ISYE6740-HW2

pkubsad

February 2021

1 Political blogs dataset [50 points.]

1. (10 points) Assume the number of clusters in the graph is k. Explain the meaning of k here intuitively.

   Answer: The value of k represents the number of clusters we want to divide the given data-set. In data set given in the questions, the hypothesis is that similar nodes connect with each other and form a community. If we draw the graph of all the nodes, 'k' represents the number of partitions we want the entire network to be partitioned into. When k=2, we would partition the graph into 2 networks. From the output of the code below, we can see that nodes 182 and 666 are clustered together as 1 cluster. All the remaining nodes are clustered together as 2nd cluster. If we see the edges txt file, we can see that nodes 182 and 666 are connected only to each other and form a good connected community.

2. (10 points) Use spectral clustering to find the $k = 2, 5, 10, 20$ clusters in the network of political blogs (each node is a blog, and their edges are defined in the file edges.txt). Then report the majority labels in each cluster, for different k values, respectively.

   Answer: I have used the spectral clustering method explained at the beginning of lecture 3 (not special spectral method, or the svd method). The approach is:

   (a) Remove disconnected nodes resulting in 1224 nodes.
   (b) Create a temporary list to maintain mapping between all the nodes in edges file to reduced 1224 nodes.
   (c) Create adjacency matrix based on nodes appearing in edges file.
   (d) Create a diagonal matrix based on degrees of each node.
   (e) Calculate graph laplacian L = D - A.
   (f) Calculate eigen values and eigen vector using L
   (g) Sort eigen values asecending, and pick the lowest k eigen vectors.
   (h) Run k-means algorithm using the eigen vectors shortlisted in the step above.

   Based on the output of the code at the time, here are the observations:

k	Labels	Counts	Cluster Mismatch	Overall Mismatch
2	cluster 0 ->1 cluster 1 ->0	cluster 0 ->1: 636 ; 0:586 cluster 1->0:2	cluster 0->0.478 cluster 1 ->0	0.478
5	cluster 0 ->1 cluster 1 ->0 cluster 2 ->0 cluster 3 ->0 cluster 4 ->1	cluster 0 ->1:4 cluster 1 ->0:2 cluster 2 ->0:2 ; 1:2 cluster 3 ->0:2 cluster 4 ->1: 630 ; 0:582	cluster 0 ->0 cluster 1 ->0 cluster 2 ->0.5 cluster 3 ->0 cluster 4 ->0.481	0.477
10	cluster 0 ->1 cluster 1 ->1 cluster 2 ->0 cluster 3 ->0 cluster 4 ->1 cluster 5 ->0 cluster 6 ->1 cluster 7 ->0 cluster 8 ->1 cluster 9 ->0	cluster 0 ->1:2 cluster 1 ->1:584 , 0: 556 cluster 2 ->0:2 cluster 3 ->0:2 cluster 4 ->1: 2 cluster 5 ->0:2 cluster 6 ->1:44 , 0:17 cluster 7 ->0:2 cluster 8 ->1:4 , 0:2 cluster 9 ->0:5	cluster 0 ->0 cluster 1 ->0.487 cluster 2 ->0 cluster 3 ->0 cluster 4 ->0 cluster 5 ->0 cluster 6 ->0.278 cluster 7 ->0 cluster 8 ->0.333 cluster 9 ->0	0.469
20	cluster 0 ->1 cluster 1 ->0 cluster 2 ->0 cluster 3 ->0 cluster 4 ->0 cluster 5 ->1 cluster 6 ->0 cluster 7 ->0 cluster 8 ->0 cluster 9 ->0 cluster 10 ->1 cluster 11 ->0 cluster 12 ->1 cluster 13 ->0 cluster 14 ->1 cluster 15 ->0 cluster 16 ->1 cluster 17 ->0 cluster 18 ->0 cluster 19 ->0	cluster 0 ->1:22, 0:7 cluster 1 ->0:5 cluster 2 ->1:15, 0:466 cluster 3 ->0:2 cluster 4 ->0:2 cluster 5 ->1:4, 0:2 cluster 6 ->0:1 cluster 7 ->0:2 cluster 8 ->0:2 cluster 9 ->1:3, 0:43 cluster 10 ->1:564,0:28 cluster 11 ->0:1 cluster 12 ->1:2 cluster 13 ->0:1 cluster 14 ->1:21,0:1 cluster 15 ->0:13 cluster 16 ->1:2 cluster 17 ->0:2 cluster 18 ->1:3, 0:8 cluster 9 ->0:2	cluster 0 ->0.241 cluster 1 ->0 cluster 2 ->0.031 cluster 3 ->0 cluster 4 ->0 cluster 5 ->0.333 cluster 6 ->0 cluster 7 ->0 cluster 8 ->0 cluster 9 ->0.065 cluster 10 ->0.047 cluster 11 ->0 cluster 12 ->0 cluster 13 ->0 cluster 14 ->0.045 cluster 15 ->0 cluster 16 ->0 cluster 17 ->0 cluster 18 ->0.272 cluster 19 ->0	0.0482

3. (10 points) Now compare the majority label with the individual labels in each cluster, and report the *mismatch rate* for each cluster, when $k = 2, 5, 10, 20$.

   Answer:

   The table above also has recordings of the observed mismatch rate for each cluster. The overall mismatch rate for k is calculated by $\frac{noofmismatchesinalltheclusters}{totalnoofnodes}$.

4. (10 points) Tune your k and find the number of clusters to achieve a reasonably small *mismatch rate*. Please explain how you tune k and what is the achieved mismatch rate.

   Answer: I followed brute force approach of iterating through Ks from 2 to 300. The gain in the mismatch rate is marginal once the value of k¿19. So, I reduced the runs for k=2 to 150 and tracked

the mismatch rate for each value of k. As per the run during this documentation, I see:

$$k = 99, mismatch - rate = 0.0359$$

The plot of mismatch rate vs k also looks like elbow method where we see that after k=19, there is not much improvement in the mismatch rate.

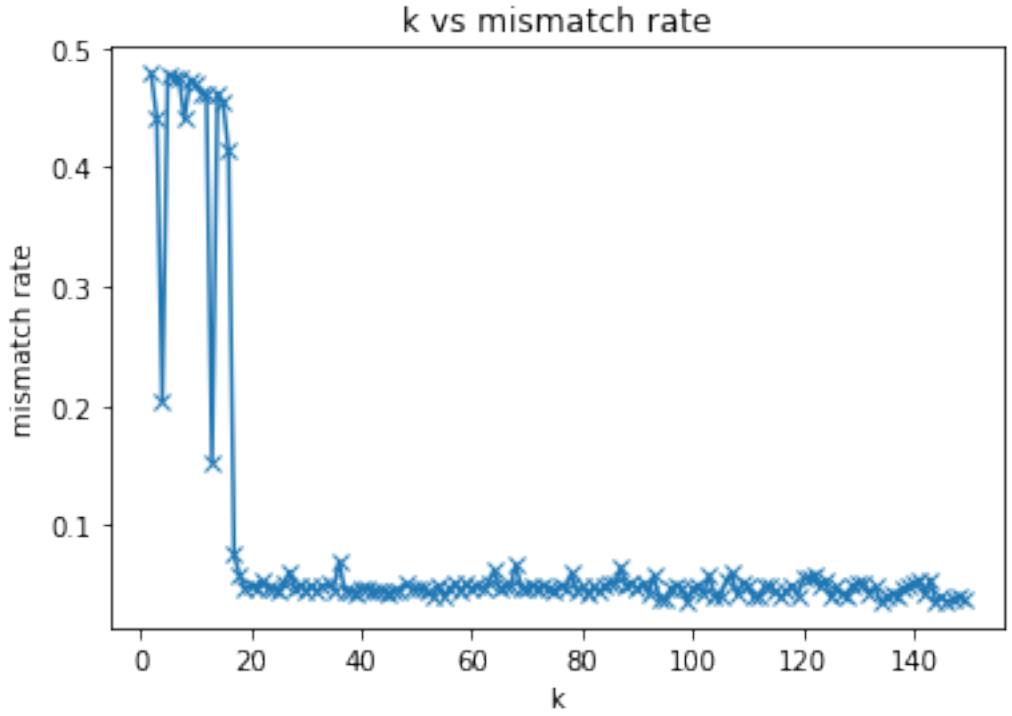

Figure 1: mismatch rate v/s k

5. (10 points) Please explain the finding and what can you learn from this data analysis.

Answer: The hypothesis of this experiment is that, similar nodes are connected to each other. These connected nodes create a community within a graph. Spectral Clustering on this graph data would divide the graph into 'k' sub networks as a cluster. Each cluster will mostly have similar nodes.
When k=2, we can see that in the 2nd cluster only 2 nodes were included. These 2 nodes, 182 and 666 are connected only to each other and form a good connected community. We can visualise by creating the graph of all the edges, we can see, nodes 636 and 586 are separated away from the overall network.

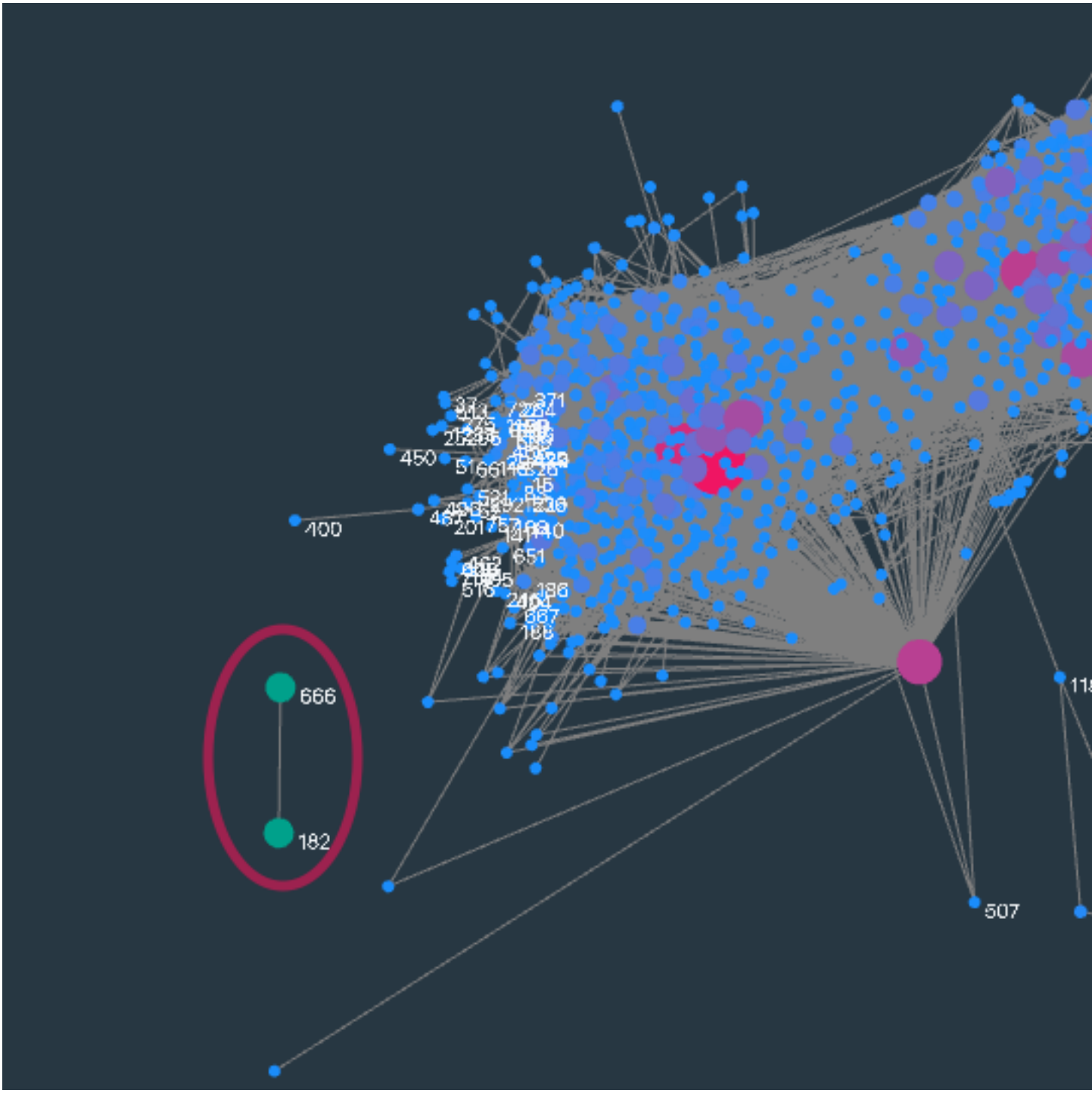

Figure 2: visualization of the edges.txt, credits: https://poloclub.github.io/argo-graph-lite/

After k value goes beyond 19, we dont see much improvement in the mismatch rate. But within the clusters formed, we see that the nodes have same political orientation are linked together. We can conclude the initial hypothesis mentioned in the homework is true.

If I run the same problem with special clustering method given in the demo code, I start seeing higher accuracy to earlier values of k. The difference between the special clustering approach v/s the normal approach comes down to the point of how python handles determination of eigen values. In my experiment and couple of posts on piazza, python performs better in determining higher eigen values compared to calculating lower eigen values. In special clustering method, we take top 'k' eigen values, it performs sligthly better than calculating lowest 'k' eigen values.

Since the approach uses kmeans at the end, the outcome is different every time we run the code. This can be attributed to the randomness of the initial centroids selected.As far as practical applications are concerned, I can easily see this approach/algorithm can be used for targeted marketing. Determine community of people within a network that share common interest and customize the marketing message to that community.

2.1 (25 points) Perform analysis on the Yale face dataset for Subject 1 and Subject 2, respectively, using all the images EXCEPT for the two pictures named subject01-test.gif and subject02-test.gif. **Plot the first 6 eigenfaces for each subject.** When visualizing, please reshape the eigenvectors into proper images. Please explain can you see any patterns in the top 6 eigenfaces?

Answer: Each eigen face that is generated using the eigen vectors from PCA output, captures some variability from the original picture, like shadow. In higher valued eigen faces these features are distinctly seen, like shadow on left side, and with eigenfaces of lower values, these features start diminishing. Each eigen face seems like approximation of the original image.

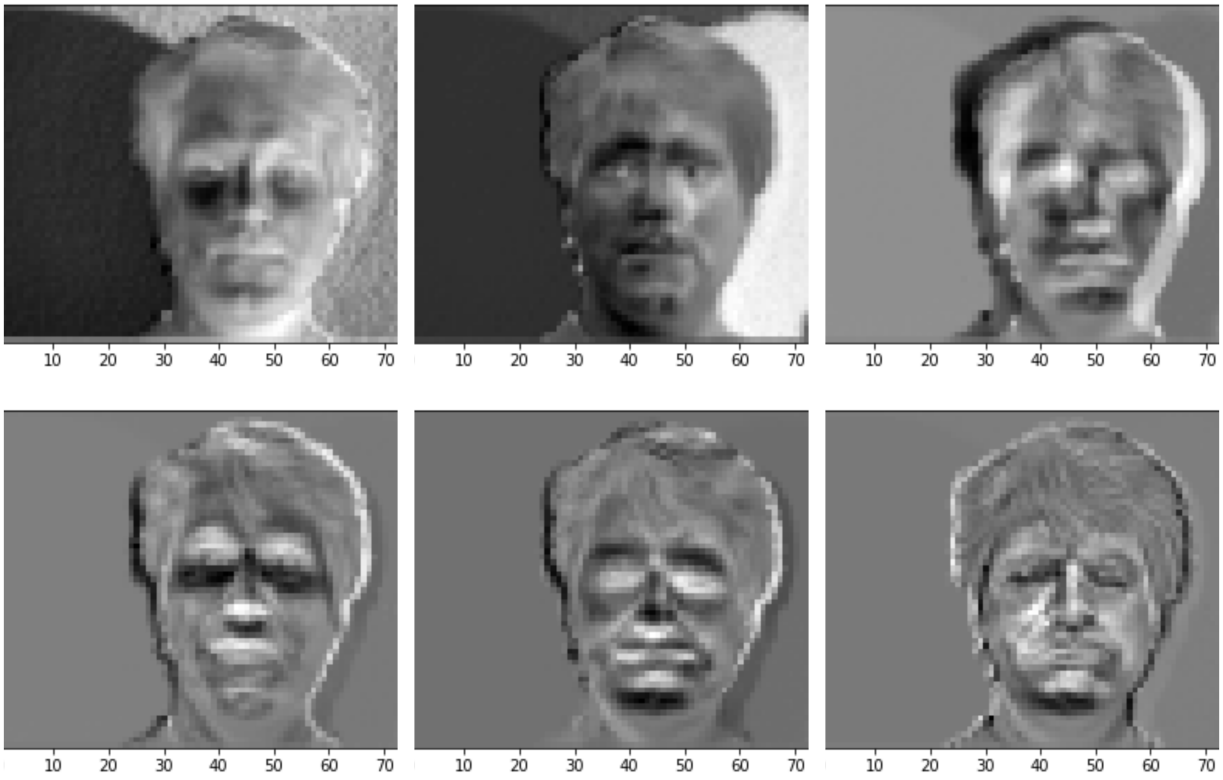

Figure 3: eigen faces of subject 1

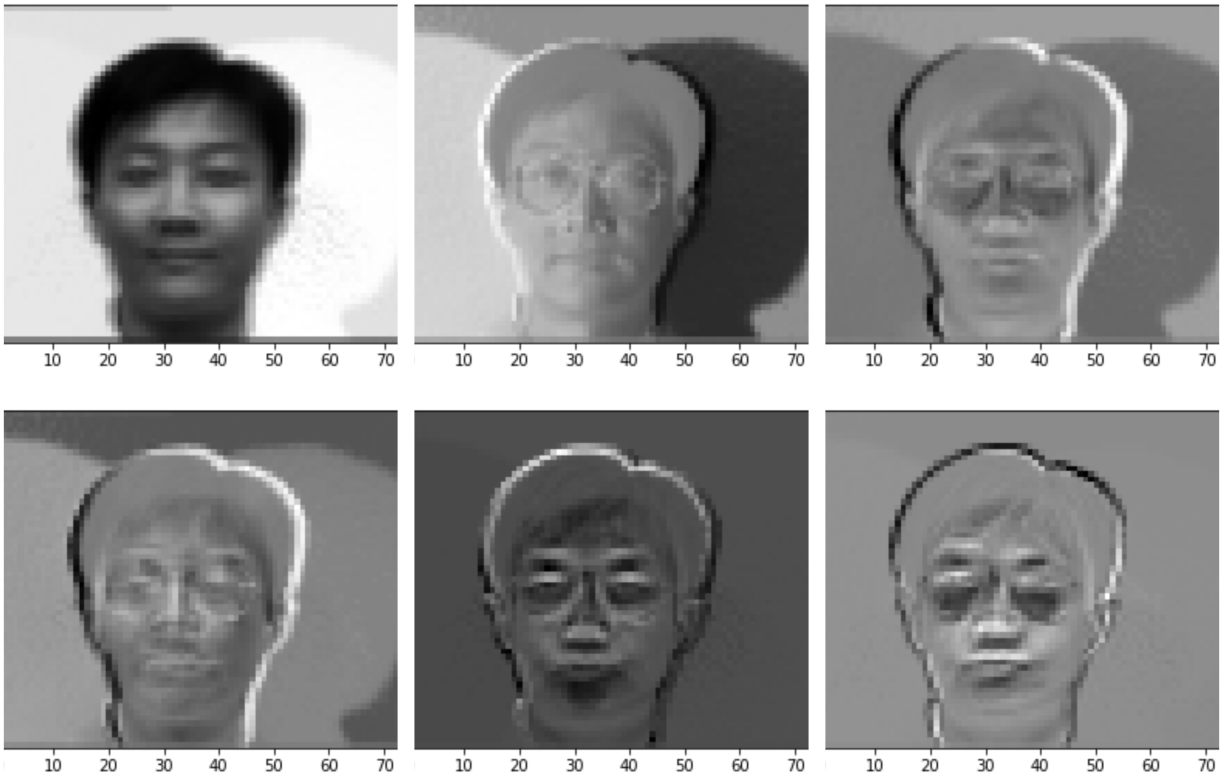

Figure 4: eigen faces of subject 2

1. (25 points) Now we will perform a simple face recognition task. ace recognition through PCA is proceeded as follows. Given the test image subject01-test.gif and subject02-test.gif, first downsize by a factor of 4 (as before), and vectorize each image. Take the top eigenfaces of Subject 1 and Subject 2, respectively. Then we calculate the *normalized inner product score* of the 2 vectorized test images with the vectorized eigenfaces:

$$s_{ij} = \frac{(\text{eigenface})_i^T (\text{test image})_j}{\|(\text{eigenface}_i)\| \cdot \|(\text{test image})_j\|}$$

Answer: Using the calculation mentioned above,I have the following table:

$$\begin{bmatrix} 0.87740349 & 0.70005538 \\ 0.0933761 & 0.41782208 \end{bmatrix}$$

The normalized inner product, also referred to with $\cos\theta$ represents how similar 2 vectors are. Value closer to 1, represents the features point in the same direction, -1 represents the features are pointing in opposite direction and 0 represents the features are orthogonal. Picking the large value closer to +1, we can conclude that S11 and S22 are matching faces.

2. (Bonus: 5 points) Explain if face recognition can work well and discuss how we can improve it possibly.

Answer: In the experiment above, we can see that S12 is somewhat close to S11. Also, the model is matching the test subject 2 more with subject1 compared to subject2. If we can train our model with

more number of faces and for each subject we can get better results with the test case.
Instead of picking the top eigen face, when I run the test images against all the eigen faces, I see better results for test2 face with 2nd and 3rd eigen face. So, I looked up, how to select best eigen face to compare against. I found a paper that talks about averaging eigen face vectors.
Reference: https://www.face-rec.org/algorithms/PCA/jcn.pdf