STAT 453 **Name:** _____
SS 2023
Midterm Exam
03/09/2023
Time: 4:00 – 5:15 pm am (75 minutes) **Email:** _____ @wisc.edu
Instructor: Yiqiao Zhong
Teaching Assistant: Hongzhi Liu

This exam contains 7 pages (including this cover page) and 9 questions.
Total of points is 100.

By submitting this exam, I (the student)

- acknowledge that I am required to follow the academic integrity and conduct policies of UW-Madison.

Grade Table (for teacher use only)

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	20	
Total:	100	

1. (10 points) In McCulloch & Pitt's neuron model (from lecture 2), we introduced neurons as a mathematical abstraction of biological neurons. What logical operations can a single neuron implement? Choose all possible answers that apply.

   A. AND
   B. OR
   C. NOT
   D. XOR

   Solution:

2. (10 points) What can be said about the perceptron algorithm that we learned in class? Choose all possible answers that apply.

   A. The perceptron algorithm is exactly stochastic gradient descent algorithm.
   B. If the input data are not linearly separable, then we can try applying basis transformation before running the perceptron algorithm.
   C. If we use the perceptron algorithm on the training dataset and obtain 98% accuracy, then we will achieve 98% on the test dataset as well.
   D. The threshold function used in the perceptron algorithm is a discontinuous function.

   Solution:

3. (10 points) Suppose that you have a minibatch of colored image data. The batch size is 128, and each image is represented by 224×224 pixels. These image data are represented by a 4-th order tensor X using PyTorch. You want to check the dimensions of the tensor X.
   (1) What is the output of `X.ndim` in PyTorch?
   (2) If you type `X.shape`, you get an output

   ```
   torch.Size([a,b,c,d])
   ```

   where a, b, c, d are four integers. What are a, b, c, d?

   Solution:

4. (10 points) What can be said about the (minibatch) stochastic gradient descent and back propagation? Choose all possible answers that apply.

   A. Stochastic gradient descent is an improvement of the full-batch gradient descent and will converge regardless of the learning rate.

   B. If we increase the batch size, then we have less noisy gradients despite bigger computational costs.

   C. In general, the exploding/vanishing gradient issue becomes more severe if add more layers in a deep neural network.

   D. Since the forward pass and the backward pass are independent, we don't need intermediate values from the forward pass when calculating back propagation.

   Solution:

5. (10 points) Suppose that $\sigma_1, \sigma_2, \sigma_3$ are three real-valued smooth activation functions (recall that a smooth function is must differentiable). Suppose

   - x is a d-dimensional input vector,
   - W_3 is a matrix of size $p \times d$,
   - W_1, W_2 are both matrices of size $m \times p$,
   - a is a m-dimensional vector.

   Consider a neural network

   $$f(x) = a^\top \sigma_3(W_1 \sigma_1(W_3 x) + W_2 \sigma_2(W_3 x)).$$

   (Note: as usual, the activation functions are computed in a coordinate-wise way.) For simplicity, we also denote

   $$z_1 = \sigma_1(W_3 x), \quad z_2 = \sigma_2(W_3 x), \quad z_3 = \sigma_3(W_1 \sigma_1(W_3 x) + W_2 \sigma_2(W_3 x)).$$

   What is the gradient of f with respect to the weight matrix W_3? Please use the chain rule to write down the gradient calculation.

   Solution:

6. (10 points) It is know that the leaky ReLU activation function is

$$\sigma(u) = \begin{cases} u & \text{if } u \geq 0 \\ -au & \text{if } u < 0 \end{cases}$$

where $a > 0$ is a pre-specified hyperparameter. An activation through leaky ReLU has the form $z = \sigma(w^\top x + b)$ where x is a d-dimensional input vector and w is a d-dimensional weight vector. What is the range of z if we change the weight w?

Solution:

7. (10 points) Suppose that you implement a multilayer perceptron model and observe severe overfitting phenomenon: the training error drops to almost zero but the validation error is still quite large. What can you do to reduce overfitting? Choose all possible answers that apply.

   A. Change the width (number of neurons in each layer) and depth (number of layers) appropriately.

   B. Increase the weight decay hyperparameter in the optimizer.

   C. Add a dropout layer between every two hidden layers.

   D. Add new training examples using data augmentation.

Solution:

8. (10 points) Suppose we have a two-class classification problem: the input is a vector x of dimension d, and the label has the one-hot encoding:

$$y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Consider the simple linear classifier with the softmax function

$$\hat{y} = \Big(\frac{\exp(w_1^\top x + b_1)}{\exp(w_1^\top x + b_1) + \exp(w_2^\top x + b_2)}, \quad \frac{\exp(w_2^\top x + b_2)}{\exp(w_1^\top x + b_1) + \exp(w_2^\top x + b_2)} \Big)^\top$$

We know that the cross-entropy loss (for a single training example) is given by

$$\ell(y, \hat{y}) = \sum_{j=1}^{2} -y_j \log(\hat{y}_j).$$

Is the cross-entropy loss equivalent to the logistic loss? Please briefly explain why.

Solution:

9. (20 points) One of your friends wants to use dropout in a simple neural network. She wrote her code but could not make it work.

```
import torch
import numpy as np

class MultilayerPerceptron():

    def __init__(num_features, num_classes=10, drop_proba=0.5,
                 num_hidden_1, num_hidden_2):
        super().__init__()

        self.my_network = torch.nn.Sequential(
            # hidden layer
            torch.nn.Flatten(),
            torch.nn.Linear(num_hidden_1, num_features),
            torch.nn.ReLU(),
            torch.nn.Dropout(drop_proba),
            # output layer
            torch.nn.Linear(num_classes, num_hidden_2)
        )

    def forward(x):
        logits = my_network(x)
        return logits

model = MultilayerPerceptron(num_features=28*28,
                             num_hidden_1=50,
                             num_hidden_2=50)

optimizer = torch.optim.SGD(model, lr=0.1)
```

Could you help her identify the bugs in her code? Please correct as many bugs as you can find (correcting 4 individual bugs for the full points).

Solution: