8주차

순환신경망

▼ 1. 순차 데이터와 순환 신경망

목적

이전에 훈련한 합성곱 신경망(CNN) 모델을 불러와 가중치 및 특성 맵을 시각화하고, 케라스 함수형 API를 활용하여 모델을 구성하는 방법을 알아본다.

데이터 준비

- 모델 불러오기: keras.models.load_model() 사용.
- 모델 구조 확인: model.layers를 통해 층 구성 확인.
- 가중치 조사:
  - 각 층의 가중치 정보는 weights 속성에 저장됨.
  - numpy()로 변환 후 평균 및 표준편차를 계산.
  - 예시:
    - 첫 번째 합성곱 층의 가중치 크기: (3, 3, 1, 32) (커널: 3x3, 채널: 1, 필터: 32개).
    - 가중치 평균과 표준편차: 0.019, 0.23

가중치 시각화

- 가중치 분포 히스토그램
  - 방법: plt.hist()를 사용하여 가중치의 분포를 시각화.
  - 결과: 가중치 값은 0을 중심으로 종 모양의 분포를 가짐.

커널 시각화

- 방법: 32개의 커널을 16개씩 두 줄로 나열해 출력.
- 결과: 특정 패턴이 있는 커널을 확인.

훈련되지 않은 모델 가중치와 비교

- 초기화된 모델 가중치
  - 평균은 0.001, 표준편차는 0.078.
  - 균등 분포를 따르며, 가중치가 밋밋한 분포를 보임.
- 결과 비교:
  - 훈련된 모델은 특정 패턴을 학습.
  - 초기화된 모델은 랜덤한 분포를 가짐.

함수형 API를 사용한 모델 구성

- Sequential 모델의 한계: 복잡한 모델 구조를 표현하기 어려움.
- 함수형 API:
  - keras.Model 클래스를 사용.
  - keras.Input()으로 입력 정의 후 층을 차례로 연결.

```
python
코드 복사
inputs = keras.Input(shape=(784,))
hidden = keras.layers.Dense(100, activation='sigmoid')(i
nputs)
outputs = keras.layers.Dense(10, activation='softmax')(h
idden)
model = keras.Model(inputs, outputs)
```

특성 맵 시각화

- 첫 번째 층 특성 맵
  - 방법:

1. 모델의 입력과 첫 번째 합성곱 층의 출력을 연결해 서브모델 생성.
2. 훈련 데이터 샘플을 표준화하여 입력 후 predict()로 특성 맵 출력.
3. plt.subplots()로 시각화.

- 결과:
  - 32개의 28x28 크기 특성 맵 생성.
  - 필터에 따라 입력 이미지의 특정 부분이 강조됨.

- 두 번째 층 특성 맵
  - 방법: 동일한 방식으로 두 번째 합성곱 층의 특성 맵 시각화.
  - 결과:
    - 64개의 특성 맵 생성.
    - 시각적으로 추상적인 정보를 학습하며, 첫 번째 층보다 복잡한 특성을 나타냄.

결론

- 첫 번째 합성곱 층은 이미지의 시각적 특징을 감지.
- 뒤쪽 층은 앞쪽 층에서 감지된 정보를 추상화하여 학습.
- 함수형 API는 복잡한 모델 구성에 유용

▼ 2. 순환신경망으로 IMDB 리뷰 분류하기

목적

- 딥러닝으로 댓글의 긍정/부정을 분석.
- 순차 데이터와 순환 신경망(RNN)의 개념을 학습하고, IMDB 리뷰 데이터를 학습하여 분류.

순차 데이터

- 정의: 순서가 중요한 텍스트, 시계열 데이터 등.
- 특징: 단어 순서가 중요한 텍스트 데이터처럼 입력의 순서를 유지해야 함.
- 피드포워드 신경망: 데이터를 처리한 후 샘플을 버림.

- 순환 신경망 (RNN):
    - 이전 샘플의 정보를 기억하고 재사용.
    - 데이터가 순환하는 구조로 샘플을 처리하며 타임스텝 단위로 계산.

순환 신경망 (RNN)

- 특징
    - 출력이 다시 입력으로 전달되는 고리 구조.
    - 은닉 상태로 이전 타임스텝 정보를 유지.
    - 활성화 함수: tanh (출력 범위: -1 ~ 1).
- 단점: 오래된 정보는 희미해짐.

IMDB 리뷰 분류

- 데이터 준비
    - IMDB 데이터셋: 긍정(1) / 부정(0) 리뷰로 이루어진 50,000개의 샘플.
    - 데이터 전처리:
        - 자주 등장하는 단어 500개만 사용.
        - 정수로 변환된 데이터 로드: imdb.load_data(num_words=500).
        - 훈련 데이터와 테스트 데이터: 25,000개씩 분리.
        - 훈련 데이터의 20%를 검증 세트로 분리.

데이터 분석 및 패딩

- 리뷰 길이 분석:
    - 리뷰 길이는 대부분 300 미만.
    - 리뷰의 길이를 100으로 통일하기 위해 pad_sequences() 사용.
    - 짧은 리뷰: 앞부분에 0으로 패딩.
    - 긴 리뷰: 뒤를 자름.
- 결과: (20000, 100) 크기의 2차원 배열 생성.

원-핫 인코딩

- 필요성: 정수 데이터가 큰 활성화 출력을 만드는 문제 해결.
- 방법:
  - to_categorical() 사용.
  - 각 단어를 고유한 500차원의 배열로 변환.
- 결과: (20000, 100, 500) 크기의 배열.

순환 신경망 훈련

- 구성:
  - SimpleRNN 층: 뉴런 8개, 입력 크기 (100, 500).
  - 출력층: 이진 분류를 위해 1개의 뉴런과 sigmoid 활성화 함수.
- 훈련 설정:
  - 학습률 0.0001로 조정된 RMSprop 사용.
  - 에포크 100회, 배치 크기 64.
  - 체크포인트 및 조기 종료 설정.
- 결과:
  - 검증 세트 정확도 약 79%.
  - 손실 변화 그래프에서 40번째 에포크 이후 성능 둔화 확인.

단어 임베딩

- 필요성: 원-핫 인코딩의 차원 폭발 문제 해결.
- 임베딩:
  - 각 단어를 고정 크기의 실수 벡터로 변환.
  - Embedding 클래스:
    - input_dim=500: 어휘 사전 크기.
    - output_dim=16: 임베딩 벡터 크기.
    - input_length=100: 입력 시퀀스 길이.
- 모델 구성:

    - Embedding(500, 16, input_length=100).
    - SimpleRNN(8) + Dense(1, activation='sigmoid').

결론

- 순환 신경망: 이전 데이터를 기억하여 순차 데이터를 처리.
- IMDB 리뷰 분류:
    - 순환 신경망으로 긍정/부정 분류 가능.
    - 단어 임베딩으로 성능 향상 가능.
- 활용: 텍스트, 음성, 시계열 데이터 등 순차 데이터 분석에 효과적.

▼ 3. LSTM과 GRU셀

- 문제점: 기본 순환층은 긴 시퀀스를 학습하기 어렵고, 멀리 떨어진 단어 정보를 학습하는 데 한계가 있음.
- 해결 방법: LSTM(Long Short-Term Memory)과 GRU(Gated Recurrent Unit) 셀을 이용한 고급 순환층 활용.
- 목표: LSTM과 GRU를 활용한 다양한 순환 신경망 모델을 구현하고 학습 성능을 비교.

훈련 데이터 준비 및 패딩

1. 데이터 로드 및 분리:
    - IMDB 리뷰 데이터셋 사용.
    - 500개의 단어로 제한 (num_words=500).
    - 훈련 데이터와 검증 데이터로 분리.
2. 패딩 처리:
    - pad_sequences()를 사용해 시퀀스 길이를 100으로 통일.
    - 짧은 리뷰는 앞에서부터 0으로 패딩.

LSTM

- LSTM 셀 기반 순환층
  - 모델 구성:
    - Embedding: 어휘 크기(500), 임베딩 벡터 크기(16), 입력 길이(100).
    - LSTM: 8개의 뉴런.
    - Dense: 1개의 뉴런과 sigmoid 활성화 함수.
  - 훈련 설정:
    - 옵티마이저: RMSprop(학습률 0.0001).
    - 에포크: 100, 배치 크기: 64.
    - 체크포인트 및 조기 종료 사용.
  - 결과:
    - 훈련 및 검증 손실 그래프에서 과대적합이 억제되면서 학습이 잘 진행됨을 확인.

드롭아웃 적용

- 설정:
  - dropout=0.3: 입력에 드롭아웃 적용.
  - recurrent_dropout: 순환 은닉 상태에 드롭아웃 적용(GPU 사용 불가로 제외).
- 결과:
  - 드롭아웃 적용 후 훈련 손실과 검증 손실 차이가 줄어들며 안정적 학습 확인.

LSTM 층 쌓기

- 구성:
  - return_sequences=True: 모든 타임스텝의 은닉 상태를 출력.
  - 두 번째 LSTM 층은 마지막 타임스텝의 은닉 상태만 출력.
- 결과:
  - 층을 쌓은 모델이 과대적합을 잘 억제하며 손실 감소.

GRU

- 특징:
  - LSTM의 간소화된 버전.
  - 셀 상태를 계산하지 않고 은닉 상태만 유지.
  - 가중치가 적어 계산량이 적음.
- 모델 구성:
  - GRU: 8개의 뉴런.
  - Dense: 1개의 뉴런과 sigmoid 활성화 함수.
- 결과:
  - 드롭아웃 없이도 훈련 손실과 검증 손실이 잘 수렴.

결과 비교

1. LSTM:
   - 긴 시퀀스 학습에 강점.
   - 과대적합 방지에 효과적.
   - 드롭아웃 및 층 쌓기를 통해 성능 향상.
2. GRU:
   - 계산량이 적고 빠른 학습 가능.
   - LSTM에 근접한 성능을 보이며 간결한 구조로 효율적 학습.

결론

- LSTM과 GRU는 순차 데이터 분석에 적합한 강력한 순환층.
- 과대적합 방지를 위해 드롭아웃 및 층 쌓기를 적용하면 더 나은 성능을 얻을 수 있음.
- GRU는 계산 효율성이 뛰어나며 LSTM과 유사한 성능을 제공.