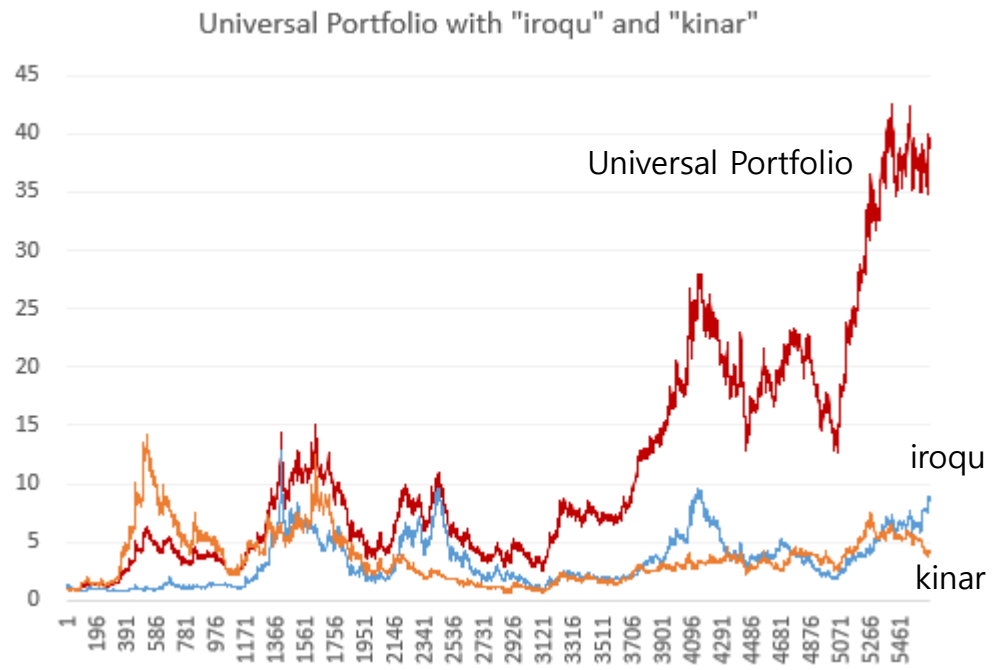
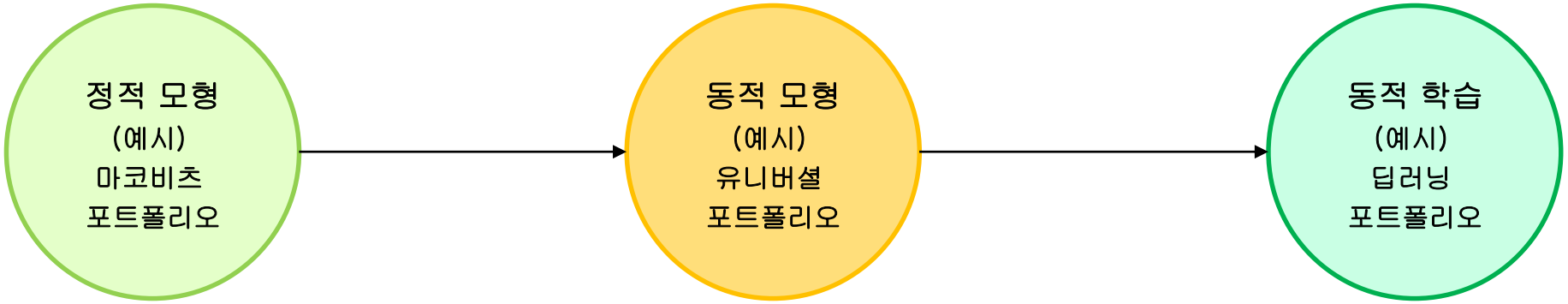


딥러닝을 활용한 동적 포트폴리오 구축 방안

포트폴리오 최적화 (Portfolio optimization, Dynamic Asset Allocation)

- 자산 관리의 핵심인 포트폴리오 최적화는 정적 모형부터 동적 모형, 딥러닝 기술의 활용까지 현재도 활발히 연구되고 있음.



- 1952년 마코비츠의 Portfolio Selection Theory
- 정적인 모형으로 위험 자산들의 수익률과 변동성으로 자산들의 최적 투자 비중을 산출함.
- 당시에는 자산들의 관계인 대량 계산이 필요한 공분산 행렬 등 계산이 어려워 실용화되지는 못함.
- 현대는 대량 계산이 용이하므로 현대 포트폴리오 이론으로 발전함.

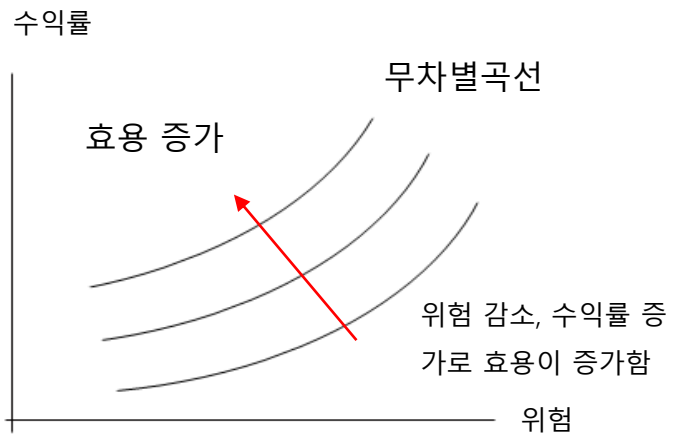
- 1996년 Tomas M. Cover의 Universal Portfolio.
- 동적인 모형으로 위험 자산들로 균등 비중을 적용한 무수히 많은 CRP를 구성하고 (CRP-1, CRP-2, ...), 이 CRP들을 대상으로 포트폴리오를 구성함.
- 이론적으로는 자산들 중 가장 좋은 성과를 보였던 것보다 더 높은 성과를 보이는 포트폴리오로 알려져 있으나, 무한히 많은 CRP들의 계산이 현실적으로 불가함.
- GPU/CUDA 등의 대량 병렬 처리를 이용하여 근사적으로 접근함.

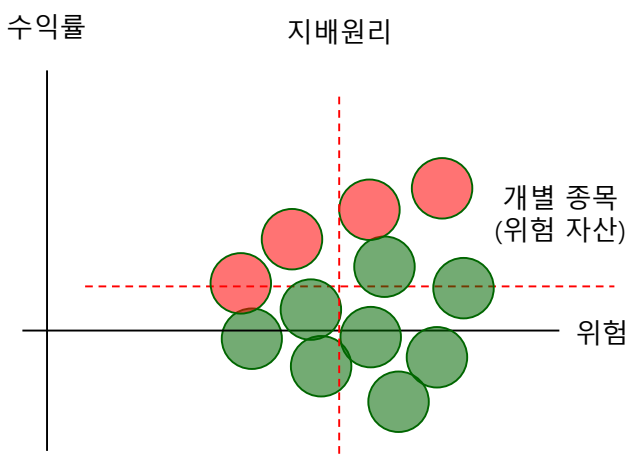
- 모형에 기반한 것이 아니라, 시장 데이터를 학습하여 동적으로 포트폴리오를 구축함.
- 위험 자산들의 시장 데이터를 LSTM/CNN 등의 딥러닝 기술로 학습하여, 과거 데이터를 통해 미래 성과를 예측하여 자산들의 최적 투자 비중을 추정함.
- 현재 가장 관심을 보이는 분야로 로보어드바이저 자산 관리 등에 활용되고 있으며, 향후 발전성이 매우 큰 분야임.

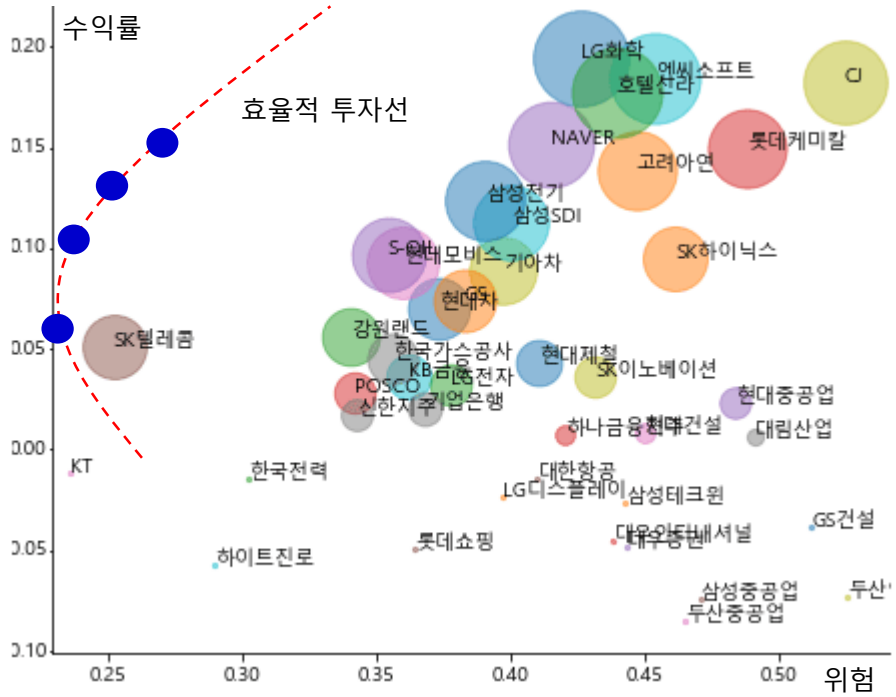
마코비츠의 포트폴리오 이론 (Portfolio Selection Theory, 1952)

- 마코비츠는 1952년 위험 자산들의 효율적 투자선을 제안하고, 최적 포트폴리오 구성 방안을 이론적으로 정립했다.
- 개별 종목의 위험과 수익률을 측정하여 상태공간에 배치하고, 각 종목의 투자 비율을 조정하여 위험대비 수익률이 극대화되는 지점을 찾는다. → 효율적 투자선 (Efficient Frontier)
- 마코비츠의 포트폴리오 이론은 종목 별로 전 조합의 상관 관계를 분석해야 하므로 계산량이 많아 1952년 당시에는 다수의 종목으로 포트폴리오를 구성하기 어려웠지만, 현재는 컴퓨터의 계산 능력이 매우 뛰어나므로 포트폴리오 투자가 다시 주목 받고 있다.
  → 현대 포트폴리오 이론 (Modern Portfolio Theory : MPT)

수익률-위험 상태공간과 효율적 투자선





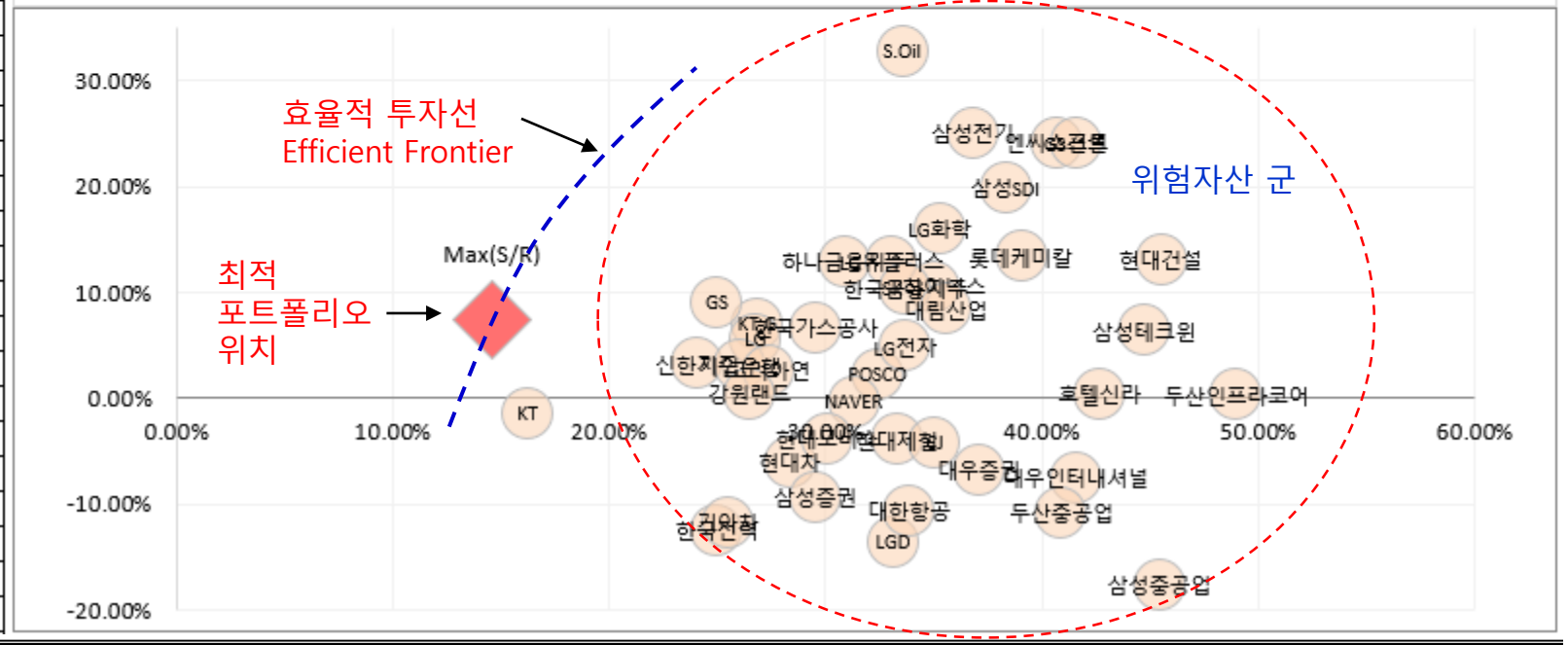


1. 마코비츠의 포트폴리오 최적화

포트폴리오 최적화 예시

No	종목	연간 변동성	연간 수익률	Weight	w ^ 2
1	현대차	28.32%	-5.92%	1.95%	0.0004
2	SK하이닉스	34.99%	10.61%	3.02%	0.0009
3	한국전력	24.96%	-12.42%	1.55%	0.0002
4	POSCO	32.38%	2.34%	2.25%	0.0005
5	NAVER	31.28%	-0.37%	2.34%	0.0005
6	현대모비스	30.06%	-3.80%	2.09%	0.0004
7	신한지주	24.02%	3.29%	2.50%	0.0006
8	기아차	25.46%	-11.65%	1.57%	0.0002
9	LG화학	35.26%	16.12%	3.12%	0.0010
10	LGD	33.10%	-13.54%	1.42%	0.0002
11	KT.G	26.84%	7.02%	2.99%	0.0009
12	LG	26.75%	5.60%	2.61%	0.0007
13	LG전자	33.67%	4.98%	2.57%	0.0007
14	하나금융지주	30.84%	12.99%	3.08%	0.0009
15	KT	16.22%	-1.42%	2.38%	0.0006
16	기업은행	26.01%	3.12%	2.48%	0.0006
17	삼성SDI	38.29%	20.00%	3.51%	0.0012
18	현대제철	33.27%	-3.85%	1.75%	0.0003
19	고려아연	27.36%	2.66%	2.45%	0.0006
20	강원랜드	26.47%	0.50%	2.43%	0.0006
21	롯데케미칼	38.99%	13.40%	2.94%	0.0009
22	S.Oil	33.53%	32.82%	4.30%	0.0019
23	LG유플러스	33.04%	12.97%	3.28%	0.0011
24	삼성중공업	45.40%	-17.57%	0.76%	0.0001
25	현대건설	45.46%	13.09%	2.70%	0.0007
26	한국가스공사	29.52%	6.70%	2.65%	0.0007
27	CJ	35.03%	-4.16%	1.98%	0.0004
28	엔씨소프트	40.65%	24.22%	3.97%	0.0016
29	삼성전기	36.78%	25.08%	3.88%	0.0015
30	GS	24.96%	9.08%	2.83%	0.0008
31	호텔신라	42.64%	0.50%	2.25%	0.0005

R(p)	7.49%	C	50
Vol(p)	14.57%	sum(w)	100.00%
Max(S/R)	0.41	Target	-0.94

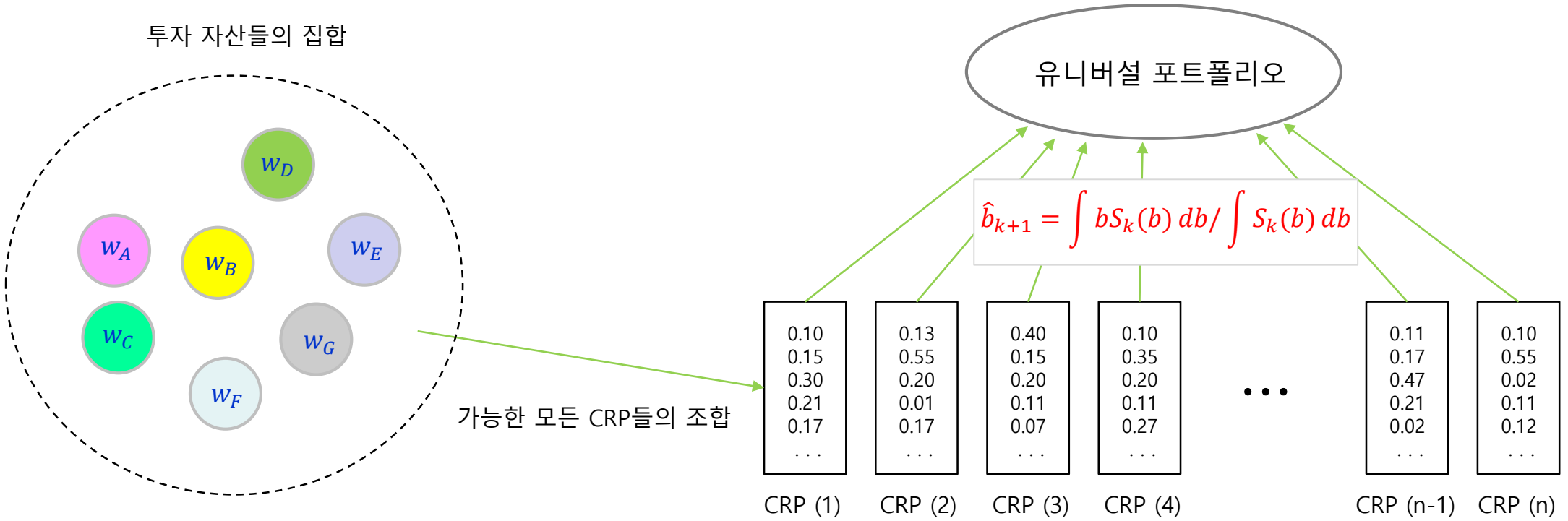


동적 자산 배분과 포트폴리오 리밸런싱 (Rebalancing)

- 최적 포트폴리오 비율대로 초기 투자를 집행해도 이후의 자산 가격들이 변동하기 때문에 최적 비율은 계속 달라진다.
- A와 B자산의 초기 가격이 각각 100원씩이고, 자산 배분 비율을 0.5: 0.5 (50원 : 50원)로 집행했다. 일정 기간 후에 A자산의 가격은 150원으로 상승했고 (+50%), B자산의 가격은 60원으로 하락했다면 (-40%), 현재의 자산은 75원 : 30원 = 71% : 29%로 비율이 변한다. → 최적 비율에서 벗어났음.
- 현재의 비율을 0.5 : 0.5로 유지하려면 A자산을 22.5원 매도하여 B자산을 22.5원 매수하면 52.5원 : 52.5원이 되고 비율은 0.5 : 0.5로 유지된다.
- 이와 같이 자산 가격 변동에 따라 자산 배분 비율을 조절해 주는 것을 포트폴리오 리밸런싱 전략이라 한다.
- 포트폴리오 리밸런싱을 적용하면 일반적으로 위험이 감소하고 투자 성과를 향상 시킬 수 있다.
- 효율적인 리밸런싱을 위해서는 비율 조절 규모, 주기, 거래 비용, 기회 비용, 시장 충격 비용 등의 영향을 고려해야 한다.
- 리밸런싱 주기가 너무 짧으면 거래 비용, 세금, 시장 충격 비용 등이 과도하게 발생하고, 너무 길면 자산 가격 변동 비용 (기회 비용 개념)이 발생한다.
- 리밸런싱은 주기적 (월간, 분기, 반기, 혹은 연간 단위)으로 적용할 수도 있고, 특정 종목의 비율이 임계치 (Threshold)를 초과하면 적용할 수도 있다.
- 또한, 총 거래 비용 (거래 비용 + 기회 비용 + 충격 비용 + …)을 측정하여 비용이 최소가 되는 조건에서 리밸런싱을 적용할 수도 있다.
- 비율 조절 방식의 예시로는 초기에 정한 자산 배분 비율을 계속 유지하는 방법 (Constant Rebalanced Portfolio : CRP), Universal Portfolio 등이 있다.

유니버설 포트폴리오 (Universal Portfolio)

- 유니버설 포트폴리오는 1996년 스탠포드 대학의 토마스 커버 (Thomas M. Cover) 교수가 제안한 것이다.
- 유니버설 포트폴리오는 여러 자산 중 가장 높은 성과를 보인 개별 자산보다 더 높은 성과를 보이는 전략이다 (이론적으로 증명했음).
- 가능한 모든 CRP들의 조합을 대상으로 포트폴리오를 구성한다. 개별 종목들의 포트폴리오가 아닌 모든 CRP들의 포트폴리오 → 유니버설 포트폴리오.
- 유니버설 포트폴리오는 Best CRP (BCRP)를 추구하는 성과를 보인다 (BCRP가 되는 것은 아님).
- 그러나 모든 CRP의 조합은 무한히 크기 때문에 현실적으로는 다수의 종목으로 유니버설 포트폴리오를 구축하기는 어렵다. → 근사적 접근이 필요하다.
- 현실적으로는 모든 CRP 조합이 아닌 이산화된 부분적 CRP 조합으로 근사적 유니버설 포트폴리오를 구축하기도 한다.



- 논문 리뷰 (2) : 유니버설 포트폴리오 (Universal Portfolio)

Universal Portfolio

Thomas M. Cover, Stanford University, October 23, 1996

Abstract

We exhibit an algorithm for portfolio selection that asymptotically outperforms the best stock in the market. Let $X_i = (x_{i1}, x_{i2}, \dots x_{im})$ denotes the performance of the stock market on day i, where x_{ij} is the factor by which the j-th stock increases on day i. Let $B_i = (b_{i1}, b_{i2}, \dots b_{im}), b_{ij} \geq 0, \sum_j b_{ij} = 1$, denote the proportion b_{ij} of wealth invested in the j-th stock on day i. Then $S_n = \prod_{i=1}^{n} B_i^t \cdot X_i$ is the factor by which wealth is increased in n trading days.

Consider as a goal the wealth $S_n^* = \max_B \prod_{i=1}^{n} B_i^t \cdot X_i$ that can be achieved by the best constant rebalanced portfolio chosen after the stock outcomes are revealed. It can be shown that S_n^* exceeds the best stock, the Dow Jones average, and the value line index at time n. In fact, S_n^* usually exceeds these quantities by an exponential factor.

Let $X_1, X_2, \dots$, be an arbitrary sequence of market vectors. It will be shown that the non-anticipating sequence of portfolios $\hat{B}_k = \int B \prod_{i=1}^{k-1} B^t \cdot X_i dB \,/\, \int \prod_{i=1}^{k-1} B^t \cdot X_i dB$ yields wealth $\hat{S}_n = \prod_{i=1}^{n} \hat{B}_i^t \cdot X_i$ such that $\left(\frac{1}{n}\right) ln\left(\frac{S_n^*}{\hat{S}_n}\right) \to 0$, for every bounded sequence $X_1, X_2, \dots$, and, under mild conditions, achieves, ...

유니버설 포트폴리오 모형

We propose a more ambitious goal. To motivate this goal let us consider all constant rebalanced portfolio strategies. Let $X = (x_1, x_2, \dots, x_m)^t \geq 0$ denote a stock market vector for one investment period, where x_i is the price relative for the i-th stock, i.e., the ratio of closing price for stock i. A portfolio $B = (b_1, b_2, \dots, b_m)^t, b_i \geq 0, \sum b_i = 1$, is the proportion of the current wealth invested in each of the m stocks. Thus $S = B^t \cdot X = \sum b_i x_i$, where B and X are considered to be column vectors, is the factor by which wealth increases in one investment period using portfolio B.

Consider an arbitrary (nonrandom) sequence of stock vectors $X_1, X_2, \dots, X_n \in R_+^m$. Here x_{ij} is the price relative of stock j on day i. A constant rebalanced portfolio strategy B achieves wealth

$$S_n(B) = \prod_{i=1}^{n} B^t \cdot X_i \qquad \cdots \quad (1.1)$$

where the initial wealth $S_0(B) = 1$ is normalized to 1. Let

$$S_n^* = \max_B S_n(B) \qquad \cdots \quad (1.2)$$

denote the maximum wealth achievable on the given stock sequence maximized over all constant rebalanced portfolios. Our goal is to achieve S_n^*.

유니버설 포트폴리오 모형

We will be able to show that there is a **"universal"** portfolio strategy $\hat{B}_k$, where $\hat{B}_k$ is based only on the past $X_1, X_2, \dots, X_{k-1}$, that will perform asymptotically as well as the best constant rebalanced portfolio based on foreknowledge of the sequence of price relatives. At first it may seem surprising that the portfolio $\hat{B}_k$ should depend on the past, because the future has no relationship to the past. Indeed the stock sequence is arbitrary, and a malicious nature can structure future X_k's to take advantage of past beliefs as expressed in the portfolio $\hat{B}_k$. Nonetheless the resulting wealth can be made to track S_n^*.

The proposed universal adaptive portfolio strategy is the performance weighted strategy specified by

$$\hat{B}_1 = \left(\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m}\right), \qquad \hat{B}_{k+1} = \frac{\int B \cdot S_k(B) dB}{\int S_k(B) dB}, \qquad S_k(B) = \prod_{i=1}^{k} B^t \cdot X_i \quad \cdots \quad (1.3), (1.4)$$

The wealth $\hat{S}_n$, resulting from the universal portfolio is given by

$$\hat{S}_n = \prod_{k=1}^{n} \hat{B}_k^t \cdot X_k \quad \cdots \quad (1.6)$$

유니버설 포트폴리오 계산 방법

$$S_k = B^t \cdot X_k = \sum b_i x_i \qquad S_k(B) = \prod_{i=1}^{k} B^t \cdot X_i$$

$W_0 \rightarrow$	200								
No	**A**	**B**	x_A	x_B	b_A	b_B	**S**	**S(B)**	**W**
0	100	100			0.5	0.5			
1	110	80	1.100	0.800	0.5	0.5	0.950	0.950	190.0
2	100	110	0.909	1.375	0.5	0.5	1.142	1.085	217.0
3	120	90	1.200	0.818	0.5	0.5	1.009	1.095	219.0

CRP (0.5, 0.5)

CRP (Random)

No	**A**	**B**	x_A	x_B	b_A	b_B	**S**	**S(B)**	**W**
0	100	100			0.15	0.85			
1	110	80	1.100	0.800	0.15	0.85	0.846	0.846	169.28
2	100	110	0.909	1.375	0.15	0.85	1.303	1.103	220.56
3	120	90	1.200	0.818	0.15	0.85	0.877	0.967	193.5

여러 개의 random CRP들

$$\hat{B}_{k+1} = \frac{\int B \cdot S_k(B) dB}{\int S_k(B) dB} \qquad \hat{S}_n = \prod_{k=1}^{n} \hat{B}_k^t \cdot X_k$$

					B_1				B_2				B_3				Universal Portfolio				
No	**A**	**B**	x_A	x_B	b_A	b_B	**S**	**S(B)**	b_A	b_B	**S**	**S(B)**	b_A	b_B	**S**	**S(B)**	b_A	b_B	**S**	**S(B)**	**W**
0	100	100					0.626	0.374			0.375	0.625			0.084	0.916					
1	110	80	1.100	0.800	1.100	0.800	0.626	0.374	0.988	0.988	0.375	0.625	0.913	0.913	0.084	0.916	0.825	0.825	0.378	0.622	0.913
2	100	110	0.909	1.375	0.909	1.375	0.626	0.374	1.083	1.070	0.375	0.625	1.200	1.095	0.084	0.916	1.336	1.102	0.359	0.641	1.208
3	120	90	1.200	0.818	1.200	0.818	0.626	0.374	1.057	1.131	0.375	0.625	0.962	1.053	0.084	0.916	0.850	0.937	0.379	0.621	0.963

2 종목의 유니버설 포트폴리오 구성 예시 : 논문의 8. Examples

We now test the portfolio algorithm on real data. Consider, for example, Iroquois Brands Ltd. and Kin Ark Corp., two stocks chosen for their volatility listed on the New York Stock Exchange. During the 22-year period ending in 1985, Iroquois Brands Ltd. increased in price (adjusted in the usual manner for dividends) by a factor of 8.9151, while Kin Ark increased in price by a factor of 4.1276, as shown in Figure 8.1.

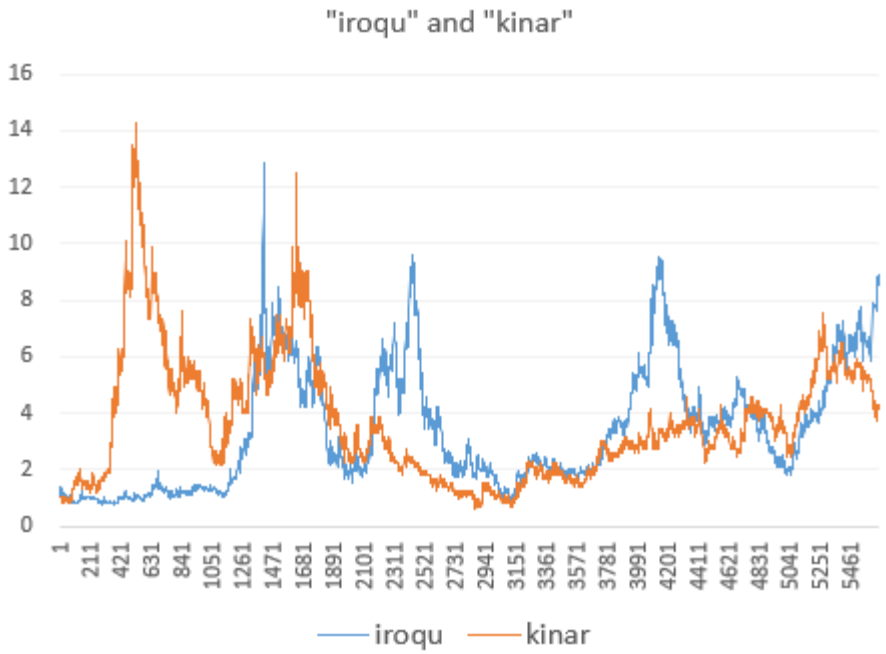
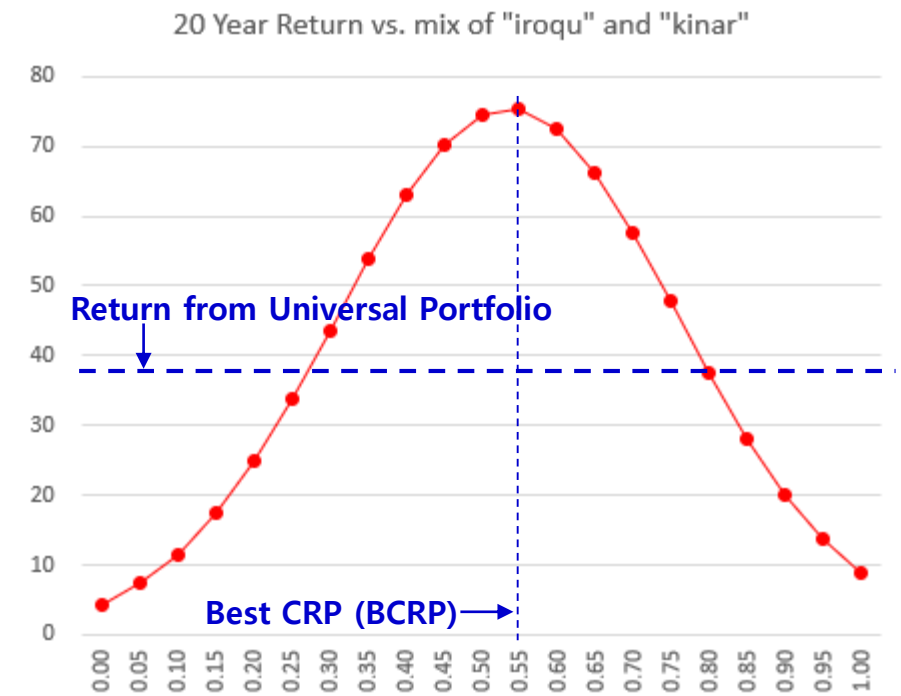


Figure 8.1. Performance of Iroquois brands and Kin Ark.

Table 8.1

b	Sn
0.00	4.422
0.05	7.333
0.10	11.568
0.15	17.364
0.20	24.804
0.25	33.718
0.30	43.621
0.35	53.707
0.40	62.934
0.45	70.187
0.50	74.496
0.55	75.250
0.60	72.337
0.65	66.172
0.70	57.599
0.75	47.704
0.80	37.589
0.85	28.175
0.90	20.088
0.95	13.621
1.00	8.782

Figure 8.2



2 종목의 유니버설 포트폴리오 구성 예시 : 논문의 8. Examples

Prior knowledge (in 1963) of this information would have enabled an investor to buy and hold the best stock (Iroquois) and earn a 791% profit. However, a closer look at the time series reveals some cause for regret. Table 8.1 lists the performance of the constant rebalanced portfolios B = (b, 1-b). The graph of $S_n(B)$ is given in Figure 8.2. For example, reinvesting current wealth in the propositions B=(0.8, 0.2) at the start of each trading day would have resulted in an increase by a factor of 37.5. In fact, the best rebalanced portfolio for this 22-year period is $B^* = (0.55, 0.45)$, yielding a factor $S_n^* = 73.619$. Here S_n^* is the target wealth ...

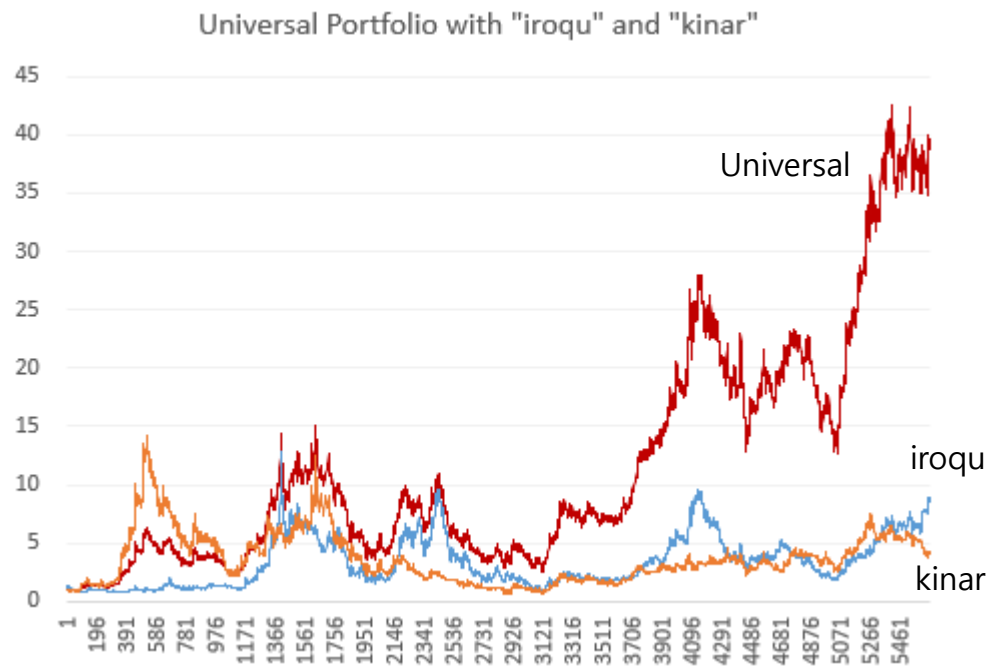


Figure 8.3. Performance of Universal Portfolio

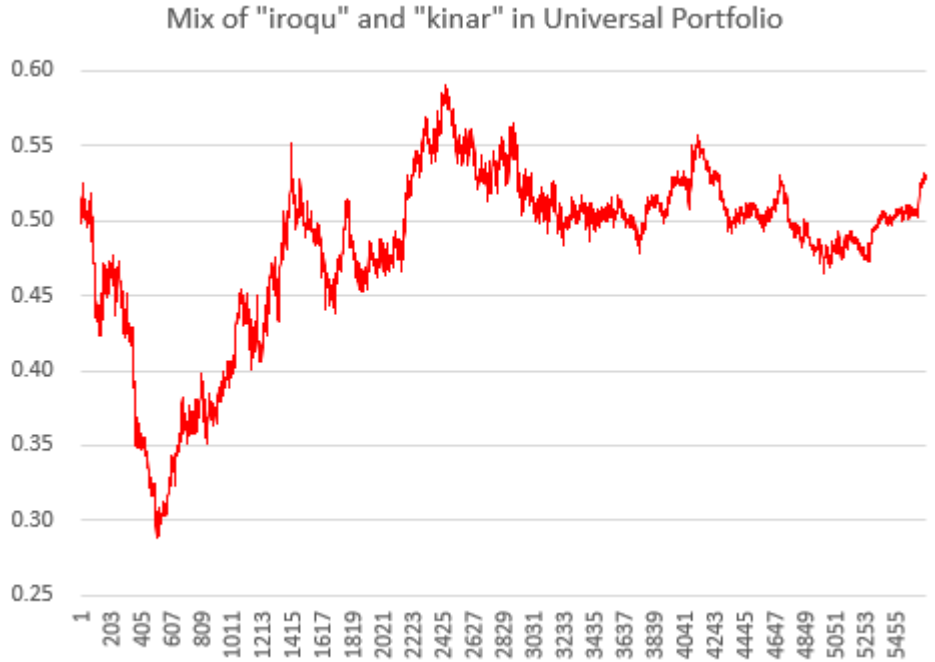


Figure 8.4. The portfolio $\hat{B}_k$

유니버셜 포트폴리오 구현 방법론

- 유니버설 포트폴리오를 현실적으로 구현하기는 매우 어렵다. 그 이유는 CRP의 조합이 무한히 많기 때문이다.
- 근사적으로 구현할 수 있는 방법으로는 GPU에서 대량의 병렬 처리를 생각해 볼 수 있다. CUDA와 같은 툴로 GPU에서 수 백만개의 쓰레드를 동시에 돌린다.
- 각 쓰레드가 하나의 CRP를 구성하고 모든 결과를 종합하여 근사적으로 유니버셜 포트폴리오를 구축해 볼 수 있다.
- 각 쓰레드가 자신의 CRP를 할당하는 것이 쉬운 문제는 아니다. 전 범위의 CRP를 골고루 할당해야 하지만 현실적인 문제로, 각 쓰레드는 랜덤하게 CRP를 구성해 볼 수 있다. 이 방식은 일종의 Monte Carlo simulation 방식으로 올바른 결과를 보장할 수 없다.
- 아래 자료는 약 10년간의 코스피 11개 종목에 대한 relative price이다. 이 데이터를 가지고 GPU에서 monte carlo 방식을 적용해 본다.

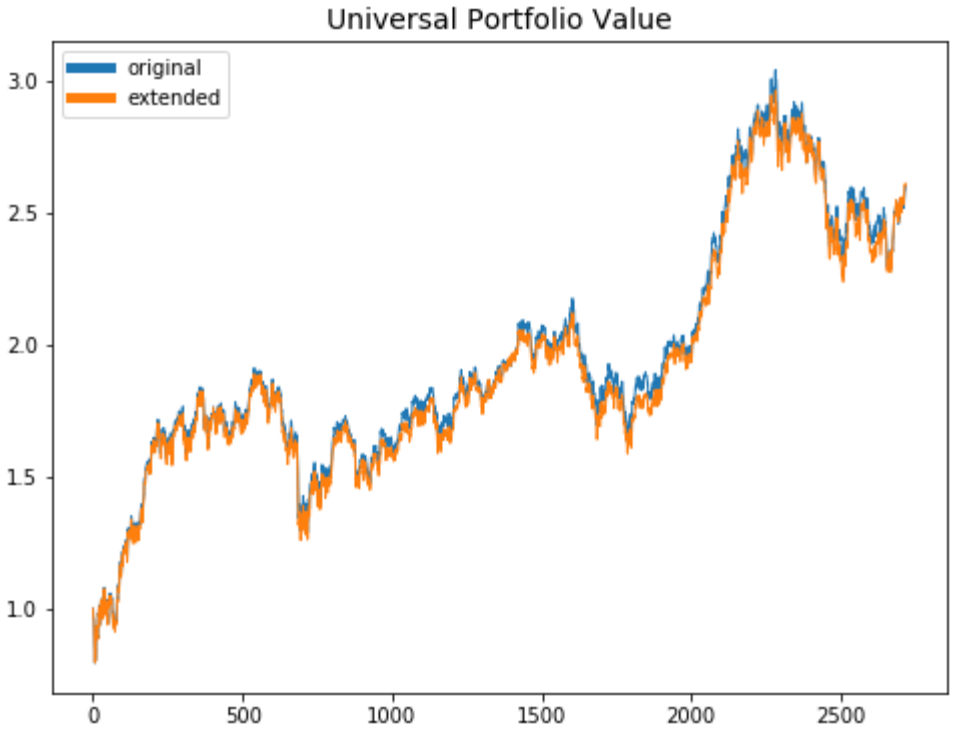
date	a005930	a000660	a005490	a035420	a017670	a055550	a105560	a066570	a006800	a009150	a015760
2008-11-12	1.0213	1.0294	0.9957	1.0065	1.0159	0.9744	0.9817	0.9919	0.9741	0.9987	1.0075
2008-11-13	0.9865	0.9333	0.9320	1.0203	1.0157	0.9357	0.9486	0.9662	0.8871	0.9706	1.0243
2008-11-14	0.9810	0.9980	0.9891	1.0159	0.9780	1.0000	1.0558	0.9626	1.0420	0.9947	0.9654
2008-11-17	0.9688	0.9335	0.9702	0.9412	1.0248	1.0000	1.0078	0.9524	1.0240	1.0066	0.9792
2008-11-18	0.9656	0.8883	0.9709	0.9092	0.9538	0.9375	0.9290	0.9474	0.9437	0.9566	0.9923
2008-11-19	1.0069	0.9260	0.9850	0.9477	0.9770	0.9750	0.9585	1.0153	0.9651	0.9203	0.9883
2008-11-20	0.9658	0.8509	0.9475	0.9101	1.0000	0.9026	0.8509	0.9576	0.8505	0.9015	0.9352
2008-11-21	1.0449	1.0423	1.0768	1.1424	0.9976	1.0795	1.0183	1.1100	1.0921	1.1076	1.0441
2008-11-24	0.9955	0.8784	0.9867	0.9256	0.9764	0.9141	0.9120	0.9524	0.9090	0.9731	0.9658
2008-11-25	1.0261	1.0017	1.0689	1.0653	1.0508	0.9827	1.0526	1.0027	1.0171	1.0046	1.0479
2008-11-26	1.0476	1.0358	1.0377	1.0349	1.0138	1.1445	1.1500	0.9959	1.1488	1.0092	1.0775
2008-11-27	1.0254	1.1450	1.0424	1.0629	0.9841	1.0119	1.0870	1.0433	1.0241	1.0091	1.0111
2008-11-28	1.0021	1.0662	0.9884	1.0373	1.0046	1.0287	1.0067	1.0182	1.0714	1.0345	1.0164
2008-12-01	0.9805	1.0256	0.9809	0.9306	0.9839	0.9672	0.9603	1.0115	1.0952	0.9869	0.9605
2008-12-02	0.9528	0.9474	0.9535	0.9474	0.9790	1.0170	0.9655	0.9610	1.0087	0.9647	0.9421
2008-12-03	0.9714	0.9847	0.9969	0.9583	1.0119	1.0550	1.0464	0.9659	1.0475	0.9878	1.0079
2008-12-04	0.9762	1.0296	1.0284	1.0097	1.0000	0.9716	0.9966	0.9769	0.9424	0.9552	0.9823
2008-12-05	0.9919	1.0356	1.0567	1.0287	1.0165	1.0309	1.0428	1.0208	1.0830	1.0129	1.0541
2008-12-08	1.0867	1.0317	1.0653	1.0512	1.0046	1.0725	1.0837	1.0816	1.0887	1.0813	1.1369

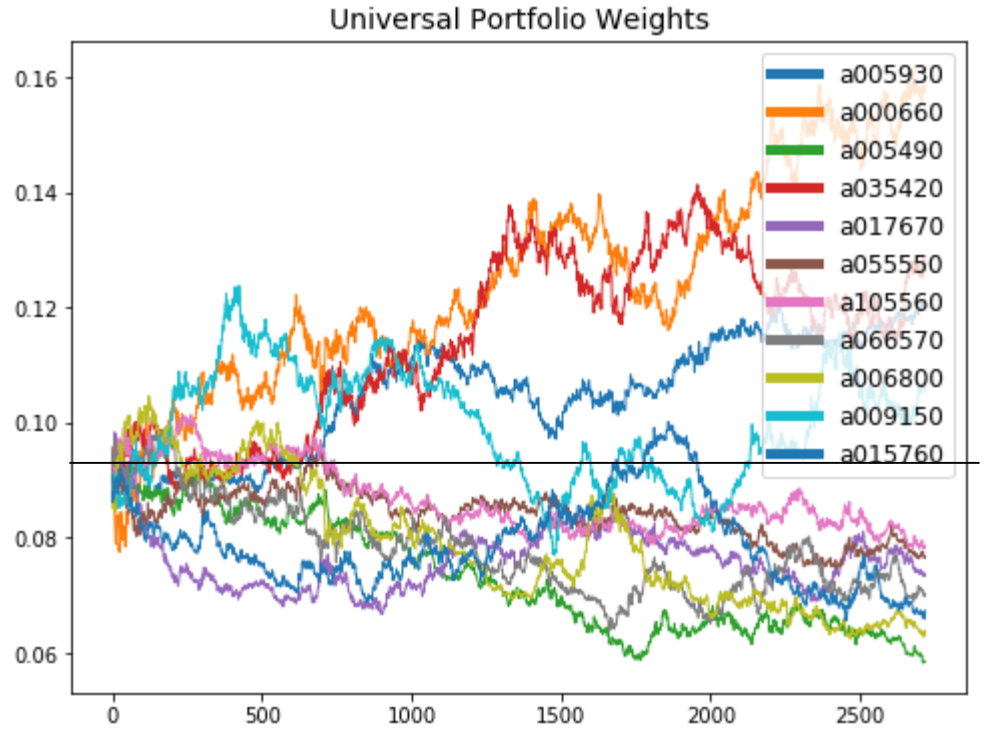
종목별 relative price

relative price
= 현재가격 / 이전가격

유니버셜 포트폴리오 구현 방법론

- GPU에 100만개의 쓰레드를 할당하고, Grid = (32 x 32), Block = (32 x 32), 각 쓰레드는 랜덤 비율로 CRP를 구성한다.
- 각 쓰레드는 relative price 데이터에 자신의 CRP 비율을 적용해서 $S_k(B)$를 산출한다.
- 각 쓰레드가 산출한 $S_k(B)$를 종합해서 유니버셜 포트폴리오를 만든다. 이 때는 parallel reduction 알고리즘으로 병렬처리한다.
- 아래 결과는 위의 과정을 거쳐 산출된 유니버셜 포트폴리오의 성과이다. 이론대로 11개 종목 중 가장 성과가 좋았던 것보다 높은 성과가 나오지 못했다.
- 몬테카를로 방식보다 더 개선된 방법을 찾아야 한다.

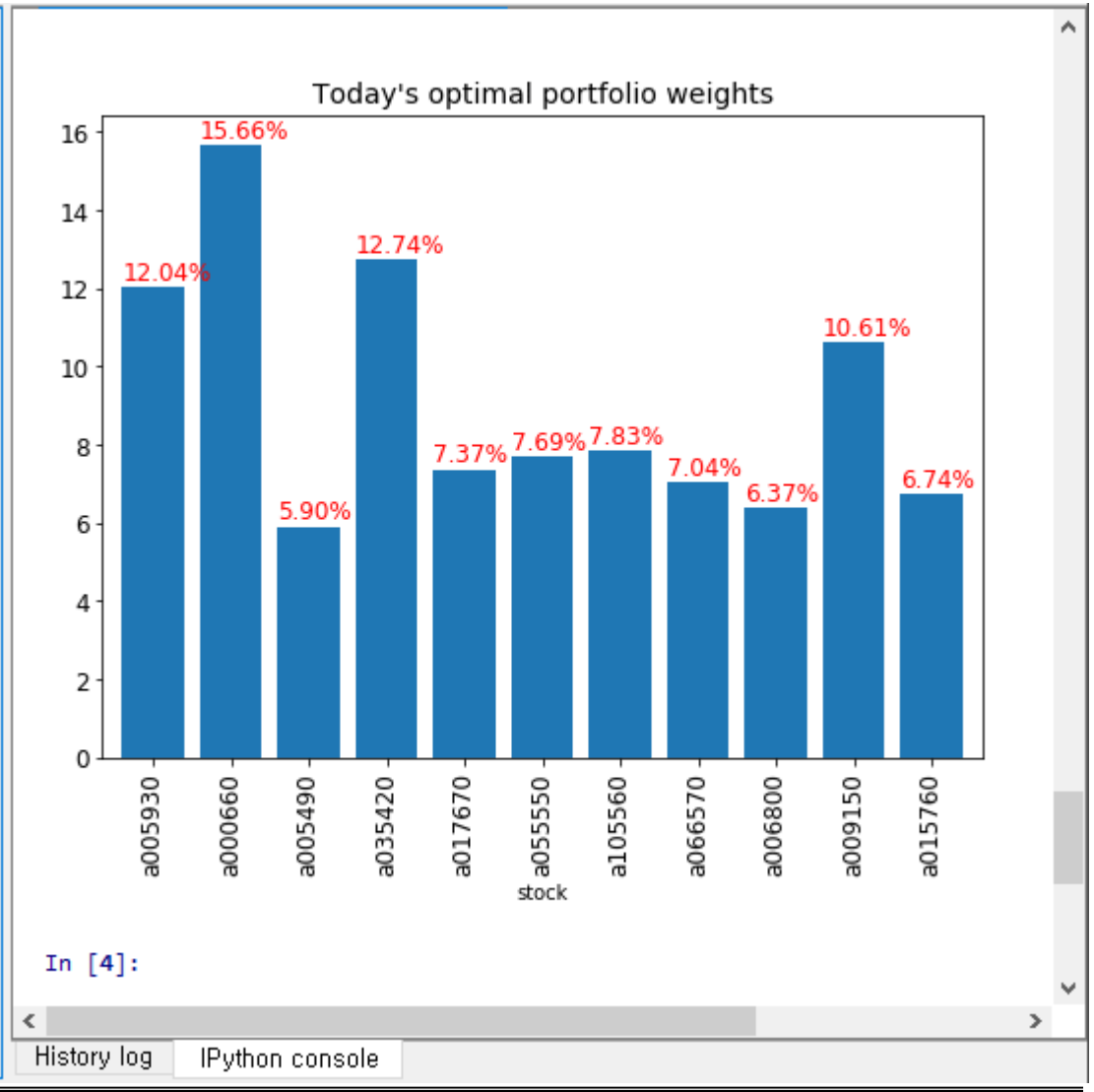




유니버셜 포트폴리오 구현 방법론

- 몬테카를로 방식을 적용했을 때 현재의 최적 투자 비율은 아래와 같다. 원하는 결과는 아니고, 단지 trial 임.

```python
# 유니버셜 포트폴리오 성능 확인 모듈
# 관련 논문 : Tomas M. Cover, 1996, Universal Portfolio
# 사용법 :
# 1. 종목 별로 relative price를 계산해서 stockData/relative_priceA.csv 파일을 생성한
# 2. CUDA로 작성된 uportfolio.cu를 실행하면 stockData/relative_priceA.csv를 읽어서
#    100만개의 CRP로 유니버셜 포트폴리오를 구축하고, 그 결과를 result/Bk2.csv 파일에
#    기록한다. (CUDA가 설치돼 있어야 한다.)
# 3. 이 프로그램인 verifyResult.py를 실행하면 result/Bk2.csv를 읽고, 승수인 pctX를 곱
#    종목 별 rebalancing 비중과 portfolio value, 그리고 오늘의 최적 비율을 확인한다.

# 2019.11.20 아마추어 퀀트 (blog.naver.com/chunjein)
# ------------------------------------------------------------------------------
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

pctX = 500
prcData = 'stockData/relative_priceA.csv'
wResult = 'result/Bk2.csv'      # uportfolio.cu로 relative_priceA.csv를 학습한 결고

def softmax(x):
    e = np.exp(x - np.max(x))
    return e / np.sum(e)

# 시계열 데이터 (univR)의 변동성을 optK 배 확대한다.
def extendWeight(univR, optK):
    # univR의 증분을 계산하고, optK를 곱한다 (확장).
    chg = univR.diff(1)
    chg = chg.fillna(0) * optK

    # 확장된 증분을 다시 누적한다
    chg = chg.cumsum()
```



딥러닝 (순환신경망)을 활용한 동적 포트폴리오 구축 예시 : labeling 및 학습 데이터 구축

- 1) 개별 종목 주가의 일일 수익률을 구간별로 나눠서 딥러닝의 입력값 (x)을 구성함. 2) 구간별로 수익률/변동성을 계산함. 3) 마코비츠의 포트폴리오 최적화 비율을 산출함. 4) 딥러닝의 출력값 $(y, label)$을 구성함.

* 개별 종목 주가의 일일 수익률 데이터 : 딥러닝 입력값 (X)

a005930	a000660	a005490	a035420	a017670	a055550	a105560	a066570	a006800	a009150	a015760
-0.0152	-0.0157	0.0294	-0.0122	-0.0041	0.0126	0.0023	0.0043	0.0134	0.0174	0.0019
0.0075	0.0012	0.0023	0.0245	0.0229	-0.0023	-0.0012	0.0058	-0.0013	-0.0087	0.0056
0.0113	-0.0190	-0.0092	0.0000	-0.0021	-0.0012	-0.0161	0.0058	-0.0186	0.0000	0.0596
0.0076	0.0047	0.0092	-0.0184	0.0084	0.0151	0.0172	0.0177	0.0227	0.0087	-0.0020
0.0213	0.0191	0.0163	-0.0151	0.0000	0.0035	0.0211	-0.0015	0.0427	0.0000	-0.0020
0.0157	0.0133	0.0094	0.0181	-0.0042	0.0035	0.0083	0.0045	-0.0056	0.0088	-0.0039
0.0000	0.0061	0.0024	0.0597	0.0127	-0.0152	-0.0118	-0.0236	-0.0028	-0.0132	-0.0156
-0.0138	-0.0182	-0.0047	0.0098	-0.0043	-0.0195	-0.0035	-0.0073	-0.0042	-0.0216	0.0294
-0.0039	0.0012	0.0024	-0.0258	-0.0085	0.0080	-0.0070	0.0029	-0.0028	0.0000	-0.0060
[illegible]										
[illegible]										
-0.0098	0.0292	0.0022	0.0033	0.0105	0.0069	0.0023	0.0058	-0.0069	0.0134	-0.0039
0.0000	-0.0179	-0.0022	-0.0098	-0.0084	0.0035	-0.0045	-0.0116	0.0028	-0.0222	-0.0098
0.0177	0.0204	0.0113	0.0131	-0.0021	0.0128	0.0218	0.0043	0.0083	0.0132	0.0019
0.0080	0.0013	-0.0180	0.0099	0.0000	0.0177	0.0070	0.0044	-0.0083	0.0270	-0.0078
-0.0120	-0.0455	0.0000	-0.0165	-0.0021	-0.0142	-0.0012	0.0000	-0.0083	-0.0091	-0.0153
-0.0040	-0.0171	-0.0111	-0.0098	0.0084	0.0000	0.0047	-0.0216	-0.0177	-0.0179	0.0096
0.0119	0.0147	0.0022	0.0196	-0.0042	0.0035	0.0000	0.0114	0.0095	0.0179	0.0096
0.0020	0.0087	-0.0110	-0.0293	0.0063	0.0012	-0.0023	0.0087	-0.0108	0.0045	0.0097
0.0171	0.0062	0.0154	0.0000	0.0000	0.0274	0.0201	0.0015	0.0245	-0.0180	0.0098
0.0123	0.0126	0.0224	0.0162	0.0149	0.0109	0.0266	0.0073	0.0125	-0.0045	-0.0118
-0.0072	-0.0238	-0.0045	0.0000	-0.0085	-0.0122	-0.0206	0.0000	-0.0166	0.0179	0.0059
0.0238	0.0074	0.0114	0.0033	0.0085	0.0073	0.0109	-0.0160	0.0110	0.0369	0.0099
-0.0052	0.0000	0.0046	0.0165	-0.0064	0.0037	-0.0097	0.0442	-0.0069	0.0047	0.0020
0.0084	0.0100	-0.0091	-0.0423	-0.0021	-0.0169	-0.0048	0.0015	-0.0137	0.0095	-0.0079
-0.0259	-0.0310	-0.0291	-0.0189	-0.0169	-0.0167	-0.0154	-0.0030	-0.0215	-0.0095	-0.0272
-0.0041	-0.0024	-0.0022	0.0189	-0.0104	0.0131	-0.0047	-0.0134	0.0027	0.0287	0.0058

x_1

* 구간별 수익률/변동성

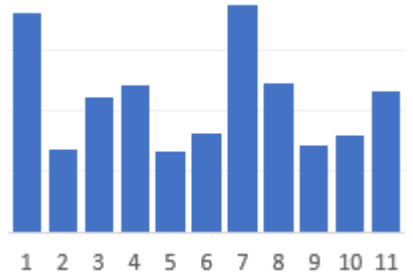
종목	수익률	변동성
a005930	0.5872	0.1971
a000660	-0.1831	0.2074
a005490	1.2689	0.1851
a035420	1.7551	0.4208
a017670	0.6861	0.1546
a055550	0.1730	0.1750
a105560	0.2927	0.1884
a066570	0.4395	0.1746
a006800	0.9536	0.2806
a009150	0.2182	0.2024
a015760	1.3910	0.3584

* 딥러닝 출력값 (Y)

a005930	9.80%
a000660	8.68%
a005490	9.11%
a035420	9.20%
a017670	8.66%
a055550	8.80%
a105560	9.86%
a066570	9.23%
a006800	8.71%
a009150	8.80%
a015760	9.15%

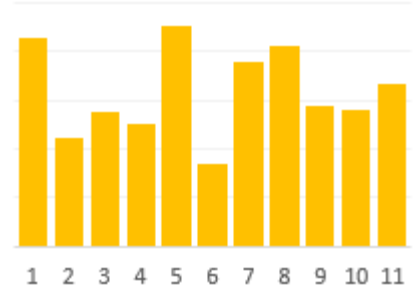
구간 별로 마코비츠의 포트폴리오를 최적화함

종목별 투자 비중



리밸런싱

종목별 투자 비중



* 다음 구간의 딥러닝 입력값 (X)

종목	수익률	변동성
a005930	0.7277	0.1717
a000660	1.8425	0.2819
a005490	-0.4561	0.2700
a035420	1.1983	0.4083
a017670	0.0000	0.1424
a055550	0.0290	0.1705
a105560	-0.0859	0.2248
a066570	-0.4026	0.1917
a006800	1.1897	0.2543
a009150	0.2201	0.2403
a015760	-0.4941	0.1991

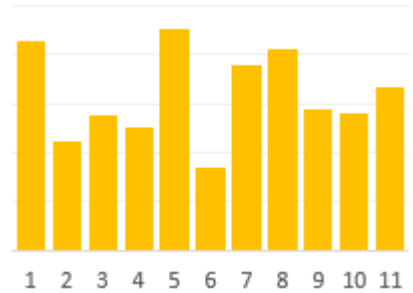
* 다음 구간의 딥러닝 출력값 (Y)

a005930	9.64%
a000660	8.62%
a005490	8.88%
a035420	8.77%
a017670	9.77%
a055550	8.35%
a105560	9.39%
a066570	9.57%
a006800	8.95%
a009150	8.90%
a015760	9.16%

딥러닝 (순환신경망)을 활용한 동적 포트폴리오 구축 예시 : 학습 (Learning)

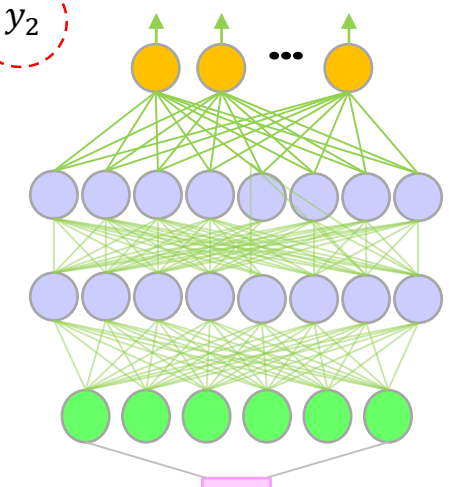
- 개별 종목 주가의 일일 수익률을 구간별로 나눠서 순환신경망 (ex: LSTM)에 순차적으로 입력함. $(x_1, x_2, x_3, \ldots x_{m-1})$
- 순환신경망의 출력값을 일반 신경망 (feed forward network : FFN)의 입력층에 입력함.
- FFN의 출력층에서는 각 종목의 투자 비중이 나오도록 함 (softmax activation function).
- FFN의 출력층에는 입력 구간 다음 구간의 label이 나오게 함. $(y_2, y_3, y_4, \ldots y_m)$
- 학습이 완료된 후 현재 구간의 입력값을 넣으면 출력값은 다음 구간의 최적 포트폴리오 값이 출력됨. ← 최종 결괏값
- 결과 : $x_m \rightarrow y_{m+1}$

a005930	9.64%
a000660	8.62%
a005490	8.88%
a035420	8.77%
a017670	9.77%
a055550	8.35%
a105560	9.39%
a066570	9.57%
a006800	8.95%
a009150	8.90%
a015760	9.16%



y_2

다음 구간의 포트폴리오 최적값이 나오도록 학습함. 학습이 완료된 후 현재 구간을 입력하면 다음 구간의 예측값이 출력됨.



FFN

* 개별 종목 주가의 일일 수익률 데이터 : 딥러닝 입력값 (X)

x_2

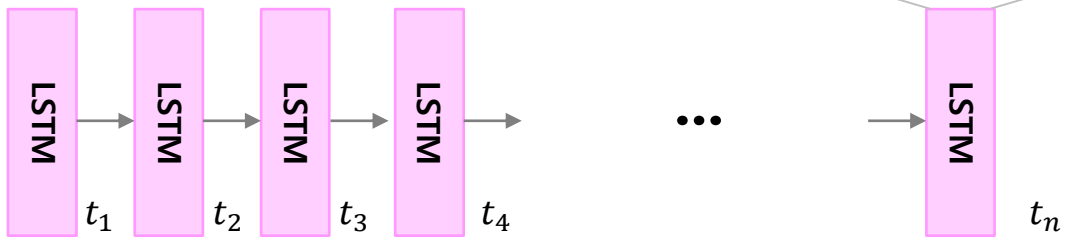
a005930	a000660	a005490	a035420	a017670	a055550	a105560	a066570	a006800	a009150	a015760
-0.0152	-0.0157	0.0294	-0.0122	-0.0041	0.0126	0.0023	0.0043	0.0134	0.0174	0.0019
-0.0075	0.0012	0.0023	0.0245	0.0229	-0.0023	-0.0012	0.0058	-0.0013	-0.0087	0.0056

t

	a005930	a000660	a005490	a035420	a017670	a055550	a105560	a066570	a006800	a009150	a015760
0.0113	0.0078	0.0000	-0.0071	0.0291	0.0063	0.0023	0.0023	0.0088	-0.0055	0.0172	-0.0118
0.0076	0.0039	0.0356	-0.0459	-0.0033	-0.0127	-0.0069	-0.0277	-0.0117	0.0069	0.0175	0.0059
0.0213	-0.0098	0.0292	0.0022	0.0033	0.0105	0.0069	0.0023	0.0058	-0.0069	0.0134	-0.0039
0.0157	0.0000	-0.0179	-0.0022	-0.0098	-0.0084	0.0035	-0.0045	-0.0116	0.0028	-0.0222	-0.0098
0.0000	0.0177	0.0204	0.0113	0.0131	-0.0021	0.0128	0.0218	0.0043	0.0083	0.0132	0.0019
-0.0138	0.0080	0.0013	-0.0180	0.0099	0.0000	0.0177	0.0070	0.0044	-0.0083	0.0270	-0.0078
-0.0039	-0.0120	-0.0455	0.0000	-0.0165	-0.0021	-0.0142	-0.0012	0.0000	-0.0083	-0.0091	-0.0153
	-0.0040	-0.0171	-0.0111	-0.0098	0.0084	0.0000	0.0047	-0.0216	-0.0177	-0.0179	0.0096
	0.0119	0.0147	0.0022	0.0196	-0.0042	0.0035	0.0000	0.0114	0.0095	0.0179	0.0096
	0.0020	0.0087	-0.0110	-0.0293	0.0063	0.0012	-0.0023	0.0087	-0.0108	0.0045	0.0097

x_1

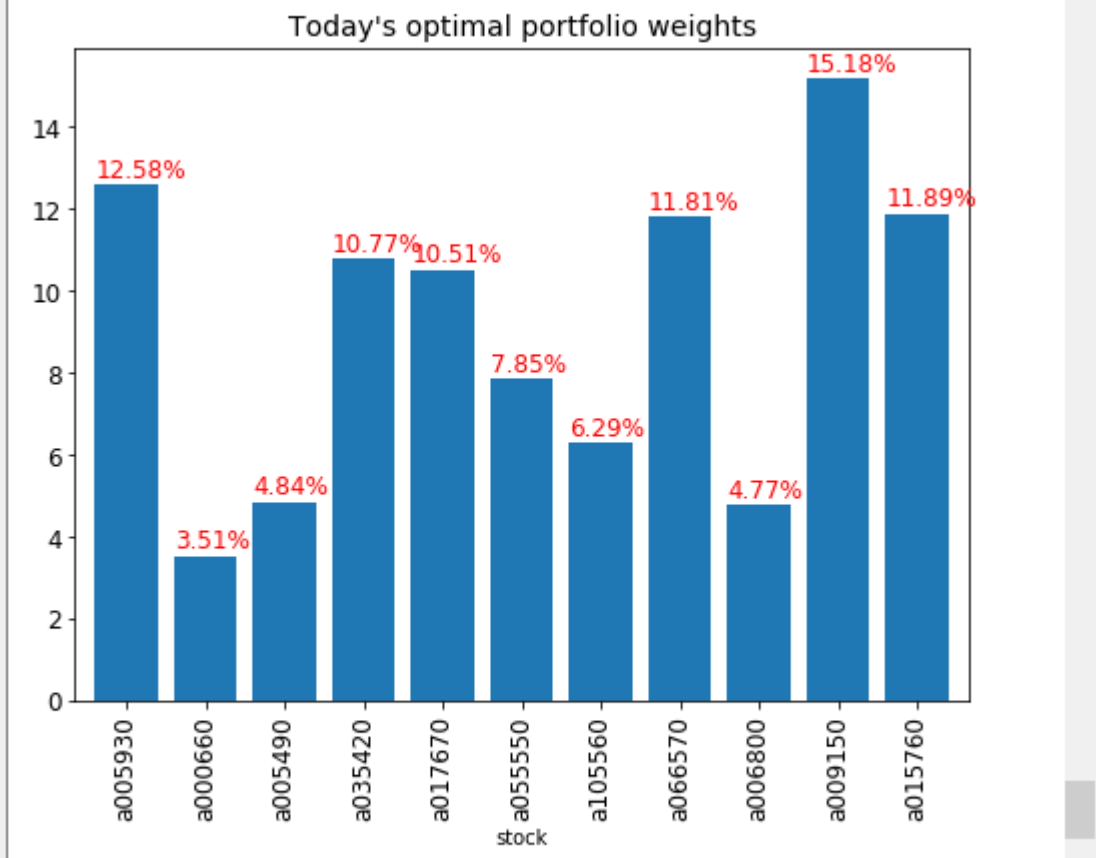
순환신경망 (RNN/LSTM/GRU)



- 딥러닝 (순환신경망)을 활용한 동적 포트폴리오 구축 예시 : 성과 분석

```python
# 지도학습으로 찾은 최적 포트폴리오의 성과를 육안으로 확인한다.
# 거래 비용은 고려하지 않았음.
#
# 2019.11.11 아마추어 퀀트 (blog.naver.com/chunjein)
# ------------------------------------------------------------
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

# 종목 별 수익률에 투자 비율을 적용해서 포트폴리오 가치를 계산한다.
def portfolioValue(rtnX, w):
    # 데이터 프레임을 배열로 변환한다
    x = np.array(rtnX)
    y = np.array(w)

    # 수익률 * weight를 계산한 후 결과를 데이터 프레임으로 변환한다.
    v = np.sum(x * y, axis = 1)
    v = pd.DataFrame(v)
    v.index = rtnX.index
    v.columns = ['Rp']

    # S의 누적 relative price를 계산한다.
    v['Rel_prc'] = np.exp(v['Rp'])
    v['Sp'] = pd.DataFrame(v['Rel_prc']).cumprod()
    return v

# 종목 별 w 변화와 누적 relative price 차트를 관찰한다.
def showPV(rtnX, w):
    # 종목 별 w 변화를 관찰한다.
    plt.figure(figsize=(12,8))
    plt.plot(np.array(w), linewidth=1)
    plt.title("Portfolio Weights", fontsize=14)
    leg = plt.legend(list(rtnX.columns), prop={'size': 12})
```

```
* 포트폴리오 수익률 = 275.62 (%), 연간 25.94 (%)
```

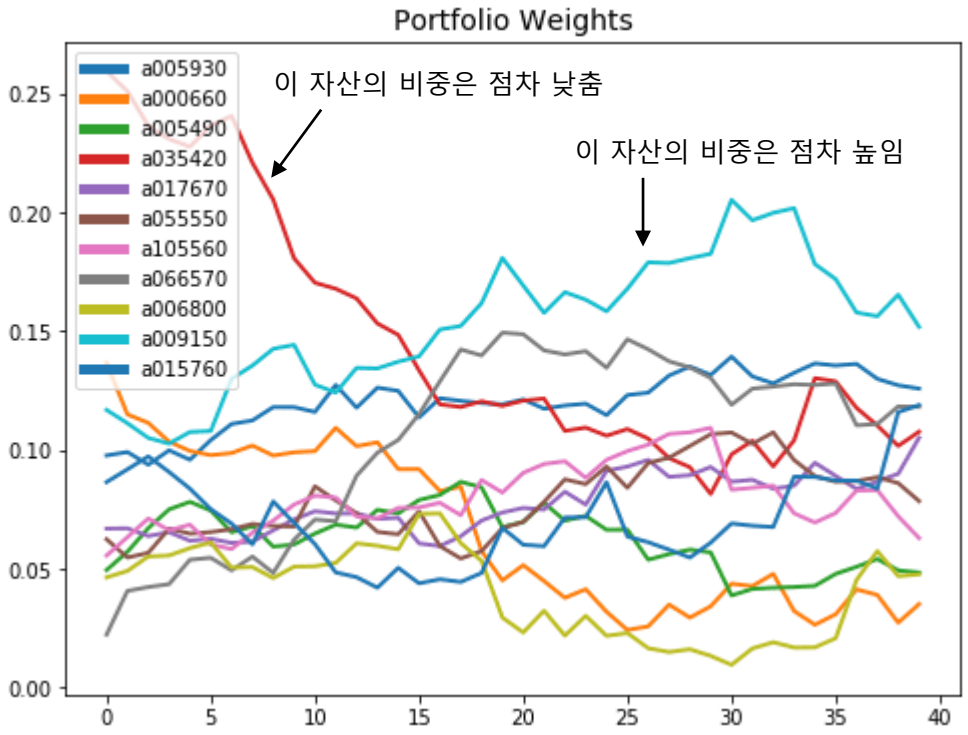


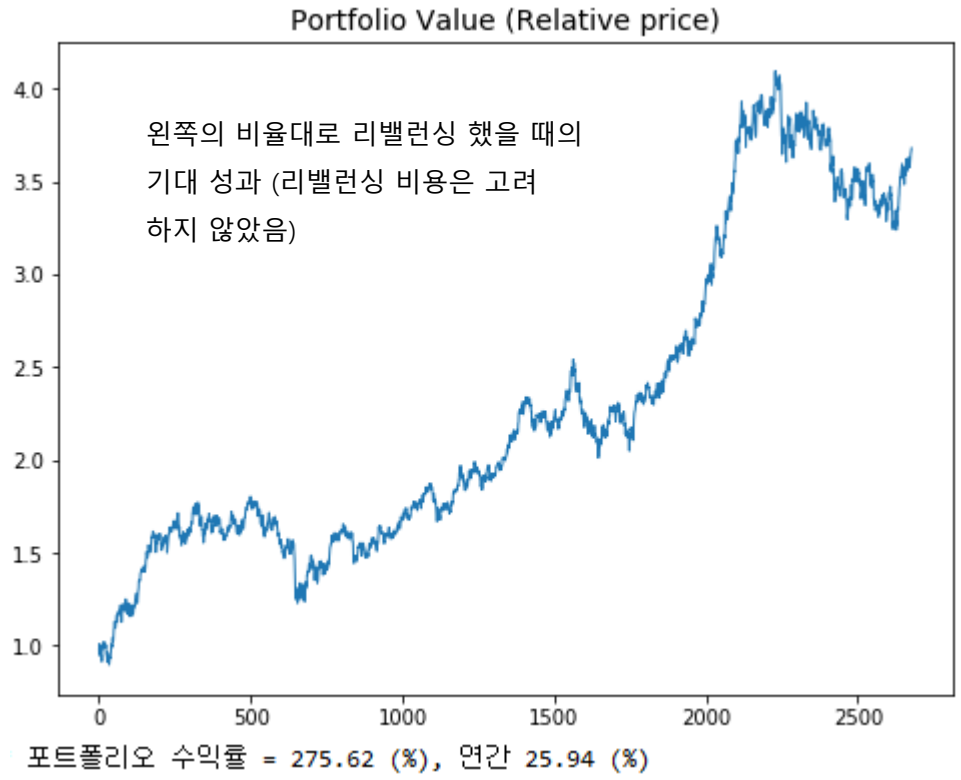

```
In [6]:
```

History log | IPython console

딥러닝 (순환신경망)을 활용한 동적 포트폴리오 구축 예시 : 성과 분석

- 아래 왼쪽 그림은 종목별 투자 비중의 변화임. 어떤 종목은 시간이 지남에 따라 보유 비중을 점차 높이고, 어떤 종목은 점차 낮추는 것이 가장 효율적이라는 결과임. → 포트폴리오 리밸런싱 결과
- 오른쪽 그림은 각 종목의 비중을 왼쪽과 같이 적용했을 때의 포트폴리오 성과임.
- 단, 이 결과는 리밸런싱에 소용되는 거래 비용을 고려하지 않은 결과임. 또한 딥러닝은 과잉적합 (overfitting)의 가능성이 있으므로 이에 대한 해결책을 마련해야 함 (regularization 혹은 dropout 등).





강화학습을 활용한 포트폴리오 구축 사례

- 관련 논문 : Yoshiharu Sato , 2019.5.3, Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey
- 아래 예시는 딥러닝 기술 중 이미지 처리에 뛰어난 성능을 보이는 합성곱 신경망 (Convolutional Neural Network : CNN)과 강화학습 (Reinforcement Learning)을 이용한 동적 포트폴리오 전략임.
- 종목별 주가 (고가, 저가, 종가의 3 features)를 기간별로 나눠서 CNN의 convolution 층으로 입력함.
- Convolution 층의 출력에 이전의 포트폴리오 비중을 추가하여 FFN으로 입력함.
- FFN의 출력은 현재 적용할 (예측) 최적 포트폴리오 비중임.

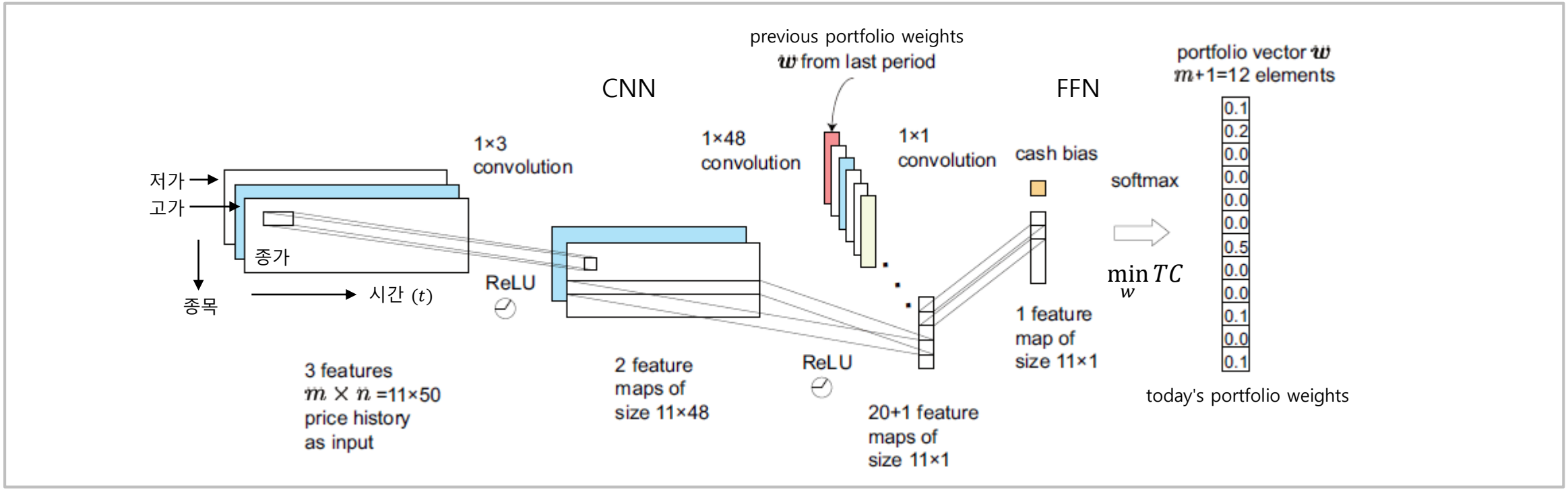


그림 출처 : 관련 논문의 Figure 2 : Convolutional Neural Network (CNN) version of the Ensemble of Identical Independent Evaluators (EIIE) topology.

- 강화학습을 활용한 포트폴리오 구축 사례

Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey

Yoshiharu Sato †

First Version: March 24th, 2019
This Version: May 3rd, 2019

Abstract

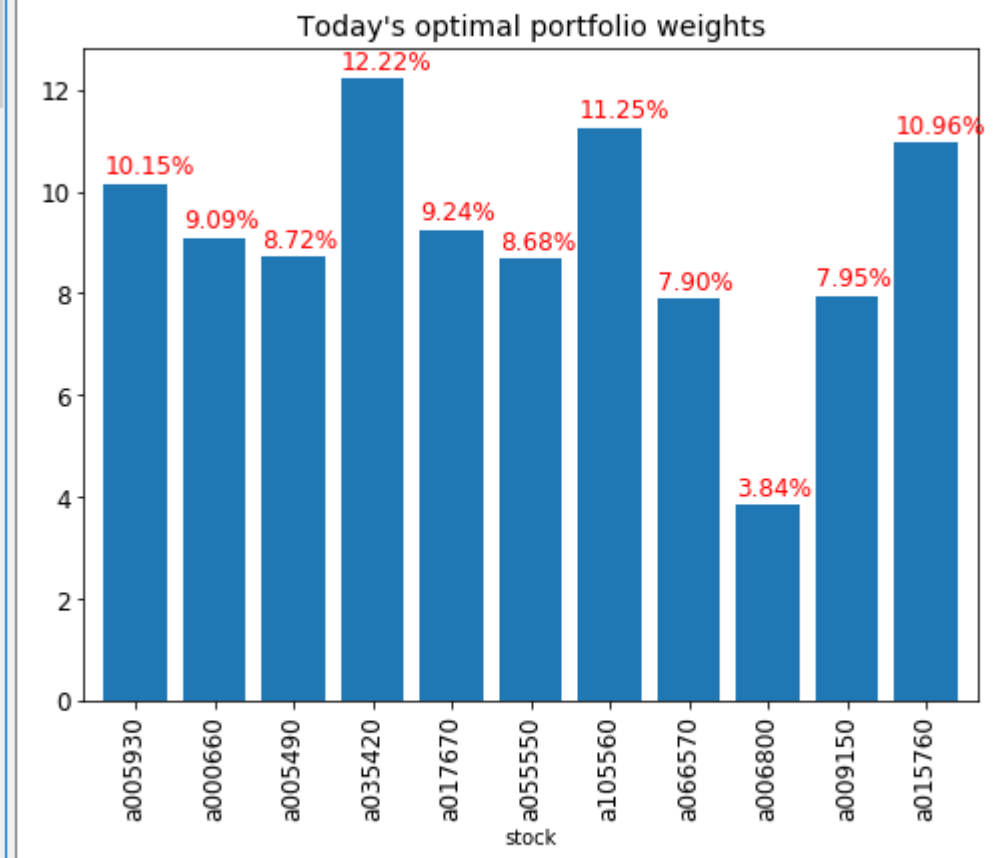
Financial portfolio management is one of the problems that are most frequently encountered in the investment industry. Nevertheless, it is not widely recognized that both Kelly Criterion and Risk Parity collapse into Mean Variance under some conditions, which implies that a universal solution to the portfolio optimization problem could potentially exist. In fact, the process of sequential computation of optimal component weights that maximize the portfolio's expected return subject to a certain risk budget can be reformulated as a discrete-time Markov Decision Process (MDP) and hence as a stochastic optimal control, where the system being controlled is a portfolio consisting of multiple investment components, and the control is its component weights. Consequently, the problem could be solved using model-free Reinforcement Learning (RL) without knowing specific component dynamics. By examining existing methods of both value-based and policy-based model-free RL for the portfolio optimization problem, we identify some of the key unresolved questions and difficulties facing today's portfolio managers of applying model-free RL to their investment portfolios.

강화학습을 활용한 포트폴리오 구축 사례

- LSTM/CNN과 강화학습 (Actor-Critic)을 활용하여 포트폴리오 최적화에 적용해 보았으나, 올바른 값으로 수렴하지 못해 실패함. → 추후 보완 예정임

```
# N-step TD LSTM-Actor-Critic 실습 : Portfolio 최적화. max(Sharp Ratio)
#
# 학습이 전혀 안되고 있음. --> 원인 추적 중.
# 이 프로그램은 일종의 challenge로 개념적으로나 기술적으로 보완해야할 점이 많다.
# 여기서는 labelling 없이 (unsupervised) 강화학습으로 포트포리오를 최적화하는
# 방법론을 소개하는 것 정도로 의미를 둔다.
#
# 2019.11.11 아마추어 퀀트 (blog.naver.com/chunjein)
# ------------------------------------------------------------------------
import numpy as np
import pandas as pd
from keras.models import Model
from keras.layers import Input, Dense, LSTM
from keras.optimizers import Adam
from keras import backend as K
import pickle

ALPHA =  0.0005
GAMMA = 0.95
MAX_ENTROPY = 0.2
INC_WEIGHT = 0.3
nStock = 11
nAction = nStock
nHidden = 16
nActOut = nAction
nStepTD = 1
nLstmTimeStep = 40
nLstmHidden = 32

trainData = "data/2.price_return.pickle"
actor_saver = "data/2-1.actor_weights.h5"
critic_saver = "data/2-2.critic_weights.h5"
resultW = "data/2-3.result.csv"

# LSTM-Actor-Critic Network을 생성한다.
```

```
* 포트폴리오 수익률 = 86.69 (%), 연간 8.15 (%)
```




```
In [8]:
```

History log | IPython console

The End

수고하셨습니다