EE 263 Homework 7

Luis A. Perez

Properties of symmetric matrices

Solution:

(a) For the proof, note the following:

$$\begin{aligned} P &\geq 0 && \text{(Given)} \\ \implies P - 0 &\geq 0 && \text{(Zero matrix does not affect inequality)} \\ \implies P - (Q - Q) &\geq 0 && \text{(Rewriting 0 matrix as } Q - Q\text{)} \\ \implies (P + Q) - Q &\geq 0 && \text{(Matrix addition is associative)} \\ \implies P + Q &\geq Q && \text{(By definition)} \end{aligned}$$

(b) For the proof, note the following:

$$\begin{aligned} P &\geq Q && \text{(Given)} \\ \implies P - Q &\geq 0 && \text{(By definition)} \\ \implies -(P - Q) &\leq 0 \\ & && \text{(Multipling by } -1 \text{ makes the matrix } P - Q \text{ negative semidefinite)} \\ \implies -Q + P &\leq 0 && \text{(Distributing scalar multiplication)} \\ \implies -P &\leq -Q && \text{(By definition)} \end{aligned}$$

(c) The proof is not as straight forward. However, note that $P > 0$ means that all eigenvalues of P exists and are positive ($\lambda_1, \cdots, \lambda_n > 0$). The corresponding eigenvalues of P^{-1} ane $\frac{1}{\lambda_i}$, and since $\lambda_i > 0$ for all i, we have that $\frac{1}{\lambda_i} > 0$. As such, $P^{-1} > 0$.

(d) We first prove the given hint. We wish to show that $Z \geq I \implies Z^{-1} \leq I$. To begin, $Z \geq I$ means that the eignevalues of $Z - I$ are all non-negative. Letting λ_i be an eigenvalue of Z, note that we have:

$$(Z - I)v_i = Zv_i - v_i = \lambda_i v_i - v_i = (\lambda_i - 1)v_i$$

So the eigenvalues of $Z - I$ are all of the form $\lambda_i - 1$. Given that $Z - I \geq 0$, we know that $\lambda_i - 1 \geq 0 \implies \lambda_i \geq 1$.

Note that Z^{-1} is defined an exists, since $\lambda_i \geq 1 > 0$. In fact, it has eigenvalues $\frac{1}{\lambda_i}$. By a similar argument as the one made above, we must have that the eigenvalues of $Z^{-1} - I$ are all $\frac{1}{\lambda_i} - 1$. Since $\lambda_i \geq 1$, we must have that $\frac{1}{\lambda_i} \leq 1$, which implies $\frac{1}{\lambda_i} - 1 \leq 0$, which by definition means that $Z^{-1} - I \leq 0$ (or $Z^{-1} \leq I$).

Now that we've proven the hint, we focus on proving the remaining statement. First, note that $Q > 0$ implies that $Q^{-\frac{1}{2}}$ exists. Furthermore, we have:

$$\begin{aligned} Q^{-\frac{1}{2}} &= (U\Lambda U^T)^{-\frac{1}{2}} && (Q \text{ is a symetric matrix}) \\ &= U\Lambda^{-\frac{1}{2}}U^T && (\text{Properties of diagonilization}) \\ &= (U^T\lambda^{-\frac{T}{2}}U)^T && (\text{Transposing twice}) \\ &= Q^{\frac{T}{2}} \end{aligned}$$

As such, we have that $Q^{-\frac{1}{2}} = Q^{-\frac{T}{2}}$. As such, by congruence (since Q is just a rotation), we have the followig:

$$\begin{aligned} P &\geq Q && (\text{Given}) \\ \implies P - Q &\geq 0 && (\text{By definition}) \\ \implies Q^{-\frac{T}{2}}(P-Q)Q^{-\frac{1}{2}} &\geq 0 && (\text{By congruence}) \\ \implies Q^{-\frac{T}{2}}PQ^{-\frac{1}{2}} - Q^{-\frac{T}{2}}QQ^{-\frac{1}{2}} &\geq 0 && (\text{Distributive properties}) \\ \implies Q^{-\frac{T}{2}}PQ^{-\frac{1}{2}} - Q^{-\frac{1}{2}}QQ^{-\frac{1}{2}} && (Q^{-\frac{T}{2}} = Q^{-\frac{1}{2}}) \\ \implies Q^{-\frac{T}{2}}PQ^{-\frac{1}{2}} - I &\geq 0 \\ \implies (Q^{-\frac{T}{2}}PQ^{-\frac{1}{2}} - I) &\leq 0 && (\text{By hint proven previously}) \\ \implies Q^{\frac{1}{2}}P^{-1}Q^{\frac{T}{2}} - I &\leq 0 \\ \implies Q^{-\frac{1}{2}}(Q^{\frac{1}{2}}P^{-1}Q^{\frac{T}{2}} - I)Q^{-\frac{T}{2}} &\leq 0 && (\text{Congruence}) \\ \implies P^{-1} - Q^{\frac{1}{2}}Q^{-\frac{T}{2}} &\leq 0 && (\text{Distributive property}) \\ \implies P^{-1} - Q^{-1} &\leq 0 && (Q^T = Q) \\ \implies P^{-1} &\leq Q^{-1} && (\text{By definition}) \end{aligned}$$

(e) This statement is false. For the simplest counter-example, take a scalar 1×1 matrix $P = \begin{bmatrix} -1 \end{bmatrix}$ and $Q = \begin{bmatrix} -2 \end{bmatrix}$. Then note that $P \geq Q$, but $P^2 \leq Q^2$.

Drawing a representation of a graph

Solution:

(a) Let us begin by rephrasing the problem a bit first. Specifically, we'll beging by focusing on our objective. Note that we can write the objective function in quadratic form (we assume that each edge is only counted once). We have:

$$\begin{aligned} J &= \sum_{(i,j)\in E} (x_i - x_j)^2 + (y_i - y_j)^2 \\ &= \sum_{(i,j)\in E} x_i^2 + \sum_{(i,j)\in E} x_j^2 - 2\sum_{(i,j)} x_i x_j + \sum_{(i,j)\in E} y_i^2 + \sum_{(i,j)\in E} y_j^2 - 2\sum_{(i,j)} y_i y_j \end{aligned}$$

The above is the result of just expanding out. However, note that the term $2\sum_{(i,j)\in E} x_i x_j$ can be written simply as $x^T A x$ where A is the adjacency matrix. This follows from the definition of quadratic form, noting that the factor of 2 is taken care of by the fact that A is symetric. As such, we have:

$$J = \sum_{(i,j)\in E} x_i^2 + \sum_{(i,j)\in E} x_j^2 + \sum_{(i,j)\in E} y_i^2 + \sum_{(i,j)\in E} y_j^2 - x^T A x - y^T A y$$

Now, focusing on the first two terms above, we note that they too can be written in quadratic form. We're basically summing x_i^2 for $i = 1, \cdots, n$, where x_i appears a total of $\deg(i)$ times (the degree of node i). As such, let us define the matrix B as a diagonal matrix as follows:

$$B = \begin{bmatrix} \deg(1) & 0 & 0 & \cdots & 0 \\ 0 & \deg(2) & 0 & \cdots & 0 \\ 0 & 0 & \deg(3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \deg(n) \end{bmatrix} \in \mathbb{R}^{n\times n}$$

With the above, we can easily rewrite our objective as:

$$\begin{aligned} J &= x^T B x - x^T A x + y^T B y - y^T A + y \\ &= x^T (B + A) x^T + y^T (B + A) y \\ &= \mathbf{Tr}\left(\begin{bmatrix} x^T \\ y^T \end{bmatrix} (B - A) \begin{bmatrix} x & y \end{bmatrix} \right) \\ &= \mathbf{Tr}\left(\begin{bmatrix} x & y \end{bmatrix}^T (B - A) \begin{bmatrix} x & y \end{bmatrix} \right) \\ &= \mathbf{Tr}(X^T (B - A) X) \end{aligned}$$

Similarly, we can express our centering and spreading constraings simply as:

$$\begin{aligned} X^TX &= \begin{bmatrix} x^T \\ y^T \end{bmatrix} \begin{bmatrix} x & y \end{bmatrix} \\ &= \begin{bmatrix} x^Tx & x^Ty \\ y^Tx & y^Ty \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} && \text{(The given constraints)} \\ &= I_2 \end{aligned}$$

Putting everything together, the problem becomes:

$$\begin{aligned} \text{minimize} \quad & \mathbf{Tr}(X^T(B-A)X) \\ \text{subject to} \quad & X^TX = I_2 \end{aligned}$$

where we know that $B + A$ is symetric, and X is as defined above.

We know from lecture that the solution to this problem is given simply by:

$$X = \begin{bmatrix} q_n & q_{n-1} \end{bmatrix}$$

achieving optimal value $\lambda_n + \lambda_{n-1}$ where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of $B - A$ with corresponding orthonormal $q_1, \cdots, q_n$.

Finally, we know that shifting all the coordinates by a fixed amount won't affect our value J, so our final solution (so that we satisfy all constraints) will be given by:

$$\begin{aligned} x &= q_n - \frac{1}{n}\sum_{i=1}^{n} q_{n,i} \\ y &= q_{n-1} - \frac{1}{n}\sum_{i=1}^{n} q_{n-1,i} \end{aligned}$$

We note that, as explained in the problem, these values of x and y are not unique (any rotation or reflection of the points will be equally valid).

(b) We apply our method to the graph described by A. Our optimal value of J is:

$$J_{opt} = 0.1072932869526993$$

We can see this connectivity structure in Figure 1.

For comparison, the corresponding value of J associated with the circle is:

$$J_{circ} = 5.32711444$$

and we can see the resulting connectivity structure in Figure 2.

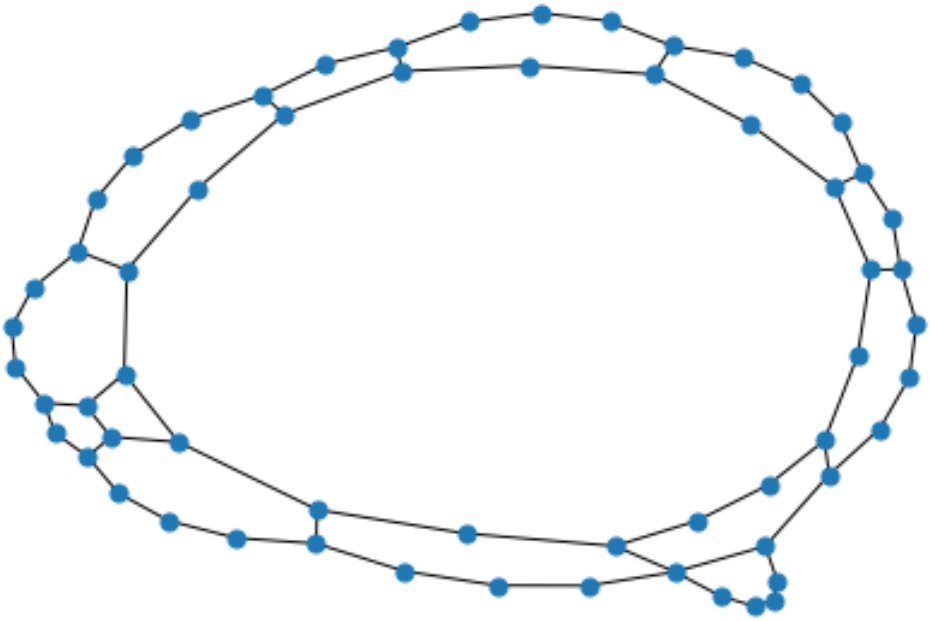


Figure 1: Optimal graph layout for the given connectivity structure

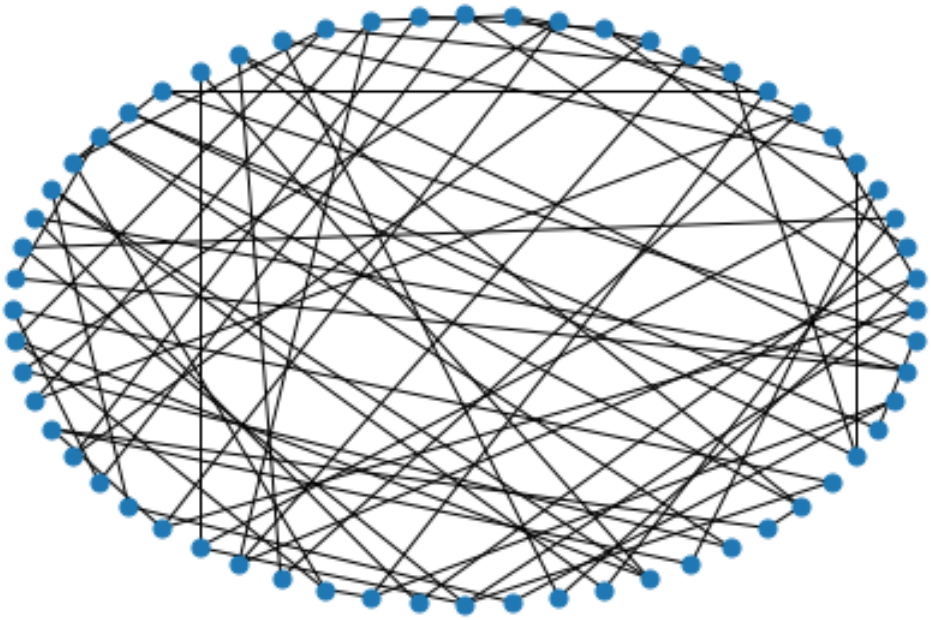


Figure 2: Graph layout with the given circle points

Blind signal detection

Solution:

(a) We restate the problem in a framework we're familiar with. In fact, our goal is to:

$$\begin{aligned} \text{minimize} \quad & ||w||^2 \\ \text{subject to} \quad & \frac{1}{T}||Y^T w||^2 = 1 \end{aligned}$$

Note that this is equivalent to:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||w||^2 \\ \text{subject to} \quad & ||Y^T w||^2 = T \end{aligned}$$

where:

$$Y = \begin{bmatrix} | & | & & | \\ y_1 & y_2 & \cdots & y_T \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times T}$$

The above inequality leads rise to the following Lagrangian:

$$L(w, \lambda) = \frac{1}{2}||w||^2 + \lambda(||Y^T w||^2 - T)$$

and taking derivates, we have the following optimality conditions:

$$\begin{aligned} \nabla L_w &= w + \lambda Y Y^T w = 0 \\ \nabla L_\lambda &= w^T Y Y^T w - T = 0 \end{aligned}$$

The first condition implies that our solution w must be an eigenvector of YY^T since we have:

$$YY^T w = -\frac{1}{\lambda} w$$

Let v_i be such an eigenvector, with eigenvalue λ_i. Then by the second condition, plugging $w = v_i$, we have:

$$v_i^T Y Y^T v_i - T = \lambda_i ||v_i||^2 - T \implies ||v_i||^2 = \frac{T}{\lambda_i}$$

Since our goal is to minimize $||v_i||^2$, and given T fixed, the second condition implies that we must choose $w = v$ as the eigenvector with the largest eigenvalue.

As such, let v_1 be the vector with eigenvalue $\lambda_1 = \max_i\{\lambda_i\}$ (this is unique if the eigenvalues are distinct, otherwise note). Then our w will be given by:

$$w = \frac{T}{\lambda_i}\frac{v_1}{||v_1||}$$

In the above, we simply make sure that the length of our chosen vector satisfies the last condition.

(b) We apply our method as described above. Our w is given by:

$$w = \begin{bmatrix} -0.00723673610484132 \\ -0.0215465730774432 \\ 0.00276054423894599 \\ 0.00445883002657630 \\ -0.0142168567061821 \\ 0.0148725035632431 \\ 0.0174363984377083 \\ -0.000841241581701471 \\ 0.00370664530172645 \\ 0.00282072925213337 \\ -0.00314005727052161 \\ 0.00835470364239948 \\ -0.00707868337950637 \\ 0.0281116810022352 \\ -0.00307470864581426 \\ 0.00234891789799632 \\ 0.0136042058350907 \\ 0.00226861097845394 \\ -0.00206656065858226 \\ -0.0106556910541729 \end{bmatrix}$$

The requested histogram can be seen in Figure 3. We can clearly see that we have two peaks. We compute the error rate of our model, which turns out to be 2.9%.

For the above, we used the code:

```
% Load the data
bs_det_data;

% Find the eigenvalues of Y*Y.T
[U,S,V] = svd(Y);

% Take the eigenvector with largest eigenvalue.
eigenValue = S(1,1)^2;
u = U(:,1);
```

```
10
11 % Normalize it to be the right size.
12 w = u / norm(u) * T / eigenValue;
13
14 % Verify w satisfies optimality conditions with some
       tolerance.
15 eps = 0.00001;
16 assert(all((w  - 1/eigenValue * Y * Y' * w) < eps));
17 assert(all((w' * Y * Y' * w - T) < eps));
18
19 % Plot histogram
20 figure
21 hist(w'*Y, 50);
22 title('Histogram of w^TY (estimated signal) values');
23 xlabel('Estimated signal value')
24 ylabel('Count');
25 saveas(gcf, 'blind_signal.jpg');
26
27 % Compute the estimated \hat{s} (or it's negation).
28 shat = sign(w'*Y);
29 errorRate = sum(shat ~= s) / T;
```

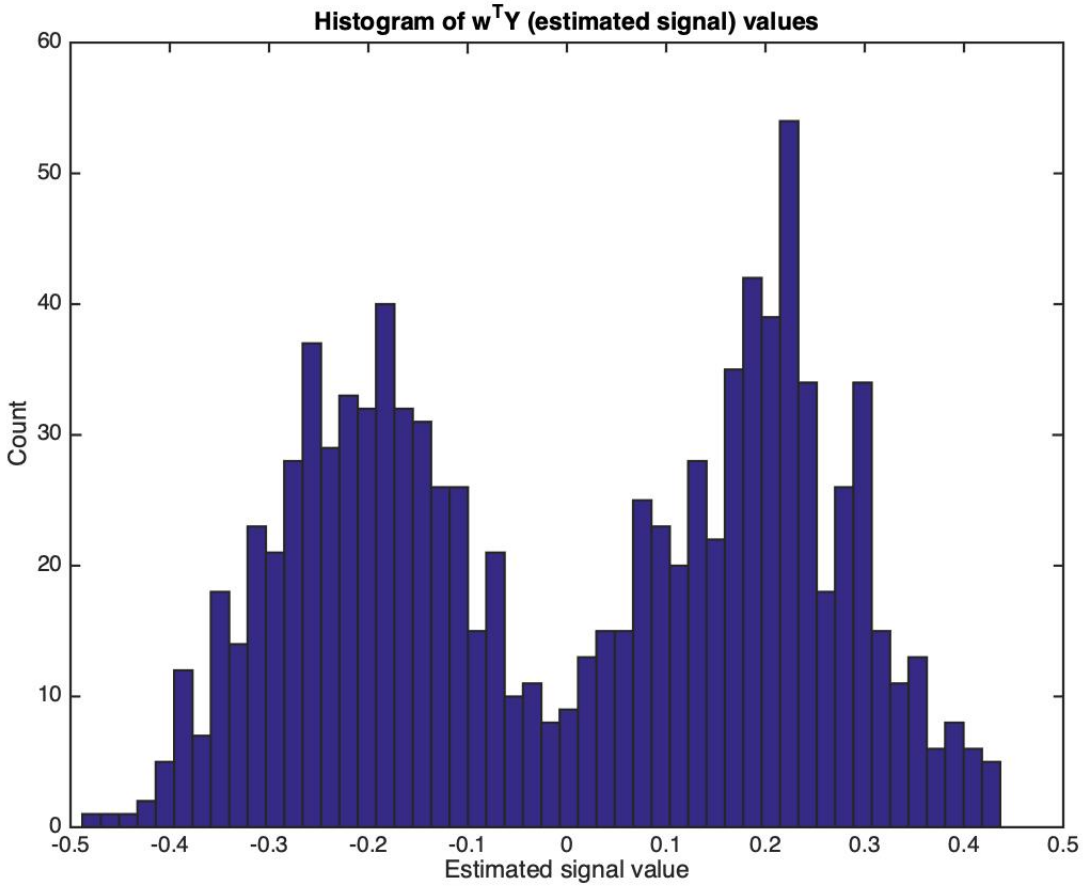


Figure 3: Histogram of estimated signal values using our weight vector.

Simultaneously estimating student ability and exercise difficulty

Solution: Our problem is to find $\hat{G}$ which approximates G. In fact, what we wish to do is find the matrix $\hat{G}$ of the form:

$$\hat{G} = \begin{bmatrix} \frac{1}{d_1} \\ \frac{1}{d_2} \\ \vdots \\ \frac{1}{d_m} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

which minimizes:

$$J = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (G_{ij} - \hat{G}_{ij})^2}$$

Note that since $\sqrt{\cdot}$ is monotonic and mn are constants, minimizing J is equivalent to minimizing:

$$\begin{aligned} \sqrt{mn}J &= \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (G_{ij} - \hat{G}_{ij})^2} \\ &= ||G - \hat{G}||_F \end{aligned}$$

As such, we can rephrase our problem as:

$$\begin{array}{rr} \text{minimize} & ||G - \hat{G}||_F \\ \text{subject to} & \mathbf{Rank}(\hat{G}) = 1 \end{array}$$

From lecture, we know that this problem has the solution of:

$$\hat{G} = \sigma_1 u_1 v_1^T$$

where the SVD decomposition of G is given by:

$$G = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > 0$. This approximation is unique of $\sigma_1 > \sigma_2$, otherwise multiple approximations achieve the same loss.

From the above model, we basically have what we want, expcet for the fact that we must

make sure that the exam question difficulties are normalized. We therefore have:

$$\begin{aligned} d_i &= \frac{1}{u_{1i}} \left(\frac{1}{m} \sum_{j=1}^{m} \frac{1}{u_{1j}} \right)^{-1} \\ &= \frac{m}{u_{1i} \sum_{j=1}^{m} \frac{1}{u_{ij}}} \\ \implies a_i &= \frac{m \sigma_1 v_{1i}}{\sum_{j=1}^{m} \frac{1}{u_{1j}}} \end{aligned}$$

Where the implication above comes from the fact that $\frac{a_i}{d_i} = \sigma_1 u_{1i} v_{1i}$. The above gives us our solution, ignoring the constraints that a_j be non-negative and d_i be positive.

Carrying out the proposed method above, we obtain the normalized difficulties as the vector:

$$d = \begin{bmatrix} 0.943 \\ 1.278 \\ 0.902 \\ 0.920 \\ 0.773 \\ 1.042 \\ 1.143 \end{bmatrix}$$

for which the optimal value achieved is $J_{\text{opt}} = 5.675923069899214$. The ratio of J_{opt} and the RMSE of G is 0.35742617468116206.

Square matrices and the SVD

Solution:

(a) False.

(b) False.

(c) False.

(d) True.

(e) False.

(f) True.

Principal-componenets analysis of decathlon data

Solution:

(a) A plot showing σ_j versus j is shown in Figure 4. A plot showing p_j versus j is shown in Figure 5. We also see that:

$$p_2 = 0.490183068840158$$

(b) The requested plot of the first two principal components can be seen in Figure 6. It's clear from the plot that some events with similarity appear to be close to each other. For example, shot put and discus are both throwing related events. Similarly, the pole value and javelin bouth require similar physical attributes.

(c) The plot generated by the given command can be seen in Figure 7. It's clear from the plot that v_1 most closely appears to represent r, since the intensitve varies as the points move along the axis corresponding to v_1.

(d) The requested plots are presented in Figure 8 and Figure 9.

From the plots, we can derive and intuitive interpretation for the first two left singular vectors of X. u_1 appears to correlate with the athletes total standardized scores, t. u_2 on the other hand appears to correlated most closely with δ, which is a measure of how much better an athlete is at running/leg events (100m, long jump, 100m hurdles) compared to throwing events (shot put, discus, javelin).

The code used for the above problems is now presented:

```
% load data
decathlon_pca_data

X = zscore(scores);
[U,S,V] = svd(X);
D = diag(S);

% part a
figure
plot(1:n, D);
title('\sigma_j vs j');
xlabel('j')
ylabel('\sigma_j');
saveas(gcf, 'sigma_vs_j.jpg');

p = cumsum(D.^2) / sum(D.^2);
```

```
figure
plot(1:n, p);
title('p_j vs j');
xlabel('j')
ylabel('p_j');
saveas(gcf, 'power_vs_j.jpg');

p2 = p(2);

% part b
v1 = V(:,1);
v2 = V(:,2);

figure
scatter(v1, v2);
text(v1, v2, events);
title('Top 2 components of event data')
xlabel('v_1')
ylabel('v_2')
saveas(gcf, 'principal_components.jpg')


% part c
t = sum(X,2);
r = t' * X;
spatial_plot(v1, v2, r, 3);
title('Spatial plot with r for intensity (color)')
xlabel('v_1')
ylabel('v_2')
saveas(gcf, 'intensity_plot.jpg')

% part d
u1 = U(:, 1);
u2 = U(:, 2);
delta = (X(:, 1) + X(:,5) + X(:,6)) - (X(:,3) + X(:,7) + X(:,9));

spatial_plot(u1 , u2 , t , length(names));
title('Spatial plot with t (athlete standardized scores) for
   intensity (color)')
xlabel('u_1')
ylabel('u_2')
saveas(gcf, 'spatial_t.jpg')

```

```
59 spatial_plot(u1 , u2 , delta , length(names));
60 title('Spatial plot with r for intensity (color)')
61 xlabel('u_1')
62 ylabel('u_2')
63 saveas(gcf, 'spatial_delta.jpg')
```

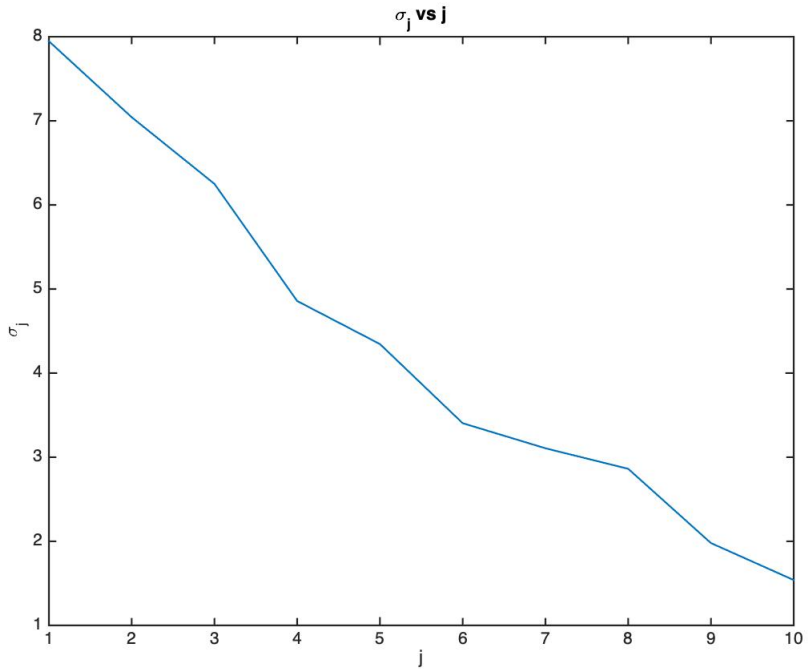


Figure 4: Plot of total power, σ_j, versus j

HW7

August 14, 2019

1 HW7

Author: Luis Perez
Last Updated: Aug. 10th, 2019

[37]:
```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

1.1 Problem 2: Drawing a representation of a graph

[10]:
```
def getProblem2Inputs():
    n = 60
    m = 73
    A = np.array([
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, ],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, ],
```

```
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 1, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, ],
```

```
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 1, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 1, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ],
   [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
```

```
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ],
   [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ],
   [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 1, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ],
   [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ],
```

```
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,␣
↪0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ],
   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
  ])
  x_circ = np.array([
       [0.1816],
       [0.1786],
       [0.1736],
       [0.1668],
       [0.1581],
       [0.1477],
       [0.1357],
       [0.1222],
       [0.1073],
       [0.0913],
       [0.0743],
       [0.0564],
       [0.0380],
       [0.0191],
       [0.0000],
       [-0.0191],
       [-0.0380],
       [-0.0564],
       [-0.0743],
       [-0.0913],
```

```
        [-0.1073],
        [-0.1222],
        [-0.1357],
        [-0.1477],
        [-0.1581],
        [-0.1668],
        [-0.1736],
        [-0.1786],
        [-0.1816],
        [-0.1826],
        [-0.1816],
        [-0.1786],
        [-0.1736],
        [-0.1668],
        [-0.1581],
        [-0.1477],
        [-0.1357],
        [-0.1222],
        [-0.1073],
        [-0.0913],
        [-0.0743],
        [-0.0564],
        [-0.0380],
        [-0.0191],
        [-0.0000],
        [0.0191],
        [0.0380],
        [0.0564],
        [0.0743],
        [0.0913],
        [0.1073],
        [0.1222],
        [0.1357],
        [0.1477],
        [0.1581],
        [0.1668],
        [0.1736],
        [0.1786],
        [0.1816],
        [0.182],
       ])

y_circ = np.array([
        [0.0191],
        [0.0380],
        [0.0564],
        [0.0743],
```

```
        [0.0913],
        [0.1073],
        [0.1222],
        [0.1357],
        [0.1477],
        [0.1581],
        [0.1668],
        [0.1736],
        [0.1786],
        [0.1816],
        [0.1826],
        [0.1816],
        [0.1786],
        [0.1736],
        [0.1668],
        [0.1581],
        [0.1477],
        [0.1357],
        [0.1222],
        [0.1073],
        [0.0913],
        [0.0743],
        [0.0564],
        [0.0380],
        [0.0191],
        [0.0000],
        [-0.0191],
        [-0.0380],
        [-0.0564],
        [-0.0743],
        [-0.0913],
        [-0.1073],
        [-0.1222],
        [-0.1357],
        [-0.1477],
        [-0.1581],
        [-0.1668],
        [-0.1736],
        [-0.1786],
        [-0.1816],
        [-0.1826],
        [-0.1816],
        [-0.1786],
        [-0.1736],
        [-0.1668],
        [-0.1581],
        [-0.1477],
```

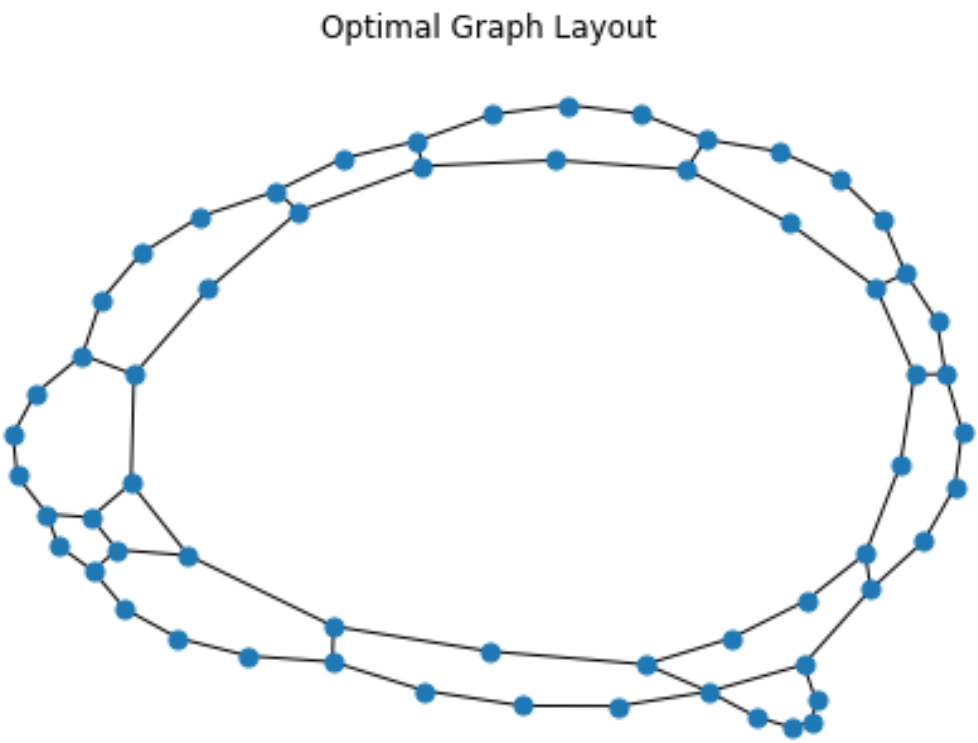
```
        [-0.1357],
        [-0.1222],
        [-0.1073],
        [-0.0913],
        [-0.0743],
        [-0.0564],
        [-0.0380],
        [-0.0191],
        [-0.000],
    ])
    return A, x_circ, y_circ, n, m
```

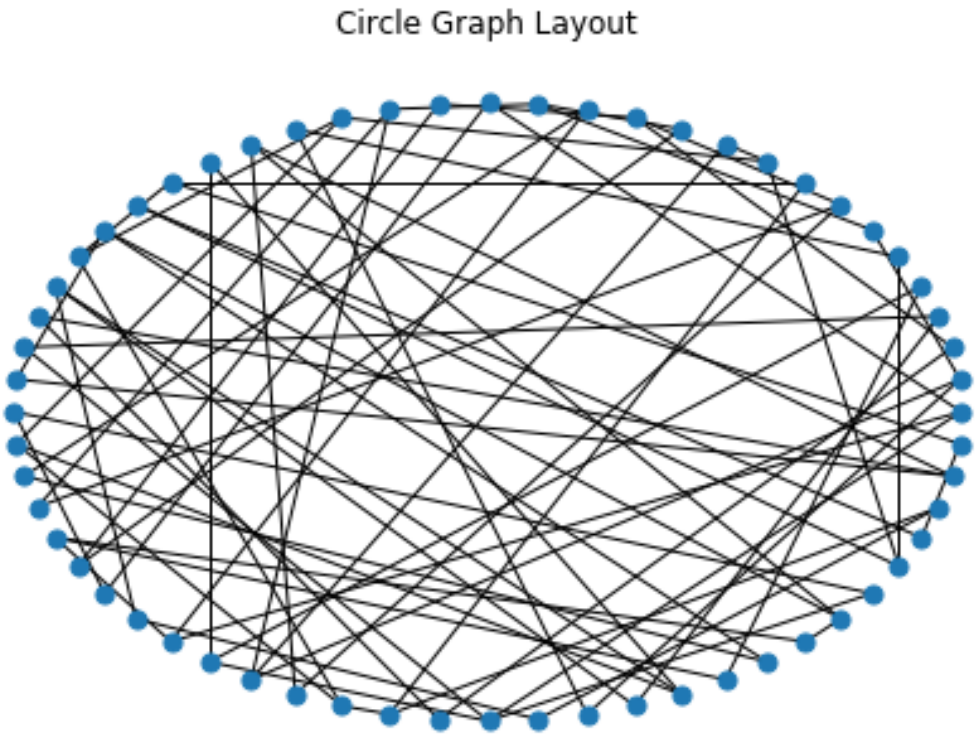
[54]:
```
def solveProblem2():
    A, x_circ, y_circ, n, m = getProblem2Inputs()
    def computeJ(x, y):
        ret = 0
        # Don't double count edges.
        for j in range(n):
            for i in range(j):
                if A[i,j] == 1:
                    ret += ((x[i] - x[j])**2 + (y[i] - y[j])**2)
        return ret
    # Find the degree of node i
    deg = np.sum(A, axis=0)
    B = np.diag(deg)

    # Our symmetric matrix.
    C = B - A

    # Need to find the smallest two eigvalues and vectors.
    # Values is in ascending order.
    values, vectors = np.linalg.eigh(C)

    # Optimal value is sum of smallest two.
    print("J optimal is %s." % (values[1] + values[2]))

    # Compute xopt and yopt
    v1 = vectors[:,1]
    v2 = vectors[:,2]
    optX = v1 - np.mean(v1)
    optY = v2 - np.mean(v2)

    # Verify constraints are met.
    assert np.allclose(np.sum(optX), 0)
    assert np.allclose(np.sum(optY), 0)
    assert np.allclose(np.sum(optX**2), 1)
    assert np.allclose(np.sum(optY**2), 1)
```

```
    assert np.allclose(np.sum(optX * optY), 0)
    print("J optimal is %s." % computeJ(optX, optY))
    print("J for circle is %s" % computeJ(x_circ.flatten(), y_circ.flatten()))

    # Plot the graph and whatnot.
    graph = nx.from_numpy_matrix(A)

    # Plot optimal.
    nx.draw(graph, list(zip(optX, optY)), node_size=50)
    plt.title('Optimal Graph Layout')
    plt.savefig('../hw7/data/optimal_graph')
    plt.show()

    nx.draw(graph, list(zip(x_circ.flatten(), y_circ.flatten())), node_size=50)
    plt.title('Circle Graph Layout')
    plt.savefig('../hw7/data/circle_graph')
```

[55]:
```
solveProblem2()
```

```
J optimal is 0.1072932869526993.
J optimal is 0.10729328695269982.
J for circle is 5.32711444
```





1.2 Problem 4: Simultaneously estimating student ability and exercise difficulty

[56]:
```
def getProblem4Inputs():
    m = 7;
    n = 78;
    G = np.array([
    [20, 20, 20, 12, 20, 15, 20],
    [20, 20, 0, 10, 16, 20, 17],
    [20, 20, 20, 10, 14, 18, 2],
    [20, 0, 0, 20, 20, 8, 2],
    [7, 20, 10, 10, 12, 0, 0],
    [6, 5, 20, 20, 20, 4, 1],
    [8, 20, 15, 10, 20, 7, 0],
    [16, 0, 20, 10, 20, 20, 16],
    [12, 5, 20, 20, 17, 19, 16],
    [17, 20, 20, 20, 20, 13, 20],
    [10, 18, 20, 10, 20, 20, 18],
    [19, 20, 10, 10, 20, 7, 17],
    [10, 0, 10, 10, 5, 13, 2],
    [20, 0, 20, 10, 20, 19, 4],
    [17, 20, 20, 10, 20, 15, 20],
```

```
    [18, 20, 10, 10, 20, 20, 18],
    [20, 20, 20, 15, 18, 0, 0],
    [20, 20, 20, 20, 15, 19, 20],
    [10, 0, 20, 10, 20, 0, 13],
    [0, 0, 20, 10, 20, 0, 1],
    [20, 0, 20, 20, 20, 5, 20],
    [19, 20, 20, 20, 20, 15, 8],
    [20, 20, 20, 20, 20, 4, 6],
    [20, 20, 20, 10, 19, 12, 20],
    [10, 20, 15, 20, 20, 20, 3],
    [17, 20, 0, 20, 18, 15, 10],
    [15, 0, 5, 8, 20, 6, 15],
    [7, 20, 20, 20, 19, 13, 10],
    [9, 0, 20, 20, 20, 9, 12],
    [18, 0, 0, 20, 20, 15, 18],
    [8, 20, 20, 20, 20, 15, 16],
    [16, 20, 20, 20, 20, 20, 15],
    [20, 20, 20, 20, 20, 16, 17],
    [18, 20, 20, 20, 20, 20, 15],
    [13, 0, 15, 20, 20, 20, 16],
    [20, 20, 20, 20, 20, 20, 0],
    [15, 10, 20, 20, 20, 20, 20],
    [6, 0, 20, 20, 20, 20, 5],
    [16, 15, 20, 20, 20, 20, 10],
    [20, 0, 20, 20, 20, 20, 18],
    [20, 20, 20, 10, 20, 20, 19],
    [10, 0, 5, 18, 12, 7, 5],
    [10, 0, 5, 20, 17, 15, 2],
    [16, 20, 0, 10, 17, 20, 17],
    [9, 0, 20, 10, 20, 20, 20],
    [20, 20, 20, 10, 20, 20, 20],
    [20, 20, 20, 10, 20, 7, 20],
    [17, 0, 20, 10, 20, 16, 20],
    [20, 20, 20, 10, 17, 20, 0],
    [10, 15, 20, 20, 18, 8, 0],
    [20, 20, 20, 19, 19, 13, 18],
    [18, 20, 20, 20, 20, 15, 20],
    [17, 0, 20, 20, 20, 20, 0],
    [8, 0, 10, 17, 14, 0, 0],
    [20, 0, 20, 20, 20, 12, 14],
    [15, 0, 5, 20, 14, 0, 0],
    [14, 0, 20, 20, 18, 15, 8],
    [10, 0, 5, 10, 20, 15, 5],
    [19, 20, 5, 20, 20, 15, 20],
    [8, 0, 20, 10, 18, 20, 10],
    [20, 0, 10, 10, 17, 10, 20],
    [12, 0, 10, 10, 20, 10, 6],
```

```
    [16, 20, 20, 20, 20, 20, 20],
    [20, 0, 20, 5, 16, 0, 4],
    [12, 0, 10, 10, 20, 5, 20],
    [20, 20, 20, 20, 20, 15, 19],
    [20, 20, 20, 18, 19, 12, 20],
    [11, 0, 0, 10, 20, 14, 16],
    [10, 0, 20, 18, 20, 15, 8],
    [11, 15, 15, 20, 20, 15, 20],
    [18, 0, 5, 18, 20, 0, 0],
    [12, 0, 20, 20, 20, 0, 0],
    [13, 0, 15, 20, 20, 20, 16],
    [9, 0, 10, 18, 15, 0, 5],
    [20, 20, 15, 20, 20, 20, 16],
    [20, 20, 20, 20, 19, 20, 20],
    [10, 0, 20, 10, 20, 20, 19],
    [15, 0, 10, 10, 18, 0, 5]
    ]).T
    return G, n, m
```

[114]:
```
def solveProblem4():
    G,n,m = getProblem4Inputs()
    # Skinny SVD.
    U, S, VT = np.linalg.svd(G, full_matrices=False)
    sigma = S[0]
    u, v = U[:, 0], VT[0,:]

    # Compute difficulties.
    d = m / (np.sum(1 / u) * u)
    with np.printoptions(formatter={'float': '{: 0.3f}'.format}):
        print("The difficulties are:")
        print(d)
    a = m * sigma / np.sum(1 / u) * v

    # Compute optimal value J
    d.shape = (d.shape[0], 1)
    a.shape = (a.shape[0], 1)
    Jopt = 1 / np.sqrt(m*n) * np.linalg.norm(G - np.dot(1 / d, a.T))
    print("The optimal value achieved is $J_{\\text{opt}} = %s$." % (Jopt))

    rmse = 1 / np.sqrt(m*n) * np.linalg.norm(G)
    print("The ratio of $J_{\\text{opt}}$ and the RMSE of $G$ is %s." % (Jopt /␣
↪rmse))
```

[115]:
```
solveProblem4()
```

```
The difficulties are:
[ 0.943  1.278  0.902  0.920  0.773  1.042  1.143]
```

The optimal value achieved is $J_{\text{opt}} = 5.675923069899214$.
The ratio of J_{opt} and the RMSE of G is 0.35742617468116206.

1.3 Problem 8: Sensor selection and observer design

[116]:
```
A = np.array([
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 0, 0, 0]
])
C = np.array([
    [1, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 0, 1]
])
```

[116]: 3

[117]:
```
np.linalg.matrix_rank(A), np.linalg.matrix_rank(C)
```

[117]: (3, 3)

[120]:
```
# Since C by itself is only rank 3, we can't reconstruct.
O = np.vstack((C, np.dot(C,A)))
```

[122]:
```
# Using just the first derivative is enough. We now have rank 4.
np.linalg.matrix_rank(O)
```

[122]: 4

[125]:
```
# We need to find the left inverse of the matrix. The matrix is skinny and
# full rank, so we have.
F = np.dot(np.linalg.inv(np.dot(O.T, O)), O.T)
```

[128]:
```
# Verify left inverse.
assert np.allclose(np.dot(F, O), np.identity(4))
```

[130]:
```
# But not right inverse.
assert not np.allclose(np.dot(O, F), np.identity(6))
```

[137]:
```
# Split into Fk
F0 = F[:, :3]
F1 = F[:, 3:]
```

[138]:
```
F0
```

[138]:
```
array([[-0.25 ,  0.125,  0.   ],
       [ 0.75 , -0.375,  0.   ],
       [-1.   ,  1.   ,  0.   ],
       [ 0.   ,  0.   ,  1.   ]])
```

[139]:
```
F1
```

[139]:
```
array([[ 3.75000000e-01, -1.25000000e-01,  6.25000000e-01],
       [-1.25000000e-01,  3.75000000e-01, -8.75000000e-01],
       [-6.66133815e-16,  4.44089210e-16,  1.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00]])
```

[ ]:

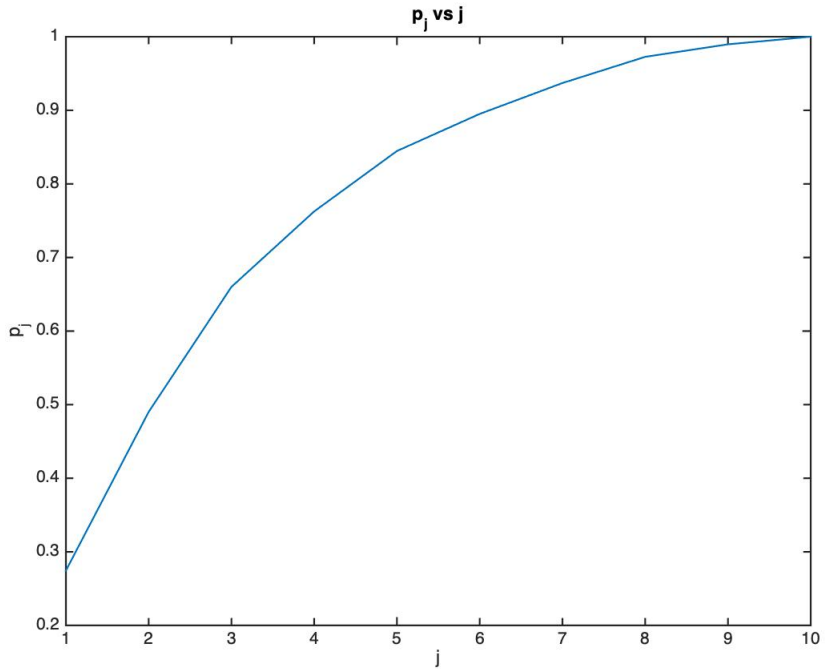


Figure 5: Plot of fraction of total power, p_j, versus j

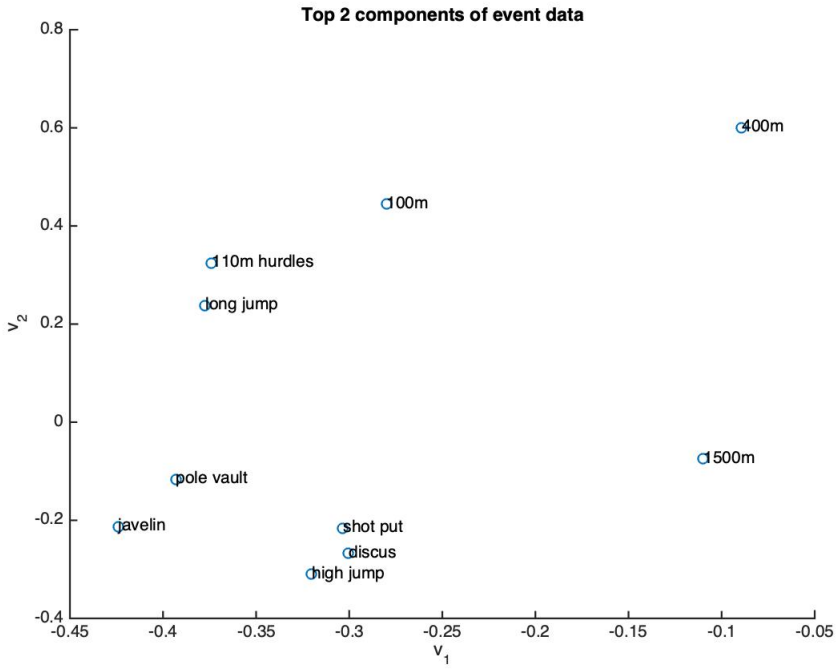


Figure 6: Plot of events by the first two principal components

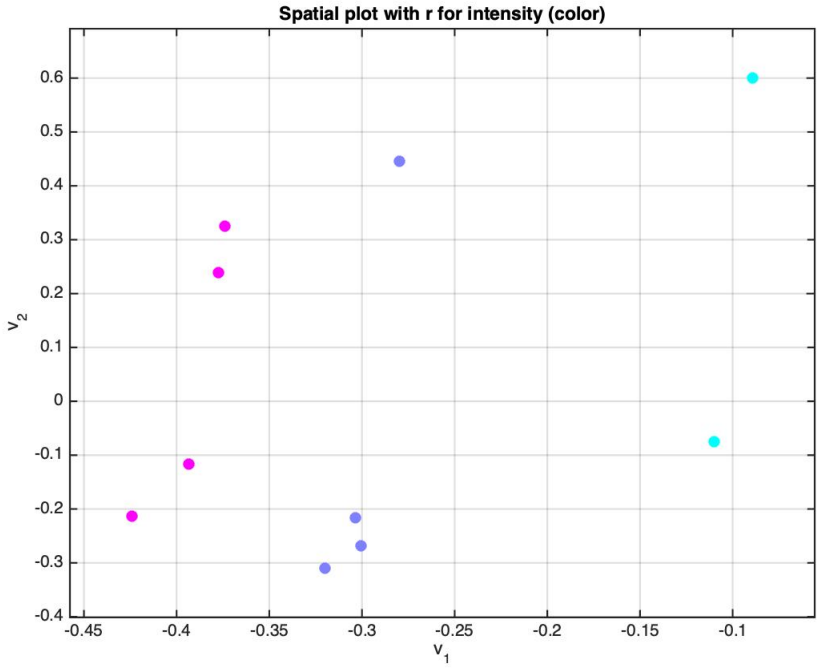


Figure 7: Plot of events colored by correlation between event and athlete's standardized test scores.

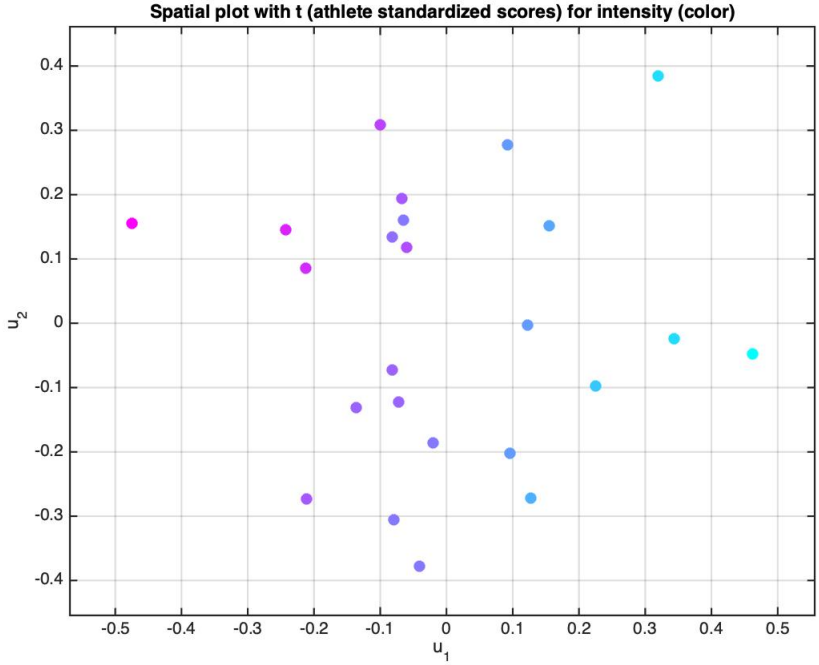


Figure 8: Plot of events colored by correlation athlete's standardized test scores.

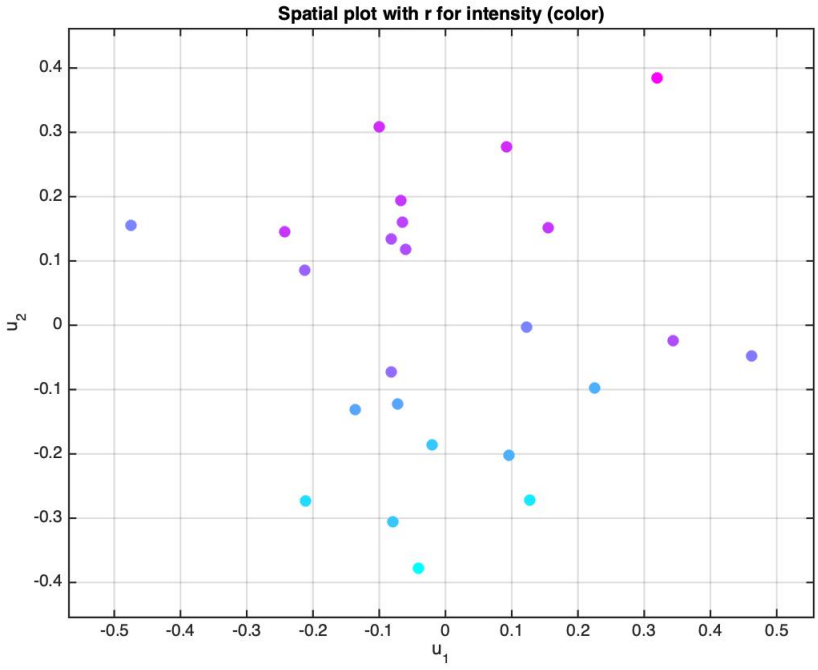


Figure 9: Plot of events colored by delta in scores between (100m, long jump, 100m hurdles) and (shot put, discus, javelin).