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/09_credit_risk_analysis.html

本教程包含：
1. 信用风险分析
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/09_credit_risk_analysis.py
https://gitee.com/mymagicpower/qubits/tree/main/quantum_qiskit_finance/09_credit_risk_analysis.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

教程目录

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

信用风险分析

本教程讲解了量子计算算法如何用于信用风险分析。更准确的说，如果用量子振幅估计(QAE)算法去估计风险度量，对经典蒙特卡罗模拟的二次加速。

具体可以参考教程：

[1] Quantum Risk Analysis. Stefan Woerner, Daniel J. Egger. [Woerner2019]
https://www.nature.com/articles/s41534-019-0130-6

[2] Credit Risk Analysis using Quantum Computers. Egger et al. (2019) [Egger2019]
https://arxiv.org/abs/1907.03044

QAE介绍：

[3] Quantum Amplitude Amplification and Estimation. Gilles Brassard et al.
https://arxiv.org/abs/quant-ph/0005055

1. 问题定义 - Problem Definition

在本教程里，我们分析 K个资产投资组合的信用风险分析。每个资产的默认概率符合高斯条件独立分布，例如：给定一个值 z，其采样于潜在的随机变量 Z，其符合标准正态分布，资产 k 默认概率为：

$$p_k(z) = F\left(\frac{F^{-1}(p_k^0) - \sqrt{\rho_k}z}{\sqrt{1-\rho_k}}\right)$$

- F：表示累积分布函数 Z
- p_k^0: 资产 k 在 z = 0时的的默认概率
- ρ_k: 是资产 k 关于 Z 的默认概率敏感度

分析风险度量的损失函数定义为：

$$L = \sum_{k=1}^{K} \lambda_k X_k(Z)$$

- λ_k：表示资产 k 的默认损失
- $X_k(Z)$： 表示伯努利变量，代表资产k的默认事件

L 的 VaR 和 CVaR定义为：

$$\mathrm{VaR}_\alpha(L) = \inf\{x \mid \mathbb{P}[L <= x] \geq 1 - \alpha\}$$

$$\mathrm{CVaR}_\alpha(L) = \mathbb{E}[L \mid L \geq \mathrm{VaR}_\alpha(L)].$$

Regulatory Capital Modeling for Credit Risk. Marek Rutkowski, Silvio Tarca.
https://arxiv.org/abs/1412.1183

1. 问题定义 - Problem Definition

问题由如下参数定义:

- number of qubits used to represent Z, denoted by n_z
- truncation value for Z, denoted by z_{max}, i.e., Z is assumed to take 2^{n_z} equidistant values in $\{-z_{max}, \ldots, +z_{max}\}$
- the base default probabilities for each asset $p_0^k \in (0, 1), k = 1, \ldots, K$
- sensitivities of the default probabilities with respect to Z, denoted by $\rho_k \in [0, 1)$
- loss given default for asset k, denoted by λ_k
- confidence level for VaR / CVaR $\alpha \in [0, 1]$.

```
# set problem parameters
n_z = 2
z_max = 2
z_values = np.linspace(-z_max, z_max, 2**n_z)
p_zeros = [0.15, 0.25]
rhos = [0.1, 0.05]
lgd = [1, 2]
K = len(p_zeros)
alpha = 0.05
```

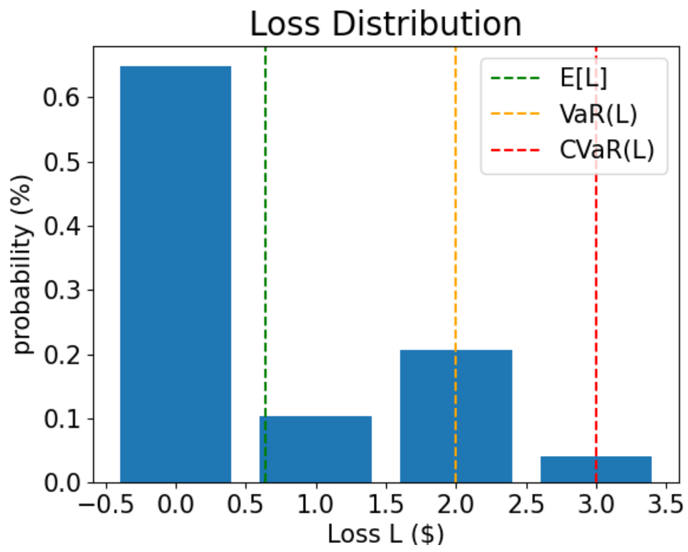
2. Uncertainty Model

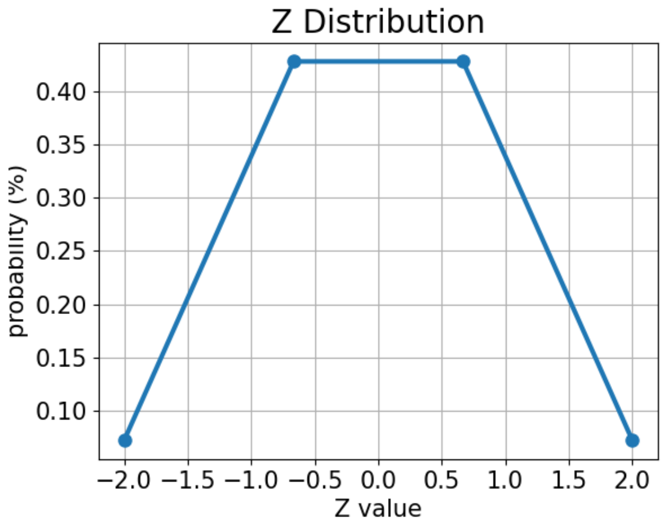
我们构造一个量子线路，通过创建 n_z 量子比特寄存器表示 Z，数据分布符合标准正态分布，初始化量子态，然后这个量子态用于控制第2个K量子寄存器的单量子的Y轴旋转。
量子比特 k 的量子态 |1> 表示资产 k 的默认事件,幺正变换算子如下：

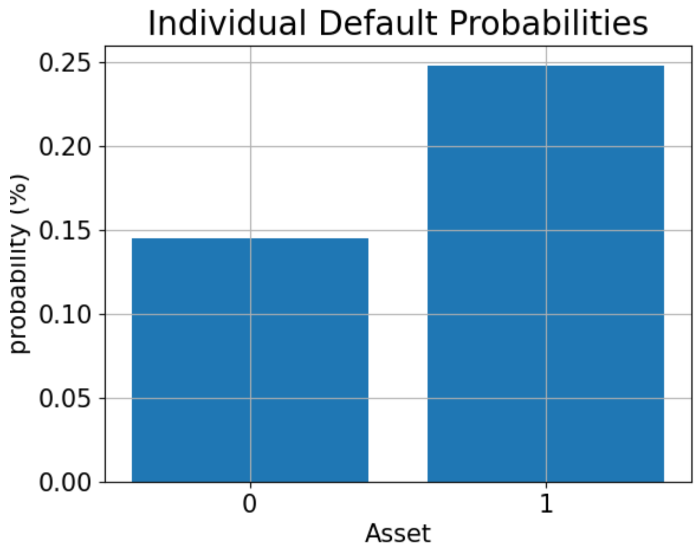
$$|\Psi\rangle = \sum_{i=0}^{2^{n_z}-1} \sqrt{p_z^i}|z_i\rangle \bigotimes_{k=1}^{K} \left(\sqrt{1-p_k(z_i)}|0\rangle + \sqrt{p_k(z_i)}|1\rangle\right)$$

Z_i表示离散且截断Z的第 i个值。

2. Uncertainty Model







Expected Loss E[L]: 0.6409
Value at Risk VaR[L]: 2.0000
P[L <= VaR[L]]: 0.9591
Conditional Value at Risk CVaR[L]: 3.0000

3. 预期损失 - Expected Loss

为估计期望损失，我们首先使用带权重求和算符，对独立损失求和：

$$\mathcal{S} : |x_1, \ldots, x_K\rangle_K |0\rangle_{n_S} \mapsto |x_1, \ldots, x_K\rangle_K |\lambda_1 x_1 + \ldots + \lambda_K x_K\rangle_{n_S}.$$

所需的量子比特数为：

$$n_s = \lfloor \log_2(\lambda_1 + \ldots + \lambda_K) \rfloor + 1.$$

将整体损失 L 映射为目标量子比特振幅：

$$|L\rangle_{n_s}|0\rangle \mapsto |L\rangle_{n_s}\left(\sqrt{1 - L/(2^{n_s} - 1)}|0\rangle + \sqrt{L/(2^{n_s} - 1)}|1\rangle\right),$$

$$L \in \{0, \ldots, 2^{n_s} - 1\}$$

3. 预期损失 - Expected Loss

```
qi = QuantumInstance(Aer.get_backend("aer_simulator"),
shots=100)
problem = EstimationProblem(
    state_preparation=state_preparation,
    objective_qubits=[len(qr_state)],
    post_processing=objective.post_processing,
)
# construct amplitude estimation
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha,
quantum_instance=qi)
result = ae.estimate(problem)

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

```
Exact value:            0.6409
Estimated value:        0.6466
Confidence interval:    [0.6182, 0.6750]
```

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

4. 累积分布函数 - Cumulative Distribution Function (CDF)

替代期望损失（使用经典计算更有效），我们使用CDF估计损失。
典型的，需要评估所有资产的可能组合，或者很多蒙特卡罗模拟经典采样。基于QAE算法有潜在的显著加速效果。
为了估计 CDF，算子如下：

$$\mathcal{C} : |L\rangle_n|0> \mapsto \begin{cases} |L\rangle_n|1> & \text{if} \quad L \leq x \\ |L\rangle_n|0> & \text{if} \quad L > x. \end{cases} \qquad \mathbb{P}[L \leq x]$$

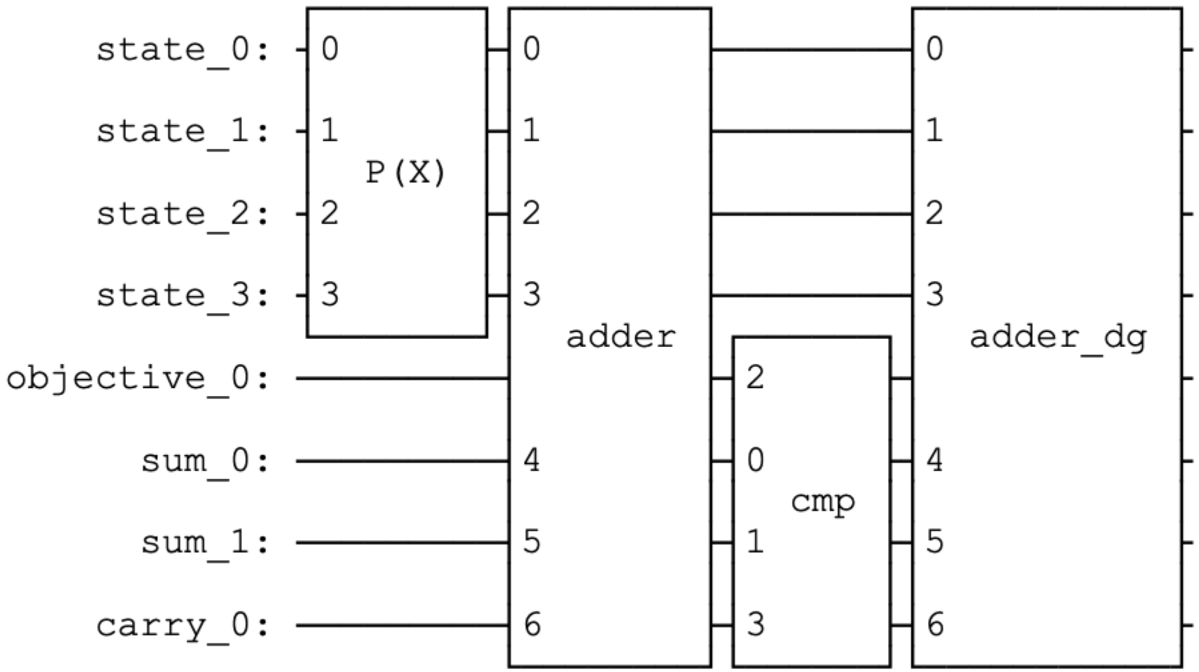
结果量子态写为：

$$\sum_{L=0}^{x} \sqrt{p_L}|L\rangle_{n_S}|1\rangle + \sum_{L=x+1}^{2^{n_S}-1} \sqrt{p_L}|L\rangle_{n_S}|0\rangle,$$

CDF(x) 等于测量|1>的概率。

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

4. 累积分布函数 - Cumulative Distribution Function (CDF)



Operator CDF(2) = 0.9591	
Exact CDF(2) = 0.9591	
Exact value:	0.9591
Estimated value:	0.9592
Confidence interval:	[0.9583, 0.9602]

5. 在险价值 - Value at Risk（VaR)

```python
# run bisection search to determine VaR
objective = lambda x: run_ae_for_cdf(x)
bisection_result = bisection_search(
    objective, 1 - alpha, min(losses) - 1, max(losses), low_value=0, high_value=1
)
var = bisection_result["level"]

print("Estimated Value at Risk: %2d" % var)
print("Exact Value at Risk:     %2d" % exact_var)
print("Estimated Probability:    %.3f" % bisection_result["value"])
print("Exact Probability:        %.3f" % cdf[exact_var])
```

```
--------------------------------------------------------------------
start bisection search for target value 0.950
--------------------------------------------------------------------
low_level     low_value     level     value     high_level     high_value
--------------------------------------------------------------------
-1            0.000           1       0.750      3             1.000
 1            0.750           2       0.960      3             1.000
--------------------------------------------------------------------
finished bisection search
--------------------------------------------------------------------
```

```
Estimated Value at Risk: 2
Exact Value at Risk: 2
Estimated Probability: 0.960
Exact Probability: 0.959
```

6. 条件风险值 - Conditional Value at Risk (CVaR)

最后，我们计算CVaR，我们使用线性分段函数评估，其公式如下：

$$f(L) = \begin{cases} 0 & \text{if} \quad L \leq VaR \\ L & \text{if} \quad L > VaR. \end{cases}$$

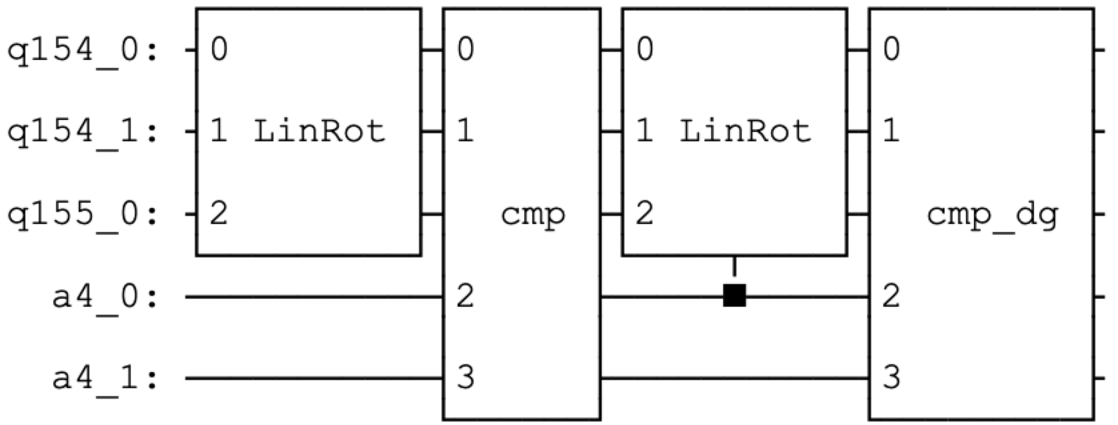
为了归一化，我们将结果期望值通过VaR概率分割，例如：

$$\mathbb{P}[L \leq VaR].$$

6. 条件风险值 - Conditional Value at Risk (CVaR)

```
# define linear objective
breakpoints = [0, var]
slopes = [0, 1]
offsets = [0, 0]  # subtract VaR and add it later to the estimate
f_min = 0
f_max = 3 - var
c_approx = 0.25

cvar_objective = LinearAmplitudeFunction(
    agg.num_sum_qubits,
    slopes,
    offsets,
    domain=(0, 2**agg.num_sum_qubits - 1),
    image=(f_min, f_max),
    rescaling_factor=c_approx,
    breakpoints=breakpoints,
)
```



Exact CVaR:	3.0000
Estimated CVaR:	3.2831

Thank

You