Portfolio Optimization
Using Minimax Rule

Ιωάννης Κοκκινίδης

ΑΜ 1053535, ΗΜΤΥ

Εισαγωγή

Το πρόβλημα της βελτιστοποίησης χαρτοφυλακίου (portfolio optimization) είναι ένα κλασσικό πρόβλημα στον χώρο των οικονομικών. Ο στόχος του είναι η βέλτιστη κατανομή των οικονομικών πόρων σε ένα χαρτοφυλάκιο, έτσι ώστε να μεγιστοποιηθεί το αναμενόμενο κέρδος, ελαχιστοποιώντας ταυτόχρονα το ρίσκο.

Γενικότερα, η επένδυση ολόκληρου του κεφαλαίου σε ένα μόνο προϊόν (asset) ενέχει υψηλό ρίσκο σε περίπτωση που αυτό δεν αποδώσει κατά τα αναμενόμενα.

Όταν όμως, ένα χαρτοφυλάκιο περιλαμβάνει διαφορετικά προϊόντα, κι ειδικότερα διαφορετικών τομέων (sectors), μειώνεται το ρίσκο να έχουν όλα πτωτική πορεία, δηλαδή να έχουμε ζημία.

Αυτό ονομάζεται διαφοροποίηση (diversification) ή διάχυση ρίσκου κι ο γραμμικός προγραμματισμός είναι ένα πολύ χρήσιμο εργαλείο για την βέλτιστη υλοποίηση της.

Περιγραφή Προβλήματος και Μοντελοποίηση

Σε αυτή την εργασία θα ασχοληθούμε με το Minimax μοντέλο του Martin R. Young [1]. Σύμφωνα με τον Young, επιλέγεται το χαρτοφυλάκιο που ελαχιστοποιεί τις μέγιστες απώλειες σε κάθε περίοδο παρακολούθησης, όταν η μέση απόδοσή του είναι φραγμένη από κάποιο κάτω όριο.

Το παραπάνω πρόβλημα μετασχηματίζεται στο πιο κατανοητό μοντέλο, όπου επιλέγεται το χαρτοφυλάκιο που μεγιστοποιεί τη μέση απόδοσή του, όταν η απόδοσή του σε κάθε περίοδο παρακολούθησης είναι φραγμένη από κάποιο κάτω όριο.

Μόνη προϋπόθεση για να αξιοποιήσουμε το μοντέλο του Young είναι να έχουμε στη διάθεσή μας ιστορικά δεδομένα αποδόσεων (returns).

Το μοντέλο για N προϊόντα και T περιόδους παρακολούθησης, (j∈N, t∈T):

$$y_{jt} = Απόδοση\ προϊόντος\ j\ την\ περίοδο\ t$$

$$\bar{y_j} = \frac{1}{T}\sum_{t=1}^{T} y_{jt} = Μέση\ απόδοση\ προϊόντος\ j$$

$$w_j = Ποσοστό\ κατανομής\ χαρτοφυλακίου\ στο\ προϊόν\ j$$

$$y_{pt} = \sum_{j=1}^{N} w_j * y_{jt} = Απόδοση\ χαρτοφυλακίου\ την\ περίοδο\ t$$

$$E_p = \sum_{j=1}^{N} w_j * \bar{y_j} = Μέση\ απόδοση\ χαρτοφυλακίου$$

Σκοπός μας είναι να μεγιστοποιήσουμε το E_p , όταν $y_{pt} \geq H$ για κάθε t, όπου H κάποιο επιθυμητό όριο. Δηλαδή:

$max \sum_{j=1}^{N} w_j * \bar{y_j}$ (1)

όταν

$\sum_{j=1}^{N} w_j * y_{jt} \geq H$, για t = 1, 2, … , T (2)

$\sum_{j=1}^{N} w_j \leq 1$ (3)

$w_j \geq 0$, για j = 1, 2, …, N (4)

Για την ακόμα καλύτερη διαφοροποίηση του χαρτοφυλακίου προστέθηκαν δυο ακόμα περιορισμοί.

- Κανένα προϊόν δεν μπορεί να κατέχει πάνω από το 15% του χαρτοφυλακίου:

  $w_j \leq 0.15$, για j = 1, 2, …, N (5)

- Λαμβάνοντας υπόψιν ότι, το κάθε προϊόν ανήκει σε κάποιον τομέα i (sector), τότε κάθε τομέας δεν μπορεί να κατέχει πάνω από ένα όριο Li του χαρτοφυλακίου:

  $\sum_{j=1}^{N} w_j \leq L1$, j∈S1 (6)
  $\sum_{j=1}^{N} w_j \leq L2$, j∈S2 (7)
  κ.ο.κ.

  Δηλαδή το άθροισμα των ποσοστών όλων των προϊόντων ενός τομέα είναι φραγμένο από ένα ανώτατο όριο, το οποίο μπορεί είναι διαφορετικό για κάθε τομέα.

Υλοποίηση

Οι μεταβλητές απόφασης είναι οι w_j, δηλαδή τα «βάρη» των προϊόντων που επιλέγονται για το χαρτοφυλάκιο (όπου ορίζεται κι ο περιορισμός (4):

```
# Decision variables
W = []
for i in range(N):
    W.append(LpVariable(name='W{}'.format(i), lowBound=0, upBound=1))
```

Αντικειμενική συνάρτηση (1):

```
# Objective function
prob += lpSum([mean[i]*W[i] for i in range(N)]), "Average Return"
```

Περιορισμοί (2),(3):

```
# Return of each period examined greater than Threshold H
for t in range(1, T):
    prob += lpSum([W[i]*y.iloc[i,t] for i in range(N)]) >= H, "Return of {} period".format(t)

prob += lpSum([W[i] for i in range(N)]) <= 1, "Sum of weights less than 1"
```

Επιπλέον περιορισμοί (5),(6):

```
# Weight of each stock less than 0.15 (No stock can account for more than 15% of the portfolio)
for i in range(N):
    prob += W[i] <=0.15, "Weight {} less than 0.15".format(i)

# Sector constraints (No sector can account for more than each limit of the portfolio)
prob += lpSum([W[i] for i in Info_Tech]) <= 0.4, "Info_Tech"
prob += lpSum([W[i] for i in Consumer_Staples]) <= 0.2, "Consumer_Staples"
prob += lpSum([W[i] for i in Consumer_Discretionary]) <= 0.3, "Consumer_Discretionary"
prob += lpSum([W[i] for i in Energy]) <= 0.15, "Energy"
prob += lpSum([W[i] for i in Financials]) <= 0.3, "Financials"
prob += lpSum([W[i] for i in Health_Care]) <= 0.3, "Health_Care"
prob += lpSum([W[i] for i in Industrials]) <= 0.2, "Industrials"
prob += lpSum([W[i] for i in Materials]) <= 0.1, "Materials"
prob += lpSum([W[i] for i in Real_Estate]) <= 0.1, "Real_Estate"
prob += lpSum([W[i] for i in Communication_Services]) <= 0.25, "Communication_Services"
prob += lpSum([W[i] for i in Utilities]) <= 0.1, "Utilities"
```

Για το dataset συλλέχθηκαν τα δεδομένα τιμών των 50 μεγαλύτερων μετοχών του δείκτη S&P500 (weight), της τελευταίας πενταετίας από το Yahoo Finance [4] (αρχείο data.xlsx).

Με περίοδο τον ένα μήνα, και για λόγους ευκολίας, για την εύρεση του βέλτιστου χαρτοφυλακίου αξιοποιήθηκαν τα δεδομένα από 1-1-2019 έως 1-1-2023 (T=48).

```
# Import data from excel file into a dataframe
df = pd.read_excel("data.xlsx")

# Get the return of each period/month for every stock (4 years = 48 months)
y = df.iloc[:,4:T+5].pct_change(axis='columns') #y(it) = (q(it) - q(it-1))/q(it-1)

# Get the mean of the returns for every stock
mean = y.mean(axis=1) #y(i) = (y(i1) + y(i2) + ... + y(iT))/T
```

Τέλος, έχοντας δώσει σαν παράμετρο στην επίλυση του προβλήματος το συνολικό πόσο επένδυσης, τα αποτελέσματα για την επιλογή μετοχών και τη βέλτιστη κατανομή του ποσού σε αυτές, εξάγονται στο αρχείο export.xlsx για πιο εύκολη ανάγνωση του αποτελέσματος.

```
# Export results
export = pd.DataFrame(columns=[0,1,2,3], dtype=object)
j=1
for i in range(N):
    if (W[i].varValue != 0):
        export.loc[j, 0] = df.iloc[i, 0]
        export.loc[j, 1] = df.iloc[i, 1]
        export.loc[j, 2] = df.iloc[i, 3]
        export.loc[j, 3] = W[i].varValue * B
        j+=1
export.to_excel("export.xlsx", index=False, header=["Stock", "Symbol", "Sector", "Allocation"])
```

Αποτελέσματα - Συμπεράσματα

Με N=50 (top50 of S&P500), T=48 (months) και B=100.000$ και
H = -0.15 (threshold) παίρνουμε:

```
Status: Optimal
Expected monthly portfolio return =  0.03573174228924071
Weight of Apple Inc = 0.1 or 10000.0$
Weight of Nvidia Corp = 0.15 or 15000.0$
Weight of Tesla Inc = 0.15 or 15000.0$
Weight of Eli Lilly & Co = 0.15 or 15000.0$
Weight of Danaher Corp = 0.15 or 15000.0$
Weight of Advanced Micro Devices = 0.15 or 15000.0$
Weight of Conocophillips = 0.15 or 15000.0$
Sum of weights =  1.0
```

Για την τελική αξιολόγηση του μοντέλου και του βέλτιστου χαρτοφυλακίου αξιοποιήθηκαν τα δεδομένα που συλλέχτηκαν για το 2023, δηλαδή από 1-1-2023 έως 1-9-2023 (8 μήνες).

```python
# Test the portfolio built by the model
def test(df, result, weights):
    future = df.iloc[:,4+T::8].pct_change(axis='columns') # Return from 1/1/2023 to 1/9/2023 (8 months)
    sum = 0
    for i in result:
        sum += (1+future.iloc[i,1:]) * weights[i] * B
    total_return = (sum - B)/B
    return total_return.iloc[0]
```

Το μοντέλο απέδωσε 37.67% για τους πρώτους 8 μήνες του 2023:

```
Total return on portfolio (in 2023):  0.3766825596506192
```

Στο ίδιο διάστημα, ο S&P500 είχε απόδοση (περίπου) 10.8%.

Βιβλιογραφία

[1] Young (1998). A minimax-portfolio selection rule with linear programming solution, Management Science

[2] Mansini R., Ogryczak W., and Speranza M.G., (2015) Linear and Mixed Integer Programming for Portfolio Optimization

[3] Papahristodoulou, C., Optimal portfolios using Linear Programming models

[4] https://finance.yahoo.com/