

생성형 AI

Day 16

딥러닝 IV



목차

1. PyTorch 기초지식
2. Seq2Seq 모델
3. 오토인코더
4. 생성적 적대 신경망 (GAN)
5. 모델 최적화
6. 실습과제



PyTorch 기초지식

Pytorch?

- Facebook AI Research(FAIR)에서 개발한 오픈 소스 머신 러닝 라이브러리
- 동적 계산 그래프, 간편한 디버깅, 그리고 Pythonic한 코드 스타일이 특징

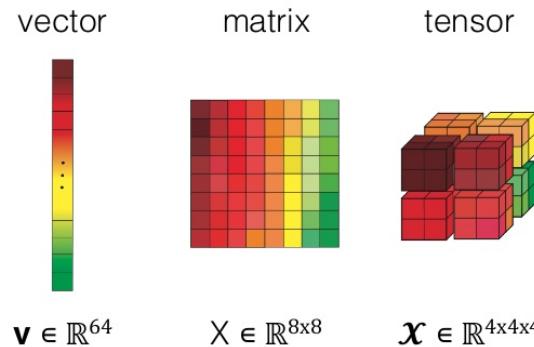


PyTorch 기초지식

텐서(Tensor)

- PyTorch에서 진행하는 대부분의 연산의 기본 데이터 구조
- 다차원 배열 또는 행렬과 유사
- GPU나 다른 연산 가속을 위한 특수한 하드웨어에서 실행할 수 있다는 점을 제외하면, NumPy의 ndarray와 매우 유사
- GPU를 활용한 고성능 연산이 가능

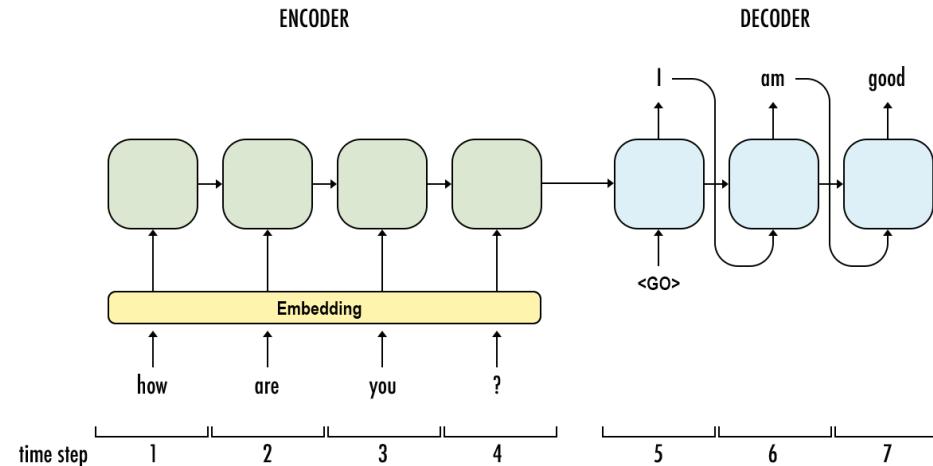
tensor = multidimensional array



Seq2Seq 모델

Seq2Seq 모델?

- 입력 시퀀스(Seq)를 출력 시퀀스(Seq)로 변환하는 데 사용되는 딥러닝 모델
- 주로 기계 번역, 텍스트 요약, 대화 시스템 등에서 사용
- 인코더(Encoder)와 디코더(Decoder)로 구성



Seq2Seq 모델

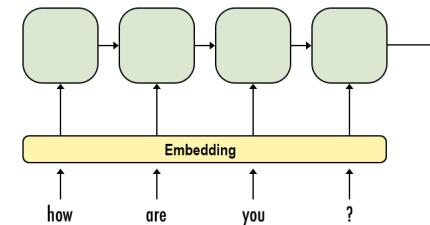
인코더(Encoder)

- 인코더는 입력 시퀀스를 고정된 길이의 벡터로 변환
- 보통 RNN, LSTM, GRU와 같은 순환 신경망으로 구성

인코더 작동원리

- 입력 시퀀스를 한 번에 한 요소씩 입력
- 각 요소를 처리하면서 은닉 상태 업데이트
- 마지막 요소가 처리되면 마지막 은닉 상태를 고정된 길이의 컨텍스트 벡터로 출력

ENCODER



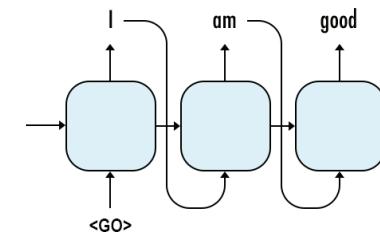
디코더(Decoder)

- 인코더에서 생성된 고정된 길이의 벡터를 입력으로 받아 출력 시퀀스를 생성
- RNN, LSTM, GRU와 같은 순환 신경망으로 구성

디코더 작동원리

- 인코더의 마지막 숨겨진 상태를 초기 하든 상태로 사용
- 시작 토큰을 입력으로 받아 다음 토큰을 예측
- 예측된 토큰은 다시 디코더의 입력으로 사용되어 다음 토큰을 예측하는 과정을 반복
- 훈련과정에서는 정답값을 입력으로 사용

DECODER



Seq2Seq 모델

교사강요(Teacher Forcing)

- 디코더가 이전 시간 단계에서 예측한 출력을 사용하지 않고, 실제 목표 시퀀스의 요소를 다음 입력으로 사용하는 방법
- 모델이 더 빠르고 안정적으로 수렴할 수 있도록 보조(강요)

Seq2Seq 장점

- 입력 시퀀스와 출력 시퀀스의 길이가 다를 수 있는 문제를 유연하게 해결 가능
- 입력 문장을 고정된 길이의 벡터로 인코딩하고, 디코딩하여 출력 문장을 생성할 수 있기 때문에 자연어 처리에 특화
- 기계 번역, 텍스트 요약, 대화 시스템, 음성 인식 등 응용분야가 넓음

Seq2Seq 단점

- 마지막 숨겨진 상태를 고정된 길이의 벡터로 사용하기 때문에, 긴 시퀀스를 처리할 때 중요한 정보가 손실 가능
- 학습에 많은 데이터와 계산 자원을 필요
- 순차적으로 데이터를 처리하기 때문에 병렬화가 어려움
- 이전 출력에 의존하여 다음 출력을 생성하기 때문에 초기 단계에서의 오류가 이후 단계로 전파되어 전체 시퀀스 품질에 영향

오토인코더

오토인코더?

- 데이터의 효율적인 표현을 학습하는 비지도 학습 인공신경망 모델
- 데이터 압축, 차원 축소, 노이즈 제거, 이상 탐지 등에 사용
- 입력 데이터를 압축하여 잠재 공간(latent space) 표현으로 변환한 후, 다시 원래 데이터로 복원하는 과정을 통해 학습

오토인코더의 구조

- 인코더

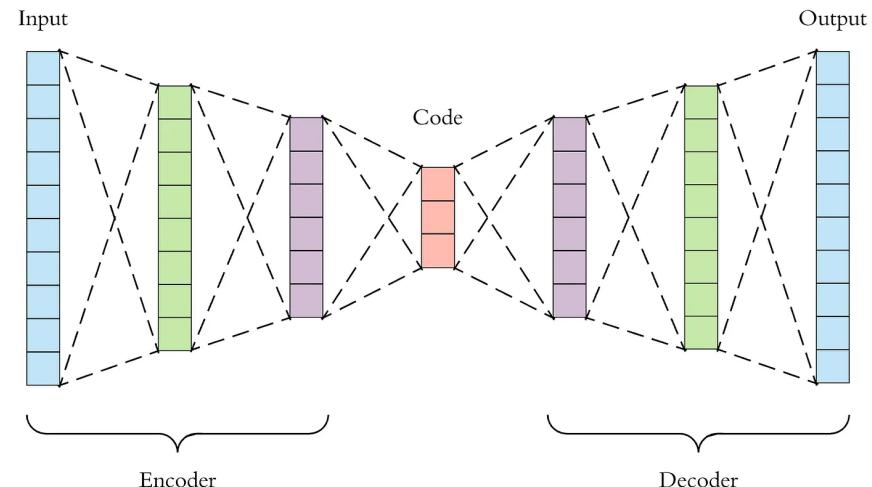
입력 데이터를 저차원 잠재 공간 벡터로 변환

입력층에서 잠재 공간 표현까지의 여러 층의 신경망으로 구성

- 디코더

잠재 공간 벡터를 원래 데이터로 복원

잠재 공간 표현에서 출력층까지의 여러 층의 신경망으로 구성



오토인코더

오토인코더의 유형

기본 오토인코더 (Basic Autoencoder)

- 인코더와 디코더로 구성된 가장 기본적인 형태의 오토인코더

딥 오토인코더 (Deep Autoencoder)

- 인코더와 디코더가 여러 개의 은닉층을 가지는 심층 신경망 구조로 구성

희소 오토인코더 (Sparse Autoencoder)

- 잠재 공간 벡터 z 의 많은 요소가 0이 되도록 제약을 가하여, 희소한 표현을 학습
- 데이터의 중요한 특징을 더 잘 잡아냄

변이형 오토인코더 (Variational Autoencoder, VAE)

- 잠재 공간을 확률 분포로 모델링하여, 데이터의 잠재 구조를 더 잘 파악
- 생성 모델로 사용되어 새로운 데이터를 샘플링

잡음 제거 오토인코더 (Denoising Autoencoder)

- 입력 데이터에 노이즈를 추가한 후, 이를 원래 데이터로 복원하도록 학습
- 데이터의 노이즈 제거와 더 강인한 특징 학습

오토인코더

오토인코더의 응용

데이터 압축(Data Compression)

- 입력 데이터를 저차원 임시 공간 벡터로 압축하여 저장 공간을 절약

차원 축소 (Dimensionality Reduction)

- 고차원 데이터를 저차원으로 축소하여 시각화하거나, 다른 머신러닝 알고리즘의 성능 향상에 사용

노이즈 제거 (Noise Reduction)

- 잡음 제거 오토인코더를 사용하여, 입력 데이터의 노이즈를 제거하고 깨끗한 데이터를 복원

이상 탐지 (Anomaly Detection)

- 정상 데이터로 학습한 오토인코더가 이상 데이터를 복원하지 못하는 특성을 이용하여, 이상 데이터 탐지

생성 모델 (Generative Model)

- 변이형 오토인코더를 사용하여, 새로운 데이터를 생성 가능

오토인코더

오토인코더 장점

- 레이블이 없는 데이터로 학습 가능
- 데이터 압축, 노이즈 제거, 이상 탐지 등 다양한 응용 분야에 사용 가능
- 학습된 잠재 공간 벡터를 분석하여, 데이터의 중요한 특징을 파악 가능

오토인코더 단점

- 복원된 데이터가 원본 데이터와 정확히 일치하지 않을 수 있음
- 매우 복잡한 데이터에 대해서는 처리가 어렵기 때문에 성능이 제한될 수 있음
- 적절한 정규화가 없으면 과적합이 발생할 수 있음

생성적 적대 신경망 (Generative Adversarial Networks, GAN)

생성적 적대 신경망 (GAN)?

- 생성적 적대 신경망은 두 개의 신경망, 생성자(Generator)와 판별자(Discriminator)가 경쟁적으로 학습하는 방식으로, 현실과 유사한 데이터를 생성하는 모델
- 이미지 생성, 데이터 증강, 비디오 생성 등 다양한 응용 분야에서 사용

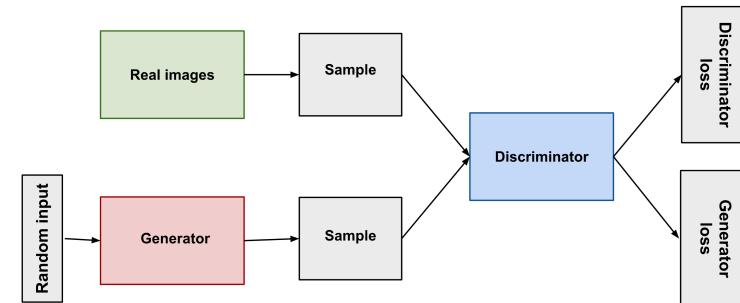
GAN 구성 요소

생성자 (Generator)

- 임의의 노이즈 벡터를 받아들여 현실감 있는 데이터를 생성하는 역할
- 입력 노이즈 벡터를 고차원 데이터로 변환하는 신경망
- 생성된 데이터를 실제 데이터처럼 보이게 만드는 것이 목표

판별자 (Discriminator)

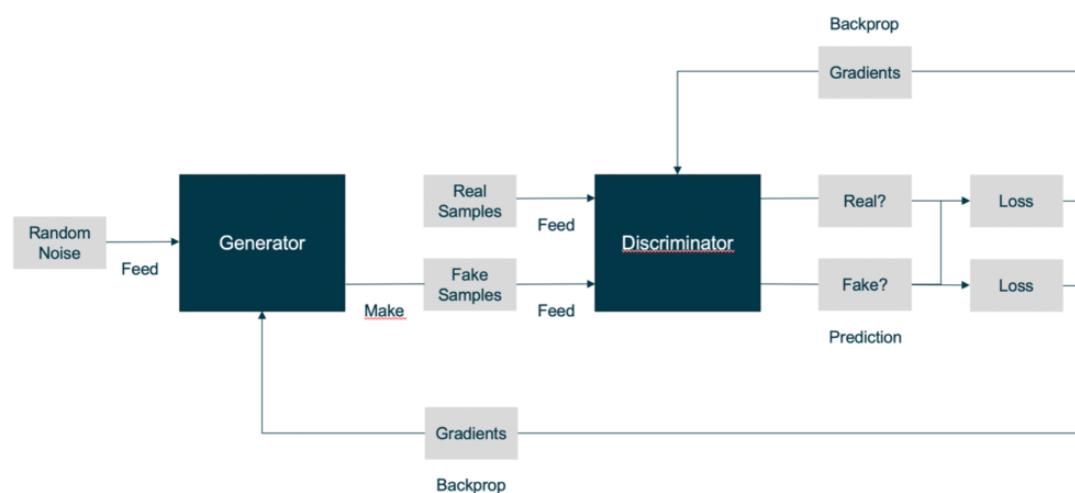
- 입력된 데이터가 실제 데이터인지 생성된 데이터인지 구분하는 역할
- 입력된 데이터를 분류하는 신경망
- 실제 데이터와 생성된 데이터를 정확하게 구분하는 것이 목표



생성적 적대 신경망 (Generative Adversarial Networks, GAN)

GAN 학습 절차

- 생성자는 노이즈 벡터를 통해 가짜 데이터를 생성
- 판별자는 실제 데이터와 생성된 데이터를 입력받아 구분
- 판별자의 손실을 최소화하는 방향으로 판별자의 가중치를 업데이트
- 생성자의 손실을 최소화하는 방향으로 생성자의 가중치를 업데이트



<https://www.statworx.com/en/content-hub/blog/generative-adversarial-networks-how-data-can-be-generated-with-neural-networks/>

생성적 적대 신경망 (Generative Adversarial Networks, GAN)

GAN 변형모델

DCGAN (Deep Convolutional GAN)

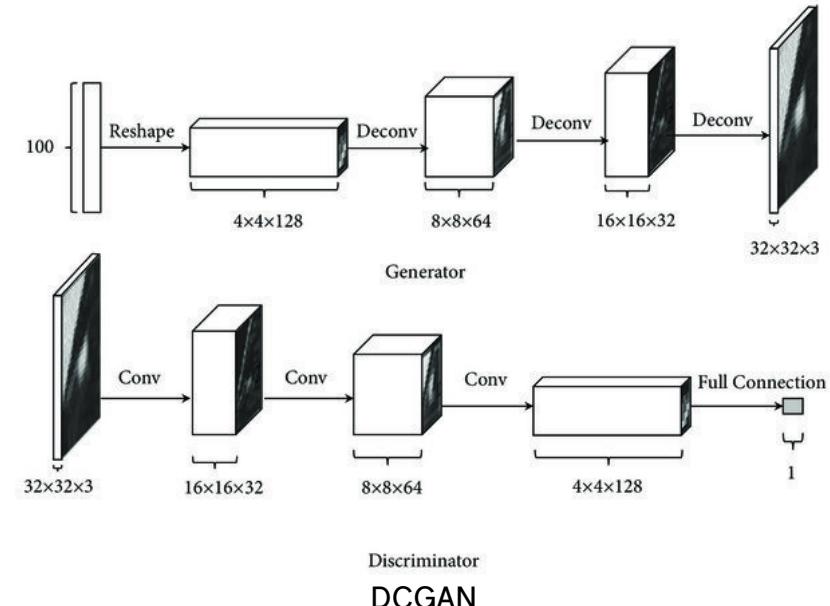
- 합성곱 신경망(CNN)을 사용하여 이미지를 생성
- 보다 높은 해상도의 이미지를 생성 가능

CycleGAN

- 두 가지 도메인 간의 이미지 변환을 수행
- 여름을 겨울로 변환, 말 이미지를 얼룩말 이미지로 변환 등

StyleGAN

- 고해상도의 이미지를 생성하며, 이미지의 스타일을 조절 가능
- 이미지의 특정 스타일 요소를 제어할 수 있음



생성적 적대 신경망 (Generative Adversarial Networks, GAN)

GAN의 응용분야

- 이미지/비디오 생성, 데이터 증강
- 텍스트 생성, 이미지 변환

GAN 장점

- 현실과 유사한 데이터를 생성할 수 있어 다양한 응용 분야에 활용 가능
- GAN의 기본 구조를 바탕으로 여러 변형 모델이 제안되고 있는 만큼, 특정 문제에 더 잘 맞는 모델을 개발/선택할 수 있음
- 학습된 GAN 모델은 무한히 새로운 데이터를 생성 가능

GAN 단점

- GAN의 학습 과정은 매우 불안정할 수 있으며, 생성자와 판별자가 균형을 이루지 못하면 학습이 실패할 수 있음
- 생성자가 제한된 종류의 데이터만 생성하게 되는 문제가 발생할 수 있음
- GAN의 학습에는 많은 계산 자원이 필요하며, 학습 시간이 오래 걸릴 수 있음

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

1. 학습률 (Learning Rate)

- 경사 하강법에서 가중치가 업데이트되는 크기를 조정하는 하이퍼파라미터

영향

- 높은 학습률: 학습 속도가 빨라지지만, 손실 함수가 최솟값에 도달하지 못하고 발산할 수 있음
- 낮은 학습률: 학습 속도가 느려지지만, 손실 함수가 더 안정적으로 최솟값에 도달할 수 있음

설정 방법

- 학습률을 실험적으로 설정하고, 모델의 성능을 기준으로 조정
- 일반적으로 0.001 또는 0.01부터 시작
- 학습률 스케줄링 기법을 사용하여 학습 진행에 따라 학습률을 조정 가능

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

2. 배치크기 (Batch Size)

- 배치 크기는 한 번의 업데이트에 사용되는 훈련 샘플의 수를 의미

영향

- 작은 배치 크기: 더 빈번하게 가중치가 업데이트되어 학습 속도가 빨라질 수 있지만, 노이즈가 증가할 위험 존재
- 큰 배치 크기: 더 안정적인 가중치 업데이트가 가능하지만, 학습 속도가 느려짐

설정 방법

- 일반적으로 32, 64, 128, 256과 같은 2의 제곱수로 설정
- 메모리 사용량과 학습 속도를 고려하여 적절한 배치 크기를 선택

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

3. 드롭아웃 비율 (Dropout Rate)

- 드롭아웃 비율은 학습 중 각 학습 단계에서 무작위로 비활성화할 뉴런의 비율을 의미

영향

- 높은 드롭아웃 비율: 과적합을 방지할 수 있지만, 학습 속도가 느려지고 모델 성능이 떨어질 수 있음
- 낮은 드롭아웃 비율: 모델이 과적합할 가능성이 증가

설정 방법

- 일반적으로 0.2에서 0.5 사이의 값을 사용
- 드롭아웃 비율을 실험적으로 설정하고, 모델의 성능을 기준으로 조정

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

4. 정규화 방법 (L1, L2)

- 정규화는 손실 함수에 정규화 항을 추가하여 모델의 복잡성을 제어하고 과적합을 방지하는 방법

L1 정규화

- 가중치의 절대값 합을 최소화하는 정규화 방법
- 가중치를 희소하게 만들어 일부 가중치를 0으로 만듦

L2 정규화

- 가중치의 제곱합을 최소화하는 정규화 방법
- 가중치의 크기를 줄여 과적합을 방지

설정 방법

- λ 값을 실험적으로 설정하고, 모델의 성능을 기준으로 조정
- 일반적으로 L2 정규화가 더 자주 사용

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

5. 네트워크 깊이와 폭

- 네트워크 깊이는 신경망의 층(layer)의 수를 의미하고, 네트워크 폭은 각 층의 뉴런 수를 의미

영향

- 깊은 네트워크: 더 복잡한 패턴을 학습할 수 있지만, 기울기 소실/폭발 문제와 과적합의 위험이 증가
- 넓은 네트워크: 더 많은 특징을 학습할 수 있지만, 많은 계산 자원을 요구

설정 방법

- 문제의 복잡성과 데이터셋의 크기에 따라 적절한 깊이와 폭을 선택
- 과적합을 방지하기 위해 드롭아웃, 정규화와 같은 기법을 함께 사용

딥러닝 모델 최적화

딥러닝 모델 최적화를 위한 하이퍼 파라미터

6. 에포크

- 에포크(Epoch)는 딥러닝 모델 학습에서 전체 데이터셋을 한 번 완전히 학습하는 주기를 의미
- 에포크 수는 모델이 전체 데이터셋을 몇 번 반복해서 학습했는지를 의미

학습 과정에서의 역할

- 학습 과정: 에포크 수가 증가할수록 모델은 데이터셋에 있는 모든 샘플을 여러 번 학습하고, 이를 통해 모델의 가중치가 점진적으로 조정되고, 최적의 가중치를 찾는 과정이 이루어짐

적절한 에포크 수 선택의 중요성

- 과소적합: 에포크 수가 너무 적으면 모델이 데이터셋을 충분히 학습하지 못해 학습이 덜 된 상태가 될 수 있음
- 과적합: 에포크 수가 너무 많으면 모델이 데이터셋을 과도하게 학습하여 과적합 상태가 될 수 있음
- 적절한 에포크 수: 조기 종료(Early Stopping)와 같은 기법을 사용하여 적절한 에포크 수를 설정

이론 실습

colab주소

PyTorch: [https://colab.research.google.com/drive/1V2G7QWOJrw5erjrjJuPpqvEg4s2SK7cu?](https://colab.research.google.com/drive/1V2G7QWOJrw5erjrjJuPpqvEg4s2SK7cu?usp=sharing)
usp=sharing

Autoencoder: [https://colab.research.google.com/drive/1qeQRip5K9KkYgRtKmWnGVjYx53FjSc44?](https://colab.research.google.com/drive/1qeQRip5K9KkYgRtKmWnGVjYx53FjSc44?usp=sharing)
usp=sharing

실습 과제 (LSTM)

1. PyTorch의 기본 개념을 이해하고, 텐서 연산, 자동 미분, 간단한 신경망 모델을 구현해보기
2. 간단한 Seq2Seq 모델 구현해보기
3. 실제 데이터셋 사용: 실제 데이터셋을 사용해보기(<https://www.gutenberg.org>, <https://storage.googleapis.com>)
4. 생성한 모델들에 다양한 학습률, 배치크기, 드롭아웃 비율들을 조절해서 성능 변화 비교해보기

실습 진행