MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

AML 5202 | Deep Learning | Sessional-1 Grading Scheme

1. [10 points] [L4, CO 1] You want to predict if a patient survived (label 1) or not (label 0) by training a logistic regression model on a dataset with 4 samples $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$ corresponding to correct labels $1, 1, 0, 0$, respectively, using full-batch gradient descent. The following quantities 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. Are you surprised by the answer? Write your observation in one sentence.

Solution:

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

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

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

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

The model is overfitting; however, the surprising observation is that training loss increases while the training accuracy also increases. Therefore, using just the loss vs. epoch curve to ascertain model accuracy could be misleading.

2. [10 points] [L5, CO 1] Suppose $\mathbf{X}$ represents the data matrix (samples along columns) containing information about 100 individuals

$$\text{with features} \begin{cases} \text{gender (female or male),} \\ \text{age,} \\ \text{weight,} \\ \text{BMI,} \\ \text{blood glucose,} \\ \text{total cholesterol,} \end{cases} \qquad \text{and categories} \begin{cases} \text{non-diabetic,} \\ \text{pre-diabetic,} \\ \text{diabetic,} \\ \text{severely diabetic.} \end{cases}$$

(a) Suppose we want to apply softmax classifier to the dataset. What will be the shape of the weights matrix $\mathbf{W}$ assuming that the bias trick has been done?

Solution: Note that gender is a categorical feature which will be one-hot encoded resulting in the new features genderfemale and gendermale. This results in 7 features:

(1) genderfemale (2) gendermale (3) age (4) weight (5) BMI (6) blood glucose (7) total cholesterol

As there are 4 output categories, the weights matrix W will have shape $4 \times (7+1) = 4 \times 8$, taking into account the bias feature.

(b) In plain English and using the data as context, explain what each of the following represents assuming indexing starts from 1:

$$w_{:,2},\ w_{4,:},\ w_{18},\ w_{25}.$$

Solution:

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

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

w_{18}: bias for the 1^{st} output category non-diabetic.

w_{25}: weight applied to the 5^{th} feature BMI to get the 2^{nd} output category pre-diabetic raw score.

(c) Suppose the gradient of the loss with respect to some weight parameter evaluated at its current value is 4. Justify what will happen to the loss if we increase that weight parameter a little bit while keeping the other parameters fixed? What if we decrease it a little bit?

Solution: A positive gradient implies that the loss will behave the same as the change in the weight parameter; that is, increasing that weight parameter a little bit while keeping the other parameters fixed will result in an increase in the loss, and decreasing that weight parameter a little bit while keeping the other parameters fixed will result in a decrease in the loss.

3. [10 points] [L5, CO 2] You want to train a neural network to classify a sample with 19 continuous features into one of 3 possible categories.

(a) How many parameters will have to be trained if you use an 8-layer deep neural network with 8 nodes in each of the hidden layers?

Solution: We have $n^{[0]} = 19, n^{[1]} = 8, n^{[2]} = 8, n^{[3]} = 8, n^{[4]} = 8, n^{[5]} = 8, n^{[6]} = 8, n^{[7]} = 8, n^{[8]} = 3$ resulting in the following shapes for the weights matrices for each layer with the bias trick done:

$$\mathbf{W}^{[1]} : 8 \times 20,\ \mathbf{W}^{[2]} : 8 \times 9,\ \mathbf{W}^{[3]} : 8 \times 9,\ \mathbf{W}^{[4]} : 8 \times 9,\ \mathbf{W}^{[5]} : 8 \times 9,\ \mathbf{W}^{[6]} : 8 \times 9,\ \mathbf{W}^{[7]} : 8 \times 9,\ \mathbf{W}^{[8]} : 3 \times 9.$$

This results in a total of $8 \times 20 + 8 \times 9 + 8 \times 9 + 8 \times 9 + 8 \times 9 + 8 \times 9 + 8 \times 9 + 3 \times 9 = 619$ parameters.

(b) Now you want to train using a 2-layer shallow neural network such that the number of parameters will not exceed the number identified in the previous part for the deep neural network. How many nodes will this shallow neural network have in the hidden layer?

Solution: We have $n^{[0]} = 19, n^{[1]} = ?, n^{[2]} = 3$ resulting in the following shapes for the weights matrices for each layer with the brick done:

$$\mathbf{W}^{[1]} : n^{[1]} \times 20,\ \mathbf{W}^{[2]} : 3 \times \left(n^{[1]} + 1\right).$$

This results in a total of $n^{[1]} \times 20 + 3 \times \left(n^{[1]} + 1\right)$ parameters. This should not exceed 619 from the previous part, so we need

$$n^{[1]} \times 20 + 3 \times \left(n^{[1]} + 1\right) \leq 619 \Rightarrow n^{[1]} \leq 26.8 \Rightarrow n^{[1]} = 26 \text{ nodes.}$$

(c) Which architecture would be preferable, the deep or the shallow one? Justify using one or two sentences at maximum.

Solution: A deep architecture with more capacity would be preferable than a shallow one with more complexity as a shallow network will immediately tend to overfit the output to the input.

4. [10 points] [L2, CO 2] 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 $\mathbf{x}$ with one-hot encoded correct label 3-vector $\mathbf{y}$ and predicted probability vector $\hat{\mathbf{y}}$ (indexing starts from 0):

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

Identify the missing shapes corresponding to the question marks.

Solution:

$$\boxed{\underbrace{\mathbf{x}_B}_{1025} = \begin{bmatrix}\mathbf{x}\\1\end{bmatrix}} \rightarrow \boxed{\underbrace{\mathbf{z}^{[1]}}_{256} = \underbrace{\mathbf{W}^{[1]}}_{256\times1025}\mathbf{x}_B} \rightarrow \boxed{\underbrace{\mathbf{a}^{[1]}}_{256} = \text{ReLU}\left(\mathbf{z}^{[1]}\right)} \rightarrow \boxed{\mathbf{a}_B^{[1]} = \begin{bmatrix}\mathbf{a}^{[1]}\\1\end{bmatrix}} \rightarrow \boxed{\underbrace{\mathbf{z}^{[2]}}_{3} = \underbrace{\mathbf{W}^{[2]}}_{3\times257}\underbrace{\mathbf{a}_B^{[1]}}_{257}} \rightarrow \boxed{\mathbf{a}^{[2]} = \text{softmax}\left(\mathbf{z}^{[2]}\right)} \rightarrow \boxed{L = \sum_{k=0}^{2} -y_k \log\left(\hat{y}_k\right)}.$$

5. [10 points] [L5, CO 3] Suppose we have the following raw scores vector corresponding to dense layer 4 of a deep neural network:

$$\mathbf{z}^{[4]} = \begin{bmatrix}-10\\1\\10\\-1\end{bmatrix}.$$

(a) Calculate the local gradient of activation layer 4 which is tanh activated. Round your calculations to 2 decimal places.

Solution: The local gradient of activation layer 4 is $\nabla_{\mathbf{z}^{[4]}}\left(\mathbf{a}^{[4]}\right)$ where

$$\mathbf{a}^{[4]} = \tanh\left(\mathbf{z}^{[4]}\right) = \begin{bmatrix} \tanh\left(z_1^{[4]}\right) \\ \tanh\left(z_2^{[4]}\right) \\ \tanh\left(z_3^{[4]}\right) \\ \tanh\left(z_4^{[4]}\right) \end{bmatrix} \Rightarrow \nabla_{\mathbf{z}^{[4]}}\left(\mathbf{a}^{[4]}\right) = \begin{bmatrix} \nabla_{z_1^{[4]}}\left(a_1^{[4]}\right) & \nabla_{z_1^{[4]}}\left(a_2^{[4]}\right) & \nabla_{z_1^{[4]}}\left(a_3^{[4]}\right) & \nabla_{z_1^{[4]}}\left(a_4^{[4]}\right) \\ \nabla_{z_2^{[4]}}\left(a_1^{[4]}\right) & \nabla_{z_2^{[4]}}\left(a_2^{[4]}\right) & \nabla_{z_2^{[4]}}\left(a_3^{[4]}\right) & \nabla_{z_2^{[4]}}\left(a_4^{[4]}\right) \\ \nabla_{z_3^{[4]}}\left(a_1^{[4]}\right) & \nabla_{z_3^{[4]}}\left(a_2^{[4]}\right) & \nabla_{z_3^{[4]}}\left(a_3^{[4]}\right) & \nabla_{z_3^{[4]}}\left(a_4^{[4]}\right) \\ \nabla_{z_4^{[4]}}\left(a_1^{[4]}\right) & \nabla_{z_4^{[4]}}\left(a_2^{[4]}\right) & \nabla_{z_4^{[4]}}\left(a_3^{[4]}\right) & \nabla_{z_4^{[4]}}\left(a_4^{[4]}\right) \end{bmatrix}$$

$$\Rightarrow \nabla_{\mathbf{z}^{[4]}}\left(\mathbf{a}^{[4]}\right) = \begin{bmatrix} 1-\tanh\left(z_1^{[4]}\right)^2 & 0 & 0 & 0 \\ 0 & 1-\tanh\left(z_2^{[4]}\right)^2 & 0 & 0 \\ 0 & 0 & 1-\tanh\left(z_3^{[4]}\right)^2 & 0 \\ 0 & 0 & 0 & 1-\tanh\left(z_4^{[4]}\right)^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1-\tanh(-10)^2 & 0 & 0 & 0 \\ 0 & 1-\tanh(1)^2 & 0 & 0 \\ 0 & 0 & 1-\tanh(10)^2 & 0 \\ 0 & 0 & 0 & 1-\tanh(-1)^2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.42 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.42 \end{bmatrix}.$$

(b) Clearly state for which nodes of dense layer 4, learning of the corresponding parameters will be very minimal? Justify your answer briefly. What could you do to improve the learning of the parameters of all nodes in dense layer 4?

Solution: Weights associated with nodes 1 and 3 of dense layer 4 will have practically zero updates after the gradient is calculated. This is because the backward flowing gradient from activation layer 4 through those nodes is almost 0. We could replace tanh activation by Leaky ReLU activation to ensure that all nodes in layer 4 learn their weights.