Deep Learning Numerical Problems

The MAHE registrar has the complete list of courses taken by each graduating student in a program. This data is represented as a matrix X with m rows (samples) and n columns (features) as follows:

Student \ Course	1	2	⋯	n
1	1	1	⋯	0
2	0	1	⋯	0
⋮	⋮	⋮	⋮	⋮
m	1	0	⋯	1

The entries of the data matrix are 1 and 0s representing whether a particular student has taken a particular course. For example, the red highlighted entry 1 means the 1st student has taken the 1st course and the blue highlighted entry 0 means that the mth student has not taken the 2nd course. Recall that the ith student vector is represented as $X^{(i)}$ and the jth course vector is represented as X_j. Then answer the following questions:

1. The total number of courses the 5th student has taken is given as $X^{(5)^T}1$
2. The term $X^{(3)^T}1$ means total number of courses the 3rd student has taken
3. Total number of students that taken both course 1 and course 6 is $X_1^T X_6$
4. The quantity $\left\|X^{(1)} - X^{(6)}\right\|^2$ means the number of courses that exactly one among the 1st student and the 6th student has taken
5. The term $X^T 1$ means an n-vector whose jth entry is the total number of students in jth course

Consider a logistic regression classifier applied to a sample X (bias feature added) with correct label y using a weights vector W (bias corrected). The forward propagation for calculating the loss function is given as:

$$\boldsymbol{Z = W^T X \rightarrow a = \sigma(Z) \rightarrow L = -\log(a^y \times (1-a)^{1-y})}$$

Calculate the gradient of the loss function using the chain rule with respect to the weight's matrix W

$\nabla_W(L) = \nabla_W(Z) \times \nabla_Z(a) \times \nabla_a(L)$

$\Rightarrow \nabla_W(L) = \nabla_W(W^T X) \times \nabla_Z\big(\sigma(Z)\big) \times \nabla_a[-\log\{a^y \times (1-a)^{1-y}\}]$

$\Rightarrow \nabla_W(L) = X \times \{\sigma(Z)(1-\sigma(Z))\} \times \dfrac{-1}{a^y \times (1-a)^{1-y}}[ya^{y-1}(1-a)^{1-y} + a^y(1-y)(1-a)^{1-y-1} \times (-a)]$

$\Rightarrow \nabla_W(L) = X \times \{a(1-a)\} \times \dfrac{(1-y)a - y(1-a)}{a(1-a)}$

$\Rightarrow \nabla_W(L) = X(a-y)$

For a particular binary classification task, your friend modifies the logistic regression loss function as:

$$\boldsymbol{L = -log\big(a^{\alpha y} \times (1-a)^{\beta(1-y)}\big)}$$

Where, α & β are coefficients set by your friend for a dataset in hand. What kind of binary classification task might your friend be working on?

This setup is used for binary classification task based on an imbalanced dataset. For example, say we have 900 positive samples and 100 negative samples; we choose α & β values such that α down-weights the loss for the positive samples and β up-weights the loss for the negative samples. In this case we have 9 times more positive samples than negative samples, so β should be 9 times of α. For example, if $\alpha = 0.1$, then $\beta = 0.9$

Consider a (5×5) matrix B defined as: $B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$. Then for a 5 vector X how BX and X are related with each other? Calculate the matrix B^2 by matrix-matrix product. From the calculated B^2 matrix, calculate the matrix B^5

$$BX = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}_{5\times5} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}_{5\times1} = \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{5\times1}$$

So, BX is the downward shifted version of X with the topmost shifted value is turned into 0

$$B^2 = B \times B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

So, for the matrix B^2, the first two rows become zeros. Hence for B^5 we will have a zero matrix

For the following vector Z, with simple calculations write the $SoftMax(Z)$ approximately, and answer to which category (among 1,2 and 3) would a sample be predicted to belong to if the raw scores vector Z is given as:

$$Z = \begin{bmatrix} 10^{-10} \\ 10^{6} \\ 10^{-12} \end{bmatrix}$$

The softMax activated raw scores vector from Z is calculated as:

$$SoftMax(Z) = SoftMax\left(\begin{bmatrix} 10^{-10} \\ 10^{6} \\ 10^{-12} \end{bmatrix}\right) = \begin{bmatrix} \frac{10^{-10}}{10^{-10}+10^{6}+10^{-12}} \\ \frac{10^{6}}{10^{-10}+10^{6}+10^{-12}} \\ \frac{10^{-12}}{10^{-10}+10^{6}+10^{-12}} \end{bmatrix} \cong \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

The object belongs to class 2 (in human indexing) and class 1 (in python indexing)

You want to predict whether a patient survived (label 1) or not (label 0) by training a logistic regression model on a dataset with 4 samples namely $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$, corresponding to the correct class labels 1,1,0,0 respectively using full-batch gradient descent. The following results are observed in consecutive epochs:

epoch k	epoch $k+1$
$\sigma\left(w \cdot x^{(1)}\right) = 0.97$	$1 - \sigma\left(w \cdot x^{(1)}\right) = 0.01$
$1 - \sigma\left(w \cdot x^{(2)}\right) = 0.09$	$1 - \sigma\left(w \cdot x^{(2)}\right) = 0.07$
$\sigma\left(w \cdot x^{(3)}\right) = 0.51$	$\sigma\left(w \cdot x^{(3)}\right) = 0.49$
$1 - \sigma\left(w \cdot x^{(4)}\right) = 0.04$	$\sigma\left(w \cdot x^{(4)}\right) = 0.98$

For each epoch, calculate the training loss and the proportion of samples that are correctly classified and compare them across the epochs. What is the main observation?

Remember that in Logistic regression, the term $a^{(i)}$, also denoted as $\hat{y}^{(i)}$ is called the predicted probability vector. It states that the i^{th} sample belongs to label 1. So, the predicted probability that the i^{th} sample will belong to label 0 is given as $\left(1 - a^{(i)}\right)$ *or* $\left(1 - \hat{y}^{(i)}\right)$.

So, combining it all we can write that,

$$a^{(i)} = \begin{cases} \sigma\left(W.x^{(i)}\right) \; if \; i^{th} \; sample \; belong \; to \; label \; 1 \\ 1 - \sigma\left(W.x^{(i)}\right) \; if \; i^{th} sample \; belong \; to \; label \; 0 \end{cases}$$

epoch k, avg. training loss =0.44, training accuracy = 2/4 = 50%

$$\sigma\left(w \cdot x^{(1)}\right) = 0.97 \quad \text{loss} = -\log(0.97)\checkmark$$
$$1 - \sigma\left(w \cdot x^{(2)}\right) = 0.09 \quad \text{loss} = -\log(1 - 0.09)\checkmark$$
$$\sigma\left(w \cdot x^{(3)}\right) = 0.51 \quad \text{loss} = -\log(1 - 0.51)\times$$
$$1 - \sigma\left(w \cdot x^{(4)}\right) = 0.04 \quad \text{loss} = -\log(0.04)\times$$

epoch $k+1$, avg. training loss =0.51, training accuracy = 3/4 = 75%

$$1 - \sigma\left(w \cdot x^{(1)}\right) = 0.01 \quad \text{loss} = -\log(0.99)\checkmark$$
$$1 - \sigma\left(w \cdot x^{(2)}\right) = 0.07 \quad \text{loss} = -\log(0.93)\checkmark$$
$$\sigma\left(w \cdot x^{(3)}\right) = 0.49 \quad \text{loss} = -\log(1 - 0.49)\checkmark$$
$$\sigma\left(w \cdot x^{(4)}\right) = 0.98 \quad \text{loss} = -\log(1 - 0.98)\times$$

It is observed that even if the training loss increases, the accuracy doesn't decrease. So, it can't be stated that lower loss means higher accuracy.

Suppose *X* represents a data matrix (samples along columns) containing information about 100 individuals

with features { gender (female or male), age, weight, BMI, blood glucose, total cholesterol, } and categories { non-diabetic, pre-diabetic, diabetic, severely diabetic. }

Suppose we want to apply a SoftMax classifier to the dataset. What will be the shape of the weight's matrix *W* assuming the bias trick has been performed? In plain English, using the data as context write what the following terms mean starting index from 1: $w_{(:,2)}, w_{(4,:)}, w_{(1,8)}, w_{(2,5)}$

There will be 7 features as input for the weight matrix W given as: genderfemale, gendermale, age, weight, BMI, blood glucose and total cholesterol. There is total 4 output categories. So, assuming that the bias trick is performed, the shape of the weight matrix W will be $\{4 \times (7+1)\} = (4 \times 8)$

$w_{:,2}$ = weights applied to the 2nd feature gendermale to get all output category raw scores.

$w_{4,:}$ = weights applied to all features to get the raw score for the 4th output category severely diabetic.

$w_{1,8}$ = bias for the 1st output category non-diabetic

$w_{2,5}$ = weight applied to the 5th feature BMI to get the 2nd output category pre-diabetic raw score

You want to train a neural network to classify a sample with 19 continuous features into 3 possible categories. How many parameters will have to be trained if you use an 8-layered deep neural network with 8 nodes in each of the hidden layers? Now, you want to train the same model using a 2-layered shallow neural network such that the number of parameters will not exceed the number of parameters of the previous part. How many nodes will this shallow neural network have in the hidden layer?

There is 19 continuous features and 3 output categories.

For an 8-layered deep neural network, layer index starts from 0 (input) and ends at 8 (output). The number of nodes in those 8 layers are:

$$n^{[0]} = 19, n^{[1]} = n^{[2]} = \cdots = n^{[7]} = 8, n^{[8]} = 3$$

So, the shapes of 8 weights matrix with the bias trick are given as:

$$W^{[1]} = 8 \times 20, W^{[2]} = W^{[3]} = \cdots = W^{[7]} = 8 \times 9, W^{[8]} = 3 \times 9$$

So, the total number of parameters will be

$$(8 \times 20) + 6 \times (8 \times 9) + (3 \times 9) = 619$$

For a 2-layered shallow neural network, the layer index starts form 0 (input) and ends at 2 (output). In 0^{th} layer there is 19 features and for the last layer there is 3 nodes as mentioned in the question. So, number of nodes in the first hidden layer (layer index 1) is unknown and say it is x

So, $n^{[0]} = 19, n^{[1]} = x, n^{[2]} = 3$

The shape of the three weights matrix will be $W^{[1]} = (x \times 20)$ and $W^{[2]} = 3 \times (x + 1)$ [bias trick has done]

As number of parameters will not exceed 619, so

$$(x \times 20) + 3 \times (x + 1) \leq 619$$
$$\Rightarrow 23x \leq 616$$
$$\Rightarrow x \leq 26.78$$
$$\Rightarrow x = 26$$

So, there will be 26 nodes in the hidden layer of the 2-layered shallow neural network

However, the deep neural network architecture has more capacity than the shallow neural network and thus is not likely to overfit like shallow neural network. Hence, deep neural network is a better choice.

Consider the following forward propagation through a fully connected deep neural network architecture (256 nodes in hidden layer) for a 32×32 image sample represented as vector X with one hot encoded correct label 3-vector y and predicted probability vector $\hat{y}$ (indexing starts form 0)

$$\underbrace{\mathbf{x}_B}_{?} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \rightarrow \underbrace{\mathbf{z}^{[1]}}_{?} = \underbrace{\mathbf{W}^{[1]}}_{?\times?}\mathbf{x}_B \rightarrow \underbrace{\mathbf{a}^{[1]}}_{?} = \mathrm{ReLU}\left(\mathbf{z}^{[1]}\right) \rightarrow \mathbf{a}_B^{[1]} = \begin{bmatrix} \mathbf{a}^{[1]} \\ 1 \end{bmatrix} \rightarrow \underbrace{\mathbf{z}^{[2]}}_{?} = \underbrace{\mathbf{W}^{[2]}}_{?\times?}\underbrace{\mathbf{a}_B^{[1]}}_{?} \rightarrow \mathbf{a}^{[2]} = \mathrm{softmax}\left(\mathbf{z}^{[2]}\right) \rightarrow L = \sum_{k=0}^{?} -y_k \log\left(\hat{y}_k\right)$$

Identify the missing shapes denoted by '?' in the above picture.

X is a 32×32 image vector that is equal to 1024 samples and considering the bias feature, we get the following shapes as:

$|X_B| = 1025$ or 1025×1

$\left|W^{[1]}\right| = 256 \times 1025$ and $\left|Z^{[1]}\right| = 256$ or 256×1

$\left|a^{[1]}\right| = \left|Z^{[1]}\right| = 256$ or 256×1

$\left|a_B^{[1]}\right| = 257 \times 1$ or 257; $\left|W^{[2]}\right| = 3 \times 257$; $\left|Z^{[2]}\right| = 3 \times 1$ or 3

The k value ranges from 0 to 2

Consider a convolutional neural network defined by the layers in the left column in the table below. Fill in the shape of the output volume and the number of parameters corresponding to each layer using the notations below:

CONVx-N denotes a convolutional layer with N filters with kernel height and width both equal to x. Padding is 2, and stride is 1.

POOL-N denotes an $N \times N$ max-pooling layer with stride of N and no zero padding.

FLATTEN flattens its inputs.

FC-N denotes a fully-connected layer with N neurons.

Fill in the missing values denoted by '?' marks in the below picture:

Layer	Output Volume Shape	Number of Parameters
Input	$32 \times 32 \times 3$	0
CONV3-16	?	?
Leaky ReLU	?	?
POOL-2	?	?
FLATTEN	?	?
FC-10	?	?

CONV3-16 means the filter shape is (3×3) and there are 16 nodes or neurons. So, there are 16 kernel or neurons that have width 3, height 3 and depth 3 (depth is obtained by the last number of input)

Layer	Output Volume Shape	Number of Parameters
Input	$32 \times 32 \times 3$	0
CONV3-16	$34 \times 34 \times 16$	$16 \times (3 \times 3 \times 3 + 1)$
Leaky ReLU	$34 \times 34 \times 16$	0
POOL-2	$17 \times 17 \times 16$	0
FLATTEN	$17 * 17 * 16 = 4624$	0
FC-10	10	$10 \times (4624 + 1)$