CS 7641 CSE/ISYE 6740 Homework 1 Solution

1 Probability [15 pts]

(a) Stores A, B, and C have 50, 75, and 100 employees and, respectively, 50, 60, and 70 percent of these are women. Resignations are equally likely among all employees, regardless of stores and sex. Suppose an employee resigned, and this was a woman. What is the probability that she has worked in store C? [5 pts]

Answer:

$$\begin{aligned} P(worked\ in\ C|woman) =& \frac{P(worked\ in\ C) \times P(woman|worked\ in\ C)}{P(woman)} \\ =& \frac{\frac{100}{50+75+100} \times 0.7}{\sum_{s \in Stores} P(s) \times P(woman|worked\ in\ s)} \\ =& \frac{\frac{4}{9} \times 0.7}{\frac{2}{9} \times 0.5 + \frac{1}{3} \times 0.6 + \frac{4}{9} \times 0.7} \\ =& \frac{1}{2} \end{aligned} \tag{1}$$

(b) A laboratory blood test is 95 percent effective in detecting a certain disease when it is, in fact, present. The test also yields a false positive result for 1 percent of the healthy persons tested. That is, if a healthy person is tested then with probability 0.01 the test result will imply he has the disease. If 0.5 percent of the population actually has the disease, what is the probability a person has the disease given that his test result is positive? [5 pts]

Answer:

$$\begin{aligned} P(ill|possitive) =& \frac{P(ill) \times P(possitive|ill)}{P(ill) \times P(possitive|ill) + P(healthy) \times P(possitive|healthy)} \\ =& \frac{0.005 \times 0.95}{0.005 \times 0.95 + 0.995 \times 0.01} \\ \approx& 0.323 \end{aligned} \tag{2}$$

[c-d] On the morning of September 31, 1982, the won-lost records of the three leading baseball teams in the western division of the National League of the United States were as follows:

Each team had 3 games remaining to be played. All 3 of the Giants games were with the Dodgers, and the 3 remaining games of the Braves were against the San Diego Padres. Suppose that the outcomes of all remaining games are independent and each game is equally likely to be won by either participant. If two teams tie for first place, they have a playoff game, which each team has an equal chance of winning.

Team	Won	Lost
Atlanta Braves	87	72
San Francisco Giants	86	73
Los Angeles Dodgers	86	73

(c) What is the probability that Atlanta Braves wins the division? [2 pts]

Answer:

There are several ways to win:

s1: Giants or Dodgers wins another 2 games, and Atlanta Braves wins 2 or 3 games.

$$P(win|s1) = P(s1) = \frac{3}{4} * (\frac{3}{8} + \frac{1}{8}) = \frac{3}{8}$$

s2: Giants or Dodgers wins another 2 games, and Atlanta Braves wins 1 game.

$$P(win|s2) = P(s2) * P(win|playoff) = \frac{3}{4} * \frac{3}{8} * \frac{1}{2} = \frac{9}{64}$$

s3: Giants or Dodgers wins another 3 games, and Atlanta Braves wins 3 games.

$$P(win|s3) = P(s3) = \frac{1}{4} * \frac{1}{8} = \frac{1}{32}$$

s4: Giants or Dodgers wins another 3 games, and Atlanta Braves wins 2 games.

$$P(win|s4) = P(s4) * P(win|playoff) = \frac{1}{4} * \frac{3}{8} * \frac{1}{2} = \frac{3}{64}$$

So the total chance to win is:

$$P(win) = \frac{3}{8} + \frac{9}{64} + \frac{1}{32} + \frac{3}{64} = \frac{19}{32}$$

(d) What is the probability to have an additional playoff game? [3 pts]

Answer:

It equals to the probability to have situation 2 and 4, shown in question c).

$$P(additional) = \frac{3}{4} * \frac{3}{8} + \frac{1}{4} * \frac{3}{8} = \frac{3}{8}$$

2 Maximum Likelihood [15 pts]

Suppose we have n i.i.d (independent and identically distributed) data samples from the following probability distribution. This problem asks you to build a log-likelihood function, and find the maximum likelihood estimator of the parameter(s).

(a) Poisson distribution [5 pts]

The Poisson distribution is defined as

$$P(x_i = k) = \frac{\lambda^k e^{-\lambda}}{k!} (k = 0, 1, 2, ...).$$

What is the maximum likelihood estimator of λ?

Answer:

The likelihood function is:

$$f(x_1, \ldots, x_n|\lambda) = \frac{\lambda^{x_1} e^{-\lambda}}{x_1!} \ldots \frac{\lambda^{x_n} e^{-\lambda}}{x_n!}$$

The log-likelihood function is:

$$\begin{aligned} L(x_1, \ldots, x_n|\lambda) &= \sum_{i=1}^{n} -\lambda + x_i \ln \lambda - \ln(x_i!) \\ &= -n\lambda + (\ln \lambda) \sum_{i=1}^{n} x_i - \ln(\prod_{i=1}^{n} x_i!) \end{aligned}$$

Take derivative of λ:

$$\frac{dL}{d\lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda} = 0$$

So

$$\lambda = \frac{\sum_{i=1}^{n} x_i}{n}$$

(b) Multinomial distribution [5 pts]

The probability density function of Multinomial distribution is given by

$$f(x_1, x_2, \ldots, x_k; n, \theta_1, \theta_2, \ldots, \theta_k) = \frac{n!}{x_1! x_2! \cdots x_k!} \prod_{j=1}^{k} \theta_j^{x_j},$$

where $\sum_{j=1}^{k} \theta_j = 1, \sum_{j=1}^{k} x_j = n$. What is the maximum likelihood estimator of $\theta_j, j = 1, \ldots k$?

Answer:

The log-likelihood function is:

$$L(x_1, x_2, \ldots, x_k; n, \theta_1, \theta_2, \ldots, \theta_k) = \ln n! - \sum_{j=1}^{k} \ln x_j! + \sum_{j=1}^{k} x_j \ln \theta_j$$

Our target is to solve a constrained optimization problem:

$$\begin{aligned} \underset{\theta_j, j=1,\ldots,k}{\arg\max} \quad & L(x_1, x_2, \ldots, x_k; n, \theta_1, \theta_2, \ldots, \theta_k) \\ \text{s.t.} \quad & \theta_j \geq 0, j = 1, \ldots, k \\ & \sum_{j=1}^{k} \theta_j = 1 \end{aligned} \tag{3}$$

We can get the following gradient about $\vec{\theta}$:

$$\nabla L(L(x_1, x_2, \ldots, x_k; n, \theta_1, \theta_2, \ldots, \theta_k) = (\frac{x_1}{\theta_1}, \ldots, \frac{x_k}{\theta_k})$$

Let g be the constrain function,

$$g(\vec{\theta}) = \sum_{j=1}^{k} \theta_j = 1$$

So the gradient is:

$$\nabla g = (1, 1, \ldots, 1)$$

According to Lagrange multiplier, there exists real value λ, such that $\nabla L = \lambda \nabla g$. So we can get $\theta_j = \frac{\lambda}{x_j}$. Since $g(\vec{\theta}) = 1$, we can have $\lambda = n$, so the maximum likelihood estimation of θ_j is:

$$\theta_j = \frac{x_j}{n}, j = 1, \ldots, k$$

(c) Gaussian normal distribution [5 pts]

Suppose we have n i.i.d (Independent and Identically Distributed) data samples from a univariate Gaussian normal distribution $\mathcal{N}(\mu, \sigma^2)$, which is given by

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

What is the maximum likelihood estimator of μ and σ^2?

Answer:

The log-likelihood function is:

$$L(x_1, \ldots, x_n | \mu, \sigma) = -\frac{1}{2} n \ln(2\pi) - n \ln \sigma - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}$$

Take partial derivative of μ:

$$\frac{\partial L}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu)$$

Let it be zero, so we get:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Take partial derivative of σ^2:

$$\begin{aligned}
\frac{\partial L}{\partial \sigma^2} =& -\frac{n}{2\sigma^2} - [\frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2] \times (-\frac{1}{\sigma^4}) \\
=& -\frac{n}{2\sigma^2} + [\frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2] \frac{1}{\sigma^4} \\
=& \frac{1}{2\sigma^2} [\frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 - n]
\end{aligned}$$

Let it be zero, so we get:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

3 Principal Component Analysis [20 pts]

In class, we learned that Principal Component Analysis (PCA) preserves variance as much as possible. We are going to explore another way of deriving it: minimizing reconstruction error.

Consider data points $\mathrm{x}^n (n = 1, ..., N)$ in D-dimensional space. We are going to represent them in $\{\mathrm{u}_1, ..., \mathrm{u}_D\}$ orthonormal basis. That is,

$$\mathrm{x}^n = \sum_{i=1}^{D} \alpha_i^n \mathrm{u}_i = \sum_{i=1}^{D} (\mathrm{x}^{n T} \mathrm{u}_i)\mathrm{u}_i.$$

Here, α_i^n is the length when x^n is projected onto u_i.

Suppose we want to reduce the dimension from D to $M < D$. Then the data point x^n is approximated by

$$\tilde{\mathrm{x}}^n = \sum_{i=1}^{M} z_i^n \mathrm{u}_i + \sum_{i=M+1}^{D} b_i \mathrm{u}_i.$$

In this representation, the first M directions of u_i are allowed to have different coefficient z_i^n for each data point, while the rest has a constant coefficient b_i. As long as it is the same value for all data points, it does not need to be 0.

Our goal is setting u_i, z_i^n, and b_i for $n = 1, ..., N$ and $i = 1, ..., D$ so as to minimize reconstruction error. That is, we want to minimize the difference between x^n and $\tilde{\mathrm{x}}^n$ over $\{\mathrm{u}_i, z_i^n, b_i\}$:

$$J = \frac{1}{N} \sum_{n=1}^{N} \|\mathrm{x}^n - \tilde{\mathrm{x}}^n\|^2.$$

(a) What is the assignment of z_j^n for $j = 1, ..., M$ minimizing J? [5 pts]

Answer:

Let's rewrite J as follows:

$$J = \frac{1}{N} \sum_{n=1}^{N} (x^n)^T x^n - 2(x^n)^T \tilde{x}^n + (\tilde{x}^n)^T \tilde{x}^n$$

We now try to get the partial derivatives of J about z_j^n

$$\begin{aligned} \frac{\partial J}{\partial z_j^n} &= \frac{\partial}{\partial z_j^n} \left(\frac{1}{N} \sum_{t=1}^{N} (x^t)^T x^t - 2(x^t)^T \tilde{x}^t + (\tilde{x}^t)^T \tilde{x}^t \right) \\ &= \frac{\partial}{\partial z_j^n} \left(\frac{1}{N} \left(-2(x^n)^T \tilde{x}^n + (\tilde{x}^n)^T \tilde{x}^n \right) \right) \end{aligned}$$

In the last step of the equation above, we just ignore the terms not related to z_j^n. We set the above equation to be zero, and get the following:

$$\begin{aligned} 0 &= \frac{\partial}{\partial z_j^n} \left(-2(x^n)^T \tilde{x}^n + (\tilde{x}^n)^T \tilde{x}^n \right) \\ &= \frac{\partial}{\partial z_j^n} \left(-2 z_j^n (x^n)^T u_j + \sum_{i,j}^{M} z_i^n z_j^n u_i^T u_j + 2 \sum_{i=1}^{M} \sum_{j=M+1}^{D} z_i^n b_j u_i^T u_j \right) \end{aligned}$$

Since $u_i^T u_j = 0$ for $i \neq j$, and $u_i^T u_j = 1$ for $i = j$, we can get:

$$\begin{aligned} 0 =& \frac{\partial}{\partial z_j^n} \left(-2 z_j^n (x^n)^T u_j + (z_i^n)^2 \right) \\ =& -2 (x^n)^T u_j + 2 z_j^n \end{aligned}$$

So the conclusion is:

$$z_j^n = (x^n)^T u_j, j = 1, \ldots, M$$

(b) What is the assignment of b_j for $j = M+1, ..., D$ minimizing J? [5 pts]

Answer:

Similarly, we take the partial derivatives of J with respect to b_j, and then we get:

$$\begin{aligned} \frac{\partial J}{\partial b_j} =& \frac{\partial}{\partial b_j} \left(\frac{1}{N} \sum_{n=1}^{N} (x^n)^T x^n - 2 (x^n)^T \tilde{x}^n + (\tilde{x}^n)^T \tilde{x}^n \right) \\ =& \frac{-2}{N} \frac{\partial}{\partial b_j} \sum_{n=1}^{N} \left(b_j (x^n)^T u_j + (\tilde{x}^n)^T \tilde{x}^n \right) \\ =& -2 \frac{\sum_{n=1}^{N} (x^n)^T}{N} u_j + \frac{1}{N} \frac{\partial}{\partial b_j} \sum_{n=1}^{N} (\tilde{x}^n)^T \tilde{x}^n \\ =& -2 \bar{x} u_j + \frac{1}{N} \frac{\partial}{\partial b_j} \sum_{n=1}^{N} \left(\sum_{i,j}^{M} z_i^n z_j^n u_i^T u_j + 2 \sum_{i=1}^{M} \sum_{j=M+1}^{D} z_i^n b_j u_i^T u_j + \sum_{i=M+1, j=M+1}^{D} b_i b_j u_i^T u_j \right) \end{aligned}$$

Here $\bar{x}^T = \frac{\sum_{n=1}^{N} (x^n)^T}{N}$, i.e., the mean value of samples. Since u_i are orthonormal, only the terms with $i == j$ have non-zero values. So we get:

$$\begin{aligned} \frac{\partial J}{\partial b_j} =& -2 \bar{x} u_j + \frac{1}{N} \frac{\partial}{\partial b_j} \sum_{n=1}^{N} \left(b_j^2 \right) \\ =& -2 \bar{x} u_j + 2 b_j \end{aligned}$$

Set the derivatives to be zero, we can get:

$$b_j = \bar{x}^T u_j, j = M+1, \ldots, D$$

(c) Express optimal $\tilde{\mathbf{x}}^n$ and $\mathbf{x}^n - \tilde{\mathbf{x}}^n$ using your answer for (a) and (b). [2 pts]

Answer:

$$\tilde{x}^n = \sum_{i=1}^{M} (x^n)^T u_i + \sum_{i=M+1}^{D} (\bar{x}^T u_i) u_i$$

$$x^n - \tilde{x}^n = \sum_{i=M+1}^{D} \left((x_n - \bar{x})^T u_i \right) u_i$$

(d) What should be the $\mathbf{u}_i$ for $i = 1, ..., D$ to minimize J? [8 pts]

Hint: Use $S = \frac{1}{N}\sum_{n=1}^{N}(\mathrm{x}^n - \bar{\mathrm{x}})(\mathrm{x}^n - \bar{\mathrm{x}})^T$ for sample covariance matrix.

Answer:
Plugin the c) into loss function J, we can get:

$$\begin{aligned}
J =& \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x^n)^T u_i - \bar{x}^T u_i)^2 \\
=& \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} u_i^T(x^n - \bar{x})((x^n)^T - \bar{x}^T)u_i \\
=& \sum_{i=M+1}^{D} u_i^T \left(\frac{1}{N}\sum_{n=1}^{N}(x^n - \bar{x})((x^n)^T - \bar{x}^T)\right) u_i \\
=& \sum_{i=M+1}^{D} u_i^T S u_i
\end{aligned}$$

Here we can formulate the above problem as a constrained optimization problem:

$$\begin{aligned}
&\underset{u_i, i=M+1,\ldots,D}{\arg\min} && \sum_{i=M+1}^{D} u_i^T S u_i \\
&\text{s.t.} && u_i^T u_i = 1, i = M+1, \ldots, D
\end{aligned} \tag{4}$$

Using Lagrange multiplier, we can re-formulate it to unconstrained optimization problem. We now consider the minimization of

$$\tilde{J} = \sum_{i=M+1}^{D} u_i^T S u_i + \sum_{i=M+1}^{D} \lambda_i(1 - u_i^T u_i)$$

Take the derivatives with respect to u_i, and set it to be zero, we can get:

$$S u_i = \lambda_i u_i$$

So the ith component should be an eigenvector of matrix S. In order to get the minimum loss, the λ_i should be as small as possible. That is to say, we should set u_i to be the eigenvector corresponding to ith largest eigenvalue.

4 Clustering [20 pts]

[a-b] Given N data points $\mathrm{x}^n (n = 1, \ldots, N)$, K-means clustering algorithm groups them into K clusters by minimizing the distortion function over $\{r^{nk}, \mu^k\}$

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} \|\mathrm{x}^n - \mu^k\|^2,$$

where $r^{nk} = 1$ if x^n belongs to the k-th cluster and $r^{nk} = 0$ otherwise.

(a) Prove that using the squared Euclidean distance $\|\mathbf{x}^n - \mu^k\|^2$ as the dissimilarity function and minimizing the distortion function, we will have

$$\mu^k = \frac{\sum_n r^{nk} \mathbf{x}_n}{\sum_n r^{nk}}.$$

That is, μ^k is the center of k-th cluster. [5 pts]

Answer:

Take the derivatives with respect to μ^k, we can get:

$$\begin{aligned}
\frac{\partial J}{\partial \mu^k} =& \frac{\partial}{\partial \mu^k} \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} \left((x^n)^T - 2(x^n)^T \mu^k + (\mu^k)^T \mu^k \right) \\
=& \frac{\partial}{\partial \mu^k} \sum_{n=1}^{N} r^{nk} \left(-2(x^n)^T \mu^k + (\mu^k)^T \mu^k \right) \\
=& \sum_{n=1}^{N} r^{nk} \left(-2(x^n) + 2\mu^k \right)
\end{aligned}$$

Set the derivatives to be zero, and we get the conclusion:

$$\mu^k = \frac{\sum_n r^{nk} \mathrm{x}_n}{\sum_n r^{nk}}$$

(b) Prove that K-means algorithm converges to a local optimum in finite steps. [5 pts]

Answer:

In each iteration, if we change the assignment of cluster index of the data points, then it must be the situation that such kind of change will result in the decrease of loss function J. Similarly, if we need to adjust the center of each cluster, it will also decrease the loss. So generally, in each step, the J decreases.

What's more, each kind of assignment to r^{nk} corresponds to a step in K-means, and vice versa. Since the number of assignments is limited, there should be limited step for K-means.

That is to say, it will finally converges to a local minimum value, in finite steps.

[c-d] In class, we discussed bottom-up hierarchical clustering. For each iteration, we need to find two clusters $\{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_m\}$ and $\{\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_p\}$ with the minimum distance to merge. Some of the most commonly used distance metrics between two clusters are:

- Single linkage: the minimum distance between any pairs of points from the two clusters, i.e.

$$\min_{\substack{i=1,\ldots,m \\ j=1,\ldots,p}} \|\mathrm{x}_i - \mathrm{y}_j\|$$

- Complete linkage: the maximum distance between any parts of points from the two clusters, i.e.

$$\max_{\substack{i=1,\ldots,m \\ j=1,\ldots,p}} \|\mathrm{x}_i - \mathrm{y}_j\|$$

- Average linkage: the average distance between all pair of points from the two clusters, i.e.

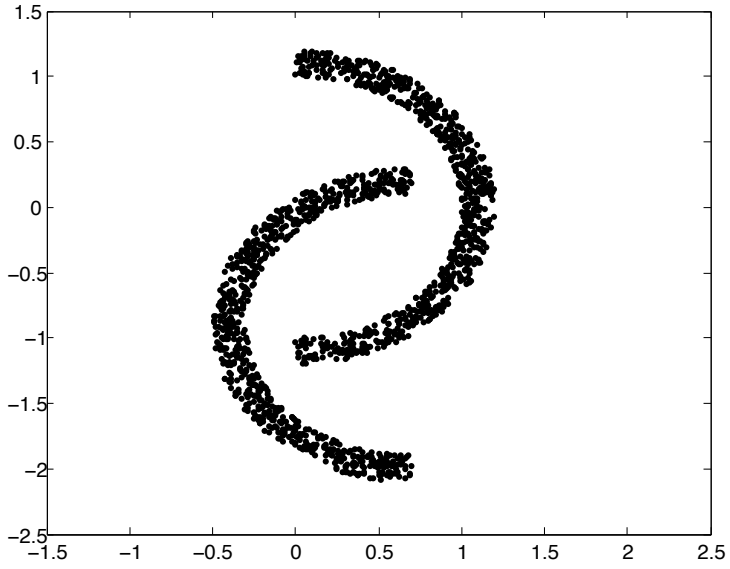
$$\frac{1}{mp} \sum_{i=1}^{m} \sum_{j=1}^{p} \|\mathrm{x}_i - \mathrm{y}_j\|$$

(c) When we use the bottom up hierarchical clustering to realize the partition of data, which of the three cluster distance metrics described above would most likely result in clusters most similar to those given by K-means? (Suppose K is a power of 2 in this case). [5 pts]

Answer:

The **Average linkage** would be the most similar to K-means, since it also tries to minimize the average distance in each step, and hope to approximate the overall minimum of average by such kind of greedy algorithm.

(d) For the following data (two moons), which of these three distance metrics (if any) would successfully separate the two moons? [5 pts]



Answer:

The **Single linkage** would successfully separate the two moons.

Let's consider building a graph from the data points. We set a threshold d, such that d is smaller than the minimum distance between two moons. If the distance between two points is smaller than or equals to d, then we add an un-directed edge to them. It is easy to see that we will get two connected components finally, where each component corresponds to a moon.

The **Single linkage** behaves like the method above. If we check the $\min_{\substack{i=1,\ldots,m \\ j=1,\ldots,p}} \|\mathrm{x}_i - \mathrm{y}_j\|$ in each step, we can see that it will never exceeds the threshold d, since there is always shorter edge connecting clusters within each moon.

The **Complete linkage** will fail. The maximum distance between two ends of each moon is larger than the minimum distance between moons. So there is possibility that this metric will merge two clusters from different moons.

Similarly, **Average linkage** behaves like K-means, which might merge halves of different moons.

5 Programming: Image compression [30 pts]

In this programming assignment, you are going to apply clustering algorithms for image compression. Before starting this assignment, we strongly recommend reading PRML Section 9.1.1, page 428 – 430.

To ease your implementation, we provide a skeleton code containing image processing part. `homework1.m` is designed to read an RGB bitmap image file, then cluster pixels with the given number of clusters K. It shows converted image only using K colors, each of them with the representative color of centroid. To see what it looks like, you are encouraged to run `homework1('beach.bmp', 3)` or `homework1('football.bmp', 2)`, for example.

Your task is implementing the clustering parts with two algorithms: *K-means* and *K-medoids*. We learned and demonstrated K-means in class, so you may start from the sample code we distributed.

The file you need to edit is `mykmeans.m` and `mykmedoids.m`, provided with this homework. In the files, you can see it calls Matlab function `kmeans` initially. Comment this line out, and implement your own in the files. You would expect to see similar result with your implementation of K-means, instead of `kmeans` function in Matlab.

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.

Given N data points $\mathrm{x}^n (n = 1, ..., N)$, K-medoids clustering algorithm groups them into K clusters by minimizing the distortion function $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} D(\mathrm{x}^n, \mu^k)$, 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^k is the center of k-th cluster; and $r^{nk} = 1$ if x^n belongs to the k-th cluster and $r^{nk} = 0$ otherwise. In this exercise, we will use the following iterative procedure:

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

Formatting instruction

Both `mykmeans.m` and `mykmedoids.m` take input and output format as follows. You should not alter this definition, otherwise your submission will print an error, which leads to zero credit.

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 `mykmeans.m` and `mykmedoids.m` files with your implementation. In your report, answer to the following questions:

1. 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.

   Answer:

   - Choose representatives:

     Since it is unrealistic to use PAM method for such large scaled clustering, we should do some approximation here. Specifically, I want to mimics the K-means algorithm. So my intuition is to choose the center which is close to the mean value of current cluster. Below is the pipeline:

     (a) Get mean value of current cluster;
     (b) Make the points unique, since there will be many pixels with same color;
     (c) Select top T candidates which are close to mean point;
     (d) Calculate the overall distance for each candidate;
     (e) Select the candidate with minimum overall distance.

   - Distance measures:

     I've tried the euclidean distance, cityblock distance and chebychev distance. I'v also tried to map the original RGB space to LAB space. But it showed that euclidean distance in RGB space produced the most decent result.

   - Stop criteria:

     I've set three criteria. Only when all of these three satisfied will the algorithm stop.

     – The current loss decreases. This is to prevent some precision errors.
     – The difference between loss in last two iteration is small, e.g., 0.1.
     – The relative change of loss in last two iteration is small, e.g., $1e - 8$.

2. Attach a picture of your own. We recommend size of 320×240 or smaller.

3. Run your K-medoids implementation with the picture you chose above, 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?

4. 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.)

5. 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?

Note

- You may see some error message about empty clusters even with Matlab implementation, 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 Matlab function `kmeans` or other clustering functions is not allowed.