Lab 7: Portfolio Optimization

Learning Objectives:

- Transform non-linear constraints into linear constraints using auxiliary variables. (Analyze)
- Write code on paper to implement an optimization formulation. (Code)

Lab 7 requires using the techniques described last class (see course notes here). To prepare us for the lab, we first consider an example illustrating the techniques.

Example 2 From Last Class

Example 2 from last class (see handout here) describes the following non-linear optimization problem, which adds certain constraints to the Lab 5 LP. The goal is to transform it into a linear formulation by adding auxiliary variables. This is called **linearizing the non-linear program.**

Non-linear formulation:

Decision variable:

- Let x_{ijk} denote the number of units of item k to ship from FC i to region j.(continuous)
- Let t_{ik} denote the total inventory for item k at FC i. (continuous)

Optimization program (not linear):

Maximize: $1.38 \sum_{i,j,k} w_k d_{ij} x_{ijk} + 0.01 \sum_{i,k} |y_{ik} - t_{ik}|$

subject to:

(FC capacity)	$\sum_{j,k} s_k x_{ijk} \leq q_i$	for all fulfilment center i.
(Satisfying all demand)	$\sum_{i} x_{ijk} \geq d_{jk}$	for all region j and item k.
(Total inventory)	$t_{ik} = \sum_{j} x_{ijk}$	for all FC i and item k.
(At most 5 FC)	# of i for which $t_{ik} > 0$ is at most 5	for each item k.
(Minimum inventory)	t_{ik} is either 0 or at least 100	for each i and k.
(Non-negativity)	$x_{ijk} \geq 0$	for all i, j, and k.

Linearized formulation

The idea is to create an auxiliary variable z_{ik} to turn on/off the variable t_{ik} for item k at FC i. Moreover, we encode the absolute value constraint by noting that $|y_{ik} - t_{ik}| = \max(y_{ik} - t_{ik}, -(y_{ik} - t_{ik}))$.

Decision variable:

- Let x_{ijk} denote the number of units of item k to ship from FC i to region j.(continuous)
- Let t_{ik} denote the total inventory for item k at FC i. (continuous)
- Let z_{ik} denote whether we place item k at FC i. (binary)
- Let a_{ik} be an auxiliary variable to linearize each absolute value term. (continuous)

Linear program:

Maximize: $1.38 \sum_{i,j,k} w_k d_{ij} x_{ijk} + 0.01 \sum_{i,k} a_{ik}$

subject to:

(FC capacity)	$\sum_{j,k} s_k x_{ijk} \leq q_i$	for all fulfilment center i.
(Satisfying all demand)	$\sum_{i} x_{ijk} \geq d_{jk}$	for all region j and item k.
(Total inventory)	$t_{ik} = \sum_{j} x_{ijk}$	for all FC i and item k.
(Inventory bounds)	$100 z_{ik} \leq t_{ik} \leq (\sum_{j} d_{jk}) z_{ik}$	for each i and k.
(At most 5 FC)	$\sum_{i} z_{ik} \leq 5$	for each item k.
(Absolute value-1)	$y_{ik} - t_{ik} \leq a_{ik}$	for each i and k.
(Absolute value-2)	$-(y_{ik} - t_{ik}) \leq a_{ik}$	for each i and k.
(Non-negativity)	$x_{ijk} \geq 0$	for all i, j, and k.

We illustrate the same techniques on another problem from finance (Lab 7).

Lab 7 Problem Description

Shanice is an analyst at Trojan capital management, which has recently been exploring optimization-based techniques for portfolio management. Since Shanice received training in optimization as part of her Master's degree, she has been given the task of implementing a prototype of the following optimization in Python.

Data:

- S: the set of stocks.
- w_i: the old weight of stock $i \in S$ before optimization. (The "weight" of a stock is % of total funds invested in the stock; weights of all stocks should add to one.)
- R_i: the expected annual return of stock $i \in S$.
- C_{ij}: the estimated covariance between stocks $i, j \in S$.
- σ_{target}: the maximum volatility of the final portfolio.
- δ: the maximum total change allowed between the old weights and the new weights.
- k: the maximum # of stocks allowed in the portfolio.
- ϵ: the minimum non-zero weight allowed.

Decision variables:

- x_i: the new weight of stock i. (Continuous)

Formulation (to be linearized): All summations are over the set S of stocks.

$$
\begin{aligned}
&\text{Maximize:} && \sum_i R_i x_i && \text{(Average Return)}\\
&\text{subject to:}\\
&\text{(Valid weights)} && \sum_i x_i = 1\\
&\text{(Risk tolerance)} && \sum_{i,j} C_{ij} x_i x_j \le \sigma^2_{target}\\
&\text{(Change in weights)} && \frac{1}{2}\sum_i |x_i - w_i| \le \delta\\
&\text{(Simplicity)} && (\#\text{ of stock } i \text{ with } x_i > 0) \le k\\
&\text{(Non-negligible weights)} && \text{If } x_i > 0 \text{ then } x_i \ge \epsilon && \text{for each stock } i.\\
& && x_i \ge 0
\end{aligned}
$$

However, the last three constraints are not allowed in Gurobi, as they are not linear. (The risk tolerance constraint, on the other hand, is allowed in Gurobi because the LHS can be expressed as a sum of squares.)

Linearized Formulation

Exercise A. Use auxiliary decision variables to rewrite the last three constraints in a linear way. Then rewrite the entire formulation on a blank sheet of paper.

Decision variables:

- x_i: the new weight of stock i. (Continuous)
- z_i: whether to give stock i a positive weight. (Binary)
- a_i: an auxiliary variable to encode $|x_i - w_i|$. (Continuous)

Formulation (last 3 constraints linearized):

$$
\begin{aligned}
&\text{Maximize:} && \sum_i R_i x_i && \text{(Average Return)}\\
&\text{subject to:}\\
&\text{(Valid weights)} && \sum_i x_i = 1\\
&\text{(Risk tolerance)} && \sum_{i,j} C_{ij} x_i x_j \le \sigma^2_{target}\\
&\text{(Change in weights)} && \frac{1}{2}\sum_i a_i \le \delta\\
&\text{(At most } k \text{ stocks)} && \sum_i z_i \le k\\
&\text{(Absolute value-1)} && x_i - w_i \le a_i && \text{for each stock } i.\\
&\text{(Absolute value-2)} && -(x_i - w_i) \le a_i && \text{for each stock } i.\\
&\text{(Stocks to include)} && x_i \le z_i && \text{for each stock } i.\\
&\text{(Non-negligible weights)} && x_i \ge \epsilon z_i && \text{for each stock } i.\\
& && x_i, a_i \ge 0\\
& && z_i \in \{0,1\}
\end{aligned}
$$

Python Code on Paper

Exercise B. Write code on a piece of paper to implement the formulation from part a) in Python and Gurobi. The code should output the final portfolio in an Excel sheet named

`portfolio.xlsx` **where the first column is the name of the stock and the second column is the corresponding weights. The output file should only contain stocks with positive weights.**

You may assume that the inputs are supplied in the following formats:

- w_i: A DataFrame named `oldPortfolio` with row index being the stock i and a column named `Weight` containing the w_i. (see below)
- R_i: A Series named `ret` with mapping each stock i to its expected annual return R_i. (see below)
- C_{ij}: A DataFrame named `cov` with row index being the i's and column index being the j's. (see below)
- σ_{target}: a variable named `stdMax`.
- δ: a variable named `maxChange`.
- k: a variable named `k`.
- ϵ: a variable named `eps`.

[1]:
```
import gurobipy as grb
mod=grb.Model()

# Obtaining the index set
S=ret.index

# Defining decision variables
x={}
z={}
a={}
for i in S:
    x[i]=mod.addVar(lb=0)
    z[i]=mod.addVar(vtype=grb.GRB.BINARY)
    a[i]=mod.addVar(lb=0)

# Alternatively, you can define variables as follows. (See solution to Homework 8 for fu
# x=mod.addVars(S,lb=0)
# z=mod.addVars(S,vtype=grb.GRB.BINARY
# a=mod.addVars(S,lb=0)

# Setting the objective
mod.setObjective(sum(ret.loc[i]*x[i] for i in S),sense=grb.GRB.MAXIMIZE)

# Defining constraints
mod.addConstr(sum(x[i] for i in S)==1)
mod.addConstr(sum(cov.loc[i,j]*x[i]*x[j] for i in S for j in S)<=stdMax**2)
mod.addConstr(sum(a[i] for i in S)/2<=maxChange)
mod.addConstr(sum(z[i] for i in S)<=k)

for i in S:
    mod.addConstr(x[i]-oldPortfolio.loc[i,'Weight']<=a[i])
    mod.addConstr(-x[i]+oldPortfolio.loc[i,'Weight']<=a[i])
    mod.addConstr(x[i]<=z[i])
    mod.addConstr(x[i]>=eps*z[i])

mod.optimize()
```

```
table=[]
for i in S:
    if x[i].x>0:
        table.append([i,x[i].x])
portfolio=pd.DataFrame(table,columns=['Stock','Weight'])
portfolio=portfolio.sort_values(by='Weight',ascending=False)
portfolio.to_excel(outputFile,index=False)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-1-22c2e7e6af7c> in <module>()
  3
  4 # Obtaining the index set
----> 5 S=ret.index
  6
  7 # Defining decision variables


NameError: name 'ret' is not defined
```

To see this code in action, see this supplementary portfolio optimization tutorial based on Lab 7. This tutorial create the inputs from data downloaded from Yahoo finance, and illustrate the use of sensitivity analysis to select the parameters k, δ, σ_{target}. It also illustrate qualitative insights based on the analysis.