1 SoftMax for Multi-Class Classification

1.1 The probabilities of the x belong to Class1, Class2 and class 3 should be equal at the intersection point. Therefore

$$p(c1|x) = p(c2|x) = p(c3|x) = 1/3$$

1.2. For the points in the red line, the probabilities are equal which are divided by two of the classes.

$$p(c1|x) = p(c2|x)$$

When the probabilities are moved along each of the red lines, the probabilities for the points belonging to the two classes which are divided by the red line are equal. However, the probability for the points belongings to the third class is reducing while the probability of belongings to the other two classes increase. In the example above, $p(c3|x)$ will reduce and $p(c1|x)$ and $p(c2|x)$ will increase.

1.3. If we move far away from the intersection point and staying in the middle of one region, the probability for the points belonging to this region will increase and it will be larger than that of the remaining two regions (decrease as the point move). That is

For Probability of points belonging to region 1: -

$$p(c1|x) > p(c2|x) \text{ and } p(c3|x)$$

For Probability of points belonging to region 2: -

$$p(c2|x) > p(c1|x) \text{ and } p(c3|x)$$

For Probability of points belonging to region 3: -

$$p(c3|x) > p(c1|x) \text{ and } p(c2|x)$$

2 Error Backpropagation

2.1

- Calculate $\frac{\partial E_n(w)}{\partial a_1^{(4)}}$. Note that $a_1^{(4)}$ is the activation of the output node, and that $\frac{\partial E_n(w)}{\partial a_1^{(4)}} \equiv \delta_1^{(4)}$.

We know that the activation function h(a)=a, so we can rewrite the error function as the below: -

$$E_n(w) = \tfrac{1}{2}(h(a_1^{(4)}) - t_n)^2 = \tfrac{1}{2}(a_1^{(4)} - t_n)^2$$

Applying this to $\frac{\partial E_n(w)}{\partial a_1^{(4)}} = \frac{1}{2}\frac{\partial (a_1^{(4)} - t_n)^2}{\partial a_1^{(4)}}$

$$= (a_1^{(4)} - t_n)$$

2.2

Calculate $\frac{\partial E_n(\omega)}{\partial w_{12}^{(3)}}$

By using the chain rule, we can derive the following: -

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

Applying the result received in the above sum, we can further conclude as

$$(a_1^{(4)} - t_n)\left[\frac{\partial a_1^{(4)}}{\partial w_{12}^{(3)}}\right].$$

$$\frac{\partial a_1^{(4)}}{\partial w_{12}^{(3)}} = \frac{\partial}{\partial w_{12}^{(3)}} \sum_{k=1}^{3} w_{1k}^{(3)} z_k^{(3)}$$

Where $a_1^{(4)} = w_{11}^{(3)} z_1^{(3)} + w_{12}^{(3)} z_2^{(3)} + w_{13}^{(3)} z_3^{(3)}$

$$\therefore \boxed{\frac{\partial a_1^{(4)}}{\partial w_{12}^{(3)}} = z_2^{(3)}}$$

$$\therefore \frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = (a_1^{(4)} - t_n)\, z_2^{(3)}$$

given $z_2^{(3)} = h(a_2^{(3)})$

$$\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = (a_1^{(4)} - t_n)\left[h(a_2^{(3)})\right]$$

$$h(a) = \frac{1}{1+e^{-a}} \qquad \therefore \frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = (a_1^{(4)} - t_n)\frac{1}{1+e^{-a_2^{(3)}}}$$

2.3

Write an expression for $\frac{\partial E_n(w)}{\partial a_1^{(3)}}$

By using the chain rule, we can derive the following: -

$$\frac{\partial E_n(w)}{\partial a_1^{(3)}} = \sum_{K=1}^{1} \frac{\partial E_n(w)}{\partial a_K^{(4)}} \frac{\partial a_K^{(4)}}{\partial a_j^{(3)}} \quad \text{where} \sum_{K=1}^{1} \frac{\partial E_n(w)}{\partial a_K^{(4)}} = \delta_1^{(4)} \quad -(1)$$

$$\frac{\partial a_M^{(4)}}{\partial a_1^{(3)}} = \frac{\partial \sum_{M=1}^{3}\sum_{K=1}^{3} w_{KM}^{(3)} z_M^{(3)}}{\partial a_1^{(3)}} = \frac{\partial \sum_{M=1}^{3}\sum_{K=1}^{3} w_{KM}^{(3)} h(a_M^{(3)})}{\partial a_1^{(3)}}$$

$$= h'(a_1^{(3)}) \sum_{K=1}^{1} w_{K1}^{(3)} = h'(a_1^{(3)}) w_{11}^{(3)} \quad -(2)$$

Therefore from (1) & (2) $\frac{\partial E_n(w)}{\partial a_1^{(3)}} = h'(a_1^{(3)}) \sum_K w_{K1}^{(3)} \delta_K^{(4)}$

$$= h'(a_1^{(3)})\, w_{11}^{(3)}\, \delta_1^{(4)}$$

if $h(a) = \frac{1}{1+e^{-a}}$ then $\frac{\partial E_n(w)}{\partial a_1^{(3)}} = \sigma(a_1^{(3)})\left[1-\sigma(a_1^{(3)})\right] w_{11}^{(3)} \delta_1^{(4)}$

2.4

calculate $\frac{\partial E_n(w)}{\partial w_{11}^{(2)}}$

By using the chain rule, we can derive the following: -

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \delta_1^{(3)} \frac{\partial a_1^{(3)}}{w_{11}^{(2)}} \quad \text{where} \quad \frac{\partial a_1^{(3)}}{w_{11}^{(2)}} = z_1^{(2)}$$

$$\delta_1^{(3)} = \frac{\partial E_n(w)}{\partial a_1^{(3)}} = h'(a_1^{(3)}) w_{11}^{(3)} \delta_1^{(4)}$$

Therefore,

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = h'(a_1^{(3)}) w_{11}^{(4)} \delta_1^{(4)} z_1^{(2)}$$

if $h(a) = \frac{1}{1+e^{-a}}$ then,

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \sigma(a_1^{(3)}) \left[1 - \sigma(a_1^{(3)})\right] w_{11}^{(3)} \delta_1^{(4)} z_1^{(2)}$$

2.5

Write an expression for $\frac{\partial E_n(w)}{\partial a_1^{(2)}}$

$$\frac{\partial E_n(w)}{\partial a_1^{(2)}} = \sum_{k=1}^{3} \frac{\partial E_n(w)}{\partial a_k^{(3)}} \frac{\partial a_k^{(3)}}{\partial a_j^{(2)}}$$

$$\Rightarrow \sum_{k=1}^{3} \frac{\partial E_n(w)}{\partial a_k^{(3)}} = \sum_{k=1}^{3} \delta_k^{(3)} \qquad -(1)$$

$$\frac{\partial a_m^{(3)}}{\partial a_1^{(2)}} = \frac{\partial \sum_{m=1}^{3} \sum_{k=1}^{3} w_{km}^{(2)} z_m^{(2)}}{\partial a_1^{(2)}}$$

$$= \frac{\partial \sum_{m=1}^{3} \sum_{k=1}^{3} w_{km}^{(2)} h(a_m^{(2)})}{\partial a_1^{(2)}}$$

$$= h'(a_1^{(2)}) \sum_{k=1}^{3} w_{k1}^{(2)} \qquad -(2)$$

Therefore from (1) & (2) we get

$$\frac{\partial E_n(w)}{\partial a_1^{(2)}} = h'(a_1^{(2)}) \sum_{k=1}^{3} w_{k1}^{(2)} \delta_k^{(3)}$$

if $h(a) = \frac{1}{1+e^{-a}}$ then

$$\frac{\partial E_n(w)}{\partial a_1^{(2)}} = \sigma(a_1^{(2)}) \left[1 - \sigma(a_1^{(2)})\right] \sum_{k=1}^{3} w_{k1}^{(2)} \delta_k^{(3)}$$

2.6

calculate $\frac{\partial E_n(w)}{\partial w_{11}^{(1)}}$

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

where $\boxed{\frac{\partial a_1^{(2)}}{w_{11}^{(1)}} = z_1^{(1)}}$

Finding for $\delta_1^{(2)} = \frac{\partial E_n(w)}{\partial a_1^{(2)}} - h'(a_1^{(2)}) \sum_{K=1}^{3} w_{K1}^{(2)} \delta_K^{(3)}$

Therefore,

$$\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = h'(a_1^{(2)}) \sum_{K=1}^{3} w_{K1}^{(2)} \delta_K^{(3)} z_1^{(1)}$$

if $h(a) = \frac{1}{1+e^{-a}}$

$$\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = \sigma(a_1^{(2)}) \left[1 - \sigma(a_1^{(2)}) \sum_{K=1}^{3} w_{K1}^{(2)} \delta_K^{(3)} x_1^{(1)} \right]$$

3 Vanishing Gradients

3.1

Write an expression for $\frac{\partial E_n(w)}{\partial w_{11}^{(l)}}$ for all layers l in the network.

$$\frac{\partial E_n(w)}{\partial w_{11}^{(152)}} = \frac{\partial E_n(w)}{\partial a_1^{(153)}} \frac{\partial a_1^{(153)}}{\partial w_{11}^{(152)}}$$

$$= (a_1^{(153)} - t_n) \frac{\partial a_1^{(153)}}{\partial w_{11}^{(152)}} = (a_1^{(153)} - t_n) z_1^{(152)}$$

as derived in the above problem.

Consider if $(n \geq 153)$ & $(l > 152)$, then

$$\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = (a_1^{(153)} - t_n).$$

else when $n = 152$, $l = 151$, $l = n-1$

$$\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = (a_1^{(153)} - t_n) \prod_{n=l+1}^{153} h'(a_1^{(n)}) w_{11}^{(n)} z_1^{l}$$

3.2

Describe what would happen to the gradient for weights early in the network $\frac{\partial E_n(w)}{\partial w_{11}^{(l)}}$ for smaller l

When the gradients for weights occur early in the network for smaller l, then

- The gradient descent tends to become o when the output of some sigmoid function goes to o.
- Considering the gradients for weights early in the network, the gradient tends to be small and hence for the front layers in an n layer network, **"Vanishing Gradient"** problem arises.
- For the layers backward in the network, the gradients will have quite good magnitude.

3.3

Suppose we use rectified linear units (ReLU) in activation functions. When would the gradients be zero?

- Considering the input as $a_j^{(e)}$,

ReLU function can be written as

$$h(a_j) = \max(0, a_j)$$

The gradient value depends on the ReLU value.

1.) The ReLU goes to 0, when the input $a_j^{(e)}$ is 0 or less. Hence the corresponding gradients will become 0.

2.) When the input $a_j^{(l)}$ is more or greater than 0, then the ReLU makes $\boxed{h(a_j)' = 1}$. This could solve the "Vanishing gradients" problem, but this will resolve completely.

3.4

Suppose we modify the graph to have bipartite connections at each layer, still using ReLU activation functions. When would the gradients be zero?

When the graph to have bipartite connections is modified, then

$$\frac{\partial E_n(w)}{\partial w_{11}^{(l)}} = \sigma(a_1^{(l+1)})\left(1-\sigma(a_1^{(l+1)})\right)\sum_{k=1}^{2} w_{k1}^{(l+1)(l+1)(l)} \delta_k z_1$$

For bipartite graph, we need to take both 2 nodes for the layers before $(l+1)$ layers.

The gradient becomes zero when $h'(a_1^{(l+1)})$ and $h'(a_2^{(l+1)})$ becomes zero. By summing both two, we get $\frac{\partial E_n(w)}{\partial w_{11}^{(l)}}$ which goes to 0.

4 Logistic Regression

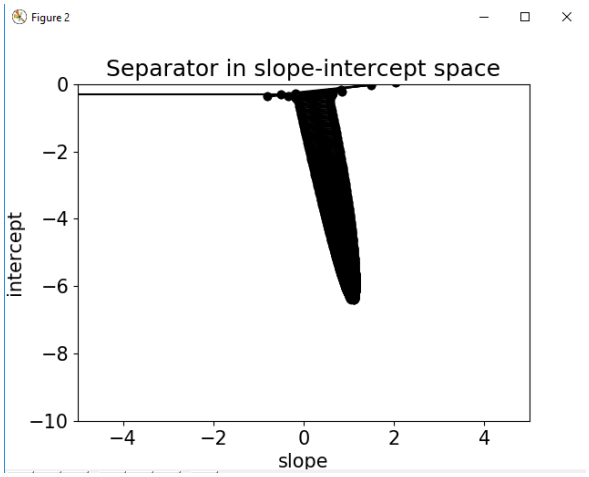
4.1

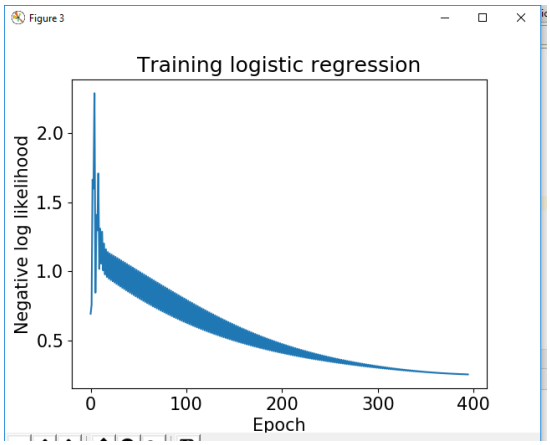
Download the assignment 2 code and data from the website. Run the script `logistic_regression.py` in the lr directory. This code performs gradient descent to find w which minimizes negative log-likelihood (i.e. maximizes likelihood).

The plot of slope Intercept Space in shown in Figure 2 and the plot of neg. log likelihood over epochs is shown in Figure 3. In Figure 2, the oscillation is because of the learning rate and the direction of the gradient descent. The value of learning rate eeta value is 0.5, hence it will be huge, and the gradient descent will jump too large in iterations.

```
# Step size for gradient descent.
eta = 0.5
```

In Figure 3, the direction of the gradient is not completely pointing to the bottom; hence it will cause lots of oscillations.



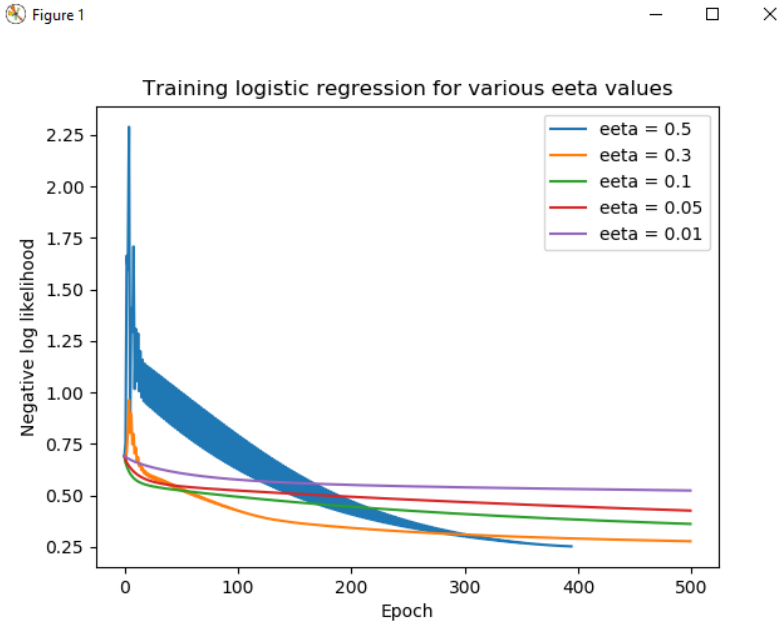


4.2

Create a Python script `logistic_regression_mod.py` for the following.

Modify `logistic_regression.py` to run gradient descent with the learning rates $\eta = 0.5, 0.3, 0.1, 0.05, 0.01$.

The below Figure 1 shows the plot between negative log likelihood and the iterations for various eeta values. From this figure, we can conclude that as the learning rate eeta value increases, the oscillation increases even though the speed of convergence has improved. But as the eeta value decreases, we can see there is no oscillation, but the speed of convergence will be reduced. We can also see that the negative log likelihood is higher as the eeta value is smaller and negative log likelihood is less when the eeta value is higher.



4.3

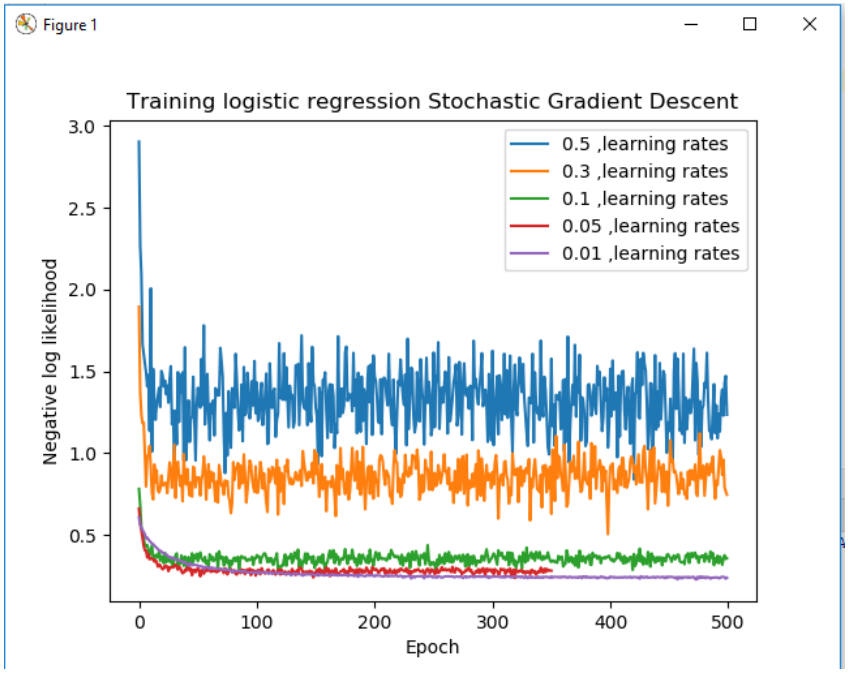
Create a Python script `logistic_regression_sgd.py` for the following.

Modify this code to do stochastic gradient descent. Use the parameters $\eta = 0.5, 0.3, 0.1, 0.05, 0.01$.

The Python script has been created as logistic regression sgd.py by using the learning rate values as such as 0.5, 0.3, 0.1, 0.05, 0.01 respectively. The data's shuffled using the np.random.shuffle(data) and ran for 500 epochs with negative log likelihood and weights for every epoch (sample output is provided below).

```
C:\Users\Ab\AppData\Local\Programs\Python\Python37-32\python.exe "C:/Vignesh/Machine L
epoch 0, negative log-likelihood 3.0392, w=[ 4.58553044 -0.94122982 -7.30975251]
epoch 1, negative log-likelihood 2.3916, w=[  3.78275326  -6.06333421 -14.73982634]
epoch 2, negative log-likelihood 1.6110, w=[  0.77917659  -5.91595082 -18.90110355]
epoch 3, negative log-likelihood 2.0662, w=[  4.34589504  -4.62428427 -21.88589368]
epoch 4, negative log-likelihood 1.4082, w=[  3.69803028  -6.33905005 -24.21125222]
epoch 5, negative log-likelihood 1.5100, w=[  3.13063796  -6.62485323 -25.84765373]
epoch 6, negative log-likelihood 1.7493, w=[  4.45543331  -6.40632361 -28.05769847]
epoch 7, negative log-likelihood 1.5268, w=[  5.06662806  -5.71498776 -29.86092918]
epoch 8, negative log-likelihood 1.1825, w=[  4.90639537  -3.33261501 -31.89210617]
epoch 9, negative log-likelihood 1.4543, w=[  5.59460184  -6.9468048  -34.01180135]
epoch 10, negative log-likelihood 1.4058, w=[  5.70785129  -5.38675538 -35.91095972]
epoch 11, negative log-likelihood 1.4246, w=[  7.76112517  -8.47051117 -36.2917026 ]
epoch 12, negative log-likelihood 1.3635, w=[  7.05298148  -4.19754944 -38.32722134]
epoch 13, negative log-likelihood 1.4784, w=[  8.19123476  -6.69144672 -38.963848  ]
epoch 14, negative log-likelihood 1.5492, w=[  4.88974768  -7.9495648  -39.92460556]
epoch 15, negative log-likelihood 1.3037, w=[  8.6147974   -6.56019087 -41.98642238]
epoch 16, negative log-likelihood 1.2333, w=[  7.60343582  -6.38556324 -43.60608429]
epoch 17, negative log-likelihood 1.3273, w=[  5.25141493  -7.6974766  -44.89960876]
epoch 18, negative log-likelihood 1.5898, w=[ 11.19547932  -8.99300955 -44.72186795]
epoch 19, negative log-likelihood 1.2934, w=[  8.44260327  -9.4043     -46.32175553]
epoch 20, negative log-likelihood 1.5658, w=[  9.11266718  -7.83434055 -47.00322391]
epoch 21, negative log-likelihood 1.2977, w=[ 11.4582633   -5.30151589 -47.70611362]
```

When the epochs run, the stochastic gradients for each learning rates are received. The below Figure 1 shows the plot between the Negative log likelihood and the Logistic Regression by stochastic gradient descent for various eeta values.



We can see that based on the learning rate eeta value, the stochastic gradient value differs with the gradient descent. When the learning rate eeta is large, the stochastic gradient will be slower than the gradient descent. But when the learning rate eeta is small, the stochastic gradient will be faster than the gradient decent. Therefore, the **stochastic gradient is not always faster than the gradient descent.**

5) Fine-Tuning a Pre-Trained Network

Run validation of the model every few training epochs on validation or test set of the dataset and save the model with the best validation error.

I have provided the Training and Validation losses for 5 epochs and the best Validation loss is in the 5th epoch.

Figure 1: - Showing the epochs and Training and Validation losses while execution.

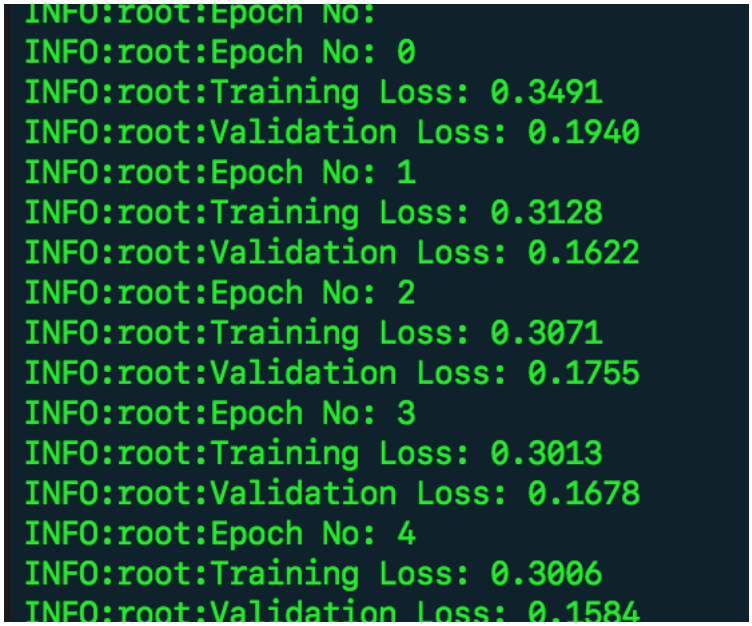


Figure 2: - Showing the 5 epochs and the respective Training and Validation loss.

Epoch	Training Loss	Validation Loss
0	0.3491	0.194
1	0.3128	0.1622
2	0.3071	0.1755
3	0.3013	0.1678
4	**0.3006**	**0.1584**

Kindly refer the readme.txt file for the execution and coding logic.