Lab 1b: Clustering

By: Malachi Jones, PhD

Outline

- Objectives
- Lab 1b.1: Implementing an Approximate Clustering Algorithm
- Lab 1b.2: Utilizing Elbow Method for K-means Clustering
- References

LAB 1B OBJECTIVES

Lab 1b Objectives

- After this lab, students should be able
  - Have a solid intuition for unsupervised ML Algorithms
  - Use pandas, NumPy, and Matplotlib to read in, process, and visualize data
  - Utilize and (if needed) implement clustering algorithms in python that are suitable for the performance and accuracy tradeoffs that are needed for the target problem

LAB 1B OVERVIEW

Lab 1B Overview

- This lab will consist of the following sections:
  - Lab 1b.1: **Implementing an Approximate Agglomerative Clustering Algorithm (4 points)**
  - Lab 1b.2: **Utilizing Elbow method for k-means clustering (1 point)**

- **Reminder**: ***No additional imports shall be added to any of the python files. Any additions will result in an automatic 0 for the portion of the lab***

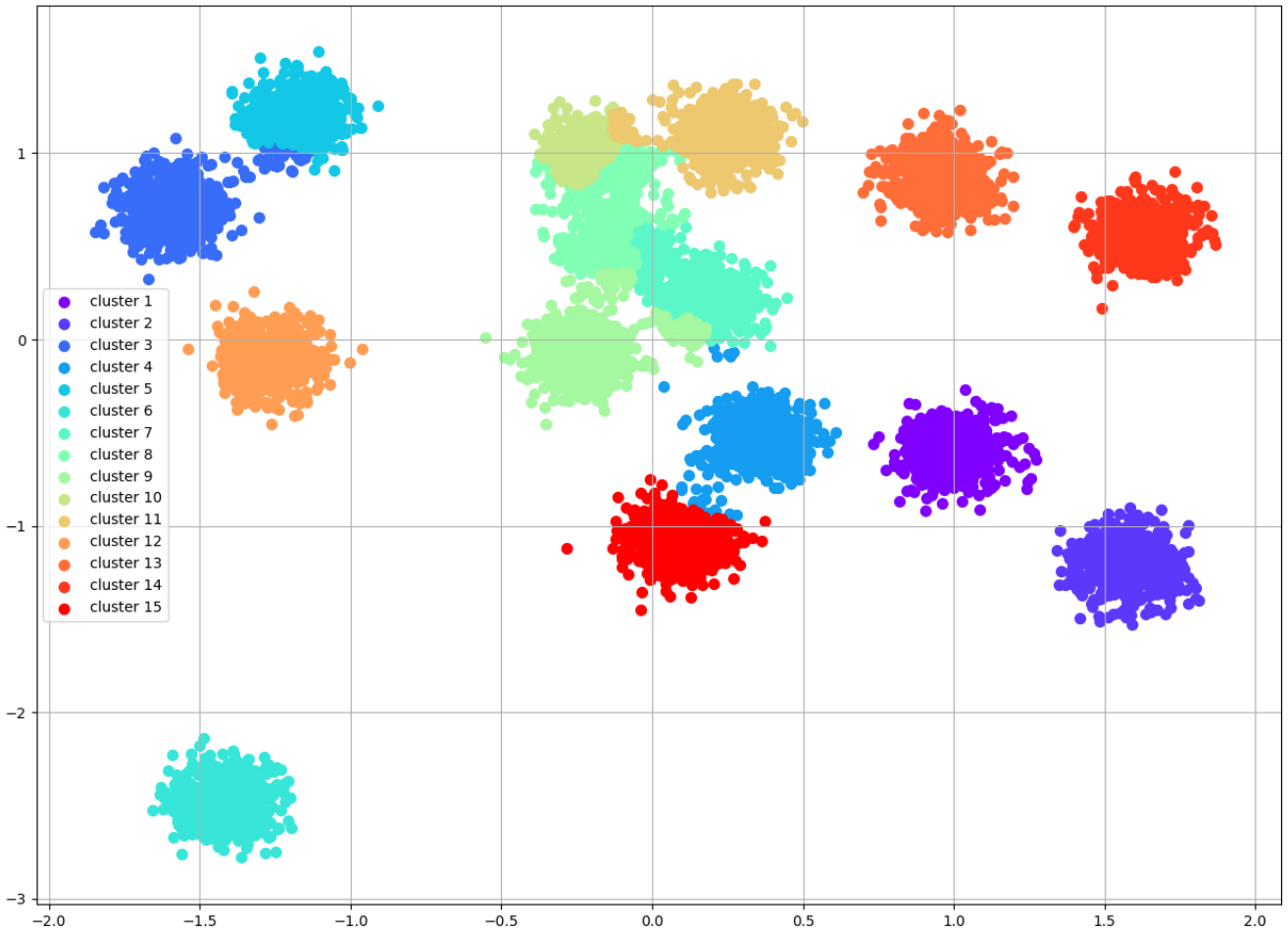
LAB 1B.1: IMPLEMENTING AN APPROXIMATE CLUSTERING ALGORITHM

Lab 1b.1 Implementing Approx. Clustering

- **Lab 1b.1 Objectives**:
  - Implement the approximate agglomerative clustering presented in [1] and run against lab1bsamples_large.csv dataset
  - Once implemented properly and executed against the specified dataset, the resulting plot should look like the plot shown on the next slide

Lab 1b.1 Implementing Approx. Clustering

- **Lab 1b.1 Objectives**:



Lab 1b.1 Implementing Approx. Clustering

- **Lab 1b.1 Steps:**
  1. Implement the Euclidean distance metric

```python
@staticmethod
def distance(node_a, node_b):
    node_a_data_point = node_a.data_point
    node_b_data_point = node_b.data_point

    distance = 0

    logger.warning("@todo: Implement euclidean distance function")

    return distance
```

The `distance()` method that will need to be updated is in the file `ApproxAgglomerativeClustering.py`

Lab 1b.1 Implementing Approx. Clustering

- **Lab 1b.1 Steps:**
  2. Compute the complete linkage distance of Node A and Node B

```
@staticmethod
def complete_linkage_distance(cluster_a, cluster_b):

    """
     Computes the complete linkage distance of two clusters
    :param cluster_a:
    :param cluster_b:
    :return:
    """

    # Note: Complete linkage computes the maximum distance of a pair of nodes from cluster a
    #       and cluster b

    max_distance = 0

    logger.warning("@todo: Implement complete linkage distance")

    return max_distance
```

The `complete_linkage_distance()` method that will need to be updated is in the file `ApproxAgglomerativeClustering.py`

Lab 1b.1 Implementing Approx. Clustering

- **Lab 1b.1 Steps:**
  3. Determine which pair of clusters are the minimum distance apart

```
@staticmethod
def get_min_cluster_distance(cluster_distance_dict):

    """
    :param cluster_distance_dict: 2-dimmensional dictionary that stores the distance of a pair of cluster nodes
                                  Example: To retrieve the distance of cluster_a and cluster, do the following:
                                           cluster_distance = cluster_distance_dict[cluster_a][cluster_b]

    :return: A tuple, where the first element of tuple is itself a tuple of the pair of clusters who have the smallest distance.
             The second element of tuple is the distance of those two tuples
    """

    min_cluster_distance = 0

    min_cluster_pair = ()

    logger.warning(
        "@Todo: return a tuple where the first element is the a pair of clusters who has the minimal distance "
        "and the second element of tuple is the distance")

    return (min_cluster_pair, min_cluster_distance)
```

The `get_min_cluster_distance()` method that will need to be updated is in the file `ApproxAgglomerativeClustering.py`

Lab 1b.1 Implementing Approx. Clustering

- The relevant files for Lab 1b.1 (included in folder Lab1) ***that will be updated*** are the following:
  - `ApproxAgglomerativeClustering`.py

- Files you may need to reference but ***should NOT be updated***
  - Lab1bsamples_large.csv

Lab 1b.1 Implementing Approx. Clustering

- **Submission**
  - You will submit the following file(s) into a folder called lab_1b :
    i. Your implementation of ApproxAgglomerativeClustering.py

LAB 1B.2 UTILIZING ELBOW METHOD FOR K-MEANS

Lab 1b.2 Utilizing Elbow method for K-means

- **Lab 1b.2 Objectives**:
  1. Use the elbow method to determine what the number of clusters, *k*, should be
  2. Perform k-means clustering with the k determined from (1)

Lab 1b.2 Utilizing Elbow method for K-means

- **Lab 1b.2 Steps**:
  1. Use the elbow method to determine k

```
def main():

    # Load the dataset from file
    df = pd.read_csv("lab1bsamples_large.csv", header=None)
    df.tail()
    print(df.tail())

    # get all of the points from the dataset as an array of 2-dimensional vectors
    X = df.iloc[:].values

    # Value that you want to adjust
    num_clusters=2

    #perform_k_means_clustering(X, num_clusters)

    elbow_method_plot(X)

    print("Number of clusters is '{}'".format(num_clusters))
```

The file that that will need to modified is named `kmeans.py`

Lab 1b.2 Utilizing Elbow method for K-means

- **Lab 1b.2 Steps**:
  2. Perform k-means clustering with the k determined from the previous step

```
def main():

    # Load the dataset from file
    df = pd.read_csv("lab1bsamples_large.csv", header=None)
    df.tail()
    print(df.tail())

    # get all of the points from the dataset as an array of 2-dimensional vectors
    X = df.iloc[:].values

    # Value that you want to adjust
    num_clusters=4

    perform_k_means_clustering(X, num_clusters)

    #elbow_method_plot(X)

    print("Number of clusters is '{}'".format(num_clusters))
```

Variable that will need to be updated

The file that that will need to modified is named `kmeans.py`

Lab 1B.2 Utilizing Elbow method for K-means

- The relevant files **to be modified** for Lab 1b.2 (included in folder Lab1) consists of the following:
  - kmeans.py

- Files you may need to reference but ***should NOT be updated***
  - Lab1bsamples_large.csv

Lab 1b.2 Utilizing Elbow method for K-means

- **Submission**
  - You will submit the following file(s) into a folder called lab_1b :
    i. A `kmeans.py` file with the appropriate `num_clusters` value determined from the elbow methd

References

1. Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, *19*(4), 639-668.