|  |  |  |  |
| --- | --- | --- | --- |
| **HW#03** | **딥러닝 과제보고서** | **학번: 2018741035** | **이름: 한준호** |
| #1. 과제 목표 | | | |
| Recurrent Neural Network (RNN)을 이용하여 주가를 예측하는 모델을 만들어본다. 만든 주가 예측 RNN 모델이 Test Set에서 높은 성능을 보여줄 수 있도록 Data Augmentation, Validation Set 생성, RNN 모델의 레이어 구조 변경, 학습 파라미터 변경 등을 고려하여 RNN 모델을 개선하여 Test Set에 대해서 일반화된 모델을 만들어본다. | | | |
| #2. 코드 수행 결과 (사진 첨부)  1. Data Import    텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명  KODEX 인버스(종목코드 114800) 주식 2010년 1월 1일부터 2015년 12월 31일까지 주가를 학습데이터로 활용하고 2016년 1월 1일부터 2020년 5월 22일까지 주가를 테스트데이터로 활용하여 데이터를 불러왔다.    불러온 training 데이터와 test 데이터를 학습을 위하여 0에서 1 사이의 값으로 Scaling을 진행하였고, 데이터를 일정한 길이인 60인 sequence로 나누어 주었다.  2. RNN Layer  텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진  자동 생성된 설명  RNN 모델을 만들기 위해서 GRU를 사용했다. 실습에서는 LSTM을 사용했지만, LSTM에서는 출력, 입력, 삭제 게이트라는 3개의 게이트가 존재하지만, GRU에서는 업데이트 게이트와 리셋 게이트 두 가지 게이트만이 존재한다. 평가에서 GRU는 LSTM과 비슷한 성능을 보인다고 알려져 있지만 GRU는 LSTM보다 학습 속도가 빠르다고 알려져 있어 GRU를 사용했다.  사용하는 주식 Data는 일정 시간 간격 순차적으로 배치된 시계열 데이터이므로 Data의 입력 순서가 중요하기 때문에, 앞선 진행한 실습과 같이 Dropout을 각 RNN 층마다 해주지 않았고, 모든 층을 쌓고 난 후, 비교적 낮은 drop-out rate로 한번만 Dropout을 진행하였다.  안정적인 학습을 위해 dense을 두번으로 나누어 학습을 진행하였다.  4. 학습 파라미터  텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진  자동 생성된 설명  학습을 여러 번 진행하면서 학습이 완료된 모델의 성능을 확인하고, 이를 비교하면서 학습 파라미터를 설정해주었다. 옵티마이저(Optimizer)를 RMSprop와 Momentum 두 가지를 합친 듯한 방법인 ‘Adam’으로 설정하였고, 손실함수(loss)는 시계열 데이터 학습에 비교적 좋은 퍼포먼스를 내는 MSE와 MAE의 가장 좋은 특성을 결합한 Huber loss를 사용하였다. 보통 적은 ‘batch size’, 큰 ‘epochs’로 설정하는 것이 학습에 좋다고 알려져있기 때문에 ‘batch\_size’는 64, ‘epochs’는 250번으로 설정해주었다.  5. 결과  텍스트, 영수증, 폰트, 라인이(가) 표시된 사진  자동 생성된 설명  학습을 진행하면서 비교적 낮은 loss가 출력되는 것을 확인했다.  텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명  training data의 Prediction 결과와 test data의 prediction 결과를 확인해본 결과, 평균 training error: 14.886941162622636, 평균 test error: 12.58853765778941로 학습한 모델의 성능을 확인했다.  텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진  자동 생성된 설명  학습된 모델의 Prediction 결과를 그래프로 확인해본 결과, 실제 그래프인 파란선을 예측 그래프인 빨간색이 거의 제대로 따라가는 것을 확인했다. | | | |
| #3. 고찰  •주식 Data는 시계열 데이터이므로 시간에 흐름에 따른 데이터의 입력 순서가 중요하기 때문에 저번 CNN 과제와 같이 데이터를 임의로 조작할 수가 없어서 Data Augmentation을 진행하지 않았다.  •처음에 model.fit( )을 사용해 데이터를 학습할 때, model.fit( )은 기본으로 ‘shuffle=True’로 설정되어 epoch마다 Data를 랜덤으로 섞어주는데, Sequence Data는 Data의 입력 순서가 중요하기 때문에 ‘shuffle=False’로 설정하여 Data가 자동으로 섞이지 않도록 해주었고, 이에 따른 성능 향상을 기대했지만 기대와 달리 성능이 눈에 띄게 좋아지지 않아 최종 학습을 진행할 때는 shuffle 파라미터를 제거하고 학습을 진행하였다.  • 학습된 모델의 training data의 Prediction 결과와 test data의 prediction 결과를 확인해보는 과정에서, 평균 error로 성능을 비교해서 그런지 같은 모델과 같은 학습 파라미터로 학습을 진행해도 비교적 차이가 큰 다른 결과가 나와 객관적인 성능 비교가 어려웠다. | | | |
| #4. 결론  학습 모델의 목표에 맞는 RNN Layer의 구조 설정, 적절한 학습 파라미터를 설정을 통해 RNN 모델을 개선하여 모델의 성능을 높일 수 있었다. 하지만 적절한 RNN Layer의 구조와, 적절한 학습 파라미터를 설정 하는 것은 쉽지 않았다.  주식 Data와 같이 일정 시간 간격 순차적으로 배치된 시계열 데이터는 데이터의 입력 순서가 중요하기 때문에, 임의로 Data를 조작해 Data Augmentation을 진행하기는 어려웠다. 따라서 Data Augmentation를 통해 모델의 성능을 올리기는 어려웠다.  같은 모델과 같은 학습 파라미터로 모델의 학습을 진행하여도 학습을 진행할 때마다 다른 성능의 결과가 나왔기 때문에 모델 Layer 변경과 학습 파라미터 변경에 따른 모델의 성능을 비교하기 어려웠다. | | | |