1.

truth table of XOR :

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$X = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T$$

$$Y = \left[y^{(i)} \right] = \left[0 \; 1 \; 1 \; 0 \right]^T$$

$$W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

MSE loss function :-

$$J(\theta) = \frac{1}{4} \sum_{x \in X} \left(f^*(x) - f(x;\theta) \right)^2,$$

where θ : w_1, w_2, b

$$\therefore J(\underbrace{w_1, w_2, b}^{\theta}) = \frac{1}{4} \sum_{x \in X} \left(f^*(x) - (X^T W + b) \right)^2 \text{ —— (i)}$$

$$\left\{ f^*(x) = y, \; X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \; W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \right\}$$

$$\nabla_{w_1} J(\theta) = \frac{1}{2} \sum_{x \in X} (-x_1) \left(y - x_1 w_1 - x_2 w_2 - b \right)$$

$$\nabla_{w_2} J(\theta) = \frac{1}{2} \sum_{x \in X} (-x_2) \left(y - x_1 w_1 - x_2 w_2 - b \right)$$

eq. (i) can be re written as :-

$J(w_1, w_2, b)$

$$= \frac{1}{4} \left[\underbrace{[0 \; 1 \; 1 \; 0]^T}_{1 \times 4} - \left(\underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}}_{4 \times 2} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{2 \times 1} + \underbrace{[b]}_{1 \times 1} \right) \right]^2$$

$$J(w_1, w_2, b) =$$

$$\frac{1}{4}\left[[0\ 1\ 1\ 0]^T - \left(\begin{bmatrix} 0 \\ w_2 \\ w_1 \\ w_1 + w_2 \end{bmatrix}_{4\times 1} + b \right) \right]^2$$

$$= \frac{1}{4}\left[[0\ 1\ 1\ 0]^T - \left(\begin{bmatrix} b \\ w_2 + b \\ w_1 + b \\ w_1 + w_2 + b \end{bmatrix}_{4\times 1} \right) \right]^2$$

$$= \frac{1}{4}\left[\begin{bmatrix} -b \\ 1 - (w_2 + b) \\ 1 - (w_1 + b) \\ -(w_1 + w_2 + b) \end{bmatrix} \right]^2$$

$$= \frac{1}{4}\left[b^2 + (1 - (w_2 + b))^2 + (1 - (w_1 + b))^2 + (w_1 + w_2 + b)^2 \right]$$

$$\therefore J(w_1, w_2, b) = \frac{1}{4}\left[\underbrace{b^2 + (1 - (w_2 + b))^2 + (1 - (w_1 + b))^2 + (w_1 + w_2 + b)^2}_{\text{(simplifying this term :)}} \right]$$

$$\rightarrow b^2 + 1 + (w_2 + b)^2 - 2(w_2 + b) + 1 + (w_1 + b)^2 + 2(w_1 + b) - 2(w_1 + b)$$
$$+ (w_1 + w_2)^2 + 2(w_1 + w_2)b + b^2$$

$$= b^2 + 2 + w_2^2 + b^2 + 2w_2 b - 2w_2 - 2b + 1 + w_1^2 + b^2 + 2w_1 b +$$
$$\cancel{2w_1 + 2b}\ w_1^2 + w_2^2 + 2w_1 w_2 + 2w_1 b + 2w_2 b + b^2$$

$$\Rightarrow 4b^2 + 2 + 2(w_1^2 + w_2^2) + 4b(w_1 + w_2) -$$
$$2(w_1 + w_2 + 2b) + 2w_1 w_2$$

$$J(w_1, w_2, b) = \frac{1}{4}\Big(4b^2 + 2 + 2(w_1^2 + w_2^2) + 4b(w_1 + w_2) - 2(w_1 + w_2 + 2b) + 2w_1 w_2\Big) \quad \text{--- (ii)}$$

taking gradient of eq (ii) wrt w_1, w_2 & b and equating to 0 :-

$$\nabla J(w_1) = [0 + 4w_1 + 4b + 2w_2 - 2] = 0 \quad \text{--- (iii)}$$

$$\nabla J w_2 = [0 + 4w_2 + 4b + 2w_1 - 2] = 0 \quad \text{--- (iv)}$$

$$\nabla J b = [8b + 4(w_1 + w_2) - 4] = 0 \quad \text{--- (v)}$$

eq (iii), (iv) & (v) can be rewritten as :-

$$\cancel{4w_1 + 4b + 2w_2 = 2} \quad \text{--- (iii)}$$

$$\nabla J w_1 \Rightarrow 2w_1 + 2b + w_2 = 1 \quad \text{--- (iii)}$$

$$\nabla J w_2 \Rightarrow 2w_2 + 2b + w_1 = 1 \quad \text{--- (iv)}$$

$$\nabla J b \Rightarrow 2b + w_1 + w_2 = 1 \quad \text{--- (v)}$$

subtracting ~~equating~~ eq (iii) & (iv), we will get :

$$w_1 - w_2 = 0$$

$$\Rightarrow w_1 = w_2 \quad \text{--- (vi)}.$$

equating eq (vi) in eq (v). & (iii).

$$2b + 2w_1 = 1. \quad \text{--- (vii)}$$

$$\cancel{b + w_1 = 1/2}.$$

$$3w_1 + 2b = 1. \quad \text{--- (viii)}.$$

eq (vii) × 3 − eq (viii) × 2 :

$$
\begin{array}{rcl}
6b + 6w_1 &=& 3 \\
\underline{-4b \pm 6w_1} &=& \underline{2} \\
2b &=& 1
\end{array}
$$

$\Rightarrow b = 1/2$

substituting $b = 1/2$ in eq (vii)

$2 \times \frac{1}{2} + 2w_1 = 1$

$\Rightarrow 2w_1 = 0$

$\Rightarrow w_1 = 0.$

from eq (vi), we can deduce that:

$w_1 = w_2 = 0$

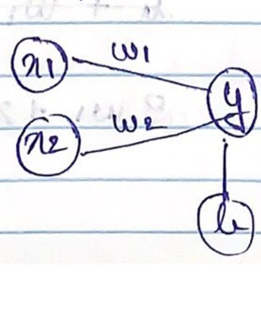
∴ after equating gradient to 0, we have found the value of w_1, w_2 b as:

$w_1 = 0$

$w_2 = 0$

$b = 1/2 = 0.5$

∴ hence proved.



2 $\hat{y} = \sigma(x^T w + b)$

(a) ~~here~~ for exact, all the training examples are positive. ~~A~~ Hence, ~~the~~ 'b' (bias) will converge to a value that will result a positive output for all positive input (i.e. positive training examples), this will not be same for different examples.

The training examples should be linearly seperable, then 'b' will converge to a unique value, which ~~will~~ will minimize log. loss.

(b) similar to the above scenario, convergence of weight vector 'w' will depend on the exact training examples. For all positive training examples weight vector will yield positive outputs for all positive inputs.

Different set of examples will lead to different distribution of weights that minimizes ~~loss~~ log loss.

(c) The ~~traing~~ training loss will converge. The optimization algorithm will continuously update values of 'w' & 'b' untill loss function has reached minima

(d) The testing loss may not necessarily converge, even if ~~traing~~ training loss converges. This because, since, training examples contains only positive examples, and testing examples has both positive & negative examples. ~~So~~ Hence, when the model predicts on testing set for positive examples the loss will tend to infinity, however for negative examples, the model will miss-classify.

~~2.~~

3. Derivation of softmax regression gradient updates

$$W = [w^{(1)} \dots w^{(c)}]$$

$$\hat{y}_k = \frac{\exp z_k}{\sum_{k'=1}^{c} \exp z_{k'}}$$

$$z_k = x^T w^{(k)} + b_k$$

$$f_{CE}(W, b) = \frac{-1}{n} \sum_{i=1}^{n} \sum_{k=1}^{c} y_k^{(i)} \log \hat{y}_k^{(i)}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{c} y_{(k)}^{(i)} \left(\frac{\nabla_{w^{(l)}} \hat{y}_k^{(i)}}{\hat{y}_k^{(i)}} \right)$$

We handle the two cases $l = k$ and $l \neq k$ seperately,

(i) Solving first for $l = k$, we get

$$\nabla_{w^{(l)}} \hat{y}_k^{(i)} = \nabla_{w^{(l)}} \left[\frac{\exp(x^{(i)T} w^{(l)} + b_l)}{\sum_{k'=1}^{c} \exp(x^{(i)T} w^{(k')} + b_{k'})} \right]$$

Using quotient rule ~~where~~ $\frac{u'v - v'u}{v^2}$, we get

$$= \cancel{\frac{x^{(i)} \times \nabla_{w^{(l)}} (x^{(i)T} w^{(l)} + b_l)}{\sum_{k'=1}^{c} \exp z_{k'}}} -$$

$$= x^{(i)} \frac{\exp x^{(i)T} w^{(l)}}{\sum_{k'=1}^{c} \exp(x^{(i)T} w^{(k')})} - \frac{x^{(i)} \exp(x^{(i)T} w^{(l)})}{(\sum_{k'=1}^{c} \exp(x^{(i)T} w^{(k')}))^2} \cdot \exp(x^{(i)T} w^{(l)})$$

$$= x^{(i)}\left[\frac{\exp(x^{(i)T}w^{(l)})}{\sum_{k'=1}^{C}\exp(x^{(i)T}w^{(k')})} - \frac{(\exp(x^{(i)T}w^{(l)}))^2}{(\sum_{k=1}^{C}\exp(x^{(i)T}w^{(k')}))^2}\right]$$

We can see that $\hat{y}_l^{(i)} = \frac{\exp(x^{(i)T}w^{(l)})}{\sum_{k'=1}^{C}\exp(x^{(i)T}w^{(k')})}$.

Substituting this value above, we get.

$$= x^{(i)}\left[\hat{y}_l^{(i)} - (\hat{y}_l^{(i)})^2\right]$$

$$= x^{(i)}\hat{y}_l^{(i)}\left[1 - \hat{y}_l^{(i)}\right]$$

ii) Solving now for $l \neq k$, we get

$$\nabla_{w^{(l)}}\hat{y}_k^{(i)} = \nabla_{w^{(l)}}\left[\frac{\exp(x^{(i)T}w^{(k)})}{\sum_{k=1}^{C}\exp(x^{(i)T}w^{(k')})}\right]$$

$$= -x^{(i)}\frac{\exp(x^{(i)T}w^{(k)}) \cdot \exp(x^{(i)T}w^{l})}{(\sum_{k'=1}^{C}\exp(x^{(i)T}w^{(k')}))^2}$$

$$= \frac{-x^{(i)}(\exp(x^{(i)T}w^{(k)})) \times (\exp(x^{(i)T}w^{l}))}{(\sum_{k=1}^{C}\exp(x^{(i)T}.w^{(k')}))^2}$$

$$= -x^{(i)}\hat{y}_k^{(i)}\hat{y}_l^{(i)}$$

(iii) To compute ~~over~~ the total gradient of J_{CE}, wrt each $w^{(k)}$, we have to sum over all examples and over $l = 1, \ldots C$.

$$\nabla_{w^{(l)}} J_{CE}(W,b) = -\frac{1}{n}\sum_{i=1}^{n} \sum_{k=1}^{C} y_k^{(i)} \nabla_{w^{(l)}} \log \hat{y}_k^{(i)}$$

$\because \sum_k a_k = a_l + \sum_{k\neq l} a_k$

$$= -\frac{1}{n}\sum_{i=1}^{n} \left(y_l^{(i)} \times \left(\frac{x^{(i)} \times \cancel{\hat{y}_l^{(i)}} (1-\hat{y}_l^{(i)})}{\cancel{\hat{y}_l^{(i)}}}\right) + \sum_{k\neq l}^{C} y_k^{(i)} \times \left(\frac{-x^{(i)} \times \cancel{\hat{y}_k^{(i)}} \times \hat{y}_l^{(i)}}{\cancel{\hat{y}_k^{(i)}}}\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} \left(y_l^{(i)} \times x^{(i)} \times (1-\hat{y}_l^{(i)}) - x^{(i)} \left(\sum_{k\neq l}^{C} y_k^{(i)}\right) \hat{y}_l^{(i)}\right)$$

$\sum_{k=1}^{C} y_k^{(i)} = 1 \quad \therefore \sum_{k\neq l}^{C} y_k^{(i)} = (1 - y_l^{(i)})$

$$= -\frac{1}{n}\sum_{i=1}^{n} \left\{ x^{(i)} \left(y_l^{(i)} - \cancel{y_l^{(i)}\hat{y}_l^{(i)}} - \hat{y}_l^{(i)} + \cancel{y_l^{(i)}\hat{y}_l^{(i)}} \right)\right\}$$

$$= -\frac{1}{n}\sum_{i=1}^{n} x^{(i)} \left(y_l^{(i)} - \hat{y}_l^{(i)}\right)$$

$$= \nabla_{w^{(l)}} J_{CE}(W,b)$$

iv) The gradient of b will follow the same steps

For $l = k$

$$\nabla_{b^l} \hat{y}_k^{(l)} = \frac{\nabla_{b^l} \exp(z_l)}{\sum_{k'=1}^{C} \exp z_{k'}}$$

$$= \frac{\exp x^{(i)T} w^{(l)}}{\sum_{k'=1}^{C} \exp x^{(i)T} w^{(k')}} - \frac{\exp(x^{(i)T} w^{(l)})}{\left(\sum_{k'=1}^{C} \exp(x^{(i)T} w^{(k')})\right)^2} \exp(x^{(i)T} w^{(l)})$$

$$= \left[\frac{\exp(x^{(i)T} w^{(l)})}{\sum_{k'=1}^{C} \exp(x^{(i)T} w^{(k')})} - \frac{(\exp(x^{(i)T} w^{(l)}))^2}{(\sum_{k'=1}^{C} \exp x^{(i)T} w^{(k')})^2}\right]$$

$$= \hat{y}_l^{(i)} - (\hat{y}_l^{(i)})^2$$

$$= \hat{y}_l^{(i)} (1 - \hat{y}_l^{(i)})$$

For $l \neq k$

$$\nabla_{b^{(l)}} \hat{y}_k^{(i)} = \nabla_{b(l)} \left[\frac{\exp(x^{(i)T} w^{(k)})}{\sum_{k'=1}^{C} \exp(x^{(i)T} w^{(k')})}\right]$$

$$= -\frac{\exp(x^{(i)T} w^{(k)})}{(\sum_{k'=1}^{C} \exp x^{(i)T} w^{(k')})^2} \exp(x^{(i)T} w_l)$$

$$= -\frac{\exp(x^{(i)T} w^{(k)})(\exp(x^{(i)T} w_l))}{(\sum_{k'=1}^{C} \exp x^{(i)T} w^{(k')})^2}$$

$$= -\hat{y}_k^{(i)} \hat{y}_l^{(i)}$$

To compute the total gradient of J_{CE} wrt each $b^{(l)}$,

$$\nabla_{b^{(l)}} J_{CE}(w,b) = \frac{-1}{n}\left[y_l^{(i)}\hat{y}_l^{(i)}(1-\hat{y}_l^{(i)}) - \sum_{k\neq l} \frac{y_k^{(i)}\hat{y}_k^{(i)}\hat{y}_l^{(i)}}{\hat{y}_k^{(i)}}\right]$$

$$= \frac{-1}{n}\sum_{i=1}^{n}\left[y_l^{(i)}(1-\hat{y}_l^{(i)}) - \sum_{k\neq l} y_k^{(i)}\hat{y}_l^{(i)}\right]$$

$$= \frac{-1}{n}\sum_{i=1}^{n}\left[\hat{y}^{(i)}\left(y_l^{(i)} + \sum_{k\neq l} y_k^{(i)}\right) - y_l^{(i)}\right]$$

$\sum_k y_k = 1$ and $y_l + \sum_{k\neq l} y_k = 1$.

$$\therefore \nabla_b J_{CE}(w,b) = \frac{-1}{n}\sum_{i=1}^{n}\left(y_l^{(i)} - \hat{y}_l^{(i)}\right)$$

Hence Proved.

```
[Running] python -u "c:\Users\Sumeet\OneDrive - Worcester Polytechnic Institute (wpi.edu)\WPI Sem 2\Deep Learning\Submission\HW3\homework3_SSHANBHAG_UMAHANTI.py"
Finding best hyperparameters from 3*3*3*3 combinations
Best Hyperparameters:

Epochs =  35
Alpha =  0.001
Learning Rate =  1e-05
Mini Batch Size =  32
Cost function value using above hyperparameter values  1.4192011617995408

Now combining Training and validation Sets

Cost & Accuracy values

 Cost =  1.8416920764703457
 Accuracy = 76.51 %



[Done] exited with code=0 in 19.675 seconds
```

```
# Below values are tuned values for the same for highest accuracy
Mb = [32]
Epoch = [35]
Alpha = [0.001]
Eps = [0.00001]
```