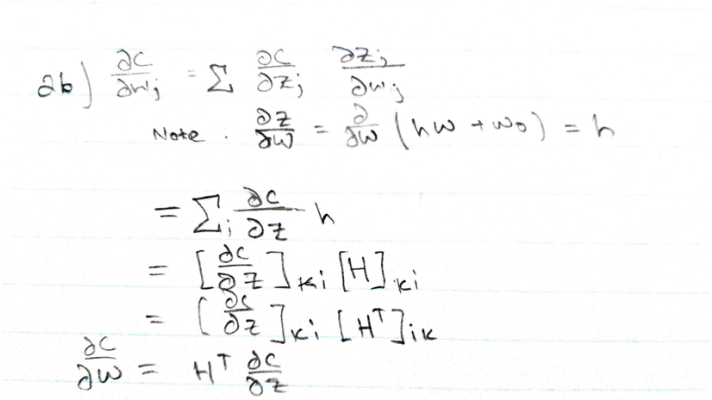
1.
B) With only a single hidden unit, the neural network produced a result which is logistic regression with a multiclassifer. As seen in previous assignments, this ultimately fits a straight line through the data.

F) The function with 4 or less hidden units in one layer lacks the complexity to fit the data to receive the results of Gaussian Bayes Discriminant Analysis. For similar results we would need to increase the complexity by adding more hidden layers and/or more hidden units.

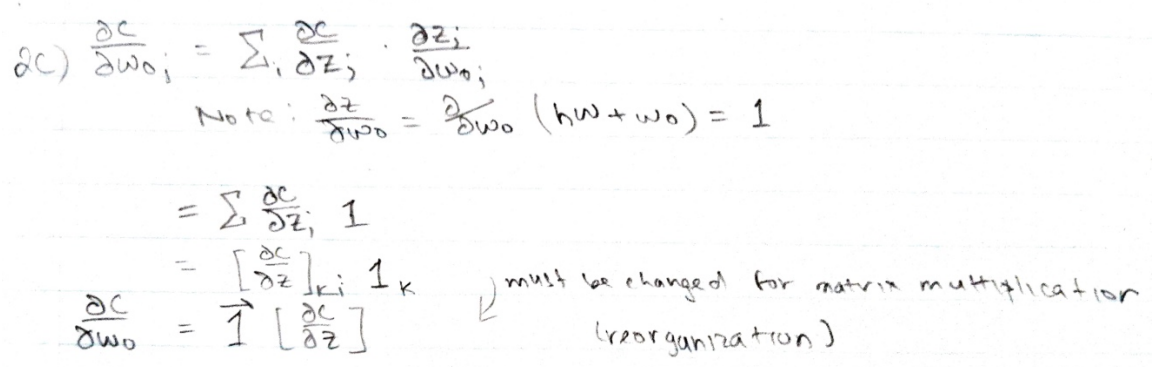
G) The various different boundaries are due to the random initialization in weights and the shape of the data. Additionally, the decision boundaries are slightly different, but nonetheless they all classify about the same points in the region.

2.
A) I don't know

B)



C)



D)

2d) $\frac{\partial C}{\partial h_k} = \sum_i \frac{\partial C}{\partial z_i} \frac{\partial z_i}{\partial h_k}$

Note: $\frac{\partial z_i}{\partial h} = \frac{\partial}{\partial h}(hW + w_0) = W$

$= \sum_i \frac{\partial C}{\partial z_i} w_i$

$\partial C / \partial H = \frac{\partial C}{\partial Z} W^T$ ↙ (Transpose due to W being (hidden units, output nums) and $\frac{\partial C}{\partial Z}$ being the same shape as well)

E)

2e) $\frac{\partial C}{\partial u} = \sum_n \frac{\partial C}{\partial h_k} \frac{\partial h_k}{\partial u_k}$

Note: $\frac{\partial h_k}{\partial u} = (h_k)(1-h_k)$

$= \sum_n \frac{\partial e}{\partial h_k} [(h_k)(1-h_k)]$

$= \left[\frac{\partial C}{\partial h}\right]_{nk} [h_k (1-h_k)]_{nk}$

$\left[\frac{\partial C}{\partial u}\right]_{nk} = (H_{nk})(1-H_{nk}) \left[\frac{\partial C}{\partial H}\right]_{nk}$

F)

F) $\frac{\partial C}{\partial v_n} = \sum_m \frac{\partial C}{\partial u_k} \frac{\partial u_k}{\partial v_n}$

⇒ Note $\frac{\partial u}{\partial v} = x$

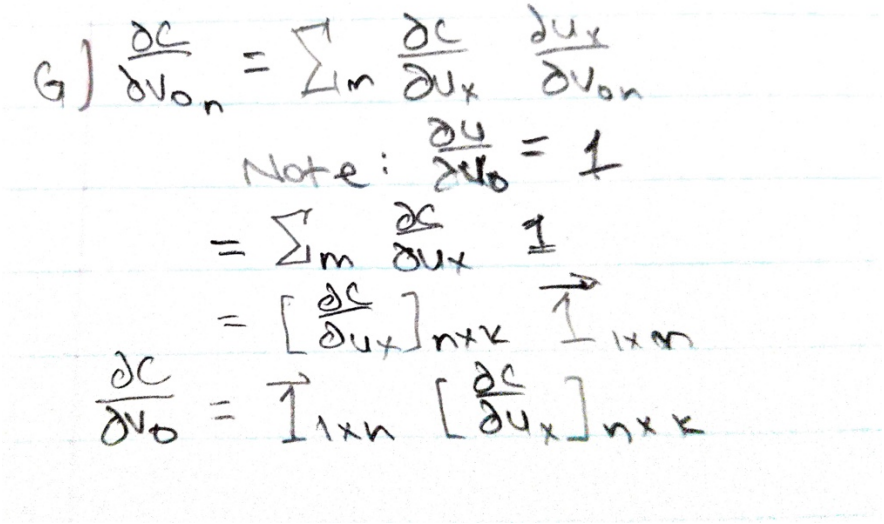
$= \sum_m \frac{\partial C}{\partial u_k} x_n$

$= \left[\frac{\partial C}{\partial u}\right]_{nxk} X_{nxm}$ ↙ (matrix transpose property)

$= \left[\frac{\partial C}{\partial u}\right]_{nxk} [X^T]_{mxn}$ ↙ [reorganize]

$\frac{\partial C}{\partial V} = [X^T]_{mxn} \left[\frac{\partial C}{\partial u}\right]_{nxk}$

G)



H) dCdw0 = np.sum(dCdZ, axis=0).reshape(1,-1)

3.

B) This is because stochastic gradient descent does more iterations of learning in our example (i.e. calculate the gradient/fix the cost more times although same number of epochs).

D) This is due to the learning rate being standardized based on the batch size. Thus, it doesn't matter how much big your dataset is, the learning rate will adjust accordingly.