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。
https://qiskit.org/documentation/finance/tutorials/06_basket_option_pricing.html

本教程包含：
1. 一篮子期权 (Basket Options)
2. 量子算法 - 通过QAE求解问题
3. 代码实例
 * TODO：完善算法的详细解读

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/06_basket_option_pricing.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/06_basket_option_pricing.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]
```

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

一篮子期权 (Basket Options)

- **奇异期权**是相对于普通（标准）期权来说的，在场外衍生品市场上还有很多非标准化产品，也就是奇异产品（exotic product）。奇异产品的产生当然是来自于需求，比如满足某种对冲需要、规避法律或监管等，当然也不排除为了使产品更加诱人，吸引经验少的投资者等。

- **一篮子期权（basket potion）**是奇异期权的一种，也称之为一揽子期权、篮筐式期权，是以一篮子资产组合的期权，篮子里的资产不是一只而是多只，资产可以包括股票、股指和货币等资产。由于期权的本质是风险管理工具，所以这种期权的设计也是为了对冲投资组合风险。

- 常见的一篮子期权基本结构有两种：
  - 一篮子最好认购期权（Best of Basket Call Option）
    到期时只按照篮子中能带来最大收益的那只认购期权进行结算；
  - 一篮子最坏认沽期权（Worstof Basket Put Option）
    到期时按照跌幅最大的，即带来最大收益的那只认沽期权来进行结算；

一篮子期权 (Basket Options) – 案例分析

比如某机构投资者持有现金3000万元，现在对市场中5只股票都看好，即茅台、隆基股份、片仔癀、招商银行和宝钢股份。一种方法是按一定比例买入并持有5只股票，这样到期时获得的收益就是这5只股票按照买入比例加权后的收益；另一种方法是要获得涨幅最大的那只股票带来的收益，于是买入一篮子最好认购期权（Best of Basket Call Option），到期时只**按照篮子中能带来最大收益的那只认购期权进行结算。**
其中：合约金额3000万的一篮子最好认购期权的期权费为8%。合约到期时，上面五只股票有涨有跌，涨幅分别为10%，15%，-2%，4%和8%。

两种方法的区别：

- 用现金等比例购买5只股票的收益，不考虑手续费：

  收益：3000 *（10%+15%+（-2%）+4%+8%）/ 5 = 210万

- 一篮子最好认购期权的收益，按照最高15%进行结算：
  期初期权费 = 3000*8% = 240万
  期权收益 = 3000*15% = 450万
  最终收益 = 450-240 = 210万

一篮子期权 (Basket Options) – 案例分析

这个例子中，两个方法的收益虽然一样，但是从组合风险角度看区别是很大：

- 在第一种方法中要用真金白银5000万去买股票，要承担市场系统性风险和个股风险。
- 在第二种方法中，仅需要付出期权费240万，除非5只股票全部下跌会损失掉全部的期权费，但凡有一只股票上涨，期权都会有收益，最终收益取决于期权收益和期权费的高低。一篮子期权最后的收益率=210/240=87.5%

同理，如果同时看跌几只股票，则可以买入一篮子最坏认沽期权（Worstof Basket Put Option），到期时按照跌幅最大的，即带来最大收益的那只认沽期权来进行结算。

一篮子期权 (Basket Options) – （简化模型）

假定一篮子期权，损益函数定义为： $\max\{S_T^1 + S_T^2 - K, 0\}$

- T：期权到期日
- S_T^1，S_T^2：现货市价 (spot price)
- K：成交价 (strike price)

在后面的例子里，量子计算算法基于振幅估计实现，估算到期损益：

$$\mathbb{E}\left[\max\{S_T^1 + S_T^2 - K, 0\}\right].$$

Uncertainty Model

在后面的例子里，量子计算算法基于振幅估计实现，估算到期损益：

我们构造一个量子线路，加载多变量对数正态分布(Log-Normal Distribution)数据，初始化 n 量子位量子态。对每一个维度 j = 1,...d，数据分布区间[low_j,high_j]，使用 2^{n_j} 个网格点离散化，n_j 表示用于维度 j 使用的量子位数，例如： $n_j + \ldots + n_d = n$ 。幺正变换算子如下：

$$|0\rangle_n \mapsto |\psi\rangle_n = \sum_{i_1,\ldots,i_d} \sqrt{p_{i_1\ldots i_d}} |i_1\rangle_{n_1} \ldots |i_d\rangle_{n_d},$$

$P_{i1\ldots id}$表示离散分布概率，i为对应正确区间的仿射映射：

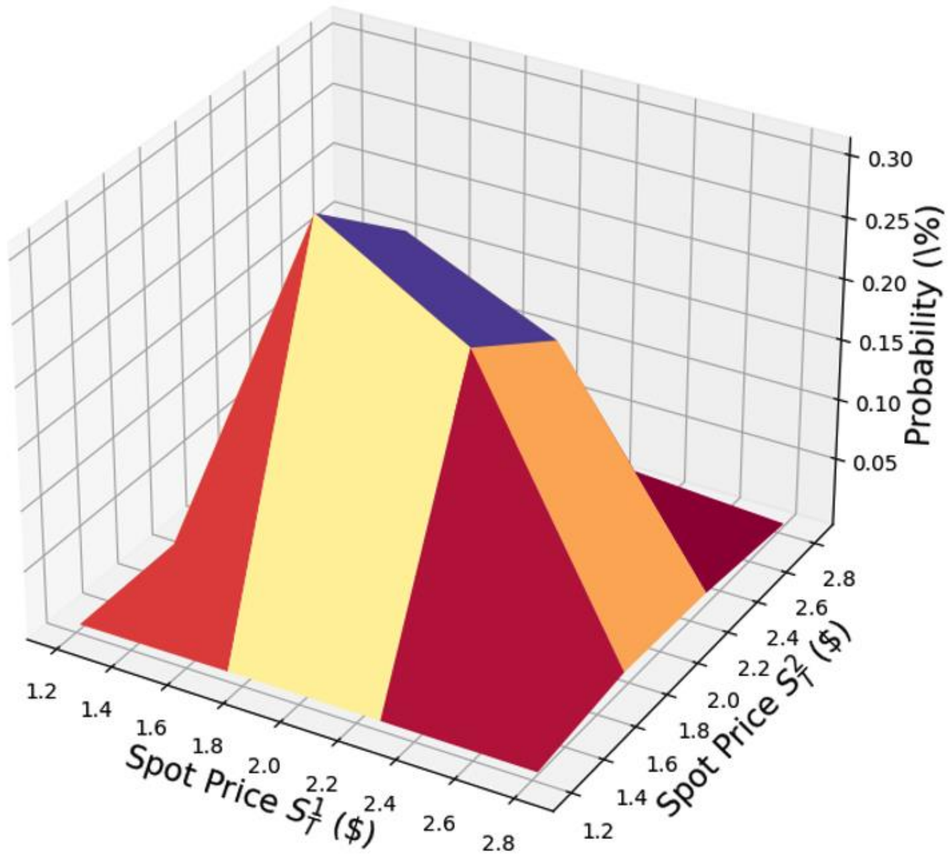
$$\{0,\ldots,2^{n_j}-1\} \ni i_j \mapsto \frac{\text{high}_j - \text{low}_j}{2^{n_j}-1} * i_j + \text{low}_j \in [\text{low}_j, \text{high}_j].$$

为了简化，我们假定两个股票价格是独立同分布的。当前实现最重要的假设是不同维度离散化网格具有相同的步长。

Uncertainty Model

```python
# resulting parameters for log-normal distribution
mu = (r - 0.5 * vol**2) * T + np.log(S)
sigma = vol * np.sqrt(T)
mean = np.exp(mu + sigma**2 / 2)
variance = (np.exp(sigma**2) - 1) * np.exp(2 * mu + sigma**2)
stddev = np.sqrt(variance)
# lowest and highest value considered for the spot price; in
between, an equidistant discretization is considered.
low = np.maximum(0, mean - 3 * stddev)
high = mean + 3 * stddev
# map to higher dimensional distribution
# for simplicity assuming dimensions are independent and
identically distributed)
dimension = 2
num_qubits = [num_uncertainty_qubits] * dimension
low = low * np.ones(dimension)
high = high * np.ones(dimension)
mu = mu * np.ones(dimension)
cov = sigma**2 * np.eye(dimension)
```

```python
# construct circuit
u = LogNormalDistribution(num_qubits=num_qubits,
mu=mu, sigma=cov, bounds=list(zip(low, high)))
```



损益函数 （Payoff Function）

$S_T^1 + S_T^2 <$ K 时损益函数等于0，然后损益函数线性增加。
使用一个带权重的sum算子计算现货市价 (spot prices)求和，保存于一个辅助寄存器，然后使用一个比较器，当 $S_T^1 + S_T^2 >=$K 时，会交换辅助量子比特: |0> → |1>, 辅助量子比特用于控制损益函数的线性部分。
当 |y| 足够小时：

$$\sin^2(y + \pi/4) \approx y + 1/2$$

因此，对于一个给定的近似缩放因子： $c_{\text{approx}} \in [0, 1]$ and $x \in [0, 1]$

有： $\sin^2(\pi/2 * c_{\text{approx}} * (x - 1/2) + \pi/4) \approx \pi/2 * c_{\text{approx}} * (x - 1/2) + 1/2$

我们使用受控绕Y轴旋转，很容易构造一个算子：

$$|x\rangle|0\rangle \mapsto |x\rangle \left(\cos(a * x + b)|0\rangle + \sin(a * x + b)|1\rangle\right)$$

最后，我们需要做的是测量|1>的概率振幅： $\sin^2(a * x + b)$

c_{approx} 越小越准，但是量子位m也需要相应调整。

量子线路

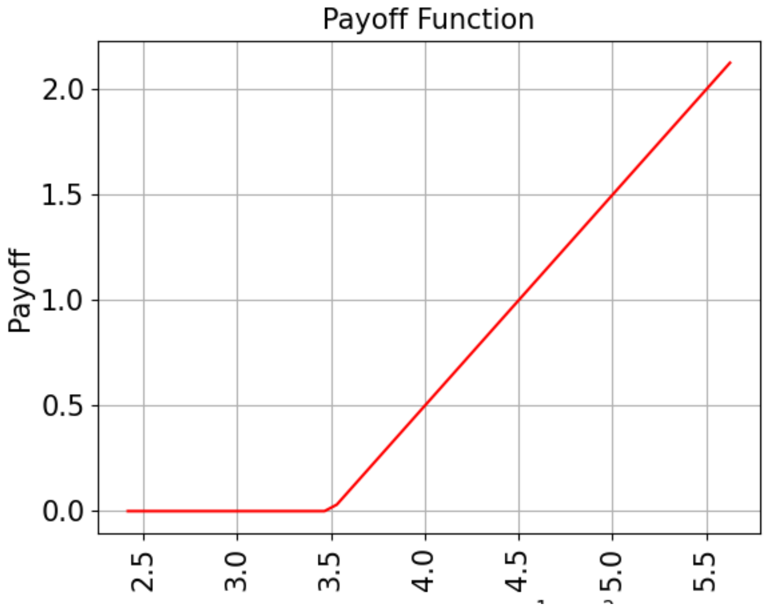
```python
# define overall multivariate problem
qr_state = QuantumRegister(u.num_qubits, "state")  # to load the probability distribution
qr_obj = QuantumRegister(1, "obj")  # to encode the function values
ar_sum = AncillaRegister(n_s, "sum")  # number of qubits used to encode the sum
ar = AncillaRegister(max(n_aux, basket_objective.num_ancillas), "work")  # additional qubits

objective_index = u.num_qubits

basket_option = QuantumCircuit(qr_state, qr_obj, ar_sum, ar)
basket_option.append(u, qr_state)
basket_option.append(agg, qr_state[:] + ar_sum[:] + ar[:n_aux])
basket_option.append(basket_objective, ar_sum[:] + qr_obj[:] + ar[:basket_objective.num_ancillas])

print(basket_option.draw())
print("objective qubit index", objective_index)
```

```
         ┌──────────┐ ┌──────────┐
state_0: ┤ 0        ├┤ 0         ├──────────
         │          ││           │
state_1: ┤ 1        ├┤ 1         ├──────────
         │   P(X)   ││           │
state_2: ┤ 2        ├┤ 2         ├──────────
         │          ││           │
state_3: ┤ 3        ├┤ 3         ├──────────
         └──────────┘│           │┌──────┐
  obj_0: ────────────┤           ├┤ 3    ├
                     │           ││      │
  sum_0: ────────────┤ 4  adder  ├┤ 0    ├
                     │           ││      │
  sum_1: ────────────┤ 5         ├┤ 1    ├
                     │           ││      │
  sum_2: ────────────┤ 6         ├┤ 2  F ├
                     │           ││      │
 work_0: ────────────┤ 7         ├┤ 4    ├
                     │           ││      │
 work_1: ────────────┤ 8         ├┤ 5    ├
                     │           ││      │
 work_2: ────────────┤ 9         ├┤ 6    ├
                     └───────────┘└──────┘
objective qubit index 4
```

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

损益函数

```
basket_objective = LinearAmplitudeFunction(
    n_s,
    slopes,
    offsets,
    domain=(0, max_value),
    image=(f_min, f_max),
    rescaling_factor=c_approx,
    breakpoints=breakpoints,
)
# define overall multivariate problem
qr_state = QuantumRegister(u.num_qubits, "state")  # to load the probability distribution
qr_obj = QuantumRegister(1, "obj")  # to encode the function values
ar_sum = AncillaRegister(n_s, "sum")  # number of qubits used to encode the sum
ar = AncillaRegister(max(n_aux, basket_objective.num_ancillas), "work")  # additional qubits
objective_index = u.num_qubits
…
```



结果：

```
objective qubit index 4
exact expected value:   0.4870
```

振幅估计

```python
problem = EstimationProblem(
    state_preparation=basket_option,
    objective_qubits=[objective_index],
    post_processing=basket_objective.post_processing,
)
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)
result = ae.estimate(problem)
conf_int = (
    np.array(result.confidence_interval_processed)
    / (2**num_uncertainty_qubits - 1)
    * (high_ - low_)
)
print("Exact value:        \t%.4f" % exact_value)
print("Estimated value:    \t%.4f"
    % (result.estimation_processed / (2**num_uncertainty_qubits - 1) * (high_ - low_))
)
print("Confidence interval:\t[%.4f, %.4f]" % tuple(conf_int))
```

结果:

```
Exact value:
0.4870
Estimated value:
0.5397
Confidence interval:
[0.5124, 0.5670]
```

参考

[1] Quantum Risk Analysis. Woerner, Egger. 2018.
https://www.nature.com/articles/s41534-019-0130-6

[2] Option Pricing using Quantum Computers. Stamatopoulos et al. 2019.
https://quantum-journal.org/papers/q-2020-07-06-291/

Thank

You