Algorithm Trading
with
Weight Agnostic Neural Networks

KU-BIG


Algorithm Trading

Reinforcement Learning

ALPHAGO

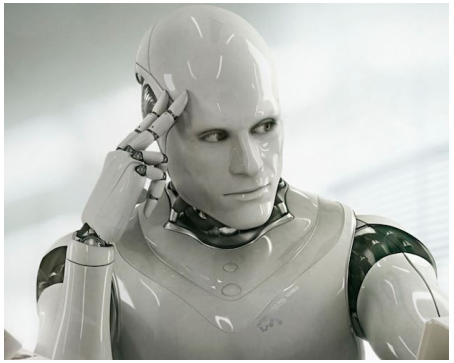
What is Algorithm Trading?

- 컴퓨터를 사용하여 인간(트레이더)이 할 수 없는 속도와 주기로 수익을 내도록 거래를 하는 일련의 명령들을 수행하는 프로그램
- 정의된 규칙들은 타이밍, 가격, 수량 혹은 다른 수학적 모델에 기초를 두고 있음
- 트레이더에게 수익을 낼 기회를 줄 뿐 아니라, 시장을 좀 더 유동적으로 만들고 트레이딩 활동에서 인간의 감정적인 영향을 없애 좀 더 체계적으로 거래를 할 수 있음
- Reference : '알고리즘 트레이딩의 기초 : 개념과 예시'

What is Algorithm Trading?

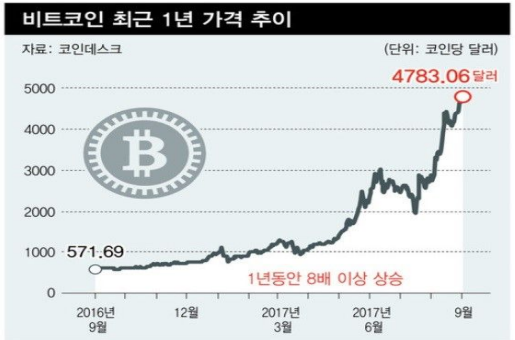
트레이더가 다음의 단순한 거래 규칙을 따른다고 가정합시다:

- 50일 이동평균선이 200일 이동평균선 **위로 올라가면** 주식 50주를 **매수**한다.
- 50일 이동평균선이 200일 이동평균선 **아래로 내려가면** 주식 50주를 **매도**한다.



이 두 가지의 간단한 규칙을 사용하면, 주식 가격(이동평균선 포함)을 자동으로 모니터링하고 매수 및 매도 주문을 넣는 컴퓨터 프로그램을 작성하는 것은 그리 어렵지 않겠죠. 트레이더는 더이상 실시간 가격과 그래프를 계속 보면서 직접 주문을 할 필요가 없습니다. 알고리즘 트레이딩 시스템은 자동으로 거래 기회를 포착하고 주문을 넣어주죠.

What is Algorithm Trading?



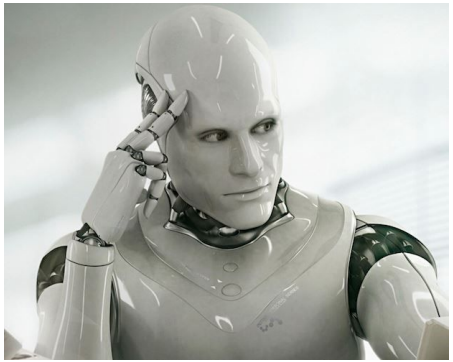
2017년 11월의 내 모습

수업 안듣고 하루종일
차트만 봄

이젠 안녕..

Algorithm Trading with Reinforcement Learning

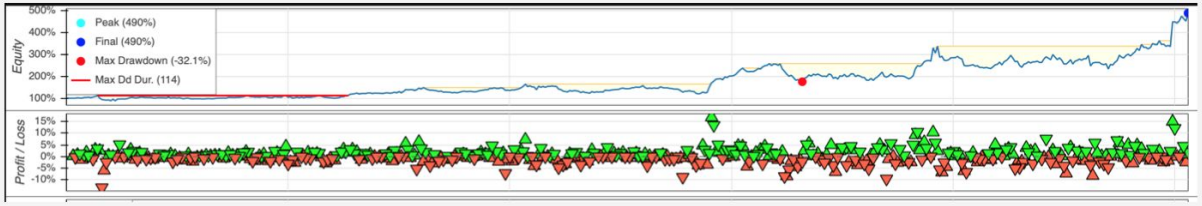
Alternative Data



+

= ?

6-1. Backtesting 결과 (이화산업 000760)



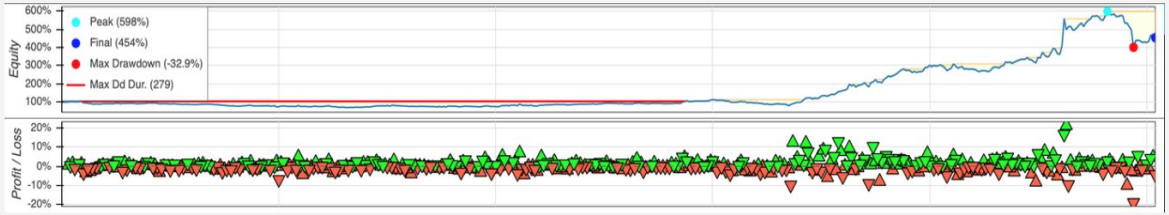
Price Prediction

- Training Period : 2013.08.01 ~ 2016.12.31
- Validation Period : 2017.01.01 ~ 2018.12.31
- Training R2 Score : 0.9987
- Validation R2 Score : 0.7461

Backtesting Result

- End Equity : 490%
- Maximal Drawdown : -32.1%
- Sortino Ratio : 0.15
- Sharpe Ratio : 0.10
- Exposure : 99.67%
- Win Rate : 49.01%

6-2. Backtesting 결과 (한국주철관공업 000970)



Price Prediction

- Training Period : 2013.08.01 ~ 2016.12.31
- Validation Period : 2017.01.01 ~ 2018.12.31
- Training R2 Score : 0.9994
- Validation R2 Score : 0.7682

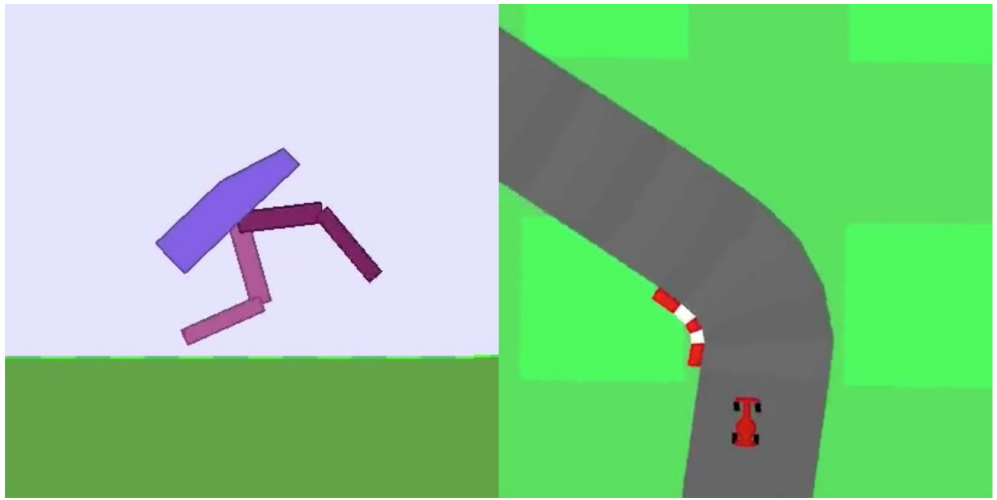
Backtesting Result

- End Equity : 454%
- Maximal Drawdown : -32.9%
- Sortino Ratio : 0.09
- Sharpe Ratio : 0.06
- Exposure : 99.60%
- Win Rate : 46.41%

Weight Agnostic Neural Network (WANN)

[Introduction]

- 랜덤으로 Weight가 초기화 되어 있어도 특정한 Task를 수행할 수 있는 뉴럴넷 구조를 찾는 방법론
- 동물이 태어나자마자 걸을 수 있다는 것에서 영감을 받은 연구



[Idea]

- 모든 가중치를 동일하게 설정하여 Topology의 성능을 평가하여 기존의 NAS(Network Architecture Search)에 비해 빠르게 네트워크 아키텍처를 탐색함

https://arxiv.org/pdf/1906.04358.pdf

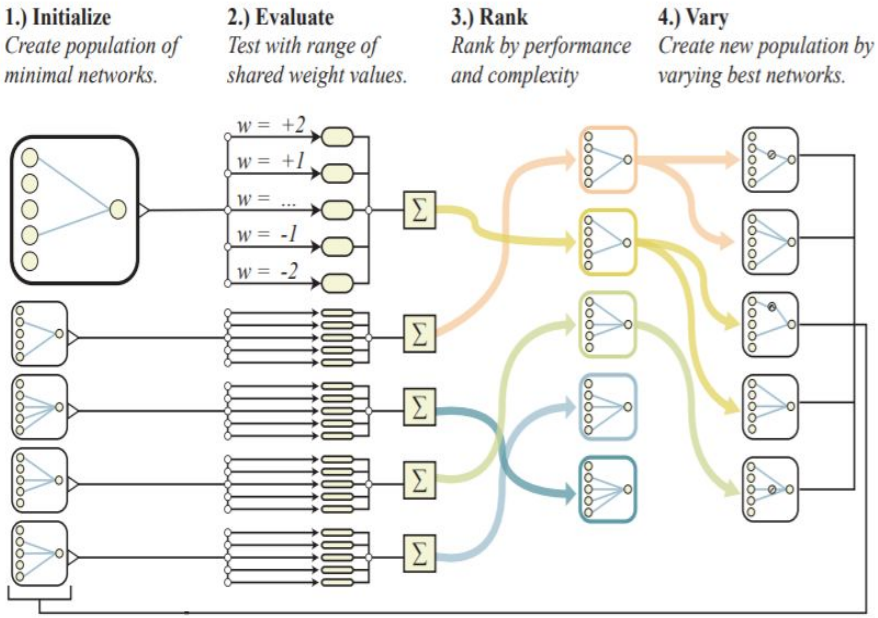
WANN

- weight training이 아닌 **weight random sampling**
- 퍼포먼스는 weight와는 무관하게 **네트워크의 형태(topology)**에 따라 결정됨
- 아키텍쳐 스스로 문제를 푸는 과정을 좀 더 효과적으로 만들기 위해, weight를 최소화하고자 → **하나의 공유된 weight** → computational cost 감소 → 계산이 더욱 빨라짐.

- 즉, weight training을 통해 optimal한 **weight를 찾는 과정이 costly**하다는 문제점을 개선하고자 함.
- 아키텍쳐가 weight에 관대하도록 진화하게 됨.
- WANN에서 **simple**의 정의
  - soft-weight sharing을 통해 모델을 단순화
  - weight의 정보량을 감소
  - weight의 search space를 단순화한다.

WANN Topology Search

1. 작은 단위의 뉴럴 네트워크를 형성.
2. 각 네트워크가 서로 다르지만 공유된 weight값들로 테스트됨
3. NEAT (Genetic Algorithm의 개선된 버전)으로 Topology 탐색
   a. 퍼포먼스와 모델의 복잡도에 따라 네트워크들의 순위가 매겨짐
   b. 가장 높은 랭크를 차지한 네트워크 모형이 변형되어 새로운 뉴럴 네트워크가 만들어짐
   c. 성능이 비슷할 경우에는 # of params. 가 작은 Topology를 선택



https://arxiv.org/pdf/1906.04358.pdf

WANN 성능 평가 & 실험계획

WANN	Test Accuracy
Random Weight	*82.0% ± 18.7%*
Ensemble Weights	*91.6%*
Tuned Weight	*91.9%*
Trained Weights	*94.2%*

ANN	Test Accuracy
Linear Regression	*91.6%* [62]
Two-Layer CNN	*99.3%* [15]

Figure 6: *Classification Accuracy on MNIST.*

A.2.4 Using backpropagation to fine-tune weights of a WANN.

We explored the use of autograd packages such as JAX [24] to fine-tune individual weights of WANNs for the MNIST experiment. Performance improved, but ultimately we find that black-box optimization methods such as CMA-ES and population-based REINFORCE can find better solutions for the WANN architectures evolved for MNIST, suggesting that the various activations proposed by the WANN search algorithm may have produced optimization landscapes that are more difficult for gradient-based methods to traverse compared to standard ReLU-based deep network architectures.

Fine-Tuning을 Backpropagation을 통해 했을때 성능이 별로였다

- REINFORCE라는 방법으로 업데이트
- 다양한 Activation Function을 사용하는 와중에 Gradient Vanishing이 일어나서?
- Input에서 Output으로 가는 Route의 길이가 제각각인 것도 학습을 방해하지 않을까?

https://arxiv.org/pdf/1906.04358.pdf

WANN 성능 평가 & 실험계획

WANN	Test Accuracy
Random Weight	*82.0% ± 18.7%*
Ensemble Weights	*91.6%*
Tuned Weight	*91.9%*
Trained Weights	*94.2%*

ANN	Test Accuracy
Linear Regression	*91.6%* [62]
Two-Layer CNN	*99.3%* [15]

Figure 6: *Classification Accuracy on MNIST.*

Backpropagation으로 Fine-Tuning했을때 얼마나 성능이 떨어질까?

- 제시한 성능은 REINFORCE를 쓰고도 Linear Regression과 크게 차이 나지 않으며 Two-Layer CNN보다 떨어짐
- Backpropagation 했을때는 어느 정도?

실험계획

- 실험에서 제시한 동일한 세팅으로 Backpropagation 후 성능 비교
- ReLU만 사용해서 동일한 Task에 대한 Topology 찾고 Accuracy, # of Params. 비교
- Gradient Vanishing을 크게 일으키는 Activation Function들 제외하고 탐색후 성능 비교

https://arxiv.org/pdf/1906.04358.pdf

WANN 성능 평가 & 실험계획

WANN	Test Accuracy
Random Weight	*82.0% ± 18.7%*
Ensemble Weights	*91.6%*
Tuned Weight	*91.9%*
Trained Weights	*94.2%*

ANN	Test Accuracy
Linear Regression	*91.6%* [62]
Two-Layer CNN	*99.3%* [15]

Figure 6: *Classification Accuracy on MNIST.*

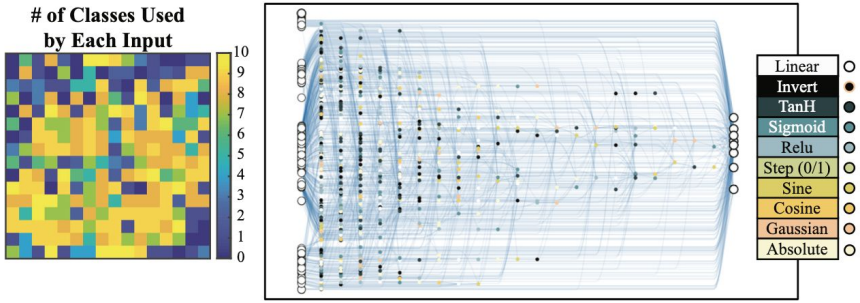


Figure 7: *MNIST classifier network (1849 connections)*

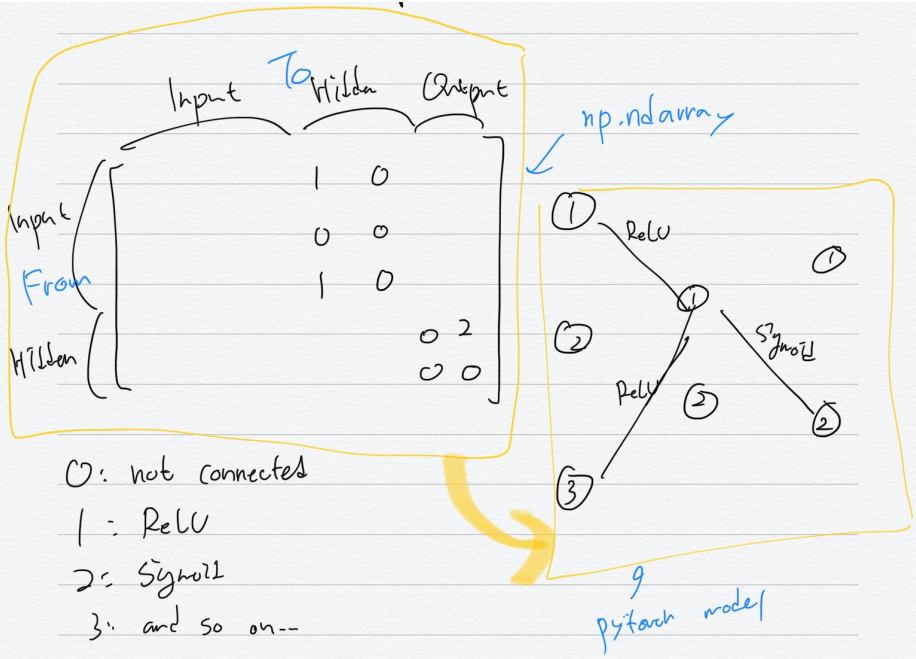
실험계획

- 1849개의 Connection을 가진 Fully connected neural network와는 얼마나 성능 차이가 나는지?

https://arxiv.org/pdf/1906.04358.pdf

WANN PyTorch Implementation

1. Class converts matrix representing graph to PyTorch NN model



2. NEAT Algorithm

It might be already implemented in other packages

If it is, just use it

If not, implement like this(sklearn-genetic)

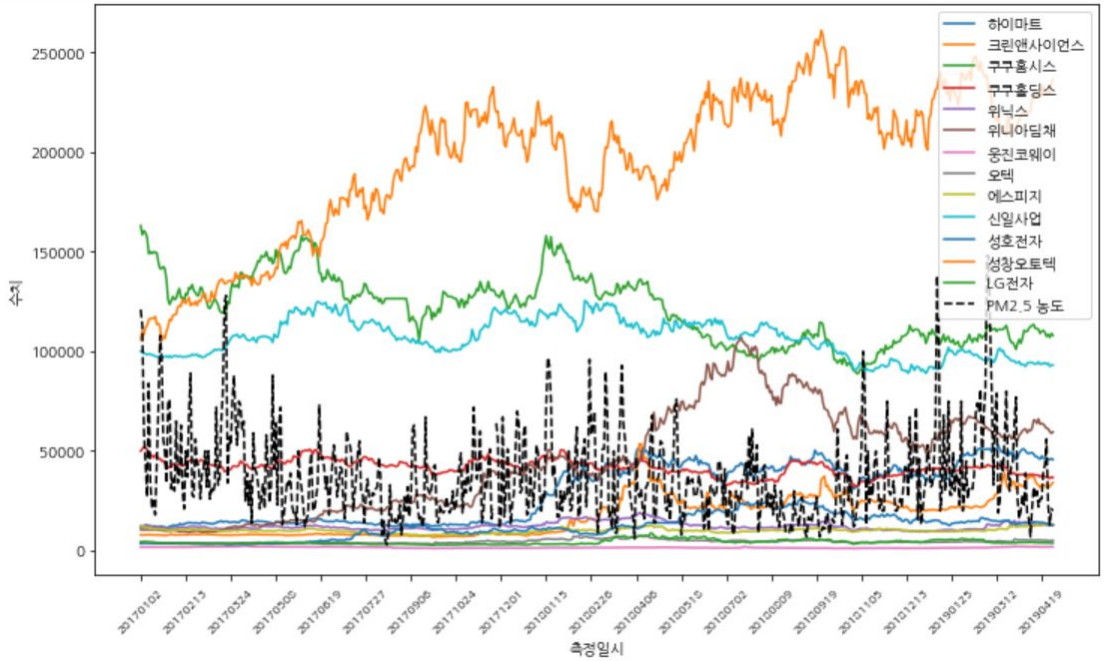
- it should have estimator which caculate the fitness
- GeneticSelectionCV() class
- GeneticSelectionCV().fit() performs search, it saves history and the optimal result inside the class
- and so on .. (need further digging)

https://arxiv.org/pdf/1906.04358.pdf

선정 회사 리스트

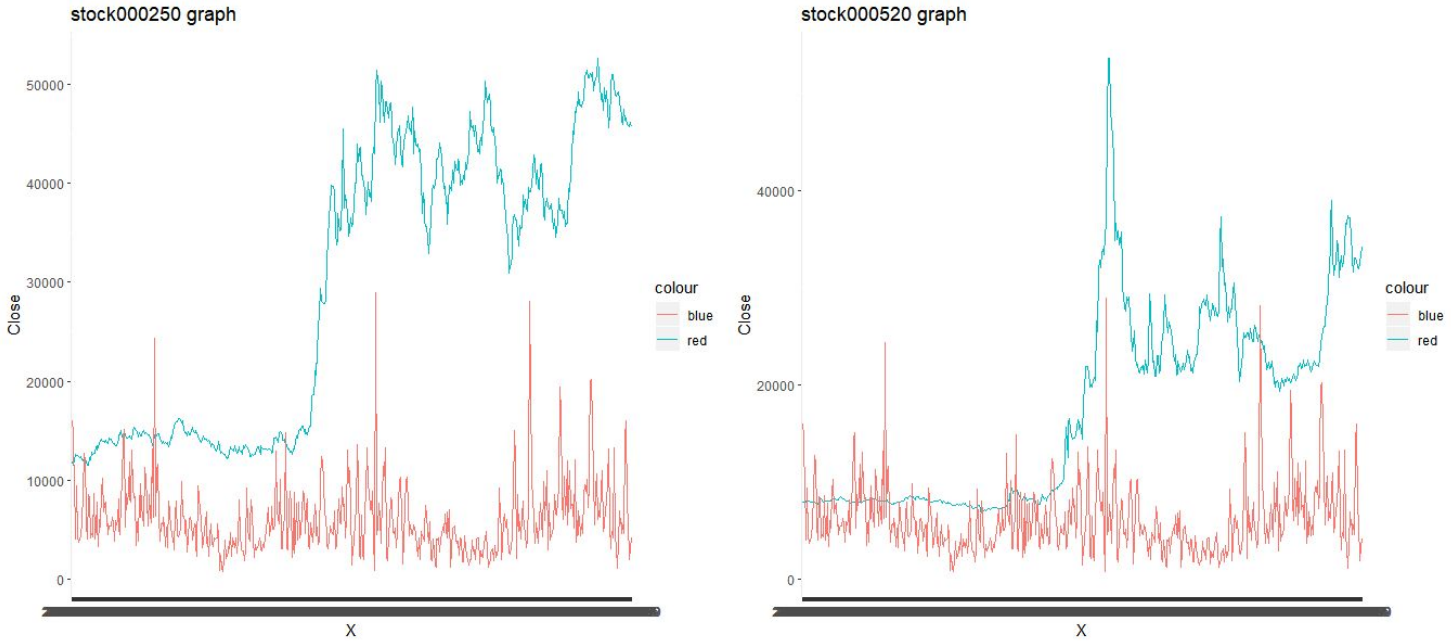
공기청정기	071840	하이마트
공기청정기	045520	크린앤사이언스
공기청정기	284740	쿠쿠홈시스
공기청정기	192400	쿠쿠홀딩스
공기청정기	044340	위닉스
공기청정기	071460	위니아딤채
공기청정기	021240	웅진코웨이
공기청정기	067170	오텍
공기청정기	058610	에스피지
공기청정기	002700	신일사업
공기청정기	043260	성호전자
공기청정기	080470	성창오토텍
공기청정기	066570	LG전자
대기환경설비	053590	한국테크놀로지
대기환경설비	066130	하츠
대기환경설비	086520	에코프로
대기환경설비	007610	선도전기
대기환경설비	148140	비디아이
대기환경설비	069140	누리플랜
대기환경설비	187790	나노
대기환경설비	009440	KC그린홀딩스
마스크	042040	케이피엠테크
마스크	016920	카스
마스크	000640	동아쏘시오홀딩스
수소차	298040	효성중공업
수소차	126880	제이엔케이히터
수소차	095190	이엠코리아
수소차	011320	유니크
수소차	288620	에스퓨얼셀
수소차	033530	세종공업
수소차	011230	삼화전자
수소차	059090	미코
수소차	007720	대명코퍼레이
수소차	012340	뉴인텍
여과설비	052690	한국전력기술
여과설비	022100	포스코ICT
여과설비	034020	두산중공업
여과설비	100840	S&TC
여과설비	119650	KC코트렐
전기차	093370	후성
전기차	009520	포스코엠텍
전기차	108230	톱텍
전기차	005070	코스모신소재

주식 데이터 EDA



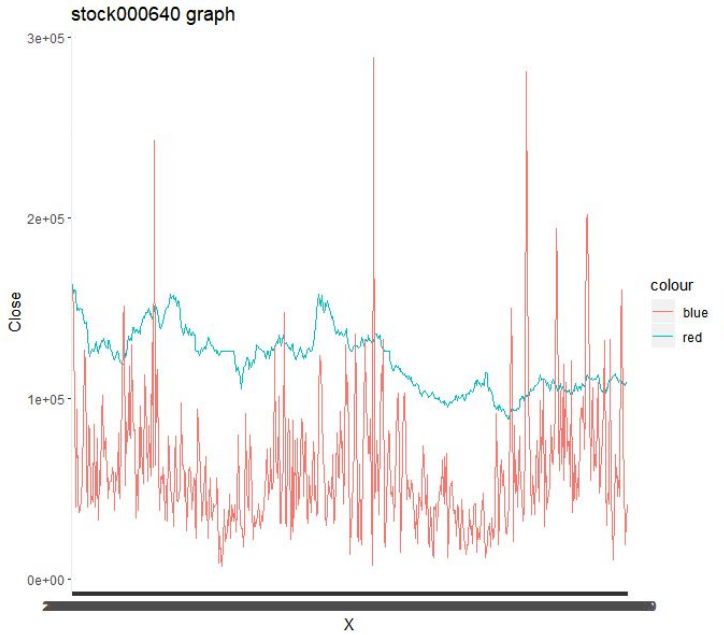
2017 - 1분기부터 2019 -4분기(현재 나온 일자까지) 나타낸 그래프.

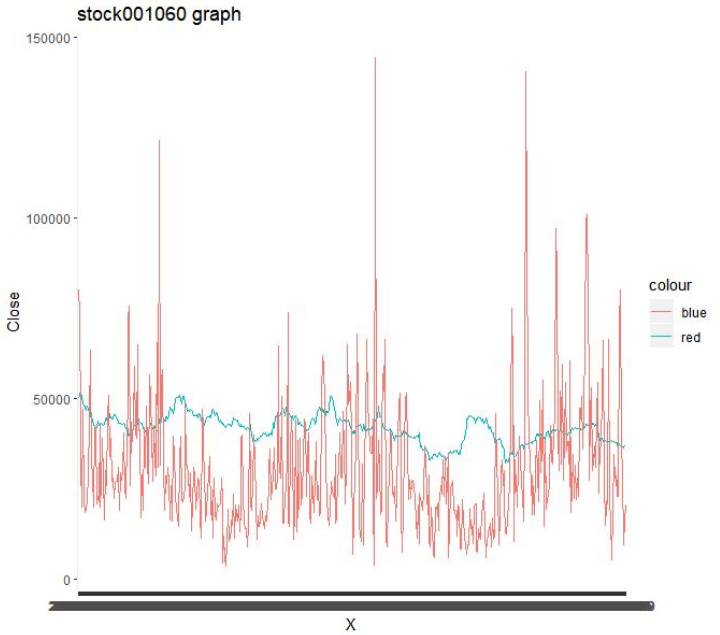
주식 데이터 EDA



삼천당제약(000250), 삼일제약(000520) 2017~현재 미세먼지 및 주가데이터

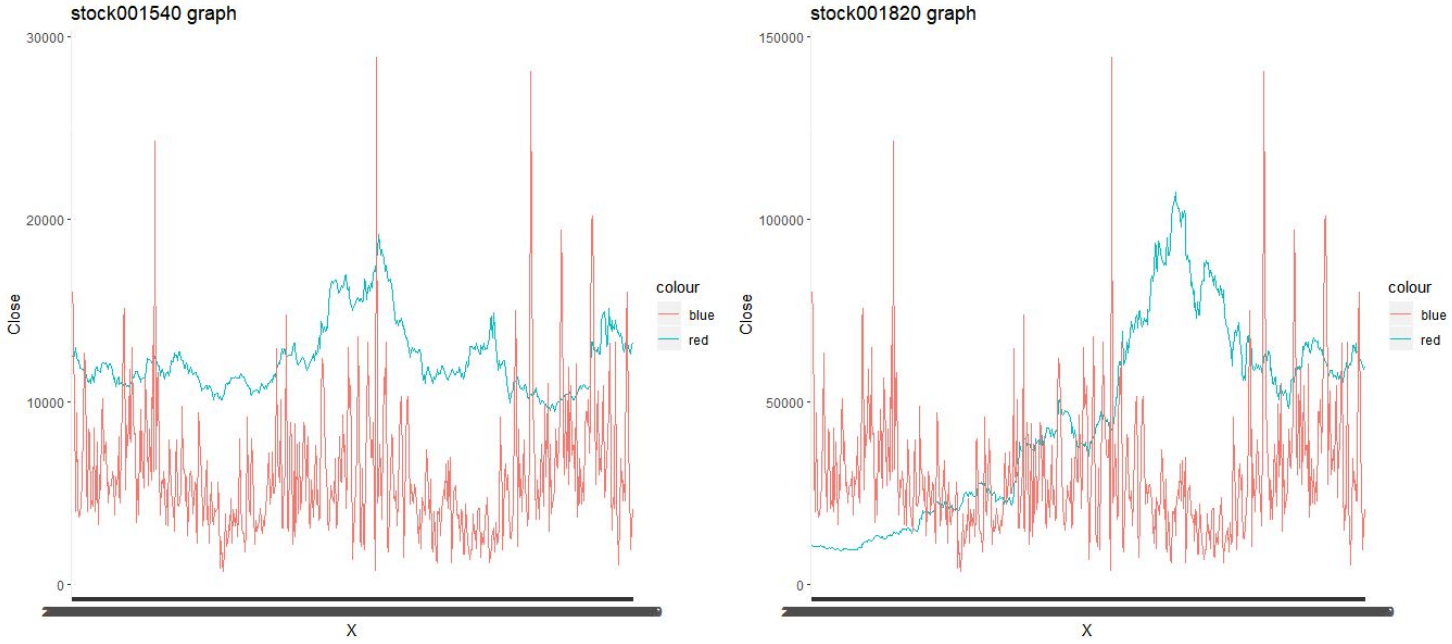
주식 데이터 EDA





동아쏘시오홀딩스(000640), JW중외제약(001060) 2017~현재 미세먼지 및 주가데이터

주식 데이터 EDA

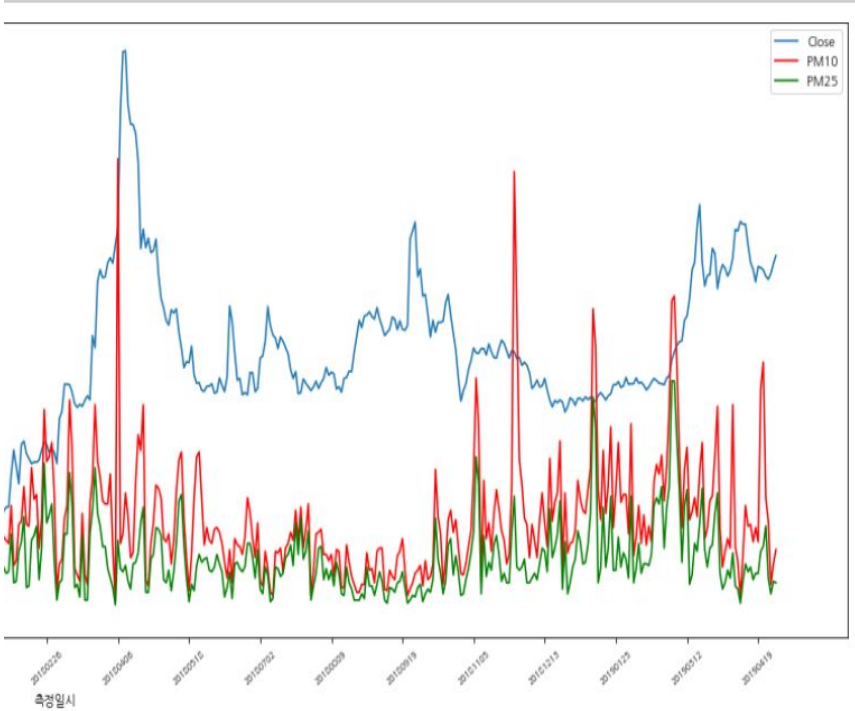


안국약품(001540), 삼화콘덴서(001820) 2017~현재 미세먼지 및 주가데이터

주식 데이터 EDA

- **한계점**: 미세먼지 관련주는 미세먼지가 심한 12~3월에 급등하는 경향이 있음. 하지만 현재는 너무 장기적인 기간으로 EDA를 진행해 추세가 잘 보이지 않는다.
- 미세먼지가 심한 12~3월과, 미세먼지가 심하지 않은 5~10월의 주가를 비교하는 것이 나을 것으로 예측.
- 또한, 미세먼지 관련 주는 **미세먼지 농도 자체보다는, 미세먼지에 대한 사람들의 관심**이 더욱 큰 영향요소라는 생각이 들었다(정부정책 등).
- 따라서 미세먼지 키워드 SNS 분석과 같은 부분이 추가되면 좋을 것 같다.
-

주식 데이터 EDA



In [91]:
```
data1 = data1.rename(columns={data1.columns[0]:'측정일시'})
list_day = []
```

In [92]:
```
for i in tqdm(range(len(data1)),mininterval = 1):
    list_day.append(str(data1['측정일시'][i]))
data1['측정일시'] = list_day
```

```
100%|██████████| 7785/7785 [00:00<00:00, 60984.22it/s]
```

In [93]:
```
merged = pd.merge(df,data1,on='측정일시')
```

In [95]:
```
plt.rc('font', family='NanumGothic')
plt.figure(figsize=(25,8))
plt.plot(merged['측정일시'].ravel(), merged['Close'].ravel(),label = 'Close')
plt.plot(merged['측정일시'].ravel(), merged['PM10'].ravel()*150,c='r',label = 'PM10')
plt.plot(merged['측정일시'].ravel(), merged['PM25'].ravel()*150,c='g',label = 'PM25')
plt.xlabel('측정일시')
plt.xticks(fontsize=7, rotation = 40)
plt.xticks(np.arange(0,568,28))
plt.legend()
plt.show()
```

추가적으로 수집한 데이터

- 미세먼지 농도 상승에 대하여 **반대**의 영향을 받는 주식을 찾고자 하였음
- 이에 **화력발전 관련주** 를 추후 EDA를 통해 비교할 예정(비디아이, 화력발전 관련업체)
- **급식 납품업체**의 경우, **실내에서 유동인구가 잔류**하게 될 수 있다는 의견에 따라 주가 데이터를 구해보려 했으나, 급식 납품만을 주력으로 하는 업체는 비상장 종목이 많아 보류
- 미세먼지 키워드 데이터등이 있다면 찾아볼 예정

Risk Management

1주차 - Risk management

2주차 - Modern portfolio theory – 개별 자산의 고유 리스크 대한 risk management

3주차 - Pair trading – 시장 전체에 적용되는 리스크에 대한 risk management
(주가의 흐름이 유사한 종목을 대상으로 하기 때문에 적합해보임)

Risk Management

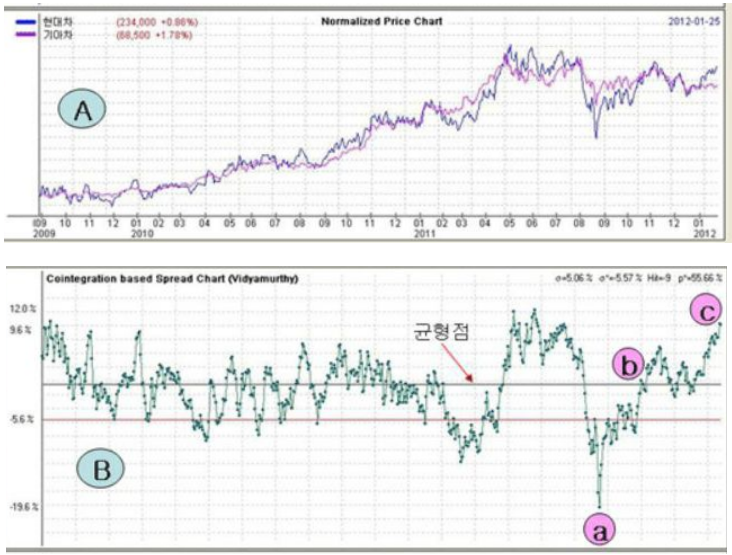
1. Idea

- 장기적으로 볼 때 유사 기업의 주가가 비슷한 경향을 보이지만 단기적으로는, 유사 기업의 주가도 서로 다르게 움직일 때가 있다. 이를 일시적 불균형이라 한다.

- 일시적 불균형 상태에 있는 두 종목을 이용하면 일종의 차익거래 (Arbitrage)를 할 수 있다. 불균형 상태에서 균형 상태로 갈 때 차익을 얻을 수 있는 것이다.

Risk Management

1. Idea(현대차와 기아차의 예시)

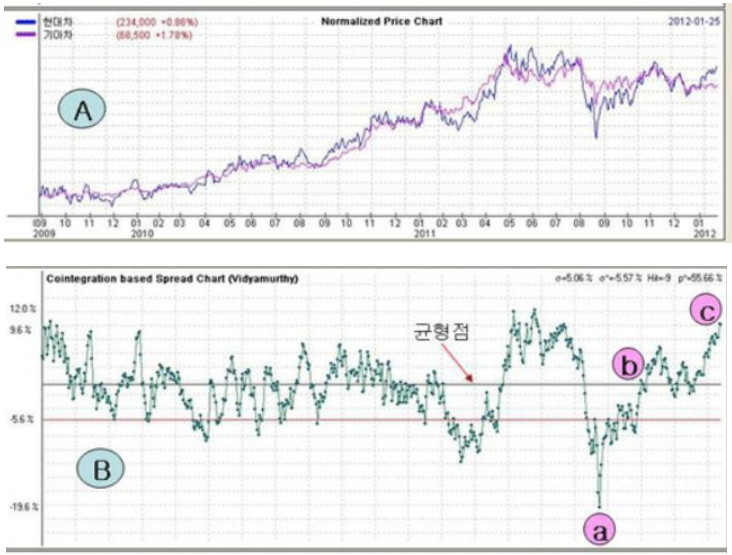


A – 두 종목의 주가차트
-> 주가의 흐름이 유사해보임
B – 두 종목간의 상대적 평가 차트

a – 현대차가 기아차에 비해 저평가
b – 두 종목이 균형상태
c – 현대차가 기아차에 비해 고평가

Risk Management

1. Idea(현대차와 기아차의 예시)



- a지점에서 현대차를 매수하고, 기아차를 매도한 상태에서 b지점까지 기다렸다가 b지점에서 현대차를 매도하고, 기아차를 상환하면 수익을 올릴 수 있다.
- c지점에서 기아차를 매수하고 현대차를 매도한 후 균형점까지 기다렸다가 균형점에서 기아차를 매도하고 현대차를 상환하면 수익을 올릴 수 있다.

□ 이러한 방식으로 수익을 올리고 매수-매도 포지션을 동시에 취하기 때문에 시장전체의 리스크에 대해 중립적이다.

Risk Management

2. In python(페어찾기 위한 함수정의)

```python
# p-value가 지정된 값보다 작은 pair 쌍을 반환한다.
def find_cointegrated_pairs(data):
    n = data.shape[1]
    score_matrix = np.zeros((n, n))
    pvalue_matrix = np.ones((n, n))
    keys = data.keys()
    all_pairs = []
    pairs = []

    # result
    stock1 = []
    stock2 = []
    pvalue_list = []
    check_95 = []
    check_98 = []

    for i in range(n):
        for j in range(i+1, n):
            S1 = data[keys[i]]
            S2 = data[keys[j]]
            result = coint(S1, S2)
            score = result[0]
            pvalue = result[1]
            score_matrix[i, j] = score
            pvalue_matrix[i, j] = pvalue

            if pvalue < 0.05:
                pairs.append((keys[i], keys[j]))
                check_95.append('Y')
            else:
                check_95.append('N')

            if pvalue < 0.02:
                check_98.append('Y')
            else:
                check_98.append('N')

            # result
            stock1.append(keys[i])
            stock2.append(keys[j])
            pvalue_list.append(pvalue)

    pair_pvalue = pd.DataFrame()
    pair_pvalue['s1'] = stock1
    pair_pvalue['s2'] = stock2
    pair_pvalue['pvalue'] = pvalue_list
    pair_pvalue['check_95'] = check_95
    pair_pvalue['check_98'] = check_98

    pair_pvalue.sort_values('pvalue', ascending=True, inplace=True) # ascending=True 오름차순

    return score_matrix, pvalue_matrix, pair_pvalue, pairs
```

Risk Management

2. In python(유의미한 페어 찾기)

```
pair_pvalue.head(5)
```

s1	s2	pvalue	check_95	check_98
CJ	CJ씨푸드	0.015815	Y	Y
CJ	갤럭시아에스엠	0.071204	N	N
CJ씨푸드	갤럭시아에스엠	0.071977	N	N
OCI	SK네트웍스	0.153834	N	N
OCI	SH에너지화학	0.155383	N	N

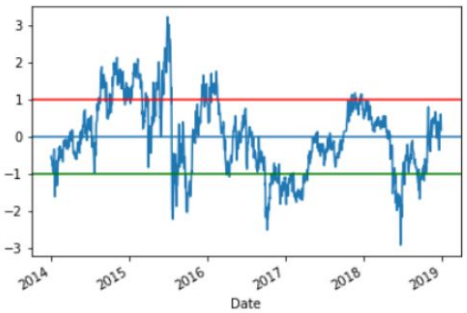
* cj와 cj씨푸드가 유의미한 페어

Risk Management

2. In python(페어의 비율 시각화)

```
def zscore(series):
    return (series - series.mean()) / np.std(series)

zscore(ratios).plot()
plt.axhline(zscore(ratios).mean())
plt.axhline(1.0, color='red')
plt.axhline(-1.0, color='green')
plt.show()
```



S1 = cj의 주가, S2 = cj씨푸드의 주가

Ratio = S1/S2

□균형점으로 회귀하려는 성질을 보임

Risk Management

2. In python(트레이딩 전략 수립)

   - 매수 신호(-1이하) 포착시, S1을 n개 매수하고, S2를 n * Ratio개 매도한다.
   - 매도 신호(+1이상) 포착시, S1을 n개 매도하고, S2를 n * Ratio개 매수한다

Risk Management

2. In python(z- score 기반으로 매도-매수신호 포착)

```
ratios_mavg5 = train.rolling(window=5,
                             center=False).mean()
ratios_mavg60 = train.rolling(window=60,
                              center=False).mean()
std_60 = train.rolling(window=60,
                       center=False).std()
zscore_60_5 = (ratios_mavg5 - ratios_mavg60)/std_60
# Plot the ratios and buy and sell signals from z score
plt.figure(figsize=(15,7))
train[60:].plot()
buy = train.copy()
sell = train.copy()
buy[zscore_60_5>-1] = 0
sell[zscore_60_5<1] = 0
buy[60:].plot(color='g', linestyle='None', marker='^')
sell[60:].plot(color='r', linestyle='None', marker='^')
x1,x2,y1,y2 = plt.axis()
plt.axis((x1,x2,ratios.min(),ratios.max()))
plt.legend(['Ratio', 'Buy Signal', 'Sell Signal'])
plt.show()
```