Part A Document Clustering

Question 1

I. Derive Expectation and Maximisation steps of the hard-EM algorithm for Document Clustering, show your work in the submitted report.

Expectation Step

1. Calculate the posterior $P(Z|X,\theta^{old})$.
2. For each point, assign P = 1 to where the maximum probability is found in the posterior and P = 0 for others.

Maximisation Step

1. Recalculate the effective number of points in cluster k (N_k).
2. Recalculate the relative cluster size (N_k/N).
3. Recalculate the new cluster centroids (μ_k).
4. Recalculate the covariance matrix (Σ_k).

II. Implement the hard-EM (you derived above) and soft-EM (derived in Chapter 5 of Module 4). Please provide enough comments in your submitted code.

See Jupyter notebook.

III. Load Task3A.text file and necessary libraries (if needed, perform text preprocessing similar to what we did in Activity 4.2), set the number of clusters K=4, and run both the soft-EM and hard-EM algorithms on the provided data.

See Jupyter notebook.

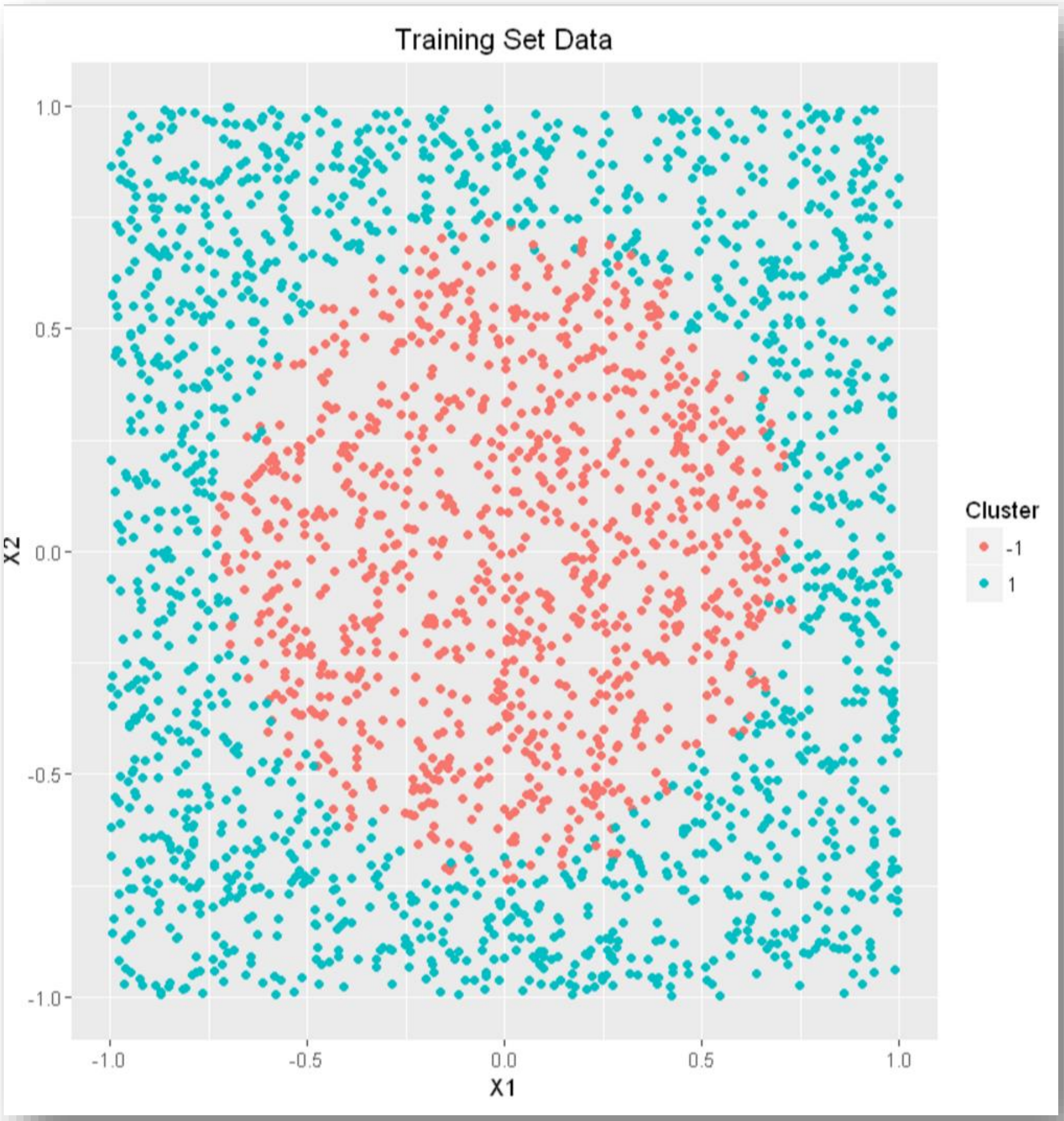
IV. Perform a PCA on the clusterings that you get based on the hard-EM and soft-EM in the same way we did in Activity 4.2. Then, visualise the obtained clusters with different colors where x and y axes are the first two principal components (similar to Activity 4.2). Save your visualizations as plots, and attach them to your report.

I couldn't get this to work because of my code and I ran out of time. Will look into it at the end of the subject to make sure I understand the implementation.

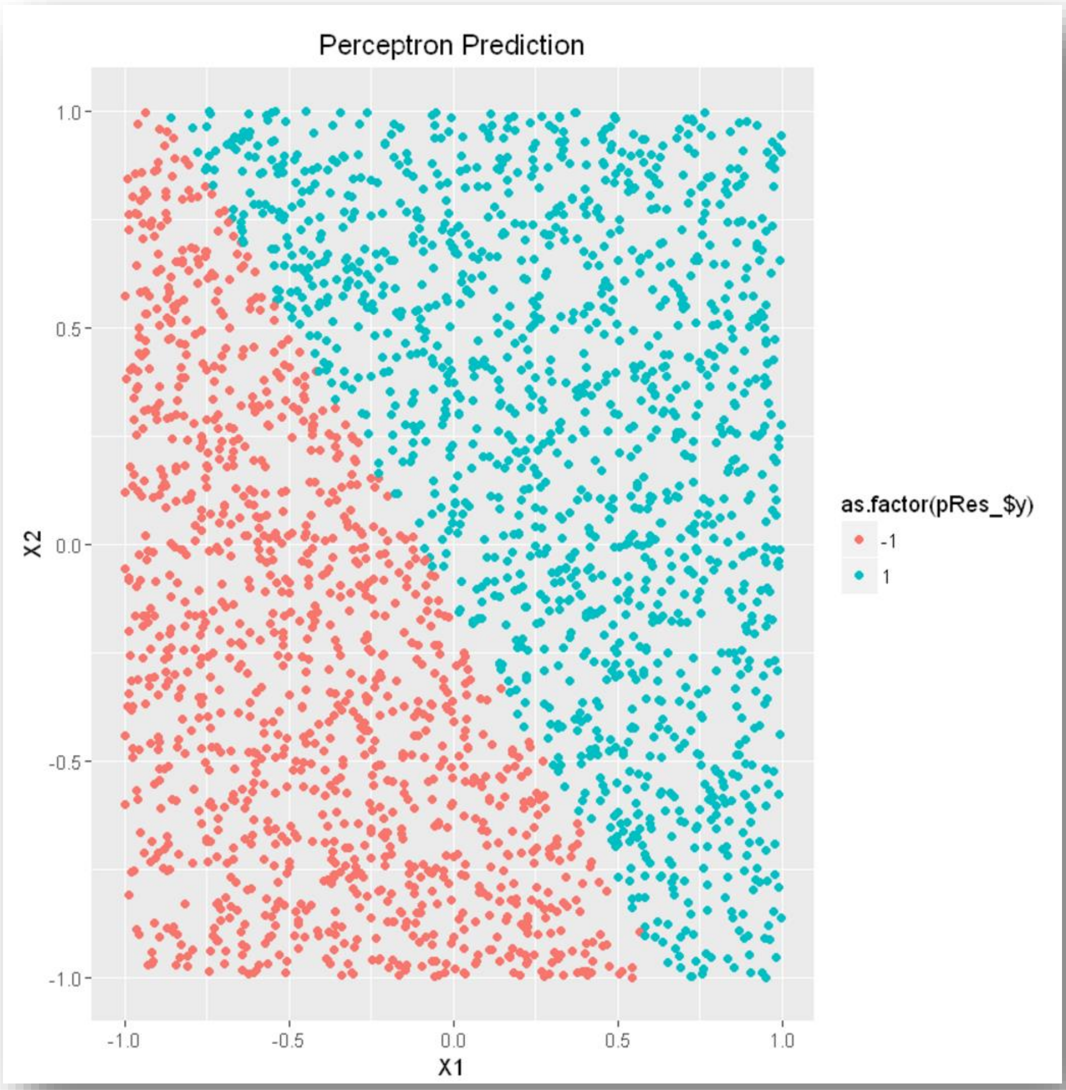
Part B. Neutral Network vs Perceptron

Question 2

I. Load Task3B_train.csv and Task3B_test.csv sets, plot the training data with classes are marked with different colors, and attach the plot to your report.


Training Set Data
1.0
0.5
0.0
-0.5
-1.0
X2
-1.0
-0.5
0.0
0.5
1.0
X1
Cluster
● -1
● 1

II. Run the implementations of Perceptron given to you in Activity 3.1, calculate the test error, and plot the test data while the points are colored with their estimated class labels; attach the pdf to your report.



III. Run the 3-layer Neural Network given to you in Activity 5.1 with different values of K (i.e, number of units in the hidden layer) and record testing error for each of them; plot the error vs K and attach it to your report. Based on this plot, find the best K and the corresponding model, then plot the test data while the points are colored with their estimated class labels using the best model that you have selected; attach the plot to your report.

IV. In a table, report the error rates obtained by the perceptron and all variants of NN. Then bold the best model (with minimum error). Add this table to your report.

K	Error
perceptron	0.514
2	0.5768
4	0.4344
6	0.432
8	0.4332
10	0.434
12	0.4348
14	0.4348
16	0.4324
18	0.4348
20	0.4344
22	0.4332
24	0.4336
26	0.4336
28	0.4332
30	0.4324
32	0.4352
34	0.4332
36	0.4344
38	0.4332
40	0.434
42	0.4328
44	0.4348
46	0.4336
48	0.4332
50	0.4336
52	0.4344
54	0.4332
56	0.4336
58	0.4348
60	0.4336
62	0.4336
64	0.4332
66	0.4328
68	0.2616
70	0.4336
72	0.2628
74	0.2664
76	0.262
78	0.2644
80	0.2616
82	0.4368
84	0.2648
86	0.2628
88	0.264
90	0.2628
92	0.2672
94	0.2648
96	0.2624
98	0.266
100	0.2644

V. In your report explain the reason(s) responsible for such difference between perceptron and a 3-layer NN.

The poor prediction by the perceptron is most likely due the classes not being able to be linearly separable. The Neural Network does perform better at this however, fails to capture the full separation. Perhaps with a larger number of neurons in the hidden layer or deep architecture, successful separation may be achieved.

Part C. Self-Taught Learning

Question 3

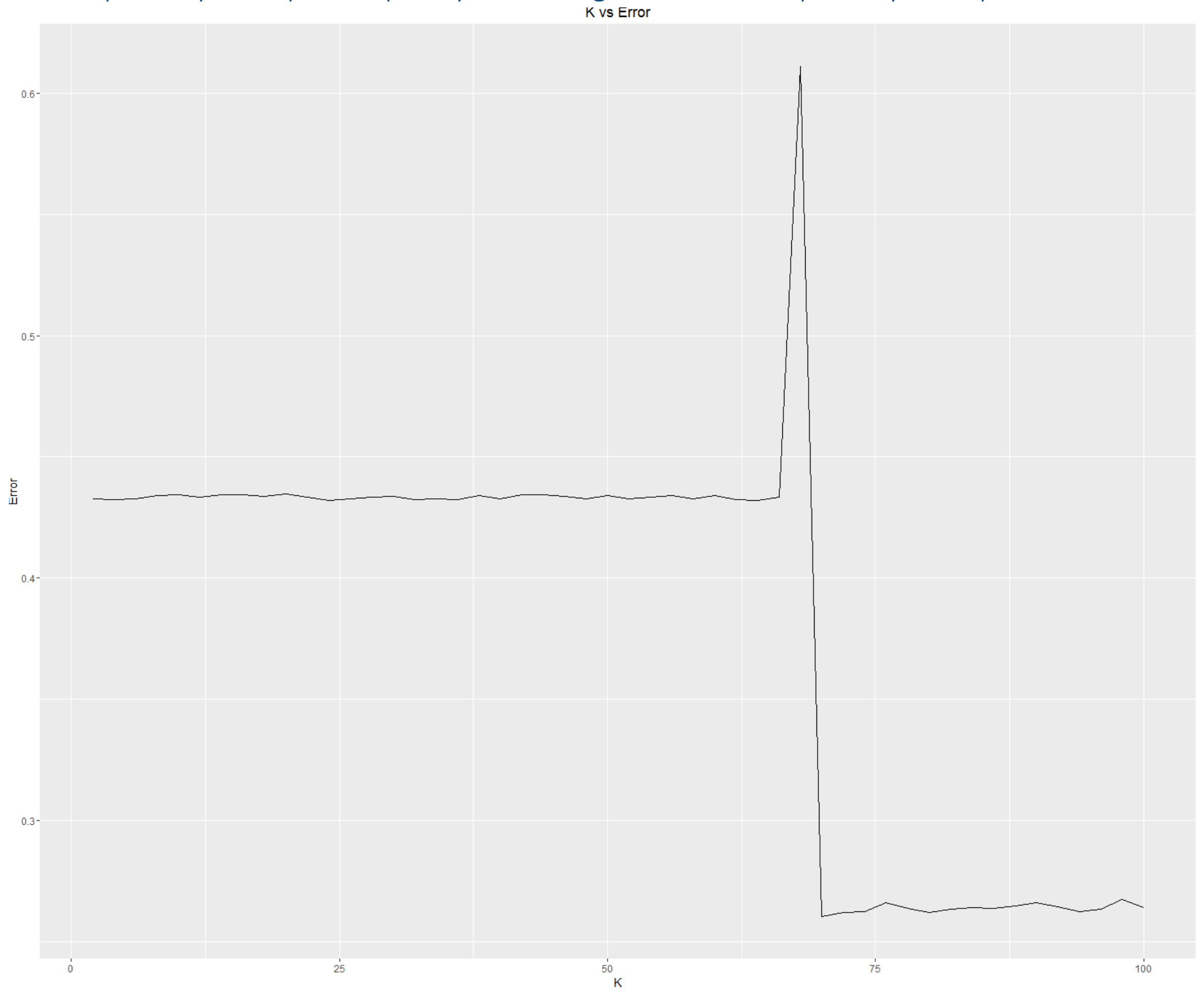
I. Load Task3C_labeled.csv, Task3C_unlabeled.csv and Task3C_test.csv data sets and required libraries (e.g., H2O). Note that we are going to use Task3C_labeled.csv and Task3C_unlabeled.csv for training the autoencoder. We are going to use Task3C_labeled.csv for training the classifier. Finally, we evaluate the trained classifier on the test Task3C_test.csv.

See Jupyter Notebook.

II. Train an autoencoder (similar to Activity 5.2) with only one hidden layer and change the number of its neurons to: 20, 40, 60, 80, .., 500 (i.e. from 20 to 500 with a step size of 20).

See Jupyter Notebook.

III. For each model in Step II, calculate and record the reconstruction error which is simply the average (over all data points while the model is fixed) of Euclidian distances between the input and output of the autoencoder (you can simply use "h2o.anomaly()" function). Plot these values where the x-axis is the number of units in the middle layer and the y-axis is the reconstruction error. Then, save and attach the plot to your report. Explain your findings based on the plot in your report.



Seems after a threshold value of K, the error seems to drop and stabilise. Not as expected where K increases, less reconstruction error would be observed.

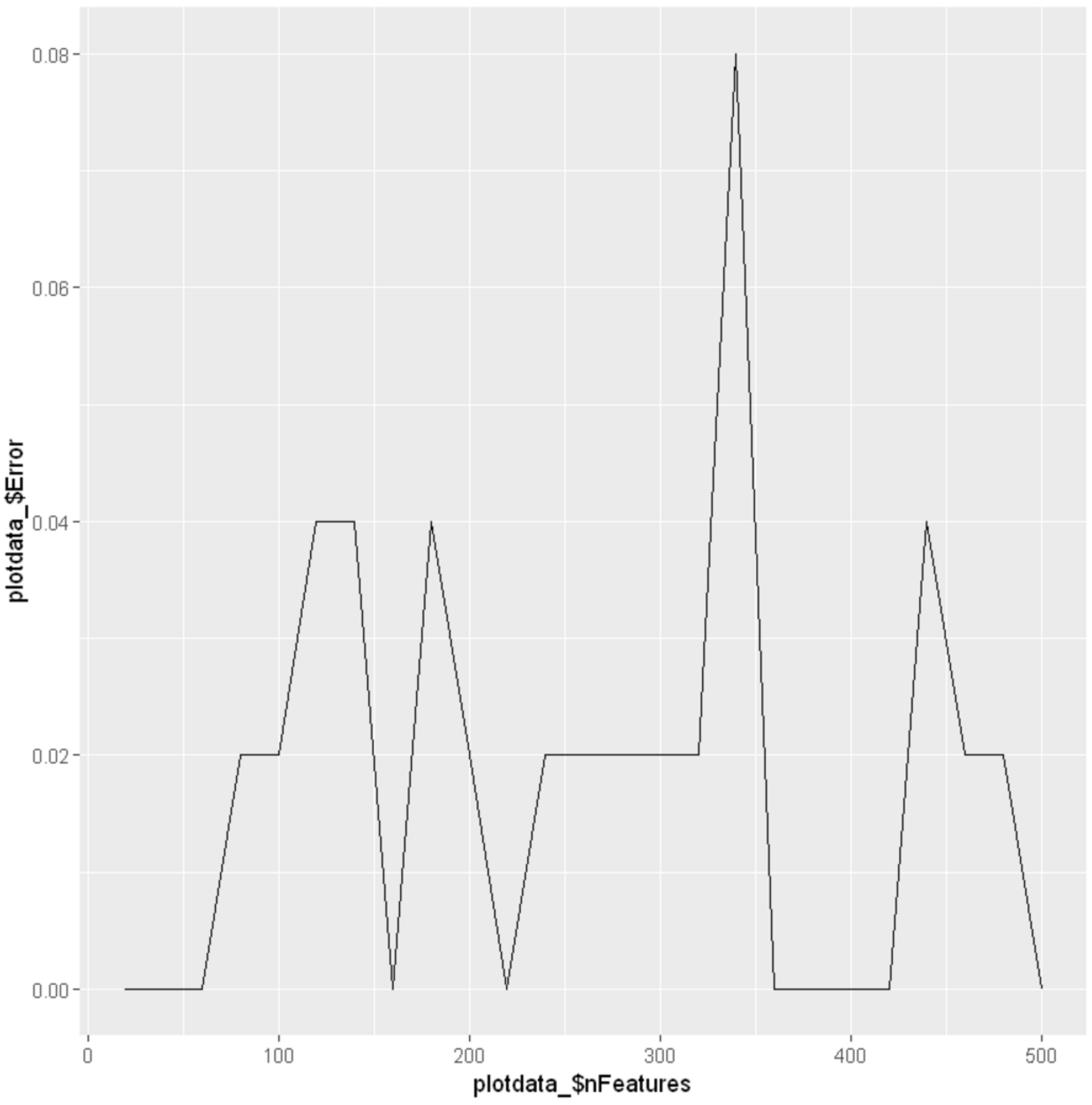
IV. Use the 3-layer NN from Activity 3.1 or "h2o.deeplearning" function (make sure you set " autoencoder = FALSE") to build a model with 100 units in the hidden layer using all the original attributes from the training set. Then, calculate and record the test error.

See Jupyter Notebook; Test error recovered : 0.344

V. Build augmented self-taught networks using the models learnt in Step II. For each model: A. Add the output of the middle layer as extra features to the original feature set, B. Train a 3-layer NN (similar to Step IV) using all features (original + extra). Then calculate and record the test error.

See Jupyter Notebook.

VI. Plot the error rates for the 3-layer neural networks from Step IV and V while the x-axis is the number of features and y-axis is the classification error. Save and attach the plot to your report.



VII. Report the optimum number(s) of units in the middle layer of the autoencoder in terms of the reconstruction and misclassification errors.

Obviously this plot is incorrect, but I would expect as the number of features increase, the classification error would decrease due to the increasing amount of information passed into the classifier.