HW 7

Gawon Kim
ECE M146
UID: 305-186-572.

① $f(\mu_k) = \sum_{n \in C_k} \|x_n - \mu_k\|_1 = \sum_{i=1}^{15} \sum_{n \in C_k} |x_{ni} - \mu_{ki}|$

1-a $f(\bar{y}) = \sum_{j=1}^{N_k} |y_j - \bar{y}|$ for $y_j \in R$. * subgradient *

$f(z) \geq f(x) + g(z-x) \rightarrow f(z) - f(x) \geq g(z-x)$

g is a subgradient of f at x when $f(x)+g(z-x)$ is a global estimator of f

$f \leftarrow$ Subdifferentiable at x when there is at least one subgradient at x

f is subgradient of x = subdifferentiable of f at x.

* $f(x) = |x|$

When $x<0$ the subgradient is unique.

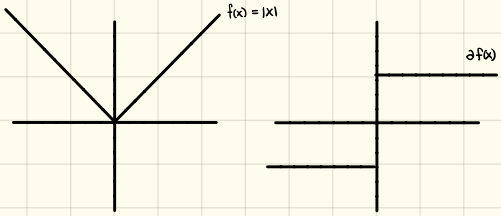
$\partial f(x) = \{-1\}$

When $x>0$, $\partial f(x) = \{1\}$

When $x=0$, the subdifferential is defined by

$|x| \geq gx$ when g is 1 or -1.

$\rightarrow \partial f = \{1, -1\}$



1-b $\min \sum_{j=1}^{N_k} |y_j - \bar{y}|$

To satisfy subdifferentiable, there should be zero at least in one element and y_i value is getting increasing depends on the movement of index $\bar{i}$. When doing partial derivative of $f(y_i)$, $\partial f(y) = \partial f(y_1) + \partial f(y_2) + \cdots \partial f(y_K)$ the value will be -1 or 1

To minimize it, the median can be zero and the smaller elements can be -1 and the bigger element (y_i) can be +1. So that canceling out each other.

$\bar{y}^*$ will minimize $\sum_{j=1}^{N_k} |y_j - \bar{y}|$ is the median of $\{y_1 \sim y_{N_k}\}$

1-c

Step 1, r_{nk} is assigning x_n to the closest cluster for each x_n. As the equation, $k = \arg\min |x_n - x_i^*|$. In this equation x_i^* is a median of i cluster and r_{nk} is one when it is belonged to the cluster

Step 2 for $\bar{i} \in \{1, 2, \ldots K\}$, for all $\bar{i}$, update the value of median for $\bar{i}$

* median $\{x_n : r_{n\bar{i}} = 1\}$

②

2-a $A = \begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix}$ $\lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$ $A - \lambda I = \begin{bmatrix} 3-\lambda & 2 \\ 2 & -\lambda \end{bmatrix}$

$\det \begin{bmatrix} 3-\lambda & 2 \\ 2 & -\lambda \end{bmatrix} = -(3-\lambda)\lambda - 4 = \lambda^2 - 3\lambda - 4 = 0$

$\lambda^2 - 3\lambda - 4 = (\lambda - 4)(\lambda + 1) = 0$ $\lambda = 4, -1$

$A - \lambda I = \begin{bmatrix} 3-\lambda & 2 \\ 2 & -\lambda \end{bmatrix} = B$

When $\lambda = 4$ $B = \begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix}$ $Bx = 0$ $-x_1 + 2x_2 = 0$

$x_1 = 2x_2$

Let $x_1 = 2$, $x_2 = 1$

eigen value : 4

eigen vector : $\left[\frac{2}{\sqrt{5}} \quad \frac{1}{\sqrt{5}} \right] = u_1$

When $\lambda = -1$ $B = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$ $Bx = 0$ $2x_1 + x_2 = 0$

$2x_1 = -x_2$

Let $x_1 = 1$, $x_2 = -2$

eigen value : -1

eigen vector : $\left[\frac{1}{\sqrt{5}} \quad \frac{-2}{\sqrt{5}} \right] = u_2$

2-b $u_1^T u_1 = 1$ $u_2^T u_2 = 1$

$A = U \Lambda U^{-1}$ U : matrix (each colum is eigen value of A

Λ : dragonal matrix

$U = \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix}$, $U^{-1} = -\frac{1}{5}\begin{bmatrix} -2 & -1 \\ -1 & 2 \end{bmatrix}$

$\Lambda = \begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix}$

$A = \left[\begin{array}{c|c} 2 & 1 \\ 1 & -2 \end{array}\right] \left[\begin{array}{c|c} 4 & 0 \\ 0 & -1 \end{array}\right] \left[\begin{array}{c|c} -2 & -1 \\ -1 & 2 \end{array}\right] \frac{-1}{5}$

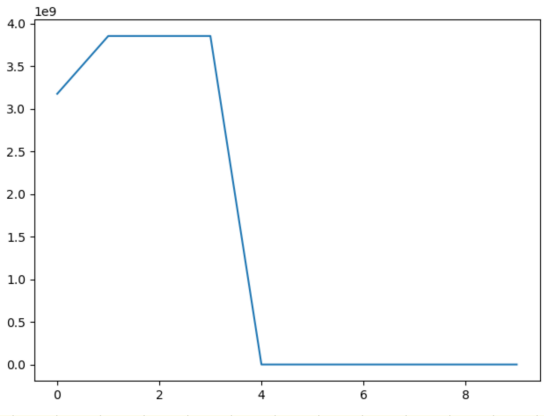
$= \left[\begin{array}{c|c} 8 & -1 \\ 4 & 2 \end{array}\right] \left[\begin{array}{c|c} -2 & -1 \\ -1 & 2 \end{array}\right] \left(-\frac{1}{5}\right)$

$= \begin{bmatrix} -15 & -10 \\ -10 & 0 \end{bmatrix} \left(-\frac{1}{5}\right) = \begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix}$

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

"""
 MU value for b
 [[147. 200. 250.]
  [  0.   0.   0.]
  [137. 141.  57.]
  [ 31.  70. 125.]]

ITR=10
J = [3.17714549e+09 3.85576888e+09 3.85584076e+09 3.85584076e+09
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00]

 ITR=10      K = 16
 MU value is
 [[204.81911369 223.51605521 229.27264059]
 [172.76879134 189.95224768 194.98604428]
 [171.33775539 188.33198992 191.81679261]
 [169.10443734 186.53001412 190.8369641 ]
 [180.1574569  192.45258183 191.58246094]
 [177.98176647 190.45088211 190.13512524]
 [174.32042506 186.65266953 184.95372542]
 [185.28253622 194.40710332 190.35956802]
 [188.35814331 194.59698956 187.97761779]
 [177.00181633 182.14370675 174.66152041]
 [139.04322238 143.81714317 135.72721039]
 [143.23318653 147.56318002 139.26170862]
 [143.44539854 147.63004615 139.19552234]
 [143.05820491 147.00532338 138.40510813]
 [143.02923084 147.04781564 138.53027858]
 [141.49944227 145.80183048 137.77128362]]

"""


##########################################
iteration = 10
k = 16  # k value in k-means algorithm
##########################################
img = cv2.imread("UCLA_Bruin.jpg")   # by using imread, read the image
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#cv2.imshow("UCLA_Bruin.jpg", img)
#cv2.waitKey(0)
#cv2.destroyAllWindows()
INIT_I = 0; INIT_J = 0
for i in range(0,300,1):
    for j in range(0,400,1):
        if img[i][j][0] == 147:
            if img[i][j][1] == 200:
                if img[i][j][2] == 250:
                    INIT_J = j
                    INIT_I = i
MU_ARR = np.zeros((k, 3))
MU_ARR[0] = img[INIT_I][INIT_J]

for K_VAL in range(1,k):
    MAX_X = 0; MAX_Y = 0
    MAX_diff = 0
    for y in range(300):
        for x in range(400):
            MIN_diff_ARR = []
            for index in range(0,K_VAL):
                distance = (MU_ARR[index][0]-img[y][x][0])**2 + \
                           (MU_ARR[index][1]-img[y][x][1])**2 + \
                           (MU_ARR[index][2]-img[y][x][2])**2
                MIN_diff_ARR.append(distance)
            if np.min(MIN_diff_ARR) > MAX_diff:
                MAX_X = x
                MAX_Y = y
                MAX_diff = np.min(MIN_diff_ARR)
    MU_ARR[K_VAL] = img[MAX_Y][MAX_X]

print(MU_ARR)
```

```
J = np.zeros(iteration)
r = np.zeros((300, 400, k))
for itr in range(iteration):
    for i in range(300):
        for j in range(400):
            DISTANCE_ARR = []
            for k_val in range(k):
                DISTANCE_ARR.append(np.sqrt(np.linalg.norm(img[i][j] - MU_ARR[k_val])**2))
            T = np.argmin(DISTANCE_ARR)
            r[i][j][T] = 1
    n = np.zeros(3)
    d = 0
    for k_val in range(k):
        for i in range(300):
            for j in range(400):
                if r[i][j][k_val] == 1:
                    n += img[i][j]
                    d += 1
        MU_ARR[k_val] = n/d
    for k_val in range(k):
        for i in range(300):
            for j in range(400):
                if r[i][j][k_val] == 1:
                    J[itr] += np.linalg.norm(img[i][j] - MU_ARR[k_val])**2

#plt.plot(J)
#plt.show()
img2 = img
for itr in range(iteration):
    for i in range(300):
        for j in range(400):
            Close_diff = (img2[i][j][0] - MU_ARR[0][0])**2 + \
                         (img2[i][j][1] - MU_ARR[0][1])**2 + \
                         (img2[i][j][2] - MU_ARR[0][2])**2
            Close_diff = np.sqrt(Close_diff)
            Close_K = 0
            for k_val in range(1,k):
                data = (img2[i][j][0] - MU_ARR[k_val][0])**2 + \
                       (img2[i][j][1] - MU_ARR[k_val][1])**2 + \
                       (img2[i][j][2] - MU_ARR[k_val][2])**2
                data = np.sqrt(data)
                if data < Close_diff:
                    Close_diff = data
                    Close_K = k_val
            img2[i][j] = MU_ARR[Close_K]
img2 = cv2.cvtColor(img2, cv2.COLOR_RGB2BGR)

cv2.imshow("NEW_Bruin", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

③



(b) By the graph, the K means algorithm function is converged to zero for the increasing iteration.

(c) K is the number of clusters and K made the picture to be more precise and bigger K value in picture is more similar to original picture. because it would have more clusters and could have more various color for consisting image

(d) For image, 24N bits which is N: pixels

24 · 300 · 400 bits. = 2880000 bits.

For K cluster, $N \log_2 K + 24K$ (bits)

When K=4, $300 \cdot 400 \log_2 4 + 24 \cdot 4$
$= 240096$ (bits)

When K=8, $300 \cdot 400 \log_2 8 + 24 \cdot 8$
$= 360192$ (bits)

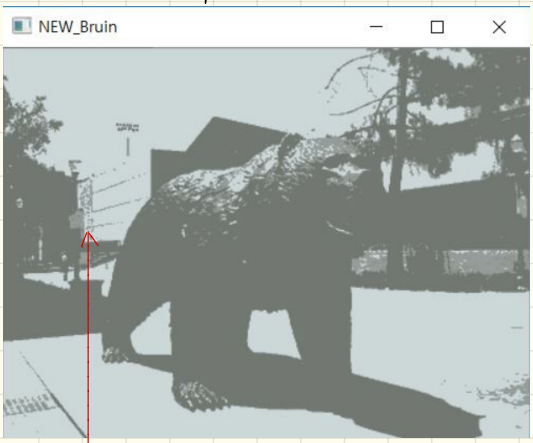
When K=16, $300 \cdot 400 \log_2 16 + 24 \cdot 16$
$= 480384$ (bits)

original : K=4 : K=8 : K=16
$= 1 : 0.083 : 0.125 : 0.167$

(1.4) Size(K=4) ≈ Size(K=8)

(1.4) Size(K=8) ≈ Size(K=16)

When K=4



there is difference.

When K=8

when K=16