Algorithm Project Presentation

Markowitz Portfolio with Integer Constraint

Sun Woo Lim

June 8, 2021

Original Markowitz Portfolio

Portfolio : a range of investments held by a person or organization (source : google dictionary). e.g., APPL, TSLA, USBond

✓ You want to 1) **maximize expected profit** + 2) **reduce the risk** : **Tradeoff**!

Markowitz Portfolio : Optimize the **proportion vector (x)** of items in a portfolio (Markowitz, H. (1959)).

$$max_{x \in \mathbb{R}^n}(e^T x - \gamma x^T C x) = \sum_i e_i x_i - \gamma \sum_i \sum_j C_{ij} x_i x_j$$

$$s.t. x_i \geq 0, \forall i \in \{1, 2, ..., n\}, \sum_i x_i = 1$$

- n : **number of items in a portfolio** $\vec{r} \in \mathbb{R}^n$: random vector of **returns** (standardize the price e.g., $\$100$ to make sense).
- $\vec{e} := E(\vec{r})$, $C := E(r-e)(r-e)^T$: Unknown → Use point estimates of them!
- $\gamma > 0$: **risk-sensitivity parameter**. $\uparrow \gamma$: **risk averse**, $\downarrow \gamma$: **risk seeking**.
- Two Constraints : Because x represents **proportion**.
- **Need Efficient Algorithms** : $\because$ 1) So many options to choose from → $n \uparrow\uparrow$, 2) **Inference** of e and C fastly varies!

Relaxed Version of Markowitz Portfolio

Added Integer Constraint on Fraction. Force x_i : finite decimal where $k = \#$ of decimal points : **Simplification**

✓ e.g., $(n, k) = (4, 1) : (0.3, 0.1, 0.2, 0.4)$ & $(n, k) = (3, 2) : (0.32, 0.61, 0.07)$.

✓ I assume $k = 1$ or 2 and $n >> k$ in calculation of time / space complexity.

$$max_{x \in \mathbb{R}^n} (e^T x - \gamma x^T C x)$$

$$s.t. \sum_i x_i = 1, x_i = \frac{z}{10^k}, z \in \mathbb{N} \cup \{0\}, k \in \mathbb{N}$$

Why Integer Constraints?

1. **Why I can**

   1) People understand fractions in finite decimals and only need them

   2) Computers : Finite Decimal Representation ($\because$ limited memory)

2. **Why I did**.

   1) Original Problem : requires deep knowledge of optimization, relies on whether the problem is convex (C : PSD)
   2) QP algorithms : integer point methods, extensions of simplex algorithms,... : hard to apply in class concepts
   3) Original Problem : No brute Force

Ways to Solve : Brute Force

```
def BruteForce(n, k):
    ans_vec = []
    1) Get every possible combinations of x
    2) Calculate the objective function for all x
    3) return max(e^T x - gamma x^T C x) and argmax(e^T x - gamma x^T C x)
```

Figure: Pseudocode for Brute Force Algorithm

Time Complexity $T(n) \in \Theta(\frac{n^{10^k+2}}{10^k!}) = \Theta(n^{10^k+2})$: Best = Avg = Worst .... considering that $n >> k$. $k = 1$ or 2.

Two **basic operations** : 1. Elementwise multiplication for $e^T x - \gamma x^T C x$, 2. Elementwise Comparison to get minimum.

1. **Multiplication** $M(n) =$ (Total number of combinations of x) $\times$ (Complexity of $e^T x - \gamma x^T C x$ calculation).

   a) Combinations of $x = \#$ of multicombinations of $(a_1, ..., a_n)$ s.t. $a_1 + ... + a_n = 10^k, a_i \in \mathbb{N} \cup \{0\}$.

   $= P(n + 10^k - 1; 10^k, n - 1) = \binom{n+10^k-1}{10^k} \in \Theta(\frac{n^{10^k}}{10^k!})$.

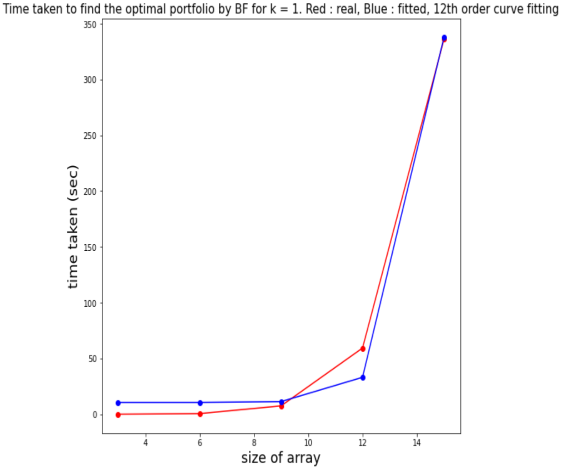
   b) Complexity of $e^T x - \gamma x^T C x$: $\Theta(n^2)$ in general, $\Theta(n)$ for Diagonal C.

   $\therefore M(n) = \Theta(\frac{n^{10^k+2}}{10^k!})$

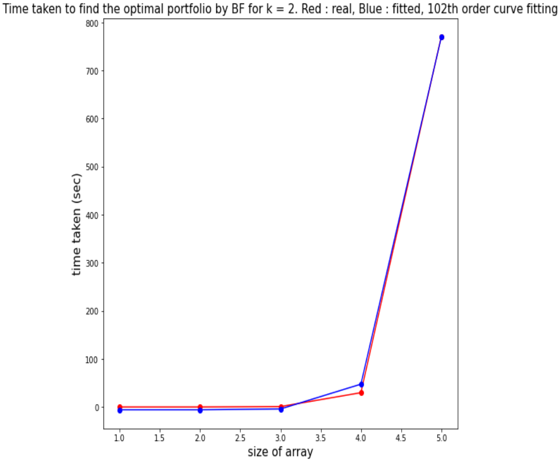
2. **Elementwise Comparison** $C(n) =$ Length of ans_vec $= \Theta(\frac{n^{10^k}}{10^k!})$.

Space Complexity $S(n) \in \Theta(\frac{n^{10^k}}{10^k!}) = \Theta(n^{10^k})$: Best = Avg = Worst

Brute Force Empirical Results For General C



(a) k $= 1$ Case, $n \in \{3, 6, 9, 12, 15\}$, $T(n) \in \Theta(n^{12})$



(b) k $= 2$ Case,$n \in \{1, 2, 3, 4, 5\}$, $T(n) \in \Theta(n^{102})$

What If C is Diagonal : Greedy Algorithm

Interpretation) Return of an investment item A and item B consisting a portfolio are uncorrelated.

- Proper Assumption? Not Quite, but a **good starting point**.

Assuming C is diagonal,$e^T x - \gamma x^T C x = \sum_{i=1}^{n} (e_i x_i - \gamma C_{ii} x_i^2)$: Objective Function is **Separable** !

Candidate 1 : Greedy Algorithm

1. Greedy Algorithm Idea) Find item i s.t. 1) e_i : big, 2) c_{ii} : small $\rightarrow$ only invest on i.
2. Does not work $\because$ **no reasonable metric** that shows how big expectation is compared to variance.
3. Looking at the figure below, multiple options are chosen : greedy algorithm is not plausible.

Item	1	2	3	4	5	6	7	8
Expected Return	0.71	0.97	-0.22	0.13	1.49	-0.24	-0.14	1.01
Variance	0.66	0.78	0.1	0.06	0.96	0.62	0.09	0.56
Optimal Proportion	10%	20%	0	0	40%	0	0	30%

Figure: Example showing that Greedy Algorithm may not work. $\gamma = 1, k = 1, n = 8$, objective function value $= 0.9222$

Slightly changing the proportion between item 1,2,5,8 does not lead to the optimum
(e.g, with $x = (0.1, 0.2, 0, 0, 0.3, 0, 0, 0.4)$, function value $= 0.9022$ & Only invest on item 5 $\rightarrow$ function value $= 0.55$))

Dynamic Programming Guarantees Optimal Solution

Terms

$Mark(n, k, p) = max_{\sum_i x_i = p}(e^T x - \gamma x^T C x)$. Call $Mark(n = n, k = k, p = 1)$ $\because \sum_i x_i = 1$ but p necessary for recursion.

Alerts

1) The recursive equation is very complicated for non-diagonal C. For simplicity, I first deal with diagonal C.
2) The record table has 10^k columns. For visualization purposes, let $k = 1$ and ignore this parameter for a while.

n \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	1	2	3	4	5	6	7	8	9	10
2	11	15	16	17	18	19	20	21	22	23
3	12	24	25	26	27	28	29	30	31	32
4	13	33	34	35	36	37	38	39	40	41
5	14	42	43	44	45	46	47	48	49	50
6										51

Figure: The order of execusion in case of $Mark(n = 6, k = 1, p = 1)$

1. **Base Case**

   1) $Mark(n = 1, p = p) = max(pe_1 - \gamma p^2 C_{11}, 0)$ $\because$ Only one item. If the objective is positive, invest on it.

   2) $Mark(n = n, p = 0.1) = max(max_i(0.1e_i - 0.1^2\gamma C_{ii}), 0)$ $\because$ with $p = 0.1$, there is only one item to choose from.

2. **Recursive Equation**

   $Mark(n, p) = max[Mark(n-1, p), Mark(n-1, p-0.1) + (0.1e_n - 0.1^2\gamma C_{nn}), ..., Mark(n-1, 0.1) +$
   $((p-0.1)e_n - (p-0.1)^2\gamma C_{nn}), (pe_n - \gamma p^2 C_{nn})]$

Review

1. $Mark(n, 0.1) = max(max_i(0.1e_i - 0.1^2\gamma C_{ii}), 0)$, $Mark(1, p) = max(pe_n - \gamma p^2 C_{nn}, 0)$: **Base Case**
2. $Mark(n, p) = max[Mark(n-1, p), Mark(n-1, p-0.1) + (0.1e_n - 0.1^2\gamma C_{nn}), ..., Mark(n-1, 0.1) + ((p-0.1)e_n - (p-0.1)^2\gamma C_{nn}), (pe_n - \gamma p^2 C_{nn})]$: **Recursive Relation**

Time Complexity $T(n) \in \Theta(100^k \cdot n) = \Theta(n)$: Best = Avg = Worst

Two **basic operations** : 1. Elementwise multiplication for $e_i x_i - \gamma C_{ii} x_i^2$, 2. Elementwise Comparison to get maximum.

1. Multiplication $M(n, k)$ = Total number of $e_i x_i - \gamma C_{ii} x_i^2$ calculations : Because calculating $e_i x_i - \gamma C_{ii} x_i^2$: $\Theta(1)$
2. Comparison $C(n, k)$ = Total number of elementwise comparisons.

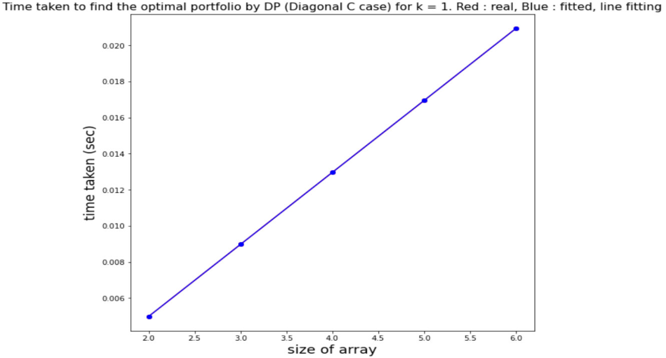
n \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	1	1	1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6										10

Figure: $(n, k) = (6, 1)$ case. $M(n, k, p = 1) = C(n, k, 1) = \sum$ shaded numbers

$$M(n, k) = C(n, k) \approx \frac{(10^k)(10^k + 1)}{2} \cdot n \in \Theta(100^k \cdot n) \in \Theta(n).$$

Space Complexity $S(n)$ =Size of the record table $= S_n = 10^k \cdot n \in \Theta(n)$: Best = Avg = Worst

Dynamic Programming Empirical Results




```
time_list
```

```
[0.004986286163330078,
 0.008975505828857422,
 0.012965202331542969,
 0.016945838928222656,
 0.020943880081176758]
```

Figure: Case $k = 1$, $n \in \{2, 3, 4, 5, 6\}$. C is diagonal

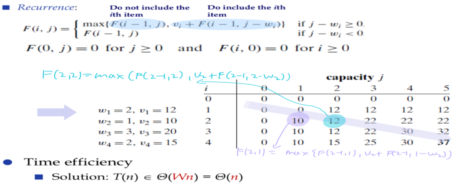
Comparison with Original Knapsack

1. $Mark(n, 0.1) = max(max_i(0.1e_i - 0.1^2\gamma C_{ii}), 0)$, $Mark(1, p) = max(pe_n - \gamma p^2 C_{nn}, 0)$: **Base Case**
2. $Mark(n, p) = max[Mark(n-1, p), Mark(n-1, p-0.1) + (0.1e_n - 0.1^2\gamma C_{nn}), ..., Mark(n-1, 0.1) + ((p-0.1)e_n - (p-0.1)^2\gamma C_{nn}), (pe_n - \gamma p^2 C_{nn})]$: **Recursive Relation**

$T(n) \in \Theta(100^k \cdot n)$ while $S(n) \in \Theta(10^k \cdot n)$

n \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	1	1	1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6										10

(a) $M(n=6, k=1, p=1) = C(6,1,1) = \sum$ shaded numbers



i	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	37

(b) Knapsack Problem (from Algorithm Lecture #8 DP class material)

Difference from the Original Knapsack. Format : this problem $\leftrightarrow$ knapsack

1. Considering k, asymptotic $T(n) \neq$ asymptotic $S(n) \leftrightarrow T(n) = S(n) = \Theta(Wn)$.
2. Moving right of the record table, more numbers to multiply and compare $\leftrightarrow$ compare two items everywhere
3. Memory function : easy pattern $\leftrightarrow$ random looking pattern.

Importance & Value of This Problem

For this specific problem

: Foundational role in Mean Variance Analysis : maximize expectation for a given level of risk.

Only that?

Generalized into Proportion Optimization Problem

- You are often faced with **proportion choice problems** in various options!
- Important thing : Same Constraints, only different **objective function**!
  In Markowitz Portfolio, the objective function is $e^T x - \gamma x^T C x$
- Other applications :
  1. Capacity of Discrete Memoryless Channel : $max_{p,q} - \sum q_j log q_j - p^T r$ s.t. $p^T P = q, p^T s \leq S, \sum p_i = 1, p_i \geq 0.$
  2. Water Filling : $min_x - \sum_i log(\alpha_i + x_i)$ s.t. $x_i \geq 0, \sum_i x_i = 1$

Approaches or variants of Markowitz Portfolio on the Internet

1. Fu et al., (1998) : Approximation algorithms for QP. Quadratic Time complexity.
2. Kamath et al., (1992) : Interior Point Approach for QP
3. Faaland (1974) : Quadratic Integer Programming. "Integer" because the investor limits the number of securities (증권) of a portfolio. Applied Knapsack (similar!) but the problem is different.

laxed problem. The enumeration scheme selected is motivated by the successful algorithm due to Gilmore and Gomory [3] for the knapsack problem.

Step 1. Reorder the variables $X_1, \cdots, X_n$ so that $C_1/a_1 \geq C_2/a_2 \geq \cdots \geq C_n/a_n$, and reorder the risk levels $B_1, \cdots, B_p$ so that $B_1 > B_2 > \cdots > B_p$.

Step 2. Calculate

$$Q = \min_{\sum_{j=1}^{n} X_j = K;\ X_j \geq 0,\ \text{integer}} \{ \textstyle\sum_{j=1}^{n} Q_j X_j^2 \}.$$

Reset p (if necessary) at the largest t for which $B_t - Q \geq 0$, and set

$$b_t = (B_t - Q)^{1/2}, \qquad t = 1, \cdots, p.$$

The calculation of Q may be done in a straightforward manner using dynamic programming. Let $f_1(k) = Q_1 k^2$ for $k = 0, 1, \cdots, K$, and for $r \geq 2$, let

$$f_r(k) = \min_{0 \leq X_r \leq k;\ X_r\ \text{integer}} \{Q_r X_r^2 + f_{r-1}(k - X_r)\}.$$

Then $Q = f_n(K)$. The term b_t computed in Step 2 is used as the right hand side of a relaxed linearized form of constraint (2.2). The algorithm applies a modified version of the Gilmore-Gomory enumeration scheme to relaxed problems of the form

(7.1) maximize $\sum_{j=1}^{n} C_j X_j$

(7.2) subject to $\sum_{j=1}^{n} a_j X_j \leq b_t$

(7.3) $\sum_{j=1}^{n} X_j = K$

(7.4) $X_j \geq 0$, integer, $\quad j = 1, \cdots, n.$

Figure: Source : Faaland (1974)

Conclusion

1. Want to solve Markowitz Portfolio : Find the proportion which maximizes return but keeps variability low
2. Relax the original problem to 1) apply class concepts 2) not to resort to optimization knowledge
3. **Brute Force** : For every grid of x, calculate the objective. $T(n) \in \Theta(\frac{n^{10^k+2}}{10^k!})$ for general C, $S(n) \in \Theta(\frac{n^{10^k}}{10^k!})$.
4. For Diagonal C, Greedy Algorithm was tempting but did not work.
5. **Dynamic Programming** approach is a modification of Knapsack problem. $T(n) \in \Theta(100^k \cdot n)$, $S(n) \in \Theta(10^k \cdot n)$
6. Since we deal with $k = 1$ or 2 while $n >> k$, **DP** works more efficiently than **BF**.
7. Not only solving the specific portfolio problem, this can be generalized to various scope of **proportion allocation**.

References

1) Markowitz, H. (1959). Portfolio selection. : Original Markowitz Portfolio

2) Faaland, B. (1974). An integer programming algorithm for portfolio selection. Management Science, 20(10), 1376-1384.
: Integer Programming version of Markowitz Portfolio

3) Fu, M., Luo, Z. Q., Ye, Y. (1998). Approximation algorithms for quadratic programming. Journal of combinatorial optimization, 2(1), 29-50.
: One way to solve Markowitz Portfolio

4) A.P. Kamath, N.K. Karmarkar, K.G. Ramakrishnan, and M.G.C. Resende, "A continuous approach to inductive inference," Mathematical Programming, vol. 57, pp. 215–238, 1992.

5) Algorithm Lecture #8 Dynamic Programming class material : For Knapsack Image