Qubits
qubits.top

量子计算
——量子金融

Quantum Finance

网址：www.qubits.top

作者：Calvin Tang

邮箱：179209347@qq.com

介绍

本教程基于 IBM 的 **Qiskit，Qiskit[finance]** 编写。

本教程包含：

1. 投资组合优化 - 均值方差模型
2. 量子算法
   - 变分量子本征求解器(VQE)
   - 量子近似优化算法（QAOA）
3. 代码实例

Qiskit：
https://qiskit.org/documentation/getting_started.html
Qiskit finance：
https://qiskit.org/documentation/finance/tutorials/index.html

Github & Gitee 代码地址：
https://github.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/01_portfolio_optimization.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/01_portfolio_optimization.py

Calvin，QQ: 179209347 Mail: 179209347@qq.com

Qiskit，Qiskit[finance] 配置和安装

虚拟环境

```
# 创建虚拟环境
conda create -n ENV_NAME python=3.8.0
# 切换虚拟环境
conda activate ENV_NAME
# 退出虚拟环境
conda deactivate ENV_NAME
# 查看现有虚拟环境
conda env list
# 删除现有虚拟环境
conda remove -n ENV_NAME --all
```

安装 Qiskit

```
pip install qiskit
```

```
# install extra visualization support
# For zsh user (newer versions of macOS)
# pip install 'qiskit[visualization]'

pip install qiskit[visualization]
```

安装 Qiskit[finance]

```
# For zsh user (newer versions of macOS)
# pip install 'qiskit[finance]'

pip install qiskit[finance]
```

均方差投资组合优化

本教程给出了给定n个资产，如何解决如下均方差投资组合优化问题的方法：

$$\min_{x\in\{0,1\}^n} qx^T\Sigma x - \mu^T x$$

$$\text{subject to: } 1^T x = B$$

- $x \in \{0,1\}^n$ 表示 2 值决策变量组成的向量， x [i] = 1表示选取该资产， x [i] = 0表示不选取
- $\mu \in \mathbb{R}^n$ 定义期望的资产投资回报率
- $\sum \in \mathbb{R}^{nxn}$ 指定资产间的协方差
- q > 0 控制决策者的风险偏好
- B表示预算，例如，从n个资产中选择的资产数量

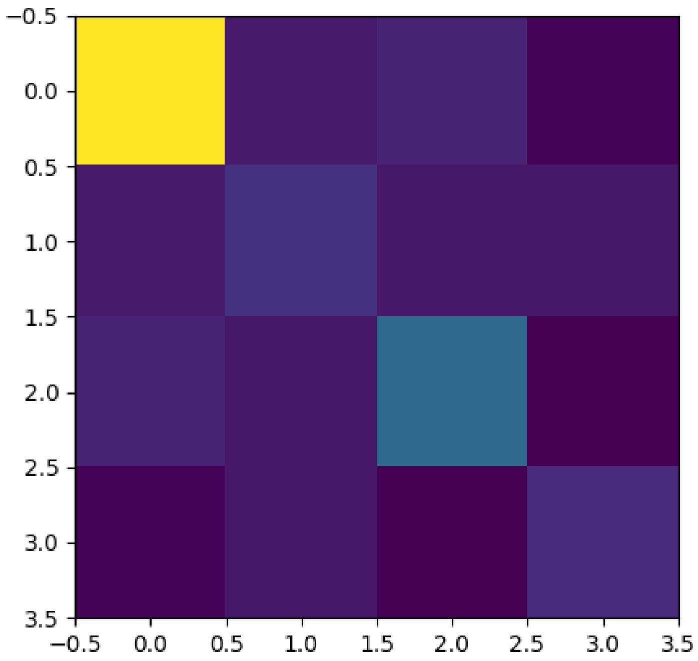
我们做如下简化假设：

- 所有的资产有同样的价格（归一化为1）
- 全部预算B都投入

约束条件$1^T x = B$ 是惩罚项$(1^T x - B)^2$的映射。
问题结果可以映射为哈密顿量，它的基态代表了最优解。
本教程给出了变分量子本征求解器(VQE)，及量子近似优化算法（QAOA）的求解示例，找到给定参数下的最优解。

数据初始化

```python
# 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),
    seed=seed,
)
data.run()
mu = data.get_period_return_mean_vector()
sigma = data.get_period_return_covariance_matrix()
```



1. 经典算法（作为参照）

```
exact_mes = NumPyMinimumEigensolver()
exact_eigensolver = MinimumEigenOptimizer(exact_mes)
result = exact_eigensolver.solve(qp)
print_result(result)
```

```
---------------- Full result ---------------------
selection     value                 probability
---------------------------------------------------
[1 0 0 1]     -0.0149               1.0000
[1 1 1 1]     4.0656                0.0000
[0 1 1 1]     1.0199                0.0000
[1 0 1 1]     1.0049                0.0000
[0 0 1 1]     -0.0010               0.0000
[1 1 0 1]     1.0060                0.0000
[0 1 0 1]     0.0002                0.0000
[0 0 0 1]     1.0191                0.0000
[1 1 1 0]     1.0069                0.0000
[0 1 1 0]     0.0008                0.0000
[1 0 1 0]     -0.0140               0.0000
[0 0 1 0]     1.0197                0.0000
[1 1 0 0]     -0.0130               0.0000
[0 1 0 0]     1.0208                0.0000
[1 0 0 0]     1.0059                0.0000
[0 0 0 0]     4.0795                0.0000
```

2. 变分量子本征求解器(VQE)

```python
from qiskit.utils import algorithm_globals

algorithm_globals.random_seed = 1234
backend = Aer.get_backend("statevector_simulator")

cobyla = COBYLA()
cobyla.set_options(maxiter=500)
ry = TwoLocal(num_assets, "ry", "cz", reps=3,
entanglement="full")
quantum_instance =
QuantumInstance(backend=backend,
seed_simulator=seed, seed_transpiler=seed)
vqe_mes = VQE(ry, optimizer=cobyla,
quantum_instance=quantum_instance)
vqe = MinimumEigenOptimizer(vqe_mes)
result = vqe.solve(qp)

print_result(result)
```

```
---------------- Full result --------------------
selection      value                probability
-------------------------------------------------
[1 0 0 1]      -0.0149              0.8489
[0 0 1 1]      -0.0010              0.0885
[1 1 0 0]      -0.0130              0.0589
[1 0 1 0]      -0.0140              0.0018
[0 1 1 0]       0.0008              0.0016
[0 1 0 1]       0.0002              0.0003
[1 1 1 0]       1.0069              0.0000
[0 0 0 1]       1.0191              0.0000
[0 0 1 0]       1.0197              0.0000
[0 1 1 1]       1.0199              0.0000
[1 0 0 0]       1.0059              0.0000
[1 1 0 1]       1.0060              0.0000
[1 0 1 1]       1.0049              0.0000
[1 1 1 1]       4.0656              0.0000
[0 1 0 0]       1.0208              0.0000
[0 0 0 0]       4.0795              0.0000
```

3. 量子近似优化算法（QAOA）

```
algorithm_globals.random_seed = 1234
backend = Aer.get_backend("statevector_simulator")

cobyla = COBYLA()
cobyla.set_options(maxiter=250)
quantum_instance =
QuantumInstance(backend=backend,
seed_simulator=seed, seed_transpiler=seed)
qaoa_mes = QAOA(optimizer=cobyla, reps=3,
quantum_instance=quantum_instance)
qaoa = MinimumEigenOptimizer(qaoa_mes)
result = qaoa.solve(qp)

print_result(result)
```

```
----------------- Full result ---------------------
selection      value                    probability
---------------------------------------------------
 [1 0 0 1]     -0.0149                  0.1683
 [1 0 1 0]     -0.0140                  0.1682
 [1 1 0 0]     -0.0130                  0.1679
 [0 0 1 1]     -0.0010                  0.1654
 [0 1 0 1]     0.0002                   0.1652
 [0 1 1 0]     0.0008                   0.1650
 [1 1 1 1]     4.0656                   0.0000
 [0 0 0 0]     4.0795                   0.0000
 [1 1 1 0]     1.0069                   0.0000
 [1 0 1 1]     1.0049                   0.0000
 [1 1 0 1]     1.0060                   0.0000
 [0 1 0 0]     1.0208                   0.0000
 [0 0 1 0]     1.0197                   0.0000
 [0 0 0 1]     1.0191                   0.0000
 [0 1 1 1]     1.0199                   0.0000
 [1 0 0 0]     1.0059                   0.0000
```

参考论文

[1] Improving Variational Quantum Optimization using CVaR. Barkoutsos et al. 2019.
https://arxiv.org/abs/1907.04769

Qubits
qubits.top

Thank You