


(인공지능과빅데이터) 연습문제_10

1. 다음 중 순환신경망(Recurrent Neural Network, RNN) 딥러닝 기법에 대한 설명으로 **가장 적절하지 않은** 것을 고르시오?
 1. 시퀀스 데이터(예: 텍스트, 음성) 처리에 특화된 신경망 구조이다.
 2. 이전 스텝의 출력이 현재 스텝의 입력으로 재사용되어 정보를 '기억'할 수 있다.
 3. 장기 의존성 문제(long-term dependency problem) 해결을 위해 게이트(gate) 메커니즘을 도입한 LSTM이나 GRU와 같은 변형 모델들이 개발되었다.
 4. 모든 입력 스텝에서 동일한 가중치(weights)와 편향(biases)을 공유한다.
 5. 피드포워드 신경망(Feedforward Neural Network, FNN)과 달리, 정보의 흐름이 단방향으로만 이루어진다. 
2. 다음 중 기본적인 순환신경망(RNN)이 가지는 한계에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
 1. **장기 의존성 문제 (Long-term Dependency Problem):** 시퀀스 길이가 길어질수록 초기에 입력된 정보의 중요도가 희석되어 장기적인 맥락을 학습하기 어렵다.
 2. **기울기 소실 (Vanishing Gradient):** 역전파 과정에서 기울기 (gradient)가 매우 작아져 가중치 업데이트가 제대로 이루어지지 않아 학습이 어렵다.
 3. **기울기 폭주 (Exploding Gradient):** 역전파 과정에서 기울기가 너무 커져 가중치가 비정상적으로 업데이트되어 모델의 안정성을 해칠 수 있다.
 4. **병렬 처리의 용이성:** 시퀀스 상의 이전 스텝의 계산 결과가 현재 스텝의 입력으로 사용되므로, 시퀀스 전체에 대한 병렬 처리가 어렵다.
 5. **다양한 종류의 시퀀스 데이터 처리 불가능:** 텍스트, 음성 등 다양한 시퀀스 데이터를 처리하는 데 근본적인 제약이 있어 특정 종류의 데이터에만 적용 가능하다. 
3. 다음 중 LSTM(Long Short-Term Memory)의 특징과 기본적인 순환신경망(RNN)과의 차이점에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
 1. LSTM은 셀 상태(cell state)와 게이트(gate) 메커니즘(입력, 망각, 출력 게이트)을 도입하여 장기 의존성 문제를 효과적으로 해결한다.
 2. 기본 RNN은 과거 정보를 단순히 현재 스텝의 은닉 상태(hidden state)에 반영하는 반면, LSTM은 별도의 셀 상태를 통해 장기적인 정

보의 상태를 유지하고 흐름을 제어한다.

3. LSTM의 게이트는 시그모이드 활성화 함수를 사용하여 0과 1 사이의 값을 출력함으로써 정보의 흐름을 조절한다.
4. 기본 RNN과 달리 LSTM은 기울기 소실(vanishing gradient) 문제로부터 완전히 자유로워 어떠한 경우에도 기울기 소실이 발생하지 않는다.



5. LSTM은 기본 RNN보다 더 많은 파라미터(가중치와 편향)를 가지므로, 학습해야 할 파라미터의 수가 더 많다.




4. 다음 중 LSTM(Long Short-Term Memory)의 내부 구조에서 게이트(gate), 셀 상태(cell state), 은닉 상태(hidden state)에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?



1. **셀 상태(Cell State):** LSTM의 핵심으로, 시퀀스 전체에 걸쳐 장기적인 정보를 전달하는 '컨베이어 벨트' 역할을 하며, 게이트에 의해 정보가 추가되거나 삭제된다.
2. **망각 게이트(Forget Gate):** 이전 셀 상태에서 어떤 정보를 '잊어버릴'지 결정하며, 시그모이드 활성화 함수의 출력을 통해 해당 정보의 보존 여부를 조절한다.
3. **입력 게이트(Input Gate):** 현재 스텝의 입력과 이전 은닉 상태를 바탕으로 새로운 정보 중 어떤 것을 셀 상태에 '추가할'지 결정한다.
4. **출력 게이트(Output Gate):** 셀 상태의 정보를 바탕으로 현재 스텝의 최종 은닉 상태를 '생성'하며, 이는 다음 스텝의 입력으로도 사용된다.
5. **은닉 상태(Hidden State):** 셀 상태와 동일한 정보를 직접적으로 전달하며, 게이트의 제어를 받지 않고 매 스텝마다 모든 과거 정보를 그대로 축적한다.


5. 다음 중 GRU(Gated Recurrent Unit)와 LSTM(Long Short-Term Memory)의 차이점에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가요?

1. **게이트 수:** LSTM은 입력, 망각, 출력 게이트의 세 가지 게이트를 사용하는 반면, GRU는 업데이트 게이트와 리셋 게이트의 두 가지 게이트를 사용한다.
2. **셀 상태 유무:** LSTM은 장기 기억을 위한 별도의 셀 상태(cell state)를 가지지만, GRU는 셀 상태 없이 은닉 상태(hidden state)만으로 정보를 전달한다.
3. **복잡성:** GRU는 LSTM보다 구조가 더 단순하고 파라미터 수가 적어, 학습 속도가 더 빠르거나 더 작은 데이터셋에서 효과적일 수 있다.
4. **성능:** 특정 작업에서 LSTM이 GRU보다 항상 우수한 성능을 보이는 것은 아니며, 데이터셋의 특성이나 문제 유형에 따라 GRU가 더 좋은 결과를 낼 수도 있다.
5. **기울기 소실 문제 해결:** 두 모델 모두 기본적인 RNN이 가지는 기울기 소실(vanishing gradient) 문제를 완전히 해결하여, 어떠한 경우에도 기울기 소실이 발생하지 않는다.


6. 자연어 처리(Natural Language Processing, NLP)와 딥러닝(Deep Learning)의 관계에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?

1. 딥러닝 기술의 발전은 자연어 처리 분야에 혁명적인 변화를 가져왔으며, 특히 단어 임베딩, 시퀀스 모델링 등에서 큰 발전을 이루었다.
 2. NLP는 딥러닝이 등장하기 전부터 규칙 기반 시스템이나 통계적 머신러닝 기법을 활용하여 존재해왔던 분야이다.
 3. 딥러닝은 NLP 문제 해결을 위한 하나의 강력한 '도구' 또는 '접근 방식'이며, 딥러닝이 NLP 자체를 완전히 대체한 것은 아니다.
 4. RNN, LSTM, GRU와 같은 순환신경망 계열 모델은 언어의 순차적인 특성을 잘 반영하여 기계 번역, 텍스트 생성 등 다양한 NLP 태스크에서 핵심적인 역할을 한다.
 5. 딥러닝 모델은 NLP 문제에서 복잡한 언어 규칙을 사람이 직접 프로그래밍하는 방식을 완전히 없애버렸으며, 이제는 어떠한 NLP 태스크에서도 수동으로 규칙을 정의할 필요가 없다. 
7. 자연어 처리(Natural Language Processing, NLP) 과정에서 '토큰화(Tokenization)'에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
1. 토큰화는 텍스트를 의미 있는 작은 단위로 분리하는 과정을 의미하며, 이 작은 단위를 '토큰(token)'이라고 부른다.
 2. 분리된 토큰은 단어, 형태소, 또는 문자 단위가 될 수 있으며, 언어의 특성과 분석 목적에 따라 토큰화의 기준이 달라질 수 있다.
 3. 토큰화는 텍스트 데이터를 딥러닝 모델의 입력으로 사용하기 위한 전처리 과정 중 가장 첫 단계에 해당한다.
 4. 토큰화 과정에서는 구두점, 특수 문자, 대소문자 변환 등 텍스트 정규화 작업이 자동으로 수행되어 항상 깨끗한 토큰을 생성한다. 
 5. 공백(띄어쓰기)을 기준으로 단어를 분리하는 '단어 토큰화'는 가장 일반적인 토큰화 방식 중 하나이다.
8. 자연어 처리를 위한 딥러닝 모델에서 '임베딩(Embedding)'에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
1. 임베딩은 단어, 문장, 또는 문서를 컴퓨터가 이해하고 처리할 수 있는 연속적인 값의 밀집된 벡터(dense vector) 형태로 표현하는 기법이다.
 2. 단순한 원-핫 인코딩(One-Hot Encoding)과 달리, 임베딩은 단어 간의 의미적 또는 문법적 유사성을 벡터 공간상에서 거리로 표현할 수 있게 한다.
 3. Word2Vec, GloVe, FastText와 같은 모델은 단어 임베딩을 학습하는 대표적인 방법론이며, 미리 학습된 임베딩을 전이 학습(transfer learning)에 활용할 수 있다.
 4. 임베딩 벡터의 차원(dimension)은 일반적으로 매우 커서 모델의 계산 효율성을 크게 저하시키고 과적합(overfitting) 문제를 야기한다. 
 5. 임베딩은 텍스트 분류, 감성 분석, 기계 번역 등 다양한 NLP 태스크의 입력으로 사용되어 모델의 성능 향상에 기여한다.
9. 정성적(범주형) 데이터를 딥러닝 모델에 적용하기 위해 '더미 변수화(Dummy Variable Encoding)'를 수행할 때, 다음 설명 중 **가장 적절하지 않은** 것은 무엇인가?

1. 더미 변수화는 '성별', '혈액형', '도시'와 같이 순서나 크기 관계가 없는 범주형 데이터를 모델이 인식할 수 있는 숫자 형태로 변환하는 기법이다.
 2. 하나의 범주형 변수를 여러 개의 이진(0 또는 1) 변수로 확장하며, 각 이진 변수는 특정 범주의 존재 여부를 나타낸다.
 3. N개의 범주를 가진 변수를 더미 변수화할 때, 보통 N-1개의 더미 변수를 생성하여 다중공선성(multicollinearity) 문제를 피한다.
 4. 더미 변수화는 범주형 변수 내에 내재된 순서나 크기 관계를 명확하게 모델에 전달하여, 모델이 범주 간의 관계를 더 잘 학습할 수 있도록 돕는다. 
 5. 원-핫 인코딩(One-Hot Encoding)은 더미 변수화의 한 형태로, 각 범주에 해당하는 더미 변수를 1로, 나머지를 0으로 설정하는 방식이다.
10. 데이터 과학 및 머신러닝 분야에서 '데이터 포지셔닝(Data Positioning)'이 가지는 일반적인 의미와 중요성에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
1. 데이터 포지셔닝은 데이터를 다차원 공간(예: 특징 공간, 잠재 공간)의 특정 지점에 배치하여, 데이터 간의 관계나 유사성을 공간적 거리로 표현하는 과정이다.
 2. 효과적인 데이터 포지셔닝은 유사한 특성을 가진 데이터 포인트들을 서로 가깝게 위치시키고, 다른 특성을 가진 데이터 포인트들을 멀리 떨어뜨림으로써 데이터의 구조를 명확히 한다.
 3. 데이터 포지셔닝은 주로 시각화를 위한 목적이며, 머신러닝 모델의 학습 성능에는 직접적인 영향을 미치지 않는다. 
 4. 임베딩(Embedding)은 텍스트, 이미지, 사용자 등 복잡한 비정형 데이터를 저차원 벡터 공간에 포지셔닝하는 대표적인 기법 중 하나이다.
 5. 데이터 포지셔닝은 분류, 군집화, 추천 등 다양한 머신러닝 태스크에서 모델이 데이터의 패턴을 더 잘 인식하고 예측할 수 있도록 돕는 기반이 된다.
11. 딥러닝에서 '임베딩(Embedding)' 기법이 '데이터 포지셔닝(Data Positioning)'과 관련하여 가지는 중요한 특징에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?
1. 임베딩은 원시 데이터를 고차원 벡터 공간의 한 지점(point)으로 매핑하여, 각 데이터 항목을 해당 공간 내에 '위치(position)'시킨다.
 2. 의미적으로 유사하거나 관련성이 높은 데이터 항목(예: 단어, 이미지, 사용자)은 임베딩 공간 내에서 서로 가까운 거리에 '포지셔닝'되는 경향이 있다.
 3. 임베딩을 통한 데이터 포지셔닝은 단순히 데이터를 숫자로 변환하는 것을 넘어, 데이터 간의 복잡한 관계와 패턴을 벡터 공간에 내재화하는 과정이다.
 4. 임베딩 공간에서 데이터의 포지셔닝은 모델이 데이터를 이해하고 새로운 관계를 추론하는 데 중요한 기반이 되며, 이는 추천 시스템, 의미 검색 등에 활용된다.

5. 임베딩은 모든 데이터 항목을 무작위로 벡터 공간에 포지셔닝하여, 모델이 데이터의 분포에 독립적으로 학습하도록 돕는다. 

12. 다음 중 PyTorch의 `torch.nn.LSTM` 레이어를 구성할 때 사용되는 주요 옵션(인자)에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가?

1. `input_size`: 입력 시퀀스의 각 스텝에 대한 특성(feature)의 크기를 정의한다. (예: 단어 임베딩 벡터의 차원)
2. `hidden_size`: LSTM 은닉 상태(hidden state)의 크기이자 다음 스텝으로 전달되는 출력 벡터의 크기를 정의한다.
3. `num_layers`: 스택(stacked) LSTM의 깊이를 의미하며, 여러 개의 LSTM 레이어를 수직으로 쌓을 때 사용한다.
4. `batch_first`: 입력 텐서의 배치(batch) 차원이 첫 번째에 올지 ((batch, seq, feature)) 또는 두 번째에 올지 ((seq, batch, feature))를 결정하는 부울(boolean) 값이다.
5. `output_size`: LSTM 레이어의 최종 출력 시퀀스의 각 스텝에 대한 특성 크기를 정의하며, `hidden_size`와는 독립적으로 설정할 수 있다. 

13. 다음 중 PyTorch의 `torch.nn.LSTM` 레이어에 대한 입력 텐서의 일반적인 구조 (shape)와 그 의미에 대한 설명으로 **가장 적절하지 않은** 것은 무엇인가? (단, `batch_first=True`로 설정된 경우를 가정한다.)

1. 입력 텐서는 기본적으로 3차원 ((batch_size, sequence_length, input_size)) 형태를 가진다.
2. `batch_size`는 한 번에 처리되는 시퀀스의 개수를 나타내며, 모델 학습 시 병렬 처리를 위해 사용된다.
3. `sequence_length`는 각 시퀀스 내의 스텝(또는 토큰)의 개수를 나타내며, LSTM이 처리할 시간 스텝의 길이를 의미한다.
4. `input_size`는 각 개별 스텝(토큰)에 대한 특성(feature)의 차원을 나타내며, 보통 임베딩 벡터의 크기와 일치한다.
5. 초기 은닉 상태(initial hidden state)와 셀 상태(initial cell state)는 입력 텐서와 동일한 3차원 ((num_layers * num_directions, batch_size, hidden_size)) 구조를 가지며, 항상 사용자가 직접 초기화하여 제공해야 한다. 