An Experimental Study into Spectral and Geometric Approaches to Data Clustering

Prashant Sridhar

October 2015
CMU-CS-15-149

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15289

Thesis Committee:
Dr. Gary Miller, Chair
Dr. Alex Smola

Submitted in partial fulfillment of the requirements for the degree of Masters in Computer Science.

Keywords: Geometry, Nonparametric, Clustering

For my parents

Acknowledgments

I would like to thank my advisor, Dr. Gary Miller, for allowing me to work with him through this project.

Abstract

Common clustering techniques involve assumptions on the distribution that the data is drawn from, with linearity often being a standard assumption. These techniques work poorly on irregular data sets or on rare clusters. Non-parametric solutions are often inefficient to use in practice and still fail to find rare clusters. This thesis explores geometric approaches to non-parametric clustering that should be much better at identifying these rare clusters. The first approach explores how the Gabriel graph can be used to compute density based distance metrics and how the Gabriel graph itself might be used to cluster data. The second approach views the probability density function as a heat distribution over a metal plate. Traditionally, finite element or control volume methods construct graphs over meshes on the metal plate. This approach explores how these meshes could be used to partition space. Finally, we present the results of running these clustering algorithms on flow cytometry data and compare these to present state of the art non-parametric methods on the field.

Contents

List of Figures

Chapter 1

Introduction

Clustering of data points is a problem that has existed in machine learning literature for quite some time now. The general definition is to partition a data set into "clusters" based on their similarity. Clustering techniques have several applications to fields like computational biology, computer vision, robotics, and others. The defining characteristic of clustering that also makes it so powerful is that it is traditionally defined in the unsupervised sense. This means that data is presented to the algorithms without any prior labels designating clusterings. This unsupervised nature is part of the appeal since unlabeled data is often easy to collect and quite plentiful, while labeling data is often laborious, requiring a human to hand label the data set. However, while unsupervised learning is often easy to collect data for, it is notoriously difficult to judge when the algorithm has produced a good clustering.Parametric clustering methods often have a well defined statistical interpretation, but perform poorly on data sets that do not satisfy the model assumptions. Non-parametric methods, on the other hand, are either very slow to implement in practice or not very well defined statistically. This thesis provides a statistical basis for a non-parametric clustering method that is efficient in practice.

Spectral methods [21] are a popular way to cluster data sets non-parametrically. The data points are connected together to form a graph with heavy weight edges between similar points and light edges between dissimilar points. This graph can then be cut in order to generate the appropriate clustering. Strong edges will rarely be cut because they would greatly decrease the cut quality. Conversely, weak edges are encouraged to be cut in order to keep the cut value low. This leaves similar nodes in the same cluster while separating dissimilar nodes. As a non-parametric method, it is resistant to non-linearities in the data set, however, it is often quite slow to run in practice. If a complete graph of the data is constructed, storing the graph itself will require $O(N^2)$ space. While quadratic time algorithms are slow to run in practice on larger data sets, quadratic space algorithms are completely infeasible even on smaller samples. Storing the graph itself will require several gigabytes of memory in data sets larger than a few ten thousands of points preventing most computers from even loading the graph into RAM. Further, if the graph is to be partitioned into k clusters, algorithm will require finding the k largest eigenvalues of graph laplacian. If the graph is very near complete, and consequently the laplacian is very dense, this operation will be very expensive making it infeasible to run on anything but the smallest data sets. Finally, deciding exactly how to weight the edges of the graph is still a matter of trial and error. The most common method is to use a Gaussian kernel to decide the edge weights, but

there is little statistical basis to this. It is merely a way to handle non-linearity of the input data. Choosing the bandwidth of this Gaussian kernel, again, must be done with grid search and cross validation.

The method of clustering proposed in this thesis solves all the problems mentioned above. Our method also relies on partitioning a graph, however, we provide some statistical explanation for the edge weights. Additionally, our method constructs a graph with only a linear number of edges in $O(nlogn)$ time. This linear size graph is feasible to store and results in a sparse laplacian, which can also be solved easily. We view clustering of data points as a partitioning of the probability space. Our interpretation of a good partition is one separating two high density regions while having a low cross-sectional cut volume. This encourages splitting up maxima across lines drawn through density troughs. This idea is also very similar to sparsest cuts on graphs and lends itself very easily to graphical interpretations. The goal is to construct a sparse graph wherein cut quality is similar to that of the distribution generating the data. The probability distribution can be viewed as a heat distribution over a metal plate. Such heat distributions can be modelled by triangular meshes over the data with added Steiner points and can be solved by control volume methods. Since the mesh is a triangulation, it will only have a linear number of edges. The triangular mesh can then be cut with standard spectral methods in order to retrieve the clustering. More specifically, the mesh approximates second nearest neighbour distances for the distribution, but these distances can easily be used as an estimate of the density function. Since we are only concerned with clusters and not the point-wise estimation of density, we do not face the traditional problems faced by other knn approaches of having non-convergent variance [9]. Integrated over finite measure sets, variance of the estimator drops off with the size of the sample. Further, clusters of at least $O(logn)$ elements should be identified by our approach since by Chernoff bounds, enough points will be sampled from these clusters to determine the cut. This gives provides a guarantee that small clusters will be identified by our approach. Finally, as a spectral, non-parametric method, this algorithm is insulated against non-linearities in the data allowing it to identify clusters with non elliptical shapes. Thus, the approach overcomes the shortcomings of previous clustering algorithms.

This thesis serves mostly as an experimental exploration into the idea of sparse graphs for data clustering since much of the theory is still being developed. Primarily, this paper focuses on data with ”rare” cluster populations and highly non-elliptical clusters that are difficult to cluster with standard methods like gaussian mixture modeling [7] or k-means. Specifically, we will deal with flow cytometry data. Flow cytometry is a laser based approach to differentiating between populations of cells based on certain bio markers on the cells surface or inside its body. Cytometers can take measurements from several thousands of cells per second leading to large datasets. Flow cytometry is often used in the diagnosis of several health disorders including various types of cancers. It is also used in clinical trials including research to autonomously compute prognoses for HIV patients. Many of these applications require clustering cell types as a primitive before prediction can take place, and at present, such clustering is often done by hand. Several model based clustering algorithms have been developed for this problem such as, FLAME, flowClust, flowMerge. These methods are all mixture models over t-distributions or skew t-distributions. Skew t-distributions, in theory, should be able to account for eccentricities in the data, but the time complexity scales to the fourth power with dimension rendering it useless in practice beyond 5 dimensions. Further, these model based methods often fail to find rare cell

populations which are of particular interest to biological applications. Non-parametric methods have also been tried in this space and mostly rely on spectral clustering. SAMSpectral solves the issue of quadratic space by heuristically sampling data, which the authors call faithful sampling, to form a representative population. As with all heuristics, there is no theoretical basis to this approach and it provides no provable guarantees. Our geometric spectral clustering method outperforms SAMSpectral while being able to produce guarantees on quality and run time.

Chapter 2

Historical Spectral clustering

Given a set of data points $x_1, x_2, \ldots, x_n$ and a notion of similarity s_{ij}, the goal of any clustering algorithm is to assign points of low similarity to different clusters while keeping high similarity points grouped together. Spectral clustering builds on the notions of graph cuts in order to achieve our goal of clustering. To do this, we build a similarity graph, $G = (V, E)$. In this graph, each vertex v_i corresponds to a data point x_i. Two vertices v_i, v_j are connected by an edge e_{ij} with weight equal to the similarity between the corresponding data points, s_{ij}. Clustering data points can now be rewritten to split the similarity graph into groups such that edges running between different clusters are weak while those running within a cluster are strong.

To begin with, we go over basic terminology and notation used in this chapter. Let $G = (V, E)$ be the affinity graph defined on the data points with vertex set $V = v_1, v_2, \ldots, v_n$. G is an undirected graph with weighted edges such that the edge weight e_{ij} between vertices v_i, v_j is equal to then similarity function of the two data points $s_{ij} \geq 0$. The degree d_i of vertex v_i is defined to be

$$d_i = \sum_j e_{ij}$$

The graph has a corresponding weighting adjacency matrix A whose entries a_{ij} are defined as follows

$$a_{ij} = \begin{cases} 0 & \text{if } i = j \\ s_{ij} & \text{otherwise} \end{cases}$$

Finally, the degree matrix D of the graph, with entries d_{ij}, can be defined as follows

$$d_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Consider a set of vertices A and its corresponding inverse set, V/A, the set of all vertices not in A. As shorthand, let $A' = V/A$, and $i \in A$ mean that $v_i \in A$. Define the volume of A, $|A|$ as follows,

$$|A| = \sum_{i \in A} e_{ij}$$

A serves as a partition of the vertices and there is a cut separating A from A'. Let $cut(A, A')$ be the surface area of the cut defined as follows,

$$cut(A, A') = \sum_{i \in A, j \in A'} e_{ij}$$

Therefore, $|A|$ is the sum of edge weights such that both edge end points lie within A while $cut(A, A')$ is the sum of edge weights such that only one end point is within A.

For data drawn from $\mathbb{R}^d$, the similarity between two vertices can be defined in terms of the Euclidean distance between the corresponding points. We define $s_{ij} = k(x_i, x_j)$ for some function k based on only the distance between the two points. This function k must be monotonically decreasing on the distance to ensure that near by points are very similar while points far away are very dissimilar. Traditionally, a Gaussian kernel is used for k.

$$k(x_i, x_j) = exp\left(\frac{-||x_i - x_j||_2^2}{2\sigma^2}\right)$$

Where σ is a parameter(width) controlling the diameter of the clusters. Other methods of constructing the similarity graph include only connecting each point to its k nearest neighbours, or only connecting it to points within an ϵ ball around it.

Given a similarity graph with an adjacency matrix A, and degree matrix D, define the graph laplacian L as

$$L = D - A$$

The main tool of spectral clustering is the graph laplacian and its eigenvalues and eigenvectors. For example the number of connected components in a graph is equal to the number of 0 eigenvalues in its graph laplacian. As discussed earlier, clustering on a graph can be phrased as partitioning the vertex set so that the volume of each vertex cluster is large while the surface area of the cut is small. This can be formalized in the Normalized cut. Let $A_1, A_2, \ldots A_k$ be the k partitions of the vertices. Define $NCut(A_1, A_,2, \ldots A_k)$ as follows.

$$NCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, A_i')}{|A_i|}$$

Before detailing how to minimize the Normalized cut, define the normalized graph laplacian as follows

$$\mathcal{L} = D^{-1} L$$

The traditional spectral clustering algorithm proceeds as follows,

- Compute similarity graph G.
- Compute normalized graph laplacian $\mathcal{L}$.
- Compute Λ_k, the matrix of the k smallest eigen vectors of $\mathcal{L}$.
- Run k-means on the columns of Λ_k.
- Return the result as the cluster assignment for the data set.

It can be shown that this algorithm solves a relaxed version of the NCut problem. The problem with spectral clustering is that results vary widely based on the type of similarity metric used and how points are connected to form a similarity graph. The rest of this thesis discusses principled approaches to solving this problem.

Chapter 3

Density based distance metrics and clustering

Our initial approach to combating the non-linearity of cytometry data, was to try non-euclidean distance metrics. Under a density based metric, two points are closer together if joined by a path through high density space, and farther apart if only connected through low density space. This should allow cluster detection even for clusters with highly non-elliptical shape. However, density based metrics are often intractable to compute exactly leading most such approaches to use reliable approximations. A recent approximation method developed by Choen et. al., [2], provides a graph based method to approximate a density based metric by computing shortest paths on a Gabriel graph over the points. The final graph has a linear(in the points) number of edges leading to quick single source shortest path queries. However, naive implementations of k-means replacing Euclidean distance with density based distance will still run into the problem of having to store all pairs shortest paths. This will still require $O(n^2)$ space rendering the approach infeasible. While, straightforward applications of the distance metric might not be feasible, spectrally cutting the Gabriel graph itself intuitively leads to a good clustering. This section of the thesis does not contain fully reasoned theory. Instead, it is an intuitive exploration of the usefulness of the Gabriel graph. While it is hard to arrive at provable bounds for cluster quality, Gabriel graphs are not as prone to the curse of dimension. While the number of edges grows exponentially with dimension, it is still upper bounded by $O(n^2)$, or the complete graph. On the other hand, the main method presented in this paper is based on Delauny triangulations and require time exponential in the dimension to construct. Therefore, the Gabriel graph may provide a clue on how to deal with high dimensional data sets. Before explaining how to cluster using the Gabriel graph, this thesis will first go over the work of Cohen et. al. to provide some background for the discussion and define relevant terms.

3.1 Approximation of density based distance metrics

If $\mathbb{R}^d$ space is Endowed with a distance metric, we can define the length of an arbitrary path γ through this space as follows. Let the length function of a path under Euclidean distance be ℓ_e, and let γ begin at the point x at time 0 and end at the point y at time 1.

$$\begin{aligned}\ell_e(\gamma) &= \int_\gamma ds \\ &= \int_0^1 \left|\frac{d\gamma(t)}{dt}\right| dt\end{aligned}$$

Which is simply the path integral of γ in $\mathbb{R}^d$. Here, $\left|\frac{d\gamma(t)}{dt}\right|$ is the velocity of trajectory at time t. Note that this integral is minimized by using the straight line path between the two points. Using this formulation for the length of a path, we can define the Euclidean distance, $\mathbf{d}_e$ between the points x and y to be,

$$\mathbf{d}_e(x, y) = \inf_\gamma \ell_e(\gamma)$$

This is merely a mathematical precise way of saying that the Euclidean distance between two points is the length of straight line shortest path between the points.

With Euclidean distance formally defined, this can now be extended to more general distance metrics. By adding a cost function based on location in the space, segments of a path through different regions are priced differently thereby contorting the shortest path away from the straight line path between the points. Let the cost function for a certain metric k be $c(x)$. We can now define the length of a path γ for this metric, ℓ_k, as follows.

$$\begin{aligned}\ell_k(\gamma) &= \int_\gamma c(s)ds \\ &= \int_0^1 c(s)\left|\frac{d\gamma(t)}{dt}\right| dt\end{aligned}$$

There is also an analogous definition for distance, $\mathbf{d}_k$, for this metric.

$$\mathbf{d}_k(x, y) = \inf_\gamma \ell_k(\gamma)$$

Density based distance metrics penalize paths through low density regions and reward paths through high density regions. Therefore, when used as a similarity metric, it will cluster together points belonging to the same high density regions. Let P be the probability distribution over $\mathbb{R}^d$, and $f(x)$ be its density function at point x. Define $c_p(x) = f(x)^{\frac{2(1-p)}{d}}$ to be the cost function used for the density based distance metric. Also define $\ell_p(\gamma)$ and $(d)_p(x, y)$ to be the associated path length function and distance metric. Note that $p = 1$ returns the Euclidean metric.

The problem with using density based metrics is that for $p \neq 1$, computing $\ell_p(\gamma)$ is near intractable for many distributions and paths. Often the true distribution is not even known and must therefore be estimated. Worse, computing $\mathbf{d}_p(x, y)$ involves finding an infimum over an infinite number of paths. Hence, the distance can never be exactly computed and must be approximated.

Assume that n points are uniformly randomly sampled from P to form the training data set D. We can now define an undirected graph G over these points such the edge length, $e_{xy}; x, y \in D$, is equal to the p^{th} power of the Euclidean distance between the points, $e_{xy} = \mathbf{d}_e(x, y)^p$. Define $\mathbf{d}_G(x, y)$ to be the shortest path distance from x to y in the complete graph G. The following theorem from Hwang et. al., [12] proves a strong connection between $\mathbf{d}_G(x, y)$ and $\mathbf{d}_p(x, y)$ at the price of a few assumptions on the sample space and the density function. Assume that P is instead defined over a manifold M, the same manifold that D is drawn from.

Theorem 3.1.1. *Assume M is compact, and that f is continuous and supported everywhere over M. There exists a constant $C(d, p) > 0$, which only depends on d and p, satisfying the following. Let $\epsilon > 0$ and $b > 0$, then there exists $\theta > 0$ such that*

$$P\left(\sup_{x,y}\left|\frac{\boldsymbol{d}_G(x,y)}{n^{\frac{(1-p)}{d}}\boldsymbol{d}_p(x,y)} - C(d,p)\right| > \epsilon\right) < e^{-\theta n^{\frac{1}{(d+2p)}}}$$

for all sufficiently large n where $x, y \in M$ and $\boldsymbol{d}_e(x, y) \geq b$.

In the case where M is closed instead of compact, the paper presents a second theorem giving an almost surely limit relating the two metrics.

Theorem 3.1.2. *Assume M is closed and that f is continuous and supported everywhere over M. There exists a constant $C(d, p) > 0$, which only depends on d, and p, satisfying the following.*

$$\lim_{n\to\inf} n^{\frac{p-1}{d}}\boldsymbol{d}_G(x,y) = C(d,p)\boldsymbol{d}_p(x,y)$$

for fixed $x, y \in M$.

The assumptions of note here are that M is either compact or closed. This means that the theorems do not work in the general case where data can be drawn from anywhere in $\mathbb{R}^d$. In practice, however, this assumption is not a problem because the set of valid data to be clustered can often be bounded between some finite intervals. The density function is also assumed to be continuous and supported everywhere over its domain. This can be achieved by simply shrinking its domain to only contain space in which f is supported. Since points cannot be drawn from 0 probability regions, this assumption is easily met in practice. What these assumptions buy though, are strong bounds relating $\mathbf{d}_p$ and $\mathbf{d}_g$ thereby giving a way to approximate density based distance metrics. However, while computing shortest paths on G is tractable, computing G itself requires quadratic space rendering this approach useless on larger data sets. This problem can be avoided by considering the special case of $p = 2$, as is done by Cohen et. al.

3.2 Gabriel graphs and approximate Gabriel graphs

Once we restrict $p = 2$, we can immediately make a few observations about the graph G. Since edge weights are now squared Euclidean distance, the shortest path between two points will not a long edge if another point exists inside of the circle inscribed by the edge as a diameter. Let the graph constructed for $p = 2$ be G_2.

Definition 3.2.1. *An edge e_{xy} between points x, y is said to be Gabriel if there exists no z such that z is contained within the circle inscribed on e_{xy} as diameter.*

Lemma 3.2.1. *For all $x, y \in G_2$, the shortest path from x to y will only contain edges that are Gabriel.*

Proof. Assume for the sake of contradiction that the shortest path from x to y contains an edge e_{uv} that is not Gabriel. By 3.2.1, there exists a point z inside the circle formed using e_{uv} as diameter. Consider the triangle formed by u, v, z. By the properties of triangles, the angle formed at z will be at least $90°$. Therefore, $e_{uv} \geq e_{uz} + e_{vz}$ by the Pythagorean theorem. Therefore we have found a shorter path and have arrived at a contradiction. □

Definition 3.2.2. *The graph GG_2, with $V = D$ and $E = \cup e_{xy}, e_{xy}$ is said to be the Gabriel graph over the points in D.*

Note that the set of Gabriel edges is a subset of Delauny edges since the Gabriel requirement is a stricter form of the Delauny requirement over edges.

The Gabriel graph serves as a possibly sparse graph that can be used to compute the density metric. Because the Gabriel graph is a subset of the Delauny triangulation, it serves as a linear spanner for $d = 2$. This is because the Delauny triangulation in the 2D case forms a planar graph which necessarily has a linear number of edges. However, even in 3 dimensions, examples can be constructed where the Gabriel graph has $O(n^2)$ edges. Consider the unit sphere in 3 dimensions and two small arcs on along this sphere on the XY plane and the YZ plane with both arcs centered around $Y = 0$. Arrange $n/2$ points equally spaced on each arc. By construction, every edge between a pair of points on different arcs will be Gabriel since each of those edges is approximately a diametric chord for the sphere. Therefore, this example leads to at least $n^2/4$ edges. This example can be seen in 3.1.

This prompts the need for spanners on the Gabriel graph. It is worth noting that even in higher dimensional cases where the Gabriel graph is linear, there is no fast algorithm to compute it. The naive approach of checking if each edge is Gabriel requires $O(n^3)$ time even when only a linear number of edges are Gabriel.

While very efficient algorithms [5], [3], working in $O(nlogn)$, time exist for computing Euclidean spanners, these algorithms are still exponential in dimension because they rely on space partitioning trees. In practice, these algorithms do not work well for $d > 10$. For this reason, other approaches that do not scale so poorly with dimension are desirable.

While the original paper did not present a way to compute the Gabriel graph quickly, they do present an algorithm to produce a linear, $(1 + \epsilon)$ spanner given the Gabriel graph. The simple construction involves constructing a conflict graph for each node p based on the angles formed by its neighbours. For any node p, construct a graph, H_p by connecting p to all its neighbours, and connect any pair of neighbours a, b if the angle formed by e_{pa} and e_{pb} is sufficiently small, say, smaller than θ. Define I_p to be the set of edges in the maximal independent set H_p. The linear spanner of the Gabriel graph is the graph H_θ such that

$$E_{H_\theta} = \{(p, q) | e_{pq} \in (I_p \cup I_q)\}$$

Before presenting the proof of quality for the spanner H_θ, we must first prove two supporting lemmas. We show that if two edges make a sufficiently small angle, then the length of the path does not change by much depending on which we pick.

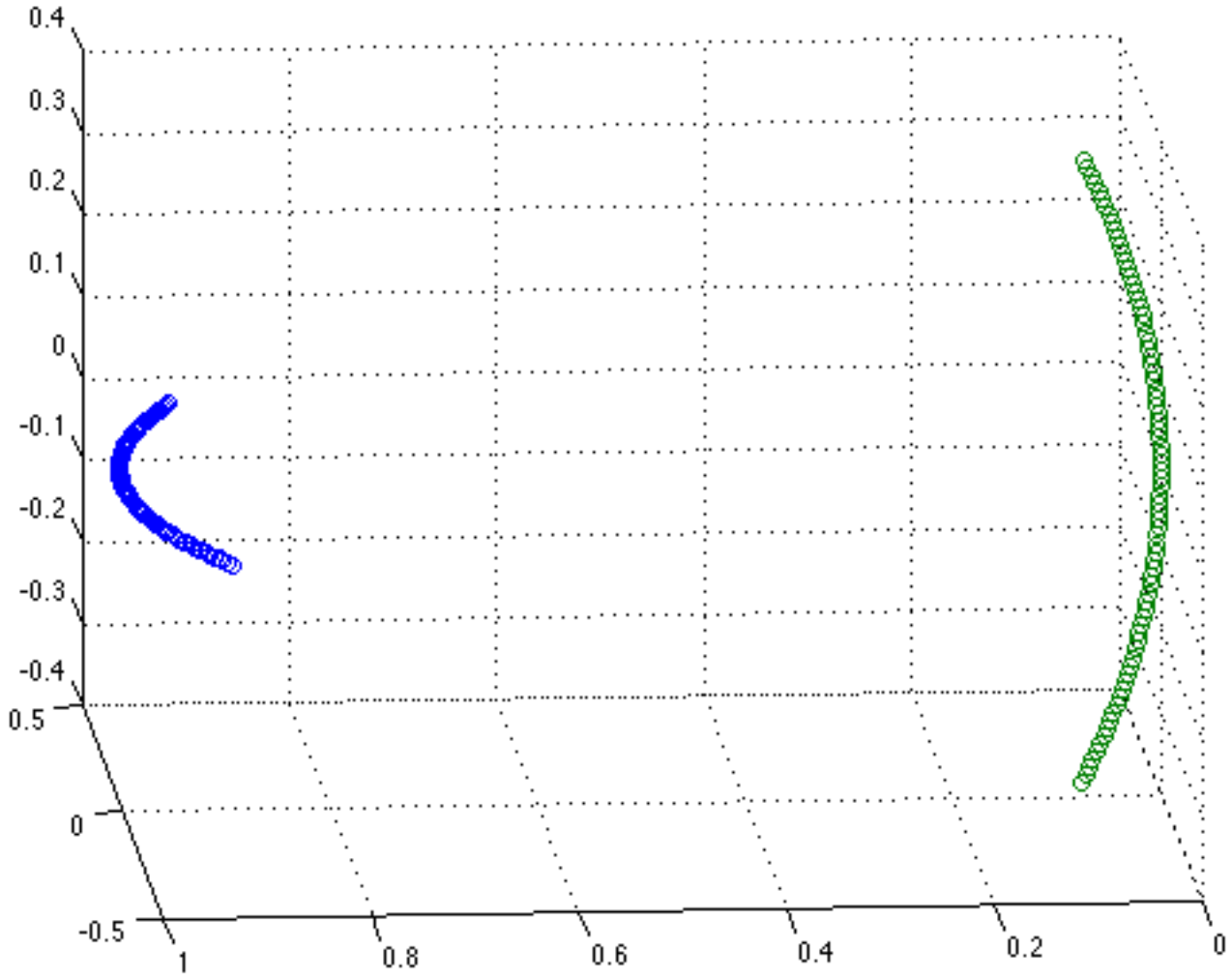

Figure 3.1: Quadratic size Gabriel graph example

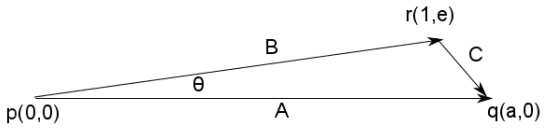

Figure 3.2: Gabriel edges with small angle

Theorem 3.2.2. *Consider a point p with neighbours q, r such that e_{pq}, e_{pr} are both Gabriel and make an angle of at mot θ. Let A be the Euclidean length of e_{pq} and B be the Euclidean length of e_{pr}. Then it is the case that for any path through the graph containing e_{pq}, its length would not change by more than $(1 + 2tan^2\theta)|A|^2$ if e_{pr} were traversed instead.*

Proof. Begin by noticing that we can assume e_{qr} exists in the graph. If e_{qr} is not in the graph, then it is not Gabriel, implying that there exists a point s within the circle with qr as diameter. Then, traversing e_{qs} followed by e_{sr} would be shorter than directly traversing e_{qr}. Therefore, e_qr's existence serves as an upper bound assumption for the stretch generated by not traversing e_{pq} and does not weaken the proof.

Let C be the Euclidean length of e_{qr}. We want to show that $|B|^2 + |C|^2 \leq (1 + 2tan^2\theta)|A|^2$.

Without loss of generality, assume that p is at the origin, q is at $(a, 0)$ and r is at $(1, e)$. This is shown in 3.2

$$\begin{aligned}\frac{|C|^2 + |B|^2}{|A|^2} &= \frac{(1-a)^2 + e^2 + 1 + e^2}{a^2} \\ &= \frac{2 - 2a + a^2 + 2e^2}{a^2}\end{aligned}$$

This ratio is at its worst when $a = 1$, or when A and C are perpendicular. To see this, differentiate the above ratio with respect to a. The derivative is negative for $1 \leq a \leq 1 + e^2$. For a outside of this range, either q or r will encroach on the ball of the other. Therefore, if the angle between A and B is less than θ,

$$\begin{aligned}\frac{|B|^2+|C|^2}{|A|^2} &\leq \frac{|B|^2+|B|^2 sin^2\theta}{|B|^2 cos^2\theta} \\ &= \frac{1+sin^2\theta}{cos^2\theta} \\ &= 1+2tan^2\theta \\ \implies |B|^2+|C|^2 &\leq (1+2tan^2\theta)|A|^2\end{aligned}$$

□

This theorem justifies why a maximal independent set of the conflict graph of a point is sufficient to preserve path lengths. If an edge is not present in the maximal independent set, then it must have been the case that it conflicted with some other edge at a low degree. Therefore, the other edge can be traversed instead with small cost to total length. One other lemma is needed for the the proof of the spanner. We need to show that the $|C|$ is not too large as compared to A.

Theorem 3.2.3. *Consider a point p with neighbours q, r such that e_{pq}, e_{pr} are both Gabriel and make an angle of at mot θ. Let A be the Euclidean length of e_{pq}, B be the Euclidean length of e_{pr} and C be the Euclidean length of e_{qr}. Then $|C|^2 \leq tan^2\theta|A|^2$.*

Proof. Once again, without loss of generality, assume p is on the origin, q is at $(a, 0)$ and r is at $(1, e)$.

$$\begin{aligned}\frac{|C|^2}{|A|^2} &= \frac{(1-a)^2+e^2}{a^2} \\ &= \frac{1-2a+a^2+e^2}{a^2}\end{aligned}$$

This ratio is again maximized if e_{qr} and e_{pq} are perpendicular. Therefore, if the angle between the edges is θ, we have

$$\begin{aligned}\frac{|C|^2}{|A|^2} &\leq \frac{|B|^2 sin^2\theta}{|B|^2 cos^2\theta} \\ &= tan^2\theta \\ \implies |C|^2 &\leq tan^2\theta|A|^2\end{aligned}$$

□

With this theorem in place, we are now ready to prove correctness for the spanner. The proof will be by induction.

Theorem 3.2.4. *For any pair of points p, q in GG_2, the path from p to q through H_θ will larger by at most a fraction of $(1+\epsilon)$ for any θ such that $\tan^2\theta \leq \frac{\epsilon}{2+\epsilon}$. Further, H_θ will have a linear number of edges in fixed dimension.*

Proof. This theorem will be proved using induction over the length of the longest edge in a path. For the base case, consider (p, q) the closest pair in the data set. Since they are the closest pair, p is q's nearest neighbour. Nearest neighbour edges are always Gabriel because if they were not, a point would lie within the circumscribing circle thereby generating a nearer neighbour. e_{pq} is also in H_θ because if it were not, then it must conflict with another edge, say, e_{pr}. By 3.2.3, e_{qr} must be shorter still meaning p, q is not the closest pair.

For the inductive step, assume that for all paths between pairs (p, q) with longest edges shorter than $|A|$, H_θ is a $(1 + \epsilon)$ spanner. Now consider all path with a longest edge of length $|A|$. If this edge is in H_θ, then the above theorem holds. If the edge is not in H_θ, then it must conflict with some neighbour. Let the endpoints of the longest edge be p, q, and the end points of the edge it conflicts with be p, r. Let B be the length of e_{pr} be $|B|$ and the length of e_{qr} be $|C|$. Let P be the path from q to r in H_θ. Since $|C|^2 \leq tan^2\theta|A|^2$, the shortest path from q to r in GG_2 cannot have any edges as long as $|A|$. Therefore by the inductive hypothesis,

$$|P| \leq (1 + \epsilon)|C|^2$$

Again, note that if e_{qr} is not in GG_2, the length of the path would be even shorter. Hence the assumption that e_{qr} is in GG_2 is an overestimate.

We now want to bound $|B|^2 + |P|$ since that is the path from p to q in H_θ.

$$\begin{aligned} |B|^2 + |P| &\leq |B|^2 + |C|^2 + \epsilon|C|^2 \\ &\leq (1 + 2tan^2\theta)|A|^2 + \epsilon tan^2\theta|A|^2 \\ &\leq (1 + (2 + \epsilon)tan^2\theta)|A|^2 \\ &\leq (1 + \epsilon)|A|^2 \end{aligned}$$

Finally, H_θ must have a linear number of edges in a fixed dimension because in any fixed dimension, each node may only have a constant out degree when constrained to only have neighbours separated by a minimum degree. □

3.3 Weakly Gabriel graph and fast linear spanner

The last section went over how to construct Gabriel graphs to compute density based distance metrics and how to sparsify these graphs. This section weakens the conditions on Gabriel graphs to allow for faster algorithms to construct them. Further, we show that we can sparsify these more general Gabriel graphs to still retain linear sized graphs. To begin with, we define the notion of an edge being weakly Gabriel. The edge to a neighbour is said to be weakly Gabriel if there are no other neighbours within the dihedral ball of the edge. However, other, non-neighbours are still allowed to encroach the ball. Formally, we define weakly Gabriel as follows,

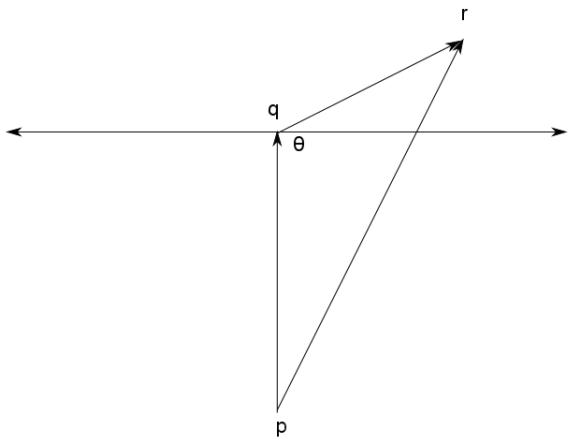

Figure 3.3: Candidate hyperplane elimination

Definition 3.3.1. *Consider a point p, with its neighbour set V_p. Let q be a point in the neighbour set, $q \in V_p$. Define the edge e_{pq} to be weakly Gabriel if, $\forall r \in V_p$, r is not in the dihedral ball of e_{pq}.*

Definition 3.3.2. *Consider a point p. The neighbour set V_p of p is said to be weakly Gabriel if $\forall q \in D, e_{pq}$ is Gabriel, $q \in V_p$, and $\forall q \in V_p, e_{pq}$ is weakly Gabriel.*

Definition 3.3.3. *For a data set D, the graph WG_2 is a weakly Gabriel graph over D if, $\forall p \in D$, V_p is weakly Gabriel.*

To begin with, notice that each data set has a unique Gabriel graph, but need not have a unique weakly Gabriel graph. Secondly, notice that a weakly Gabriel graph still contains all Gabriel edges and therefore preserves all shortest paths. Hence, the shortest path between any two nodes, and, by extension, the approximate density based distance, is not changed if a weakly Gabriel graph is used instead of a Gabriel graph. The trade-off is that a weakly Gabriel graph will always be at least as dense as a Gabriel graph and could potentially have a quadratic number of edges even when the Gabriel graph is linear. However, the main gain of using the weakly Gabriel graph is that its definition lends itself to a fast iterative, input sensitive algorithm where Gabriel graphs must be computed in $O(n^3)$ time.

The first observation needed for this algorithm is that for a given point, the edge to its nearest neighbour must necessarily be Gabriel. Necessarily, there may be no points in the shared edges dihedral circle. This serves as a base case for the algorithm. Once we have identified a Gabriel edge, several candidate neighbours can be immediately eliminated as seen in the 3.3.

The points above the hyperplane can all be eliminated because the nearest neighbour would encroach on their ball. The hyperplane check can be performed simply by checking if the angle θ in the figure is obtuse. This is equivalent to checking if the cosine is negative, or simply checking if the dot product is negative. If only the Gabriel edges were desired, every single point would

have to draw its hyperplane and eliminate potentially offending points. However, this will again require $O(n^3)$ time. Instead, we need only eliminate points with successive nearest neighbours in order to obtain a weakly Gabriel neighbour set. This is formalized in 1

Algorithm 1: Compute Weakly Gabriel Graph

Data: Data set D
Result: Weakly Gabriel Graph, $WGG = \{V, \{V_p\}\}$ over D
$WGG \longleftarrow \{\ \}$
for $p \in D$ **do**
  $V_p \longleftarrow \{\ \}$
for $p \in D$ **do**
  $d_p \longleftarrow [\ ]$
  for $q \in D - \{p\}$ **do**
    $append(d_p, (q, \mathbf{d}_e^2(p, q))$
  $s_p \longleftarrow sort(d_p)$ ascending by distance
  while s_p *is not empty* **do**
    $r \longleftarrow pop(s_p)$ **for** $s \in s_p$ **do**
      if $cos(rp, rs) > 0$ **then**
        $remove(s_p, s)$
    $add(V_p, r)$
  $add(WGG, \{p, V_p\})$
  for $q \in V_p$ **do**
    if $p \notin V_q$ **then**
      $add(V_q, p)$

The run time of this algorithm is $O(n^2 logn)$ for the sorting step, and $O(n^2 e)$ for the elimination steps, where e is the maximum out degree of a point. In practice, this algorithm works well at building a sparse graph on which to run shortest path queries because many real world data sets have linear size Gabriel graphs. For those applications, The weakened Gabriel graph algorithm produces only a small fraction of non-Gabriel edges. For example, in one flow cytometry data-set consisting of roughly 70,000 points in 16 dimensions, only 2% of the edges generated were non-Gabriel. This highlights the algorithms input sensitivity. In most practical use cases, the bottle-neck in runtime is the sorting step giving an overall complexity of $O(n^2 logn)$ since e is often a constant. However, it is possible to construct edge cases with a linear number of Gabriel edges for which 1 produces a quadratic number of edges leading to a $O(n^3)$ time and $O(n^2)$ space complexity.

The good news is that sparsifying based on angle will continue to work on weakly Gabriel graphs. If a non-Gabriel edge is ever picked in favour of a Gabriel edge, the stretch will still remain small because the edges must be comparable in length. If this were not the case, one would encroach on the dihedral ball of the other violating the requirement that both edges be at least weakly Gabriel. This preserves lemmas, 3.2.2 and 3.2.3. With these lemmas in place, the main proof carries through very similarly.

Instead of first running 1 and then sparsifying the resultant graph, the two steps can be combined into one. When a point is added to the neighbour set and used to prune potential neighbours, the only check performed is the hyperplane check. Instead, the neighbour can prune points by its hyperplane and also by the angle made with p. Finally, because the resultant graph will have constant(in dimension) out degree from each node by virtue of being a linear spanner, the sorting step can be removed in favour of picking the closest neighbour each time. Since we will only have to pick a linear number of neighbours, this reduces the runtime from $O(n^2 logn + n^2 e)$ to $O(n^2 e)$, which is the same as $O(n^2)$. This algorithm is formally represented in 2

Algorithm 2: Compute Sparse Weakly Gabriel Graph

Data: Data set D
Data: Error threshold ϵ
Result: Weakly Gabriel Graph, $WGG = \{V, \{V_p\}\}$ over D
$WGG \longleftarrow \{\ \}$
for $p \in D$ **do**
  $V_p \longleftarrow \{\ \}$
for $p \in D$ **do**
  $d_p \longleftarrow [\ ]$
  for $q \in D - \{p\}$ **do**
    $append(d_p, (q, \mathbf{d}_e^2(p, q))$
  while d_p *is not empty* **do**
    $r \longleftarrow findmin(d_p)$
    $remove(d_p, r)$
    for $s \in d_p$ **do**
      if $cos(rp, rs) > 0 \ || \ tan^2(pr, ps) > \frac{\epsilon}{2+\epsilon}$ **then**
        $remove(s_p, s)$
    $add(V_p, r)$
  $add(WGG, \{p, V_p\})$
  for $q \in V_p$ **do**
    if $p \notin V_q$ **then**
      $add(V_q, p)$

3.4 Results and Discussion

While asymptotically we can achieve $O(n^2)$, in practice it is often faster to perform the sorting step and run at $O(n^2 logn)$. It can be shown that for a point placed on the origin and surrounded by neighbours drawn uniformly from the unit ball, the expected number of Gabriel edges is $O(2^d)$. It can also be empirically observed that the average angle between edges is $90°$. Therefore, each node often has a large out degree making repeatedly picking the nearest neighbour very slow if 2^d exceeds $logn$. This also demonstrates that for data with no irregularities, sparsification is not necessary and constructing a weakly Gabriel graph is sufficient. Also, the number of non-Gabriel

edges added by 2 is a very small percentage of the total edges. For the applications discussed in this paper, no more than 5% of the edges generated are non-Gabriel. Therefore, 2 is a significant improvement in computing approximations to density based distance.

Clustering with non-Euclidean distance metrics poses a challenge because computing the mean of a cluster is difficult thereby rendering k-means impossible to use. The solution is to assume that the cluster center will be a point in the data set. This gives rise to the k-medioids algorithm [13]. The most common realization of the idea is Partitioning Around Mediods (PAM) [22] which proceeds as follows.

- Select k random points without repetition to serve as cluster medioids.
- Assign each point to a cluster based on medioid distance.
- Compute total cost of clustering and repeat until cost converges
  - For each medioid m and non-medioid o
    - Swap m and o and recompute cost.
    - Keep swap if cost decreases.

Other medioid based algorithms also exist such as the following based on Voronoi iterations [16].

- Select k random points without repetition to serve as cluster medioids.
- Assign each point to a cluster based on medioid distance.
- Compute total cost of clustering and repeat until cost converges
  - For each cluster, select m as medioid if m minimizes the sum of distance to all other points.
  - Reassign points to the closest medioid.

If the number of iterations required to converge is assumed to be t and that pairwise distances are stored in a Gram Matrix G, then the runtime of PAM is $O(tkn^2)$. Each internal step require $O(n^2)$ time to swap a medioid with each non-medioid point and recompute the cost. The second method also has a similar runtime of $O(tkn^2)$ since it take $O(n^2)$ time to find the optimal medioid for each cluster. The advantage of the second approach is it more closely mirrors the more familiar k-means algorithm and is this easier to reason about. The main drawback to both of these approaches is that when the distance metric is density based distance, computing the distance between two points requires a shortest path computation using Dijkstra's algorithm, which will cost $O(enlogn)$ where e is the maximum out degree. This makes computing distance on the fly infeasible. The other alternative is to precompute all pairs shortest paths and store the result in a Gram matrix. However, computing and storing the Gram matrix will have a $O(n^2)$ space complexity. While this approach is not further explored in this thesis, a solution to this problem might arise from the graph structure used to compute density based metrics. To begin, let us define the notion of closeness centrality [19] in a graph. The closeness centrality $C(x)$ of a graph is defined as follows, $C(x) = \frac{1}{\sum_y \mathbf{d}_e(y,x)^2}$. The reciprocal of the closeness centrality is the distance from a node to all the other nodes in the graph. This is very similar to the k-medioids requirement of computing the distance from the medioid to every other node in the cluster. Several algorithms such as [23][17][6] deal with computing closeness centrality or a related metric

called betweenness centrality [8] which is often considered harder to compute than closeness centrality. These algorithms rely on sampling to build approximations to closeness and the methods therein could significantly speed up medioid computation. Sampling techniques could also be useful in cluster assignment since the subsampled paths or nodes could build approximations to the distance to different medioids for each point.

The other alternative is to cluster directly based on the properties of the Gabriel graph. Instead of setting the edge weight e_{xy} to be the squared Euclidean distance, $e_{xy} = \mathbf{d}_e(x, y)^2$, set the edge weight to be the reciprocal, $e_{xy} = \frac{1}{\mathbf{d}_e(x,y)^2}$. Geometrically close points in this graph are usually connected by a single Gabriel edge or a series of short hops. In either of those cases, short edges will have very high weights leading the cut between the points to be very expensive. On the flip side, long edges on the graph are not very common since they are very likely to have some other point encroach on them. They mostly arise from the boundaries of clusters. These long edges are also much cheaper to cut than short edges, thereby incentivising cuts between clusters. In non-sparsified Gabriel graphs, there is some chance that there many connecting edges between two clusters if the clusters are sufficiently compact and far away. This is similar to the scenario where $O(n^2)$ Gabriel edges exist in the graph. These dense inter cluster connections are problematic for clustering because they drive up the cost of cutting the two clusters with several nearly identical edges. Sparsified Gabriel graphs fix this issue by removing such "duplicate" edges. If several edges arise from and end with similar nodes, the edges will make small angles with each other and will hence be pruned. This makes sparsified Gabriel graphs very good to cluster over. Note that instead of the reciprocal of the squared Euclidean distance, a gaussian kernel could have been used for edge length. While results vary, they do not change significantly by altering the kernel. The weights that are mentioned in this section are used because they tie in to ideas of effective resistance and conductance of the edges. The experimental results section will go into more detail but for now consider figure 3.4 of a clustering constructed by partitioning Gabriel graphs.

The data set used here is called the Barcode data [11] set and it captures the difficulty of clustering rare populations. The Barcode data set also provides an easily verified correct answer that is not available for other flow cytometry applications(where experts often also disagree on clusterings). If we contrast the quality of this clustering with K-means, we see that cutting the Gabriel graph performs much better. The primary win of this approach is that the Gabriel graph is still fast to construct in high dimensions. While the runtime will depend on the dimensionality of the data, the number of edges in the graph is bounded by $O(n^2)$ and since 2 is an input sensitive algorithm, the runtime of that algorithm is also bounded. On the other hand, the approach mentioned later in this thesis requires computing the Delauny triangulation, a procedure that scales exponentially with dimension. The primary downside of Gabriel graphs is that clustering on them is not based soundly in theory while this is not true for the approach to be discussed in the next chapter. However, our algorithm grounded in theory still calls upon the Delauny triangulation, and since Gabriel edges are a subset of the Delauny edges, the Gabriel graph might serve as an approximation to the more principled clustering approach in high dimension.

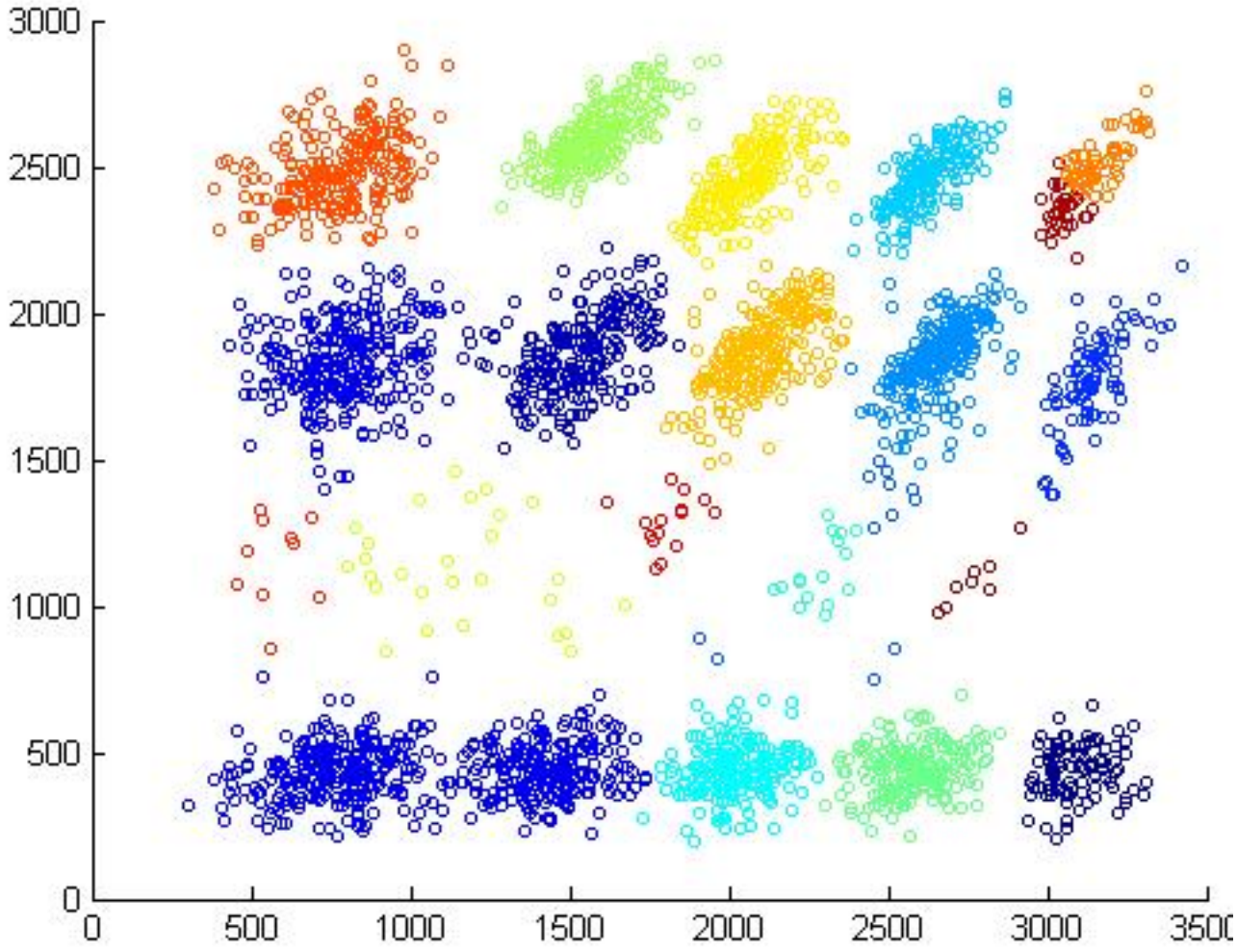

Figure 3.4: Barcode data clustered with a Gabriel graph

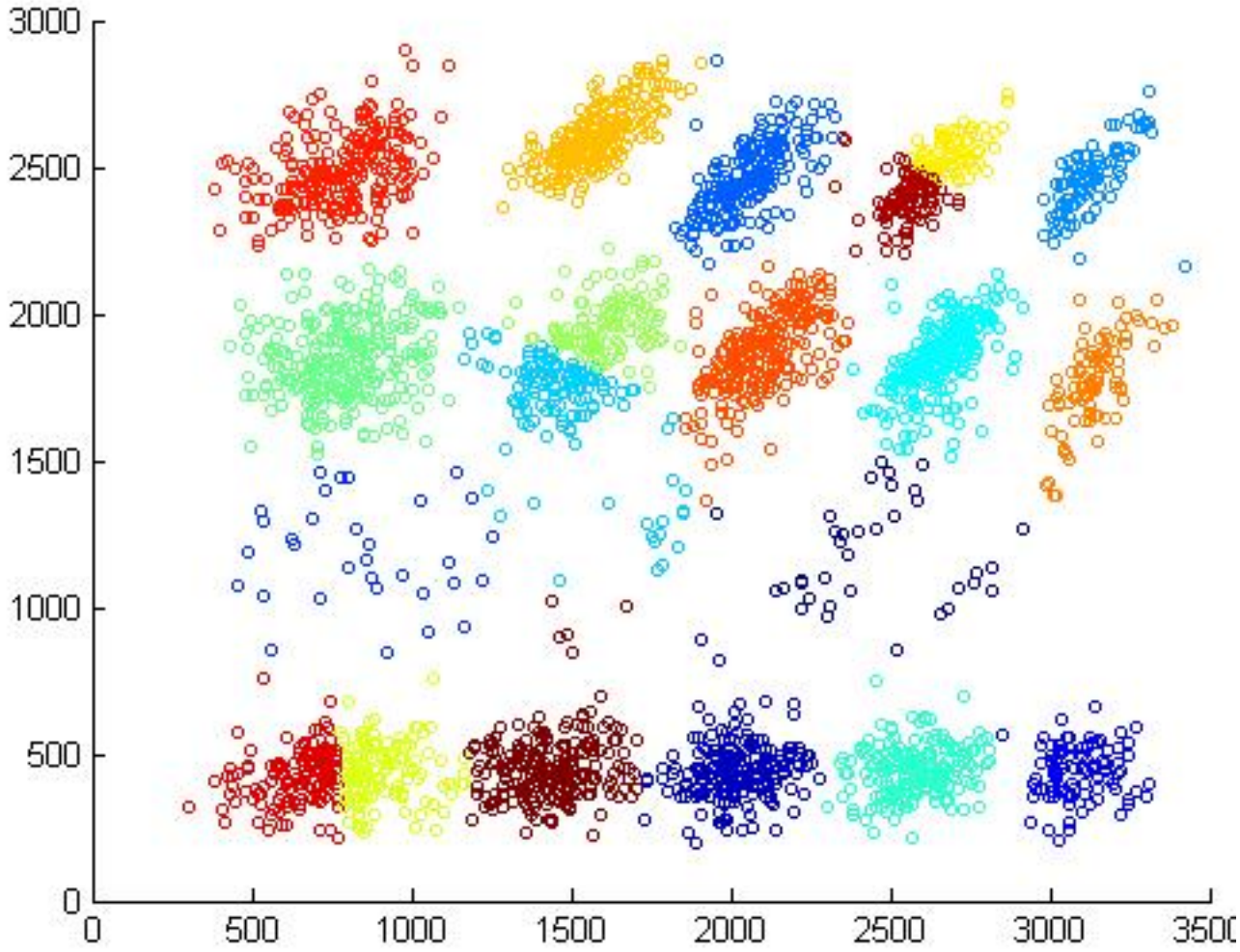

Figure 3.5: Barcode data clustered with a K-means

Chapter 4

Geometric non-parametric clustering

In this section we present a statistical framework to clustering and a geometric approach to solve the resulting optimization problem. The goal of this section is to quickly compute linear graphs with which to cluster. Further, we ground the graphs built in theory in order to prove bounds on the quality of the clusters obtained.Traditionally, spectral clustering does not have a statistical interpretation behind it. However, spectral clustering relies on notions of graph cuts. Define the Normalized cut (NCut) [21] as follows,

$$NCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{cut(A_i, A_i')}{|A_i|}$$

Here, $A_1, \ldots, A_k$ are the k partitions of the node set of the graph. $|A_i|$ is the volume of a cluster and is equal to $\sum_{j,k} e_{jk}$ such that x_j or $x_k \in A_i$. A_i' is the set of nodes in the graph that do not belong to A_i. Finally, $cut(A_i, A_i')$ is the surface area of the cut separating A_i from the rest of the graph. $cut(A_i, A_i') = \sum_{j,k} e_{jk}$ such that $x_j \in A_i, x_k \in A_i'$. Note that the quantity is divided by 2 so as to not double count the edges.

The most common definition of graph cuts is defined as follows,

$$cut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i} cut(A_i, A_i')$$

However this definition does not provide good clusters because there is strong incentive to remove single nodes with small out degree. The normalized cut circumvents this problem by normalizing by cluster volume thereby encouraging larger clusters. The NCut quality would be minimized by ensuring that $|A_i$ is the same for all i, i.e. equal volume clusters. As explained in the section on spectral clustering, the spectral clustering algorithm solves a relaxed version of the normalized cut problem.

In this thesis, we extend the notion of normalized cuts to probability distributions. Define the Normalized Distribution cut (NDCut) as follows,

$$NDCut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{\oint_{\gamma(A_i . A_i')} dA}{\int_{A_i} dV}$$

At this point it is important to note the k-way expansion cut [14]. Let the cut, Kcut, be defined as follows.

$$Kcut = max \left(\frac{cut(A_i, A_i')}{|A_i|}, i = 1, \ldots, k \right)$$

This k-way expansion cut more closely resembles the isoparametric cut for which Cheeger bounds exist [1]. Cheeger bounds bound the quality of this cut by the second eigenvalue of the graph laplacian. Unfortunately, similar bounds do not exist for the normalized cut. While the k-way expansion cut has good theoretic properties, practitioners still use normalized cuts for their link to spectral clustering. In this thesis, we will continue to do the same.

In this context, let A_i be a closed subspace of Ω corresponding to a cluster. A_i' is the conjugate space of A_i, $A_i' = \Omega - A_i$. $\int_{A_i} dV$ is the volume integral of A_i and $\gamma(A_i, A_i')$ is the surface seperating A_i from A_i'. This means $\oint_{\gamma(A_i, A_i')} dA$ is the surface integral of the cut seperating out A_i. The intuition behind the normalized distribution cut is very similar to that of the vanilla normalized cut. The formulation encourages cutting the probability space into maximum volume clusters such that the surface area separating them is minimized. Since this formulation makes no assumptions on the underlying distribution, it remains non-parametric. Further, in order to prove effectiveness on small clusters, it only needs to be shown that volume and surface area estimates are close even if only a few points from the cluster are sampled. As we do not have access to the true density distribution, the density function must be approximated from the data. For this purpose, we will use a second nearest neighbour approximator.

Density approximation is traditionally done in two forms, k nearest neighbours and kernel methods. Both methods involve measuring the number of points in a region as an approximation of density. KNN typically constrains the number of points in the region to be fixed (k) and adjusts the radius until a large enough volume in covered. Conversely, kernel methods hold the region volume constant while counting the number of points in the space. For the purposes of the paper, we will use a second nearest neighbour estimator. For both approaches, say we are interested in a density approximation of points X, $f(X)$. Let $L(X)$ be a small neighbourhood of X. If $L(X)$ is sufficiently small, the probability mass of $L(X)$ can be approximated by $f(X)v$ where v is the volume of the local neighbourhood. The probability mass of the empirical region can also be approximated by sampling a large number of points from the distribution and counting the number that fall within $L(X)$. Let n points be sampled and let k of them fall within $L(X)$. Let $\hat{f}(X)$ be the desired empirical estimate of density at X.

$$\hat{f}(X)v = \frac{k}{N}$$
$$\hat{f}(X) = \frac{k}{Nv}$$

As was mentioned, KNN estimators work by varying v until k points fall in the local neighbourhood of X. However, the form of the estimator must be slightly modified as follows. Let v_r be the distance of the k^{th} nearest neighbour to X. Let v_r be the volume of the d dimensional ball of radius r. Again, let $\hat{f}(X)$ be the empirical density estimate.

$$\hat{f}(X) = \frac{k-1}{Nv_r}$$

As shown in [10], the reason $k-1$ is used instead of k in the numerator is that the former leads to an unbiased estimator. With this modified form, it can be shown that the bias of the KNN estimator goes to 0 as n goes to infinity. However, the variance of the estimator remains constant as n increases and only reduces by increasing k. If accurate pointwise estimates are required, it can be shown that k must be some function of n. Our needs do not face this shortcoming because we are not interested in accurate pointwise estimates; rather, we are interested in accurate integrals of the pdf over volumes of clusters or surface areas of cuts. While the proofs have not yet been completed rigourously, we believe that the large variance will integrate out as we consider larger spaces. For any clusters of non-zero measure, it should be possible to invoke Chernoff bounds and claim that if it had a sufficiently large probability mass, sufficient points must fall in the area to generate an accurate estimate.

4.1 Control volume methods

With a pointwise estimate of density in place, we are ready to construct a graph to partition in order to derive our clusters. For this purpose, a probability density distribution can be thought of as analogous to a heat distribution over a metal plate of varying conductivity. Regions of high density can be thought of as parts of the metal plate with high conductivity. Solving the heat equation over a space is hard to do exactly but can be approximated well by finite element methods. Finite element methods solve partial differential equations over closed sets by splitting the set into small units and solving the equation on unit boundaries. The total solution can then be extracted from these boundary solutions. In the 2D case, space is usually quantized by a triangulation.

In order to ensure an accurate approximation, it is important that the triangles be close to equilateral in some sense. This condition is achieved by bounding the aspect ratio for each triangle in the mesh. The aspect ratio of a triangle is defined to be the ratio of the radius of the circumcircle to the smallest edge in the triangle. Ruppert's algorithm [18] provides a mesh composed entirely of these high quality triangles. The algorithm consits of the following operations and invokes the Delauny Triangulation [4] as a subroutine.

1. Compute the Delauny Triangulation of the points.
2. Repeat the following two steps until convergence.
3. For each segment such that its dihedral ball is non-empty, add its midpoint to the triangulation.
4. For each triangle with bad aspect ratio, add its circumcenter to the triangulation.

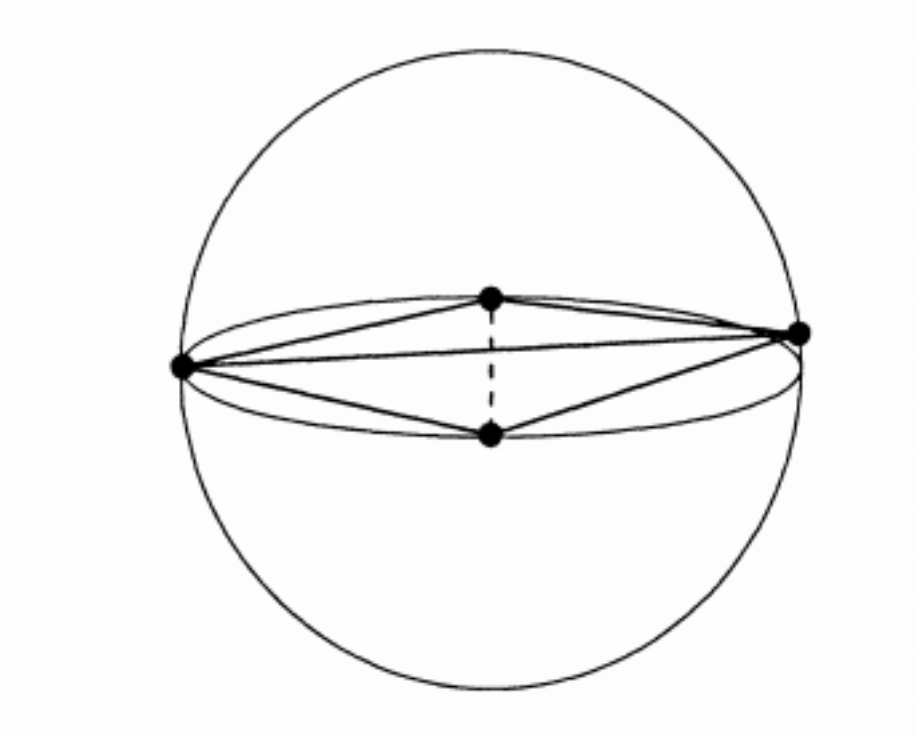

Figure 4.1: 3D slivers

In 3D, the mesh is generated by covering space with tetrahedrons. This, however, is plagued by ”sliver” tetrahedrons in which the four points are nearly coplanar. Finite element methods perform poorly in meshes containing these slivers despite their having good aspect ratio. An image of these slivers can be found in, [15]. The image is reproduced here in 4.1. [15] shows that the control volume method has a low error even in the presence of these slivers. Therefore, extensions of Ruppert's algorithm can be used to solve estimation problems even in three dimensions.

4.2 Constructing the mesh graph

The problem remains to construct a graph over the input data as opposed to simply over a bounding box. In order for the control volume method to be applicable, it is still necessary for the Voronoi cells in the mesh to have good aspect ratio. In lower dimensions, this implies good aspect ratio Delauny triangles. This is ensured by adding extra points to the data set called Steiner points. The Steiner points added can be shown to have the following properties.

Theorem 4.2.1. *For any data set D with n points, Steiner points D' added to ensure the triangulation only has good aspect ratio triangles will increase the size of the data set by at most a constant fraction.*

Theorem 4.2.2. *For any data set D with n points, Steiner points D' added to ensure the triangulation only has good aspect ratio triangles will not increase the second nearest neighbour*

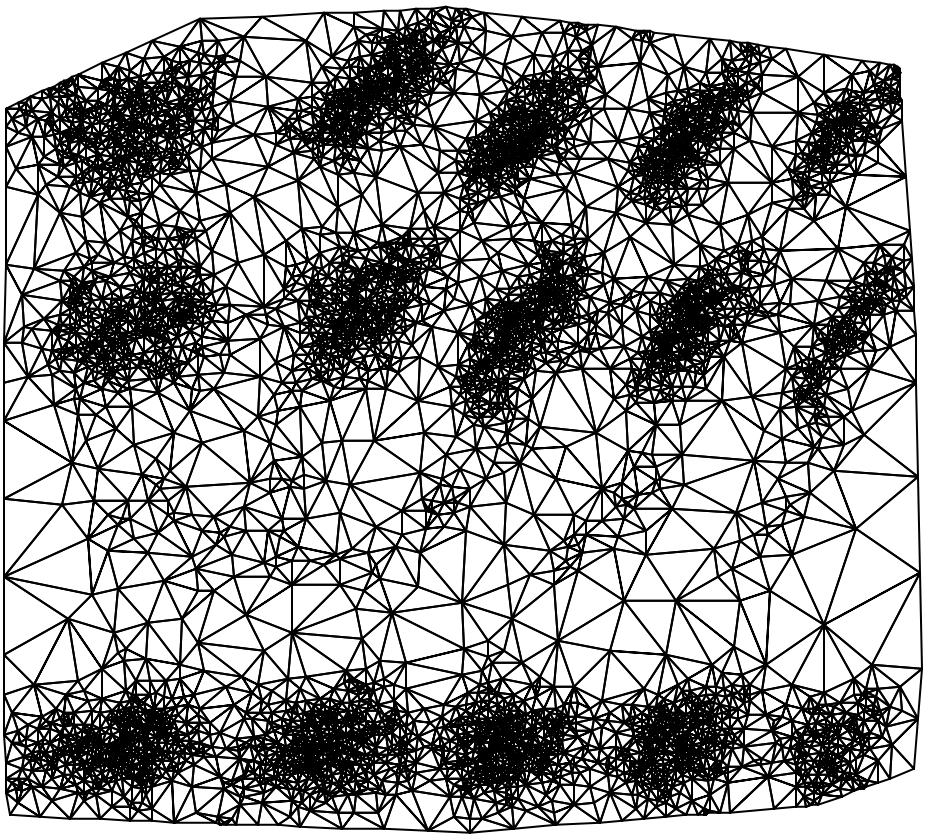

Figure 4.2: Delauny triangulation of Barcode data with Steiner points added

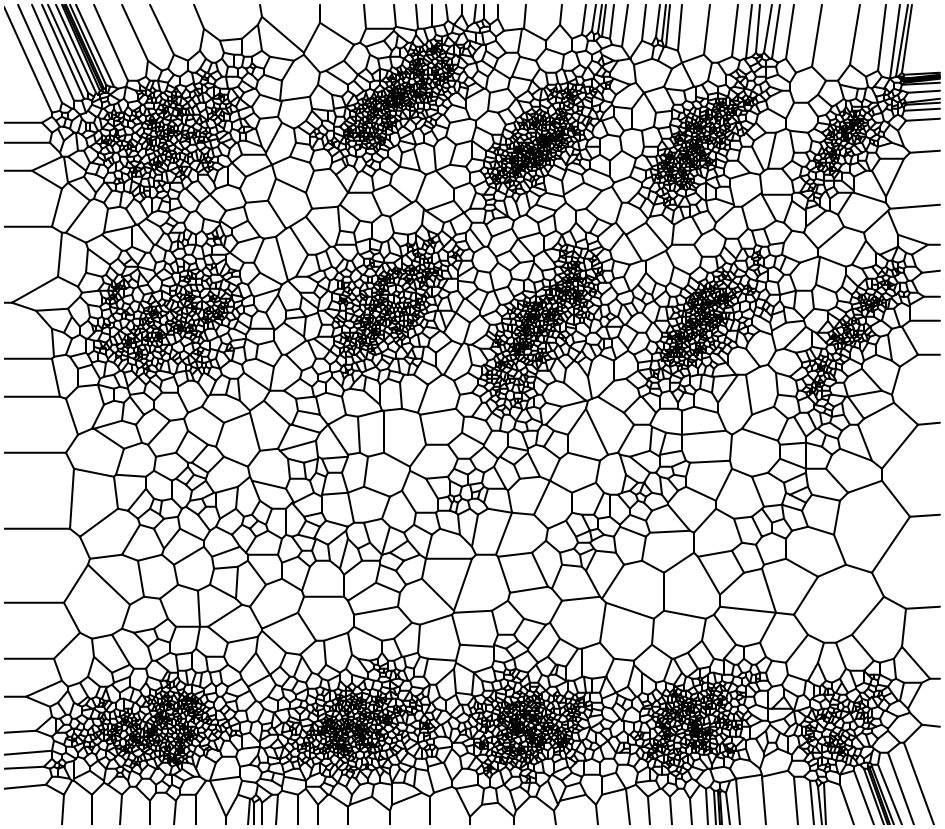

Figure 4.3: Voronoi diagram of Barcode data with Steiner points added

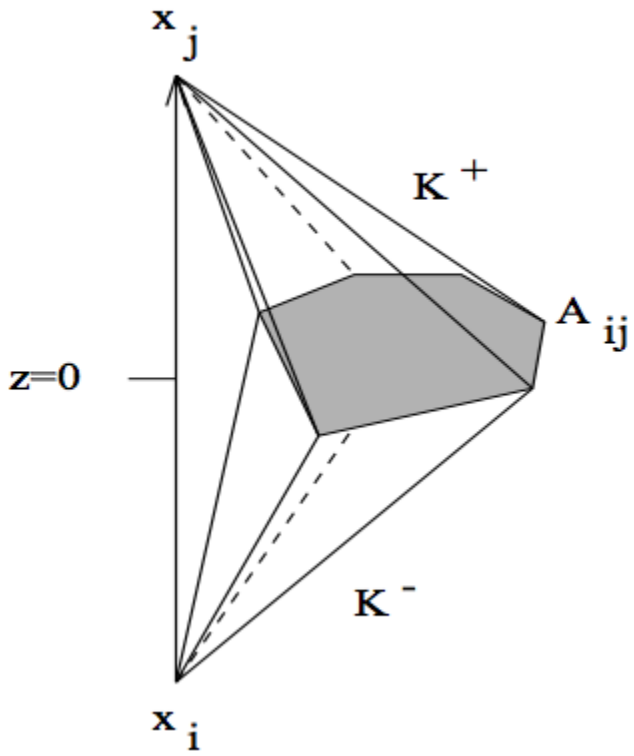

Figure 4.4: Kite between two points in three dimensions

distance of any point by at most a constant fraction.

Theorem 4.2.3. *For any data set D, consider any point x. Let Steiner points D' be added to enforce quality of the triangulation. Then the edge length from x to any of its neighbours will not differ by more than constant fraction.*

These theorems allow us to triangulate the data set and ensure that all triangles have good quality. The resulting triangulation graph is used as the affinity graph in order to cluster the data set. In order to estimate the cut, boundary conditions on this graph must be solved. Consider the figure 4.4 from [15]. This figure represents the connection between two neighbouring points in the triangulation, also known as a kite. The points are, without loss of generality, positioned along the z-axis.Let the points be x_i and x_j and the Delauny edge between then be h_{ij}. K^+ and K^- refer to sections of the Voronoi space of each point. A_{ij} is the Voronoi face between the two points. Let e_{ij} denote the weight of the edge in the affinity graph and h and A denote the length of the edge and surface area of the Voronoi face respectively. Let C_{ij} be the conductance at the mean of the Voronoi face. The Control volume method dictates that the edge weight of the Delauny edge is as follows.

$$
\begin{aligned}
e_{ij} &= \frac{A}{h} C_{ij} \\
&= \frac{A}{h} \frac{k-1}{n v_h} \\
&= C \frac{A}{h^{(d+1)}}
\end{aligned}
$$

For some constant C. This is because the conductance on the Voronoi face is simply the density estimate at that point. This estimate is equal to $\frac{k-1}{nv_h}$. Note here that $k = 2$ since we use second nearest neighbour distance. In d dimensional space, $v_h = C'h^d$ for some other constant C'. Combining these observations and separating the constants into a single term gives us the above form. Because this constant is independent of position and therefore edge independent, it may simply be dropped in assigning edge weights.

Chapter 5

Results

In this paper, we apply our clustering approaches to flow cytometry data. Cytometry refers to the measurement of cell characteristics such as size, count, DNA content, or the existence of certain bio markers. Flow cytometry in particular refers to the measurement of cell characteristics by aligning cells using flow techniques. Cytometry data provides valuable insight into cell populations and is used often in medical research and diagnostics. Its applications include providing diagnosis and prognosis information on diseases like HIV or leukemia. Clustering these cell populations is a standard first step in many pipelines using cytometry data. However, these data clusters tend to be highly irregular in shape. Further, rare cell populations are often vital to perform diagnostics. Therefore, traditional clustering methods such as K-means or Gaussian Mixture Models fail to produce satisfactory results.

The state of the art non-parametric clustering approach on this data is SamSPECTRAL [24]. SamSPECTRAL is also based around spectral clustering but constructs a complete graph over the data. Rather than attempt to reduce the size of the graph, the algorithm relies on a heuristic to sample a ”representative” subset of the data points. The graph is then built only over these representatives. The edge weight between points in the graph depend on the pairwise distance between all points assigned to each representative. This graph is then cut in order to cluster the data set. Finally, clusters with low separation are combined. While SamSPECTRAL performs well on cytometry tasks, it relies heavily on heuristics and provides no theoretical bounds on quality.

Flow cytometry data poses another challenge and that is that the output of a clustering algorithm is hard to judge. Typically, cytometry data is somewhat high dimensional. Standard cytometry data sets contain 70,000-150,000 cells per person with 16-20 different measurements per cell. Therefore, clusters must be judges in 20 dimensions. Even if dimensionality reduction techniques are used, the problem is sufficiently hard that expert judgements are required. Cytometry data is nuanced enough that non-experts cannot easily determine the number of clusters or the cluster positions while expert annotated data sets are hard to come by. For this reason, much of the research is done on ”toy” data sets to demonstrate feasibility of the approach. We present our approaches on two toy data sets. The first is a small set of non-linear data with a few rare populations. The second is a so called ”Barcode” data set that was presented in Section 3.4. The data is reproduced here without clustering labels. This data has been built by experts to resemble problems faced in clustering true cytometry data and is presented here [11]. However,

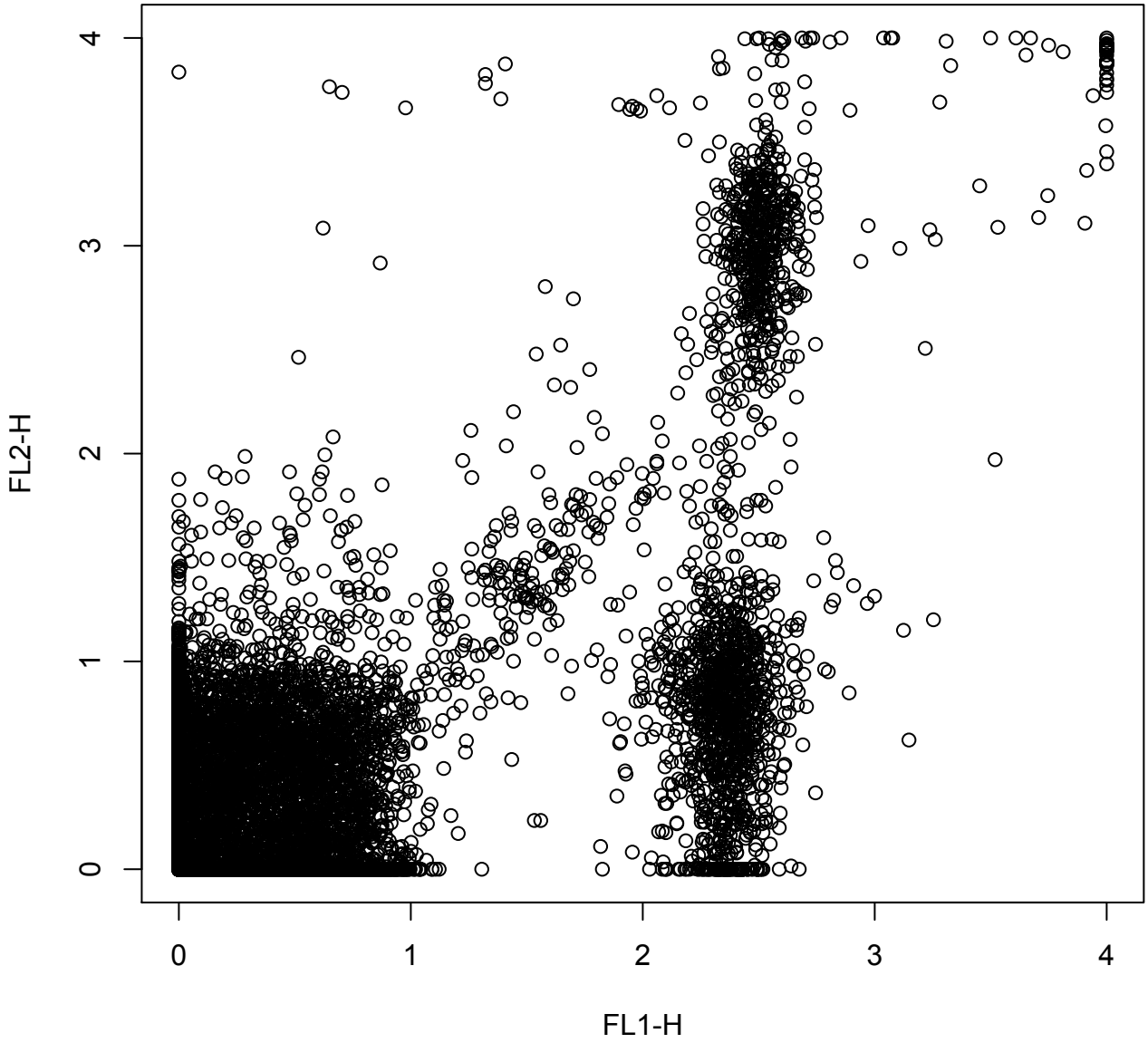

Figure 5.1: Small data set

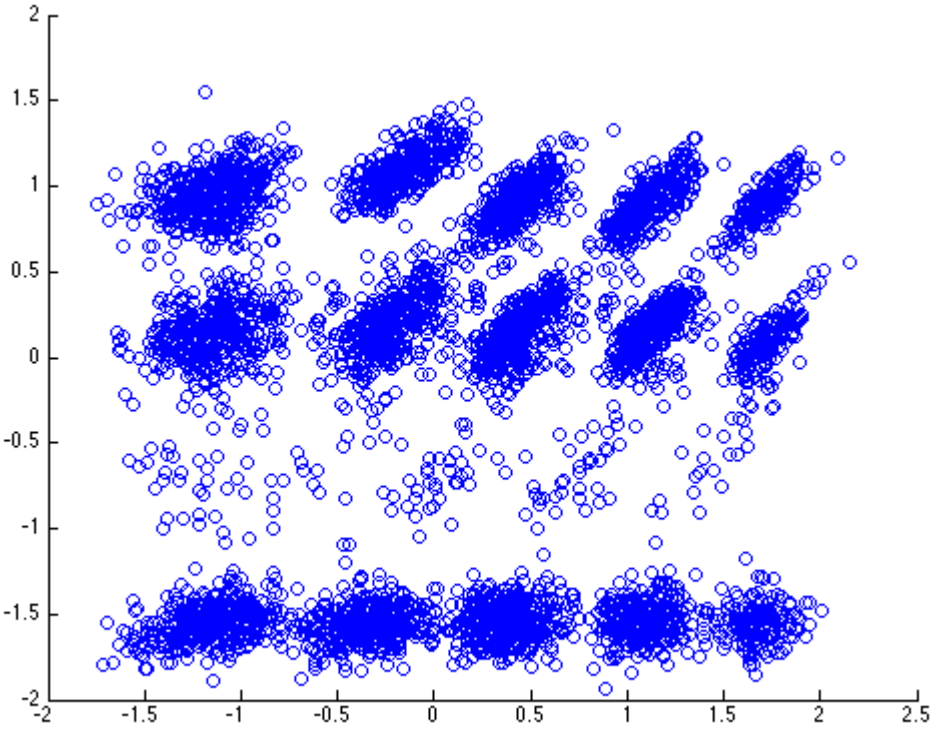

Figure 5.2: Barcode data set

the barcode data is easy for a non-expert to cluster on inspection.

In the small toy example, both SamSPECTRAL and our clustering approaches vastly outperform K-means. Geometric clustering performs about as well as SamSPECTRAL (depending on the designation of clusters) but is far less dependent on external parameters being set to the right values. In order for SamSPECTRAL to perform well, several parameters must be tuned through an exhaustive grid search.

The following pages contain the different clusterings for the toy data set and the barcode data set. 5.3 is the figure showing the results of K-means on the toy data set. 5.4 presents how SamSPECTRAL clusters the set, while 5.5 shows the clustering produced through a geometry based approach.

Similarly, **??** is the result of applying K-means to the Barcode data, 5.6 is the clustering produced by SamSPECTRAL and 5.7 is the clustering produced by the geometry based approach. Finally, **??**

For the Barcode data, we immediately notice that all approaches make mistakes. K-means performs the worst with several split clusters. SamSPECTRAL provides clear clusters but cannot differentiate between the several rare groups. It only detects the presence of 16 clusters instead of 20. Both Gabriel clustering and Geometric clustering give similar results. Geometric clustering has a huge win on speed in low dimensions because computing a good quality triangulation is very fast while computing the Gabriel graph is still super-quadratic time. Triangulation is done with the help of the Triangle package. [20].

These results show that, in 2D, a geometry based approach is both very fast and decently performant at identifying rare clusters. Spectrally clustering a mesh with Steiner points added performs similarly to SamSPECTRAL, a highly tuned, heuristic method. Further, meshing data sets in low dimensions is computationally cheap and produces only a linear sized graph. In higher dimensional problems, Gabriel graphs are cheaper to construct while still yielding good clustering results and a linear graph. Further, the geometric approach has a strong theoretical foundation from which proofs of quality could be derived. This is a good avenue for future exploration.

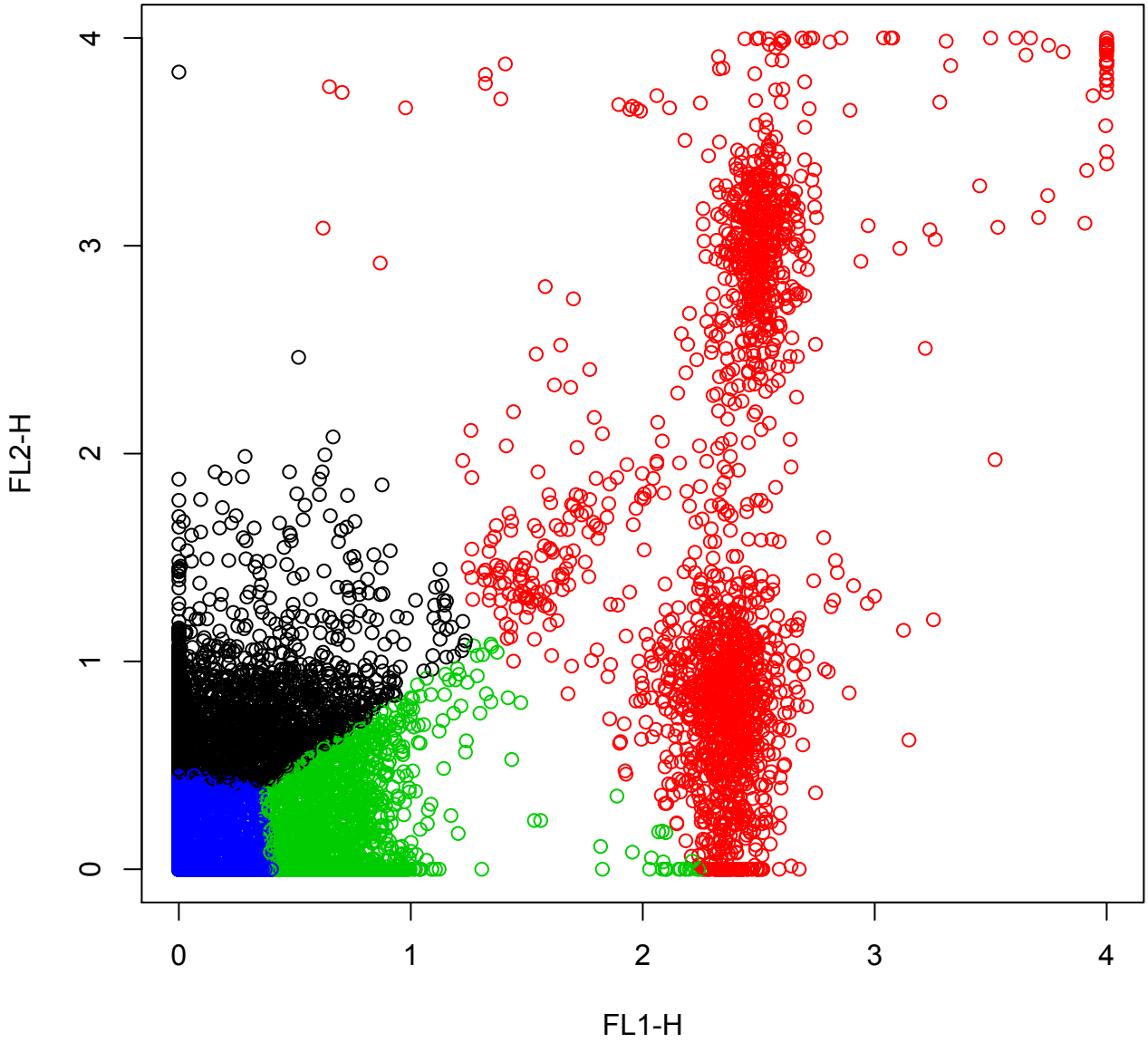

Figure 5.3: K-means on toy data set

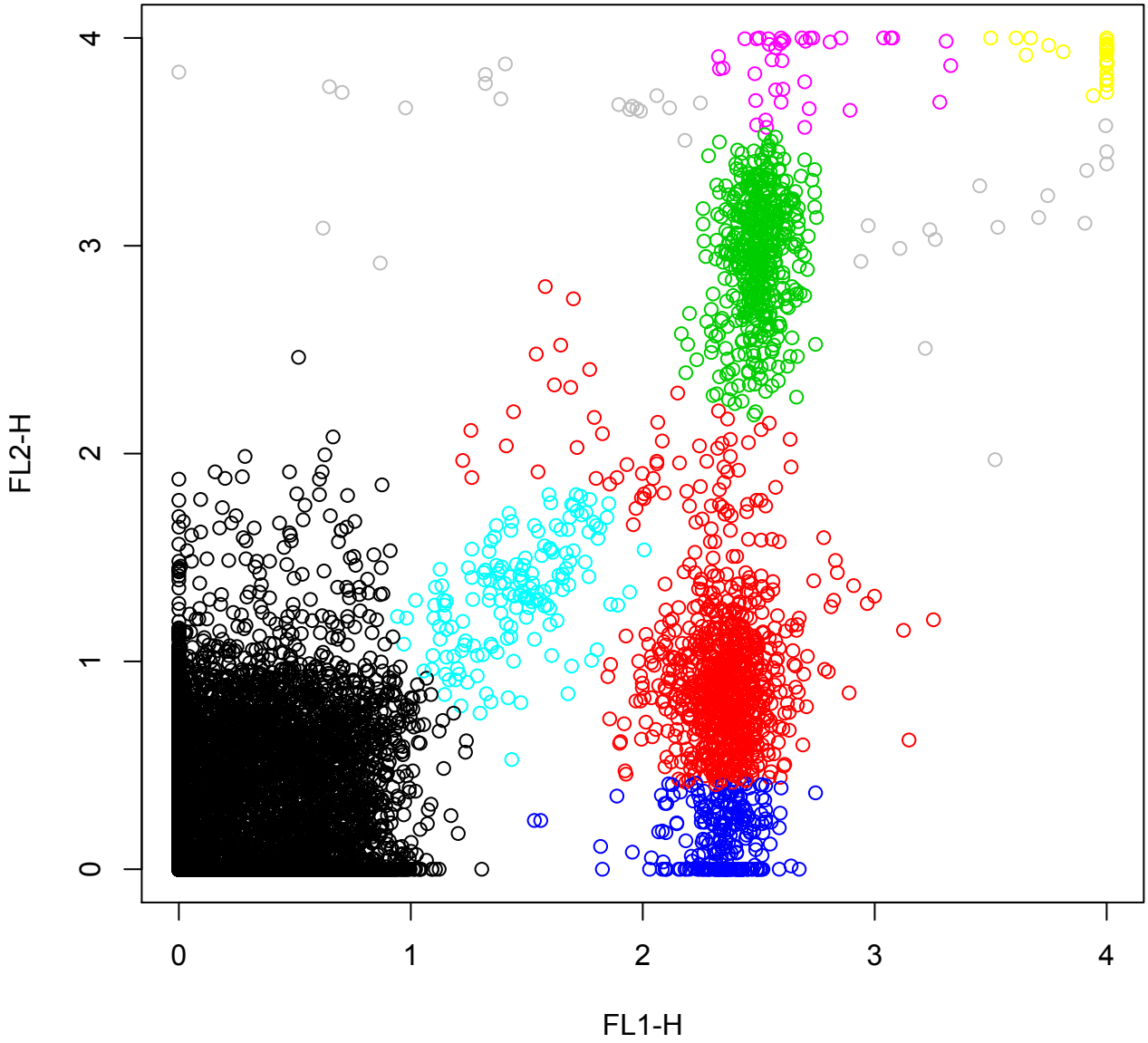

Figure 5.4: SamSPECTRAL on toy data set

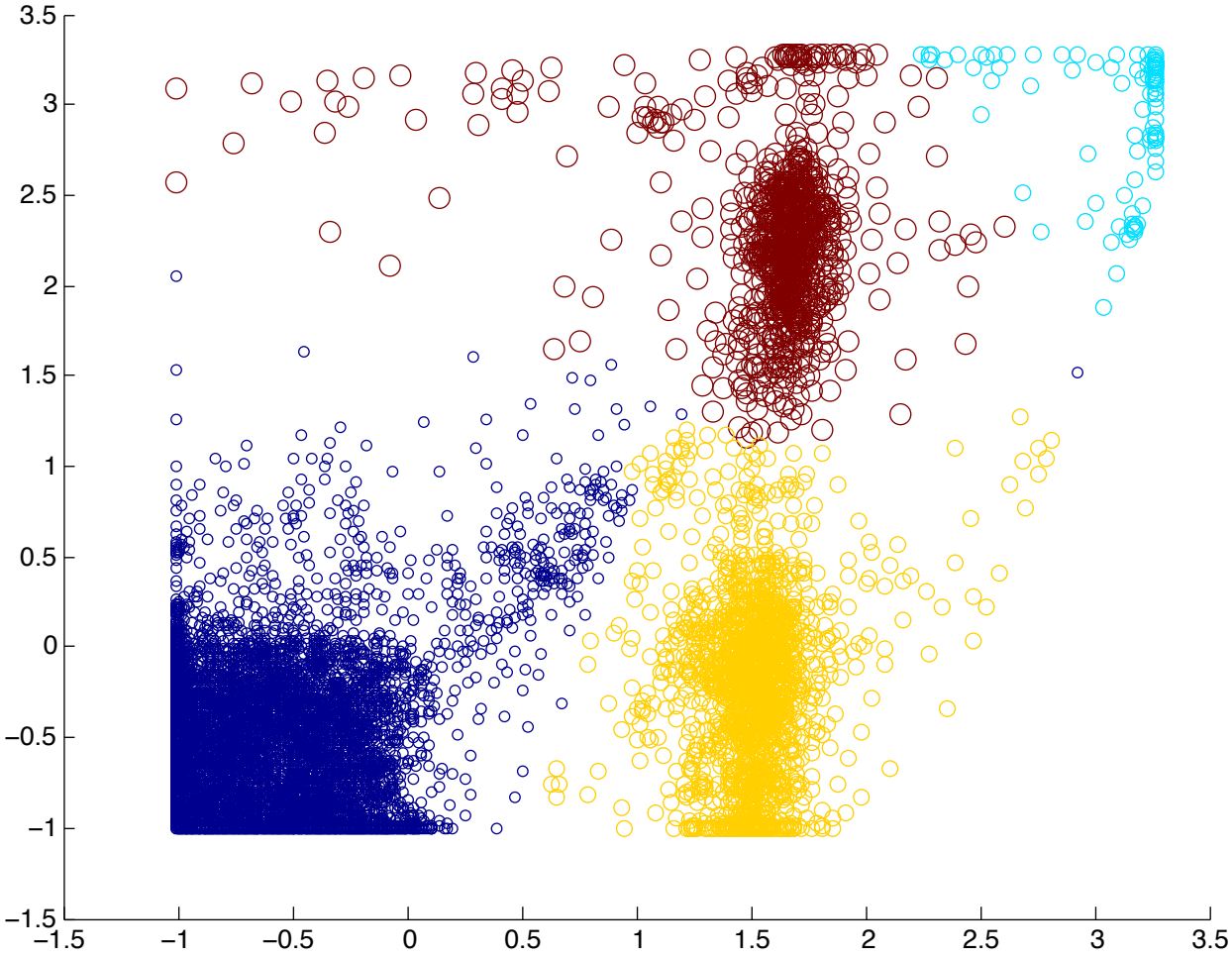

Figure 5.5: Geometry based clustering on toy data set

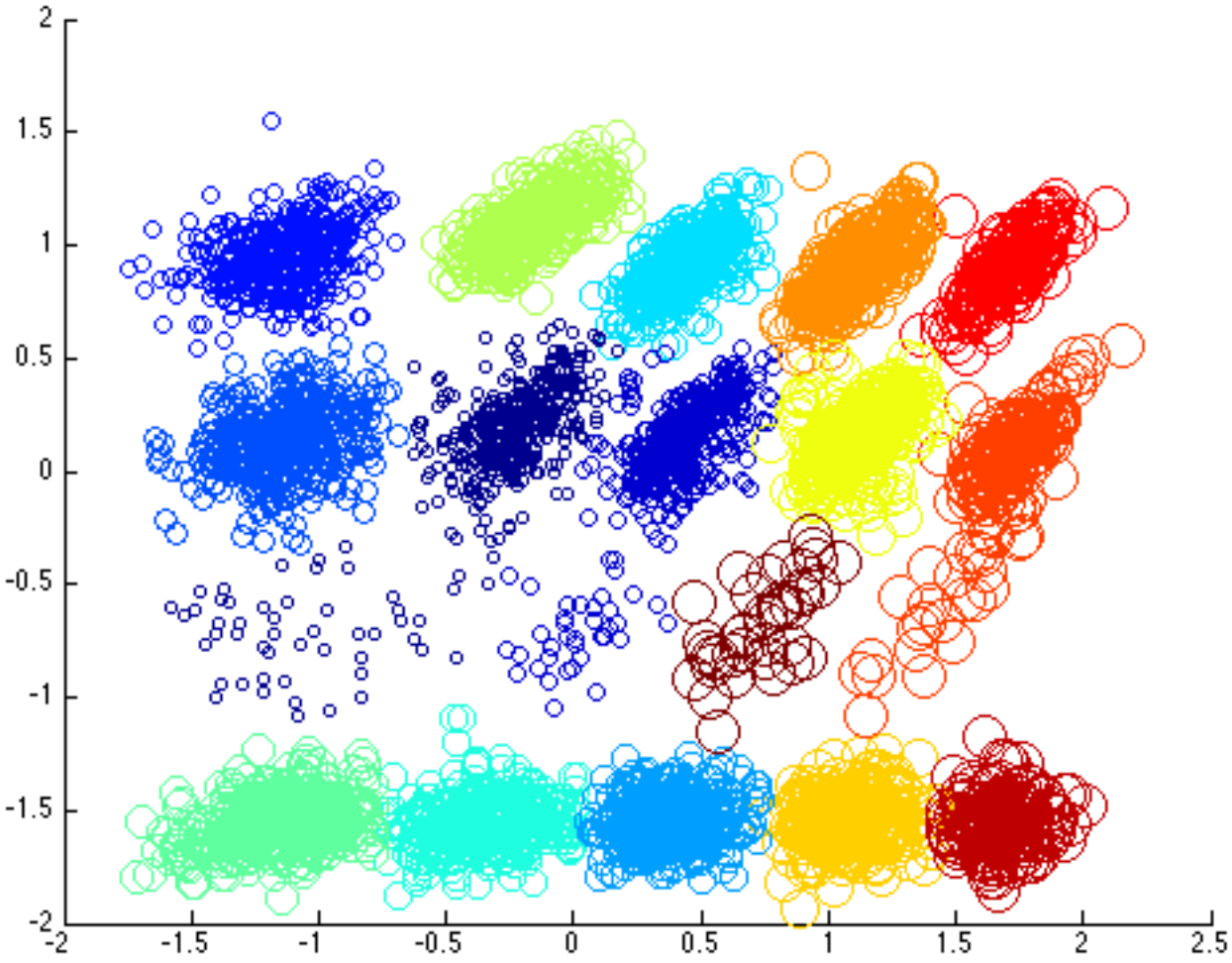

Figure 5.6: SamSPECTRAL on Barcode data

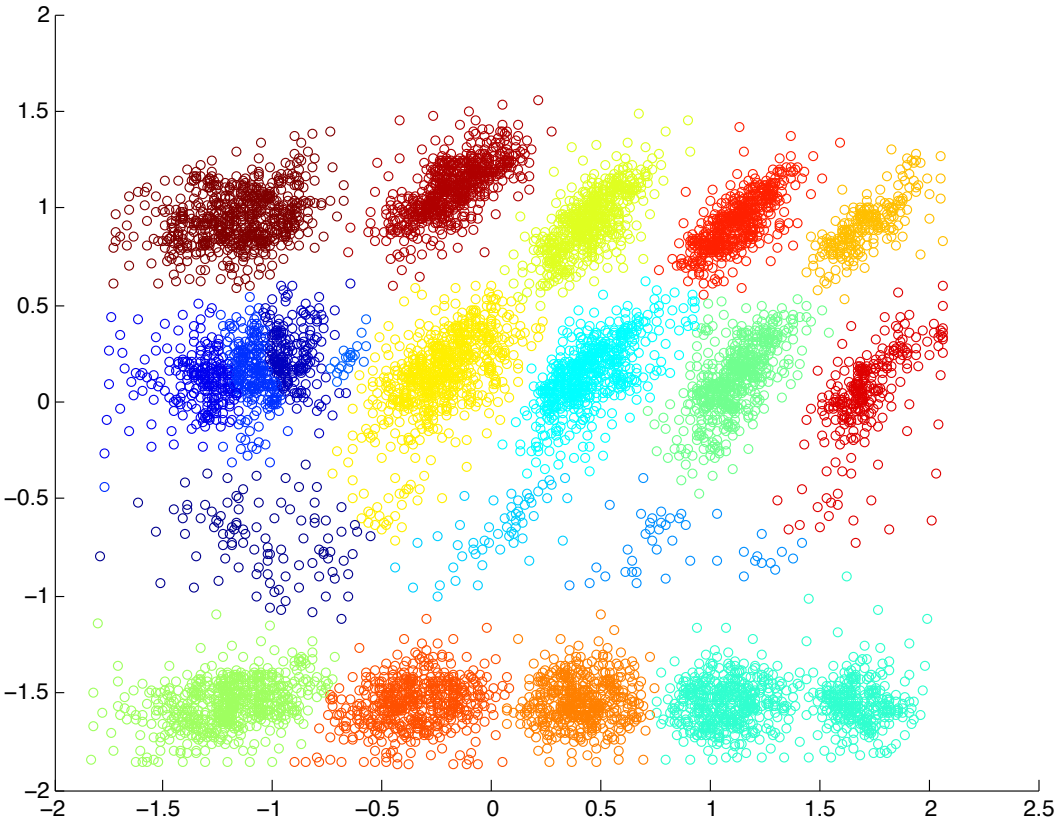

Figure 5.7: Geometric approach on Barcode data

Bibliography

[1] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, 625:195–199, 1970. 4

[2] Michael B. Cohen, Brittany Terese Fasy, Gary L. Miller, Amir Nayyeri, Donald Sheehy, and Ameya Velingker. Approximating nearest neighbor distances. *CoRR*, abs/1502.08048, 2015. URL http://arxiv.org/abs/1502.08048. 3

[3] Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse euclidean spanners. *International Journal of Computational Geometry & Applications*, 7(04):297–315, 1997. 3.2

[4] B Delaunay. Sur la sphere vide. a la memoire de george voronoi, 1934. 4.1

[5] Michael Elkin and Shay Solomon. Optimal euclidean spanners: really short, thin and lanky. *CoRR*, abs/1207.1831, 2012. URL http://arxiv.org/abs/1207.1831. 3.2

[6] David Eppstein and Joseph Wang. Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8:39–45, 2004. 3.4

[7] Mario AT Figueiredo and Anil K Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002. 1

[8] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977. 3.4

[9] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001. 1

[10] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic press, 2013. 4

[11] Yongchao Ge and Stuart C Sealfon. flowpeaks: a fast unsupervised clustering for flow cytometry data via k-means and density peak finding. *Bioinformatics*, 28(15):2052–2058, 2012. 3.4, 5

[12] Sung Jin Hwang, Steven B Damelin, and Alfred O Hero III. Shortest path through random points. *arXiv preprint arXiv:1202.0045*, 2012. 3.1

[13] L. Kaufman and P.J. Rousseeuw. Clustering by means of medoids. in statistical data analysis based on the l1–norm and related methods. 1987. 3.4

[14] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014. 4

[15] Gary L Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. On the radius-edge condition in the control volume method. *SIAM Journal on Numerical Analysis*, 36(6): 1690–1708, 1999. 4.1, 4.1, 4.2

[16] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009. 3.4

[17] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 413–422. ACM, 2014. 3.4

[18] Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *SODA*, volume 93, pages 83–92, 1993. 4.1

[19] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. 3.4

[20] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996. 5

[21] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000. 1, 4

[22] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, USA, 2006. ISBN 0123695317. 3.4

[23] Vladimir Ufimtsev and Sanjukta Bhowmick. An extremely fast algorithm for identifying high closeness centrality vertices in large-scale networks. In *Proceedings of the Fourth Workshop on Irregular Applications: Architectures and Algorithms*, IA3 '14, pages 53–56, Piscataway, NJ, USA, 2014. IEEE Press. ISBN 978-1-4799-7056-8. doi: 10.1109/IA3.2014.12. URL http://dx.doi.org/10.1109/IA3.2014.12. 3.4

[24] Habil Zare, Parisa Shooshtari, Arvind Gupta, and Ryan R Brinkman. Data reduction for spectral clustering to analyze high throughput flow cytometry data. *BMC bioinformatics*, 11(1):403, 2010. 5