Portfolio Optimization

1 Key Concepts and Knowledge Review

1.1 Expected Return

Expected return is a measure of the average return anticipated on an asset or portfolio over a specific period, based on historical performance. It is the mean of past returns and serves as an estimate of future performance.

For an individual asset i over T periods, the expected return μ_i is calculated as:

$$\mu_i = \frac{1}{T} \sum_{t=1}^{T} A_{t,i}$$

where $A_{t,i}$ represents the return of asset i in period t.

For a portfolio with weights p, the expected return is a weighted sum of the expected returns of individual assets:

$$\mu^T p = \sum_{i=1}^{N} \mu_i p_i$$

where μ is the vector of expected returns, p is the vector of portfolio weights, and N is the number of assets.

1.2 Risk and Variance

Variance represents the risk or volatility of an asset's returns, measuring how much returns fluctuate around the mean. Higher variance indicates greater uncertainty in returns, which is perceived as higher risk.

For a portfolio, risk is typically modeled as the **portfolio variance**, which depends on both the variances of individual assets and their covariances.

1.3 Covariance and the Covariance Matrix

Covariance measures the degree to which two assets move in relation to each other: - *Positive covariance* indicates that two assets tend to move in the same direction. - *Negative covariance* indicates that two assets tend to move in opposite directions.

The **covariance matrix** V captures the variances and covariances of multiple assets in a portfolio. For assets i and j over T periods:

$$V_{i,j} = \frac{1}{T-1} \sum_{t=1}^{T} (A_{t,i} - \mu_i)(A_{t,j} - \mu_j)$$

The **portfolio variance**, which measures portfolio risk, is calculated as:

$$\text{Portfolio Variance} = p^T V p$$

where p represents the portfolio weights.

1.4 Optimization Objective: Risk-Adjusted Return

The goal of portfolio optimization is to balance return and risk, creating a portfolio that maximizes returns for a given level of risk, or vice versa. This is typically achieved by maximizing the **risk-adjusted return**:

$$\mu^T p - \lambda p^T V p$$

where λ is the **risk-aversion parameter**, which determines the trade-off between expected return and risk. A higher λ places more emphasis on risk minimization.

1.5 Common Constraints in Portfolio Optimization

Portfolio optimization typically involves constraints to reflect real-world investment requirements. Common constraints include:

- **Fully Invested**: $\sum p_i = 1$ ensures that all available capital is allocated across assets.
- **No Short-Selling**: $p_i \geq 0$, preventing negative weights (short positions) in any asset.
- **Minimum Return Requirement**: $\mu^T p \geq R_{\min}$, enforcing a minimum expected return for the portfolio.

2 Example 1: Simple Portfolio Optimization

This example introduces a simplified portfolio optimization scenario using hypothetical assets. Here, we aim to minimize risk while achieving a minimum expected return.

2.1 Problem Statement

Consider two assets, A and B, with the following characteristics:

- **Expected Returns**: $\mu_A = 0.02$ (2%), $\mu_B = 0.01$ (1%).
- **Variances**: $\text{Var}(A) = 0.0004$, $\text{Var}(B) = 0.0001$.
- **Covariance**: $\text{Cov}(A, B) = 0.00005$.

Objective: Minimize the portfolio variance $p^T V p$ while achieving a minimum expected return of 1.5%.

2.2 Solution with Explanation

The covariance matrix V and expected return vector μ are given by:

$$V = \begin{bmatrix} 0.0004 & 0.00005 \\ 0.00005 & 0.0001 \end{bmatrix}, \quad \mu = \begin{bmatrix} 0.02 \\ 0.01 \end{bmatrix}$$

We will use `cvxpy` to set up and solve the optimization problem.

```
import cvxpy as cp
import numpy as np

# Define the expected return vector and covariance matrix
mu = np.array([0.02, 0.01])
V = np.array([[0.0004, 0.00005], [0.00005, 0.0001]])

# Define the portfolio weights as optimization variables
p = cp.Variable(2)

# Objective: Minimize portfolio variance
objective = cp.Minimize(cp.quad_form(p, V))

# Constraints
constraints = [
    mu.T @ p >= 0.015,  # Minimum expected return of 1.5%
    cp.sum(p) == 1,     # Fully invested
    p >= 0              # No short-selling
]

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

# Display results
print("Optimal Weights:", p.value)
print("Expected Return:", mu.T @ p.value)
print("Portfolio Variance:", p.value.T @ V @ p.value)
```

- **Define Variables**: We start by defining p as the vector of portfolio weights.
- **Objective**: Our objective is to minimize $p^T V p$, representing the portfolio's risk.
- **Constraints**: We ensure that the portfolio achieves a minimum return of 1.5%, is fully invested, and avoids short-selling.

3 Example 2: Real-World Portfolio Optimization with Historical Data

In this example, we construct an optimal portfolio using real stock data (SPY, TSLA, and MSFT) over the past five years to maximize risk-adjusted return.

3.1 Problem Statement

1. Download historical monthly data for SPY, TSLA, and MSFT. 2. Calculate monthly returns. 3. Use these returns to compute the expected returns vector μ and covariance matrix V. 4. Objective: Maximize $\mu^T p - \lambda p^T V p$ with $\lambda = 0.5$.

3.2 Step-by-Step Solution with Explanation

```
import yfinance as yf
import cvxpy as cp
import numpy as np
import pandas as pd

# Step 1: Download historical data
tickers = ['SPY', 'TSLA', 'MSFT']
data = yf.download(tickers, start="2018-01-01", end="2023-01-01", interval="1mo")['Adj Close
returns = data.pct_change().dropna()

# Step 2: Calculate expected returns and covariance matrix
mu = returns.mean().values
V = returns.cov().values

# Step 3: Define the portfolio weights variable
p = cp.Variable(len(tickers))

# Step 4: Set the risk-aversion factor and define the objective function
lambda_ = 0.5
objective = cp.Maximize(mu.T @ p - lambda_ * cp.quad_form(p, V))

# Step 5: Define constraints
constraints = [cp.sum(p) == 1, p >= 0]
```

```
# Step 6: Solve the problem
problem = cp.Problem(objective, constraints)
problem.solve()

# Step 7: Display the results
print("Optimal Weights:", p.value)
print("Expected Return:", mu.T @ p.value)
print("Portfolio Variance:", p.value.T @ V @ p.value)
```