DAP: Robust Spectral Clustering via Iterated Diffusion Reweighting

Carlton Downey

December 8, 2016

Abstract

BACKGROUND: Clustering is one of the fundamental problems in machine learning and is widely used in a variety of scientific disciplines. Spectral clustering is a popular modern clustering algorithm which uses the eigendecomposition of a similarity matrix to find an embedding of the data, then clusters the resulting embedding. Unfortunately the poor conditioning of the eigendecomposition operation causes the performance of spectral clustering to suffer in the presence of noise.

AIM: In this paper we aim to develop a noise-robust spectral clustering algorithm.

METHOD: We build on the work of Coifman et al. We show that the notion of diffusion can be used to identify and reduce the weight of noisy edges (entries) in the similarity matrix. We propose a new iterative algorithm consisting of three key steps: A diffusion step, a thresholding step, and a normalization step. By iterating these 3 steps we drive noisy edges to zero, denoising the adjacency matrix. We call this algorithm Iterative Diffusion Reweighting (IDR).

RESULTS: We apply IDR to 2 datasets: The Utah teapot dataset and a geotagged Twitter dataset. Using the Utah teapot we show that IDR can recover a planted manifold in the presence of noise where alternative approaches cannot. Specifically IDR can handle almost twice as much noise as alternative approaches (Gaussian noise with standard deviation of 0.1 vs 0.2). Using the geotagged Twitter dataset we show that IDM results in a superior clustering. Qualitatively the clusters appear to be more informative when visualized, and quantitatively the clusters correspond superior tweet prediction accuracy (0.32 vs 0.30).

CONCLUSION: We show that an iterative diffusion based algorithm can be used to improve the performance of spectral clustering in the presence of noise by denoising the adjacency matrix.

1 Introduction

Clustering is one of the fundamental problems in machine learning. In clustering we attempt to partition a data set into groups of objects called clusters, such that objects within a cluster are similar, while objects in different clusters are dissimilar.

Clustering is a key problem in a wide variety of scientific disciplines: in biology clustering is used to study proteins by identifying groups of genes with similar expression patterns [Vesth et al., 2016]; In medicine clustering is used to analyze patterns of antibiotic resistance [Donia et al., 2014]; In sociology clustering is used to identify patterns of criminal behavior in former foster youth [McMahon and Fields, 2015]; and in computer vision clustering is used to detect edges and objects in pictures [Dinh et al., 2009]. These are only a few examples of the many problems practitioners use clustering algorithms to solve.

In recent years, spectral clustering [Luxburg, 2007, Ng et al., 2002, Spielman and Teng, 1996] has become one of the most popular modern clustering algorithms. It is simple to implement, can be solved efficiently by standard linear algebra software, and very often outperforms traditional clustering algorithms such as the k-means algorithm.

Spectral clustering applies traditional clustering techniques (such as k-means) to a low dimensional, non-linear embedding of the data. This embedding is obtained by analyzing the spectrum, or eigendecomposition, of a matrix representation of the data. The goal of this embedding is to retain the meaningful information present in the data, while removing the non-meaningful information.

Unfortunately this embedding is highly susceptible to noise [Zhu et al., 2014, Balakrishnan et al., 2011] due to the poor conditioning of the eigendecomposition operator. This is a result of adjacency matrix for most naturally occurring datasets possessing a small eigengap. Poor conditioning means that a small noise perturbation to the data set can drastically change the resulting embedding, in turn resulting in a drastic (and usually negative) change to the resulting clustering. This significantly limits the effectiveness of spectral clustering on noisy real world data sets.

A variety of techniques have been proposed to help alleviate this problem, the most popular of which is known as Diffusion Maps [Coifman and Lafon, 2006]. Diffusion Maps uses the theory of random walks to help de-noise the data, resulting in improved spectral embeddings, and hence a more noise-robust spectral clustering algorithm.

While Diffusion Maps outperforms vanilla spectral clustering in many settings, we believe there is still significant room for improvement.

In this document we introduce a new spectral clustering algorithm which offers improved robustness to noise over existing techniques. This algorithm is based on an extension of the random walk analysis made popular in the diffusion maps algo-

rithm. We show experimentally that this new algorithm significantly outperforms existing techniques on two data sets.

2 Background

2.1 Spectral Clustering

Spectral clustering is a popular modern clustering algorithm based on the concept of manifold embeddings. Spectral clustering algorithms consist of three high level steps:

(1) Form a similarity matrix based on the data (2) Find a low dimensional embedding of the data via the eigendecomposition of this similarity matrix (3) Determine a good clustering using this embedding

Spectral clustering has several advantages over conventional clustering techniques such as k-means: In many settings spectral clustering is guaranteed to converge to the global optimum, and spectral clustering makes very few assumptions about the shape of the clusters.

We now present the spectral clustering algorithm in detail. Let $X = x_1, ..., x_n \subset \mathbb{R}^n$ be a dataset. The first step in spectral clustering is to select a similarity function $S(x, y) : X \times X \to \mathbb{R}$ which intuitively measures the similarity between pairs of points. Using S we construct a similarity matrix W such that $W_{ij} = S(x_i, x_j)$, i.e., W consists of all n^2 pairwise similarities. We normalize the rows of W to produce $W_{\text{rw}} = D^{-1}W$ where D is the diagonal matrix of row sums. We take the eigendecomposition, $U\Sigma U^T = W_{\text{rw}}$ and use the first few eigenvectors as our embedding coordinates. Finally we apply a conventional clustering algorithm, such as k-means, to our embedding to obtain our clusters.

2.2 Diffusion Maps

There have been several attempts to produce spectral clustering algorithms which are robust to noise. One of the most successful is called the diffusion maps algorithm.

Diffusion maps is based on the simple, yet powerful insight that we can use random walks to measure the "manifold distance" between two points. The idea is that we want to take our original similarities and replace them with new similarities based on the manifold distance between two points. Manifold distances provide a more robust measure of the similarity between points, because they reflect the relative connectedness of the points, taking into account other nearby points.

The diffusion maps algorithm is deceptively simple: We replace the similarity matrix W_{rw} with the transformed similarity matrix W_{rw}^k where $k \in \mathbb{N}$. We take the

eigendecomposition $U\Sigma^k U^T = W_{\text{rw}}^k$ and use the first few columns of $U\sigma^k$ as our embedding. Behind this simple transformation is a beautiful abstraction, where we can use structural information about the local topology of the manifold to reweight edges and help remove noise.

Diffusion maps can obtain good results on noisy data sets where vanilla spectral clustering fails to do so. However there are still datasets which contain a modest amount of noise where diffusion maps does not perform well. We aim to build on the diffusion maps framework to further improve the noise-robustness of spectral clustering.

3 Related Work

Spectral clustering dates back to the work of [Donath and Hoffman, 1973] and [Fiedler, 1973] who in the same year both proposed partitioning graphs based on the spectrum of the adjacency matrix. In the following years this approach was independently rediscovered in a wide variety of fields, see [Spielman and Teng, 1996] for a nice overview of this history and [Luxburg, 2007] for an excellent and practical tutorial on the method itself.

Since then there has been significant work on improving the performance of spectral clustering on noisy data sets. The majority of these approaches can be categorized into two categories: (1) methods which attempt to create robust affinity matrices based on the original data [Zelnik-manor and Perona, 2004, Pavan and Pelillo, 2007, Premachandran and Kakarala, 2013, Wang et al., 2008, Zhu et al., 2014] and (2) approaches which attempt to improve the quality of the clustering based on a fixed (sparse) affinity matrix with no access to the original data [Shi and Malik, 2000, Ng et al., 2002, Xiang and Gong, 2008].

Within category (1) one popular approach is to use an adaptive scaling parameter which controls the number of neighbors at each point when learning the nearest neighbor graph [Zelnik-manor and Perona, 2004, Wang et al., 2008]. This approach aims to mitigate the problem of regions with different scales existing within a single data set. Unfortunately this approach has little effect on outliers.

[Pavan and Pelillo, 2007] and [Premachandran and Kakarala, 2013] attempt to solve the problem of outliers by using graph properties to remove outliers from the affinity matrix. [Pavan and Pelillo, 2007] uses an approach based on cliques, while [Premachandran and Kakarala, 2013] uses an approach based on k nearest neighborhood evaluations at different scales. [Zhu et al., 2014] suggest an approach where they learn a robust similarity metric based on non-euclidean distance and feature selection with the goal of producing an improved affinity matrix

Within category (2) [Shi and Malik, 2000] propose a hierarchical spectral clustering algorithm which recursively subdivides the dataset using spectral clustering

at each step. This approach allows them to solve a more simple binary classification problem at each step. [Xiang and Gong, 2008] note that some eigenvectors are more useful for clustering than others, and propose an approach based on selecting an optimal set of eigenvectors.

4 Method

In this section we describe a new, noise-robust spectral clustering algorithm for manifold data.

One important property of any good clustering algorithm is being robust to noise: Even if significant noise is added to the data we can expect that it does not change the output of the clustering algorithm.

Spectral clustering is a powerful and flexible technique; however due its dependence on the eigendecomposition of a matrix it is inherently susceptible to noise. One approach to improving the robustness of spectral clustering is to add an additional step which de-noises the adjacency matrix prior to calculating the embedding. This is the approach taken in diffusion maps, and the approach we will also take.

We assumed all data points lie on a low dimensional manifold embedded in the high dimensional space. We assume edges in our graph fall into two categories: intra-manifold edges and between-manifold edges. Intra-manifold, or data edges, are edges between two points which are close together on the manifold, while between-manifold, or noise edges, are edges between two points which are far apart on the manifold. These edges, which short-circuit the manifold, are the primary cause of low-quality embeddings.

We want to identify and remove these between-manifold edges from the graph. Unfortunately to determine which edges are between-manifold edges we first need to know the manifold, and if we knew the manifold we would have already solved the embedding problem. Therefore we turn instead to an alternative characterization of these edges which does not require knowledge of the entire manifold. We achieve this via the random walk interpretation of spectral clustering.

Let W be a row normalized similarity matrix, i.e., $\sum_j W_{ij} = 1 \quad \forall i$. Let G be the graph corresponding to this matrix, which has one node for each row/column, and one weighted edge for each non-zero entry. We can view W as the transition matrix of a Markov chain acting on G. Under this view each $W_{ij} = p(i, j)$ is the conditional probability of moving to vertex in j in one step given we started in vertex i. Furthermore if W is the 1-step transition probabilities, then W^n consists of the n-step transition probabilities. In other words W^n_{ij} is the probability of moving to vertex j in n steps given we started in vertex i.

The n-step transition probability from i to j can also be calculated combina-

torially as the number of distinct length n paths from i to j divided by the total number of distinct length n paths starting at i. The first key insight behind this work is that if (i, j) is a between-manifold edge, then there are few paths of length n between i and j, while if (i, j) is an intra-manifold edge, then there are many paths of length n between i and j.

Consider a vertex and its edges. If (i, j) is an intra-manifold edge then it connects two distinct regions of the manifold, region A and region B. By assumption A and B are well separated on the manifold, therefore for short random walks we can view A and B as two distinct manifolds. This means that any length n path connecting two points in different regions must pass through one of the inter-cluster edges. We make the (reasonable) assumption that the number of between-manifold edges is small relative to the total number of edges. Hence the number of length n paths between two adjacent points in different regions is small, while the number of length n paths between two adjacent points in the same region is large. In other words $\forall i, j \in A$ and $\forall k \in B$ such that $W_{ij} > 0$ and $W_{ik} > 0$ then $p^n(i, j) > p^n(i, k)$. This discussion suggests that if we replace W with W^n in the spectral clustering algorithm, this will act to denoise the similarity matrix by decreasing the weight on inter-cluster edges, while leaving the weights on intra-manifold edges unchanged. This is exactly the diffusion maps algorithm.

The problem with replacing W with W^n is that W^n contains many more edges than W. This is due to the fact that it is possible to move between two nodes in n steps even if there is no edge directly connecting them. These edges are undesirable for both computational and statistical reasons. Diffusion map attempts to solve this by thresholding the entries of the resulting matrix, however this is problematic for a number of reasons. This is an additional parameter to tune, it increases the cost of the eigendecomposition operation, and most importantly a single data set may contain multiple different structures which require different thresholds.

We suggest that an alternative approach is to restrict the non-zero weights of W^n to the original non-zero weights of W, then re-normalize the rows of the resulting matrix to produce a new transition matrix. In other words we create a new matrix X such that $X = W^n_{ij}$ if $W_{ij} > 0$ and $X = 0$ otherwise. This edge re-weighting procedure is summarized in algorithm 1.

This allows us to reweight the edges of the graph according to the graph topology without introducing additional edges.

4.1 Iterated Diffusion Reweighting

By replacing W with the thresholded power matrix X we can de-weight noise edges and decrease their influence on the embedding, however we have not entirely removed them as desired. We now show how to construct an iterative algorithm using this procedure which can completely remove such noise edges.

Input: Normalized Similarity Matrix W
Random Walk Parameter n

$W_{\text{old}} \leftarrow W$
$W \leftarrow W^n$
$W(W_{\text{old}} = 0) \leftarrow 0$
normalize(W)
return W

Algorithm 1: Diffusion Reweighting

Unfortunately if we naively iterate the diffusion embedding procedure of the previous section we will not get the desired result. It will indeed drive the noise edges to have zero weight, however it will also drive many other edges to also have zero weight. The problem is that even if two edges which are adjacent with the same vertex are both intra-vertex edges, they will be assigned different values by the diffusion embedding procedure — in fact they will converge to a multiple of the stationary distribution of the matrix. In other words the weight assigned to edge (i, j) will be proportional to the degree of j, $d(j)$. This is not an issue if we only apply this procedure once, however if we iterate it will quickly get out of hand. Specifically in each iteration we will increase the degree of high degree vertices and decrease the degree of low degree vertices. As we continue to iterate, edges adjacent to low degree vertices will be driven to zero, disconnecting the graph and resulting in poor embeddings.

The issue that's arising here is that we are interested in the short term dynamics of the system, but W^n consists of a mixture of short term and long term dynamics.

We solve this problem by noting that the short term dynamics of the system are present in both the rows and the columns of W^n, however the long term dynamics of the system are only present in the *rows* of W^n. If we examine W^n we see that each row of the matrix is converging to the stationary distribution, with probability mass spreading out from the initial vertex following the topology of the graph. The ith row and the i column of W^n exhibit nearly symetrical behavior, because if $p^n(i, j)$ is large, then $p^n(j, i)$ is also going to be large. The key difference is that each column is converging to a multiple of the all ones vector, rather than a multiple of the stationary distribution. This means that the values in each column reflect the short term dynamics of the system, while each column is weighted according to the long term dynamics of the system.

Using this insight and our original projection algorithm we can now construct an iterative algorithm which completely removes noise edges; see Algorithm 2 for details. Essentially we repeatedly take an n-step random walk on the graph, take the transpose, remove all edges which were not present in the original graph, then

normalize to obtain a transition matrix. Raising the matrix to a power de-weights between-manifold edges. Taking the transpose of the matrix and normalizing remove the long term dynamics from the system so that at each step we reweight the edges based only on the short term dynamics of the system.

Input: Normalized Similarity Matrix W
Random Walk Parameter n
Power Parameter p
Iteration parameter k
for $iter = 1 : k$ **do**
  $W_{\text{old}} \leftarrow W$
  $W \leftarrow W^n$
  $W \leftarrow W^T$
  $W(W_{\text{old}} = 0) \leftarrow 0$
  normalize(W)
end
return W

Algorithm 2: Iterative Diffusion Reweighting (IDR)

5 Theory

We show the effectiveness of our algorithm in a highly simplified setting using a barbell graph. While this setting is highly unrealistic, and can easily be solved using vanilla spectral clustering, it nonetheless provides us with a great deal of intuition into why the algorithm works.

A barbell graph $G = (V, E)$ is a graph consisting of two cliques connected by a single edge or isthmus. Note that all edges in the graph initially have weight 1. Let $X \subset V$ and $Y \subset V$ be the two cliques present in this graph. Suppose $x \in X$ and $y \in Y$ are the two points connected by the isthmus, and $x' \in X$ is another point in X. Let $p(a, b)$ be the transition probability of moving from a to b in one step of a Markov chain acting on the graph, and $p^k(a, b)$ be the k step transition probability. If each clique is of size n then $p(x, x') = p(x, y) \approx \frac{1}{n}$. This implies that the 2 step transition probability between x and x' can be approximated as:

$$p^2(x, x') = \sum_{v \in V} p(x, v)p(v, x') \approx \sum_{v \in X} p(x, v)p(v, x') = \sum_{v \in X} \frac{1}{n}\frac{1}{n} = \frac{1}{n}$$

And that the 2 step transition probability between x and y can be approximated as:

$$p^2(x, y) = \sum_{v \in V} p(x, v)p(v, y) \approx \sum_{v \in X} p(x, v)p(v, y) = p(x, x)p(x, y) = \frac{1}{n^2}$$

A single iteration of our IDR algorithm replaces each transition probability with its n step transition probability. This implies that if we apply a single iteration of diffusion reweighting to this graph it will decrease the weight of the isthmus edge by a factor of $\frac{1}{n}$ while leaving all other edge weights unchanged. Therefore applying the IDR algorithm on the barbell graph with random walks of length 2 will achieve exponentially fast convergence of the isthmus edge weight to zero while leaving the other edges unchanged.

6 Experiments

6.1 Utah Teapot

We begin with a well known semi-synthetic embedding problem called the Utah Teapot Problem. This dataset consists of a sequence of images taken by a camera panning 360 degrees around a decorative teapot. Given the images, the goal is to recover the relationship between them — in other words reconstruct the original video sequence from the unordered set of images. In order to recover the correct ordering we embed the set of images into 2D space. In a good embedding the points are organised in a circle, with the position in the circle based on camera angle. Given this embedding it is clear that we can easily reconstruct the original video.

To increase the difficulty of this problem, we corrupt each pixel in the original images with random Gaussian noise.

The data set we use consists of 400 images, each of size 76x101 pixels. The Gaussian noise has mean zero and standard deviation 0.20, with the standard deviation selected to be sufficiently large to cause existing algorithms to fail. We calculate a binary mutual-$k$nn adjacency matrix W based on the Euclidean distance between images.

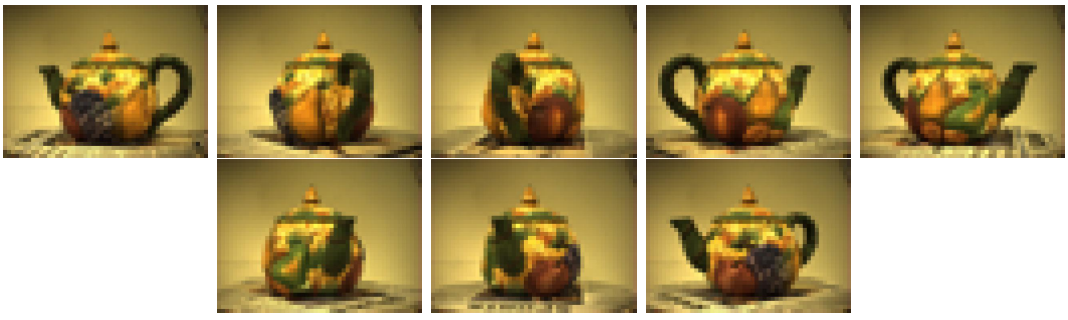


Figure 1: Example images from Utah Teapot dataset

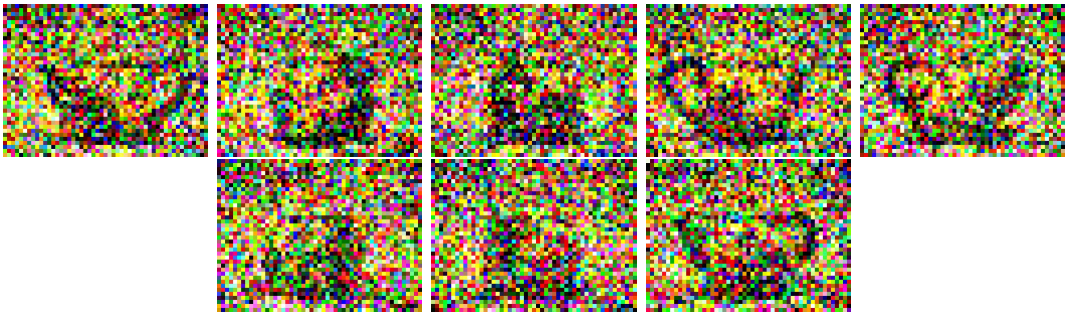
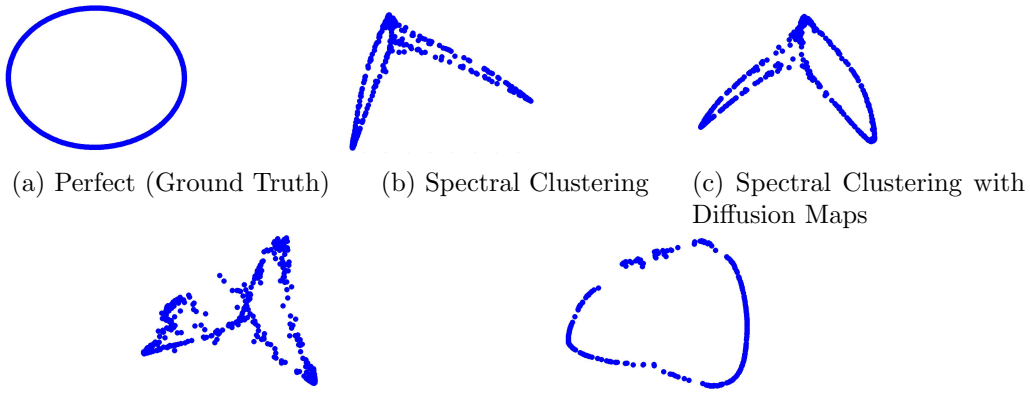


Figure 2: Example images from Utah Teapot dataset perturbed with random Gaussian noise

We apply 4 distinct spectral clustering algorithms to the featurized data: (1) Spectral Clustering, (2) Spectral Clustering with Diffusion Maps, (3) Spectral Clustering with Diffusion Maps and Thresholding, (4) Spectral Clustering with IDR. We use cross validation to tune the parameters for each of these methods.

6.1.1 Results



(a) Perfect (Ground Truth)

(b) Spectral Clustering

(c) Spectral Clustering with Diffusion Maps

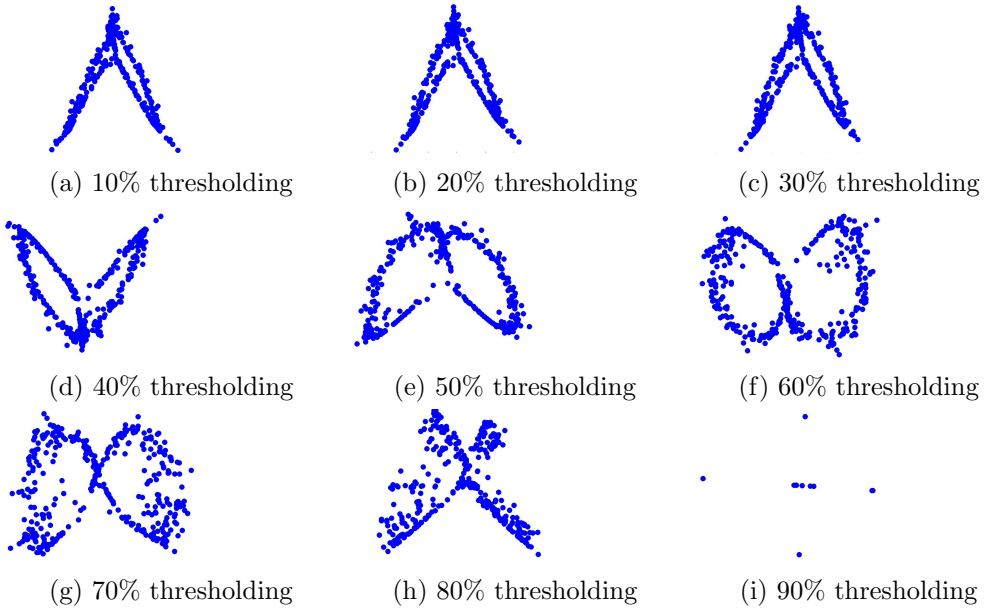
(d) Spectral Clustering with Diffusion Maps and Thresholding

(e) Spectral Clustering with IDR

Figure 3: 2D embeddings for the Utah Teapot dataset for 4 different embedding algorithms, together with a ground truth embedding

Figure 3 presents the results of applying each embedding algorithm to the Utah Teapot dataset described above. We see that IDR clearly outperforms the other 3 techniques — in fact it is the only technique which is able to recover the correct embedding. We found that Diffusion maps can correctly recover the planted manifold when the standard deviation of the noise is at most 0.1, whereas

we found that IDR can recover the planted manifold when the standard deviation of the noise is at most 0.2;



(a) 10% thresholding (b) 20% thresholding (c) 30% thresholding

(d) 40% thresholding (e) 50% thresholding (f) 60% thresholding

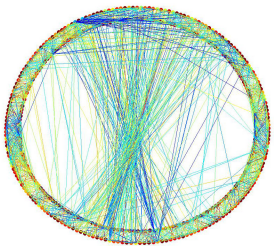
(g) 70% thresholding (h) 80% thresholding (i) 90% thresholding

Figure 4: Embeddings using Diffusion Maps with Thresholding for different levels of thresholding

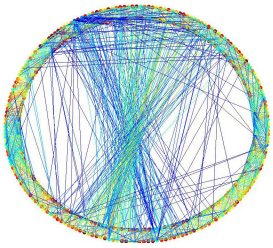
Figure 4 displays the embeddings resulting from the spectral clustering with diffusion maps and thresholding, for different levels of thresholding. We see that a moderate amount of thresholding improves the embedding; however, too much thresholding causes the embedding to collapse.

Diffusion maps, thresholding, and IDR all aim to improve the quality of the embedding by modifying the weights matrix. Figure 5 provides a visualization of the weights matrix resulting from each of these techniques applied to the Utah Teapot dataset. Each point represents an image, and two points x and y will be connected by an edge if x is a knn of y and y is a knn of x. The weight of the edge is indicated by its color: Red edges have high weight while blue edges have low weight. We have arranged the points based on their location in the ground truth embedding.

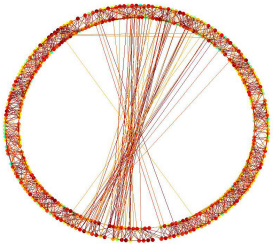
The goal of modifying the weights matrix is to preserve edges between points which are close together, while removing edges between points which are distant. In particular we can see a large number of noise edges between points on opposite sides of the circle in one particular location that we want to remove. Diffusion maps removes many of these noise edges, but many more are not removed. Thresholding the weights matrix resulting from diffusion maps removes even more noise edges,



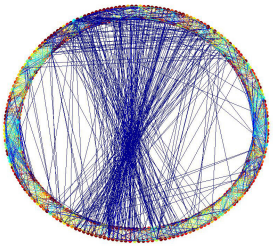
(a) Original Weights



(b) Weights after Diffusion Maps



(c) Weights after Diffusion Maps and optimal Thresholding



(d) Weights after IDR

Figure 5: Visualization of the Weights generated by each embedding algorithm. Color corresponds to edge weight: blue edges have small weights, red edges have large weights. Edges which cross the circle are noise edges and should be eliminated. Dark blue weights are zero to within numerical precision. In panel (d) all noise edges have been eliminated, while in figures (a)-(c) a significant number of noise edges remain.

but at the cost of removing many signal edges as well. Furthermore even after thresholding many noise edges still remain. IDR removes virtually all of the noise edges while preserving the signal edges.

6.2 NY-NJ Twitter Data

For our second experiment we work with a proprietary Twitter data set courtesy of Norman Sadeh's group (sadeh@cs.cmu.edu). This data set consists 6,099,005 tweets from 232,280 New York City (NY) and New Jersey (NJ) Twitter users collected over a 3 month period. Each tweet consists of a short message, a time stamp, and lat-long coordinates.

Our goal is to use this dataset to identify mobility patterns and indirectly generate abstractions of the types of activities in which different groups of people engage during the course of the day. We investigate different ways of organizing this data in both time and space, and evaluate the stability and predictive power of different clustering techniques. Specifically we will use clustering techniques to

partition the set of Twitter users into k distinct groups based on the time/location of their tweets. Users in the same group should have similar tweeting habits i.e they should tweet at similar locations at similar times; while users in different groups should have dissimilar tweeting habits. An example would be a group of users who tweet in a particular suburb of Brooklyn in the morning and downtown Manhattan at night. We will compare different clusterings qualitatively via visualization, and quantitatively by their predictive power, i.e. our ability to predict tweet time/location based on group membership.

We featurize each user using an estimate of their tweet probability at a set of randomly chosen points in time/space. We use only the time/location of the tweet, and do not use the text of the tweet in any way. We estimate the tweet probability using a kernel density estimate: if $X_i = \{x_{i,1}, ..., x_{i,m}\}$ is the list of all m tweets for user U_i, where $x_{i,j} = (a, b, c)$ is a 3-tuple of time, latitude, and longitude, then we can calculate the probability of user U_i making tweet y as:

$$p_i(y) = \frac{1}{m} \sum_{i=1}^{m} N(y, x_i, \Sigma)$$

where σ is a user specified covariance matrix which determines the level of smoothing. We choose a set of n points $(z_1, ..., z_n)$ uniformly at random from all possible (time, lat, long) tuples which lie inside the convex hull of our data set. Using these two components our feature vector is obtained by evaluating the probability distribution at each of the random points:

$$F_{\text{kde}}(U_i) = (p_i(z_1), ..., p_i(z_n))$$

This featurization preserves the manifold structure of the data. The euclidean distance between two users using this density based featurization can be thought of as a Monte-Carlo approximation to the distance between their tweet probability distributions. Hence the distance between two users is a smooth function of the proximity of their tweets.

We apply 4 distinct spectral clustering algorithms to the featurized data: (1) Spectral Clustering, (2) Spectral Clustering with Diffusion Maps, (3) Spectral Clustering with Diffusion Maps and Thresholding, (4) Spectral Clustering with IDR.

We evaluate the resulting clusterings quantitatively based on their predictive power. Specifically we use a metric which reflects the ability of group membership to predict tweet time/location. The idea is that in a good clustering all clusters will be tight, and that all points in the cluster will have similar behaviour, therefore group membership provides a great deal of information about the tweeting behaviour of group members. In a poor clustering, the clusters are loose,

and each cluster will contain a wide variety of disparate behaviour. In this setting cluster membership provides us very little information about the behaviour of group members. We use a metric which determines the top k most popular tweet time/location tuples, called hotspots, for each cluster, then measures the proportion of tweets for each user which are occur in one of these hotspots.

This real world dataset is a good testbed for our technique because it is a high dimensional dataset with a naturally occurring low dimensional manifold structure. Each user is a high dimensional point (in fact each user is a probability distribution over time/space), but we expect the topology of the set of all users to be low dimensional due to geographical constraints.

Details: We begin by splitting our dataset into a training set and a test set. We featurize both sets, then cluster the training set. Given a cluster of users $C_i = \{U_1, ..., U_m\}$ let $F(C_i) = \frac{1}{n}\sum_{i=1}^{m} F(U_i)$ be the cluster centroid. For each point in the test set y let $N(y)$ be the cluster with closest cluster centroid and assign it to that cluster. To create the hotspots we partition the space of observations using a grid and assign tweets to the corresponding grid cells. Hotspots corresponding to the k grid cells containing the largest number of observations. Let $H(C_i) = \{h_1, ..., h_k\}$ be the set of the k hotspots for cluster C_i. We define the predictive accuracy of our model for a single user U_j given cluster assignment C_i as:

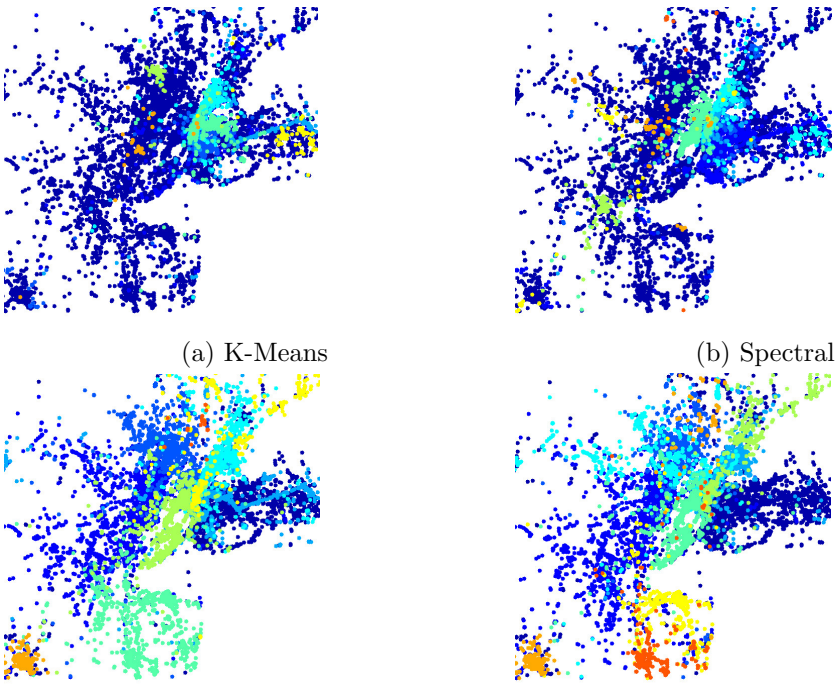
$$\text{Acc}(U_j|C_i) = \frac{\sum_{x \in X_j} \mathbb{1}(x \in H(C_i))}{|X_j|}$$

An accuracy of 1 would indicate that all user tweets lie within the cluster hotspots. An accuracy of 0 would indicate that no user tweets lie within the cluster hotspots. We measure the performance of each clustering algorithm using the mean accuracy over all users. This metric is particularly useful because it will only be large if the clusters are well balanced. While small clusters will result in good prediction accuracy for nearby points, this will result in other large clusters which perform poorly on all remaining points.

$$\text{Acc} = \frac{\sum_{i=1}^{|U|} \sum_{x \in X_i} \mathbb{1}(x \in H(N(x)))}{|X_j|}$$

6.2.1 Results

In Figure 6 we visualize the results of applying our 4 different clustering algorithms to the NY-NJ Twitter data using a density featurization. We see that spectral and k-means both produce comparable, low-quality clusterings which contain a single "mega cluster". Based on this visualization we can conclude that vanilla spectral clustering offers little to no benefit over k-means. This theory is supported by the the data in Table 1 which lists the accuracy of each clustering algorithm based on



(a) K-Means (b) Spectral

(c) Spectral with Diffusion Maps (d) Spectral with IDR

Figure 6: Results of clustering NY-NJ Twitter data. Point color indicates cluster membership

Algorithm	Accuracy
K-Means	0.23
Spectral	0.23
Spectral + Diffusion	0.30
Spectral + IDR	0.32

Table 1: Hot spot prediction accuracy

the metric discussed above. We see that both k-means and spectral have identical accuracy.

In contrast diffusion maps and IDR both have superior quality embeddings. In both cases there is no mega cluster, the clusters are well balanced, and they follow the roughly the geographical outlines we would expect. However based on the visualization it would appear that using IDR we can identify a couple of clusters not identified using diffusion maps. Again this theory is supported by the data in Table 1: We see that both Diffusion maps and IDR offer significant improvements in accuracy over k-means and Spectral clustering. Furthermore IDR offers a (slight) further performance improvement over Diffusion Maps.

The superior clustering provided by IVR presents us with numerous insights into the dataset. We see clusters emerging for many of the important ny-nj geo-political boundaries: These include Brooklyn, Staten island, Jersey City, Trenton, New Brunswick, and Philadelphia, Newark, Brick, Manhattan, and Queens. Many of these these clusters overlap: For example the individuals in the Brooklyn cluster also tweet heavily in Manhattan. Conversely individuals in the Manhattan cluster tend not to tweet in Brooklyn. This matches our expectations: Many people who live in Brooklyn tend to commute to Manhattan for work and leisure, however people who live in Manhattan tend not to commute to Brooklyn. Interestingly some clusters do not have this overlap. For instance the Brick cluster has relatively little overlap, suggesting that people who live in Brick tend to be far less likely to commute to another location for work. The Philadelphia cluster is another example of this pattern..

Furthermore we see many of the major expressways clearly picked out. We can see which expressways are used primarily by local traffic, and which are used by commuters, in addition to which expressways are used to commute between geopolitical areas. For example the Interstate 95 is clearly picked out as being heavily used by commuters.

Taken together, these results suggest that given a dataset which satisfies our modelling assumptions (low dimensional manifold structure) IDR allows us to effectively denoise the adjacency matrix, improving the performance of spectral clustering and providing us with a useful, high quality clustering of the data.

7 Conclusions and Future Work

We presented a new spectral clustering algorithm which uses iterated diffusion to detect and remove outliers from the graph adjacency matrix. This approach is based on the key idea that local diffusion can be used to reduce the weight of noise edges, and that we can iterate this process to drive noise edges to zero without affecting the other edges. We analyzed the behavior of this algorithm on a simple barbell graph and showed that it results in exponentially fast convergence. We applied this new algorithm two data sets, and showed that in both cases it provides improved performance when compared with alternative approaches.

In the future we hope to present a thorough theoretical analysis of the IDR algorithm on general graphs which satisfy a set of reasonable assumptions. We hope to establish a convergence guarantee, a rate of convergence, and a bound on the noise tolerance.

References

[Balakrishnan et al., 2011] Balakrishnan, S., Xu, M., Krishnamurthy, A., and Singh, A. (2011). Noise thresholds for spectral clustering. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 954–962. Curran Associates, Inc.

[Coifman and Lafon, 2006] Coifman, R. R. and Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30. Special Issue: Diffusion Maps and Wavelets.

[Dinh et al., 2009] Dinh, V. C., Leitner, R., Paclik, P., and Duin, R. P. W. (2009). *Image Analysis: 16th Scandinavian Conference, SCIA 2009, Oslo, Norway, June 15-18, 2009. Proceedings*, chapter A Clustering Based Method for Edge Detection in Hyperspectral Images, pages 580–587. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Donath and Hoffman, 1973] Donath, W. E. and Hoffman, A. J. (1973). Lower bounds for the partitioning of graphs. *IBM J. Res. Dev.*, 17(5):420–425.

[Donia et al., 2014] Donia, M. S., Cimermancic, P., Schulze, C. J., Brown, L. C. W., Martin, J., Mitreva, M., Clardy, J., Linington, R. G., and Fischbach, M. A. (2014). A systematic analysis of biosynthetic gene clusters in the human microbiome reveals a common family of antibiotics. *Cell*, 158(6):1402 – 1414.

[Fiedler, 1973] Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305.

[Luxburg, 2007] Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

[McMahon and Fields, 2015] McMahon, R. C. and Fields, S. A. (2015). Criminal conduct subgroups of "aging out" foster youth. *Children and Youth Services Review*, 48(C):14–19.

[Ng et al., 2002] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press.

[Pavan and Pelillo, 2007] Pavan, M. and Pelillo, M. (2007). Dominant sets and pairwise clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):167–172.

[Premachandran and Kakarala, 2013] Premachandran, V. and Kakarala, R. (2013). Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

[Spielman and Teng, 1996] Spielman, D. A. and Teng, S.-H. (1996). Spectral partitioning works: Planar graphs and finite element meshes. In *In IEEE Symposium on Foundations of Computer Science*, pages 96–105.

[Vesth et al., 2016] Vesth, T. C., Brandl, J., and Andersen, M. R. (2016). Fungeneclusters: Predicting fungal gene clusters from genome and transcriptome data. *Synthetic and Systems Biotechnology*, pages –.

[Wang et al., 2008] Wang, J., Chang, S.-F., Zhou, X., and Wong, S. T. C. (2008). Active microscopic cellular image annotation by superposable graph transduction with imbalanced labels. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.

[Xiang and Gong, 2008] Xiang, T. and Gong, S. (2008). Spectral clustering with eigenvector selection. *Pattern Recogn.*, 41(3):1012–1029.

[Zelnik-manor and Perona, 2004] Zelnik-manor, L. and Perona, P. (2004). Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press.

[Zhu et al., 2014] Zhu, X., Loy, C. C., and Gong, S. (2014). Constructing robust affinity graphs for spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*.