Nonlinear Dimensionality Reduction II: Diffusion Maps

36-462/662, Data Mining

23 September 2019

Contents

1 Introduction

Let's re-cap what we did with locally linear embedding. We start p-dimensional data ($\vec{x}_1, \ldots \vec{x}_n$ in vector form, stacked into a data-frame $\mathbf{x}$), which we suspect are on, or are near, a q-dimensional manifold embedded in the feature space. As an exercise in least squares, we got weights w_{ij} saying how much of data-point $\vec{x}_j$ we needed to use to reconstruct $\vec{x}_i$. Since $\sum_j w_{ij} = 1$, we can think of w_{ij} as saying how similar i is to j. Next, we asked for the new vectors $\vec{y}_i$ which were best reconstructed by those weights, i.e., which minimized

$$\sum_{i=1}^{n} \left\| \vec{y}_i - \sum_{j=1}^{n} w_{ij} \vec{y}_j \right\|^2 \tag{1}$$

under the constraint that $n^{-1}\mathbf{y}^T\mathbf{y} = \mathbf{I}$. We turned this into an eigenvalue problem, that of finding the eigenvectors of

$$\mathbf{M} = (\mathbf{I} - \mathbf{w})^T(\mathbf{I} - \mathbf{w}) \tag{2}$$

It was nice to do this, because finding eigenvectors is something our computers are very good at. These eigenvectors were then the new coordinates of our data points on the low-dimensional manifold.

To introduce diffusion maps, we need to change tack a little from the geometric/optimization approach we've been taking, though we'll come back to it. Notice that when we do LLE, we identify a neighborhood of k points for each point in our data set. We can imagine drawing a graph or network which connects a point to its neighbors. We can also attach weights to the edges in this graph, since we get them from our $\mathbf{w}$ matrix. If the weights are all non-negative, we can use them to define a random walk on the data, and use that random walk to define coordinates.[1]

Let's be a little more formal, and more general. Suppose that we have a function K which takes two data-points $\vec{x}_1$, $\vec{x}_2$ are inputs, and gives us a non-negative real number as an output:

$$K(\vec{x}_1, \vec{x}_2) \geq 0 \tag{3}$$

Further suppose that it's symmetric

$$K(\vec{x}_2, \vec{x}_1) = K(\vec{x}_1, \vec{x}_2) \tag{4}$$

We'll form a matrix $\mathbf{K}$ for our data, where

$$K_{ij} = K(\vec{x}_i, \vec{x}_j) \tag{5}$$

and as a last requirement, insist that the *matrix* be non-negative definite, i.e., for any vector $\vec{v}$

$$\vec{v}^T \mathbf{K} \vec{v} \geq 0 \tag{6}$$

As we saw in the homework, this last is equivalent to requiring that all the eigenvalues of $\mathbf{K}$ be ≥ 0.

Tradition says that we should call $\mathbf{K}$ the **kernel matrix** or **Gram matrix** corresponding to the **kernel**[2] function K. The point of this matrix is, once again, to give us a weighted graph, which tells us which points are tied to which other points, and how similar those neighbors are — it's going to play much the same role in the generalized case that $\mathbf{w}$ did for LLE.[3] The advantage of abstracting away from the least-squares procedure of LLE to "some kernel function or other" is that it lets us deal with cases where least-squares does not apply — because it's not the right metric, because some of the data are categorical, etc.; I will return to the "kernel trick" below, and we'll come back to it again later in the course.

[1] We didn't *require* the weights to be non-negative, but they often will be spontaneously, and it's not, generally, a hard requirement to impose. See Saul and Roweis (2003), if you want the details.

[2] A name whose origins are lost in the mists of 19th century algebra.

[3] If you are getting suspicious because $\mathbf{w}$ is not symmetric, even if we force is entries to be non-negative, good; but notice that what ultimately shows up in finding the coordinates is $\mathbf{w}^T + \mathbf{w} - \mathbf{w}^T\mathbf{w}$, which *is* symmetric.

Given this $\mathbf{K}$, we need one more step to make a random walk on the data graph, which is to turn it into a stochastic transition matrix, where each row adds up to one:

$$\mathbf{D} = \texttt{diag(rowSums(K))} \tag{7}$$

that is, $\mathbf{D}$ is the diagonal matrix whose i^{th} entry is the sum of the i^{th} row of $\mathbf{K}$.[4] Now

$$\mathbf{A} \equiv \mathbf{D}^{-1}\mathbf{K} \tag{8}$$

defines the same graph as $\mathbf{K}$, but it's properly row-normalized. It's the transition matrix for a Markov chain — specifically or a random walk on the graph of the data. The random walk is biased towards making transitions that take it between *similar* points. If we look at the behavior of the random walk, then, it should tell us about the *geometry* of the data (how can we move around the data without ever making a big, discontinuous step?), and about *groups* of similar points (i.e., what are the clusters?). We'll take these two in order.

2 Diffusion-Map Coordinates

Let's say that we want to assign a single coordinate to every point on the data, which we'll write as f, with $f(i)$ being the coordinate for i. Because there are n data points, instead of treating f as a function, we can also treat it as an $n \times 1$ matrix, which for simplicity I'll also write f. (Here, as with PCA and with LLE, starting with a single coordinate, the $q = 1$ case, is just to simplify the initial algebra.) Let's say that we want to minimize

$$\Phi(f) \equiv \sum_{i,j} A_{ij}(f_i - f_j)^2 \tag{9}$$

This should force the coordinates of points which are very similar ($A_{ij} \approx 1$) to be close to each other, but let the coordinates of deeply dissimilar points ($A_{ij} \approx 0$) vary freely.[5] Now that we know what to do with quadratic forms, let's try and turn this into one.

$$\begin{aligned}
\sum_{i,j} A_{ij}(f_i - f_j)^2 &= \sum_{ij} A_{ij}(f_i^2 - 2f_i f_j + f_j^2) && (10)\\
&= \sum_i f_i^2 \sum_j A_{ij} - 2\sum_i f_i \sum_j A_{ij} f_j + \sum_i \sum_j A_{ij} f_j^2 && (11)\\
&= \sum_i f_i^2 - 2\sum_i f_i \sum_j A_{ij} f_j + \sum_j f_j^2 && (12)\\
&= 2f^T f - 2f^T \mathbf{A} f && (13)
\end{aligned}$$

[4] If you don't like my mix of R and matrix notation, you try writing it out in proper matrix form.

[5] This is not *quite* the same as the minimization problem for LLE.

using the facts that $A_{ij} = A_{ji}$ and $\sum_j A_{ij} = 1$. So minimizing Φ is the same as minimizing

$$f^T(\mathbf{I} - \mathbf{A})f \equiv f^T \mathbf{L} f \tag{14}$$

where of course $\mathbf{L} \equiv \mathbf{I} - \mathbf{A}$. The matrix $\mathbf{L}$ is called the **Laplacian** of the graph. Now we impose the constraint that $f^T f = 1$, to avoid the uninteresting minimum at $f = 0$, and as before we get an eigenvalue problem:

$$\mathbf{L}f = \lambda f \tag{15}$$

is the solution. Substituting back in to the expression for Φ, we see that the value of Φ is 2λ, so we want the bottom eigenvectors, as we did with LLE.

For exactly that same reasons that $\mathbf{w1} = \mathbf{1}$ for LLE, we know that $\mathbf{A1} = \mathbf{1}$ — i.e., the vector of all 1s is an eigenvector of $\mathbf{A}$ with eigenvalue 1. It turns out (see below) that this is the *largest* eigenvalue. Now, every eigenvector of $\mathbf{A}$ is also an eigenvector of $\mathbf{L}$, but the eigenvalues change:

$$\begin{aligned} \mathbf{L}f &= \lambda f & (16)\\ (\mathbf{I} - \mathbf{A})f &= \lambda f & (17)\\ f - \mathbf{A}f &= \lambda f & (18)\\ \mathbf{A}f &= (1-\lambda)f = \mu f & (19) \end{aligned}$$

where of course $\mu = 1 - \lambda$. Thus, $\mathbf{1}$ is an eigenvector of the Laplacian with eigenvalue 0. Again as with LLE, we discard this solution as useless.

The next-to-bottom eigenvector of $\mathbf{L}$ has to be orthogonal to $\mathbf{1}$, because all the eigenvectors are orthogonal. This means it must have both negative and positive entries (otherwise $\mathbf{1}^T f = \sum_i f_i > 0$). We will see later how to use the signs of the entries in this eigenvector. What matters for right now is that it gives us a non-trivial coordinate, in which similar data points are close to each other.

If we want q coordinates, then (yet again as with LLE) we just take the bottom $q+1$ eigenvectors[6] $f^{(n)}, f^{(n-1)}, \ldots f^{(n-q)}$ of $\mathbf{L}$, and their eigenvalues $0 = \lambda_n, \lambda_{n-1}, \ldots \lambda_{n-q}$. We discard the bottom one as uninteresting. The **diffusion map** Ψ uses the remaining eigenvectors and eigenvalues to find a point in $\mathbb{R}^q$ corresponding to each of the original data points. Specifically, the image of the point v is

$$\Psi(v) = \left(\mu_1 f_v^{(n-1)}, \mu_2 f_v^{(n-2)}, \ldots \mu_q f^{(n-q)}\right) \tag{20}$$

where I've abbreviated $1 - \lambda_{n-i}$ as μ_i.

The coordinates we get from the diffusion map are related to, but not identical with, the coordinates we would get from LLE.[7] They former are, however, in a reasonable sense the optimal coordinates to represent the original matrix of similarities $\mathbf{K}$ (Lee and Wasserman, 2010).

By now you may be asking what any of this has to do with "diffusion".

[6] Eigenvectors are conventionally numbered starting from 1 at the top.

[7] In fact, in some cases, it can be shown (Belkin and Niyogi, 2003, §5) that the matrix in the LLE minimization problem is related to the Laplacian, because $(\mathbf{I} - \mathbf{w})^T(\mathbf{I} - \mathbf{w}) \approx \frac{1}{2}\mathbf{L}^2$. Since the powers of $\mathbf{L}$ have the same eigenvectors as $\mathbf{L}$, when this holds the coordinates we get from the diffusion map are *approximately* the same as the LLE coordinates.

2.1 Fun with Transition Matrices

The matrix $\mathbf{A}$ is a stochastic transition matrix, so the entries in each row are non-negative and add up to one. Suppose we have a function f on the graph; since there are only n data points, we can represent it as an $n \times 1$ matrix, which for simplicity I'll also write as f. What's $\mathbf{A}f$? Well, it's another $n \times 1$ matrix:

$$(\mathbf{A}f)_i = \sum_{j=1}^{n} A_{ij} f_j \tag{21}$$

In words, the i^{th} entry of $\mathbf{A}f$ is a weighted average of the values of f at i's neighbors; the weight of j is the likelihood of going from i to j. Suppose I write Z_t for the node occupied by the random walk at time t. Then

$$\mathbf{E}\left[f(Z_{t+1})|Z_t = i\right] = (\mathbf{A}f)_i \tag{22}$$

If f encodes a function, then $\mathbf{A}f$ encodes the expected value of that function after one step of the random walk, $\mathbf{A}^2 f$ is the expected function after two steps, and so forth. So, acting to the right, $\mathbf{A}$ forecasts what the value of a fixed function is going to be later.

This is inside-out from (or, a the mathematicians say, "adjoint to") asking *where* the random walk will be later. As you know, if ρ is a distribution over the states of a Markov chain, written as a $1 \times n$ matrix, then

$$\rho\mathbf{A} \tag{23}$$

is the distribution of the Markov chain at the next time step. So from the left $\mathbf{A}$ updates distributions, and from the right $\mathbf{A}$ forecasts functions. Without getting in to the more profound aspects of this duality[8], what we care about here is how this can help us understand the data.

We know (if only because I asserted it during the lecture on page-rank) that $\mathbf{A}$ has a left eigenvector with eigenvalue 1:

$$\rho^{(1)}\mathbf{A} = \rho^{(1)} \tag{24}$$

Moreover, every entry of $\rho^{(1)}$ is > 0, and it is the top (or "dominant") eigenvector, the one with the largest eigenvalue. This means that all the other eigenvectors must have both positive and negative entries.

What happens if we let the walk evolve for many steps undisturbed? Ergodicity happens, that's what. Let's say that the let eigenvectors of $\mathbf{A}$ are $\rho^{(1)}, \rho^{(2)}, \ldots \rho^{(n)}$, with corresponding eigenvalues $1 = \mu_1 > \mu_2 > \ldots \mu_n > 0$. Since th eigenvectors are orthogonal, and there are n of them, they **span** the space — any arbitrary vector can be written as a linear combination of eigenvectors. So start with any initial distribution ρ we like; it's the case that

$$\rho = \sum_{i=1}^{n} a_i \rho^{(i)} \tag{25}$$

[8] It's the difference between the Schrödinger and Heisenberg pictures of quantum mechanics.

How does ρ evolve after t steps of the random walk?

$$\begin{aligned}
\rho \mathbf{A}^t &= \left(\sum_{i=1}^{n} a_i \rho^{(i)} \right) \mathbf{A}^t && (26) \\
&= \sum_{i=1}^{n} a_i \rho^{(i)} \mathbf{A}^t && (27) \\
&= \sum_{i=1}^{n} a_i \rho^{(i)} \mu_i \mathbf{A}^{t-1} && (28) \\
&= \sum_{i=1}^{n} a_i \rho^{(i)} \mu_i^t && (29)
\end{aligned}$$

Since all of the $\mu_i < 1$, except for $\mu_1 = 1$, as we step through the random walk, the distribution comes closer and closer to $\rho^{(i)}$, the invariant distribution. This is ergodicity.

I bring this up not only because it's a cool bit of math, but also because it's relevant to diffusion maps. The last few paragraphs have been all about *left* eigenvectors, but of course there is a close relationship between them and the *right* eigenvectors, which are the coordinates of the diffusion map. Recall what happens when you transpose a matrix product:

$$(\rho \mathbf{A})^T = \mathbf{A}^T \rho^T \tag{30}$$

If ρ should happen to be a left eigenvector of $\mathbf{A}$, then ρ^T is a right eigenvector of $\mathbf{A}^T$, with the same eigenvalue. Since we have arranged for *our* matrix $\mathbf{A}$ to be symmetric, $\mathbf{A} = \mathbf{A}^T$, the left and right eigenvectors are the same. So we know the right eigenvalues of $\mathbf{A}$ — they're the same as the left eigenvalues, starting at 1 and counting down from there towards a lowest value $\mu_n > 0$. I could write the right eigenvectors as $(\rho^{(i)})^T$, but that's ugly and awkward; let's call them $f^{(i)}$ instead.

We've already seen that $\mathbf{A}\mathbf{1} = \mathbf{1}$, so we've identified the top eigenvector: $f^{(i)} = \mathbf{1}$. In other words, if a function is constant, then its expected value after one step is also constant. Averaging a constant just gives you back the constant. What happens to other functions, especially if we are trying to predict multiple steps ahead (i.e., averaging repeatedly)? We'll pull the expansion-in-

eigenvectors trick again.

$$\begin{aligned}
\mathbf{A}^t f &= \mathbf{A}^t \left(\sum_{i=1}^{n} a_i f^{(i)} \right) && (31)\\
&= \sum_{i=1}^{n} a_i \mathbf{A}^t f^{(i)} && (32)\\
&= \sum_{i=1}^{n} a_i \mu_i \mathbf{A}^{t-1} f^{(i)} && (33)\\
&= \sum_{i=1}^{n} a_i \mu_i^t f^{(i)} && (34)\\
&\to a_i \mathbf{1} && (35)
\end{aligned}$$

since all the eigenvalues are ≤ 1. In other words, projected far enough into the future, every function looks constant. Any irregularities in the initial function f get averaged away.

This is where diffusion enters into it. In physics, diffusion refers to the process of *passive* drift of substances[9] away from regions of high concentration and towards regions of low concentration. This is accomplished by particles of the substance taking unbiased random walks. If two adjacent regions of space start with different concentrations of the substance — say it's higher in region 1 than in region 2 — then, even though the individual random walks are unbiased, there will be more random walkers going from region 1 to region 2 than going in the opposite direction; which will tend to reduce then difference in concentrations.

More specifically, if $\rho(\vec{r}, t)$ is the *density* of the substance at the point $\vec{r}$ at the time t, the **diffusion equation** is

$$\frac{\partial \rho}{\partial t} = D \left(\frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial y^2} + \frac{\partial^2 \rho}{\partial z^2} \right) = D \nabla^2 \rho \tag{36}$$

where the last equation (implicitly) defines the **Laplacian** ∇^2, which is sometimes also written as Δ, and D is the diffusion constant.[10] Remember that second derivatives are positive at minima and negative at maxima; this is saying that the substance, whose density is ρ, will move away from regions of high concentration and towards ones of low concentration, until it's evenly distributed everywhere.[11] On the one hand, it's almost Biblical[12]; on the other hand, it's

[9] Or heat; which for many purposes acts roughly like a subtle fluid.

[10] See the appendix.

[11] Actually, precisely linear gradients are *also* left alone by the diffusion equation (why?). This reflects a difference between the finite-dimensional operator $\mathbf{L}$ for graphs, and the infinite-dimensional operator ∇^2 for Euclidean space. (If this comment makes no sense, don't worry; it's mostly intended to reassure anyone who may have taken operator theory and started reading these notes by mistake.)

[12] Isaiah 40:4, "Every valley shall be exalted, and every mountain and hill shall be laid low".

the mathematical expression of what we mean when in ordinary language we say "it oozes" (Haldane, 1985, p. 33).

Without going into all the ramifications of the diffusion equation (a subject that fills volumes), the important thing to notice about it is that $\nabla^2 f$ says how much f changes per unit time. It takes the very simple form it does because Euclidean space is **isotropic** — every direction is the same as every other direction — and flat, uncurved. Now consider a diffusion process which is confined to some manifold — say some substance diffusing over the surface of a sphere. The flow can't be given by ∇^2, since that describes inequalities of concentration in the embedding space, and we don't care about (for instance) the fact that the concentration is > 0 on the manifold and $= 0$ off it — we can't diffuse of the manifold. Rather, each manifold $\mathcal{M}$ has its *own* Laplacian operator, $\nabla^2_{\mathcal{M}}$, which takes into account its curvature and anisotropy.[13] In fact, the connection also goes the other way: knowing a manifold's Laplacian basically tells you its shape, because it tells you how the manifold curves.

This is where we connect back, at last, to the random walk on the graph. $\nabla^2 f$ says how rapidly f changes through diffusion in ordinary Euclidean space. $\nabla^2_{\mathcal{M}} f$ says how rapidly f changes through diffusion on a manifold $\mathcal{M}$. $\mathcal{L}f$ says how much f changes through one step of the random walk on the graph. Suppose we build the graph by uniformly sampling points from $\mathcal{M}$; as we sample more and more densely, our graph looks more and more like a discrete approximation to the continuous manifold, and so $\mathbf{L}$ has to encode the same geometry as $\nabla^2_{\mathcal{M}}$.[14] So $\mathbf{L}$ is estimating the natural or intrinsic "shape" of the data. This remains true even if the data don't come from uniform sampling on a manifold, but some other distribution.

2.2 Multiple Scales

If $\mathbf{A}$ is the transition matrix for a Markov chain, then so is $\mathbf{A}^m$, for any integer m — it's just taking m steps at a time, rather than one. It's easy to show (see exercises) that $\mathbf{A}^m$ has the same eigenvectors as $\mathbf{A}$, but different eigenvalues — specifically, the eigenvalues of $\mathbf{A}$, all also raised to the power m. We can therefore look at diffusion maps defined not by one step of the random walk but by m steps. The advantage of doing so is that it tends to reduce the influence of small-scale local noise. Computationally, this is because it shrinks the small eigenvalues rapidly, meaning those coordinates become less influential.

[13] You might want to try to work out what the Laplacian is for a sphere, say in spherical coordinates.

[14] The exact sense in which this is true is fairly subtle. On the one hand, $\mathbf{L}$ is an $n \times n$ matrix, so multiplying by it transforms vectors in $\mathbb{R}^n$ into other vectors in $\mathbb{R}^n$. On the other hand, $\nabla^2_{\mathcal{M}}$ takes functions to functions — points in $\mathbb{R}^{\mathcal{M}}$ to $\mathbb{R}^{\mathcal{M}}$. Similarly, $\mathbf{L}$ represents an *amount* of change in *one time step*, while $\nabla^2_{\mathcal{M}}$ represents a *rate* of change *per unit time*. Roughly speaking, we need to integrate $\nabla^2_{\mathcal{M}}$ over the duration of a time-step, call it h. This gives us an operator $\mathcal{L}_h$ defined through $\mathcal{L}_h f = e^{h\nabla^2_{\mathcal{M}}} f$. If we evaluate $\mathcal{L}_h f$ only at points on the graph, we should have $\mathcal{L}_h f \approx \mathbf{L}f$. See Grimmett and Stirzaker (1992) for an introduction on how discrete-time Markov chains relate to continuous-time Markov processes, and Lee and Wasserman (2010) for a precise statement of what's going on in the present case.

Stochastically, it's because, by ergodicity, where the random walk is after m steps depends less on its starting position than where it is after 1 step. Of course if m is very large, all the points blur into each other (again by ergodicity), so m is best seen as a control setting.

2.3 Choosing q

It can be shown (through a *very* complicated argument; Lee and Wasserman 2010) that how accurately the diffusion map reconstructs the underlying geometry depends on the ratio

$$\frac{\sum_{i=1}^{q} \mu_i}{\sum_{j=1}^{n} \mu_j} \tag{37}$$

where the μ are the eigenvalues of $\mathbf{A}$.[15] This suggests that a practical rule for choosing q is to fix a value for this ratio, and use the smallest q which achieves it. This is the default implemented in the `diffuse` function in the CRAN package `diffusionMap`.

3 What to Do with the Diffusion Map Once You Have It

First, everything you might do with ordinary vector-valued data can be done with the diffusion coordinates. You can do similarity search, you can look for clusters (for instance, k-means; there's a function to do this nicely in `diffusionMap`), classification, etc. We'll be looking at regression after the midterm, and you can use the diffusion-map coordinates as the input variables in a regression.

3.1 Spectral Clustering

One cute application of diffusion maps is to clustering. Remember that only the top eigenvector of $\mathbf{A}$ is all positive; all the other eigenvectors have both positive and negative entries. What does the difference in signs mean?

To be concrete, let's think about $\rho^{(2)}$, the next-to-top eigenvector. Every point gets either a positive or a negative sign in the eigenvector. Suppose we want to start with a distribution which is concentrated solely on the points with positive sign. We can decompose any such distribution into the eigenvectors:

$$\rho = \rho^{(1)} + a\rho^{(2)} + \sum_{j=2}^{n} b_j \rho^{(j)} \tag{38}$$

but now a will be large and positive, enhancing the probability of the positive points and lowering that of the negative ones, and the b_j will be small. (The b_j

[15] More exactly, the accuracy depends on the ratio of the sums of the population quantities of which these eigenvalues are estimates, and n in the denominator has to go to infinity. I told you it was complicated.

might even manage to be zero.) After one time step, this will become

$$\rho\mathbf{A} = \rho^{(1)} + a\mu_2\rho^{(2)} + \sum_{j=2}^{n} b_j\mu_j\rho^{(j)} \tag{39}$$

. . . which looks rather like ρ: while it's closer to being uniform than ρ was, since the eigenvalues are all < 1, it's still concentrated on the positive points of $\rho^{(2)}$, since the eigenvalues are decreasing and the b_j are small. In other words, if we start with a distribution concentrated on the positive points of $\rho^{(2)}$, it will tend to stay there, though diffusion will disperse it eventually. Things would work exactly the same if we concentrated the initial probability on the negative points — the only difference would be that $a < 0$.

What about the other eigenvectors? Well, we could make a similar argument, but, because eigenvalues are smaller, initial concentrations on the positive points will diffuse away more quickly — how much more quickly will depend on the ratios of the eigenvalues. If we had to split the graph into two parts with the minimum diffusion between them, we'd split it into the points with positive and negative coordinates in $\rho^{(2)}$.

How does this relate to *clustering*? Well, suppose the data fall into two or more clusters, within which all the points are much more similar to each other than they are to outsiders. We would like to call these clusters. We could then re-arrange the kernel matrix $\mathbf{K}$ so it's block-diagonal, or nearly so. Then $\mathbf{A}$ will also be nearly block-diagonal. Thus a random walk which starts inside one of the blocks will tend to stay inside the block for a long time. This means that all the points in a block should have the same sign in at least one of the eigenvectors. The signs of the eigenvectors, in other words, act like indicator functions, or linear combinations of indicator functions, for the clusters.

In the most basic form of spectral clustering, we first divide the data into two clusters by the signs of entries in $\rho^{(2)}$. Having done that, we can further sub-divide the clusters by the signs of $\rho^{(3)}$ and so forth. This is a top-down (**divisive**) hierarchical clustering. At some point it's no longer worth it to keep splitting, and we should instead use the remaining eigenvectors as coordinates within each cluster.

3.2 Asymmetry

We required the matrix $\mathbf{K}$ we started from to be symmetric; consequently the transition matrix $\mathbf{A}$ was too. How much of what we've done depends on symmetry, and how much would apply to any Markov chain over the data?

Most of it carries through, actually. It's no longer the case that the left and right eigenvectors are the same, but the left and right eigenvalues are. And it's still the case that the top right eigenvector is $\mathbf{1}$, that all of the others have both positive and negative signs, etc. (Also, the non-dominant left eigenvectors all have both positive and negative signs.) It's also the case that the signs of the entries in the non-dominant eigenvectors correspond to clusters.

4 The Kernel Trick

Everything that went before rested on having a kernel matrix $\mathbf{K}$, derived from a kernel function. *Where* that function came from, or what kind of data it took as arguments, was irrelevant. The data could be Euclidean vectors and the kernel the ordinary inner product, but nothing required that. So long as the kernel function was mathematically OK, nothing else mattered. This is a very powerful idea in data mining: we can split our problem into (a) finding an algorithm (for prediction or clustering or whatever) that works in terms of kernels, and (b) finding a kernel function for our representation. Then we can recycle algorithms across problems. We will return to kernel methods repeatedly, but I want to close with a small illustration of how they can be powerful, something often called "the kernel trick".[16]

Suppose that we have data points $x_1, x_2, \ldots x_n$ (which may be vectors or something else). Each point is represented by certain features, but we don't think those are really the right ones for our problem. Instead, we have q different functions $\psi_1, \psi_2, \ldots \psi_q$, and we really think those are the appropriate features. So we'd like to work with the vectors

$$\Psi(x) = (\psi_1(x), \psi_2(x), \ldots \psi_q(x)) \tag{40}$$

Let us say that $K(x_i, x_j)$ is the inner product of $\Psi(x_i)$ and $\Psi(x_j)$:

$$K(x_i, x_j) = \sum_{k=1}^{q} \psi_k(x_i)\psi_k(x_j) \tag{41}$$

Since this is an inner product, it's got all the properties we need for a kernel function — symmetry, forms a positive-definite matrix, etc. So if I can express my problem in terms of inner products of the transformed features Ψ, I can also write it in terms of the kernel function K on the *original* features. In particular, if I can find a formula for the right-hand side of Eq. 41, then I never have to calculate the functions ψ at all. In fact, I can even let q go to infinity!

A concrete and very useful example is the Gaussian kernel:

$$K_h(x_i, x_j) = \frac{1}{\sqrt{2\pi h^2}} \exp -(x_i - x_j)^2/2h^2 \tag{42}$$

(Sometimes you see this without the normalizing factor.) The features here are actually *all* the powers of x_i and x_j, though they're not all equally weighted. This is in fact what people typically use with diffusion maps, rather than the Euclidean inner product, because the Gaussian distribution is preserved under diffusion.

[16] For much more about kernel methods in data mining, see Shawe-Taylor and Cristianini (2004).

References

Belkin, Mikhail and Partha Niyogi (2003). "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation*, **15**: 1373–1396. URL `http://www.cse.ohio-state.edu/~mbelkin/papers/LEM_NC_03.pdf`. doi:10.1162/089976603321780317.

Grimmett, G. R. and D. R. Stirzaker (1992). *Probability and Random Processes*. Oxford: Oxford University Press, 2nd edn.

Haldane, J. B. S. (1985). *On Being the Right Size, and Other Essays*. Oxford: Oxford University Press. Edited and introduced by John Maynard Smith.

Lee, Ann B. and Larry Wasserman (2010). "Spectral Connectivity Analysis." *Journal of the American Statistical Association*, **105**: 1241–1255. URL `http://arxiv.org/abs/0811.0121`. doi:10.1198/jasa.2010.tm09754.

Saul, Lawrence K. and Sam T. Roweis (2003). "Think Globally, Fit Locally: Supervised Learning of Low Dimensional Manifolds." *Journal of Machine Learning Research*, **4**: 119–155. URL `http://jmlr.csail.mit.edu/papers/v4/saul03a.html`.

Sethna, James P. (2006). *Statistical Mechanics: Entropy, Order Parameters, and Complexity*. Oxford: Oxford University Press. URL `http://pages.physics.cornell.edu/sethna/StatMech/`.

Shawe-Taylor, John and Nello Cristianini (2004). *Kernel Methods for Pattern Analysis*. Cambridge, England: Cambridge University Press.

Exercises

1. Let $\mathbf{B}$ be any $n \times n$ matrix. Show that if $\vec{v}$ is an eigenvector of $\mathbf{B}$, then it is also an eigenvector of $\mathbf{B}^2$, and of any power of $\mathbf{B}$. Conclude that $\mathbf{B}$ and $\mathbf{B}^2$ have the *same* eigenvectors. (*Hint:* how many eigenvectors does each matrix have?) What happens to the eigenvalues?

A From Random Walks to the Diffusion Equation

Think of a one-dimensional space with coordinate x, and a concentration $\rho(x,t)$ of particles of some substance at the point x at time t. Divide the axis into little intervals of width h, small enough that ρ is approximately constant over each interval. Also, let's divide time up into intervals of duration τ, again very small. Fix a point x at time t. How does the concentration change?

Assume that the particles each take independent random walks, and that the time it takes them to make one jump of the talk is much less than τ; also that the step-size is much less than h. The expected change in position of any one particle over the time-period τ is zero, and the variance $\propto \tau$. Thus

$$h\rho(x,t+\tau) \approx h\rho(x,t) - \frac{\tau D}{h^2} h\rho(x,t) + \frac{1}{2}\frac{\tau D}{h^2} h\rho(x-h,t) + \frac{1}{2}\frac{\tau D}{h^2} h\rho(x+h,t) \quad (43)$$

$h\rho$ is the number of particles in the interval of length h around x (because ρ is roughly constant over the interval). The left-hand side is the new number of particles in that interval. The first term on the right hand side is the old number. The second is the number of particles which jumped out of the interval — this will shrink as $\tau \to 0$, because there's less time for them to jump. In fact, since the variance of a random walk's position grows $\propto t$, this number really depends on τ/h^2 — D is a proportionality constant. The third and forth terms count particles that jump *into* the interval from adjacent intervals.

$$\begin{aligned}
\rho(x,t+\tau) - \rho(x,t) &\approx \frac{\tau D}{h^2}\rho(x,t) + \frac{1}{2}\frac{\tau D}{h^2}\rho(x-h,t) + \frac{1}{2}\frac{\tau D}{h^2}\rho(x+h,t) \\
\frac{\rho(x,t+\tau) - \rho(x,t)}{\tau} &\approx \frac{D}{2}\left(\frac{(\rho(x-h,t) - \rho(x,t)) + (\rho(x+h,t) - \rho(x,t))}{h^2}\right) \\
\rightarrow \frac{\partial \rho}{\partial t} &= \frac{D}{2}\frac{\partial^2 \rho}{\partial x^2}
\end{aligned}$$

letting τ and h shrink to zero.

Since D is just a proportionality constant, re-define it to absorb the factor of $1/2$. This gives the one-dimensional diffusion equation, and the argument for the three-dimensional version is entirely parallel, only with more book-keeping.

Of course, the idea that the particles of some substance all take independent random walks at discrete time intervals is a fable. We can pass to continuous time by making the durations and the step-sizes of the random walk smaller and smaller. The limit is a continuous stochastic process called **Brownian motion** (after Robert Brown, who discovered it experimentally in 1827) or the **Wiener process** (after Norbert Wiener, who worked out all the mathematical details in 1922). That the particles move *independently* of each other is a bigger assumption. It's a reasonable approximation when they are not too highly concentrated — say molecules of perfume diffusing through the air of a room, rather than the motion of the air molecules themselves. Chapter 2 of Sethna (2006) is a good place to start.