Hierarchical Clustering Via Localized Diffusion Folders

Gil David
Department of Mathematics
Program in Applied Mathematics
Yale University, USA

Amir Averbuch
School of Computer Science
Tel-Aviv University, Israel

Ronald R. Coifman
Department of Mathematics
Program in Applied Mathematics
Yale University, USA

Abstract

We present a short introduction to an hierarchical clustering method of high-dimensional data via localized diffusion folders.

Introduction

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 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 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.

Diffusion Maps

The diffusion maps framework (Coifman and Lafon 2006a; 2006b) 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 \( \mu \) is the distribution of the points on \( \Gamma \). We construct the graph

\[
G(V, E), |V| = m, |E| \ll m^2 \text{ on } \Gamma \text{ in order to study the intrinsic geometry of this set. A symmetric, non-negative and positive semi-definite weight function } W_e \triangleq w_e(x_i, x_j) \text{ is introduced. It measures the pairwise similarity between the points. A common choice for } W_e \text{ is } w_e(x_i, x_j) = e^{-\|x_i-x_j\|^2}. \text{ The non-negativity property of } W_e \text{ allows to normalize it into a Markov transition matrix } P \text{ where the states of the corresponding Markov process are the data points. This enables to analyze } \Gamma \text{ as a random walk. The construction of } P \text{ is known as the } \text{normalized graph Laplacian} \text{ (Chung 1997).}
\]

Formally, \( P = \{p(x_i, x_j)\}_{i,j=1,...,m} \) is constructed as

\[
p(x_i, x_j) = \frac{w_e(x_i, x_j)}{\sum_i w_e(x_i, x_j)} \text{ where } d(x_i) = \int_{\Gamma} w_e(x_i, x_j) d\mu(x_j) \text{ is the degree of } x_j. \text{ Thus, } p(x_i, x_j) \text{ can be viewed as the probability to move from } x_i \text{ to } x_j \text{ in one time step. By raising this quantity to a power } t \text{ (advance in time), this influence is propagated to nodes in the neighborhood of } x_i \text{ and } x_j \text{ and the result is the probability for this move in } t \text{ time steps. We denote this probability by } p_t(x_i, x_j).
\]

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 \( \epsilon \). Construction of \( p_t(x_i, x_j) \) is obtained from \( p_t(x_i, x_j) = \sum_{k \geq 0} \lambda_k \nu_k(x_i) \nu_k(x_j) \). A similar eigen-decomposition is obtained from \( p_t(x_i, x_j) = \sum_{k \geq 0} \lambda_k^t \nu_k(x_i) \nu_k(x_j) \) after advancing \( t \) times on the graph. Here \( \lambda_k \) and \( \nu_k \) are the eigenvalues and the corresponding eigenfunctions of the laplacian.

The map \( \Phi_m : \Gamma \rightarrow \mathbb{R}^l \) embeds the dataset into an Euclidean space. The diffusion distance is defined as

\[
D^t_i(x_i, x_j) = \sum_{k \geq 0} (\tilde{p}_t(x_i, x_k) - \tilde{p}_t(x_k, x_j))^2. \text{ This formulation is derived from the known random walk distance in Potential Theory. It is shown that the diffusion distance can be expressed in terms of the right eigenvectors of the graph Laplacian.}
\]

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
tors of $P$: $D_t^2(x_i, x_j) = \sum_{k \geq 0} \lambda_k^{(t)} (\nu_k(x_i) - \nu_k(x_j))^2$.

This facilitates the embedding of the original points in an Euclidean space $\mathbb{R}^n$, by: $\Xi_t : x_i \rightarrow (\lambda_0^{(t)}(x_i), \lambda_1^{(t)}(x_i), \lambda_2^{(t)}(x_i), \ldots, \lambda_{n}^{(t)}(x_i))$.

**Localized Diffusion Folders**

As described in the brief overview of the diffusion maps, $P$ 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 a short introduction to 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.

**Experimental Results**

**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 (Macqueen 1967) and BIRCH (T. Zhang 1996) clustering algorithms. The overall accuracy of each algorithm was evaluated for the different number of clusters (22, 10, 7 and 5). Figure 1 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 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.

### 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 and compared the accuracy of the LDF algorithm to the accuracy of K-means, CURE (S. Guha 1998) 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.

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.

### 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 $5 \times 5$ neighbors around it. This way, each pixel is transformed into a 25-dimensional vector (mega-pixel). Then, we moved with a sliding window of $9 \times 9$ 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 $9 \times 9$ 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 $9 \times 9$ 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 (A. Szlam 2006). Since our LDF method is most related to this diffusion regularization method, we compare between the performance of both methods. Figure 2 shows the results. The top-left image is the original image. The middle-left 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 bottom-left image is the result of the LDF denoising algorithm (as described above). The right images are the results of the diffusion regularization algorithm using different $\sigma$ (scale controls) values while constructing the Gaussian kernel (as described in (A. Szlam 2006)).

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst Accuracy</th>
<th>Best Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDF</td>
<td>84.2105</td>
<td>90.1316</td>
</tr>
<tr>
<td>BIRCH</td>
<td>66.2252</td>
<td>89.35</td>
</tr>
<tr>
<td>CURE</td>
<td>60.7368</td>
<td>79.7368</td>
</tr>
<tr>
<td>K-means</td>
<td>65.7895</td>
<td>89.4040</td>
</tr>
</tbody>
</table>

Table 1: Comparison between the overall accuracy results from K-Means, CURE, BIRCH and the LDF algorithms
Figure 2: Comparison between the denoising results of the LDF and the diffusion regularization algorithms

References

A. Szlam, M. Maggioni, R. C. 2006. A general framework for adaptive regularization based on diffusion processes on graphs. Technical Report YALE/DCS/TR1365, Yale University, Yale University, New Haven, CT, USA.


