Hierarchical Clustering via Localized Diffusion Folders

Gil David GIL.DAVID@YALE.EDU
Department of Mathematics
Program in Applied Mathematics
Yale University
New Haven, CT 06510, USA

Amir Averbuch AMIR@MATH.TAU.AC.IL
School of Computer Science
Tel-Aviv University
Tel-Aviv, 69978, Israel

Ronald R. Coifman COIFMAN@MATH.YALE.EDU
Department of Mathematics
Program in Applied Mathematics
Yale University
New Haven, CT 06510, USA

Editor:

Abstract

Data clustering is a common technique for statistical data analysis. It is used in many fields including machine learning, data mining, customer segmentation, trend analysis, pattern recognition and image analysis. The proposed Localized Diffusion Folders methodology performs hierarchical clustering and classification of high-dimensional datasets. The diffusion folders are multi-level data partitioning into local neighborhoods that are generated by several random selections of data points and folders in a diffusion graph and by defining local diffusion distances between them. This multi-level partitioning defines an improved localized geometry of the data and a localized Markov transition matrix that is used for the next time step in the diffusion process. The result of this clustering method is a bottom-up hierarchical clustering of the data while each level in the hierarchy contains localized diffusion folders of folders from the lower levels. This methodology preserves the local neighborhood of each point while eliminating noisy connections between distinct points and areas in the graph. The performance of the algorithms is demonstrated on real data and it is compared to existing methods. The proposed solution is generic since it fits a large number of related problems where the source datasets contain high dimensional data.

Keywords: Hierarchical Clustering, Diffusion Geometry, Spectral Graph

1. Introduction

The diffusion maps framework (Coifman and Lafon , 2006a,b) and its inherent diffusion distances provide a method for finding meaningful geometric structures in datasets. In most cases, the dataset contains high dimensional data points in $\mathbb{R}^n$. The diffusion maps construct coordinates that parameterize the dataset and the diffusion distance provides a local preserving metric for this data. A non-linear dimensionality reduction, which reveals global

geometric information, is constructed by local overlapping structures. Let $\Gamma = \{x_1, \ldots, x_m\}$ be a set of points in $\mathbb{R}^n$ and μ is the distribution of the points on Γ. We construct the graph $G(V, E)$, $|V| = m$, $|E| \ll m^2$, on Γ in order to study the intrinsic geometry of this set. A weight function $W_\epsilon \triangleq w_\epsilon(x_i, x_j)$ is introduced. It measures the pairwise similarity between the points. For all $x_i, x_j \in \Gamma$, the weight function has the following properties: symmetry: $w_\epsilon(x_i, x_j) = w_\epsilon(x_j, x_i)$, non-negativity: $w_\epsilon(x_i, x_j) \geq 0$ and positive semi-definite: for all real-valued bounded function f defined on Γ, $\int_\Gamma \int_\Gamma w_\epsilon(x_i, x_j) f(x_i) f(x_j) d\mu(x_i) d\mu(x_j) \geq 0$.

A common choice for W_ϵ is $w_\epsilon(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\epsilon}}$.

The non-negativity property of W_ϵ allows to normalize it into a Markov transition matrix P where the states of the corresponding Markov process are the data points. This enables to analyze Γ as a random walk. The construction of P is known as the *normalized graph Laplacian* (Chung , 1997).

Formally, $P = \{p(x_i, x_j)\}_{i,j=1,\ldots,m}$ is constructed as $p(x_i, x_j) = \frac{w_\epsilon(x_i, x_j)}{d(x_i)}$ where $d(x_i) = \int_\Gamma w_\epsilon(x_i, x_j) d\mu(x_j)$ is the degree of x_i. P is a Markov matrix since the sum of each row in P is 1 and $p(x_i, x_j) \geq 0$. Thus, $p(x_i, x_j)$ can be viewed as the probability to move from x_i to x_j in *one* time step. By raising this quantity to a power t (advance in time), this influence is propagated to nodes in the neighborhood of x_i and x_j and the result is the probability for this move in t time steps. We denote this probability by $p_t(x_i, x_j)$. These probabilities measure the connectivity among the points within the graph. The parameter t controls the scale of the neighborhood in addition to the scale control provided by ϵ.

Construction of $\tilde{p}(x_i, x_j) = \frac{\sqrt{d(x_i)}}{\sqrt{d(x_j)}} p(x_i, x_j)$, which is a symmetric and positive semi-definite kernel, leads to the following eigen-decomposition:
$\tilde{p}(x_i, x_j) = \sum_{k \geq 0}^{m} \lambda_k \nu_k(x_i) \nu_k(x_j)$. A similar eigen-decomposition is obtained from
$\tilde{p}_t(x_i, x_j) = \sum_{k \geq 0}^{m} \lambda_k^t \nu_k(x_i) \nu_k(x_j)$ after advancing t times on the graph. Here $\tilde{p}_t$ is the probability of transition from x_i to x_j in t time steps.

A fast decay of $\{\lambda_k\}$ is achieved by an appropriate choice of ϵ. Thus, only a few terms are required in the sum above to achieve a given relative cover $\delta > 0$. Let $\eta(\delta)$ be the number of the retained terms.

A family of diffusion maps was introduced in Coifman and Lafon (2006a). They are $\Phi_t(x)_{m \in \mathbb{N}}$ given by $\Phi_t(x) = \left(\lambda_0^t \nu_0(x), \lambda_1^t \nu_1(x), \ldots\right)^T$.

The map $\Phi_m : \Gamma \to l^{\mathbb{N}}$ embeds the dataset into an Euclidean space. The *diffusion distance* is defined as $D_t^2(x_i, x_j) = \sum_{k \geq 0} \left(\tilde{p}_t(x_i, x_k) - \tilde{p}_t(x_k, x_j)\right)^2$. This formulation is derived from the known random walk distance in Potential Theory. It is shown that the diffusion distance can be expressed in terms of the right eigenvectors of P: $D_t^2(x_i, x_j) = \sum_{k \geq 0} \lambda_k^{2t} \left(\nu_k(x_i) - \nu_k(x_j)\right)^2$. It follows that in order to compute the diffusion distance, one can simply use the eigenvectors of $\tilde{P}$. Moreover, this facilitates the embedding of the original points in an Euclidean space $\mathbb{R}^{\eta(\delta)-1}$ by:
$\Xi_t : x_i \to \left(\lambda_0^t \nu_0(x_i), \lambda_1^t \nu_1(x_i), \lambda_2^t \nu_2(x_i), \ldots, \lambda_{\eta(\delta)}^t \nu_{\eta(\delta)}(x_i)\right)$. This also provides coordinates on the set Γ. Essentially, $\eta(\delta) \ll n$ due to the fast spectral decay of the spectrum of P. Furthermore, $\eta(\delta)$ depends only

on the primary intrinsic variability of the data as captured by the random walk and not on the original dimensionality of the data. This data-driven method enables the parameterization of any set of points - abstract or not - provided the similarity matrix of the points

W_ϵ is available.

As described in the brief overview of the diffusion maps above, P is the affinity matrix of the dataset and it is used to find the diffusion distances between data points. This distance metric can be used to cluster the data points according to the propagation of the diffusion distances that are controlled by t. In addition, it can be used to construct a bottom-up hierarchical clustering of the data. For $t = 1$, the affinity matrix reflects local and direct connections between adjacent data points. The resulting clusters preserve the local neighborhood of each point. These clusters are the bottom level of the hierarchy. By raising t (advancing in time), the affinity matrix is changed accordingly and it reflects indirect connections between data points in the graph. The diffusion distance between data points in the graph represents all possible paths between these points according to a given time step. The more we advance in time, the more we increase indirect and global connections. Therefore, by raising t we can construct the upper levels of the clustering hierarchy. In each advance in time, it is possible to merge more and more lower-level clusters since there are more and more new paths between them. The resulting clusters reflect global neighborhood of each point, which is highly affected by parameter t.

The major risk in this global approach is that by increasing t, the noise (connections between points that are not related) in the affinity matrix increases as well. Moreover, errors in clustering in the lower levels of the hierarchy will diffuse to the upper levels of the hierarchy and hence, will significantly influence the upper levels clustering. As a result, some areas in the graph that should be separated will be connected by the new (noise-result and error-result) paths. This will cause fast and wrong diffusion to different areas in the graph and convergence of the data points (and clusters of data points) to only few clusters. Since the noise and errors significantly affect the diffusion process, the resulting clusters do not reflect properly the underlying physical phenomena that we model. Although these clusters consist of data points that are adjacent according to their diffusion distances, the connections among these points in each cluster can be too global and loose. Thus, the accuracy of the clustering is decreased.

In this paper, we present an hierarchical clustering method of high-dimensional data via what we call *localized diffusion folders* (LDF) (David , 2009). This method overcomes the problems that were described above. The key idea is that clustering of data points should be achieved by utilizing the local geometry of the data and the local neighborhood of each data point and by constructing a new local geometry every advance in time step. The new geometry is constructed according to the local connections and according to the diffusion distances in the previous time steps. This way, as we advance in time, the geometry of the affinity improves, the noise in the new localized matrix decreases and the accuracy of the resulting clusters is improved. In order to construct the new localized geometry, we introduce the idea of LDF. The LDF are multi-level partitioning (Voronoi diagrams) of the data into local neighborhoods that are initiated by several random selections of data points or folders of data points in the diffusion graph and by defining local diffusion distances between them. Since every different selection of initial points yields a different set of folders, it is crucial to repeat this selection process several times. The multiple system of folders (Voronoi diagrams), which we get at the end of this random selection process, define a new geometry and a new affinity on the graph. This affinity is a result of a "shake

n bake" process: first, we "shake" the multiple Voronoi diagrams together in order to get rid of the noise in the original affinity. Then, we "bake" a new clean affinity that is based on the actual geometry of the data while eliminating rare connections between points. This affinity is more accurate than the original affinity since instead of defining a general affinity on the graph, we let the data define a local affinity on the graph. In every time step, this multi-level partitioning defines a new localized geometry of the data and a new localized affinity matrix that is used for the next time step. In every time step, we use the localized geometry and the localized folders that were generated in the previous time step in order to define the localized affinity between folders. The affinity between two folders is defined by the localized diffusion distance metric between the points in the two folders. In order to define this distance between the folders, we construct a local sub-matrix that contains only the affinity of the points (or folders) of the two folders. This sub-matrix is raised by the power of the current time step (according to the current level in the hierarchy) and then it is used to find the localized diffusion distance between the two folders. The result of this clustering method is a bottom-up hierarchical data clustering where each level in the hierarchy contains LDF of folders from the lower levels. Each level in the hierarchy defines a new localized affinity (geometry) that is dynamically constructed and it is used by the upper level. This methodology preserves the local neighborhood of each point while eliminating the noisy connections between distinct points and areas in the graph.

The paper has the following structure. Section 2 presents related work. In Section 3, we present in details the methodology of the localized diffusion folders. Section 4 presents experimental results.

2. Related Work

Many clustering algorithms have been proposed over the years. Most of them cluster numerical data, where the data consists of numeric attributes whose values are represented by continuous variables. Finding similarity between numeric objects usually relies on common distance measures such as Euclidean, Manhattan, Minkowski, Mahalanobis distances to name some. A comprehensive survey of several clustering algorithms is given in Gan, Ma, and Wu (2007).

$k-$means (Macqueen , 1967) is one of the most used clustering algorithm. It was designed to cluster numerical data in which each cluster has a center called the mean. The $k-$means algorithm is classified as either partitioner or non-hierarchical clustering method. The performance of $k-$means is highly dependent on the initialization of the centers. This is its major drawback. Furthermore, it does not perform effectively on high-dimensional data.

An agglomerative hierarchical algorithm is proposed in BIRCH (Zhang, Ramakrishnan andLivny, 1996). It is used for clustering large numerical datasets in Euclidean spaces. BIRCH performs well when clusters have identical sizes and their shapes are either convex or spherical. However, it is affected by the input order of the data and it may not perform well when clusters have either different sizes or non-spherical shapes.

CURE (Guha, Rastogi and Shim , 1998) is another method for clustering numerical datasets using hierarchical agglomerative algorithm. CURE can identify non-spherical shapes in large databases that have different sizes. It uses a combination of random sam-

pling and partitioning in order to process large databases. Therefore, it is affected by a random sampler performance.

A density-based clustering algorithm is proposed in DBSCAN (Ester, Kriegel, Sander and Xu , 1996). This method is used to discover arbitrarily shaped clusters. DBSCAN is sensitive to its parameters, which in turn, are difficult to determine. Furthermore, DBSCAN does not perform any pre-clustering and it executed directly on the entire database. As a result, DBSCAN can incur substantial I/O costs in processing large databases.

DIANA (Kaufman and Rousseeuw , 1990) is a divisive hierarchical algorithm that applies to all datasets that can be clustered by hierarchical agglomerative algorithms. Since the algorithm uses the largest dissimilarity between two objects in a cluster such as the diameter of the cluster, it is sensitive to outliers.

Several clustering algorithms for categorical data have been proposed in recent years.

The $k-$modes algorithm (Huang , 1998) emerged from the $k-$means algorithm and it was designed to cluster categorical datasets. The main idea of the $k-$modes algorithm is to specify the number of clusters and then to select k initial modes, followed by allocating every object to its nearest mode. The algorithm minimizes the dissimilarity of the objects in a cluster with respect to its mode.

The inter-attribute and intra-attribute summaries of a database are constructed in CACTUS (Ganti, Gehrke and Ramakrishnan , 1999). Then, a graph, called the similarity graph, is defined according to these summaries. Finally, the clusters are found with respect to these graphs.

More recently, some clustering kernel methods were proposed. A clustering method, which uses support vector machine, is given in Ben-Hur, Horn, Siegelmann and Vapnik (2001). Data points are mapped by a Gaussian kernel to a high dimensional feature space, where it searches the minimal enclosing sphere. When this sphere is mapped back to data space, it can be separated into several components where each encloses a separate cluster.

A method for unsupervised partitioning of a data sample, which estimates the possible number of inherent clusters that generate the data, is described in Girolami (2002). It exploits the notion that performing a nonlinear data transformation into some high dimensional feature space increases the probability for linear separability between the patterns within the transformed space. Therefore, it simplifies the associated data structure. It shows that the eigenvectors of a kernel matrix, which define an implicit mapping, provide means to estimate the number of clusters inherent within the data. A computational iterative procedure is presented for the subsequent feature space that partitions the data.

A kernel clustering scheme, which is based on $k-$means for large datasets, is proposed in Zhang and Rudnicky (2002). It introduces a clustering scheme which changes the clustering order from a sequence of samples to a sequence of kernels. It employs a disk-based strategy to control the data.

Another kernel clustering scheme, which is based on $k-$means, is proposed in Couto (2005). It uses a kernel function that is based on Hamming distance to embed categorical data in a constructed feature space where clustering takes place.

An efficient algorithm for clustering large high dimensional datasets, called Smart-Sample, is described in David, Averbuch and Lazarov (2009). Its performance is compared with k-means, k-means++, LSH (Gionis, Indyk and Motwani , 1999), BIRCH and CURE.

Our proposed method is different from the above clustering methods in several aspects. First, it provides a multi-level hierarchical clustering of the data where each level t in the hierarchy expresses the diffusion geometry of the data after t time steps. A level in our hierarchy means an advance in time in the diffusion graph. Moreover, our method is unique since it constructs a new localized geometry of the data in every time step. This enables a more accurate representation of the geometry of the data and, as a result, it provides a better clustering. Last, in every level in the hierarchy, our method cleans the data from noisy data points which are the global loose connections between data points.

3. Localized Diffusion Folders (LDF)

The basic operation in the two-scale localized diffusion folders algorithm has two steps. In the first step, the bottom level of the hierarchy is constructed as follows: First, the data points are partitioned into diffusion folders. Then, multiple systems of diffusion folders are constructed. Next, a localized affinity that defines the "purified" geometry ("shake n bake" process) is constructed. Last, the data is partitioned into localized diffusion folders.

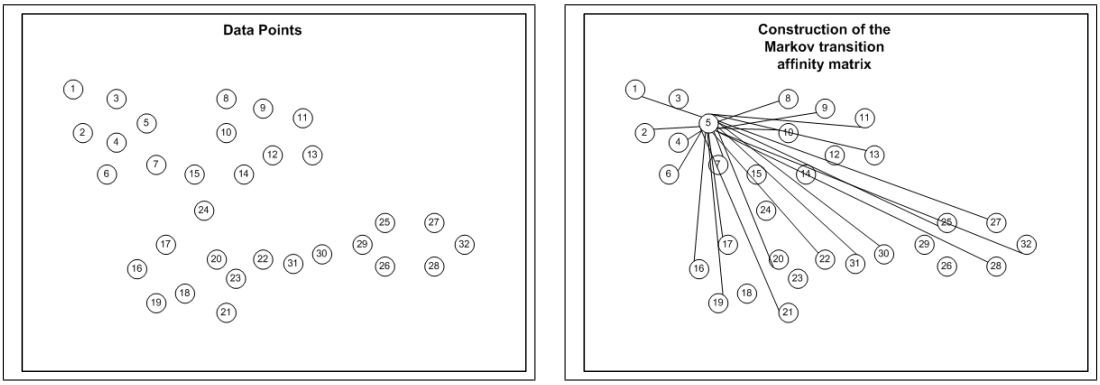
In the second step, the upper levels of the hierarchy are constructed as follows: First, the localized affinity between the lower level diffusion folders is constructed. Then, multiple systems of diffusion folders of folders are constructed. Next, a localized affinity that defines the "purified" geometry ("shake n bake" process) is constructed. Last, the folders are partitioned into localized diffusion folders. This process is repeated up to the root of the hierarchy.

In this section, we describe the LDF algorithm in details. Section 3.1 provides an illustrated intuition for this methodology. Section 3.2 describes the construction of the bottom level in the hierarchy ($t = 1$). Section 3.3 describes the construction of the upper levels in the hierarchy ($t > 1$).

3.1 Outline of the methodology of the LDF

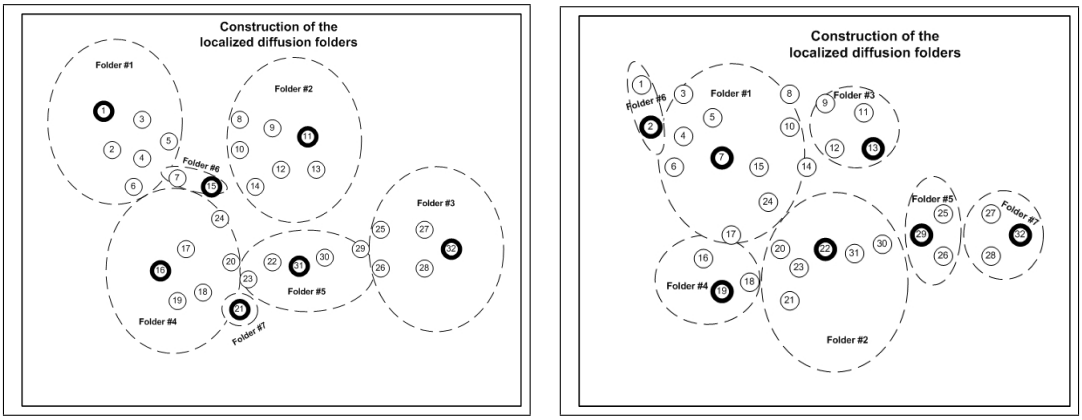
In order to provide intuition for the proposed construction, we use some simple examples to illustrate this methodology. The input to the algorithm is a set of data points as shown in Fig. 3.1(a). The first step constructs the Markov transition affinity matrix. This matrix defines the pairwise diffusion distances between every point in the graph to any other point in the graph. It is illustrated in Fig. 3.1(b). Then, the diffusion folders are constructed using this affinity matrix. Initial partitioning of the points into non-overlapping LDF according to initial random selections of data points takes place.

Figure 3.2(a) illustrates a system of non-overlapping LDF. The points in bold are the initial random selections. Since the resulting diffusion folders depend on the selection of the initial points, we repeat this process several times with different random selections of the initial points. Each process yields a different system of diffusion folders. Figure 3.2(b) shows a different system of non-overlapping LDF with different initial random selections. We repeat this process several times according to the size of the dataset. The result of this process is a set of LDF systems where each system depends on the initial selection of the random points. Figure 3.3(a) shows this set of systems. Now, we have different sets of folders where each point can reside in a different folder in each system. In other words, for every set of folders, each point can be associated to a different neighborhood. We construct



(a) Set of data points (b) Construction of the affinity matrix

Figure 3.1: First step

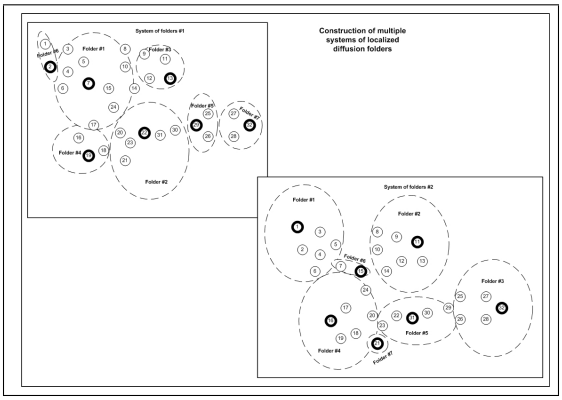


(a) Construction of the LDF (b) Construction of another set of LDF

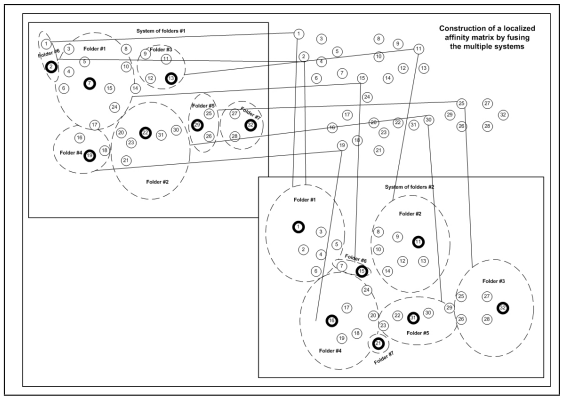
Figure 3.2: Second step

the new localized affinity matrix by fusing multiple systems from the diffusion folders. For each point, its distance to all other points is defined as the relative probability to reside in the same neighborhood in the multiple systems of the diffusion folders. This way, we use different sets of LDF to define a new affinity between points. This affinity is more accurate than the original affinity since it reflects the actual neighborhoods of each point. It also reduces the noise by eliminating rare connections. Figure 3.3(b) shows the construction of the localized affinity matrix by fusing multiple systems. Once we have this improved affinity, the super-system of localized diffusion folders is constructed using this localized affinity matrix. Figure 3.3(c) shows this constructed system. At this point, we constructed the bottom level of the hierarchy. This level contains folders of data points.

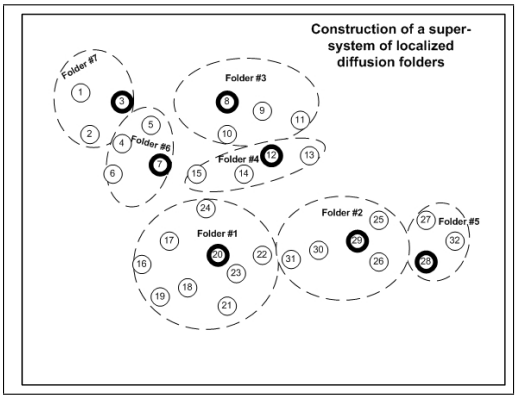
In order to improve the clustering, we build the next (upper) level in the hierarchy. This level reflects the clustering of the data after advancing one time step (propagation of the points). This level constructs LDF of the folders that were constructed in the lower level. In order to construct these folders, we define an affinity between folders. Therefore, we build this affinity between folders by defining the pairwise diffusion distances between the lower



(a) Construction of multiple systems of LDF



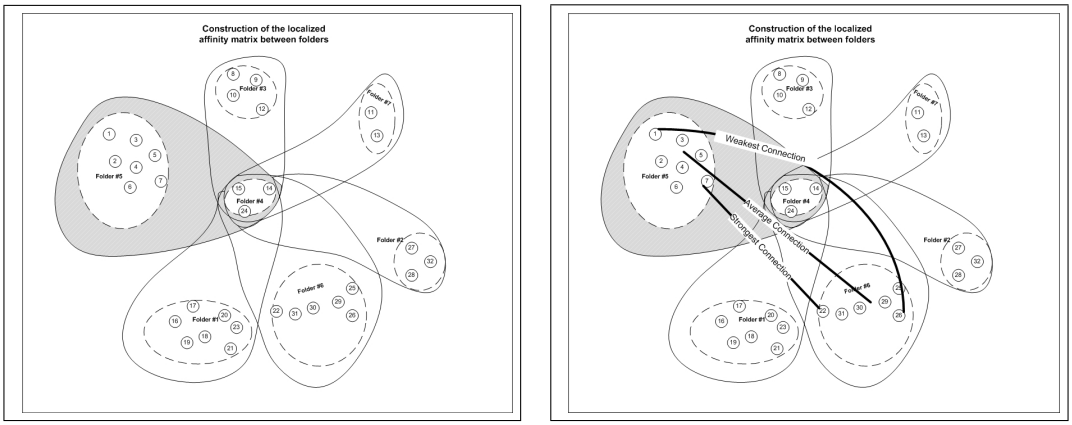
(b) Construction of the localized affinity matrix



(c) Construction of a super system of LDF

Figure 3.3: Third step

level diffusion folders using the lower level localized affinity matrix. This diffusion distance is computed locally by raising the local sub-matrix of both folders by the power of the current level that is constructed in the hierarchy (the advance in time). Figure 3.4(a) shows the construction of the localized affinity matrix between pairs of folders. Figure 3.4(b) shows the different distance metrics that are used for the construction of the localized affinity matrix between pairs of folders. Once the affinity between folders is defined, we continue with the construction process of the upper level of the hierarchy as described in the construction of the bottom level of the hierarchy. This way, we advance in time and more levels are built in the hierarchy. Figure 3.5 shows the final bottom-up hierarchy.



(a) Construction of the affinity between folders (b) Measuring the distance between folders

Figure 3.4: Fourth step

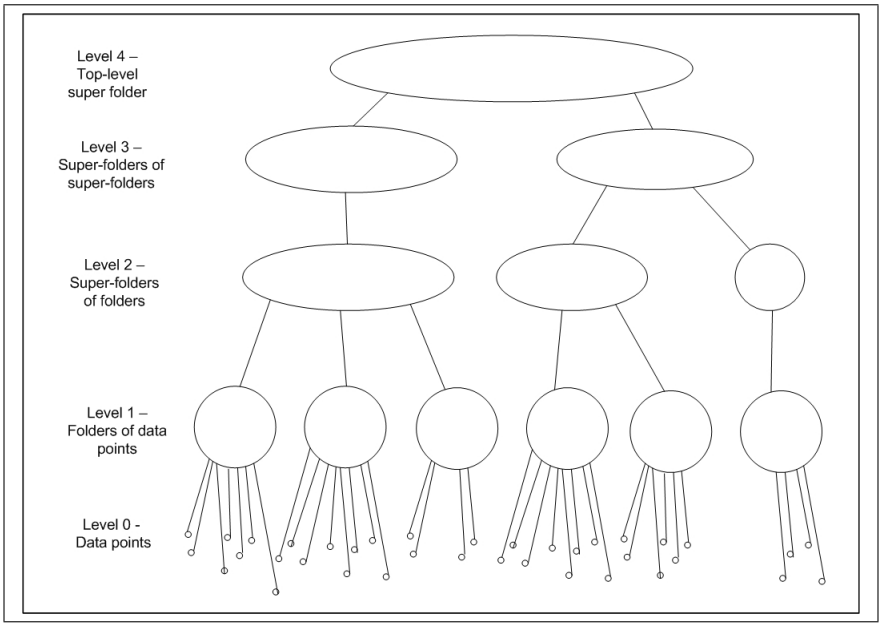


Figure 3.5: Bottom-up hierarchy

3.2 Construction of the bottom level in the hierarchy

1. **Input**
   Let C be a matrix of size $m \times n$ of data points. m is the number of data points and n is the number of features in each data point. Each data point is a row vector in C.

2. **Normalization of the matrix C**
   It is necessary in order to transform all the features to a common scale. One of the common normalization methods of the matrix C is done by taking the logarithmic value of each feature. Assume $r, 1 \leq r \leq m$, is a row in C denoted by $c_r \stackrel{\Delta}{=} \{c_{ri} : 1 \leq$

$i \leq n\}$. The normalized vector a_r is constructed by $a_r = \log(c_r)$, $r = 1, \ldots, m$. A is the output matrix after normalizing each row $r = 1, \ldots, m$ in the input matrix C.

3. **Processing the normalized matrix A - construction of the similarity matrix $\tilde{A}$**

   Denote the row vector $i, 1 \leq i \leq m$, in the normalized matrix A by $a_i \stackrel{\Delta}{=} \{a_{ik} : 1 \leq k \leq n\}$. The pairwise distances between the data points in A are computed using a weight metric to produce $\tilde{A}$ whose entries are $\tilde{a}_{ij}$. Common distance metrics are:

   Euclidean distance metric: $\tilde{a}_{ij} \stackrel{\Delta}{=} \{\sqrt{(a_i - a_j) \cdot (a_i - a_j)^T} \colon i, j = 1, \ldots, m\}$.

   Weighted Euclidean distance metric: $\tilde{a}_{ij} \stackrel{\Delta}{=} \left\{ \sqrt{(\frac{a_i - a_j}{w}) \cdot (\frac{a_i - a_j}{w})^T} \colon i, j = 1, \ldots, m \right\}$

   where $w \stackrel{\Delta}{=} \{w_k : 1 \leq k \leq n\}$ is a weight factor vector. As w_k becomes larger, the influence of the k-th feature on the distance between a_i and a_j becomes smaller.

   Cosine distance metric: $\tilde{a}_{ij} \stackrel{\Delta}{=} \left\{ \left(1 - \frac{a_i \cdot {a_j}^T}{\sqrt{{a_i}^T \cdot a_i} \cdot \sqrt{{a_j}^T \cdot a_j}} \right) \colon i, j = 1, \ldots, m \right\}$.

   Mahalanobis distance metric: $\tilde{a}_{ij} \stackrel{\Delta}{=} \{\sqrt{(a_i - a_j) \cdot \Sigma^{-1} \cdot (a_i - a_j)^T} \colon i, j = 1, \ldots, m$ and Σ is the covariance matrix$\}$. Σ can also be the features matrix.

4. **Processing the similarity matrix $\tilde{A}$ - construction of the Gaussian kernel K**

   Construction of a Gaussian kernel: We construct the Gaussian kernel $K_{ij} = e^{-\frac{\tilde{a}_{ij}}{\epsilon}}$, $i, j = 1, \ldots, m$. For each point $a_i \in A$, this Gaussian kernel pushes away from a_i all the points that are already far away from a_i (according to $\tilde{A}$). On the other hand, it pulls towards a_i all the points that are already close to a_i. This push-pull process is controlled by ϵ. Since ϵ is fixed for all the entries in $\tilde{A}$, it produces a coarse scaling control. This scale control is obviously not optimal for all the entries in the Gaussian kernel since it does not take into account the local geometry of each data point in the graph. Figure 3.6 shows an example of data points in R^3. In this example, we chose $\epsilon = 0.02$ as the scale control. The orange/red points in this image are the neighbors of the green points according to the constructed Gaussian kernel (using the fixed scale control). These are the points that were pulled towards the green points.

   In the left image, we see that although the green point is in a very dense area, it has only few neighbors. In this case, we are interested in pulling more points towards the green point. This is achieved by selecting a larger ϵ (that provides a wider radius) since it will include more points in the neighborhood of the green point. However, in the right, image we see that although the green point is an outlier, it has relatively many neighbors where some of them are in areas that are well separated from the green point. In this case, we are interested in pushing more points away from the green point. This is achieved by selecting a smaller ϵ (that provides a narrower radius) since it will include less points in the neighborhood of the green point.

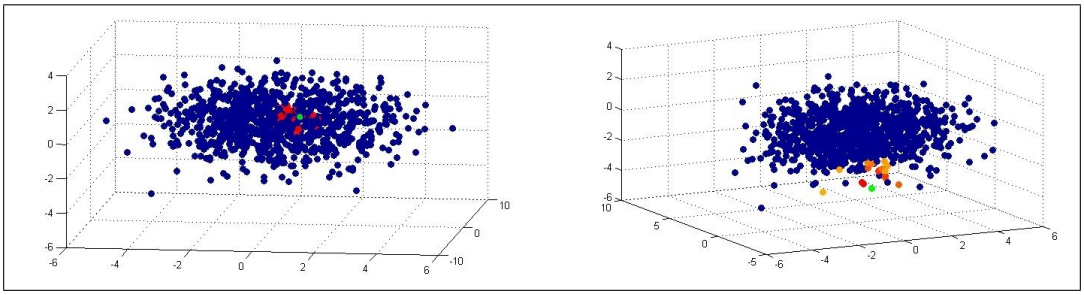


Figure 3.6: Gaussian kernel with fixed ϵ. Left: Normal point. Right: Outlier point

Therefore, a selection of an adaptive scale control is necessary in order to construct a more accurate Gaussian kernel that will express the local geometry of each data point in the graph.

Construction of an adaptive Gaussian kernel: The key idea of the construction of the adaptive Gaussian kernel is to determine automatically an adaptive scale control for each point in the graph. This adaptive scale control is a weight function that has the following property: data points in dense areas will have a large weight ("bonus") and data points in sparse areas will have a small weight ("penalty"). Since dense-areas points are close to many points and sparse-areas points are far away from many points, we define the weight function ω to be ${\omega^\epsilon}_i = \int_A e^{-\frac{\tilde{a}_{ij}}{\epsilon}} d\mu(a_j)$ where $A = \{a_1, \ldots, a_m\}^T$, μ is the distribution of the points on A and ϵ is an initial scale control. ${\omega^\epsilon}_i$, $i = 1, \ldots, m$ defines an adaptive weight for each data point. However, since we construct an affinity matrix between pairs of data points, we need to define a pairwise weight function. We determine this way not only an adaptive scale for each point in A, but we also determine an adaptive scale for each pair of points. We define the pairwise weight function Ω^ϵ to be ${\Omega^\epsilon}_{ij} = \sqrt{\int_A e^{-\frac{\tilde{a}_{ik}}{\epsilon}} d\mu(a_k) . \int_A e^{-\frac{\tilde{a}_{jk}}{\epsilon}} d\mu(a_k)} = \sqrt{{\omega^\epsilon}_i . {\omega^\epsilon}_j}$. Ω^ϵ satisfies the "bonus" and "penalty" property. Moreover, Ω^ϵ is symmetric and non-negative. Now, we construct the adaptive Gaussian kernel K as follows: $K_{ij} = e^{-\frac{\tilde{a}_{ij}}{\sqrt{{\omega^\epsilon}_i . {\omega^\epsilon}_j}}} = e^{-\frac{\tilde{a}_{ij}}{{\Omega^\epsilon}_{ij}}}$, $i, j = 1, \ldots, m$. Since both $\tilde{A}$ and Ω^ϵ are symmetric and non-negative, the constructed kernel K is symmetric and non-negative as well. This adaptive scale control provides better and compact description of the local geometric properties of the pairwise distances matrix $\tilde{A}$. This process can be repeated several times and K_{ij} represents optimally the nature of the local geometry of $\tilde{A}$.

Figure 3.7 shows an example of the same data points in R^3. In this example, we constructed the adaptive Gaussian kernel that was described above using an initial $\epsilon = 0.02$. In the left image we see that the green point has more neighbors than when we used the fixed scale control (Fig. 3.6). The reason is that this green point is in a very dense area and therefore it is close to many points. Therefore, the adaptive scale control is much bigger than the initial scale control (a "bonus" was awarded) and as a result more points were pulled towards the green point. However, in the right image we see that the green point has fewer

neighbors than when we used the fixed scale control. The reason is that this green point is an outlier and therefore it is far from most of the points. Therefore, the adaptive scale control is much smaller than the initial scale control (a "penalty" was assigned) and as a result more points were pushed away from the green point. These selections of the adaptive scale controls provide a more accurate Gaussian kernel.

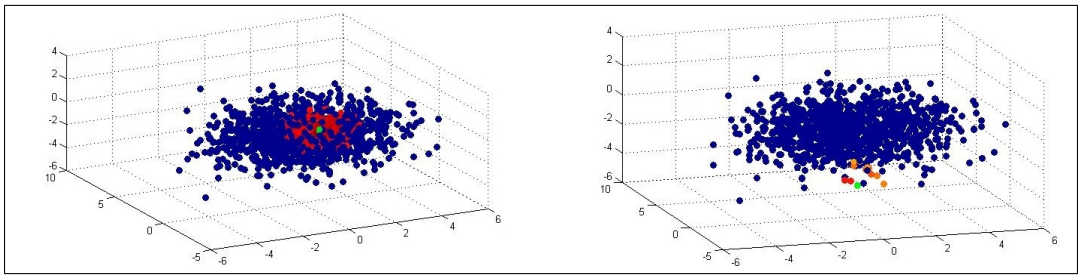


Figure 3.7: Gaussian kernel with an adaptive scale control. Left: Normal point. Right: Outlier point

Figure 3.8 shows the diffusion maps coordinates of the data points that use the Gaussian kernel with the fixed scale control and the proposed kernel with the adaptive scale control. As we can see, when we use the fixed scale control (the left image) we get a jelly-fish shape where the separation between the normal points (the body) and the outliers (the tails) is unclear. However, when we use the adaptive scale control (the right image), we get a bell shape where the separation between the normal points (the bell) and the outliers is very clear. This structure represents the geometric of the original points more accurately.

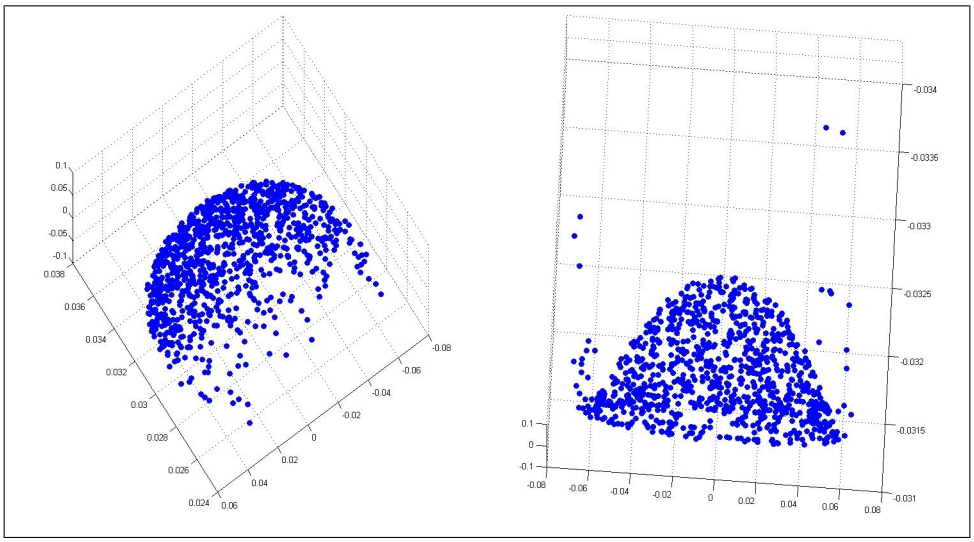


Figure 3.8: Left: Diffusion maps with fixed scale control. Right: Adaptive scale control

5. **Normalizing the Gaussian kernel K into a Markov transition matrix**
The non-negativity property of K allows to normalize it into a Markov transition matrix P^t where the states of the corresponding Markov process are the data points. This enables to analyze C as a random walk. K_{ij} is normalized into a matrix P^t_{ij} by one of the following methods:

   Graph Laplacian matrix: $P^t_{ij} = \frac{K_{ij}}{\sqrt{\sum_{q=1}^m K_{iq}}\cdot\sqrt{\sum_{q=1}^m K_{jq}}}$.

   Laplace-Beltrami matrix: First, we compute the graph Laplacian matrix $\tilde{P}^t_{ij} = \frac{K_{ij}}{\sqrt{\sum_{q=1}^m K_{iq}}\cdot\sqrt{\sum_{q=1}^m K_{jq}}}$. We repeat this process to get the Laplace-Beltrami matrix $P^t_{ij} = \frac{\tilde{P}^t_{ij}}{\sqrt{\sum_{q=1}^m \tilde{P}^t_{iq}}\cdot\sqrt{\sum_{q=1}^m \tilde{P}^t_{jq}}}$.

   P^t is a Markov matrix since the sum of each row in P^t is 1 and $P^t_{ij} \geq 0$. Thus, P^t_{ij} can be viewed as the probability to move from a_i to a_j in t time steps. P^1 is used as the initial affinity matrix of the graph.

6. **Construction of the LDF D^t using the affinity matrix P^t**
First, an initial partitioning of the points into non-overlapping diffusion folders is performed. Let a_i be a random point in the dataset. We denote by $N_\epsilon(a_i) \stackrel{\Delta}{=} \{a_j : P^t_{ij} > \epsilon, 1 \leq j \leq m, i \neq j\}$ the neighborhood of a_i. This neighborhood contains all the neighbors of the random selected point a_i that their distance to a_i, according to the affinity matrix P^t, is greater than ϵ. Now, the diffusion folder is denoted by $D^t_i \stackrel{\Delta}{=} \{a_j : a_j \in N_\epsilon(a_i), a_j \notin D^t_k, 1 \leq k \leq q, k \neq i, q \ll m\}$. Since the neighborhood $N_\epsilon(a_i)$ can contain points that were already associated to neighborhoods of other points, we add to the diffusion folder of a_i only the points that have not already been associated to any other diffusion folder. This way, we partition the points into non-overlapping diffusion folders. Next, we choose another unassociated random point a_l in the dataset where $a_l \notin D^t_k, 1 \leq k \leq q, q \ll m$ and we repeat the above process in order to construct its diffusion folder D^t_l. The whole process is repeated until all points are associated to the constructed diffusion folders.

   Once all the points are assigned and we have a set $D^t_k, 1 \leq k \leq q, q \ll m$ of diffusion folders, we have to verify that each point is associated to its nearest neighborhood in order to get accurate diffusion folders. This is achieved by reassignment of the points to their nearest neighborhood and then reconstruct the diffusion folders takes place accordingly. For each point $a_i, i = 1, \ldots m$, we calculate its average affinity $\mu(a_i, D^t_k)$ to each of the diffusion folders $D^t_k, 1 \leq k \leq q, q \ll m$ according to the affinity matrix P^t: $\mu(a_i, D^t_k) = \frac{1}{|D^t_k|}\sum_{l=1}^{|D^t_k|} P_{iD^t_{k_l}}$, where $P_{iD^t_{k_l}}$ is the affinity between a_i to the l^{th} point in D^t_k. Finally, a_i is reassigned to the diffusion folder with the maximum average affinity μ. Once this process is performed for all points, each point is reassigned to one of the diffusion folders $D^t_k, 1 \leq k \leq q, q \ll m$. Hence, at the end of this process, the members in each diffusion folder can be changed (according to the reassignments). Therefore, the whole process of the reassignment of the points and the reconstruction of the folders is repeated several times, until the convergence of

the diffusion folders is achieved and all the folders become stable (there are no more movements of points between folders). The result of this process is a Voronoi diagram (Aurenhammer , 1991) D^t of LDF that is affected by the initial random selection of the points.

7. **Construction of multiple systems $\tilde{D}^t$ of LDF**
In the previous step, we constructed a Voronoi diagram D^t of LDF. Since this system was affected by the initial random selection of the points, we repeat this step r times until we get r different Voroni systems of LDF. We denote the set of multiple systems of LDF by $\tilde{D}^t \triangleq \{D^{(t,k)} : k = 1, \ldots, r\}$, where $D^{(t,k)}$ is the k^{th} system of LDF. $\tilde{D}^t$ contains the r systems of diffusion folders where the construction of each system of folders was initiated with different selections of random points.

8. **Construction of the localized affinity matrix $\hat{P}^t$ - "shake n bake" process**
We fuse the multiple Voronoi systems of the LDF $\tilde{D}^t$ in order to get rid of the noise in the original affinity ("shake"). Then, we construct a cleaner and more accurate affinity matrix $\hat{P}^t$ ("bake"). The key idea of this process is that two points are close to each other iff they are associated to the same folder in different systems (at least in the majority of the systems).

   Figure 3.9 shows four different Voronoi diagrams that were constructed according to different selections of the centers (of the Voronoi cells). We can see that most of the time, although the centers (blue points) are changed, the data points (red points) are associated to the same Voronoi cells. However, the noisy point (green point) is associated to different Voronoi cells. This means that we need to construct and examine different Voronoi diagrams using different centers in order to find and eliminate non-consistent associations. Therefore, we define the following metric space (Frechet , 1906) (A, d) where $A = \{a_1, \ldots, a_m\}$ is the set of points and d is a metric on A such that $d : A \times A \to \mathbb{R}$ is

$$d(a_i, a_j) = \begin{cases} 0 & a_i = a_j \\ \frac{1}{2} & a_i \neq a_j \text{ and } a_j \in D_q^t(a_i) \\ 1 & a_i \neq a_j \text{ and } a_j \notin D_q^t(a_i) \end{cases}$$

   where $D_q^t(a_i)$ is the folder q in the Voronoi system D^t that contains a_i. In order to prove that (A, d) is a metric space, we have to show that for any $a_i, a_j, a_l \in A$ the following conditions are satisfied:

   i. Non-negativity: $d(a_i, a_j) \geq 0$;

   ii. Identity: $d(a_i, a_j) = 0$ iff $a_i = a_j$;

   iii. Symmetry: $d(a_i, a_j) = d(a_j, a_i)$;

   iv. Triangle inequality: $d(a_i, a_j) \leq d(a_i, a_l) + d(a_l, a_j)$.

   Conditions i-iii are satisfied by the definition of d. We prove that condition iv is satisfied by contradiction. Assume to the contrary that $d(a_i, a_j) > d(a_i, a_l) + d(a_l, a_j)$. Since d is non-negative then there are two cases to consider: (1) $d(a_i, a_j) = \frac{1}{2}$ or (2) $d(a_i, a_j) = 1$.

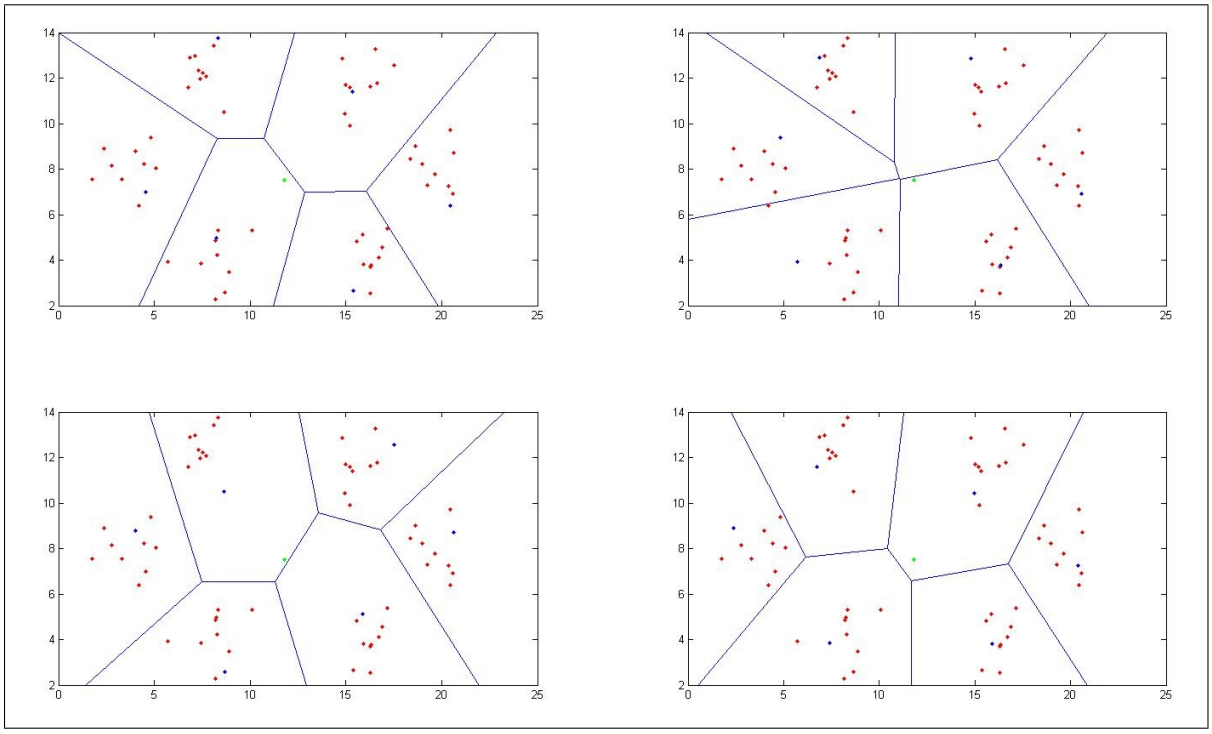


Figure 3.9: Four different Voronoi diagrams of the same dataset

(a) If $d(a_i, a_j) = \frac{1}{2}$, then $d(a_i, a_l) = 0$ and $d(a_l, a_j) = 0$. However, if $d(a_i, a_l) = 0$ and $d(a_l, a_j) = 0$ then $a_i = a_l$ and $a_l = a_j$ (identity). Therefore, we conclude that $a_i = a_j = a_l$ and $d(a_i, a_j) = 0$, contradicting our assumption that $d(a_i, a_j) = \frac{1}{2}$;

(b) If $d(a_i, a_j) = 1$, then there are three cases to consider: (a) $d(a_i, a_l) = 0$ and $d(a_l, a_j) = 0$ or (b) $d(a_i, a_l) = \frac{1}{2}$ and $d(a_l, a_j) = 0$ or (c) $d(a_i, a_l) = 0$ and $d(a_l, a_j) = \frac{1}{2}$.

   a) If $d(a_i, a_l) = 0$ and $d(a_l, a_j) = 0$ then we conclude that $d(a_i, a_j) = 0$ (identity), contradicting our assumption that $d(a_i, a_j) = 1$;

   b) If $d(a_i, a_l) = \frac{1}{2}$ and $d(a_l, a_j) = 0$ then since $a_l = a_j$ (identity) we conclude that $d(a_i, a_j) = \frac{1}{2}$, contradicting our assumption that $d(a_i, a_j) = 1$;

   c) Similarly, if $d(a_i, a_l) = 0$ and $d(a_l, a_j) = \frac{1}{2}$ then we conclude that $d(a_i, a_j) = \frac{1}{2}$, contradicting our assumption that $d(a_i, a_j) = 1$.

Therefore, the triangle inequality (condition iv) holds and (A, d) is a metric space. Moreover, as we can see in the above proof, the following condition is satisfied as well:

v. Strong inequality: $d(a_i, a_j) \leq max\{d(a_i, a_l), d(a_l, a_j)\}$.

Hence, (A, d) is an ultrametric space (Kaplansky , 1977).

In order to fuse the multiple Voronoi systems of the diffusion folders $\tilde{D}^t$, we define the following metric space (A, d_μ) where $A = \{a_1, \ldots, a_m\}$ is the set of points and d_μ is a metric on A such that $d_\mu : A \times A \to \mathbb{R}$ is

$d_\mu(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)$, where $|\tilde{D}^t|$ is the number of Voronoi systems in $\tilde{D}^t$ and $d_k(a_i,a_j)$ is the distance between a_i to a_j according to the metric d and the Voronoi system $D^{(t,k)}$.

As for (A,d), in order to prove that (A,d_μ) is a metric space, we have to show that for any $a_i,a_j,a_l\in A$ the following conditions hold:

i. Non-negativity: $d_\mu(a_i,a_j)\geq 0$;
ii. Identity: $d_\mu(a_i,a_j)=0$ iff $a_i=a_j$;
iii. Symmetry: $d_\mu(a_i,a_j)=d_\mu(a_j,a_i)$;
iv. Triangle inequality: $d_\mu(a_i,a_j)\leq d_\mu(a_i,a_l)+d_\mu(a_l,a_j)$.

Satisfying Condition i: due to the non-negativity property of d, $d_k(a_i,a_j)\geq 0$ for any k. Therefore, $d_\mu(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)\geq 0$. Hence, (A,d_μ) satisfies the non-negativity condition.

Satisfying condition ii: first we prove the left side of the identity. Due to the non-negativity property of d, $d_k(a_i,a_j)\geq 0$ for any k. Therefore, if $\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)=0$ then $d_k(a_i,a_j)=0$ for any k. Due to the identity property of d, $a_i=a_j$ for any k. Hence, if $d_\mu(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)=0$ then $a_i=a_j$.

Now we prove the right side of the identity. Since $a_i=a_j$ then for any k, $d_k(a_i,a_j)=0$. Therefore, $d_\mu(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)=0$. Hence, (A,d_μ) satisfies the identity condition.

Satisfying Condition iii: due to the symmetry property of d, $d_k(a_i,a_j)=d_k(a_j,a_i)$ for any k. Therefore,

$d_\mu(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_j,a_i)=d_\mu(a_j,a_i)$.

Hence, (A,d_μ) satisfies the symmetry condition.

Satisfying Condition iv: due to the triangle inequality property of d, $d_k(a_i,a_j)\leq d_k(a_i,a_l)+d_k(a_l,a_j)$ for any k. Therefore,

$$\begin{aligned}
d_\mu(a_i,a_j)&=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_j)\\
&\leq\frac{(d_1(a_i,a_l)+d_1(a_l,a_j))+\ldots+(d_{|\tilde{D}^t|}(a_i,a_l)+d_{|\tilde{D}^t|}(a_l,a_j))}{|\tilde{D}^t|}\\
&=\frac{(d_1(a_i,a_l)+\ldots d_{|\tilde{D}^t|}(a_i,a_l))+d_1(a_l,a_j)+\ldots d_{|\tilde{D}^t|}(a_l,a_j))}{|\tilde{D}^t|}\\
&=\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_i,a_l)+\frac{1}{|\tilde{D}^t|}\sum_{k=1}^{|\tilde{D}^t|}d_k(a_l,a_j)\\
&=d_\mu(a_i,a_l)+d_\mu(a_l,a_j).
\end{aligned}$$

Hence, (A, d_μ) satisfies the triangle inequality condition and (A, d_μ) is a metric space.

We define the localized affinity matrix $\hat{P}$ as $\hat{P}^t_{ij} = 1 - d_\mu(a_i, a_j)$.

This affinity $\hat{P}^t$ is localized and it reduces the noise by eliminating rare collections between points. This affinity defines a new geometry on the data according to the actual different partitions of the data. Instead of defining a general affinity on the data, we let the data define the local affinity by itself. We normalize this affinity into a Markov transition matrix (normalized graph Laplacian or normalized Laplace-Beltrami) as described in Section 3.2(5).

9. **Construction of the super-system $\hat{S}^t$ of LDF using $\hat{P}^t$**
Once we have the localized affinity matrix $\hat{P}^t$, which was constructed from the set of multiple systems of LDF $\tilde{D}^t$, we can construct the super-system $\hat{S}^t$ of LDF. The construction of $\hat{S}^t$ is similar to the construction of the LDF D^t that was described in Section 3.2(6) but instead of using the original affinity matrix P^t we use the improved affinity matrix $\hat{P}^t$. $\hat{S}^t$ is a super-system of the systems of LDF $\tilde{D}^t$. This last step finalizes the construction of the bottom level of the hierarchy of the LDF. An initial partitioning (clustering) $\hat{S}^t$ of the data into LDF is achieved. These LDF are the input for the next upper level in the hierarchy of the LDF. The final localized affinity matrix $\hat{P}^t$ is also an input to the next upper level.

In order to demonstrate the advantage of the clusters, which were generated by the LDF algorithm in the bottom level over the clusters that were generated in the embedding process of the diffusion maps algorithm, we use the following swiss roll examples. Figures 3.10-3.15 show the clustering results where each color in each image represents a different cluster.

First, we generate a swiss roll without any noise. We apply the diffusion maps and cluster the data in the embedding. Figures 3.10-3.11 show the clustering results from the use of different ϵ values (while constructing the Gaussian kernel). In Fig. 3.10, we can see that by using $\epsilon = 0.01$, the diffusion maps clusters the data points successfully (the clusters are changing along the swiss roll and there are no clusters that include points from two different sides of the gaps). However, once ϵ is increased (Fig. 3.11), the clusters are not accurate anymore and we can see clusters that include points from both sides of the gaps. This example demonstrates the sensitivity of the method to the ϵ parameter. Next, we generate a swiss roll with noise. We apply the diffusion maps and cluster the data in the embedding. Figures 3.12-3.13 show the clustering results from the use of different ϵ values (while constructing the Gaussian kernel). As we can see, although we try different values of ϵ, the clusters are not accurate and we can see clusters that include points from both sides of the gaps. This example demonstrates the sensitivity of the method to noise. Last, we apply the LDF to the same noisy data. Figures 3.14-3.15 show the clustering results from the use of different ϵ values (while constructing the initial Gaussian kernel in the construction of the adaptive Gaussian kernel). We see that for most ϵ values, the LDF clusters successfully the data points. The clusters are changing along the swiss roll and there are no clusters that include points from two different sides of the gaps.

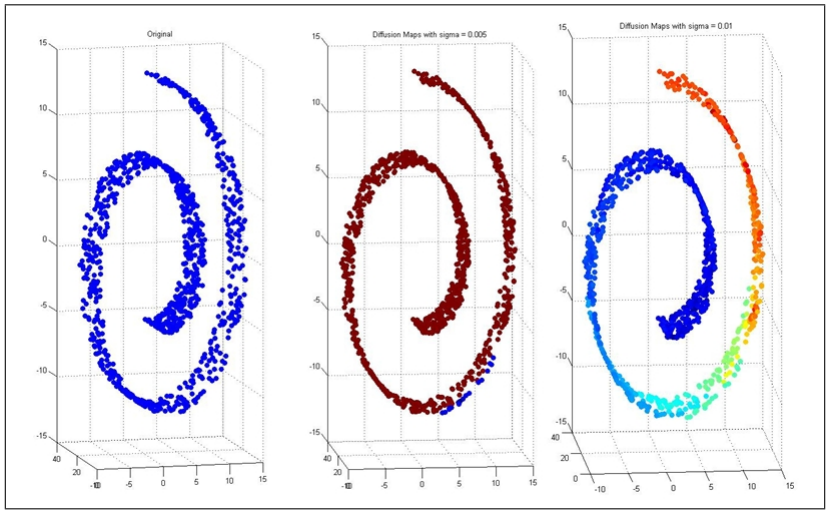


Figure 3.10: Left: Original. Middle: DM with $\epsilon = 0.005$. Right: DM with $\epsilon = 0.01$

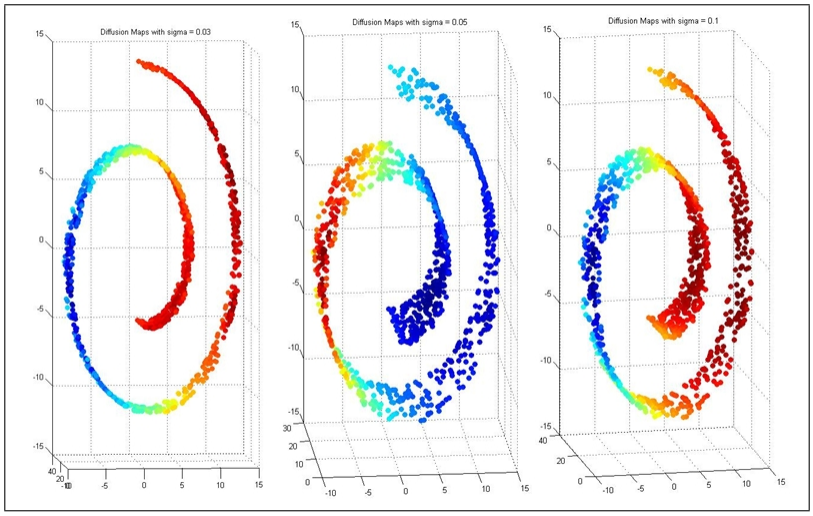


Figure 3.11: Left: DM with $\epsilon = 0.03$. Middle: DM with $\epsilon = 0.05$. Right: DM with $\epsilon = 0.1$

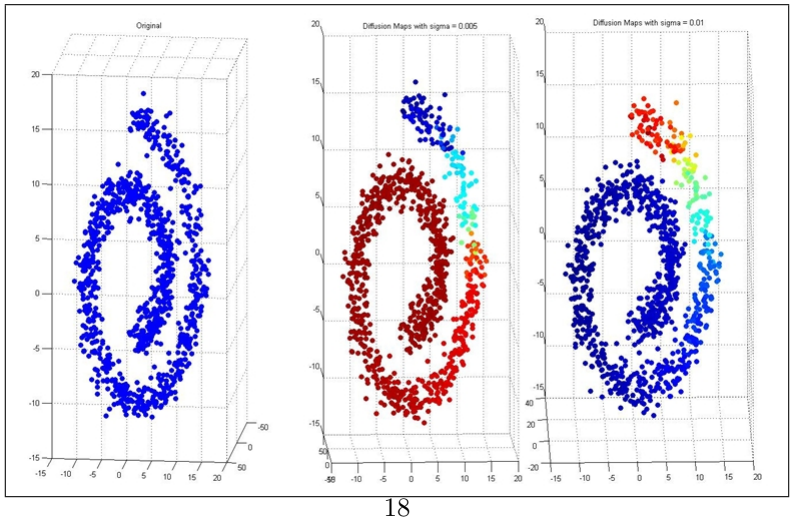


Figure 3.12: Left: Original noisy. Middle: DM with $\epsilon = 0.005$. Right: DM with $\epsilon = 0.01$

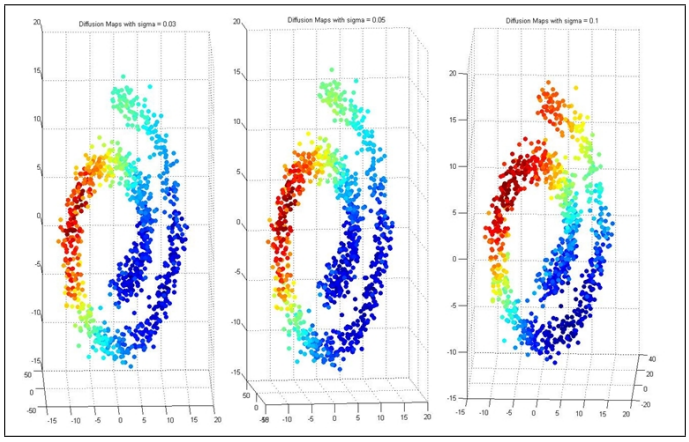


Figure 3.13: Left: DM with $\epsilon = 0.03$. Middle: DM with $\epsilon = 0.05$. Right: DM with $\epsilon = 0.1$

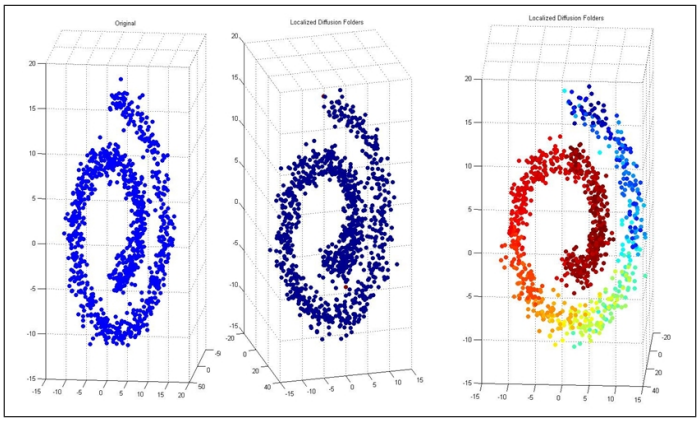


Figure 3.14: Left: Original noisy. Middle: LDF with $\epsilon = 0.005$. Right: LDF with $\epsilon = 0.01$

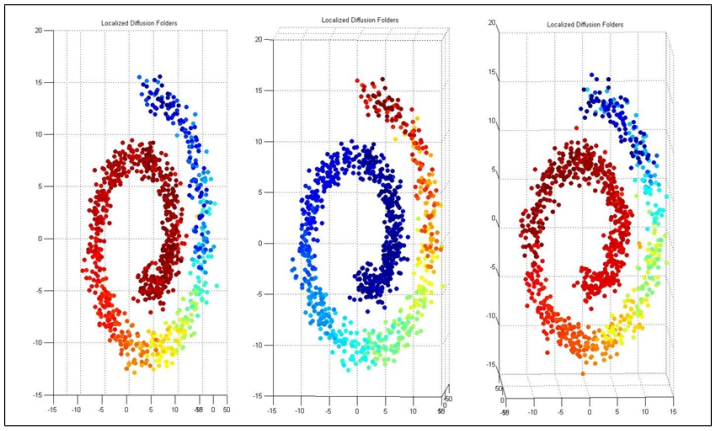


Figure 3.15: Left: LDF with $\epsilon = 0.03$. Middle: LDF with $\epsilon = 0.05$. Right: LDF with $\epsilon = 0.1$

3.3 Construction of the upper levels in the hierarchy

1. **Input**
   Let $\hat{S}^{t-1}$ be the LDF and let $\hat{P}^{t-1}$ be the localized affinity matrix that were constructed in the lower level in the hierarchy of the LDF (time t-1). Let t be the current level in the hierarchy. t is the time step.

2. **Construction of the localized affinity matrix P^t between the lower-level folders in $\hat{S}^{t-1}$**
   Let $|\hat{S}^{t-1}|$ be the number of LDF in $\hat{S}^{t-1}$. In order to construct a localized affinity matrix P^t between the lower-level folders in $\hat{S}^{t-1}$, the pairwise diffusion distances between each pair of folders in $\hat{S}^{t-1}$ are computed. Let $\hat{S}_k^{t-1}$ and $\hat{S}_l^{t-1}$ be two different folders in $\hat{S}^{t-1}$, $1 \leq k,l \leq |\hat{S}^{t-1}|$. Then, the sub-matrix of $\hat{P}^{t-1}$, which contains only the affinities between all the points in $\hat{S}_k^{t-1} \cup \hat{S}_l^{t-1}$ to all the points in $\hat{S}_k^{t-1} \cup \hat{S}_l^{t-1}$, is denoted by $\hat{P}_{kl}^{t-1}$.

   Figure 3.16 demonstrates the selection of the sub-matrix $\hat{P}_{kl}^{t-1}$. This sub-matrix includes the blue entries in the right image.

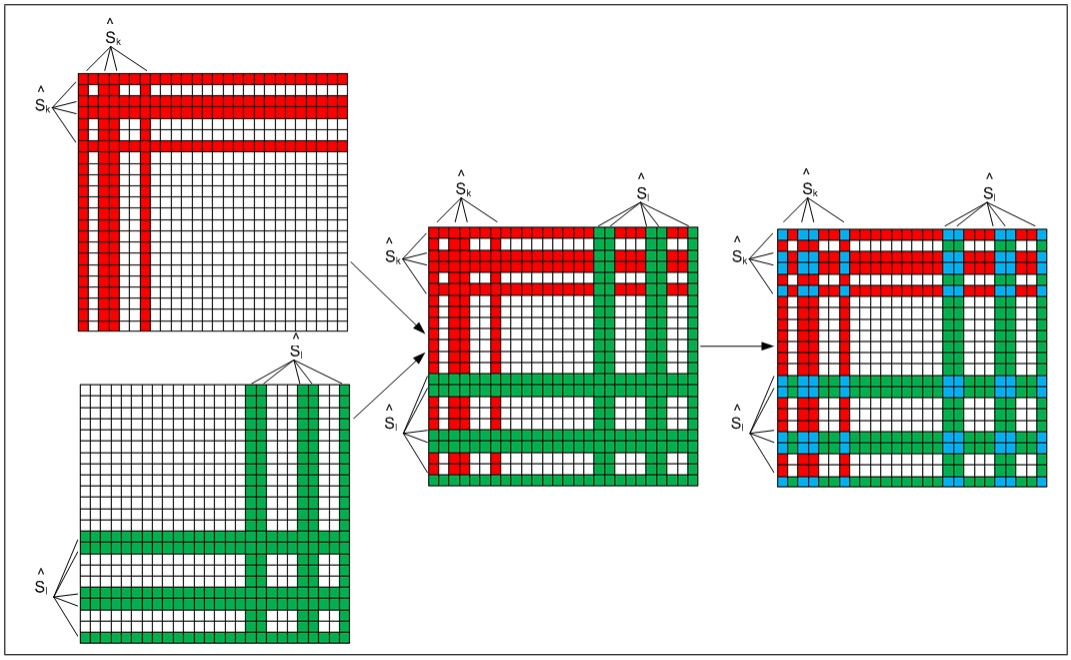


Figure 3.16: Selection of the sub-matrix $\hat{P}_{kl}^{t-1}$ of the affinities between the points in $\hat{S}_k^{t-1} \cup \hat{S}_l^{t-1}$

This localized affinity sub-matrix is raised by a power of 2^t, denoted by $Q_{kl}^t \stackrel{\Delta}{=} \hat{P}_{kl}^{t-1^{2^t}}$, to diffuse the affinities in time. Only the affinities between all the points in both folders propagate in time. This way, we eliminate distinct connections between them while reducing the noise and increasing the accuracy of the distance metric. Q_{kl}^t enables to

preserve only local collections. Since we are interested in the diffusion connectivity between the two folders, we need only the sub-matrix of Q_{kl}^t that contains the affinities between the points in $\hat{S}_k^{\,t-1}$ to the points in $\hat{S}_l^{\,t-1}$. We denote this sub-matrix by $\hat{Q}_{kl}^t$. Figure 3.17 demonstrates the process of generating the sub-matrix $\hat{Q}_{kl}^t$. This sub-matrix includes the yellow entries in the right image.

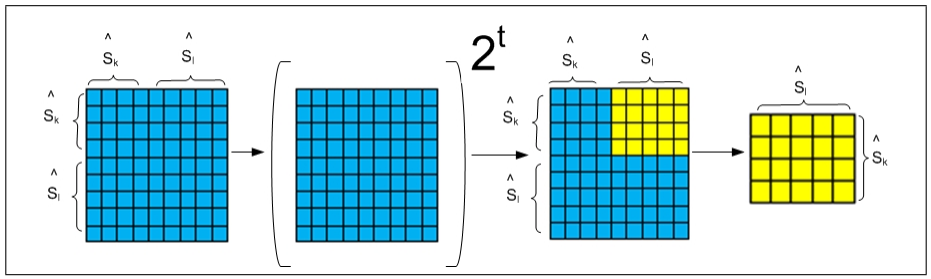


Figure 3.17: Generating the sub-matrix $\hat{Q}_{kl}^t$

Last, the affinity P_{kl}^t between the folders $\hat{S}_k^{\,t-1}$ and $\hat{S}_l^{\,t-1}$ is defined by one of the following metrics:

Fastest random runner: $P_{kl}^t = \max_{1\le i\le|\hat{S}_k^{\,t-1}|}\max_{1\le j\le|\hat{S}_l^{\,t-1}|}\hat{Q}_{kl_{ij}}^t$. This metric expresses the fastest path between the two folders. Since this path is determined by applying multiple random walks between one folder to another, we use the term *random runner*.

Slowest random runner: $P_{kl}^t = \min_{1\le i\le|\hat{S}_k^{\,t-1}|}\min_{1\le j\le|\hat{S}_l^{\,t-1}|}\hat{Q}_{kl_{ij}}^t$. This metric expresses the slowest path between the two folders.

Average random runner: $P_{kl}^t = \frac{\sum_{i=1}^{|\hat{S}_k^{\,t-1}|}\sum_{j=1}^{|\hat{S}_l^{\,t-1}|}\hat{Q}_{kl_{ij}}^t}{|\hat{S}_k^{\,t-1}||\hat{S}_l^{\,t-1}|}$. This metric expresses the average path between the two folders.

At the end of this process, P^t is a non-negative and symmetric matrix that contains the pairwise distances between any pair of folders. The size of P^t is $|\hat{S}^{t-1}|\cdot|\hat{S}^{t-1}|$. P^t is normalized into a Markov transition matrix (normalized graph Laplacian or normalized Laplace-Beltrami) as described in Section 3.2(5). P^t is the localized affinity matrix between the diffusion folders of the lower-level $(t-1)$ in the hierarchy.

3. **Construction of the upper LDF level**
The rest of the process, which constructs the upper level of the current level, is similar to the construction of the bottom level of the hierarchy that was described in Sections 3.2(6) - 3.2(9). However, the input data for this process is the set of folders of points $\hat{S}^{t-1}$ (instead of set of points as in the bottom level) and the initial localized affinity for this process is the localized affinity matrix P^t between the diffusion folders in $\hat{S}^{t-1}$. Therefore, in this level, we cluster folders and not points. At the end of this construction of each upper level in the hierarchy, a higher-level partition $\hat{S}^t$ of the data into LDF is achieved. These LDF are the input for the next upper level $(t+1)$

in the hierarchy in the LDF. In addition, the final localized affinity matrix $\hat{P}^t$, which was constructed in this process, is the input affinity matrix for the next upper level $(t+1)$.

The process of constructing the upper levels in the hierarchy is repeated up to the root of the hierarchy.

4. Experimental Results

Our proposed methodology for hierarchical clustering was tested on several datasets that belong to different domains:

- Network protocols dataset - clustering and classification of network packets (see Section 4.1);
- Wine dataset - wine recognition and classification (see Section 4.2);
- Iris dataset - clustering of iris plants (see Section 4.3);
- Image processing - denoising and restoration of images (see Section 4.4).

4.1 The network protocols dataset

It is a proprietary non-public dataset. This dataset contains records of network activities of different applications (for example, Skype, Google Talk, ICQ, Windows Messenger, Microsoft Outlook, Mozila Thunderbird, Limewire, Gnucelus, eMule, eDonkey2000, BitLord, uTorrent, etc.). Each record contains the numerical statistics of a single network activity (for example, a single chat) of a single application. These statistics include 30 parameters for each record. These features may include the duration of the activity, the number of bytes the client sent to the server and received from the server, the upload and download bit rate, the number of data packets and control packets, etc. The dataset, which we used for the experimental evaluation, contains 5,500 records where each record belongs to one of 17 applications. Our goal was to hierarchically cluster the records according to their application, protocols and meta-protocols. In the lower levels of the hierarchy, the records should be clustered according to their applications and meta-applications (for example, Windows Messenger and Windows Live Messenger). In the upper levels of the hierarchy, the records should be clustered according to their protocols (for example, chat, Voip, etc.) and their meta-protocols (for example, symmetric and asymmetric communication). Figure 4.1 presents a diagram of a possible hierarchy of such dataset. In order to evaluate the quality of the clustering produced by our proposed method, we compared its performance to the performance of several algorithms and methods. We measured the quality and the accuracy of the clustering algorithms as follows: Let $X = \{x_1, \ldots, x_m\}$ be a set of n-dimensional points in $\mathbb{R}^n$ where each point $x_i \in X$ is described by n variables $x_i = \left\{hx_i^1, \ldots, x_i^n\right\}$. Let $L = \{l_1, \ldots, l_q\}$ be a set of different classes. For each n-dimensional point $x_i \in X$, the corresponding label y_i, $y_i = l_j, 1 \leq j \leq q$ is assigned. Therefore, $Y = \{y_1, \ldots, y_m\}$ is a set of labels for the dataset X. Y is used only for measuring the quality of the clustering algorithms and not for the clustering process itself.

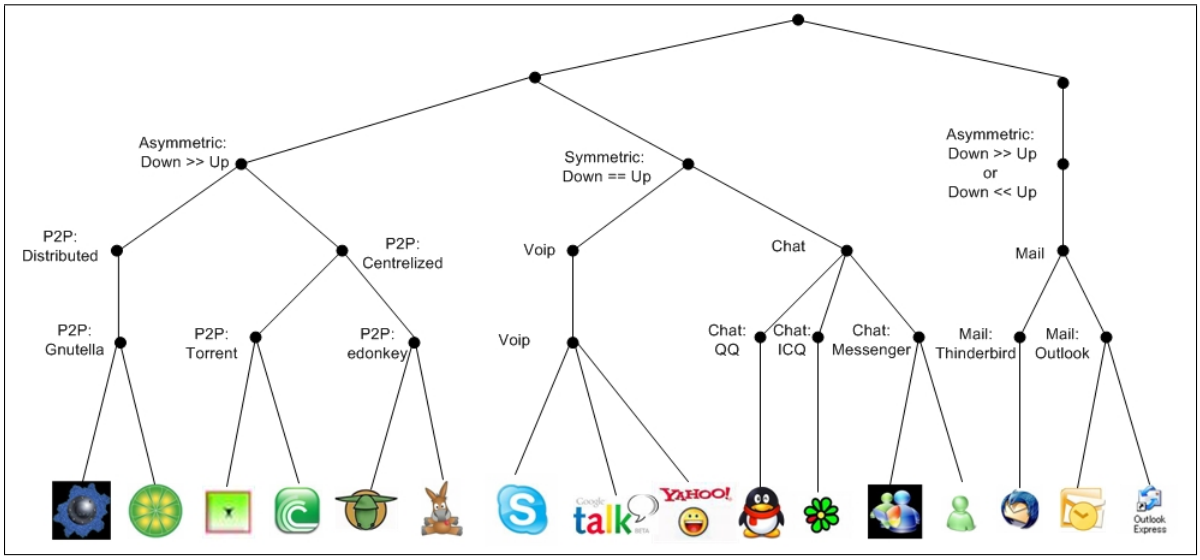


Figure 4.1: A possible hierarchy of network applications

Let f be a clustering algorithm to be evaluated. Let k be the number of clusters that f generates. Then, f_k is a clustering algorithm that associates each $x_i \in X$, $i = 1, \ldots, m$, to one of the clusters $c_r \in C$, $r = 1, \ldots, k$, where k is the number of clusters in C. For each $c_r \in C$, c_r is labeled according to the majority of the records in the cluster. Formally, let $B^i_{c_r} = \sum_{x_p \in c_r} \delta(x_p, l_i)$, where

$$\delta(x_p, l_i) = \begin{cases} 1 & \text{if } y_p = l_i \\ 0 & \text{otherwise} \end{cases}$$

and $1 \leq i \leq q, 1 \leq r \leq k$. Then, the label of each cluster $c_r, r = 1, \ldots, k$, is denoted by $M_{c_r} \triangleq \{l_i : \max_{1 \leq i \leq q} B^i_{c_r}\}$.

In order to evaluate the quality of the clustering algorithms, we measure the number of records in each cluster whose labels is equal to the label of the majority of the records in the cluster where the majority of the records is at least $P\%$ from the total number of records in the cluster. This measure determines the purity of the cluster (the intra-cluster accuracy). This $P-$purity accuracy is defined by:

$OP^P_k = \frac{\sum_{r=1}^{k} M_{c_r}}{k} = \frac{\sum_{r=1}^{k} \max_{1 \leq i \leq q} B^{iP}_{c_r}}{k}$, where k is the total number of clusters and $B^{iP}_{c_r}$ is defined as: $B^{iP}_{c_r} = \begin{cases} 1 & \text{if } \frac{B^i_{c_r} \cdot 100}{|c_r|} > P \\ 0 & \text{otherwise} \end{cases}$

In figures 4.2-4.4 , the X-axis represents the different $P-$purity levels and the Y-axis represents the percentages of the clusters that gained these levels.

First, we analyze the affect of t on the quality of the clustering. Figure 4.2 shows the comparison between the $P-$purity results from the LDF algorithm when $t = 1, 2, 3$ (first, second and third time steps which is also first, second and third levels in the hierarchy of the folders, respectively). As t increases, the accuracy improves accordingly. The reason is that at the bottom of the hierarchy ($t = 1$), we still have some clustering errors and as a result the accuracy of some folders is relatively low. However, as t increases and folders

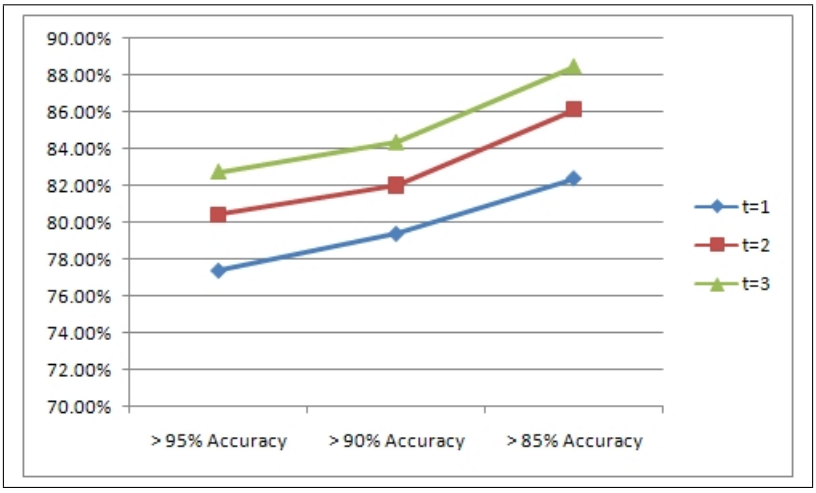


Figure 4.2: Comparison between the intra-cluster accuracy results for different t values

are merged by our method, then these clustering errors become negligible and the overall accuracy improves. This is the expected behavior of the LDF methodology.

Since our method aims to improve the clustering via the diffusion maps method, we compare between the results from the LDF algorithm and the diffusion maps. We applied the diffusion maps to the data and we clustered the points in the embedding. In order to evaluate the diffusion maps using kernels that were constructed according to different distance metrics, we used the Euclidean (L_2), Mahalanobis, Cosine and Weighted Euclidean (WL_2) distances. Figure 4.3 shows the comparison between the $P-$purity results from the LDF algorithm and the diffusion maps (using different distance metrics). We see that for

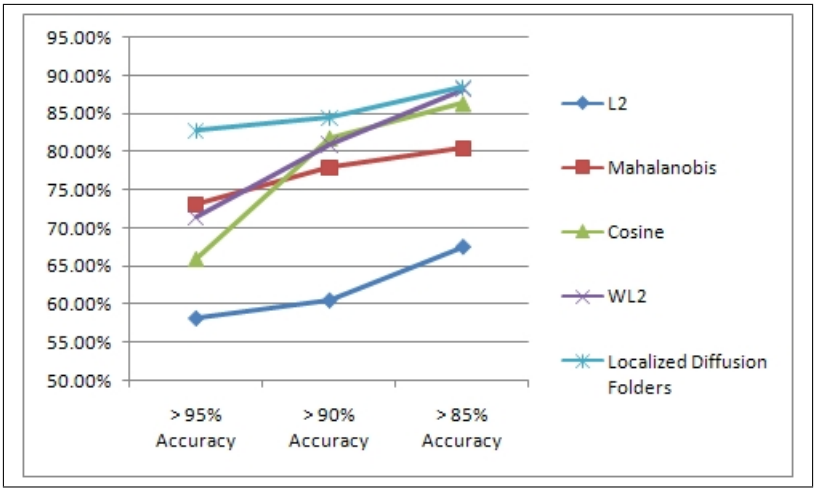


Figure 4.3: Comparison between intra-cluster accuracy results from the diffusion maps and from the LDF algorithms

95%-Purity, the LDF algorithm is more accurate than the best diffusion maps results by

about 15%. For 90%-Purity, the LDF algorithm is more accurate by about 5% and for 85%-Purity we get about the same results. This means that the LDF algorithm generates more accurate and purer clusters.

Last, we compared the accuracy of the LDF algorithm to the accuracy of k-means, BIRCH, and CURE clustering algorithms. Figure 4.4 shows the comparison between the $P-$purity results from the LDF algorithm and the different clustering algorithms.

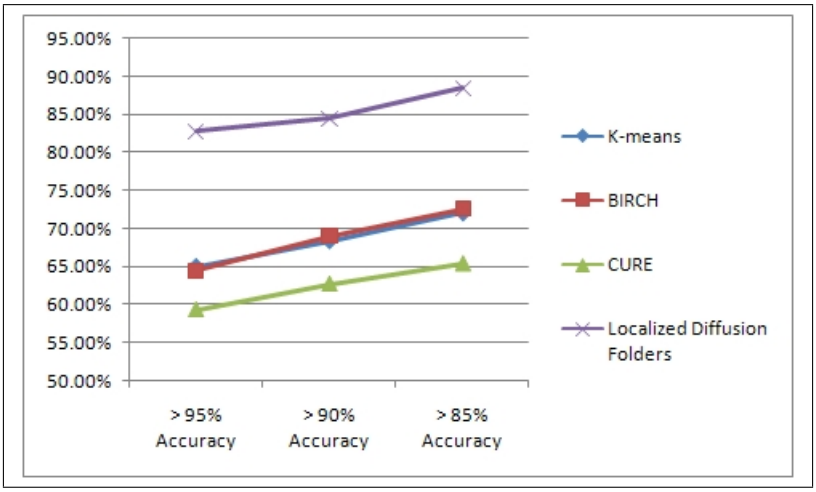


Figure 4.4: Comparison between the intra-cluster accuracy results from k-means, BIRCH, CURE and the LDF algorithms

We can see that the LDF algorithm outperforms the other clustering algorithms while achieving the best results for different $P-$purity levels.

4.2 The wine dataset (Blake and Merz , 1998)

It is a result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. For example, Alcohol, Malic acid, Magnesium, Hue and Proline. The dataset contains 178 wines that belong to three classes (the three types of wines): 59 wines belong to class 1, 71 wines belong to class 2 and 48 wines belong to class 3. In this experiment, we constructed the affinity matrix between wines according to the Euclidean distance of the log value of each wine. Then we constructed the hierarchical LDF accordingly. In the bottom level of the hierarchy ($t = 1$) we had 22 folders. In the second level ($t = 2$) we had 10 folders. In the third level ($t = 3$) we had 7 folders and in the fourth level ($t = 4$) we had 5 folders.

We measured the overall accuracy in each level as follows: each folder was labeled according to the majority of the points that have the same label. The overall accuracy is the ratio between the total number of points that have the same label as the majority (in their folder) and the total number of points in the dataset.

We compared the overall accuracy of the LDF algorithm to the accuracy of k-means and BIRCH clustering algorithms. The overall accuracy of each algorithm was evaluated for the

different number of clusters (22, 10, 7 and 5). Figure 4.5 shows the comparison results. The X-axis in this figure represents different number of clusters and the Y-axis represents the overall accuracy. We see that for 22 clusters ($t = 1$ in the LDF algorithm), the LDF

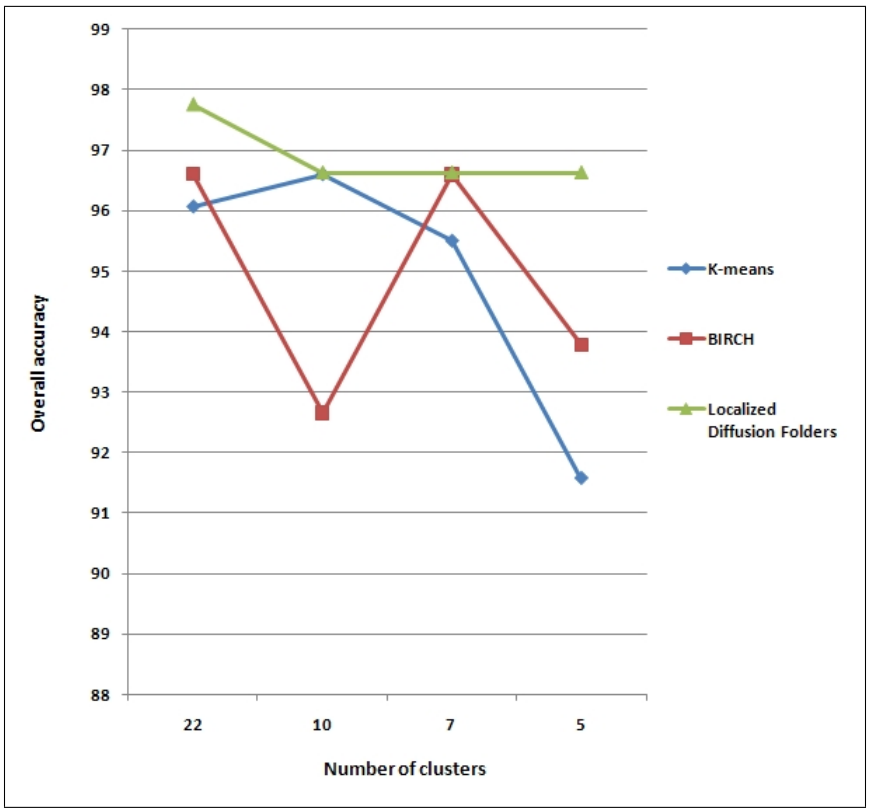


Figure 4.5: Comparison between the overall accuracy results from k-means, BIRCH and the LDF algorithms

algorithm is more accurate than BIRCH by 1.1% and from k-means by 1.8%. For 10 clusters ($t = 2$ in the LDF algorithm), the LDF algorithm is more accurate than BIRCH and as accurate as k-means. For 7 clusters ($t = 3$ in the LDF algorithm), the LDF algorithm is more accurate than k-means and as accurate as BIRCH. For 5 clusters ($t = 4$ in the LDF algorithm), the LDF algorithm is more accurate than BIRCH by 3.1% and from k-means by 5.7%. For this dataset, the overall accuracy of the LDF algorithm was better than the compared algorithms.

4.3 The iris dataset (Fisher , 1936)

This is perhaps the best known dataset to be found in the pattern recognition literature. It contains information about three types of iris plants. The plants are described by four variables (sepal, petal length and width). The dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant (Setosa, Versicolour, and Virginica). One

class is linearly separable from the other two. The latter are not linearly separable from each other.

We added Gaussian noise to the original iris dataset in order to determine its affect on different clustering algorithms. We measured the overall accuracy as described in 4.2. We compared the accuracy of the LDF algorithm to the accuracy of k-means, CURE and BIRCH clustering algorithms. Table 1 shows the comparison results. For each clustering algorithm, we measured the worst overall accuracy and the best overall accuracy.

Algorithm	Worst Accuracy	Best Accuracy
LDF	84.2105	90.1316
BIRCH	66.2252	89.35
CURE	60.7368	79.7368
k-means	65.7895	89.4040

Table 1: Comparison between the overall accuracy results from k-Means, CURE, BIRCH and the LDF algorithms

For the worst case, we see that the LDF algorithm is more accurate than BIRCH by 21.35%, from CURE by 27.87% and from k-means by 21.87%. In this case, the BIRCH, CURE and k-means algorithms failed clustering the noisy dataset. For the best case, we see that the LDF algorithm is more accurate than BIRCH by 0.87%, from CURE by 11.53% and from k-means by 0.81%. For this noisy dataset, the overall accuracy of the LDF algorithm was better than the compared algorithms.

4.4 Denoising and restoration of images

These are two important domains in image processing. We used the LDF algorithm for denoising and restoration as follows: first, we represented each pixel in the image by a window of 5×5 neighbors around it. This way, each pixel is transformed into a 25-dimensional vector (mega-pixel). Then, we moved with a sliding window of 9×9 mega-pixels over the image in order to determine the value of the center pixel of each sliding window. For each sliding window, we applied the following process: first, we constructed the hierarchical LDF according to the 9×9 neighbors around it. This way, each such window of 81 pixels was clustered into folders of pixels, super-folders (meta-pixels), super-super-folders (meta-meta-pixels), etc,. Last, we replaced the value of the center pixel (in the 9×9 window) with the average value of the pixels in the largest meta-pixel in the third level of the hierarchy.

A method for denoising using diffusion processes on graphs was described in Szlam, Maggioni and Coifman (2006). Since our LDF method is most related to this diffusion regularization method, we compare between the performance of both methods. Figure 4.6 shows the results. The top-left image is the original image. The top-middle image is the denoised image. We used a salt & pepper noise where 30% of the original pixels were replaced by salt (white) or pepper (black). The top-right image is the result of the LDF denoising algorithm (as described above). The bottom images are the results of the diffusion regularization algorithm using different σ (scale controls) values while constructing the Gaussian kernel (as described in Szlam, Maggioni and Coifman (2006)).

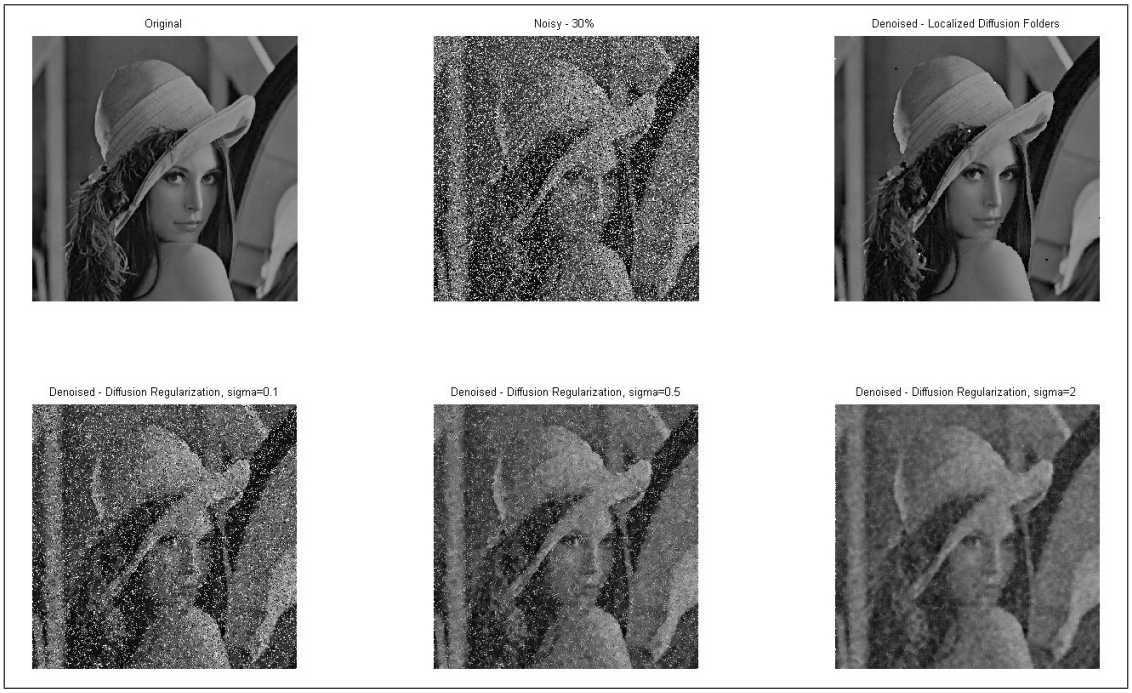


Figure 4.6: Comparison between the denoising results of the LDF and the diffusion regularization algorithms

As we can see, the LDF algorithm achieves the best denoising results.

Figure 4.7 shows another image processing result. This result relates to the image restoration problem. The left image is the original image. The middle image is the damaged image. The right image is the result of the LDF denoising algorithm (as described above).

Figure 4.7: Image restoration results

As we can see, the LDF algorithm restored the damaged image successfully.

5. Conclusions

In this paper, we presented a method for hierarchical clustering via LDF. The result of this clustering method is a bottom-up hierarchical clustering of the data while each level in the hierarchy contains LDF of folders from the lower levels. This methodology preserves the local neighborhood of each point while eliminating noisy connections between distinct points and areas in the graph. The performance of the algorithms was demonstrated on real data and it was compared to state-of-the art methods. In most cases, our proposed method outperformed the compared methods. In some cases, when we added noise to the dataset, all the compared methods failed clustering the data while the LDF succeeded.

References

F. R. K. Chung. Spectral Graph Theory. *AMS Regional Conference Series in Mathematics*, 92, 1997.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5-30, 2006.

R. R. Coifman and S. Lafon. Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31-52, 2006.

G. David, A. Averbuch and D. Lazarov. Smart-Sample: An Efficient Algorithm for Clustering Large High-Dimensional Datasets. submitted (2009).

A. Gionis, P. Indyk and R. Motwani. Similarity Search in High Dimensions via Hashing. *In Proceedings of the 25th VLDB Conference*, pages 518-528, 1999.

G. Gan, C. Ma and J. Wu. *Data Clustering: Theory, Algorithms, and Applications.* SIAM, 2007.

T. Zhang, R. Ramakrishnan and M. Livny. BIRCH: An efficient data clustering method for very large databases. *SIGMOD*, 25(2):103-114, 1996.

M. Ester, H. P. Kriegel, J. Sander and X. Xu. A density-based algorithm for discovering clusters in large spatial database with noise. *In International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96)*, pages 226–231, Portland, Oregon, AAAI Press, 1996.

S. Guha, R. Rastogi and K. Shim. CURE: An Efficient Clustering Algorithms for Large Databases. *ACM SIGMOD International Conference on Management of Data*, pages 73–84, Seattle, WA, 1998.

L. Kaufman and P. Rousseeuw. *Finding groups in Data - An introduction to Cluster Analysis.* Wiley, New York: John Wiely & Sons, Inc., 1990.

J. Macqueen. Some methods for classification and analysis of multivariate observations. *In Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, Berkeley, CA: University of California Press, 1967.

V. Ganti, J. Gehrke and R. Ramakrishnan. CACTUS: Clustering Categorical Data using Summaries. *In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 73–83, San Diego, CA, USA, ACM Press, August 1999.

Z. Huang. Extensions to the k-means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, 2(3):283-304, 1998.

A. Ben-Hur, D. Horn, H. Siegelmann and V. Vapnik. Support vector clustering. *Machine Learning Res.*, 2:125-137, 2001.

M. Girolami. Mercer Kernel Based Clustering in Feature Space. *IEEE Transactions on Neural Networks*, 13(4):780-784, 2002.

R. Zhang and A. Rudnicky. A Large Scale Clustering Scheme for Kernel k-means. *In Proceedings of the 16th International Conference on Pattern Recognition (ICPR 02)*, pages 289–292, Quebec City, Canada, August 2002.

J. Couto. Kernel k-means for Categorical Data . *Advances in Intelligent Data Analysis* VI(3646):46-56, 2005.

H. Koga, T. Ishibashi and T. Watanabe. Fast Hierarchical Clustering Algorithm Using Locality-Sensitive Hashing. *Knowledge and Information Systems*, 12:25-53, 2007.

F. Aurenhammer. Voronoi Diagrams A Survey Of A Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23:345-405, 1991.

C. L. Blake and C. J. Merz. UCI Repository of Machine Learning Databases. University of California, Dept. Information and Computer Science, Irvine, CA, USA, 1998. http://www.ics.uci.edu/ mlearn/MLRepository.html.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(2):179-188, 1936.

A. Szlam, M. Maggioni and R. Coifman. A general framework for adaptive regularization based on diffusion processes on graphs. Technical Report YALE/DCS/TR1365, Yale University, July 2006.

M. Frechet. Sur quelques points du calcul fonctionnel. *Rendiconti de Circolo Matematico di Palermo*, 22:1-74, 1906.

I. Kaplansky. *Set Theory and Metric Spaces*. AMS Chelsea Publishing, 1977.

G. David. *Anomaly Detection and Classification via Diffusion Processes in Hyper-Networks*. Phd Thesis, Tel-Aviv University, March 2009.