量子计算
—量子金融

Quantum Finance

网址：www.qubits.top

作者：Calvin Tang

邮箱：179209347@qq.com

Qubits
qubits.top

介绍

本教程基于 IBM 的 **Qiskit，Qiskit[finance]** 编写。
https://qiskit.org/documentation/finance/tutorials/10_qgan_option_pricing.html

本教程包含：

1. QGAN期权定价
2. 量子算法 - 通过qGAN, 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/10_qgan_option_pricing.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/10_qgan_option_pricing.py

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]
```

期权的损益分析

- 执行价值：期权买方执行期权权利时能获得的利润。(以欧式期权为例)
- T：期权到期日。
- S：现货市价
- K：成交价
- c：欧式买权的期权费
- p：欧式卖权的期权费

期权种类	到期损益
欧式看涨期权多头	$Max\{S_T - K - c, -c\}$
欧式看涨期权空头	$Min\{K - S_T + c, +c\}$
欧式看跌期权多头	$Max\{K - S_T - p, -p\}$
欧式看跌期权空头	$Min\{S_T - K + p, +p\}$

qGAN期权定价介绍

本教程演示如何使用量子机器学习算法 - 量子对抗生成网络(qGAN)，辅助欧式看涨期权定价。
更具体的说，qGAN可以被训练模拟欧式看涨期权资产现货价格。
结果模型可以集成于量子振幅估计，估算期望损益。

参考：
European Call Option Pricing.
http://localhost:8888/notebooks/03_european_call_option_pricing.ipynb

Quantum Generative Adversarial Networks for Learning and Loading Random Distributions.
Zoufal, Lucchi, Woerner. 2019.
https://www.nature.com/articles/s41534-019-0223-2

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，加载对数正态分布(Log-Normal Distribution)数据，初始化量子态。
我们使用qGAN训练对数正态分布数据采样。幺正变换算子如下：

$$|g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle,$$

$$p_\theta^j, \text{ for } j \in \{0, \ldots, 2^n - 1\}$$

Uncertainty Model

```
# Set upper and lower data values
bounds = np.array([0.0, 7.0])
# Set number of qubits used in the uncertainty model
num_qubits = 3

# Load the trained circuit parameters
g_params = [0.29399714, 0.38853322, 0.9557694, 0.07245791,
6.02626428, 0.13537225]

# Set an initial state for the generator circuit
init_dist = NormalDistribution(num_qubits, mu=1.0, sigma=1.0,
bounds=bounds)

# construct the variational form
var_form = TwoLocal(num_qubits, "ry", "cz",
entanglement="circular", reps=1)

...
```

probability distribution

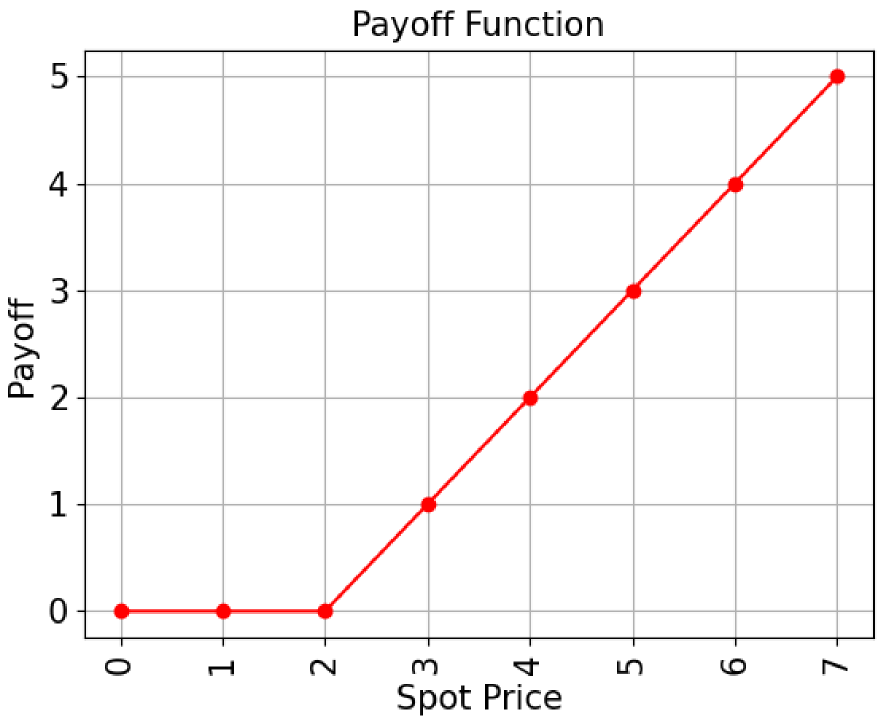
Calvin, QQ: 179209347 Mail: 179209347@qq.com

损益函数

```python
# Evaluate Expected Payoff
# Evaluate payoff for different distributions
payoff = np.array([0, 0, 0, 1, 2, 3, 4, 5])
ep = np.dot(log_normal_samples, payoff)
print("Analytically calculated expected payoff w.r.t. the target distribution:  %.4f" % ep)
ep_trained = np.dot(y, payoff)
print("Analytically calculated expected payoff w.r.t. the trained distribution: %.4f" % ep_trained)

# Plot exact payoff function (evaluated on the grid of the trained uncertainty model)
x = np.array(values)
y_strike = np.maximum(0, x - strike_price)
plt.plot(x, y_strike, "ro-")
plt.grid()
```

```python
plt.title("Payoff Function", size=15)
plt.xlabel("Spot Price", size=15)
plt.ylabel("Payoff", size=15)
plt.xticks(x, size=15, rotation=90)
plt.yticks(size=15)
plt.show()
```



振幅估计

```
# construct circuit for payoff function
european_call_pricing = EuropeanCallPricing(
    num_qubits,
    strike_price=strike_price,
    rescaling_factor=c_approx,
    bounds=bounds,
    uncertainty_model=uncertainty_model,
)
# set target precision and confidence level
epsilon = 0.01
alpha = 0.05
qi = QuantumInstance(Aer.get_backend("aer_simulator"), shots=100)
problem = european_call_pricing.to_estimation_problem()
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha,
quantum_instance=qi)
```

```
result = ae.estimate(problem)
conf_int = np.array(result.confidence_interval_processed)
print("Exact value:        \t%.4f" % ep_trained)
print("Estimated value:    \t%.4f" % (result.estimation_processed))
print("Confidence interval:\t[%.4f, %.4f]" % tuple(conf_int))
```

结果:

```
Exact value:            0.9805
Estimated value:        1.0196
Confidence interval:    [0.9885, 1.0508]
```

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

Qubits
qubits.top

Thank You