portfolio_optimisation

December 5, 2021

[1]:
```
# Generate data for long only portfolio optimization.
import cvxpy as cp
import numpy as np
np.random.seed(1)
n = 10
mu = np.abs(np.random.randn(n, 1))
Sigma = np.random.randn(n, n)
Sigma = Sigma.T.dot(Sigma)  # Covariance matrix is PSD
```

[2]:
```
# Long only portfolio optimization.
x = cp.Variable(n)
gamma = cp.Parameter(nonneg=True)
ret = mu.T @ x
risk = cp.quad_form(x, Sigma)
prob = cp.Problem(cp.Maximize(ret - gamma * risk), [cp.sum(x) == 1, x >= 0])
```

[3]:
```
# Compute trade-off curve.
SAMPLES = 100
risk_data = np.zeros(SAMPLES)
ret_data = np.zeros(SAMPLES)
gamma_vals = np.logspace(
    -2, 3, num=SAMPLES
)  # take 100 samples of gamma from 10^-2 to 10^3
for i in range(SAMPLES):
    gamma.value = gamma_vals[i]
    prob.solve()
    risk_data[i] = cp.sqrt(risk).value
    ret_data[i] = ret.value
```

[4]:
```
# Plot long only trade-off curve.
import matplotlib.pyplot as plt
%matplotlib inline
markers_on = [29, 40]
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(risk_data, ret_data, "g-")
for marker in markers_on:
    plt.plot(risk_data[marker], ret_data[marker], "bs")
```

```
    ax.annotate(
        r"$\gamma = %.2f$" % gamma_vals[marker],
        xy=(risk_data[marker] + 0.08, ret_data[marker] - 0.03),
    )
for i in range(n):
    plt.plot(
        cp.sqrt(Sigma[i, i]).value, mu[i], "ro"
    )  # standard deviation of individual asset price is plotted as well
plt.xlabel("Standard deviation")
plt.ylabel("Return")
plt.show()
```