

We usually think of a graph visually, i.e. we see vertices as points on a plane or in space, with edges as lines connecting them. When specified algebraically (e.g. by its adjacency matrix), the geometrical imagery is lost. Can we find a natural embedding of a graph in the plane or in space given the adjacency matrix? This is the question for the lecture. It will turn out that such an embedding can indeed be derived. Whether it fits our bill of “naturalness” is really a subjective, almost artistic question, and unlikely to have a crisp unique answer. We will use the resulting embeddings for two purposes: drawing a graph on paper, and partitioning a graph into large parts by removing few edges. In both cases, it will be seen that our approach is reasonably satisfactory.

The starting point for our construction is the edge-node incidence matrix B of a graph G with n vertices and m edges. B has n rows and m columns, and $b_{ie} = 1$ if vertex i is an endpoint of edge e , and 0 otherwise. Clearly, every column of B has exactly two 1s and rest 0s. Note that $BB^T = A + D$ where A is the adjacency matrix, and D a diagonal matrix in which the i th entry equals the degree d_i of vertex i . We will later consider the oriented incidence matrix in which one of the entries in each column is negated, corresponding to an orientation of the edge.

The first point to note is that the elements of the i th row B_i of the incidence matrix B can be thought of as the coordinates of vertex i in m dimensional space. In some sense, this is a nice embedding: $B_i \cdot B_j > 0$ if (i, j) is an edge, and 0 otherwise. If the graph is regular, then $\|B_i\| = \|B_j\|$ and hence in this embedding connected vertices are seen to be closer to each other than non connected vertices. This clearly seems desirable. For non-regular graphs, the situation can be fixed as follows: we define a normalized incidence matrix C where $c_{ie} = b_{ie}/\sqrt{d_i}$. Alternately $C = D^{-1/2}B$. The matrix C can be seen to have the property that unconnected vertices are located farther than connected vertices. Note however, that lengths of different edges is different, whereas it was the same for the regular case.

The matrices B or C place the vertices as a point cloud in m dimensional space. We will project the point cloud onto two dimensions to get a good picture of the graph, and onto a line to determine how to partition it. The key question is what direction to use for the projection.

When we project, we would like to preserve most distances. Say we have an actual 3 dimensional cloud. What would be the best picture of it? If there is some direction in which it is thin, then it would seem that we should really project it in that direction. This would give as big a picture, with as few points falling on top of one another, and as little distortion of distances between points as possible.

If we think of the point as an ellipsoid, with some length, breadth, width, oriented arbitrarily, we would like to know which are the directions in which the length, breadth, and width (smallest) are oriented. Then we simply project in the direction of the width.

Singular value decomposition allows us to find such directions. It is very intimately related to eigenvalue decomposition.

1 Singular value decomposition

Note that the discussion in this section applies to any matrix B , not just the incidence matrix considered above.

Given an $n \times m$ matrix B , its first (right) singular vector, customarily denoted as v_1 , is defined as a unit vector that maximizes $\|Bv_1\|_2$. In order for the norm to be maximized, it is necessary, in some sense, that v_1 approximate the “dominant” direction among the rows of B . This is the intuition we are interested in.

The first left singular vector could be defined as the right singular vector of the matrix B^T . However, there is an elegant joint definition of both left and right singular vectors:

$$u_1, v_1 = \text{unit vectors that maximize } u_1^T B v_1$$

Note that this immediately shows that the direction of Bv_1 must be the same as that of u_1 , and that of $u_1^T B$ the same as that of v_1^T . Further, we define $\sigma_1 = \|Bv_1\|_2 = \|u_1^T B\|_2$ as the first singular value.

Next we formalize the intuition that u_1 is a “rough” estimate of the directions of the rows of B . In particular, noting that Bv_1 gives the projections of the rows of B on v , the rank 1 matrix $B_1 = Bv_1v_1^T = \sigma_1 u_1v_1^T$ can be considered to be a rank 1 approximation to B . In fact we can prove that B_1 is the best such matrix in the sense of the Frobenius norm:

$$\|B - B_1\|_F = \min_R \|B - R\|_F$$

where R is any rank 1 matrix. Note that the Frobenius norm $\|B\|_F$ of a matrix is simply the sum of the squares of all its entries.

Writing $B' = B - B_1$, we simply repeat the above procedure on B' to get u_2, v_2, σ_2 and so on. Clearly the rank of B' will be 1 less than that of B , and hence the process will terminate after r steps, where r is the rank of B . Let u_i, v_i, σ_i , for $i = 1$ to r be obtained in this manner. These respectively give the i th left singular vector, the i th right singular vector, and the i th singular value. B_k is defined as $\sum_{i=1}^k \sigma_i u_i v_i^T$. Obviously $B_r = B$.

Since in $B - B_1$ we are removing from each row its projection on v_1 , we know that the rows of B' must be orthogonal to v_1 . Thus it follows that v_2 is orthogonal to v_1 and so on. Thus the vectors v_i are orthogonal to each other, and similarly u_i amongst themselves.

It is customary to define U_k as consisting of the matrix made up by using $u_1, \dots, u_k$ as its columns. Likewise V_k . Σ_k is defined to be a $k \times k$ diagonal matrix with $\sigma_1, \dots, \sigma_k$ along the diagonal. Then clearly, $BV_r = U_r \Sigma_r$. Let V be any orthogonal matrix obtained by extending V_r , i.e. by adding any columns that are orthogonal to the columns in V_r and to each other. Note that these newly added columns must be orthogonal to the rows of B . Likewise add columns to U_r to extend it to an orthogonal matrix. Also extend Σ_r into an $n \times m$ matrix by adding 0s. Then we have $BV = U\Sigma$. Alternately, we can write this as a decomposition of B , noting that $V^T = V^{-1}$.

$$B = U\Sigma V^T$$

This is the Singular Value Decomposition (SVD) of B and it can be computed in time $O(mn^2 + m^2n)$, see [2].

We note that it is easily proved that B_k is the best rank k approximation to B :

$$\|B - B_k\|_F = \min_R \|B - R\|_F$$

where R is any rank k matrix. Further the error $\|B - B_k\|_F^2 = \sigma_{k+1}^2 + \dots + \sigma_r^2$.

1.1 Examples

Our first example decomposition is

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.85 & -0.53 \\ 0.53 & 0.85 \end{pmatrix} \begin{pmatrix} 1.62 & 0 \\ 0 & 0.62 \end{pmatrix} \begin{pmatrix} 0.53 & 0.85 \\ -0.85 & 0.53 \end{pmatrix}$$

Note here that direction of the first right singular vector $\begin{pmatrix} 0.53 \\ 0.85 \end{pmatrix}$ is somewhere between the directions of the rows of A .

Our second decomposition example is

$$B = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

In this case note that the first right singular vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ snaps to the direction of the first row, does not point somewhere in between the directions of the two rows. In some sense it can be seen to identify the *dominant* direction. This happens when the rows are orthogonal, unlike the case for matrix A earlier.

Our final example consists of the matrix

$$C = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

C has a block diagonal structure, with the blocks being of size 3×3 and 3×4 . In this case it turns out that the matrix inherits the singular values of the blocks. The singular vectors are also suitably padded by zeros and inherited. In this case the largest singular value 2.36 comes from the second block, and the second largest, 2.00 from the first block. Both these are singular values of C . The first singular vector of the second block is $(.56 \ .44 \ .44 \ .56)^T$, and so the first singular vector of C will be $(0 \ 0 \ 0 \ .56 \ .44 \ .44 \ .56)^T$. The next singular value 2 comes from block 1. The singular vector in that block is $(0.58 \ 0.58 \ 0.58)^T$. So the second singular vector for C will be $(0.58 \ 0.58 \ 0.58 \ 0 \ 0 \ 0 \ 0)^T$.

1.2 Singular values vs. Eigenvalues

We have that $u_i^T B B^T = \sigma_i v_i^T B^T = \sigma_i (B v_i)^T = \sigma_i^2 u_i$. Thus u_i is an eigenvector of $B B^T$ of eigenvalue σ_i^2 . Likewise v_i is a eigenvector of $B^T B$ of eigenvalue σ_i^2 .

The matrices BB^T and B^TB are symmetrical matrices, $n \times n$ and $m \times m$ respectively. So we expect them to have respectively n and m orthogonal eigenvectors. On the other hand, our process of determining singular vectors only give r singular vectors, where r is the rank of B . How do we get the additional eigenvectors? This may be resolved as follows. We simply interpret the extra columns (if any) added to V also as right singular vectors of singular value 0 – notice that these columns are orthogonal to each row of B and hence their product with B is indeed 0. Likewise we may interpret the additional columns added to U as left singular vectors, also of singular value 0.

While singular vectors can be thought of as approximating the directions of the rows/columns of a matrix, such an interpretation is not interesting for eigenvectors.

2 Drawing regular graphs

Let $u_1, u_2, \dots$ and $v_1, v_2, \dots$ and $\sigma_1, \sigma_2, \dots$ respectively denote the left singular vectors, right singular vectors and the singular values of B . Then v_1 gives an approximation to the direction of the rows of B in the Frobenius norm. It will be seen that the point cloud is relatively far from the origin, and hence v_1 merely gives the general direction in which the point cloud lies. From the point of view of understanding the shape of the point cloud, v_1 is not interesting.

The second right singular vector v_2 is an approximation to the general direction of the rows of A after removing the components in the direction of v_1 . It turns out that this roughly discovers the “length” of the point cloud! The next singular vector v_3 likewise discovers the “breadth”, albeit in a heuristic sense.

So for the purpose of drawing, we consider the dimensions v_2, v_3 . So we need coordinates of all vertices along these dimensions. The coordinate of a vector B_i along direction v_2 is simply $b_i \cdot v_2$. Thus $B \cdot v_2$ gives the coordinates of all vertices in a single vector. Note further that $B \cdot v_2 = \sigma_2 u_2$. Thus u_2 gives the coordinates of each vertex in the direction v_2 (within scaling) and likewise u_3 the coordinate along direction v_3 .

Next, note that u_2 is the second left eigenvector of BB^T . But $BB^T = A + D$. For a k -regular graph, $D = kI$. Hence BB^T has the same eigenvectors as A , or for that matter $D^{-1}A$. So for regular graphs, this discussion indicates

that we may draw a graph using the eigenvectors of A or those of $W = D^{-1}A = A/k$. This is what you did in homework 7.

2.1 General graphs

It appears that using the eigenvectors of the walk matrix also works reasonably for general graphs[3]. Note however that W does not have the same eigenvectors as BB^T for general graphs. Nor are the eigenvectors of W the same as those of CC^T , although they are related, as will be seen in the next section.

Let x_2, x_3 be the second and third eigenvectors, and α_2, α_3 be the second and third eigenvalues of W . Then $Wx_2 = \alpha_2 x_2$, and hence $x_2(i) = (Wx_2)_i / \alpha_2 = (\sum_{j|(i,j) \in E} x_2(j) / d_i) / \alpha_2$. Now $\sum_{j|(i,j) \in E} x_2(j) / d_i$ is simply the mean of the coordinates of the neighbours of i . Thus this embedding places vertex i at $(x_2(i), x_3(i))$ which is nearly the center of mass of where its neighbours are placed, if α_2, α_3 are reasonably close to 1. For the binary hypercube Q_n , the second and third eigenvalues are $1 - 2/n$, and are thus close to 1 for large n . A similar situation also holds for the grid.

3 Graph Partitioning

We have looked at the mincut problem, where we wanted to split a graph by removing as few edges as possible. However, in that problem, there was no restriction on how large the parts had to be.

In many applications, it is required that a none of the parts be too small. Say you have a circuit that you want to place on multiple printed circuit boards. Typically, within a board there can be many wires, but each board has only a few “pins” to connect to the external world. So the partitioning must have the property that relatively large subgraphs are created, but with as few wires connecting the subgraphs as possible. In divide and conquer graph algorithms, it is again useful to split graphs into equal parts; the amount of work that needs to be done while putting together the solutions for the parts depends upon the number of edges cut, and this needs to be minimized.

We might ask for graph bisection, and ask for algorithms that find the *bisection width*, i.e. the minimum number of edges that need to be removed

so that the resulting two subgraphs have equal number of vertices. In many cases, a more general definition might be more useful.

Let $G = (V, E)$ be the graph and $S \subset V$. Let $E(S, V - S)$ denote the number of edges with one endpoint in S and the other in $\bar{S} = V - S$.

The sparsity of a cut $(S, \bar{S})$ is defined as

$$\text{sp}(S, V - S) = \frac{|E(S, V - S)|}{\min(|S|, |V - S|)}$$

The sparsity of a graph is defined as $\text{sp}(G) = \min_S \text{sp}(S, V - S)$.

Another measure is the ratio of a cut:

$$r(S, V - S) = \frac{|E(S, V - S)|}{|S||V - S|}$$

It is easily seen that $r(S, V - S)|V|/2 \leq \text{sp}(S, V - S) \leq r(S, V - S)|V|$, so these measures are not really too different. The latter might be preferred only because it seems to avoid the non-linear operator $\min$. The cut ratio for a graph is defined as $r(G) = \min_S r(S, V - S)$.

We would like to find the (cuts corresponding to) $r(G)$ or $\text{sp}(G)$.

3.1 Formulation as a numerical optimization

This formulation is from Spielman[3]. Represent a set of vertices S by its characteristic vector x , i.e. $x_i = 1$ if $i \in S$ and 0 otherwise. We first consider the ratio cut $r(S) = \frac{|E(S, V - S)|}{|S||V - S|}$ because it is algebraically easier:

The numerator may be written as $\sum_{(i,j) \in E} (x_i - x_j)^2$; for each edge that crosses from S to $V - S$ we get a 1, so that the total is $|E(S, V - S)|$.

The denominator requires a contribution of 1 for every i, j where $i \in S, j \in \bar{S}$. Thus we may write this as $\sum_{i > j} (x_i - x_j)^2$. So the problem of finding a set of minimum ratio is equivalent to solving

$$r_G = \min_S r(S) = \min_{x \in \{0,1\}^n} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_{i > j} (x_i - x_j)^2}$$

Solving for integer values is harder than solving for reals, typically. So we let x range over real vectors. This will allow the right hand side expression to possibly take even smaller values than what might be possible with integers, so we only get a lower bound. Note further that since only differences $x_i - x_j$

appear in the expression, we can uniformly increment all numbers without changing the value. So we may assert $\sum_i x_i = 0$, i.e. $x \cdot \mathbf{1} = 0$, or $x \perp \mathbf{1}$. Thus we get

$$r_G \geq \min_{x \perp \mathbf{1}} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_{i > j} (x_i - x_j)^2}$$

We next define something truly magical: the Laplacian matrix $L = D - A$. It may be seen that $x^T Lx = \sum_{i,j} l_{ij} x_i x_j = \sum_i d_i x_i^2 - \sum_{i \neq j} a_{ij} x_i x_j = \sum_i d_i x_i^2 - \sum_{(i,j) \in E} 2x_i x_j = \sum_{(i,j) \in E} (x_i - x_j)^2$. Further note that the denominator

$$\sum_{i > j} (x_i - x_j)^2 = (n-1) \sum_i x_i^2 - 2 \sum_{i > j} x_i x_j = n \sum_i x_i^2 - \left(\sum_i x_i \right)^2 = n \sum_i x_i^2$$

since $\sum_i x_i = 0$ by assumption. Thus we get

$$r_G \geq \min_{x \perp \mathbf{1}} \frac{x^T Lx}{n x^T x}$$

Notice that the denominator $x^T x$ in this expression is simply the Euclidean length of x . Asserting that this be 1 does not change the minimum. Thus we may write:

$$r_G \geq \min_{x \perp \mathbf{1}, \|x\|_2=1} \frac{x^T Lx}{n}$$

Since L is symmetric, it has real eigenvalues $\mu_1, \dots, \mu_n$ from smallest to largest (our ordering was reverse of this for eigenvalues and singular values so far), and orthogonal eigenvectors $z_1, \dots, z_n$. It is also easily seen that $\mu_1 = 0$ and $z_1 = \mathbf{1}$.

We next resolve x along the eigenvectors z_i , i.e. let c_i be such that $x = \sum_i c_i z_i$. Then $Lx = \sum_i c_i Lz_i = \sum_i c_i \mu_i z_i$. Noting that $z_i \cdot z_j = 0$ for $i \neq j$:

$$x^T Lx = \left(\sum_i c_i z_i \right) \cdot \left(\sum_i c_i \mu_i z_i \right) = \sum_i c_i^2 \mu_i$$

Since we want to minimize $x^T Lx$, it would be desirable to make c_i large for smaller indices, since μ_i is non-increasing. Note now that $x \perp \mathbf{1} = z_1$. Thus $c_1 = 0$. Thus to minimize we should set $c_2 = 1$ whereupon $x^T Lx = \mu_2 x^T x$. Thus we have shown

$$r_G \geq \frac{\mu_2}{n}$$

3.2 Algorithm

Does the previous formulation help in designing an algorithm?

It is instructive to consider a similar but far simpler problem. Suppose we want to find an integer x such that $x^2 + 3.1x$ is minimized. We would naturally do this by relaxing the condition that x be real, and taking the derivative, assert that $2x + 3.1 = 0$. This would yield $x = -1.55$. We would then venture that a *nearby* integer value, -2 or -1 would have to be the answer. We would probably just check both and pick the smaller. Note however that the integer minimum -2.2 in this case, is not the same as the real minimum, -2.4025.

We can try something similar for our problem as well. We compute $x = z_2$, the second smallest eigenvector of the Laplacian matrix L . Then we somehow determine an S from x . Remember that we started off by assuming x to be a bit-vector, which we then shifted by an amount say α so that $x \cdot \mathbf{1} = 0$. In the new x vector there were only 2 values: α , wherever there was a 0 before, and $1 + \alpha$, wherever there was a 1 before. If z_2 has exactly this form, we can right away decide which vertex to put in S .

But in general z_2 will not have this form. So what do we do? Perhaps those vertices i whose coordinate $z_2(i)$ is large should be put in S . But how do we decide what “large” means? The simple answer is we try out all possibilities! After all, there are only n possible thresholds $z_2(j)$ for the different values of j . So we define $S_j = \{i | z_2(i) > z_2(j)\}$. We calculate $r(S_j, V - S_j)$ and pick the minimum.

So the next question is: how much worse than μ_2/n can the cut ratio of the cut selected above be?

3.3 Cheeger’s Theorem

Theorem 1 (Cheeger) *Let L be the Laplacian matrix of a graph G , and $\mu_1, \dots, \mu_n$ its eigenvalues ordered from the smallest to the largest. Then*

$$\frac{r(G)^2 n}{8d_{\max}} \leq \frac{\mu_2}{n} \leq r(G)$$

where $d_{\max}$ is the maximum degree. Further there is a simple algorithm to find a cut matching the first inequality, i.e. S such that

$$r(S, V - S) \leq \frac{\sqrt{8d_{\max}\mu_2}}{n}$$

The second inequality we have proved. The first inequality is substantially harder, and we will not prove. But we note that the algorithm referred to in the theorem is indeed the algorithm we discussed, and it may not find a cut of ratio μ_2/n , but will certainly find a cut of ratio at most $\frac{\sqrt{8d_{max}\mu_2}}{n}$. Note that the cut found need not be the best, but the best cut ratio must be somewhere between $\frac{\mu_2}{n}$ and $\frac{\sqrt{8d_{max}\mu_2}}{n}$.

Sometimes the bounds are stated in terms of sparsity:

$$\frac{\text{sp}(G)^2}{2d_{max}} \leq \mu_2 \leq 2\text{sp}(G)$$

3.4 Relationship to incidence matrix embedding

For k -regular graphs, $L = D - A = kI - A$. Thus L has the same eigenvectors as A . However, the order is reversed. The eigenvector with the i th smallest eigenvalue in L has the i th largest eigenvalue in A . Note further that $BB^T = A + D = kI + A$ also has same eigenvectors as A , and hence L . Thus the left singular vector u_i of B of singular value σ_i is the same as the eigenvector z_i of L of eigenvalue μ_i .

So the algorithm above was effectively: order vertices according to their coordinate in $u_2 = z_2$, and construct set S_j by picking those vertices i whose coordinate $u_2(i) = z_2(i)$ is greater than $u_2(j)$. Of all sets S_j return the one which has the least cut ratio.

If we believe that the point cloud obtained from the vertices has its “length” along direction v_2 , then it would seem natural to believe that the “cross section” perpendicular to v_2 is the “narrowest”. So if we want to cut very few edges, we should perhaps be partitioning the graph by slicing perpendicular to v_2 . In other words we project each vertex onto the direction v_2 , and then take a set of contiguous vertices in that order. But note that the projection of the vertex i is simply the i th component of Bv_2 . But $Bv_2 = \sigma_2 u_2$. Since we do not care about the scale, it is enough to consider the entries in u_2 as giving the projections of the vertices. So to get a “narrow” cross section, we should take all vertices having their u_2 value above some threshold. This is exactly what the algorithm described above does!

3.5 General graphs

The algorithm actually also works for general graphs; however, there is no clear correspondence with the edge incidence embedding. This is because for general graphs the vectors u_2 and z_2 will be different.

However, for general graphs, using the normalized incidence matrix does give an algorithm that finds a cut of the so-called minimum conductance (ratio of number of edges cut to the minimum of the sums of the degrees on either side of the cut). The second singular vector of the normalized incidence matrix is in fact the same as the second eigenvector of the so called normalized Laplacian that is encountered. So even in this algorithm can be viewed as slicing the point cloud perpendicular to the second singular vector.

But the details get very complicated[1].

4 Eigenvalues of Expander graphs

A (non-bipartite) (α, β, n, d) expander is a graph d regular $G = (V, E)$ with $|V| = n$ in which any subset S of V has at least $\beta|S|$ neighbours outside of S . We can use randomized constructions to construct such expanders.

Given such an expander, we note that by Cheeger's theorem, $\mu_2 \leq r(G)n$. If $\Gamma(S)$ denotes the neighbourhood of S , then we have $|\Gamma(S)| \geq |E(S, V - S)|/d$. Thus $\frac{|\Gamma(S)|}{|S|d} \geq \frac{r(S, V-S)|V-S|}{d} \geq r(G)\frac{(1-\alpha)n}{d} \geq \frac{\mu_2(1-\alpha)}{d}$. Thus, if $\frac{\mu_2(1-\alpha)}{d} \geq \beta$, we can rest assured that G is indeed an (α, β, n, d) expander.

Something similar works also for the bipartite case.

5 Eigenvalues of the hypercube

It is quite easy to find the eigenvalues of the hypercube by exploiting its recursive structure. First consider the adjacency matrix.

By simple algebra, it can easily be seen that the $-1, 1$ are the eigenvalues of Q_2 . Now observe that if A_i is the adjacency matrix of Q_i , then we have:

$$A_{i+1} = \begin{pmatrix} A_i & I \\ I & A_i \end{pmatrix}$$

From this it is easily verified that if λ is any eigenvalue of A_i and x the corresponding eigenvector, then $\lambda \pm 1$ are eigenvalues of A_{i+1} , and $\begin{pmatrix} x \\ \pm x \end{pmatrix}$

are the corresponding eigenvectors.

6 A note on eigenvalues and eigenvectors

The following facts are important for finding eigenvalues and eigenvectors and are easily verified.

1. If λ, x are respectively an eigenvalue and eigenvector of a matrix M , then $k\lambda, x$ are of kM , for any real k .
2. If λ, x are respectively an eigenvalue and eigenvector of a matrix M , then $k\lambda + k, x$ are of $kI + M$, for any real k .
3. If λ, x are respectively an eigenvalue and eigenvector of a matrix M , then $\lambda, Q^{-1}x$ are of $Q^{-1}MQ$, for any matrix Q .

These should enable you to relate the eigenvalues of walk matrices and adjacency matrices and Laplacians, especially when the graphs are regular. And also to singular values as appropriate.

Exercises

1. From first principles (i.e. by using elementary calculus) find the singular value decompositions for the matrices

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

You may check correctness by using a program such as scilab.

2. Find the largest singular value and the range of singular values for the edge incidence matrix of K_4 , the complete graph on 4 vertices.
3. Show that for a regular graph, the first singular vector passes through the center of mass of the vertices.
4. Draw a picture of the edge incidence embedding of all connected graphs having 3 edges.

5. Suppose you have a solid body made up of unit masses (discretized) at points $p_1, \dots, p_n$. Suppose the center of mass of the body is at the origin. The first principal axis of rotation of the solid body is that line through the origin which minimizes $\sum_j r_j^2$, where r_j is the perpendicular distance to the line from p_j . Show that the first principal axis of rotation can be found using singular value decomposition.
6. Show that all eigenvalues of any Laplacian matrix L are non-negative, and observe that $\mathbf{1}$ is an eigenvector of eigenvalue 0.
7. What is $\text{sp}(G)$ where G is (a) Q_n , (b) $K_{3n/4}$ connected by a single edge to $K_{n/4}$, (c) $P_{\sqrt{n}} \square P_{\sqrt{n}}$.
8. Use scilab to draw Q_3 using the second and third singular value. Also try other graphs.
9. Use scilab to find a cut of small ratio of different graphs using the second smallest eigenvector of the Laplacian, say $P_n \square P_n$.
10. Use scilab to find μ_2 for a random d -regular graph for various values of d . Comment on how much expansion you get.

References

- [1] F. Chung. *Spectral Graph Theory*. Americal Mathematical Society, 1997.
- [2] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.
- [3] D. Spielman. Graphs and networks: Random walks and spectral graph drawing, September 2007. www.cs.yale.edu/homes/spielman/462/lect4-07.pdf.