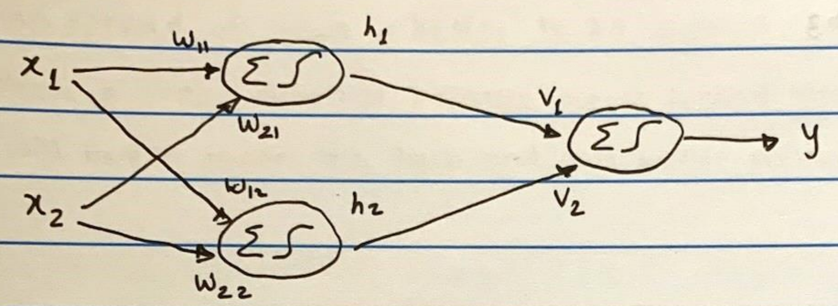
1(a).

1.5/2

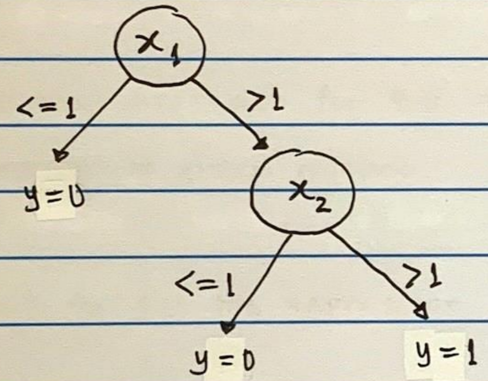


(b). This is because the output y is continuous value $[0, 1]$ whereas tanh function only ~~gives~~ returns the values -1 and 1.

0.5/1

(c).

3/3



(d). For $x_1 = (2,2)^T$, $h_1 = \tanh(2(100) + 2(0) - 100) = 1$

$h_2 = \tanh(2(0) + 2(100) - 100) = 1$

$y = \sigma(1(100) + 1(100) - 100) = \frac{1}{1+e^{-100}} = 1 > 0.5$, hence classified as (+) class

2/3

For $x_2 = (-2,2)^T$, $h_1 = \tanh(-2(100) + 2(0) - 100) = -1$

$h_2 = \tanh(-2(0) + 2(100) - 100) = 1$

$y = \sigma(-1(100) + 1(100) - 100) = \frac{1}{1+e^{100}} = 3.72 \times 10^{-44} \leq 0.5$, hence classified as (−) class

Therefore...

For x_1, $y = 1$

For x_2, $y = 0$

This gives same predictions as decision tree.

2 (a). The spread of news is better to be modelled as an exponential ~~than~~ function than a linear function because mews spread through 'word of mouth'. Hence will better model the task and give better predictions.

1/2

(b). The gradient descent update rule is updating the weight vector w,

$$w \leftarrow w - \eta g$$

1/2

where η is the learning rate

and g is the gradient of E given w, $\frac{\partial E}{\partial w}$

It is a good approach for this model because the exponential function has one unique global minima.

(c). We want to get the expression $\frac{\partial E}{\partial w}$

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

3/3

$$= \sum_{i=1}^{n} \frac{\partial}{\partial w_j} (y_i - f(x_i))^2$$

$$= \sum_{i=1}^{n} 2(y_i - f(x_i)) \frac{\partial}{\partial w_j} (y_i - f(x_i))$$

$$= 2\sum_{i=1}^{n} (y_i - f(x_i)) \left(\frac{\partial}{\partial w_j} y_i - \frac{\partial}{\partial w_j} f(x_i) \right)$$

$$= 2 \sum_{i=1}^{n} (y_i - e^{wx})(we^{wx})$$

2(d).

$$\frac{\partial E}{\partial w_0} = 2[(2-e^1)(e^1) + (3.25-e^1)(e^1) + (11-e^1)(e^1)]$$

$$= 2[(2-e)e + (3.25-e)e + (11-e)e] = 44$$

1/3

$$\frac{\partial E}{\partial w_1} = 2[(2-e^{0.5})(e^{0.5}) + (3.25-e^1)(e^1) + (11-e^2)(e^2)]$$

$$= 57.41$$

$$W = W - \eta g = 1 - (0.001)(57.41)$$

$$= 0.94$$

2(e)

0/2

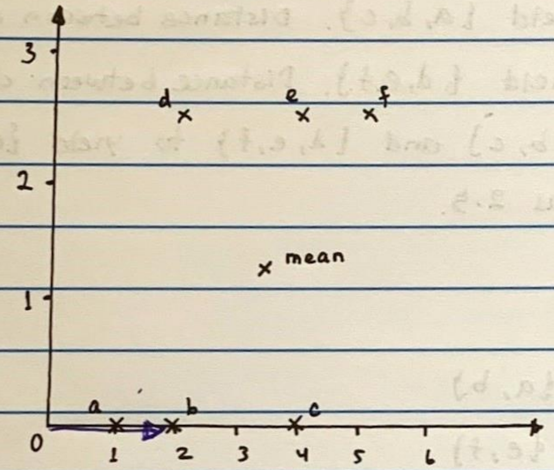
2(f). Gradient descent is computationally expensive method as we take small steps downhill. This is a problem with large datasets.

0/1

3. (a). mean $\mu = \left[\begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}2\\0\end{pmatrix} + \begin{pmatrix}4\\0\end{pmatrix} + \begin{pmatrix}2\\2.5\end{pmatrix} + \begin{pmatrix}4\\2.5\end{pmatrix} + \begin{pmatrix}5\\2.5\end{pmatrix}\right] \times \frac{1}{6}$

$= \frac{1}{6}\begin{pmatrix}18\\7.5\end{pmatrix} = \begin{pmatrix}3\\1.25\end{pmatrix}$

1.5/3



(b) For first iteration, calculate Euclidean distance from m_1 and m_2 to each data point; alternatively, we can visually see from the plot above that we can assign the following points to clusters m_1 and m_2:

$m_1 = [d, e, f]$ and $m_2 = [a, b, c]$

4/4

The new means of each centroids will be:

$m_1 = \frac{1}{3}\left[\begin{pmatrix}2\\2.5\end{pmatrix} + \begin{pmatrix}4\\2.5\end{pmatrix} + \begin{pmatrix}5\\2.5\end{pmatrix}\right] = \begin{pmatrix}11/3\\5/2\end{pmatrix}$

$m_2 = \frac{1}{3}\left[\begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}2\\0\end{pmatrix} + \begin{pmatrix}4\\0\end{pmatrix}\right] = \begin{pmatrix}7/3\\0\end{pmatrix}$

Repeat for second iteration. Again, we get the following assignments:

$m_1 = [d, e, f]$ and $m_2 = [a, b, c]$

The algorithm terminates since there are no centroid changes.

$\therefore$ In cluster m_1, we have instances d, e, f

In cluster m_2, we have instances a, b, c

$m_1 = \begin{pmatrix}11/3\\5/2\end{pmatrix}$ $m_2 = \begin{pmatrix}7/3\\0\end{pmatrix}$

Q3

3/3

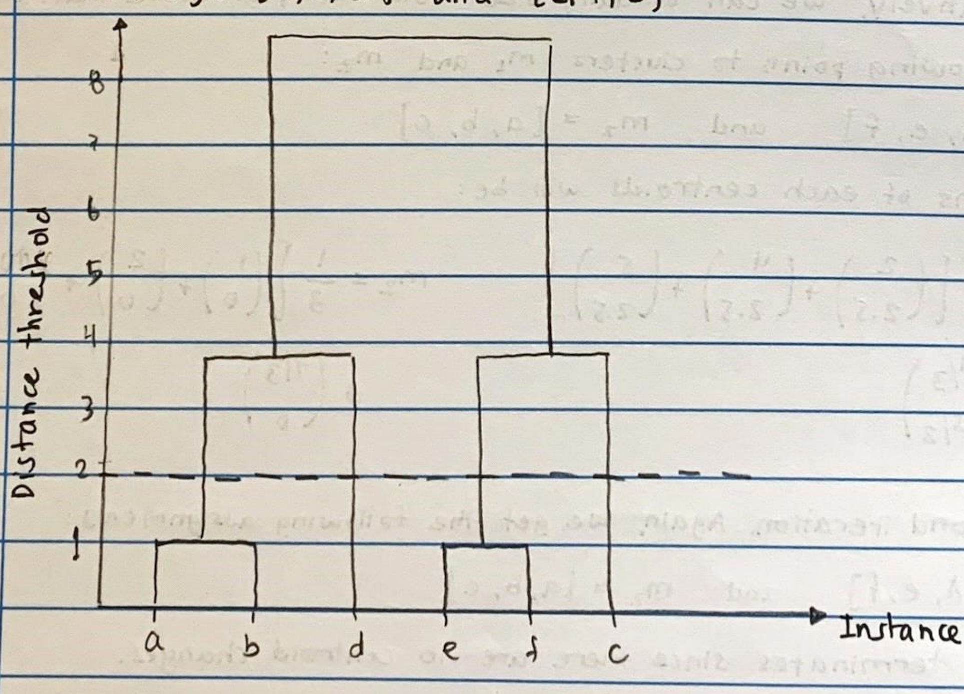
3(c). In single-link, merge clusters that are closest in distance between their 2 closest points.

1) Merge a and b to yield {a, b}
2) Merge e and f to yield {e, f}
3) Merge c to {a, b} to yield {a, b, c}. Distance between clusters is 2.
4) Merge d to {e, f} to yield {d, e, f}. Distance between clusters is 2.
5) Merge the two clusters {a, b, c} and {d, e, f} to yield {a, b, c, d, e, f}. Distance between clusters was 2.5.

3/4

3(d). In complete-link,

1) Merge a and b to yield {a, b}
2) Merge e and f to yield {e, f}
3) Merge d to {a, b} to yield {a, b, d} because distance between farthest points a and d is 2.7.
4) Merge c to {e, f} to yield {e, f, c}. Distance between clusters is 2.7.
5) Merge {a, b, d} and {e, f, c}



Hence, the remaining clusters are {a, b}, {d}, {e, f}, {c}.

Q4

4(a). I would check for missing values and possible outlier as these will likely affect predictions.

0.5/3

4(b). To give an unbiased estimate of the performance of each method, I would set aside a part of training data to be my testing set and learn the models without using any of the test set. I ~~then~~ will then calculate the testing error.

1.5/2

4(c). No. Having too many hidden units may give low training error but high generalization error because of overfitting

2/2

4(d). I think k-nearest-neighbours would handle this better. k-NN gives non-linear decision boundary while log regression gives linear boundaries.

0/2

Another classifier could be a Naive Bayes classifier as it handles missing values well.

2/2

4(e). Disparate impact is when a decision disproportionately harms people with certain protected attributes.
I would compare ROC curves for the different values of the attribute A and analyze how they differ.