CS230: Deep Learning

Fall Quarter 2022

Stanford University

Midterm Examination

Suggested duration: 180 minutes

Problem	Full Points	Your Score
Multiple Choice	14	
Short Answer	30	
Feed-Forward Neural Network	15	
Backpropagation	19	
Discrete Functions in Neural Networks	11	
Debugging Code	18	
Total	88	

The exam contains 14 pages including this cover page.

- This exam is open book, but collaboration with anyone else, either in person or online, is strictly forbidden pursuant to The Stanford Honor Code.

- In all cases, and especially if you're stuck or unsure of your answers, **explain your work, including showing your calculations and derivations!** We'll give partial credit for good explanations of what you were trying to do.

Name: _____

SUNETID: _____@stanford.edu

The Stanford University Honor Code:
I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

Signature: _____

Question (Multiple Choice, 14 points)

For each of the following questions, circle the letter of your choice. Each question has AT LEAST one correct option unless explicitly mentioned. No explanation is required.

(a) **(2 points)** Imagine you are tasked with building a model to diagnose COVID-19 using chest CT images. You are provided with 100,000 chest CT images, 1,000 of which are labelled. Which learning technique has the best chance of succeeding on this task? **(SELECT ONLY ONE)**

   (i) Transfer Learning from a ResNet50 that was pre-trained on chest CT images to detect tumors

   (ii) Train a GAN to generate synthetic labeled data and train your model on all the ground truth and synthetic data

   (iii) Supervised Learning directly on the 1,000 labeled images

   (iv) Augment the labeled data using random cropping and train using supervised learning

(b) **(2 points)** Imagine you are tasked with training a lane detection system that can detect between two different types of lanes: lanes in the same direction as the car moves and lanes in the opposite direction. Assume all the images are taken in common two-way streets in California from a car's front-view camera. What are the following data augmentation techniques can be used for your task?

   (i) Flipping vertically (across x-axis)

   (ii) Flipping horizontally (across y-axis)

   (iii) Adding artificial fog to your images

   (iv) Applying random masking to a (small) portion of the image

(c) **(2 points)** You are training a binary classifier and are unsatisfied with the F1-score as a good metric to combine precision and recall into a single number. You are considering alternatives to F1-score. Which of the following would be reasonable candidate metric(s):

   (i) |precision - recall|

   (ii) recall/precision

   (iii) precision $\times$ recall

   (iv) max(precision, recall)

(d) **(2 points)** Dropout can be considered as a form of ensembling over variants of a neural network. Consider a neural network with N nodes, each of which can be dropped during training independently with a probability $0 < p < 1$. What is the total number of unique models that can be realized on applying dropout?

(i) $\lfloor N \times p \rfloor$

(ii) $(\lfloor N \times p \rfloor)^N$

(iii) $2^{\lfloor N \times p \rfloor}$

(iv) 2^N

(e) **(2 points)** In practice when using Early Stopping, one needs to set a "buffer" hyperparameter, which determines the number of epochs model training continues when no improvement in validation performance is observed before training is terminated. After training is terminated, the model with the best validation performance is used. What is the benefit of setting the buffer parameter to a value $k = 5$ epochs instead of 0:

   (i) Robustness to noise in validation performance from epoch to epoch

   (ii) Reduced training time on average

   (iii) Reduced inference time on average

   (iv) None of the above

(f) **(2 points)** Suppose that you are training a deep neural network and observe that the training curve contains a lot of oscillations, especially at early stages of training. Which of the following techniques can help stabilize training?

   (i) Early stopping

   (ii) Learning rate scheduling

   (iii) Data augmentation

   (iv) Gradient clipping

(g) **(2 points)** You have a 2-layer MLP with Sigmoid activations in the hidden layers that you want to train with SGD. Your network weights are initialized from $\mathcal{N}(10, 1)$. From the very first epoch, you observe that some weights in the first layer are not getting updated or are updated very slowly compared to the second layer. Which of the following can fix this issue?

   (i) Initializing the weights to be from $\mathcal{N}(0, 1)$

   (ii) Adding more hidden layers

   (iii) Switching the activation function to tanh

   (iv) Switching the activation function to leaky ReLU

Question (Short Answer, 30 points)

The questions in this section can be answered in less than 3 sentences. Please be concise in your responses.

(a) Imagine that you are building an app to optimize wait times in US emergency rooms while prioritizing severe cases. You build a deep learning-based app that works as follows:

- **Input:** a patient's demographic information (i.e, ethnicity, age), health history and reasons for emergency

- **Output:** ranking of patients currently in the emergency room from most to least severe

You trained and tested your model using 3 months worth of data from hospitals in the US, before deploying it to several hospitals in the San Francisco Bay Area.

(i) **(2 points)** Now you want to deploy your app internationally. Do you think your app will work well? Why or why not?

You noticed that the app tends to rank African American and Hispanic patients lower than patients from other ethnic backgrounds, even if those patients came into the emergency department with more severe cases.

(ii) **(1 point)** Why is this a problem?

(iii) **(2 points)** What may have caused this problem?
Hint: Think about how the model was trained and the input data that was provided

(iv) **(2 points)** How can we fix this problem?

(b) Graph Neural Networks (GNNs) are a family of neural networks that can operate on graph-structured data. Here, we describe a basic 2-layer GNN. Consider a graph with k nodes labeled $\{1, 2, \ldots, k\}$. For simplicity, assume that each node i is associated with a scalar input x_i. The first layer of our GNN, parameterized by scalar parameters $w^{[1]}$ and $b^{[1]}$ performs the following operation to compute $a_i^{[1]}$ at each node i:

$$a_i^{[1]} = \text{ReLU}\left(x_i + w^{[1]}\left(\sum_{n \in N(i)} x_n\right) + b^{[1]}\right) \tag{1}$$

where $N(i)$ is the set of neighbors of node i in the graph (i.e, all nodes that share an edge with node i). The second layer, parameterized by scalar parameters $w^{[2]}$ and $b^{[2]}$, analogously computes $a_i^{[2]}$ for each node i:

$$a_i^{[2]} = \text{ReLU}\left(a_i^{[1]} + w^{[2]}\left(\sum_{n \in N(i)} a_n^{[1]}\right) + b^{[2]}\right) \tag{2}$$

Answer the following questions for the graph in the figure below, with labels as shown in the nodes.

(i) **(2 points)** What is $\partial a_1^{[2]}/\partial x_6$?
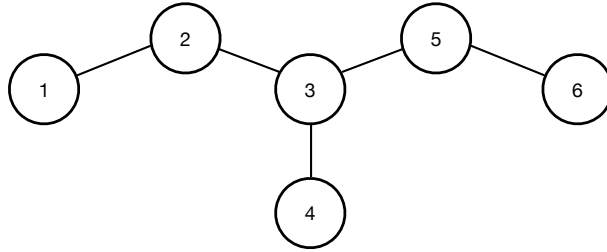
(ii) **(2 points)** You are allowed to add one additional node (suppose this is node 7) and accompanying edges such that the value of $\partial a_1^{[2]}/\partial x_6$ changes from the value computed in part (i). Describe how you would do this with fewest number of edges accompanying node 7.

(c) Consider the graph in figure below representing the training procedure of a GAN. The figure shows the cost function of the generator plotted against the output of the discriminator when given a generated image $G(z)$. Concerning the discriminator's output, we consider that 0 means that the discriminator thinks the input "has been generated by G", whereas 1 means the discriminator thinks the input "comes from the real data".
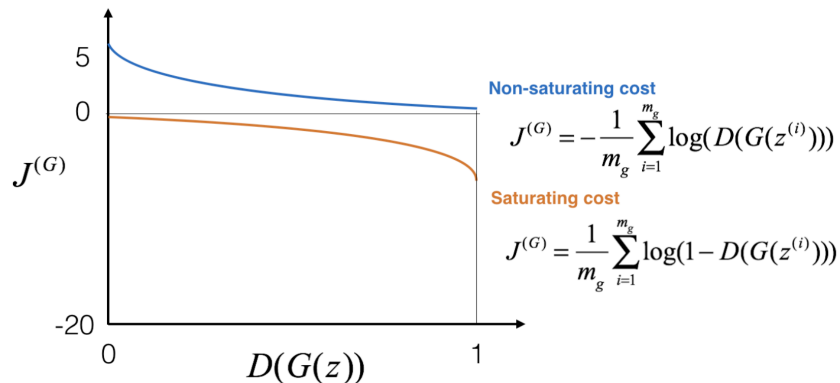

Figure 1: GAN training curve

(i) **(2 points)** After one round of training the generator and discriminator, is the value of $D(G(z))$ closer to 0 or closer to 1? Explain.

(ii) **(2 points)** Two cost functions are presented in Figure 1 above. Which one would you choose to train your GAN? Justify your answer.

(iii) **(2 points)** True or false. Your GAN is finished training when $D(G(z))$ is close to 1. Please explain your answer for full credit.

(d) We would like to train a self-supervised generative model that can learn encodings z of a given input image X by reconstructing the same input image as $\hat{X}$. For our example, lets say our input images are MNIST digits. Consider the architecture shown below:
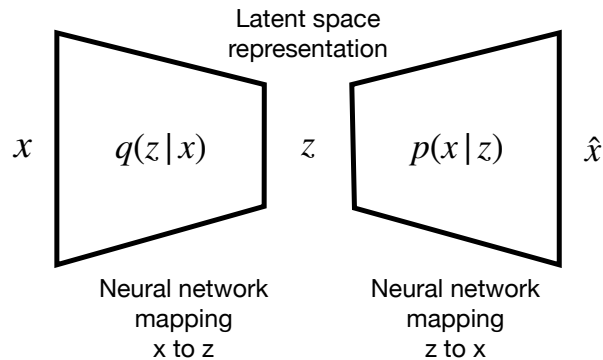
Figure 2: Architecture of proposed generative model

Assume the encoder $q(z \mid x)$ is parameterized to output a normal distribution over z. Alice, Bob and Carol propose 3 different loss functions to train this model end-to-end.

- **Alice:** $\mathbb{KL}(q(z \mid x) \,||\, \mathcal{N}(0, \mathrm{I}))$
- **Bob:** $\mathrm{MSE}(X - \hat{X}) + \mathbb{KL}(q(z \mid x) \,||\, \mathcal{N}(0, \mathrm{I}))$
- **Carol:** $\mathrm{MSE}(X - \hat{X})$

The entire network is end-to-end differentiable for all 3 loss functions.

Here $\mathbb{KL}$ is the KL-divergence which is a measure of similarity of two different probability distributions. $\mathcal{N}(0, \mathrm{I})$ is the multivariate standard Normal distribution where I is the identity matrix. MSE is the mean squared error.

(i) **(3 points)** In plain English, intuitively, explain what each loss function is trying to optimize.

(ii) **(3 points)** Say we choose the dimension of z to be 2 so we can plot the z's on a graph. Consider the three graphs below where each of the two axes is a dimension of z. The different colours indicate different MNIST digits as indicated by the legend. The plots are numbered left to right as (1), (2) and (3).

Match each graph to Alice, Bob and Carol (draw lines connecting the two columns if you printed the midterm) and explain your reasoning for each.

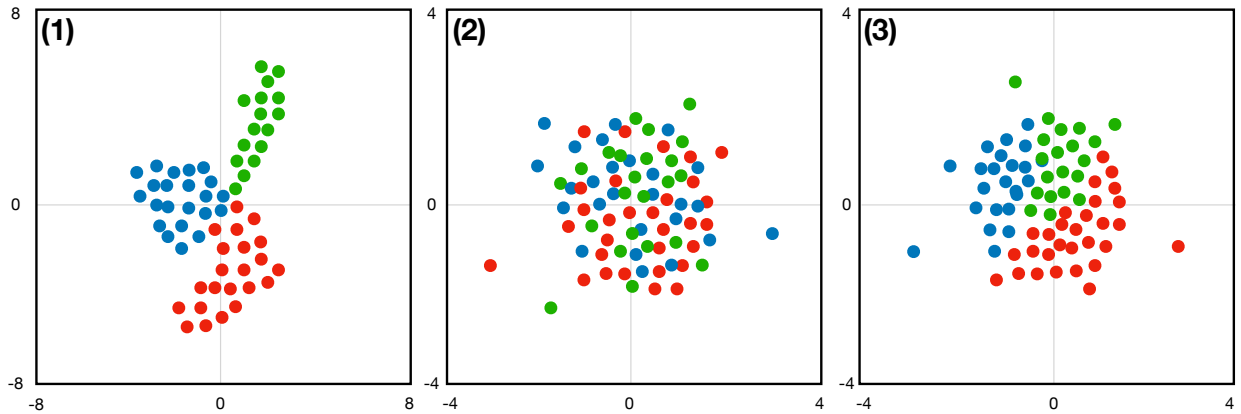|	
Alice	**(1)**
Bob	**(2)**
Carol	**(3)**

Figure 3: Plotted graphs for different loss functions. Plots are numbered, left to right as (1), (2) and (3).

Question (Backpropagation, 19 points)

Consider the following neural network with arbitrary dimensions (ie, $\mathbf{x}$ is not necessarily 5-dimensional, etc.):

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{h} = \text{ReLU}(\mathbf{z}^{[1]})$$
$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{h} + \mathbf{b}^{[2]}$$
$$\hat{\mathbf{y}} = \sigma(\mathbf{z}^{[2]})$$
$$\mathcal{L} = \sum_{i=1}^{k} \max(0, 1 - \mathbf{y}_i\hat{\mathbf{y}}_i)$$

where σ is the sigmoid activation function, and $\odot$ is the operator for element-wise products, and $\mathbf{y}$ is a k-dimensional vector of 1's and 0's. Note that $\mathbf{y}_i$ represents the i-th element of vector $\mathbf{y}$, and likewise for $\hat{\mathbf{y}}_i$.

(i) **(3 points)** What is $\partial\mathcal{L}/\partial\hat{\mathbf{y}}_i$? You must write the most reduced form to get full credit.

(ii) **(2 points)** What is $\partial\mathcal{L}/\partial\hat{\mathbf{y}}$? Refer to this result as $\overline{\hat{\mathbf{y}}}$. Please write your answer according to the shape convention, i.e., your result should be the same shape as $\hat{\mathbf{y}}$.

(iii) **(2 points)** What is $\partial\mathcal{L}/\partial\mathbf{z}^{[2]}$? Refer to this result as $\overline{\mathbf{z}^{[2]}}$. To receive full credit, your answer must include $\overline{\hat{\mathbf{y}}}$ and your answer must be in the most reduced form.

(iv) **(2 points)** What is $\partial\mathcal{L}/\partial\mathbf{W}^{[2]}$? Please refer to this result as $\overline{\mathbf{W}^{[2]}}$. Please include $\overline{\mathbf{z}^{[2]}}$ in your answer.

(v) **(2 point)** What is $\partial\mathcal{L}/\partial\mathbf{b}^{[2]}$? Please refer to this result as $\overline{\mathbf{b}^{[2]}}$. Please include $\overline{\mathbf{z}^{[2]}}$ in your answer.

(vi) **(2 points)** What is $\partial\mathcal{L}/\partial\mathbf{h}$? Please refer to this result as $\overline{\mathbf{h}}$. Please include $\overline{\mathbf{z}^{[2]}}$ in your answer.

(vii) **(2 points)** What is $\partial\mathcal{L}/\partial\mathbf{z}^{[1]}$? Refer to this result as $\overline{\mathbf{z}^{[1]}}$. Please include $\overline{\mathbf{h}}$ in your answer.

(viii) **(2 point)** What is $\partial\mathcal{L}/\partial\mathbf{W}^{[1]}$? Please refer to this result as $\overline{\mathbf{W}^{[1]}}$. Please include $\overline{\mathbf{z}^{[1]}}$ in your answer.

(ix) **(2 point)** What is $\partial\mathcal{L}/\partial\mathbf{b}^{[1]}$? Please refer to this result as $\overline{\mathbf{b}^{[1]}}$. Please include $\overline{\mathbf{z}^{[1]}}$ in your answer.

Question (Discrete Functions in Neural Networks, 11 points)

In this problem, we will explore training neural networks with discrete functions. Consider a neural network encoder $\mathbf{z} = \text{softmax}[f_\theta(\mathbf{X})]$. You can think of f_θ as an MLP for this example. $\mathbf{z}$ is the softmax output and we want to discretize this output into a one-hot representation before we pass it into the next layer. Consider the operation `one_hot` where `one_hot`(z) returns a one-hot vector where the 1 is at the argmax location. For example, `one_hot`$([0.1, 0.5, 0.4]) = [0, 1, 0]$. Say we want to pass this output to another FC layer g_ϕ to get a final output y.

(i) **(1 points)** Is there a problem with the neural network defined below?

$$y = g(\text{one\_hot}(\text{softmax}(f(\mathbf{X}))))$$

(ii) **(2 points)** Consider the following function:

$$z = S_\tau(f(\mathbf{X})) = \text{softmax}(f(\mathbf{X})/\tau)$$

Here dividing by τ means every element in the vector is divided by τ. Obviously, when $\tau = 1$, this is exactly the same as the regular softmax function. What happens when $\tau \to \infty$? What happens when $\tau \to 0$?
Hint: You don't need to prove these limits, just showing a trend and justifying is good enough.

(iii) **(4 points)** Assume $f(\mathbf{X}) = \mathbf{w}^\top \mathbf{X}$ where $\mathbf{w}$ is a weight vector. What is the derivative of $S_\tau(f(\mathbf{X}))_i$ with respect to $\mathbf{w}$ for a fixed τ? In other words, what is $\partial S_\tau(\mathbf{w}^\top \mathbf{X})_i/\partial \mathbf{w}$, the derivative of the i-th element of $S_\tau(\mathbf{w}^\top \mathbf{X})$ with respect to $\mathbf{w}$? You must write your answer in the most reduced form to receive full credit.

(iv) **(2 points)** How can we use this modified softmax function S to get discrete vectors in our neural networks? Perhaps we cannot get perfect one-hot vectors but can we get close?

(v) **(2 points)** What problems could arise by setting τ to very low values?

Question (Debugging Code, 18 points)

Consider the pseudocode below for an MLP model to perform regression. The model takes an input of dim 10, hidden layer of size 20 with ReLU activations and outputs a real number. There are biases in both layers.

Weights are initialized from the random normal distribution and biases to 0.

Point out the errors in the code with line numbers and suggest fixes to them.

Your fixes should suggest code changes and not just English descriptions.

Functions/classes that are not implemented completely can be assumed to be correctly written and have no errors in them.

```
 1  import numpy as np
 2
 3  def mse_loss(predictions, targets):
 4      """
 5      Returns the Mean Squared Error Loss given the
 6      predictions and targets
 7
 8      Args:
 9          predictions (np.ndarray): Model predictions
10          targets (np.ndarray): True outputs
11
12      Returns:
13          Mean squared error loss between predictions and targets
14      """
15      return 0.5 * \
16              (predictions.reshape(-1) - targets.reshape(-1))**2
17
18
19  def dropout(x, p=0.1):
20      """
21      Applies dropout on the input x with a drop
22      probability of p
23
24      Args:
25          x (np.ndarray): 2D array input
26          p (float): dropout probability)
27
28      Returns:
29          Array with values dropped out
30      """
31      ind = np.random.choice(x.shape[1]*x.shape[0], replace=False,
32                             size=int(x.shape[1]*x.shape[0]*p))
33      x[np.unravel_index(indices, x.shape)] = 0
34      return x / p
```

```
35
36
37  def get_grads(loss, w1, b1, w2, b2):
38      """
39      This function takes the loss and returns the gradients
40      for the weights and biases
41      YOU MAY ASSUME THIS FUNCTION HAS NO ERRORS
42      """
43      ...
44      return dw1, db1, dw2, db2
45
46  def sample_batches(data, batchsize):
47      """
48      This function samples of batches of size `batchsize`
49      from the training data.
50      YOU MAY ASSUME THIS FUNCTION HAS NO ERRORS
51      """
52      ...
53      return x, y
54
55  class Adam:
56      """
57      The class for the Adam optimizer that
58      accepts the parameters and updates them.
59      YOU MAY ASSUME THIS CLASS AND ITS METHODS HAVE
60      NO ERRORS
61      """
62      def __init__(self, w1, b1, w2, b2):
63          ...
64
65      def update(self):
66          """
67          Updates the params according to the
68          Adam update rule
69          """
70          ...
71
72  class MLP:
73      """
74      MLP Model to perform regression
75      """
76      def __init__(self):
77          super().__init__()
78          self.w1 = np.random.randn(10, 20)
79          self.b1 = np.zeros(10)
```

```
80          self.w2 = np.random.randn(20, 1)
81          self.b2 = np.zeros(20)
82          self.optimizer = Adam(w1, b1, w2, b2)
83
84      def forward(self, x):
85          """
86          Forward pass for the model
87
88          Args:
89          x (np.ndarray): Input of shape batchsize x 10
90
91          Returns:
92          out (np.ndarray): Output of shape batchsize x 1
93          """
94          x = self.w1 * x + b1
95          x = dropout(x)
96          x = self.w2 * x + b2
97          return x
98
99
100     def train(self, training_data, test_data):
101          """
102          This method trains the neural network and outputs
103          predictions for the test_data
104
105          Args:
106          training_data (np.ndarray):
107                  Training data containing (x, y) pairs
108                  x is 10-dimensional and y is 1-dimensional
109          test_data (np.ndarray): 100 test points of shape
110                                          (100, 10)
111
112          Returns:
113          predictions (np.ndarray): The predictions for
114                              the 100 test points.
115                              Final shape is (100,1)
116          """
117          batchsize = 32
118          for _ in range(num_epochs):
119              for x, y in sample_batches(training_data, batchsize):
120                  # Shape of x is (32, 10) and y is (32, 1)
121                  out = self.forward(x)
122                  loss = mse_loss(x, y)
123                  dw1, db1, dw2, db2 = get_grads(loss, self.w1,
124                                                  self.b1, self.w2,
```

```
125                                                      self.b2)
126              self.optimizer.update()
127
128        # Assume test_data is of shape (100, 10)
129        predictions = self.forward(test_data)
130
131        return predictions
132
```

END OF PAPER