ISYE 6740 Fall 2020
Homework 1

1 Clustering [25 points]

Given m data points x^i, $i = 1, \ldots, m$, K-means clustering algorithm groups them into k clusters by minimizing the distortion function over $\{r^{ij}, \mu^j\}$

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} r^{ij} \|\mathrm{x}^i - \mu^j\|^2,$$

where $r^{ij} = 1$ if x^i belongs to the j-th cluster and $r^{ij} = 0$ otherwise.

1. (5 points) Prove (using mathematical arguments) that using the squared Euclidean distance $\|\mathrm{x}^i - \mu^j\|^2$ as the dissimilarity function and minimizing the distortion function, we will have

$$\mu^j = \frac{\sum_i r^{ij} \mathrm{x}^i}{\sum_i r^{ij}}.$$

That is, μ^j is the center of j-th cluster.
Hint: consider taken derivative of J with respect to μ^j.

Solution: With r^{ij} fixed, to minimize J, we only need to find μ^j such that

$$\frac{\partial J}{\partial \mu^j} = -2 \sum_{i=1}^{m} r^{ij} (x^i - \mu^j) = 0,$$

from which we can find that $\mu^j = \frac{\sum_i r^{ij} \mathrm{x}^i}{\sum_i r^{ij}}$.

2. (5 points) Now suppose we replace the similarity function (the squared ℓ_2 distance here: $\|\mathrm{x}^i - \mu^j\|^2$) by another distance measure, the quadratic distance (also known as the Mahalanobis distance) $d(x, y) = (\mathrm{x} - y)^T \Sigma (\mathrm{x} - y)$, where the given weight matrix Σ is symmetrical and positive definite (meaning that the corresponding $d(x, y) > 0$ when $x \neq y$).
(i) Show (prove) that the centroid in this case will be the same

$$\mu^j = \frac{\sum_i r^{ij} \mathrm{x}^i}{\sum_i r^{ij}}.$$

Solution: Now we have

$$J = \sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij} d(x^i, \mu^j) = \sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij}(x^i - \mu^j)^T \Sigma (x^i - \mu^j).$$

Since $\frac{\partial d(x^i,\mu^j)}{\mu^j} = -2\Sigma(x^i - \mu^j)$, we know that

$$\frac{\partial J}{\partial \mu^j} = -2\sum_{i=1}^{m} r^{ij}\Sigma(x^i - \mu^j) = 0,$$

which is equivalent to

$$\sum_{i=1}^{m} r^{ij}(x^i - \mu^j) = 0$$

since Σ is positive definite. Therefore, as before, we can find that $\mu^j = \frac{\sum_i r^{ij}\mathbf{x}^i}{\sum_i r^{ij}}$.

(ii) However, the assignment function will be different – comment how the assignment function r^{ij} should be in this case. Thus, the point is here that, depending on the choice of the similarity function (for generalized k-means, the corresponding centroid will be different as well.)
Hint: consider taken derivative of J with respect to μ^j, and use the multivariate calculus rule that the derivative of $z^T\Sigma z$ with respect to z is given by Σz.

Solution: Since the similarity function has been changed from $\|x - y\|^2$ to $d(x, y) = (x - y)^T\Sigma(x - y)$, the assignment function will be changed correspondingly. Namely, we should have $r^{ij} = 1$ if $j = \text{argmin}_{j=1,\ldots,k} d(x^i, \mu^j)$. Otherwise, $r^{ij} = 0$.

3. (5 points) Prove (using mathematical arguments) that K-means algorithm converges to a local optimum in finite steps.

Solution: Suppose there are n different data points and we want to cluster them into k groups. Then there are k^n ways to do that because each point has k options. Initially, we randomly select k centroids which give us k initial clusters after a step of cluster assignment. After that, we do centroids adjustment and cluster assignment alternatively until the group assignment dose not change anymore. Both centroids adjustment and cluster assignment decrease the objective function J. When the clusters remain unchanged, the algorithm stops and we reach a local optimum since J is not a convex function. As we mentioned before, there are only k^n ways to make k clusters. Therefore, the algorithm will converge to a local optimum after finite many steps.

4. (10 points) Calculate k-means by hands. Given 5 data points configuration in Figure 1. Assume $k = 2$ and use Manhattan distance (a.k.a. the ℓ_1 distance: given two 2-dimensional points (x_1, y_1) and (x_2, y_2), their distance is $|x_1 - x_2| + |y_1 - y_2|$). Assuming the initialization of centroid as shown, after one iteration of k-means algorithm, answer the following questions.

   (a) Show the cluster assignment;

   (b) Show the location of the new center;

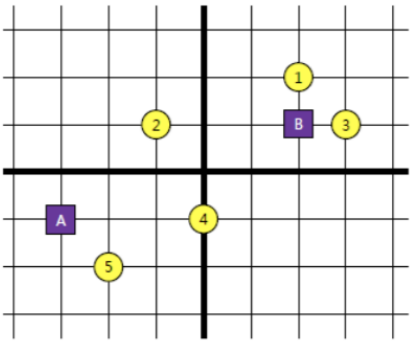


Figure 1: K-means.

(c) Will it terminate in one step?

Solution:

(a) The cluster assignment:
Cluster A: $\{4, 5\}$
Cluster B: $\{1, 2, 3\}$

(b) The new center of cluster A is $(-1, -1.5)$ and the new center of cluster B is $(4/3, 4/3)$.

(c) The algorithm will not stop after one step because point 2 will belong to cluster A in the next step as it is closer to the new center of cluster A with a distance of $5/2$. So we will need to do at least one more step of cluster assignment and centroid adjustment.

2 Image compression using clustering [40 points]

In this programming assignment, you are going to apply clustering algorithms for image compression. Your task is implementing the clustering parts with two algorithms: *K-means* and *K-medoids*. **It is required you implementing the algorithms yourself rather than calling from a package.**

K-medoids

In class, we learned that the basic K-means works in Euclidean space for computing distance between data points as well as for updating centroids by arithmetic mean. Sometimes, however, the dataset may work better with other distance measures. It is sometimes even impossible to compute arithmetic mean if a feature is categorical, e.g, gender or nationality of a person. With K-medoids, you choose a representative data point for each cluster instead of computing their average. Please note that K-medoid is different from generalized K-means: Generalized K-means still computes centre of a cluster is not necessarily one of the input data points (it is a point that minimizes the overall distance to all points in a cluster in a chosen distance metric).

Given m data points $\mathrm{x}^i (i = 1, \ldots, m)$, K-medoids clustering algorithm groups them into K clusters by minimizing the distortion function $J = \sum_{i=1}^{m} \sum_{j=1}^{k} r^{ij} D(\mathrm{x}^i, \mu^j)$, where $D(\mathrm{x}, \mathrm{y})$ is a distance measure between

two vectors x and y in same size (in case of K-means, $D(x, y) = \|\mathrm{x} - \mathrm{y}\|^2$), μ^j is the center of j-th cluster; and $r^{ij} = 1$ if x^n belongs to the k-th cluster and $r^{ij} = 0$ otherwise. In this exercise, we will use the following iterative procedure:

- Initialize the cluster center μ^j, $j = 1, ..., k$.
- Iterate until convergence:
  - Update the cluster assignments for every data point x^i: $r^{ij} = 1$ if $j = \arg\min_j D(\mathrm{x}^i, \mu^j)$, and $r^{ij} = 0$ otherwise.
  - Update the center for each cluster j: choosing another representative if necessary.

There can be many options to implement the procedure; for example, you can try many distance measures in addition to Euclidean distance, and also you can be creative for deciding a better representative of each cluster. We will not restrict these choices in this assignment. You are encouraged to try many distance measures as well as way of choosing representatives (e.g., ℓ_1 norm).

Formatting instruction

Input

- `pixels`: the input image representation. Each row contains one data point (pixel). For image dataset, it contains 3 columns, each column corresponding to Red, Green, and Blue component. Each component has an integer value between 0 and 255.
- `k`: the number of desired clusters. Too high value of K may result in empty cluster error. Then, you need to reduce it.

Output

- `class`: cluster assignment of each data point in pixels. The assignment should be 1, 2, 3, etc. For $k = 5$, for example, each cell of class should be either 1, 2, 3, 4, or 5. The output should be a column vector with `size(pixels, 1)` elements.
- `centroid`: location of k centroids (or representatives) in your result. With images, each centroid corresponds to the representative color of each cluster. The output should be a matrix with K rows and 3 columns. The range of values should be [0, 255], possibly floating point numbers.

Hand-in

Both of your code and report will be evaluated. Upload them together as a zip file. In your report, answer to the following questions:

1. (10 points) Within the k-medoids framework, you have several choices for detailed implementation. Explain how you designed and implemented details of your K-medoids algorithm, including (but not limited to) how you chose representatives of each cluster, what distance measures you tried and chose one, or when you stopped iteration.
2. (10 points) Attach a picture of your own. We recommend size of 320×240 or smaller. Run your k-medoids implementation with the picture you chose, as well as two pictures provided (beach.bmp and football.bmp), with several different K. (e.g, small values like 2 or 3, large values like 16 or 32) What did you observe with different K? How long does it take to converge for each K? Please write in your report.

3. (10 points) Run your k-medoids implementation with different initial centroids/representatives. Does it affect final result? Do you see same or different result for each trial with different initial assignments? (We usually randomize initial location of centroids in general. To answer this question, an intentional poor assignment may be useful.) Please write in your report.

4. (10 points) Repeat question 2 and 3 with k-means. Do you see significant difference between K-medoids and k-means, in terms of output quality, robustness, or running time? Please write in your report.

Note

- You may see some error message about empty clusters when you use too large k. Your implementation should treat this exception as well. That is, do not terminate even if you have an empty cluster, but use smaller number of clusters in that case.

- We will grade using test pictures which are not provided. We recommend you to test your code with several different pictures so that you can detect some problems that might happen occasionally.

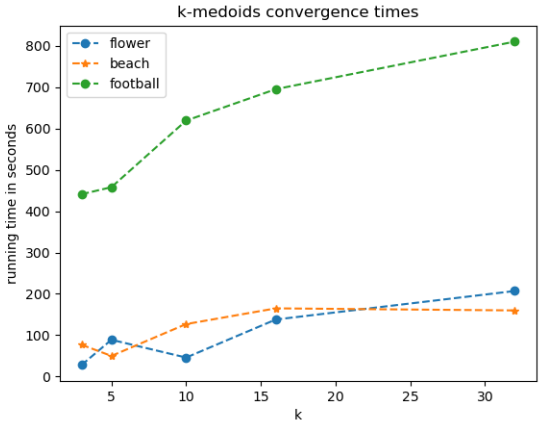
- If we detect copy from any other student's code or from the web, you will not be eligible for any credit for the entire homework, not just for the programming part. Also, directly calling built-in functions or from other package functions is not allowed.

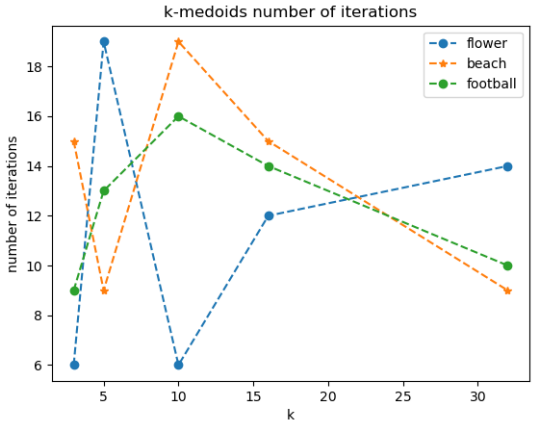
Report on problem 2

2.1 k-medoids

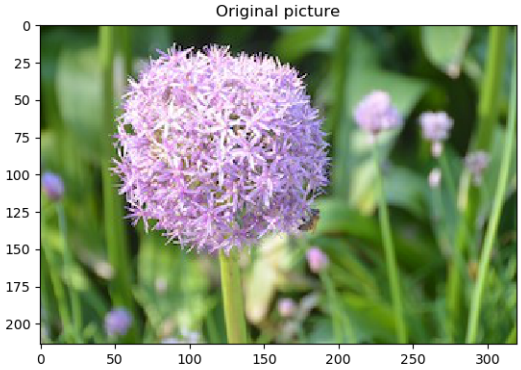
1. In my k-medoids algorithm, I randomly generated k pixels as the inital medoids for k clusters. (I also tried to randomly take k pixels from the original picture, in which case the resulting number of clusters is very close to k.) My algorithm is designed to support three different kinds of similarity functions, including l^1, l^2 and l^∞ norms. To update the representative of a cluster, I randomly choose a subset of $n \times 5\% + 1$ pixels from each cluster where n is the total number of pixels in that cluster. Then for each pixel in the subset together with the old medoid, we assume it to be the new representative and calculate the objective function value inside the cluster. We choose the pixel with the least objective function value as the new representative for that cluster. My algorithm will terminate when two consecutive differences of objective function values are less than 10^{-8} or after 20 times of iterations.

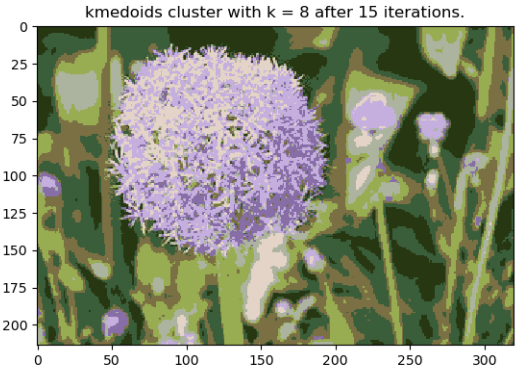
2. I tried $k = 3, 5, 10, 16, 32$, and recorded the running times in seconds and number of iterations for convergence. My results are obtained using l^1 norm as the similarity function. I used my own flower picture (can be find below in part 3) to test the algorithm and plotted the results I obtained. The plot on the left shows that k-medoids is very slow to converge for the more complicated football picture, but faster for the simpler beach and flower pictures. Since for simpler pictures, there are a lot of pixels with the same values, so the difference matrix will end up with a lot of zeros, making it much easier to calculate similarities, label pixels and update medoids. Also, the first plot shows that as k increases, the running time increases correspondingly. This is more obvious for complicated pictures like football. However, the plot on the right shows that the number of iterations seems like has no dependence on k.

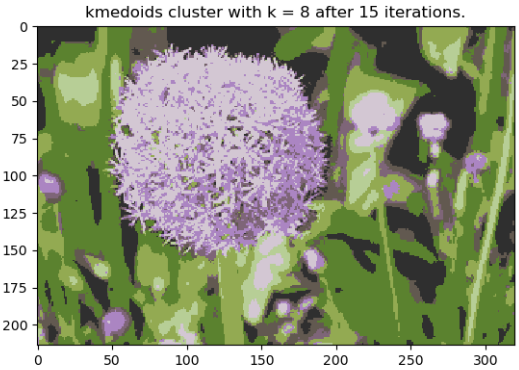




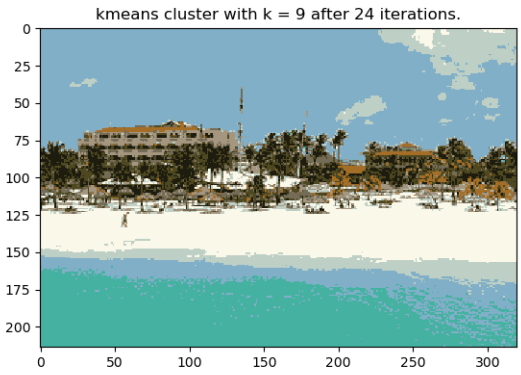
3. The initial representations dose have affect in final results. Different initial representations result in different final resulting clusters. As we can see from the following pictures. The one on the top is the original picture. And the two pictures on the bottom were obtained by running kmedoids twice both with $k = 8$, but different random initial representations.

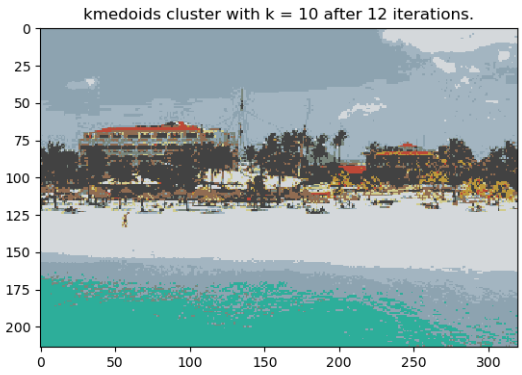




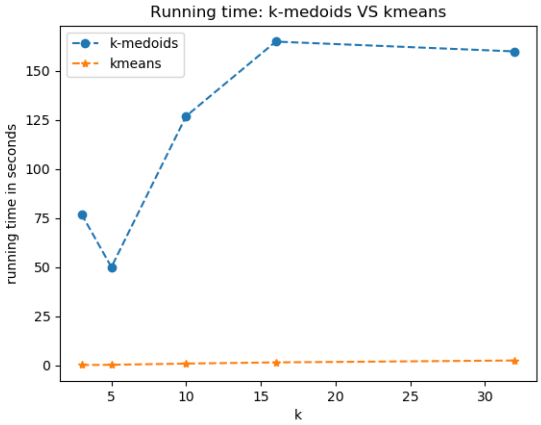


4. Question 2 and 3 for kmeans are answered in the next subsection. In terms of the output quality, k-medoids did a better job because it uses the pixels from the original picture so that the color of the resulting compressed picture is much closer to the original picture. This is not the case for kmeans because the representative color for each cluster is obtained by averaging the colors in that cluster. Also, since k-medoids uses representative pixels from the original picture, it is more robust to noise and outliers compared to kmeans.





However, kmeans has its own strength on efficiency since calculating the similarities and centroids are much easier compared with kmedoids as everything in kmeans can be done with matrix multiplications. The following plot compares the running time of kmeans and kmedoids on the picture of beach.

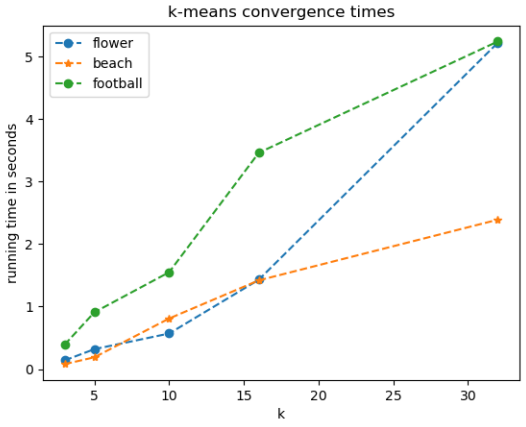


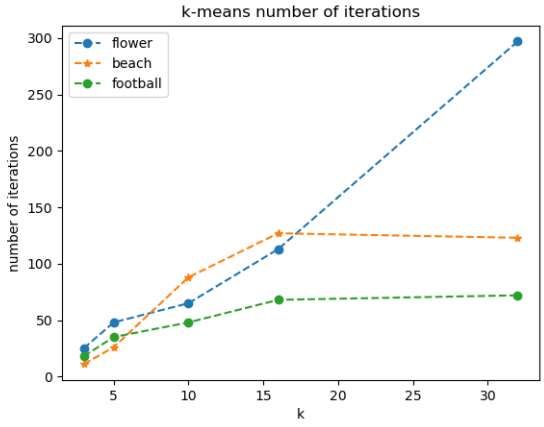
Besides, I have also observed the phenomenon that when k is large, both the kmeans and kmedoids will result in less clusters. For example, when $k = 32$, both of the two clustering algorithms ended up with around 24 clusters, instead of 32 clusters. I believe this is related to the way the initial centroids/medoids were selected. As it shows, if we initialize medoids from the picture, than the resulting number of clusters will still be k or at least very close to k.

2.2 kmeans

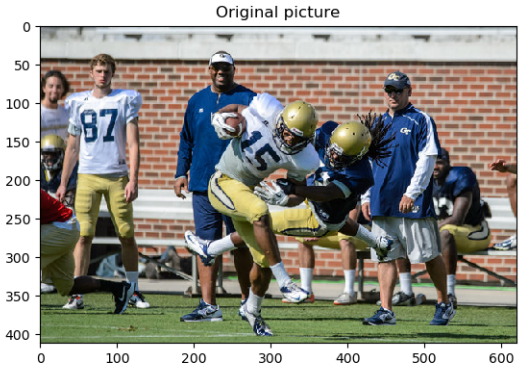
1. Nothing need to do for kmeans.

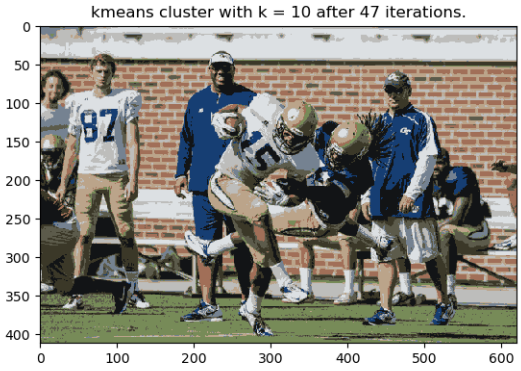
2. I tried $k = 3, 5, 10, 16, 32$, and recorded the running times in seconds, number of iterations for convergence. The results can be easily seen from the following plots. As we can see, with k increases, both the running times and the number of iterations increase. And when k is large, the number of resulting clusters is much smaller than k.

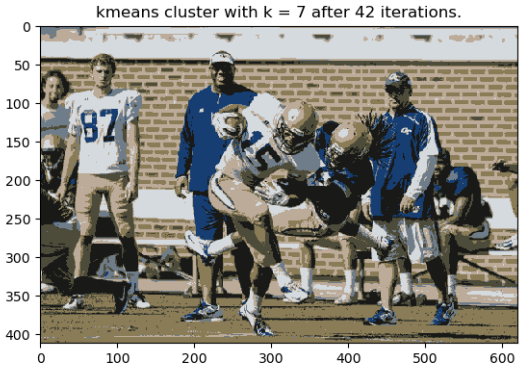




3. Different initial centroids will result in different resulting clusters. For example, I ran my kmeans with $k = 10$ and obtained the following different results. The picture on the left ended up with $k = 10$. However, the picture on the right ended up with $k = 7$.







3 Political blogs dataset [35 points]

We will study a political blogs dataset first compiled for the paper Lada A. Adamic and Natalie Glance, "The political blogosphere and the 2004 US Election", in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem (2005). The dataset nodes.txt contains a graph with $n = 1490$ vertices ("nodes") corresponding to political blogs. Each vertex has a 0-1 label (in the 3rd column) corresponding to the political orientation of that blog. We will consider this as the true label and try to reconstruct the true label from the graph using the spectral clustering on the graph. The dataset edges.txt contains edges between the vertices. You may remove isolated nodes (nodes that are not connected any other nodes).

1. (5 points) Assume the number of clusters in the graph is k. Explain the meaning of k here intuitively.

   Solution: It means that there are k tightly connected components with sparsely connected edges in the graph.

2. (10 points) Use spectral clustering to find the $k = 2$, $k = 3$, and $k = 4$ clusters in the network of political blogs (each node is a blog, and their edges are defined in the file edges.txt). **We will treat the network as an undirectly graph; thus, when constructing the adjacency matrix,**

make sure to it is symmetrical. Then report the majority labels in each cluster, when $k = 2, 3, 4$, respectively. For example, if there are $k = 2$ clusters, and their labels are $\{0, 1, 1, 1\}$ and $\{0, 0, 1\}$ then the majority label for the first cluster is 1 and for the second cluster is 0. **It is required you implementing the algorithms yourself rather than calling from a package.**

Solution:

- $k = 2$: the majority labels are [1,1];
- $k = 3$: the majority labels are [0,1,1];
- $k = 4$: the majority labels are [0,1,1,1].

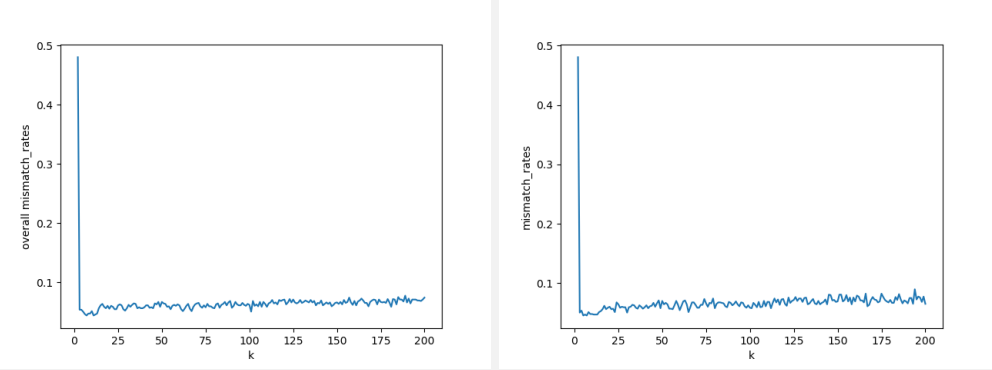
3. (5 points) Now compare the majority label with the individual labels in each cluster, and report the *mismatch rate* for each cluster, when $k = 2, 3, 4$. For instance, in the example above, the mismatch rate for the first cluster is 1/4 (only the first node differs from the majority) and the the second cluster is 1/3.

Solution:

- $k = 2$: the mismatch rates are [0.481, 0.333];
- $k = 3$: the mismatch rates are [0.022, 0.030, 0.439];
- $k = 4$: the mismatch rates are [0.022, 0.027, 0.455, 0.300].

4. (10 points) Now 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.

Solution: In this part, for each $k \in \{2, 3, ..., 200\}$, we calculated the overall mismatch rate, which is a weighted sum of the mismatch rates of all the k clusters. I can then pick the smallest k that gives the lowest overall mismatch rate. Here I present two results. The first one was obtained by calling Kmeans from Sklearn. And the second one used the kmeans algorithm I implemented myself from problem 2 but the initial centroids were randomly taken from the rows of k eigenvectors. As we can see, the results are pretty similar and pretty stable. When $k >= 3$, the mismatch rates dropped to very small values and stayed small. The reason why this happened is explained in part 5.



5. (5 points) Please explain the finding and what can you learn from this data analysis (e.g., node within same community tend to share the same political view, or now? Did you find blogs that share the same political view more tightly connected than otherwise?)

Solution: By analyzing the results, I found that there are mainly two communities and they hold different political views. The first community accounts for around 44.6% of the blogs and 97.8% of them were labeled as 0. The second community accounts for around 48.4% of the blogs and 97.3% of them were labeled as 1. So, I conclude that nodes from the same community tend to share the same political view and blogs sharing the same political view more tightly connected than otherwise.