MACHINE LEARNING ASSIGNMENT 2

Mehdi Nikkhah (301324692)

Mehdi Nikkhah
mnikkhah@sfu.ca

Question1)

Answer the following questions. For 2 and 3, you may provide qualitative answers (i.e. no need to analyze limits).

1. (3 marks) What are the probabilities p(C_k|x) at the green point?

a. The probability of p(C_k|x) is defined with Softmax, so the sum of probability of three classes is 1.

$$P(C_k|x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \Rightarrow \sum_k P(C_k|x) = \frac{\sum_k \exp(a_k)}{\sum_j \exp(a_j)} = 1 \quad \text{eq.1}$$

b. At the intersection point (green point), the values of a_1, a_2, a_3 are equal. So, the probability of three classes are equal.

$$a_1=a_2=a_3 \Rightarrow \frac{\exp(a_1)}{\sum_j \exp(a_j)} = \frac{\exp(a_2)}{\sum_j \exp(a_j)} = \frac{\exp(a_3)}{\sum_j \exp(a_j)}$$

$$P(C_1|x) = P(C_2|x) = P(C_{k3}|x) \quad \text{eq.2}$$

$$\text{eq.1, eq.2} \Rightarrow \quad P(C_1|x) = P(C_2|x) = P(C_{k3}|x) = \frac{1}{3}$$

2. (3 marks) What happens to the probabilities along each of the red lines? What happens as we move along a red line (away from the green point)?

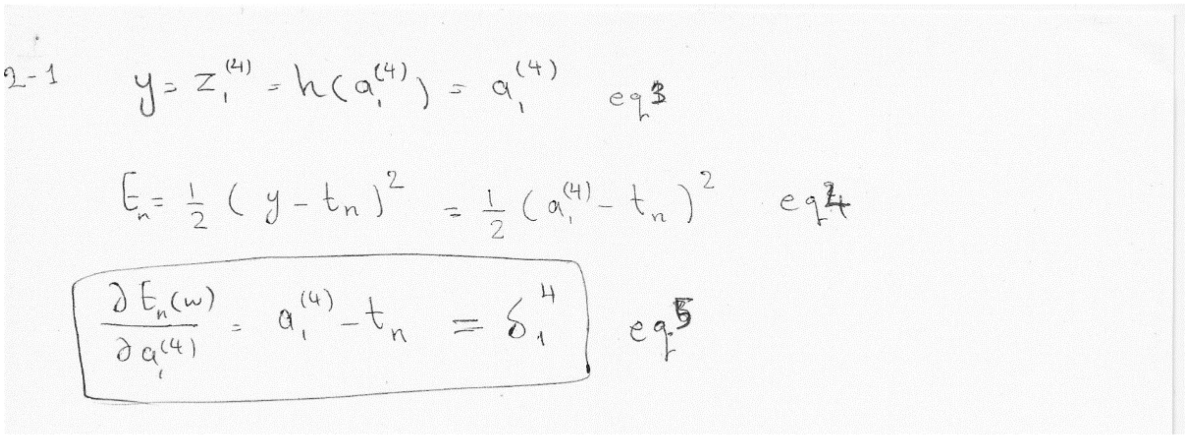
On the red lines the probability of two adjacent regions (classes) are equal.
As we move along a red line away from the green point, the probability of the third region (which is not adjacent to the specified red line) decreases. This way if we go to infinity on the red line the probability of two adjacent regions (classes) will go to ½ and the probability of third region will go to zero.

3. (4 marks) What happens to the probabilities as we move far away from the intersection point, staying in the middle of one region?

In this case as we go away from the green point in the middle of one of the regions, that region's probability (we are in the middle, let's call it C_a) will increase. The probability of two other classes (let's call them C_b and C_c) probability will decrease. This way P(C_a|x) goes to 1 and P(C_b|x) and P(C_c|x) go to zero.

Question 2)

Part1:

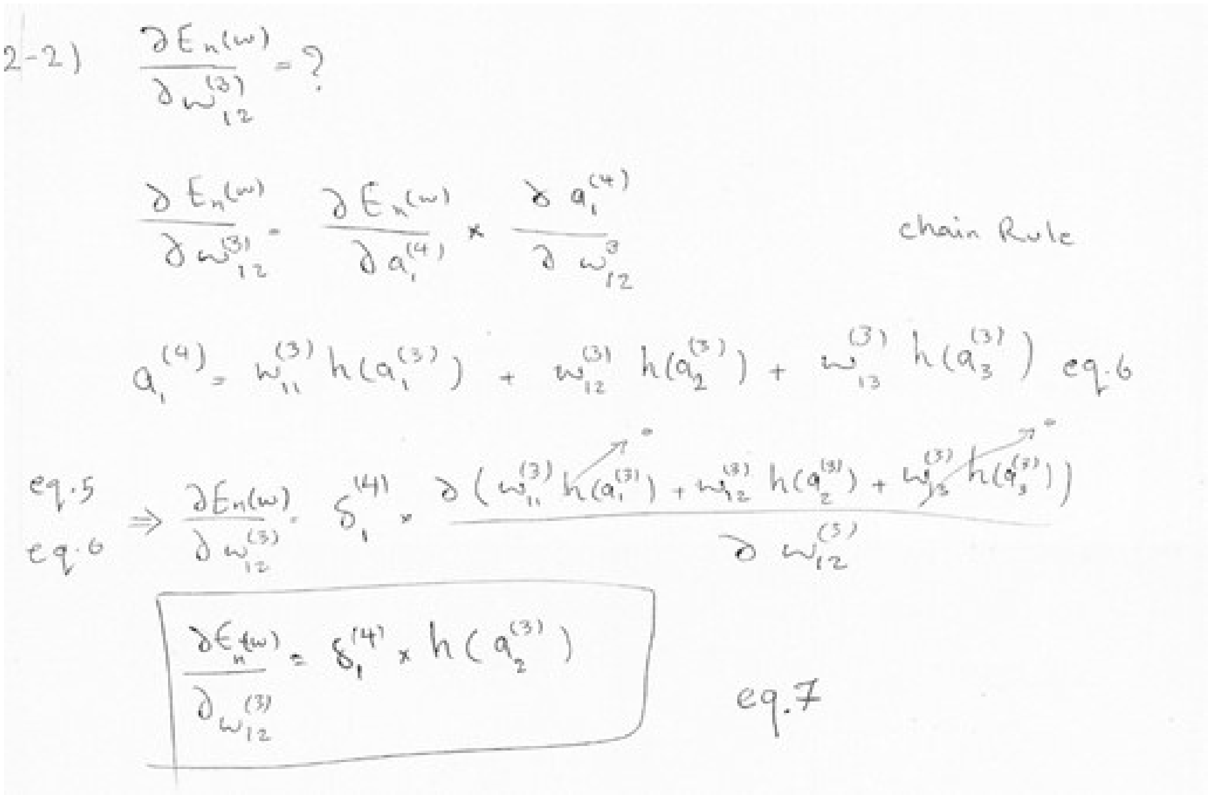


2-1 $y = z_1^{(4)} = h(a_1^{(4)}) = a_1^{(4)}$ eq3

$E_n = \frac{1}{2}(y - t_n)^2 = \frac{1}{2}(a_1^{(4)} - t_n)^2$ eq4

$$\boxed{\frac{\partial E_n(w)}{\partial a_1^{(4)}} = a_1^{(4)} - t_n = \delta_1^4}$$ eq5

Part 2:



2-2) $\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = ?$

$\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = \frac{\partial E_n(w)}{\partial a_1^{(4)}} \times \frac{\partial a_1^{(4)}}{\partial w_{12}^{3}}$ chain Rule

$a_1^{(4)} = w_{11}^{(3)} h(a_1^{(3)}) + w_{12}^{(3)} h(a_2^{(3)}) + w_{13}^{(3)} h(a_3^{(3)})$ eq.6

eq.5
eq.6 $\Rightarrow \frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = \delta_1^{(4)} \times \frac{\partial(w_{11}^{(3)} h(a_1^{(3)}) + w_{12}^{(3)} h(a_2^{(3)}) + w_{13}^{(3)} h(a_3^{(3)}))}{\partial w_{12}^{(3)}}$

$$\boxed{\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = \delta_1^{(4)} \times h(a_2^{(3)})}$$ eq.7

Part 3:

2-3) $$\frac{\partial E_n(w)}{\partial a_1^{(3)}} = ?$$

$$\frac{\partial E_n(w)}{\partial a_1^{(3)}} = \frac{\partial E_n(w)}{\partial a_1^{(4)}} \times \frac{\partial a_1^{(4)}}{\partial a_1^{(3)}} = \delta_1^{(4)} \cdot \frac{\partial a_1^{(4)}}{\partial a_1^{(3)}}$$

$$= \delta_1^{(4)} \times \frac{\partial\left(w_{11}^{(3)} h(a_1^{(3)}) + \overset{0}{\cancel{w_{12}^{(3)} h(a_2^{(3)})}} + \overset{0}{\cancel{w_{13}^{3} h(a_3^{(3)})}}\right)}{\partial a_1^{(3)}}$$

$$\boxed{\delta_1^{(3)} = \frac{\partial E_n(w)}{\partial a_1^{(3)}} = \delta_1^{(4)} \times w_{11}^{(3)} \times h'(a_1^{(3)})} \quad \text{eq.8}$$

Part 4:

2-4) $$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = ?$$

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \frac{\partial E_n(w)}{\partial a_1^{(3)}} \times \frac{\partial a_1^{(3)}}{\partial w_{11}^{(2)}} = \delta_1^{(3)} \times \frac{\partial a_1^{(3)}}{\partial w_{11}^{(2)}}$$

$$= \delta_1^{(3)} \times \frac{\partial\left(w_{11}^{(2)} \times h(a_1^{(2)}) + \overset{0}{\cancel{w_{12}^{(2)} h(a_2^{(2)})}} + \overset{0}{\cancel{w_{13}^{(2)} \cdot h(a_3^{(2)})}}\right)}{\partial w_{11}^{(2)}}$$

$$\boxed{\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \delta_1^{(3)} \times h(a_1^{(2)})} \xrightarrow{\text{eq.8}} \frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \delta_1^{(4)} \times w_{11}^{(3)} \times h'(a_1^{(3)}) \times h(a_1^{(2)})$$

Part 5:

2-5) $\frac{\partial E_n(w)}{\partial a_1^{(2)}} = ?$

$$\frac{\partial E_n(w)}{\partial a_1^{(2)}} = \sum_{1}^{3} \frac{\partial E_n(w)}{\partial a_K^{(3)}} \times \frac{\partial a_K^{(3)}}{\partial a_1^{(2)}} = \sum_{K=1}^{3} \delta_K^{(3)} \times \frac{\partial a_K^{(3)}}{\partial a_1^{(2)}}$$

$$= \sum_{K=1}^{3} \delta_K^{(3)} \times \frac{\partial\left(w_{K1}^{(2)} \cdot h(a_1^{(2)}) + \cancelto{0}{w_{K2}^{(2)} \cdot h(a_2^{(2)})} + \cancelto{0}{w_{K3}^{(2)}\, h(a_3^{(2)})}\right)}{\partial a_1^{(2)}}$$

$$\boxed{\delta_1^{(2)} = \frac{\partial E_n(w)}{\partial a_1^{(2)}} = \sum_{K=1}^{3} \delta_K^{(3)} \times w_{K1}^{(2)} \times h'(a_1^{(2)})} \quad eq.9$$

Part 6:

2-6) $$\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = \frac{\partial E_n(w)}{\partial a_1^{(2)}} \times \frac{\partial a_1^{(2)}}{\partial w_{11}^{(1)}} = \delta_1^{2} \cdot \frac{\partial a_1^{(2)}}{\partial w_{11}^{(1)}}$$

$$= \delta_1^{2} \times \frac{\partial\left(w_{11}^{(1)} \cdot x_1 + \cancelto{0}{w_{12}^{(1)} \cdot x_2} + \cancelto{0}{w_{13}^{(1)} \cdot x_3}\right)}{\partial w_{11}^{(1)}}$$

$$\boxed{\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = \delta_1^{(2)} \times x_1} \xrightarrow{eq.9} \boxed{\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = h'(a_1^{(2)}) \times x_1 \times \sum_{K=1}^{3} w_{K1}^{(2)} \delta_K^{(3)}}$$

eq.10

Question3)

Part1:

3-1) $\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = ?$

chainRule

$$\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = \frac{\partial E_n(w)}{\partial a_1^{(153)}} \times \frac{\partial a_1^{(153)}}{\partial a_1^{(152)}} \times \frac{\partial a_1^{(152)}}{\partial a_1^{(151)}} \times \cdots \times \frac{\partial a_1^{(l+2)}}{\partial a_1^{(l+1)}} \times \frac{\partial a_1^{(l+1)}}{\partial w_{11}^{(l)}}$$

$$\Rightarrow \boxed{\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = \frac{\partial E_n(w)}{\partial a_1^{(153)}} \times \left(\prod_{K=l+1}^{152} \frac{\partial a_1^{(K+1)}}{\partial a_1^{(K)}}\right) \times \frac{\partial a_1^{(l+1)}}{\partial w_{11}^{(l)}}} \quad eq.1$$

$$E_n(w) = \frac{1}{2}(y-t_n)^2 = \frac{1}{2}\left(h(a_1^{(153)}) - t_n\right)^2 \Rightarrow \boxed{\frac{\partial E_n(w)}{\partial a_1^{(153)}} = \left(h(a_1^{(153)}) - t_n\right) \times h'(a_1^{(153)}) \quad eq.2}$$

$$a_1^{(K+1)} = w_{11}^{(K)} \times h(a_1^{(K)}) \Rightarrow \boxed{\frac{\partial a_1^{(K+1)}}{\partial a_1^{(K)}} = w_{11}^{(K)} \times h'(a_1^{(K)})} \quad eq.3$$

$$a_1^{l+1} = w_{11}^{(l)} \cdot h(a_1^{(l)}) \Longrightarrow \boxed{\frac{\partial a_1^{(l+1)}}{\partial w_{11}^{(l)}} = h(a_1^{(l)})} \quad eq.4$$

replacing eq2. , eq.3, eq.4 in eq.1.

$$\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = \left(h(a_1^{(153)}) - t_n\right) \times h'(a_1^{(153)}) \times \left(\prod_{K=l+1}^{152} w_{11}^{(K)} \times h'(a_1^{(K)})\right) \times h(a_1^{(l)})$$

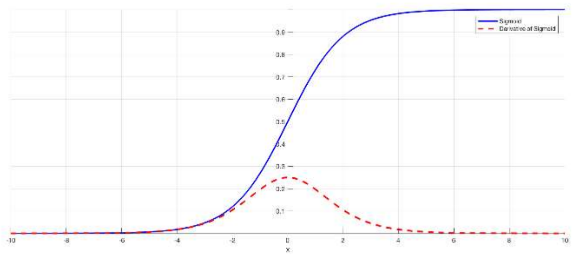
Part2:

In backpropagation training method, which is used to train a Neural Network the error calculation of each layer will start from the very last layer (output layer). Then by considering the target value the error will be calculated and will be back propagated to the first layer. In the way going back to the first layer, as shown in the above equation, there are several element that have effect on the last value of the above equation. So, the weight vector multiplies the derivation of the activation function in the activation point and all will multiplies to the value of very first error.

As far as we know from the problem that, the activation function is sigmoid which output is between 0 to 1. Using sigmoid function, very large positive input generates a positive one output, values between -4 to 4 generates output between 0 to 1 and very small negative value generates zero output (these numbers are approximately). The derivation of sigmoid function is mostly zero and only for input values distributed around 0.5 (input between -4 to 4 approximately), it can return maximum output of 0.25. So, if the value of any of layer's weight is very large positive or very small negative, then the activation of the next layer will be very large positive or very small negative. As result, the derivative of the sigmoid function for this input will be close to zero, which causes Vanishing Gradient.

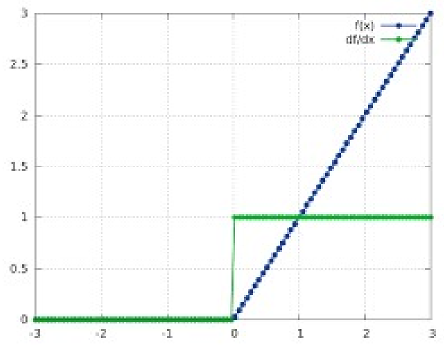
On the other hand, if the weight is very close to zero or zero it again causes the whole formula goes to zero from that layer back to the first layer, which is exactly called Vanishing Gradient. As a result, selecting proper initial value for the weights can help us to reduce the chance of Vanishing Gradient to more steps in the back-propagation process. Because there is always the chance of vanishing gradient exists.

There is also another possibility, which is the error value of the output layer is very small, which can cause the Vanishing Gradient as well. Bellow the sigmoid function and its derivative is illustrated.



Part 3:

The *ReLU* activation function return 0 if its input is negative or zero and return the input value for positive input value. As a result, the derivation of this function is zero for negative value and one for positive values. If the input is set to negative, then the derivation will be zero and from that point the gradient will have no effect (no learning) as the value will be zero. The back propagated error can be cancelled out whenever there is a negative input into a given neuron and therefore the gradient will also fall to zero. But if the weight vector set properly, ReLU can behave better than sigmoid in the case of Vanishing Gradient.

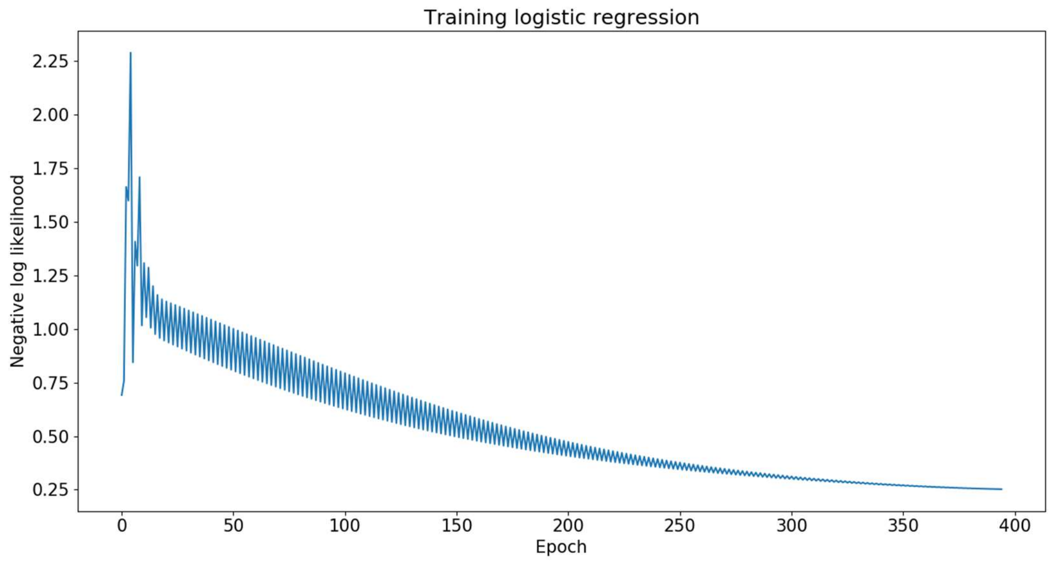


Part 4:

If the derivation of the both connected node from the previous layer goes to zero, then the value of the gradient will go to zero by doing back propagation. Because, the activation of each node in the layer l +1 is sum of multiply of two nodes in the previous layer to the related weight. So, their gradient will become zero when both $\boldsymbol{h'(a_1^{(l)})}$ and $\boldsymbol{h'(a_2^{(l)})}$ become zero. Or if for the same reasoning as above, both weight of a layer goes to zero, then the gradient goes to zero as well.

Question 4)

Part 1:

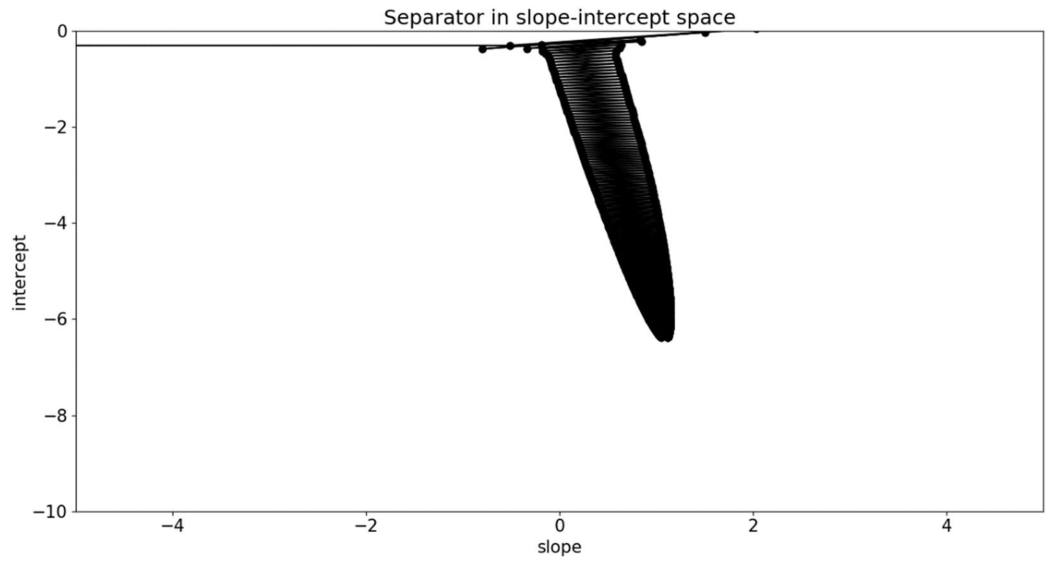


The plots produced by logistic_regression.py is shown above. To train the model using the gradient decent following steps must be repeated over the training dataset. By reaching a specific number of itteration or getting an error smaller than a pre-set threshold, the training will stop.

1- Calculate the output, using the current W (weight).
2- Calculate the error based on the Eq. 4.90.
3- Save the error to plot the changing trend of the error.
4- Calculate the gradient based on the Eq. 4.91.
5- Calculate the new weight vector by deducting the derivation (gradient) times the step factor (eta).

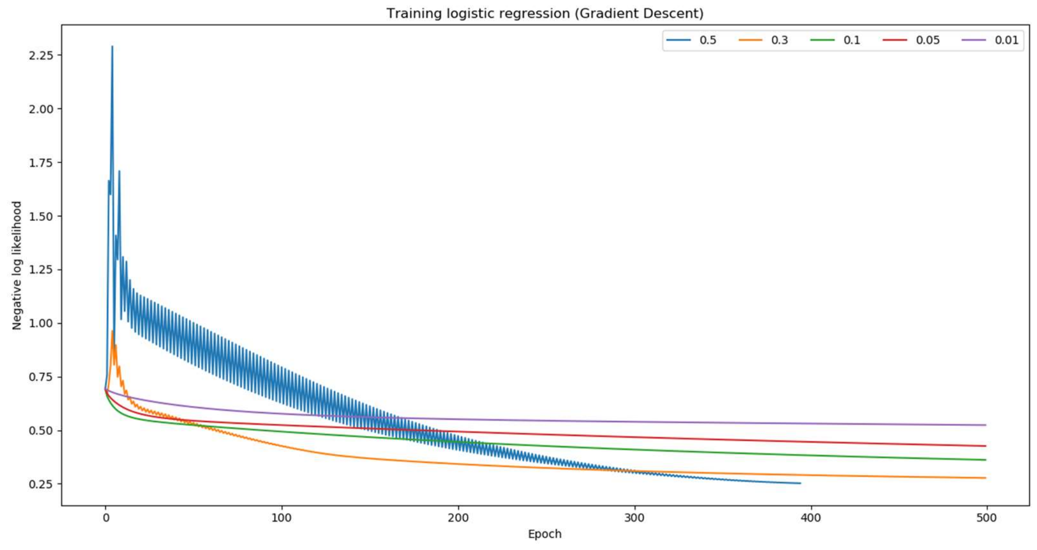
Plot's Analysis:
As the model is training, the error (negative log likelihood) decreased overally. But there is an oscillation which the reason is the large value of the learning rate. As it is shown in the next plot, the value of the slope interchangeably decreases and increase. This means that the separator line is moving over the data to provide better error value (decrease it).
Although the overall trend is downward. The reason for this oscillation is the size of the learning rate. The bigger learning rate can cause overshooting and since this learning rate is large, the back and forth of the separator line is apparently visible.
The second plot illustrates the changes in the slope and the intercept of the separator line as the model is learning. As it is shown, the jumping range of the separator line’s slope is decreasing and it is converging to a value around 1 and the intercept is decreasing to some value. So, changing the slope as shown in the following image cause changes in data classification, and consequently, this causes the

oscillation of the Negative log likelihood in the above plot.



Part 2:

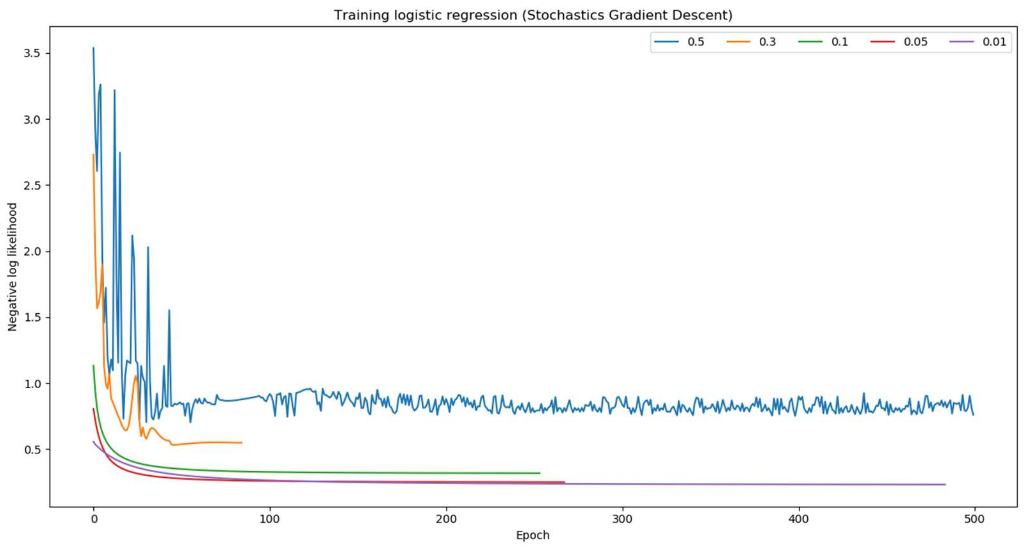
Create a Python script logistic_regression_mod.py for the following. Modify logistic_regression.py to run gradient descent with the learning rates eta= 0:5; 0:3; 0:1; 0:05; 0:01.



In the second problem, we are studying the effect of changing the learning step size (eta) in the training process of the model. This time, same gradient descent algorithm is applying over the training dataset using different step size vary from 0.5 to 0.01. Based on the achieved result, the larger eta causes more oscillation and longer converging but smaller error value. In other words, the larger step size causes more fluctuation and later convergent but smaller error. On the other hand, the smaller step size value causes sooner convergent and less fluctuation but the larger value of error because it might convert to the local minimum. So, as a result, we can imply that the moderate step size cause a balance between accuracy and convergent's speed.

Part 3:

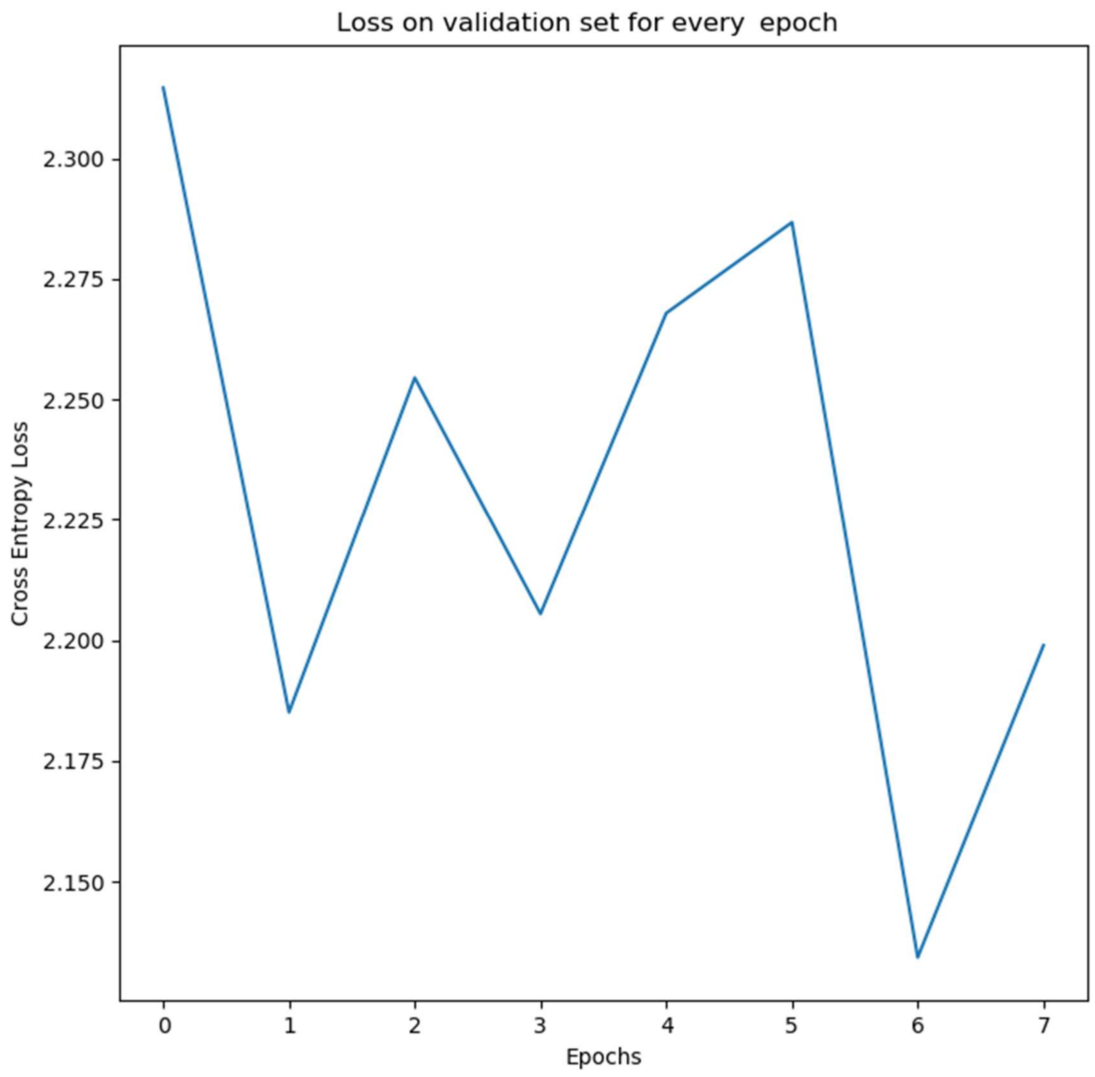
Create a Python script logistic_regression_sgd.py for the following. Modify this code to do stochastic gradient descent. Use the parameters eta = 0:5; 0:3; 0:1; 0:05; 0:01.



For this question the stochastic gradient descent was implemented. In this method using the proper number of learning rate or step size, causes to faster convergent, with acceptable accuracy. But such a behaviour happens in a specific range of etha. In the above plot, the range of [0.1 to 0.01], provides accepteable acuracy and fast convergent (the red, green and pink lines). But more or less than that the speed is slow and for the largest value (0.5) it did not converge properly and it has flactuation. But in this scenario the stochastic gradient descent converged faster, as shown above.

Qustion 5)

Second part is selected and implemented. So the following diagram sjows how the loss value changes on validation of the model after training model on each epoch.
The number of epochs was 8 and the size of the training dataset was 32000 data and the test and the validation dataset size was 3000. For more information on how the code is structured and work please check the readme file in the code.zip folder.


Loss on validation set for every epoch
Cross Entropy Loss
Epochs