HW4

December 2, 2022

1 Homework 4: Due Sunday 11/27 11:30pm

In this assignment, we are working on a list of 1200 bitstrings, where each of them contains 16 bits. We will apply Agglomerative Clustering, K-means Clustering, and PCA to this dataset.

1.1 Background and Data Information

For a bitstring S in this dataset, we describe $S = \{s_{15}, s_{14}, s_{13}, s_{12}, \dots, s_0\}$, where s_{15} is often known as the most significant bit (MSB) and s_0 as the least significant bit (LSB).

There are duplicated bitstrings in this dataset, but they will not affect this assignment. Don't worry about them.

1.2 Equivalence Relation

This is an important concept to Exercise 1.

Let's say if we have two bitstrings, $A = \{a_{15}, a_{14}, a_{13}, \dots, a_0\}$ and $B = \{b_{15}, b_{14}, b_{13}, \dots, b_0\}$.

We can flip one bit a_i in A to get another bitstring A', such that the difference of A and A' is only one bit. We define the above transformation to be $A \to A'$.

We call two bitstrings A and B to be **equivalent** ($A \sim B$) if there exists a sequence $A \to C_1 \to C_2 \to \dots \to C_n \to B$, where $\forall i, C_i$ belongs to the dataset.

It can be seen that equivalence is both **commutative** ($A \sim B \iff B \sim A$) as well as **transitive** ($A \sim B, B \sim C \implies A \sim C$).

We can say that the elements in the above sequence $\{A, C_1, \dots, C_n, B\}$ form an equivalence class. Given a new bitstring X, we can see that if $X \sim C_i$, $1 \le i \le n$, then X will be added to the above equivalence class, and by the transitive property of equivalence relations, $X \sim A$, and $X \sim B$.

1.2.1 Example

Let's say we have 4 bitstrings, each of them is 4 bits long. They are $0000, 0010, 0110, 1100$, respectively.

We can say $0000 \sim 0110$ because $0000 \to 0010 \to 0110$.

However, $0000 \nsim 1100$. There may be sequences like $0000 \to 1000 \to 1100$ or $0000 \to 0100 \to 1100$, but neither 1000 nor 0100 is in our dataset.

Ultimately, $\{0000, 0010, 0110\}$ form an equivalence class, whereas $\{1100\}$ is the other. As a result, there are two classes.

1.2.2 Libraries that can be used: numpy, scipy, pandas, scikit-learn, matplotlib, seaborn

Any libraries used in the discussion materials are also allowed.

2 Exercises

2.1 Exercise 1 - Agglomerative Clustering (40 points in total)

Using agglomerative clustering with a distance threshold for early stopping, we can calculate the number of equivalence classes by counting the number of clusters. In order to perform agglomerative clustering, we have to consider what parameters may be used:

2.1.1 Exercise 1.1 - Choosing Parameters (20 points)

- Explain why you would pick these parameters.
  - Which linkage rule should be used? (single-linkage, complete-linkage, or average-linkage)
  - Which distance function should be used? (Euclidean distance, Manhattan distance, or cosine distance)
  - What should the threshold distance be?

Hints: - How the distance threshold works: Whenever two clusters are picked to consider merging them, the distance between those clusters is compared to the distance threshold. If the distance is smaller than the threshold, the clusters merge and the algorithm continues; Otherwise, they will not be merged. - How to choose a linkage rule: Think about how you would figure out which equivalence class the string 0001 belongs to in the previously given example.

We should be using the `Manhattan distance`, as we are dealing with discrete bits. If we imagine these 16 bit strings as vectors in 16 dimensions, adjustments for them to be equal need to be done in discrete steps, by incrementing or decrementing by 1 in a direction. There is no “diagonal” movement in our system.

The distance between `0101` and `1001` for example, should be 2, 1 for each bit flip. This means for 16 bits, the maximum distance between any 2 strings is 16. For 2 bit strings to be equivalent, (excluding other data) there can only be 1 flip maximum, therefore the `distance threshold` should be 1 (inclusive).

`single linkage` will be used, as we want to merge equivalent sets.

[ ]:
```python
import pandas as pd
import numpy as np
df = pd.read_csv('./bitstrings.csv')
```

2.1.2 Exercise 1.2 - Agglomerative Clustering for Equivalence Classes (20 points)

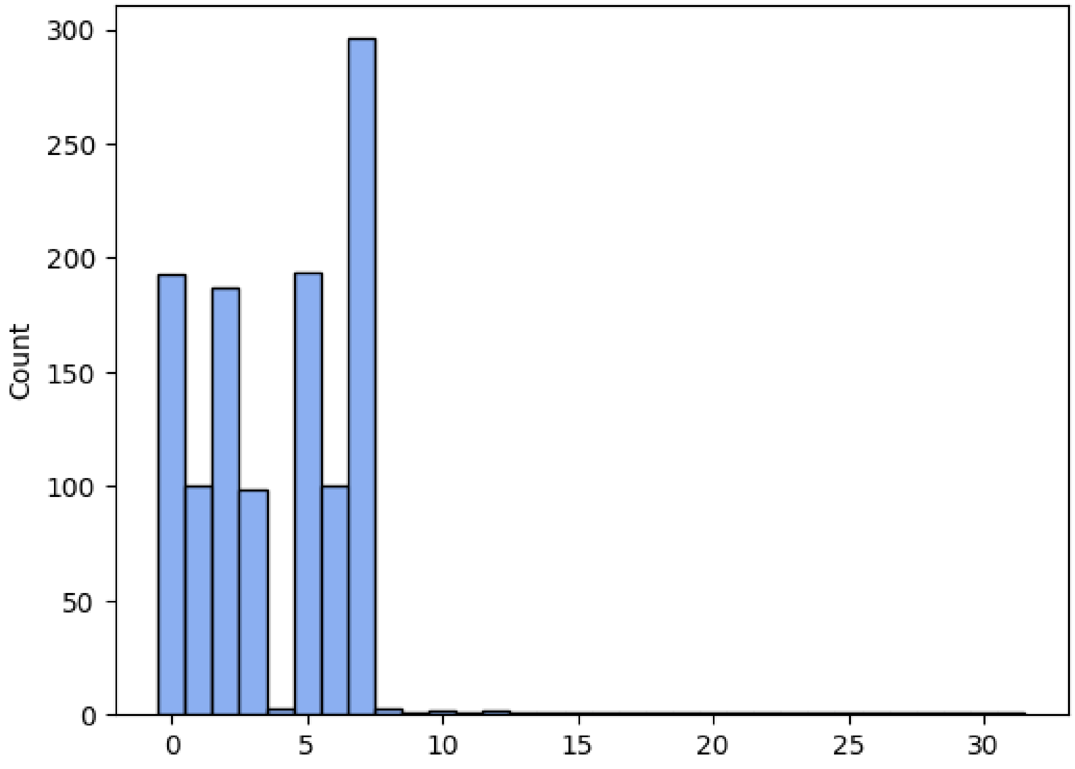
- Perform the agglomerative clustering with the parameters you picked in the above three questions.
- Show the frequency(number of members) of each cluster. You are encouraged to create a bar chart to show the distribution as it will help you in Exercise 2, but printing only the numbers is also fine.

Hints: - The value of `distance_threshold` in the arguments should be **slightly** higher than what you picked. This is because we only merge two clusters when their distance is **strictly smaller** than the threshold.

```
[ ]: from sklearn import cluster
     from seaborn import histplot

     # perform clustering
     clustering = cluster.AgglomerativeClustering(n_clusters=None,␣
      ↪affinity='manhattan', linkage='single', distance_threshold=1.1,␣
      ↪compute_full_tree=True).fit(df)

     # plot cluster distribution
     histplot(clustering.labels_, discrete=True, color='cornflowerblue'); #␣
      ↪semicolon removes jupyter output
```



```
[ ]: # 32 groups total, could also use range(max(clustering.labels_))
     print([sum(clustering.labels_ == i) for i in range(32)])
```

```
[193, 100, 187, 99, 3, 194, 100, 296, 3, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

2.2 Exercise 2 - K-Means Clustering (30 points in total)

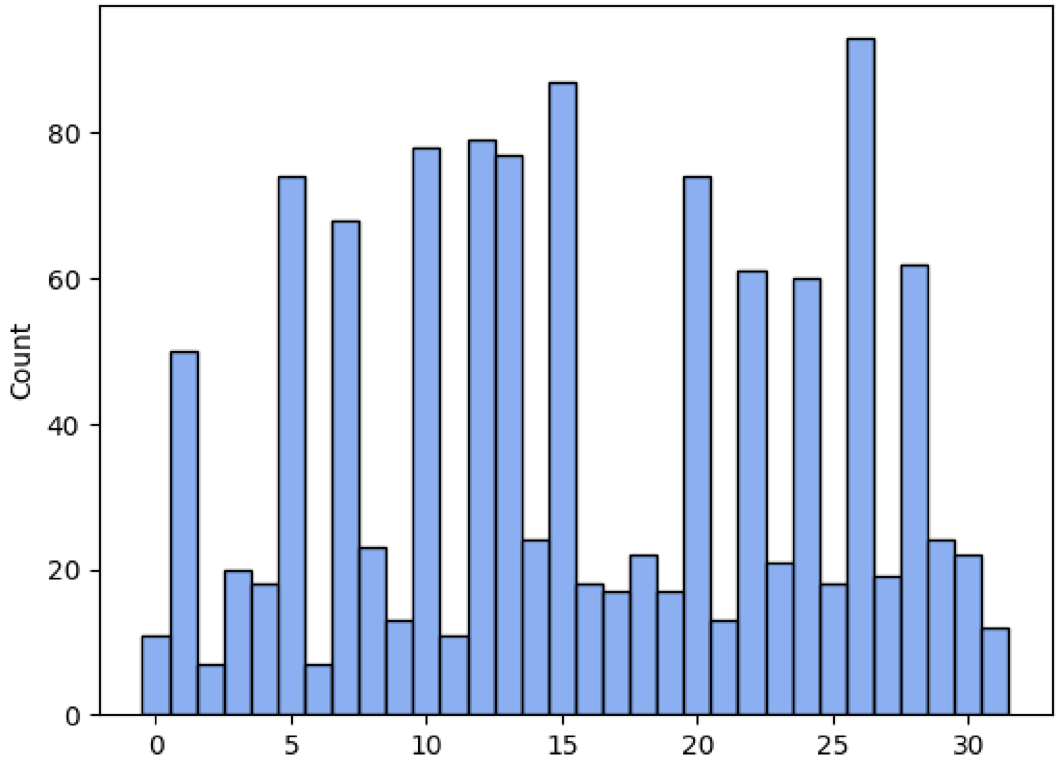
Let's see how k-means behave differently from agglomerative clustering.

2.2.1 Exercise 2.1 - K-Means Clustering for Equivalence Classes (20 points)

- Re-cluster the dataset with k-means, but with the number of clusters you obtained from Exercise 1.
- Show the frequency(number of members) of each cluster. Again, you are encouraged to create a bar chart, but printing the numbers is also fine.

[ ]:
```
# perform clustering
clustering = cluster.KMeans(n_clusters=32).fit(df)

# plot cluster distribution
histplot(clustering.labels_, discrete=True, color='cornflowerblue');
```



[ ]:
```
from tabulate import tabulate
data = [sum(clustering.labels_ == i) for i in range(32)]
tabulate([data], tablefmt='html')
```

[ ]:
```
'<table>\n<tbody>\n<tr><td style="text-align: right;">11</td><td style="text-
align: right;">50</td><td style="text-align: right;">7</td><td style="text-
```

```
align: right;">20</td><td style="text-align: right;">18</td><td style="text-
align: right;">74</td><td style="text-align: right;">7</td><td style="text-
align: right;">68</td><td style="text-align: right;">23</td><td style="text-
align: right;">13</td><td style="text-align: right;">78</td><td style="text-
align: right;">11</td><td style="text-align: right;">79</td><td style="text-
align: right;">77</td><td style="text-align: right;">24</td><td style="text-
align: right;">87</td><td style="text-align: right;">18</td><td style="text-
align: right;">17</td><td style="text-align: right;">22</td><td style="text-
align: right;">17</td><td style="text-align: right;">74</td><td style="text-
align: right;">13</td><td style="text-align: right;">61</td><td style="text-
align: right;">21</td><td style="text-align: right;">60</td><td style="text-
align: right;">18</td><td style="text-align: right;">93</td><td style="text-
align: right;">19</td><td style="text-align: right;">62</td><td style="text-
align: right;">24</td><td style="text-align: right;">22</td><td style="text-
align: right;">12</td></tr>\n</tbody>\n</table>'
```

2.2.2 Exercise 2.2 - Difference between Agglomerative Clustering and K-Means Clustering (10 points)

Compare the result from Exercise 2.1 with that from Exercise 1.2, and explain - How the two results are different - Why there is such a difference

[ ]:

```
```

2.3 Exercise 3 - Principal Component Analysis (30 points in total)

We can visualize how the bitstrings are distributed using principal component analysis.

2.3.1 Exercise 3.1 - Generate 2 Clusters (10 points)

- Re-do the k-means clustering on our dataset again, but this time we only consider `k=2`.
- Show the frequency(number of members) of each cluster.

[ ]:

```
```

2.3.2 Exercise 3.2 - PCA for Feature Extraction (20 points)

- Retrieve the projected dataset with PCA, using `n_components=2`.
- Generate a scatter plot to visualize the projected points, where they should be colored differently based on the assigned cluster in Exercise 3.1.
- In the first principal component, **print out** the weights of all features.
- Report which feature has the **highest positive** weight in the first principal component.

[ ]:

```
```

2.4 Exercise 4 - Collaborative Statement (5 points)

2.4.1 You must fill this out even if you worked alone to get credit.

It is mandatory to include a Statement of Collaboration in each submission, that follows the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed. All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments in particular, I encourage students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, possible bugs in the support code, and the relevant technical content before they start working on it. However, you should not discuss the specific solutions, and as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially after you have started working on the assignment, try to restrict the discussion to Piazza as much as possible, so that there is no doubt as to the extent of your collaboration.

[ ]:

```

```