LSTM을 활용한 기술적 주가 분석 방법

201502127 허세훈
201502046 김백준
201602055 이현경

1. Introduction

주식 가격을 예측하는 것은 주식 투자자들이 지속적으로 높은 수익을 얻을 수 있게 할 수 있습니다. 한 학기 동안 본 수업에서는 데이터를 분석하고 미래를 예측하는 방법을 배웠기 때문에 이를 활용한다면 주식 가격 또한 예측 가능할 것이라는 생각하였습니다.

주식 가격을 예측하는 일반적인 방법에는 기본적 주가 분석과 기술적 주가 분석이 있습니다. 기본적 주가 분석은 재무제표, 환율, 해당 기업의 산업 자료, 통계자료 등을 통해 해당 기업에 투자하기 위한 중요한 정보를 얻고 이를 통한 투자 전략을 세우는 방법입니다. 하지만 이 방법은 기업 분석을 위한 회계 및 금융과 같은 전문적인 지식이 필요하므로 전문가가 아니면 이를 통해 주식 가격을 예측하는 것은 쉽지 않습니다.

반면에 기술적 주가 분석은 과거 주식의 가격이나 거래량 같은 자료를 이용하여 주가 변화의 추세를 발견해내어 미래의 주가를 예측하는 방법입니다. 이 방법은 과거의 데이터를 통해 변화의 추세를 발견하고 이를 통해 통찰력을 얻는 방법이므로 그간 배웠던 데이터 분석 및 모델링 방식을 적용하는데 적합합니다. 따라서 저희는 기술적 주가 분석을 통해 미래의 주식 가격을 예측하였습니다.

본 문서는 기술적 주가 분석을 하기 위한 문제 정의 과정, 데이터 수집 과정, 모델링 과정, 모델링의 결과 순서로 구성되어 있습니다.

2. Problem Definition

기술적 주가 분석은 주가는 항상 반복하여 변화하는 속성이 있고 이 추세를 파악할 수 있다면 현재 기업의 재무상태나 전제 시장의 상태와는 무관하게 변동하는 주가를 예측할

수 있다는 전제를 가지고 있습니다. 이 전제를 검증하기 위해 저희는 실제 기업의 과거 주가 데이터를 통해 모델을 학습시키고 미래의 주식 가격을 예측하여 그 성능을 측정하기로 결정했습니다. 또한 이번 프로젝트에서는 방대한 양의 데이터를 모두 수집하기에 어려움이 있어 하루 동안의 주가 변동을 모두 예측하는 것은 무리가 있다고 판단하였습니다. 따라서 이번 프로젝트에서는 다음 날의 주식 가격을 예측하여 주식 시장 개시 이전에 시장에 참여할지 여부에 대한 통찰을 제공하는 것을 목표로 하였습니다.

한편 주식 시장이 개시되면 주가는 5초마다 변화하기 때문에 하루 단위의 주가를 하나의 숫자로 나타내기는 어렵습니다. 따라서 하루 단위로 주가를 분석하기 위해서는 주식 시세를 나타내는 지표를 이용해야 합니다. 이 때, 사용될 수 있는 지표는 시가, 고가, 저가, 종가, 거래량이 있습니다. 시가는 그 날 최초로 체결된 거래의 가격을 의미하고 종가는 마지막으로 체결된 거래 가격을 의미합니다. 고가와 저가는 당일 거래된 가격 중 최고가와 최저가를 의미하고 거래량은 당일 증권시장에서 거래된 주식의 수를 말합니다. 이 때, 고가와 저가는 하루 동안의 주가 변동 량을 나타내는 지표이기 때문에 이번 프로젝트에서는 다루지 않기로 결정하였습니다. 반면 시가와 종가를 비교하는 것과 거래량을 예상해보는 것은 주식 시장 개시 이전에 주식 매도와 매수에 대한 상당한 통찰을 줄 수 있기 때문에 시가, 종가, 거래량은 예측할 가치가 있는 지표들이라고 생각하였습니다. 따라서 이번 프로젝트에서는 시가, 종가, 거래량을 예측하기로 결정하였습니다.

일별시세

날짜	종가	전일비	시가	고가	저가	거래량
2020.06.05	55,500	▲ 900	54,400	55,900	54,000	22,399,065
2020.06.04	54,600	▲ 100	55,800	57,000	54,600	40,220,334
2020.06.03	54,500	▲ 3,100	51,800	55,000	51,700	49,257,814
2020.06.02	51,400	▲ 200	51,000	51,500	50,800	14,247,933
2020.06.01	51,200	▲ 500	50,800	51,200	50,600	16,949,183
2020.05.29	50,700	▲ 300	50,000	50,700	49,700	27,596,961
2020.05.28	50,400	▲ 500	51,100	51,200	49,900	31,309,318
2020.05.27	49,900	▲ 650	48,950	50,000	48,800	19,548,479
2020.05.26	49,250	▲ 400	48,700	49,450	48,600	15,127,490
2020.05.25	48,850	▲ 100	48,750	48,900	48,450	14,337,913

출처 : https://finance.naver.com/item/sise_day.nhn?code=005930&page=1

시가, 종가, 거래량은 어느 정도의 연관성을 가지기는 하지만 그들의 관계를 통해 서로가 어떻게 변화하는지를 알 수는 없습니다. 예를 들어, 종가와 거래량의 관계를 모델링하고 이를 통해 종가나 거래량을 예측하는 것은 둘 사이의 연관성이 있다고 생각할 수 없기에 불가능합니다. 또한 과거의 데이터를 통해 미래의 데이터를 예측하는 기술적 주가 분석 방법론을 따르기 위해 각 지표의 연관성에 집중하기 보다는 각 지표들이 시간에 따라 어떻게 변화하는지를 살펴보는 것이 좋겠다고 판단하였습니다. 따라서 저희는 당일의 주식 시세 지표를 Label로 정의하고 해당 지표의 과거 데이터를 Variable로 설정하여 최종적으로 회귀 문제로 문제를 정의하였습니다.

마지막으로 국내 주식 시장은 시장의 규모에 따라 코스피와 코스닥으로 나뉩니다. 코스피는 우리나라를 대표하는 대기업들의 주식이 거래되는 시장이고 코스닥은 중소기업이나 벤처기업과 같은 상대적으로 규모가 작은 기업들의 주식이 거래되는 시장입니다. 코스피

에 상장되어 있는 기업들의 주식 가격은 변동성이 크지 않아 안정적인 반면 코스닥 시장에서의 주가는 높은 변동성으로 인해 주가가 불안정합니다. 저희는 이번 프로젝트에서 저희의 방법론이 서로 다른 특성을 가진 두 가지 시장에서도 적용될 수 있는지를 알아보기 위해 코스피와 코스닥에서의 예측 성능을 모두 측정하기로 결정하였습니다.

3. Data Explanation

3-1. 데이터 수집

저희는 pandas의 read_html 모듈을 사용하여 네이버 금융 웹 페이지를 크롤링하여 데이터를 수집하였습니다.

코스피 시장에서는 삼성 전자의 주가 데이터를 수집하였고 코스닥 시장에서는 셀트리온 헬스케어의 주가 데이터를 수집하였습니다.

문제 정의 단계에서 저희는 하루 단위의 시가, 종가, 거래량을 예측하기로 결정하였기 때문에 데이터 수집 단계에서도 각 종목별로 일별 시가, 종가, 거래량을 수집하였습니다.

삼성 전자의 경우, 2001.10.29 ~ 2018.04.27까지(기간 내 주말 제외)의 총 4084일의 데이터를 수집하였습니다. 삼성 전자는 2018.04.27에 액면 분할을 통해 주식 가격이 1주당 250만원 선에서 5만원 선으로 내려왔습니다. 액면 분할을 통한 주식 가격의 하락은 과거의 주식 가격을 통해 예측할 수 있는 것이 아닙니다. 따라서 저희는 2018.04.28 이전의 주식 가격만 수집하고 이들을 모델링에 활용하기로 하였습니다.

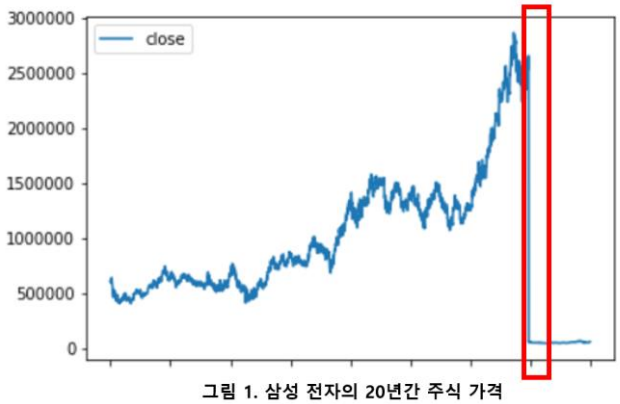

그림 1. 삼성 전자의 20년간 주식 가격

셀트리온 헬스케어의 경우, 2017.07.28 ~ 2020.06.08까지(기간 내 주말 제외)의 총 700일의 데이터를 수집하였습니다. 셀트리온 헬스케어는 코스닥에 상장된 지 약 3년이 되었기에 삼성 전자에 비해 데이터가 비교적으로 적게 수집되었습니다.

[그림 2]와 [그림 3]은 최종적으로 수집된 삼성전자 주식 데이터와 셀트리온 헬스케어의 주가 데이터입니다.

```
In [36]: samsung.head()
Out[36]:
```

	date	close	open	volume
0	2001-10-29	184000	183000	509692
1	2001-10-30	174000	177000	676544
2	2001-10-31	173500	170500	579648
3	2001-11-01	179000	177000	685945
4	2001-11-02	183000	181500	758497

그림2. 수집된 삼성 전자 주식 데이터

```
In [33]: celtrion.head()
Out[33]:
```

	date	close	open	volume
0	2017-07-28	50300	43650	14075726
1	2017-07-31	49750	49800	4965563
2	2017-08-01	51300	50900	3611026
3	2017-08-02	51000	50800	2495045
4	2017-08-03	51600	50500	3930618

그림3. 수집된 셀트리온 헬스케어 주식 데이터

3-2. 데이터 전처리 및 데이터 분리

앞선 문제 정의 과정에서 기술적 주가 분석 방법론을 따르기 위해 과거의 데이터를 Variable로 삼기로 하였습니다. 따라서 데이터 전처리 과정에서 수집된 데이터의 Column을 조정해줄 필요가 있었습니다. 저희는 몇 일 동안의 주가 데이터에 기반하여 다음날 가격을 예측할 지에 따라 Column을 그에 맞게 복제해주었습니다. [그림 4]는 종가를 3일 이전의 데이터에 기반하여 예측하기로 한 경우의 예시입니다.

```
In [87]: window_size = 3

         for s in range(1, window_size+1):
             celtrion['close_{}'.format(s)] = celtrion['close'].shift(s)

In [88]: celtrion.head()
Out[88]:
```

	date	close	close_1	close_2	close_3
0	2017-07-28	50300	NaN	NaN	NaN
1	2017-07-31	49750	50300.0	NaN	NaN
2	2017-08-01	51300	49750.0	50300.0	NaN
3	2017-08-02	51000	51300.0	49750.0	50300.0
4	2017-08-03	51600	51000.0	51300.0	49750.0

그림4. 3일 이전의 데이터에 기반하여 종가를 예측할 경우의 데이터

Min-Max 스케일링을 통해 시가, 종가, 거래량을 0 ~ 1 사이의 값으로 정규화 해주었습니다. 또한 결측 치를 대비하여 결측 치가 발생한 행은 제거하였습니다.

저희는 모델의 학습 과정에서 Overfitting을 방지하기 위해 훈련 데이터와 테스트 데이터를 분리할 필요가 있다고 판단하였습니다. 하지만 붓꽃의 품종 분류를 할 때와 같이 데이터를 무작위로 분리하는 것은 저희 프로젝트에서 의미가 없었습니다. 즉, 주가는 시

간 순서로 변화하는 시계열 데이터(Time Series Data)이기 때문에 데이터를 섞게 될 경우 그 시간 순서가 훼손된다고 생각하였습니다.

따라서 저희는 삼성 전자의 데이터는 수집된 양이 많았기에 가장 최근의 2년간 데이터를 테스트 데이터로 사용하였고 셀트리온 헬스케어 데이터는 최근 4달(약 120일)간 데이터를 테스트 데이터로 사용하였습니다.

4. Modeling Method

4-1. 예측 모델 선정

주식 데이터는 관측치가 시간적 순서를 가지기 때문에 시계열 데이터(Time Series Data)입니다. 따라서 저희는 모델을 선정하기 위해 시계열 데이터에 특화된 모델에는 어떠한 것이 있을지에 대해 조사를 진행했습니다. 조사 결과, RNN 기반의 모델들, Transformer 기반의 모델들이 후보로 선정되었습니다. Transformer 기반의 모델들의 경우, 자연어 처리 분야에서 최근 상당히 좋은 성과를 내고 있는 모델이었지만 그 모델의 구조를 이해하고 이들을 훈련 시키는 것은 이번 학기에서 배운 지식들의 범위를 넘어선다고 판단하여 RNN 기반의 모델을 사용하기로 결정했습니다.

기본 RNN은 입력 시퀀스의 길이가 길 경우, Back Propagation시에 Vanishing Gradient 문제가 발생하여 학습능력이 크게 저하되는 것으로 알려져 있습니다. 주식 가격을 예측하기 위해서는 입력 시퀀스의 길이. 즉, 입력 데이터가 긴 시간 축을 가져도 좋은 성능을 낼 수 있는 모델이 필요했습니다. 따라서 저희는 기본 RNN에서의 Vanishing Gradient 문제를 해결하기 위해 고안된 LSTM 모델을 사용하기로 결정하였습니다.

LSTM 모델은 기본 RNN에서 Cell State와 Gate가 추가된 것입니다. Cell State는 장기적으로 이전 시간에 예측된 정보를 기억하는 역할을 합니다. Gate는 Cell State에 정보를 추가할지, 추가하지 않을 지를 결정하는 역할을 합니다. LSTM은 이렇게 추가적인 Layer를 추가함으로써 기본 RNN의 문제를 해결할 수 있었습니다.

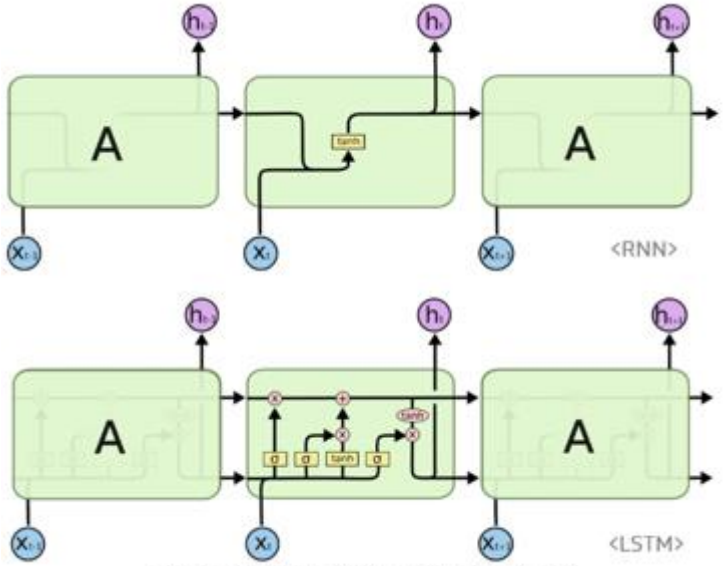

그림5. RNN과 LSTM의 구조를 비교한 그림

4-2. LSTM 모델링 방법

우선, LSTM 모델을 쉽게 구현하기 위해서 PyTorch를 사용하였음을 알려드립니다.

LSTM 모델의 출력은 2개입니다. 하나는 Cell State의 출력이고 다른 하나는 모델의 예측 결과입니다. 자연어 처리나 음성과 같은 시계열 데이터의 경우에는 문장이나 발화의 단위로 입력이 구분될 수 있기 때문에 이전 입력 데이터에 대한 Cell State 출력과 모델의 예측 결과를 저장할 필요가 없습니다.

하지만 주식 데이터의 경우에는 시간 순서로 계속 입력이 되기 때문에 이전 입력 데이터의 Cell State 출력과 모델의 예측 결과를 모델 내부에 저장을 해두어야 합니다. 따라서 저희는 LSTM 클래스 내부의 hidden이라는 변수에 이 출력 값들을 저장할 수 있게 해주었습니다.

주식 가격을 예측하는 것은 회귀 문제이기 때문에 LSTM Layer를 통해 나온 예측 결과를 주식 가격의 형태로 바꿔줄 수 있는 Layer가 추가적으로 필요했습니다. 따라서 Linear Layer를 추가하여 LSTM을 통해 나온 벡터 차원들을 출력 크기에 맞게끔 변환해주도록 했습니다.

또한 DropOut Layer를 추가하여 모델이 학습 데이터에 Overfitting되는 것을 방지하도록 했습니다.

```python
class LSTM(nn.Module):
    def __init__(self, input_dim, hidden_size, output_size, num_layer, hps):
        super(LSTM, self).__init__()
        self.hps = hps # 모델의 하이퍼파라미터 정보 저장
        self.hidden_size = hidden_size # hidden unit의 수를 저장
        self.num_layer = num_layer # LSTM Layer의 수를 저장

        self.rnn = nn.LSTM(input_dim, hidden_size, num_layer)
        self.linear = nn.Linear(hidden_size, output_size) # 회귀 문제로 바꾸기 위한 선형 Layer

        self.dropout = nn.Dropout(p=0.1) # 과적합 방지를 위한 DropOut

        self.hidden = None # 이전 예측 결과 사용을 위해 hidden unit 예측 결과 저장
```

그림6. LSTM 모델 구현 코드 - 1

학습을 위해서 초기에는 Cell State의 출력과 모델의 출력을 초기화해줄 필요가 있었기에 init_hidden() 함수를 작성하였습니다.

마지막으로 PyTorch에서 CPU나 GPU 장치에 따라 다른 Tensor를 사용하기에 이를 위한 코드도 작성해주었습니다.

```python
def init_hidden(self, batch_size):
    h0 = torch.zeros(self.num_layer, batch_size, self.hidden_size).requires_grad_()
    c0 = torch.zeros(self.num_layer, batch_size, self.hidden_size).requires_grad_()

    return (h0, c0)

def forward(self, input):
    device = input.device

    h0, c0 = self.hidden

    h0 = h0.to(device)
    c0 = c0.to(device)

    out, (hn, cn) = self.rnn(input, (h0.detach(), c0.detach()))

    self.hidden = (hn, cn)

    out = self.dropout(out)

    out = self.linear(out[:, -1, :])

    return out
```

그림7. LSTM 모델 구현 코드 - 2

4-3. 선정된 모델 훈련 방법

모델을 학습시키기 위해서는 우선 적절한 손실 함수를 선정할 필요가 있었습니다. 회귀 문제의 경우, 일반적으로 MSE(Mean Square Error)를 손실 함수로 사용하기 때문에 이번 프로젝트에서는 MSE를 손실 함수로 선정하였습니다.

LSTM은 딥러닝 모델입니다. 딥러닝 모델은 옵티마이저를 잘 선정하여 학습을 빨리 진행할 수 있도록 하는 것이 중요합니다. 따라서 Momentum과 Learning Rate 업데이트 기능을 가진 Adam 옵티마이저를 정의하여 사용했습니다.

```
# loss function
criterion = torch.nn.MSELoss(reduction='mean')

# optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=hps.learning_rate)
```

그림8. 손실 함수와 옵티마이저 코드

5. Model Assessment

앞선 모델링 과정에서는 MSE를 손실 함수로 사용했습니다. 모델을 평가하는 과정에서도 이와 비슷한 평가 지표인 RMSE를 사용하여 그 성능을 평가했습니다. 또한 몇 일 동안의 주가 데이터에 기반하여 다음날 가격을 예측하는 것이 좋은지를 알아보기 위해 입력 시퀀스의 길이를 다르게 하여 모델을 학습시키고 그에 따른 성능도 측정했습니다.

삼성 전자의 주식 시세를 나타내는 각 지표들을 입력 시퀀스 별로 학습시킨 RMSE 성능은 다음과 같습니다.

	시가 (open)	종가 (close)	거래량 (volume)
Seq Length = 10	0.15	0.04	0.09
Seq Length = 30	0.06	0.08	0.08
Seq Length = 50	0.03	0.04	0.08

셀트리온 헬스케어의 주식 시세를 나타내는 각 지표들을 입력 시퀀스 별로 학습시킨 RMSE 성능은 다음과 같습니다. 셀트리온 헬스케어의 경우, 데이터가 삼성전자에 비해 턱없이 부족했기 때문에 RMSE 성능이 삼성전자에 비해 낮게 측정되었습니다.

	시가 (open)	종가 (close)	거래량 (volume)
Seq Length = 10	0.14	0.13	0.19
Seq Length = 30	0.16	0.13	0.22
Seq Length = 50	0.13	0.12	0.13

[그림 9]는 50일 이전의 데이터를 기반으로 삼성 전자의 종가를 LSTM으로 예측한 결과와 실제 삼성 전자의 종가를 비교하기 위해 시각화한 Plot입니다.

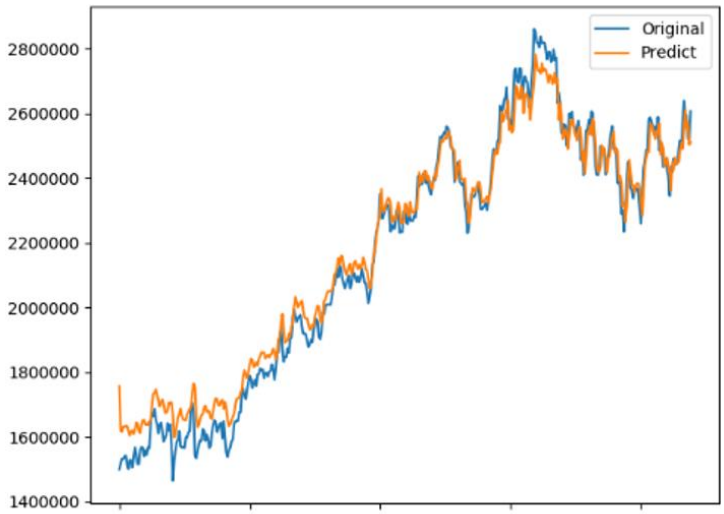

그림9. 삼성 전자의 종가 예측 결과와 실제 가격 비교

[그림 10]은 50일 데이터를 기반으로 셀트리온 헬스케어의 종가를 LSTM으로 예측한 결과와 실제 셀트리온 헬스케어의 비교하기 위해 시각화한 Plot입니다.

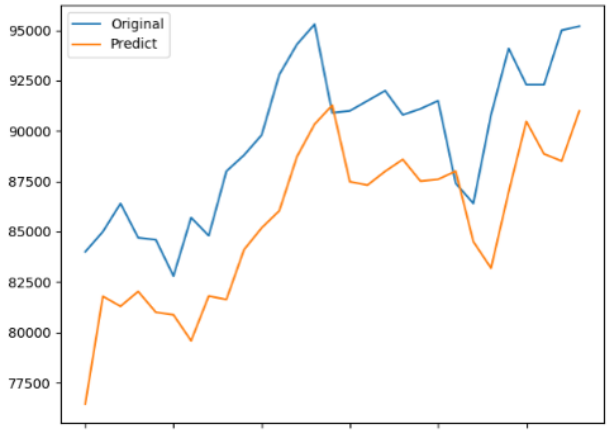

그림10. 셀트리온 헬스케어의 종가 예측 결과와 실제 가격 비교

6. Conclusion

이번 프로젝트에서는 LSTM을 활용하여 실제 코스피와 코스닥에 상장되어 있는 기업들의 주가 데이터를 통해 시가, 종가, 거래량을 예측했습니다. 그 결과, 각 지표들의 실제 수치를 정확히 예측하는 데에는 어려움이 있었지만 그들의 추세를 예측하는 데에는 상당한 효과가 있었습니다. 따라서 이번 프로젝트에서는 기술적 주가 분석 방법론의 전제인 주식 가격은 과거의 데이터를 통해 어느 정도 예측할 수 있다는 것을 다소 증명했습니다.

하지만 데이터를 수집하는 과정에서 삼성전자의 액면 분할은 기술적 주가 분석 방법을 적용할 수 없다는 것을 밝혔듯이 기술적 주가 분석 방법이 주식 가격을 예측하는 절대적인 방법은 아닙니다. 따라서 이번 프로젝트에서는 기술적 주가 분석 방법이 미래의 주가를 예측하는데 어느 정도의 도움은 되지만 맹신할 수는 없으며 기본적 주가 분석도 같이 행해져야 함을 시사하였습니다.