Portfolio Optimization and Modern Portfolio Theory

Full Definitions, Explanations, and Implementations

Chapter 1: Introduction to Portfolio Optimization

1.1 What Is Portfolio Optimization?

Portfolio optimization is the process of selecting the best asset allocation to maximize expected return for a given level of risk, or equivalently, to minimize risk for a given expected return. In quantitative finance, rigorous mathematical and statistical models are used to achieve this objective.

1.2 Markowitz Modern Portfolio Theory (MPT)

Modern Portfolio Theory (MPT), introduced by Harry Markowitz in 1952, is the foundation of quantitative portfolio optimization. The key insight is that the variance (risk) of a portfolio depends not only on the variances of individual assets but also on their correlations.

Mathematical Formulation:

Let there be n assets with weights w_i, expected returns μ_i, and covariances Σ_{ij}. Portfolio expected return:

$$\mathbb{E}[R_p] = \sum_{i=1}^{n} w_i \mu_i$$

Portfolio variance:

$$Var(R_p) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \Sigma_{ij}$$

Optimization Problem:

$$\text{minimize} \quad w^T \Sigma w$$

$$\text{subject to} \quad w^T \mathbf{1} = 1, \quad w^T \mu = \mu_p \text{ (required return)}, \quad w_i \geq 0 \ (no\ short\ selling)$$

1.3 Efficient Frontier

The set of optimal portfolios (max return for given risk, min risk for given return) forms the efficient frontier, a concave curve in risk-return space. Portfolios below the frontier are sub-optimal.

1.4 Sharpe Ratio and Tangency Portfolio

- **Sharpe Ratio:** $\frac{\mathbb{E}[R_p]-R_f}{\sigma_p}$ measures return per unit risk versus risk-free rate R_f
- Portfolio maximizing Sharpe is the "tangency portfolio."

Chapter 2: Implementing Mean-Variance Optimization

2.1 Data Preparation

- Gather time-synchronous price data
- Compute returns (log or arithmetic)
- Calculate expected returns and covariance matrix

```
import numpy as np
import pandas as pd

# prices: DataFrame of asset prices
returns = prices.pct_change().dropna()
mu = returns.mean()
Sigma = returns.cov()
```

2.2 Solving the Optimization

Numerical Solution (Scipy):

```
from scipy.optimize import minimize

n = len(mu)

# Objective: Minimize portfolio variance
def portfolio_variance(w, Sigma):
    return w.T @ Sigma @ w

# Constraints: Full investment, expected return
def constrain_sum(w): return np.sum(w) - 1
def constrain_return(w): return w.T @ mu - target_return

w0 = np.ones(n) / n
constraints = ({'type': 'eq', 'fun': constrain_sum},
               {'type': 'eq', 'fun': constrain_return})
bounds = [(0,1)] * n
result = minimize(portfolio_variance, w0, args=(Sigma,), constraints=constraints, bounds=
optimal_weights = result.x
```

2.3 Plotting the Efficient Frontier

```
# Vary target_return to trace the frontier
results = []
targets = np.linspace(mu.min(), mu.max(), 50)
for tr in targets:
    constraints[1]['fun'] = lambda w: w.T @ mu - tr
    res = minimize(portfolio_variance, w0, args=(Sigma,), constraints=constraints, bounds
    results.append((tr, np.sqrt(res.fun)))
results = np.array(results)

import matplotlib.pyplot as plt
plt.plot(results[:,1], results[:,0])
plt.xlabel('Risk (Std Dev)'), plt.ylabel('Return')
plt.title('Efficient Frontier')
plt.show()
```

2.4 Maximum Sharpe Portfolio

Set risk-free rate R_f. Maximize Sharpe:

```
Rf = 0.01
def neg_sharpe(w): return -((w @ mu - Rf) / np.sqrt(w @ Sigma @ w))
cons = ({'type': 'eq', 'fun': constrain_sum})
res = minimize(neg_sharpe, w0, constraints=cons, bounds=bounds)
sharpe_weights = res.x
```

Chapter 3: Extensions and Advanced Concepts

3.1 Risk-Free Asset and Capital Market Line

Adding a risk-free asset allows construction of portfolios anywhere on the capital market line (CML), tangent to the efficient frontier.

3.2 Factor Models

Use linear factor models (e.g., Fama-French, Carhart) to explain/estimate asset returns.

3.3 Robust and Regularized Portfolio Optimization

- L2 regularization to penalize large weights
- Uncertainty bounds on mean and covariances
- Shrinkage estimators (Ledoit-Wolf)

3.4 Constraints and Real-World Considerations

- Weight bounds (minimum, maximum allocation per asset)
- Transaction costs, liquidity
- Tax implications and round lots

Chapter 4: Implementation Using PyPortfolioOpt

```
from pypfopt import EfficientFrontier, expected_returns, risk_models

# Compute inputs
er = expected_returns.mean_historical_return(prices)
cov = risk_models.sample_cov(prices)
ef = EfficientFrontier(er, cov)
# Max Sharpe
raw_weights = ef.max_sharpe()
cleaned_weights = ef.clean_weights()
print(cleaned_weights)
# Portfolio performance
perf = ef.portfolio_performance(verbose=True)
```

Chapter 5: Capital Asset Pricing Model (CAPM)

5.1 Definition and Explanation

The **CAPM** gives the expected return for a security or portfolio as:

$$E[R_i] = R_f + \beta_i(E[R_m] - R_f)$$

Where:

- $E[R_i]$: expected return
- R_f: risk-free rate
- $E[R_m]$: expected market return
- β_i: sensitivity to market, calculated as:

$$\beta_i = \frac{Cov(R_i, R_m)}{Var(R_m)}$$

5.2 Implementation in Python

```
# Assume asset_returns, market_returns, risk_free_rate available
excess_asset = asset_returns - risk_free_rate
excess_market = market_returns - risk_free_rate
beta = np.cov(excess_asset, excess_market)[0,1] / np.var(excess_market)
capm_return = risk_free_rate + beta * (excess_market.mean())
```

Chapter 6: Practice Projects

- Build a 10-asset portfolio optimizer and plot the frontier
- Sharpe maximization with constraints (min, max per asset)
- Replicate S&P500 by optimizing tracking error
- Backtest CAPM on historical market data and visualize SML

Further Reading

- Elton, Gruber, Brown, Goetzmann: Modern Portfolio Theory and Investment Analysis
- Luenberger: Investment Science
- PyPortfolioOpt official docs
- MIT OCW - Finance Theory (CAPM video lectures)

This textbook gives you the rigorous theory, mathematical formulations, and stepwise Python/CVXPY code for all aspects of portfolio optimization and modern portfolio theory, fully aligned to industry practice and research.