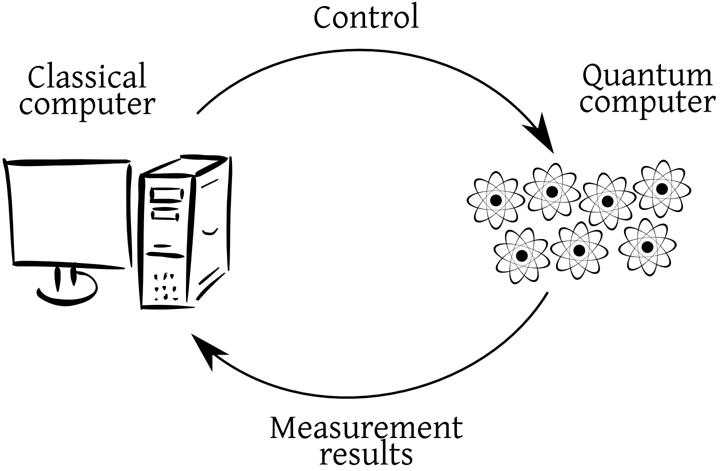
Quantum Machine Learning with AQMLator

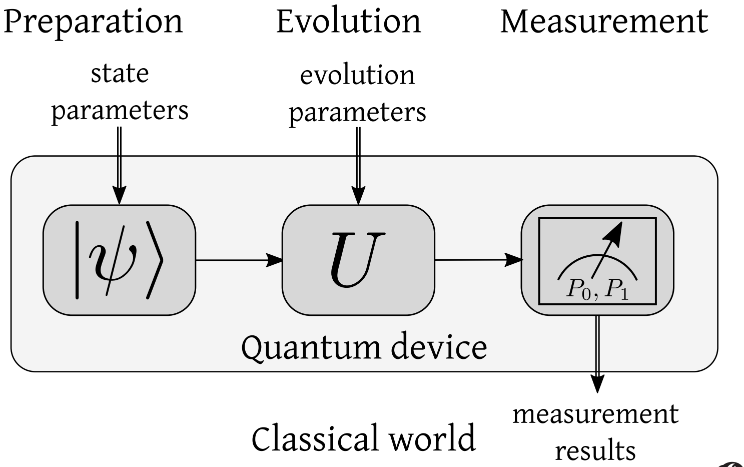
Tomasz Rybotycki

Systems Research Institute, Polish Academy of Sciences
Nicolaus Copernicus Astronomical Center, Polish Academy of Sciences
Center of Excellence in Artificial Intelligence, AGH University of Cracow

2nd International Workshop on Machine Learning and Quantum Computing Applications in Medicine and Physics
7 VI 2024, Warsaw



Thanks to Piotr Gawron for the figure!



Preparation

Evolution

Measurement

state parameters

evolution parameters

$|\psi\rangle$

U

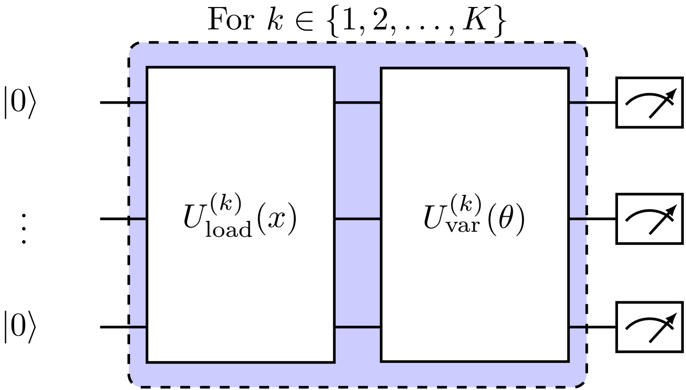
P_0, P_1

Quantum device

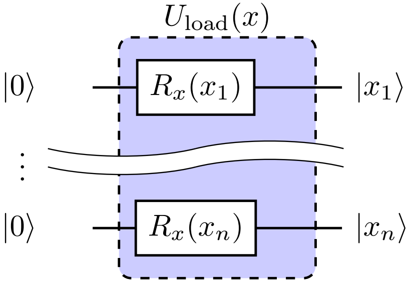
Classical world

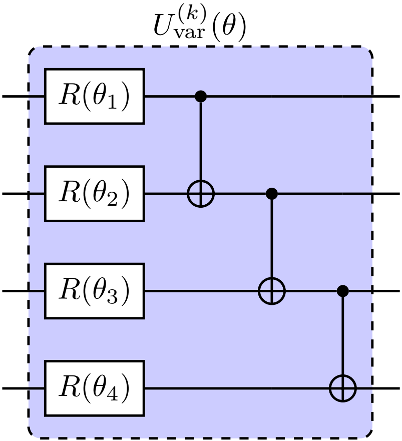
measurement results

Thanks to Piotr Gawron for the figure!


For $k \in \{1, 2, \ldots, K\}$
$|0\rangle$
$\vdots$
$|0\rangle$
$U_{\text{load}}^{(k)}(x)$
$U_{\text{var}}^{(k)}(\theta)$

Thanks to Piotr Gawron for the figure!





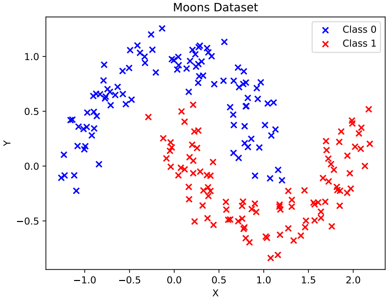
Thanks to Piotr Gawron for the figures!

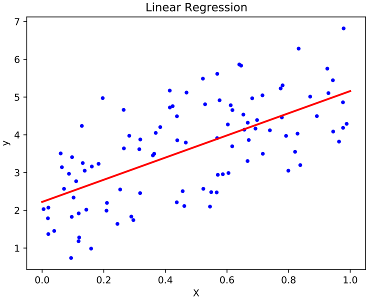
AQMLator: Auto QML e-platform

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Great comic strip by xkcd. (https://xkcd.com/license.html)

- Black-box approach for quantum ansatze search and learning.
- Open source.
- **Requires no quantum knowledge**.
- Can be used with real quantum devices (IBM, D-Wave, **IQM**, ...).
- Generates PyTorch models that can be integrated with existing pipelines.
- sklearn model mixins!



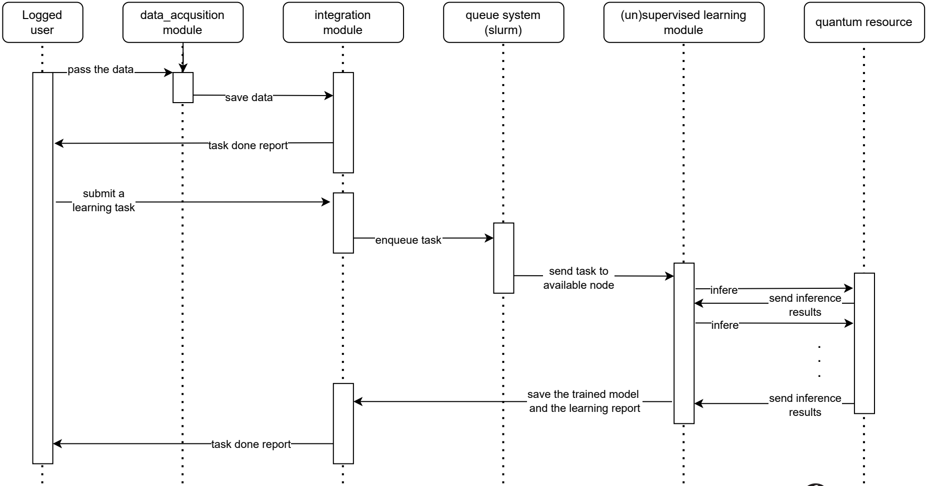


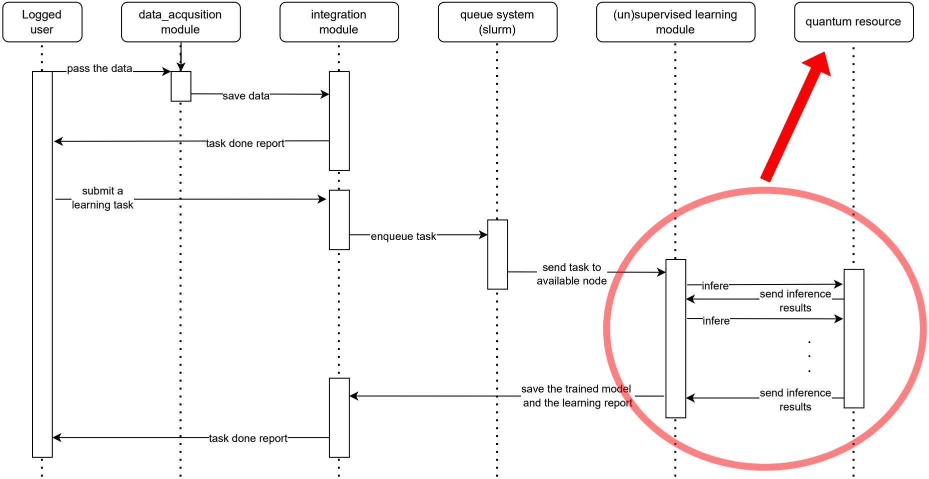
- Classification
- Clustering (RBM Quantum Training)[1]
- Multi-label Classification*

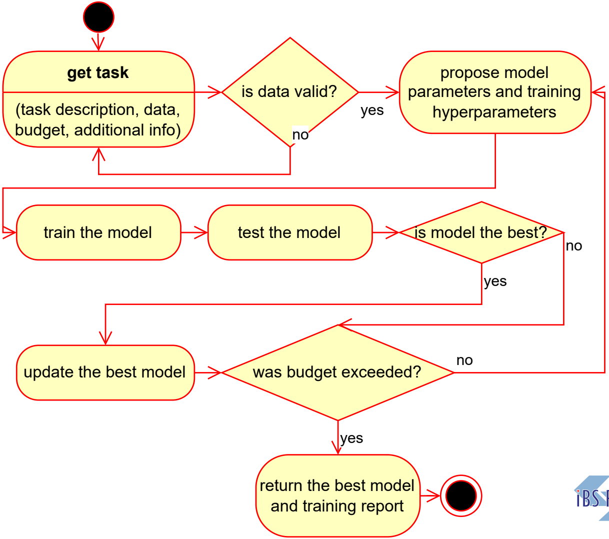
- Linear Regression

[1] Thanks to the QBM4EO project team!

* After some tweaking.


Logged user
data_acqusition module
integration module
queue system (slurm)
(un)supervised learning module
quantum resource
pass the data
save data
task done report
submit a learning task
enqueue task
send task to available node
infere
send inference results
infere
save the trained model and the learning report
send inference results
task done report


Logged user
data_acqusition module
integration module
queue system (slurm)
(un)supervised learning module
quantum resource
pass the data
save data
task done report
submit a learning task
enqueue task
send task to available node
infere
send inference results
infere
save the trained model and the learning report
send inference results
task done report



get task

(task description, data, budget, additional info)

is data valid?

yes

no

propose model parameters and training hyperparameters

train the model

test the model

is model the best?

no

yes

update the best model

was budget exceeded?

no

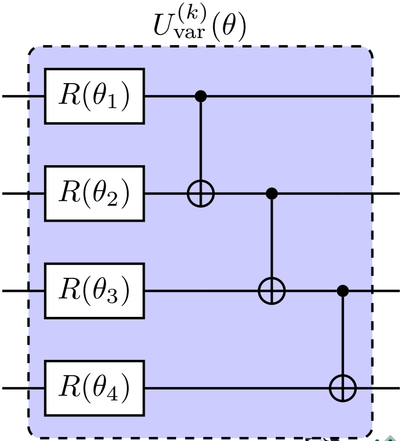
yes

return the best model and training report

How does it work?

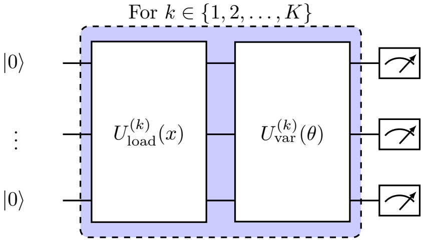
VQC Layers as building blocks

- Number of layers have been proposed over the years.
- There are layers known to perform well.
- Well structured, hence easy to handle automatically.
- Customizable and flexible!



Thanks to Piotr Gawron for the figure!

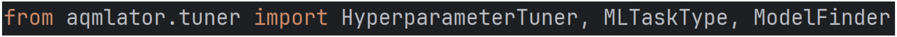
Layers as model hyperparameters


For $k \in \{1, 2, \ldots, K\}$
$|0\rangle$
$\vdots$
$|0\rangle$
$U_{\text{load}}^{(k)}(x)$
$U_{\text{var}}^{(k)}(\theta)$

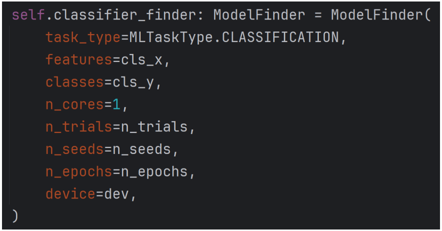
- Multiple approaches and libraries for hyperparameters optimization.
- Optuna library for ansatze finding and other hyperparameters optimization.
- Access to Optuna-adjacent tools (like optuna dashboard).
- Parallel processing.

Thanks to Piotr Gawron for the figure!

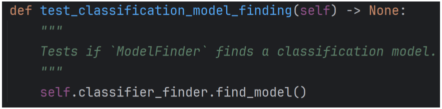
Code



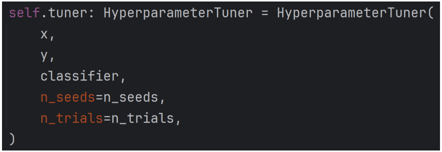
```
from aqmlator.tuner import HyperparameterTuner, MLTaskType, ModelFinder
```



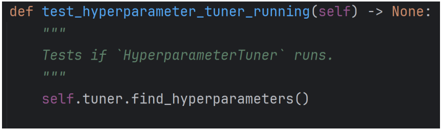
```
self.classifier_finder: ModelFinder = ModelFinder(
    task_type=MLTaskType.CLASSIFICATION,
    features=cls_x,
    classes=cls_y,
    n_cores=1,
    n_trials=n_trials,
    n_seeds=n_seeds,
    n_epochs=n_epochs,
    device=dev,
)
```



```
def test_classification_model_finding(self) -> None:
    """
    Tests if `ModelFinder` finds a classification model.
    """
    self.classifier_finder.find_model()
```

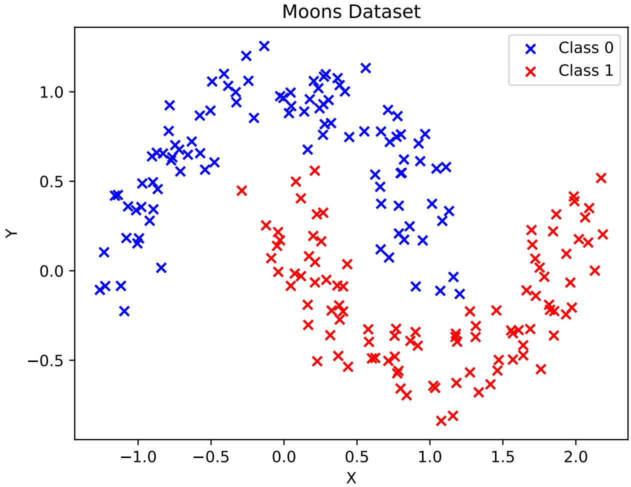


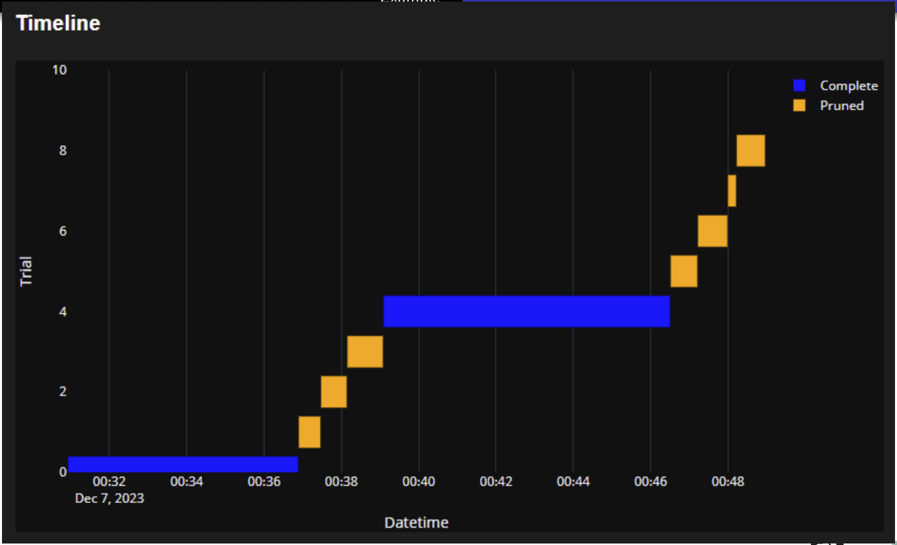
```
self.tuner: HyperparameterTuner = HyperparameterTuner(
    x,
    y,
    classifier,
    n_seeds=n_seeds,
    n_trials=n_trials,
)
```



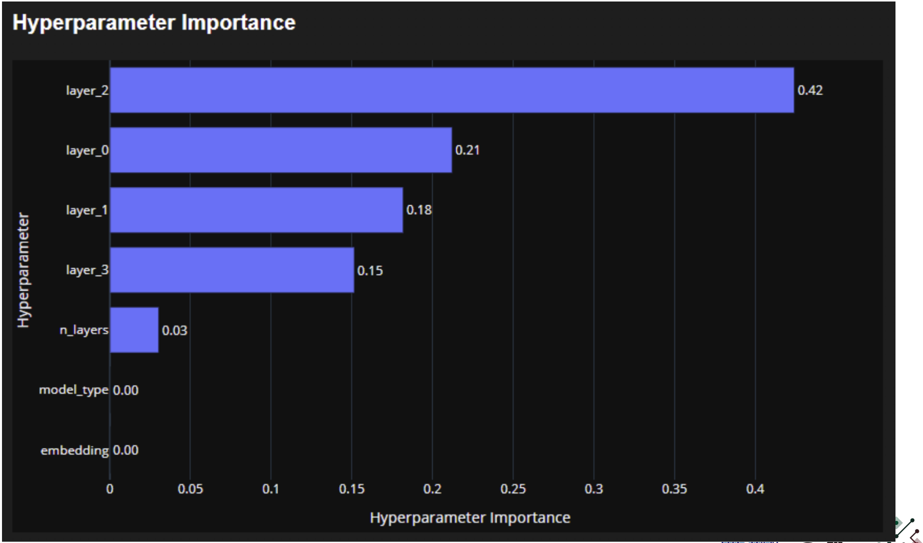
```
def test_hyperparameter_tuner_running(self) -> None:
    """
    Tests if `HyperparameterTuner` runs.
    """
    self.tuner.find_hyperparameters()
```

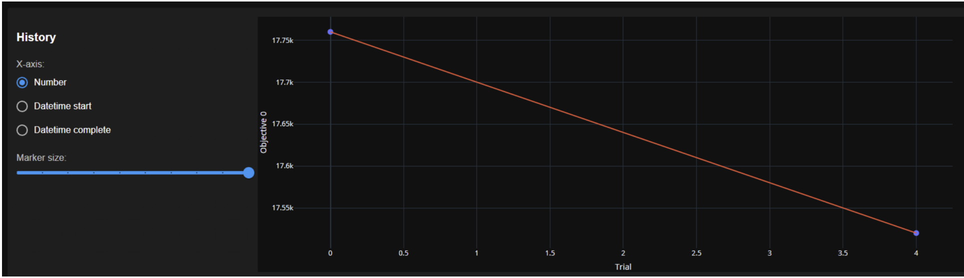
Binary Classification





Pruning threshold was accuracy lower than 0.85.


Hyperparameter Importance
Hyperparameter
layer_2
layer_0
layer_1
layer_3
n_layers
model_type
embedding
0.42
0.21
0.18
0.15
0.03
0.00
0.00
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Hyperparameter Importance



History

X-axis:

● Number

○ Datetime start

○ Datetime complete

Marker size:

Objective 0 is the number of quantum device calls.

Thank you for your attention!

Conclusions

- Open source, black-box, auto quantum machine learning framework.
- VQC architecture as model's hyperparameter.
- For ML practitioners with(out) quantum computing knowledge.
- Based on standard libraries (sklearn, PyTorch, Optuna, Pennylane).
- Usable with existing quantum devices.

Find me!

- rybotycki.tomasz@gmail.com
- https://twitter.com/TRybotycki
- https://orcid.org/0000-0003-2493-0459
- https://github.com/Tomev

AQMLator

- https://pypi.org/project/AQMLator/
- https://aqmlator.readthedocs.io/en/latest/