Clustering Analysis on IoT-Detection and NSL-KDD Datasets

By: Bilel RAHMOUNI

Objective

In this lab, we aimed to apply two basic clustering algorithms—K-means and Hierarchical Clustering (CAH)—on two datasets: NSL-KDD (used for network intrusion detection) and IoT-detection (used for device activity detection). By the end of the lab, we aimed to successfully run these algorithms and interpret the resulting clusters.

Part 1: Introduction to Clustering

Clustering is an unsupervised learning technique that involves grouping similar data points together. In network analysis, clustering can help identify "normal" behavior versus "suspicious" activity in network traffic. Similarly, clustering can group IoT devices based on their activity patterns, aiding in detecting abnormal behaviors.

Clustering Algorithms We Used:

1. **K-means:** This algorithm divides the dataset into a pre-defined number of clusters (in this case, 3). It identifies clusters by minimizing the variance within each cluster.
2. **Hierarchical Clustering (CAH):** This method builds a hierarchy of clusters by iteratively merging similar data points. The hierarchical nature allows us to explore the nested structure of the data, visualized using a dendrogram.

Part 2: Loading the Data

We worked with two datasets:

1. **NSL-KDD:** This dataset contains network traffic data for intrusion detection.
2. **IoT-detection:** This dataset contains activity data from various IoT devices, which we can cluster based on behavior patterns.

For this analysis, we used a pre-processed version of the IoT-detection dataset. We focused on selecting relevant numerical features to perform clustering effectively.

Steps:

- We loaded the dataset into a DataFrame.
- We selected a subset of relevant numerical features: `['IPLength', 'IPHeaderLength', 'TTL', 'SourcePort', 'DestPort', 'WindowSize']`.

Part 3: K-means Clustering

1. Running the K-means Algorithm

- We applied the K-means algorithm on the dataset using 3 clusters. Before applying the algorithm, we scaled the selected features using `StandardScaler` to standardize the data. This step is crucial because K-means is sensitive to the scale of the features.
- We then used PCA (Principal Component Analysis) to reduce the dataset to two principal components for visualization.

2. Code Summary

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Select relevant features and standardize them
selected_features = ['IPLength', 'IPHeaderLength', 'TTL', 'SourcePort',
'DestPort', 'WindowSize']
data_selected = data[selected_features]
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_selected)

# Apply PCA for visualization
pca = PCA(n_components=2)
```

```
data_pca = pca.fit_transform(data_scaled)

# K-means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(data_scaled)

# Plotting the clusters
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=kmeans_labels,
cmap='rainbow')
plt.title('K-means Clustering on PCA-transformed Features')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

3. Results

- **Number of Clusters:** We specified 3 clusters for the K-means algorithm.
- **Visualization:** The plot shows that there is a clear separation between clusters in the PCA-transformed feature space. This indicates that K-means was effective in identifying some natural groupings within the dataset.

4. Observations

- The clusters in the plot indicate some structure in the data, suggesting that K-means can group network activities or device behaviors based on their similarities. However, the clustering depends on the initial choice of the number of clusters (`n_clusters`), which is a limitation of K-means.

Part 4: Hierarchical Clustering (CAH)

1. Running the Hierarchical Clustering Algorithm

- We used hierarchical clustering with the Ward method to progressively merge similar data points into clusters.
- We scaled the selected features using `StandardScaler` and created a dendrogram to visualize the clustering process.

2. Code Summary

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

# Create the linkage matrix using the Ward method
linked = linkage(data_scaled, method='ward')

# Plot the truncated dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked, truncate_mode='lastp', p=30)
plt.title('Dendrogram - Hierarchical Clustering (Truncated)')
plt.xlabel('Cluster Index')
plt.ylabel('Distance')
plt.show()
```

3. Results

- **Dendrogram:** The dendrogram provides a visual representation of how clusters are merged based on their similarity. We used truncation to show the last 30 merged clusters for simplicity.
- **Number of Clusters:** By examining the dendrogram, a natural choice would be to cut at a distance of around 100, resulting in approximately 3 clusters. The height at which we make the cut affects the number of clusters we identify.

4. Observations

- **Hierarchical Clustering:** Unlike K-means, hierarchical clustering does not require us to pre-define the number of clusters. We can decide on the number of clusters by choosing where to "cut" the dendrogram. This flexibility is an advantage of hierarchical clustering.
- The hierarchical nature of this algorithm allows us to explore nested groupings within the dataset, which is helpful for understanding the relationships between different data points.

Part 5: Questions to Answer

1. **K-means Clustering:**
    - **How many clusters did K-means create?**
      We specified 3 clusters for the K-means algorithm.
    - **Can we see any clear separation between clusters in the plot?**
      Yes, the PCA-transformed plot shows a clear separation between clusters. This suggests that the K-means algorithm was able to identify meaningful groupings in the data.
2. **Hierarchical Clustering (CAH):**
    - **Look at the dendrogram. How many clusters do we think are present based on the dendrogram?**
      By examining the dendrogram, we can estimate the number of clusters to be around 3, particularly when cutting at a distance of approximately 100.
    - **How does Hierarchical Clustering differ from K-means in grouping the data?**
        - Hierarchical Clustering builds a tree of clusters and does not require specifying the number of clusters in advance. We can determine the number of clusters by analyzing the dendrogram.
        - K-means requires us to specify the number of clusters before running the algorithm. It partitions the data into spherical clusters centered around centroids.
        - Hierarchical clustering can capture more complex relationships between data points, while K-means is more computationally efficient for larger datasets but assumes spherical clusters.

Conclusion

In this lab, we successfully applied K-means and Hierarchical Clustering on the NSL-KDD and IoT-detection datasets. By scaling the data and reducing dimensions using PCA, we visualized the clusters formed by both algorithms. K-means identified 3 clusters based on our initial specification, while hierarchical clustering provided a more flexible way to explore the cluster structure through a dendrogram. Both algorithms demonstrated clear groupings, indicating the presence of distinct patterns within the dataset.

This analysis highlights the strengths and limitations of each clustering method. While K-means is computationally efficient, it requires pre-defining the number of clusters. Hierarchical clustering, on the other hand, offers flexibility in exploring nested clusters but can become computationally expensive for larger datasets.