Hybrid Quantum-Classical Methods for Time-Series Forecasting

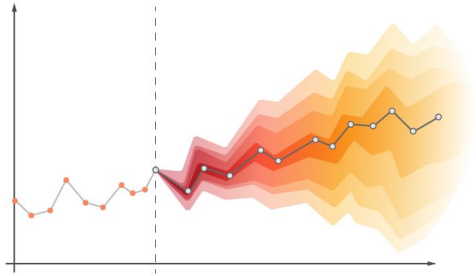
Authors: Maksims Dimitrijevs, Martins Kalis, and Ilja Repko

Presentation by Cihan, Dante, Elio, and Richard S

Paper: https://arxiv.org/abs/2412.05615v1

Table of Contents

Purpose



Time series forecasting is a tool used in fields where previous observations of a value can help to predict the future observations of the value.

Financial markets -> stock price of a company over time
Meteorology -> temperature at a location over time

To do so, we use historical data to learn the patterns in observations.

“develop a model that effectively captures underlying patterns in historical data and extrapolates them to future points.” [Dimitrijevs et al.]

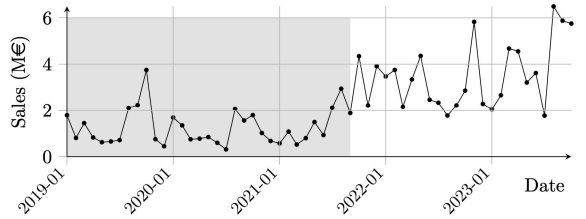
This paper tries to answer:

1. **Can we use NISQ era quantum computers for time series forecasting?**
2. **How effective are the quantum methods compared to classical methods for time series forecasting?**

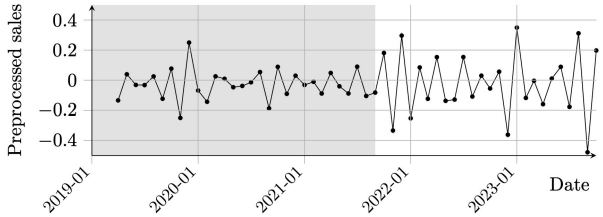
Image: https://medium.com/analytics-vidhya/time-series-forecasting-c73dec0b7533

Paper Overview

- 2 Classical Methods:
  - Linear Regression
  - Deep Learning
- 2 Quantum Methods:
  - Parameterized Quantum Circuit (PQC)
  - Variational Quantum Linear Solver (VQLS)
- 58 synthetic data points mimicking typical sales data
- Feature window of 12 data points for each prediction
- Data Preprocessing:
  - Differencing: enhance stationarity and removes a point
  - Normalization + Scaling: enable quantum circuit encoding



(a) Original data. Data in the shaded area is used for training and scaling adjustments The rest is used for testing.



(b) Preprocessed data after applying differencing (change in sales vs. absolute sales) and scaling the differences to the range [-0.25, +0.25]. Note that test-time data can fall outside of this range.

Data from Dimitrijevs et al.

Methods and Results

Classical Methods: Linear Regression & Neural Network

"... performance was compared against two **classical baselines:** a **linear regression model** and a simple **deep learning model**." [Dimitrijevs et al.]

"The deep learning model consisted of two hidden layers, each with 12 units and ReLU activation, **to mirror the limitations of the currently available quantum devices.**" [Dimitrijevs et al.]

"The primary evaluation metric used across all models was the Mean Squared Error (MSE)." [Dimitrijevs et al.]

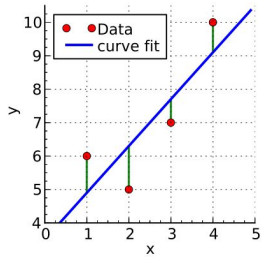


Image: https://en.wikipedia.org/wiki/Linear_regression

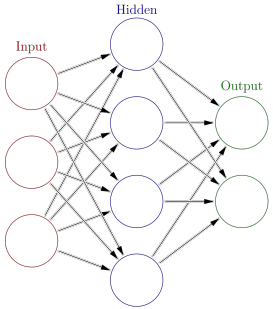
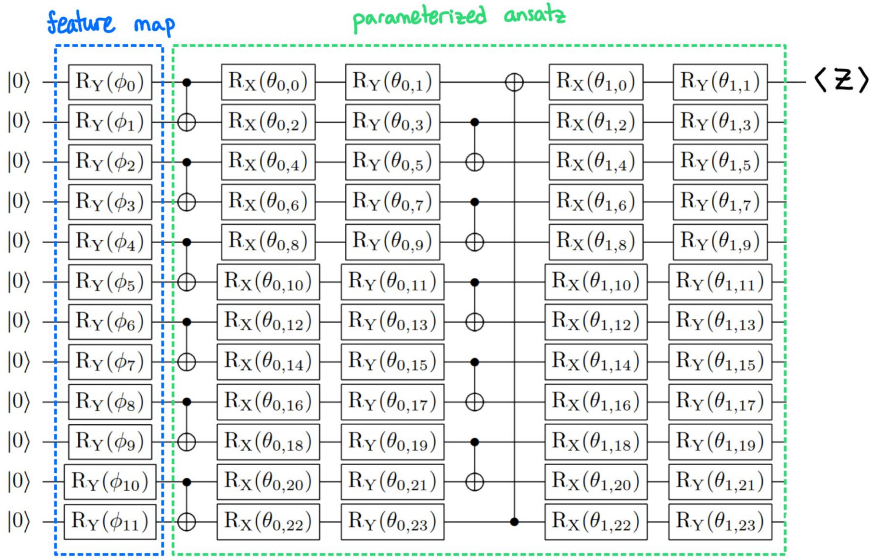


Image: https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

Method 3: Parameterized Quantum Circuit (PQC)

- Feature Map: RY gates to angle encode features
- Parameterized Ansatz: 2 layers of single qubit parameterized gates + entangling gates
- Cost Function: MSE of ⟨Z⟩ of wire 0 and expected value
- Classical Optimizers:
  - L-BFGS-B: Gradient based method
  - COBYLA: Linear approximation method



Quantum Circuit from Dimitrijevs et al.

Method 4: Variation Quantum Linear Solver (VQLS)

We can turn a linear regression problem into a quantum optimization problem:

Inputs: X, y
Unknowns: w

$$Xw = y$$
$$X^T X w = X^T y \longrightarrow M|w\rangle = |b\rangle$$
$$Mw = b$$

We can use a variation solver on w with a minimized cost when the vectors are the same.

$$C(\phi) = 1 - \frac{\langle b|M|w(\phi)\rangle}{|M|w(\phi)\rangle|} \longrightarrow M|w(\phi)\rangle = |b\rangle$$

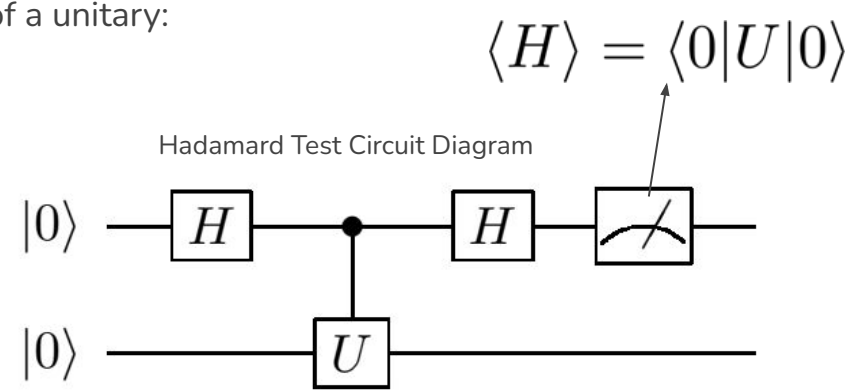
Method 4: Variation Quantum Linear Solver (VQLS) - Expectation Values

How can we find $\langle b|M|w\rangle$?

Hadamard Test gives us the expectation of a unitary:

$$\langle H\rangle = \langle 0|U|0\rangle$$

But we need to make M unitary!



Hadamard Test from Dimitrijevs et al.

Method 4: Variation Quantum Linear Solver (VQLS) - M Decomposition

Decompose M into products of Pauli Matrices For each qubit:

$$M = \sum_{i=0}^{4^n-1} \alpha_i M_i \qquad M_i = \bigotimes_{j=0}^{n-1} \sigma_{imod4^j} \qquad \alpha_i = \frac{Tr(M_i M)}{2^n}$$

Computing the expectation value as:

$$\langle b|M|w\rangle = \sum_{i=0}^{4^n-1} \alpha_i \langle b|M_i|w\rangle$$

Using the Hadamard Test:

$$\langle H\rangle = \langle b|M_i|w\rangle = \langle 0|B^t M_i W|0\rangle$$

Where B and W generate the states $|b\rangle$ **and** $|w(\phi)\rangle$

Method 4: Variation Quantum Linear Solver (VQLS) - Paper Issues

The paper reports no successful tests of VQLS.

We identified two issues with their cost function:

- Incorrect sign is permitted
$$c(|w\rangle) = c(-|w\rangle)$$
- Normalization is lost
$$|M|w(\phi)\rangle|^2 \neq 1$$

$$M|w(\phi)\rangle = |b\rangle$$

Is not the lowest energy solution anymore!

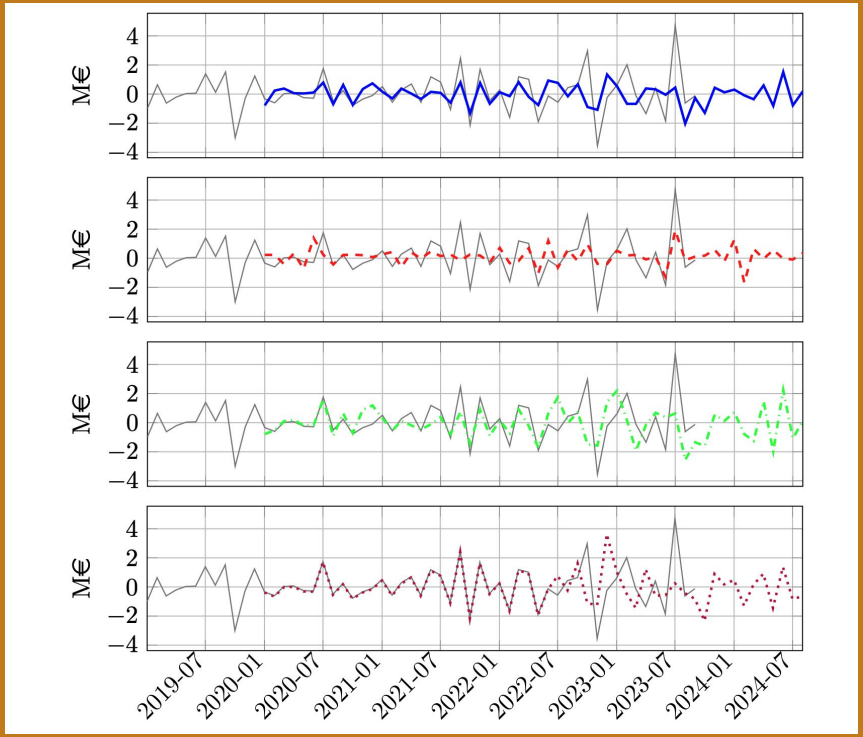
Paper Cost Function

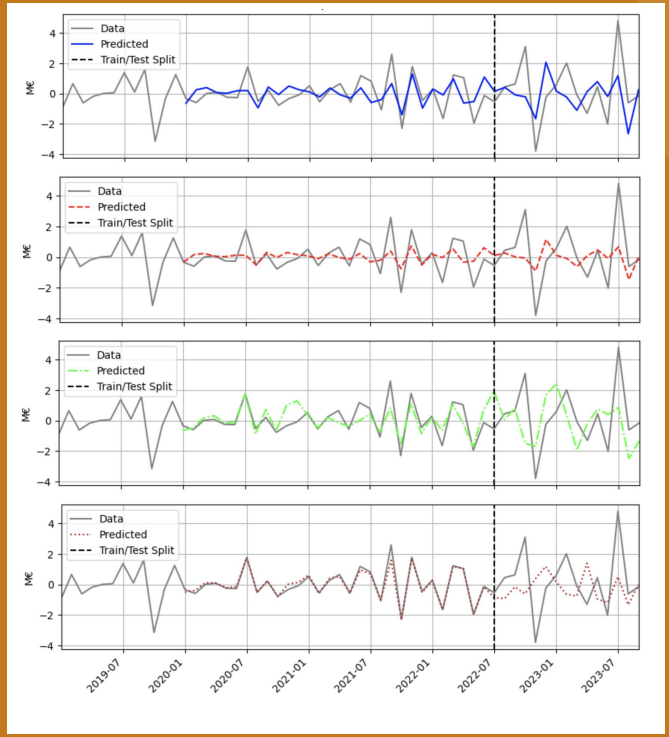
$$C = 1 - |\langle b|M|w(\phi)\rangle|^2$$

Our Cost Function

$$C = 1 - \frac{\langle b|M|w(\phi)\rangle}{|M|w(\phi)\rangle|}$$

Results Summary





L-BFGS-B

COBYLA

Linear Model

Neural Network

Data

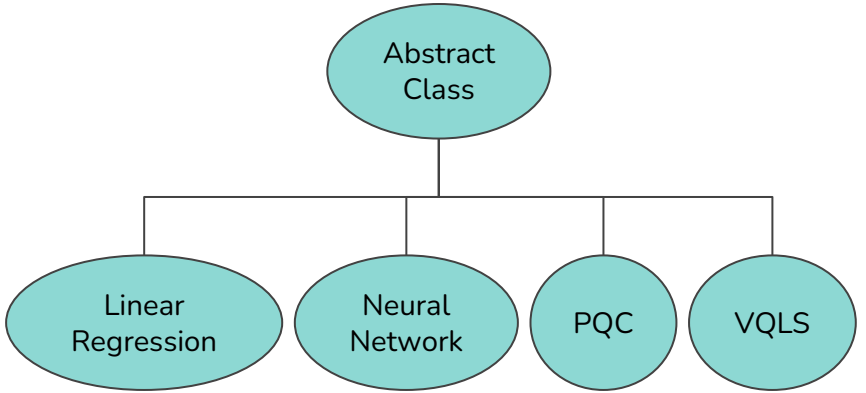
Paper

Model	Training Set MSE	Test Set MSE
PQC with COBYLA	0.01257	0.02106
PQC with L-BFGS-B	0.00612	0.04418
Classical Linear Model	0.00514	0.05177
Classical Neural Network	0.00003	0.05767

Ours

Model	Training Set MSE	Test Set MSE
PQC with COBYLA	0.00637	0.02512
PQC with L-BFGS-B	0.00582	0.02583
Classical Linear Model	0.00360	0.03680
Classical Neural Network	0.00035	0.03659

Software Overview



- Design
  - Abstract class provides a standard interface for everything
    - Training, predicting, scoring, saving and loading parameters to a file
  - Static methods on the abstract class for data loading, preprocessing, and postprocessing
  - Individual or combined training options
  - Automated formatting and linting as well as GitHub Actions checks
- Implementation
  - Scikit-learn for linear regression
  - Keras for neural network
  - Scipy for hybrid optimizations
  - Pennylane for quantum components

Paper Shortcomings & Reproducibility

- **Dataset:** The dataset used in the paper had only 58 data points. Moreover, it is a “synthetic dataset designed to mimic typical sales data patterns.” [Dimitrijevs et al.]
- **Resources Translation:** “The deep learning model consisted of ..., to mirror the limitations of the currently available quantum devices.” [Dimitrijevs et al.]
- **Overfitting Baseline:** The authors overfitted their classical baseline neural network for unclear reasons.
- **VQLS Implementation:** Their VQLS implementation failed. We found errors in their implementation which were corrected to calculate accurate weights.
- **Resource Requirements for VQLS:** VQLS training took over 24hrs to converge for 3 qubits (8 weights) due to the 4^n simulation requirements with n qubits.

Live Demo: Time-Series Forecasting

Live Demo: VQLS

Questions?