Machine Learning - MPhil AI Fall 2024 Morning
Quiz – 04

Max Marks: 20 Roll Number: _____
Available Time: 40-minutes Name: _____

1). Consider a neural network with the sigmoid activation function defined as: $\sigma(x)=1/\ 1 + e^{-x}$

During backpropagation, which of the following is the correct expression for the derivative of the sigmoid activation function with respect to its input?

a) $\sigma'(x)=\sigma(x)(1-\sigma(x))$ b) $\sigma'(x)=1-\sigma(x)$
c) $\sigma'(x)=1/\sigma(x)(1-\sigma(x))$ d) $\sigma'(x)=e^{-\sigma(x)}$

2). You are given a dataset with points in a 2-dimensional space. You apply the K-means clustering algorithm with k=2. After one iteration, the following centroids are computed:

Centroid 1: (3,4) Centroid 2: (8,6)

Which of the following is the correct update rule for the next iteration, assuming Euclidean distance?

a) Reassign the points to the centroid with the smallest x-coordinate
b) Reassign the points to the centroids that minimize the squared Euclidean distance
c) Reassign the points to the centroid with the largest y-coordinate
d) No reassignment is necessary; centroids remain the same

3). Which of the following properties is true for the hyperbolic tangent (tanh) activation function tanh(x)?

a) It is symmetric around the origin, mapping negative inputs to positive outputs
b) Its derivative is always greater than 1, leading to faster convergence in gradient descent
c) Its output is always bounded between 0 and 1
d) It suffers from the vanishing gradient problem for very small or very large inputs

4). You are optimizing a neural network with two layers: one with 128 neurons and another with 64 neurons. Which of the following is most likely to result in an unstable training process?

a) Using a large learning rate and no regularization
b) Using a small batch size and a large dropout rate
c) Using a learning rate scheduler with a linear decay
d) Using an optimizer like Adam with default parameters

5). You are working on a neural network for image classification. The model has been defined with one hidden layer, and you have already prepared the dataset and initialized the optimizer and loss function.

The model takes an input of shape (batch_size, 784) (flattened 28x28 images) and has 10 output classes. You need to train the model for 5 epochs using the Adam optimizer and cross-entropy loss. The following code is provided:

```
import torch
import torch.nn as nn
import torch.optim as optim

# Define the simple neural network
model = nn.Sequential(
    nn.Linear(784, 128),  # Input to hidden layer
    nn.ReLU(),            # ReLU activation
    nn.Linear(128, 10)    # Hidden to output layer
)

# Set up the optimizer and loss function
optimizer = optim.Adam(model.parameters(), lr=0.001)
loss_fn = nn.CrossEntropyLoss()

# Assume inputs and targets are given as random tensors for this example
inputs = torch.randn(32, 784)  # Example input batch (batch_size=32)
targets = torch.randint(0, 10, (32,))  # Example targets (batch_size=32, 10 classes)
```

You are required to complete the following code to:
1. Perform the forward pass
2. Calculate the loss
3. Perform the backward pass and optimizer step

6). In the following training loop for a neural network, which part of the code needs to be modified to properly compute the gradients and update the weights?

```
for epoch in range(num_epochs):
    for data in train_loader:
        inputs, targets = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = loss_fn(outputs, targets)
        # missing part for gradient update
        optimizer.step()
```

Which operation is missing in the training loop to ensure the model learns?

a) optimizer.zero_grad() b) inputs.backward()
c) loss.backward() d) outputs.backward()

7). The Leaky ReLU activation function is defined as: $\text{LeakyReLU}(x)=\{x \text{ if } x>0, \quad \alpha x \text{ if } x\leq 0 \}$

Where α is a small constant. Which of the following is the key advantage of Leaky ReLU over ReLU?

a) It introduces a non-zero gradient for negative inputs, preventing dead neurons during training
b) It guarantees that the output of the function is always positive
c) It accelerates the training process by providing higher gradients for negative inputs
d) It is computationally more expensive than ReLU but results in better accuracy

8). Write the code to initialize a simple neural network in PyTorch with:

- 1 hidden layer with 64 neurons
- Uses ReLU activation
- Output layer with 10 neurons for classification (10 classes)
- Input size is 784 (e.g., flattened 28x28 images)

9). Consider a neural network where the gradient with respect to weights w at layer l is computed as:

$\partial E / \partial w^{(l)} = (\partial E / \partial a^{(l)}) \cdot (\partial a^{(l)} / \partial w^{(l)})$.

Where $a^{(l)}$ is the activation at layer l. Which of the following is the best approach to verify the correctness of this computed gradient?

a) Perform gradient checking by numerically approximating the gradients and comparing them
b) Use the chain rule to compute gradients for each layer
c) Implement weight decay to prevent large gradients during training
d) Apply the ReLU activation function to each gradient step

10). In the following scenario, you are building a simple convolutional neural network. What is the correct code to implement the convolutional layer with 32 filters, kernel size 3, and stride 1?

```
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1)
        # missing part for activation function
        self.fc1 = nn.Linear(32 * 32 * 32, 10)

    def forward(self, x):
        x = self.conv1(x)
        # missing part for activation
        x = self.fc1(x)
        return x
```

What is the missing part of the code?

a) x = torch.sigmoid(x) b) x = torch.relu(x)
c) x = torch.tanh(x) d) No activation function is needed after the convolutional layer

11). Derive the first derivative of the sigmoid activation function $\sigma(x) = 1/ 1 + e^{-x}$ with respect to x. Show your work in 4-5 lines.

12). Consider you are working on a Naive Bayes classification task with a high-dimensional dataset. You are using the Naive Bayes classifier with the assumption that features are conditionally independent given the class. Which of the following methods will most likely lead to overfitting when applied directly to the training data?

a) Feature selection using mutual information to reduce dimensionality
b) Applying Naive Bayes with Laplace smoothing to handle zero frequencies
c) Removing highly correlated features before training the model
d) Using a kernel density estimate to estimate the conditional probabilities of features

Solution

1). a) $\sigma'(x)=\sigma(x)(1-\sigma(x))$
2). b) Reassign points to minimize squared Euclidean distance.
3). d) Suffers from vanishing gradient problem for very small/large inputs.
4). a) Large learning rate and no regularization.
5). Complete code:

```
outputs = model(inputs)
loss = loss_fn(outputs, targets)
loss.backward()
optimizer.step()
```

6). c) loss.backward()
7). a) Non-zero gradient for negative inputs prevents dead neurons.
8). Code for initializing NN:

```
model = nn.Sequential(
nn.Linear(784, 64),
nn.ReLU(),
nn.Linear(64, 10)
)
```

9). a) Perform gradient checking by numerical approximation.
10). b) x = torch.relu(x)
11). Derivative of $\sigma(x)$ is $\sigma'(x)=\sigma(x)(1-\sigma(x))$
12). d) Using a kernel density estimate can lead to overfitting.