El problema de la cartera de valores

Penadillo Lazares Wenses Johan

UNIVERSIDAD NACIONAL DE INGENIERÍA

9 de octubre de 2021

Índice

Explicación

Un inversor es propietario de b_i participaciones de los valores bursátiles $A_i, i = 1, 2, \ldots, m$. Los precios actuales de estos valores son v_i. Considerando que se puede predecir los dividendos que se pagarán al final del año que comienza y los precios finales de los valores bursátiles. A_i pagará d_i y tendrá un nuevo precio w_i.

Objetivo

El objetivo sera ajustar la cartera, es decir, el numero de participaciones en cada valor, de modo que se maximicen los dividendos. Las incógnitas serán x_i, el cambio en el número de participaciones que ya se tienen.

Datos

- m: el numero de valores bursátiles
- b_i: el numero actual de participaciones del valor bursátil i
- v_i: el precio actual del valor i por participación
- d_i: el dividendo que se pagará al final del año en el valor bursátil i
- w_i: el nuevo precio del valor bursátil i
- r: porcentaje mínimo r del valor actual de toda la cartera que no debe superarse en el ajuste
- s: porcentaje mínimo del valor total actual que no debe superarse por el valor futuro total de la cartera, para hacer frente a la inflación.

Restricciones

El numero de participaciones debe ser no negativa.

$$x_i + b_i \geq 0$$

La cartera debe evitar depender en exceso de un valor cualquiera; esta condición puede establecerse exigiendo que el capital asociado a todo valor concreto, después del ajuste, represente al menos una cierta fracción r del capital total actual de la cartera.

$$r(\sum v_i(b_i + x_i)) \leq v_j(b_j + x_j); \forall j$$

Restricciones

El capital total de la cartera no debe cambiar en el ajuste.

$$\sum v_i x_i = 0$$

Para hacer frente a la inflación, el capital total en el futuro debe ser al menos un cierto porcentaje s mayor que el capital invertido actualmente.

$$\sum w_i(b_i + x_i) \geq (1 + s) \sum v_i b_i$$

Función a optimizar

Como el objetivo es maximizar los dividendos.

$$Z = \sum d_i(b_i + x_i)$$

```
import math
import numpy as np


def get_pivot_position(tableau):
    z = tableau[-1]
    column = next(i for i, x in enumerate(z[:-1]) if x > 0)

    restrictions = []
    for eq in tableau[:-1]:
        el = eq[column]
        restrictions.append(math.inf if el <= 0 else eq[-1] / el)

    if (all([r == math.inf for r in restrictions])):
        raise Exception("Linear program is unbounded.")

    row = restrictions.index(min(restrictions))
    return row, column


def pivot_step(tableau, pivot_position):
    new_tableau = [[] for eq in tableau]

```

```
    i, j = pivot_position
    pivot_value = tableau[i][j]
    new_tableau[i] = np.array(tableau[i]) / pivot_value

    for eq_i, eq in enumerate(tableau):
        if eq_i != i:
            multiplier = np.array(new_tableau[i]) * tableau[eq_i][j]
            new_tableau[eq_i] = np.array(tableau[eq_i]) - multiplier

    return new_tableau


def is_basic(column):
    return sum(column) == 1 and len([c for c in column if c == 0]) == l


def get_solution(tableau):
    columns = np.array(tableau).T
    solutions = []
    for column in columns[:-1]:
        solution = 0
        if is_basic(column):
            one_index = column.tolist().index(1)
            solution = columns[-1][one_index]
```

```
            solutions.append(solution)

    return solutions


def to_tableau(c, A, b):
    xb = [eq + [x] for eq, x in zip(A, b)]
    z = c + [0]
    return xb + [z]


def can_be_improved(tableau):
    z = tableau[-1]
    return any(x > 0 for x in z[:-1])


def simplex(c, A, b):
    tableau = to_tableau(c, A, b)

    while can_be_improved(tableau):
        pivot_position = get_pivot_position(tableau)
        tableau = pivot_step(tableau, pivot_position)

    return get_solution(tableau)
```

```
c = [0, 3, 5, 0, 0, 0, 0, 0, 0, 0]
A = [
    [-1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, -1, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, -1, 0, 0, 1, 0, 0, 0, 0],
    [-15, 5, 25, 0, 0, 0, 1, 0, 0, 0],
    [5, -15, 25, 0, 0, 0, 0, 1, 0, 0],
    [5, 5, -75, 0, 0, 0, 0, 0, 1, 0],
    [-18, -23, -102, 0, 0, 0, 0, 0, 0, 1],
    [20, 20, 100, 0, 0, 0, 0, 0, 0, 0]
]
b = [75, 100, 35, -250, 250, 1750, -1880, 0]

solution = simplex(c, A, b)
print('solution: ', solution)
```