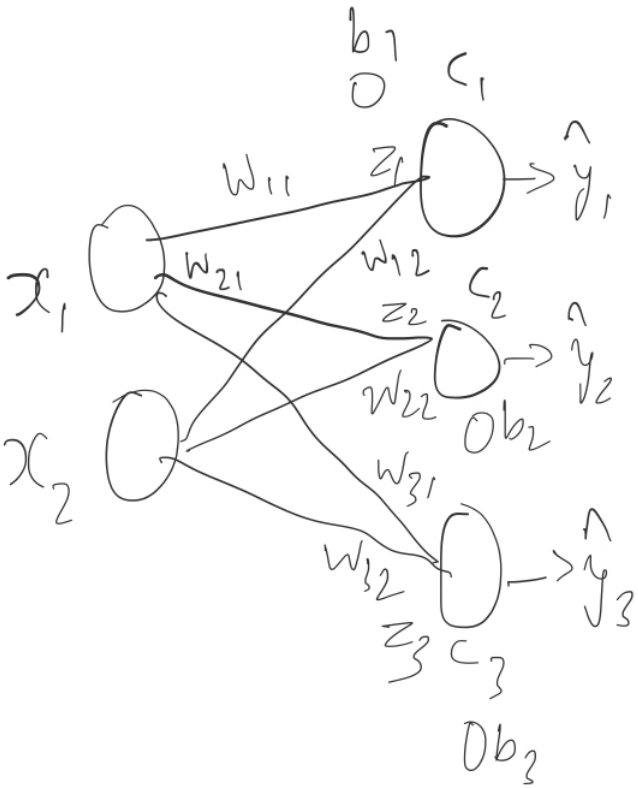
Assignment 1

Problem 1

0.1 Q1



Ans 1: There are total of 9 parameters, 6 w and 3 b. The given input data is 2D and there are 3 classes. We have to train a logistic regression model. So above is the model that i am training.:

Ans 2: The loss for a single sample is given by $L = -(y_1 Log(\hat{y}_1) + y_2 Log(\hat{y}_2) + y_3 Log(\hat{y}_3))$, the same can be extended to any number of samples, it will simply add.
$\hat{y}_i = \sigma(z_i)$ (softmax)
$z_i = w_{i1}x_1 + w_{i2}x_2 + b_i$
$L = -(y_1 Log(\sigma(w_{11}x_1 + w_{12}x_2 + b_1)) + y_2 Log(\sigma(w_{21}x_1 + w_{22}x_2 + b_2)) + y_3 Log(\sigma(w_{31}x_1 + w_{32}x_2 + b_3)))$

Ans 3: Let $W_i = [W_{i1}, W_{i,2}], B_i, for$ i=1 to 3
For a single sample, the loss is given by below equation, the same can be easily extended for multiple samples.
$L = -(y_1 Log(\hat{y}_1) + y_2 Log(\hat{y}_2) + y_3 Log(\hat{y}_3))$
$L = -\Sigma_{j=1}^{j=3} y_i Log(\hat{y}_i)$
$\hat{y}_i = \sigma(z_i)$
$z_i = w_{i1}x_1 + w_{i2}x_2 + b_i$
$\frac{\partial L}{\partial w_{i_1}} = \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial w_{i_1}}$
Now i will calculate each of the above partial derivatives separately.

$\frac{\partial L}{\partial z_i} = -\Sigma_{j=1}^{3} \frac{y_j}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_i}$

There are two cases where i=j, i $\neq$ j. The partial derivatives for both cases will be different. First calculating for the case i=j.

$\frac{\partial \hat{y}_i}{\partial z_i} = \frac{\partial \frac{e^{z_i}}{\Sigma_j e^{z_j}}}{\partial z_i} = \frac{e^{z_i}(\Sigma_j e^{z_j}) - e^{z_i} e^{z_i}}{(\Sigma_j e^{z_j})^2} = \hat{y}_i(1 - \hat{y}_i)$

Now solving i≠j.

$\frac{\partial \hat{y}_j}{\partial z_i} = \frac{\partial \frac{e^{z_j}}{\Sigma_l e^{z_l}}}{\partial z_i} = -e^{z_j} \frac{e^{z_i}}{(\Sigma_l e^{z_l})^2} = -\hat{y}_j \hat{y}_i$

Putting these values into above expression:

$\frac{\partial L}{\partial z_i} = -(\frac{y_i}{\hat{y}_i}(\hat{y}_i)(1 - \hat{y}_i) + \Sigma_{i \neq j}^{3} \frac{y_j}{\hat{y}_j}(-\hat{y}_j \hat{y}_i))$

$\frac{\partial L}{\partial z_i} = -(y_i(1 - \hat{y}_i) + \Sigma_{i \neq j}^{3} y_j(-\hat{y}_i))$

$\frac{\partial L}{\partial z_i} = -(y_i - y_i \hat{y}_i \Sigma_{i \neq j}^{3} y_j(-\hat{y}_i))$

$\frac{\partial L}{\partial z_i} = -(y_i - y_i \Sigma_{j=1}^{3} y_j(\hat{y}_i))$

$\frac{\partial L}{\partial z_i} = -(y_i - \hat{y}_i \Sigma_{j=1}^{3} y_j)$

$\frac{\partial L}{\partial z_i} = -(y_i - \hat{y}_i)$

$\frac{\partial z_i}{\partial \hat{w}_i} = x_1$

$\frac{\partial L}{\partial w_{i1}} = (y_i - \hat{y}_i)x_1$

$\frac{\partial L}{\partial \hat{w}_i} = \begin{bmatrix} -\Sigma_{n=1}^{n=3}(y_{i,n} - \hat{y}_{i,n})x_{1,n} \\ -\Sigma_{n=1}^{n=3}(y_{i,n} - \hat{y}_{i,n})x_{2,n} \end{bmatrix}$ eq1

$\frac{\partial L}{\partial b_i} = -\Sigma_{n=1}^{n=3}(y_{i,n} - \hat{y}_{i,n})$ eq2

Let each y be one hot encoded, so if x belongs to class 0, then $y = [1, 0, 0]$ and $y_1 = 1, y_2 = 0, y_3 = 0$as there are three classes. $\frac{\partial L}{\partial w_1} = \begin{bmatrix} -(0 - \hat{y}_{1,1})5 - (1 - \hat{y}_{1,2})40 - (0 - \hat{y}_{1,3})10 \\ -(0 - \hat{y}_{1,1})10 - (1 - \hat{y}_{1,2})(-9) - (0 - \hat{y}_{1,3})2 \end{bmatrix}$

$\frac{\partial L}{\partial b_1} = -(0 - \hat{y}_{1,1}) - (1 - \hat{y}_{1,2}) - (0 - \hat{y}_{1,3})$

$\frac{\partial L}{\partial w_2} = \begin{bmatrix} -(1 - \hat{y}_{2,1})5 - (0 - \hat{y}_{2,2})40 - (0 - \hat{y}_{2,3})10 \\ -(1 - \hat{y}_{2,1})10 - (0 - \hat{y}_{2,2})(-9) - (0 - \hat{y}_{2,3})2 \end{bmatrix}$

$\frac{\partial L}{\partial b_2} = -(1 - \hat{y}_{2,1}) - (0 - \hat{y}_{2,2}) - (0 - \hat{y}_{2,3})$

$\frac{\partial L}{\partial w_3} = \begin{bmatrix} -(0 - \hat{y}_{3,1})5 - (0 - \hat{y}_{3,2})40 - (1 - \hat{y}_{3,3})10 \\ -(0 - \hat{y}_{3,1})10 - (0 - \hat{y}_{3,2})(-9) - (1 - \hat{y}_{3,3})2 \end{bmatrix}$

$\frac{\partial L}{\partial b_3} = -(0 - \hat{y}_{3,1}) - (0 - \hat{y}_{3,2}) - (1 - \hat{y}_{3,3})$

Ans 3:

$X \leftarrow [[5, 10], [40, -9], [10, 2]], Y \leftarrow [[0, 1, 0], [1, 0, 0], [0, 0, 1]], lr \leftarrow \eta$, $w_{i,j} \leftarrow$ random values in (0,1) where $i \epsilon (1, 3)$ and $j \epsilon (1, 2)$, $b_{i,j} \leftarrow$ random value in (0,1), epoch $\leftarrow say1000, it \leftarrow 0$

while $it \leq epoch$ **do**

Use the entire training data for each epoch. Calculate gradients as per below algorithm.

$i = 0$

while $i \leq 3$ **do**

$\frac{\partial L}{\partial w_i} =$ Using eq 1

$w_i = w_i$(value at previous time)$-\frac{\eta}{3} \frac{\partial L}{\partial w_i}$

$\frac{\partial L}{\partial b_i} =$ Using eq 2

$b_i = b_i$(value at previous time)$-\frac{\eta}{3} \frac{\partial L}{\partial b_i}$

end while

end while

Ans 4:

$X \leftarrow [[5, 10], [40, -9], [10, 2]], Y \leftarrow [[0, 1, 0], [1, 0, 0], [0, 0, 1]], lr \leftarrow \eta$, $w_{i,j} \leftarrow$ random values in (0,1) where $i \epsilon (1, 3)$ and $j \epsilon (1, 2)$, $b_{i,j} \leftarrow$ random value in (0,1), epoch $\leftarrow say1000, it \leftarrow 0$, beta1 $\leftarrow 0.9$, $beta2 \leftarrow 0.999$, $shuffledata \leftarrow$ list of integers from 0-2, $mtwi \leftarrow [[0], [0]], vtwi \leftarrow [[0], [0]]$,mtbi $\leftarrow [0], vtbi \leftarrow [0]$

▷ Each 0 in mt and vt are for one parameter each

while $e \leq epoch$ **do**
shuffle(shuffledata)
it=0

while $it \leq$len(shuffledata) **do**

▷ Use shuffledata[it] as the data and calculate gradient only on that point. So for equations 1-2, N will be equal to 1 instead of 3.

$i \leftarrow 0$

while $i \leq 3$ **do**
$\frac{\partial L}{\partial w_i}$ = Using eq 1
$\frac{\partial L}{\partial b_i}$ = Using eq 2
mtWi=beta1*mtWi+(1-beta1)$\frac{\partial L}{\partial w_i}$
vtWi=beta2*vtWi+(1-beta2)*$\frac{\partial L}{\partial w_i}^2$
mhatWi=mtWi/(1-beta1**(it+1))
vhatWi=vtWi/(1-beta2**(it+1))
mtbi=beta1*mtbi+(1-beta1)$\frac{\partial L}{\partial b_i}$
vtbi=beta2*vtbi+(1-beta2)*$\frac{\partial L}{\partial b_i}^2$
mhatbi=mtWi/(1-beta1**(it+1))
vhatbi=vtWi/(1-beta2**(it+1))

$w_i = w_i - \frac{\eta}{1} \frac{mhatWi}{(sqrt(vhatWi)+1e-8)}$
$b_i = b_i - \frac{\eta}{1} \frac{mhatbi}{sqrt(vhatbi)+1e-8)}$
it+=1
end while
end while
end while

Q2 Ans: When implementing all the different optimizers i observed that the step size plays a critical role. When I used a large step size, i the loss dropped very quickly close to zero but as the iterations progressed, there were small spikes in the loss graph. I think this is because large step size can lead to some oscillations around the local minima. When i used a smaller step size, the loss decreased gradually, but there were no spikes seen. I observed that the MSE is quite large because the data cannot be properly fit using a linear model, it requires a non linear model to properly fit the data. I also observed that by modifying the learning rate, i was able to get similar fitting lines for the different optimizers, however this will always not be the case because the adaptive learning rate optimizers will have a better performance. This is because when we are very close to the local minima, having large step sizes will lead to oscillatory behavior, but the adaptive learning rate strategy will help avoid this problem. I observed that ADAM and the full gradient descent have similar performance. However, since this was a small dataset, full gradient descent was feasible and gave good results, in practise ADAM would be better i think.

Q3 Ans: I tried using Logistic regression as the classifier for obtaining the double descent but it took a lot of simulation time and the runs crashed many times. So i chose SGDClassifier, the default loss in this classifier is "hinge loss" so basically i used a SVM classifier for obtaining the double descent. The maximum number of parameters was 30,000, and i observed that the loss curve started to rise towards the end of 30,000 features. However, i was not able to observe the double descent phenomenon, but i feel that on increasing the number of parameters, the double descent would be clearly visible. Since the execution always crashed on my local machine, i could not run that experiment. In my graph, the train error also starts increasing after a certain number of features, i think this is happening because we have a lot of features to train in our model and we are still using the same number of iterations for training hence the model is not getting sufficient time to train.

For fourier random features, i used the cauchy kernel whose fourier transform is the laplace. I used the laplace as the probability distribution and sampled points from that distribution. The first waveform in my notebook is when i used SGDClassifier and RFF, i think the double descent is visible there as well because i see the test accuracy

dropping and then increasing and decreasing again. However, this is not very clear because the scales are different. Then i ran the same experiment but using RidgeClassifier and plotted the test and train loss separately, and the double descent phenomenon can be clearly seen. I observed that even though the train accuracy was getting higher with the number of features, the test accuracy did not change significantly. It was very low compared to the previous case. I also observed that there were a lot of sudden spikes in the test loss compared to the example that was provided with the assignment. It proves that the quality of the features are very important int he performance of a model.