Dimension Induced Clustering

Aristides Gionis
Basic Research Unit, HIIT
University of Helsinki, Finland

Alexander Hinneburg*
Department of Computer Science
Martin-Luther-University Halle, Germany

Spiros Papadimitriou†
Department of Computer Science
Carnegie Mellon University, USA

Panayiotis Tsaparas
Basic Research Unit, HIIT
University of Helsinki, Finland

ABSTRACT

It is commonly assumed that high-dimensional datasets contain points most of which are located in low-dimensional manifolds. Detection of low-dimensional clusters is an extremely useful task for performing operations such as clustering and classification, however, it is a challenging computational problem. In this paper we study the problem of finding subsets of points with low intrinsic dimensionality. Our main contribution is to extend the definition of fractal correlation dimension, which measures average volume growth rate, in order to estimate the intrinsic dimensionality of the data in local neighborhoods. We provide a careful analysis of several key examples in order to demonstrate the properties of our measure. Based on our proposed measure, we introduce a novel approach to discover clusters with low dimensionality. The resulting algorithms extend previous density based measures, which have been successfully used for clustering. We demonstrate the effectiveness of our algorithms for discovering low-dimensional m-flats embedded in high dimensional spaces, and for detecting low-rank submatrices.

1. INTRODUCTION

Real datasets exhibit patterns and regularities. As a main consequence, points typically lie on low-dimensional manifolds, rather than being evenly spread out. Detecting subsets of points with low intrinsic dimensionality is useful in tasks such as indexing and classification. It has recently been proved [18] that the well-known "curse of dimensionality" translates essentially to a "curse of intrinsic dimensionality," in terms of finding efficient approximations to nearest-neighbor queries. Furthermore, separating points based on some notion of "local dimensionality" is helpful in identifying subsets of points that are qualitatively different. For example assuming a geographical setting, locations along a river belong to a 1-D manifold, whereas locations on a lake would belong to a 2-D manifold (for example Figure 1). Similarly, road intersections along a highway are on a 1-D manifold, while intersections within a city belong to a 2-D manifold. Thus, discovering low-dimensional manifolds is also useful in its own right.

However, we are faced with three main challenges. The first question that naturally arises is what "dimension" exactly means. Data patterns may be fairly complicated. Assuming that points follow linear trends and always lie on hyper-planes is fairly restrictive; in fact, they typically follow complex shapes, with limited extents. For example, in a more abstract setting the lake and river may lie on the same 2-D plane, but they still differ in dimension. Second, to complicate matters even further, the observations may not even belong to a vector space. Yet, we should be still able to define the dimension of a subspace. Finally, in practical applications, the dimension of the embedding space is large, in the order of thousands. Any method of practical interest should be able to deal with spaces of arbitrarily high dimension and still successfully find low-dimensional subsets embedded in the original space. It is thus desirable to characterize manifolds of complex shape embedded in any space of high dimension, and devise algorithms for identifying them. As we shall see, it is possible to intuitively define a topological notion of dimension, which does not depend on the notion of a linear subspace. Furthermore, our algorithms for identifying low-dimensional manifolds are not sensitive to the dimension of the original space and, thus, do not suffer from the "curse of dimensionality".

In order to argue about and detect the existence of a low-dimensional manifold in our data, there must exist a sufficiently large number of points that are densely packed on this manifold. Therefore, it seems reasonable, that using density based methods it would be possible to detect such subspaces. However, we argue that density alone is not enough for this task. For the sake of example consider city locations interspersed among highway intersections. The cities, lying on a 2-D surface, form the first cluster. The road intersections form the second cluster, as a complex network of denser 1-D lines which occupies the *same* space as the city manifold cluster. Density-based clustering approaches have a limited ability to detect clusters-within-clusters. In this simple example, they would typically produce either a large number of separate city clusters (one for each group

*This work was done while the author was working at Basic Research Unit, HIIT, University of Helsinki, Finland.
†Part of this work was done while the author was visiting Basic Research Unit, HIIT, University of Helsinki, Finland.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

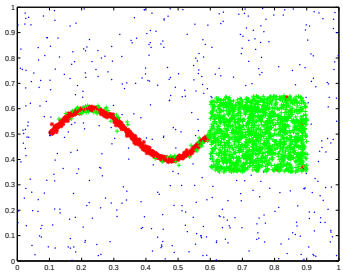


Figure 1: A dataset that contains two subsets of different intrinsic dimensionality

of cities enclosed by roads), or one single cluster containing both intersections and cities, depending on the density thresholds.

Consider also the example in Figure 1. In this case we have three qualitatively different types of points. The set of points that lie on the curved 1-D line, the set of points that lie on the 2-D cloud, and the noise points that are scattered in the 2-D plane. If the line and the square have the same density, using a density based method is not possible to detect all three of these subsets. Any density threshold will just separate the noise points from the rest. Note also that dimensionality by itself would not be able to separate the square from the noise points, since the noise points are also 2-dimensional. The synergy of density and intrinsic dimensionality gives a clear separation of the three distinct datasets, as it is shown in the figure.

In this paper we propose the idea of creating a local-growth model for each point. This growth model depends, in principle, only on pairwise point distances and captures how each point "views" its local neighborhood. Using this model we can characterize each point x_i with two variables (d_i, c_i), where d_i is the *local dimensionality* of the point x_i, and c_i is the *local density*. Intuitively, d_i depends on the growth rate of the number of points in the neighborhood of x_i, while c_i depends on the density of points in the neighborhood of x_i.

Both variables are estimated from *local growth curves*. Local growth curves can be computed directly from the data. Our algorithms require only a limited number of nearest-neighbor (NN) queries. These are well-studied and several efficient algorithms exist to answer them. For each point x_i, the local growthcurve of x_i is computed, and a line is fitted on a subset of the points of the curve that corresponds to a local neighborhood of x_i. Then, the local dimensionality d_i is defined as the slope of the fitted line, while the local density c_i is defined as the value of the fitted line for a specific radius r^*. We choose r^* so as to maximize the information captured by the set of feature pairs (d_i, c_i), in the sense of minimizing the correlation between d_i and c_i.

Using the local density and local dimensionality, each point x_i is represented by the feature pair (d_i, c_i). Therefore, we map our dataset in a two dimensional space. This has the following advantages. First, we can easily cluster the dataset using an off-the-shelf, two-dimensional clustering algorithm, like EM. Second, in a user interactive system, the number of clusters and the correct partition can be identified through visual inspection.

Our main contributions are the following:

- Drawing upon ideas from fractals, we propose a general way to characterize the *local* dimensionality of points. Our definitions are topological and independent of the notion of a linear subspace. Our methods can be applied to datasets of arbitrary dimensionality and our algorithms are independent of the number of dimensions in the original dataset.
- Our method maps the dataset into a 2-dimensional space. We show how to chose the feature pairs (d_i, c_i), so as to maximize the information they retain about the dataset, and enhance the visual representation of the dataset.
- We demonstrate how local dimensionality and local density can be used to detect low dimensional m-flats and low-rank sub-matrices. Our algorithms can successfully detect low-dimensional manifolds embedded in high-dimensional spaces, even when they are spatially overlapping

Additionally, our method does not assume that the points lie in a vector space and can be also applied to metric data, when dimensionality is not directly obtainable from the data representation itself.

The rest of the paper is organized as follows. Section 2 discusses briefly the related work. Section 3 introduces the key concepts and definitions. Section 4 explains their properties and elaborates on effectively selecting feature pairs (d_i, c_i). Section 5 presents our algorithms and section 6 applies them to the problems of detecting low-dimensional m-flats and low-rank sub-matrices. Finally, we conclude in section 7.

2. RELATED WORK

In this section we briefly discuss related work, broadly divided in two categories: methods that use some notion of density, and methods based on intrinsic dimensionality.

Density-based clustering: Similar to our method, density-based clustering approaches also rely on local density information in order to partition the dataset.

Hierarchical single linkage is a well-known method to find clusters with respect to density. To overcome problems in cases, when clusters are connected by small chains, popular variants like DBSCAN [11] and OPTICS [5] use a modified linkage hierarchy, where points within a cluster have to be reachable via core points (points having a certain minimum number of neighbors). DBSCAN computes a clustering corresponding to a cut in the linkage hierarchy, while OPTICS finds an ordering of the points from which a lower part of the linkage hierarchy can be deduced. Both algorithms fall short in case of clusters within clusters and the true hierarchy contains nodes with degree one. However, as our approach does not rely on spatial separation of the clusters, but focuses on detecting subsets with low intrinsic dimensionality, it can also deal with those cases.

Another density-based clustering method is DenClue [16], which employs kernel density estimation and uses density thresholds to define the clusters to be found. CLIQUE [4] is a density-based method that can also detect subspaces such that high-density clusters exist in them. However, it is grid-based and thus assumes that points lie in vector space. Furthermore, CLIQUE considers only hyper-rectangular clusters and projections parallel to the axes.

Projective clustering: There are some algorithms which can find clusters which are dense in a projection of the original data space. Proclus [2] and DOC [21] search the space of axes-parallel projections to find good clusterings of the data. More advanced techniques like Orclus [3] and projective k-means [1] analyze eigenvalues of subsets of the data and can find arbitrary linear projections, in which points are clustered.

Fractals-related work: Concepts of intrinsic dimensionality from fractals have been successfully used in the database field for numerous problems, such as nearest-neighbor queries [19] and spatial query selectivity estimation [8, 12]. Recent results [18] discuss doubling dimension as measure for intrinsic dimensionality. The proposed algorithm works efficiently, when the intrinsic dimensionality is bounded.

Barbará et al. [7] propose a clustering approach that uses the fractal dimension (box-counting). It computes the fractal dimension of each individual cluster X and of $X \setminus x_i$ and puts x_i into the cluster for which the change is minimal. This requires some initial seed clusters, which may be difficult to guess.

Finally, LOCI [20] is an outlier detection method based on the local distribution of pairwise distances at multiple scales. Although the key concepts are related, LOCI focuses on an entirely different application and does not use the concept of intrinsic dimensionality in any way.

3. OVERVIEW OF THE APPROACH

In this section we give an overview of our method. As we discussed before, the method draws upon and extends previous density-based algorithms, as well as concepts of intrinsic dimensionality. The two key measures it uses are *local density* and *local dimensionality*. Both are obtained by fitting a line on a subset of points of the *local growth curve*.

First, we review basic facts about the notion of intrinsic dimensionality. Then, we describe local growth curves, and we explain how local density and local dimensionality are computed, and how they are used for clustering the dataset.

3.1 Background on intrinsic dimensionality

As an underlying basis of our method, we use the notion of *correlation dimension*, which is a measure of the *intrinsic dimensionality* of a dataset. In the following discussion, we assume that the dataset X is a subset of $\mathbb{R}^m$, and for definition purposes we assume that the number of points n in X approaches the infinity. Let $d : X \times X \to \mathbb{R}$ be a distance function between pairs of points of X, and let $C(r)$ be the average fraction of pairs of points within distance r, that is,

$$C(r) = \lim_{n\to\infty} \frac{1}{n^2} \sum_{x \in X} |B(x, r)|,$$

where $B(x, r) = \{y \mid y \in X, d(x, y) \leq r\}$ is the subset of points contained in a ball of radius r, centered at point x. The correlation dimension is then defined as

$$d_{corr} = \lim_{r,r'\to 0} \frac{\log[C(r)/C(r')]}{\log[r/r']}. \quad (1)$$

We assume that all the limits exist. Alternative definitions of intrinsic dimensionality can be found in the literature, such as *capacity* or *box counting dimension* and *information dimension*. Intrinsic dimensionality measures are sometimes also collectively referred to as *fractal dimension*. The interested reader can find a comprehensive development of the topic in a standard textbook, e.g., Rasband [22].

In practice, we deal with finite sets, so the definition of correlation dimension in Equation (1) is not applicable. In this case, we define the function $C(r)$ as

$$C(r) = \frac{1}{n^2} \sum_{x \in X} |B(x, r)|, \quad (2)$$

and estimate the correlation dimension by the *slope* of the function $C(r)$ in the log-log scale. The reason is that, since $\frac{\log[C(r)/C(r')]}{\log[r/r']} = \frac{\log[C(r)]-\log[C(r')]}{\log r - \log r'}$, the correlation dimension expresses the increase rate of $\log[C(r)]$ between $\log r$ and $\log r'$.

The intuition is shown in Figure 2: For points that are arranged on a line, as shown in Figure 2(a), one expects to find twice the number of points when doubling the radius. On the other hand, for points that are scattered on the 2-D plane, as shown in Figure 2(c), when doubling the radius, we expect the number of points to increase quadratically. The growth rates of the number of points in Figures 2(a) and 2(c) can then be estimated from the slope of the $C(r)$ curve in log-log scale, as shown in Figures 2(b) and 2(d), respectively. These are close to one and two, respectively. To enable visual comparison, the scales of the Figures 2(b) and Figures 2(d) are the same.

3.2 Local Correlation Dimension

The function $C(r)$ as defined in Equation (2) computes the average fraction of neighbours of a point within distance r, where the average is taken over *all* points of the dataset. However, due to averaging, if the dataset is non-homogeneous, the estimated correlation dimension will not reflect the "true" dimensionality of the data. Figure 3 illustrates this point. Figure 3(a) shows a dataset with two distinct subsets of points: in the first subset the points lie on a 1-D curve, while the second subset consists of a cloud of 2-D points. As a consequence of taking averages, the intrinsic dimension of the whole dataset is somewhere between one and two. Figure 3(b) shows the line fitted to the $C(r)$ curve and, for comparison, lines with slopes one and two.

Therefore, in the case that a dataset consists of subsets with different intrinsic dimensionality, the correlation dimension of a dataset does not correctly characterize the dimensionality of the dataset. To overcome this problem we extend the definition of correlation dimension for each point in the dataset.

DEFINITION 1 (LOCAL-GROWTH CURVE). *For each point x, we define the* local-growth curve, *to be the function of r, $G_x : \mathbb{R} \to \mathbb{N}$ that computes the fraction of neighbors of x in a ball of radius r,*

$$G_x(r) = \lim_{n\to\infty} \frac{1}{n} |B(x, r)|.$$

The local growth curve G_x describes the density of the local neighborhood of x for all distances r. In addition, the G_x curve contains information about the growth rate of the number of neighbors of x. We can now define the *local-correlation dimension* (or local dimension) of point x.

DEFINITION 2 (LOCAL-CORRELATION DIMENSION). *We*

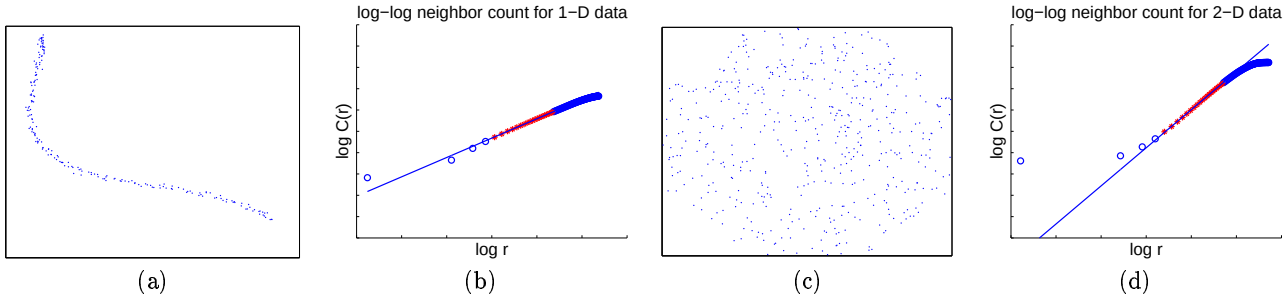


Figure 2: Intuition behind the intrinsic dimensionality (correlation dimension).

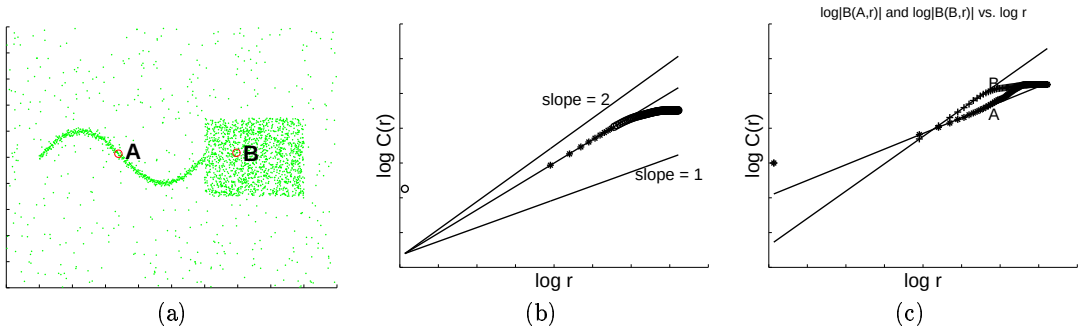


Figure 3: A dataset that contains two subsets of different intrinsic dimensionality

define the local-correlation dimension d_x *of point* x*, as*

$$d_x = \lim_{r,r' \to 0} \frac{\log[G_x(r)/G_x(r')]}{\log[r/r']}. \quad (3)$$

As in the case of correlation dimension, when dealing with finite sets, we define the local growth curve to be $G_x(r) = \frac{1}{n}|B(x,r)|$, and we compute the local-correlation dimension d_x of a point x by the slope of G_x curve in log-log scale. Notice that for finite sets the G_x curve is step-wise – its value chances only when the radius grows to include the next neighbor of a point. As a result, the local-growth curve can be represented without loss of information by specifying its value on a finite set of radii $\mathcal{D}_x \subset \mathbb{R}$, which we call the *domain* of G_x.

3.3 Local representation

We now describe how to use the local growth curves and the local-correlation dimension in order to represent the dataset.

Let $X = \{x_1, \ldots, x_n\}$ be a dataset of n points in some metric space. For each point x_i we define its domain $\mathcal{D}_{x_i}$, and we compute the local growth curve G_{x_i}. We then take the G_{x_i} curve in log-log scale, and we find the line that fits it best, in a least squares sense. We use L_{x_i} to denote this line, and we call it the *linear growth model* for point x_i.

DEFINITION 3 (LINEAR GROWTH MODEL). *We refer to the line*

$$L_{x_i}(\log r) = d_i \log r + b_i$$

as the Linear Growth Model *for the point* x_i.

The slope d_i of the line L_{x_i} is an estimate of the local-correlation dimension of point x_i. The value b_i is the coefficient computed by the line fitting. Using the linear growth model, we can now represent the point x_i using just two numbers. The first number is the value d_i, the *local dimension* of the point x_i. The second number is denoted by $c_i = L_{x_i}(\log r^*)$, and it corresponds to the density of the dataset in a ball of radius r^*, centered on x_i *as it is estimated* by the linear growth model L_{x_i} for point x_i. For example, for $r^* = 1$, $c_i = b_i$. We defer the discussion about the choice of value for r^* to Section 4, Lemma 1. We call c_i the *local density* of the point x_i. We write $l(x_i) = (d_i, c_i)$ to denote the representation of x_i by these two parameters.

DEFINITION 4 (LOCAL REPRESENTATION). *The mapping* $l(x_i) = (d_i, c_i)$ *is called the* local representation *of* x_i*, where* $c_i = L_{x_i}(\log r^*)$.

To illustrate the intuition behind local representation, consider two specific points A and B that come from the two different subsets in the example of Figure 3. Panel 3(c) shows $G_x(r)$ for $x = A$ and $x = B$ together with the fitted lines. In our example, the local dimensionality of point A is 1.20 and that of point B is 1.98. Therefore, we are able to distinguish the subsets with different intrinsic dimensionality using the local dimensionality d_i.

As we will explain in the next section, it is meaningful to ignore the parts of the local growth curve that correspond to very small and very large radii. The reason is that, for small r, the value of $G_x(r)$ is sensitive to local noise effects. Thus, ignoring small radii improves the robustness of the estimated dimension. On the other hand, for large r too many points contribute to the value of $G_x(r)$. Therefore, G_x does not capture local-neighborhood structure around x any more; most curves look identical. For these reasons we restrict the domain $\mathcal{D}_x$ of $G_x(r)$ to a smaller subset $\mathcal{F}_x \subseteq \mathcal{D}_x$, which we call the *fitting set* of $G_x(r)$, and it is precisely the range over

which we fit the linear-growth model L_x. The details of how the fitting set is determined are discussed in Section 4.2.

3.4 Overall clustering

The final step of our method is to detect clusters of points that form low-dimensional manifolds in the ambient space of the dataset. Taking advantage of the simplicity of the local representation $l(x_i) = (d_i, c_i)$, we can perform this step using a standard clustering algorithm. Assuming that the local representation maintains well the information about the local density and the local dimensionality of the points, the clustering process is relatively easy since it is an operation on two-dimensional data. For our experiments we used the standard EM algorithm with full covariance matrices. Furthermore, the 2-D representation offers an informative visualization of the dataset. In a user-interactive system it is usually easy to determine the underlying clusters in the dataset.

Finally, we note that the output of our algorithm can be further processed to discover dimensions of interest. First, in case that the algorithm places two manifolds of same dimension and density into one cluster, but the two manifolds do not intersect, then they can be separated by the single-linkage clustering. Second, in case of axis aligned subspaces, we can easily discover the attributes of interest by measuring the variance along each dimension. Finally, in case that we are interested in linear subspaces, running PCA will reveal the directions of interest. All of these three tasks become significantly easier once the appropriate subset of points has been identified by our method.

4. ANALYSIS OF OUR METHOD

In this section we discuss the properties of our definitions, and their implications in the overall approach. As our guide in this discussion we consider the simple cases of points lying on a 2-D grid and on a 1-D line.

4.1 Discussion and Examples

The definition of correlation dimension in Section 3 assumes that the size of the set X is infinite. For an infinite real line and an infinite real plane the dimensions are precisely 1 and 2, respectively. In practice, however, we deal with finite sets with finite extent, so we can only compute an estimate of the actual dimension. We will now study the effect of finiteness on the local representation of the points by investigating two simple cases. The first dataset L consists of n one-dimensional points equally spaced on a line. The second dataset G consists of n two-dimensional points arranged on a grid. Ideally, the intrinsic dimensionality of the line should be one, and the dimensionality of the grid should be two. However, due to the finite size of the datasets, the estimated dimensionalities are different.

Consider the points in the set L and assume that the point x_i is located at position i of the real line. Consider one of the endpoints of the line, e.g., the leftmost point x_1 of the line. The local growth curve of x_1 is $G_{x_1}(r) = \frac{r}{n}$ (when computing $|B(x_1, r)|$ we do not count the point itself). For the point x_m in the middle of the line $m = n/2$, the local growth curve is $G_{x_m}(r) = \frac{2r}{n}$. In both of these cases, the local dimensionality of points x_1 and x_m is one, as expected. However, consider the point x_p that lies in position $p = n/4$. For radii $r = 1..\frac{n}{4}$, $G_{x_p}(r) = \frac{2r}{n}$, while for radii $r = \frac{n}{4}...\frac{3n}{4}$, $G_{x_p}(r) = \frac{r}{n}$. Due to this change in the local growth curve, when fitting a line, the local dimensionality of the point x_p is underestimated. For example, for a line with 500 points, the local dimensionality is estimated to be around 0.87. The d_i values for all points are shown in Figure 4.

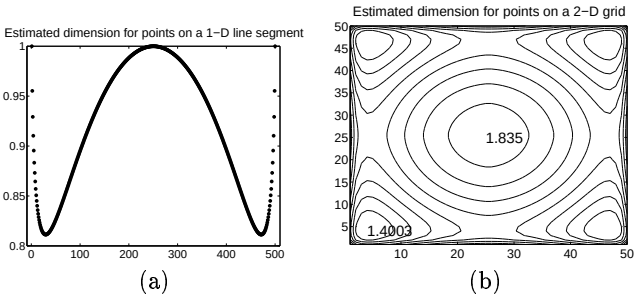


Figure 4: Boundary effects on the estimation of the correlation dimension.

Determining the correct slope is even harder for the two-dimensional grid. Consider the point x_m in the middle of the grid. For simplicity we will assume that the distance between points is measured using the L_∞ norm. It is not hard to see that the local growth curve for this point is $G_{x_m}(r) = \frac{1}{n}((2r+1)^2 - 1) = \frac{1}{n}(4r^2 + 4r)$ (again, the point x_m is not counted in the computation). Due to the additive term $4r$ we need to have $r \to \infty$ in order for the local dimension d_m of x_m to tend to 2. In practice, this results in underestimating the dimension of the point. For a 50×50 grid the local dimension of the middle points is estimated to be close to 1.8. Furthermore, simple computations show that for a point x_s on the side of the grid, $G_{x_s}(r) = \frac{1}{n}(2r^2 + 3r)$, while for a point x_c on the corner of the grid, $G_{x_c}(r) = \frac{1}{n}(r^2 + 2r)$. Again, the local growth curve on the boundary of the grid is different from that inside the grid. Therefore, when the curve hits the boundary there is a change in the local growth curve, which results in further underestimation of the local dimension. An example with a 50×50 grid is shown in Figure 4. The contour lines show how the local dimension changes for different points of the grid. The maximum and minimum values are shown on the plot.

We next consider the case of a dataset consisting of a line embedded in a grid. We assume that the line consists of grid points which are replicated μ times. The value μ is the density of the line. For simplicity, assume that the line lies in the middle of the grid and it is parallel to one axis of the grid. Furthermore, in order to avoid dealing with boundary points, assume that the grid extends to infinity in all directions. For a point on the line x_ℓ it is not hard to show that the local growth curve is $G_{x_\ell}(r) = \frac{1}{n}((2r+1)^2 + \mu(2r+1) - 1)$. When the value μ is large enough compared to r the growth of $G_{x_\ell}(r)$ is dominated by the linear term. Of course as $r \to \infty$, the quadratic part becomes dominant. For a grid point, the growth is the same as before as long as the ball around the point has not reached the line. When the line is reached, the local growth curve becomes the same as for a line point (this is also due to the fact that we consider the L_∞ distance, and the line is parallel to the axis).

In the next subsection we will see how to address the issues raised by the previous examples. The idea is to compute the local dimensionality d_x of each point by fitting a line not to the entire local growth curve G_x, but only on the fitting set

$\mathcal{F}_x$. We discuss how to determine $\mathcal{F}_x$ next.

4.2 Determining the fitting set

An important issue in the definition of local growth curves is the domain of radii over which they are defined. An immediate idea is to define the curves over the interval ranging from the minimal pairwise distance up to the diameter of the dataset. Let $\mathcal{R}$ denote this interval. This is the maximal interval over which the local growth curves can be defined. However, this approach is extreme, since for most points, the low part of the curve will be zero (balls with small radius contain no points), while the upper part of the curve will be one (balls with large radius contain the whole dataset). This will result in poor estimates for the local dimension of these points.

One approach for dealing with this problem is to restrict the definition of the local growth curves over an interval $[r_{\min}, r_{\max}] \subset \mathcal{R}$, which one might believe that captures the useful information of the local growth curve. However, this approach is also problematic when the density of the dataset differs in different regions of the space. Furthermore, for points that lie in large dimensional spaces, the minimum and maximum distances converge, so it is challenging to find a meaningful interval.

To address these issues, we choose to define a different domain $\mathcal{D}_x$ for each point in the dataset. This domain is defined by growing a ball around x such that at each step we extend the ball to include (at least) one more neighbor. In other words, the domain $\mathcal{D}_x$ is precisely the set of radii $\{r_1, r_2, ..., r_n\}$, where r_k is the distance of x to its k-th nearest neighbor.

Following the discussion in Section 4.1, it becomes clear that it is beneficial to restrict the domain $\mathcal{D}_x$ by considering the distances only up to some k_{max}-th nearest neighbor, instead of all possible neighbors. This has the following advantages. First, it captures best the idea of locality upon which our approach is based. As it was demonstrated in the case of the line embedded in the grid, this can help discriminate between points that lie on different manifolds. Second, it helps in avoiding strong boundary effects, since less points hit the boundaries, and thus we can better estimate their "real" dimension. This is shown in Figures 5 *(a)*, *(b)*, and *(c)*, where by restricting the interval from above, we obtain a better estimation of the dimension for more points of the line and the grid.

We further restrict the interval R_x from below, by considering only the neighbors that are no closer than the $k_{\min}$-th nearest neighbor. As discussed in Section 4.1, in the case of the grid this helps obtain a better estimate of the dimension. This becomes obvious when comparing the the figures *(b)* and *(c)* in Figure 5. Figure *(c)* is obtained by restricting the interval R_x from below, where we obtain an estimate of the dimension closer to 2.

Furthermore, for small values of k (i.e., for the very first nearest neighbors) the radius r_k might be affected by small local variations of the density of the points. Such density variations might include isolated points or unusually dense areas. Our point is illustrated in Figure 5*(d)*. We generated 100 points uniformly at random in a d-dimensional hypercube, for $d = 2$, 5, and 10. By repeating the process of random point generation 1000 times, we estimate the expected distance and the variance of the k-th nearest neighbor from a randomly selected point as a function of k. Figure 5*(d)* shows that the variance of the distances of the very first nearest neighbors is large. Therefore, by ignoring the k-th nearest neighbors for $k < k_{\min}$ we obtain a more robust estimation of the local dimension. We are now ready to summarize our observations with the following simple definition.

DEFINITION 5 (FITTING SET). *The set of radii* $\mathcal{F}_x = \{r_k \in \mathcal{D}_x \mid r^{(x)}_{k_{\min}} \leq r_k \leq r^{(x)}_{k_{\max}}\}$, *where* $r^{(x)}_{k_{\min}}$ *and* $r^{(x)}_{k_{\max}}$ *are the distances of the* $k_{\min}$*-th and* $k_{\max}$*-th nearest neighbors of* x *is called the* fitting set.

In our experiments, we have found that the algorithm is not particularly sensitive in the choice of $k_{\min}$ and $k_{\max}$. For example, the values $k_{\min} = 0.01 \cdot n$ and $k_{\max} = 0.1 \cdot n$ give good quality of results in a wide variety of datasets.

4.3 Estimating local density

In this paragraph we derive an estimation for the radius r^*, which is used for computing the local density $c_i = L_{x_i}(\log r^*)$ for each x_i. For simplicity of notation we rewrite Equation (3) as $Y = d_i X + b_i$. We also write $X^* = \log r^*$ and $Y^* = d_i X^* + b_i = c_i$. Notice that by fitting a line to the curve G_{x_i} we obtain the parameters d_i and b_i. The goal is to compute the "best" choice of parameter $\log r^*$ that achieves the local representation $l(x_i) = (d_i, c_i)$ for each x_i.

The main observation is that by setting $\log r^* = +\infty$ in Equation (3), the resulting values c_i of local densities are perfectly positively correlated with the values d_i of local dimensions. The reason is that for $\log r^* = +\infty$ the ordering of c_i's is completely determined by the ordering of d_i's. Similarly, for $\log r^* \to -\infty$, the c_i's are perfectly negatively correlated with d_i's. Since our goal is to use the pair (d_i, c_i) that captures as much information for each x_i as possible, we would like to choose r^* so that the parameters d_i and c_i are uncorrelated. Based on this idea we can estimate the optimal radius r^* for the local representation $l(x_i)$. Note that it makes a considerable difference for the visualization as well as for automated clustering algorithms, whether the two-dimensional data (d_i, c_i) are correlated or not.

LEMMA 1. *The value of* r^* *for which* d_i *and* c_i *are uncorrelated is given by*

$$\log r^* = -\frac{\sum (d_i - \bar{d})(b_i - \bar{b})}{\sum (d_i - \bar{d})^2}$$

PROOF. The correlation between the variables d_i and c_i can be estimated by the coefficient

$$r_{dc} = \frac{\sum_i (d_i - \bar{d})(c_i - \bar{c})}{\sqrt{\sum_i (d_i - \bar{d})^2 \sum_i (c_i - \bar{c})^2}},$$

where $\bar{d} = E[d_i]$ and $\bar{c} = E[c_i]$ are the expectations of d_i's and c_i's, respectively. To make the correlation zero we need to choose r^* so that the numerator of r_{dc} is equal to zero. Let $\bar{b} = E[b_i]$ be the expectation of b_i's. Since $c_i = d_i X^* + d_i$, by linearity of expectation we get $\bar{c} = E[c_i] = E[b_i + X^* d_i] = \bar{b} + X^* \bar{d}$. The numerator of the correlation coefficient can

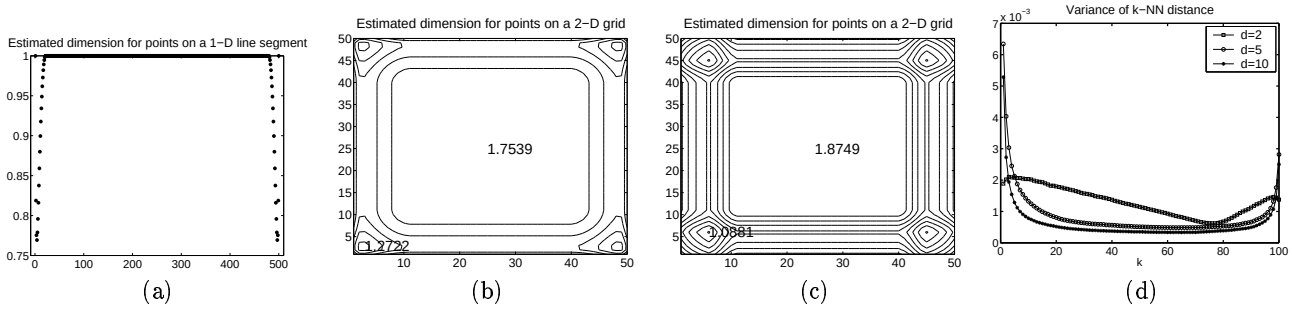


Figure 5: Restricting the fitting interval

now be written as

$$\begin{aligned}
&\sum(d_i - \bar{d})(c_i - \bar{c}) \\
&\quad = \sum(d_i - \bar{d})(b_i + X^* d_i - \bar{b} - X^* \bar{d}) \\
&\quad = \sum(d_i - \bar{d})(X^*(d_i - \bar{d}) + (b_i - \bar{b})) \\
&\quad = X^* \sum(d_i - \bar{d})^2 + \sum(d_i - \bar{d})(b_i - \bar{b}).
\end{aligned}$$

Setting $r_{dc} = 0$ gives the optimal value of $\log r^*$.

5. THE ALGORITHM

We now present the Dimension Induced Clustering (DIC) algorithm. The algorithm works on the representation of the dataset defined in Section 3. The objective of the algorithm is to partition points so that points in the same cluster lie on dense manifolds of the same dimension.

5.1 The DIC algorithm

The outline of the DIC algorithm is shown in Algorithm 1. The input to the algorithm is a set X of n elements, that we want to cluster in b clusters. In first step, the algorithm computes for each element x_i, the distance of x_i to its k-th nearest neighbor for all $k = k_{\min}, \ldots, k_{\max}$. The distances of the nearest neighbors of x_i specify completely the local growth curve G_{x_i}. By fitting the linear growth model L_{x_i} on G_{x_i} and by estimating the local density, as in section 4.3, we compute the local representation $l(x_i) = (d_i, c_i)$ for each x_i. Thus, we map the set X into a two dimensional set X_{LR} that contains the local representation of all points. The task now becomes to cluster the two-dimensional points in X_{LR}. Clustering in two dimensions is conceptually much simpler than clustering in high-dimensional spaces. The correct clustering can often be determined even by simple visual inspection. In the automated case applying an EM (Expectation Maximization) algorithm [15] for fitting b Gaussian distributions on the data works well in most cases. If the set X consists of b sufficiently dense subsets that lie on manifolds of different dimension, which are sufficiently separated, the algorithm will be able to separate these subsets.

5.2 Efficiency of the DIC algorithm

The complexity of the DIC algorithm is dominated by the complexity of computing for every point x_i the distance to the k_{min} to $k_{\max}$ neighbors of the point x_i. The simple solution to this problem is to compute the distances between all points in the set X, and for each point x_i sort the points with respect to their distance from point x_i, and retrieve the necessary information. The time for computing all pairwise distances is $O(n^2)$.

Algorithm 1 The DIC algorithm

Input: Dataset X of n points, number of clusters b
Ouput: Clustering of X into b clusters
1: **for all** $i \in \{1, \ldots, n\}$ **do**
2: Compute k-th NN of x_i, for $k = k_{\min} \ldots k_{\max}$
3: Compute the local representation (d_i, c_i) of x_i.
4: **end for**
5: $X_{LR} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$
6: Cluster the set X_{LR} into b clusters.

A different approach is to construct an index for the elements in X that supports fast execution of k-nearest neighbor queries. In case that X consists of vector data, spatial index structures can be used for the efficient calculation such as [6,9,17]. In case of metric data the OMNI framework [13], or data structures like the M-tree [10] can be used. Since the computation of the local representation is inherently approximate, the use of approximative methods for k-nearest neighbor queries such as locality-sensitive hashing [14], is also possible.

Investigating the construction of the appropriate nearest neighbor index is beyond the scope of this paper. We assume that such an index exists, and we use it as a black box for obtaining the distances of the k-th nearest-neighbor queries for $k = k_{\min}, \ldots, k_{\max}$. The efficiency of the DIC algorithm is determined by the efficiency of this index.

6. EXPERIMENTS

In this section we study experimentally the properties and the performance of the DIC algorithm.

6.1 Applications and Datasets

We apply our algorithms on the following types of datasets.

Embedded m-flats: Consider a set X of n points in $\mathbb{R}^d$ that can be decomposed in two subsets N, and F, of size s and f respectively, where $n = s + f$. The points in N are distributed uniformly at random in $(0,1)^d$. The points in F take values normally distributed around 0.5, with variance 0.01 in the first $d - m$ coordinates. In the last m coordinates, they take values uniformly distributed in $(0,1)$. As $s, f \to \infty$ the intrinsic dimensionality of the sets N and F approaches d and m respectively. We call the set F an *m-flat*. The value m is the dimension of the m-flat. The set

N can be thought of as an m-flat of dimension d, so we say that N has *full dimension*.

The objective of the algorithms is to partition the set X into sets N and F. We apply the DIC algorithm on X, requesting 2 clusters. We will demonstrate that the DIC algorithm, is able to return the sets F and N as the clusters even when m and d are relatively close. The flat F is the set of nodes with the smaller average intrinsic dimensionality.

Manifolds within manifolds: The setting is similar to the previous one, only this time the set X contains more than one m-flats of different dimensions. Namely, the set X can be decomposed into sets $N, F_1, \ldots, F_p$, where N has full dimension d, and $F_1, F_2, \ldots, F_p$ are m-flats with dimensions $m_1 < m_2 < \ldots < m_p$ respectively. The m-flats are constructed as described above. Note that since for every flat F_i we always "fix" the first $d - m_i$ coordinates, the m-flats with lower dimension are embedded within the m-flats of higher dimensions. This results in creating a chain hierarchy of manifolds where every manifold is embedded in all the preceding ones in the chain.

Again, we apply the DIC algorithm, requesting $p+1$ clusters. When the dimensionalities of the m-flats are sufficiently separated, the algorithm returns as clusters that p flats and the set N. The average estimated dimension values for each set are ordered according to the actual dimension of the flats.

Low Rank Sub-Matrices: The input is an $n \times m$ matrix that takes values in $[0, 1]$. Within the matrix there is a collection of k rows and ℓ columns, such that the combinatorial $k \times \ell$ sub-matrix has low rank. The objective is to identify the rows and columns of this sub-matrix.

We generate such datasets as follows. First we generate a $k \times \ell$ matrix S of rank exactly r, where $r \ll \min\{n, m\}$. We then plant it in the matrix M. The remaining elements of M are generated uniformly at random, scaled so that the mean is zero and the standard deviation is one. Therefore, if we remove either the k rows, or the ℓ columns of matrix S from M, we obtain a matrix of rank $\min\{n-k, m\}$ and $\min\{n, m-\ell\}$ respectively. To this matrix we add a "noise" matrix X with entries distributed normally around 0, with variance 0.05.

In order to extract S from the matrix M we apply the DIC algorithm in two steps. First we perform a clustering of the rows, and we identify the rows of the matrix S. The dimension of these rows is $m-\ell+r$, as opposed to m which is for the rest of the rows, so it is easy for the DIC algorithm to identify them. We then cluster the columns of M. The dimension of the columns in S is $n-k+r$ as opposed to n for the rest of the columns, so again DIC manages to partition the rows. Given the rows and columns we can extract matrix S.

6.2 Experiments with the DIC algorithm

In this section we present experiments with the DIC algorithm on various datasets. In all runs of the algorithm, we set $k_{\min} = 10$, and $k_{\max} = 100$, two values that we observed that they work well in practice.

We start by experimenting with datasets that contain a single m-flat F, embedded in a space of higher dimension d, together with a set N of noise points distributed uniformly at random. The datasets are constructed as described in section 6.1. Since the objective is to separate the sets F and N, we evaluate our algorithm by looking into the *total classification error* of the algorithm. The total classification error E_{tot} is computed as follows: We first compute the confusion matrix C whose C_{ij} entry contains the number of overlapping points between the i-th cluster of the ground truth and the j-th cluster of the clustering found by the algorithm. Then $E_{tot} = 1 - (\sum_i \max_j C_{ij})/n$.

Our experiments indicate that the DIC algorithm performs exceptionally well in this setting, even in the case that the dimension of the host space and the m-flat are very close, or if the m-flat is embedded in a high dimensional space. Figure 6(a) plots the local representation of the data points when $d = 3$ and $m = 2$, and their clustering. Figure 6(b) shows the the case where $d = 50$ and $m = 40$. In both cases, the size of the dataset is 1,000 points, of which 500 belong to the m-flat. We observe that the algorithm manages to identify the m-flats successfully. The total classification error is 8.1% in the first case, and 1.2% in the second case.

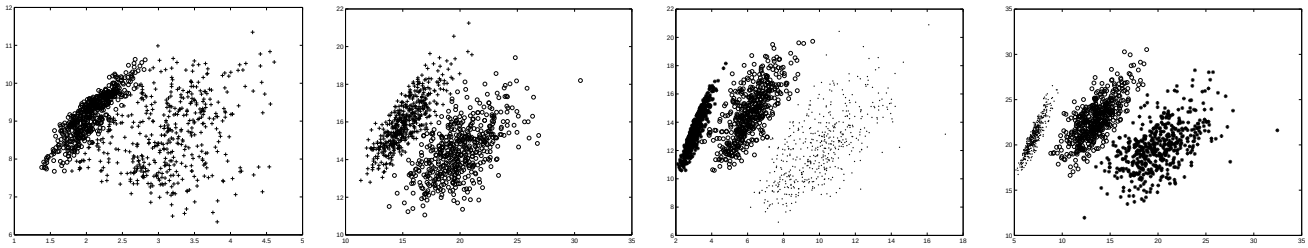
In order to better understand the performance of DIC, we performed a more detailed experiment, generating datasets with the dimension of the host space being $d = 2 \ldots 10$, and the dimension of the m-flat ranging from 1 to $d-1$. In all cases, the dataset consists of 1,500 points, 500 of which belong to the m-flat. Table 1 reports the average classification error for 20 runs of the algorithm (the numbers are percentages). We observe that the classification error is never more than 39%, and this occurs in the case that the dimension of the host space and that of the m-flat differ by just one.

We now turn our attention to cases where there are more than one m-flats in the dataset. Figures 6(c) and (d) show the plots of the local representations of two datasets that contain flats of different dimension. In the first case the dimension of the host space is $d = 10$ and the two manifolds have dimension $m_1 = 3$, and $m_2 = 6$. In the second case we have $(m_1, m_2, d) = (10, 20, 30)$, In both cases all three sets of points F_1, F_2, N contain 500 points each. We observe that the DIC algorithm manages to discriminate the three sets. The average classification error is 1.53% for the first case, and 0.51% for the second case, where the average is taken over 20 runs. Datasets with more than three flats are examined in the full version of the paper.

We also experiment with low rank matrices, trying to detect a (combinatorial) 100×100 submatrix of rank 2, within a 1000×1000 matrix. The algorithm proves to be quite successful, obtaining classification error just 0.16%, where the average is taken over 10 runs. In this case the large dimension of the matrix works in favor of our algorithm. The algorithm often achieves a perfect partition of the matrix (4 out of the 10 runs).

The problem of finding low rank sub-matrices has been applied to microarray data. Wang et. al. [23] report a solution for rank-1 sub-matrices. We experiment with the same microarray data used in [23][1]. The data contains the expression levels of 2884 genes (rows) for 17 different patients (columns). Finding combinatorial low-rank matrices in such types of data is important since they represent subsets of genes that are co-regulated on some subsets of patients. Figure 7(a) shows a plot of the local representation of the rows (genes) of the matrix. The scatter plot shows no obvious clustering except a few outliers. A more detailed density

[1]The data can be obtained at `http://arep.med.harvard.edu/biclustering/yeast.matrix`

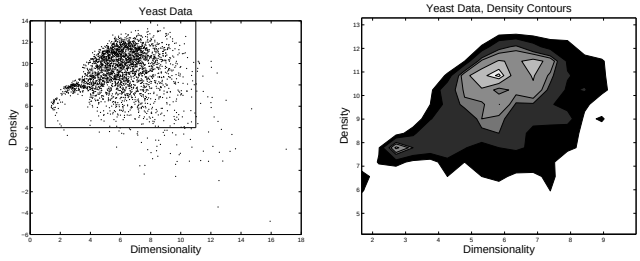


(a) 2D-flat in 3D (b) 40D flat in 50D (c) 3D and 6D flats in 10D (d) 10D and 20D flats in 30D

Figure 6: Discovering m-flats with DIC, (x: dimensionality, y: density)

	1	2	3	4	5	6	7	8	9
2	9.2								
3	13.0	20.14							
4	14.9	1.53	29.28						
5	16.1	0.26	6.74	26.42					
6	15.4	0.08	0.68	6.99	31.1				
7	7.1	0.02	0.17	1.25	13.7	33.4			
8	10.5	0.00	0.02	0.41	2.1	14.6	36.3		
9	1.4	0	0.01	0.08	0.6	2.9	18.7	37.9	
10	7.4	0.01	0.01	0.04	0.2	0.9	4.2	20.7	38.3

Table 1: Classification error of discovering m-flat clusters



(a) (b)

Figure 7: Analysis of microarray data, (a) dimensionality and density for each row, (b) density contours in the rect. area of (a) reveal two clusters.

estimation of the 2D data reveals two hidden clusters. The figure 7(b) shows the density contour lines, estimated by a 30×30 histogram (light gray means high density). There is one small cluster the rows of which has dimensionality around 3 and lower density than the bigger cluster with intrinsic dimensionality of about 6. Note that the two clusters can not be found using either density or intrinsic dimensionality. This shows that combining density with intrinsic dimensionality is an improvement upon previous methods.

6.3 Comparison with OPTICS

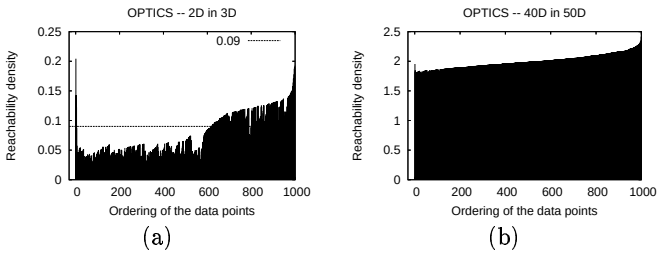
In this subsection we compare our algorithm with OPTICS on the task of finding m-flats within noise. OPTICS takes as input the parameter ϵ_{max} which is the maximum linkage distance, and it produces an ordered visualization of the points in the dataset from which the lower part of a cluster hierarchy can be derived. In order to get rid of the dependency from ϵ_{max} we set it in all cases to the maximum distance in the particular data set, so that OPTICS computes the whole hierarchy.

The primary output of the OPTICS algorithm is a plot. The x axis of the plot shows the indices of the data points in the ordering produced by OPTICS. The y axis in the visualization is reachability distance, which is small if the density at that point is high. The computed hierarchy by OPTICS is similar to a single linkage hierarchy. The plot produced by OPTICS defines clusters as valleys. A valley is defined by a horizontal cutting line, which is chosen by the user. In case of sub-clusters within larger clusters, the plot by OPTICS consists of a large valley, which includes at the bottom smaller valleys divided by small hills.

The OPTICS algorithm assumes that each cluster in the true hierarchy consists of at least two sub-clusters. However, in case of m-flats embedded in other m-flats of higher dimension, this is not true. But one can still look for knees at the right side of a valley in the visualization plot of OPTICS, as it is shown in Figure 8(a). This allows to specify a cutoff value for the hierarchical algorithm. The cutting line should be set to the beginning of the knee. Note that in case that we have multiple m-flats, one embedded within the other it is not possible for OPTICS to identify all m-flats using a single cutoff value.

In our comparison with OPTICS we consider datasets where a single m-flat is embedded in a higher dimensional space. The datasets contain 1,000 points, while both the m-flat, as well as the noise set (containing points, uniformly distributed in the full-dimensional space) consists of 500 points each. The following data sets are generated: 2D-flat in 3D, 3D-flat in 5D, 5D-flat in 8D, 6D-flat in 10D, 40D-flat in 50D, and 90D-flat in 100D.

Figure 8(a) shows the plot generated by OPTICS for the 2D-flat in 3D. The knee at the right side of the lowest valley is clearly visible and so we choose the cutting value to be 0.09. However, for data with dimension larger than 10, although the algorithm produces the correct ordering with most of the points of the m-flat being in the beginning of the ordering, the knee is not longer visible. An example is shown in figure 8(b). Therefore, we could not compute a



(a) (b)

Figure 8: OPTICS plots for (a) 2D-flat within 3D noise (b) 40D-flat within 50D noise

Data	E_{tot}(OPTICS)	E_{tot}(DIC)
2D in 3D	0.115	0.077
3D in 5D	0.045	0.029
5D in 8D	0.087	0.024
6D in 10D	0.045	0.010
40D in 50D	n.a.	0.010
90D in 100D	n.a.	0.072

Table 2: Classification Error of OPTICS and DIC

clustering with OPTICS for the last two high-dimensional data set. Any value seems equally good, resulting in arbitrarily good, or bad results. The same problem arises when trying to use OPTICS for identifying low-rank sub-matrices.

In Table 2 we compare the classification error of the clusterings found by OPTICS and the DIC algorithm. The classification errors are similar, with DIC being a little more accurate. The main conclusion from this experiment is that the density-based clustering method OPTICS fails for high-dimensional data. In such cases density alone is not sensitive enough to reveal the structure of the data. Furthermore, the OPTICS algorithm requires careful fine-tuning of the parameters in order to produce a meaningful clustering, as opposed to the DIC algorithm which has only few, easy to set parameters. Finally, we note that the DIC algorithm is order independent, as opposed to OPTICS which is sensitive to the order in which the points are visited.

The case of multiple m-flats is considerably more difficult for OPTICS. Multiple m-flats will appear as a single valley since they differ only by density and not by location, which means that one has to find multiple knees. However, the visibility of the knees degrades as the dimensionality of the m-flats increases.

7. CONCLUSIONS

We address the problem of discovering clusters of points that lie on low-dimensional manifolds. Our approach is to extend the definition of fractal correlation dimension and create a local-growth model for each point. Based on this model, each point in the dataset can be mapped to a local representation consisting of a density coefficient and a dimensionality coefficient. We argue that the local representation preserves well the information about the manifolds that points belong to, and discovering those manifolds becomes a two-dimensional clustering problem.

Our method is able to discover low-dimensional manifolds that are not necessarily linear, it can find clusters within clusters, as well as clusters that occupy the same space. Furthermore the method does not require a vector-space representation of the data; it can be used equally well for metric datasets. We perform experiments in which we demonstrate the effectiveness of our algorithms for discovering low-dimensional m-flats and for detecting low-rank submatrices. We also show that our method outperforms other approaches that are based only on density and do not take into account the notion of dimensionality.

8. REFERENCES

[1] P. K. Agarwal and N. H. Mustafa. k-means projective clustering. In *PODS*, pages 155–165, 2004.

[2] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proc. SIGMOD*, pages 61–72. ACM, 1999.

[3] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. SIGMOD*, pages 70–81. ACM, 2000.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. SIGMOD*, pages 94–105, 1998.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. SIGMOD*, pages 49–60, 1999.

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *JACM*, 45(6):891–923, 1998.

[7] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proc. KDD*, pages 260–264, 2000.

[8] A. Belussi and C. Faloutsos. Self-spacial join selectivity estimation using fractal concepts. *ACM TOIS*, 16(2):161–201, 1998.

[9] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proc. VLDB*, pages 28–39, 1996.

[10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. VLDB*, pages 426–435, 1997.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusteris in large spatial databases with noise. In *Proc. KDD*, pages 226–231, 1996.

[12] C. Faloutsos and I. Kamel. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In *Proc. PODS*, pages 4–13, 1994.

[13] R. F. S. Filho, A. J. M. Traina, J. Caetano Traina, and C. Faloutsos. Similarity search without tears: The omni family of all-purpose access methods. In *Proc. ICDE*, pages 623–630, 2001.

[14] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, pages 518–529, 1999.

[15] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[16] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. KDD*, pages 58–65, 1998.

[17] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *SIGMOD*, 1997.

[18] R. Krauthgamer and J. R. Lee. The black-box complexity of nearest neighbor search. In *Proc. ICALP*, pages 858–869, 2004.

[19] B.-U. Pagel, F. Korn, and C. Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *Proc. ICDE*, pages 589–598, 2000.

[20] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proc. ICDE*, pages 315–325, 2003.

[21] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proc. SIGMOD*, pages 418–427. ACM Press, 2002.

[22] S. N. Rasband. *Chaotic Dynamics of Nonlinear Systems*. Wiley-Interscience, 1990.

[23] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proc. SIGMOD*, pages 394–405. ACM Press, 2002.