Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining | Assignment 4

Due: March 1, 2023(100 points)

Yang Yao
Stervt
yao.yan@northeastern.edu

Question 1

1a.) K-Means algorithm has 2 main steps, assign a closest centroid for each sample and recalculate centroids for each cluster. So the implementation is shown below, the function receives 4 arguments, data, function of distance, initial states of centroids and max iteration.

```
def K_means(data,distance,inital_state,max_iteration=100):
  centers = deepcopy(inital_state)
  for iteration in range(max_iteration):
      new_centers = np.zeros_like(centers)
      centers_count = np.zeros(centers.shape[0])
      for i in range(data.shape[0]):
        # assign a center for each sample
        min_dist = np.inf
        c = None
        for j in range(centers.shape[0]):
          dist = distance(data[i],centers[j])
          if dist < min_dist:
            c = j
            min_dist = dist
        new_centers[c]+=data[i]
        centers_count[c]+=1
      # update centers
      for k in range(new_centers.shape[0]):
        new_centers[k] = new_centers[k]/centers_count[k]
      if (new_centers == centers).all():
        break
      centers = new_centers
  return centers
```

1b.) In this question, I use Euclidean distance as distance. The result is shown below.

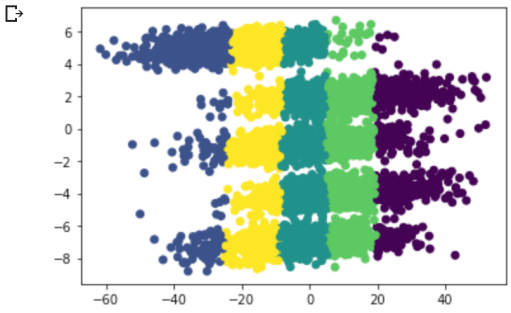
```
[ ] import matplotlib.pyplot as plt
```

```
[ ] centers = K_means(data, lambda x,y: np.linalg.norm(x - y),inital_centers)
```

```
[ ] labels = np.zeros(data.shape[0])
```

```
[ ] for i in range(data.shape[0]):
        # cluster samples
        min_dist = np.inf
        c = None
        for j in range(centers.shape[0]):
          dist = np.linalg.norm(data[i] - centers[j])
          if dist < min_dist:
            c = j
            min_dist = dist
        labels[i] = c
```

```
plt.scatter(x = data[:,0],y = data[:,1], c = labels)
plt.show()
```



1c.) The number of cluster be set to 5 is logical, because the dataset contains trucks of 5 different years. It is reasonable to believe data of trucks produced in different years has slightly difference and the data of trucks produced in the same year have some common patterns. So we may guest the data can be sperate to 5 different clusters.

After plotting my result, I find the cluster result is not very ideal. It split the data vertically which doesn't match the pattern shown in the graph.

I think there is no an initialization that can make it cluster well. Because we can see the "right" cluster is spindle shaped, the Euclidean distance of samples on the both corners of the spindle will be large so that K-Means will not consider them belong to the same cluster.

Question 2

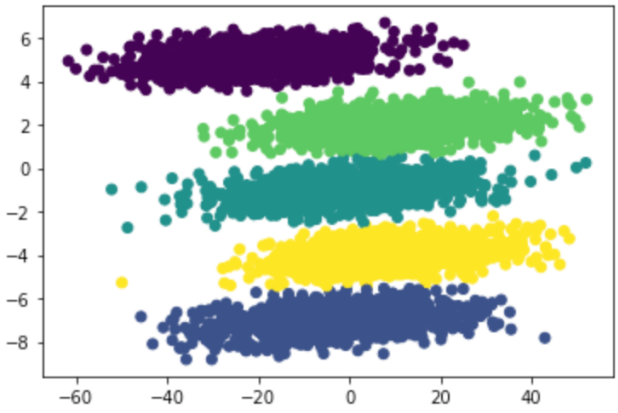
2a.) Because I have implement K-Means algorithm, I just need to pass a function to calculate the Mahalanobis Distance between 2 points. The result is shown below. I find that this distance works well, it can well measure the distance between a point and a distribution that all samples in a spindle shape area are clustered together.

```
[ ]  P = np.array([[10,0.5],[-10,0.25]]).astype('float64')
     R = np.linalg.inv((P.T.dot(P)))
     def Mahalanobis_Distance(x,y):
       return (x-y).T.dot(R).dot(x-y)
```

```
[ ]  centers = K_means(data, Mahalanobis_Distance,inital_centers)
```

```
     labels = np.zeros(data.shape[0])
     for i in range(data.shape[0]):
           # cluster samples
           min_dist = np.inf
           c = None
           for j in range(centers.shape[0]):
             dist = Mahalanobis_Distance(data[i],centers[j])
             if dist < min_dist:
               c = j
               min_dist = dist
           labels[i] = c
```

```
[ ]  plt.scatter(x = data[:,0],y = data[:,1], c = labels)
     plt.show()
```



2b.) I use PCA class in sklearn library to compute the principle components. The first components is a horizontal direction which match the our intuition that all samples are T mainly spread out along the horizontal direction.

```
[ ]  from sklearn.decomposition import PCA
```

```
[ ]  pca = PCA(n_components=2)
```

```
[ ]  pca.fit(data)
```

```
     PCA(n_components=2)
```

```
[ ]  pc1 = pca.components_
     print(pc1)
```

```
     [[-0.99838317  0.05684225]
      [-0.05684225 -0.99838317]]
```

2c.) Use the same method former question to get the principle components of 5 clusters. They are not the same as the aggregate data, and they are not the same as each other. However, all the principle components are similar, the first principle component is nearly horizontal direction and the second component is nearly vertical direction.

```
for c in range(5):
  cluster = data[labels==[c]]
  pca = PCA(n_components=2)
  pca.fit(cluster)
  print('cluster {}\'s principle components:'.format(c))
  print(pca.components_)
  print('-'*40)
```

```
cluster 0's principle components:
[[ 0.99993527  0.01137789]
 [ 0.01137789 -0.99993527]]
----------------------------------------
cluster 1's principle components:
[[ 0.99992533  0.01222027]
 [ 0.01222027 -0.99992533]]
----------------------------------------
cluster 2's principle components:
[[ 0.99990986  0.01342629]
 [ 0.01342629 -0.99990986]]
----------------------------------------
cluster 3's principle components:
[[ 0.99993306  0.01157047]
 [-0.01157047  0.99993306]]
----------------------------------------
cluster 4's principle components:
[[-0.99989374 -0.01457781]
 [-0.01457781  0.99989374]]
----------------------------------------
```

2d.) I don't figure out what's the relationship between P' and the data. It may be better to tell us the physical meaning of P or R to help us find out the relationship.

```
[ ]  v,w = np.linalg.eig(P.T)
```

```
[ ]  v = np.array([[v[0],0],[0,v[1]]])
```

```
[ ]  P_ = v*w
```

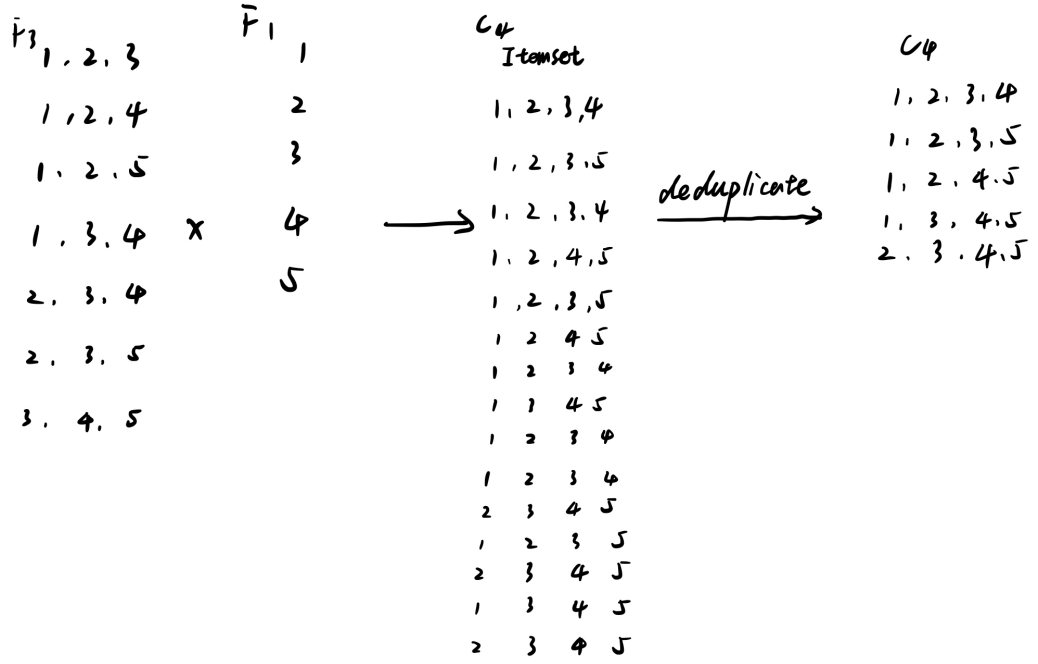
```
[ ]  P_
```

```
array([[9.44301625, 0.58344264],
       [0.51282107, 0.53717161]])
```

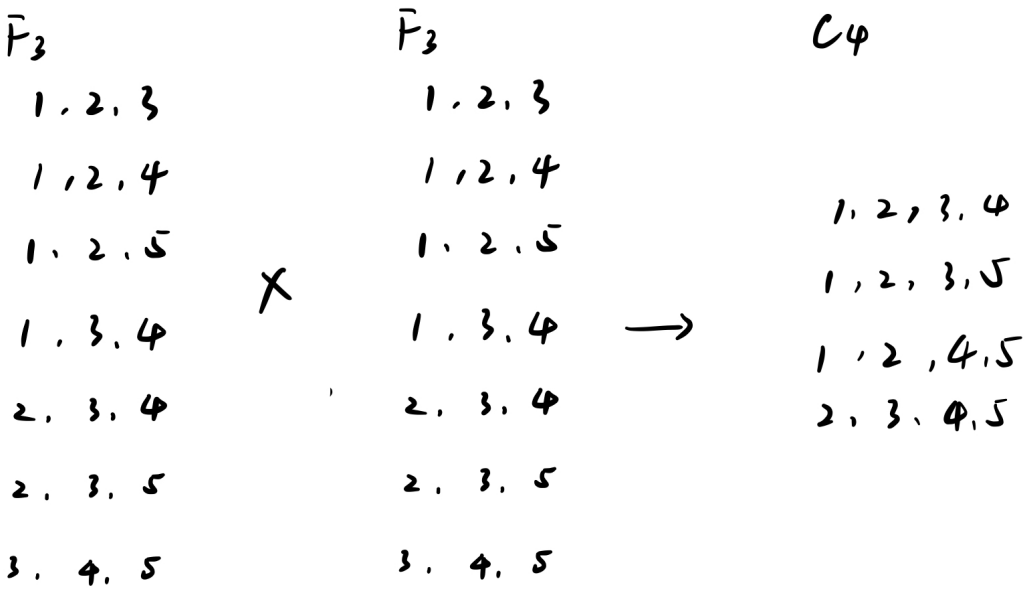
$P' = \Lambda\Phi = P.T\Phi$

Question 3

3a.)



3b.)



3c.)

C4:

✓ 1, 2, 3, 4

✗ 1, 2, 3, 5 (pruned, {1, 3, 5} is not in F_3)

✗ 1, 2, 4, 5 (pruned, {2, 4, 5} is not in F_3)

✗ 2, 3, 4, 5 (pruned, {2, 4, 5} is not in F_3)

pruning → 1, 2, 3, 4

Question 4

4a.) There are 7 unique items: {Beer, Diapers, Milk, Bread, Butter, Cookies, Eggs}, so all possible rules can generate are: $R = 3^7 - 2^8 + 1 = 1932$

4b.) $Confidence = \frac{\sigma(\{Milk,Diapers,Butter\})}{\sigma(\{Milk,Diapers\})} = \frac{2}{4} = 50\%$

4c.) $Support = \frac{\sigma(\{Milk,Diapers,Butter\})}{T} = \frac{2}{10} = 20\%$

4d.) True. Apriori Principle tells us if an itemset is frequent set, then all its subsets are frequent sets.

4e.) False. We can not know whether an itemset is a frequent set by some of its subsets are frequent set.

4f.) False. The support of {b} show greater or equal to 30.

4g.) False. The maximum number of possible frequent itemsets of size 2 equals $C_5^2 = \frac{5\times4}{2!} = 10$.

4h.)