

## CX4010 HW2

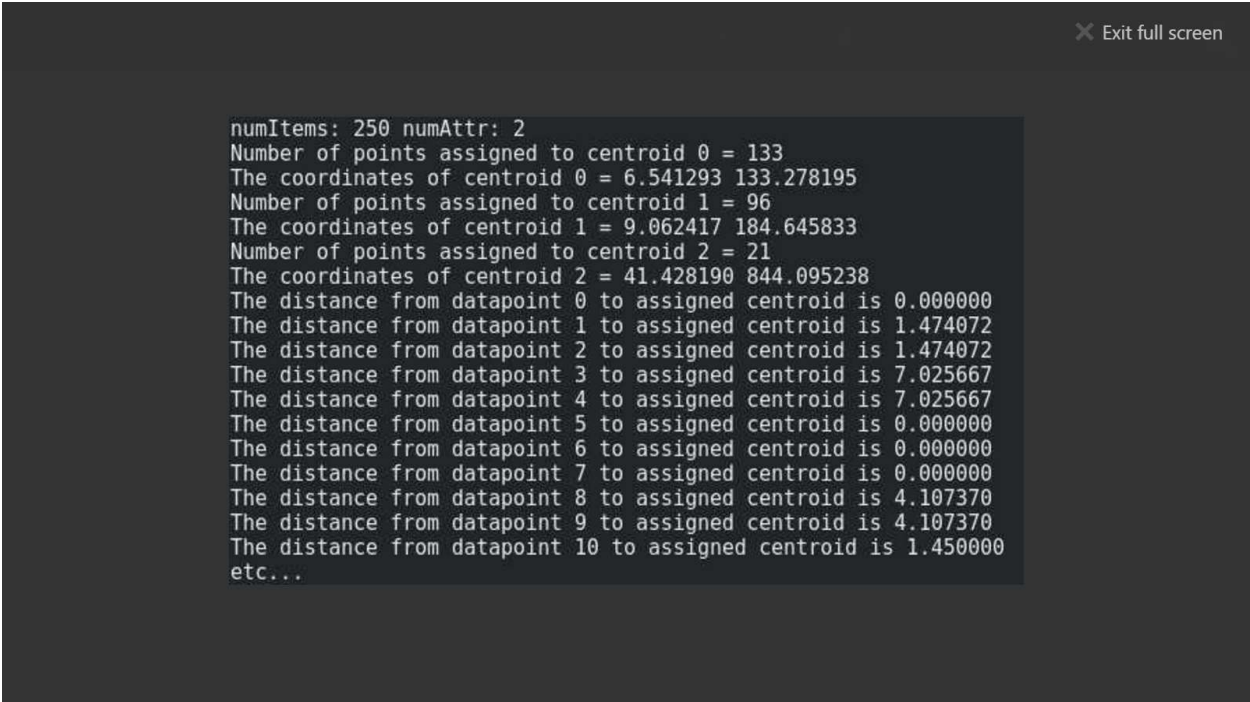
Thomas Hervieu and IsaK Thomas

### KMEANS:

If the amount of centroids is too large, some of the clusters will become empty.

If the amount of centroids is too small then it wouldn't be an accurate representation of the data.

The way we determined that the optimal amount of centroids is three is that there was a fair distribution among the centroids. One always had significantly more, but the distribution of the other two were fairly close.

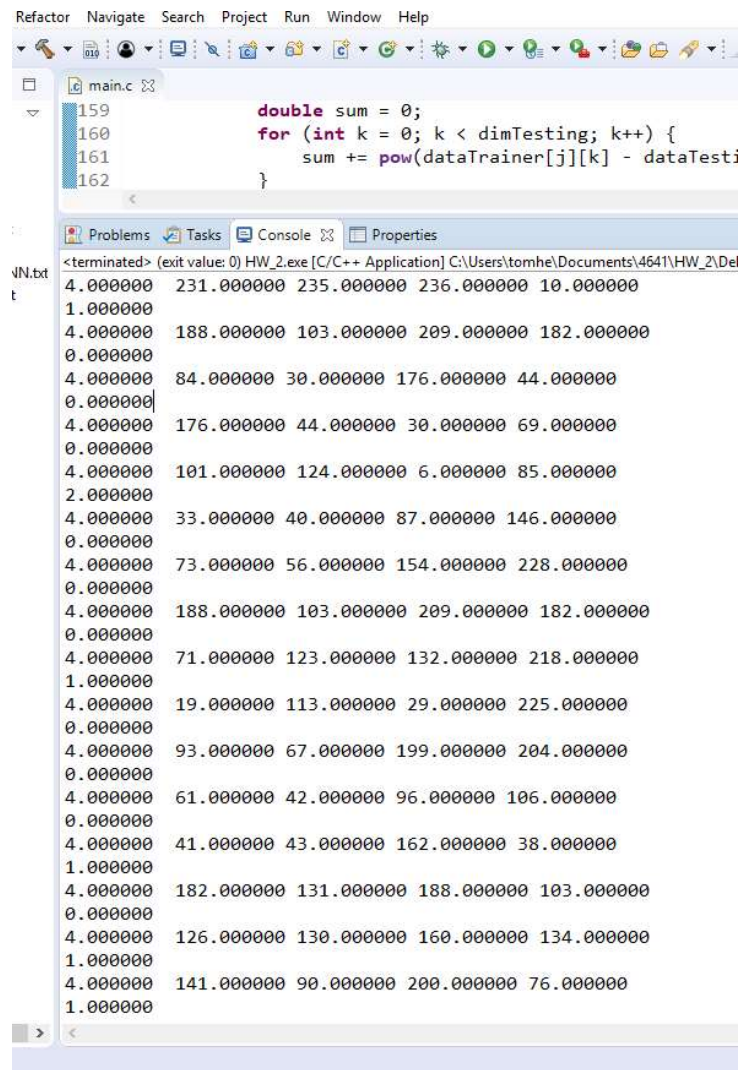


The screenshot shows a terminal window with a dark background and light gray text. In the top right corner, there is a small icon of a window with an 'X' and the text 'Exit full screen'. The terminal output displays the results of a K-means clustering process on a dataset with 250 items and 2 attributes. It lists the number of points assigned to each of the three centroids, the coordinates of each centroid, and the distance from the first 11 data points to their assigned centroid. The output shows that centroid 0 has 133 points, centroid 1 has 96 points, and centroid 2 has 21 points. Distances for points 0-10 are shown, with some points (0, 5, 6, 7) having a distance of 0.000000.

```
numItems: 250 numAttr: 2
Number of points assigned to centroid 0 = 133
The coordinates of centroid 0 = 6.541293 133.278195
Number of points assigned to centroid 1 = 96
The coordinates of centroid 1 = 9.062417 184.645833
Number of points assigned to centroid 2 = 21
The coordinates of centroid 2 = 41.428190 844.095238
The distance from datapoint 0 to assigned centroid is 0.000000
The distance from datapoint 1 to assigned centroid is 1.474072
The distance from datapoint 2 to assigned centroid is 1.474072
The distance from datapoint 3 to assigned centroid is 7.025667
The distance from datapoint 4 to assigned centroid is 7.025667
The distance from datapoint 5 to assigned centroid is 0.000000
The distance from datapoint 6 to assigned centroid is 0.000000
The distance from datapoint 7 to assigned centroid is 0.000000
The distance from datapoint 8 to assigned centroid is 4.107370
The distance from datapoint 9 to assigned centroid is 4.107370
The distance from datapoint 10 to assigned centroid is 1.450000
etc...
```

## KNN:

Code successfully ran for a synthetic version of the Kmeans output file. If K is less than the number of rows of the training set, the code will run correctly. I realized that a k of around 10 will be ideal for labeling the testing data points. A k that is too low tends to label the points prematurely because there may be many more data points of another class near the testing point that are in closest proximity to the testing point apart from only a couple points from another class. Also, if k is too large (k = 100) the bigger clusters (which contain the most labels) will dominate almost every testing point, which will result in all the testing points being classified to one class.



The screenshot shows a C++ IDE with a file named `main.c` open. The code defines a function `double sum = 0;` and a `for` loop that iterates over `dimTesting` and calculates the sum of squared differences between training and testing data points. The console output shows the results of the K-Nearest Neighbors (KNN) algorithm for a synthetic dataset. The output is a list of 20 rows, each containing 5 values. The first column represents the predicted class (0 or 1), and the other four columns represent the distances to the four nearest neighbors. The output shows that the algorithm correctly classifies most points, with some misclassifications (e.g., the first row is classified as 0 but the second row is classified as 1).

```
double sum = 0;
for (int k = 0; k < dimTesting; k++) {
    sum += pow(dataTrainer[j][k] - dataTesti
```

<terminated> (exit value: 0) HW\_2.exe [C:/C++ Application] C:\Users\tomhe\Documents\4641\HW\_2\Del  
4.000000 231.000000 235.000000 236.000000 10.000000  
1.000000  
4.000000 188.000000 103.000000 209.000000 182.000000  
0.000000  
4.000000 84.000000 30.000000 176.000000 44.000000  
0.000000  
4.000000 176.000000 44.000000 30.000000 69.000000  
0.000000  
4.000000 101.000000 124.000000 6.000000 85.000000  
2.000000  
4.000000 33.000000 40.000000 87.000000 146.000000  
0.000000  
4.000000 73.000000 56.000000 154.000000 228.000000  
0.000000  
4.000000 188.000000 103.000000 209.000000 182.000000  
0.000000  
4.000000 71.000000 123.000000 132.000000 218.000000  
1.000000  
4.000000 19.000000 113.000000 29.000000 225.000000  
0.000000  
4.000000 93.000000 67.000000 199.000000 204.000000  
0.000000  
4.000000 61.000000 42.000000 96.000000 106.000000  
0.000000  
4.000000 41.000000 43.000000 162.000000 38.000000  
1.000000  
4.000000 182.000000 131.000000 188.000000 103.000000  
0.000000  
4.000000 126.000000 130.000000 160.000000 134.000000  
1.000000  
4.000000 141.000000 90.000000 200.000000 76.000000  
1.000000

The 2 input files and output file have been included under the KNN folder to show that the calculations are correct. To the left is a sample execution with K=4 nearest neighbors.