Task 1 (70 points, programming)

In this task you will implement EM clustering.

Command-line Arguments

You must implement a Matlab function or a Python executable file called `em_cluster`. Your function will be invoked as follows:

```
em_cluster(<data_file>, <k>, <iterations>)
```

If you use Python, just convert the Matlab function arguments shown above to command-line arguments. The arguments provide the following information:

- The first argument, <data_file>, is the path name of a file where the data is stored. The path name can specify any file stored on the local computer.
- The second argument, <k>, specifies the number of clusters.
- The third argument, <iterations>, specifies the number of iterations of the main loop. The initialization stage (giving a random assignment of objects to clusters) does not count as part of an iteration. After the initialization, you start with the M-step, and follow with the E-step. Each iteration consists of an M-step and an E-step.

The data file will follow the same format as the training and test files in the UCI datasets directory. A description of the datasets and the file format can be found on this link. Your code should also work with ANY OTHER training and test files using the same format as the files in the UCI datasets directory.

As the description states, **do NOT use data from the last column (i.e., the class labels) as features**. In these files, all columns except for the last one contain example inputs. The last column contains the class label.

An example file containing 2D points, that you can use to test your code, is point_set1.txt. In that file, the third column should be ignored, in the same way that you ignore the last column in every UCI dataset file.

Implementation Guidelines

- To initialize your variables before the EM iterations, you should assign randomly each point to a cluster. In other words, using the notation on slides 69-70 of the clustering slides, you initialize the p_{ij} values so that, for each j, one p_{ij} value (for some randomly selected i) is set to 1, and the other p_{ij} values are set to 0. After this initialization, you start with the M-step, and you follow that with the E-step. Each iteration consists of one M-step and one E-step.

- For each covariance matrix, you compute values as described on slide 70 of the clustering slides. To make sure that the covariance matrices are well-behaved, when you compute the entries for diagonal values (i.e., when r = c), if you end up with a value less than 0.0001, replace that value with 0.0001. In other words, diagonal values of the covariance matrix should not be allowed to be smaller than 0.0001.

Output on the Screen

After each iteration, you should print the weight and mean of each Gaussian. After all iterations are done, you should print out the mean and covariance matrix for each Gaussian.

The output should follow this format:

```
After iteration 1:
weight 1 = %.4f, mean 1 = (%.4f, ..., %.4f)
weight 2 = %.4f, mean 2 = (%.4f, ..., %.4f)
...
weight <k> = %.4f, mean 1 = (%.4f, ..., %.4f)
After iteration 2:
weight 1 = %.4f, mean 1 = (%.4f, ..., %.4f)
weight 2 = %.4f, mean 2 = (%.4f, ..., %.4f)
...
weight <k> = %.4f, mean 1 = (%.4f, ..., %.4f)
...
<k>
After final iteration:
weight 1 = %.4f, mean 1 = (%.4f, ..., %.4f)
Sigma 1 row 1 = %.4f, ..., %.4f
Sigma 1 row 2 = %.4f, ..., %.4f
...
Sigma 1 row <d> = %.4f, ..., %.4f
weight 2 = %.4f, mean 2 = (%.4f, ..., %.4f)
Sigma 2 row 1 = %.4f, ..., %.4f
Sigma 2 row 2 = %.4f, ..., %.4f
...
Sigma 2 row <d> = %.4f, ..., %.4f
...
weight <k> = %.4f, mean 1 = (%.4f, ..., %.4f)
Sigma <k> row 1 = %.4f, ..., %.4f
Sigma <k> row 2 = %.4f, ..., %.4f
...
Sigma <k> row <d> = %.4f, ..., %.4f
```

In the above format specifications, you should substitute the actual values for k and d where you see <k> and <d> respectively.

Output for answers.pdf

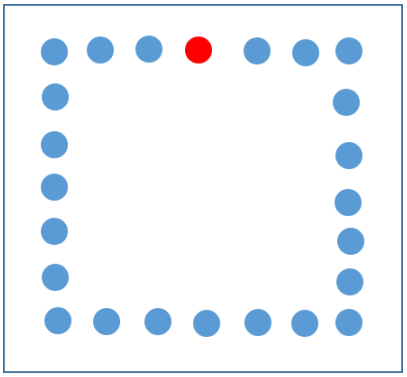
In your answers.pdf document, you need to provide the complete output for the following invocations of your program:

```
em_cluster('point_set1.txt', 2, 5)
em_cluster('point_set1.txt', 3, 5)
```

Grading

- 55 points: Correct implementation of EM clustering.
- 15 points: Following the specifications in producing the required output.

Task 2 (10 points)



Consider the set of points above. Each dot corresponds to a point. Consider a clustering consisting of two clusters, where the first cluster is the set of all the blue dots, and the second cluster has the red dot as its only element. Can this clustering be the final result of the k-means algorithm? Justify your answer.

Your answer should be based on your visual estimations of Euclidean distances between dots. For this particular question, no greater precision is needed.

Task 3 (10 points)

For both parts of this task, assume that the algorithm in question **never needs to break a tie** (i.e., it never needs to choose between two distances that are equal).

Part a: Will the k-means algorithm always give the same results when applied to the same dataset with the same k? To phrase the question in an alternative way, is there any dataset where the algorithm can produce different results if run multiple times, with the value of k kept the same? If your answer is that k-means will always give the

same results, justify why. If your answer is that k-means can produce different results if run multiple times, provide an example.

Part b: Same question as in part a, but for agglomerative clustering with the d_{min} distance. Here, as "result" we consider all intermediate clusterings, between the first step (with each object being its own cluster) and the last step (where all objects belong to a single cluster). Will this agglomerative algorithm always give the same results when applied to the same dataset? If your answer is "yes", justify why. If your answer is "no", provide an example.

Task 4 (10 points)

Consider a dataset consisting of these eight points: 2, 4, 7, 11, 16, 22, 29, 37.

Part a: Show the results (all intermediate clusterings) obtained by applying agglomerative clustering to this dataset, using the d_{min} distance.

Part b: Show the results (all intermediate clusterings) obtained by applying agglomerative clustering to this dataset, using the d_{max} distance.