K-Means Group Assignment

29th November, 2018

ONG JIA AUN WEA150022

For each group to produce a report detailing the followings:

* Detail the steps to perform K-Means
* Initialize number of cluster, k
* Initialize random vector of means
* Classify each input data according to mean
* Update cluster(centroid) until no change
* Repeat step 2 and 3 until convergence had been reach
* Detail the explanation on the cost function for K-Means
* The cost function of K-Means is distortion function.
* K-Means attempts to minimize distortion defined by the sum of the squared distances between each observation and its closest centroid.
* Explain on different methods on choosing K for K-means
* Elbow method – Increment k and plot graph. Find the sharp decline of increase of performance as the best number of k
* X-means – Refines cluster by repeatedly attempt subdivision. Keep the best resulting split.
* Information-Theoretic – Apply rate distortion theory as “jump” method
* Provide sample code and dataset presenting the usage of K-Means
  + Have to be other than the one provided in the lab

from copy import deepcopy

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

from matplotlib import pyplot as plt

Generate Random data

# Set three centers, the model should predict similar results

center\_1 = np.array([1,1])

center\_2 = np.array([5,5])

center\_3 = np.array([8,1])

# Generate random data and center it to the three centers

data\_1 = np.random.randn(200, 2) + center\_1

data\_2 = np.random.randn(200,2) + center\_2

data\_3 = np.random.randn(200,2) + center\_3

data = np.concatenate((data\_1, data\_2, data\_3), axis = 0)

plt.scatter(data[:,0], data[:,1], s=7)

# Number of clusters

k = 3

# Number of training data

n = data.shape[0]

# Number of features in the data

c = data.shape[1]

# Generate random centers, here we use sigma and mean to ensure it rep

resent the whole data

mean = np.mean(data, axis = 0)

std = np.std(data, axis = 0)

centers = np.random.randn(k,c)\*std + mean

# Plot the data and the centers generated as random

plt.scatter(data[:,0], data[:,1], s=7)

plt.scatter(centers[:,0], centers[:,1], marker='\*', c='g', s=150)

centers\_old = np.zeros(centers.shape) # to store old centers

centers\_new = deepcopy(centers) # Store new centers

data.shape

clusters = np.zeros(n)

distances = np.zeros((n,k))

error = np.linalg.norm(centers\_new - centers\_old)

# When, after an update, the estimate of that center stays the same, e xit loop

while error != 0:

# Measure the distance to every center

for i in range(k):

distances[:,i] = np.linalg.norm(data - centers[i], axis=1)

# Assign all training data to closest center

clusters = np.argmin(distances, axis = 1)

centers\_old = deepcopy(centers\_new)

# Calculate mean for every cluster and update the center

for i in range(k):

centers\_new[i] = np.mean(data[clusters == i], axis=0)

error = np.linalg.norm(centers\_new - centers\_old)

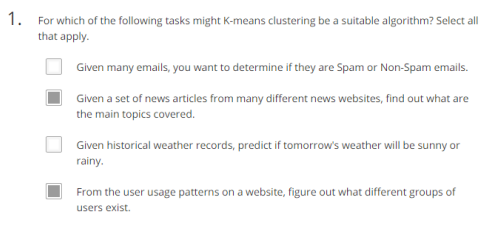
centers\_new

# Plot the data and the centers generated as random

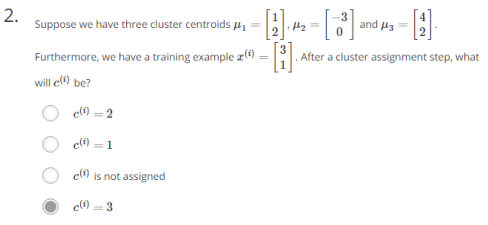
plt.scatter(data[:,0], data[:,1], s=7)

plt.scatter(centers\_new[:,0], centers\_new[:,1], marker='\*', c='g', s=150)

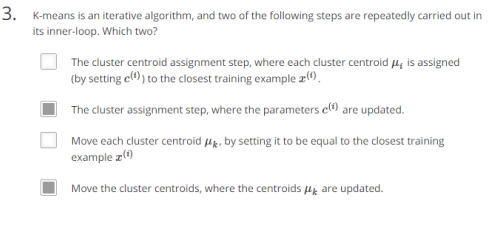
* Provide explanation for the answers to the quiz



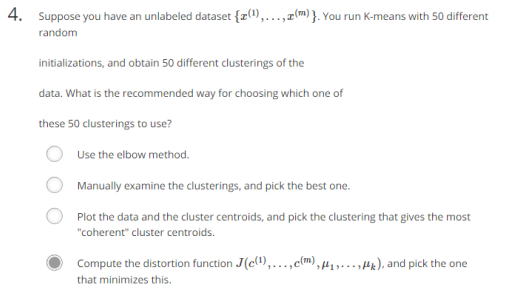
* It is easier to use supervised learning for spam email and weather prediction since it can be labelled easily



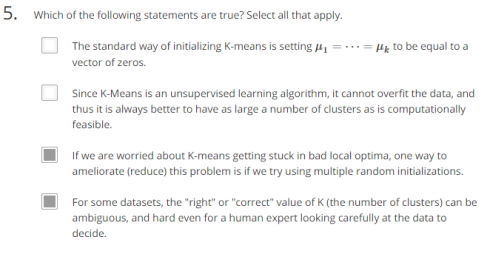
* For u1, the value is sqrt((3-1)^2 + (1-2)^2) = sqrt(5)
* For u2, the value is sqrt((3+3)^2 + (1-0)^2) = sqrt(37)
* For u3, the value is sqrt((3-4)^2 + (1-2)^2) = sqrt(2)
* Since value of u3 is lowest, hence x(i) is closest to u3



* Move the cluster centroids, where the centroids, μk are updated is the second step
* The cluster assignment step, where the parameters c(i) are updated is the first step
* Move uk by equating to x(i) is wrong, move by slight increase
* Cluster centroid assignment is outside of inner-loop



* Elbow method is not suitable, all the init is with the same k
* Too many init to manually choose
* The distortion cost function is the one we aim to minimize



* The standard way of init is random init
* Having a big k value will result in confusion as each k only represent a small range of data
* Yes
* Yes
* Detail other variants of K-means to overcome the weaknesses of K-means
* Arithmetic mean is not robust to outliers. Outlier may pull the centroid away from the optimal location. To overcome this, we can use median instead of mean.