Comparison of gradient and derivative-free learning methods for quantum circuit Born machine

Vlastimil Hudeček[1] Aurél Gábris[1, 2]

[1]Department of Physics, Czech Technical University in Prague: Faculty of Nuclear Sciences and Physical Engineering

[2]Institute for Solid State Physics and Optics, HUN-REN Wigner Research Centre for Physics

June 17, 2024

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML
  - QCBM as a for of PQC with classical analogy

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML
  - QCBM as a for of PQC with classical analogy
- Learning of PQC through optimization of parameters

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML
  - QCBM as a for of PQC with classical analogy
- Learning of PQC through optimization of parameters
- Optimization methods: derivative and gradient-free

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML
  - QCBM as a for of PQC with classical analogy
- Learning of PQC through optimization of parameters
- Optimization methods: derivative and gradient-free
- Comparison of methods with COBYLA, CMA-ES and ADAM optimizers

Quantum generative machine learning and objectives of the work

- Quantum generative machine learning (QML) and parameterized quantum circuits (PQC)
  - Generating samples from probability distributions
  - PQC as a circuit for QML
  - QCBM as a for of PQC with classical analogy
- Learning of PQC through optimization of parameters
- Optimization methods: derivative and gradient-free
- Comparison of methods with COBYLA, CMA-ES and ADAM optimizers
- Numerical simulations and verification on quantum computers

Main results of the research

- *New finding:* Connection between the method of optimization of QCBM parameters and the circuit depth and the number of circuit parameters.

Main results of the research

- *New finding:* Connection between the method of optimization of QCBM parameters and the circuit depth and the number of circuit parameters.
  - For a limited depth circuit the gradient-free optimization is more effective

Main results of the research

- *New finding:* Connection between the method of optimization of QCBM parameters and the circuit depth and the number of circuit parameters.
  - For a limited depth circuit the gradient-free optimization is more effective
  - For complex cases with deeper circuits the derivative optimization has an advantage

Main results of the research

- *New finding:* Connection between the method of optimization of QCBM parameters and the circuit depth and the number of circuit parameters.
  - For a limited depth circuit the gradient-free optimization is more effective
  - For complex cases with deeper circuits the derivative optimization has an advantage
- Effective implementation of the QCBM for quantum computer with parallel execution of QCBM circuits.

Restricted Boltzmann Machine

- Two layer classical neural network for generative deep learning CITE, same as QCBM serves as a generator for joint probability distribution

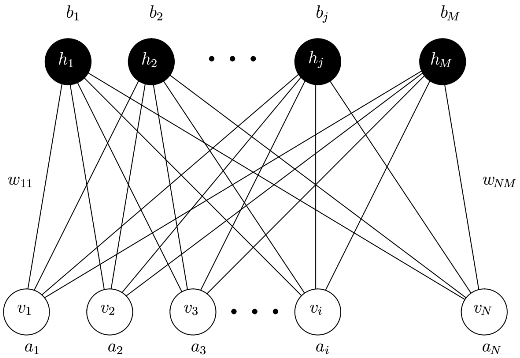

Figure 1: Restricted Boltzman Machine as a two layer neural network with a visible (white) and a hidden (black) layer nodes CITE

Restricted Boltzmann Machine

- Two layer classical neural network for generative deep learning CITE, same as QCBM serves as a generator for joint probability distribution
- Joint probability determined by the *Boltzmann distribution function*

$$P(\mathbf{v}, \boldsymbol{h}) = \frac{\exp\{-E(\mathbf{v}, \boldsymbol{h})\}}{Z}$$

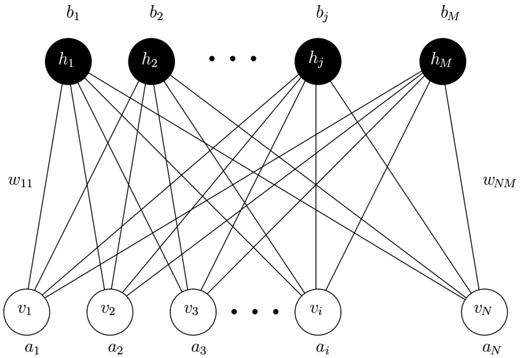

Figure 1: Restricted Boltzman Machine as a two layer neural network with a visible (white) and a hidden (black) layer nodes CITE

Quantum Circuit Born Machine

- Using the Born rule as a mean of generating (probability) distribution instead of relying of Boltzman distribution

$$P(\mathbf{v}, \boldsymbol{h}) = \left\langle \mathcal{P}^{\dagger}_{\mathbf{v},\boldsymbol{h}} \mathcal{P}_{\mathbf{v},\boldsymbol{h}} \right\rangle ,$$

  where $\mathcal{P}_{\mathbf{v},\boldsymbol{h}}$ denotes measurement operator which fulfills $\sum_{\mathbf{v},\boldsymbol{h}} \mathcal{P}_{\mathbf{v},\boldsymbol{h}} = \mathbb{I}$

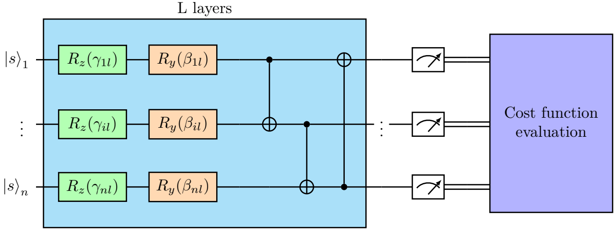

Figure 2: Schematic of a QCBM

Quantum Circuit Born Machine

- Using the Born rule as a mean of generating (probability) distribution instead of relying of Boltzman distribution

$$P(\mathbf{v}, \boldsymbol{h}) = \left\langle \mathcal{P}^{\dagger}_{\mathbf{v},\boldsymbol{h}} \mathcal{P}_{\mathbf{v},\boldsymbol{h}} \right\rangle,$$

  where $\mathcal{P}_{\mathbf{v},\boldsymbol{h}}$ denotes measurement operator which fulfills $\sum_{\mathbf{v},\boldsymbol{h}} \mathcal{P}_{\mathbf{v},\boldsymbol{h}} = \mathbb{I}$
- For the case of PQC with parameter vector $\boldsymbol{\theta}$ this can be rewritten as $P = \langle \phi | \psi_{\boldsymbol{\theta}} \rangle$

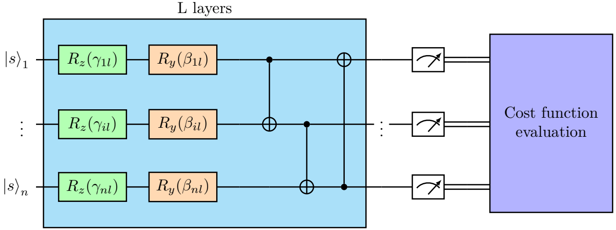

Figure 2: Schematic of a QCBM

Test distribution

- Mixing of normal distributions

$$P(x) = \frac{1}{N}\sum_{i=1}^{4} w_i \exp\left\{-\frac{(x-p_i)^2}{2\sigma_i^2}\right\}$$

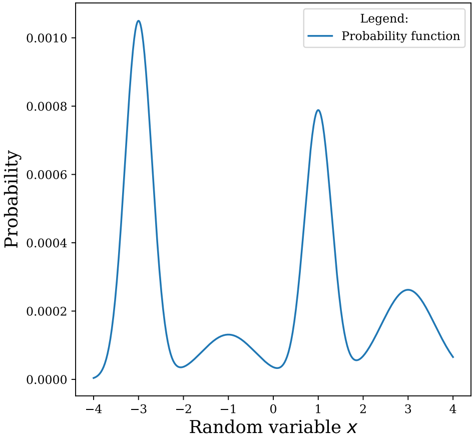

Figure 3: Graph of the benchmark probability distribution

Test distribution

- Mixing of normal distributions

$$P(x) = \frac{1}{N}\sum_{i=1}^{4} w_i \exp\left\{-\frac{(x-p_i)^2}{2\sigma_i^2}\right\}$$

- Table of parameters of mixed normal distributions

i	p_i	σ_i	w_i
1	-3	0.3	0.4
2	-1	0.6	0.1
3	$+1$	0.3	0.3
4	$+3$	0.6	0.2

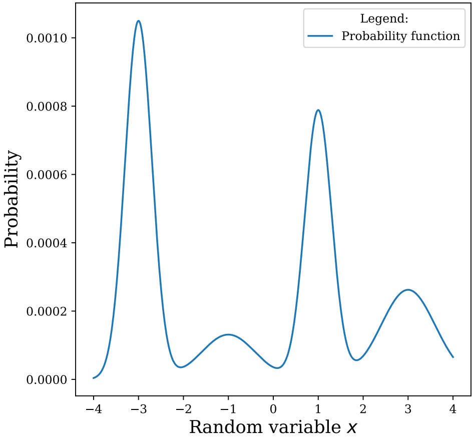

Figure 3: Graph of the benchmark probability distribution

Cost function for optimization algorithms

Total Variation (TV)

Maximum Mean Discrepancy (MMD)

- Cost function formula

$$
\begin{aligned}
f_{\mathsf{TV}}(\boldsymbol{\theta}) &= f_{\mathsf{TV}}\left(P_{\mathsf{model},\boldsymbol{\theta}}(x), P_{\mathsf{data}}(x)\right) \\
&= \sum_x \Big(P_{\mathsf{model},\boldsymbol{\theta}}(x) - P_{\mathsf{data}}(x)\Big)
\end{aligned}
$$

Cost function for optimization algorithms

Total Variation (TV)

- Cost function formula

$$f_{\mathrm{TV}}(\boldsymbol{\theta}) = f_{\mathrm{TV}}\left(P_{\mathrm{model},\boldsymbol{\theta}}(x), P_{\mathrm{data}}(x)\right) = \sum_x \left(P_{\mathrm{model},\boldsymbol{\theta}}(x) - P_{\mathrm{data}}(x)\right)$$

Maximum Mean Discrepancy (MMD)

- Cost function formula

$$f_{\mathrm{MMD}}(\boldsymbol{\theta}) \approxeq \mathbb{E}\left[K(P_{\mathrm{model},\boldsymbol{\theta}}, P_{\mathrm{model},\boldsymbol{\theta}})\right] - 2\mathbb{E}\left[K(P_{\mathrm{model},\boldsymbol{\theta}}, P_{\mathrm{data}})\right] + \mathbb{E}\left[K(P_{\mathrm{data}}, P_{\mathrm{data}})\right]$$

Cost function for optimization algorithms

Total Variation (TV)

- Cost function formula

$$\begin{aligned} f_{\mathsf{TV}}(\boldsymbol{\theta}) &= f_{\mathsf{TV}}\left(P_{\mathsf{model},\boldsymbol{\theta}}(x), P_{\mathsf{data}}(x)\right) \\ &= \sum_x \Big(P_{\mathsf{model},\boldsymbol{\theta}}(x) - P_{\mathsf{data}}(x)\Big) \end{aligned}$$

Maximum Mean Discrepancy (MMD)

- Cost function formula

$$\begin{aligned} f_{\mathsf{MMD}}(\boldsymbol{\theta}) \approxeq\ & \mathbb{E}\left[K(P_{\mathsf{model},\boldsymbol{\theta}}, P_{\mathsf{model},\boldsymbol{\theta}})\right] \\ & - 2\mathbb{E}\left[K(P_{\mathsf{model},\boldsymbol{\theta}}, P_{\mathsf{data}})\right] \\ & + \mathbb{E}\left[K(P_{\mathsf{data}}, P_{\mathsf{data}})\right] \end{aligned}$$

- Kernel function

$$K(x, y) = \frac{1}{k} \sum_{i=1}^{k} \exp\left\{-\frac{1}{2\sigma_i}|x - y|^2\right\}$$

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer
  - Usage of first derivative

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer
  - Usage of first derivative
  - Batch calculation of gradient

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer
  - Usage of first derivative
  - Batch calculation of gradient
  - Analytic gradient for MMD cost

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer
  - Usage of first derivative
  - Batch calculation of gradient
  - Analytic gradient for MMD cost
  - Momentum can increase noise resilience

Tested optimization algorithms

Gradient-free optimization

- COBYLA optimizer
  - Low number of cost function evaluations
  - Low noise resilience, higher number of shots
- CMA-ES optimizer
  - Allows for batch calculations
  - Able to converge fast

Derivative optimization

- ADAM optimizer
  - Usage of first derivative
  - Batch calculation of gradient
  - Analytic gradient for MMD cost
  - Momentum can increase noise resilience

- Finding the relation between circuit depth and the learning result for each of the optimization algorithms.

Optimal depth for the COBYLA optimized QCBM

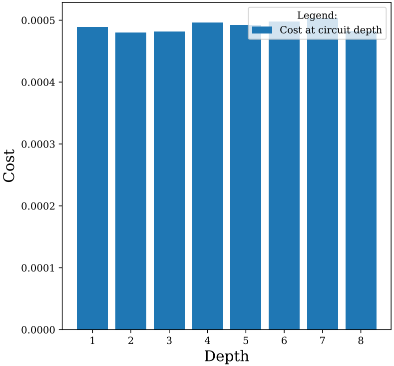

Figure 4: The TV cost reached for different depths of the QCBM circuit with the COBYLA optimizer

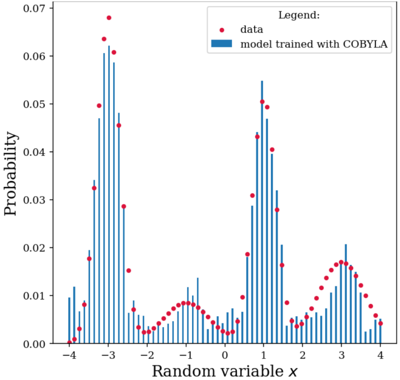

Figure 5: Probability distribution with lowest cost generated with the COBYLA optimizer

Optimal depth for the CMA-ES optimized QCBM

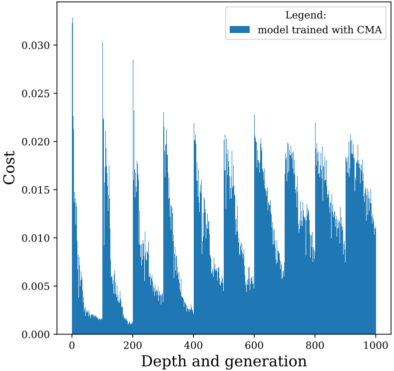

Figure 6: TheTV cost reached for different depths of the QCBM circuit with CMA-ES optimizer

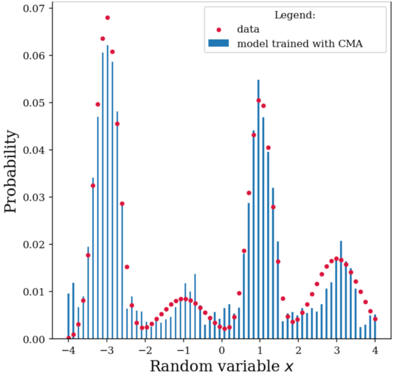

Figure 7: Probability distribution with lowest cost generated with the CMA-ES optimizer

Optimal depth for the ADAM optimized QCBM

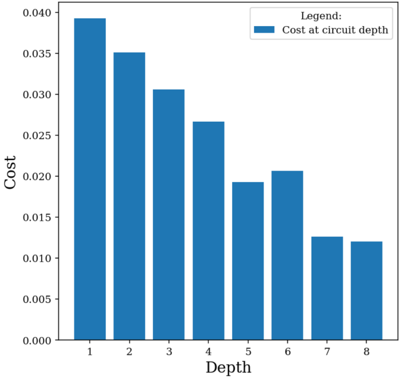

Figure 8: TheMMD cost reached for different depths of the QCBM circuit with ADAM optimizer

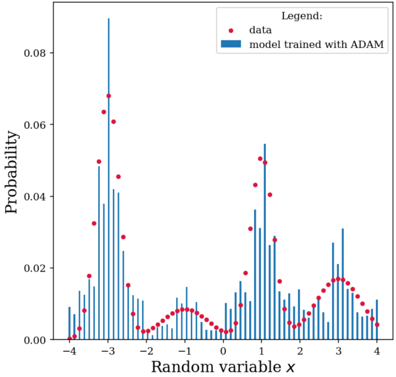

Figure 9: Probability distribution with lowest cost generated with the ADAM optimizer

Plot TV and MMD cost during learning

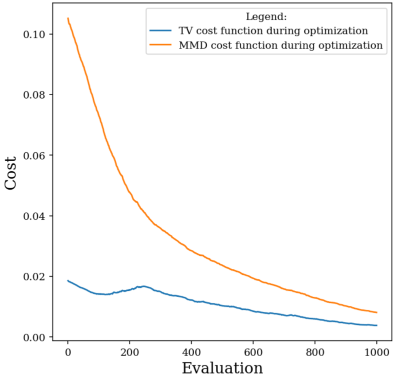

Figure 10: Plot of TV and MMD cost function during learning on simulator

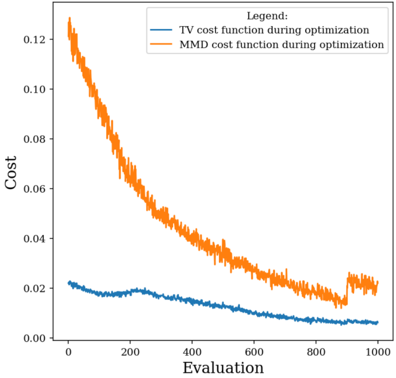

Figure 11: Plot of TV and MMD cost function during learning on `ibmq_mumbai`

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.

Open questions:

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.
- Observed similarity to learning of classical RBM in the connection depth and optimal learning method.

Open questions:

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.
- Observed similarity to learning of classical RBM in the connection depth and optimal learning method.
- Cost functions, e.g. MMD, used in classical machine learning can be applied to quantum machine learning for performance advantage.

Open questions:

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.
- Observed similarity to learning of classical RBM in the connection depth and optimal learning method.
- Cost functions, e.g. MMD, used in classical machine learning can be applied to quantum machine learning for performance advantage.
- Tendency of QCBM to produce periodic distributions.

Open questions:

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.
- Observed similarity to learning of classical RBM in the connection depth and optimal learning method.
- Cost functions, e.g. MMD, used in classical machine learning can be applied to quantum machine learning for performance advantage.
- Tendency of QCBM to produce periodic distributions.

Open questions:

- Scaling the learning for larger PQC leads to *sampling problem*.

Conclusion

- *Numerical results* show the connection between the depth of PQC and the optimal choice of optimizer for learning.
- Observed similarity to learning of classical RBM in the connection depth and optimal learning method.
- Cost functions, e.g. MMD, used in classical machine learning can be applied to quantum machine learning for performance advantage.
- Tendency of QCBM to produce periodic distributions.

Open questions:

- Scaling the learning for larger PQC leads to *sampling problem*.
- Introduction of *hints* for the learning of difficult distributions.

Thank you for your attention

Figure 12: QR code leading to the project repository

QCBM learning video