STAT 453 **Name:** ____________________

SS 2024

Midterm Exam

03/19/2024

Time: 4:00 – 5:15 pm am (75 mins) **Email:** ____________________ @wisc.edu

Instructor: Yiqiao Zhong

Teaching Assistant: Zhexuan Liu

Access Code: 819771-ZHONG

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) Among the four logical gates, which one cannot achieve 100% using a linear decision boundary?

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

   Solution:

2. (10 points) Which statement is wrong about stochastic gradient descent (SGD) for training neural networks?

   A. Compared with GD, SGD is much faster due to fewer examples a network computes at each iteration.
   B. Due to noise, SGD shows less oscillatory behavior compared with GD.
   C. SGD and perceptron algorithm share many similarities.
   D. When training neural networks, the standard practice is to only use first-order derivatives for SGD.

   Solution:

3. (10 points) CNNs are more effective than MLPs in processing images. Which one is not a reason for the effectiveness of CNNs?

   A. CNNs benefit from hand-crafted features to extract features such as edges corners from images.
   B. The kernels of CNNs use patches to compute local information, which reduces the number of parameters.
   C. CNNs have kernels that are shared across all patches, which benefit from our prior knowledge that images are invariant to translations.
   D. Many network-in-network modules such as those in Inception further reduce the number of parameters and improve computational efficiency.

4. (10 points) Which one of the statement is false about dropout?

   A. Dropout is an important regularization technique for training neural networks.

   B. For neural networks that use dropout, the forward pass is slightly different between the training phase and inference phase.

   C. During the inference phase, a neural network with dropout is equivalent to a smaller neural network with some neurons removed from computation.

   D. Dropout layers in PyTorch contain a probability (hyper-)parameter as an input argument.

   Solution:

5. (10 points) Which statement is false about normalization and initialization.

   A. Appropriate initialization and normalization help alleviate the exploding/vanishing gradient issue.

   B. If we don't use batch normalization, the training dynamics may suffer from slow convergence due to oscillation.

   C. Xavier initialization and He initialization ensures that gradients have similar magnitude during the backward pass.

   D. Batch normalization contains trainable parameters, which are updated using the usual backpropagation.

   Solution:

6. (10 points) Suppose that you use a CNN to process a minibatch of images. The batch size is 32. One convolutional layer is created as follows.

```
conv_layer = torch.nn.Conv2d(in_channels=4, out_channels=2,
                             kernel_size = (5, 5), stride = (2, 2))
```

   What is the **total** number of parameters in this layer?

   Solution:

7. (10 points) Suppose that σ is a generic differentiable activation function. Suppose

   - x is a d-dimensional input vector;
   - W is a matrix of size $d \times d$.

   Consider a nonstandard neural network

   $$f(x) = x + Wx + W\sigma(Wx).$$

   The activation functions are computed in a coordinate-wise way. We compute the loss based on MSE:

   $$\ell(W, w) = (y - f(x)^\top w)^2.$$

   We are interested in the gradient of $\ell(W, w)$ with respect to W. Can you derive the gradient? Please do one of the two following tasks: (1) use the chain rule to write down the gradient calculation, (2) or draw a computational graph to represent the forward pass from x to the loss value.

   Solution:

8. (10 points) Suppose that we have a neural network model `model` in PyTorch with three layers named "conv_1", "conv_2", "fc". We know that we can access the layers by typing `model.conv_1`, `model.conv_2`, and `model.fc` respectively.

   Suppose that the model is already trained, and we want to finetune the "fc" layer for a downstream application. To accomplish this, we need to freeze the parameters of the other layers before finetuning. How do you achieve this in PyTorch? Please write code below. *Hint: use* `requires_grad`.

   Solution:

9. (20 points) Pleaes explain in words what the following PyTorch code means.
(i) `torchvision.transforms.Resize(size=(32,32))`
(ii) `class MLP(torch.nn.Module):`
(iii) `loss.backward()`
(iv) `optimizer = torch.optim.AdamW(model.parameters(), weight_decay=0.01)`
(v) `torch.manual_seed(20)`

Solution: