Statistical Learning for Complex Systems an Example-driven Introduction

Greg Leibon[1]
Scott Pauls[2]
Dan Rockmore[3]
Robert Savell[4]

June 11, 2010

[1]Memento Security and Department of Mathematics, Dartmouth College
[2]Department of Mathematics, Dartmouth College
[3]Departments of Mathematics and Computer Science, Dartmouth College and The Santa Fe Institute
[4]Thayer School of Engineering, Dartmouth College

Chapter 1

Preface

Complex adaptive systems exist in many forms, but the main form that this book concerns itself with is as data, generally numerical or categoical, high-dimensional, and lots of it. When we set out to understand or explore a complex system, often what this means is sifting through the data in search of structure. This structure is often revealed in some notion of geometry in the data.

The goal of this little book is to give people who work in complex systems some sense of the tools and techniques that can be brought to bear on the kinds of data measured in a complex system. Examples include the families of time series that might summarize the activities of markets, economies, brains, or communication networks, the roll call votes of a legislature, the feature vectors of a corpus of drawings, or the words of a corpus of documents, or inputs and outputs of an economic network.

Broadly speaking, the kinds of tools that we are talking about are part of the world of statistical learning, sometimes called machine learning, or statistical learning, and it comes in unsupervised and supervised flavors. There are of course already many great books on learning, among which are [**?**, **?**, **?**]. This **little** book is not meant to compete with these. Rather, in an example-driven fashion it is meant to give the reader an idea of some of the basic tools at his or her disposal with a few case studies that might at least give a starting point when addressing some new data. Our hope is that this first handhold in the data will be enough for you to get your start and set you off on the fun work of really getting to know your complex system!

Chapter 2

Introduction

The phrase **complex adaptive systems** is often used as a catchall for many kinds of natural phenomena that vary across many different kinds of traditional disciplines. We'll be looking at tools that try to say something sensible and quantifiable about economies and ecologies, organisms and organizations, societies and sensory data, markets and minds. Mainly we'll make use of some combination of linear algebra, probability, statistics, network analysis, and computing (simulation and analysis).

Despite the obvious differences, many kinds of complicated living (in the sense of "evolved" and of the world) phenomena have important commonalities, commonalities that enable us to at least believe that we can find some kinds of general tools to study them and in so doing, find ways to link understandings of the different phenomena into some sort of whole. Commonalities allow for ideas to flow between different subjects.

These commonalities include

- Diversity and individuality of components
- Localized Interactions among components
- Emergent phenomena
- Hierarchical
- Multiple stable states
- Adaptation
- Hysteresis
- Natural Selection and Evolution
- Criticality

- Nonlinearities

Many complex systems can be regarded as dynamic networks with physical or abstract interactions. This opens the door to mathematical and numerical analysis. The dominant approach of the last decade has a physics flavor to it.

We'll now proceed to look at several examples of complex systems, generally with a network flavor, to get ourselves oriented to the subject, all the time keeping in mind this idea that we are aiming to bring geometry (and topology) to natural phenomena.

2.1 Example: Physical networks

We'll be trying to understand all kinds of phenomena that don't come with a geometry, but of course some really do! Physical networks abound, examples include river networks, neural networks, trees and leaves, physiological networks such as your pulmonary or circulatory system, computer networks (e.g., the internet or your local area network), the power grid, road networks.

So, what kinds of commonalities do we have in these systems? In fact, many! All sorts of branching and a branching that seems to exhibit aspects of regularity.

These examples help us to keep in mind that often networks provide a substrate enabling the delivery of some resources. In the case of physical networks, it is often a "nutrient" (e.g., blood, oxygen, water, food, electricity), but it can also be merchandise, capital, influence, ideas, disease, power. In some cases the network mainly works to deliver a resource from A to B (e.g., the veins in a leaf or a road system), while others may also serve to circulate a resource (e.g., our circulatory system).

2.2 Example: Social systems

In the 1990s there began a renewed interest in social networks, largely brought about by the quantification by Steve Strogatz and Duncan Watts of Stanley Milgram's famous six degrees of separation through the invention of the idea of "small world" networks [16]. Their work, which extended beyond the discussion of social networks to discuss also physical and biological networks, articulated various structural similarities in these networks that helped spur the vast amount of research that is taking place today.

That said, today, we are all well-aware of the notion of social networks, through entities such as Facebook or Linked-In or Myspace. Social networks are part of the data explosion. Different kinds of social networks codify different kinds of social relationships. They confer power, influence, access, and a host of other social resources. In Figure **??**[1] is a famous early social network, a "sexual relationship" network in an Indiana high school, studied by one of the great figures in early social network research, R. Moody. This network tells a social story, but also shows a substrate for the transmission of disease.

[1] This image is missing

2.3 Example: Voting network

Does a legislative body have a geometry? Well, it has a color, or rather colors, if we think of the usual Democrat/Republican Red/Blue (with a sprinkling of Green denoting independents) coding of modern American politics, but this is just one aspect of a legislator's "ideology." So in trying to create a geometry of a legislative space, we're getting at the geometry of ideology. Political scientists call this the "action space," whose components might include party affiliation, regional identification, religion, morality (itself a multidimensional thing), etc. [**?**] How can we get at measuring these sorts of abstract traits? One method is to look at the "data trail" of the voting record as some sort of proxy for all of this.

That is, what we can "measure" is the voting record - the so-called record of **roll call** votes. In this case we have legislators $L_1, \ldots, L_n$ where the action on roll call vote j is $L_i(j)$ and

$$L_i(j) = \begin{cases} 1 & \text{aye} \\ -1 & \text{nay} \\ 0 & \text{otherwise.} \end{cases}$$

In this sense, a legislative body comes with a measured dimensionality equal to the number of votes taken by the body of a session. This is a very rough estimate of the dimensionality. Political scientists use a variety of techniques, contained broadly in the field of factor analysis, to better estimate the dimensionality of the data. Almost all of these methods are model based - positing a rubric by which legislators vote and fitting the parameters of the model to available data. The most popular family of such models are the NOMINATE models developed (initially) by Poole and Rosenthal [13]. As an example of our explorations, we will use a different technique called *partition decoupling* to analyze roll call data. Figure 2.1 is an example of this technique applied to create a dimension reduced version of the 88^{th} U.S. Senate.

2.4 Example: Netflix Challenge

In 2003 Netflix (the movie-on-DVD delivery company) put forth a challenge. They would provide a dataset of customer preferences (movie choices and their ratings) so that researchers could work on coming up with a "better" recommendation system than was/is currently in place at Netflix. The winner would not only be able to publish a pretty good paper, but would also receive $1M.

From our geometric point of view, the goal would be to articulate a sensible notion of distance in the movie ratings network so that two customers who were close together would be likely to enjoy the recommendations of one another. Network - movie renters and their ratings. What is the correct notion of distance to enable a good recommendation system?

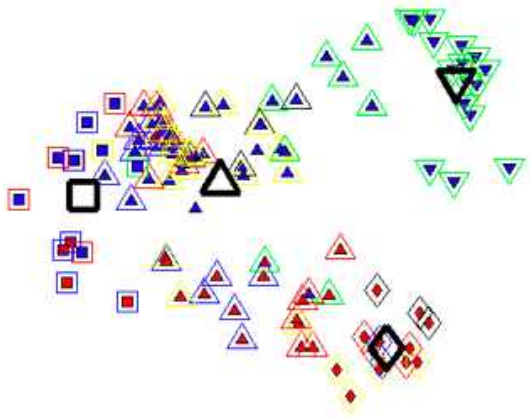


Figure 2.1: Two-dimensional embedding of the 88th Senate (1968). Node colors indicate party affiliation (blue=Democratic Party, red=Republican Party), shape (square, triangles and diamond) label the senators by cluster and the color of the larger outline denotes geography (red=midwest, blue=northeast, green=south, black=southwest, yellow=west)

2.5 Example: Microarray Data

A microarray is effectively a multidimensional sensor able to measure the levels (so-called expression levels) of many proteins in your body simultaneously. This vector then gives a multidimensional chemical summary of an individual. Microarray data is thought to provide insight into the interaction of genes and multigenic phenotypes, especially complex disease. Once again, we seek to find tools to articulate the geometry of this data space, thereby uncovering the geometry of disease and even (from the point of view of genetics) humanity. It turns out that once again, the partition decoupling method [8] is useful for this purpose [2].

2.6 Example: Literature and the arts

Some, if not most, works of art are an example of emergence. Individual words, taken one at a time, can evoke very little, but when combined according (or even in opposition) to a set of rules (some explicit, some not so explicit), sentences are formed, then paragraphs, and so on, and a work, a story emerges. Perhaps this reductionist view goes down even to the level of the phoneme. Similarly, a drawing or painting emerges from a collection of marks on paper or some other medium. Here the "atoms" are not so obvious. Digitization hands us the pixel basis (neglecting issues of digital capture, which for paintings can be significant). In both cases, we can ask sensible questions regarding the measurements appropriate to distinguishing style and/or content – i.e., the measurements that might articulate a useful notion of geometry in the world of art.

As it turns out, in the case of literature the easy-to-capture measurements of word frequencies (and their close derivatives) are quite informative to this end (see e.g., [4]). For images we need to do more work. That

is, the pixel space seems to need to be dimension-reduced via a change of basis akin to Fourier analysis. In this representation a coordinate system is achieved that has useful discriminatory power. Figure 2.2 shows the result of using multidimensional scaling (MDS) on a *wavelet decomposition* of a collection of drawings, some of which are known to have been executed by the famous Flemish draughtsman Pieter Bruegel the Elder (1542–1596) while the remainder were, at one time, thought to have been [10].

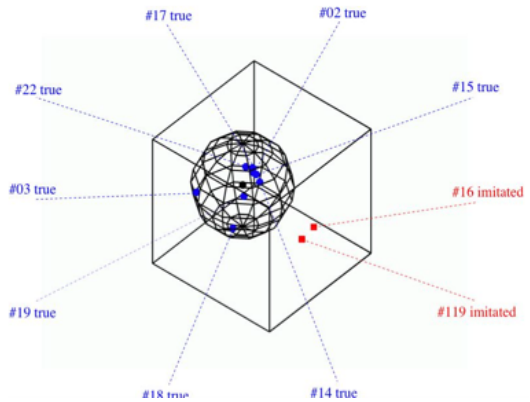


Figure 2.2: Three-dimensional embedding of a distance matrix obtained from "clouds" of feature vectors derived from drawings attributed to Pieter Bruegel the Elder (created by P. Kostelec).

2.7 Example: Geometry of thought

Functional magnetic resonance imaging (fMRI) is the method by which is measured the average oxygenated blood flow through each of a collection of small brain volumes ("voxels" – the 3D version of pixels) whose union makes up the brain. Neuroscientists believe that regional increased blood flow is a proxy for activity (the so-called BOLD Hypothesis) and that understanding the neural process is directly linked to understanding "patterns of activity" in fMRI data. A voxelization of the brain might amount to the designation of over one million measurement positions, each of which will index a time series of several hundred or thousand points. Thus, the articulation of patterns of voxel activity is a tricky and computationally challenging business. For example, the discovery of unusual activity would seem to entail the understanding of the resting state of the brain – a problem which has only recently begun to be studied. Even given that, there are (presumably) multiple layers of activity, blood shifting about to support both physiological as well as teleological purposes. Then we have the added challenge that the voxelization of the brain overlays a simple-minded geometry on top of the complex geometry that is the neural sheet – neighboring regions in voxel space need not be neighbors in physiological space. Our goal is to understand the interplay of these geometries.

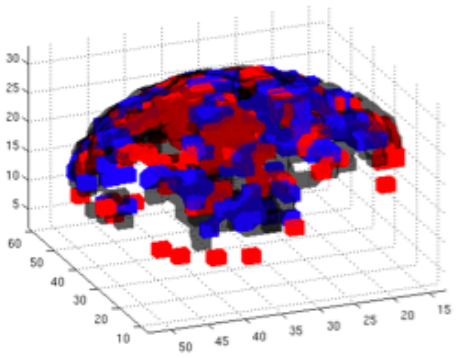


Figure 2.3: Clustering of fMRI time series, labelled by cluster, embedded in three dimensions with the given (aggregated) voxel geometry.

2.8 Example: Foodwebs

Ecosystems are summarized by their foodwebs, networks that encode the collection of predator-prey relationships between the species in the ecosystem. The geometry of this network and related patterns of dynamic allocation of energy resource (the shifting of biomass via feeding) say much about the health of an ecosystem.

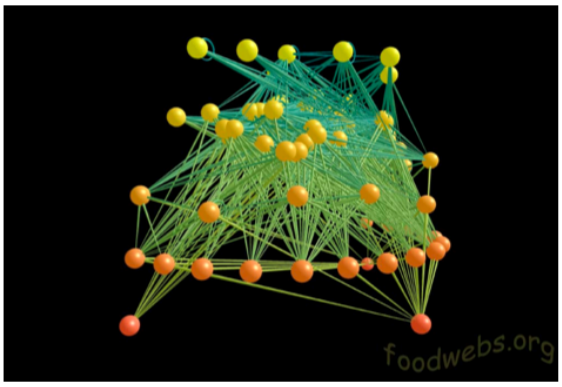


Figure 2.4: A Caribbean reef foodweb. Nodes represent species in the reef and links indicate a predator-prey relationship. From, Dunne, J.A., R.J. Williams, N.D. Martinez. 2004. Network structure and robustness of marine food webs. *Marine Ecological Press Series*, vol. 273, pp. 291-302.

2.9 Example: Economic networks

Economies and ecologies seem to have much in common, at least on the level of analogy. The predator-prey relationship of an ecology can be analogized to the feeding relations between industries as they provide resource to one another for the purposes of production. This industrial feeding relationship is summarized in the I/O (input/output) tables of the economy. Our goal is to understand these geometries so as to better understand the linkages and vulnerabilities in our economy and even the world economy. In this case, the underlying data is effectively a weighted directed network.

2.10 Example: Financial networks

This will be an important example for us. Like brain data this will be encoded in a large family of time series of significant length (on the order of a thousand time points) from which we derive a correlation network worthy of study. In particular, we consider the correlation structure of the combined NYSE and NASDAQ equities markets (which in this case is about 2500 entities), which we view both as embedded in the larger

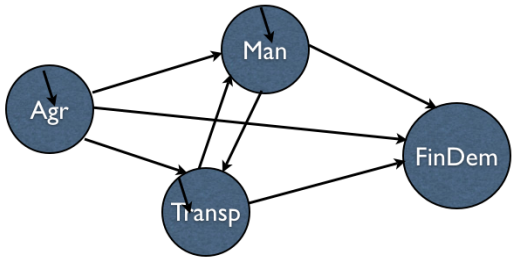


Figure 2.5: Cartoon input-output economic network. Directed and weighted edges indicating the fraction of a unit's worth of output from industry i required to produce a unit of industry j.

network of the global markets of all kinds of instruments (see Figure 2.6) as well as encompassing a variety of scales and layers of activity. Figure 2.7 gives a two-dimensional embedding of the NYSE/NASDAQ correlation network with labels for the discovered clustering.

2.11 Closing thoughts and organization of the book

The conjunction of these seemingly disparate examples suggest some intriguing analogies. For example many of these networks can be viewed abstractly as a geometry of resource flow. In ecosystems we have the movement of biomass according to the relations represented by feeding relations between species. In short we can view an ecosystem as a particular kind of resource processing unit. Therein we can find trophic levels, considerations of the role of exogenous effects (sunlight or pollution), a notion of keystone species, and the need for diversity. Analogously, consider the brain – it is also a processing unit with high connectivity, localized functionality, in which we monitor oxygenated blood flow (the resource) as a proxy for functionality. We of course also note that its highly adaptive and with great redundancy in its working parts. Similarly, consider metabolism as a resource processing device, societies, economies, markets as information processing entities.

In what follows we first review the necessary network theoretic and linear algebra concepts and then move to some basic clustering theory and practice, a discussion of the partition decoupling method (PDM) [8] a particular kind of statistical learning tool that seems to be useful for the analysis of a variety of complex systems phenomena. We then move on to the consideration of the geometry of Markov processes. We follow with a survey of a number of other useful learning tools. We close with a reading list. Along the way we include many examples and some homework problems, generally tied to real examples and datasets.

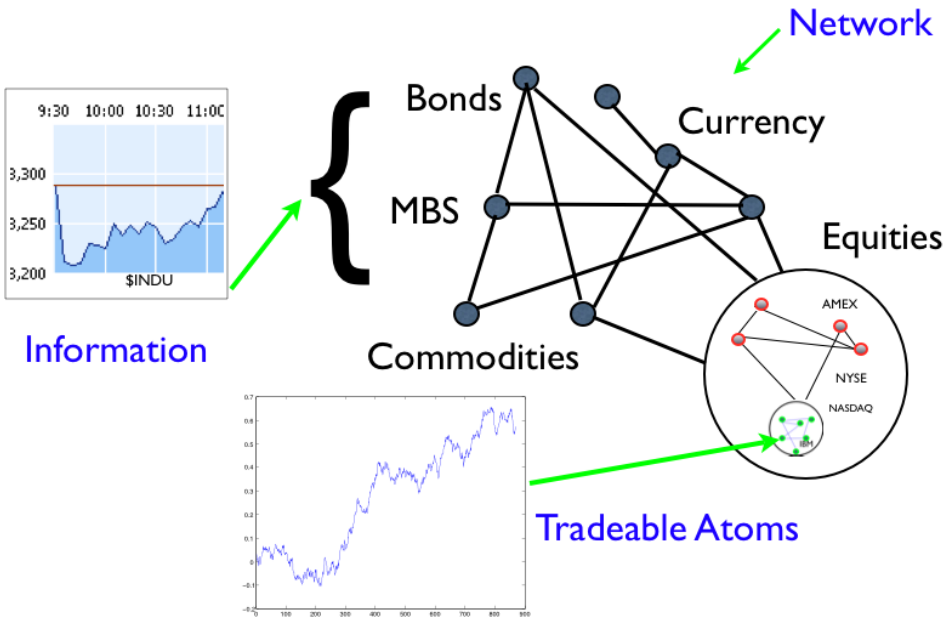


Figure 2.6: Cartoon Diagram of the Full Market. Note that the equities market does not exist in a vacuum. It is clearly influenced by the activities related to the other markets.

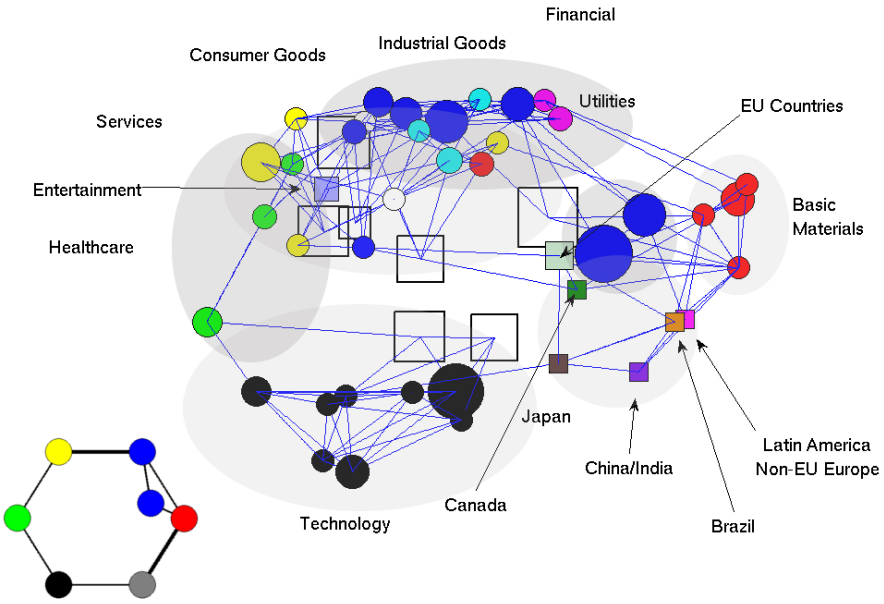


Figure 2.7: A two-dimensional embedding of the combined equities of the NYSE and NASDAQ. Colors (with the associated labels) represent dominant sector or geographical labeling of the clustered equities.

Chapter 3

Some Background

3.1 Networks beginnings

In this chapter we present the necessary backgrounds for networks and linear algebra. Thorough references for these topics are [**?**] for networks and [**?**] for linear algebra.

Basic definitions

Networks are made up of **nodes** or **vertices** that are connected by **edges.** A network can be **directed**, **undirected**, or **weighted** (the last of these effectively includes the previous two by allowing the weight on edge to be negative and thus give an orientation to the edge).

Note: it is worth thinking about the fact that a network is essentially a one-dimensional object (a collection of lines). Are there higher dimensional objects that might encode more complicated notions of relatedness?

Adjacency Matrix: Networks are encoded by an **adjacency matrix**. Rows and columns are indexed by the node labels $\{1,2,\ldots,n\}$. If A is the adjacency matrix for a network Γ, then A_{ij} encodes the "relatedness" of i and j as described by the network as follows:

1. Undirected (and unweighted) case:

$$A_{ij} = \begin{cases} 1 & i \sim j \\ 0 & \text{otherwise} \end{cases}$$

   These are the simplest and most basic kinds of networks. The fact is that most network analyses reduce to this case. In this case the adjacency matrix is binary and symmetric. In this definition, we use the notation $i \sim j$ to indicate that node i and j are connected by an undirected edge. This is a binary, symmetric matrix, possibly with some 1's on the diagonal.

2. Directed (unweighted) case:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge } i \leftarrow j \\ 0 & \text{otherwise} \end{cases}$$

3. Weighted case:

$$A_{ij} = w_{ij}$$

for the edge **from** j **to** i.

Note that if we allow for all possible weights between nodes, then in this case the adjacency matrix could be anything. Often, given the implicit orientation in the subscripts (A_{ij} means that the edge goes **from** j **to** i) then we can usually add the constraint that all weights are positive and thus the matrix A is positive. However, this need not be the case. An important class of weighted matrices allow for values of 1 and -1 [**?**].

Often, we have the case that in addition to being positive, the weight matrix is **symmetric** ($A = A^T$), so that the network is effectively undirected and weighted. This has significant implications for the spectral analysis of A, as we will recall below.

Degree. In the undirected case, we are familiar with the notion of the **degree** of a vertex. It is simply the number of edges connected to the vertex. We can write this as

$$degree(j) = \sum_i A_{ij}.$$

Note that this definition makes sense generally. Thus, we adopt this definition for the weighted, symmetric case as well. In the case of a directed network, weighted or not, we have a slightly expanded notion of degree, including a notion of **in-degree** and **out-degree**:

$$\begin{aligned} \text{out-degree}(j) &= \textstyle\sum_i A_{ij} \\ \text{in-degree}(j) &= \textstyle\sum_i A_{ji} \end{aligned}$$

Network data comes in many flavors and often, given a network that is of the most general kind, say weighted and directed, then we often try to look at it in various simplified forms, by (1) symmetrizing and (2) thresholding. Symmetrizing makes the matrix undirected – this accomplished by finding some way to create a symmetric weight between the nodes and then thresholding can take the symmetric weight and set this equal to either 1 (if the weight is above a threshold) or 0 (if the weight does not exceed the threshold) to turn the network into a simple undirected network. Note that one can also threshold the original directed weights to obtain a simple directed network. Here are a few examples:

1. **World Trade Web.** Considered the example of the World Trade Web (see e.g., [3]). Here the given data w_{ij} is the total U.S. dollars in exports from country j to country i.

   One could choose a given export value and threshold the network according to that, obtaining a directed network that only shows "significant" trade. Here are a few ways to symmetrize:

   (a) $c_{ij} = (1/2)[w_{ij} + w_{ji}] \qquad C = (1/2)[A + A^T]$

(b) $c_{ij} = \max\{w_{ij}, w_{ji}\}$

but there are others! After symmetrizing, you could then threshold to create a sparser matrix if desired.

2. **International Telephone Calls.**

3. **Market (NYSE) Data.**

Connections to Markov Chains

For many networks it makes sense to think of it as giving the underlying structure/geometry of a Markov chain. Recall that a Markov chain is defined by its **transition matrix** which is a **stochastic matrix,** a $(n \times n)$ matrix P such that p_{ij} is the probability of transition from state j to i, so that $p_{ij} \geq 0$ and $\sum_i p_{ij} = 1$. Given a state distribution π (so $\pi(k) =$ probability of being in state k) then the distribution after one step is $P\pi$ and after k steps is $P^k\pi$. An equilibrium state is a distribution π such that $P\pi = \pi$, so that π is a an eigenvector with eigenvalue 1 and all nonnegative entries. Note that if P is regular[1] (so that there is some $k > 0$ with $P^k > 0$) then there exists a unique equilibrium distribution.

Given a directed network, then by normalizing the edge weights (e.g., dividing the weights of all outgoing edges by the total out-degree) we obtain a Markov chain. In the case of an undirected weighted network we often associate a Markov chain simply by viewing each undirected edge as standing for edges in both directions with the associated weight. This is one way in which we might view a network as encoding dynamics.

Degree Distributions

Degree gives the first source of a statistic to consider in thinking about networks. The **degree sequence** of a network is the list of degrees in non-decreasing order. From this we can compute the **degree distribution** of a network. In the simple case of an unweighted network this is simply the probability distribution such that $P(k)$ is the fraction of nodes of degree k. For the weighted case, it is more natural to consider the associated cumulative distribution,

$$F(x) = P(degree \leq x) \qquad \text{or} \qquad 1 - F(x) = P(degree > x).$$

3.2 Spectral Analysis

The Markov theory re-introduces us to the use of eigenvector analysis for networks. Generally speaking, eigenvector/eigenvalue analysis is called spectral analysis. As a quick reminder, for an $n \times n$ real matrix A,

[1] This is equivalent to the existence of a k such that all states are reachable from any state in at k steps. Note that this is not equivalent to simple reachability.

a vector $\mathbf{v} \in \mathbb{R}^n$ is an **eigenvector** for A with **eigenvalue** λ if

$$A\mathbf{v} = \lambda\mathbf{v}.$$

Note that λ is in general in $\mathbb{C}$, but for most of our applications we will be able to ensure that $\lambda \in \mathbb{R}$. Note that this is equivalent to the property that $\mathbf{v} \in Nul(A - \lambda I)$, the nullspace of $A - \lambda I$, which is also —em the λ-eigenspace of A.

The eigenvalues for A are the roots of the characteristic polynomial

$$char_A(t) = det(A - tI)$$

where I is the $(n \times n)$ identity matrix. The set of eigenvalues of A is the **spectrum** of A. The multiplicity of a root λ of $char_A(t)$ is the **algebraic multiplicity** of λ, while the **geometric multiplicity** of λ is $dim(Nul(A - \lambda I)$.

The matrix A is **diagonalizable** if:

1. For each eigenvalue the algebraic multiplicity equals the geometric multiplicity;

2. There exists a basis of $\mathbb{R}^n$ of eigenvectors for A. In this case, if $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are the eigenvectors for A with eigenvalues $\lambda_1, \ldots, \lambda_n$, then we have

$$A(\mathbf{v}_1|\ldots|\mathbf{v}_n) = (\mathbf{v}_1|\ldots|\mathbf{v}_n)Diag(\lambda_1, \ldots, \lambda_n)$$

where

$$Diag(\lambda_1, \ldots, \lambda_n) = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_1 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{pmatrix}$$

and $(\mathbf{v}_1|\ldots|\mathbf{v}_n)$ denotes a matrix with the columns given by the $\mathbf{v}_i$.

Since the $\mathbf{v}_i$ form a basis the matrix $V = (\mathbf{v}_1|\ldots|\mathbf{v}_n)$ is invertible, so we can write

$$V^{-1}AV = Diag(\lambda_1, \ldots, \lambda_n).$$

Note that in this case, the matrix V effects a **change of basis** for the matrix A (given with respect to some initial basis), re-expressing the effect of A on $\mathbb{R}^n$ in terms of the basis given by $\mathcal{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, i.e., if given a vector in $\mathbb{R}^n$ in coordinates relative to $\mathcal{B}$, then multiplying that vector of coordinates by $V^{-1}AV$ will give the effect of the linear transformation determined by A (originally considered as expressing a linear transformation relative to the standard basis).

We are particularly interested in the case of **symmetric** matrices. Recall that in that case, not only is A diagonalizable, but even more, the eigenvalues must all be **real** and we can find an **orthogonal** and

hence, **orthonormal**, basis of eigenvectors. This fact is equivalent to the **spectral theorem** (for symmetric matrices), which defines the **spectral decomposition** of a real symmetric matrix

$$U^t AU = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

for

$$U = (u_1|u_2|\dots|u_n)$$

so that $Au_i = \lambda_i u_i$ where $u_i \in \mathbb{R}^n$ and $\langle u_i, u_j \rangle = u_i \cdot u_j = u_i^t u_j = \delta_{ij}$, so that $U^t U = UU^t = I$ (i.e., U is an orthogonal matrix). The spectral decomposition of A is then

$$A = \lambda_1 u_1 u_1^t + \cdots \lambda_n u_n u_n^t.$$

Note that for any $v \in \mathbb{R}^n$, $u_i u_i^t v$ computes the projection of v onto u_i and $u_i u_i^t$ is the associated projection matrix.

Rayleigh quotient and the Rayleigh-Ritz Theorem[2]

In the case of a symmetric matrix, A, there is also another characterization of eigenvalues via a maximization criterion. Since the eigenvalues will all be real we can order them

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

The Rayleigh-Ritz Theorem states that the eigenvalues of A are the critical points of

$$R(x) = \frac{\langle \mathbf{x}, A\mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle}$$

where $x \in \mathbb{C}^n \setminus 0$. $R(x)$ is called the **Rayleigh quotient** (with respect to A and $\mathbf{v}$). As a consequence have the following fundamental fact:

$$\lambda_1 = \max_{\mathbf{v} \neq \mathbf{0}} \frac{\langle \mathbf{v}, A\mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} = \max_{\|\mathbf{v}\|=1} \langle \mathbf{v}, A\mathbf{v} \rangle.$$

This reflects the fact that the largest eigenvalue measures the maximum amount of stretching that A can effect in any given direction, with the direction of maximal stretching being the so-called "first" eigenvector (in the ordering according to eigenvalue). Similarly, the smallest eigenvalue can be characterized as

$$\lambda_n = \min_{\mathbf{v} \neq \mathbf{0}} \frac{\langle \mathbf{v}, A\mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle}.$$

The eigenvalues in between can also be characterized by adding the constraint that the maximizing/minimizing vector also be orthogonal to the eigenvectors for the other eigenvalues.

[2]Originally, the Rayleigh-Ritz Theorem is never stated in this section. I inserted a pointer to it but this could be changed if you like.

3.3 PCA

The principal components for a collection of random vectors (with mean 0) are the eigenvectors for the sample covariance matrix. They represent the directions (basis) that successively account for the "most" variance in the data. In this case, the eigenvectors for the covariance matrix effectively give the linear combinations of the original random variables that result in uncorrelated random variables. These new random variables provide us with a means of accomplishing a simple dimension reduction of the original data by taking some subset of the new random variables that account for a good chunk of the total variance. Note that if given a cloud of points about the origin (meant to be a scatter plot of mean zero data), then a direction with a lot of the total variance of the data is a direction such that if the data were all projected onto that vector, then the relative positions of those projections would give you a lot of information as to the relative distances between those points in the original data. Let's see how we get to that.

We start with a collection of random vectors, i.e., a set of m **observations** or **measurements** of multiple attributes of a number of n **subjects**, $\mathbf{X}_1, \ldots, \mathbf{X}_n$ where

$$\mathbf{X}_i = \begin{pmatrix} X_i^1 \\ X_i^2 \\ \vdots \\ X_i^m \end{pmatrix}.$$

For a classic example, the $\mathbf{X}_i$ represent people and the $m = 2$ observations are height and weight. In that example, we generally have $n > 2$, so that we have $m < n$. The other kind of example also occurs, i.e., $m > n$. A typical example of this occurs in microarray data where we will have m on the order of tens or even hundreds of thousands (gene expression levels) and again, n is the number of people in the study, generally a number much smaller than m. The analysis in these two cases will be slightly different.

Given such data, each sample corresponds to a point in $\mathbb{R}^n$. The first thing we do is remove the sample mean:

$$\bar{\mathbf{X}} = \frac{1}{n}\sum_i \mathbf{X}_i$$

which is nothing but the vector of sample means for each of the individual observations. The data is then "mean-centered" – i.e., we consider the observations $\mathbf{X}_1 - \bar{\mathbf{X}}, \ldots, \mathbf{X}_n - \bar{\mathbf{X}}$. We'll call the new centered data $\mathbf{B}_i$ and these will make up the columns of the $n \times m$ matrix B.

Given n measurements of two observations X and Y (i.e., two rows of the B matrix), the **sample covariance** of those (random) variables is defined as

$$Cov(\mathbf{X}, \mathbf{Y}) = \frac{\mathbf{1}}{\mathbf{n-1}}\sum_{\mathbf{i}} (\mathbf{X_i} - \bar{\mathbf{X}})(\mathbf{Y_i} - \bar{\mathbf{Y}})$$

it is large (and positive) if they move together (about their respective means), large (and negative) if they move in opposite directions (around their means) and (in theory) roughly cancel out if they move independently of one another. Thus, if $Cov(\mathbf{X}, \mathbf{Y}) = \mathbf{0}$ we say that $\mathbf{X}$ and $\mathbf{Y}$ are **uncorrelated**. The quantity

$Cov(\mathbf{X},\mathbf{X})$ is the (sample) **variance** of $\mathbf{X}$. Finally, the quantity

$$\frac{Cov(\mathbf{X},\mathbf{Y})}{\sqrt{\mathbf{Var}(\mathbf{X})}\sqrt{\mathbf{Var}(\mathbf{Y})}}$$

is the **correlation** of $\mathbf{X}$ and $\mathbf{Y}$. Note that the correlation is the inner product of the normalized vectors of samples, so in fact is the cosine of the angle between the normalized samples.[3]

Given a covariance matrix,

$$C_{i,j} = Cov(X^i, X^j)$$

the spectral decomposition gives

$$U^t C U = Diag(\lambda_1, \ldots, \lambda_m)$$

where $U = (\mathbf{u_1} \cdots \mathbf{u_m})$ is the matrix of (orthonormal) eigenvectors with (real) eigenvalues $\lambda_1 \geq \cdots \geq \lambda_m \geq 0$. We assume an ordering such that the eigenvalues are listed in decreasing order.[4]

The implication of the spectral decomposition for the mean-centered data is

$$\frac{1}{n-1} U^t B (U^t B)^t = ([\mathbf{u_1} \cdots \mathbf{u_m}]^t \tilde{X_1} \ldots [u_1 \cdots u_m]^t \tilde{X_n}) = \Lambda$$

where $\tilde{X_j} = X_j - \bar{X}$ is the mean-centered sample vector. Thus,

1. For each measurement X_k, the entries in U provide a systematic way to reorganize the measurements so that we arrive at uncorrelated combinations of the measurements.

2. Moreover, because the u_i are orthonormal, the matrix-vector product $[u_1 \cdots u_m]^t \tilde{X_k}$ gives the coordinates of $\tilde{X_k}$ with respect to the basis $u_1, \ldots, u_m$.

The second of these implies that we have discovered a new coordinate system for representing the data in such a way that the new basis is uncorrelated and the coordinates are listed in terms of decreasing variance.

Dimension reduction and visualization.

Given a collection of multidimensional data, $s_1, \ldots, s_n \in \mathbb{R}^m$ (n is the number of samples and m as the number of measurements), a "dimension reduction" of the data is the process of finding a change of basis for the data space ($\mathbb{R}^m$) and then considering only a subset of the new coordinates as a reasonable summary

[3]Note that the correlation is the cosine of the angle between two mean-centered normalized vectors of measurements, e.g., the mean-centered heights and weights of a population of size n. Thus, the normalized mean-centered samples are turned into unit vectors in $\mathbb{R}^n$ and the correlation is the angle between these vectors. If the angle between these vectors is θ (with $0 \leq \theta \leq \pi$) then the **distance** between these points is $2\sin\frac{\theta}{2}$. In this way the correlation is tied a notion of distance and gives a way to assign a weight to the connection between the underlying measurements. Note that the stronger the correlation, the shorter the distance, so that the weight function would have to take this into account. For example, $exp(-d(a,b))$ would do this. Such a weighted network would be an example of a **correlation** network.

[4]Note that since $C = \frac{1}{n-1} BB^t$, C is symmetric positive semidefinite, so that its eigenvalues are all nonnegative.

of the data. The hope is that in this new reduced summary, it is easier to find structure in the (presumably) very high-dimensional original data.

PCA gives a principled (no pun intended!) approach to accomplishing this dimension reduction. The idea is that the first several principal components give the appropriate coordinate system.

The eigenvalues of the covariance matrices are the variances and they are all real and nonnegative. Generally, one hopes to see an "elbow" in the plot of index versus fraction of variance, i.e., potting $1 - a_k$ versus k where

$$a_k = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_r}.$$

indicating a dramatic shift in the contribution to the overall variance of the Y_k. The position of this elbow marks a natural place to cut-off as the true dimensionality of the data, with remaining dimensions attributed to noise or intrinsic randomness.

Elbowology

The *ad hoc* method for picking the an appropriate number of principle components to form a dimension reduction is a commonly used intuitive method for selecting parameters in various models. As it is so widely used, we give it name - Elbowology. In the context of the problems we consider in this book, we will often use elbowology to pick or estimate parameters such as the number of clusters in a clustering algorithm or (as above) the number of significant eigenvalues.

Suppose we wish to estimate a parameter ℓ. To use elbowology, we need to first identify a *loss function*, $f(\ell)$ which gives some measure of goodness of fit for our problem. In the application to PCA given above, the loss function is $f(\ell) = 1 - a_\ell$. As discussed above, it measures the fraction of total variance captured by the first k eigenvalues.

The "method" of elbowology is then to consider the graph of $f(\ell)$ and to look for ℓ where there is a natural cut-off - the elbow - in the loss function. This method is also sometimes called the scree method in the literature. This method was first introduced in the context of PCA and Factor Analysis by Cattell (1966).

PCA and the SVD

As long as $m << n$, the approach outlined above for computing PCA for a data matrix is completely feasible. On the other hand, if $m >> n$ (as is generally the case in something like microarray data, where n may be on the order of 10s or 100s, but the number of measuremens is thousands of times larger) then we've set ourselves up to find the eigenvalues for a huge matrix, even though the rank of the data matrix can be at most the smaller of number of the rows and columns.[5] However, what we will see is that we work with B^tB (instead of BB^t) to find the principal components. This is through the use of the **singular value decomposition** or SVD.

[5]Recall that for a real $m \times n$ matrix A, the **rank** of $A = dim(col(A)) = dim(row(A))$. Furthermore $rk(A^tA) = rk(A) = rk(AA^t)$.

The SVD is a generalization of the diagonalization of a matrix. Not every matrix is diagonalizable, but every matrix has a singular value decomposition.

We'll frame this via the eigenvalue problem: for a square matrix A ($m \times m$) find $\mathbf{v} \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$ such that

$$A\mathbf{v} = \lambda\mathbf{v}.$$

If we can find a basis of such v then we have a diagonalization of A,

$$A[\mathbf{v}_1|\ldots|\mathbf{v}_m] = [\mathbf{v}_1|\ldots|\mathbf{v}_m]Diag(\lambda_1,\ldots,\lambda_m) \qquad \text{or} \qquad V^{-1}AV = Diag(\lambda_1,\ldots,\lambda_m).$$

The SVD comes from taking on a different problem: given an $m \times n$ matrix A, find $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$ such that

$$A\mathbf{v} = \lambda\mathbf{u}.$$

As it turns out, we can find orthonormal bases $\{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ for $\mathbb{R}^n$ and $\{\mathbf{u}_1,\ldots,\mathbf{u}_m\}$ for $\mathbb{R}^m$ such that

$$A\mathbf{v}_i = \sigma_i\mathbf{u}_i$$

for $i = 1,\ldots,r = rk(A)$ and where $\sigma_i > 0$ (note that $\sigma_i^2 \geq 0$). The σ_i are called the **singular values** for A. When A is symmetric, these will be the absolute value of the eigenvalues of A. The rest of the basis $\{\mathbf{v}_{r+1},\ldots,\mathbf{v}_n\}$ will span the null space of A. Similarly, the vectors $\{\mathbf{u}_{r+1},\ldots,\mathbf{u}_m\}$ will span the orthogonal complement of the range of A.

We find the SVD as follows: Consider A^tA. This is real symmetric and positive semidefinite. Assume it has rank r. Then we can write the nonzero eigenvalues of A^tA as

$$\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_r^2 > 0$$

with associated orthonormal eigenvectors $\mathbf{v}_i$ (so that for $i > r$, these are just an orthonormal basis for the nullspace). Thus,

$$A^tA\mathbf{v}_i = \sigma_i^2\mathbf{v}_i.$$

Multiplying both sides by A, we have that

$$(AA^t)(A\mathbf{v}_i) = A\sigma_i^2\mathbf{v}_i = \sigma_i^2(A\mathbf{v}_i)$$

so that the vectors $A\mathbf{v}_i$ are eigenvectors for the real symmetric and positive semidefinite matrix AA^t. In addition,

$$\|A\mathbf{v}_i\|^2 = (A\mathbf{v}_i)^t(A\mathbf{v}_i) = \mathbf{v}_i^tA^tA\mathbf{v}_i = \mathbf{v}_i^t\sigma_i^2\mathbf{v}_i$$

So $\|A\mathbf{v}_i\| = \sigma_i$. Letting

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|},$$

we have

$$A\mathbf{v}_i = \sigma_i \mathbf{u}_i.$$

Note that the singular values are the eigenvalues for both A^tA and AA^t.

Now, let's take this back to the problem of finding our principal components when $m >> n$. Using the above, we can find the nonzero eigenvalues $\lambda_1 \geq \cdots \geq \lambda_r$ for B^tB and then for each λ_i solve the system of linear equations

$$BB^t\mathbf{v} = \lambda_i \mathbf{v}$$

in order to find the principal components, as needed.

The Reduced SVD. Let A have rank r so that

$$A[v_1\ v_2\ \ldots\ v_r] = [u_1\ u_2\ \ldots u_r]Diag(\sigma_1, \ldots, \sigma_r).$$

Then using the fact that $[v_1\ v_2\ \ldots\ v_r][v_1\ v_2\ \ldots\ v_r]^t = I_n$ we can write

$$A = [u_1\ u_2\ \ldots u_r]Diag(\sigma_1, \ldots, \sigma_r)[v_1\ v_2\ \ldots\ v_r]^t.$$

This is the reduced SVD.

Computational Issues. We have discussed the theory behind PCA. Now we will briefly focus on a practical issue that arises when computing principal components for a *high-dimensional* data set on a *real* computer. In the case where $m << n$ (*low-dimensional*) we simply implement the theory as described above, computing the eigenvectors of the covariance matrix. In this case the covariance matrix will be reasonably sized and there are good algorithms to accurately compute the spectral decomposition of symmetric matrices. In the case where $m >> n$ (*high-dimensional*) we saw that we could use the SVD to alleviate the need to compute the spectral decomposition of the *large* $m \times m$ covariance matrix. One could compute the SVD of the data matrix X and use the columns of the resulting matrix U as the principal components, however, it is ill-advised to do so if numerical accuracy is important. It turns out that the algorithms to compute the SVD of a given matrix are numerically stable, this means that if A is our input matrix and $\tilde{U}$, $\tilde{\Sigma}$ and $\tilde{V}$ are the matrices returned from our SVD algorithm, then $||A - \tilde{U}\tilde{\Sigma}\tilde{V}^t||$ will be *small*. Now, if U, Σ and V are the true components of the SVD of A, unfortunately it isn't the case that $||\tilde{U} - U||$, $||\tilde{V} - V||$ and $||\tilde{\Sigma} - \Sigma||$ are *small*. In fact, the differences between the individual matrices and their exact values could be orders of magnitude larger than the difference between A and the product of the computed matrices (10^{-5} vs. 10^{-15}).

In practice one should explicitly compute the eigenvectors of the $n \times n$ covariance matrix, and then use the data matrix X to transform these eigenvectors into the correct space. Since algorithms for computing eigenvectors are guaranteed to provide *accurate* approximations to the true eigenvectors of a matrix (on the order of machine round-off) we won't have the problems above where our principal components are only accurate to three or four decimal places.

The numerically accurate method of computing principal components for high-dimensional data is:

1. Compute the small covariance matrix $C_s = X^t X$
2. Find the eigenvalues and eigenvectors of C_s using an accurate eigenvector algorithm
3. For each eigenvector, compute $u_i = X v_i$
4. Normalize each u_i

Exercises: (1) Insert Nick's Matlab exercise and a link to do PCA on the face database.
(2) Doing PCA on a microarray set (include a picture). Here the number of measurements is (in general) much larger than the number of samples, so the SVD trick is necessary.

Other applications of the SVD and singular values

(1) Eigenvector Centrality. Let A be the weighted symmetric adjacency matrix for a network. One notion of centrality (or importance) for a vertex can be phrased as: the importance/centrality of a given node is proportional to the sum of the importances/centralities of my neighbors. If you're connected to important positions, you are important. We make math out of this as follows. Let $c(i)$ be the centrality of node i, whatever that means. Then the importance relation becomes

$$c(i) = \mu \sum_j a_{ij} c(j)$$

or as a matrix-vector equation

$$c = \mu A c$$

or, to put it in spectral form,

$$Ac = \frac{1}{\mu} c$$

where μ is some positive parameter. We now make use of the important, but slightly difficult-to-prove (see e.g., [7]) Perron-Frobenius Theorem:

Perron-Frobenius Theorem. Let A be a nonnegative matrix then the largest eigenvalue of A is positive and the associated eigenvector has nonnegative entries.

So, let's take as a measure of centrality to be c, the largest eigenvector for A. Then $c(i)$ is defined to be the **eigenvector centrality of** i.

(2) Hubs and Authorities. The "trick" of considering $A^t A$ and AA^t comes up in other areas of network analysis. In particular, in various notions of centrality. J. Kleinberg generalizes the notion of eigenvector centrality [6] to create a more textured view of centrality in a directed (i.e., weighted, but not necessarily symmetric) network quantified in the twinned notions of **hubs** and **authorities.**

In a directed network there are two ways in which a node may be important. If we keep in mind the fundamental directed network that is the WWW, we have those nodes that everyone points to as sources

of information (e.g., Wikipedia) and those nodes that everyone goes to as central locations for getting to information (e.g., Yahoo! or Google). The former are (colloquially) "authorities" and the latter are "hubs."

How great an authority is someone? Well, if that source of information is pointed to by places whose "job" it is to get you to sources (i.e., hubs), then that is some measure of how good an authority you are. Conversely, how great a hub are you? Well, if all the god authorities say that you are a go-to place, then you are a good hub.

As in the case of eigenvector centrality, quantifying that common sense comes out to a set of matrix-vector and ultimately eigenvector equations. If $a(i)$ is the authority index of node i and $h(j)$ is the hub index of node j, then the above anecdote is quantified in the following relations

$$\begin{aligned} a(i) &= \sum_j w_{ij} h(j) \\ h(i) &= \sum_j w_{ji} a(j) \quad = \quad \sum_j w^t_{ij} a(j) \end{aligned}$$

or

$$\begin{aligned} a^{(1)} &= Wh \\ h^{(1)} &= W^t a \end{aligned}$$

View this as a first approximation to the actual values. When we iterate this relation we get

$$\begin{aligned} a^{(2)} &= Wh^1 = WW^t a \\ h^{(2)} &= W^t a^1 = W^t W h \end{aligned}$$

so that

$$\begin{aligned} a^{(2k)} &= (WW^t)^k a \\ h^{(2k)} &= (W^t W)^k h. \end{aligned}$$

If we let k go to infinity we see that the authority and hub values go multiples of the leading eigenvectors of WW^t and $W^t W$ respectively.

Multidimensional Scaling (MDS)

In another instance, we might be given only similarity or dissimilarity data. This too has a dimensionality, this time in the sense that we can try to find a collection of points in some Euclidean space of a given number of dimensions whose interpoint distances are either equal to or close to (in a way that we need to make precise) the set of dissimilarities.

Multidimensional scaling (MDS) is the general set of techniques we use to find such a set of points. Suppose D is our $n \times n$ dissimilarity matrix[6] and we wish to find a good low dimensional representation of the dissimilarity data. Is D is given as the distances between objects, then we may proceed. However, if the entries of D do not come from a measurement of distance, we may choose to apply a continuous monotone

[6]If given a similarity matrix S, there are a number of ways to transform it into a dissimilarity matrix. If the entries of S are between 0 and 1, the simplest is to form $D_{ij} = 1 - S^2_{ij}$.

function to the entries to make them behave more like distances. We call the resulting matrix D_f. For the purposes of understanding the method, we may focus on the case where f is the identity map. We wish to find a dimension k and a set of points $\{x_i\} \subset \mathbb{R}^k$ so that if

$$D'_{ij} = d(x_i, x_j)$$

where d is a metric on $\mathbb{R}^k$, then $\|D_f - D'\|$ is minimal. If d is the Euclidean distance, this is called *classical multidimensional scaling*, otherwise it is called *metric multidimensional scaling* or simply *multidimensional scaling*.

As with many dimension reduction schemes, the choice of k is potentially problematic. There are two standard tacks to take. First, we can *a priori* specify k using some prior information concerning the dimensionality or because we wish to produce useful visualizations (in the latter case, k is usually 2 or 3). Upon specifying k, finding the points $\{x_i\}$ is then a problem in numerical optimization, which can be approached via a variety of algorithms. Second, we may wish to find the smallest dimension k where we can find an embedding where D and D' match as closely as possible. In this case, the problem may be approached linear algebraically.

The general idea stems from considering the simpler case when the entries of D are distances between points, $\{x_i\}_{i=1}^n$ in k-dimensional Euclidean space. Given the distances, can we recover the points? As a first step, we try to recover the matrix of inner products of the x_i, A (i.e. $A_{ij} = \langle x_i, x_j \rangle$). Since

$$\begin{aligned} D_{ij}^2 &= \langle x_i - x_j, x_i - x_j \rangle \\ &= \langle x_i, x_i \rangle + \langle x_j, x_j \rangle - 2\langle x_i, x_j \rangle \\ &= A_{ii} + A_{jj} - 2A_{ij} \end{aligned}$$

This is a system of n^2 equations in n^2 unknowns and, it turns out, is non-degenerate and can be solved to yield:

$$A_{ij} = -\frac{1}{2}\left(D_{ij}^2 - \frac{1}{n}\sum_{k=1}^n D_{kj}^2 - \frac{1}{n}\sum_{k=1}^n D_{ik}^2 + \frac{1}{n^2}\sum_{k,l=1}^n D_{kl}^2 \right)$$

From this matrix, we may recover the $\{x_i\}$. Let X be the matrix whose columns are the $\{x_i\}$. Then,

$$A = XX^t$$

But, A is a symmetric positive semidefinite matrix and hence has a spectral decomposition,

$$A = V\Lambda V^t$$

Thus, we can realize $X = V\Lambda^{\frac{1}{2}}$.

In general, for dissimilarity matrices that do not (necessarily) come from distances, we can form the same matrix A and, so long as A is positive semidefinite[7], find the $\{x_i\}$ using the same method.

[7] If A is not positive semidefinite, there are a variety of techniques to amend the procedure. For example, one may often add a constant to A to make it positive semidefinite and proceed from there.

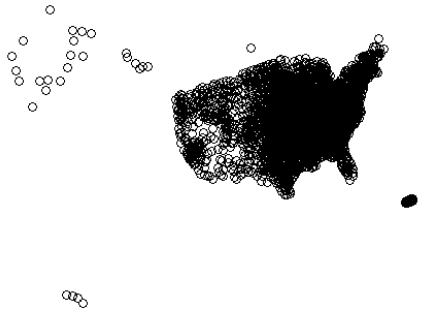


Figure 3.1: MDS applied to distances between geographic centroids of US counties.

Example 3.3.1. *The first use of multidimensional scaling by Young and Householder [**?**] was to recover a configuration of points in Euclidean space from their matrix of distances. Some of the initial applications were to construct maps of cities based on railway transit times to estimate efficiencies of rail travel.*

To see a version of this type of application, we can take as our data the geographic centroids of each county in the United States in coordinates given by stereographic projection of the sphere onto the plane. If our initial data is simply the Euclidean distance between those points, can we recover the points themselves? Applying MDS while fixing the embedding dimension at two, we indeed recover a familiar picture, shown in Figure 3.1.

Example 3.3.2. *As a last example, we apply multidimensional scaling to the S&P 500 data set. The 438 equities each have 902 data points associated to them. Thus, the embedding of the equities given by the data is into $\mathbb{R}^{902}$. As this is impossible to visualize, we use multidimensional scaling to find the best representation we can into three dimensional space. The results are shown in Figure **??**. While the picture is not obviously informative, it does hint at additional structure - why are some blobs denser than others? what does it mean to be in the center rather than the edge? These are questions we will begin to address when we discuss clustering.*

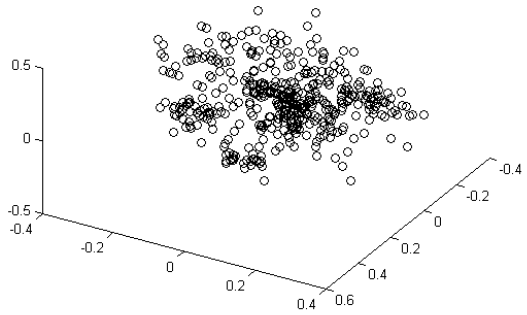


Figure 3.2: Three dimensional multidimensional scaling for the S&P 500 data

??

Chapter 4

Unsupervised Learning – Cluster Analysis

Summary: Having accomplished dimension reduction we now look for "clusters" which in this case reflect neighborhoods of correlated activity. We consider in detail two approaches, the well-known k-means clustering algorithm as well as spectral clustering, which is perhaps less well-known, but much more powerful for interesting geometries. This is "unsupervised" learning in the sense that the we look at the raw data without annotation, thus attempting to discover structure in the data without any additional meta-information. We also explore a variation on this approach, where the clustering precedes the multidimensional scaling.

4.1 Clustering

Dimension reduction is just one tool for better understanding the delightful mess that can be a complex system. A sensible dimension reduction (and accompanying embedding in some Euclidean space) should be such that the fundamental relations of the objects of interest are preserved. One basic structure to look for is the "cluster," that is, points which appear to be closer together in space as a whole than they are to other points in space (which are then further divided into other clusters).

For example, in the complex system of the markets (e.g., any subset of the equities market), the time series of the equities become points in space, whose distance from one another reflects (inversely) their correlation. Part of our goal in understanding the market is to understand the correlation structure, to see which equities move together, or apart, as a means of understanding the market as a whole. These equities that move together are clusters and will be subsets of equities that are more correlated as a whole (and pairwise) than they are with other equities. If our example came from the feature vectors of artworks or books, the clusters would correspond to works that are more similar to each other than to other works.

If we are looking to "discover" the structure, then we are looking for an **unsupervised** way of finding clusters in the data. A well-known technique for finding clusters is the ***k*-means algorithm**. We describe it below. As we will see, *k*-means "suffers" from needing to be given a number of clusters (the *k* in *k*-means) to seek. A second method, **spectral clustering** helps to identify the number of clusters in a network via an

eigenvalue analysis of the corresponding weighted network.

k-means.

Once the number of clusters, k, is specified, k-means is an unsupervised method to find clusters. This means that no additional intervention is needed - the set of points is passed to the algorithm which returns a clustering. The algorithm is simple to explain: given a set of points $X \in R^m$ and number of clusters k, go about finding a clustering into c clusters as follows:

- **Step 0 (initialization):**

  A. Initialize a set of k points in R^m, $C^0 = \{c_1^0, \ldots, c_k^0\}$ as a guess for the centroids of the k clusters.

  B. For each $x \in X$ assign x to cluster j if x is closest to c_j^0 and initialize $i := 0$.

- **Step 1: Repeat**

  A. For each cluster, recompute the centroids $C^i = \{c_1^i, \ldots, c_k^i\}$.

  B. For each $x \in X$ assign x to cluster j if within C^i, x is closest to c_j^i.

  C. Increment i.

  Until No point changes its cluster.

- **Step 2:** Let $C(k) = \{c_1, \ldots, c_k\}$ be the clustering $C^i = \{c_1^i, \ldots, c_k^i\}$ from the termination of Step 1.

The initialization step of choosing an initial set of centroids can certainly influence the outcome of the algorithm. In fact, k-means generally finds a local optimal clustering which depends on the initialization of the algorithm. Because of this, it is standard practice to run k-means many times on the same dataset and to pick the clustering which minimizes the total of the distances of the points to the ending centroids:

$$totaldist(C(k)) = \sum_{c \in C(k)} \sum_{x \in c} |x - c|$$

A key question for this method, which is also a key question in many unsupervised clustering algorithms, is how to pick a good number of clusters? As in the case of dimension reduction one can use a form of elbowology, using as a loss function $k \to totaldist(C(k))$. Note that at the extreme of number of clusters equal to number of points and the loss is equal to 0. This approach has the disadvantage of being very computationally intensive.

Just as there is no one clustering method which is effective for all possible situations, there is no one method for picking a good number of clusters. Elbowology provides an *ad hoc* method for picking clusters. Next, we present two other methods.

Cluster consistency

A clustering partitions X into k subsets. If the clustering is robust, then the assignments of members of X to the clusters should not change if we remove some members of X before clustering. In other words, in a good

clustering, the clusters remain consistent even if omit some of the original points. If we randomly sample from our set independently cluster the subset using the same method, we say that the clustering exhibits cluster consistency for this splitting if the percentage of members of the subset of X which are categorized differently is small.

The method of *cluster consistency* is a formalism of this idea: Given a clustering of the data with k clusters.

Step 1: Randomly select half of the points of X, denoted $\tilde{X}$.

Step 2: Perform clustering on both $\tilde{X}$ with k clusters.

Step 3: Compute the percentage of members in X that are classified differently in the two clusterings

Needless to say, steps 1-3 should be repeated as many times as feasible to get a good picture of the consistency of a given number of clusters. To use the consistency check to aid in the selection of k, the number of clusters, we can use repeat steps 1-3 a fixed number of times for each k. We select the k that has the minimum mean consistency score.

Information Criteria

There are other, information theoretic, approaches as well.

Example 4.1.1. ***k*-means works well for "ball" clusters:** *To initially illustrate the algorithm, we create a dummy dataset by generating 400 points in $\mathbb{R}^2$ in 4 groups. For each group of 100 points, we generate their positions randomly by adding two dimensional Gaussian noise to a center point.*[1] *We then use elbowology to pick k. Figure 4.1 shows the results: (a) shows a plot of the generated points (b) $totaldist(C(k))$ for k from 2 to 10 and (c) the graph from (a) colored by the $k = 4$ clustering. Note that Figure 4.1 (b) shows a relatively clear drop off around $k = 4$.*

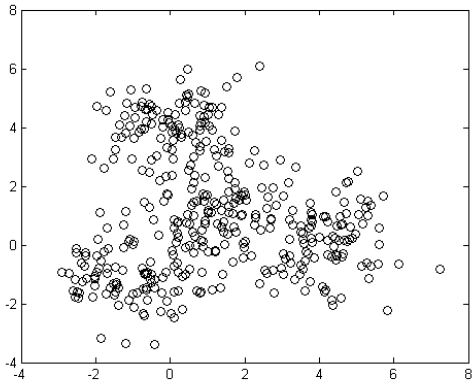
Example 4.1.2. "No free lunch": *To illustrate the algorithm further, we will consider the S & P 500 data. In Section* **??***, we saw how multidimensional scaling can be used to embed the 438 equities as points in a small dimensional space. However, there are still 438 points! The picture (as seen in Figure* **??***) is not particularly informative. We will use this as an example of how one might use clustering to find a more accessible presentation of the data.*

Figure **??** *hints that there might be good clusters within the data - there are darker patches indicating denser sets of points. Part of our job is to see if these visual clusters are real or simply artifacts of the*

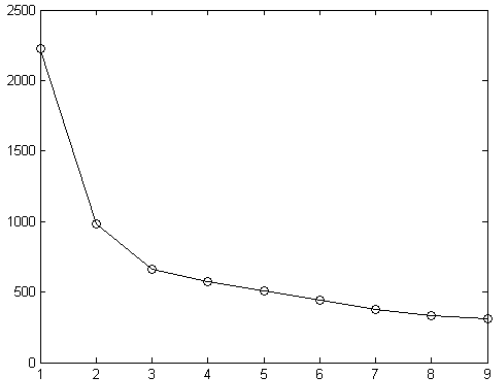
[1] An example of how to do this in MATLAB is given by

```
X = [randn(100,2)+ones(100,2);randn(100,2)-ones(100,2);...
randn(100,2)+repmat([4,0],100,1);randn(100,2)+repmat([0,4],100,1)];.
```

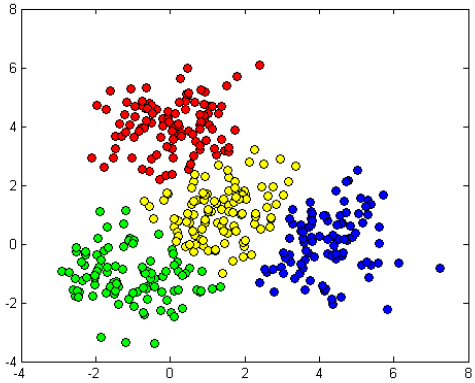
In this case, the 4 groups are centered at $(1,1),(-1,-1),(4,0)$ and $(0,4)$ respectively.



(a) Data

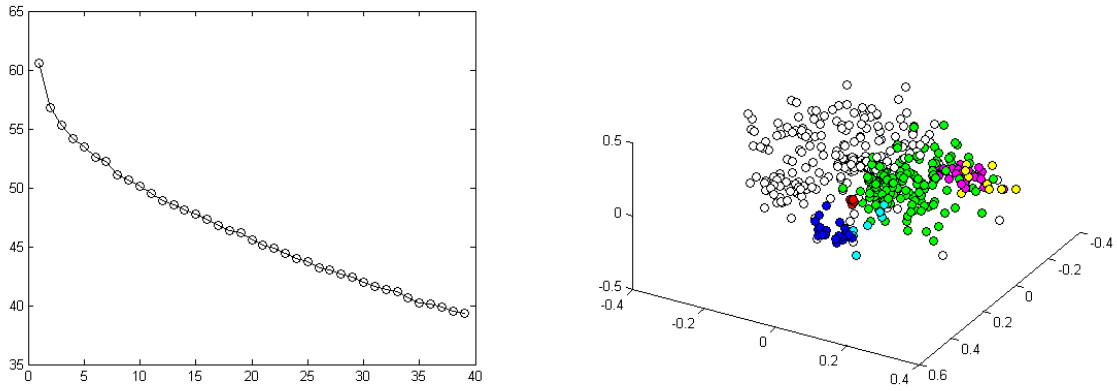


(b) Elbowology



(c) 4 clusters

Figure 4.1: (a) 400 points of centered Gausian data (b) $totaldist(C(k))$ for k from 2 to 10 and (c) Data points colored by the 4 cluster labeling.



(a) Elbowology

(b) 7 clusters

Figure 4.2: Using k-means on the S&P 500 data. (a) Elbowology for picking k in k-means. (b) Three dimensional multidimensional scaling for the data with colors given by cluster labels with 7 clusters.

dimension reduction and/or the visualization. Figure 4.2 (a) shows $totaldist(C(k))$ *as a function of k from 2 to 20. Alas, there is no clear elbow, but there does seem to be a slight drop for* $k = 7$*. Figure 4.2 (b) shows the same plot as Figure* **??** *colored according to the clusters. We can see from this illustration that some of the clusters seem like "real" clusters (red, blue, purple) while others seem rather diffuse (white, green).*

We emphasize that this application of k-means and the associated elbowology is neither satisfying nor robust. To use these results for anything other than exploration of the data set would be inappropriate.

Example 4.1.3. "No free lunch" part two: *Suppose we have data, such as that shown in Figure 4.1, with two obvious groupings but that are not groupings that are shaped like blobs. Consider this toy data set in which there are two natural clusters: the exterior circle and interior ball. Reviewing the algorithm for k-means should convince you that k-means - with any k - will cluster this data set poorly. In Figure 4.4 we see typical results from 100 different runs of k-means (with* $k = 2$*) on this dataset. This example demonstrates that while k-means works quite well for "ball" clusters, it can work quite poorly for clusters which take other shapes.*

Linkage analysis and dendrology

Linkage analysis and hierarchical or agglomerative clustering – has the (potential) disadvantage that you get a tree-like structure for the clusters.

Modularity and divisive clustering – Break up the dataset/network until you can break it up no further - i.e., no natural communities.

There are other forms of clustering that are also widely used and may yield structures different from either the k-means or spectral clustering methods. One such is "linkage analysis." In this technique we

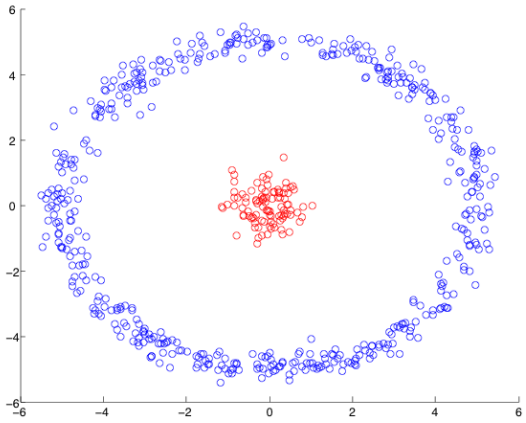


Figure 4.3: A toy data set with two groupings which are not balls.

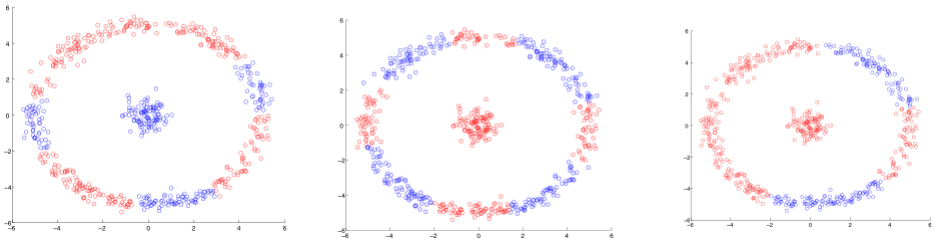


Figure 4.4: Typical results from one hundred runs of k-means with $k = 2$ on a dataset of concentric circles.

assume that we start with some sort of similarity matrix. From this clusters are constructed by successively aggregating entities via an iterative process whereby at every stage in the process the two nearest clusters are merged into a single cluster. Thus, we start with each point in its own cluster and end with all points in one cluster. The aggregation process is illustrated via its associated dendrogram whose geometry encodes the process of aggregation. Note that the process requires some way to measure the distance/similarity between two clusters. Some examples are

- complete linkage
- single linkage

The information encoded in the dendogram construction is nicely illustrated by the dendogram for the S& P cluster centroids. The steps of joining are as follows:

1. 6 is joined to 13
2. 10 is joined to 18

3. 9 is joined to 14
4. 7 is joined to 8
5. then 9 and 14
6. then 13 is joined to 10 and 16
7. ...

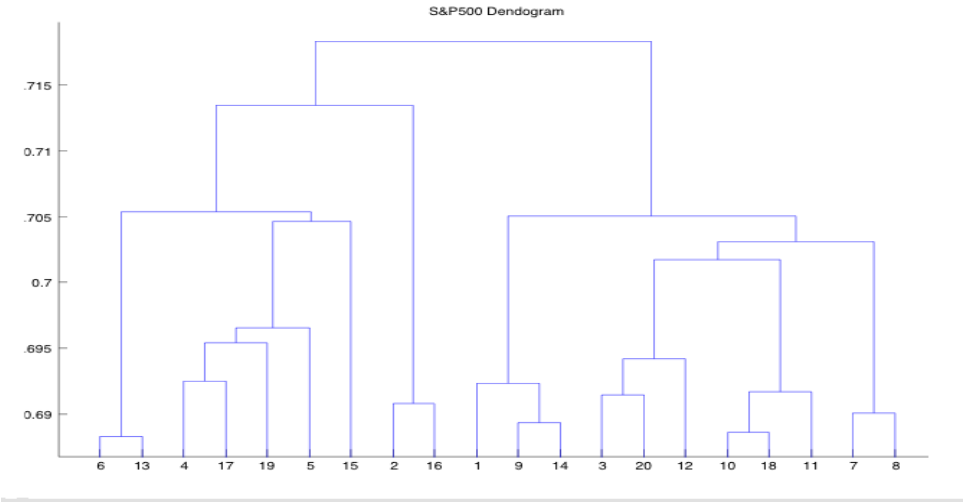


Figure 4.5: Dendrogram for the 20 spectral clusters of the S& P.

The same process can be carried out with the entire S& P data set, although the picture is a bit less iformative.

On the flip side of this aggregation approach, we have processes of successive division. That is, we successively split the data in the "best way," then split the splits and so on. As applied to networks, this might be the iteration of a techniques like **modularity** (see [12].

4.2 Spectral clustering

We discussed in some detail the k-means algorithm. The fact is that k-means is very good at finding ball-like clusters, but is not very good for finding nontrivial structure such as the circle that we saw above. One method to overcome this is called Spectral Clustering. Spectral Clustering relies on a fundamental geometric operator - the Laplacian - to encode the global geometry of the data set. The algorithm then uses this encoding to produce geometrically meaningful clusters.

While our goal is to describe the spectral clustering algorithm, we note that there are many features of the analysis of the Laplacian which are interesting and useful in their own right.

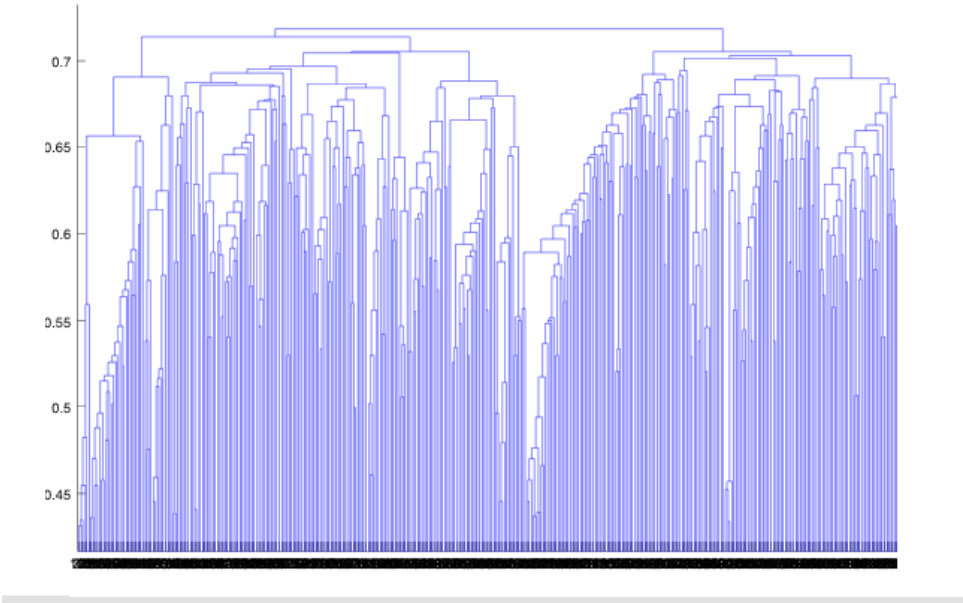


Figure 4.6: Dendrogram for the S& P equities.

Introduction of the Laplacian

To begin, we introduce the graph **Laplacian**. We assume we have a graph G given by an adjacency matrix A. A graph Laplacian is the discrete analogue of the continuous Laplacian, which we are familiar with in two and three dimensional Euclidean space and which generalizes to arbitrary manifolds. One of the great uses of the Laplacian is given by its role in the heat equation. The heat equation is a model of heat flow which is a simple model based on our real world experience: heat flows from warmer areas to colder ones, evening out the temperature in an (insulated) object over time. The Laplacian's role in the heat equation is that it calculates the direction the temperature should change at each point to achieve the evening out. Simply put, the Laplacian, when applied to a function, performs an averaging procedure. Roughly speaking, on an infinitesimal level, the continuous Laplacian applied to a function returns the difference of the function value at a point and the average of the function over a neighborhood of the point. If that difference is positive, it indicates that heat will flow to the point, raising it temperature. If negative, it indicates that heat will flow away from the point, lowering its temperature.

The continuous (flat) Laplacian in two dimensions is the operator

$$\Delta f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Our infinitesimal averaging operator comes from the fact that we can view this as

$$\Delta f(x,y) \approx \frac{1}{\varepsilon^2}[4f(x,y) - f(x+\varepsilon,y) - f(x-\varepsilon,y) - f(x,y+\varepsilon) - f(x,y-\varepsilon)].$$

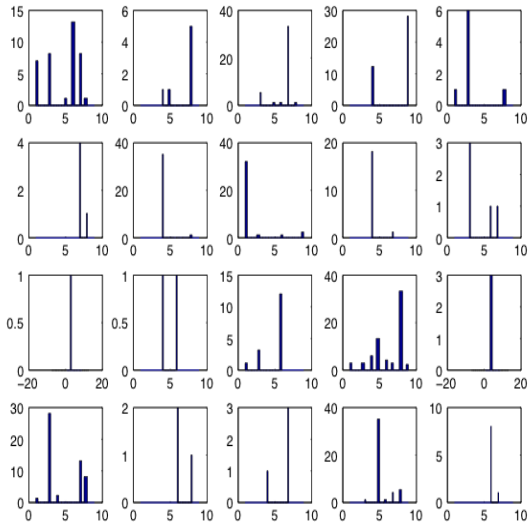


Figure 4.7: Sector breakdown of the linkage clusters.

If we see this as an operator on a function on a grid in the plane with nodes at distance ε from one another, then this is the (discrete) operator whose value at a given node is the sum of the differences between f at the node and at each of his four neighbors. Thus, a generalization to the network situation has

$$Lf(k) = \sum_{i \sim k} f(k) - f(i) = (A - D)f(k)$$

where A is the adjacency matrix of the network and D is the degree matrix with diagonal entries equal to the degree. Note that it is not too hard to prove that L is real, symmetric, positive semidefinite. We can assume that the network has no isolated points, thus doing away with rows and columns in A and D that are zero. This version of the Laplacian is sometimes called the unnormalized Laplacians.

We may also work with a weighted, but undirected matrix, thus replacing A by a (symmetric) positive weight matrix. In these cases, we may want to work with a slight variant

$$L = D^{-1}A - I.$$

This, the normalized Laplacian, has essentially the same features as the unnormalized Laplacian but can sometimes be computationally better suited to a particular problem. Similarly, we introduce the symmetrized Laplacian,

$$L = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} - I$$

as a third alternative.

The Nullspace of the Laplacian

Suppose our $n \times n$ adjacency matrix A has a connected component with j vertices. Without loss of generality, we can reorder the rows and columns of A so that the subgraph of these j vertices is the $j \times j$ submatrix of A consisting of the first j rows and columns. Then, letting $\mathbf{v} = \langle 1, \ldots, 1, 0, \ldots, 0 \rangle$ be the vector with ones in the first j entries and zeros elsewhere, we have

$$L\mathbf{v} = A\mathbf{v} - D\mathbf{v} = \begin{cases} \sum_{j \sim i} 1 - d_i & \text{for } i \in \{1, \ldots j\} \\ 0 & \text{otherwise} \end{cases}$$

But, $\sum_{j \sim i} 1 - d_i = d_i - d_i = 0$ so $\mathbf{v}$ is an eigenvector for eigenvalue zero. Since we may repeat this for any connected component, we see that the dimension of the null space of the Laplacian is equal to the number of connected components of G.

Exercise: Show the same result for the normalized and symmetrized Laplacia.

4.3 The Fiedler vector

Given the unnormalized Laplacian L, the Fiedler vector is the eigenvector associated to the first nonzero eigenvalue. Since L is symmetric, positive semi-definite,

$$\lambda_{max} = \lambda_n \geq \cdots \lambda_2 \geq \lambda_1 = 0$$

it has only nonnegative eigenvalues and an orthonormal basis of eigenvectors. The Fiedler vector is necessarily orthogonal to the constant vector so it has positive and negative entries.

As we've seen, the nullspace of the Laplacian (0-eigenspace) gives some coarse scale toplogical information - the number of connected components. As we continue up the spectrum we gain more and more information. The Fiedler vector is the next stop.

It is useful to recall the optimization characterization of the spectrum of a symmetric matrix given by the Rayleigh-Ritz Theorem:

$$\begin{aligned} \lambda_{max} &= \max_{v \neq 0} \frac{v^t A v}{v^t v} = \max_{\|v\| = r} \frac{v^t A v}{r^2} \\ \lambda_{min} &= \min_{v \neq 0} \frac{v^t A v}{v^t v} = \min_{\|v\| = r} \frac{v^t A v}{r^2} \end{aligned}$$

with the associated eigenvectors being precisely those vectors that achieve these extrema. The various extensions for the intermediate eigenvectors are obtained by adding constraints of orthogonality for the previously determined eigenvectors. The proof of these facts is not difficult and uses the polar decomposition (see e.g., [7]).

With this in hand we can now see that the Fiedler vector v_{Fdlr} solves the minimization problem

$$\min_{v \perp \mathbf{1}} \sum_{i \sim j} [v(i) - v(j)]^2$$

subject to the constraint of v having norm 1. This is a scattering of points on the line that tries to keep connected nodes close to one another. Note that the most (degree) central nodes affect the most sums so their placement has the greatest effect on this quantity. The orthogonality condition is equivalent to centering the data around 0.

This is a *spectral* embedding of the network in one dimension, where spectral refers back to the fact that we are using an eigenvector of some matrix associated or derived from the adjacency information of the network (in this case the unnormalized Laplacian). We could get higher dimensional embeddings by taking more and more eigenvectors and using the i coordinate of the j eigenvector ("after" the Fiedler vector) as the $j+1$ coordinate of the embedding of node i in $\mathbb{R}^{j+1}$. ******* Present the minimization problem that the j-dimensional embedding solves.

I.e., the "best" j-dimensional embedding is a solution to the problem

$$\min_{\substack{V \in \mathbb{R}^{n\times j} \\ V^tV = I_j \\ V \perp Null(L)}} \sum_{k,\ell} a_{k,\ell} \|V(k) - V(\ell)\|^2$$

where $V(i)$ is the i^{th} row of V.

How do we know when we have a "good embedding?" This is another case where we can use elbowology. In the case of a network that represents a dissimilarity, d_{ik}, then a natural measure of "goodness of representation" - the loss function - would be the same as the one we used when discussing MDS, a measurement of the deviation of the actual distances between points in the j-dimensional embedding ($\tilde{d}_{ik}^j$) and the original dissimilarity measure:

$$f(k) = \Delta^k = \sum_{i,k} |d_{ik} - \hat{d}_{ik}^j|$$

We first outline the algorithm and then explain the various steps.

The Spectral Clustering Algorithm

1. Compute the correlation $\rho(i,j)$ between all pairs of n data points i and j.

2. Form the similarity matrix S defined by

$$s_{ij} = \exp(-sin^2(\frac{cos^{-1}(\rho_{ij})}{2})/sigma^2),$$

   where σ is a scaling parameter. Remove the diagonal of S.

3. Define D to be the diagonal matrix whose i,i element is the column sums of S.

4. Choose a Laplacian.

5. Find the eigenvectors $(v_0, v_1, v_2, \ldots v_n)$ with corresponding eigenvalues $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ of L.

6. Determine from the eigendecomposition the optimal dimensionality l and natural number of clusters k.

7. Construct the embedded data by using the first l eigenvectors to provide coordinates for the data (i.e., sample i is assigned to the point in the Laplacian eigenspace with coordinates given by the ith entries of each of the first l eigenvectors, similar to PCA).

8. Using k-means, cluster the l-dimensional embedded data into k clusters.

Steps 1 and 2

The first step is to transform your data into a matrix which records how similar the data streams are to one another. For the purposes of exposition, we use correlation as it is a common solution to this problem. The second step first transforms the correlation into the chordal spherical distance and then exponentiates the entries of the resulting matrix to create the analogue of an adjacency matrix for the data. The parameter σ is a scale parameter, it essentially allows you to choose the level of correlation you deem significant. Removing the diagonal of S simply remove self loops from the resulting network.

Step 3

D is just the degree matrix of S, the adjacency matrix.

Step 4

Natural choices for the Laplacian are $L = S - D$ or $L = D^{-1/2}SD^{-1/2} - I$. The former is connected to the so-called RatioCut problem while the latter to the NCut problem. Next, we explain these two problems to help guide the choice of the Laplacian in this step.

RatioCut: Simply put for this problem we wish to divide the graph, G, given by S into two parts which have, in some sense, the smallest number of edges between them. In others, it costs the least in terms of cutting edges to cut the graph in half via this partition.

For A, B with $A \cap B = \emptyset$, define $cut(A,B) = \sum_{i \in A, j \in B} S_{ij}$. Then for a partition of G, $\{A, \overline{A}\}$, define

$$CutRatio(A,\overline{A}) = \frac{cut(A,\overline{A})}{|A|} + \frac{cut(\overline{A},A)}{|\overline{A}|}$$

The second largest eigenvalue and associated eigenvector of $L = S - D$ give an indication of the best RatioCut. That is, the Fiedler vector is the solution to a relaxed form of the minimization of RatioCut for a partition of two parts. From this we get to a best division via either

1. k-means on the coordinate given by v_{Fdlr} with $k = 2$.

2. Assign to clusters according to the signs of v_{Fdlr}. Then optimize by swapping the clusters of the coordinates near 0, evaluating in terms of the mean distance from the centroids of the clusters.

If one wants to solve the RatioCut problem for a partition with more than two pieces, we have the generalized RatioCut problem. For a partition of G as $A_1,\ldots,A_m$ define

$$CutRatio(A_1,\ldots,A_m) = \sum_{i=1}^{m} \frac{cut(A_i,\overline{A_i})}{|A_i|}$$

In this case, the solution to the relaxed RatioCut problem is solved (approximately) using the first m nontrivial eigenvalues of $L = S - D$ and their associated eigenvectors. One simply uses k-means with $k = m$ on the coordinates for the vertices given by the entries in the eigenvectors.

One problem with RatioCut is that if your graph has a vertex with a single edge, the solution is often to let A be this vertex. Then $CutRatio(A) = \frac{1}{1} + \frac{1}{\#vertices-1}$. This is often not a useful solution to the problem. One method around this issue is to encourage partitions of roughly equal size. NCut is one such method.

NCut: The NCut problem is a variant of RatioCut which aims to create partitions with pieces of rougly equal size. For a partition of G as $A_1,\ldots,A_m$ define

$$NCut(A_1,\ldots,A_m) = \sum_{i=1}^{m} \frac{cut(A_i,\overline{A_i})}{assoc(A_i)}.$$

where $assoc(A_i) = \sum_{a\in A_i, g\in G} S_{ag}$. As you can see from the formula, the only difference is in the denominator of the sum where NCut uses $assoc(A_i)$ instead of $|A_i|$. Since $assoc(A_i)$ captures the total number of edges from A_i to the rest of the graph, it prevents the pathological type of partition described above. If we let A be a single vertex with one edge, then $NCut(A) = \frac{1}{1} + \frac{1}{1} = 2$. This is substantially larger than the RatioCut of the partition (at least for large dense graphs) and usually leads to optima which are of similar size.

For a full discussion (and proofs!) see [**?**].

Step 6

This step is the hardest and least defined step in this procedure. It is more art than science. We will discuss both aspects below as part of Example 4.3.1.
Further comments and related ideas:

- **Connection with Random Walks:** The consideration of a random walk on a weighted graph gives a nice interpretation of NCut and hence of spectral clustering using the symmetrized Laplacian. The intuition is that a random walk will likely be concentrated in clusters so that the equilibrium distribution of random walk should reveal something about the clustering of the network. We can now consider eigenvalue–random walk convergence:

  Let P be the transition matrix for a regular Markov chain (aperiodic and irreducible). Let's assume that P has eigenvalues $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ with left (row) eigenvectors $u_1,\ldots,u_n$. The row

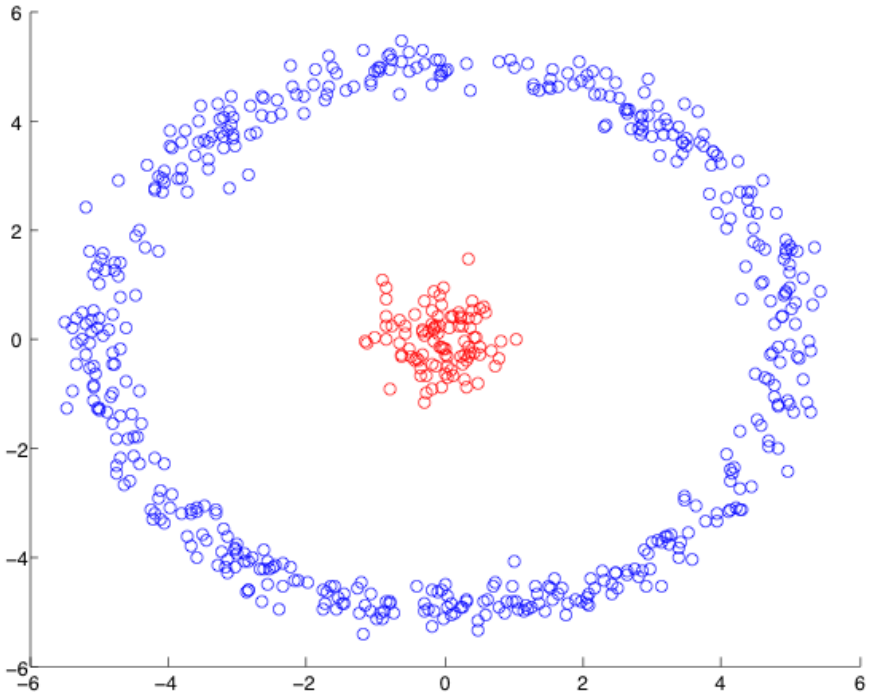


Figure 4.8: Result of applying spectral clustering.

vector u_1 is the equilibrium distribution (which in class we denoted as π). Let U be the matrix with row i given by u_i. Then

$$UP = \Lambda U$$

where Λ is diagonal with λ_i in the i,i position. Thus, note that the columns of U^{-1} are (right) eigenvectors for P (with the same eigenvalues) and in particular, the first column must be a multiple of the vector $(11\ldots1)^t$ (since the row sums of a Markov matrix are equal to 1). Since $UU^{-1} = I$, it must be the case that the first column of U^{-1} is $(11\ldots1)^t$.

Powers of P are given by

$$P^n = U^{-1}\Lambda^n U$$

and in the limit

$$\lim_{n\mapsto\infty} P^n = U^{-1}\begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \ldots & 0 \end{pmatrix} U$$

which by the above (first column of U^{-1} is $(11\ldots1)^t$) implies that P^n goes to the matrix with all rows equal to u_1 and the rate of convergence is governed by λ_2.

As is recapitulated in Luxburg's paper there is a formal connection between random walk and NCut:

$$NCut(A,\overline{A}) = P(\overline{A}|A) + P(A|\overline{A})$$

(for a random walk starting in the equilibrium distribution), so that *NCut* is minimized for partition into groups with the least likely transitions between them.

- PageRank — The *PageRank* algorithm (created by Google's founders Larry Page and Sergey Brin) is (from one point of view) a random walk-based approach to rating webpages. The idea is that a page has great "importance" if a random surfing of the web (within some context) takes you to that page. The question then becomes what is a model of a random surfing of the web. One simple model is as follows: start at a random position, then with probability $1-\alpha$ (for some predetermined $0 \leq \alpha \leq 1$) choose a random webpage and with probability α choose uniformly at random any of the pages linked to the current page and if the current page is not linked to any page, choose a page at random.

  Under this "random surfer" model, the corresponding Markov chain is (generically) regular and thus will have a unique stationary distribution. The convergence to this distribution is in part dictated by the choice of α. Brin and Page initially determined that the value $\alpha = 0.85$ gave reasonable output. The "importance" of a page is then the proportion of time that is spent at that page according to the equilibrium distribution.

- Commute distance: Finally, we only mentioned[2] the idea of *commute distance*, defined between any two states (vertices) as the expected time to travel back and forth between them. Amazingly, it turns out that the square root of this is in fact a distance! From this (using "metric multidimensional scaling") we get a natural way in which to embed this in a Euclidean space.

Example 4.3.1. *As we saw earlier, k-means is a good clustering method for "ball" clusters but failed for data sets which were clustered in obvious but un-ball-like ways (e.g. Figure 4.1). Using spectral clustering on the toy data set from Example 4.1.3 with $\ell = 1$ and $k = 2$ yields exactly the desired clustering: the points in the clump in the middle is assigned to one cluster while points in surrounding circle are assigned to the other. See Figure 4.8.*

4.3.1 Example: S&P 500

In Example 4.1.2, we saw that *k*-means provided a less than clear picture of clustering in the S&P 500 data. In this example, we will apply spectral clustering to compare results. As part of this example we will discuss aspects of Step 6 above - picking appropriate values for ℓ and k. Following the steps, we create the correlation matrix from the data, form S (with $\sigma = 0.5$) and the symmetrized Laplacian. Figure 4.9 shows a

[2] We don't seem to have mentioned it before. Should we keep this in?

histogram of distribution of eigenvalues of this Laplacian. The eigenvalue farthest to the right is the Fiedler value, i.e. the eigenvalue associated to the eigenvector. Notice that there are several "isolated" eigenvalues to the right of the "bulk" where many eigenvalue collect. This gives us a hint at the structure we are looking for - the isolated eigenvalues (and their associated eigenvectors) may be encoding the coarse relationships between the parts of the market.

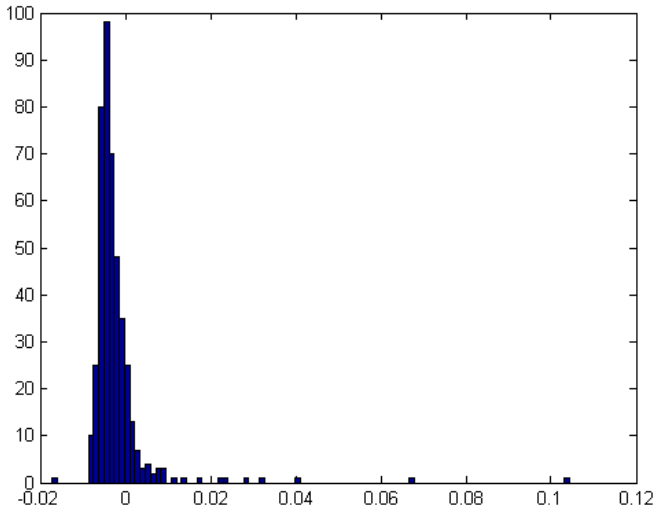


Figure 4.9: Distribution of the spectrum for the Laplacian of the correlation of the S& P 500 data. Note the sequence of outliers that then clump up at the end. The smallest eigenvalue is that which corresponds to the Fiedler vector.

This observation lands us smack in the middle of the difficulties inherent in Step 6. How do we pick the number of actually isolated eigenvalues? What criteria do we use? Once we have that settled, how many clusters do we look for? In practice, we often use a two stage process. The first step is exploratory - we eyeball it, do the computations and see what the results look like. If the first step shows some promise, we use more principled (and unsupervised) yet computationally more expensive methods for making the decisions.

We will explicitly engage in the step step - eyeballing it - not only to illustrate the method (such as it is) but to succinctly illustrate the full spectral clustering methodology. Looking back at Figure 4.9, we see that there are clearly at least 10 eigenvalues to the right of the bulk. To be conservative, we will begin our investigation with $\ell = 10$. In our discussion of NCut and its relation to the symmetrized Laplacian, we learned that ℓ eigenvectors are used to create a partition of size ℓ. So, to begin, we will also take $k = 10$. Figure **??** shows the results. The left hand side of the figure shows the three dimensional spectral embedding of the graph - i.e. the coordinates assigned to the vertices are given by the first three nontrivial eigenvectors.

On the right, we see the same plot annotated by color or shape to indicate the 10 clusters.

This *ad hoc* choice of ℓ and k has produced a reasonable picture. It shows three fairly tight clusters (at least observed through the lens of the three dimensional spectral embedding): white, black and purple. The other clusters are close together but still relatively isolated from one another. This is evidence that spectral clustering is providing reasonable results for this data set, leading us to keep our choice of $\sigma = 0.5$ and proceed with an unsupervised method for picking the other parameters.

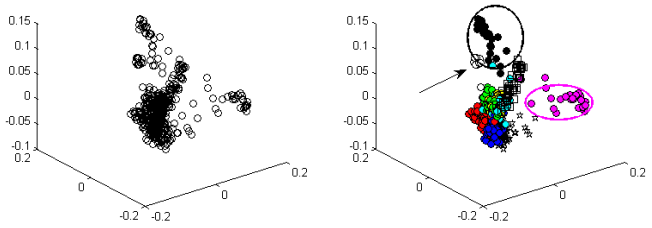


Figure 4.10: (a) Three dimensional spectral embedding of S&P 500 data. (b) Results of spectral clustering with $\ell = 10, k = 10$.

Null models

When estimating parameters for a particular algorithm, it is often useful to build an appropriate null model to aid in the computation. In this context a null model is a mechanism for generating data streams with characteristics that are similar to the data stream we are analyzing. For example, if you believe your data looks much like samples from a particular distribution - say, a beta distribution - you might create a null model by sampling from an appropriate beta distribution or family of distributions.

In our case, the S&P data looks like a fat-tailed Gaussian. So, we could use Gaussians or Gaussian ensembles to generate a null model data stream. Instead, we will introduce a data-driven method of producing a null model: the bootstrap null model. The basic idea of the bootstrap model is to simply randomize the data that you are working with. In our case, for each trading day, we will permute the data for that day randomly. Doing this (independently) for each day creates a data stream which has the same statistics as the original data but should have none of the structure. To use this null model to estimate the parameter ℓ, we calculate the Fiedler value for the null model data. If our randomization has truly destroyed the clustering structure, then the structure inherent in the spectral decomposition is the result of chance. In comparing the null Fiedler value to the eigenvalues of the Laplacian associated to the real data, we declare that eigenvalues of the Laplacian of the real data that are larger than the Fiedler value of the null data are more likely to represent real structure within the data.

To add more evidence to these comparison, we repeat this process many times - randomize data, compute

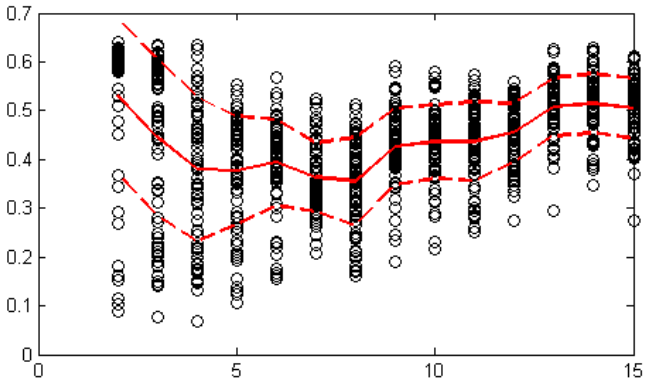


Figure 4.11: Cluster consistency for $k \in \{2, \ldots, 15\}$ with 50 trials.

the Fiedler value of the Laplacian associated to the null model data, and count the number of nontrivial eigenvalues of the Laplacian associated to the real data which are less than that Fiedler value. Taking the minimum of these counts over many such experiments gives us a number of eigenvalues for which we have the best evidence that they encode nontrivial structure.

For example, for the S&P 500 data with 100 instances of the null model we estimated $\ell = 6$.

Number of clusters

We have already seen the difficulty in picking the number of clusters in our discussion of k-means. To pick the number of clusters for spectral clustering, we could use elbowology (section **??**) or cluster consistency (section 4.1) as described above. To illustrate the technique, we use cluster consistency. Figure 4.11 shows the results of the method cluster consistency based on 100 random samples of the data. We see that the minimum of the mean occurs for $k = 8$. Thus, for our unsupervised analysis, we use $\ell = 6, k = 8$.

Figure 4.12 shows results for this combination of parameters just as we did in Figure 4.10. Again, we see the same tight clusters we found earlier (again colored white, black, and purple) and the clusters dividing the main portion again do not overlap significantly.

Validation

But what, if anything, do these clusters mean? Our analysis indicates that they may have significance in explaining variation in the data, but ideally we would like an explanation for their existence. For this, we turn to information external to the data set to help understand the clusters. If we look at the companies whose equities are grouped together, we see the hints of substantial commonalities in the types of businesses that are grouped together. To test this observation, we can look at the sector classifications of the equities and see if they have any relationship to the clusters. Figure 4.13 shows histograms by sector for each of the clusters.

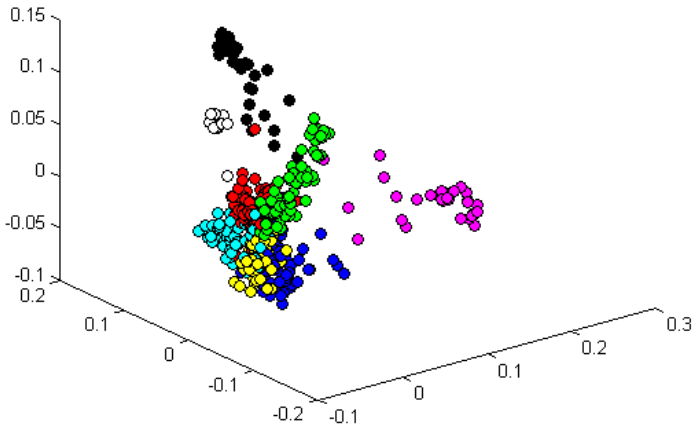


Figure 4.12: Spectral clustering for the S&P 500 data with $\ell = 6, k = 8$

The x-axis in each plot is labeled 1-8 for the eight different sectors (Basic Materials, Consumer Goods, Financial, Healthcare, Industrial Goods, Services, Technology, Utilities). Each histogram given the percent of the cluster classified in each sector and the title give the sector with the highest percentage, together with that percentage. As we can readily see, most clusters are dominated by one sector (particularly the last four) and seven sectors (omitting Consumer Goods) are represented.

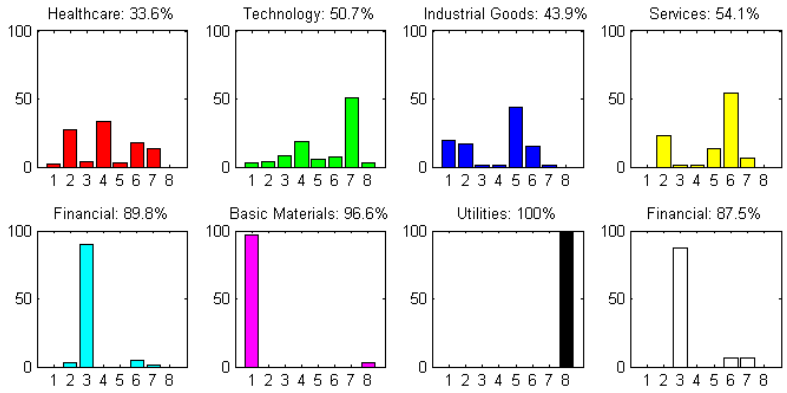


Figure 4.13: Sector classification by cluster for the S&P 500 data with $\ell = 6, k = 8$

Chapter 5

The Partition Decoupling Method

In this chapter we discuss the *Partition Decoupling Method* (PDM), a flexible method that incorporates a mixture of spectral clustering and multiscale decomposition for the analysis of high-dimensional datasets with the aim of extracting multiple layers of interacting networks. That is, our point of view is that most interesting phenomena is a manifestation of multiple (interacting) networks working at multiple time scales and in an interleaved and overlapping manner that is not necessarily hierarchical. The markets are an obvious example, as is the brain, or any large social system.

5.1 PDM Overview

A general framework for the PDM is as follows. Assume some initial data D of feature vectors $\{v_i\}_i$. These feature vectors might be categorical, but regardless, they are numerical.[1]

Given D we now have the following PDM algorithm:

1. Compute the correlation network ρ of D.

2. Clustering of the correlation network of D (using spectral clustering)

3. Compute "low-passed" summary of D (one per cluster - e.g., the cluster mean $= \frac{1}{|C|}\sum_{i\in C} v_i$)

4. "Scrub D" (remove cluster effect and compute the residual features) from original network. This produces the "highpass" information.

5. Compute correlation structure of the low-pass summaries (now a network of size equal to the number of clusters) and compute the highpass correlation network (of size equal to the original network).

[1] As we will indicate at the end of the Chapter, there are possible extensions to situations in which all we have is a similarity matrix as input.

6. Iterate on both low-pass and high-pass networks until indistinguishable either indistinguishable from an appropriate null model or the cluster means are linearly dependent.[2]

A sketch of the output of the PDM is shown in Figure 5.1.

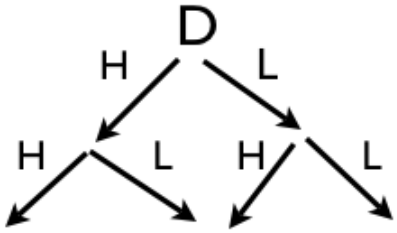


Figure 5.1: A cartoon outline of the PDM process. The "H" denotes the highpass step and the "L" is the lowpass step.

An example: the NYSE Market

Initial data and the correlation network

Here we have as our initial data $X_i(t) =$ close price of equity on day t. So, the time series is the feature vector with the close price on day t as the t^{th} feature. We took a window of 1251 (five year's of trading days) and 2457 equities. We then processed this a bit – in fact, working not with the close prices but the "logarithmic derivative" or fractional change

$$Y_i(t) = \frac{X_i(t) - X_i(t-1)}{X_i(t-1)}.$$

Finally, we then normalized each Y_i by removing the (sample) mean (over t) and dividing by the (sample) variance to give us $D_i(t)$ as the working data. Notice that as vectors, the D_i can be thought as points on the N-sphere where $N = 2457$.

From this, we compute the correlation matrix. Since we've normalized the data this is simply the matrix of inner products

$$\rho_{ij} = \langle D_i, D_j \rangle.$$

Keeping the geometric point of view, correlation is the then the *arccos* of the angle between the equities, which is directly related to chordal distance of the points or the distance between the points given by a

[2]Note that in this situation we can still project onto the lowpass space – span of the means – but the interpretation of the individual vectors is less obvious.

geodesic (arc) on the sphere. At an rate, correlation is related to distance, with closely correlated equities corresponding to points that are close on this sphere.

Clustering the feature vectors

As we mentioned in Chapter 3, given the geometric embedding of the equities in $\mathbb{R}^n$ we could now apply various clustering techniques. We choose to use spectral clustering. Recall the basic algorithm for this:

1. Input: Similarity matrix $S = (s_{ij})$
2. Construct Laplacian - we use the "symmetrized Laplacian" $L_{\text{sym}} = I - D^{-1/2}SD^{-1/2}$ (related to Ncut and RW)
3. Choose number m of significant eigenvectors for a spectral embedding
4. Run k-means (multiple times) to find clusters in the m-dimensional embedding

Let's take each of these steps in turn.

(1) The similarity matrix. We need a similarity matrix derived from the correlation network. This is easily obtained via the relation

$$s_{ij} = exp\left(\frac{-\sin^2(arccos(\rho_{ij})/2)}{\sigma^2}\right).$$

The parameter σ can be fiddled with so as to vary the scale of similarity (a large value of σ smoothes things out, making all things equally similar, while a small value of σ does a better job of distinguishing among the different distances). Note that in this way it is possible to vary the scale at which structure is discovered.

(2) The Laplacian. Recall that D is the diagonal degree matrix. The Laplacian that we use is that whose eigenvectors give a solution to a relaxed version of Ncut. Thus, the spectral embedding given by the first ℓ eigenvectors is an approximation to the best decomposition (in terms of Ncut) of a network into ℓ pieces.

(3) Choice of dimension. Dimensional considerations raise the sometimes thorny issue of the choice of an appropriate null model. Since we are performing spectral clustering (with an underlying model of Ncut), this is really a question of what are the "significant" eigenvectors of L_{sym}. Assuming we are looking N time series of length r, then for this, we compare our correlation matrix with the correlation matrix of N time series of length r drawn from a (standard) Gaussian process. This so-called "Gaussian ensemble" $G(N,r)$ is our null model. For each run we obtain a correlation matrix – we do this 100 times and consider the smallest Fiedler value. Recalling that the Fiedler value is (for a given network) a measure of the best (in the Ncut sense) decomposition of the network/dataset into two pieces, then the smallest of these values represents the best choice of an axis for the data that would give a good separation. Thus, we use this Fiedler value as a benchmark, and in the actual NYSE data, consider all m eigenvalues that are less than or equal to this

and use the associated eigenvectors for the spectral embedding. Note that each of these axes reveals more structure (from this Ncut view) than any axis for a random dataset.

It is also possible to build in more structure to the null model. For example we could assume a hierarchical structure of the form of families of time series who clustered separately and then whose means were, say, from some $G(N', r)$. This kind of *partition decoupled null model* was used in [8]. Details of its construction can be found there.

(4) Clustering. Following the usual spectral clustering yoga, we take the (spectral) embedding determined in $\mathbb{R}^m$ in step (3) and run k-means over and over again, looking for the best decomposition (in terms of discrepancy) into m clusters.

When we do this with at the first nontrivial level of the PDM on the NYSE data, we get forty-nine clusters. Figure 5.2 shows the composition of each of these forty-nine clusters. That is, we use the standard sector labeling and give the distribution of SAIC labels for the learned sector. This is effectively a data-driven notion of sector.

As per the PDM description we compute the means of each of the forty-nine clusters. We now have two paths to pursue: (1) construct the correlation network on the (normalized) cluster means and (2) "Scrub out" the mean effect from the data and start over.

When we do the former, the result is our sector network show that we saw early on in the book, Figure 2.7. Note the apparent "cycle" present in this coarse version of the network. More on that later. In the latter case, a clustering anew the scrubbed market data (and repeating the procedure outline above) we find this time sixty-two clusters.

Remark: It is possible that recent work in computational homology (see e.g., [17]) may also be used to extract the circle or even some other nontrivial manifold structure in the data.

We can continue this kind of process. For the details of the outcome, see [8].

The market data is just one instance of a useful application of the PDM. Another good example comes from its application to microarray data.

Roll call data and the PDM

As noted in the introduction, roll call data can provide a geometric summary, or at least a vector space summary, of a legislative body. In this case the actions of each legislator are summarized by a feature vectors with entries $-1, 1$, and 0 indicating votes of nay, yea, and any other state, on the various issues that come before the body.

In [9] we apply the PDM technique to this kind of data, in particular, looking at the roll call votes for United States Congress at different time periods. To apply the PDM to this data (which we should note is categorical and can be found at [15]) we proceed as above, but with a few significant modifications:

1. The "null model" that we use as comparison for the actual data is a set of roll call votes of the same size (i.e., number and dimensionality) as the original, but with the votes for each legislator randomized.

2. Other

The results of the analysis pick up party affiliation and region as having significant effects on voting record, but also indicate other sorts of influence, thereby articulating a slightly more textured view of "ideology." The tool of "boosting" [11] is applied to identify significant votes. We anticipate that this kind of approach could be usefully applied in other areas that produce vectors of categorical data.

Microarrays and the PDM

Microarrays are effectively high-dimensional summaries of your genetic "activity." The point is that while we are all bags of roughly the same set of genes, these genes are "expressed" in different amounts in each of us. From a given cell sample it is possible to measure the relative abundance of many genes at once.[3] The measurements are obtained by *probes* and a single microarray will carry tens or even hundreds of thousands of probes. The belief is that it is in the multivariate differences in these genetic profiles that our phenotypic differences can be found [2].

The initial microarray data requires various kinds of preprocessing or normalization, generally specific to the actual microarray chip (see e.g[1]). In what follows we assume that all the preprocessing has been done and we're ready to go with the analysis.

Partition decoupling for microarray data proceeds much like the methodology for the markets. A collection of microarray samples produces data in a high-dimensional space of dimension equal to the number of probes. We are interested in using the PDM on this data to explore the correlation structure in the population under investigation. That is, if we are given m samples each with n probes, then we can use the PDM to articulate structure in the n-dimensional data set via the analysis of the correlation structure of the m samples. THere are a few ways in which the PDM has been used in microarray analysis (see [2] for details)[4] :

Population classification. This is a straightforward clustering of the samples according to spectral clustering. Experiments with a set of cancer-related cells produce perfect classification, in great distinction to the results of k-means. In this particular study this first layer clustered according to a cell treatment. The second layer gave some strong indication of clustering according to cell type.

Combining studies. Studies performed on different populations using different chips, but a common collection of probes, can be naively combined by taking the union of the array data. The first layer of the PDM seems to cluster according to experiment. This can be cleaned and the analysis continued. Preliminary results show that phenotypic characteristics are preserved in this way.

[3] In fact, what is actually measured are various protein levels and each protein corresponds to a gene.

[4] In this application a "random rewiring" null model was used for the correlation network. That is, the correlations were randomly permuted many times in order to produce a distribution of Fiedler values. Eigenvalues were considered to be significant if they were in the bottom fifth percentile or smaller.

Pathway analysis. A pathway is a subset of genes identified to be important for some biological process (e.g., metabolism of some protein).[5] Suppose we are given samples from a population of several phenotypes. Let the pathway consist of a subset S of the set of probe indices $\{1,\ldots,n\}$. If X is the original (full) microarray data, then left X_S denote the pathway subset of microarray data. Then we say that the pathway S is significant if (1) there is a nontrivial (as usual, compared to an appropriate null model) clustering of X_S (as revealed by the PDM applied the correlation network on X_S) and (2) If the clustering, say $C_1,\ldots,C_k$, respects, or is at least close to respecting, the partition of the original data according to phenotype – i.e., each cluster is mainly of one phenotype.[6]

The significant pathways can then be scrubbed from the data and the analysis continued.

In [2] this analysis is applied to a population of cells some with cancer, others normal. In this case there is an interesting finding: the fraction of significant pathways (chosen from some a priori set of pathways that were deemed interesting) identified in the first layer is smaller than the number identified in the second layer – i.e., for each pathway found to be significant, its effect can be scrubbed and then the set of pathways can be considered again. When we do this over all pathways, a greater number of significant pathways are discovered. One interpretation of this is that at least **some** of the layer one effect is due simply to the fact that these are all living cells, so that the PDM succeeds in removing the baseline common biological activity or background noise enabling the discovery of underlying activity that is differentially expressed.

Pathway-based classification. Suppose that a pathway respects a given phenotype in the above sense. Given a new sample of unknown phenotype (e.g., a cell sample for which we are looking for a classification of cancer or not), we can try to perform classification by tossing the unknown sample in with the original population and seeing where it classifies. Note that this is, in part, related to the problem of combining studies. Preliminary results of this kind of predictive work are promising.

Exercises.

1. Take the S&P 500 data from the website and "apply" the PDM to it.
2. Other?

[5] The KEGG (Kyoto Encyclopedia of Genes and genomes) database [5] is a publicly available compendium of pathways.

[6] This can be measured using Fisher's exact test.

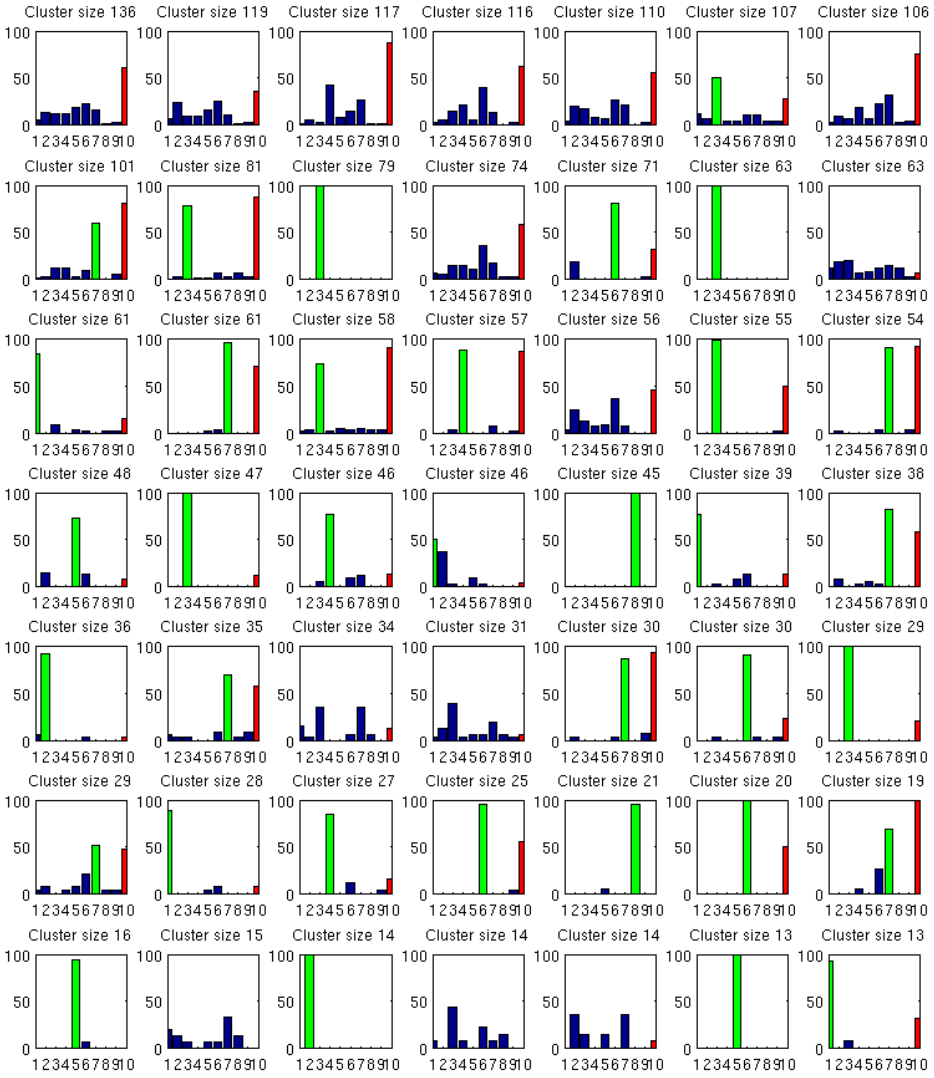


Figure 5.2: We see here the composition of each of the forty-nine clusters in the NYSE data. Each figure contains a histogram of the sector and index breakdown of the clusters. Columns 1–8 represent Yahoo! Finance sector labels. The ninth column is reserved for equities that had no sector labeling. The height of each column is the percentage of the total cluster falling in that category (the cluster size is listed above each histogram). A cluster is defined as "sector dominated" if one sector comprises more than 50% of the cluster. In that case, the sector is denoted in green. The tenth column, in red, gives the percentage of the cluster that is listed on the NASDAQ.

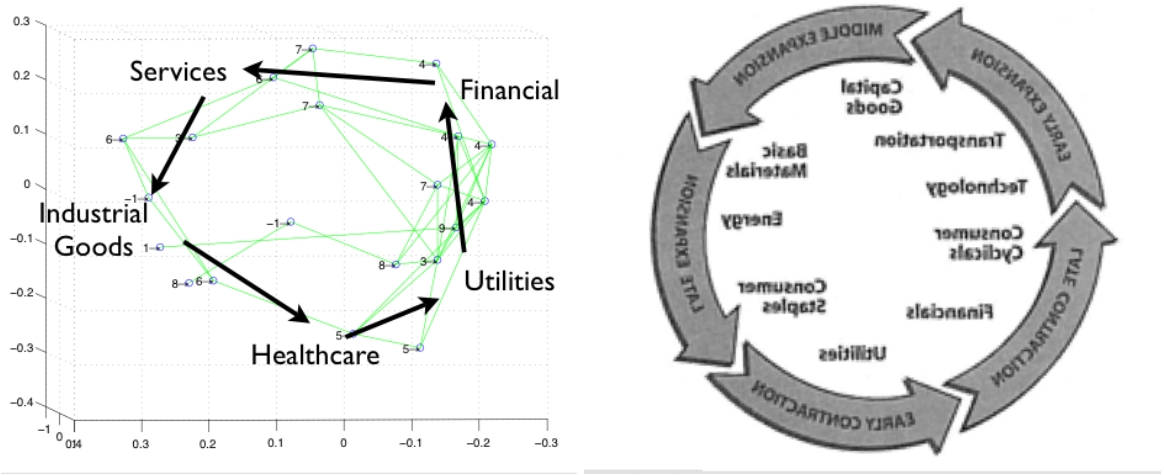


Figure 5.3: Sector flow. Closer investigation of the circle in the network of clusters suggests a manifestation of the well-known phenomenon of "sector flow" whereby capital flows from the financial sector to the services sector to industrial goods to healthcare and then to the utilities sector and back again.

Chapter 6

Topological Data Analysis

As indicated at the end of the previous chapter, we believe that in the analysis of the NYSE market data we believe that we have "articulated" a structure in the data known as "sector rotation," the phenomenon of a cycle of capital flow in the markets from one sector to another.

Brief explanation of what exactly we did - using the Delaunay empty sphere method to show the "persistent" existence of a cycle in the data. This is in the sense of connecting vertices under some range of conditions of being in the same neighborhood. This is akin to the general methods of "topological data analysis" in which one hypothesizes that the dataset (already coming with some kind of embedding in $\mathbb{R}^n$) comes as a sample from some kind of object and then looks to infer the topology of that object.

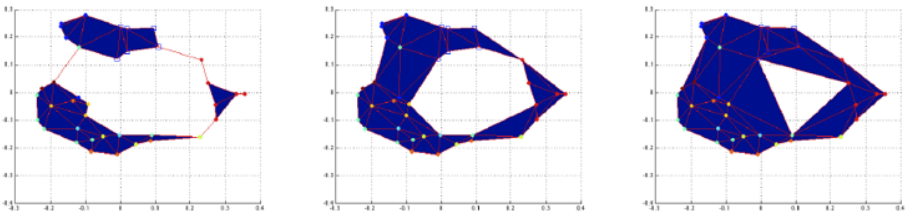


Figure 6.1: Use of Delaunay "Empty Sphere Method" to detect the (persistent) homology in a 2-d MDS of the first scale.

Chapter 7

Further Reading, References

Bibliography

[1] B.M. BOLSTAD, R.A . IRIZARRY, M. ASTRAND, AND T.P. SPEED. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. (2003) *Bioinformatics,* 19(2):185–193.

[2] R. BRAUN, G. LEIBON, S. PAULS, AND D. ROCKMORE (2010). Partition decoupling for multi-gene analysis of gene expression profiling data.

[3] N. FOTI, S. PAULS, D. N. ROCKMORE, AND T. KASTELLE (2010) A longitudinal analysis of the World Trade Web.

[4] D. I. HOLMES AND J. KARDOS (2003). Who was the author? An introduction to stylometry. *Chance 16*, No. 2, 5–8.

[5] HTTP://WWW.GENOME.JP/KEGG/

[6] J. KLEINBERG (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM* **46** (5): 604632. doi:10.1145/324133.324140

[7] P. LAX Linear Algebra and Its Applications.

[8] G. LEIBON, S. PAULS, D. N. ROCKMORE, AND R. SAVELL (2008). Topological structures in the equities market network. *Proceedings of the National Academy of Sciences 101*, 105(52):20589-94.

[9] G. LEIBON, S. PAULS, D. N. ROCKMORE, R. SAVELL, AND M.HERRON (2010). Partition decoupling for roll call voting. *in preparation.*,

[10] S. LYU, D. N. ROCKMORE, AND H. FARID A digital technique for art authentication. *Proceedings of the National Academy of Sciences 101*, 49 (December 2004), 17006–17010.

[11] SOMEONE Something.

[12] M. NEWMAN Modularity and community structure in networks *PNAS* (2006) vol. 103 no. 23, 8577-8582

[13] K. T. POOLE AND H. ROSENTHAL (2007). *Ideology and Congress.* New York: Oxford University Press.

[14] D. N. ROCKMORE AND Y. WANG (2010) Empirical mode decomposition for visual stylometry. Preprint.

[15] WWW.VOTEVIEW.COM

[16] D. J. WATTS AND S. H. STROGATZ (1998) Collective dynamics of 'small-world' networks. *Nature* 393 (6684): 40910. doi:10.1038/30918.

[17] A. ZOMORODIAN (2009) *Computational Homology*