CSE-6363 Assignment 3

Swetha Vijaya Raghavan

1001551229

K-means clustering algorithm performed on Iris data.

First data preprocessing, of reading the data from iris.txt and then extracting the class label column from the whole data and X_train has the all the features excluding class label column.

```
# Fetching Iris Data
data = pand.read_csv('iris.txt')

# Extracting Class label from the data
col = data.columns[data.columns.str.startswith('I')]

# Seperating classlabel and data
classlabel = data[col]
features = data.drop(data.columns[4], axis=1)

X_train = features.as_matrix()
```

I have used k= 3 because we can evaluate, as there are 3 labels (0 Iris-setosa, 1 Iris-versicolor, 2 Iris-virginica) in the dataset. This can be done intuitively. Also, the data is uniformly distributed over all the clusters. A good clustering with smaller k such has 3 can have a lower Sum of Squared Error (SSE) than a poor clustering with higher k.

The below function will calculate the initial centroids to start our algorithm.

```
# Fuction for Calculating Initial Centriods
def initialCentriod(k=3):
    np.random.seed(0)
    cen = np.random.random((k,4))
    return cen

initialcen =initialCentriod()*5
```

Function to calculate the Euclidean distance from centroid to each data point so as to form a neighborhood which easily identifies which data point belongs to which cluster.

```
# function to calculate the distance between centriod and data points
from math import sqrt
def pairwisedistance(datapoint,cent):
    return sqrt(np.sum((datapoint-cent)*(datapoint-cent)))
```

KmeansClustering is the function which performs the kmeans clustering algorithm (mentioned below) using 2 parameters the training data and number of iterations.

Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change

The Complexity of the above algorithm is O(n * K * I * d) where n=number of points(), K= number of clusters, I = number of iterations, d = number of attributes.

```
# Function for calculating Kmeans along with training data and iterations
def kmeansClustering(X_train,iterations):
    size = X_train.shape[0]
    centriod = initialCentriod(3)*5
    kvalue = centriod.shape[0]
    distance = np.zeros([size, kvalue])
    classAssign = np.zeros([size, ])
    cenn = centriod
    temp = np.zeros([1, 4])
    count = 0
    for t in range(iterations):
        # print(centriod)
        for r in range(0, size):
            for c in range(0, kvalue):
                distance[r][c] = pairwisedistance(X_train[r], centriod[c])
        classAssign = (np.argmin(distance, axis=1)).reshape((-1,))
        cenn = np.concatenate((cenn, centriod))
        print("\nThe Centriod values for each iteration:\n")
        print(centriod)
        for c in range(0, kvalue):
            temp = np.zeros([1, 4])
            count = 0
            for r in range(0, size):
                temp = temp + (0.98) * (classAssign[r] == c) * X_train[r] + (0.02) * (classAssign[r] != c) * X_train[r]
                count = count + (0.98) * (classAssign[r] == c) + (0.02) * (classAssign[r] != c)
            centriod[c] = (temp.reshape((-1,))) / (count)
    return centriod, classAssign, cenn
```

The kmeansClustering results in centroids computed in each iterations along with the assignments of the data points to each cluster.

```
centroid ,classAssign,cenn = kmeansClustering(X_train,iterations=10)
print("\nKmeans Result Assignments:\n")
print(classAssign)
```

10 Iterations centroids are shown below:

```
The Centriod values for each iteration:

[[2.74406752 3.57594683 3.01381688 2.72441591]
 [2.118274   3.22947057 2.18793606 4.458865  ]
 [4.8183138  1.91720759 3.95862519 2.6444746 ]]

The Centriod values for each iteration:

[[5.00089837 3.43009545 1.64985963 0.33262212]
 [5.84832215 3.05100671 3.77449664 1.20536913]
 [6.11455283 2.93191039 4.44198271 1.47955548]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.57892116 2.64381743 3.82473029 1.17278008]
 [6.44176727 2.95633188 5.15407141 1.79573594]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.60516048 2.67249843 3.96677155 1.23071114]
 [6.52430892 2.97754346 5.23522371 1.83767455]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.6665918  2.67546322 4.02425604 1.24626614]
 [6.54452819 2.99360868 5.27778113 1.86445583]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.69651948 2.70722078 4.06732468 1.26275325]
 [6.56621494 2.98878033 5.30854081 1.88250867]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.71849975 2.71434364 4.07541814 1.26609225]
 [6.565696   2.988608   5.322496   1.88992   ]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.71849975 2.71434364 4.07541814 1.26609225]
 [6.565696   2.988608   5.322496   1.88992   ]]

The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.71849975 2.71434364 4.07541814 1.26609225]
 [6.565696   2.988608   5.322496   1.88992   ]]
The Centriod values for each iteration:

[[5.05437825 3.39456218 1.60287885 0.30211915]
 [5.71849975 2.71434364 4.07541814 1.26609225]
 [6.565696   2.988608   5.322496   1.88992   ]]
```

The result of our algorithm is:

```
Kmeans Result Assignments:

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 1 2 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1
 1 2 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2]

Process finished with exit code 0
```

References:

http://madhugnadig.com/articles/machine-learning/2017/03/04/implementing-k-means-clustering-from-scratch-in-python.html

http://benalexkeen.com/k-means-clustering-in-python/

https://www.kaggle.com/andyxie/k-means-clustering-implementation-in-python

GitHub & Wikipedia