CSCI 4140 Assignment Three

Portfolio Optimization

Due: December 4, 2020

Introduction

Portfolio optimization is a very important problem in finance. One of the main reasons why financial institutions are interested in quantum computers. The statement of this problem is quite simple, but the solution in general can be quite complicated. You have a fixed amount of money and you want to invest it in a number of assets (stocks, bonds, mutual funds, etc.) to get the maximum return with the minimum risk.

Mathematically we state our simple version of the problem in the following way:

$$\min_{x} (q x^T \Sigma x - \mu^T x)$$

Subject to:

$$\sum_i x_i = B$$

Where Σ is the covariance matrix for the assets, μ is the expected return on the assets, q is the risk appetite of the investor (use the value q=0.5) and B is the total budget.

To make things simple we will assume that all assets have the same price and that we can only buy one unit of them. In this case the x_i become binary variables. We will have 4 assets and our total budget is 2. Clearly this is an optimization problem that you will need to solve.

Data

Where are we going to get the data from? It turns out that Qiskit has a procedure for doing this and it will compute the covariance matrix and the expected return for you.

Here are some imports to get you started:

```
from qiskit import Aer
from qiskit.aqua import QuantumInstance
from qiskit.finance.data_providers import RandomDataProvider
from qiskit.aqua.algorithms import VQE, QAOA, NumPyMinimumEigensolver
import numpy as np
import matplotlib.pyplot as plt
import datetime
from qiskit.optimization import QuadraticProgram
from qiskit.optimization.converters import QuadraticProgramToQubo
```

We use the random data provider to generate some random stock information to be used in this assignment:

```
num_assets = 4

# Generate expected return and covariance matrix from (random) time-series
stocks = [("TICKER%s" % i) for i in range(num_assets)]
data = RandomDataProvider(tickers=stocks,
                 start=datetime.datetime(2016,1,1),
                 end=datetime.datetime(2016,1,30))
data.run()
mu = data.get_period_return_mean_vector()
sigma = data.get_period_return_covariance_matrix()
```

This provides the covariance matrix and average returns that you need to get started.

Procedure

Start by constructing a QuadraticProblem for the optimization and data. Then you can basically follow the steps that we have used in the lecture. First, use a classical optimization solver to determine the correct answer. This will be your starting point.

From there you can choose any of the quantum optimization techniques that we described in class. I found that Grover worked the best, but it is more work. You will need to experiment a bit with the parameters. Remember, for many of these problems the first attempt doesn't always work out. You may need to run it several times and experiment with the parameters.

Assignment Report

For the assignment report show how you converted the optimization problem into a quantum program and the results you obtained. Submit this as a PDF file using Canvas.

Note

When setting up the quadratic problem you can specify that the variables are binary and give them a name using

    model.binary_var(name='variable name')

You would do this for each of the variables in the model.