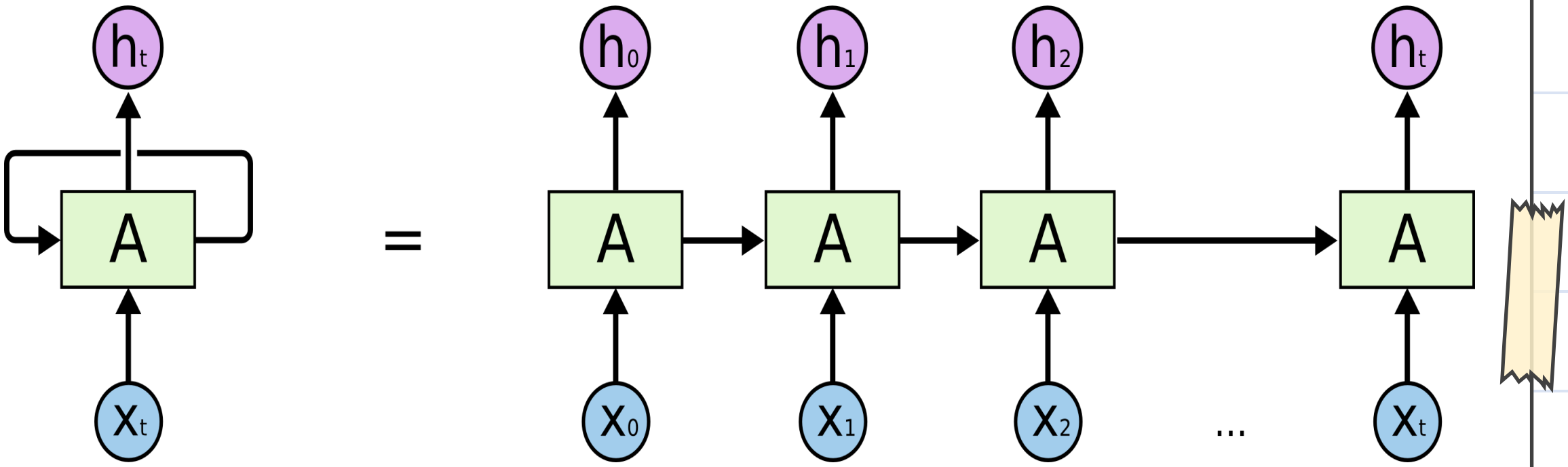
Stock Market Prediction with LSTM

B611137 이강복

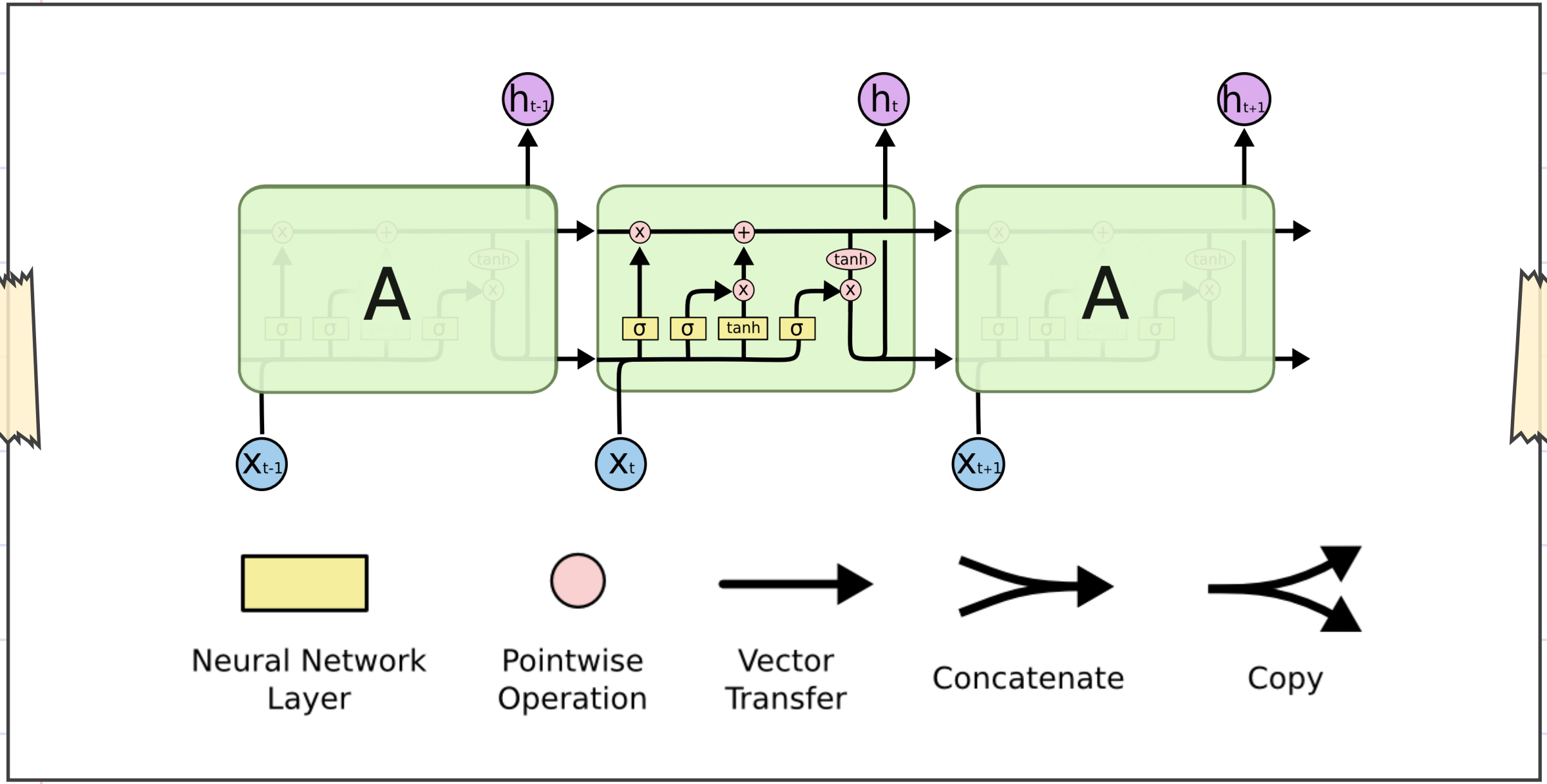
목차

Using Model - RNN



- Sequence 데이터를 다루는데 용이하다.
- 하지만 장기 의존성(Long-term dependency)의 문제를 가지고 있다.

Using Model - LSTM



Dataset

- S&P 500 Dataset 사용
- FinanaceDataReader 라이브러리를 사용하여 데이터를 불러들임
- 2017-01-03 ~ 2020-12-18 까지의 Data 사용
- 2017년 이후에 등록된 주식의 경우에는 데이터가 충분치 않아 제거하고 실시 (총 486개 사용)
- 각 주식당 Train : Test = 530 : 230 (2017~2019년의 데이터를 학습해서 2020년의 데이터를 Test)

Data Preprocessing

$$R_t^{m,s} = \frac{P_t^s}{P_{t-m}^s} - 1.$$

Change Value

$$\tilde{R}_t^{m,s} = \frac{R_t^{m,s} - \mu_{train}^m}{\sigma_{train}^m}.$$

Standardization

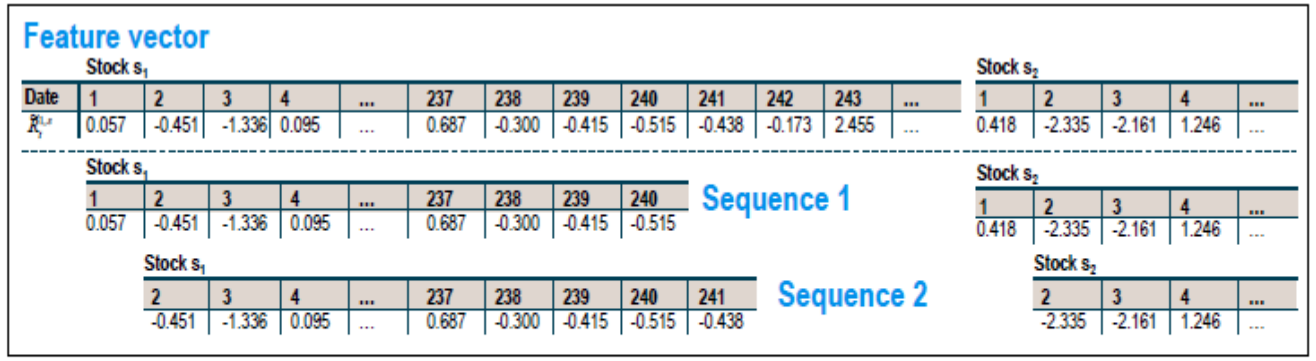


Figure 1: Construction of input sequences for LSTM networks (both, feature vector and sequences, are shown transposed)

Date	Change
2017-01-03	-0.0029
2017-01-04	0.0015
2017-01-05	-0.0034
2017-01-06	0.0029
2017-01-09	-0.0054
...	...
2020-12-14	-0.0054
2020-12-15	0.0092
2020-12-16	0.0042
2020-12-17	0.0071
2020-12-18	-0.0013

'3M Company' Change Value

Data Preprocessing

Target Data는 약 500개의 Stock들의 Change_Value를 정렬 후, 중앙값 보다 크거나 같으면 1, 아니면 0으로 labeling 해준다.

Date	Change0	Change1	Change2	Change3	Change4	Change5	Change6	Change7	Change8	Change9	...
2017-01-03	-0.0029	0.0011	0.0011	-0.0028	-0.0057	0.0147	0.0051	0.0079	0.0088	0.0034	...
2017-01-04	0.0015	0.0097	0.0097	0.0301	0.0024	0.0197	0.0064	0.0000	0.0082	-0.0086	...
2017-01-05	-0.0034	-0.0033	-0.0033	-0.0080	-0.0150	0.0155	0.0170	-0.0166	-0.0007	-0.0130	...
2017-01-06	0.0029	0.0032	0.0032	0.0053	0.0114	-0.0008	0.0226	0.0071	-0.0131	0.0359	...
2017-01-09	-0.0054	-0.0028	-0.0028	0.0146	-0.0112	-0.0055	0.0025	0.0150	-0.0006	-0.0245	...
...	...	...	...	...	...	...	...	...	...	...	...
2020-12-14	-0.0054	-0.0002	-0.0002	0.0355	-0.0085	0.0270	0.0221	0.0342	-0.0006	0.0071	...
2020-12-15	0.0092	0.0154	0.0154	0.0289	0.0135	0.0114	-0.0078	0.0247	0.0145	0.0460	...
2020-12-16	0.0042	-0.0083	-0.0083	-0.0044	0.0016	0.0150	0.0150	-0.0028	0.0093	-0.0171	...
2020-12-17	0.0071	0.0129	0.0129	0.0286	0.0688	0.0027	0.0111	-0.0001	-0.0061	0.0087	...
2020-12-18	-0.0013	0.0057	0.0057	0.0404	0.0067	0.0216	0.0153	-0.0095	-0.0022	0.0204	...

999 rows × 487 columns

Data Preprocessing

Sequence와 label한 데이터를 concat을 통해 하나로 합쳐준다.

	Sequence0	Sequence1	Sequence2	Sequence3	Sequence4	Sequence5	Sequence6	Sequence7	Sequence8	Sequence9	...
2017-01-03	-0.204062	0.062153	-0.234113	0.147617	-0.356012	-0.265892	0.418387	-0.180361	-0.048039	-0.071874	...
2017-01-04	0.062624	-0.234766	0.147763	-0.355372	-0.265184	0.418483	-0.181174	-0.047144	-0.072239	0.388062	...
2017-01-05	-0.234368	0.146987	-0.355344	-0.264470	0.419051	-0.181102	-0.047938	-0.071365	0.387566	0.037058	...
2017-01-06	0.147479	-0.355958	-0.264421	0.420323	-0.180411	-0.047861	-0.072163	0.388836	0.036662	-0.096081	...
2017-01-09	-0.355589	-0.265064	0.420532	-0.179629	-0.047198	-0.072087	0.388106	0.037630	-0.096439	-0.023460	...
...	...	...	...	...	...	...	...	...	...	...	...
2019-02-05	0.250517	0.116689	-0.973619	0.408202	-0.416563	0.842431	0.485004	3.410424	1.337425	-0.319997	...
2019-02-06	0.117173	-0.974034	0.408409	-0.415973	0.842914	0.485103	3.410133	1.339516	-0.320291	-0.362359	...
2019-02-07	-0.973816	0.407549	-0.415959	0.844530	0.485658	3.410349	1.338924	-0.319632	-0.362642	-0.616534	...
2019-02-08	0.408104	-0.416553	0.844839	0.486984	3.410310	1.339057	-0.320466	-0.362019	-0.616744	0.708806	...
2019-02-11	-0.416199	0.843838	0.487209	3.414018	1.339439	-0.320399	-0.362859	-0.616341	0.708219	-0.168702	...

257580 rows × 240 columns

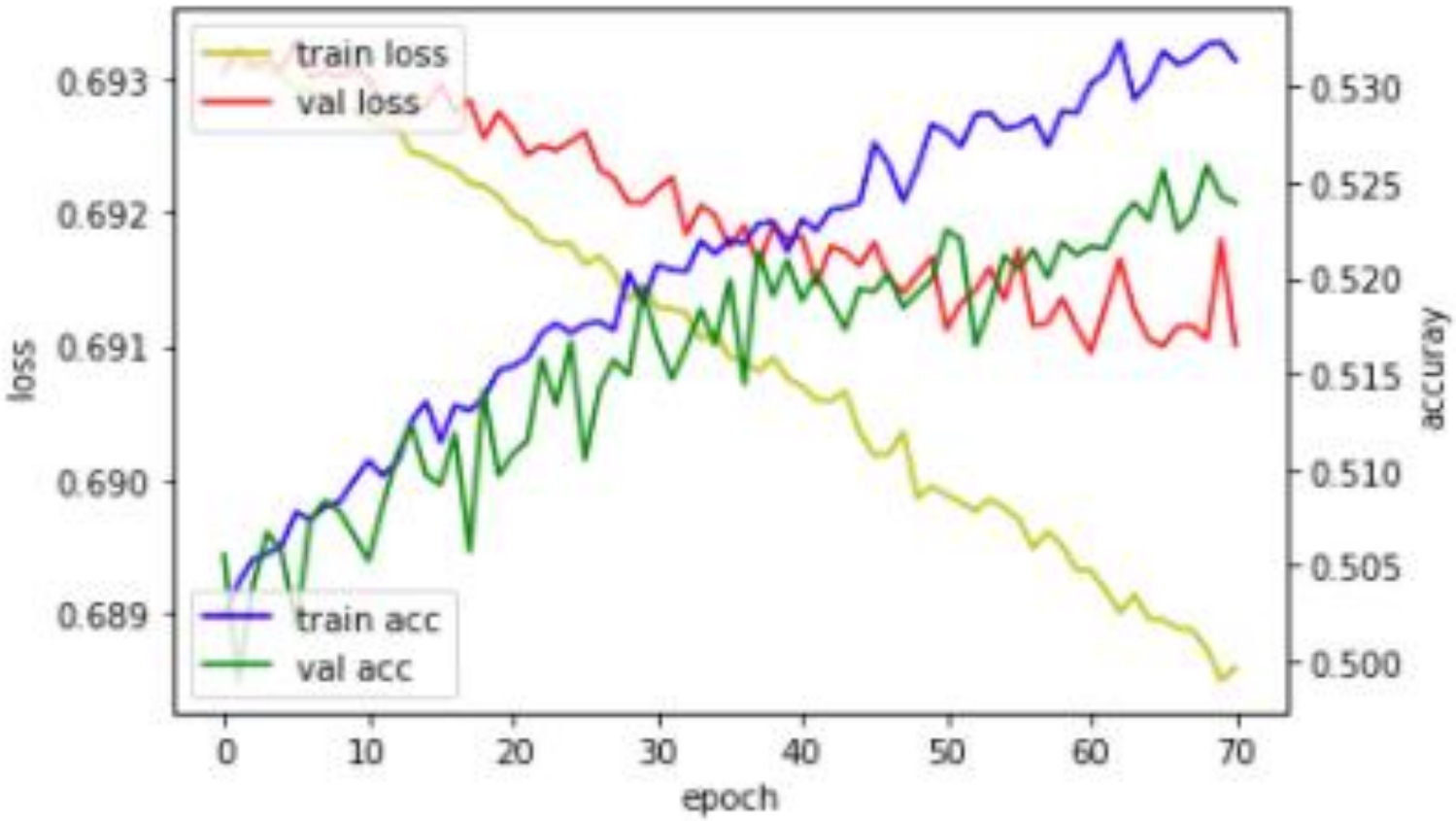
0	1.0
1	0.0
2	0.0
3	1.0
4	0.0
...	...
257575	1.0
257576	1.0
257577	0.0
257578	0.0
257579	0.0

Training

- 단층 LSTM 사용
- TimeStep : 240, hidden neurons : 25, dropout :0.1
- Output : Dense with 2 neurons, activation : softmax
- Epoch : 1000, Batch_Size = 230, Optimizer : RMSprop, loss : cross_entropy
- Early_stopping patience = 10, Use Callback

Training

- Early Stopping in Epoch 71



Result

```
1 from sklearn.metrics import confusion_matrix
2 from sklearn.metrics import precision_score
3 from sklearn.metrics import accuracy_score
4 confusion_matrix(y_t,y_p)
```

```
array([[25672, 30042],
       [24327, 31739]], dtype=int64)
```

```
1 precision_score(y_t,y_p)
```

```
0.5137339958887036
```

```
1 accuracy_score(y_t,y_p)
```

```
0.5136070853462158
```

Trading

- 각 주식 S_n에 대한 Sequence_test_data를 예측한 후 나온 확률값들을 정렬하고, Class 0에 속할 확률이 높은 10가지의 주식을 팔고, Class 1에 속할 확률이 높은 10가지의 주식을 사는 방식으로 진행

- 높은 10가지의 주식을 사는 방식으로 Trading 한다.

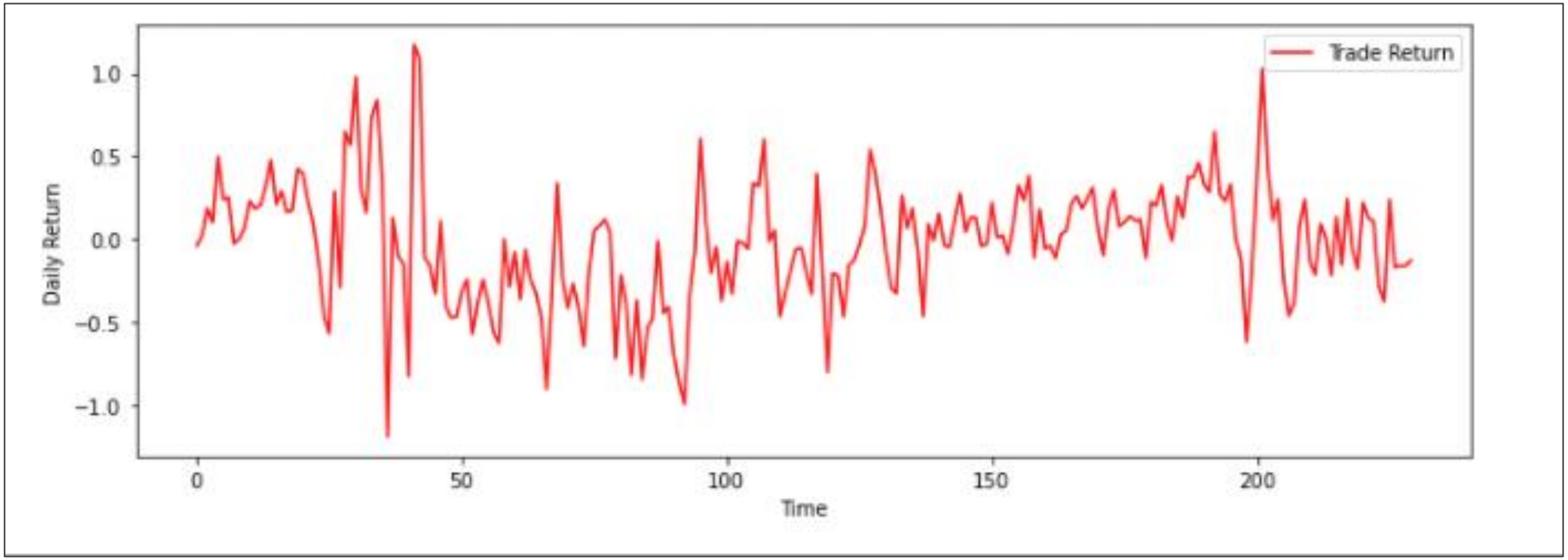
- Class 1에 속할 확률 = 1 – Class 0에 속할 확률

- 구매 : + Change_value, 판매 : - Change_Value

Stock0	Stock1	Stock2	Stock3	Stock4	Stock5	Stock6	Stock7	Stock8	Stock9	...
0.518397	0.474813	0.474813	0.518093	0.484867	0.548728	0.502323	0.498831	0.409823	0.507264	...

Class 1에 속할 확률

Trading




```
1  np.average(trade_value)
```

```
-0.015645217391304435
```

Conclusion

- Data Feature이 다양하지 못했다.
- Data Preprocessing의 방법이 다양하지 못했다.
- 최적의 HyperParameter나 규제 방법을 찾지 못했다.
- 다른 모델과의 비교를 통해 LSTM이 시계열 데이터에서 뛰어나다는 것을 보여주지 못했다.
- Trading 방법이 현실적이지 못하였다.