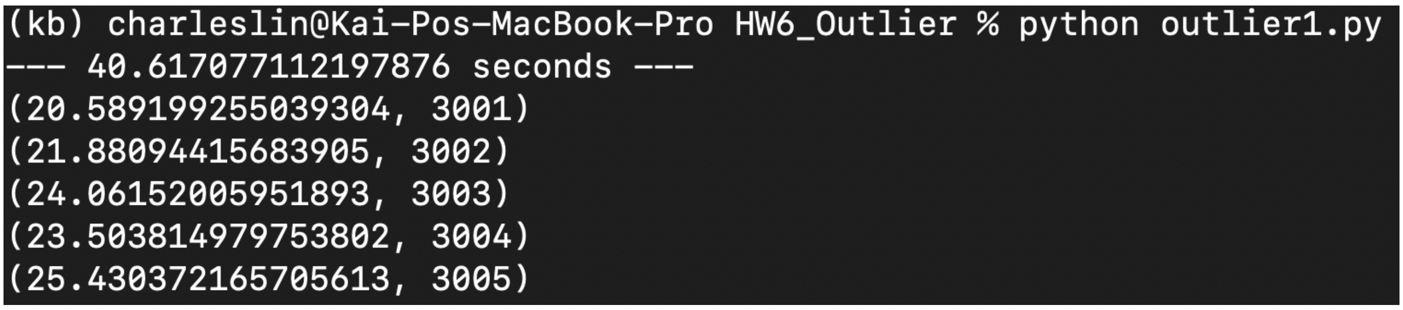
Comp 543 HW6-Outliers

Kai-Po Lin (kl72)

【Code – Task 1】

```python
import heapq as hq
import numpy as np
import time


if __name__ == '__main__':
    # Create the covariance matrix
    covar = np.zeros((100,100))
    np.fill_diagonal(covar, 1)

    # And the mean vector
    mean = np.zeros(100)

    # Create 3000 data points
    all_data = np.random.multivariate_normal(mean, covar, 3000)

    # Now create the 20 outliers
    for i in range(1, 20):
        mean.fill(i)
        outlier_data = np.random.multivariate_normal(mean, covar, i)
        all_data = np.concatenate((all_data, outlier_data))

    # k for kNN detection
    k = 10

    # The number of outliers to return
    m = 5

    # Start the timer
    start_time = time.time()

    # The priority queue of outliers
    outliers = list()

    #YOUR CODE HERE!
    outliersDict = dict()
    for idx, i in enumerate(all_data):
        maxPriority = list()
        hq.heapify(maxPriority)
        for jdx, j in enumerate(all_data):
            if idx == jdx:
                continue
            hq.heappush(maxPriority, -np.linalg.norm(j - i))
            if len(maxPriority) > k:
                hq.heappop(maxPriority)
        # Insert idx into with key max(maxPriority)
        outliersDict[-hq.heappop(maxPriority)] = idx

        if len(outliersDict) > m:
            minPriority = list(outliersDict.keys())
            hq.heapify(minPriority)
            outlierKey = hq.heappop(minPriority)
            del outliersDict[outlierKey]

    # Get key by dict() value
    for key, value in outliersDict.items():
        outliers.append((key, value))

    print("--- %s seconds ---" % (time.time() - start_time))

    # Print the outliers...
    for outlier in outliers:
        print(outlier)
```

【Result – Task 1】=> About 40.617 seconds

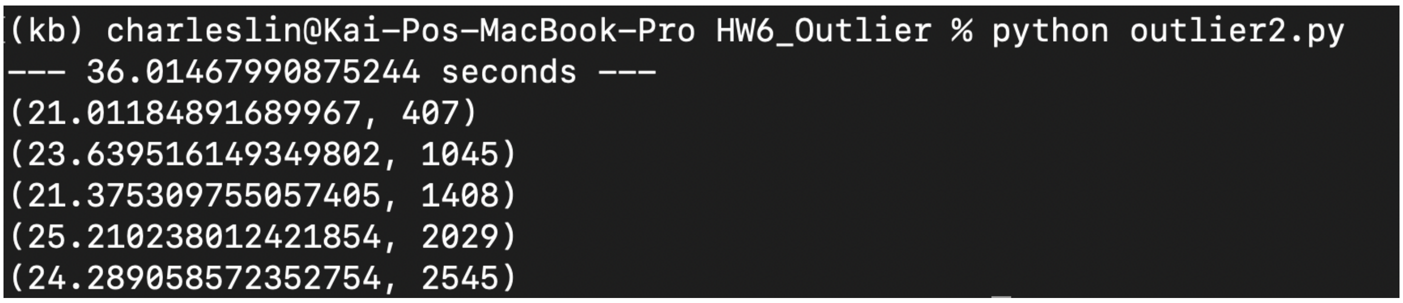


```
(kb) charleslin@Kai-Pos-MacBook-Pro HW6_Outlier % python outlier1.py
--- 40.617077112197876 seconds ---
(20.589199255039304, 3001)
(21.88094415683905, 3002)
(24.06152005951893, 3003)
(23.503814979753802, 3004)
(25.430372165705613, 3005)
```

【Code – Task 2】

```
import heapq as hq
import numpy as np
import time


if __name__ == '__main__':
    # Create the covariance matrix
    covar = np.zeros((100,100))
    np.fill_diagonal(covar, 1)

    # And the mean vector
    mean = np.zeros(100)

    # Create 3000 data points
    all_data = np.random.multivariate_normal(mean, covar, 3000)

    # Now create the 20 outliers
    for i in range(1, 20):
        mean.fill(i)
        outlier_data = np.random.multivariate_normal(mean, covar, i)
        all_data = np.concatenate((all_data, outlier_data))

    # Randomly shuffle the data
    np.random.shuffle(all_data)

    # k for kNN detection
    k = 10

    # The number of outliers to return
    m = 5

    # Start the timer
    start_time = time.time()

    # The priority queue of outliers
    outliers = list()

    #YOUR CODE HERE!
    outliersDict = dict()
    minOutlierVal = 0
    for idx, i in enumerate(all_data):
        flag = False
        maxPriority = list()
        hq.heapify(maxPriority)
        for jdx, j in enumerate(all_data):
            if idx == jdx:
                continue
            hq.heappush(maxPriority, -np.linalg.norm(j - i))
            if len(maxPriority) > k:
                hq.heappop(maxPriority)
            elif len(maxPriority) == k and len(outliersDict) == m and -
hq.nsmallest(1, maxPriority)[0] < minOutlierVal:
                flag = True
                break
        if flag:
            continue
        # Insert idx into with key max(maxPriority)
        outliersDict[-hq.heappop(maxPriority)] = idx

        if len(outliersDict) >= m:
            minPriority = list(outliersDict.keys())
            hq.heapify(minPriority)
            minOutlierVal = hq.nsmallest(1, minPriority)[0]

            if len(outliersDict) > m:
                outlierKey = hq.heappop(minPriority)
                del outliersDict[outlierKey]

    # Get key by dict() value
    for key, value in outliersDict.items():
        outliers.append((key, value))

    print("--- %s seconds ---" % (time.time() - start_time))

    # Print the outliers...
    for outlier in outliers:
        print(outlier)
```

【Result – Task 2】=> About 36.015 seconds



```
(kb) charleslin@Kai-Pos-MacBook-Pro HW6_Outlier % python outlier2.py
--- 36.01467990875244 seconds ---
(21.01184891689967, 407)
(23.639516149349802, 1045)
(21.375309755057405, 1408)
(25.210238012421854, 2029)
(24.289058572352754, 2545)
```