Generative Quantum Learning of Joint Probability Distribution Functions

Quantum Journal Club

Florian Rehm

13.01.2022

Summary (abstract)

- Goal: generating joint probability distributions
- Two approaches:
  - Quantum Generative Adversarial Networks (QGAN)
  - Quantum Circuit Born Machines (QCBM)
- Run on trapped ion quantum computers from IonQ for up to 8 qubits
- Outperform classical generative learning
  - (a neural network with the same number of parameters as the quantum circuit)
- Exponential advantage in model‘s expressivity

- Link to paper: https://arxiv.org/pdf/2109.06315.pdf (2021)

Use Case / Data Set

- Daily returns of stock exchange courses for Apple and Microsoft

Stock courses | X space (real data space) | U space (copula space)

For this distribution they train

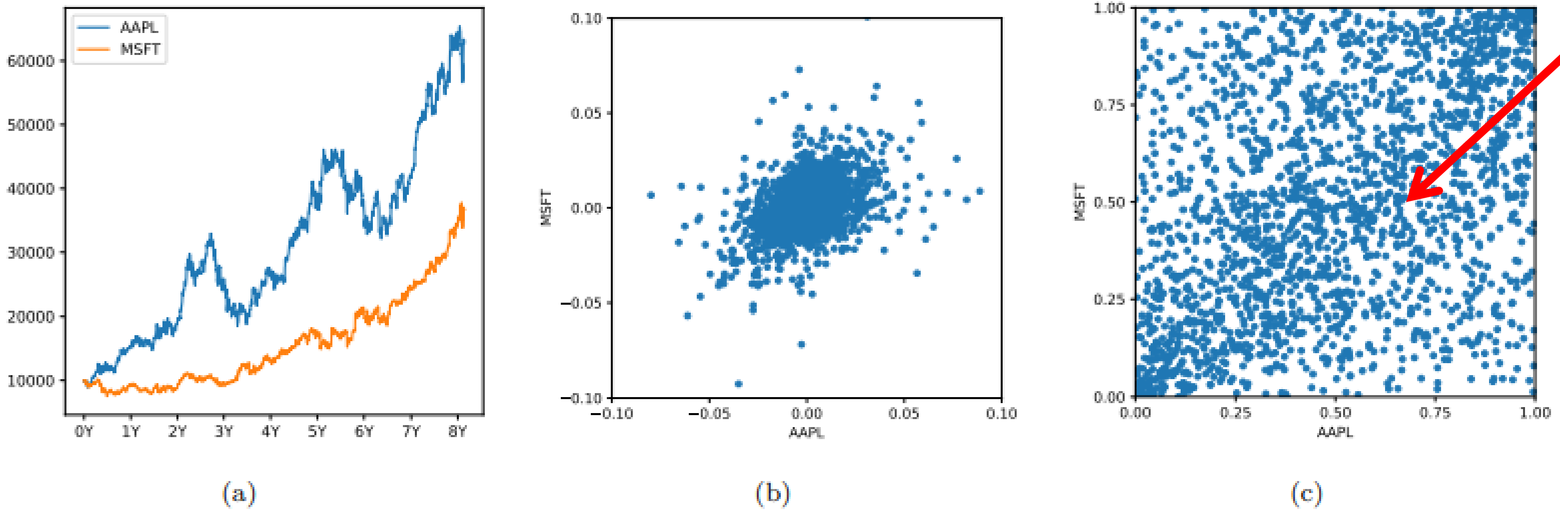
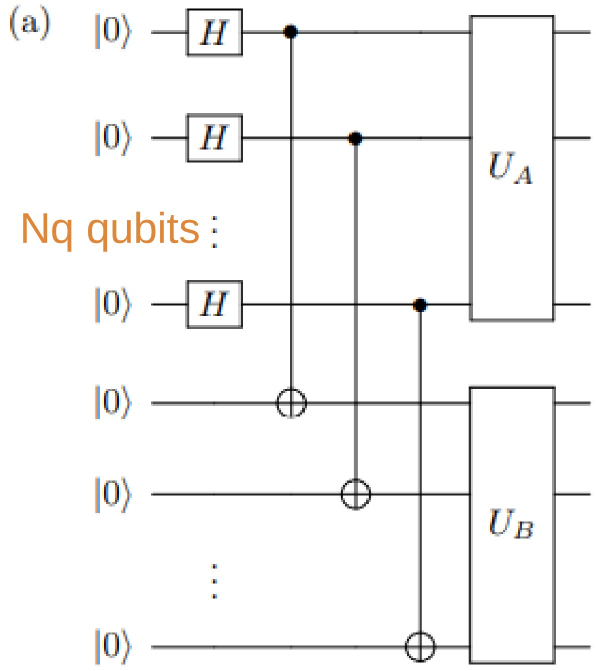


Fig. 2: (a) Hypothetical Growth of $10,000 invested in AAPL and MSFT between 2010-2018 (b) Scatter Plot of Daily Returns (c) Scatter Plot of Data after Probability Integral Transform

A copula is a multivariate cumulative distribution function for which the marginal probability distribution of each variable is uniform on the interval [0, 1].
Copulas are used to describe/model the dependence (inter-correlation) between random variables.
https://en.wikipedia.org/wiki/Copula_(probability_theory)

Qopula Circuit

- Qopula Circuit: A quantum circuit that can represent a maximally entangled state for every copula



1. Creation of Nq Bell pairs between two registers A and B what results in:

$$\frac{1}{2^{N_q/4}} \sum_{i=0}^{2^{N_q/2}-1} |i_A\rangle |i_B\rangle .$$

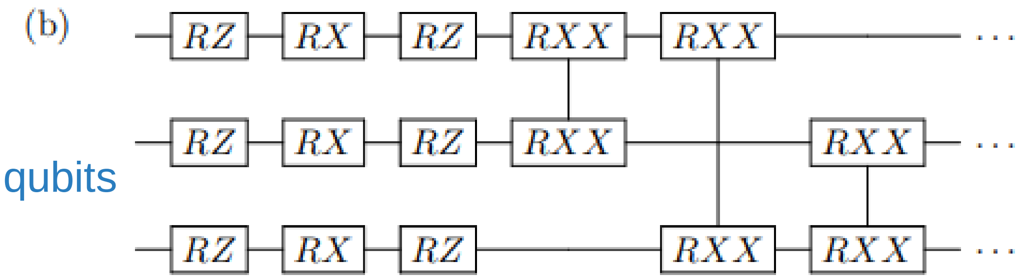
   1. Nq=2: number of qubits

2. Unitaries U_A and U_B act on register A and B
   1. U_A provides samples for the first random variable
   2. U_B for the second

Quantum Circuit

- One layer of the quantum circuit for U_A or U_B

n qubits



  - RXX can be replaced by standard gates

- The number of qubits can be freely chosen and determines the discretization of the output → More qubits, more accurate results.
  - The outputs of the registers are measured to a bitstring and transformed to a variable in the domain [0, 1] by:

$$x = \frac{1}{2^{1+n}} + \sum_{i=0}^{n} \frac{1}{2^n} x_n.$$

QGAN

- Generator comprises 6 qubits and 1 layer of ansatz
  → 24 trainable parameters

- They add additional random bits to increase the discretization
  - x1, x2, x3 → x1, x2, x3, x4, …, x22

- Discriminator has one hidden layer with dimension 32

GAN vs QGAN

- They make a classical generator with the same number of parameters as the quantum circuit

- QGAN training: usually parameter-shift rule
  - 24 parameters -> 48 circuit executions to compute the gradients (sequentially computed)

- Too slow -> they use Simultaneous Pertubation Stochastic Approximation (SPSA) algorithm
  - Only 10 circuit execution steps (5 iterations with 2 executions each)

QGAN Training Algorithm

Algorithm 2: QGAN Training Loop

Result: Trained Generator G

Initialize Network and angles of the Quantum Circuit ansatz;

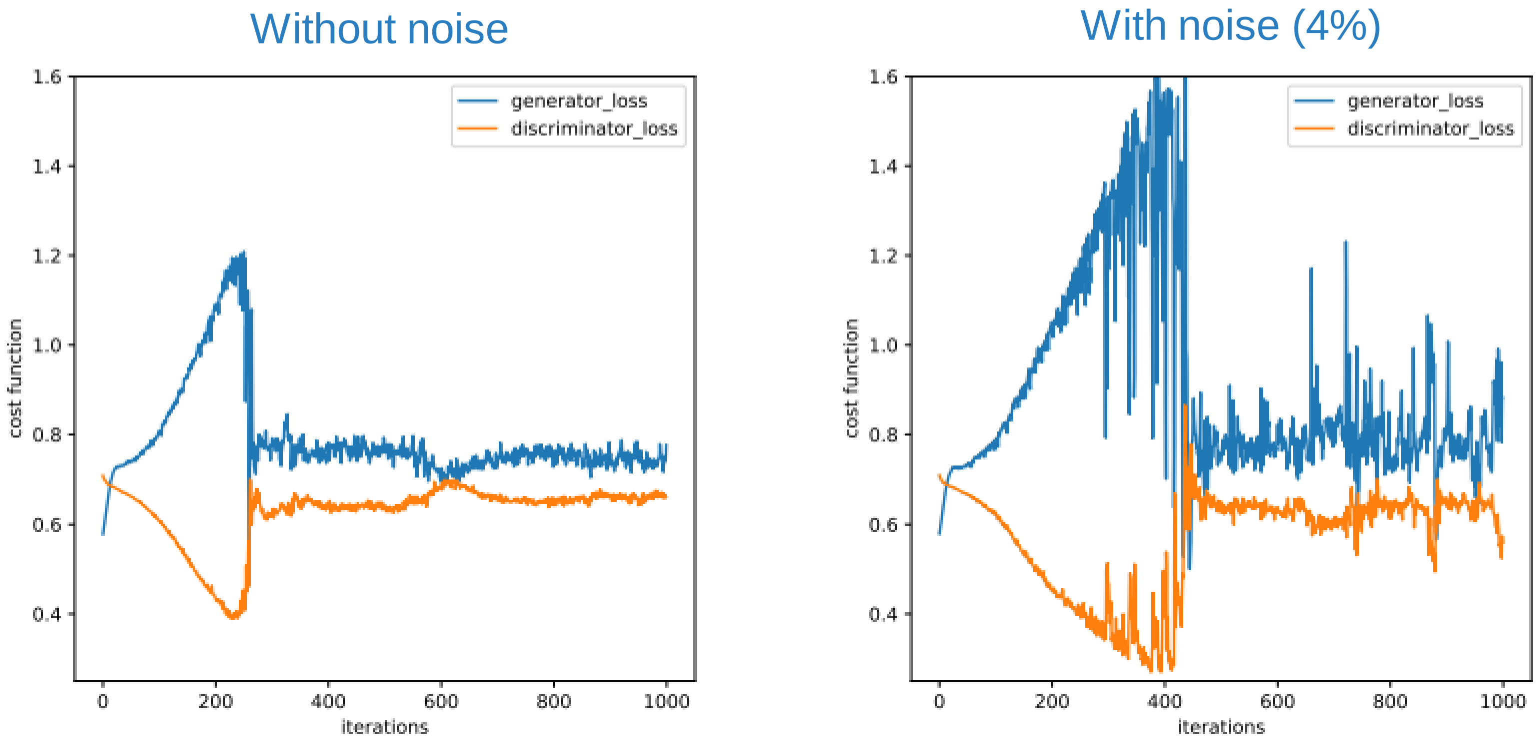
for $i \leftarrow 1$ **to** *iterations* **do**

- Run the quantum generator with m shots to generate m measurements ;
- Sample minibatch of m data examples $\{x^{(l)}\}$;
- Calculate Discriminator Loss L_D ;
- Update the discriminator by descending its stochastic gradient $\nabla_{\theta_d} L_D(\theta_g, \theta_d)$;
- Calculate Generator Loss L_G ;
- Update the Quantum Generator by using SPSA algorithm ;

end

QGAN Training

- More fluctuations with hardware noise
- 4% depolarizing noise



(a) Loss Plot from Simulator

(b) Loss Plot from Experiment

Fig. 4: Plot of the loss functions from QGAN training for $N_q = 6$ qubits.

Quantum Circuit Born Machine (QCBM)

- QCBM similar as generator of QGAN but with different training objective
- QCBM minimizes the distance between the output distribution and the target distribution
  - They use Kullback-Leibler (KL) divergence
    - KL: quantifies how different two probability distributions are

Algorithm 3: QCBM Training Loop

Result: Trained Quantum Circuit

Initialize angles of the Quantum Circuit ansatz;

for $i \leftarrow 1$ **to** *iterations* **do**

  Run the quantum circuit with m shots to generate m measurements ;

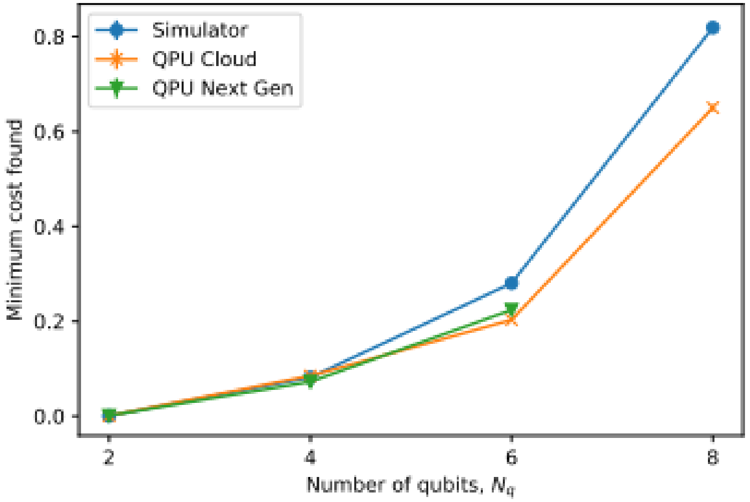
  Calculate the KL divergence d_{KL} ;

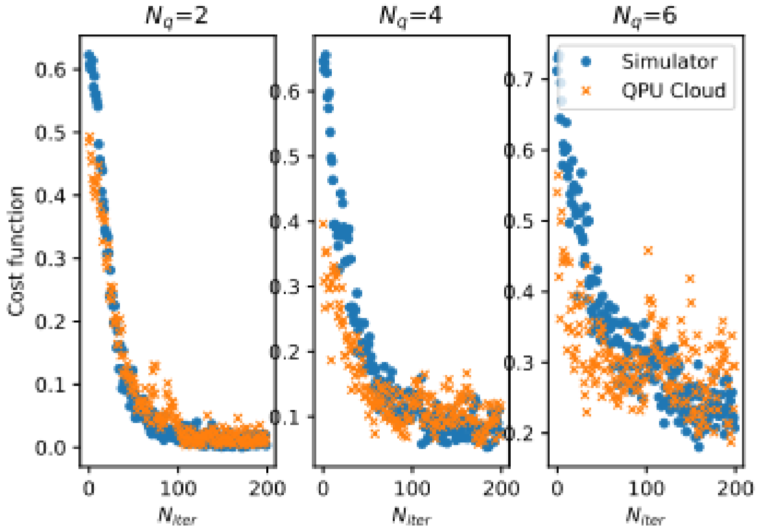
  Update the circuit angles by using SPSA algorithm ;

end

QCBM Training

- (left plot) Noise increases accuracy (green and orange curve with noise are lower than simulator without noise in blue)





- (right plots) The more qubits one uses the more unstable the training becomes due to noise (more qubits lead to more noisy gates)

Evaluation

Model	D_{KS} (the smaller the better)
Parametric model	0.0449
Classical GAN	0.0363 - 0.0508
QGAN simulation	0.0320 - 0.0396
QGAN experiment, QPU cloud	0.0352
QCBM simulation	0.0425 - 0.0520
QCBM experiment, QPU cloud	0.0373 - 0.0515
QCBM experiment, QPU Next Gen	0.0330 - 0.0510

- Kolmogorov-Smirnov test for evaluation
- Table shows average over 20 iterations
- QGAN outperforms GAN
- QCBM worse then QGAN

- QGAN vs GAN:
  - both models could perform better with deeper networks / circuits
  - QGAN model has higher expressivity
  - Quantum models can be trained with a higher learning rate than classical GAN what leads to much less training iterations (in theory faster training)
  - QGAN took 2 weeks to complete training on IonQ QPU
    - QGAN simulation only 4 minutes, classical GAN 6 minutes

Conclusion

- The number of parameters scales quadratically with the number of qubits (undesierable)
- Noise plays a positive role in machine learning models
- QGAN performs best (better than classical GAN and QCBM)
- Since quantum models cannot be tested at scale their efficancy cannot be nummerically predicted