.Abstract

Geometry and Analysis of Dual Networks on Questionnaires

Jerrod Isaac Ankenman

2014

We investigate databases of binary questionnaire type, which map the Cartesian product of two sets into two distinct values. These databases naturally take the form of matrices, where the rows and columns each represent one of the sets and can be expected to have independent structure and correlations. We define a generative model for data of this type by assuming that the data is generated by independent Bernoulli draws from a probability field which is a function from the Cartesian product into [0,1]. We define a notion of function smoothness with respect to a metric induced by a partition tree on a set, and we further define a basis for the space of these matrices which is the tensor product of Haar-like systems defined on a pair of partition trees on the rows and columns. Then we define a notion of smoothness in two variables with respect to this bi-Haar basis, and assume that the probability field which generates the observed data is smooth with respect to some such dual geometry on the rows and columns. We describe an algorithm for discovering such a geometry which involves building graphs on the rows and on the columns, and constructing partition trees on each based on a metric induced by a random walk on these graphs.. We then iteratively refine the partition trees using an Earth Mover's Distance with respect to the tree metric in the other direction. After discovering the geometry of the sets, we describe an algorithm for reconstructing the underlying probability field subject to the smoothness condition previously defined. We further introduce a novel method of estimating the probability field for out-of-sample columns by adaptively selecting and observing a small subset of the data.

Geometry and Analysis of Dual Networks on Questionnaires

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Jerrod Isaac Ankenman

Dissertation Director: Ronald R. Coifman

May 2014

Contents

Acknowledgements

A doctoral dissertation is not, and is not generally intended to be, the project of a student working alone. But given how unlikely it would have seemed at many times in the past that I would be writing an acknowledgements section on such a dissertation, I feel both especially gratified to have the opportunity to do so and especially grateful to the many individuals who have made this possible.

As his many collaborators can surely attest, Raphy Coifman is a first-rate mathematician. Not only that, he was a first-rate advisor, finding a project that was nicely suited to my skills and allowing me enough freedom to put my own mark on it, while remaining always accessible, supporting, and vastly knowledgeable. Without his guidance none of this would be possible. There's no one else I'd prefer to tell me "No, you're wrong. It's a theorem!"

My officemate and friend Will Leeb has been a great help to me over the past two years, participating in fruitful discussions, and especially in helping to clarify bits of analysis that escaped me the first time. He had many useful suggestions and pointed out many errors in drafts of this dissertation. I'm so glad that I lobbied to have him moved into my office way back when; my graduate school career has been much more rewarding for it.

Of all the people on this list, Bill Chen, my longtime business partner and book co-author, probably had to most to do with my ending up in the field of mathematics. His willingness to completely overlook disparities in experience and credentials and just invite me to work together on math and game theory in poker was really the catalyst for "waking up" my previously languishing abilities, and getting me back on a track to success many years ago.

My wife Michelle definitely had the most to do with my ending up in graduate school. It was she that first encouraged me to look into programs in mathematics, and after I finished at Columbia, it was she who encouraged me to apply to graduate school and to aim high and apply to places like Yale. Her patience and willingness to sacrifice to support me through this process has been nothing short of amazing.

I would like to thank Peter Jones, Yuval Kluger, and Mauro Maggioni for agreeing to serve as readers for this dissertation.

I also would like to thank all the people who I have worked with in one capacity or another, who helped to make this dissertation possible. Among others, Doug Costa, Oksana Katsuro-Hopkins, and Catherine Bliss all wrote me letters of recommendation along the way, and they seem to have worked. Daniel Beylkin and Roy Lederman both entered the Applied Math program at the same time as I did, and I hope I was able to help them as much as they helped me in getting through courses and preparing for qualifying exams. Karen Kavanaugh has always been more than willing to help with all the details that really keep things on track.

This would also never have been possible without the love and support of my mother and my stepchildren Michael and Abby.

Most of all, this dissertation is dedicated to my daughter Mara, who delights me with her delight at how Daddy is going to be a "doctor of math." Yes, honey, people will come to me with their math problems and I will try to make them better.

A Note on the Interactive Dissertation

This dissertation, while it contains some theoretical results, is fundamentally empirical in nature. As a result, a substantial part of the value of the dissertation resides in code written to experiment with and implement the algorithms it describes. Almost all of this work was done using the excellent open source scientific software stack for Python, consisting of *numpy*[22], *scipy*[16], *matplotlib*[15], *scikit-learn[23], and IPython*[24]. We have not yet reached a point where a dissertation can be submitted online as an interactive, executable software package. However, in that spirit, in lieu of the usual figures and tables, we include several IPython notebooks rendered in the text, which show the results of interactive sessions, including code, plots, and markdown that should be considered part of the text. All these notebooks, as well as almost all the data that supports them (for legal reasons, we cannot release one data set), will be available for download at https://github.com/hgfalling/pyquest. In the spirit of open source software, we encourage the reader to download, experiment with, and possibly extend the software both as a companion to this thesis and as a powerful tool for data analysis.

1 Introduction

Problems involving the collection, analysis, and inference on large sets of high-dimensional Euclidean data are increasingly important in applications. The development of methods for dealing with such datasets has largely focused on setting the data in low-dimensional Euclidean space or on linear methods which struggle with the curse of dimensionality and other difficulties. Many such methods attempt to exploit the global geometry of data points by clustering, kernel methods, or classification by hyperplanes, but these methods often ignore the geometry in the other direction (correlations among the dimensions of observations). In this thesis, we further develop previously published methods for iteratively organizing data using correlations among observations to inform the organization of the dimensions of those observations, and vice versa. We propose a generative model for data of the binary questionnaire type, and propose methods of reconstructing the hidden parameters of that model. We further propose an adaptive method of reconstructing the model for a new person by only observing a selected but highly-reduced subset of the questions.

1.1 Problem setting

Consider the following common problems:

- A questionnaire consisting of m questions, to which n patients answer either yes or no.
- A database of sports match information, where n players or teams face each other in matches, whose outcomes are recorded as a win or loss.
- A database of documents, where n documents are queried for occurrences of m words, with the counts of each recorded.

- A recommendation platform, where n individuals rate a subsample of m movies/books/songs.

Each of these problems shares a common structure and a natural representation as a two-dimensional $m \times n$ matrix, and we will refer to data of this form as being of **questionnaire type**. Particularly, if the answers are restricted to two different options (ie yes/no), we refer to the data as **as being of binary questionnaire type**. Often we will refer to the rows of such a matrix as **questions** and the columns of such a matrix as **people**, and frame the matrix as the results of a questionnaire where n people were interrogated in m different ways. But this formulation is completely general; the method treats the transposed matrix in exactly the same way; both rows and columns are organized identically.

Let the dataset we want to consider be $X = \{x_1, ..., x_n\}$, a set of n observations in $\mathbb{R}^m$. Then the dataset naturally organizes into a $m \times n$ rectangular matrix. Now the standard assumptions of classical statistics, independence and identical distribution, immediately break down because there are significant, potentially non-linear correlations both among the rows and among the columns. Importantly, there is no particular reason to prefer treating one dimension as independent, while the other is treated as dependent. We first describe a generative model for data of binary questionnaire type, and follow that with a thorough description of the methods by which we find that model's parameters.

2 A Generative Model for Questionnaire-type Data

Suppose we are given X, a binary questionnaire type dataset. The answer to each question $X(i, j)$ is either "yes" $(+1)$ or "no" (-1). There are a few questions to be asked. What does it mean to find the "best" organization of rows and columns? What does that organization imply? And if we knew how to find it, what could we do with it?

2.1 Probability Fields

Suppose that we consider the process by which individuals respond to the questions in the questionnaire as a signal to be processed. In one sense, there is no "noise" in the record of the responses, assuming the respondent actually did mark "yes" or "no" in response to each question and those choices were accurately recorded and passed on. If the response of each individual to each question were deterministic, then this would be a completely noiseless setting, and organizing the data by its bidirectional similarity as well as possible would be an end goal. However, we can make the following empirical observation: in the list of questions, there are many pairs or groups which appear to be nearly identical in meaning. Yet it is not uncommon for individuals to answer these questions differently. It may be that there is some subtlety to the precise wordings that causes this effect. But surely any test-taker can relate to situations where the right answer is unclear, and the decision to answer "yes" or "no" is made in some arbitrary way that may not be repeatable. The fact that the answer to a question must be quantized into "yes" or "no" is therefore a form of noise. So instead of modeling the dataset as a true and noiseless capture of the intersection of people and questions, we instead model each question/person pair as a random variable instead.

- We assume that there exists an $m \times n$ matrix F whose elements are on $[0, 1]$. Then X (the observed data) is the result of sampling independent Bernoulli variables with parameters equal to the entries of F; that is:

$$X(i,j) \sim (2)\text{Bernoulli}(F(i,j)) - 1$$

2.2 Smoothness

The above model, though, is insufficient to allow for meaningful description of F. The observed data X exhibits what we might call **quantization noise**, which occurs because the output values are restricted to $\{-1, 1\}$. For example, suppose that under the above model we sought a maximum likelihood estimator F; since there are no constraints on the structure of F, we would find that

$$F(i,j) = \begin{cases} 1 & X(i,j) = +1 \\ 0 & X(i,j) = -1 \end{cases}$$

One useful condition to place on F is that it satisfy some smoothness condition with respect to a geometry on the rows and columns of the matrix. The condition we will enforce was first introduced in [26], and updated by [6], and can be characterized as a **bi-Hölder** condition: that averages of data on rectangles in the matrix have a mixed derivative that is bounded by a power of the sizes of the rectangles. We will define the rectangles and this condition precisely in 4.4, but we include the following brief example of estimating a smooth probability field from sampled binary data when the geometry is known.

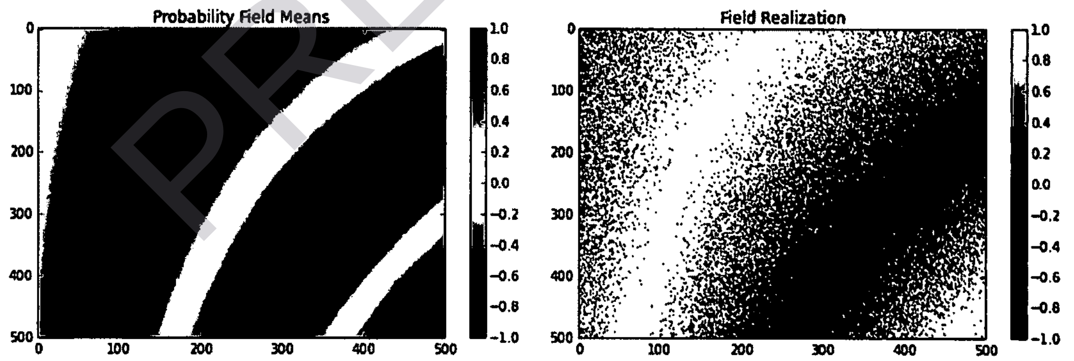
Reconstruction of a Sampled Smooth Function

In this short example, we generate a smooth artificial probability field. We then sample the field as independent Bernoulli variables, and then recover the underlying probability field from the sampled data, using binary trees built on the known geometry.

```
In [1]:  from imports import *
```

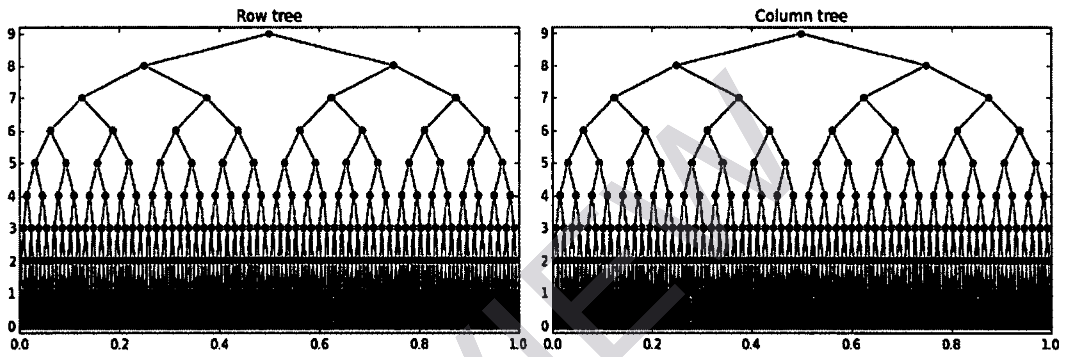
We define the sample probability field to be a 500×500 matrix P. We let i vary uniformly on $\left[0, \frac{\pi}{4}\right]$ and j vary uniformly on $[0, 2\pi]$ over the rows and columns of the matrix respectively. Then we let the entries of the matrix be $P(i,j) = \frac{1}{2}\left(1 + \sin\left(\frac{i+j+2ij}{2}\right)\right)$. We then sample from the field, taking successes to be $+1$ and failures to be -1.

```
In [2]:  n_rows,n_columns = 500,500
         means_matrix = np.zeros([n_rows,n_columns])
         for i in xrange(n_rows):
             for j in xrange(n_columns):
                 ii = i*0.25*np.pi/n_rows
                 jj = j*2.0*np.pi/n_columns
                 means_matrix[i,j] = np.sin((ii+jj+2*ii*jj)/2.0)*1.0

         np.random.seed(20140304)
         pf = artificial_data.ProbabilityField(means_matrix/2.0+0.5)
         orig_data = pf.realize()

         fig = plt.figure(figsize=(12,4))
         fig.add_subplot(121)
         cplot(means_matrix,colorbar=True,title="Probability Field Means")
         fig.add_subplot(122)
         bwplot2(orig_data,colorbar=True,title="Field Realization")
         plt.tight_layout()
         plt.show()
```



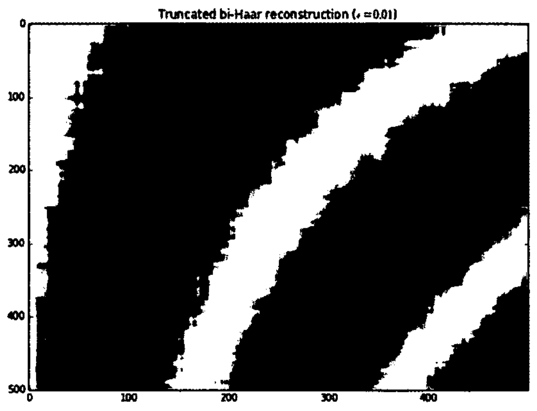
Next we define binary trees on the rows and columns.

```
In [3]:  rt = bin_tree_build.random_binary_tree(n_rows,1.0)
         ct = bin_tree_build.random_binary_tree(n_columns,1.0)

         fig = plt.figure(figsize=(12,4))
         fig.add_subplot(121)
         plot_tree(rt,title="Row tree")
         fig.add_subplot(122)
         plot_tree(ct,title="Column tree")
         plt.tight_layout()
         plt.show()
```



We then reconstruct the probability field by expressing the sampled data in the bi-Haar basis implied by these trees, and truncating the coefficients corresponding to folders smaller than ϵ to 0.

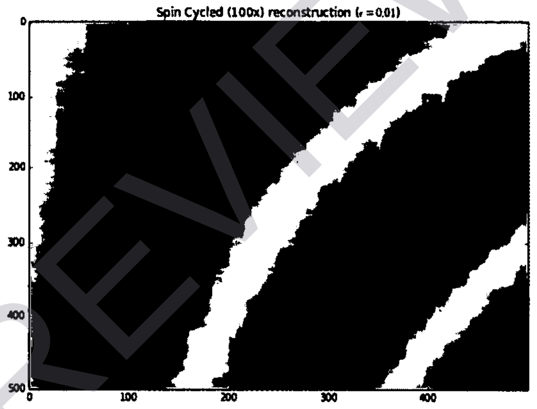
```
In [4]:  tr = tree_recon.recon_2d_haar_folder_size(orig_data,rt,ct,0.01)
         tree_recon.threshold_recon(tr,-1.0,1.0)
         fig = plt.figure(figsize=(8,6))
         fig.add_subplot(111)
         cplot(tr,title="Truncated bi-Haar reconstruction ($\epsilon = 0.01$)")
         plt.tight_layout()
         plt.show()
```



We now consider the result of "spin cycling" the reconstruction to remove artifacts. We introduce a slight element of randomness to the tree construction, and generate ten pairs of trees. Then we cross-match the row and column trees with each other, generating 100 pairs of trees, each of which has a slightly perturbed basis. Then we reconstruct as before, and average over the 100 tree-pairs.

```
         row_trees, col_trees = [],[]
In [5]:  spun_recon = np.zeros(orig_data.shape)
         for i in xrange(10):
             row_trees.append(bin_tree_build.random_binary_tree(n_rows,1.2+0.2*i))
             col_trees.append(bin_tree_build.random_binary_tree(n_columns,1.2+0.2*i))
         for rt in row_trees:
             for ct in col_trees:
                 tr = tree_recon.recon_2d_haar_folder_size(orig_data,rt,ct,0.01)
                 tree_recon.threshold_recon(tr,-1.0,1.0)
                 spun_recon += tr
         spun_recon /= 100.0
```

```
         fig = plt.figure(figsize=(8,6))
In [6]:  fig.add_subplot(111)
         cplot(spun_recon,title="Spin Cycled (100x) reconstruction ($\epsilon = 0.01$)")
         plt.tight_layout()
         plt.show()
```



Now in this toy example, the reconstruction method here is not particularly efficient. Because the geometry of the image is uniform in each direction, standard signal processing techniques could be applied with good effect. We present this only as a short example of the process of recovering a smooth function from a given geometry. In other settings, the geometry of the rows or columns ("people") might be of higher dimension, or oscillating at different rates. In these cases, these reconstruction techniques continue to work just as well.

The point here is that our model assumes the following:

- There exists some permutation of the rows and some permutation of the columns such that if we consider F on the permutation, F is smooth.

Suppose we are given a potentially shuffled matrix of binary data where we can reasonably make these assumptions. Our goals are to first discover (from the empirical data) the dual geometry on the rows and columns with respect to which F is smooth. We can then use the geometry we learn, in conjunction with the empirical data, to estimate F.

3 Diffusion Maps

Suppose that we have X, an $m \times n$ matrix, with both m and n of reasonable size, and we are seeking a geometry on the rows and columns. One natural thing to do is to use the geometry of the original Euclidean space from which the data came. However, this will fail as a general method of defining a geometry, because of the well-known result that the discriminatory power of distances falls off rapidly as the dimension of the data increases. Most common approaches apply some type of dimensionality reduction to the problem. Additionally, in settings that contain significant noise (such as the binary samples from a probability field), we are especially interested in metrics robust to noise which heavily weight local geometry and deemphasize the preservation of large distances. For these purposes, we will employ diffusion maps as introduced by [5, 18].

3.1 Diffusion geometry and the diffusion distance

Let Ω be a set of n points. Then we define an **affinity** $k : \Omega \times \Omega \to \mathbb{R}$ as a kernel function with the following three properties:

- $k(x, y) = k(y, x)$ (symmetric)
- $k(x, y) \geq 0 \ \forall x, y$ (non-negative)
- $\sum_x \sum_y k(x, y) f(x) f(y) \geq 0$ (positive semidefinite)

Given any function of this type, we can immediately construct a graph G, where the nodes are the n points, and edges of weight $k(i, j)$ link points $i, j \in \Omega$. Now let K be an $n \times n$ matrix and let $K_{ij} = k(i, j)$. Now we define a random walk on the vertices of G as follows: Let $\omega(x)$ be sum of the row x in K and $\omega(y)$ be the sum of column y in K; that is,

$$\omega(x) = \sum_y k(x, y)$$

Let D be the diagonal matrix with entries $D(x, x) = \omega(x)$. Then $M = D^{-1}K$ is a Markov matrix which can be thought of as defining a random walk on the graph G, where $M\,(x, y)$ is the probability of transitioning in one time step from point x to point y. Likewise $\left(M^t\right)(x, y)$ is the probability of transitioning from point x to point y in t time steps. So now we can use this random walk as the basis for a distance between points. We define the **diffusion distance** between points x and y at a particular time t as:

$$D_t^2(x, y) = \sum_{z=1}^{n} \frac{1}{\pi(z)} \left(\left(M^t\right)(x, z) - \left(M^t\right)(y, z) \right)^2$$

where π is the stationary distribution of the Markov chain. This distance is the weighted Euclidean distance between the distributions induced by the random walk of t steps on x and y . It can be characterized as measuring the overall rate of connectivity between points of the data set. It will be small if the random walk contains many paths of length less than t between x and y, while it will be large if the number of such connections is small.

This diffusion distance is often signficantly more robust to noise than, say, the Euclidean distance in the original space. The reason for this robustness is that the effect of spurious or incoherent connections between points are dampened. Suppose point x and point y are strongly connected in the graph G, but this is in truth a product of some kind of noise and the points x and y are unrelated. When we look at the diffusion distance, we would see that point x diffuses partly to point y, but mostly to its neighbors who are not connected to y. Likewise y diffuses partly to x but mostly to its neighbors who are not connected to x. This continues as the diffusion time increases; essentially there is only one path between x and y, because the neighbors of x are far from the neighbors of y. So diffusion distance is a more **coherent** distance on the graph; the diffusion distance does not only consider the strength of the direct link between x and y, but the strength of all the paths from x to y.

3.2 The diffusion embedding

$M = D^{-1}K$ is not necessarily symmetric, but it is easily shown that it is similar to a symmetric matrix. Since K is symmetric by the symmetry of the kernel function and D is a non-singular diagonal matrix, we have that $D^{-\frac{1}{2}}KD^{-\frac{1}{2}}$ is symmetric. Then $M = D^{-1}K = D^{-\frac{1}{2}}\left(D^{-\frac{1}{2}}KD^{-\frac{1}{2}}\right)D^{\frac{1}{2}}$. Hence M and $D^{-\frac{1}{2}}KD^{-\frac{1}{2}}$ have the same eigenvalues and the eigenvectors of M can be readily obtained by a diagonal transformation of the eigenvectors of $D^{-\frac{1}{2}}KD^{-\frac{1}{2}}$.

Now take$\{\lambda_1, ..., \lambda_n\}$ to be the eigenvalues and $\{\psi_1, ..., \psi_n\}$ the corresponding eigenvectors of M. We define the **diffusion embedding** $\varphi_t : \Omega \to \mathbb{R}^n$ by

$$\varphi_t(x) = \left\{\lambda_1^t\psi_1(x), ..., \lambda_n^t\psi_n(x)\right\}$$

φ_t maps Ω into $\mathbb{R}^n$ but with point locations given by the diffusion distance metric instead of the Euclidean metric. Now notice here that the eigenvalues λ_i are the eigenvalues of a Markov process; hence we have immediately that $|\lambda_i| \leq 1$ for all i, and further if the graph G is connected, $\lambda_1 = 1$ and the other eigenvalues decay in absolute value from there. Truncating φ_t by taking the first k eigenvalues and eigenvectors leads to a k-dimensional embedding within the original space.

Recall that the diffusion distance at time t was defined as:

$$D_t^2(x,y) = \sum_{z=1}^{n} \frac{1}{\pi(z)} \left(\left(M^t\right)(x,z) - \left(M^t\right)(y,z)\right)^2$$

It can be shown (see [5, 18]) that:

$$D_t^2(x,y) = \sum_{j \geq 0} \lambda_j^t \left\|\psi_j(x) - \psi_j(y)\right\|^2$$

So the Euclidean distance between the coordinates of two points in the diffusion map at time t is exactly the diffusion distance between those two points at time t. For more on the theory of diffusion maps and examples, consult [5, 7, 18, 21]. In our work, we will use diffusion maps for three main purposes:

1. To generate diffusion distances used in creation of initial and flexible trees, as discussed in 4.1.2.

2. As part of the process of generating eigenvector-cut binary trees, as discussed in section 4.1.1.

3. For visualization purposes after running the iterated questionnaire process discussed in section 5.3.

3.3 Affinity

In the previous sections, we described a method for utilizing diffusion maps to embed high-dimensional data into a low-dimensional manifold in the ambient space. This technique was based on building a graph structure on the data based on a given affinity between points. But different choices of affinity can lead to quite different results, and the choice of affinities will normally be driven by the characteristics of the data. For example, suppose that the data is of binary questionnaire type, like a psychological questionnaire where a broad population answers a long panel of questions. Then we can characterize a person's responses as a function supported on the set of questions. More colloquially, a person's responses form an overall response profile. Then the affinity we construct between two people is an attempt to characterize the similarity of the response profiles to the panel of questions.

A commonly used kernel is based on a metric:

$$k(i,j) = e^{-\frac{\|x_i - x_j\|^\alpha}{\epsilon}}$$

where α is some prespecified power and the parameter ϵ is data-dependent and causes the affinity to decay rapidly as the distance between points grows, thus limiting the affinity to (relatively) near neighbors. This helps to mitigate the difficulties in high dimension, because we ignore the problems in discrimination among objects that are far away; only local distances are preserved. Euclidean distance with a Gaussian($\alpha = 2$) is popular here, although other distances may also be utilized. In 5.2, we will suggest that an affinity based on a distance equivalent to Earth Mover's Distance will be useful.

[20]proposed a characterization of the affinity between two points i and j by making the assumption that there are some number of (hidden) equivalence

classes, around which the data is sampled with some unknown noise. Then the affinity between i and j might be the probability that i and j came from the same equivalence class. Other kernels not based on distances are likewise possible. For many data sets, affinities based on the inner product between two data points, such as correlation or cosine similarity, are natural, with a threshold added to set negative values to zero.

In any event, any of these might make useful affinities on appropriate datasets. The point is that the affinity is a proxy for the **global** closeness of two points – that is, people i and j are close (have high affinity) if their response profile against the entire panel of questions is similar in whatever metric we have chosen.

For questionnaire-type data, we often use a normalized correlation-like function, thresholded cosine similarity. Let X_i and X_j be the vectors of data associated with points i and j and let $t \geq 0$ is a threshold parameter discussed below. Then let

$$c_{ij} = \frac{\langle X_i, X_j \rangle}{\|X_i\| \, \|X_j\|}$$

The affinity would then be

$$A(i,j) = \begin{cases} c_{ij} & c_{ij} \geq t \\ 0 & c_{ij} < t \end{cases}$$

Unthresholded, this produces affinities between -1 (perfect anti-correlation between answers) and +1 (perfect correlation). In fact, when the data are coded as +1 (yes) and -1 (no), then this function is simply the number of matching answers minus the number of non-matching answers, divided by the total number of questions. We then want to at least threshold at 0, since one of the requirements on affinities is that they be non-negative.

It is often helpful to even further sparsify the graph by removing edges corresponding to affinities below some threshold. With a carefully chosen threshold, this has the effect of helping to denoise the graph, from the assumption that weak affinities are more likely to be the product of noise than they are to represent something true about the dataset. A threshold chosen too low (or no threshold at all) may result in a graph which is highly connected with little differentiation between clusters. On the other hand, a high threshold will result in a disconnected graph with many components, a situation that is undesirable for the diffusion map, as mass will then be unable to diffuse from one component to a different one. In general, choosing the proper threshold is somewhat data-specific; one possible idea is to slowly increase the threshold until the graph becomes disconnected. Calculating this is a simple matter of monitoring the value of the second eigenvalue of the Markov matrix; as long as it is 1, the graph is disconnected. At the point when the graph does become connected, we might drop the threshold a little so as not to force the connectedness of the graph to rely on one single edge. We may also consider thresholding determined locally, based on some number of neighbors with the highest affinity instead of a global threshold. Again we must be careful to avoid disconnecting the graph. Any thresholding technique can be applied both to the affinity itself or to the Markov matrix – in the former case, edges are removed because of absolute weakness, while in the latter case edges are removed because they are relatively weak compared to other edges attached to a node.

4 Partition Trees

A key tool we will use in our data organization methods is the **partition tree**, defined in the next section. Partition trees serve three main purposes in our overall process. First, they serve as an encoding of similarity between rows or