(인공지능과빅데이터) 연습문제_05

1. 다층 퍼셉트론(MLP) 신경망 모델에서 과적합(Overfitting)을 방지하기 위해 Dropout 기법을 사용할 때, 학습 단계와 평가 또는 테스트 단계에서 유의해야 하는 가장 중요한 차이점은 무엇가?
    1. 학습 시에는 Dropout을 적용하여 일부 뉴런의 출력을 무작위로 0으로 만들지만, 평가/테스트 시에는 모든 뉴런을 사용하고 각 뉴런의 출력에 학습 시의 Dropout 비율을 곱하여 최종 출력을 결정한다. ❤
    2. 학습 시에는 모든 뉴런을 활성화하여 모델의 표현력을 최대화하고, 평가/테스트 시에만 Dropout을 적용하여 일반화 성능을 측정한다.
    3. 학습 시와 평가/테스트 시 모두 동일한 Dropout 비율로 무작위로 뉴런을 비활성화하여 모델의 안정성을 유지해야 한다.
    4. 학습 시에는 Dropout 비율을 점진적으로 증가시켜 모델이 점차 더 많은 노이즈에 강해지도록 훈련하고, 평가/테스트 시에는 Dropout을 적용하지 않는다.
    5. Dropout은 학습 단계에서만 사용되는 규제 기법이며, 평가/테스트 단계에서는 모델의 실제 성능을 정확하게 측정하기 위해 반드시 비활성화해야 한다.
2. PyTorch로 구현된 신경망 모델에서 Dropout 레이어를 사용할 때, 학습 단계에서 `model.train()`을 호출하는 것과 평가/테스트 단계에서 `model.eval()`을 호출하는 것의 가장 중요한 차이점은 무엇가?
    1. `model.train()`은 모델의 모든 파라미터의 requires_grad 속성을 True로 설정하여 역전파 계산을 활성화하고, `model.eval()`은 이를 False로 설정하여 불필요한 계산을 방지한다. Dropout의 동작에는 영향을 미치지 않는다.
    2. `model.train()`은 모델 내의 Dropout 레이어를 활성화하여 학습 과정에서 무작위로 뉴런을 비활성화시키고, `model.eval()`은 Dropout 레이어를 비활성화하여 평가/테스트 시 모든 뉴런이 사용되도록 한다. ❤
    3. `model.train()`은 배치 정규화(BatchNorm) 레이어의 통계(평균과 분산)를 현재 배치 기준으로 업데이트하고, `model.eval()`은 학습 과정에서 계산된 이동 평균(running average)과 이동 분산(running variance)을 사용하여 배치 정규화를 수행한다. Dropout과는 관련이 없다.
    4. `model.train()`은 모델의 학습 가능한 파라미터를 CUDA 장치로 이동시키고, `model.eval()`은 CPU로 이동시켜 메모리 사용량을 줄인다. Dropout의 활성화 여부는 별도로 설정해야 한다.

    5. `model.train()`은 모델의 최적화기(optimizer)를 활성화하여 파라미터 업데이트를 가능하게 하고, `model.eval()`은 최적화기를 비활성화하여 파라미터가 고정되도록 한다. Dropout의 동작 방식은 두 모드에서 동일하다.

3. TensorFlow로 구현된 신경망 모델에서 Dropout 레이어를 사용할 때, 학습 단계와 평가/테스트 단계에서 Dropout의 동작 방식을 제어하기 위해 일반적으로 사용되는 방법으로 가장 **옳은** 것은 무엇가?
    1. TensorFlow의 Dropout 레이어는 학습 단계와 평가/테스트 단계에서 자동으로 다르게 작동하므로, 별도의 코드 변경 없이 모델을 학습하고 평가할 수 있다.
    2. 학습 단계에서는 Dropout 레이어를 활성화하여 `training=True`로 설정하고, 평가/테스트 단계에서는 `training=False`로 설정하여 Dropout을 비활성화한다. ❤

    > TensorFlow의 tf.keras.layers.Dropout 레이어는 training이라는 불리언(Boolean) 매개변수를 가진다. 학습 시에는 training=True로 설정하여 Dropout을 활성화하고, 평가/테스트 시에는 training=False로 설정하여 Dropout을 비활성화한다.

    3. 모델을 학습하기 전에 Dropout 레이어의 활성화 확률을 1.0으로 설정하여 모든 뉴런이 항상 활성화되도록 하고, 평가/테스트 전에 활성화 확률을 학습 시 사용한 비율로 조정한다.
    4. TensorFlow에서는 학습과 평가/테스트를 위한 별도의 Dropout 레이어 인스턴스를 생성하여, 학습 시에는 활성화하고 평가/테스트 시에는 비활성화하는 방식으로 제어한다.
    5. Dropout 레이어의 동작 방식은 모델 컴파일 시 설정되는 손실 함수(loss function)에 따라 자동으로 결정되므로, 사용자는 별도로 Dropout의 활성화 여부를 제어할 필요가 없다.
4. 데이터 전처리 과정에서 사용되는 Min-Max 정규화와 Standardization에 대한 설명으로 가장 **옳지 않은** 것은 무엇인가?
    1. Min-Max 정규화는 데이터를 특정 범위 (일반적으로 0과 1 사이)로 변환하며, 데이터의 상대적인 크기 관계를 유지한다.

        > 데이터 값을 설정된 최소값과 최대값 사이의 범위로 선형 변환하며, 원래 데이터 간의 비율 관계는 유지

    2. Standardization은 데이터의 평균을 0으로 만들고 표준편차를 1로 조정하며, 이상치(outlier)의 영향을 Min-Max 정규화보다 덜 받는다.

> 데이터의 중심을 평균으로 이동시키고 스케일을 표준편차로 조정하여, 이상치의 영향을 덜 받는다. 이는 이상치가 평균과 표준편차에 미치는 영향이 제한적이기 때문이다.

3. Min-Max 정규화는 데이터의 분포가 가우시안 분포를 따르지 않거나, 특정 범위로 스케일링하는 것이 중요한 경우에 유용하게 사용될 수 있다.

> Min-Max 정규화는 특정 범위로 출력을 제한해야 하는 경우에 효과적

4. Standardization은 데이터가 가우시안 분포를 따른다고 가정하는 알고리즘 (예: 선형 회귀, SVM)에서 더 나은 성능을 보이는 경향이 있으며, 이상치에 민감하게 반응한다. ❤
5. Min-Max 정규화와 Standardization 모두 학습 데이터셋을 기준으로 변환 파라미터 (최솟값, 최댓값, 평균, 표준편차)를 계산하고, 이를 동일하게 테스트 데이터셋에 적용해야 한다.

> 학습 데이터셋에서 계산된 변환 파라미터를 테스트 데이터셋에 동일하게 적용하는 것은 데이터 유출(data leakage)을 방지하고 일관된 전처리를 유지하는 데 매우 중요

5. 데이터 전처리 기법 중 하나인 비상관화(Decorrelation)와 화이트닝(Whitening)에 대한 설명으로 가장 **옳지 않은** 것은 무엇인가?
    1. 비상관화는 데이터의 특징(feature)들 간의 선형적인 상관 관계를 제거하는 것을 목표로 하며, 주성분 분석(PCA) 등을 통해 수행될 수 있다.
    2. 화이트닝은 비상관화된 데이터의 각 특징이 동일한 분산(일반적으로 1)을 갖도록 추가적으로 스케일링하는 과정을 포함한다.
    3. 비상관화는 모델 학습 시 특정 특징에 과도하게 의존하는 것을 방지하고, 학습 효율성을 높이는 데 도움을 줄 수 있다.
    4. 화이트닝된 데이터는 특징 간의 상관 관계가 없고 분산이 동일하므로, 일부 머신러닝 알고리즘의 수렴 속도를 향상시키고 성능을 개선할 수 있다.
    5. 비상관화는 데이터의 정보 손실을 최소화하면서 특징 간의 독립성을 최대한 확보하는 것을 목표로 하지만, 화이트닝은 데이터의 중요한 패턴까지 왜곡시킬 수 있어 항상 적용하는 것이 바람직하다. ❤

> 비상관화는 정보 손실을 최소화하려고 노력하지만, 차원 축소를 동반할 경우 일부 정보 손실이 발생할 수 있다. 화이트닝이 항상 중요한 패턴을 왜곡시킨다고 단정할 수는 없지만, 데이터의 분산을 조정하는 과정에서 특정

> 방향의 정보가 강조되거나 약화될 수 있다. 따라서 항상 적용하는 것이 바람직한 것은 아니다.

6. 머신러닝 모델 학습 시 발생하는 공변량 변화(Covariate Shift)와 내부 공변량 변화(Internal Covariate Shift)에 대한 설명으로 가장 **옳지 않은** 것은 무엇가?
    1. 공변량 변화는 학습 데이터의 입력 변수(feature) 분포와 테스트 데이터의 입력 변수 분포가 서로 다를 때 발생하는 현상을 의미한다.
    2. 내부 공변량 변화는 심층 신경망 학습 시, 이전 계층의 파라미터 변화로 인해 현재 계층의 입력 분포가 지속적으로 변하는 현상을 의미한다.
    3. 공변량 변화는 모델이 학습 데이터에 과적합되어 새로운 테스트 데이터에 대한 일반화 성능이 저하되는 주요 원인 중 하나이다.
    4. 배치 정규화(Batch Normalization)는 내부 공변량 변화 문제를 완화하여 심층 신경망의 학습을 안정화시키고 수렴 속도를 높이는 데 효과적인 기법이다.
    5. 공변량 변화와 내부 공변량 변화는 동일한 현상을 지칭하는 다른 용어이며, 모델의 성능 저하에 유사한 영향을 미친다. ❤

    > 공변량 변화와 내부 공변량 변화는 서로 다른 현상이다. 공변량 변화는 데이터 분포 자체의 차이에서 비롯되는 반면, 내부 공변량 변화는 모델 내부 학습 과정 중 파라미터 변화로 인해 발생한다.

7. 심층 신경망 모델에서 학습을 안정화하고 성능을 향상시키는 데 사용되는 배치 정규화(Batch Normalization)에 대한 설명으로 가장 **옳지 않은** 것은 무엇가?
    1. 배치 정규화는 각 계층의 활성화 출력을 미니배치 단위로 평균이 0, 분산이 1이 되도록 정규화한 후, 학습 가능한 스케일(scale)과 이동(shift) 파라미터를 적용하는 기법이다.
    2. 배치 정규화를 적용하면 내부 공변량 변화(Internal Covariate Shift) 문제를 완화하여 학습 속도를 향상시키고, 더 높은 학습률을 사용할 수 있도록 돕는다.
    3. 배치 정규화는 모델의 과적합(Overfitting)을 줄이는 규제(regularization) 효과를 가지며, Dropout과 함께 사용될 경우 그 효과가 감소할 수 있다.
    4. 평가(evaluation) 또는 테스트 단계에서는 학습 과정에서 계산된 각 계층별 이동 평균(running average)과 이동 분산(running variance)을 사용하여 입력을 정규화한다.
    5. 배치 정규화는 모든 신경망 구조에 필수적으로 적용해야 하며, 특히 순환 신경망(RNN)과 같은 순차 데이터 처리 모델에서는 그 효과가 매우 뛰어나다. ❤

> 모든 구조에 필수적으로 적용해야 하는 것은 아니다. 배치 크기가 매우 작거나 불안정한 경우 오히려 성능 저하 가능성. 순환 신경망(RNN)에서는 배치 정규화 적용이 더 복잡하다.

8. 신경망 모델에서 배치 정규화(Batch Normalization)의 대안으로 사용되는 레이어 정규화(Layer Normalization)에 대한 설명으로 가장 **옳은** 것은 무엇가?
    1. 레이어 정규화는 각 계층의 특정 뉴런에 대해 모든 데이터 샘플에 걸쳐 통계(평균과 분산)를 계산하여 정규화를 수행한다.
    2. 레이어 정규화는 각 데이터 샘플 내의 모든 특징(feature) 또는 뉴런에 걸쳐 통계(평균과 분산)를 계산하여 정규화를 수행한다. ❤

    > 레이어 정규화는 배치 크기에 크게 의존하지 않으므로, 작은 배치 크기에서도 안정적인 성능

    3. 레이어 정규화는 배치 정규화와 마찬가지로 미니배치 크기가 클수록 안정적인 정규화 효과를 보이며, 작은 배치 크기에서는 성능이 불안정해진다.
    4. 레이어 정규화는 주로 이미지 처리 분야의 컨볼루션 신경망(CNN)에서 배치 정규화보다 더 뛰어난 성능을 보이는 것으로 알려져 있다.
    5. 레이어 정규화는 학습 단계에서 계산된 이동 평균(running average)과 이동 분산(running variance)을 평가 또는 테스트 단계에서 사용하여 일관된 정규화 효과를 유지한다.

    > 레이어 정규화는 각 샘플 내에서 통계를 계산하므로, 학습 단계에서 계산된 이동 평균이나 이동 분산을 별도로 저장하고 평가/테스트 단계에서 사용할 필요가 없다.

9. 배치 정규화(Batch Normalization)와 레이어 정규화(Layer Normalization)를 비교한 설명 중 가장 **옳지 않은** 것은 무엇가?
    1. 배치 정규화는 각 계층의 특정 뉴런에 대해 미니배치 내의 모든 데이터 샘플에 걸쳐 평균과 분산을 계산하여 정규화하는 반면, 레이어 정규화는 각 데이터 샘플 내의 모든 뉴런에 걸쳐 평균과 분산을 계산하여 정규화한다.
    2. 배치 정규화는 작은 배치 크기에서 통계량 추정의 불안정성으로 인해 성능이 저하될 수 있지만, 레이어 정규화는 배치 크기에 영향을 받지 않고 안정적인 성능을 유지한다.

> 배치 정규화와 레이어 정규화의 중요한 차이이다.

3. 배치 정규화는 주로 컨볼루션 신경망(CNN)에서 효과적인 것으로 알려져 있으며, 레이어 정규화는 순환 신경망(RNN)과 같이 시퀀스 데이터를 처리하는 모델에서 더 유리한 경우가 많다.
4. 배치 정규화는 평가 또는 테스트 시 학습 과정에서 계산된 이동 평균(running average)과 이동 분산(running variance)을 사용하는 반면, 레이어 정규화는 각 입력 샘플에 대해 독립적으로 통계를 계산하여 정규화를 수행한다.
5. 배치 정규화와 레이어 정규화 모두 학습 가능한 스케일(scale)과 이동(shift) 파라미터를 도입하여, 모델이 데이터의 원래 표현력을 완전히 잃지 않도록 한다. 그러나 배치 정규화는 각 샘플별로 이러한 파라미터를 학습하고, 레이어 정규화는 각 뉴런별로 학습한다. ❤

> `배치 정규화와 레이어 정규화 모두 학습 가능한 스케일과 이동 파라미터를 도입하고 있다.'

`- 배치 정규화 (Batch Normalization): 각 계층의 특정 뉴런에 대해 하나의 스케일 파라미터 (γ, 감마)와 하나의 이동 파라미터 (β, 베타)를 학습한다. 이 파라미터들은 미니배치 내의 모든 샘플에 대해 공유된다. 즉, 특정 뉴런의 출력을 정규화한 후, 모든 샘플에 대해 동일한 스케일과 이동 값을 적용한다.`
`- 레이어 정규화 (Layer Normalization): 각 데이터 샘플 내의 모든 뉴런에 대해 하나의 스케일 파라미터 (γ, 감마)와 하나의 이동 파라미터 (β, 베타)를 학습한다. 즉, 각 샘플별로 자신만의 스케일과 이동 파라미터를 가진다.`
`위 보기는 배치 정규화와 레이어 정규화의 학습 가능한 파라미터 학습 주체를 반대로 설명하고 있다.`

10. PyTorch 신경망 모델에서 배치 정규화(BatchNorm) 레이어를 사용할 때, 학습 과정의 안정화 및 성능 향상을 위해 배치 정규화 코드를 **어떤 위치**에 삽입하는 것이 가장 적절한가?
    1. 각 활성화 함수 이후 (예: `nn.Linear -> activation -> nn.BatchNorm1d`)

> 활성화 함수의 출력을 정규화하는 것보다 선형/컨볼루션 연산 후의 값을 정규화하여 활성화 함수에 안정적인 입력을 제공하는 것이 더 효과적

2. 각 선형 레이어 또는 컨볼루션 레이어의 **출력 직후**, 활성화 함수 **직전** (예: `nn.Linear -> nn.BatchNorm1d -> activation`) ❤

> 배치 정규화는 일반적으로 각 선형 레이어 (Linear) 또는 컨볼루션 레이어 (Conv2d)의 출력 직후, 그리고 해당 레이어에 적용될 활성화 함수 (ReLU, Sigmoid 등) 직전에 삽입된다.

3. 모델의 가장 첫 번째 레이어의 입력 바로 이전

> 첫 번째 레이어 이전에 배치 정규화를 적용하는 것은 입력 데이터 자체를 정규화하는 것과 유사한 효과를 낼 수 있지만, 각 내부 레이어의 공변량 변화를 해결하는 주요 목적과는 거리가 있다.

4. 모델의 가장 마지막 레이어의 출력 바로 이후

> 가장 마지막 레이어 이후에 배치 정규화를 적용하는 것은 모델의 최종 출력을 정규화하는 것으로, 일반적으로 사용되지 않다.

5. 모델 정의 시, 모든 선형 레이어와 활성화 함수를 하나의 순차 컨테이너 (`nn.Sequential`)로 묶은 후, 컨테이너의 가장 마지막에 배치 정규화 레이어를 추가

> 컨테이너의 마지막에 배치 정규화를 추가하는 것은 각 레이어의 출력을 개별적으로 정규화하는 목적에 부합하지 않다.