Midterm Exam I
ECE 685D– Introduction to Deep Learning
Fall 2024

Instructor: Prof. Vahid Tarokh
ECE Department, Duke University

Oct 9th 2024
10:05 AM - 11:20 AM

Name: ____
Duke ID: ____

This exam contains 10 pages and 10 questions. This is a closed-book exam. No exam aids are permitted except for a one-sided letter-sized cheat sheet. Communication with others is strictly prohibited.

Distribution of Marks

Question	Points	Score
1	3	
2	3	
3	3	
4	3	
5	3	
6	3	
7	5	
8	25	
9	35	
10	17	
Total:	100	

Definition of Leaky ReLU function. Given the non-negative coefficient α (i.e., $\alpha < 1$) and the input x, the leaky ReLU is defined as:

$$LeakyReLU(x, \alpha) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

For the multiple-choice questions, circle ALL of the correct answers. These questions can have more than one correct answer.

A.

1. (3 points) After training a convolutional neural network, you observe a large gap between the training accuracy (e.g., 99%) and the test accuracy (e.g., 50%). Which of the following methods is commonly used to resolve this issue?
   (a) Dropout
   (b) Leaky ReLU
   (c) Sigmoid activation

A, C

2. (3 points) Which of the following activation functions can lead to vanishing gradients?
   (a) Sigmoid
   (b) Leaky ReLU
   (c) Tanh

B.

3. (3 points) Which of the following is true about Batchnorm
   (a) Batchnorm is another way of performing dropout.
   (b) Batchnorm makes the training process converge faster.
   (c) Batchnorm is a non-linear transformation to center the dataset around the origin.

C.

4. (3 points) If the size of the input is 64×64×16 (i.e., number of channels = 16), how many parameters are there, including bias, in a convolution filter with kernel size=1, stride=1, padding=0?
   (a) 1
   (b) 2
   (c) 17

Weight param = K × K × # channels.

Total = weight param + # bias. (= # filter).

Each filter has its own bias

(d) 4097

The # filters determine # output channels.

5. (3 points) Consider an input of dimensions (n_h, n_w, n_c), where n_c is the number of channels. A convolutional layer with kernel size=1, stride=1, padding=0 is applied to this input. After the convolution, a standard max-pooling layer (e.g., stride=2) is applied. Which of the following statements about the output after both operations is correct?

~~C.~~

B.C

(a) The convolution can help reduce the dimension of n_h, n_w, but not n_c.
(b) The convolution can help reduce the dimension of n_c, but not n_h, n_w.
(c) The standard maxpooling can help reduce the dimension of n_h, n_w, but not n_c.
(d) The standard maxpooling can help reduce the dimension of n_c, but not n_h, n_w.

6. (3 points) Recall that the L_p norm of $\mathbf{w} = [w_1, w_2, ..., w_n]^T$ is defined as:

B.

$$||\mathbf{w}||_p = \left(\sum_i^n |w_i|^p\right)^{1/p}.$$

Which of the following regularization metrics is convex and most likely leads to weight sparsity?

(a) $L_{1/2}$
(b) L_1
(c) L_2
(d) L_∞

	Convex	Sparsity	
$L_{\frac{1}{2}}$	✗	✓	strong sparsity
L_1	✓	✓	Lasso, drives weights to 0.
L_2	✓	✗	weights rarely to 0.
L_∞	✓	✗	

7. (5 points) What are the advantages of using convolutional layers instead of fully connected layers for visual tasks (e.g., image classification, object detection)? Explain your answer.

Fully connected layers have too many params.

not practical to train.

8. Consider an image X with 2 channels. The image is represented as a $4 \times 4 \times 2$ matrix where the last dimension correspond to the number of channel of this image. We denote a kernel W with dimension $3 \times 3 \times 2$. The values of the image pixels and weights of the kernel are given below.

$$X[:,:,0] = \begin{bmatrix} 3 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 3 \end{bmatrix} \qquad W[:,:,0] = \begin{bmatrix} -1 & 1 & 1 \\ 0 & -1 & 0 \\ -1 & 1 & 1 \end{bmatrix}$$

$$X[:,:,1] = \begin{bmatrix} 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 \end{bmatrix} \qquad W[:,:,1] = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

The output of the convolutional filter is given as:

$$Y = LeakyReLU\left(X' * W - \mathbb{1}_{4\times 4}, 0.1\right), \tag{1}$$

where Y is the output image, X' is the input image after applying zero-padding around the edges (i.e. each channel is converted to a 6×6 matrix such that a row of zeros is added to the top and bottom and a column of zeros is added to the left and right.), $X' * W$ is the convolution between X' and W with stride size=1, and $\mathbb{1}_{4\times 4}$ is a 4×4 matrix with elements equal to one. Here, α is the coefficient of the LeakyReLU function, where the leaky ReLU function is defined as:

$$LeakyReLU(x, \alpha) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha x & \text{if } x < 0. \end{cases}$$

(a) (15 points) Compute the output Y of the image X.

(b) (5 points) Apply max pooling on non-overlapping 2×2 sub-matrices of the output image and compute the output.

(c) (5 points) Apply average pooling on non-overlapping 2×2 sub-matrices of the output image and compute the output.

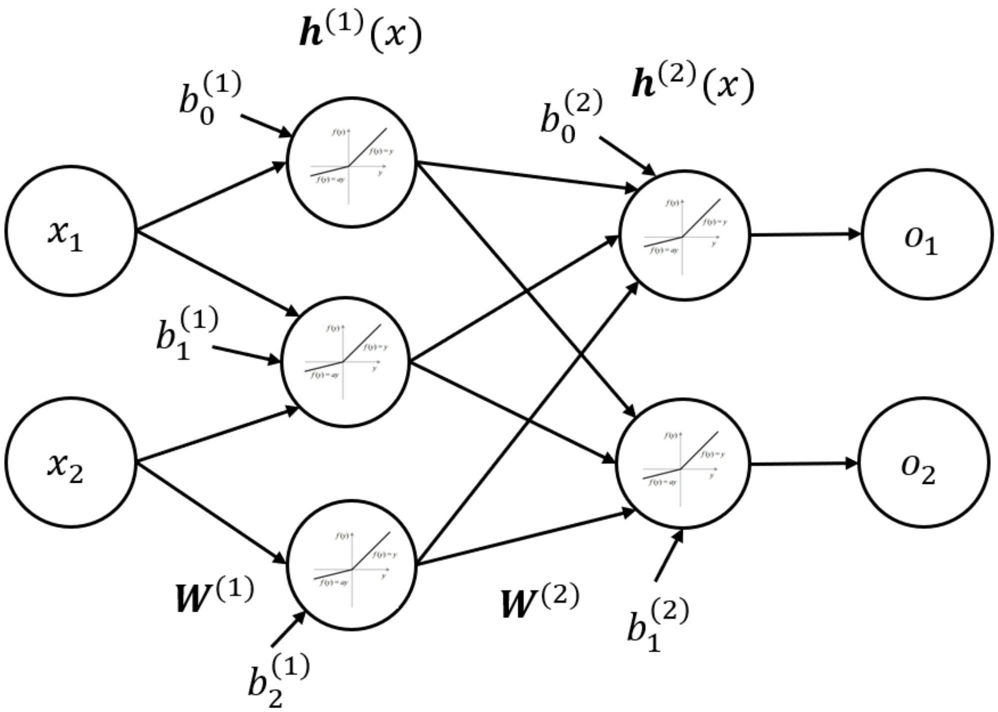
9. Given the neural network with $L = 2$ hidden layers, the input units $\mathbf{x} = [x_1, x_2]^T$, and LeakyReLU activations (as shown below). The weights and bias of hidden units are denoted $\mathbf{W}$ and $\mathbf{b}$. The hidden layers are defined as below:

$$\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x} = [x_1, x_2]^T$$

$$\mathbf{h}^{(i)}(\mathbf{x}) = \sigma\left(\mathbf{b}^{(i)} + \mathbf{W}^{(i)}\mathbf{h}^{(i-1)}(\mathbf{x})\right), \text{for } i = \{1, 2\}$$

where σ is the LeakyReLU activation with the coefficient $\alpha = 0.5$. The output $\mathbf{o} = [o_1, o_2]^T$ is defined as:

$$\mathbf{o}(\mathbf{x}) = \mathbf{h}^{(2)}(\mathbf{x})$$



The values of the inputs are $\mathbf{x} = [1, 2]^T$. The ground truth values of the output are $\mathbf{t} = [4, 1]^T$. The weights of the network are given as follows:

$$\mathbf{W}^{(1)} = \begin{bmatrix} -0.5 & 0 \\ 0.5 & 1 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{W}^{(2)} = \begin{bmatrix} 0.5 & 2 & 0.5 \\ 1 & -0.5 & -2 \end{bmatrix}$$

$$\mathbf{b}^{(1)} = \begin{bmatrix} 2.5 & -0.5 & 1 \end{bmatrix}^T \qquad \mathbf{b}^{(2)} = \begin{bmatrix} -5 & -1 \end{bmatrix}^T$$

(a) (10 points) Compute the output $[o_1, o_2]^T$ of the input $[x_1, x_2]^T$ using the network parameters as specified above. Write down all calculations of the intermediate layers.

(b) (5 points) Compute the mean squared error (MSE) between the output $[o_1, o_2]^T$ and the target ground truth $[t_1, t_2]^T$.

(c) (5 points) Using the calculated MSE, update the weight $W_{0,1}^{(2)}$ (i.e., entry $\{0, 1\}$ of matrix $\mathbf{W}^{(2)}$) using gradient descent and the backpropagation algorithm with the learning rate of 0.1. You do not need to simplify your answer.

(d) (5 points) Similarly, update the weight $W_{1,1}^{(2)}$ (i.e., entry $\{1, 1\}$ of matrix $\mathbf{W}^{(2)}$) using gradient descent and the backpropagation algorithm with the learning rate of 0.1. You do not need to simplify your answer.

(e) (10 points) Update the weight $W_{1,0}^{(1)}$ (i.e., entry $\{1, 0\}$ of matrix $\mathbf{W}^{(1)}$) using gradient descent and the backpropagation algorithm with a learning rate of 0.1. Write down the details of the backward pass. You do not need to simplify your answer.

(a).
$$z^{(1)}(x) = W^{(1)} \cdot h^{(0)}(x) + b^{(1)} = \begin{bmatrix} -0.5 & 0 \\ 0.5 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2.5 \\ -0.5 \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} -0.5 \\ 2.5 \\ 2 \end{bmatrix} + \begin{bmatrix} 2.5 \\ -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}$$

$$h^{(1)}(x) = \sigma(z^{(1)}(x)) = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}$$
$$z^{(2)}(x) = W^{(2)} \cdot h^{(1)}(x) + b^{(2)} = \begin{bmatrix} 0.5 & 2 & 0.5 \\ 1 & -0.5 & -2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} -.5 \\ -1 \end{bmatrix}$$
$$= \begin{bmatrix} 6.5 \\ -5 \end{bmatrix} + \begin{bmatrix} -.5 \\ -1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -6 \end{bmatrix}$$

$$h^{(2)}(x) = \sigma(z^{(2)}(x)) = \begin{bmatrix} 1.5 \\ -3 \end{bmatrix}$$

$$[o_1, o_2]^T = h^2(x)^T = [1.5 \quad -3]^T.$$

(b).
$$L = \frac{1}{2} \sum_{i=1}^{2} (t_i - o_i)^2 = \frac{1}{2} \cdot ((4 - 1.5)^2 + (1+3)^2) = 11.125.$$

(c).
$$\frac{\partial L}{\partial W^{(2)}_{0,1}} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h} \cdot \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial W^{(2)}_{0,1}}$$

$$= (o_1 - t_1) \cdot 1 \cdot 1 \cdot 2 = -5.$$

$$W^{(2)}_{0,1} = W^{(2)}_{0,1} - 0.1 \times -5 = 2.5$$

(d).
$$\frac{\partial L}{\partial W^{(2)}_{1,1}} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h} \cdot \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial W^{(2)}_{1,1}}$$

$$= (o_2 - t_2) \cdot 1 \cdot 0.5 \cdot 2 = -4.$$

$$W^{(2)}_{1,1} = W^{(2)}_{1,1} - 0.1 \times -4 = -0.1$$

(e).
$$\frac{\partial L}{\partial W^{(1)}_{1,0}} = \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial h^{(2)}_1} \cdot \frac{\partial h^{(2)}_1}{\partial z^{(2)}_1} \cdot \frac{\partial z^{(2)}_1}{\partial h^{(1)}_2} \cdot \frac{\partial h^{(1)}_2}{\partial z^{(1)}_2} \cdot \frac{\partial z^{(1)}_2}{\partial W^{(1)}_{1,0}}$$

$$+ \frac{\partial L}{\partial o_2} \cdot \frac{\partial o_2}{\partial h^{(2)}_2} \cdot \frac{\partial h^{(2)}_2}{\partial z^{(2)}_2} \cdot \frac{\partial z^{(2)}_2}{\partial h^{(1)}_2} \cdot \frac{\partial h^{(1)}_2}{\partial z^{(1)}_2} \cdot \frac{\partial z^{(1)}_2}{\partial W^{(1)}_{1,0}}$$

$$= (o_1 - t_1) \cdot 1 \cdot 1 \cdot 2 \cdot 1 \cdot 1 + (o_2 - t_2) \cdot 1 \cdot 0.5 \cdot -0.5 \cdot 1 \cdot 1$$

$$= -5 + 1 = -4.$$

$$W^{(1)}_{1,0} = W^{(1)}_{1,0} - 0.1 \times -4 = 0.9.$$

10. The L_1 regularization of the model parameter $\mathbf{w} = [w_1, w_2, ..., w_n]^T$ is defined as:

$$||\mathbf{w}||_1 = \sum_i^n |w_i|.$$

Below, we apply L_1 regularization to a neural network model, of which the original objective function is defined as $J(\mathbf{w}; X, y)$, where $\mathbf{w}$ is the model weights, X is the input, and y is the ground truth labels. The regularized objective function after adding the L_1 normalization term becomes:

$$J_R(\mathbf{w}; X, y) = J(\mathbf{w}; X, y) + \alpha||\mathbf{w}||_1.$$

(a) (2 points) Write the gradient $\partial J_R/\partial \mathbf{w}$ in terms of $\partial J/\partial \mathbf{w}$.

(b) (15 points) Finding the closed form solution for the root $\mathbf{w}_R = [w_{R1}, w_{R2}, ..., w_{Rn}]^T$ of $\partial J_R/\partial \mathbf{w} = 0$ is difficult. Hence, we apply Taylor expansion to approximate $J_R(\mathbf{w}; X, y)$, and discard the high-order terms:

$$\hat{J}(\mathbf{w}; X, y) \approx J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T H(\mathbf{w} - \mathbf{w}^*),$$

where H is the Hessian matrix, and $\mathbf{w}^*$ is the solution of $\partial J/\partial \mathbf{w} = 0$ (i.e., the optimal parameter for the objective function J without the regularization term). Suppose that it is known that H is a diagonal matrix with $H = diag([H_{1,1}, ..., H_{n,n}])$, where $H_{i,i} > 0, \forall i \in n$. Write down a closed form expression for the root of $\mathbf{w}_R$.

(a). $\frac{\partial J_R}{\partial W} = \frac{\partial J}{\partial W} + \cancel{\lambda}\ \alpha \cdot \mathrm{sign}(w).$

(b).