Quantum Reinforcement Learning for Enhanced Portfolio Optimization

Review 2: Final Presentation

Outline

Team & Contribution

Member	Contribution
SaiRahul Kodeboina 21BCE9121	Quantum & Classical ML Coding, Documentation
	Non-Machine Learning Optimization, Documentation & Classical ML Coding
	Quantum & Classical ML Coding, Documentation
Chitteti Kusela 21BCE9158	Classical ML Coding & Documentation
	Classical ML Coding & Documentation
Dunde Sumathi 21BCE9401	Quantum ML Documentation
	Classical ML Coding & Documentation

Introduction: Motivation

Financial Systems are complex, strongly correlated and difficult to predict, full of optimization problems, Monte Carlo sampling, stochastic differential equations, machine learning…

Quantum Mechanics principles native to Quantum Computing, make it suitable for implementing financial systems, also inline with business objectives, as reduced execution time due to Quantum Advantage can also ensure more savings.

Problem Mapping

Problem	High Level Solution
Which assets should be included in an optimized portfolio, and how should one change its composition according to the market?	Classical: Optimization models **Quantum optimization** **Quantum-Inspired Optimization**
How to detect opportunities in the different assets in the market, and take profit by trading them?	Classical: Machine learning **Quantum Reinforcement learning**
How to estimate the risk of a portfolio, a company, or even the whole financial system?	Classical: Monte Carlo **Quantum amplitude estimation** **Quantum Monte Carlo**

Introduction: Portfolio Optimization

Financial portfolio optimization is the problem of optimal allocation of a fixed budget to a collection of assets (commodities, bonds, securities etc.) which produces random returns over time.

Portfolio Optimization

Goals:

- Maximize returns
- Minimize risk
- Stay within budget

Output:

- A portfolio representing a list of investments and the expected return

Inputs:

- Uniform random historical price data
- Budget
- Risk tolerance

Project Overview

Goal : Implement quantum algorithms for optimization financial portfolio. To reduce errors and improve accuracy for computing expected returns from assets.

Background : Combinatorial optimization problem is a class of problems where goal is to find the best solution from a finite set of solutions.

Approach: CVaR (Conditional Value-at-Risk) adaption to **QAOA** (Quantum approximate optimization algorithm) is good for combinatorial optimization though it is a special case of **VQE** (Variational-Quantum-Eigensolver).

Why not VQE : In **VQE** goal is to find the ground state energy and in order to do that ground state need to be need reproduced. While in **QAOA** goal is to find the solution to the problem. To do that there is no need to find the ground state — just need to find a state which has a high enough probability of finding the right solution aligned to requirement of combinatorial optimization problem .

What is the special about this Approach : CVaR (Conditional Value-at-Risk) is an aggregation function which leads to faster convergence to better solutions for all combinatorial optimization problems.

Project Overview : Considerations

During implementation of Quantum portfolio optimizer framework, intent was to also highlight comparative features , to achieve same and keep approach within Quantum Computing Simulator limits, a portfolio of 5 assets from US Stock Exchange was considered (based on lesser correlation) .

Input portfolio dataset was generated using Yahoo finance historical (2015 – 2020) trading data for 5 stocks :

- **IBM** (IT Industry)
- **Pfizer** (Healthcare / Pharmacy)
- **Exxon Mobil Corp.** (Oil & Gas)
- **Bank of America** (Finance / Banking)
- **Tesla** (Automobile / Technology)

Technology Ecosystem for Methodologies Comparison

Non-Machine Learning :

Tool (s): MS Excel

Solver / Optimizer : GRG (Generalized Reduced Gradient) Nonlinear

Platform : Windows 10

Classical Machine Learning:

Programming Language : Python 3.x >= 3.7

ML Library : Keras (2.4) for Neural Network, Pandas, Numpy, Yahoo Finance , Scikit Learn, SciPy

Neural Network Type : Long Short-Term Memory (**LSTM**)

Platform : Jupyter Notebook on Google Colab / Kaggle (with GPU access)

Quantum Reinforcement Learning:

Programming Language : Python 3.x >= 3.7

Quantum Programing SDK : Qiskit (latest >=0.29)

Algorithm : CVaR + QAOA

Platform : Qiskit QASM Simulator

Implementation Methodology

Portfolio Optimization Concepts

Portfolio Expected Return

- The expected return of a portfolio is calculated by multiplying the weight of the asset by its return and summing the values of all the assets together. To introduce a forward looking estimate, probability may be introduced to generate and incorporate features in business and economy.

  - Expected return:

$$\mathrm{E}(R_p) = \sum_i w_i \, \mathrm{E}(R_i)$$

Portfolio Optimization Concepts

❖ **Portfolio Variance**

Portfolio variance is used as the measure of risk in this model. A higher variance will indicate a higher risk for the asset class and the portfolio. The formula is expressed as

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij},$$

Portfolio Optimization Concepts

- **Sharpe Ratio**

  The Sharpe ratio measures the return of an investment in relation to the risk-free rate (Treasury rate) and its risk profile. In general, a higher value for the Sharpe ratio indicates a better and more lucrative investment. Thus, if comparing two portfolios with similar risk profiles, given all else equal, it would be better to invest in the portfolio with a higher Sharpe Ratio.

$$\frac{R_p - R_f}{\sigma_p}$$

R_p = return of portfolio

R_f = risk-free rate

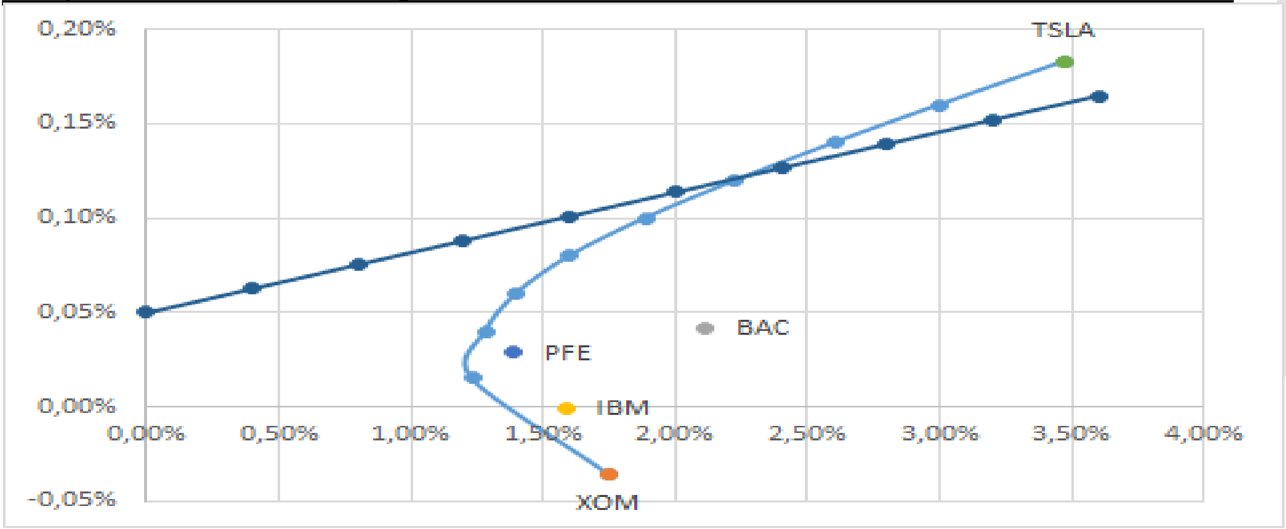
σ_p = standard deviation of the portfolio's excess return

Non Machine-Learning Results

	XOM	BAC	IBM	PFE	TSLA
Expected Return	-0.04%	0.04%	0.00%	0.03%	0.18%
Variance	0.03%	0.04%	0.03%	0.02%	0.12%
Std.Dev.	1.75%	2.11%	1.59%	1.39%	3.47%

	XOM	BAC	IBM	PFE	TSLA	**Weights**
XOM	0.030%	0.024%	0.016%	0.010%	0.015%	0.0%
BAC	0.024%	0.044%	0.020%	0.013%	0.021%	11.9%
IBM	0.016%	0.020%	0.025%	0.011%	0.015%	0.0%
PFE	0.010%	0.013%	0.011%	0.019%	0.009%	28.1%
TSLA	0.015%	0.021%	0.015%	0.009%	0.120%	60.0%
					SUM	100%

Portfolio Return	0.12%
Portfolio Risk	0.05%
Portfolio Std Dev	2.29%
Risk Free Rate	0.05%
Sharpe Ratio	0.032



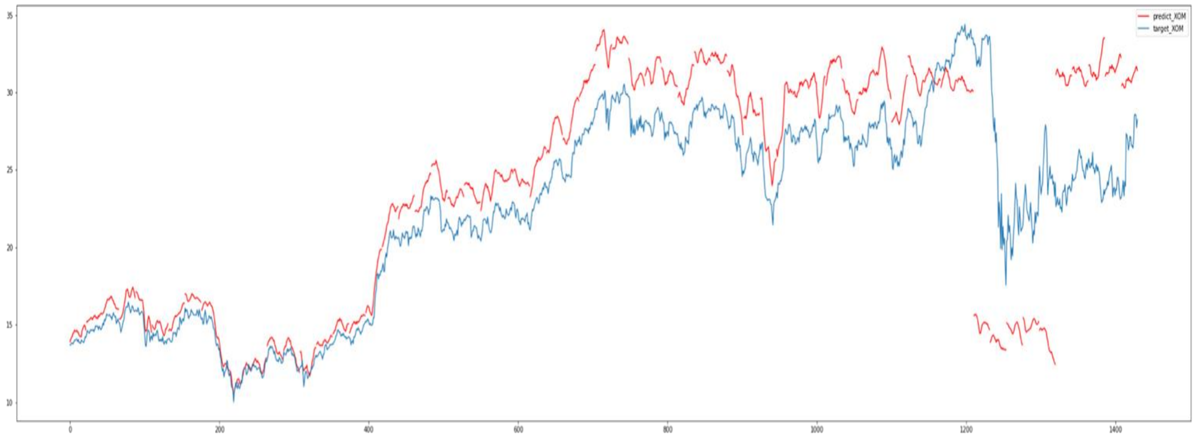
Classical Machine-Learning (LSTM +PCA + SCIPY) Overview

Long short-term memory (LSTM) is **an artificial recurrent neural network (RNN) architecture used in the field of deep learning.** Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video).

- To simulate Classical ML code **LSTM- Long Short-Term Memory model was used** with 4 layers (LSTM +Dense) of 800 neurons was used to forecast the trend of the assets for next 22 days with 60 days of feedback loop.
- Scaled the data using MinmaxScaler as LSTM is sensitive non stationary component
- PCA **(Principal Component Analysis)** was used to retrieve principal components to keep most of the useful features dataset.
- Prediction results from LSTM was used to optimize the portfolio using optimizer of SciPy library
- Optimized weights for portfolio were based on Optimized Sharpe Ratio as an input, the negative of sharpe ratio was minimized with the help of SciPy optimizer.

Classical Machine-Learning Results

LSTM Prediction Accuracy on complete Dataset



Optimized Portfolio Weights for optimized sharpe ratio

```
================================================================================
OPTIMIZED SHARPE RATIO:
--------------------------------------------------------------------------------
     fun: -0.8557134672913682
     jac: array([ 1.34512782e-04,  2.13749655e-01,  1.24640763e-04, -7.00280070e-05,
        4.42517310e-01])
 message: 'Optimization terminated successfully.'
    nfev: 65
     nit: 9
    njev: 9
  status: 0
 success: True
       x: array([2.75805628e-03, 1.06034972e-16, 3.56834884e-01, 6.40407060e-01,
       0.00000000e+00])
--------------------------------------------------------------------------------
```

- Bank of America: 0.003
- IBM: 0
- Pfizer: 0.357
- Tesla: 0.640
- Exxon: 0

Final sharpe ratio of 0.86 compared to 0.20 using optimized volatility

QAOA Overview

- QAOA (**Quantum Approximate Optimization Algorithm)** introduced by Farhi et al. is a quantum algorithm that attempts to solve such combinatorial problems.

- It is a variational algorithm that uses a unitary U(β,γ) characterized by the parameters (β,γ) to prepare a quantum state |ψ(β,γ)⟩. The goal of the algorithm is to find optimal parameters ($\beta_{opt,}\gamma_{opt}$) such that the quantum state |ψ($\beta_{opt,}\gamma_{opt)}$⟩ encodes the solution to the problem.

- The unitary U(β,γ) has a specific form and is composed of two unitaries $U(\beta)=e^{-i\beta B}$ and $U(\gamma)=e^{-i\gamma HP}$, where B is the mixing Hamiltonian and H_P is the problem Hamiltonian. Such a choice of unitary drives its inspiration from a related scheme called quantum annealing.

QAOA : Role of Mixer

- H_P is a diagonal Hamiltonian in Z basis and without mixing Hamiltonian, from a physical point, the system evolves under closed-system dynamics and the energy (represented here by H_p) is conserved, which means energy (that is, in "cost") by choosing different values of γ (Gamma).

- All of this changes, if a mixer is applied (where *e.g.* $B=\sum_i \sigma_i^x$) $U(\beta)\equiv\exp(-i\beta B)$ to state $|\psi(\gamma)\rangle$ so that:

$$|\psi(\gamma,\beta)\rangle\equiv\exp(-i\beta B)\exp(-i\gamma H_p)|\psi\rangle$$

- Since H_p and B do not commute, the dynamics generated by B will *not* in general conserve the "energy" (*i.e.* the cost) H_p.Now the (correct) QAOA energy functional

$$f(\gamma,\beta)=\langle\psi(\gamma,\beta)|HC|\psi(\gamma,\beta)\rangle$$

is no longer a constant function of γ,β and classical optimizer can be to minimize its value. That is in general values $\gamma*,\beta*$ can be found such that

$$\langle\psi(\gamma*,\beta*)|Hp|\psi(\gamma*,\beta*)\rangle<\langle\psi|Hp|\psi\rangle.$$

The exact same argument applies to any depth p of the QAOA variational Ansatz.

Conditional Value at Risk (CVaR)

The idea behind calculating the **Conditional Value at Risk** (or CVaR) is an adaptation to QAOA .

Instead of calculating the mean of all cut values c_i obtained from the measurement outcomes of the circuit to compute the cost function f, the algorithm only takes into account a fraction **α** of the highest measured cut values.

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} c_i \rightarrow f(\theta) = \frac{1}{\lceil \alpha n \rceil}\sum_{i=1}^{\lceil \alpha n \rceil} c_i$$

,

where the $\mathbf{c_i}$ are ordered decreasing in size. Since interest is in obtaining a single good solution for the original optimization problem, looking only at a fraction of the best cuts can help to speed up the optimization process.

Quantum Reinforcement-Learning (QAOA) for Portfolio Optimization

- Using the Quantum Approximation Optimization Algorithm

- Goal to minimize the following function

$$qx^T\Sigma x - \mu^T x$$

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

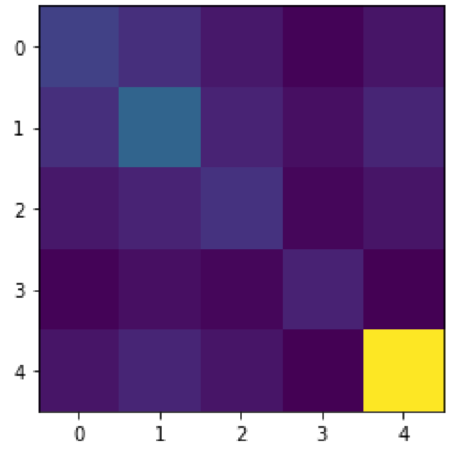
- $x \in \{0,1\}^n$ denotes the vector of binary decision variables,

which indicate which assets to pick ($x[i] = 1$) and which not to pick ($x[i] = 0$),

- $\mu \in \mathbb{R}^n$ defines the expected returns for the assets,
- $\Sigma \in \mathbb{R}^{n\times n}$ specifies the covariances between the assets,
- $q > 0$ controls the risk appetite of the decision maker,
- and B denotes the budget, i.e. the number of assets to be selected out of n.

Quantum Reinforcement-Learning Results

```
Initial Stocks ['XOM′(0),
'BAC′(1), 'IBM'(2),
'PFE'(3), 'TSLA′(4)]
```



Covariance matrix for the assets

Minimum Variance For Optimal Portfolio Selection:

Using Classical EigenSolver

```
['BAC', 'TSLA']
```

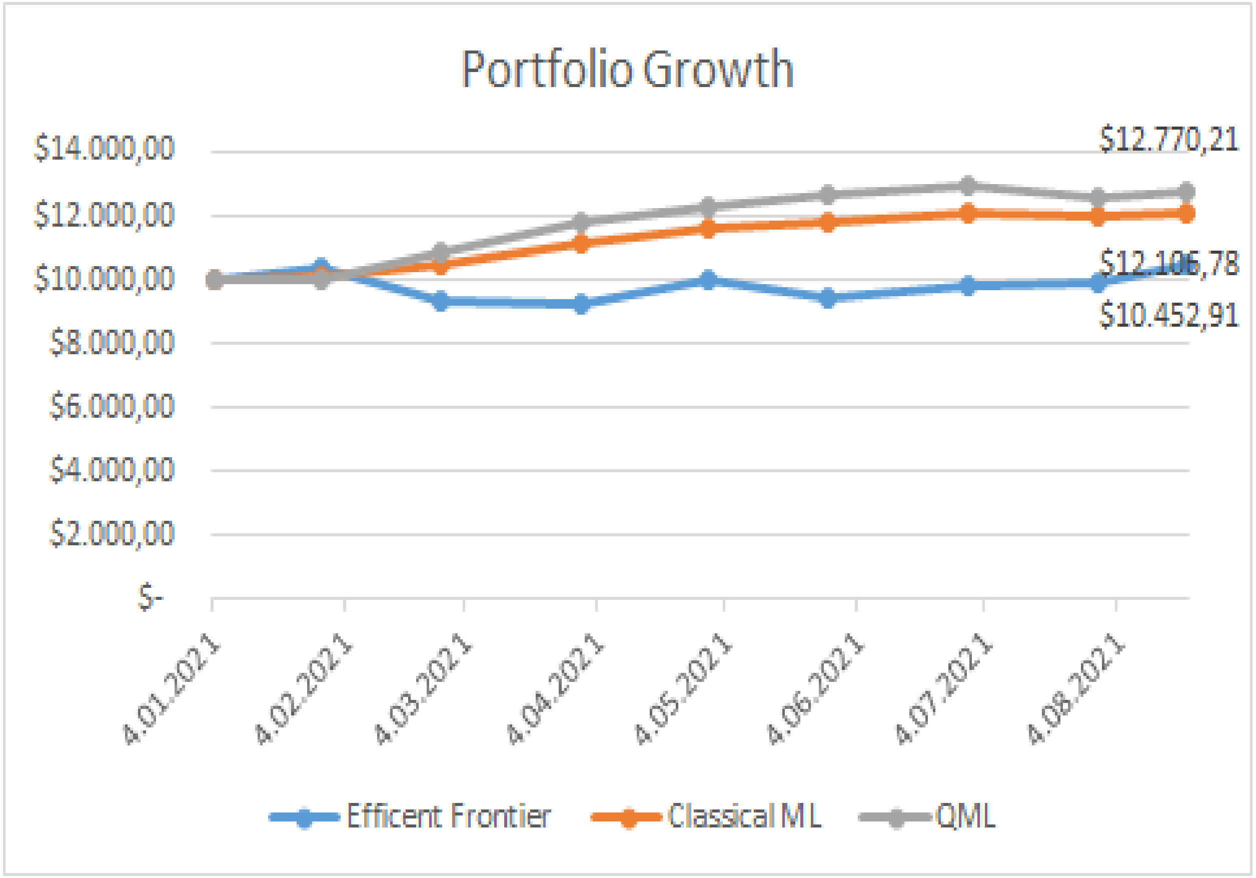
Using QAOA without Conditional Value At Risk

```
[‘XOM', 'TSLA']
```

Using QAOA with Conditional Value At Risk (**alpha= 0.05**)

```
['XOM', 'BAC', 'IBM', 'PFE′]
```

Comparative Analysis



References

- Markowitz Portfolio Optimization with a Quantum Anneale: Erica Grant, Travis Humble , University of Tennessee https://www.dwavesys.com/sites/default/files/5_Grant_Erica_ORNL.pdf

- Improving Variational Quantum Optimization using CVaR – https://arxiv.org/abs/1907.04769

- **Quantum computing for finance: overview and prospects https://arxiv.org/abs/1807.03890**

- **Qiskit Textbook**

Conclusion

- Classical ML can leverage Tensor Networks which highly improvable (GPUs, etc)
- VQE & QAOA (in NISQ) highly limited by underlying hardware advancements, though somewhat resistant to noise
- **CVaR** (Conditional Value-at-Risk) is an aggregation function which leads to faster convergence to better solutions for all combinatorial optimization problems.

Largest portfolio optimization so far possible with quantum and Tensor Network methods and with real data.
Improvable, promising

Thank You for Attention
!