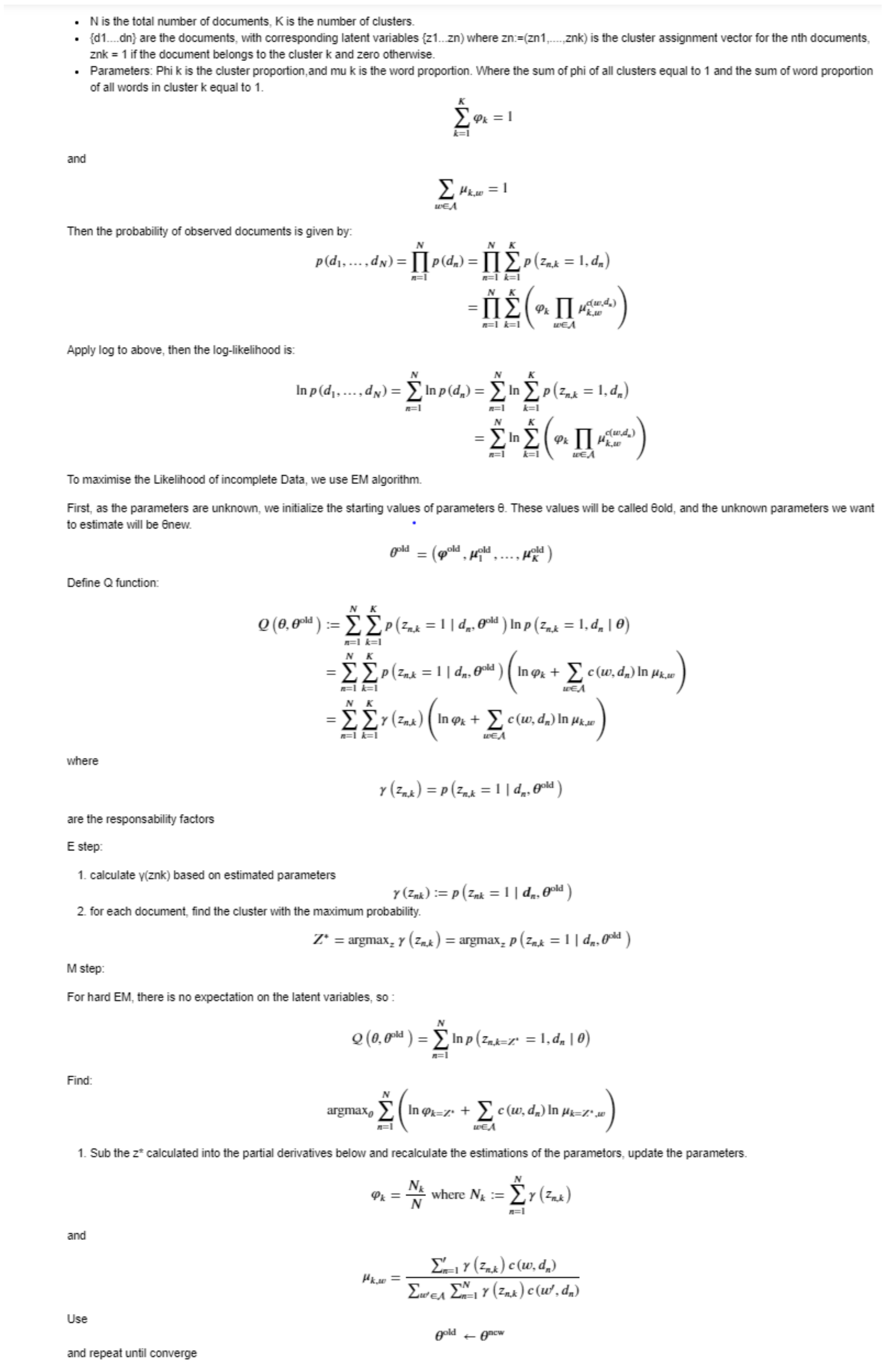
Part A. Document Clustering

Question1.

1. Derive Expectation and Maximization steps of the hard-EM algorithm for Document Clustering



- N is the total number of documents, K is the number of clusters.
- {d1....dn} are the documents, with corresponding latent variables {z1...zn) where zn:=(zn1,....,znk) is the cluster assignment vector for the nth documents, znk = 1 if the document belongs to the cluster k and zero otherwise.
- Parameters: Phi k is the cluster proportion,and mu k is the word proportion. Where the sum of phi of all clusters equal to 1 and the sum of word proportion of all words in cluster k equal to 1.

$$\sum_{k=1}^{K} \varphi_k = 1$$

and

$$\sum_{w \in \mathcal{A}} \mu_{k,w} = 1$$

Then the probability of observed documents is given by:

$$\begin{aligned} p(d_1, \ldots, d_N) &= \prod_{n=1}^{N} p(d_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} p\left(z_{n,k} = 1, d_n\right) \\ &= \prod_{n=1}^{N} \sum_{k=1}^{K} \left(\varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d_n)} \right) \end{aligned}$$

Apply log to above, then the log-likelihood is:

$$\begin{aligned} \ln p(d_1, \ldots, d_N) &= \sum_{n=1}^{N} \ln p(d_n) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} p\left(z_{n,k} = 1, d_n\right) \\ &= \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \left(\varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d_n)} \right) \end{aligned}$$

To maximise the Likelihood of incomplete Data, we use EM algorithm.

First, as the parameters are unknown, we initialize the starting values of parameters θ. These values will be called θold, and the unknown parameters we want to estimate will be θnew.

$$\theta^{\text{old}} = \left(\boldsymbol{\varphi}^{\text{old}}, \boldsymbol{\mu}_1^{\text{old}}, \ldots, \boldsymbol{\mu}_K^{\text{old}}\right)$$

Define Q function:

$$\begin{aligned} Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right) &:= \sum_{n=1}^{N} \sum_{k=1}^{K} p\left(z_{n,k} = 1 \mid d_n, \boldsymbol{\theta}^{\text{old}}\right) \ln p\left(z_{n,k} = 1, d_n \mid \boldsymbol{\theta}\right) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} p\left(z_{n,k} = 1 \mid d_n, \boldsymbol{\theta}^{\text{old}}\right) \left(\ln \varphi_k + \sum_{w \in \mathcal{A}} c(w, d_n) \ln \mu_{k,w} \right) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma\left(z_{n,k}\right) \left(\ln \varphi_k + \sum_{w \in \mathcal{A}} c(w, d_n) \ln \mu_{k,w} \right) \end{aligned}$$

where

$$\gamma\left(z_{n,k}\right) = p\left(z_{n,k} = 1 \mid d_n, \boldsymbol{\theta}^{\text{old}}\right)$$

are the responsability factors

E step:

1. calculate γ(znk) based on estimated parameters

$$\gamma\left(z_{nk}\right) := p\left(z_{nk} = 1 \mid \boldsymbol{d}_n, \boldsymbol{\theta}^{\text{old}}\right)$$

2. for each document, find the cluster with the maximum probability.

$$Z^* = \text{argmax}_z\, \gamma\left(z_{n,k}\right) = \text{argmax}_z\, p\left(z_{n,k} = 1 \mid d_n, \theta^{\text{old}}\right)$$

M step:

For hard EM, there is no expectation on the latent variables, so :

$$Q\left(\theta, \theta^{\text{old}}\right) = \sum_{n=1}^{N} \ln p\left(z_{n,k=Z^*} = 1, d_n \mid \theta\right)$$

Find:

$$\text{argmax}_\theta \sum_{n=1}^{N} \left(\ln \varphi_{k=Z^*} + \sum_{w \in \mathcal{A}} c(w, d_n) \ln \mu_{k=Z^*,w} \right)$$

1. Sub the z* calculated into the partial derivatives below and recalculate the estimations of the parametors, update the parameters.

$$\varphi_k = \frac{N_k}{N} \text{ where } N_k := \sum_{n=1}^{N} \gamma\left(z_{n,k}\right)$$

and

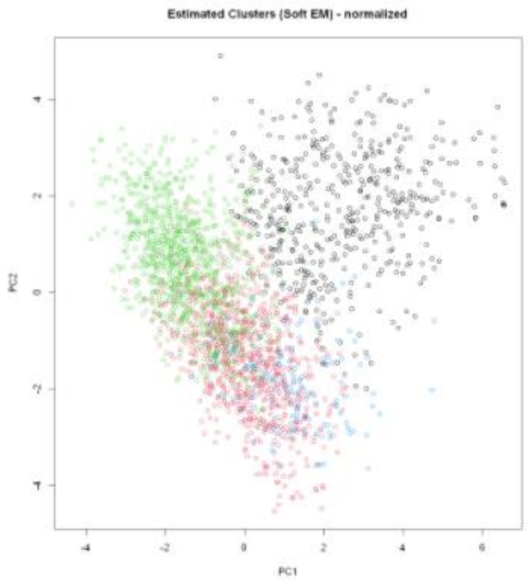
$$\mu_{k,w} = \frac{\sum_{n=1}^{\prime} \gamma\left(z_{n,k}\right) c(w, d_n)}{\sum_{w' \in \mathcal{A}} \sum_{n=1}^{N} \gamma\left(z_{n,k}\right) c(w', d_n)}$$

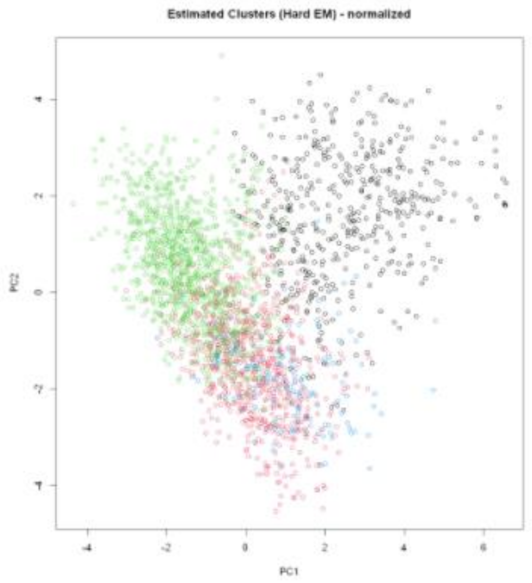
Use

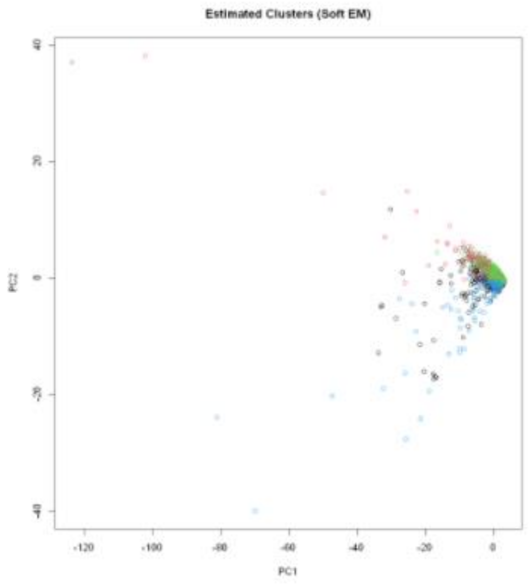
$$\theta^{\text{old}} \leftarrow \theta^{\text{new}}$$

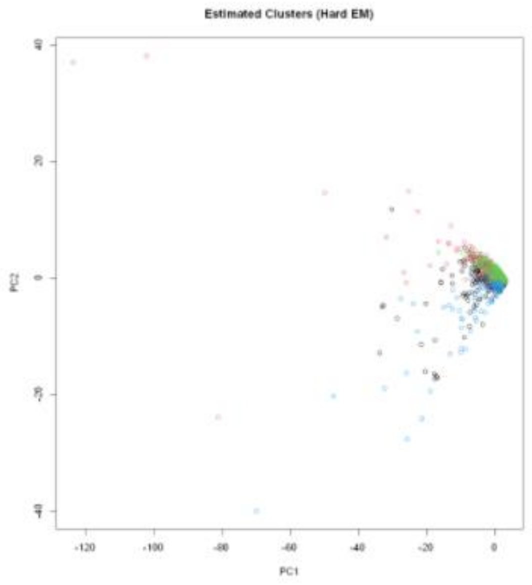
and repeat until converge

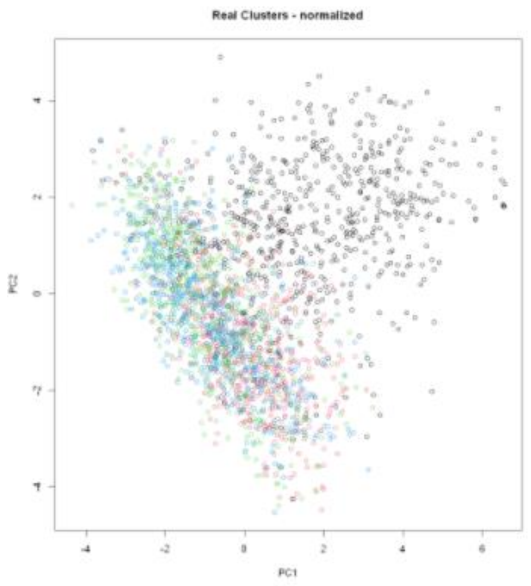
4. How and why the hard and soft-EM are different

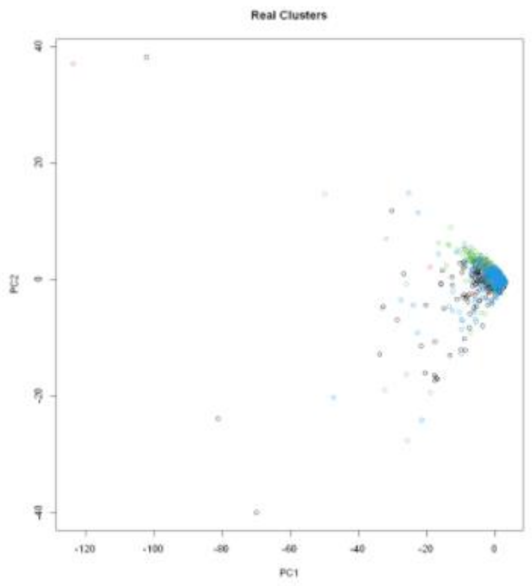












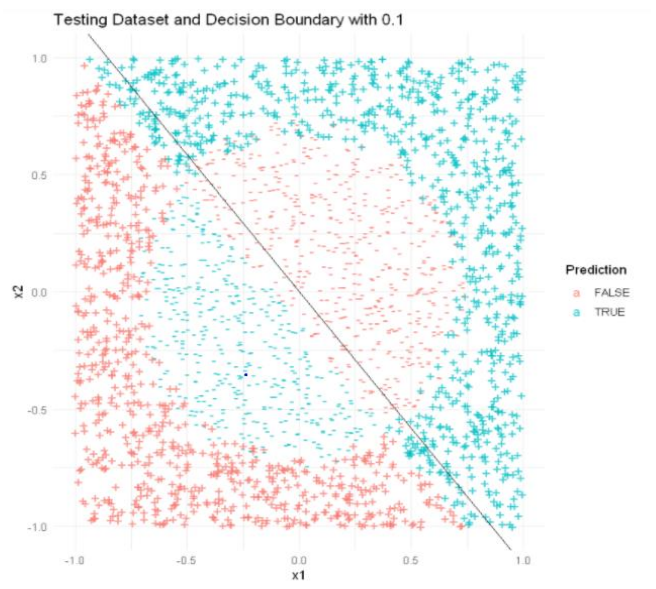
The clusters on the plots without normalization are hard to identify as the points are crowd in one area. Let's look at the graphs produced by normalized results.

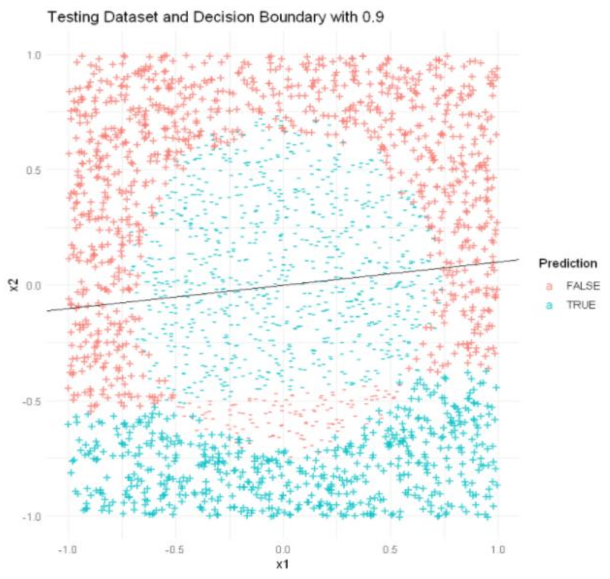
Comparing the clustering results of the soft and hard EM, it is hard to identify any clear differences between the two plots. They all have one clear cluster on the right and three clusters mixed together on the left. Green cluster appears to dominate the top left area, where the other two clusters mixed together at the lower area on the left. Usually, we would expect that in contrast with soft assignments, hard assignments give a more clear picture of clustering. As in hard assignments, each document is only assigned to one cluster.

Compare the two type of cluster assignments to real clusters, both soft and hard assignments seems to fail to estimate the clusters on the left side accurately. Where for the left side of the graph, the three clusters of documents are mixed together, but both our soft and hard assignments indicates that green cluster dominates the top left side. And, due to the fact that the real clusters are mixed together, it is extremely hard for our algorithm to estimate the clusters correctly.

Part B. Neural Network vs. Perceptron

2. Plot of the perceptron model





Based on test error, perceptron with learning rate of 0.9 is a better model in this case.

For the first model with η = 0.1, after running the code a few times, it appears that the model always only have a test error rate around 0.5 where the decision boundary always appears to be a diagonal.

The second model with η = 0.9, although still have a linear decision boundary(as we are using perceptron), it has a better performance on estimation on labels and the testing errors. Where the estimations on label and decision boundary seems to make more sense compare to the previous model.

Plot of the best perceptron model (0.9): with points coloured with their estimated class labels
Labels: ‘-’: -1 (0) , ‘+’:1

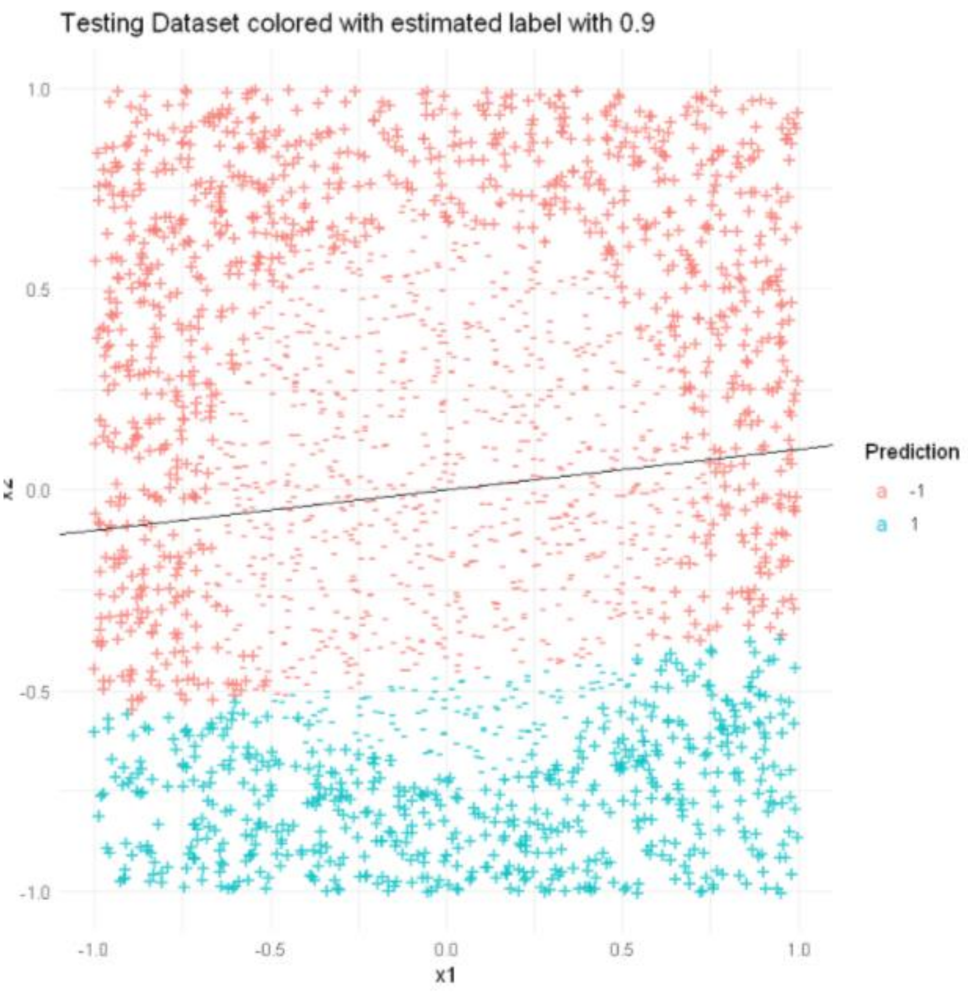
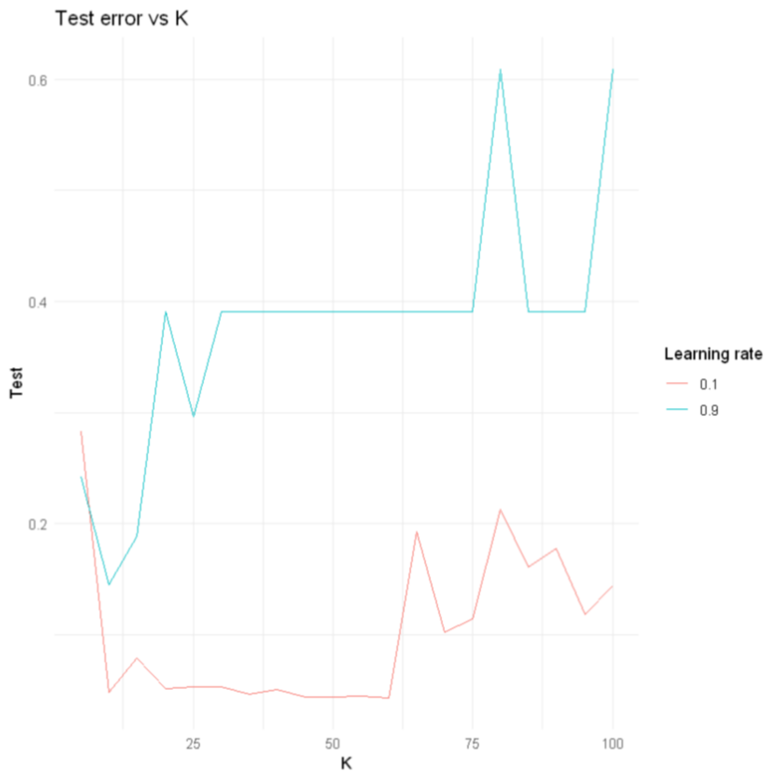


Figure 1. Perceptron

3. Plot the error for μ 0.01 and 0.09 vs K



The plot above illustrates that learning rate of 0.1 with K = 60 is the best model according to test error.

Plot the test data of the best model

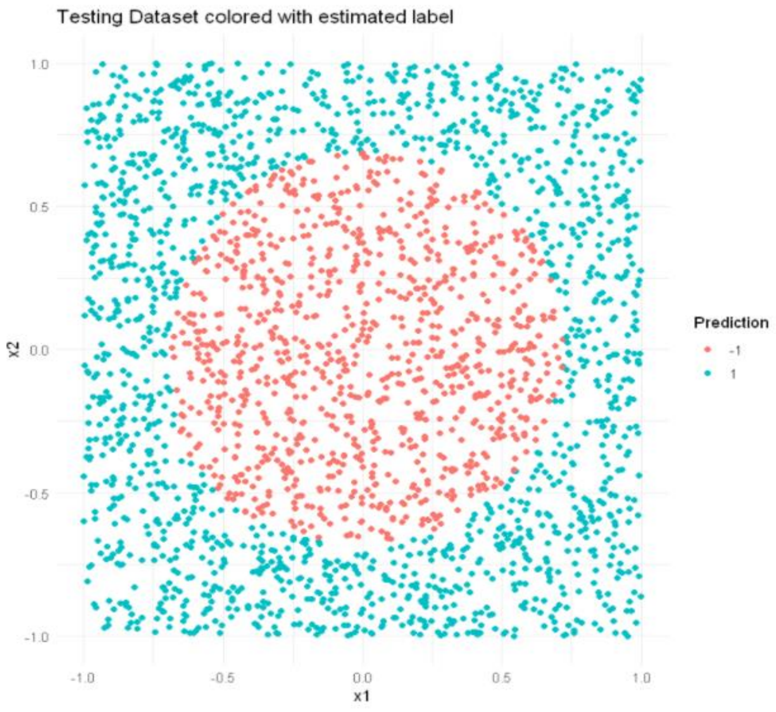


Figure 2. 3 layer NN

4. Difference between perceptron and a 3-layer NN

   The perceptron perform poorly compare to 3-layer NN in this case. As perceptron works as a linear binary classifier, it is suitable for this task, however, the shape of the two cluster are not two separate group that can be easily classified under a linear boundary concept. As we can see in figure 1, the results from the best perceptron is unable to find the real boundary of the two clusters. It can only determines the labels based on the linear boundary, which is insufficient in this case. Thus, the single layer network perceptron is not able to successfully cluster the data points into the real clusters as a single layer network can only learn linear functions.. On the other hand, the three layer neural network is able to learn non-linear functions as it has extra layers. Which in this case, the boundary is non-linear, it is necessary for our model to be able to learn non-linear functions to be able to produce accurate and sensible prediction. In figure 2, the 3-layer NN with 60 units produces accurate predictions on labels and successfully identify the non-linear boundary of the two cluster.

Part C. Self-Taught Learning

3. plot units vs reconstruction error from autoencoder

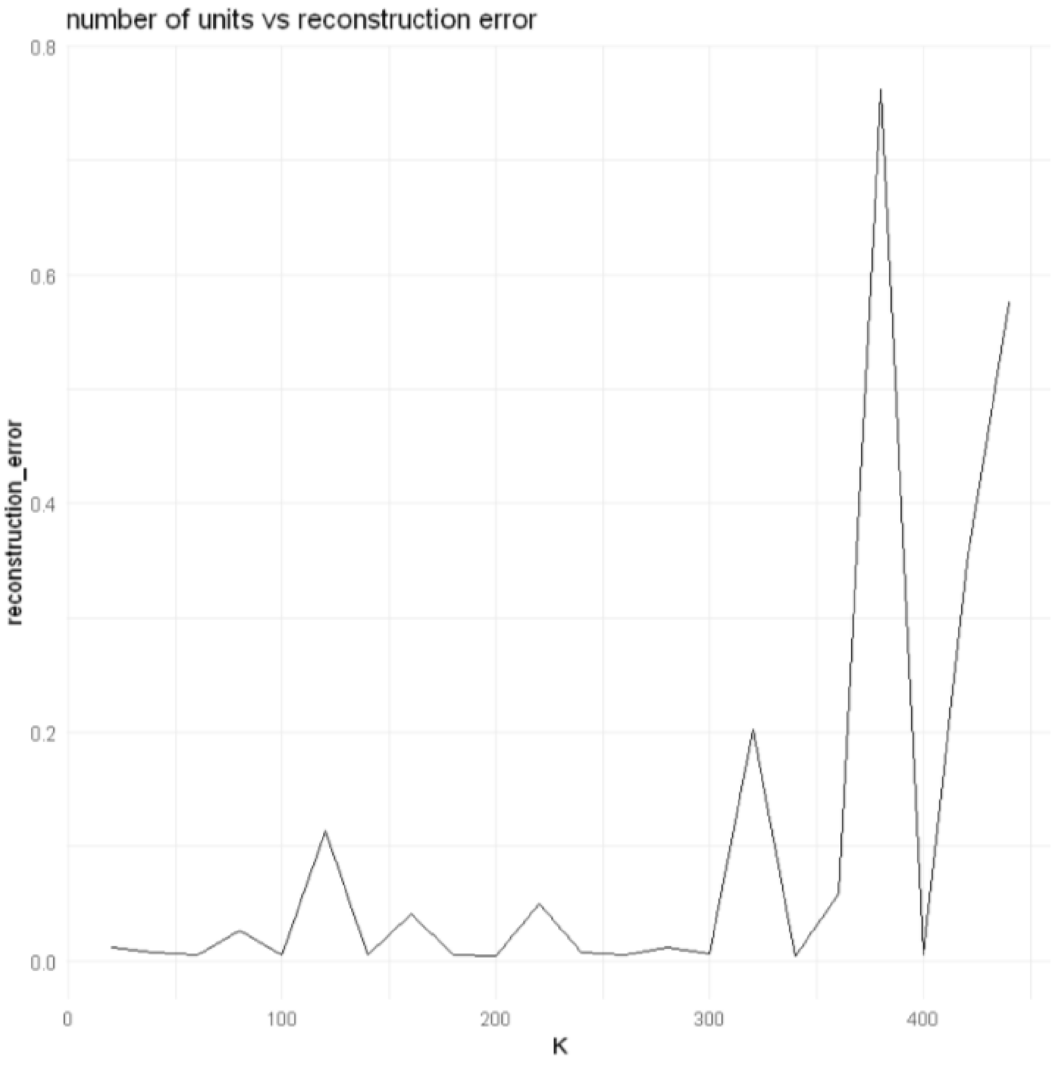
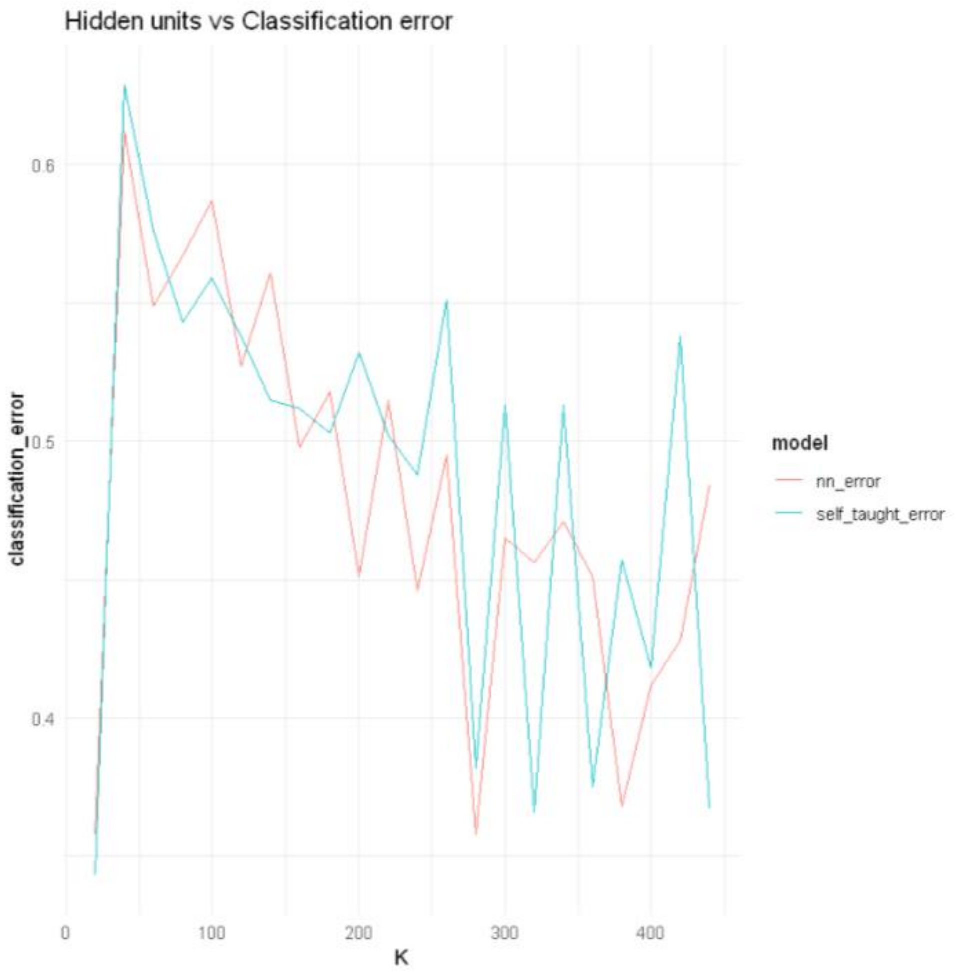


Figure 3. Reconstruction error of autoencoder

From figure 3 above, we can observe that when the number of units in the hidden layers are low, the reconstruction error is more likely to be low. As units increases, the variation in the error also increases. This illustrates that when the units are small, the features extracted or generated in the middle player are likely to be more useful and produce more accurate estimates of the original inputs in the decode process.

6. 3-layer neural networks and the augmented self-taught networks



Based on the graph above, we may not see a clear difference between the test errors of two networks. However, in some cases, there is a large difference between the test errors of the two model, where the classification error of the self-taught networks are lower than the 3 layer neural networks. Also, note that the lowest test error is produced by the self-taught networks at k = 20. Thus, we can say that self-taught networks perform slightly better than the 3 layer neural network. This may due to the fact of the extra features learned from autoencoder boosted the performance. Compare to 3 layer neural networks which only use the original features, the self-taught networks combines features extracted from the unlabelled data and try to apply them to achieve a higher accuracy. Nevertheless, based on the graph and test errors, the extra features have impact the results leads to an overall better performance for self-taught networks, the variation in test errors seems also increases. This may also caused by the extra features, not all features included can be helpful, as some may also leads to worse performance in specific cases.