ECE 47300: Final Exam (Spring 2023), Version: D

Instructions

Please fill out this cover page but do NOT start until instructed to do so.

What is your full name and Purdue ID (10-digit number)?

Name (please PRINT, no cursive): Paloma Arellano

Purdue ID (10 digits): 0031926985

Honor Pledge Signature: PA

- **Total number of points**: 115
- **Question format notes**
  - Circular checkboxes are for multiple choice (i.e., a single correct answer)
  - Square checkboxes are for select all questions (i.e., possibly multiple correct answers).
  - Only put your answer in the boxes like [ ] .
- **No partial credit**: Every (sub-)question is all or nothing credit. Thus, you must get the answer exactly right to get credit for the question (including SELECT ALL questions). No partial credit will be given.
- **Number Format**: When giving numbers as short answers, please give in standard decimal notation with preceding "0." for decimals if needed but no trailing 0s (e.g., "0.15", "2.9", "0.001", "100" but NOT "0.15000" NOR ".15" NOR ".001" NOR "6.0").
- **Honor Pledge**: I assert that I have not received any information about this exam and will not share any exam content with anyone else, even after the exam. I understand that any violation of this will result in a failing grade for the whole class (not just this exam).

1. (3 points) Which of the following tasks can be accomplished using self-supervised learning? Select all the correct answers.
   - ☐ Predicting the sentiment of a sentence
   - ■ Image colorization (from grayscale to color image)
   - ■ Predicting missing words in a sentence
   - ☐ Image classification

2. (3 points) What are some limitations of the windowing approach to handling sequences?
   - ☐ The window size is variable.
   - ■ Lacks long-range dependencies.
   - ☐ The model's predictions of different windows are very dependent.

3. (2 points) RNNs can only be used if both the input and output are sequences that have more than one item.
   - ○ True
   - ● False

4. (2 points) Which of the following distributions are valid generative models for count-valued data? (Select all)
   - ☐ Gaussian
   - ☐ Uniform
   - ■ Multinomial
   - ☐ Mixture of Gaussians
   - ■ Mixture of Multinomials

5. (2 points) In the word2vec task for predicting the target word given context words using the CBOW architecture, the order of the words in the context is important.
   - ○ True
   - ● False

6. (3 points) Which of the following statements about LSTM (Long Short-Term Memory) networks are true? (Select all that apply)
   - ■ LSTM networks are a type of recurrent neural network (RNN) module architecture used for sequence modeling.
   - ☐ LSTMs can be used for natural language processing (NLP) tasks.
   - ☐ LSTMs have a mechanism that allows them to update and forget information that is passed to the next cell. in hidden state selectively
   - ■ LSTMs are capable of capturing long-term dependencies in sequential data.

7. (2 points) Assuming the notation from the slides, what is the correct equation for an RNN module?
   - ○ $x_\ell, y_\ell = f_\theta(h_\ell, h_{\ell-1})$
   - ● $y_\ell, h_\ell = f_\theta(x_\ell, h_{\ell-1})$
   - ○ $x_\ell, h_\ell = f_\theta(y_\ell, h_{\ell-1})$
   - ○ $x_\ell, h_{\ell-1} = f_\theta(y_\ell, h_\ell)$
   - ○ $y_\ell, h_{\ell-1} = f_\theta(x_\ell, h_\ell)$

8. (3 points) Which of the following statements about GRU (Gated Recurrent Unit) are true?
   - ☐ GRU is a type of unsupervised learning algorithm used for clustering.
   - ■ GRU has a gating mechanism that allows it to selectively update its hidden state, which helps to address the problem of vanishing gradients.
   - ■ GRU is a type of recurrent neural network (RNN) module used for sequence modeling.

9. (2 points) RNNs can be used for which of the following tasks (if any):
   - ■ Text generation
   - ■ Text classification

10. (2 points) Suppose we have an attention weights (after softmax) $M = \sigma(QK^T) \in \mathbb{R}^{15\times 20}$. What is the sum over all the elements in the matrix, i.e., $\sum_{i,j} M_{i,j}$?

    Ans: 15

11. (3 points) Which of the following describe the role of attention in Transformers?
    - ☐ A technique used to increase the computational efficiency of Transformers by reducing the number of computations required.
    - ■ A mechanism that allows Transformers to weigh the importance of different tokens in a sequence for processing.
    - ■ A model architecture that enables the decoder to use all encoder outputs efficiently.

12. (2 points) One nice property of attention is visualizing the attention map. Assuming that we have an attention model trained for English-French translation, and the input is "He has a dog" and the output is "Il a un chien" what is the size of the cross-attention map? Note that an $< EOS >$ token is used.
    - ○ 6 by 6
    - ● 5 by 5
    - ○ 7 by 7

13. (2 points) Assuming the input of self attention is $[1, 3, 2, 7]^T$ and the output is $[6, 8, 0, 4]^T$. If we permute the input to be $[1, 2, 3, 7]^T$, what is the output?
    - ○ $[0, 4, 6, 8]^T$
    - ○ $[4, 0, 6, 8]^T$
    - ○ $[8, 0, 6, 4]^T$
    - ● $[6, 0, 8, 4]^T$
    - ○ Unable to be determined by information given.

    [6,0,8,4]T

    [6,0,8,4]T

14. (2 points) For multi-headed attention, suppose the output dimension of every head is $D_{head} = 4$ and there are 8 heads and the final linear layer matrix of multiheaded attention is $W_H \in \mathbb{R}^{C\times D_{out}}$, what is C?

    Ans: 32

15. (2 points) Which of the following statement accurately describes diffusion models?

- ○ Diffusion models are a type of supervised learning model that use diffusion processes to propagate information through a graph.
- ● Diffusion models are a type of deep generative model that use probabilistic diffusion processes to model the dynamics of data creation.
- ○ Diffusion models are a type of reinforcement learning model that use diffusion processes to simulate the spread of rewards through a network.
- ○ Diffusion models are a type of deep learning model that use diffusion equations to model the flow of information through a neural network.

16. (2 points) For each diffusion model training method, there is only one method to sample from it given the constraints of the model.

- ● True
- ○ False

17. (2 points) For the diffusion models discussed in class, which of the following are known in closed-form?

- □ $q(x_t|x_{t-1})$
- ■ $q(x_{t-1}|x_t)$

18. (2 points) For the diffusion models discussed in class, which of the following are known in closed-form?

- ■ $q(x_{t-1}|x_t, x_0)$
- □ $q(x_t|x_0)$

19. (2 points) Which of the following statements accurately describes reinforcement learning?

- ○ Reinforcement learning is a type of unsupervised learning that involves finding patterns and structure in data. ✗
- ○ Reinforcement learning is a type of supervised learning that involves predicting outputs from inputs using labeled data. ✗
- ● Reinforcement learning is a type of learning in which an agent learns to make decisions by interacting with an environment and receiving rewards or punishments.
- ○ Reinforcement learning is a type of transfer learning that involves transferring knowledge from one domain to another. ✗

20. (2 points) In reinforcement learning, training can happen simultaneously with testing/deployment.

- ● True
- ○ False

21. (2 points) Rewards encode _____ feedback from the environment.

- ● Short-term
- ○ Average
- ○ Long-term

22. (2 points) In vanilla multi-armed bandits, the explore-exploit tradeoff does not exist because the environment does not change.

- ● True
- ○ False

23. (2 points) In multi-armed bandits, the greedy algorithm is the optimal algorithm because it always selects the action with the highest estimated reward.

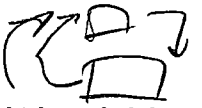
- [x] True
- [ ] False

24. (2 points) For the learning-to-walk task of a robot, the reinforcement learning agent is the whole robot including the position of its arms and legs.

- [x] True
- [ ] False

25. (2 points) In Markov decision processes, the agent provides the action and reward to the environment.

- [x] True
- [ ] False



26. (2 points) In Markov decision processes, which probability distribution defines the dynamics of the environment?

- [ ] $p(A_t|S_t)$
- [ ] $p(S_t|A_t)$
- [x] $p(S_t, R_t|S_{t-1}, A_{t-1})$
- [ ] $p(S_t, A_t|S_{t-1}, R_{t-1})$

27. (2 points) The state-value and action-value functions are independent of the agent policy.

- [ ] True
- [x] False

28. (2 points) An optimal policy can be constructed from optimal value functions.

- [ ] True
- [x] False

29. (2 points) Suppose we have near ideal word embeddings, where w_i denotes the embedding of the i-th word. Select all boxes where the equations should approximately hold.

- [ ] $w_{father} - w_{son} \approx w_{mother} - w_{daughter}$
- [ ] $w_{king} - w_{queen} \approx w_{woman} - w_{man}$
- [x] $w_{fast} - w_{faster} \approx w_{long} - w_{longer}$

30. The following questions are related to the Convolutional Neural Network, where you need to calculate either the input or output shape in the format "CxHxW" or the filter size as "f_Hxf_W".

(a) (2 points) If the stride of a convolutional layer is 3 and the filter size is 3x3, what would be the corresponding stride and filter size in the transpose convolutional layer that aims to undo the effect of the original convolutional layer?
Ans: Filter size (f_Hxf_W) = [3x3], Stride = [3]

(b) (2 points) Given an input image with dimensions 3x224x224, what would be the size of the output feature map after applying a convolutional layer with 7 filters, a filter size of 7x7, a stride of 1, and a padding of 3? (Format: CxHxW)
Ans: [7x224x224]

$\frac{224-7+2(3)}{1}+1 \rightarrow 224-7+6+1$

(c) (2 points) Given an input image with dimensions 3x224x224, what would be the size of the output feature map after applying a convolutional layer with 5 filters, a filter size of 4x4, a stride of 2, and a padding of 1? (Format: CxHxW)
Ans: [5x112x112]

$\frac{224-4+2(1)}{2}+1 = \frac{222}{2}+1 = 111+1 = 112$

31. Assume that the goal of the code below is to compute the covariance matrix correctly. However, the current code below contains a bug.

```
# Assume X is a 2D numpy array
n, d = X.shape
x_center = X.mean(axis=1)
zero_centered_data = X - x_center
cov = (1/n) * np.dot(zero_centered_data.T, zero_centered_data)
```

(a) (2 points) Which line will raise a Numpy error?
- ○ (a)
- ● (b)
- ○ (c)

(b) (2 points) Which line should be changed to correctly compute the covariance matrix?
- ○ (a)
- ● (b)
- ○ (c)

(c) (3 points) Corrected line. [zero_centered_data = X[:,1] - np.ones(X) * X_center]

32. (3 points) Fit a Kernel density model on X_train where the kernel is a *Uniform* distribution centered around the training point, i.e., Uniform($[x_i - 0.5, x_i + 0.5]$), where x_i is a training point. X_train is (already sorted):

```
X_train = [0.9, 1.3, 1.7, 2.1, 2.9, 3.0]
```

What are the kernel density PDF values for the following test points?

- x = 3.1: [1/6]
- x = 1.3: [1/6]
- x = 2.7: [1/6]

33. The code below shows the training process of the discriminator in GANs. `B` and `D` denote the batch dimension and latent dimension of the noise respectively. The criterion is the binary cross entropy loss as used in the homework and defined as:

$$\ell(x, y) \triangleq y \log x + (1 - y) \log(1 - x). \tag{1}$$

```
# Input: z, x, device

##### Update netD #####
for x in dataloader:
    netD.zero_grad()
    label = torch.zeros((B,), device=device)
    output = netD(x).view(-1)
    errD_real = criterion(output, label) # criterion = nn.BCELoss()
    errD_real.backward()
    noise = torch.randn(B, D, 1, 1, device=device)
    fake = netG(noise)
    label = torch.zeros((B,), device=device)
    output = netD(fake.detach()).view(-1)
    errD_fake = criterion(output, label)
    errD_fake.backward()
    optimizerD.step()
```

(a) (4 points) While the above code should run, what is the line number having a logical error (choose out of line 2-8)?
Ans: 5

(b) (2 points) (line 5, 11) Calling `backward()` twice before calling `step()` is logically wrong because the first gradient is overridden by the second gradient.

- ○ True
- ● False

(c) (2 points) Suppose we want to use the original GAN loss for updating a generator, which can be formulated as $\log(1 - D(G(z)))$. Assuming that `criterion(x,y)` is the binary cross entropy loss (see equation above code), choose the answer for the correct form of `errG`.

- ● `errG = criterion(output, fake_label)`
- ○ `errG = -criterion(output, fake_label)`
- ○ `errG = criterion(output, real_label)`
- ○ `errG = -criterion(output, real_label)`

(d) (2 points) To make the GAN training stable, WGAN proposed to restricts the discriminator weights to have the bounded Lipschitz constant. What is the correct answer to achieve this given `c=0.01`?

- ○
```
for p in netD.parameters():
    p.data.clamp_(-c, c)
```

- ○
```
for p in netD.parameters():
    p.data.clamp_(0, c)
```

- ●
```
for p in netD.parameters():
    p.data.clamp_(-c, 0)
```

34. Suppose the task is next character prediction. The code below chooses a random chunk of characters and returns a tuple containing two items: (1) all but the last character and (2) the last character.

```
def get_random_instance(file, chunk_len):
    start_index = np.random.randint(0, len(file) - chunk_len)
    end_index = start_index + chunk_len + 1
    chunk = file[start_index:end_index]
    return (__A__, __B__)
```

What should be the code in the two blanks?

(a) (2 points) A: chunk[:-1]

(b) (2 points) B: chunk[-1]

35. The LSTM equations are:

$$h'_{t-1} = [h_{t-1}, x_t]$$
$$\tilde{C}_t = \tanh(W_C h'_{t-1} + b_C)$$
$$f_t = \sigma(W_f h'_{t-1} + b_f)$$
$$i_t = \sigma(W_i h'_{t-1} + b_i)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma(W_o h'_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

and the following code is from the forward function of LSTMCell.

```
def forward(self, input, hidden=None):
    # Unpack hidden state and cell state
    hx, cx = hidden
    # Apply linear layers to input and hidden state
    linear = self.xh(input) + self.hh(hx)
    # Get outputs of applying a linear transform for each part of the LSTM
    input_linear, forget_linear, cell_linear, output_linear = linear.reshape(-1).chunk(4)

    i_t = torch.sigmoid(input_linear)
    f_t = torch.sigmoid(forget_linear)
    c_t = torch.tanh(cell_linear)
    o_t = torch.sigmoid(output_linear)

    #### See next line for question ####
    c_new = __A__ * f_t + i_t * __B__ → c_t
    h_new = o_t * torch.tanh(c_new)

    # Pack cell state $C_t$ and hidden state $h_t$ into a single hidden state tuple
    output = h_new # For LSTM the output is just the hidden state
    hidden = (h_new, c_new) # Packed h and C
    return output, hidden
```

What variables should be placed in the A and B blanks in the code?

(a) (2 points) A: cx

(b) (2 points) B: c_t

36. The code below calculates attention.

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where $Q \in \mathbb{R}^{L \times d_k}$, $K \in \mathbb{R}^{L \times d_k}$, and $V \in \mathbb{R}^{L \times d_v}$, where d_v is the dimension of the values. The softmax is across the column dimension. The output of attention should be a matrix $A \in \mathbb{R}^{L \times d_v}$.

```
def attention(q, k, v):
    '''
    Inputs:
    q: query vector of shape (batch_size, seq_len, d_k)
    k: key vector of shape (batch_size, seq_len, d_k)
    v: value vector of shape (batch_size, seq_len, d_v)

    Returns:
    output: attention weighted sum of the value vectors
        of shape (batch_size, seq_len, d_k)
    scores: attention weights of shape (batch_size, seq_len, seq_len)
    '''
    d_k = k.size(-1)
    assert d_k == q.size(-1), 'q and k should have the same dimensionality'
    d_v = v.size(-1)
    print(f"d_k => {d_k}")


    att_scores = torch.bmm(q, k.transpose(-2, -1)) /  math.sqrt(d_k)
    att_values = F.softmax(att_scores, dim=-1)
    output = torch.bmm(att_values, v)

    return att_scores, att_values, output
```

Here, all the output tensors are 2x2 matrices.

(a) (4 points) Suppose you are given the following inputs to self-attention:

```
q = torch.tensor([[[1., 1., 1., 1.],
                   [0., 1., 0., 1.]]])
k = torch.tensor([[[0., 1., 0., 1.],
                   [0., 1., 0., 1.]]])
v = torch.tensor([[[0., 0.],
                   [2., 1.]]])
```

What will be att_score?

2	2
2	2

(b) (4 points) Suppose that att_score is equal to:

```
att_score = torch.tensor([[[torch.log(1.0), torch.log(3.0)],
                           [torch.log(3.0), torch.log(9.0)]]])
```

What will be att_values?

3	3
9	9

37. The following is a module that computes attention.

```
class Attention(nn.Module):
    def __init__(self, input_dim, key_dim, output_dim):
        super().__init__()
        self.q_linear = nn.Linear(input_dim, key_dim)
        self.k_linear = nn.Linear(input_dim, key_dim)
        self.v_linear = nn.Linear(input_dim, output_dim)

    def forward(self, x1, x2, x3):
        q = self.q_linear(x1)
        k = self.k_linear(x2)
        v = self.v_linear(x3)
        output, att_values = attention(q, k, v)
        return output
```

(a) (2 points) Suppose we wanted to use this module to compute self-attention on variable `x` of the correct shape, what should be placed in the blank below:

```
attn = Attention(input_dim, key_dim, output_dim)
self_attn_output = ____________________
```

attn

(b) (2 points) Suppose we wanted to use this module to compute cross-attention between variables `x` and `y` of the correct shape, what should be placed in the blank below:

```
attn = Attention(input_dim, key_dim, output_dim)
cross_attn_output = ____________________
```

attn

38. This is a code snippet for the Greedy agent. It contains a minor mistake. The code runs without any errors. Find the line number and write the corrected line.

```
class GreedyAgent:
    def __init__(self, n_actions, init_value=0):
        self.init_value = init_value
        self.action_counts = torch.zeros(n_actions) #n_t
        self.action_value_func = init_value * torch.ones(n_actions)  #Q_t

    def select_action(self):
        return torch.argmax(self.action_value_func)

    def update(self, action_index, reward):
1       action_count = self.action_counts[action_index]
2       sum_rewards = action_count * self.action_value_func[action_index]
3       new_avg_reward = (sum_rewards + reward) / (action_count)

4       self.action_counts[action_index] += 1
5       self.action_value_func[action_index] = new_avg_reward
        return self

    def __str__(self):
        return f'Greedy(init={self.init_value})'
```

(a) (3 points) Line number (between 1-5 marked above): 2

(b) (3 points) Corrected line. sum_rewards = action_count * select_action()