Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining — Assignment 4

Due: March 1, 2023 (100 points)

Charlie Huang
https://github.com/Charolf
huang.cun@northeastern.edu

K-Means

The normalized automobile distributor timing speed and ignition coil gaps for production F-150 trucks over the years of 1996, 1999, 2006, 2015, and 2022. We have stripped out the labels for the five years of data.

Each sample in the dataset is two-dimensional, i.e. $\mathbf{x}_i \in \mathbb{R}^2$ (one dimension for timing speed and the other for coil gaps), and there are $N = 5000$ instances in the data.

Question 1 [20 pts total]

[10 pts] Question 1a.) Implement a simple k-means algorithm in Python on Colab with the following initialization:

$$\mathbf{x}_1 = \begin{pmatrix}10\\10\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}-10\\-10\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}2\\2\end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix}3\\3\end{pmatrix}, \mathbf{x}_5 = \begin{pmatrix}-3\\-3\end{pmatrix},$$

You need only 100 iterations, maximum, and your algorithm should run very quickly to get the results.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

```
df = pd.read_csv('./data/f150_motor_distributors.txt', header=None, names=['speed', 'gap'])
data = df.values
```

1a

```
clusters = [[], [], [], [], []]
# centroids = [X1, X2, X3, X4, X5]
centroids = np.array([[10.0, 10.0], [-10.0, -10.0], [2.0, 2.0], [3.0, 3.0], [-3.0, -3.0]])
k = len(clusters)
```

```
def calcClusters(data, centroids):
    # Init empty clusters
    clusters = [[] for _ in range(k)]
    for pt in data:
        dist = []
        for i in range(k):
            dist.append(np.linalg.norm(pt - centroids[i]))
        clusters[np.argmin(dist)].append(pt)
    return clusters

def calcCentroids(centroids, clusters):
    # Recalculates centroids
    for i in range(k):
        centroids[i] = np.average(clusters[i], axis=0)
    return centroids
```

```
for _ in range(100):
    clusters = calcClusters(data, centroids)
    centroids = calcCentroids(centroids, clusters)
```

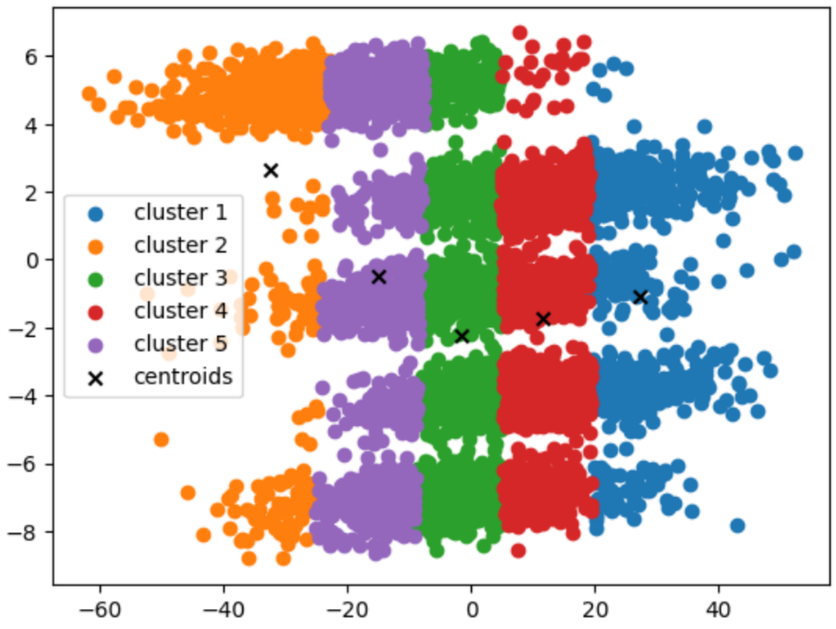
[5 pts] Question 1b.) Scatter the results in two dimensions with different clusters as different colors. You can use **matplotlib**'s **pyplot** functionality:

```
>> import matplotlib.pyplot as plt
>> plt.scatter(<YOUR CODE HERE>)
```

1b

```
for i in range(k):
    x, y = zip(*clusters[i])
    plt.scatter(x, y, label='cluster {}'.format(i+1))
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', color='black', label='centroids')
plt.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x1a5d3accb10>
```



[5 pts] Question 1c.) You will notice that in the above, there are only five initialization clusters. Why is $k = 5$ a logical choice for this dataset? After plotting your resulting clusters, what do you notice? Did it cluster very well? Is there an initialization that would make it cluster well?

Yes, $k = 5$ is logical choice for this dataset since by human eyes, there are indeed 5 clusters to be categorized. After plotting my resulting clusters, I notice that the program did not cluster the dataset well. I think there might be an initialization that can make it cluster well. For example, to normalize the data at the beginning.

Question 2)[30 pts total]

In the data from Question 1, let $\mathbf{x}$ and $\mathbf{y}$ be two instances, i.e., they are each truck with separate measurements. A common distance metric is the *Mahalanobis Distance* with a specialized matrix $P \in \mathbb{R}^{2\times 2}$ that is written as follows:

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T(P^TP)^{-1}(\mathbf{x}-\mathbf{y})$$

In scalar format (non-matrix format), the Mahalanobis Distance can be expressed as:

$$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{2}\sum_{j=1}^{2}(x_i - y_i)\cdot(P^TP)^{-1}_{i,j}\cdot(x_j - y_j)$$

where $\mathbf{x}$ and $\mathbf{y}$ are two instances of dimensionality 2, and $d(\mathbf{x},\mathbf{y})$ is the distance between them. In the case of the F150 engine components, P is a known relationship through Ford's quality control analysis each year, where it is numerically shown as below:

$$P = \begin{pmatrix} 10 & 0.5 \\ -10 & 0.25 \end{pmatrix}$$

[15 pts] Question 2a.) Using the same data as **Question 1** and the same initialization instances $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$ implement a specialized k-means with the above Mahalanobis Distance. Scatter the results with the different clusters as different colors.

What do you notice? You may want to pre-compute $(P^TP)^{-1}$ so that you aren't calculating an inverse every single loop of the the k-Means algorithm.

I noticed that this time the Mahalanibis Distance algorithm helped the k-means method to find the correct clusters.

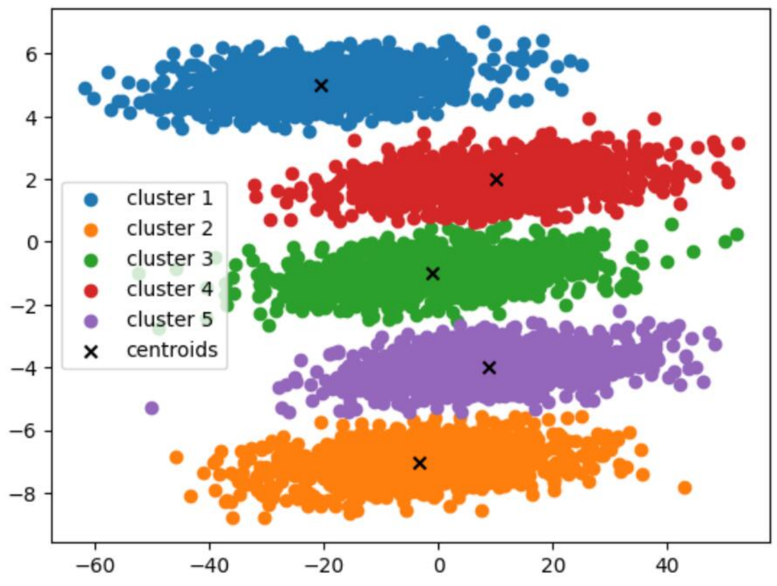
```
P = np.array([[10, 0.5], [-10, 0.25]])
PTPinv = np.linalg.inv(P.T @ P)

clusters = [[], [], [], [], []]
centroids = np.array([[10.0, 10.0], [-10.0, -10.0], [2.0, 2.0], [3.0, 3.0], [-3.0, -3.0]])
k = len(clusters)
```

```python
def calcClusters(data, centroids):
    clusters = [[] for _ in range(k)]
    for pt in data:
        dist = []
        for i in range(k):
            dist.append(mahalanobis(pt, centroids[i]))
        clusters[np.argmin(dist)].append(pt)
    return clusters

def calcCentroids(centroids, clusters):
    for i in range(k):
        centroids[i] = np.average(clusters[i], axis=0)
    return centroids

def mahalanobis(X, Y):
    dist = 0
    for i in range(len(P)):
        for j in range(len(P[0])):
            dist += (X[i] - Y[i]) * PTPinv[i,j] * (X[j] - Y[j])
    return dist
```

```python
for _ in range(100):
    clusters = calcClusters(data, centroids)
    centroids = calcCentroids(centroids, clusters)
```

```python
for i in range(k):
    x, y = zip(*clusters[i])
    plt.scatter(x, y, label='cluster {}'.format(i+1))
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', color='black', label='centroids')
plt.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x1a5d3e42390>
```



[5 pts] Question 2b.) Calculate and print out the principle components of the aggregate data.

```
std_data = StandardScaler().fit_transform(data)
corr = df.corr()
eig_vals, eig_vecs = np.linalg.eig(corr)

pc1 = data.dot(eig_vecs[:, 0])
pc2 = data.dot(eig_vecs[:, 1])
```

I got the following for the two principal components:

```
[ -2.77918957 -14.14164517  -8.15782548 ...   9.21268179
 -27.39510673 -17.79984525]
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945
 -21.16443095 -20.11909325]
```

[5 pts] Question 2c.) Calculate and print out the principle components of *each cluster*. Are they the same as the aggregate data? Are they the same as each other?

```
for clust in clusters:
std_data = StandardScaler().fit_transform(clust)
corr = pd.DataFrame(clust).corr()
eig_vals, eig_vecs = np.linalg.eig(corr)

pc1 = data.dot(eig_vecs[:, 0])
pc2 = data.dot(eig_vecs[:, 1])
print(pc1, pc2)
```

I got the following. It seems like the principle components are the same as each other.

```
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945 -21.16443095
 -20.11909325] [ 2.77918957 14.14164517  8.15782548 ... -9.21268179
 27.39510673 17.79984525]
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945 -21.16443095
 -20.11909325] [ 2.77918957 14.14164517  8.15782548 ... -9.21268179
 27.39510673 17.79984525]
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945 -21.16443095
 -20.11909325] [ 2.77918957 14.14164517  8.15782548 ... -9.21268179
 27.39510673 17.79984525]
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945 -21.16443095
 -20.11909325] [ 2.77918957 14.14164517  8.15782548 ... -9.21268179
 27.39510673 17.79984525]
[-14.14894079 -23.98755475 -10.10794322 ...   9.06673945 -21.16443095
 -20.11909325] [ 2.77918957 14.14164517  8.15782548 ... -9.21268179
 27.39510673 17.79984525]
```

[5 pts] Question 2d.) Take the eigenvector / eigenvalue decomposition of P^T and subsequently, take their product. That is to say,

$$\{\Lambda, \Phi\} = \texttt{eig}\left(P^T\right)$$

where $\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ and Φ is a 2×2 matrix with $\phi_i \in \mathbb{R}^2$, a column in Φ. Calculate a new P' such that

$$P' = \Lambda\Phi$$

What is the relationship between P' and the data?

By following the what the question asks for, the code is below:

```
PT = P.T
eig_vals, eig_vecs = np.linalg.eig(PT)
Pprime = np.diag(eig_vals) @ eig_vecs
```

I got P' to be the following, and the eigenvectors are also similar to those of the PCA components of the clusters.

```
array([[9.44301625, 6.95724558], [0.04300577, 0.53717161]])
```

Market Basket Analysis and Algorithms

Consider F_3 as the following set of frequent 3-itemsets:

```
{1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4},
{2, 3, 4}, {2, 3, 5}, {3, 4, 5}.
```

Assume that there are only five items in the data set.

Question 3 [25 pts total]

[10 pts] Question 3a.) List all candidate 4-itemsets obtained by a candidate generation procedure using the $F_{k-1} \times F_1$ merging strategy.

From the dataset, we know F_1 contains all 1-itemsets {1}, {2}, {3}, {4}, and {5}, the problem also has given us all 3-itemsets {1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {2, 3, 4}, {2, 3, 5}, and {3, 4, 5}. To generate all candidate 4-itemsets, we just combine all 3-itemsets with the 1-itemsets:

- {1, 2, 3, 4}
- {1, 2, 3, 5}
- {1, 2, 4, 5}
- {1, 3, 4, 5}
- {2, 3, 4, 5}

We can't have something like {1, 2, 4, 4} because it contains duplicates.

[10 pts] Question 3b.) List all candidate 4-itemsets obtained by the candidate generation procedure in A Priori, using $F_{k-1} \times F_{k-1}$.

Since we already have 3-itemsets {1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {2, 3, 4}, {2, 3, 5}, and {3, 4, 5}, to generate all candidate 4-itemsets, we just combine them in pairs and not to consider the ones that have duplicates.

- {1, 2, 3, 4}
- {1, 2, 3, 5}
- {1, 2, 4, 5}
- {1, 3, 4, 5}
- {2, 3, 4, 5}

Which we can see, the candidate 4-itemsets are identical to the results of $F_{k-1} \times F_1$ above.

[5 pts] Question 3c.) List all candidate 4-itemsets that survive the candidate pruning step of the Apriori algorithm.

To prune, we just need to remove those candidates where the subsets aren't in F_3. Remove the following:

- {1, 2, 3, 5}: contains {1, 3, 5} that's not in F_3
- {1, 2, 4, 5}: contains {1, 4, 5} that's not in F_3
- {1, 3, 4, 5}: contains {1, 3, 5} that's not in F_3
- {2, 3, 4, 5}: contains {2, 4, 5} that's not in F_3

After removing the above candidates, only {1, 2, 3, 4} survives. This is the only 4-itemset that survives pruning.

Question 4 [25 pts total]

Consider the following table for questions 4a) to 4c):

Transaction ID	Items
1	**{Beer, Diapers}**
2	**{Milk, Diapers, Bread, Butter}**
3	**{Milk, Diapers, Cookies}**
4	**{Bread, Butter, Cookies}**
5	**{Milk, Beer, Diapers, Eggs}**
6	**{Beer, Cookies, Diapers}**
7	**{Milk, Diapers, Bread, Butter}**
8	**{Bread, Butter, Diapers}**
9	**{Bread, Butter, Milk}**
10	**{Beer, Butter, Cookies}**

[3 pts] Question 4a.) What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?

To calculate the maximum number of association rules from the table, we need to find the number of association rules for Support and Confidence separately.

To find the association rules for Support, we have 6 unique items in the table. From here, the question basically asks to calculate the number of subsets with at least 2 items in a set of n items. The reason we need at least 2 items is that to form an association, we need to have at least a 1-1 linkage, which makes it 2.

There is a formula to calculate the above scenario:

$$S = 2^n - n - 1,$$

where S is the number of association rules for Support, and n is the unique items in the table.

We plug in $n = 6$, get:

$$S = 2^6 - 6 - 1 = 57.$$

Therefore, the maximum number of association rules for Support is 57. As for Confidence, the number of its association rules becomes very simple: $C = 6 \times 6 = 36$.

In conclusion, the total number of the association rules of this problem is $57 + 36 = 93$.

[3 pts] Question 4b.) What is the confidence of the rule {Milk, Diapers} ⇒ {Butter}?

$$C = \frac{\sigma(\{Milk, Diapers, Butter\})}{\sigma(\{Milk, Diapers\})} = \frac{2}{4} = 0.5.$$

[3 pts] Question 4c.) What is the support for the rule {Milk, Diapers} ⇒ {Butter}?

$$S = \frac{\sigma(\{Milk, Diapers, Butter\})}{|T|} = \frac{2}{10} = 0.2.$$

[3 pts] Question 4d.) `True` or `False` with an explanation: Given that {a,b,c,d} is a frequent itemset, {a,b} is always a frequent itemset.

True. Since {a,b,c,d} is a frequent itemset and {a,b} is a subset of it, that means {a,b} must also appear in the dataset with a frequency at least as much as {a,b,c,d}. Therefore, {a,b} is also a frequent itemset.

[3 pts] Question 4e.) `True` or `False` with an explanation: Given that {a,b}, {b,c} and {a,c} are frequent itemsets, {a,b,c} is always frequent.

True. Consider {a,b,c}, it contains {a,b}, {b,c}, and {a,c} as its subsets. Since each of these subsets is frequent, it follows that {a,b,c} must also appear in the dataset. Therefore, {a,b,c} is also a frequent itemset.

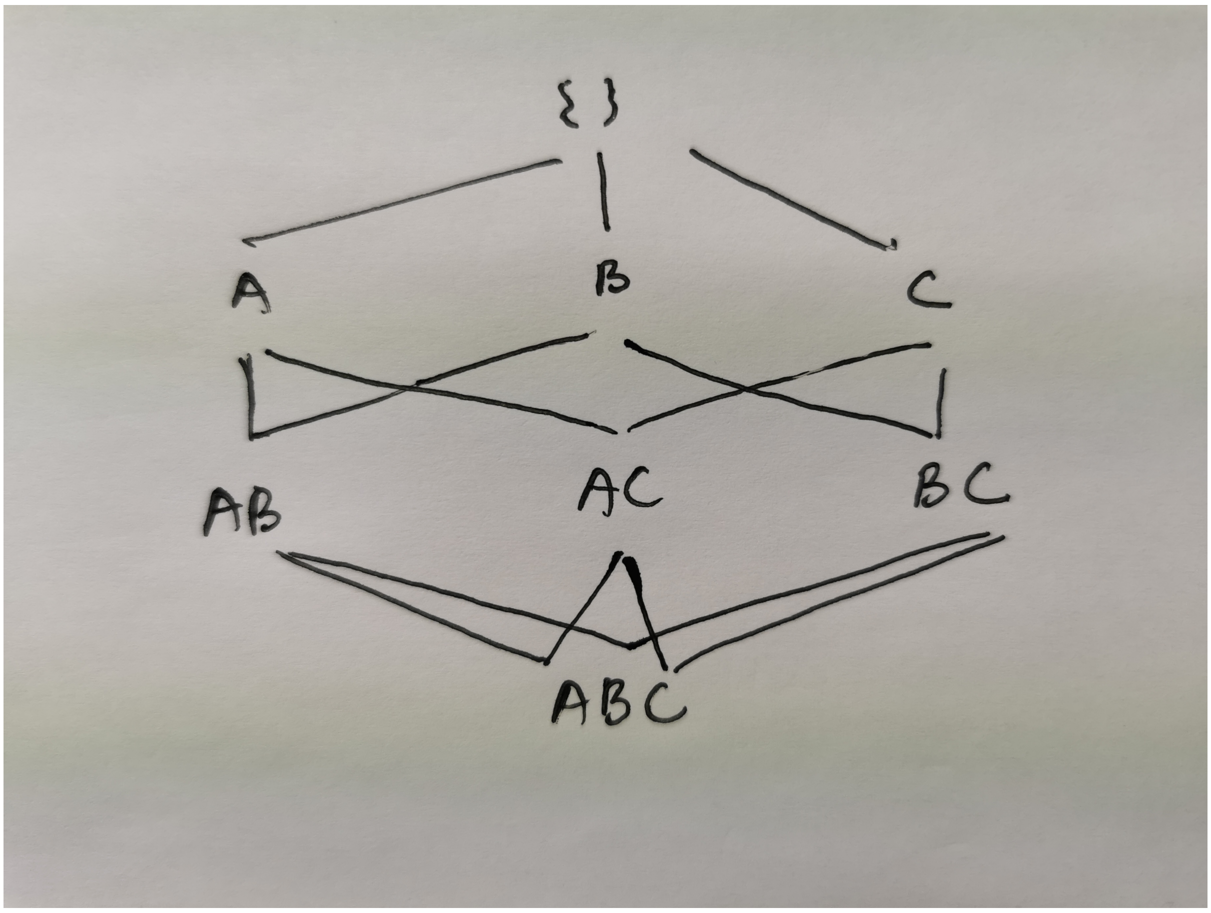
[3 pts] Question 4f.) `True` or `False` with an explanation: Given that the support of {a,b} is 20 and the support of {b,c} is 30, the support of {b} is larger than 20 but smaller than 30.

True. By definition, the Support of a subset is in between of the minimum and the maximum of its parentsets Support.

[3 pts] Question 4g.) `True` or `False` with an explanation: In a dataset that has 5 items, the maximum number of size-2 frequent itemsets that can be extracted (assuming minsup > 0) is 20.

False. In a dataset that has 5 items, the total number of possible size-2 itemsets is given by the binomial coefficient $C(5, 2) = 10$. This represents the maximum number of ways to choose 2 items out of 5. Since this number is less tha 20, the question's statement is False.

[4 pts] Question 4h.) Draw the itemset lattice for the set of unique items $\mathcal{I} = \{a, b, c\}$.


{ }
A
B
C
AB
AC
BC
ABC