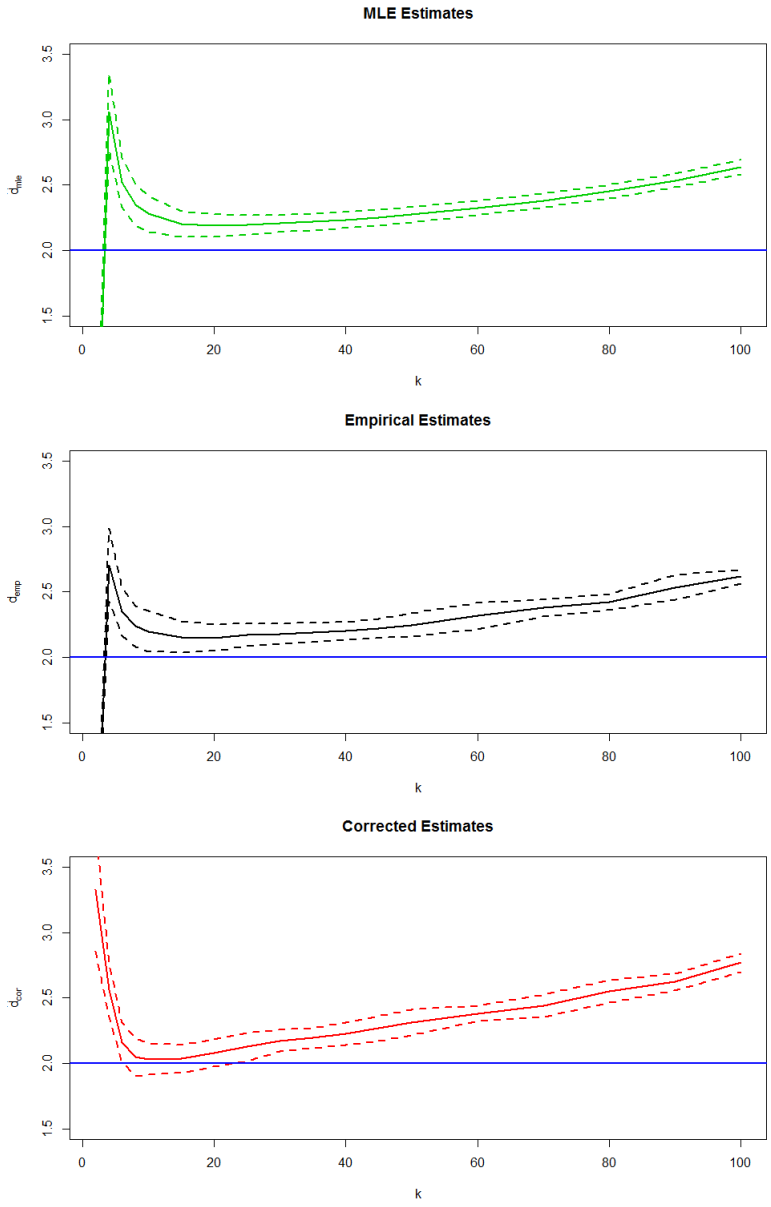

Figure 3.1: Estimates of the intrinsic dimension provided by the three methods described in Section 3.1.3, for data on a 2–dimensional flat torus embedded in $\mathbb{R}^4$. The green, black and red lines represents $d_{\texttt{mle}}$, $d_{\texttt{emp}}$ and $d_{\texttt{cor}}$ respectively. The solid lines are the average estimates over 100 iteration. The blue solid line represents the real intrinsic dimension. The dashed lines are bands centered on the average estimate $\pm$ 1 standard error.

3.2 Cross Validation for the choice of k

One of the most critical aspect of the methods presented in this chapter is the choice of the neighborhood size. In other words we have to decide either the number of element k inside the neighborhood or its radius r. In this section we will propose some ways to choose these parameters.

3.2.1 Cross Validation

In this section we propose an algorithm (Algorithm 1) to find the optimal number of neighbors k in order to obtain a good estimate for the intrinsic dimension d. What we

Algorithm 1 Cross Validation for optimal k

Input: An $(n \times D)$ dataset X
Output: Optimal number of neighbors k
Steps:
- **for** $k = 1 : (n-1)$
  - **for** $i = 1 : n$
    compute $d(k)_i$ and $\widehat{\mathbf{f}}(k)_{-i}$ described in Section 3.2.1
    compute $w\left(\widehat{\mathbf{f}}(k)_{-i}\right)$ as in Equation 3.9
    sample M values from the distribution X^*_{-i} described in Equation (3.6)
    compute the distance $d\left(k,\, g\left(X^*_{-i}\right)\right)$
  compute $C(k)$ as in Equation (3.7)
compute k^* as in Equation (3.8)
return: the optimal neighbors number k^*
- **end algorithm**

propose, for the choice of the tuning parameter k, is a leave–one–out Cross Validation (CV), which can be easily extended to a K–fold Cross Validation. The problem with the application of the CV in our case is that you have to choose what to "validate" and among what to "cross". Our idea is to select a summary statistic of a data point's k-neighborhood and measure how well it "validates" a random realization of the same statistic depending from an estimate of the parameters ($\mathbf{f}$ and d) and then "cross" through all the data points to get a CV score. Let x_i, with i in $(1, \ldots, n)$, be a D-dimensional data point and X be the $(n \times D)$ data matrix. Using the MacKay and Ghahramani (2005) method, for any given k we obtain the estimates $\widehat{\mathbf{f}}(k)_{-i}$, which length is $n-1$, and $\widehat{d}(k)_{-i}$ which are respectively the estimates of $\mathbf{f}$ and d without taking into account the $i^{\mathtt{th}}$ observation. Since we assume that if the sphere that contain k data points $S(k)$ is small enough, those points are uniformly distributed, we treat the observations as a Poisson process in that sphere. Thus, we

can predict the number of points in a multidimensional sphere around the i^{th} point as

$$X_i^* \sim \text{Poisson}\left(\widehat{f}\,(k)_i\, V\left(\widehat{d}\,(k)_i\right)\widehat{d}\,(k)_i\, T\,(k)_i^{\widehat{d}(k)_i-1}\right).$$

where $V\,(a)$ is the volume of a unit ball of dimension a and $T\,(k)_i$ is the distance between x_i and its k^{th} neighbor. However, we need to predict the number of points in $S\,(k)_i$ without taking into account the i^{th} point, in other words we can not directly compute $\widehat{f}\,(k)_i$ and $\widehat{d}\,(k)_i$. The most obvious solutions is then to replace $\widehat{d}\,(k)_i$ with $\widehat{d}\,(k)_{-i}$. On the other hand the choice of $\widehat{f}\,(k)_{-i}$ is not completely straightforward and in fact our proposal is to combine the values $\widehat{\mathbf{f}}\,(k)_{-i}$ with a suitable weight function $w\,(\cdot)$. The choice of $w\,(\cdot)$ is discussed in Section 3.2.2. With those corrections we can predict the number of points in a multidimensional sphere around the i^{th} observation, removing the point itself from the computation of the estimates, as

$$X_{-i}^* \sim \text{Poisson}\left(w\left(\widehat{\mathbf{f}}\,(k)_{-i}\right)V\left(\widehat{d}\,(k)_{-i}\right)\widehat{d}\,(k)_{-i}\, T\,(k)_i^{\widehat{d}(k)_{-i}-1}\right) \tag{3.6}$$

Then we can compute the CV score

$$C\,(k) = \mathfrak{F}\left[\sum_{i=1}^{n} d\left(k,\, g\left(X_{-i}^*\right)\right)\right], \tag{3.7}$$

where $\mathfrak{F}\,[\cdot]$ is an some optimality criterion, $d\,(\cdot)$ is a reasonable distance function and $g\,(\cdot)$ may be a representative function of the predictive distribution (e.g. the mean). The choices of $\mathfrak{F}$, $d\,(\cdot)$ and $g\,(\cdot)$ are discussed in Section 3.2.3. The optimal number of neighbor is then chosen with the rule

$$k^* = \left\{k :\, C\,(k) \leq C\,(l)\,,\ \forall\, k,\ \forall\, l\right\}. \tag{3.8}$$

The idea of this method is that point process driven by the predictive distribution X_{-i}^* within a d dimensional sphere of radius $T\,(k)_i$ with intensity f_i should, at least on average, return values close to k. So we select as optimal value for k, the one that leads to estimates of d and $\mathbf{f}$ that minimize $C\,(k)$.

3.2.2 The choice of the weight function $w\,(\cdot)$

Under the assumption that the manifold we are sampling from has a regular shape and $f(\cdot)$ is a smooth density, we can reasonably expect that if two points on the manifold are somehow close then the associated values of the density f are also similar to each other. In typical application of our model this assumption is usually

satisfied since $S(k)_i$ intersects with at least $k-1$ other spheres and the closer two spheres are, the bigger intersected area they have. Thus, it is plausible that densities evaluated on nearby spheres are more similar to the one that we want to estimate. Hence, to estimate f_i once the observation x_i is removed, we propose to average the values of $\widehat{\mathbf{f}}(k)_{-i}$ with a weight function $w(\cdot)$ that soften the impact of density estimates related to regions far from x_i. One of the most straightforward choice for $w(\cdot)$ is a symmetric D-dimensional kernel centered at x_i. Since one can not take for granted that densities in spheres that doesn't intersect with $S(k)_i$ are similar to f_i, we propose instead the weight function

$$w_I\left(\widehat{\mathbf{f}}(k)_{-i}\right) = \sum_{j \in G} \frac{\widehat{\mathbf{f}}(k)_j}{h(x_i,\, x_j)} \qquad G = \left\{ j :\, S(k)_j \cap S(k)_i \neq \emptyset \right\}, \tag{3.9}$$

where $h(\cdot)$ is a reasonable distance function which takes into account only densities in spheres that intersect with $S(k)$. The choice of $h(\cdot)$ depends on how much importance we want to give to nearby points. A standard choice is the euclidean distance, but since the intersected volume does not grow linearly with the distance, something more sophisticated might be used.

3.2.3 The choice of $\mathfrak{F}[\cdot]$, $d(\cdot)$ and $g(\cdot)$

Different combinations of $\mathfrak{F}[\cdot]$, $d(\cdot)$ and $g(\cdot)$ lead to different CV scores. Wise choice for them can lead to different properties. One of the simplest choice for $\mathfrak{F}$ is the expected value with respect to the predictive distribution

$$\mathfrak{F}[\cdot] = \mathbb{E}_{X^*_{-i}}[\cdot] = \int_{\mathbb{N}} [\cdot]\, \pi\left(x^*_{-i}\right) dx^*_{-i}$$

but more robust criteria can be used (e.g. the median).

Usually the function $g(\cdot)$ is the identity function $\mathbb{I}(\cdot)$, this choice leads to a direct comparison between X^*_{-i} and k. Another choice for $g(\cdot)$ can be any representative value (e.g. the mean) of the predictive distribution X^*_{-i} comparable with k. The function $d(\cdot)$ can be any distance function. The sum of the absolute or squared difference are the standard choice but, since the variables involved are actually counts, distances between small numbers could have a bigger impact than distance between bigger values even if their euclidean distance is the same. For this reason more fine tuned distance functions may be used.

3.2.4 Numerical Results

In this section we collect the results of three simulation studies showing the performance of the proposed method. The setting we used is $w(\cdot) = w_I(\cdot)$, $\mathfrak{F}[\cdot] = \mathbb{E}_{X^*_{-i}}[\cdot]$, $g(\cdot) = \mathbb{I}(\cdot)$ and the L^1–norm. Our cross validation score is then

$$C_1(k) = \frac{1}{n}\sum_{i=1}^{n} \|k - \mathbf{x}_i^*\|_1 .$$

Table 3.1 shows the results of estimating the intrinsic dimension with the original MLE using the optimal (cross–validated) neighbors size k^*. Results are given for an extension of Algorithm 1 that performs a 5–fold Cross–Validation under different sampling schemes. On each experiment we draw n points uniformly from a d-dimensional manifold embedded in $\mathbb{R}^D$ and corrupt each observation with an additive Gaussian noise $\eta_i \sim \frac{\sigma}{\sqrt{D}} N(0, I_D)$. We consider three different manifolds: the first is a 9-dimensional unit sphere embedded in $\mathbb{R}^{100}$ with noise level $\sigma = 0.1$; the second is a bi–dimensional rectangular region composed by two unit squares glued together having different densities levels (see Figure 3.2 for an example) embedded in $\mathbb{R}^5$ with noise level $\sigma = 0.001$; the third is a 2–dimensional unit cube embedded in $\mathbb{R}^5$ with noise level $\sigma = 0.01$ and also without noise. For each manifold we vary the number

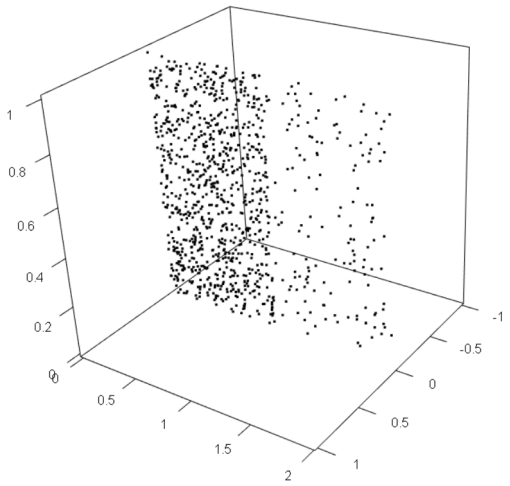

Figure 3.2: Example of 1000 samples drawn from a Twosheets manifold

of samples $n = (100, 500, 1000)$. Each experiment has been iterated 100 times and the results are averaged among those iterations. Table 3.1 shows how the estimates of the intrinsic dimension $\widehat{d}_{k^*}$ are pretty close to the true value d, with a relatively small standard error which tends to decrease as the number of samples increases.

Manifold	n	σ	D	d	$\mathbb{E}[\widehat{d}_{k^*}]$	$\mathrm{sd}[\widehat{d}_{k^*}]$
Sphere	100	0.1	100	9	8.49	1.01
	500	0.1	100	9	8.8	0.5
	1000	0.1	100	9	9.03	0.36
Twosheets	100	0.01	5	2	1.96	0.27
	500	0.01	5	2	2.03	0.11
	1000	0.01	5	2	2.08	0.08
Hypercube	100	0.01	5	2	2.19	0.25
	500	0.01	5	2	2.59	0.15
	1000	0.01	5	2	2.52	0.33
	1000	0	5	2	1.98	0.08

Table 3.1: Results on the performance of the MLE $\widehat{d}_{k^*}$ using the cross–validated neighbors size k^* obtained extending Algorithm 1 to perform a 5–fold Cross–Validation. Each experiment, described in Section 3.2.4, consider different manifolds with different sample sizes and it is iterated 100 times.

3.3 SURE

Since computational approaches to determine the optimal value for k (or r) can be very slow, in future work it could be useful to find a criterion to do it analytically. The Stein's Unbiased Risk Estimate (SURE) theory, James and Stein (1961), can help with this issue. The SURE method allows to have an unbiased estimate of the mean-squared error (MSE) from the data without requiring knowledge of the parameter true value. Let $\boldsymbol{h}(\boldsymbol{u}) = \hat{\boldsymbol{\theta}}_\alpha$ where $\boldsymbol{h}(\boldsymbol{u})$ is a function of a sufficient statistics $\boldsymbol{u}$ and $\hat{\boldsymbol{\theta}}_\alpha$ is an estimate for the parameter $\boldsymbol{\theta}$ that depends from a tuning parameter α. What we want to do is to find an optimal value for α that minimize the MSE of $\boldsymbol{h}(\boldsymbol{u}) = \hat{\boldsymbol{\theta}}_\alpha$. This can be written as

$$\mathbb{E}\left\{\left\|\hat{\boldsymbol{\theta}}_\alpha - \boldsymbol{\theta}\right\|^2\right\} = \|\boldsymbol{\theta}\|^2 + \mathbb{E}\left\{\|\boldsymbol{h}(\boldsymbol{u})\|^2\right\} - 2\mathbb{E}\left\{\boldsymbol{h}^T(\boldsymbol{u})\,\boldsymbol{\theta}\right\}.$$

In order to minimize the MSE over $\boldsymbol{h}(\boldsymbol{u})$ it is enough to minimize

$$v(\mathbf{h}, \boldsymbol{\theta}) = \mathbb{E}\left\{\|\boldsymbol{h}(\boldsymbol{u})\|^2\right\} - 2\mathbb{E}\left\{\boldsymbol{h}^T(\boldsymbol{u})\,\boldsymbol{\theta}\right\}$$

over $\boldsymbol{h}(\boldsymbol{u})$, which is impossible since it depends from the true value of $\boldsymbol{\theta}$ that we want to estimate. The second term of $v(\mathbf{h}, \boldsymbol{\theta})$ is problematic since it depends explicitly from $\boldsymbol{\theta}$. The SURE idea is based on estimating this term with an unbiased estimator that depends only on $\boldsymbol{u}$ in order to obtain an unbiased estimator for the MSE.

Part II

Low Dimensional Representation for High Dimensional Data

Chapter 4

Introduction to the MSVD approach

As mentioned in Chapter 1, a local singular value decomposition (SVD) may be used to estimate the intrinsic dimension of a point cloud. More in particular in this chapter we will focus on the approach described in Little et al. (2009b) where an estimator of the intrinsic dimension is proposed that exploits a careful study of the behavior of the singular values as functions of the radius of the sub–regions where the SVD is actually performed. This approach computes the singular values

$$\left\{\lambda_1^{(x,r)}, \ldots, \lambda_D^{(x,r)}\right\},$$

sorted in non-increasing order, on the data points $X_D^{(x,r)}$ belonging to a D-dimensional sphere $S_D^{(x,r)}$, with radius r centered at a point $x \in X$, and repeat the process for different values of r. This is the reason why this technique is called Multiscale Singular Value Decomposition (MSVD). The method is based on the study of the behavior of the SVs $\widehat{\lambda}_d(r)$ when r grows. It has been shown in Little et al. (2009b) that it exists a set of values of r such that the SVs may be grouped in three different categories according to their different growth–rate as functions of r; that is,

- the "just–noise" SVs which are almost flat;
- the "dimensionality" SVs which are the top d SVs and grow linearly w.r.t. r;
- the "curvature" SVs which grow at most linearly with r^2.

These different behaviors allow us to separate the SVs using, for instance, a simple least squared fit to the SVs and then estimating the intrinsic dimension d as the number of the non–curvature SVs among the non–noise SVs. The main issue with this approach is the choice of a good region for the values of r to perform the least squared fit. Small values of r may lead to an insufficient number of samples and

thus to noise dominated SVs. On the other hand, for large values of r, the SVs may lose the properties described above, as the curvature SVs will grow at the same rate of the dimensionality SVs. This problem surfaces also in Little et al. (2009a) and Little et al. (2011), where it is shown that, under mild assumption on the manifold structure, it exists a region R of optimal values of r such that the the larger gap $\Delta_l = \lambda_l^{(r)} - \lambda_{l-1}^{(r)}$ will be Δ_d. Thus, given R, it is straightforward to estimate the intrinsic dimension d. Numerical results and comparison between MSVD and other approaches show that it performs well even with low sample size and that it is less sensitive than other methods to noise in the data, although it is still an open problem to locate the region R and find an optimal estimator for r. These problems will be addressed in the following sections.

4.1 The MSVD algorithm

The methods and algorithms that will be introduced in the following sections will heavily rely on a MSVD of the data. For this reason, in this section, we brieflydescribe a very basic algorithm (Algorithm 2) to perform the local MSVD.

Algorithm 2 Algorithm to perform local MSVD on Data

Input: An $(n \times D)$ dataset X, a point x in X, a bound d_0 on intrinsic dimension
Output: A $(D \times J)$ MSVD matrix $\mathbf{\Lambda}$
Steps:

Define $n_{d_0} = d_0 \log(d_0)$
Define $J = n/n_{d_0}$
• **for** $j = 1 : J$
Find $r_j = \min\left(r : S_D^{(x,r)} \text{ contains } j \cdot n_{d_0} \text{ points}\right)$
Perform SVD on points in $S_D^{(x,r_j)}$ to obtain $\left\{\lambda_l^{(x,r_j)}\right\}_{l=1,\dots,D}$
return: $\mathbf{\Lambda}$ such that $\mathbf{\Lambda}_{l,j} = \lambda_l^{(x,r_j)}$
• **end**

Algorithm 2 requires as input a dataset matrix X, a point $x \in X$ and an a priori upper bound d_0 on the intrinsic dimension. The latter parameter is not strictly needed as it can be set equal to D, but in practice the ambient space dimension D may be very large and the sample size n may be not big enough to have fine enough scales to do a multiscale analysis. Thus, the use of $d_0 \ll D$ is strongly suggested, but again may not be necessary. The choice of the minimal number of points on which to perform the SVD is $n_{d_0} = d_0 \log(d_0)$ because with fewer points it could be impossible to compute the top d_0 single values. For each scale j the SVD is performed on the $j \cdot n_{d_0}$ points closest to x in euclidean distance. In other words, the SVD is performed on the points contained in the sphere $S_D^{(x,r_j)}$, where r_j is the minimum radius of the D-dimensional sphere centered iat x, that contains $j \cdot n_{d_0}$ observations. The output of the algorithm is then a $(D \times J)$ matrix $\mathbf{\Lambda}$. The j^{th} column of $\mathbf{\Lambda}$ represents the singular values at scale j sorted in non–increasing order. Therefore the l^{th} row of $\mathbf{\Lambda}$ can be interpreted as the variation of the l^{th} top singular value with respect to r. Figure 4.1 shows the result of applying the MSVD on 10^5 points uniformly sampled on the surface of a 9–dimensional unit sphere embedded in $\mathbb{R}^{100}$. Each point is corrupted with additive Gaussian noise $\eta_i \sim \frac{\sigma}{\sqrt{D}} N(0, I_D)$ with $\sigma = 1$ and $D = 100$. Each line then represents the singular value evolution with respect to the radius r of the neighborhood of x. From Figure 4.1 we notice the behavior described previously: it is easy too see that, approximately in the region $r = [1.6, 1.9]$, the biggest gap between the singular values is exactly Δ_9. This gap

correctly identifies the intrinsic dimension to be 9. For larger values of r the biggest gap will be Δ_{10}, because the constant curvature become a predominant element in the neighborhood on which the SVD's are computed. Also, it is easy to see that the growth–rate of the 10^{th} SV is higher than the first 9 SV's. In the next chapters we will introduce a method to automatically find a good region that will return Δ_d as the biggest gap between the singular values.

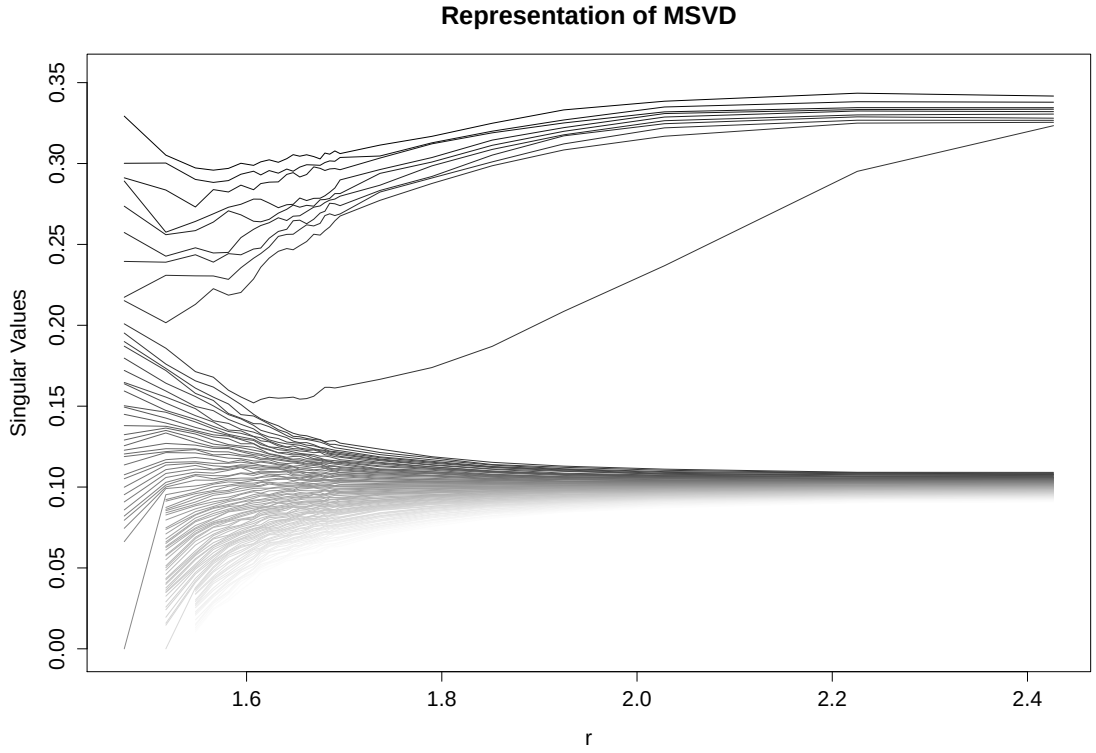

Figure 4.1: Representation of a MSVD applied to 10^5 points uniformly sampled from a unit 9–dimensional sphere embedded in $\mathbb{R}^{100}$ adding Gaussian noise $\eta_i \sim N\left(0, \frac{1}{\sqrt{100}} I_{100}\right)$.

Chapter 5

Model Selection for Plane Arrangement

In many applications, like computer vision (Sugaya and Kanatani (2004)), pattern recognition (Ma et al. (2008)) and image processing (Hong et al. (2006)), the data cloud, usually observed in high dimension, can be modelled with hybrid linear models using a union of low-dimensional planes. This approach has shown promising results in model selection, clustering and classification problems. The main practical problem to find a set of planes approximating a point cloud laying in high dimension consists in choosing the number of planes and their dimension. One of the algorithm proposed to solve these problems is the Agglomerative Lossy Coding (ALC) (Ma et al. (2007)). However, the ALC algorithm is very sensitive to the value of a tunable but otherwise unknown tolerance parameter which can lead, if changed, to very different number of clusters. In addition the ALC was found to be slow (Chen and Maggioni (2011)), does not come with any finite sample guarantees and the number of iteration needed to converge is unknown. Another algorithm that tackle both the model selection and the clustering problems is proposed in Chen and Maggioni (2011). This algorithm is based on a combination of Multiscale Singular Value Decomposition (Little et al. (2012)) and Spectral Clustering (Chen and Lerman (2009)). We focus on this last algorithm and improve its performances introducing a new method to estimate the intrinsic dimension of the planes.

5.1 Model Assumptions

Let the data set $\mathbf{X} = \{x_1, \ldots, x_n\}$ be a point cloud observed in $\mathbb{R}^D$ and sampled around a collection of K affine planes $\pi_1, \ldots, \pi_K$ of (true) dimensions $d_1, \ldots, d_K$ respectively. We require some assumptions on the distribution of the points on the planes to ensure a good and fast convergence of the proposed algorithms. Let μ be a probability measure in $\mathbb{R}^D$ with support in $\mathbf{Q}_{\mathbf{M}}^D \cap \left(\cup_{k=1}^K \pi_k\right)$, where $\mathbf{Q}_{\mathbf{M}}^D$ is the hypercube with edge length $\mathbf{M}$. We assume measure zero on the planes intersection, more formally

$$\mu\left(\pi_k \cap \pi_{k'}\right) = 0 \quad \text{for} \quad \pi_k \neq \pi_{k'}.$$

We also assume that it exists a positive constant c_1 such that

$$\mu\left(\pi_k\right) \geq \frac{c_1}{K}, \quad \forall\, k.$$

Now, we introduce some local regularity assumptions around a generic point $x \in \pi_k$. Let μ_k be the probability measure conditioned to the plane π_k, it exists a positive constant c_2 such that

$$\frac{r^{d_k}}{c_2} \leq \mu_k\left(S_D^{(x,r)}\right) \leq c_2 r^{d_k} \quad \forall\, r \leq \mathbf{M},$$

which ensure that the measure on the sphere $S_D^{(x,r)}$ is bounded by a factor proportional to its radius. Now Let $\mathbf{X_{k,x,r}}$ be a random variable with distribution μ_k restricted on the sphere $S_D^{(x,r)}$, then

$$\left\{\lambda_l^{(x,r)}\right\}_{l=1,\ldots,D}^2 \subset \frac{r^2}{d_k}\left[\lambda_{\min}, \lambda_{\max}\right] \quad \forall\, r > 0. \tag{5.1}$$

Where $\lambda_{\min}$ and $\lambda_{\max}$ are fixed positive constant. Assumption in Equation (5.1) ensure a regular shape of the ellipsoid represented by the SVs in $S_D^{(x,r)}$.

Now, for each $x \in \pi_k$ let

$$\mathbf{E}_x = \left\{r : S_D^{(x,r)} \cap \left(\cup_j^K \pi_j\right) = \emptyset\right\} \quad \text{for}\, j \neq k. \tag{5.2}$$

$\mathbf{E}_x$ represents the set of values of r such that the sphere built around x does not hit other planes and let $r_x^{\max} = \sup \mathbf{E}_x$. Let each point x_i be observed after a random Gaussian noise corruption $\eta_i \sim \frac{\sigma}{\sqrt{D}} N\left(0, I_D\right)$. Assume that

$$\exists\, c_3 < 1 : \mu_k\left(r_x^{\max} > \sigma\right) \geq 1 - c_3. \tag{5.3}$$

This assumption, jointed with the others in this section, ensure that, for c_3 small enough, there is a granted probability that it exists a non empty set of values of r such that $S_D^{(x,r)}$ contains only points in π_k and that the SVs on those points will not be dominated by the noise. Finally, the next assumption will guarantee that the computational time of our algorithm does not dependent on the sample size n. To be more specific, it exists a value $c_5\,(c_2, c_3, \sigma) > 0$ such that, if

$$n_1 \geq c_5\, K \log(K)\, d\, \log(d),$$

then, with high probability, the sphere of radius $r_x^{\max}$ around a randomly sampled point x contains enough points that lay beyond the noise zone. As shown in Little et al. (2009a), this will ensure an accurate estimate of the covariance of the points in the sphere $S_D^{(x,r_x^{\max})}$. It is also assumed that the probability to sample a points in the $k^{\texttt{th}}$ plane is proportional and close to $1/K$.

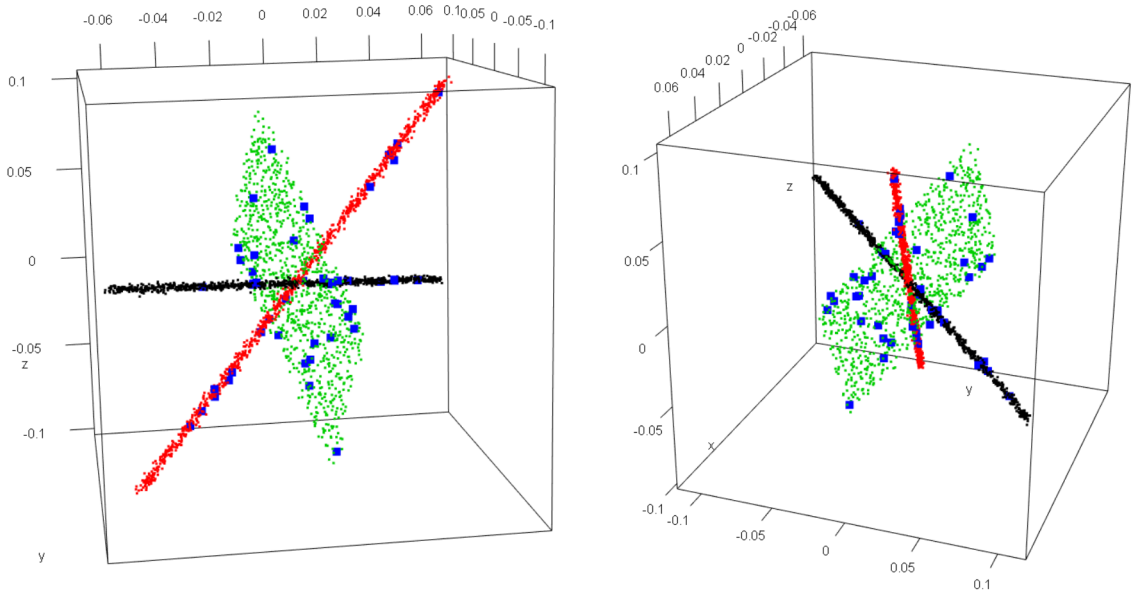

Figure 5.1: Example of points uniformly drawn on 3 hyperplanes embedded in $\mathbb{R}^6$ of dimension $d = (1, 1, 2)$ respectively. Black, red and green dots represent the points on the two lines and the plane respectively while the blue dots represent the randomly sampled points used as centers in Algorithm 4.

5.2 Local Intrinsic Dimension Estimation

In this section we propose an algorithm (Algorithm 3) for the local estimation of d and the choice of r. Let x_i be a random data point and perform Multiscale Singular Value Decomposition described in Algorithm 2 (Section 4.1) to obtain a $(D \times J)$ matrix $\mathbf{\Lambda}$, which each j^{th} column represents the Singular Values $\{\lambda_{lj}\}_{l=1}^{D}$ at the scale j in decreasing order. In this section, in order to simplify the notation, we drop the i index (e.g. $x_i = x$), although all the quantities that follow are still considered to be local conditioned to x_i. When r is small, but large enough to move away from the noise zone, with high probability all the points inside the sphere belong to the same plane π_k. Thus the first d_k Singular Values are big, while the other $D - d_k$ are close to σ. As r increases, the top d_k Singular Values grow linearly until the radius reach the change point value $r^{\max}$ at which the sphere hits another plane π'_k. At this point the sphere starts to include points from π'_k. Thus, at least another Singular Value, after the first d_k, should start to move away from 0 growing linearly with r.

Bearing these features in mind – which are also shown in Figure 5.2 – the idea that we propose to find $r^{\max}$ is based on the use of piecewise linear regressions to fit each Singular Value path independently

$$\mathbb{E}(\lambda_l) = \begin{cases} \alpha\, r & \text{for } r \leq r_l^* \\ \beta\, r & \text{for } r > r_l^* \end{cases} \tag{5.4}$$

with the constrains $\alpha \leq 0$ and $\beta > 0$. When r_l^* is also unknown, the parameters estimation turn into a nonlinear optimization problem. However, we can reduce the whole procedure to a linear optimization problem. Since we have only J values for r, which are the values of the radius r_j at each scale, we can vary r_l^* among all the values of r_j and estimate, for each λ_l, the change point $r_l^{\max}$ using the value of r_l^* that leads to the model with the best fit in term of mean squared error. Since Singular Values may change their slopes every time the growing sphere hits another planes, if we take into account all the values of λ_l and r, the piecewise regression may fail to give a good estimate of the first change point. To overcome this problem we fit the piecewise linear regression, described above, using only the first j values of λ_l for l in $1, \ldots, D$. This process is iterated for increasing values of j. At each iteration we apply a k–means algorithm with two cluster to the resulting D values $r_t^{\max}$, for t in $1, \ldots, J$, breaking at t^* when the difference between the two clusters is bigger than a tolerance value δ. In this way we obtain two well separated sets of change points. The lower change point of the first set is a good estimate for the value of r needed for the sphere to include points outside the noise zone. The lower

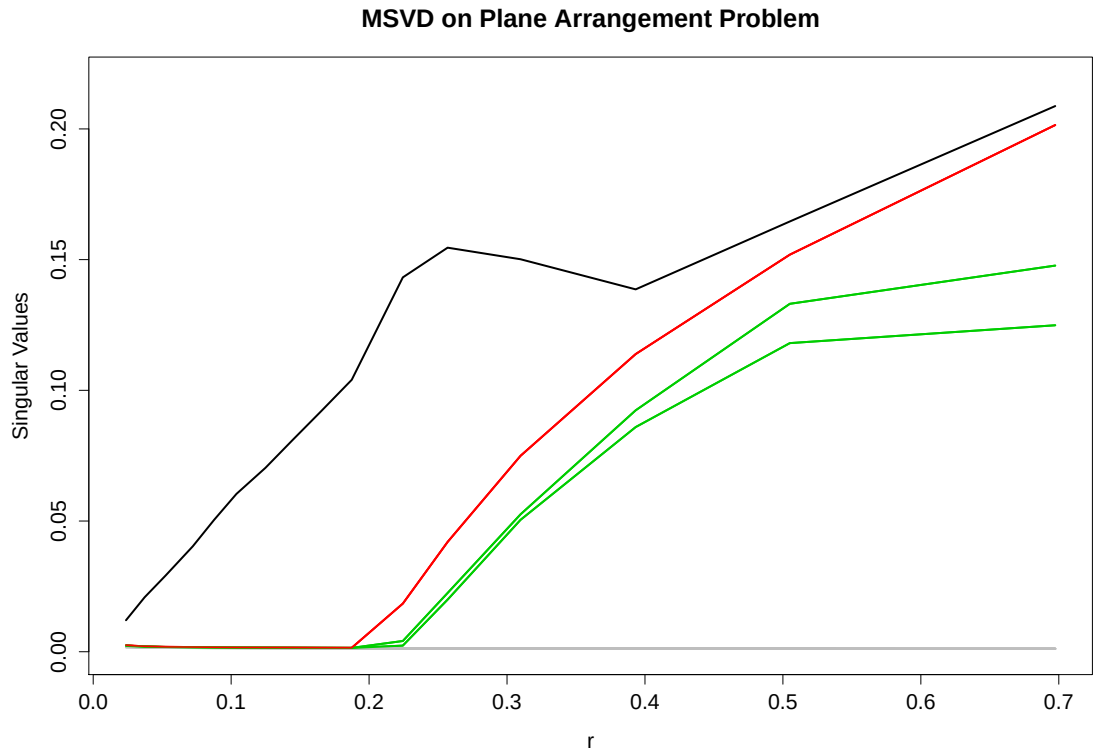

Figure 5.2: A Representative image of a MSVD on the data shown in Figure 5.1. We use for center x a point generated from the black plane ($d = 1$). Here one can appreciate the behavior described in Section 5.2. For small values of r only the first SV grows almost linearly. As r increase, around 0.2, the neighborhood of x includes points of the red plane ($d = 1$) and the second SV starts to grow linearly too. Around $r = 0.23$ the sphere hits the green plane ($d = 2$), thus, the next two SVs start to grow linearly. The remaining $2 = (D - \sum_{k=1}^{K} d_k)$ SVs stay flat because they represents only noise. **Note**: in typical applications we do not know a priori from which plane x is generated or in which order the planes are hit by the sphere, the colors of this figure are only chosen for the sake of a better comprehension.

change point of the second group is a good estimate of the value of r needed for the sphere to hit another plane. So value $r_{i,t^*}^{\max}$ can be used to estimate $r_i^{\max}$ as it represents the radius needed for the d_k+1 Singular Values to move away from 0 (i.e. have a positive slope). At this point a SVD analysis on the points in sphere $S_D^{(x,\, r^{\max})}$ gives an estimate of d, defined by the biggest SV gap Δ_d. The top d eigenvectors, along with the barycenter of the points in the sphere, give a representation of the plane π approximating the neighborhood of x.

Algorithm 3 Algorithm for finding d, $r_l^{\max}$ and $\widehat{\pi}$

Input:

- X: A $(n \times D)$ data matrix
- x: A center point
- δ: A tolerance level

Output:

- $\widehat{d}$: Local Intrinsic dimension estimate
- $r^{\max}$: Distance to the closest plane
- $\widehat{\pi}$: Best d-dimensional plane approximating the neighborhood of x
- $\widehat{\epsilon}$: Mean squared error of plane $\widehat{\pi}$

Steps:

Perform Multiscale SVD on X to obtain $\mathbf{\Lambda}$

- **for** $j = 1 : J$
  - **for** $l = 1 : D$
    - **for** $t = 1 : j$
      - use the model in Equation (5.4) with the first j values of λ_l and r, set $r_l^* = r_t$ and compute its MSE err_t
      - **end "for** t**"**
    - for each of the D values λ_l set the best change point $r_l^{\max}$ to the value of r_t which leads to the smaller err_t
    - **end "for** l**"**
  - perform a two clusters k–means on the D values $r_l^{\max}$
  - **if** the distance between the two groups is bigger than the tolerance δ **then** • estimate $r^{\max}$ with the lower value of $r_l^{\max}$ in the higher cluster
    - let $\widehat{d}$ be the rank $g^\star$ of the biggest value of $\Delta_g = \Lambda_{g,r^{\max}} - \Lambda_{g-1,r^{\max}}$
    - compute $\widehat{\pi}$ as the best d-dimensional plane on points in $S_D^{(x,\, r^{\max})}$
    - compute $\widehat{\epsilon}$ as the fitted mean squared error of plane $\widehat{\pi}$
    - **break "for** j**"**
    - **return** d, $r^{\max}$, $\widehat{\pi}$ and $\widehat{\epsilon}$
    - **end algorithm**

    else • **continue "for** j**"**
- **end "for** j**"**
- **return:** *"w.h.p x is close to the intersection of all the planes or $d = D$"*
- **end algorithm**

5.3 Global model estimation

In this section we propose a method (Algorithm 4) based on the idea of Chen et al. (2009), to find and estimate the number of planes K, the dimensions $d_1, \ldots, d_k$ and how the planes $\pi_1, \ldots, \pi_K$ are arranged in the space. The basic idea is to perform Algorithm 3 on n_0 subsamples of X in order to obtain a set of planes $\widehat{\pi}_1, \ldots, \widehat{\pi}_{n_0}$. Since many of these planes will be an estimate of the same π_k, a spectral clustering on the plane estimates is performed to find the true number of planes K. In Chen and Maggioni (2011) it is proposed a spectral approach that aligns the planes to rebuild the original plane arrangement. To have a good confidence of having, on average, approximately $c_4 > 0$ subsampled points for each plane, we need $n_0 \geq c_4\, K\, \log(K)$. This requirement have the same motivations as the famous coupon collector's problem, with the only difference that the coupons are represented by the planes. Since K is usually unknown, a parameter $K_{\max}$ representing a priori information about the maximum number of planes K should be plugged into the equation. Note that the parameter $K_{\max}$ is not strictly required as one can choose $n_0 = n$, but it is strongly suggested to improve the computational time of the algorithm. Algorithm 3 on n_0 subsampled points returns the set $\left\{\widehat{\pi}_j, \widehat{d}_j, \widehat{\epsilon}_j\right\}_{j=1}^{n_0}$. To perform spectral clustering we have to define an $(n \times n_0)$ affinity matrix A such that

$$A_{ij} = \exp\left\{-\frac{\mathcal{D}\left(x_i, \widehat{\pi}_j\right)}{2\widehat{\epsilon}_j^2}\right\} \tag{5.5}$$

where $\mathcal{D}\left(x_i, \widehat{\pi}_j\right)$ represents the euclidean distance between the point x_i and its orthogonal projection on the plane $\widehat{\pi}_j$. For the Gaussian kernel, $\widehat{\epsilon}_j$ is a scale parameter that penalizes planes with high local error. Algorithm 4 for k in $1, \ldots, K_{\max}$, iteratively encode U_k as the matrix containing the first k left singular vector matrix of A – after an opportune normalization detailed in Algorithm 4. Let V_k be a normalization of U_k such that the vectors will have unit length. We then apply the k–means algorithm to the row vectors of V_k in order to obtain k clusters $\widehat{\chi}_{tk}$, for t in $1, \ldots, k$. Let $\widehat{d}_{tk}$ be the mode of the $\widehat{d}_j$ of the points assigned to $\widehat{\chi}_{tk}$ and compute the best $\widehat{d}_{tk}$-dimensional plane on the cluster $\widehat{\chi}_{tk}$ using principal component analysis. Now compute the k–planes model error as

$$e^2(k) = \frac{D}{n} \sum_{t=1}^{k} \sum_{x \in \widehat{\chi}_{tk}} \frac{\mathcal{D}\left(x_i, \widehat{\pi}_j\right)}{D - \widehat{d}_{tk}}.$$

The k–planes model error $e^2(k)$ is computed for increasing values of k. The Algorithm 4 stops when

$$e^2(k) \leq \tau^2 = \frac{D}{n_0} \sum_{j=1}^{n_0} \frac{\widehat{\epsilon}_j^2}{D - \widehat{d}_j}, \tag{5.6}$$

and it returns

$$\widehat{K} = \min \left\{ k : e^2(k) \leq \tau^2(k) \right\}.$$

In other words, the optimal number of planes $\widehat{K}$, is the first value of k such that the error of the aligned k–planes model is lesser or equal to the model error estimate τ^2. Note that during this process we already computed:

- the estimates of the best $\widehat{K}$ planes $\widehat{\pi}_1, \ldots, \widehat{\pi}_{\widehat{K}}$ when we performed PCA on the clusters $\left\{ \widehat{\chi}_{t\widehat{K}} \right\}_{t=1}^{\widehat{K}}$;
- the plane dimensions $\widehat{d}_1, \ldots, \widehat{d}_{\widehat{K}}$ when we computed the mode of the $\widehat{d}_j$ values of the points assigned to $\widehat{\chi}_{tk}$.

Thus, we already have everything we need to reconstruct the plane arrangement. As a final remark notice that the computational time of Algorithm 4 is independent from n, this is a extremely good property when it is needed to analyze a dataset with a large number of observations.

Algorithm 4 Algorithm for Plane Arrangement Model Parameters

Input:

- X: The data
- δ: A tolerance level for Algorithm 3
- d_0: A priori bound for $\max(d_k)$ (optional)
- $K_{\max}$: A priori bound for the number of planes K (optional)
- c_4: The subsampling constant for n_0 (optional)
- c_5: The subsampling constant for n_1 (optional)

Output:

- $\widehat{K}$: estimate for the number of planes
- $\widehat{d}_1, \ldots, \widehat{d}_{\widehat{K}}$: estimates for the planes dimension
- $\widehat{\pi}_1, \ldots, \widehat{\pi}_{\widehat{K}}$: best planes approximating the plane arrangement
- $e_{\widehat{K}}$: estimated model error
- $\{\widehat{\chi}_t\}_{t=1}^{\widehat{K}}$: clusters

Steps:

– Replace X with a subsample of $n_1 = c_5\, K_{\max} \log(K_{\max})\, d_0 \log(d_0)$ points
– Subsample $n_0 = c_4\, K_{\max} \log(K_{\max})$ points from X
– Perform Algorithm 3 on the subsampled points and obtain $\left\{\widehat{\pi}_j, \widehat{d}_j, \widehat{\epsilon}_j\right\}_{j=1}^{n_0}$
– Compute A as shown in Equation (5.5)
– Normalize A to $L = \operatorname{diag}\left(AA'\mathbf{1}\right)^{-1/2} A$
– Let $U = [u_1, \ldots, u_{K_{\max}}]$ with u_k the k^{th} left singular vectors of L
– Compute $\tau^2 = \frac{D}{n_0} \sum_{j=1}^{n_0} \frac{\widehat{\epsilon}_j^2}{D - \widehat{d}_j}$

• **for** $k = 1 : K_{\max}$
– Define $U_k = [u_1, \ldots, u_k]$
– Let V_k be the matrix U_k with normalized row vector to the unit length
– Apply k–means algorithm to the rows of V_k to obtain k clusters $\{\widehat{\chi}_{tk}\}_{t=1}^{k}$
– Let $\widehat{d}_{tk}$ be the mode of the $\widehat{d}_i$'s of the points assigned to $\widehat{\chi}_{tk}$
– Do PCA on $\{\widehat{\chi}_{tk}\}_{t=1}^{k}$ and let $\widehat{\pi}_{tk}$ be the best $\widehat{d}_{tk}$–dimensional plane
– Compute $e^2(k)$ as defined in Equation (5.6)
• **if** $e(k) > \tau$ **end "for k"**
– set $\widehat{K} = k$ and $\{\widehat{\chi}_t\}_{t=1}^{\widehat{K}} = \{\widehat{\chi}_{tk}\}_{t=1}^{\widehat{K}}$
• **return** $\widehat{K}$, $\left\{\widehat{d}_1, \ldots, \widehat{d}_{\widehat{K}}\right\}$, $\{\widehat{\pi}_1, \ldots, \widehat{\pi}_{\widehat{K}}\}$, $e_{\widehat{K}}$ and $\{\widehat{\chi}_t\}_{t=1}^{\widehat{K}}$
• **end algorithm**

5.4 Simulations

In this section we show some empirical results on the performances of Algorithm 4. We denote a collection of K planes of dimension $(d_1, \ldots, d_K)$ in $\mathbb{R}^D$ by $(d_1, \ldots, d_K; D)$. On each simulation we sample 200 points uniformly on K unit hyper-cubes of dimension $d_1, \ldots, d_K$ respectively. Thereafter we corrupt each observation by adding a Gaussian noise $\eta_i \sim \frac{0.04}{\sqrt{D}} N(0, I_D)$, we rotate each hyper-plane randomly in D directions and make them intersect in the origin. We ran Algorithm 4 on 1000 iterations in different scenarios, varying the number of planes K, their dimension $(d_1, \ldots, d_K)$ and the embedding dimension D. The performance of the algorithm are shown in Table 5.1 where we collect the number of planes K, their dimension d and the success rate in assigning each point to the right plane. As we can see our novel method for estimating the dimension of the planes performs very well in all the proposed scenarios and the clustering success rate is also quite good once consider that, by construction, all the planes intersect in the origin, and so we might have some physiological error for observation close to this point.

	(1,2,3;6)	(1,2,3;10)	(1,1,2;6)
K	0.93 (0.25)	0.90(0,29)	0.88(0.31)
d	1	1	1
`cl`	0.88(0.03)	0.91((0.02)	0.91(0.07)
`time`	5.7(0.8)	6.6(0.2)	7.5(0.8)

	(1,2,3;10)	(1,1,3,3;6)	(1,1,3,3;10)
K	0.88(0.32)	0.27(0.44)	0.7(0.46)
d	1	0.982 (0.07)	0.999 (0.06)
`cl`	0.91(0.08)	0.89(0.06)	0.94(0.02)
`time`	6.4(0.2)	10.2(4.2)	10(0.8)

Table 5.1: Performances of Algorithm 4 in six different scenarios iterated 1000 times. The values on row K represent the success rate in estimating the number of planes. The values on row d represent the success rate in estimating the dimension of the planes when the number of planes is correctly identified. The values on row `cl` represent the success rate in assigning the points to the correct plane when the number of planes is correctly estimated. The `time` row shows the average time taken by each iteration in seconds. The values in parentheses are the standard errors.

Chapter 6

Manifold Learning

In Section 5 we proposed a method to estimate the model parameter when the data are generated from an unknown collection of different planes with different intrinsic dimension embedded in high dimension and perturbed with gaussian error. In the next sections we propose a method to approximate the model when the data are assumed to be generated from a manifold $\mathcal{M}$ with intrinsic dimension d embedded in $\mathbb{R}^D$. A manifold is a topological space that can be locally but not necessarily globally euclidean. In the literature there are several methods available for recovering the underlying manifold from data. These techniques tackle the problem from different perspectives and some of them will be described in the following sections.

Our goal here is to perform a data adaptive piecewise linear multiscale reconstruction of the manifold with guarantees on the approximation error. Also we want our algorithm to be fast, its computational time must be independent from the sample size n, should scale well with the intrinsic dimension d and only negligibly on the embedding dimension D. With the term multiscale we mean that the manifold approximation is made at multiple scales of precision, from coarse scales with lower precision to fine scales with higher precision. With the proposed method we also directly build a data adaptive tree structure for the data which gives nice properties to the approximation and is very useful for other kind of applications. The proposed method is consistent in the sense that for finer and finer scales and with the sample size growing the approximation error tends to zero.

6.1 Background Theory

In the literature there are several nonlinear manifold learning techniques used to recover the full low-dimensional representation of an unknown nonlinear manifold embedded in a high dimensional space. In this section we will briefly describe some of these methods while in the following sections we will propose a new one for manifold approximation.

In this part of our work we assume that a finite number of points, $\{y_i\}$, are randomly sampled from a smooth d–dimensional manifold $\mathcal{M}$ with an unknown metric given by its geodesic distance. These points are then embedded in a nonlinear fashion by a smooth embedding map ψ into a D–dimensional input space $\mathcal{X} = \mathbb{R}^D$ with $d \ll D$. The space $\mathcal{X}$ have an Euclidean metric. The map ψ yields the observed data $\{x_i\}$. Thus, $\psi : \mathcal{M} \to \mathcal{X}$ is the embedding map, and a point on the manifold $y \in \mathcal{M}$ can be expressed as $\phi(x)$, for $x \in \mathcal{X}$, where $\phi = \psi^{-1}$. The goal of a manifold learning technique is to recover $\mathcal{M}$ from $\{x_i\}$ finding the unknown map ψ and hence the $\{y_i\}$. Consequently, in general, the output of these algorithms will be an estimate $\{\widehat{y}_i\} \subset \mathbb{R}^{\hat{d}}$ of the manifold data $\{y_i\} \subset \mathbb{R}^d$. One important aspect is then to estimate correctly the true dimension of the manifold $\mathcal{M}$.

ISOMAP

The isometric feature mapping (i.e. ISOMAP) algorithm, described in Tenenbaum et al. (2000), is based on two main assumptions. The first is that the manifold is a convex region of the embedding space and the second is that the embedding function $\psi : \mathcal{M} \to \mathcal{X}$ is an isometry. The second assumption guarantees that the geodesic distance is invariant or, in other words, that the geodesic distance of any pair of points $y, y' \in \mathcal{M}$ must be equal to the euclidean distance between the same point in the corresponding coordinates $x, x' \in \mathcal{X}$. The first step of the ISOMAP algorithm is to build a neighborhood graph by evaluating all the pairwise distances and determining which data points are neighbors on the manifold $\mathcal{M}$ by connecting each point to its K nearest neighbors. In this way we produce a weighted neighborhood graph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertexes in $\mathcal{V}$ are the data points, the edges in $\mathcal{E}$ gives the neighborhood structure while the weights are proportional to the distance between two connected observations. In the second step the geodesic distance between pairs of points in the manifold is estimated by the shortest path distances between the all pairs of points on the graph $\mathcal{G}$. Finally, in the third and last step (the embedding step), the multidimensional scaling algorithm is applied to reconstruct a d dimensional space in such a way that the geodesic distances on $\mathcal{M}$

are preserved as much as possible. The ISOMAP algorithm may have difficulties with manifolds that contain holes, have high curvature or are not convex. The choice of K is crucial on the success of the algorithm: it must be sufficiently large so that the points can be well-reconstructed but also small enough to have little curvature in the neighbor. Another problem with the ISOMAP algorithm is that on the second and the third step $(n \times n)$-matrices need to be computed and the memory needed to store such matrices could be huge. This problem has been addressed in de Silva and Tenenbaum (2003) with the introduction of the LANDMARK ISOMAP algorithm which tries not to compute redundant distances with a choice of a representative landmark subset of $m \ll n$ data points. The distances are calculated only on this small set of points and then the matrices needed in the second and third step of the algorithm will be of dimension $(m \times n)$.

Local Linear Embedding

The local linear embedding (LLE) algorithm (Roweis and Saul (2000)) for nonlinear dimensionality reduction, similarly to ISOMAP, attempts to preserve the local neighborhood information on the manifold. As before in the first step of the algorithm we compute the K nearest neighborhood of each point x using the Euclidean distance. The second step consists in reconstructing the point x using a linear combination of its neighbors. The optimal weights of the linear combination can be obtained via linear optimization that provides a $(n \times n)$ matrix of optimal weights $\widehat{\mathbf{W}} = (\widehat{w}_{ij})$. The third and final step consists in finding the $(d \times n)$ matrix $\mathbf{Y} = (\mathbf{y_1}, \ldots, \mathbf{y_n})$ of embedding coordinates that solves

$$\widehat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} \widehat{w}_{ij} y_j \right\|^2 ,$$

with the constrains $\sum_n y_i = \mathbf{Y}\mathbb{I}_n = 0$ and $n^{-1} \sum_i y_i y_i^T = n^{-1}\mathbf{Y}\mathbf{Y}^T = \mathbb{I}_d$. These constrains are imposed to set the translation, rotation, and scale of the embedding coordinates so that the objective function will be invariant. Hence the aim of LLE is to preserves the local (rather than global) properties of the underlying manifold.

Laplacian Eigenmaps

The Laplacian Eigenmaps algorithm Belkin and Niyogi (2001) also consists in three steps. The first step of the algorithm is to compute the K nearest neighborhood of each point x using the Euclidean distance with a symmetrical constrain: for a K–neighbor N_i^k of the point x_i, then $x_j \in N_i^k$ if and only if $x_i \in N_j^K$. The second

step, which characterize this algorithm, compute a weight matrix $\mathbf{W} = (w_{ij})$ using the isotropic Gaussian kernel

$$w_{ij} = \begin{cases} \exp\left\{-\dfrac{\|x_i - x_j\|^2}{2\sigma^2}\right\} & \text{if } x_j \in N_i \\ 0 & \text{otherwise} \end{cases}$$

where σ is the scale parameter. In this way we have obtained a weighted graph $\mathcal{G}$. The third and last step consists in the embedding of $\mathcal{G}$ into the low–dimensional space $\mathbb{R}^d$ by the $(d \times n)$ matrix $\mathbf{Y} = (y_1, \ldots, y_n)$, where the i^{th} column of $\mathbf{Y}$ yields the embedding coordinates of the i^{th} point. This is done using the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ for the graph $\mathcal{G}$ where $\mathbf{D}$ is a $(n \times n)$ diagonal matrix where each i^{th} entry is the sum of the weight of the neighbor of x_i. The matrix $\mathbf{Y}$ is then approximated with

$$\widehat{\mathbf{L}} = \underset{\mathbf{YDY}^T = \mathbb{I}_d}{\operatorname{argmin}} \operatorname{tr}\left\{\mathbf{YDY}^T\right\}.$$

Hessian Eigenmaps

In certain situations, the assumption of convexity of the manifold may be too restrictive. The Hessian Eigenmaps algorithm has been proposed for recovering manifolds which may not be convex. Assume that the parameter space is $\Theta \subset \mathbb{R}^d$ and suppose that $\phi : \Theta \to \mathbb{R}^D$, where $d < D$. Also assume that the manifold $\mathcal{M} = \phi(\theta)$ is smooth. The isometry and convexity assumption are dropped and replaced by a local isometry assumption. The function ϕ is a locally isometric embedding of θ into $\mathcal{R}^D$ if for any point x' in a sufficiently small neighborhood around each point x on $\mathcal{M}$, the geodesic distance equals to the Euclidean distance between their corresponding parameter points $\theta, \theta' \in \Theta$ where $x = \phi(\theta)$ and $x' = \phi(\theta')$. Another requirement is that the parameter space Θ must be open and a connected subset of $\mathbb{R}^d$. The goal of the Hessian Eigenmaps algorithm is to recover the parameter vector θ. Let $\mathcal{T}_x(\mathcal{M})$ be a tangent space of the point $x \in \mathcal{M}$. We confer $\mathcal{T}_x(\mathcal{M})$ with a system of orthonormal coordinates having the same inner product as $\mathbb{R}^D$. The tangent space $\mathcal{T}_x(\mathcal{M})$ may be interpreted as an affine subspace of $\mathbb{R}^D$ that is spanned by vectors tangent to $\mathcal{M}$ and intersect the point x, with origin $0 \in \mathcal{T}_x(\mathcal{M})$ identified with $x \in \mathcal{M}$. Define N_x as a neighborhood of x such that each point in this neighborhood has a unique closest point $\xi \in \mathcal{T}_x(\mathcal{M})$. A generic point in N_x has local coordinate, or tangent coordinates, $\xi = \xi(x) = \big(\xi_1(x), \ldots, \xi_d(x)\big)^T$. Let $f : \mathcal{M} \to \mathbb{R}$ be a $\mathcal{C}^2$-function near x. If a point $x' \in N_x$ has local coordinates $\xi = \xi(x) \in \mathbb{R}^d$, then the rule $g(\xi) = f(x')$ defines a $\mathcal{C}^2$-function $g : U \to \mathbb{R}$, where U is a neighborhood of $0 \in \mathbb{R}^D$. The tangent Hessian matrix, which measures the

curvature level of f at the point $x \in \mathcal{M}$, is defined as the ordinary $(t \times t)$ Hessian matrix of g,

$$H_f^{\tan}(x) = \left[\frac{\partial^2 g(\xi)}{\partial \xi_i \partial \xi_j}\right]_{\xi=0}.$$

Thus, the average curvature level of f over $\mathcal{M}$ is represented by the quadratic form

$$\mathcal{H}(f) = \int_{\mathcal{M}} \left\| H_f^{\tan}(x) \right\|_F^2 \mathrm{d}x,$$

where $\|H\|_F^2 = \sum_i \sum_j H_{ij}^2$ is the squared Frobenius norm of a square matrix H. The Hessian Locally Linear Embedding (HLLE) algorithm performs a discrete approximation to the Hessian $\mathcal{H}$ using the data on $\mathcal{M}$. The HLLE algorithm has three steps. In the first step a neighborhood N_i^K of each point x_i is created with a K nearest neighbors based on euclidean distances. In the second step the tangend Hessian Matrices are estimated. This is done computing a $(D \times D)$ covariance matrix M_i of the K points in N_i^K and compute a PCA on it. Assuming $K \geq d$, the first d eigenvectors of M_i yield the tangent coordinates of the K points in N_i^K and provide the best-fitting d–dimensional planes corresponding to x_i. All the square and cross products of the columns of M_i are computed, up th the d^{th} order, and set the $1 + d + d(d+1)/2$ obtained vectors to be the columns of the matrix Z_i. Then apply the Gram-Schmids orthonormalization to Z_i, and let the estimate of the tangent Hessian matrix $\widehat{H}_i$ be the transposed of the last $d(d+1)/2$ orthonormal columns pf Z_i The third and last step combines the estimates local Hessian matrices $\widehat{H_i}, i = 1, \ldots, n$ to construct a sparse symmetric, $(D \times D)$ matrix $\widehat{\mathcal{H}} = \widehat{\mathcal{H}}_{kl}$, where

$$\widehat{\mathcal{H}} = \sum_i \sum_j \left[\widehat{H_i}\right]_{jk} \left[\widehat{H_i}\right]_{jl}.$$

$\widehat{\mathcal{H}}$ is a discrete approximation of the functional $\mathcal{H}$. Now we perform an eigenanalysis of $\widehat{\mathcal{H}}$ in the same way it is done on the Laplacian in the LLE algorithm. Hence the smallest $d+1$ eigenvectors of $\widehat{\mathcal{H}}$ represent the low-dimensional representation that will minimize the level of curvature of the manifold $\mathcal{M}$. The first d eigenvectors provide the embedding coordinates for $\widehat{\theta}$.

6.2 Novel Manifold Learning Method

In this section we will introduce a novel method for manifold learning. The approximation of the underlying structure is composed by a collection of planes, where each plane approximates a sub–region of the manifold. This approximation is made in a multiscale fashion, which means that you get better and better approximations as you use finer and finer scales. Hence, in essence, this method can reconstruct the manifold providing the desired level of precision together with a strict control on the error at each scale.

6.2.1 The algorithm

Under certain regularity conditions, a d–dimensional manifold embedded in high dimension can be approximated by a collection of d–dimensional planes. The approximation can be performed at different scales with the precision growing as the scales get finer. Based on this idea. we may try to extend the approach discussed in Section 5 for plane arrangement to the manifold case. The algorithm proposed here (Algorithm 5) requires as input parameters the desired approximation error τ and the maximum number of planes per node $K_{\max}$. The algorithm produces finer and finer manifold approximations until the desired precision is achieved. Similarly to how we have done in Algorithm 4, we can subsample the data to have a computational time independent from the sample size n. Let the sample sizes of the center set, of the training set and of the test set be respectively

$$\begin{aligned} n_0 &= c_4\, K_{\max}\, \log(K_{\max}) \\ n_1 &= c_5\, n_0\, d_0\, \log(d_0) \\ n_2 &= c_6\, n_1 \end{aligned} \tag{6.1}$$

where c_4 and c_5 are conceptually similar to the one introduced in Section 5.1, while $c_6 > 0$ represent the ratio between the validation and test sets sizes. At scale $j = 0$ we approximate the manifold with a single plane π_{01}. At each scale $j + 1$ of Algorithm 5 we sample $c_j \cdot (n_0 + n_1 + n_2)$ from the data X, where c_j is the number of planes $\pi_{j1}, \ldots, \pi_{jc_j}$ at scale j, and assign those points to the cluster χ_{jt}, with t in $1, \ldots, c_j$, with a criteria based on the smaller point–plane distance. Now for each cluster $\{\chi_{jt}\}_{t=1}^{c_j}$ we apply a slightly modified version of Algorithm 4 using $(n_0 + n_1)$ points to obtain at most $K_{\max}$ new planes for each cluster for a total of c_{j+1} planes at scale $j + 1$. Figure 6.1 shows how Algorithm 4 is reiterated on a single cluster.

The collection of these planes $\left\{\pi_{(j+1)1}, .., \pi_{(j+1)c_{j+1}}\right\}$ represents the approximation

Algorithm 5 Multiscale Manifold Approximation

Input:

- X: The data matrix
- δ: A tolerance level for Algorithm 3
- τ: The desired level of approximation error
- $K_{\max}$: The maximum number of planes per node (optional)
- d_0: A priori bound for $max\,(d_k)$ (optional)
- c_4: Subsampling constant for n_0 (optional)
- c_5: Subsampling constant for n_1 (optional)
- c_6: Ratio between n_2 and n_1 (optional)

Output:

- A multiscale manifold approximation organized in a tree structure

  At each scale j it is computed:

  - $\pi_{j1}, \ldots \pi_{jc_j}$: collection of planes approximating the manifold
  - $C_{j1}, \ldots, C_{jc_j}$: partition of the manifold
  - $\{e_{jt}\}_{t=1}^{c_j}$: local estimated approximation errors

Steps:

- Let n_0, n_1 and n_2 be as in Equation (6.1) and let $j = 0$ and $c_j = 1$
- Subsample $(n_0 + n_1 + n_2)$ points from X
- Apply Algorithm 4 on $n_0 + n_1$ points forcing a single plane output π_{01}
- Compute the LS error $\{e_{jt}\}_{t=1}^{c_j}$ of the plane using n_2 points as test set
- **while** $\exists\, t \in \{1, ..., c_j\} \,:\, e_{jt} > \tau$

  Subsample $c_j\,(n_0 + n_1 + n_2)$ points from X

  Compute the distances between the subsample and the planes $\{\pi_{jt}\}_{t=1}^{c_j}$

  Assign each point to the cluster $\{\chi_{jt}\}_{t=1}^{c_j}$ with the lower distance

  - **for** $t = 1 : c_j$
    - **if** $|\chi_{jt}| < (n_0 + n_1 + n_2)$
      - Clone the parent: $\pi_{(j+1)t'} = \pi_{(j+1),t}$ with $t' \in T_{jt}$ and $|T_{jt}| = 1$
    - **else**
      - Apply Algorithm 4 to $(n_0 + n_1)$ points in χ_{jt} and let $\left\{\pi_{(j+1)t'}\right\}_{t' \in T_{jt}}$ with $|T_{jt}| \leq K_{\max}$ be the output planes
      - Let $e_{(j+1)t'}$ be the LS error of plane $\pi_{(j+1)t'}$
    - Define $C_{(j+1)t'}$ as the region of the manifold which has $\pi_{(j+1)t'}$ as closest plane $\forall t' \in \cup_{t=1}^{c_j} T_{jt}$
  - Let $c_{j+1} = \sum_{c_j}^{t=1} |T_{jt}|$
  - Let $j = j + 1$
- **return: Output**
- **end algorithm**

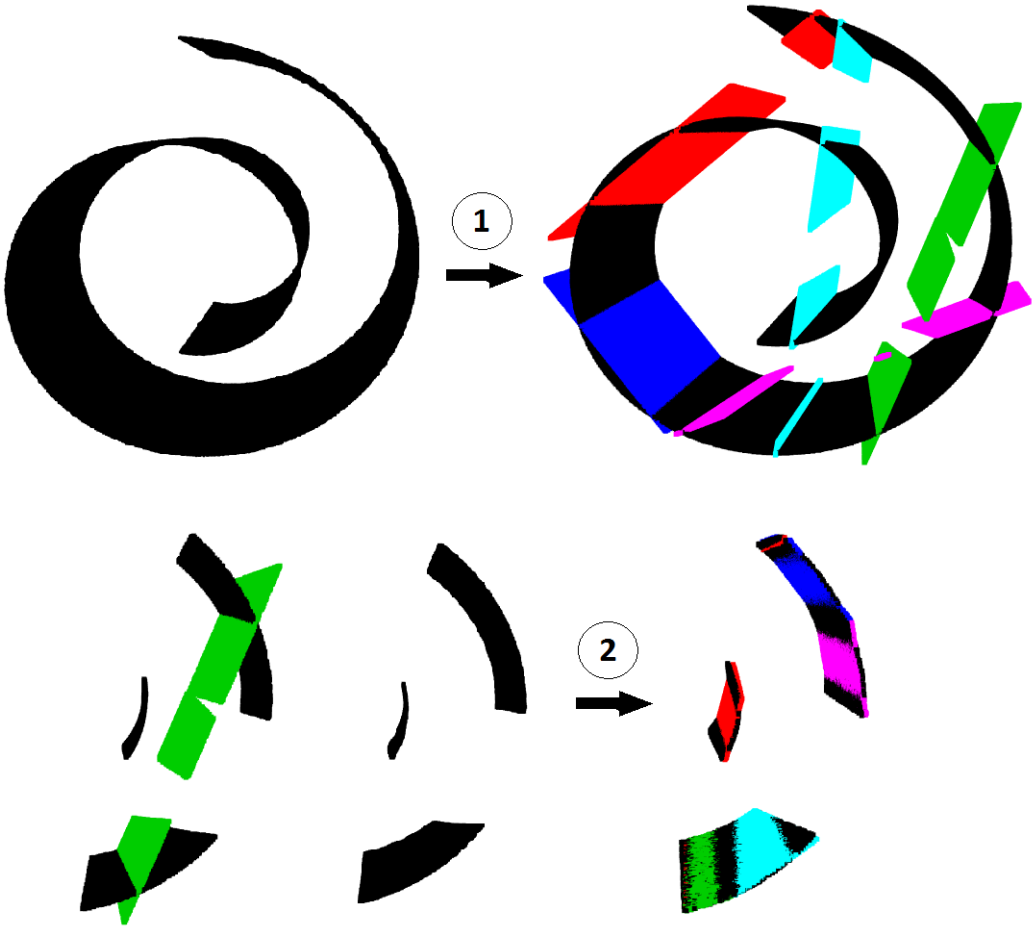

Figure 6.1: An example of the iterative nature of Algorithm 5. In the first step Algorithm 4 works on the whole dataset to obtain five best fitting planes. Then points are grouped according to the closest plane. In the second step Algorithm 4 is applied on each cluster independently: in this picture it is shown what happen in particular to the "green cluster" in order to obtain a finer approximation.

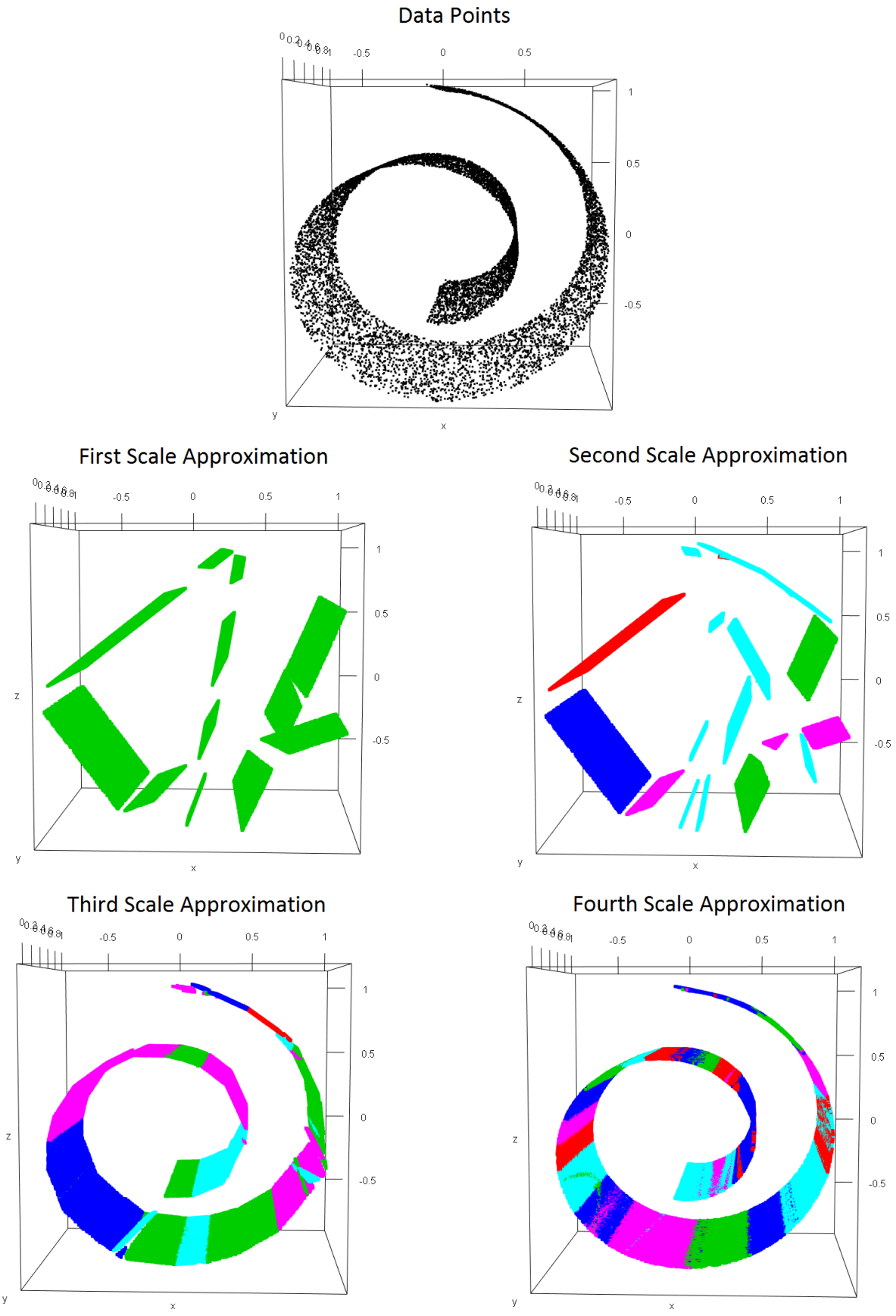

Figure 6.2: An example of multiscale reconstruction of the Swiss Roll manifold.

at scale $j+1$. The approximation error is then computed using other n_2 points of cluster χ_{jt} as test set. If a cluster χ_{jt} does not have $n_0+n_1+n_2$ points we will not perform Algorithm 4 and we will just reproduce the plane π_{jt} used to create the cluster. In this way if we do not have enough point the approximating plane remains the same until eventually enough points will be sub sampled to ensure a finer approximation. The algorithm stops at scale $j^\star$ when all the approximation errors of all the planes will be lower or equal to the desired error τ. Some representative example of this multiscale manifold approximation are shown on Figure 6.3. As mentioned before, this method automatically build a tree structure. Each plane π_{jt} is a linear approximation of a specific region C_{jt} of the manifold which will be approximated in a finer way at the next scale with $q \leq K_{\max}$ planes $\pi_{j1}, \ldots, \pi_{jq}$ which will divide the region C_{jt} in the disjointed subregions $C_{j1}, \ldots, C_{jq}$. Thus every single plane at scale j represents a node of the tree at level j and will be the parent node of up to $K_{\max}$ planes at level $j+1$.

6.2.2 Numerical Results

In this section we show some empirical results on the performances of Algorithm 5. We draw 10^6 points uniformly on a manifold, embed and randomly rotate them in $\mathbb{R}^D$ with $D = 30$. We then corrupt each observation adding a Gaussian noise $\eta_i \sim \frac{\sigma}{\sqrt{D}} N(0, I_D)$. We tested the algorithm in different settings, each repeated 100 times, varying the manifold type – a Swissroll and an S-Manifold – and the noise level σ. In all cases we have chosen a precision parameter τ equal to σ, so that we expect to have in every approximation a MSE $\leq \sigma = \tau$. In Table 6.1 we present for each scenario the MSE, the number of planes needed to obtain a precision of τ and the running time of the algorithm.

Results in Table 6.1 show that the algorithm guarantee an approximation error always lower than τ, we can see this noticing that the MSE values in square bracket, which represent the maximum MSE over the 100 iterations, is always lower than $\sigma = \tau$. As one could expect, the number of planes and the computational time needed decrease as the precision required τ decreases. We notice that the computational time for these experiments are very small relatively to the dimension of the data set which is a $(10^6 \times 30)$ matrix, since, on such a matrix, an application of standard algorithms would be impracticable. To give the reader a benchmark, we ran the Local Linear Embedding algorithm (with given dimension $d = 2$ and neighborhood size of 10 points) on the same machine and software of our experiments, on a Swissroll with only 10^4 points embedded in $\mathbb{R}^5$ with noise level $\sigma = 0.001$. The experiment using the LLE algorithm took more than 111 minutes to reconstruct the

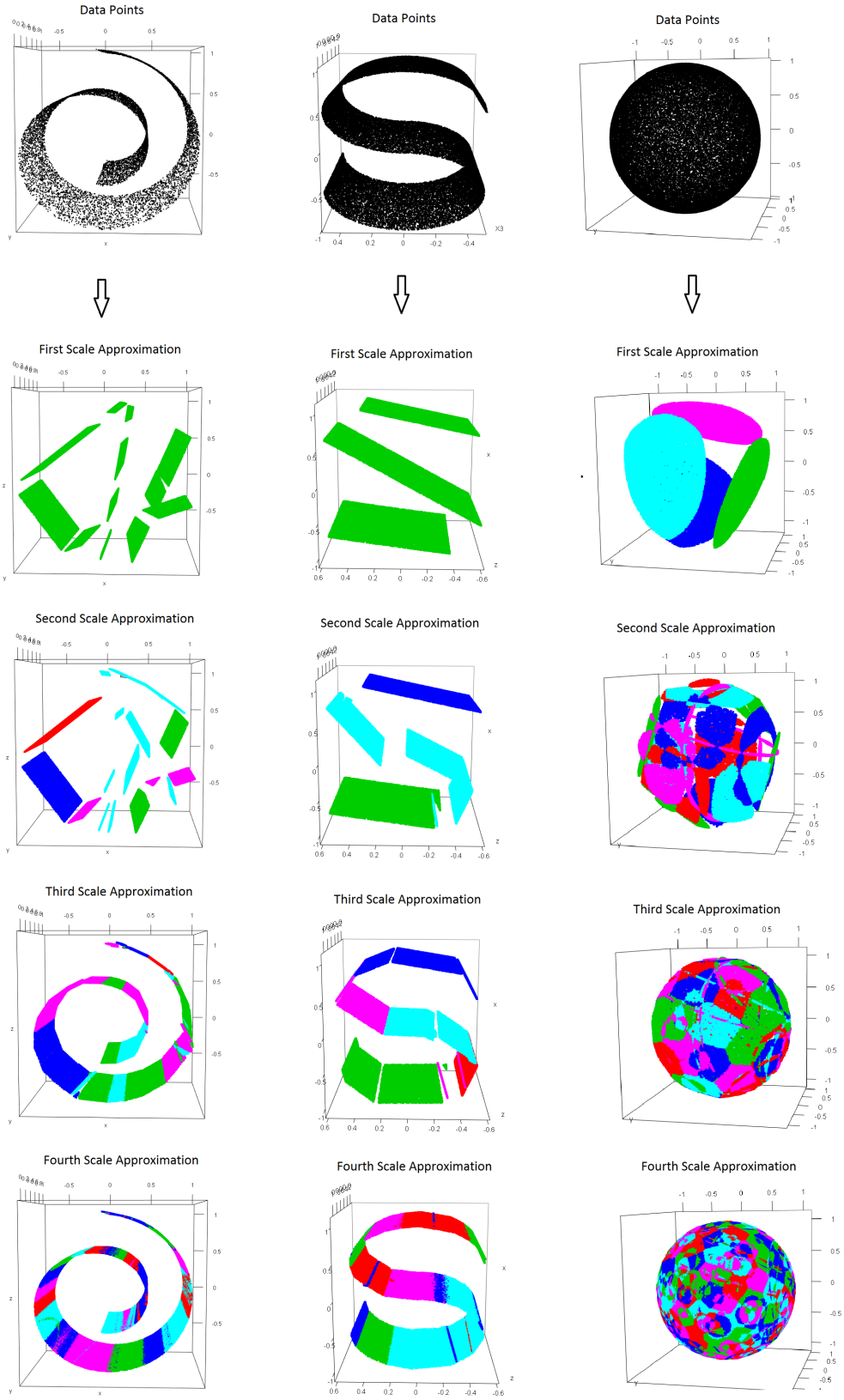

Figure 6.3: An example of the multiscale reconstruction on three different manifolds: a Swissroll, a S–Shaped manifold and a surface of a 3D sphere. All these manifold are embedded in $\mathbb{R}^{30}$ and then corrupted with gaussian noise.

manifold while Algorithm 5 (look at Table 6.1) took an average of 10.09 seconds for a matrix with a number of entries 600 times bigger than the one used on the LLE algorithm. The speed of the algorithm is a direct consequence of the subsampling techniques described in Section 6.2.1 which make the computational times be independent from n and only mildly dependent on D.

$10^3\sigma$		Swissroll	S-Manifold
1	10^3MSE	0.19 (0.13) [0.99]	0.34 (0.31) [0.98]
	Planes	25.23 (9.56)	14.12 (4.83)
	Time	70.22 (24.41)	41.25 (23.05)
5	10^3MSE	1.70 (1.10) [4.94]	3.07 (0.96) [4.99]
	Planes	19.14 (10.96)	6.85 (2.54)
	Time	48.82 (29.17)	5.28 (9.77)
10	10^3MSE	7.30 (2.27) [9.95]	7.63 (0.96) [9.55]
	Planes	9.38 (8.02)	5.31 (1.27)
	Time	10.09 (21.39)	2.23 (3.39)

Table 6.1: Experiments results for Algorithm 5. Experiments are made on two different manifolds, a Swissroll and a S–Manifold, with sample size $n = 10^6$, embedded in $\mathbb{R}^D$ with $D = 30$. All points are corrupted with an additive gaussian noise $\eta_i \sim \frac{\sigma}{\sqrt{D}} N(0, I_D)$. For each manifold we vary the noise level $\sigma = (0.001, 0.05, 0.1)$ and ask the algorithm to produce an approximation with MSE $< \sigma$ setting the algorithm parameter $\tau = \sigma$. The values in the table are averaged over 100 iterations of the experiment, the values in parentheses represent the standard errors while the values in square brackets are the maximum values. Note that some values in the table are scaled by 10^3 to let the results be more readable.

Conclusions

In this dissertation we presented some novel techniques to tackle crucial difficulties that arise when dealing with high dimensional data sets. The aim of all our proposals is to simplify the structure of the data in order to distill meaningful information on their underlying low dimensional structure.

In Part I we focused on techniques to estimate the intrinsic dimension of a dataset. Correctly identifying the minimum number of variables needed to describe a dataset is becoming an unavoidable task in several fields including physics, genomics, statistics, finance and machine learning. The method we propose model the neighbor of each observation as a point process. Looking at the resulting approximate generative model within the realm of composite likelihoods, we combine the local likelihood functions and find a closed form maximum likelihood estimator for the quantity of interest. We compute a correction term for the composite likelihood that should adjust its shape and curvature in order to compensate for the overall model miss–specification. We also present a cross–validation techniques to find the optimal neighborhoods size which is treated basically as a tuning parameter for the intrinsic dimension estimation techniques we introduced. Numerical results on artificial datasets shown that our CV scheme leads to estimates well concentrated around the true value of the intrinsic dimension under different scenarios.

In Part II we presented some methods to find the underlying low dimensional structure of a dataset observed in high dimension. The *fil rouge* of these techniques is the use of a multiscale singular value decomposition (MSVD) approach. The first method introduced is a model selection technique for the plane arrangement problem. In this setting the data points are drawn from several low dimensional intersecting planes embedded in a high dimensional space. We propose an algorithm that reconstructs the plane arrangement by estimating the number of planes needed, their dimension and how they are displaced in the space. In particular we presented a novel method to estimate the dimension of the planes which, according to the numerical simulation on artificial data sets at hand, performs well. The second method is a manifold learning technique which approximates the underlying structure of the

data (i.e. a manifold) with a collection of planes, each of which is in charge of a sub–region of the manifold. This approximation is made in a multiscale fashion, which means that one get better and better approximations simply digging into finer and finer scales. Hence, applying this method, we can reconstruct the manifold providing the desired level of precision and controlling the reconstruction error at each scale. The simulation study we conducted shows that the algorithm is able to guarantee the desired level of precision with a limited number of planes. Both methods rely on a sub–sampling technique which make the computational time independent from the sample size. This property is, of course, extremely important when handling large sample sizes and/or feature spaces. In the simulation study we have shown how our manifold learning algorithm is able to reconstruct the underlying structure of the data in few seconds whereas the use of other more standard methods would be impractical or even impossible given the unbearable computational time.

Bibliography

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591.

Camastra, F. and Vinciarelli, A. (2002). Estimating the intrinsic dimension of data with a fractal-based method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(10):1404–1407.

Chen, G., Atev, S., and Lerman, G. (2009). Kernel spectral curvature clustering (kscc). In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 765–772.

Chen, G. and Lerman, G. (2009). Foundations of a multi-way spectral clustering framework for hybrid linear modeling. *Foundations of Computational Mathematics*, 9(5):517–558.

Chen, G. and Maggioni, M. (2011). Multiscale geometric and spectral analysis of plane arrangements. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2825–2832.

David, G. and Semmes, S. (1993). *Analysis of and on uniformly rectifiable sets*, volume 38. American Mathematical Soc.

de Silva, V. and Tenenbaum, J. B. (2003). Unsupervised learning of curved manifolds. In *Nonlinear estimation and classification*, pages 453–465. Springer.

Fan, M., Gu, N., Qiao, H., and Zhang, B. (2010). Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint arXiv:1002.2050*.

Fukunaga, K. and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *Computers, IEEE Transactions on*, 100(2):176–183.

Geys, H., Molenberghs, G., and Ryan, L. M. (1999). Pseudolikelihood modeling of multivariate outcomes in developmental toxicology. *Journal of the American Statistical Association*, 94(447):734–745.

Grassberger, P. and Procaccia, I. (1983). Characterization of strange attractors. *Physical review letters*, 50(5):346.

Gupta, M. D. and Huang, T. S. (2012). Regularized maximum likelihood for intrinsic dimension estimation. *arXiv preprint arXiv:1203.3483*.

Hong, W., Wright, J., Huang, K., and Ma, Y. (2006). Multiscale hybrid linear models for lossy image representation. *Image Processing, IEEE Transactions on*, 15(12):3655–3671.

James, W. and Stein, C. (1961). Estimation with quadratic loss. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 361–379.

Kent, J. T. (1982). Robust properties of likelihood ratio tests. *Biometrika*, 69(1):19–27.

Levina, E. and Bickel, P. J. (2004). Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, pages 777–784.

Lindsay, B. G. (1988). Composite likelihood methods. *Contemporary mathematics*, 80(1):221–39.

Little, A. V., Jung, Y.-M., and Maggioni, M. (2009a). Multiscale estimation of intrinsic dimensionality of data sets. In *AAAI Fall Symposium: Manifold Learning and Its Applications*.

Little, A. V., Lee, J., Jung, Y.-M., and Maggioni, M. (2009b). Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*, pages 85–88. IEEE.

Little, A. V., Maggioni, M., and Rosasco, L. (2011). Multiscale geometric methods for estimating intrinsic dimension. *Proc. SampTA*.

Little, A. V., Maggioni, M., and Rosasco, L. (2012). Multiscale geometric methods for data sets i: Multiscale svd, noise and curvature.

Ma, Y., Derksen, H., Hong, W., and Wright, J. (2007). Segmentation of multivariate mixed data via lossy data coding and compression. 29(9):1546–1562.

Ma, Y., Yang, A. Y., Derksen, H., and Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM review*, 50(3):413–458.

MacKay, D. and Ghahramani, Z. (2005). Comments on'maximum likelihood estimation of intrinsic dimension'by e. levina and p. bickel (2004). *The Inference Group Website, Cavendish Laboratory, Cambridge University*.

Pauli, F., Racugno, W., and Ventura, L. (2011). Bayesian composite marginal likelihoods. *Statistica Sinica*, 21(1):149.

Pettis, K. W., Bailey, T. A., Jain, A. K., and Dubes, R. C. (1979). An intrinsic dimensionality estimator from near-neighbor information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–37.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Snyder, D. and Miller, M. (1991). *Random Point Processes in Time and Space*. Springer, New York.

Sricharan, K., Raich, R., and Hero, A. (2010). Optimized intrinsic dimension estimator using nearest neighbor graphs. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5418–5421. IEEE.

Sugaya, Y. and Kanatani, K. (2004). Multi-stage unsupervised learning for multi-body motion segmentation. *IEICE Transactions on Information and Systems*, 87(7):1935–1942.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Varin, C. (2008). On composite marginal likelihoods. *AStA Advances in Statistical Analysis*, 92(1):1–28.

Varin, C., Reid, N., and Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica*, 21(1):5–42.

Verbeke, G. (2005). *Models for Discrete Longitudinal Data. Springer Series in Statistics*. Springer.