Robert Irwin

ECE 3522: Stochastics

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

In this assignment we will be implementing principal component analysis to determine which cities are the most closely related based off a wide variety of criteria. The criteria are measured on a large variety of scales, which is why principal component analysis must be implemented before distance measures can yield accurate results. Principal component analysis makes use of the following equation.

$$z=∆^{-\frac{1}{2}}ßx, Equation 1$$

where z is the transformed data, x is the original data, ∆ is the eigenvalue matrix of the covariance of x, and ß is the eigenvector matrix of the covariance of x.

# Approach and Results

The first step of the analysis was to bring in the data to be analyzed. Once this was done we computed the covariance matrix of the data. Then we performed eigenvector/value analysis to the covariance matrix. We then applied Equation 1 and begin calculating Euclidean distance between every city. Once this was complete, we searched for the minimum distance, took the average of the two cities ratings, and repeated the process to find the third closest city. The results are shown below.



*Figure 1: Eigenvalue Matrix*

Note that the eigenvalues are in ascending order. This tells us that the first column of the data would impact the rating the least, while the last column impacts the data the most if principal component analysis is not applied.



*Figure 2: Eigenvector Matrix*

The eigenvector matrix seems to be normalized so that every column is on the same scale. This makes sense as Matlab automatically normalizes every eigenvector to have a magnitude of 1. Also note that each eigenvector corresponds to the eigenvalue in the same column, so the eigenvalues and eigenvectors must stay ordered this way.



 *Figure 3: Covariance Matrix of Transformed Data*

As expected, the matrix is an identity. The small errors can be attributed to numerical limitations.



*Figure 4: Percent of Variance by each Eigenvalue*

We notice that the first two eigenvalues account for around 90% of the variance. The eigenvalues arranged in the proper order is shown below.



*Figure 5: Eigenvalues Sorted from Largest to Smallest*

From Figures 1 and 2 we can see that the eight and ninth eigenvector correspond to the two largest eigenvalues. In these eigenvectors, we see the seventh and the second row have the largest values. The seventh and second criteria in the ratings matrix are arts and housing respectively.

Using the original data, we get the following results.



*Figure 6: Results for Original Data*

If we print the values for each category, we see that the second and the seventh column of each city is very closely related. This confirms our assumption about which categories matter the most. Based off of te plot in Figure 4 however, we can see that the ninth eigenvector affects the distance measure much more than the eight eigenvector. Because the ninth eigenvector had its highest value at 2, we see that the second column of “compare” is very close, which tells us the second column of the ratings matrix completely dominates the distance measure. It is for this reason that these cities were considered to be the closest even though the other columns corresponding to these cities are fairly distant.



*Figure 7: Comparison of the Three Closest Cities*

**

*Figure 8: Results for Transformed Data*

**

*Figure 9: Comparison of Transformed Three Closest Cities*

This time we see that across all categories, the stats for each city are very closely related.

We now present the same data for the first three criteria only.



*Figure 10: Results for Three Criteria*

**

*Figure 11: Comparison of Three Closest.*

In the this figure the second column still dominates, but the other two columns play more of an even role in the distance measure. This is because we are ignoring the other dominant criteria.



*Figure 12: Transformed Data (Three Criteria)*

**

*Figure 13: Comparison of Three Closest*

Now that the data has been decorrelated, we see that we get different results, and all three criteria are closely related.

# MATLAB Code



*Figure 14: Bringing in Data, Beginning PCA*

In the code above we bring in the data and begin the eigenvalue problem on the covariance matrix of the data. This is necessary in order to implement Equation 1. My\_covar() is a user defined function and will be explained below. It is also important to note that the matrix z in the code above is the transformed data.



*Figure 15: My\_covar()*

This function computes the covariance matrix of an input. We begin by fnding the mean of each row and column of the input. Then we apply the definition of covariance. This function was needed because Matlab’s function normalizes by N-1 instead of N.



*Figure 16: Ordering the Eigenvalues*

In this piece of code we are putting the eigenvalues of the matrix into a vector so that they can be ordered and summed. Once this is done we plot the current eigenvalue divided by the total sum of the eigenvalues, times 100.



*Figure 17: Finding the Closest Cities*

In this code we are simply calculating the distance between every possible combination of cities. Based on the code above, this matrix will be symmetric with 0’s on the main diagonal.



*Figure 18: Finding the Closest Cities (continued)*

Because we expect zeros on the main diagonal, we find the maximum of the matrix and add it to the main diagonal. Because all distance measures are positive, a 0 would always come up as the minimum, which is obviously inaccurate.



*Figure 18: Finding the Closest Cities (continued)*

We now find the minimum entry of the matrix. This will return two indices for minr, and minc because the matrix is symmetric.



*Figure 19: Finding the Third Closest City*

We average all the criteria for the two closest city, than store the distance measures from every other city to the average of the two closest. We check to ensure the index does not equal the index of either of the two closest cities. This is because they were already found to be the two closest, and we do not want them considered.



*Figure 20: Finding the Third Closest City (continued)*

Just as we did before, we search the matrix for the minimum entry.

The rest of the code is identical, so it will be presented and not explained. The only difference is the data set we are working with.

%repeat the process for the transformed data

%Find the two closest cities

z = z';

for i = 1:length(z(:,1))

 for j = 1:length(z(:,1))

 for k = 1:length(z(1,:))

 temp(k) = (z(i,k)-z(j,k))^2;

 end

 disz(i,j) = sqrt(sum(temp));

 end

end

% we expect zeros along the diagonal, if we search for a minimum, we will

% receive these values GET RID OF THEM

%find max value and add it to the diagonal

maxcolz = max(disz);

maxdisz = max(maxcolz);

[maxrz, maxcz] = find(disz==maxdisz);

for i = 1:length(disz(:,1))

 for j = 1:length(disz(1,:))

 if( i == j)

 disz(i,j) = disz(i,j)+maxdisz;

 end

 end

end

%now find the minimum entry

mincolz = min(disz);

mindisz = min(mincolz);

[minrz, mincz] = find(disz==mindisz);

disp(sprintf('The closest states for the transformed data are the %dth and the %dth'...

 ,minrz(1),mincz(1)));

%we now have the closest states. find the third closest

%begin by finding the average between the two closest

for i = 1:1:length(z(1,:))

 avg\_ratez(i) = (z(minrz(1),i)+z(mincz(1),i))/2;

end

for i = 1:length(z(:,1))

 for j = 1:length(z(1,:))

 temp(j) = (z(i,j)-avg\_ratez(j))^2;

 end

 dis2z(i) = sqrt(sum(temp));

 if( (i==mincz(1)) || (i==minrz(1)) )

 dis2z(i) = 1e6;

 end

end

%find the closest state

minc2z = find(dis2z==min(dis2z));

disp(sprintf('The %dth state is the closest to the %dth and %dth for the transformed data'...

 ,minc2z,minrz(1),mincz(1)));

comparez = [z(minc2,:);z(minr(1),:);z(minc(1),:)]

%now we repeat using only the first three criteria

ratings\_lim = [ratings(:,1) ratings(:,2) ratings(:,3)];

zlim = [z(:,1) z(:,2) z(:,3)];

%Find the two closest cities

for i = 1:length(ratings\_lim(:,1))

 for j = 1:length(ratings\_lim(:,1))

 for k = 1:length(ratings\_lim(1,:))

 temp(k) = (ratings\_lim(i,k)-ratings\_lim(j,k))^2;

 end

 dis\_lim(i,j) = sqrt(sum(temp));

 end

end

% we expect zeros along the diagonal, if we search for a minimum, we will

% receive these values GET RID OF THEM

%find max value and add it to the diagonal

maxcol\_lim = max(dis\_lim);

maxdis\_lim = max(maxcol\_lim);

[maxr\_lim, maxc\_lim] = find(dis\_lim==maxdis\_lim);

for i = 1:length(dis\_lim(:,1))

 for j = 1:length(dis\_lim(1,:))

 if( i == j)

 dis\_lim(i,j) = dis\_lim(i,j)+maxdis\_lim;

 end

 end

end

%now find the minimum entry

mincol\_lim = min(dis\_lim);

mindis\_lim = min(mincol\_lim);

[minr\_lim, minc\_lim] = find(dis\_lim==mindis\_lim);

disp(sprintf('The closest states for 3 criteria are the %dth and the %dth',minr\_lim(1),minc\_lim(1)));

%we now have the closest states. find the third closest

%begin by finding the average between the two closest

for i = 1:1:length(ratings\_lim(1,:))

 avg\_rate\_lim(i) = (ratings\_lim(minr\_lim(1),i)+ratings\_lim(minc\_lim(1),i))/2;

end

for i = 1:length(ratings\_lim(:,1))

 for j = 1:length(ratings\_lim(1,:))

 temp(j) = (ratings\_lim(i,j)-avg\_rate\_lim(j))^2;

 end

 dis2\_lim(i) = sqrt(sum(temp));

 if( (i==minc\_lim(1)) || (i==minr\_lim(1)) )

 dis2\_lim(i) = 1e6;

 end

end

%find the third closest state

minc2\_lim = find(dis2\_lim==min(dis2\_lim));

disp(sprintf('The %drd state is the closest to the %dth and %dth for 3 criteria',minc2\_lim,...

 minr\_lim(1),minc\_lim(1)));

comparelim = [ratings\_lim(minc2\_lim,:);ratings\_lim(minr\_lim(1),:);...

ratings\_lim(minc\_lim(1),:)]

%repeat the process for the transformed data

%Find the two closest cities

z\_lim = zlim;

for i = 1:length(z\_lim(:,1))

 for j = 1:length(z\_lim(:,1))

 for k = 1:length(z\_lim(1,:))

 temp(k) = (z\_lim(i,k)-z\_lim(j,k))^2;

 end

 disz\_lim(i,j) = sqrt(sum(temp));

 end

end

% we expect zeros along the diagonal, if we search for a minimum, we will

% receive these values GET RID OF THEM

%find max value and add it to the diagonal

maxcolz\_lim = max(disz\_lim);

maxdisz\_lim = max(maxcolz\_lim);

[maxrz\_lim, maxcz\_lim] = find(disz\_lim==maxdisz\_lim);

for i = 1:length(disz\_lim(:,1))

 for j = 1:length(disz\_lim(1,:))

 if( i == j)

 disz\_lim(i,j) = disz\_lim(i,j)+maxdisz\_lim;

 end

 end

end

%now find the minimum entry

mincolz\_lim = min(disz\_lim);

mindisz\_lim = min(mincolz\_lim);

[minrz\_lim, mincz\_lim] = find(disz\_lim==mindisz\_lim);

disp(sprintf('The closest states for the transformed data(3 criteria)'))

disp(sprintf('are the %dth and the %dth',minrz\_lim(1),mincz\_lim(1)));

%we now have the closest states. find the third closest

%begin by finding the average between the two closest

for i = 1:1:length(z\_lim(1,:))

 avg\_ratez\_lim(i) = (z\_lim(minrz\_lim(1),i)+z\_lim(mincz\_lim(1),i))/2;

end

for i = 1:length(z\_lim(:,1))

 for j = 1:length(z\_lim(1,:))

 temp(j) = (z\_lim(i,j)-avg\_ratez\_lim(j))^2;

 end

 dis2z\_lim(i) = sqrt(sum(temp));

 if( (i==mincz\_lim(1)) || (i==minrz\_lim(1)) )

 dis2z\_lim(i) = 1e6;

 end

end

%find the closest state

minc2z = find(dis2z\_lim==min(dis2z\_lim));

disp(sprintf('The %dth state is the closest to the %dth and %dth for the transformed data(3 criteria)'...

 ,minc2z,minrz\_lim(1),mincz\_lim(1)));

comparez\_lim = [z\_lim(minc2\_lim,:);z\_lim(minrz\_lim(1),:);z\_lim(mincz\_lim(1),:)]

# Conclusions

In this assignment we have learned that principal component analysis is useful when trying to make accurate comparisons to things on completely different scales. We saw how much difference the principal component analysis made in our results of the three closest cities. This makes sense because vefore principal component analysis was done on the data, the second criteria completely dominated. This means that the three closest cities would have very similar data for the second criteria, but be very different in all other categories.