Excel vs Q_eng vs Pypfopt lib

JE KIM

2020-06-22

1. 개요

세 가지의 엔진에 관하여 1. 결괏값을 구하는 과정을 분석하고 2. 최종 비중을 비교하였습니다. 그 결과, 세 가지 엔진 모두 ±1%의 차이로 비슷한 비중 값을 보였습니다.

IRA와 머니포트에 탑재할 엔진으로는 Q_eng이 적합하다고 생각되며 그 이유는, q_eng에 우리의 저작권이 있고 cvxopt 최적화 방식을 사용하였기 때문입니다.

아래의 보고서는 Q_eng 을 기준으로 작성하였습니다.

2. Q_eng 외의 포트폴리오

2.1 Excel

Q_eng 의 바탕이 되는 포트폴리오입니다. 기대 수익률은 각 종가 데이터를 로그 수익률로 변환 후, 평균을 낸 값을 연 환산하여 사용합니다.

자산배분 포트폴리오 이론에 따르면 실무적으로는 Excel의 solver 최적화 프로그래밍 기법을 활용하지만 IRA, 머니포트에 탑재하기 위해 코드화한 것이 Q_eng입니다.

2.2 Pypfopt

Efficient Frontier, Black-Litterman 등을 다룬 포트폴리오 라이브러리입니다. 시각화한 Hierarchical Risk Parity(HRP)와 The Critical Line Algorithm(CLA) 또한 도출할 수 있습니다.

더 많은 문제 클래스를 처리할 수 있는 cvxopt가 아닌 cvxpy가 사용되었습니다.

2.3 그 외의 라이브러리

(1) FinanceDataReader, quantmod

: Investing.com에서 필요한 금융자산의 가격 데이터를 불러옵니다.

(2) Pyfolio, zipline, backtrader

: 백테스팅 성과분석 tool입니다.

(3) QuantLib

: 금융공학 라이브러리입니다.

3. CVXOPT

2차 프로그래밍(quadratic programming)에 대한 인터페이스 제공합니다.

아래는 cvxopt의 공식입니다.

$$cvxopt.solvers.qp(P,q[,G,h[,A,b[,solver[,inivals]]]])$$

solve

$$\min_x \frac{1}{2} x^T P x + q^T x$$

$$\begin{aligned} \textit{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$

Cvxopt의 solver에는 P, Q, G, h, A, b에 해당하는 인수를 넣어야 합니다.

Efficient Frontier의 공식을 변환하여 max sharpe 가 목적인 인수를 대입해 solver를 실행하여 비중을 도출합니다.

$$\begin{aligned} \underset{w}{\text{minimise}} \quad & w^T \Sigma w \\ \text{subject to} \quad & w^T \mu \geq \mu^* \\ & w^T \mathbf{1} = 1 \\ & w_i \geq 0 \end{aligned}$$

μ : 기대 수익률
Σ : 공분산 행렬
wTΣw : 리스크, 분산
wTμ : 가중치 * 수익률
μ* : 목표 수익률 or 무위험 수익률
w : 가중치 벡터

P : Σ
q : 0
-w + lb <= 0
w – ub <= 0 이므로
h : 0
mu_hat@x =1 으로
A : mu_hat의 전치행렬
b : 1

4. 실험

4.1 로그 수익률과 일반 수익률

Numpy 패키지에 내장된 함수인 pct_change()은 일반 수익률을 구하는 메소드입니다. 공식은 *(i+1일의 종가 데이터 - i일의 종가 데이터) / i일의 종가 데이터* 입니다.

로그 수익률은 특정 일반 수익률이 되도록 만들어주는 무한복리약정이자율이며 *ln(i+1일의 종가 데이터 / i일의 종가 데이터)* 공식으로 구해집니다. 통계적 분석에 유용하므로 금융에서 수익률을 계산할 때 주로 로그 수익률을 사용합니다.

4.2 환율 적용

포트폴리오 사이에서 결괏값 일치가 안 되었던 원인 중에 하나로 기본 초기화가 해외 주식으로 설정되어 있어 결괏값이 다르게 도출되었습니다.

결괏값 비교는 국내 주식으로 설정을 바꾸어 준 후, 진행하였습니다.

4.3 로그 수익률 index

코드의 로그 수익률을 구하는 공식 부분에서 index가 범위 밖으로 지정되어 잘못된 값이 저장되는 부분을 수정하였습니다.

포트폴리오들의 결괏값 일치가 안된 원인 중에 하나라고 생각됩니다.

4.4 기대 수익률

마지막으로 기대 수익률을 수정하자 포트폴리오들의 결과 비중이 일치했습니다.

평균 수익률과 공분산행렬은 모두 일치했지만 결과 비중은 일치하지 않아 많은 시행착오가 있었습니다.

opt_sharpe에 대입되는 값은 평균 수익률이 아니라 평균 수익률을 연환산한 기대수익률입니다. 코드 상에서 주석 처리된 평균 수익률의 연환산 부분을 주석 해제한 뒤, 그 값을 배열로 변형하여 opt_sharpe에 대입하니 최종적으로 엑셀과 일치하는 비중이 출력되었습니다.

5. 실험 결과

		비중		
excel	SPY	29.57%	포트폴리오기대수익률	0.030647
	GLD	21.19%	포트폴리오분산	0.000205
	UUP	49.24%	포트폴리오기대샤프비율	0.618121
	IWM	0.00%	포트 연환산 표준편차	0.049581
q_eng	SPY	29.829	포트폴리오기대수익률	0.030898
	GLD	21.411	포트폴리오분산	0.000208
	UUP	48.76	포트폴리오기대샤프비율	0.618033
	IWM	0	포트 연환산 표준편차	0.049993
pypfopt	SPY	0.29147	Expected annual return	79.10%
	GLD	0.21268	포트폴리오분산	
	UUP	0.49585	Sharpe Ratio	3.46
	IWM	0	Annual volatility	22.30%
excel	AMD	73.24%	포트폴리오기대수익률	0.664659
	AAPL	26.76%	포트폴리오분산	0.229508
	LMT	0.00%	포트폴리오기대샤프비율	1.266512
			포트 연환산 표준편차	0.524795
q_eng	AMD	73.503	포트폴리오기대수익률	0.666254
	AAPL	26.497	포트폴리오분산	0.023061
	LMT	0	포트폴리오기대샤프비율	1.26651
			포트 연환산 표준편차	0.526055
pypfopt	AMD	0.72372	Expected annual return	564.00%
	AAPL	0.276	포트폴리오분산	
	LMT	0	Sharpe Ratio	4.57
			Annual volatility	122.90%

6. 결론

Q_eng은 엑셀을 그대로 구현한 것이기 때문에 비중에 미세한 차이가 생기는 이유는 최적화 방식의 차이 때문입니다. 그 차이는 ±1% 내이며 일의 자리까지 반올림하였을 때에는 같은 비중을 보입니다. Pypfopt 또한 큰 차이를 보이지는 않았지만 Q_eng이 홍광진 이사님께서 직접 작성한 코드이기 때문에 프로그래밍 의도에 대해 완벽하게 파악할 수 있어 코드 수정이 용이한 Q_eng이 적합할 것으로 보입니다. 게다가 성능이 입증된 cvxopt 모듈을 사용하고 있어 global optimum을 찾을 수 있습니다.

기대수익률을 어느 값으로 설정했느냐에 따라 결과 비중값에 많은 차이가 있었습니다. 모델링 성능을 더욱 향상시키기 위한 방안으로는 기대수익률과 리스크 값에 변화를 주는 방법이 있습니다. 그 중, 머신러닝 또는 딥러닝을 통해 기대수익률을 예측하는 방식을 선택해 연구할 예정입니다.