ML and AI

- Which statement is true? (you may interpret “kind of” as “subset of”)

A. AI is a kind of Machine Learning
B. Machine Learning is a kind of AI
C. Machine Learning is a kind of Deep Learning
D. Deep Learning is a kind of Machine Learning

ML and AI

- Which statement is true? (you may interpret “kind of” as “subset of”) D
    A. AI is a kind of Machine Learning
    B. Machine Learning is a kind of AI
    C. Machine Learning is a kind of Deep Learning
    D. Deep Learning is a kind of Machine Learning

Simulated Annealing

- In simulated annealing, how can we increase “exploitation”?
    A. Increasing the step-size
    B. Decreasing the step-size
    C. Increasing the temperature
    D. Decreasing the temperature

Simulated Annealing

- In simulated annealing, how can we increase “exploitation”? D
    A. Increasing the step-size
    B. Decreasing the step-size
    C. Increasing the temperature
    D. Decreasing the temperature

Optimization

- In gradient descent and ascent, which of the following does tell us the *direction* of solution update?

A. Gradient
B. Step-size
C. Batch-size
D. Temperature

Optimization

- In gradient descent and ascent, which of the following does tell us the *direction* of solution update? A

A. Gradient
B. Step-size
C. Batch-size
D. Temperature

kNNs

12- Which statement is Wrong about about kNNs?

A. For a given *k*, it calculates the distance to all the training instances and picks the *k* nearest ones, followed by choosing the majority class.
B. Selecting a small *k* may lead to overfitting to training data.
C. Selecting a large *k* may lead to a very general but ineffective model.
D. kNN can be trained very efficiently and it is also easy, fast, and efficient to make predictions for new instances.

kNNs

- Which statement is Wrong about about kNNs? D
    A. For a given *k*, it calculates the distance to all the training instances and picks the *k* nearest ones, followed by choosing the majority class.
    B. Selecting a small *k* may lead to overfitting to training data.
    C. Selecting a large *k* may lead to a very general but ineffective model.
    D. kNN can be trained very efficiently and it is also easy, fast, and efficient to make predictions for new instances.

Gradient computation

16- Suppose you have a function $h(x,y) = (a + bx + cy)^2$ where x and y are input variables and a, b, and c are some constants. What is the gradient of *h* at the origin (x=0, y =0)

A. (0, 0)
B. (a, b, c)
C. (2ab, 2ac)
D. (2b, 2c)

Gradient computation

16- Suppose you have a function *h(x,y) = (a + bx + cy)^2* where x and y are input variables and a, b, and c are some constants. What is the gradient of *h* at the origin (x=0, y =0) C

A. (0, 0)
B. (a, b, c)
C. (2ab, 2ac)
D. (2b, 2c)

CNNs

24- Which of the following does NOT explain why convolutional neural networks work and become popular?

A. Having an overparameterized architecture with one fully-connected and very wide layer and a few other fully-connected layers
B. Parameter sharing
C. Localized pattern detection over all the input using inner product as a similarity measure
D. Hierarchical feature/pattern recognition across layers where complex features are extracted in deeper layers

CNNs

24- Which of the following does NOT explain why convolutional neural networks work and become popular? A

A. Having an overparameterized architecture with one fully-connected and very wide layer and a few other fully-connected layers
B. Parameter sharing
C. Localized pattern detection over all the input using inner product as a similarity measure
D. Hierarchical feature/pattern recognition across layers where complex features are extracted in deeper layers

Coding Question

What is the issue with the loss reporting in this training function?

```
def train(model, train_loader, criterion, optimizer, num_epochs=5):
    # Set model to training mode
    model.train()

    for epoch in range(num_epochs):
        running_loss = 0.0

        for i, data in enumerate(train_loader):
            # Get inputs and labels
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            # Zero parameter gradients
            optimizer.zero_grad()

            # Forward pass
            outputs = model(inputs)
            loss = criterion(outputs, labels)

            # Backward pass and optimize
            loss.backward()
            optimizer.step()

            # Update statistics
            running_loss += loss.item()

        # Print epoch statistics
        print(f'Epoch {epoch+1}/{num_epochs}, Loss: {running_loss}')
```

A. The model is never set to evaluation mode between epochs
B. The `optimizer.zero_grad()` is in the wrong place
C. The `running_loss` is not normalized by the number of batches in `train_loader`
D. The loss calculation is using the wrong criterion

Coding Question

What is the issue with the loss reporting in this training function?

```
def train(model, train_loader, criterion, optimizer, num_epochs=5):
    # Set model to training mode
    model.train()

    for epoch in range(num_epochs):
        running_loss = 0.0

        for i, data in enumerate(train_loader):
            # Get inputs and labels
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            # Zero parameter gradients
            optimizer.zero_grad()

            # Forward pass
            outputs = model(inputs)
            loss = criterion(outputs, labels)

            # Backward pass and optimize
            loss.backward()
            optimizer.step()

            # Update statistics
            running_loss += loss.item()

        # Print epoch statistics
        print(f'Epoch {epoch+1}/{num_epochs}, Loss: {running_loss}')
```

A. The model is never set to evaluation mode between epochs
B. The `optimizer.zero_grad()` is in the wrong place
C. The `running_loss` is not normalized by the number of batches in `train_loader`
D. The loss calculation is using the wrong criterion

Solution: C

The code accumulates loss values with `running_loss += loss.item()` but then prints the total accumulated loss rather than the average loss per batch. This would scale the loss with the number of batches and would be misleading for comparing across datasets of different sizes.