COEN 240 Machine Learning

Homework #2

Name: Alex Cherekdjian **Student ID:** 00001083236

Problem 1

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{kn} \left\| \vec{m}_k - \vec{x}_n \right\|_2^2$$

$$\frac{\partial J}{\partial \vec{m}_k} \left(\sum_{n=1}^{N} r_{kn} \left\| \vec{m}_k - \vec{x}_n \right\|_2^2 \right) = \vec{0}$$

$$\frac{\partial J}{\partial \vec{m}_k} \sum_{n=1}^{N} r_{kn} \left(\vec{m}_k^T - \vec{x}_n^T \right) \left(\vec{m}_k - \vec{x}_n \right)$$

$$\frac{\partial J}{\partial \vec{m}_k} \sum_{n=1}^{N} r_{kn} \left(\vec{m}_k^T \vec{m}_k - \vec{x}_n^T \vec{m}_k - \vec{m}_k^T \vec{x}_n + \underbrace{\vec{x}_n^T \vec{x}_n}_{0} \right)$$

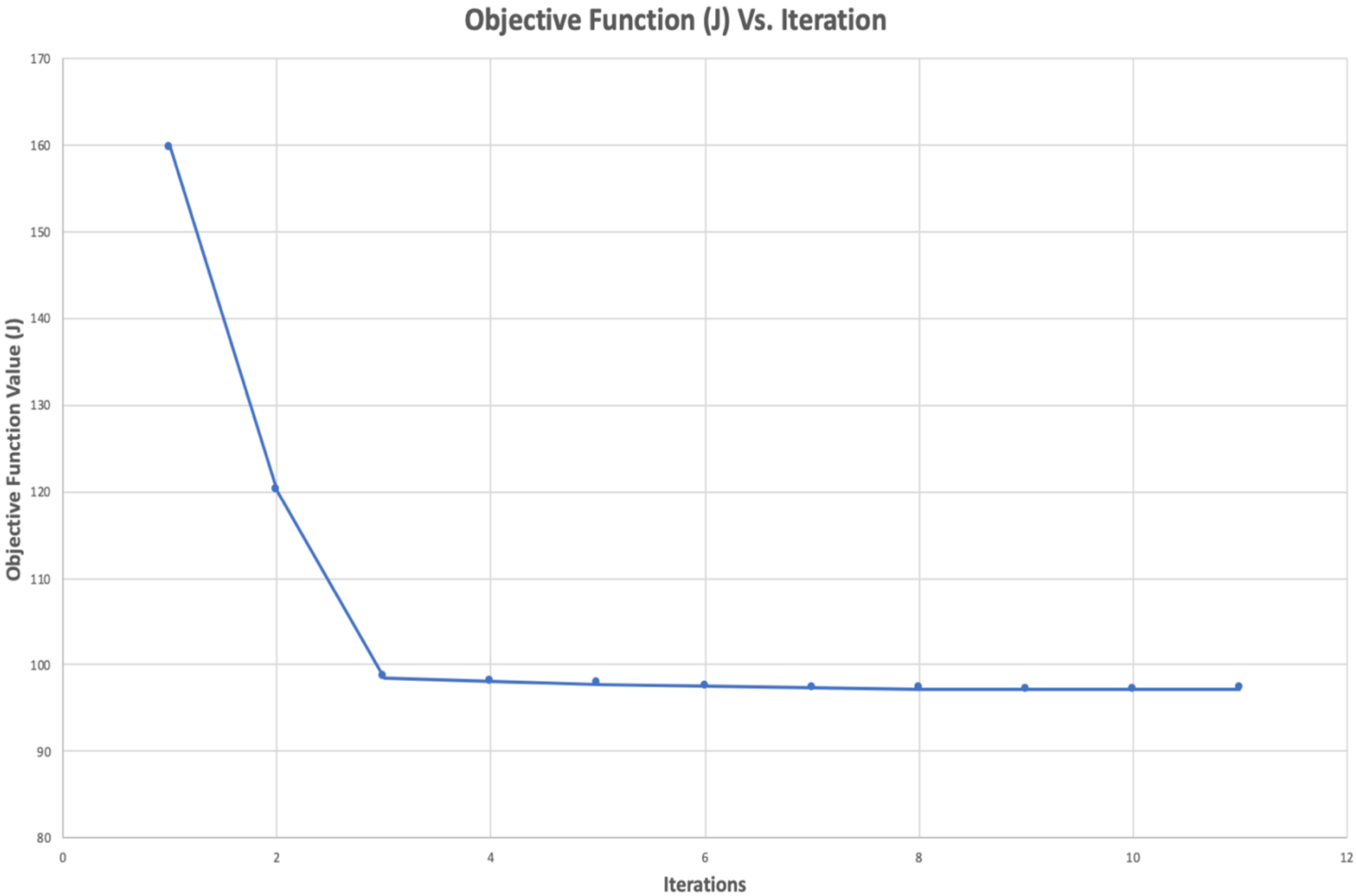
$$\sum_{n=1}^{N} r_{kn} \left(2 \cdot \cancel{I \cdot} \vec{m}_k - \vec{x}_n - \vec{x}_n \right) = \vec{0}$$

$$\sum_{n=1}^{N} r_{kn} \left(\vec{m}_k - \vec{x}_n \right) = \vec{0}$$

$$\left(\sum_{n=1}^{N} r_{kn} \right) \vec{m}_k = \sum_{n=1}^{N} r_{kn} \vec{x}_n$$

$$\boxed{\hat{\vec{m}}_k = \frac{\sum_{n=1}^{N} r_{kn} \vec{x}_n}{\sum_{n=1}^{N} r_{kn}}}$$

Problem 2



Comment: The graph shows the converging nature of a k-means clustering problem. It indicates that as the iterations increase, the objective function value J is decreasing.

Results: J values - 159.6810, 120.1270, 98.4979, 98.0508, 97.796, 97.4666, 97.2788, 97.2058, 97.1250, 97.0687, 97.2248

Best Prediction Accuracy: 133/150 = 88.67%

Problem 3

a) Math expression: $P(C1|X)$ where y is the output and $a = \vec{w}^T \vec{x}_n$

$$\downarrow$$

$$\boxed{y = \sigma(a)}$$

Criterion:

If $y > 0.5$, classify as $C1$, otherwise $C2$

b) For binary classification, $\boxed{D+1}$ parameters (weights) need to be calculated.

Problem 4

a) Math expression: $$P(C_k|X) = \frac{P(C_k)\cdot P(\vec{x}|C_k)}{\sum_{j=1}^{K} P(C_j)\cdot P(\vec{x}|C_j)}$$

$$\boxed{P(C_k|X) = y = \frac{\exp\{a_k\}}{\sum_{j=1}^{K} \exp\{a_j\}}}$$

where y is the output and $a_k = \vec{w}_k^T \vec{x} + \underbrace{\vec{w}_{k0}}_{\text{bias}}$

Criterion: If $P(C_k|X) \simeq 1$, then the max all probabilities yields which group, C_k, X belongs to the most.

b) For multi-classification problems, $\boxed{K}$ parameters (weights) need to be calculated for $\vec{w}_k$ where $k = 1 \ldots K$

Attachment

Problem 2 Code (in zip file):

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

import pandas as pd
import numpy as np
import math

# reading in the csv and outcomes
X_n = pd.read_csv("Iris.csv")
outcomes = X_n['outcome(Cluster Index)']

# cleaned up iris data
X_n = X_n.drop('outcome(Cluster Index)', axis=1)
X_n = X_n.drop('Sample Index', axis=1)

# constants
N = 150 # sample number
K = 3 # cluster number
M = 4 # attribute number

# getting column names
X_n_columns =  X_n.columns

# random initialization of cluster centers
m_k = X_n.sample(n=3)
r_kn = []

# initialize r_kn to all zeros
for i in range (0, N):
    r_kn.append([0,0,0])


J = 0.0
prev_J = 1000000.0
J_values = []
eta = 0.00001


while (1):

    iteration = 0
    # assignment step
    for sample in X_n.itertuples():
```

```
        distance = []

        # for each cluster midpoint, calc distance and store in distance list
        for m_k_row in m_k.itertuples():
            distance.append(math.sqrt((m_k_row._1 - sample._1)**2 +
(m_k_row._2 - sample._2)**2 + \
                        (m_k_row._3 - sample._3)**2 + (m_k_row._4 -
sample._4)**2))

        # get min value index
        index = distance.index(min(distance))

        # assign J value
        J += min(distance)

        # assign that r_kn value to a 1
        r_kn[iteration] = [0,0,0]
        r_kn[iteration][index] = 1
        iteration +=1

    J_values.append(J)

    if ((prev_J - J) < eta):
        break
    else:
        prev_J = J


    summation_1 = [0,0,0,0]
    summation_2 = [0,0,0,0]
    summation_3 = [0,0,0,0]
    r_totals = [0,0,0]

    # cluster-center update step
    for i in range(0, len(r_kn)):
        # if a sample is in that cluster group, add those values from each
attribute to summation, inc total r
        if r_kn[i][0] == 1:
            for j, x_column in zip(range(0, M), X_n_columns):
                summation_1[j] += X_n.at[i, x_column]
            r_totals[0] += 1

        elif r_kn[i][1] == 1:
            for j, x_column in zip(range(0, M), X_n_columns):
                summation_2[j] += X_n.at[i, x_column]
```

```
            r_totals[1] += 1

        elif r_kn[i][2] == 1:
            for j, x_column in zip(range(0, M), X_n_columns):
                summation_3[j] += X_n.at[i, x_column]
            r_totals[2] += 1

    # getting m_k indexes and columns
    m_k_indexes = m_k.head()
    m_k_columns =  m_k.columns

    # assigning new midpoints
    for i in range(0, M):
        m_k.at[m_k_indexes.index[0], m_k_columns[i]] =
summation_1[i]/r_totals[0]
        m_k.at[m_k_indexes.index[1], m_k_columns[i]] =
summation_2[i]/r_totals[1]
        m_k.at[m_k_indexes.index[2], m_k_columns[i]] =
summation_3[i]/r_totals[2]

    J = 0


print(J_values)
```