

Stock Prediction Program

Rokey 스터디

목 차

1. LSTM(Long Short Term Memory)

- RNN(순환 신경망)
- LSTM(장단기 메모리)

2. 데이터 수집

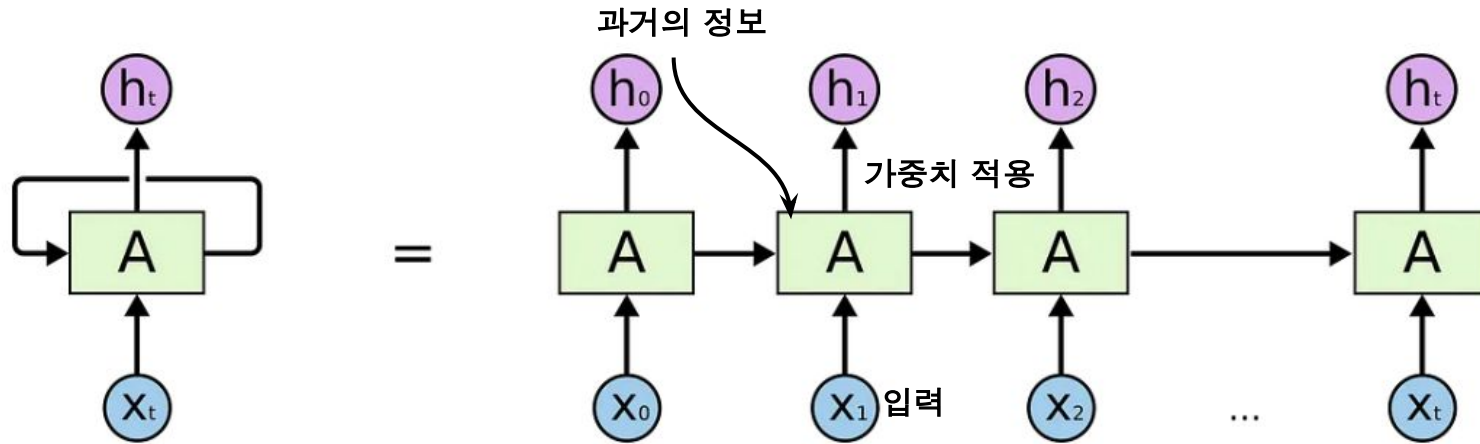
- pykrx 라이브러리

3. 데이터 전처리

- 결측치 처리
- 이상치 제거
- 정규화
- 훈련/테스트 데이터 분할
- 시퀀스 데이터 생성

4. 보완점 및 계획

- RNN



- ❑ 정의

- 시계열 또는 순차적 데이터(텍스트, 음성, 주식)를 처리하는 데 특화된 신경망
- 과거의 정보를 사용하여 현재 및 미래의 입력에 대한 신경망의 성능을 개선하는 딥러닝 구조

- ❑ 작동 원리

- 셀은 입력 데이터와 은닉 상태를 기반으로 출력을 생성하며 은닉 상태는 다음 시간 스텝으로 전달

= 데이터가 시간 순으로 순차적으로 처리되며 이전의 정보가 다음 예측에 영향을 미침



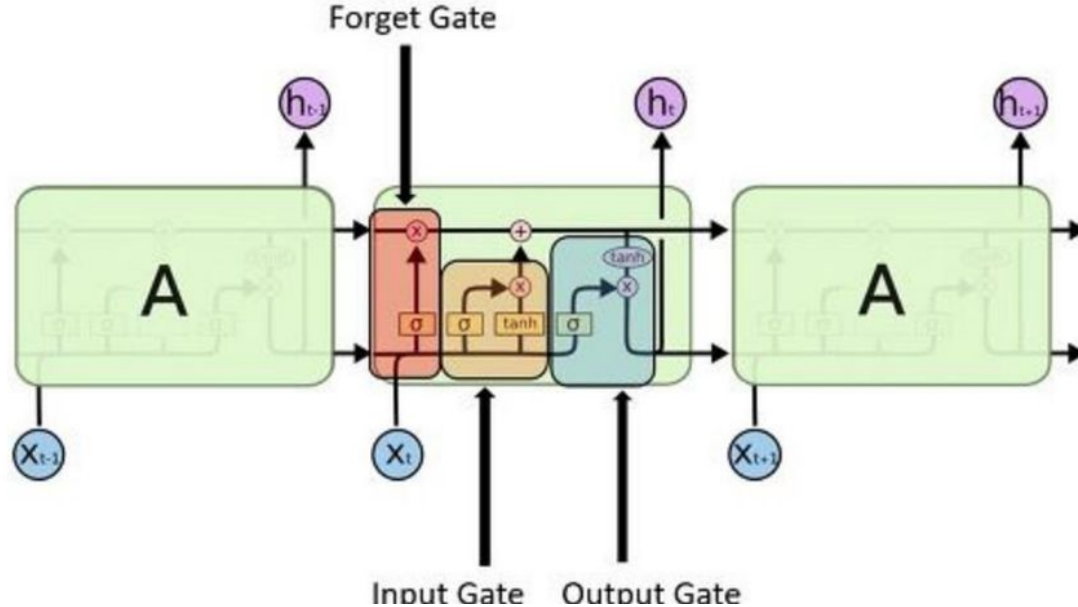
- ❑ RNN의 문제점

- 1) 기울기 소실: 과거의 데이터가 많아지면 과거 정보가 사라져 예측 성능 저하
- 2) 기울기 폭발: 기울기 값이 너무 커져 과도한 변화로 학습이 불안정

LSTM(Long Short-Term Memory)

기존 셀 상태에서 얼마나 많은 정보 버릴지 결정

- LSTM



- ❑ 정의

- 기존 RNN의 기울기 소실 문제를 해결하기 위해 고안된 순차적 데이터 처리에 특화된 모델
- 장기적인 의존성을 학습할 수 있도록 설계되어 **긴 시퀀스 데이터 처리에 강점**

- ❑ 작동 원리

- 1) 망각 게이트

- 이전 정보 중 어떤 부분을 잊을지 결정(현재 입력과 은닉 상태를 바탕으로 0,1 사이의 값 출력)

- 2) 입력 게이트

- 현재 입력과 이전 은닉 상태를 기반으로 어떤 새로운 정보를 셀 상태에 추가할지 결정

- 3) 출력 게이트

- 최종적으로 어떤 정보가 출력될지 결정하며 이는 다음 시점으로 전달될 은닉 상태

주식 데이터 수집 방법 3가지

1. 웹 크롤링

- 주식 관련 웹 사이트에서 크롤러를 이용해 데이터 수집
- 하지만 사이트의 구조가 변경되거나 크롤러의 접근을 차단할 수도 있는 문제점

2. 주식 거래소 데이터 다운로드

- 거래소에서 직접 제공하는 데이터이므로 정확성 높음
- 데이터를 실시간으로 받아오기 어려우며 매번 데이터를 다운로드해야 할 수도 있음

3. 주식 데이터 API 활용(FinaceDataReader, pykrx, pandas_datereader, yfinance)

- 무료 API인 라이브러리를 통해 주식 데이터를 실시간으로 수집 가능
- 데이터의 신뢰성이 높고 실시간으로 쉽게 접근 가능

```
from pykrx import stock
data = stock.get_market_ohlcw_by_date("2020-01-01", "2025-02-18", "454910")
print(data.head())
print('----- tiker: 주식을 식별하는 고유한 코드 -----')
print(data.tail())
```

날짜	시가	고가	저가	종가	거래량
2023-10-05	59100	67600	46450	51400	23416071 NaN
2023-10-06	49550	53400	49100	51800	6183396 0.778210
2023-10-10	51200	51300	46300	46650	4693925 -9.942085
2023-10-11	47600	49100	45400	46900	3303330 0.535906
2023-10-12	47250	50200	46700	47800	4428666 1.918977

날짜	시가	고가	저가	종가	거래량
2025-02-12	68900	69100	67200	67500	412025 -2.315485
2025-02-13	67500	71400	67500	70700	806930 4.740741
2025-02-14	69700	70400	68600	69200	577548 -2.121641
2025-02-17	70000	70100	68200	69200	290810 0.000000
2025-02-18	69800	77000	69100	76300	3720345 10.260116

Selection of prediction period

● 주식 데이터 사용할 기간 선택

❑ 훈련 데이터와 테스트 데이터의 비율 고려

- 훈련 데이터: 충분한 양을 확보해야 데이터의 패턴을 제대로 학습할 수 있기 때문에 80% 비율
- 테스트 데이터: 학습한 패턴을 실제 데이터에 얼마나 잘 적용할 수 있는지 평가

❑ 각 데이터 별 기간 지정

- 훈련 데이터: 2023년 10월 5일부터 2024년 9월 6일까지의 데이터(80%)
- 테스트 데이터: 2023년 9월 7일부터 2024년 11월 30일까지의 데이터(20%)
- 예측할 데이터: 2024년 11월30일~ 2025년 2월 xx일까지(예측 후 정확도로 모델의 성능 판단)

```
data = stock.get_market_ohlcw_by_date("2023-10-05", "2024-11-30", "454910")
```

● 종가 데이터 사용

- 하루를 마감하고 나서 시장에서 최종적으로 결정된 가격
- 다른 가격들에 비해 안정적이고 신뢰할 수 있는 지표

```
# 1) 종가 데이터만 선택
data = data[['종가']]
print("원본 데이터 개수:", len(data))
```

✓ 0.0s

원본 데이터 개수: 283

날짜	종가
2023-10-05	51400
2023-10-06	51800
2023-10-10	46650
2023-10-11	46900
2023-10-12	47800

...	...
2025-02-13	70700
2025-02-14	69200
2025-02-17	69200
2025-02-18	76300
2025-02-19	74100

Data Preprocessing

• 데이터 전처리

❑ 결측치 확인

- 모델의 정확도 향상: 결측치가 있으면 모델이 그 데이터를 처리 못하거나 왜곡된 예측
- 시계열 데이터의 연속성 유지: 시계열 데이터는 시간 순서대로 이어지는 패턴 학습

```
# 2. 결측치 확인  
print(data.isnull().sum())
```

✓ 0.0s

종가 0

거래가 없는 주말이나
공휴일의 데이터 없음

날짜	종가
2023-10-05	51400
2023-10-06	51800
2023-10-10	46650
2023-10-11	46900
2023-10-12	47800



결측치 처리?

```
# 전체 기간의 날짜 생성 (거래일이 아닌 날 포함)  
full_dates = pd.date_range(start="2023-10-05", end="2024-11-30", freq='D')
```

```
# 전체 날짜에 대한 데이터프레임 생성  
data = data.reindex(full_dates)
```

```
# 결측치 개수 확인  
ms_data = data.isnull().sum()  
print('추가된 결측치 개수:', ms_data['종가'])
```

원본 데이터 개수: 283
추가된 결측치 개수: 140

- 전체 기간의 날짜를 생성하여 reindex를 통해 데이터프레임을 다시 생성하고 이 때 기존 데이터에 없는 날짜에 대해 NAN 값을 할당

Data Preprocessing

- 데이터 전처리(bfill, interpolate, 평균값, 중앙값으로 채우기(fillna))

□ 보간법을 이용한 결측치 처리

- 결측값을 주변 데이터 값을 이용해 추정하여 채우는 기법
- 시계열 데이터에서 연속된 데이터 사이의 흐름을 자연스럽게 보정

보간법	특징	추천 데이터
linear	간단한 보간, 빠름	일반적인 데이터
time	날짜 기반 보간	시계열 데이터 (주식, 기온)
polynomial	곡선 보간 가능	비선형적인 데이터
spline	더 부드러운 곡선 보간	변동성이 큰 시계열 데이터



```
# 날짜 인덱스 설정 (결측치 처리시 필요)
data.index = pd.to_datetime(data.index)

# 결측치 처리 (선형 보간법)
data['종가'] = data['종가'].interpolate(method='time')

# 결측치 처리 후 확인
ms_data_after = data.isnull().sum()
print('결측치 처리 후 결측치 개수:', ms_data_after['종가'])
```

✓ 0.0s

결측치 처리 후 결측치 개수: 0



□ 날짜 인덱스 설정

- 주가 데이터를 다루기 위해 날짜를 인덱스로 설정하는 것 필수
- 보간, 리샘플링, 이동 평균 등의 시계열 분석 기능을 활용할 수 있음

Removal Outlier

- 이상치 제거

- ❑ IQR(사분위 범위)

- 데이터의 중앙값을 기준으로 분포를 파악하고 범위를 벗어난 값을 이상치로 간주
 - 비정규분포 데이터에 적용 가능하며 주식, 기후, 시계열 데이터에 적합

- IQR의 기본 개념

- ❑ 사분위수

- 데이터를 크기 순서로 정렬했을 때 4등분하는 기준점 (Q1: 하위 25%, Q2: 중앙값, Q3: 하위 75%)

- ❑ 계산법

- $data = [1, 2, 4, 6, 8] \rightarrow Q1, Q2, Q3 = (\text{데이터 수} + 1) \times (\text{각 사분위 수 별 퍼센티지}) \text{ 번째 데이터}$
 - $IQR = Q3 - Q1$

- ❑ 이상치 판단 기준

- 하한: $Q1 - 1.5 \times IQR$
 - 상한: $Q3 + 1.5 \times IQR$

```
# IQR 계산 (1사분위 수와 3사분위 수)
```

```
Q1 = data['종가'].quantile(0.25)
```

```
Q3 = data['종가'].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
# IQR을 이용해 이상치 제거
```

```
lower_bound = Q1 - 1.5 * IQR
```

```
upper_bound = Q3 + 1.5 * IQR
```

```
data_cleaned = data[(data['종가'] >= lower_bound) & (data['종가'] <= upper_bound)]
```

```
print('이상치 제거 후 데이터 수:', len(data_cleaned))
```

✓ 0.0s

이상치 제거 후 데이터 수: 385

Removal Outlier

- 이상치 제거 후



Data Normalization

• 데이터 정규화

❑ 정의 및 특징

- 서로 다른 스케일을 가진 데이터를 **일정 범위로 변환하는 과정**
- 각 데이터의 크기 차이가 크면 특정 변수가 모델에 더 큰 영향을 미칠 수 있음
- 딥러닝 모델은 입력값의 범위가 작을수록 학습이 빨라지고 안정적

❑ Min-Max Scaling(최소-최대 정규화)

- 데이터를 0~1 사이로 변환하여 각 데이터가 전체 데이터에서 어느 위치에 있는지 비율로 표현

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler() # MinMaxScaler 객체 생성

# '종가' 컬럼에 정규화 적용
data_cleaned['정규화_종가'] = scaler.fit_transform(data_cleaned[['종가']])
print(data_cleaned.head())
```

✓ 0.1s

	종가	정규화_종가
2023-10-05	51400.0	0.133407
2023-10-06	51800.0	0.140078
2023-10-07	50512.5	0.118608
2023-10-08	49225.0	0.097137
2023-10-09	47937.5	0.075667

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

X: 원본 데이터
Xmin : 최소값
Xmax: 최대값

❑ fit_transform

- fit() : 데이터를 보고 최소값과 최대값을 계산
- transform(): 계산된 값을 사용하여 각 데이터를 0~1 사이로 반환

Removal Outlier

- 정규화 이후



Data Splitting

• 데이터 분할

❑ 데이터 분할

- 훈련 데이터: 충분한 양을 확보해야 데이터의 패턴을 제대로 학습할 수 있기 때문에 80% 비율
- 테스트 데이터: 학습한 패턴을 실제 데이터에 얼마나 잘 적용할 수 있는지 평가
- 이상치 제거로 인해 없어진 데이터를 제외하고 **데이터 재분할**

```
# 전체 데이터 길이
total_len = len(data_cleaned)
print(total_len)

# 80% 지점 계산
train_size = int(total_len * 0.8)

# 80:20으로 분할
train_data = data_cleaned['정규화_종가'].iloc[:train_size]
test_data = data_cleaned['정규화_종가'].iloc[train_size:]

# 결과 확인
print(f"전체 데이터 길이: {total_len}")
print(f"훈련 데이터 길이: {len(train_data)}")
print(f"테스트 데이터 길이: {len(test_data)}")
```

385
전체 데이터 길이: 385
훈련 데이터 길이: 308
테스트 데이터 길이: 77



❑ data.iloc[행_인덱스, 열_인덱스]

- pandas 라이브러리에서 제공하는 메서드
- 정수 기반 인덱싱을 통해 특정 행 및 열 선택

Making Sequence Data

- 시퀀스 데이터 생성

- 과거의 데이터를 이용해 미래를 예측하는 데 필수적인 과정
- 타임스텝을 설정한 만큼 데이터를 하나의 시퀀스로 묶어 모델에 입력하면 그 만큼 학습하고 후의 가격을 예측할 수 있음

```
# 시퀀스 데이터 생성 함수
```

```
def create_sequences(data, timestep):
```

```
    x = []
```

```
    y = []
```

```
    # 타임스텝만큼 데이터를 슬라이딩 윈도우로 자르기
```


```
    for i in range(len(data) - timestep): # for i in range(278)
```

```
        x.append(data[i:i+timestep]) # data[0:30] + data[1:31] + data[2:32]...
```

```
        y.append(data[i+timestep]) # data[30]+data[31]+data[32]
```

```
    return np.array(x), np.array(y)
```

데이터를 지정한 타임스텝만큼
묶기



```
# 훈련/테스트 데이터로 시퀀스 생성
```

```
timestep = 30
```

```
x_train, y_train = create_sequences(train_data.values, timestep)
```

```
x_test, y_test = create_sequences(test_data.values, timestep)
```

□ data.values

- 데이터는 판다스의 데이터프레임 형태이기 때문에 .values를 통해 Numpy 배열로 반환할 때 사용

훈련 데이터 시퀀스 x: (278, 30)

훈련 데이터 타겟 y: (278,)

테스트 데이터 시퀀스 x: (47, 30)

테스트 데이터 타겟 y: (47,)

Plan

- 보완점 및 계획

- 보완점

- 데이터 전처리 단계 중 없는 날짜 데이터에 대해서도 결측치 처리를 할 것인지 피드백 받고 적용
 - **훈련과 테스트할 데이터에 대해서만 데이터 전처리를 진행** 하였으며 예측할 데이터에 대해서도 데이터 전처리 진행
 - 지정한 주식에 대한 데이터가 적은 것이 아닌지에 대해 피드백 받고 적으면 종목 변경 및 관찰으면 이대로 사용
 - 미래의 데이터를 예측하는 게 아닌 LSTM을 이용해 과거의 데이터를 이용해 과거의 데이터를 예측하는게 맞는지 피드백 받고 예측할 과거 데이터의 기간에 대해서도 적은지 피드백

- 계획

- LSTM 모델 구축
 - 모델 예측 및 평가
 - 모델 튜닝
 - 배포(?)
 - 과거의 데이터를 과거 예측 관찰

Coding Test

- 코딩 테스트 문제(DP: 동적 계획법 활용)

〈문제〉 1로 만들기: 문제 조건

난이도 ●○○ | 풀이 시간 20분 | 시간제한 1초 | 메모리 제한 128MB

입력 조건 • 첫째 줄에 정수 X 가 주어집니다. ($1 \leq X \leq 30,000$)

출력 조건 • 첫째 줄에 연산을 하는 횟수의 최솟값을 출력합니다.

입력 예시

26

출력 예시

3

- 정수 X 가 주어졌을 때, 정수 X 에 사용할 수 있는 연산은 다음과 같이 4가지입니다.
 - X 가 5로 나누어 떨어지면, 5로 나눕니다.
 - X 가 3으로 나누어 떨어지면, 3으로 나눕니다.
 - X 가 2로 나누어 떨어지면, 2로 나눕니다.
 - X 에서 1을 뺍니다.
- 정수 X 가 주어졌을 때, 연산 4개를 적절히 사용해서 값을 1로 만들고자 합니다. 연산을 사용하는 횟수의 최솟값을 출력하세요. 예를 들어 정수가 26이면 다음과 같이 계산해서 3번의 연산이 최솟값입니다.
 - $26 \rightarrow 25 \rightarrow 5 \rightarrow 1$



Thank you for listening!