1. 다음 중 PyTorch로 딥러닝 모델을 학습할 때, 한 배치(batch)에 대한 학습 과정의 올바른 순서를 고른 것은?

A. `optimizer.zero_grad()`
B. `output = model(input)`
C. `loss = criterion(output, target)`
D. `loss.backward()`
E. `optimizer.step()`

---

1. B → C → D → E
2. A → B → C → D → E
3. A → C → B → D → E
4. B → C → D → E → A
5. A → B → D → C → E

2. 다음은 AI 개론 5차시 이차 손실 함수 L(u,v)L(u, v)L(u,v)에 대해 경사하강법을 수행하는 코드이다. 이 코드에 대한 설명으로 옳은 것을 모두 고르시오.

```
def L(u, v):
    return 3 * u**2 + 3 * v**2 - u*v + 7*u - 7*v + 10
def Lu(u, v):
    return 6* u - v + 7
def Lv(u, v):
    return 6* v - u - 7


W = np.array([4.0, 4.0])
alpha = 0.05
for i in range(N):
    W = W - alpha * np.array([Lu(W[0], W[1]), Lv(W[0], W[1])])
```

A. 함수 `L(u, v)`는 이차식이며, 볼록 함수(convex function)이다.

B. `W`의 초기값이 다르더라도, 학습률과 반복 횟수가 적절하면 최소값에 수렴할 수 있다.

C. 이 코드는 수학적으로 유도된 편미분식을 사용하므로 역전파나 자동미분을 사용하지 않는다.

D. `alpha` 값이 너무 크면 손실 함수의 최소점을 지나쳐서 발산할 수 있다.

E. `Lu`와 `Lv`를 대신하여 수치 미분 기법(예: 중심 차분법)을 사용하면 정확도는 오히려 더 높아진다.

3. 선형 회귀에서 기울기(θ_1)를 다음과 같이 계산하는 이유 및 관련된 설명으로 옳은 것을 모두 고르면?

$$\hat{\theta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

---

① 이 공식은 **평균 제곱 오차(MSE)**를 최소화하기 위해 유도된 결과이다.

② 이 공식의 분자는 입력값과 출력값의 공분산을 의미한다.

③ 이 공식은 분류 문제에서 결정 경계를 만들기 위해 사용된다.

④ 기울기를 구한 후, 절편(θ_0)은 $\bar{y} - \hat{\theta}_1 \bar{x}$ 로 계산된다.

⑤ 이 공식은 입력값의 표준편차를 기반으로 만들어졌다..

4. 다음 중 PyTorch의 텐서(Tensor)에 대한 설명으로 옳은 것을 모두 고르시오.

① PyTorch의 텐서는 GPU에서 연산이 가능하지만, NumPy 배열은 기본적으로 CPU에서만 연산된다.

② `squeeze()` 함수는 차원이 1인 축을 제거하고, `unsqueeze()`는 지정한 위치에 차원 1을 추가한다.

③ 3차원 텐서는 일반적으로 RGB 이미지 한 장을 표현할 때 사용된다.

④ PyTorch 텐서 연산은 자동 미분을 지원하며, `autograd`를 통해 기울기를 추적할 수 있다.

⑤ 텐서 연산은 항상 입력 텐서의 차원이 같아야만 작동하며, 차원이 다르면 오류가 발생한다.

5. 다음 중 딥러닝 학습 과정에서 실제로 수행되는 동작에 대해 올바르게 설명한 것을 모두 고르시오.

① 파라미터가 학습되기 전에, 이전 반복에서 계산된 변화율 정보를 초기화하지 않으면, 새로 계산되는 값에 영향을 줄 수 있다.

② 모델에 입력 데이터를 넣어 결과를 계산할 때, 손실 함수와는 독립적으로 연산 그래프가 구성되지 않는다.

③ 정답값과 모델 출력값 간의 차이를 정량적으로 측정하는 과정은, 이후 기울기 계산에서 시작점 역할을 하게 된다.

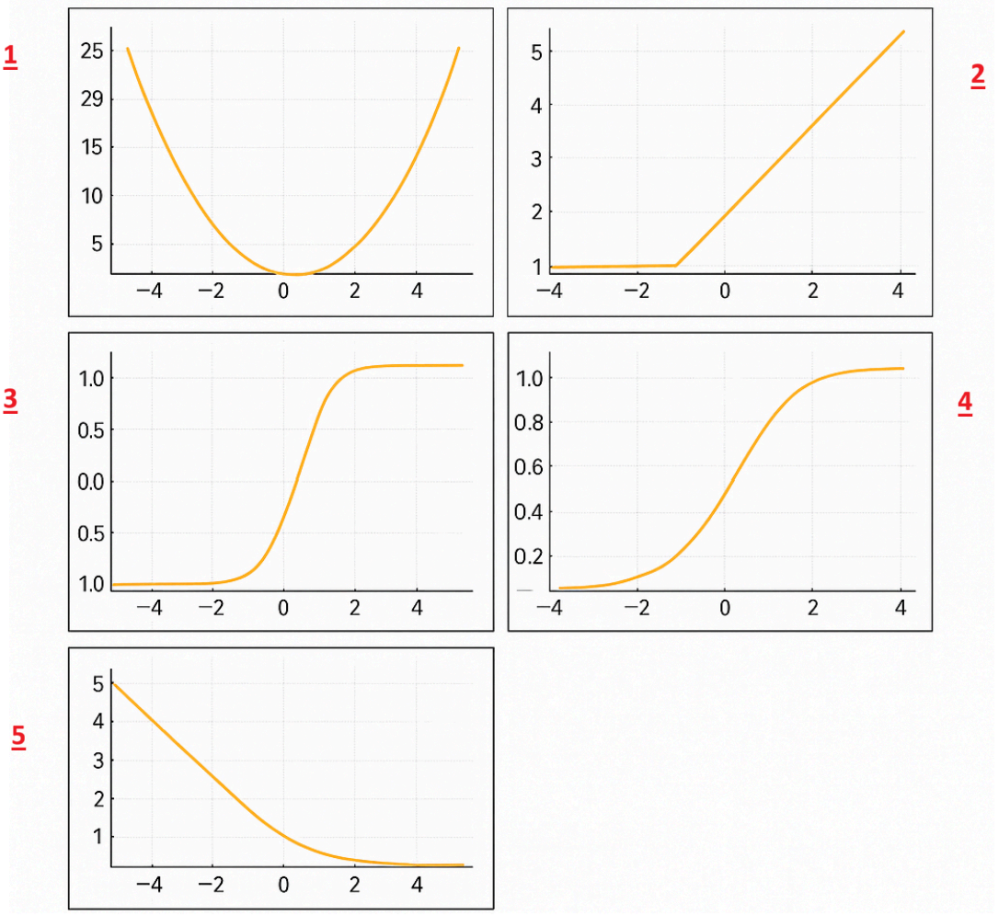
④ 손실이 계산된 후, 연산 경로를 따라 자동으로 편미분이 수행되어, 각 학습 대상 파라미터에 변화량(기울기)이 저장된다.

⑤ 학습이 끝난 후, 연산 그래프에 저장된 결과들을 다시 사용해 모델의 파라미터를 조정하며, 이 과정은 매 반복마다 정방향 계산을 새로 수행해야 한다.

6. PyTorch에서 `nn.Linear(1, 10)`을 선언했을 때 내부 동작과 관련된 설명 중 올바른 것을 모두 고르시오.

① 입력값 하나에 대해 서로 다른 10개의 가중치와 10개의 편향 조합을 적용하여 10개의 출력값이 생성된다.
② 이 계층의 가중치와 편향은 `layer.weight`, `layer.bias` 속성에 저장되며, 학습 중 자동으로 갱신된다.
③ 입력값이 1개이므로 출력값도 1개이며, 가중치 개수는 1개이다.
④ 이 계층은 비선형 활성화 함수(ReLU 등)를 포함하고 있어 음수 출력은 제거된다.
⑤ 이 계층은 선형 연산 $y = xW^T + b$ 을 수행하며, 연산 결과는 출력 차원 수만큼의 벡터로 반환된다.

7. 다음은 딥러닝에서 쓰이는 대표적인 함수의 그래프이다. 순서에 맞게 알맞게 나열한 것을 고르시오.(좌상에서 좌우로 행렬 방향순으로)



1.MSE, ReLU, Tanh, Sigmoid, Cross Entropy
2.Quadratic, ReLU, Sigmoid, Tanh, Cross Entropy
3.Cross Entropy, Linear, Sigmoid, Tanh, MSE
4.MSE, ReLU, Sigmoid, Tanh, Cross Entropy
5.Quadratic, Linear, Tanh, Sigmoid, Cross Entropy

8. 다음은 Iris 데이터를 분류하기 위한 다중 분류 모델을 PyTorch로 구현한 코드의 일부이다.이 코드에 대한 분석으로 가장 적절하지 않은 것을 고르시오.

```
class Net(nn.Module):
    def __init__(self, n_input, n_output):
        super().__init__()
        self.l1 = nn.Linear(n_input, n_output)

    def forward(self, x):
        return self.l1(x)

model = Net(n_input=2, n_output=3)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)

for epoch in range(100):
    model.train()
    optimizer.zero_grad()
    output = model(torch.Tensor(x_train))
    loss = criterion(output, torch.LongTensor(y_train))
    loss.backward()
    optimizer.step()
```

---

① 이 모델은 활성화 함수를 사용하지 않지만 `CrossEntropyLoss()`가 softmax를 포함하므로 정상적으로 학습된다.

② 출력층은 3개의 뉴런을 사용하므로 분류 클래스가 3개임을 유추할 수 있다.

③ 입력 데이터는 2차원이므로 2D 평면에 분류 경계를 시각화할 수 있다.

④ `torch.Tensor(x_train)`은 float 타입이므로 정답 라벨도 float 타입으로 맞춰야 한다.

⑤ `nn.Linear`는 입력값에 대해 선형 변환을 수행하며 학습 가능한 파라미터를 가진다.

9. 다음은 하나의 합성곱(Convolution) 레이어에 대한 파라미터 설정이다. 이 때, 입력 이미지의 가로 길이가 32일 때, 합성곱 연산 후의 출력 가로 크기(너비)를 구하시오.

입력 크기 (ninn_{in}nin) = 32

커널 크기 (k) = 5

패딩 크기 (p) = 2

스트라이드 (s) = 1

---

① 28

② 30

③ 31

④ 32

⑤ 33

10. 다음 보기 중에서 설명과 알고리즘의 연결이 올바르지 않은 묶음을 모두 고르시오.선형 회귀에서 기울기(θ_1)를 다음과 같이 계산하는 이유 및 관련된 설명으로 옳은 것을 모두 고르면?

①

- 모멘텀(Momentum) 개념을 도입하여 관성 효과로 진동을 줄이는 방식
- SGD보다 빠르게 수렴 가능
- 대표 알고리즘: SGD + Momentum, NAG

②

- 학습률을 각 파라미터마다 자동으로 조정해주는 방식
- 많이 업데이트된 파라미터는 학습률이 줄어든다
- 대표 알고리즘: Adagrad, Adam

③

- 평균 제곱 기울기 값을 이용하여 학습률을 조정한다
- 기울기가 큰 방향은 학습률을 줄이고, 작은 방향은 유지한다
- 대표 알고리즘: RMSProp, Adagrad

④

- 일정한 학습률을 고정하여 사용하는 전통적인 방법
- 데이터 1개마다 가중치를 업데이트하므로 느리다
- 대표 알고리즘: GD, SGD

⑤

- 적응형 기울기 방식과 모멘텀을 결합한 방식
- 대표적으로 학습률 감소가 없고, 항상 같은 보폭으로 움직인다
- 대표 알고리즘: Adam

11. 다음 중 He 초기화(He Initialization)에 대한 설명으로 옳지 않은 것은 무엇인가?

① He 초기화는 주로 ReLU 활성화 함수를 사용할 때 효과적으로 동작한다.
② He 초기화는 Xavier 초기화에 비해 입력 노드 수의 영향을 더 크게 반영한다.

③ He Normal 초기화는 평균이 0이고 분산이 $\frac{2}{n_{in}}$ 인 정규분포에서 가중치를 샘플링한다.

④ He Uniform 초기화는 $\left[-\sqrt{\frac{6}{n_{in}}}, +\sqrt{\frac{6}{n_{in}}}\right]$ 6] 범위에서 균등 분포로 초기화한다.
⑤ He 초기화는 입력 노드 수가 클수록 가중치의 분산을 줄이기 위한 방식이다.

12. 다음 중 정규분포(Normal Distribution)에 대한 설명으로 옳은 것을 모두 고르시오.

① 정규분포는 가우스가 중심극한정리를 수학적으로 증명하는 과정에서 처음 정의되었으며, 그 형태는 관측값이 평균 주변에 대칭적으로 밀집하는 특징을 가진다.

② 중심극한정리는 모집단의 분포와 무관하게, 독립이고 동일한 분포를 따르는 확률변수들의 평균이 표본 크기가 커짐에 따라 정규분포로 수렴한다는 이론이다.

③ 정규분포의 확률밀도함수는 표준정규분포를 기준으로 편차가 작아질수록 곡선의 폭이 넓어지고, 높이는 낮아진다.

④ 정규분포는 연속 확률분포로, 전체 확률이 항상 1이 되도록 정의되어 있으며, 확률밀도함수는 평균과 분산을 모수로 가진다.

⑤ 정규분포의 가장 큰 특징은 평균과 분산이 항상 동일하다는 점이며, 이는 표준정규분포에서 특히 두드러진다.

13. 다음 중 Adaptive Average Pooling (`AdaptiveAvgPool2d`)에 대한 설명으로 옳은 것을 모두 고르시오.

① Adaptive Average Pooling은 커널 크기와 스트라이드를 직접 지정하지 않고, 원하는 출력 크기를 기준으로 커널 크기와 스트라이드를 자동 조정한다.

② Adaptive Average Pooling을 사용하면, 서로 다른 크기의 입력 이미지도 동일한 크기의 출력 feature map으로 변환할 수 있으므로, Fully Connected Layer에 고정된 입력 크기를 제공할 수 있다.

③ Adaptive Average Pooling은 일반 Average Pooling보다 더 정확한 결과를 내기 위해 가중 평균을 사용한다.

④ Adaptive Average Pooling은 입력 텐서의 채널 수를 줄이기 위한 기법이며, 출력의 채널 수는 항상 1로 고정된다.

⑤ AdaptiveAvgPool2d((1, 1))은 글로벌 평균 풀링(Global Average Pooling)과 동일하게 작동할 수 있다.

14. 다음 중 Two-Stage Object Detection과 R-CNN 계열 모델(Fast R-CNN, Faster R-CNN 포함)에 대한 설명으로 올바른 설명을 모두 고르시오.

① R-CNN은 후보 영역을 만들기 위해 그래프 기반 세그멘테이션과 region 병합을 사용하는 selective search 기법을 활용하며, 후보 영역마다 CNN을 반복 수행하기 때문에 처리 속도가 매우 느리다.

② Fast R-CNN은 R-CNN과 달리, 입력 이미지를 한 번만 CNN에 통과시켜 Feature Map을 생성하고, Region of Interest를 그 위에서 추출하여 공유 연산 구조로 속도를 개선하였다.

③ Faster R-CNN은 Region Proposal Network(RPN)를 추가하여 후보 영역 생성을 네트워크 내부 연산으로 통합했고, 이를 통해 End-to-End 학습이 가능해졌다.

④ IOU(Intersection over Union)는 region proposal의 성능을 정량적으로 비교하는 수치로, Faster R-CNN에서는 이를 이용해 anchor box의 정답 여부를 판단하며, 높은 IOU를 가진 proposal만 남긴다.

⑤ 모든 R-CNN 계열은 속도 향상을 위해 CNN feature extractor와 SVM, Bounding box regressor를 별도의 모듈로 독립적으로 운영한다.

15. 다음 중 순환 신경망 계열(RNN, LSTM, Seq2Seq)의 구조 및 학습 특성에 대해, 옳은 설명을 모두 고르시오.

① Vanila RNN은 이전 hidden state와 현재 입력을 기반으로 현재 상태를 계산하며, tanh나 sigmoid와 같은 비선형 함수를 사용한다.

② RNN은 구조상 시퀀스가 길어질수록 gradient exploding 문제가 발생하기 쉬우며, 이를 방지하기 위해 output gate를 삽입한 구조가 LSTM이다.

③ LSTM에서는 셀 상태(cell state)를 중심으로 정보 흐름을 유지하며, 입력 게이트·망각 게이트·출력 게이트의 조합으로 중요한 정보만 기억하거나 제거할 수 있다.

④ Seq2Seq 모델은 인코더와 디코더로 구성되며, attention 메커니즘이 없어도 context vector 하나로 긴 문장의 모든 의미를 안정적으로 유지할 수 있다.

⑤ LSTM 기반 분류에서 마지막 hidden state를 사용하여 전체 시퀀스를 대표하는 출력으로 활용할 수 있다.

16. 다음 중 Vision Transformer(ViT)의 구조 및 작동 방식에 대해, 옳은 것을 모두 고르시오.

① 입력 이미지는 고정 크기의 패치로 분할된 후, 각 패치는 평탄화(Flatten)되고 선형 계층(Linear layer)을 통해 고정 차원의 임베딩 벡터로 변환되어 Transformer의 입력 시퀀스로 사용된다.

② ViT는 convolution 연산을 전혀 사용하지 않기 때문에, 이미지의 위치 정보를 효과적으로 처리하기 위해 sin/cos 기반의 fixed positional encoding만 사용한다.

③ 패치 임베딩은 `Conv2d(in_channels, embed_dim, kernel_size=patch_size, stride=patch_size)` 구조로 구현되며, 각 패치 임베딩 뒤에는 class token이 앞에 추가되고, positional encoding은 learnable 파라미터로 덧붙여진다.

④ Self-Attention은 모든 패치 간 상호작용을 고려할 수 있어 전역 정보를 학습하는 데 강점을 가지며, 따라서 local 구조의 인접성을 자동으로 보존하기 때문에 위치 인코딩이 반드시 필요한 것은 아니다.

⑤ ViT는 대규모 데이터 없이도 빠르게 수렴하고 높은 성능을 내는 것이 특징이며, 일반적인 CNN보다 적은 데이터에서도 일반화 성능이 우수하게 나타난다.

17. 다음 중 객체 추적(Object Tracking) 알고리즘 및 Optical Flow 기반 기법의 작동 방식에 대한 설명으로 옳은 것을 모두 고르시오.

① Mean-Shift는 히스토그램 기반의 커널 윈도우 내에서 픽셀 분포의 평균을 반복적으로 계산하여 객체 중심을 이동시키며, 영상 내 움직임의 방향이나 속도 정보는 고려하지 않는다.

② Camshift는 Mean-Shift의 확장으로, 커널 창의 위치뿐 아니라 크기와 방향도 동적으로 조정되며, HSV 컬러 히스토그램과 백프로젝션을 이용해 얼굴 등의 물체 추적에 자주 사용된다.

③ Lucas-Kanade Optical Flow는 작은 이동을 가정하고 밝기 불변 조건과 국소 창의 선형 근사를 이용해 속도를 계산하며, 실습에서는 `cv2.calcOpticalFlowPyrLK()`를 사용해 다중 프레임 기반 추적을 구현하였다.

④ RAFT는 고정된 이미지 피라미드를 사용하는 전통적인 Dense Optical Flow 기법으로, 고속 계산을 위해 사전 정의된 필터를 각 픽셀에 반복 적용하여 흐름 벡터를 추정한다.

⑤ RAFT는 모든 픽셀 쌍 간의 상호 작용을 계산하기 위해 All-Pairs Correlation Volume을 사용하며, 이를 GRU 기반의 recurrent unit을 통해 iterative하게 refinement 하여 높은 정확도의 Optical Flow를 제공한다.

18. 다음 중 OpenCV에서 영상의 경계(Edge) 또는 객체 검출과 관련된 설명으로 가장 부정확한 것을 고르시오.

① Sobel 필터는 영상의 수평/수직 방향 밝기 변화를 각각 GxG_xGx, GyG_yGy로 계산하고, 두 방향의 그래디언트 크기를 이용해 경계 강도를 추정할 수 있다.

② Canny 엣지 검출은 노이즈 제거, 그래디언트 계산, 비최대 억제(Non-maximum suppression), 이력 임계값 연결(Hysteresis thresholding)의 4단계를 거친다.

③ Hough 변환은 엣지 이미지에서 직선 또는 원의 좌표계에서의 위치 변화를 누적 공간에서 찾는 방법이며, 원 검출에는 누적 공간 차원이 2개로 충분하다.

④ 레이블링(Labeling)은 연결된 픽셀 집합에 고유한 ID를 부여하는 과정이며, 8-연결성을 사용할 경우 대각선 방향까지 인접성을 판단한다.

⑤ Hu 모멘트는 이진 영상의 윤곽선 형태를 대표하는 7개의 불변 특징을 추출하는 방법이며, 회전, 크기, 이동에 관계없이 동일한 객체를 비교할 수 있도록 도와준다.

19. 다음 중 GAN(GAN, cGAN, DCGAN 포함)의 구조적 특성과 학습 과정에 대한 설명으로 옳은 항목만을 모두 고르시오.

① GAN에서 Generator는 무작위 벡터를 입력받아 실제와 유사한 데이터를 생성하며, Discriminator는 입력 데이터가 실제(real)인지 생성(fake)인지 판단하여 두 모델이 상호 경쟁적으로 학습한다.

② Generator는 Discriminator의 출력을 1로 만들도록 학습되며, Discriminator 역시 Generator가 생성한 가짜 데이터를 1로 판단할 수 있도록 학습된다.

③ Conditional GAN은 조건 정보를 noise 벡터와 결합하여 Generator에 전달하며, Discriminator에도 동일한 조건 정보를 함께 입력하여, 조건 기반 구별이 가능하도록 설계된다.

④ DCGAN 구조에서는 Generator에서 ConvTranspose2d 계층을 활용해 점차적으로 해상도를 증가시키며, Discriminator에서는 Conv2d 계층으로 입력 이미지를 축소시켜 진위 여부를 판단한다.

⑤ 고해상도 이미지 생성을 위해서는 오히려 활성화 함수와 정규화 계층을 제거하는 것이 일반적이며, 단순 구조로 갈수록 학습 안정성이 향상된다.

20. 다음 중 딥러닝 모델 학습 중 발생 가능한 일반화 문제 및 그에 영향을 미치는 하이퍼파라미터와 최적화 요소에 대한 설명으로, 구조적 이해와 작동 원리를 정확히 반영한 항목만을 모두 고르시오.

① 모델의 과적합은 주로 하이퍼파라미터 설정 오류보다 데이터 수의 부족이나 네트워크 구조의 깊이 문제로 발생하므로, 학습률을 조정하는 것으로는 해결되지 않는다.

② 학습률이 너무 낮을 경우 손실 함수의 값이 감소하지 않고 정체될 수 있으며, 이는 최적화가 진행되지 않는 것처럼 보이게 한다. 반대로 너무 높으면 파라미터가 진동하거나 발산할 수 있다.

③ 하이퍼파라미터에는 학습률, 에폭 수, 배치 크기뿐 아니라 가중치(weight)도 포함되며, 이들은 학습 과정 중 동적으로 업데이트되는 값이기 때문에 적절히 초기화해야 한다.

④ 학습률 최적화를 위해 Grid Search를 사용할 경우 각 후보 값마다 전체 학습을 반복해야 하므로, 시간 복잡도는 선형이지만 실제 계산 비용은 매우 크다.

⑤ 과적합을 방지하기 위해 드롭아웃, 조기 종료(Early Stopping), Weight Decay 같은 기법이 사용되며, 이들 역시 하이퍼파라미터로 조정될 수 있다.

1번: ② A → B → C → D → E

2번: A, B, C, D

3번: ①, ②

4번: ①, ②, ③, ④

5번: ①, ③, ④, ⑤

6번: ①, ②, ⑤

7번: 1

8번: ④

9번: ④

10번: ③, ⑤

11번: ④

12번: ①, ②, ④

13번: ①, ②, ⑤

14번: ①, ②, ③, ④

15번: ①, ③, ⑤

16번: ①, ③, ④

17번: ①, ②, ③, ⑤

18번: ③

19번: ①, ③, ④

20번: ②, ④, ⑤