Project 2: Biomedical Data Analysis

Final Report

Liliko Uchida

**Description of Nearest Neighbor and K-Nearest Neighbor Functions**

*Nearest Neighbor Functions*

General Algorithm:

1. Normalize all data points to eliminate malfunctions in distance calculations.
2. Create a random test case.
3. Calculate the distance between the random test case and every pre-existing data point in the set. Create an array with all of these distances called zArray.
4. Find the minimum distance in the zArray and extract the index to which this minimum distance corresponds. Find the pre-existing data point which corresponds to this minimum distance, and finally, using this index, find the corresponding class to this pre-existing data from the array of original classifications.
5. Assign the classification of the nearest data point to the random test case.

*K-Nearest Neighbor Functions*

General Algorithm:

1. (1-4 remains the same as above)
2. repeat k times.
3. Find the average classification value of the k different pre-existing data points.
4. Assign the average classification to the random test case.

*Differences in Calculations:*

While in Nearest Neighbor classifications, the code only looks at the single closest data point and assigns the test case with the classification corresponding to it, the K-Nearest Neighbor classification method gathers data from multiple points instead of just one. K-Nearest Neighbor classification can be helpful in data sets that involve outliers. It mitigates the possibility of a test case being incorrectly classified in accordance to an outlier by looking at the general surroundings and classifying the test case based on a general trend rather than a single point.

**Description of K-Means Clustering Functions**

K-Means Clustering

Flow Chart

General Algorithm:

1. Initialize clusters.
2. Calculate distances between data points and clusters.
3. Find average value of data points associated with each cluster.
4. Reassign clusters to these average values.
5. Repeat until reassigned values are equivalent to previous values.

Results from Testing Code with K = 1, 2, 3:

*k = 1…*

|  |  |  |
| --- | --- | --- |
| **Trial** | **initialClusterArray\_x** | **initialClusterArray\_y** |
| 1 | [0.809] | [0.026] |
| 2 | ﻿[0.095] | [0.123] |
| 3 | [0.693] | [0.043] |

*k = 2…*

|  |  |  |
| --- | --- | --- |
| **Trial** | **initialClusterArray\_x** | **initialClusterArray\_y** |
| 1 | [0.939, 0.707] | [0.062, 0.143] |
| 2 | ﻿[0.891. 0.347] | [0.031, 0.138] |
| 3 | [0.810, 0.707] | [0.093, 0.064] |

*k = 3…*

|  |  |  |
| --- | --- | --- |
| **Trial** | **initialClusterArray\_x** | **initialClusterArray\_y** |
| 1 | [0.891, 0.456, 0.524] | [0.112, 0.221, 0.121] |
| 2 | ﻿[0.884, 0.707, 0.340] | [0.019, 0.143, 0.367] |
| 3 | [0.497, 0.946, 0.286] | [0.353, 0, 0.440] |