DATA 609: Final Project - A Simple Convex Optimization on Regulatory Financial Reporting

Eddie Xu

Introduction

All financial institutions are required to report their financial data in accordance to multiple regulatory frameworks such as Basel III and Dodd Frank Act. Due to the complexities with some of the framework, it is challenging for some financial companies, especially in smaller regional or community banks. They need to maximize their returns and ensure that they are to be in regulatory compliance on limited resources and time. This can cause potential reporting errors and lead financial institutions to be penalized with fines that are millions of dollars and have a significant impact on the financial market.

Abstract

This project is a modification of Lab 2, Problem 3. The goal is to leverage convex optimization applications for a simple regulatory financial reporting problem. In this problem, the focus is the optimal allocation of a proposed bank's capital across different assets (cash, bonds, stocks, etc) in order to maximize return using the Markowitz Portfolio Optimization. This will minimize the market risks while ensuring compliance with regulatory capital requirements under the Basel III.

Data Pulling

The data is pulled from Lab 2 Problem 3 as a source for the financial assets needed for a small financial institution's portfolio. It's information on daily asset returns from 2020-2024 for a group of assets, consisting mostly of large-cap stocks as exchange traded funds that correspond to US Treasury Bonds and Notes with varying maturities will be used for this project.

```python
# load dependencies
import numpy as np
import pandas as pd
import cvxpy as cp
import matplotlib.pyplot as plt
import datetime as dt
from scipy.stats import norm

# read in data
stock_url = 'https://media.githubusercontent.com/media/georgehagstrom/DATA609Spring2025/refs/
fin_tidy_data = pd.read_csv(stock_url)
fin_tidy_data['date'] = pd.to_datetime(fin_tidy_data['date'])

print(fin_tidy_data)
```

```
       Company       date    adjusted    return  log_return
0         AAPL 2020-01-02   72.796028  1.022817    0.022560
1         AAPL 2020-01-03   72.088295  0.990278   -0.009770
2         AAPL 2020-01-06   72.662704  1.007968    0.007937
3         AAPL 2020-01-07   72.320969  0.995297   -0.004714
4         AAPL 2020-01-08   73.484352  1.016086    0.015958
...        ...        ...         ...       ...         ...
592042     SHV 2024-12-23  109.588890  1.000091    0.000091
592043     SHV 2024-12-24  109.618782  1.000273    0.000273
592044     SHV 2024-12-26  109.638718  1.000182    0.000182
592045     SHV 2024-12-27  109.668602  1.000273    0.000273
592046     SHV 2024-12-30  109.698494  1.000273    0.000273

[592047 rows x 5 columns]
```

Convex Optimization

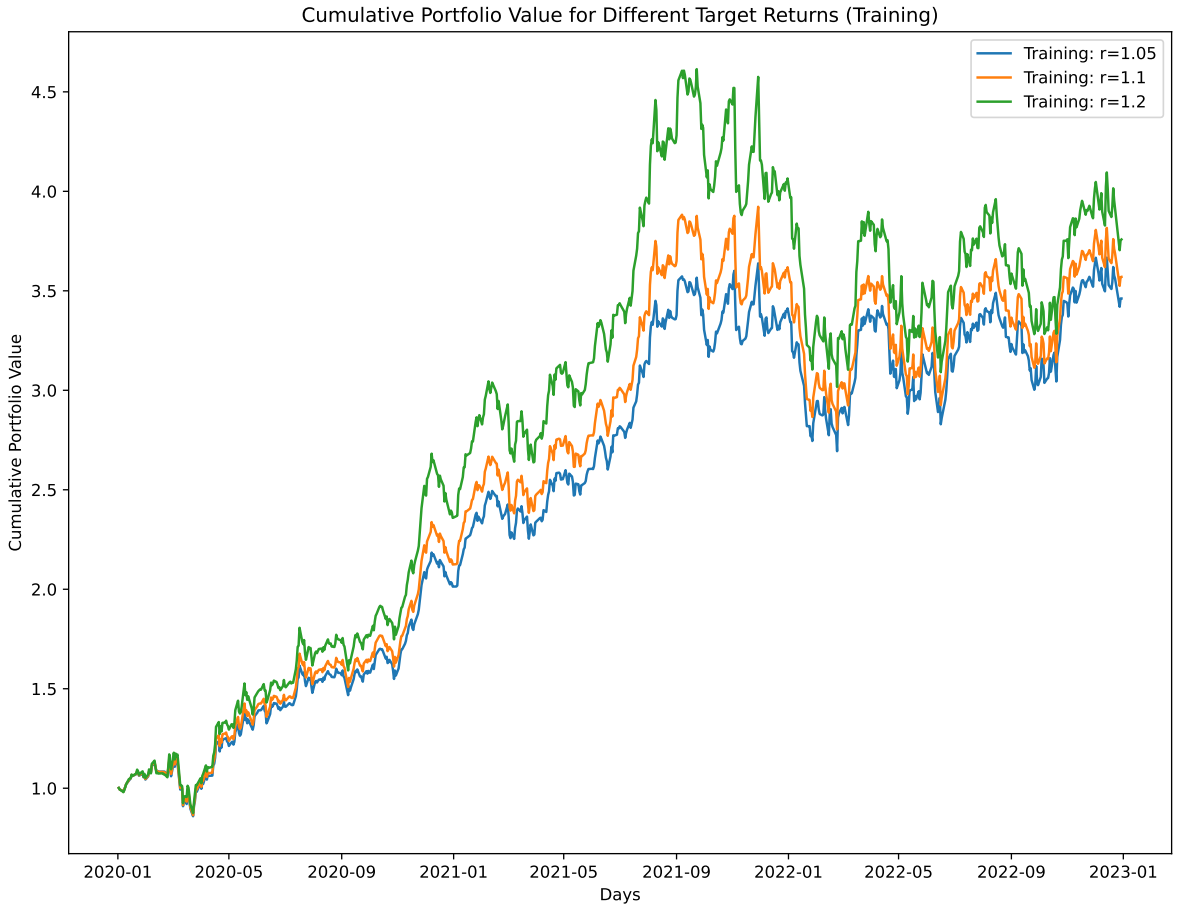
For the convex optimization problem, the data will be split into two time periods, a training period (2020-2022) and a testing period (2022-2024).The log return will be used on the Markowitz Portfolio Optimization where the number of target returns (1.05, 1.10, and 1.20) are defined and the market risks are minimized while ensuring compliance with regulatory capital requirements under the Basel III.

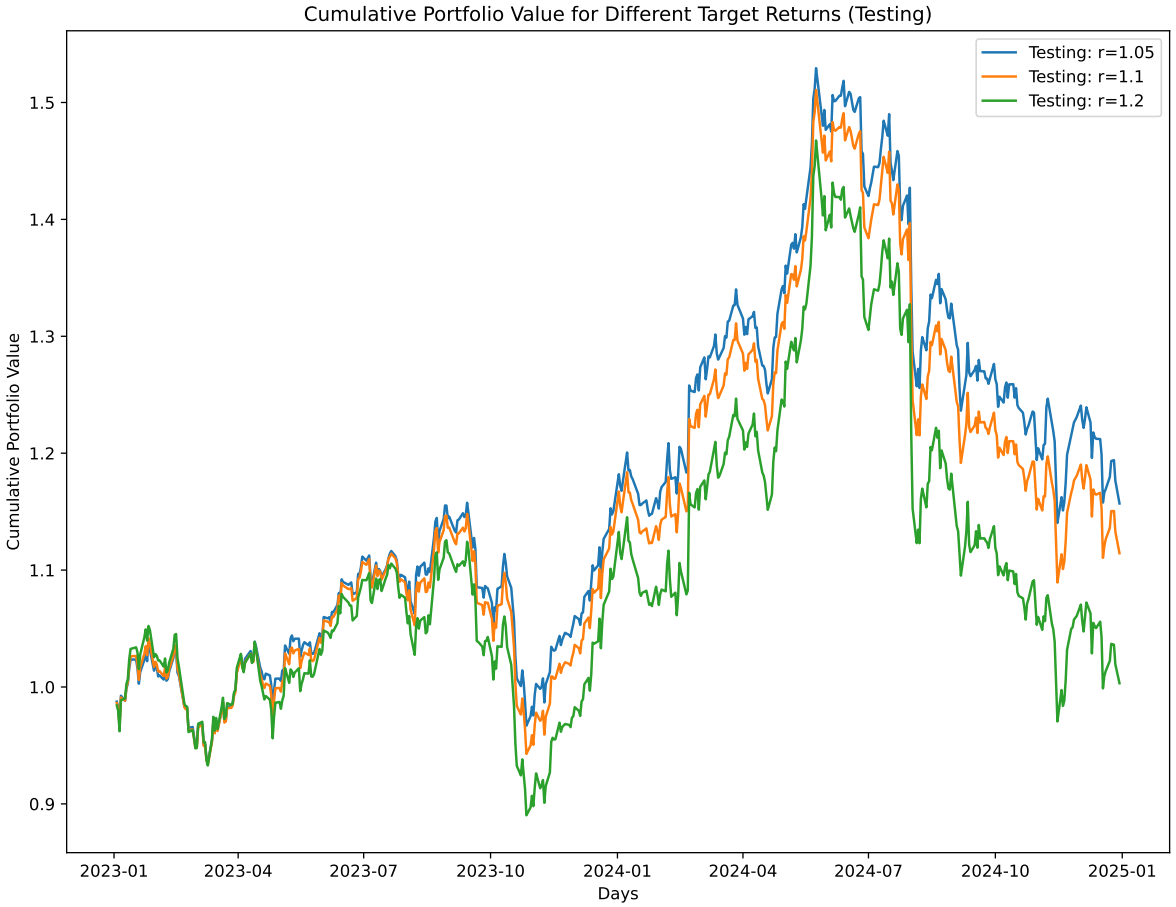
The constraint are defined from the Basel III requirements and for its simplicity, it will use few parameters. The capital ratio is set to 8% and captial available is set to 10%. The rules will also include conditional Value at Risks is a risk measure that show the average of extreme losses that exceed Value at Risk and is defined as 95% in the constraint as well.

```
# filter data into training (2020-2022) and testing (2022-2024) periods
train_period = fin_tidy_data[(fin_tidy_data['date'] >= '2020-01-01') & (fin_tidy_data['date']
test_period = fin_tidy_data[(fin_tidy_data['date'] > '2022-12-31')]

# convert the data with the log return
train_data = train_period.pivot(index='date', columns='Company', values='log_return')
test_data = test_period.pivot(index='date', columns='Company', values='log_return')

# calculate the annual mean and covariance matrix
mu = np.exp(0.5 * train_data.sum()) - 1
gamma = train_data.cov().values

# define variables
r_targets = [1.05, 1.10, 1.20]
X = train_data.values
T, n = X.shape

optimal_weights = {}
portfolio_cum_returns_train = {}
portfolio_cum_returns_test = {}

# define parameters based on Basel III and CVaR
capital_ratio = 0.08
capital_available = 0.10
risk_weights = np.ones(n)

alpha = 0.95
max_cvar = 0.05

for r in r_targets:
    # define the variable
    w = cp.Variable(len(mu))
    z = cp.Variable(T)
    VaR = cp.Variable()

    # calculate the portfolio
    portfolio_returns = X @ w
    losses = - portfolio_returns

    # calculate the conditional Value at Risk
    cvar = VaR + (1 / ((1 - alpha) * T)) * cp.sum(z)
```

```
    # define the objective function: 0.5 * w^T * Gamma * w
    objective = cp.Minimize(0.5 * cp.quad_form(w, gamma))

    # define the constraints
    constraints = [cp.sum(w) == 1,
            w.T @ mu >= r,
            w >= 0,
            cp.sum(cp.multiply(w, risk_weights)) * capital_ratio <= capital_available,
            z >= 0,
            z >= losses - VaR,
            cvar <= max_cvar]

    # Solve the problem
    problem = cp.Problem(objective, constraints)
    problem.solve()

    # store the optimal weight
    optimal_weights[r] = w.value

    # calculate portfolio returns for training period
    portfolio_returns_train = train_data.dot(w.value)
    portfolio_cum_returns_train[r] = np.cumprod(1 + portfolio_returns_train)

    # calculate portfolio returns for testing period
    portfolio_returns_test = test_data.dot(w.value)
    portfolio_cum_returns_test[r] = np.cumprod(1 + portfolio_returns_test)

# plot cumulative returns for training and testing periods
plt.figure(figsize=(12, 9))

for r in r_targets:
    plt.plot(portfolio_cum_returns_train[r], label=f'Training: r={r}')

plt.title('Cumulative Portfolio Value for Different Target Returns (Training)')
plt.legend()
plt.xlabel('Days')
plt.ylabel('Cumulative Portfolio Value')
plt.show()

plt.figure(figsize=(12, 9))
for r in r_targets:
    plt.plot(portfolio_cum_returns_test[r], label=f'Testing: r={r}')
```

```
plt.title('Cumulative Portfolio Value for Different Target Returns (Testing)')
plt.legend()
plt.xlabel('Days')
plt.ylabel('Cumulative Portfolio Value')
plt.show()
```



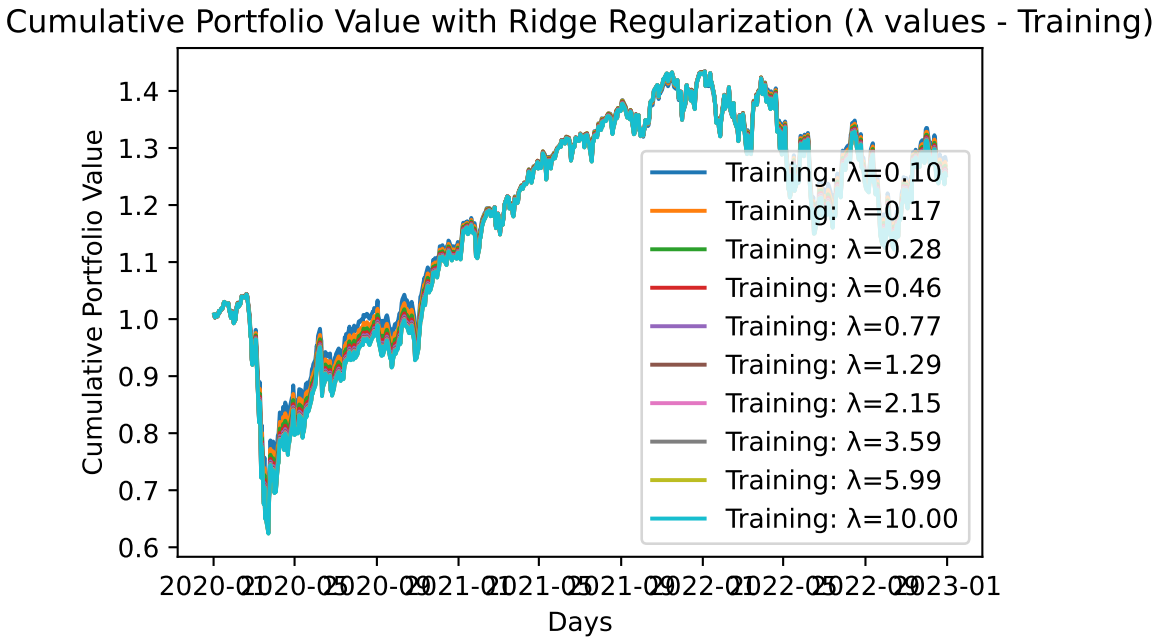


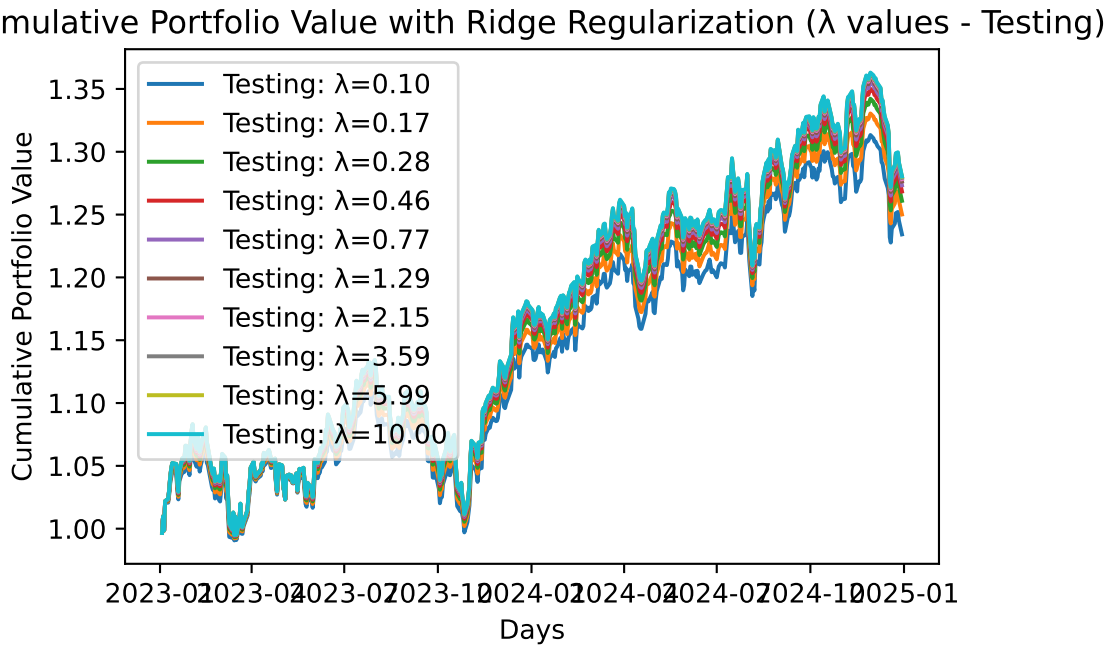
It is commonly known that optimal portfolios constructed using the Markowitz Portfolio Optimization performs poorly out of sample compared to in sample. Instead the ridge regularization and the set target return of 20% will be applied to the problem and be reevaluated.

```
# set the target return
r = 0.20

# define the lambda range
lambda_values = np.logspace(-1, 1, 10)

# define dicts for storing results
optimal_weights_ridge = {}
portfolio_cum_returns_train_ridge = {}
portfolio_cum_returns_test_ridge = {}

for lam in lambda_values:
```

```
    # define the variable
    w = cp.Variable(len(mu))
    z = cp.Variable(T)
    VaR = cp.Variable()

    # define the objective function: 0.5 * w^T * (Gamma + lambda * I) * w
    objective = cp.Minimize(0.5 * cp.quad_form(w, cp.psd_wrap(gamma + lam * np.eye(len(mu)))))

    # define the constraints
    constraints = [cp.sum(w) == 1,
            w.T @ mu == r, w >= 0,
            cp.sum(cp.multiply(w, risk_weights)) * capital_ratio <= capital_available,
            z >= 0,
            z >= losses - VaR,
            cvar <= max_cvar]

    # define the optimization problem
    problem = cp.Problem(objective, constraints)

    # Solve the problem
    problem.solve()

    # store the optimal weights
    optimal_weights_ridge[lam] = w.value

    # calculate portfolio returns for training data
    portfolio_returns_train = train_data.dot(w.value)
    portfolio_cum_returns_train_ridge[lam] = np.cumprod(1 + portfolio_returns_train)

    # calculate portfolio returns for testing data
    portfolio_returns_test = test_data.dot(w.value)
    portfolio_cum_returns_test_ridge[lam] = np.cumprod(1 + portfolio_returns_test)

# plot the lamda training period
for lam in lambda_values:
    plt.plot(portfolio_cum_returns_train_ridge[lam], label=f'Training:  ={lam:.2f}')

plt.title('Cumulative Portfolio Value with Ridge Regularization (  values - Training)')
plt.legend()
plt.xlabel('Days')
plt.ylabel('Cumulative Portfolio Value')
plt.show()
```

```
# plot the lamda testing period
for lam in lambda_values:
    plt.plot(portfolio_cum_returns_test_ridge[lam], label=f'Testing:  ={lam:.2f}')

plt.title('Cumulative Portfolio Value with Ridge Regularization (  values - Testing)')
plt.legend()
plt.xlabel('Days')
plt.ylabel('Cumulative Portfolio Value')
plt.show()
```





Observation and Insights

With a fully invested portfolio, it shows that the optimization tries to meet the target return while being compliance with the capital ratios and the CVar not exceeding over 5%. This is presented when the target return is set 1.2 as the portfolio shows to be more violiate on both training and testing data.

By introducing the ridge regularization to the convex optimization problem, it improves the stabilitiy and generalization of the portfolio. It shows improved trade-off between performance and risk epsecially when the lamda value is less than 1.