(인공지능빅데이터) 연습문제_04

1. 다음 중 **단층 신경망(Single Layer Perceptron)의 한계**에 대한 설명으로 **틀린 것**은?
    1. 단층 신경망은 선형적으로 구분 가능한 문제는 해결할 수 있다.
    2. 단층 신경망은 XOR 문제와 같은 선형적으로 구분이 불가능한 문제를 해결할 수 있다. ❤
    3. 단층 신경망은 다층 신경망보다 표현력이 제한적이다.
    4. 단층 신경망은 입력과 출력 사이의 복잡한 비선형 관계를 학습하기 어렵다.
    5. 단층 신경망의 한계를 극복하기 위해 활성화 함수와 은닉층을 추가한 다층 신경망(MLP)이 개발되었다.
2. 다음 중 **다층 신경망(MLP)에서 발생하는 가중치 소실(Gradient Vanishing) 문제**에 대한 설명으로 **틀린 것**은?
    1. 가중치 소실 문제는 역전파 과정에서 기울기(Gradient)가 점점 작아져 가중치가 거의 업데이트되지 않는 현상이다.
    2. 활성화 함수로 시그모이드(Sigmoid)나 하이퍼볼릭 탄젠트(Tanh)를 사용할 때 가중치 소실 문제가 발생할 가능성이 높다.
    3. 깊은 신경망일수록 가중치 소실 문제가 심각해질 수 있다.
    4. 가중치 소실 문제를 해결하기 위해 ReLU와 같은 활성화 함수가 도입되었다.
    5. 가중치 소실 문제는 학습률을 높이면 자연스럽게 해결된다. ❤
3. 다음 중 **다층 신경망(MLP)에서 발생하는 과적합(Overfitting) 문제**에 대한 설명으로 **틀린 것**은?
    1. 과적합이 발생하면 신경망 모델이 학습 데이터에는 높은 성능을 보이지만 새로운 데이터에는 일반화되지 않는다.
    2. 과적합을 방지하기 위해 드롭아웃(Dropout)과 같은 정규화 기법을 사용할 수 있다.
    3. 훈련 데이터가 많을수록 과적합이 심해진다. ❤
    4. 가중치의 크기를 제한하는 L1, L2 정규화 기법이 과적합 방지에 도움을 줄 수 있다.
    5. 모델의 복잡도를 조절하여 적절한 구조를 선택하는 것이 과적합을 줄이는 방법 중 하나이다.
4. 다음 중 **신경망 모델 학습에서 적절하지 않은 학습률(Learning Rate)이 초래하는 문제**에 대한 설명으로 **틀린 것**은?

    1. 학습률이 너무 크면 최적값을 지나치게 크게 이동하여 학습이 수렴하지 않을 수 있다.
    2. 학습률이 너무 작으면 학습 속도가 느려지고 지역 최적값(Local Minimum)에서 정체될 가능성이 높다.
    3. 학습률을 적절히 조절하면 신경망이 더 빠르고 안정적으로 최적값에 수렴할 수 있다.
    4. 학습률을 너무 크게 설정하면 가중치가 발산하여 모델의 성능이 급격히 향상될 수 있다. ❤
    5. 학습률 조정 기법(Adaptive Learning Rate, Learning Rate Decay 등)을 활용하면 학습 속도와 성능을 개선할 수 있다.
5. 다음 중 **신경망 모델에서 지역 최저점(Local Minimum)과 전역 최저점(Global Minimum)** 에 대한 설명으로 **틀린 것**은?
    1. 전역 최저점은 손실 함수의 모든 가능한 값 중에서 가장 낮은 값이다.
    2. 지역 최저점은 특정 영역 내에서 가장 낮은 손실 값을 가지지만, 전체 손실 함수에서 가장 낮은 값은 아닐 수 있다.
    3. 신경망의 손실 함수는 일반적으로 비선형적이기 때문에 지역 최저점에 빠지기 쉽다.
    4. 지역 최저점에 도달하면 모델의 성능이 더 이상 향상되지 않으므로 최적의 솔루션으로 간주할 수 있다. ❤
    5. 전역 최저점에 도달하는 것은 신경망의 학습 과정에서 가장 바람직한 결과이다.
6. 다음 중 **신경망 모델의 활성화 함수(Activation Function) 종류**에 대한 설명으로 **틀린 것**은?
    1. 시그모이드(Sigmoid) 함수는 출력 값을 0과 1 사이로 압축하므로 이진 분류 문제에서 주로 사용된다.
    2. ReLU(Rectified Linear Unit) 함수는 음수 입력에 대해 0을 출력하고, 양수 입력에 대해서는 입력 값을 그대로 출력한다.
    3. 하이퍼볼릭 탄젠트(Tanh) 함수는 출력 값을 -1과 1 사이로 압축하여 시그모이드 함수보다 더 나은 성능을 보일 수 있다.
    4. 소프트맥스(Softmax) 함수는 여러 클래스의 확률을 출력하기 위해 사용되며, 출력 값의 합이 1이 되도록 정규화한다.
    5. 리니어(Linear) 활성화 함수는 비선형 문제를 해결하는 데 적합하므로, 주로 중간 레이어에서 사용된다. ❤
7. 다음 중 **신경망 모델의 가중치 초기화 기법**에 대한 설명으로 **틀린 것**은?
    1. 모든 가중치를 0으로 초기화하면 학습이 정상적으로 진행되지 않는다.
    2. Xavier 초기화는 각 층의 입력 노드 수에 따라 가중치를 초기화하여 비선형 활성화 함수의 출력을 안정화하는 데 도움을 준다.
    3. He 초기화는 ReLU 활성화 함수를 사용할 때 효과적으로 작동하며, 가중치를 정규 분포를 따르도록 초기화한다.
    4. 랜덤 초기화는 가중치를 무작위 값으로 설정하여 대칭성을 깨뜨리는 방법이다.

    5. 초기화된 가중치의 크기가 너무 크면 학습 과정에서 기울기가 사라져 가중치가 업데이트되지 않는 문제를 초래할 수 있다. ❤

8. 다음 중 **신경망 모델의 학습 방법에서 사용되는 최적화 기법**에 대한 설명으로 **틀린 것**은?
    1. 경사 하강법(Gradient Descent)은 손실 함수의 기울기를 계산하여 가중치를 업데이트하는 기본적인 최적화 알고리즘이다.
    2. 확률적 경사 하강법(Stochastic Gradient Descent, SGD)은 전체 데이터셋을 사용하는 대신 임의로 선택된 일부 샘플만을 사용하여 기울기를 계산한다.
    3. 모멘텀(Momentum) 기법은 이전 기울기를 고려하여 가중치 업데이트의 방향과 크기를 조정함으로써 학습 속도를 증가시킨다.
    4. Adam(Adaptive Moment Estimation) 최적화 기법은 학습률을 일정하게 유지하며, 매개변수마다 다른 학습률을 적용한다. ❤
    5. AdaGrad(AdaGrad) 알고리즘은 각 매개변수에 대해 적응형 학습률을 적용하여 자주 업데이트되는 매개변수의 학습률을 낮춘다.

- 다음은 pytorch를 이용하여 간단한 MLP 모델을 만드는 code 의 일부분으로 학습을 정의한 부분이다.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"set device = {device}")
model = MLP().to(device)
criterion = torch.nn.CrossEntropyLoss().to(device)
optimizer = torch.optim.SGD(model.parameters(), lr = 1e-4)

epochs = 20
torch.cuda.empty_cache()
train_losses = []
test_losses = []
train_acc = []
test_acc = []

for e in range(epochs):
    # training loop
    running_loss = 0
    running_accuracy = 0
    model.train()
    for _, data in enumerate(tqdm(train_loader)):
        # training phase
        inputs, labels = data
        inputs = inputs.to(device).float()
        labels = labels.to(device).long()
        optimizer.zero_grad()

        # forward
        outputs = model(inputs)
```

```
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        # backward
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        running_accuracy += torch.sum(preds ==
labels.data).detach().cpu().numpy()/inputs.size(0)
```

10. 다음은 위 코드에 대한 설명이다. 틀린 설명을 고르시오.

1) epochs = 20, 으로 설정하여 한번에 입력되는 학습 batch 크기가 20임을 정하고 있다. ❤
2) optimizer.zero_grad() 는 새로운 batch 입력 데이터에 대해서 학습을 진행하기 위하여 기존 가중치의 변화량을 계산한 gradient를 초기화하는 명령이다.
3) criterion = torch.nn.CrossEntropyLoss().to(gpu) 는 손실함수를 정의한 부분이다.
4) optimizer = torch.optim.SGD(model.parameters(), lr = 1e-4) 는 최적하기법을 정의한 부분으로 learning rage 은 1e-4 로 하였다.
5) loss.backward() 는 Loss 를 역전파하는 과정이고, optimizer.step() 학습을 위한 Step을 진행시키는 과정이다.

11. 다음은 위 코드에 대한 설명이다. 맞는 설명을 고르시오
    1. model = MLP().to(device) 은 MLP 클라스를 무조건 cpu에서 계산하도록 하는 코드이다.
    2. device = torch.device("cuda" if torch.cuda.is_available() else "cpu")는 torch가 GPU를 사용할 수 있는지 아니면 CPU를 사용할 수 있는지 확인하여 deive 변수에 대입하는 코드이다. ❤
    3. for e in range(epochs): 부분은 epochs 파라메터를 20으로 정하였기 때문에 1부터 20까지 e 에 배당하고 반복한다.
    4. running_loss = 0 은 한 step 동안 계산할 loss를 초기화한 것이다.
    5. model.train() 은 모델이 validation 중임을 나타낸다.

- 다음은 pytorch를 이용하여 간단한 MLP 모델을 만드는 code 의 일부분으로 test 과정을 정의한 부분이다.

```
### Test
model.eval()
y_pred = []
y_true = []
```

```
with torch.no_grad():
    for _, data in enumerate(tqdm(test_loader)):
        inputs, labels = data
        inputs = inputs.to(device).float()
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        y_pred += list(preds.detach().cpu().numpy())
        y_true += list(labels.detach().numpy())
```

12. 다음은 with torch.no_grad(): 코드에 대한 설명이다. 맞는 설명을 고르시오.
    1. test 과정을 진행 할 수 있도록 가중치가 학습할 수 있게 준비하는 코드이다.
    2. epochs 동안 학습된 가중치를 초기화 하는 코드이다.
    3. test 과정을 위하여 with 의 범위 안에서는 가중치가 변화지 않다록 선언하는 것이다. ❤
    4. epochs 만큼 반복을 위하여 iterator를 정의하는 것이다.
    5. test의 loss를 계산하기 위하여 손실함수를 지정하는 코드이다.
13. 가중치를 네트워크의 입력과 출력 연결 수에 따라 조정된 값으로 초기화하는 방법으로 활성화 함수로 하이퍼볼릭 탄젠트(tanh) 또는 시그모이드(sigmoid)를 사용할 때 효과적 것은 무엇인지 고르시오.
    1. Xavier 초기화 (Xavier Initialization 또는 Glorot Initialization) ❤
    2. 정규화 초기화 (Normalization Initialization)
    3. 레이어마다 다른 초기화 (Layer-wise Initialization)
    4. 무작위 초기화 (Random Initialization)
    5. 균등 초기화 (Uniform Initialization)
14. 다음 중 신경망 모델의 과적합을 막기 위한 정규화 기법인 L1, L2 정규화에 대한 설명으로 틀린 것은?
    1. L1 정규화는 가중치의 절대값의 합을 손실 함수에 추가하여, 일부 가중치를 정확히 0으로 만들어 변수 선택의 효과를 가진다.
    2. L2 정규화는 가중치의 제곱합을 손실 함수에 추가하여, 가중치를 0에 가깝게 줄이지만 완전히 0으로 만들지는 않는다.
    3. L1 정규화는 L2 정규화에 비해 이상치에 덜 민감하며, 모델의 일반화 성능을 향상시키는 데 유리하다. ❤
    4. L1 정규화는 Lasso 회귀 모델에서, L2 정규화는 Ridge 회귀 모델에서 사용된다.
    5. L1, L2 정규화는 모두 모델의 복잡도를 줄여 과적합을 방지하는 데 사용된다.
15. Drop out에 대한 설명으로 올바른 것을 고르시오.
    1. Drop out은 모델의 학습 과정에서 가중치 값을 무작위로 조정하여 오버피팅을 방지하는 기법이다.

2. Drop out은 학습 과정에서 일부 뉴런을 무작위로 비활성화하여 오버피팅을 방지하는 기법이다. ❤
3. Drop out은 모델의 손실 함수에 패널티를 부여하여 오버피팅을 방지하는 기법이다.
4. Drop out은 학습률을 동적으로 조절하여 오버피팅을 방지하는 기법이다.
5. Drop out은 모델의 가중치를 정규화하여 오버피팅을 방지하는 기법이다.