Algorithm Trading

Week 2

KU-BIG


Algorithm Trading

Timeline

내용	참고	실명의 팀원	2019.10.31(목)	2019.11.8(금)	2019.11.14(목)	2019.11.21(목)	2019.11.28(목)	2019.12.5(목)
데이터 수집 (대체 데이터)	에어코리아 + 빅콘 팀한테 데이터 구걸							
업종 선정	회사 리스트 뽑기 + 나름의 이유	이은진, 김민석						
데이터 수집(주가 데이터 인프라)	회사 리스트 정해져야 가능	이세희, 김미라, 유현우						
선정된 업종 검증	뽑힌 회사 리스트로 EDA	이은진, 김민석						
Backtesting Meth	패키지(zipline) 사용방법, 여러 method 정리해서 공유	유현우, 양수형, 이세희, 김민석						
Risk Managemer	여러 방법, 개념 정리해서 공유	현예성, 유현우						
WANN + 문제 정의	논문읽기, 선행논문 있으면 간단하게 정리, 깃허브 한번 보고 듀토리얼까지?	이세희, 김미라, 양수형, 유현우						
RL	보류							
모델 검증	백테스팅							
...								

Backtesting Method

Backtesting.py tutorial

- backtesting을 위한 python package
- https://kernc.github.io/backtesting.py/doc/examples/Quick%20Start%20User%20Guide.html
- 설치 방법 : !pip3 install backtesting
- backtesting.test에 내장된 google 데이터를 이용해서 test를 진행.

```
from backtesting.test import GOOG

GOOG.tail()
```

	Open	High	Low	Close	Volume
2013-02-25	802.3	808.41	790.49	790.77	2303900
2013-02-26	795.0	795.95	784.40	790.13	2202500
2013-02-27	794.8	804.75	791.11	799.78	2026100
2013-02-28	801.1	806.99	801.03	801.20	2265800
2013-03-01	797.8	807.14	796.15	806.19	2175400

Backtesting Method

Backtesting.py tutorial

- 간단한 Simple Moving Average cross-over 전략을 사용해서 backtest를 진행.
- 내장된 전략을 import 하거나 직접 함수를 만들어 사용하는 것도 가능하다.

```python
import pandas as pd

def SMA(values, n):
    """
    Return simple moving average of 'values', at
    each step taking into account 'n' previous values
    """
    return pd.Series(values).rolling(n).mean()
```

```python
# above function is the same as import SMA
from backtesting.test import SMA
```

Backtesting Method

Backtesting.py tutorial

init() : 전략이 시작하기 전에 처음으로 호출된다.

next() : backtest instance에서 호출된다. 즉, 각 data frame row인 포인트마다 호출되며 시뮬레이션과 new full candlestick bar 결과를 포함.

backtesting.lib.crossover() :
`def next(self): if (self.sma1[-2] < self.sma2[-2] and self.sma1[-1] > self.sma2[-1]): self.buy() elif (self.sma1[-2] > self.sma2[-2] and self.sma1[-1] < self.sma2[-1]): self.sell()` 를 정리한 함수를 대신 사용한다.

```python
from backtesting import Strategy
from backtesting.lib import crossover

class SmaCross(Strategy):
  # Define the two MA lags as *class variables*
  # for later optimization
  n1 = 10
  n2 = 20

  def init(self):
    # precompute two moving averages
    self.sma1 = self.I(SMA, self.data.Close, self.n1)  #indicator wrapped with
    self.sma2 = self.I(SMA, self.data.Close, self.n2)

  def next(self):
    # if sma1 crosses above sma2, buy the asset
    if crossover(self.sma1, self.sma2):
      self.buy()

    #else, if sma1 crosses below sma2, sell it
    elif crossover(self.sma2, self.sma1):
      self.sell()
```

Backtesting Method

Backtesting.py tutorial

- Backtest() 함수를 사용해서 backtesting을 진행할 수 있다.
- cash, commission을 조정할 수 있으며 사용하는 전략 역시 변경 가능하다.
- 간단한 사용법과 더불어 다양한 backtest에 사용되는 정보를 제공한다.

```
from backtesting import Backtest

bt = Backtest(GOOG, SmaCross, cash=10000, commission=.002)
# begin with 10,000 units of cash
# set broker's commission to realistic 0.2%
bt.run()  #pandas Series of simulation results and statistics associated with
```

```
Start                     2004-08-19 00:00:00
End                       2013-03-01 00:00:00
Duration                   3116 days 00:00:00
Exposure [%]                          94.2875
Equity Final [$]                      69665.1
Equity Peak [$]                       69722.1
Return [%]                             596.651
Buy & Hold Return [%]                  703.458
Max. Drawdown [%]                     -33.6059
Avg. Drawdown [%]                     -5.67842
Max. Drawdown Duration      689 days 00:00:00
Avg. Drawdown Duration       41 days 00:00:00
# Trades                                   93
Win Rate [%]                          53.7634
Best Trade [%]                        56.9786
Worst Trade [%]                      -17.0259
Avg. Trade [%]                        2.44454
```

Backtesting Method

Backtesting.py tutorial

- Simple moving average crossover 전략으로 9년을 투자한 결과 700%의 Return, maximal drawdown 33%, longest drawdown period 는 거의 2년의 결과를 얻을 수 있었다.
- 비교적 간단한 코딩 방법과 시각화가 가능한 패키지.

Backtesting Method

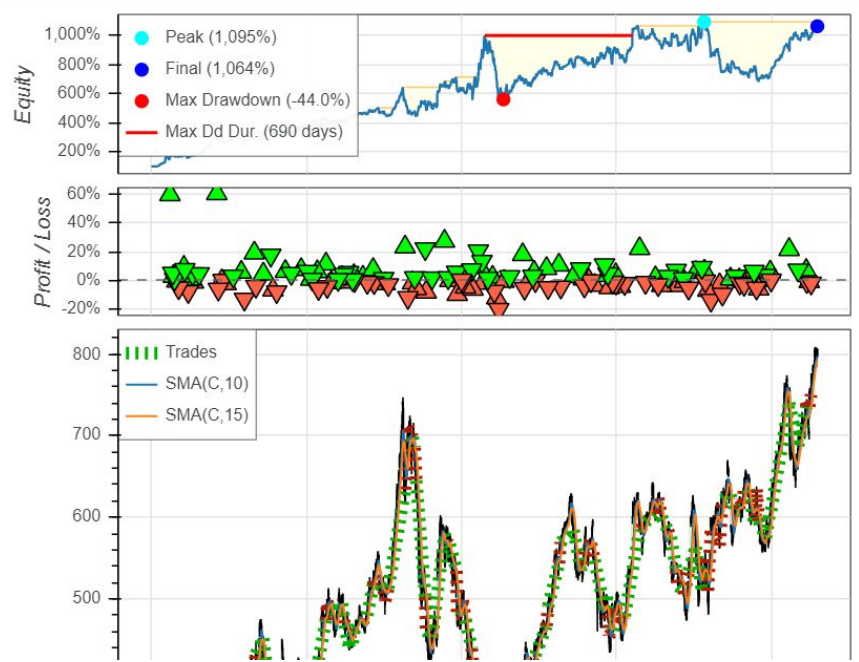
Backtesting.py tutorial

- Optimization 역시 무척 간단하다.
- lag parameter optimization : Backtest.optimize() 함수를 사용.
- 1000%의 return, 44%의 Max Drawdown, 690일의 Max Drawdown duration.

```
%%time

stats = bt.optimize(n1 = range(5, 30, 5),
                    n2 = range(10, 70, 5),
                    maximize = 'Equity Final [$]',
                    constraint = lambda p: p.n1 < p.n2)
```

```
bt.plot()
```



Backtesting Method

Backtrader _tutorial

- backtesting을 위한 python package
- https://ntguardian.wordpress.com/2017/06/12/getting-started-with-backtrader/
- 설치 방법 : !pip3 install backtrader

- 특징) 사용자가 trading strategy, broker, sizers등을 설정 가능하다. 자율성이 보장되는 backtesting package.

```
class SMAC(bt.Strategy):
  params = {'fast':20, 'slow':50, #the windows for both fast and slow moving average
            'optim':False, 'optim_fs':(20, 50)} #used for optimization. first # = fast MA's window
                                                # second in tuple : slow MA's window
  def __init__(self):
    self.fastma = dict()
    self.slowma = dict()
    self.regime = dict()

    self._addobserver(True, bt.observers.BuySell)

    if self.params.optim:
      self.params.fast, self.params.slow = self.params.optim_fs

    if self.params.fast > self.params.slow:
      raise ValueError(
          "A SMAC strategy cannot have the fast moving average's whindow"+#
          "greater than the slow moving average window"
      )

    for d in self.getdatanames():
      #the moving averages
      self.fastma[d] = btind.SimpleMovingAverage(self.getdatabyname(d),
                                                 period=self.params.fast,
                                                 plotname="FastMA: "+ d)
      self.slowma[d] = btind.SimpleMovingAverage(self.getdatabyname(d)
```

Backtesting Method

Backtrader _tutorial

- Trading strategy를 규정한 이후에 Sizer라고 불리는 sizing object를 생성해서 얼마나 구매/판매 할 것인지 결정한다.

```
class PropSizer(bt.Sizer):
  params = {'prop':0.1, 'batch':100}

  def _getsizing(self, comminfo, cash, data, isbuy):
    if isbuy: #buying
      target = self.broker.getvalue() * self.params.prop  #ideal total value
      price = data.close[0]
      shares_ideal = target / price  #How many shares are needed to get targe
      batches = int(shares_ideal / self.params.batch)
      shares = batches * self.params.batch

      if shares * price > cash :
        return 0  #Not enough money for this trade

      else:
        return shares
    else:
      return self.broker.getposition(data).size  #clear the position
```

Backtesting Method

Backtrader _tutorial

- 직접 test하기 위해서는 Cerebro 객체를 생성한다.

- Cerebro 객체는 backtest와 분석의 실행 역할을 하는 객체이다.

```
cerebro = bt.Cerebro(stdstats=False)
```

```
cerebro.broker.set_cash(1000000)  #set our starting cash
cerebro.broker.setcommission(0.02)
```

이제 Cerebro 객체에 넣을 데이터가 필요하다. pandas DataFrame, CSV files, databases, live data streams 모두 feed 가능하다.

```
[ ]  start = datetime.datetime(2010, 1, 1)
     end = datetime.datetime(2016, 10, 31)
     is_first = True

     symbols = ["AAPL", "GOOG", "AMZN", "NVDA"]
     plot_symbols = ["AAPL", "GOOG", "NVDA"]

     for s in symbols:
       data = bt.feeds.YahooFinanceData(dataname=s, fromdate = start, todate=end)
       if s in plot_symbols:
         if is_first:
           data_main_plot = data
           is_first = False
         else:
           data.plotinfo.plotmaster = data_main_plot
       else:
         data.plotinfo.plot = False
       cerebro.adddata(data)
```

Backtesting Method

Backtrader _tutorial

- cerebro에 observer, strategy, sizer를 추가해서 simulation 실행.

```
cerebro.addobserver(AcctValue)
cerebro.addstrategy(SMAC)
cerebro.addsizer(PropSizer)
```

```
cerebro.broker.getvalue()
```

```
1000000
```

```
cerebro.run()
```

```
[<__main__.SMAC at 0x7f6a28826a90>]
```

```
cerebro.plot(iplot=True, volume=Falsec)
```

```
1 cerebro.plot(iplot=True, volume=False)
```

https://ntguardian.wordpress.com/2017/06/12/getting-started-with-backtrader/

Backtesting Method

Backtrader _tutorial

- simulation 결과.

- 역시 optimization은 가능했지만,
  backtrading.py보다는 다소 복잡하다.

- backtesting methods를 직접 생성해서
  사용 가능하다.

```
cerebro.broker.getvalue()
```

```
1139128.4599999995
```

요약하자면, Backtrader 패키지는 사용자의 자율성이 강화되어있으며,
backtesting.py 패키지는 코딩을 많이 하지 않아도 필요한 모든
backtrading methods value를 자동으로 return한다.

Risk Management

MPT(Modern Portfolio Theory)

: 투자자가 주어진 위험에 대해 자신의 기대수익을 극대화하기 위한 이론

(일종의 분산투자 이론)

Risk Management

1. 가정

- 투자자는 위험회피성향을 가지고,
  기대효용 극대화를 목표
- 거래 비용과 세금은 고려 사항에서 제외
- 모든 투자자는 투자에 필요한 정보는 모두
  동등하게 접근할 수 있음
- 평균 분산 기준: 기대 수익은 평균으로
  측정하며, 위험은 분산으로 측정

Risk Management

2. 수익과 위험

- 포트폴리오의 기대 수익
  :기대 수익은 각각의 주식의 비중과 수익률을 곱한 것의 합과 같다.

$$E[R] = \sum_{i-1}^{n} R_i P_i$$

R_i is the return in scenario i;
P_i is the probability for the return R_i in scenario i; and
n is the number of scenarios.

- 포트폴리오의 위험(분산)

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_i \sigma_j p_{ij}$$

행렬로 표현하면 아래와 같다.
$\sigma_p^2 = w^T \cdot \Sigma \cdot w$
$\Sigma =$ covariance matrix of assets

Risk Management

3. 포트폴리오의 평가

- 샤프지수(Share ratio)

$$\text{Sharpe Ratio} = \frac{R_p R_{rf}}{\sigma_p}$$

R_p = Expected portfolio/asset return
R_{rf} = Risk-free rate of return
σ = Portfolio/asset standard deviation

- 변동성(Volatility)
  변동성은 포트폴리오의 위험(분산)에 대한 표준편차 값이다.

Risk Management

4. 포트폴리오의 최적화

- 목표 : 샤프지수를 최대화 하거나 변동성을 최소화하는 등의 조건들을 만족하는 자산들의 조합을 찾는 것

Risk Management

5. In python

- 기대수익과 위험 계산

```
returns = prices.pct_change()

# mean daily return and covariance of daily returns
mean_daily_returns = returns.mean()
cov_matrix = returns.cov()

# portfolio weights
weights = np.asarray([0.4,0.2,0.1,0.1,0.1,0.1])

portfolio_return = round(np.sum(mean_daily_returns * weights) * 252,2)
portfolio_std_dev = round(np.sqrt(np.dot(weights.T,np.dot(cov_matrix, weights))) * np.sqrt(252),

print("Expected annualised return: " + str(portfolio_return))
print("Volatility: " + str(portfolio_std_dev))


Expected annualized return: 0.23
Volatility: 0.18
```

Risk Management

5. In python

- 샤프지수를 최대로 하는 자산들의 조합을 찾음

```
# Expected returns and sample covariance
mu = expected_returns.mean_historical_return(prices)
S = risk_models.sample_cov(prices)

# Optimise portfolio for maximum Sharpe Ratio
ef = EfficientFrontier(mu, S)
raw_weights = ef.max_sharpe()
cleaned_weights = ef.clean_weights()
print(cleaned_weights)
ef.portfolio_performance(verbose=True)
{'msft': 0.46583, 'aapl': 0.0, 'amzn': 0.28355, 'fb': 0.0, 'brkb': 0.0368, 'jnj': 0.21382}


Expected annual return: 25.3%
Annual volatility: 18.9%
Sharpe Ratio: 1.23
(0.25302149406757357, 0.18918176710946344, 1.2317333621941686)
```