Applying IBM quantum computing to LHC physics analysis Higgs coupling to two top quarks (progress report)

Speaker: Wen Guan

Jay Chan, <u>Wen Guan</u>, Shaojun Sun, Alex Wang, Sau Lan Wu, Chen Zhou
University of Wisconsin-Madison
and
Federico Carminati
Chief Innovation Officer, CERN Openlab
and
Panagiotis Barkoutsos, Ivano Tavernelli, Stefan Woerner, Christa Zoufal
IBM Research Zurich

January 23-24, 2019
CERN Openlab workshop

Machine learning and quantum computing

- Machine Learning has become one of the most popular and powerful techniques and tools for HEP data analysis
- **Machine Learning: This is the field that gives computers "the ability to learn without explicitly programming them".**
- Issues raised by ML
  - Heavy CPU time is needed to train complex models
    - With the size of more data, the training time increases very quickly
  - May lead to local optimization, instead of global optimization
- Quantum computing
  - **A way of parallel execution of multiple processes using Qubits**
  - **Can speed up certain types of problems effectively**
  - **It is possible that quantum computing can find a different, and perhaps better, way to achieve global optimization.**

Ref: "Global Optimization Inspired by Quantum Physics", 10.1007/978-3-642-38703-6_41

Our program with IBM Qiskit

Our Goal:

Perform LHC High Energy Physics analysis with Quantum computing

Our preliminary program is to:

Employing SVM Quantum Variational (QSVM) method for LHC High Energy Physics (HEP) analysis with the environment of IBM Qiskit, for example ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis.

*** SVM = Support Vector Machine**

Our program with IBM Qiskit

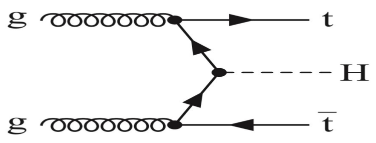
Our preliminary program can be divided into three parts with the Environment of IBM Qiskit:

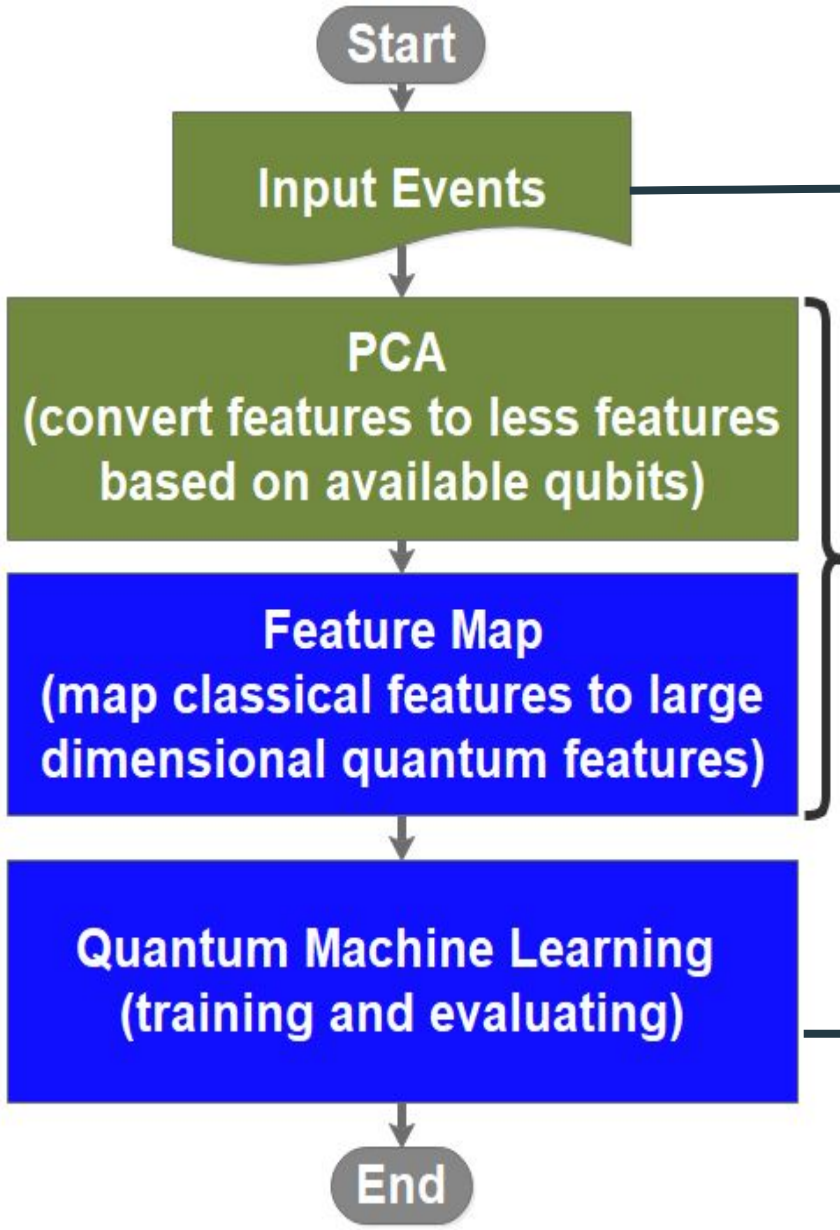
Part 1. Our workflow for quantum machine learning process.

Part 2. Employing the quantum method for LHC High Energy Physics (HEP) analysis, with quantum simulators, for example IBM Qiskit qasm simulator.

Part 3. Employing the quantum method for LHC High Energy Physics (HEP) analysis, with IBM quantum hardware, for example IBM Q Experience hardware.

Part 1: Our Workflow for Quantum Machine Learning process




Start
Input Events
PCA
(convert features to less features based on available qubits)
Feature Map
(map classical features to large dimensional quantum features)
Quantum Machine Learning
(training and evaluating)
End

ttH: number of features(variables) = 45

- **FeatureMap: Each feature(variable) of input event is encoded in the amplitude of one separate qubit, but we have much more features for an event than available qubits**
- **(Number of qubits = 5, 10, 20 for example)**
- **PCA: Principal Component Analysis method is used to convert/combine features to less features to be able to be encoded into quantum system.**

Support Vector Machine (SVM) quantum variational, for example

Part 2: Employing QSVM Variational with Q simulators

- **Employing SVM Quantum Variational for LHC HEP analysis**
  - **For example, ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis**
  - **Exploring different feature maps and entanglement methods**
  - **Training and evaluating quantum ML methods with different numbers of qubits, different number of events, different parameters and optimizers**

Part 2: Employing QSVM Variational with Q simulators

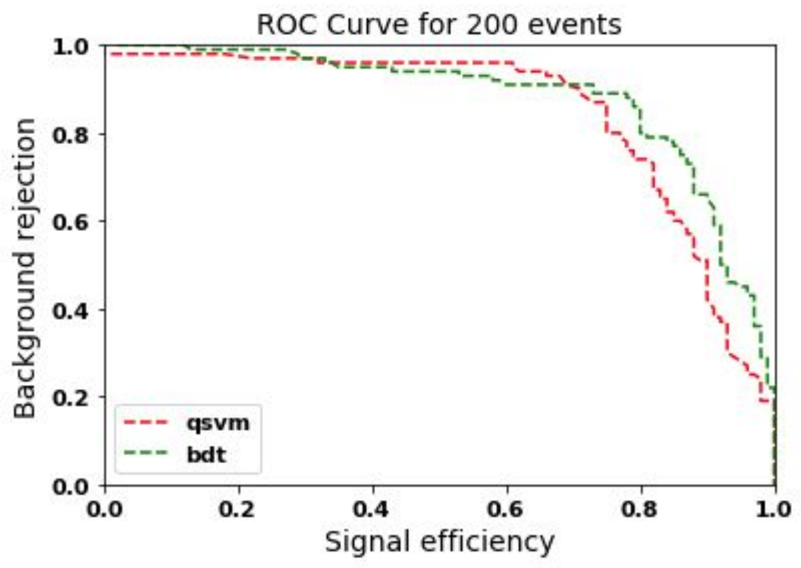
- With 5 qubits, we successfully finished training and testing with 200 events, 800 events and 3200 events with IBM Qiskit qasm simulator (where '200' events means 200 training events and 200 test events; same for others).

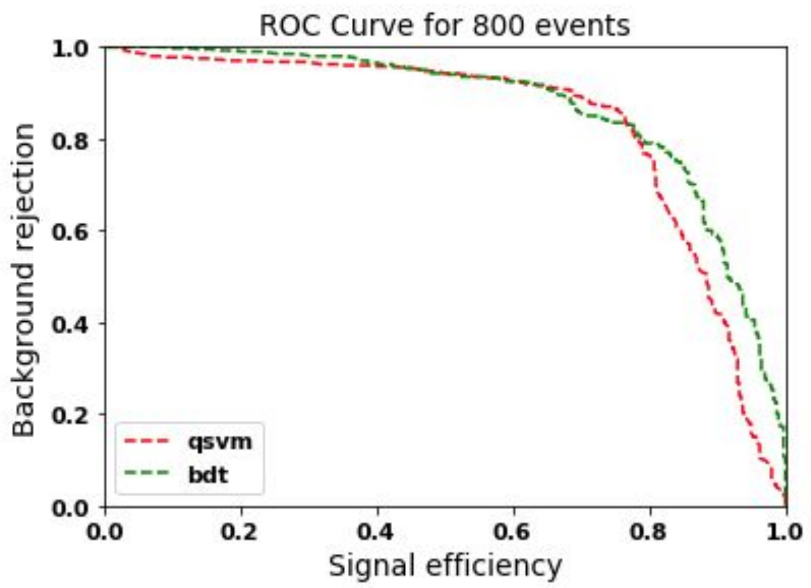
ttH(H->$\gamma\gamma$) accuracy	200	800	3200
QSVM	0.775	0.798	0.774
BDT	0.810	0.796	0.781

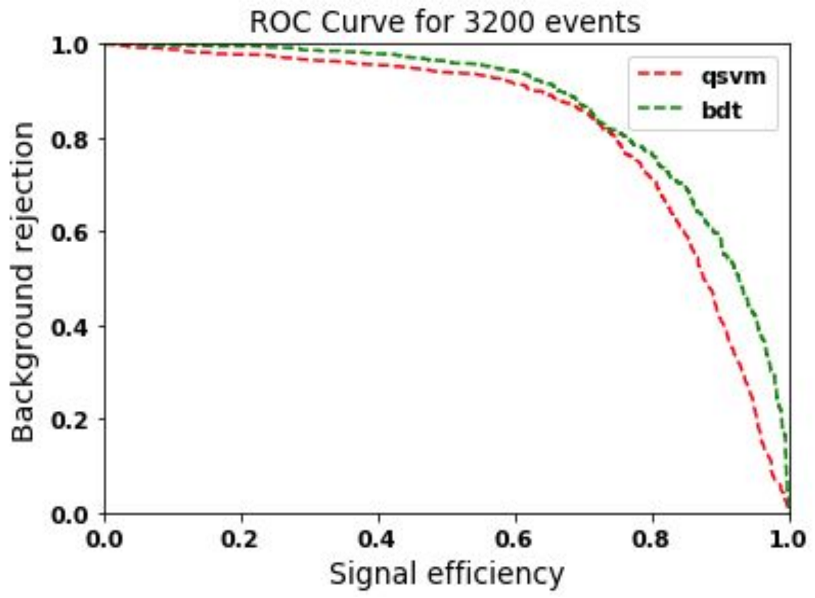
ttH(H->$\gamma\gamma$) auc	200	800	3200
QSVM	0.849	0.834	0.826
BDT	0.880	0.867	0.869

- For QSVM, SPSA optimizer is used with 3000 iterations.
- BDT(Boosted Decision Tree) method is using XGBoost, a classical method.
- BDT and QSVM are using exactly the same inputs for comparison.
- AUC: Area Under the ROC Curve(https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc).

Part 2: Employing QSVM Variational with Q simulators

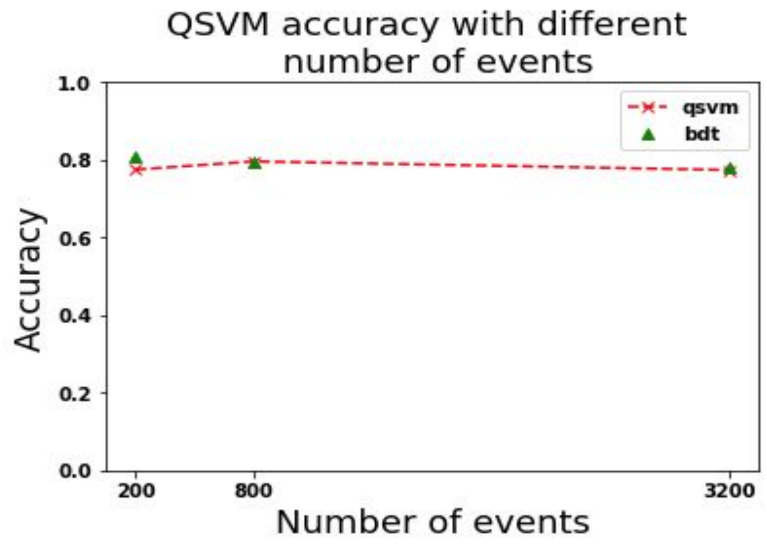


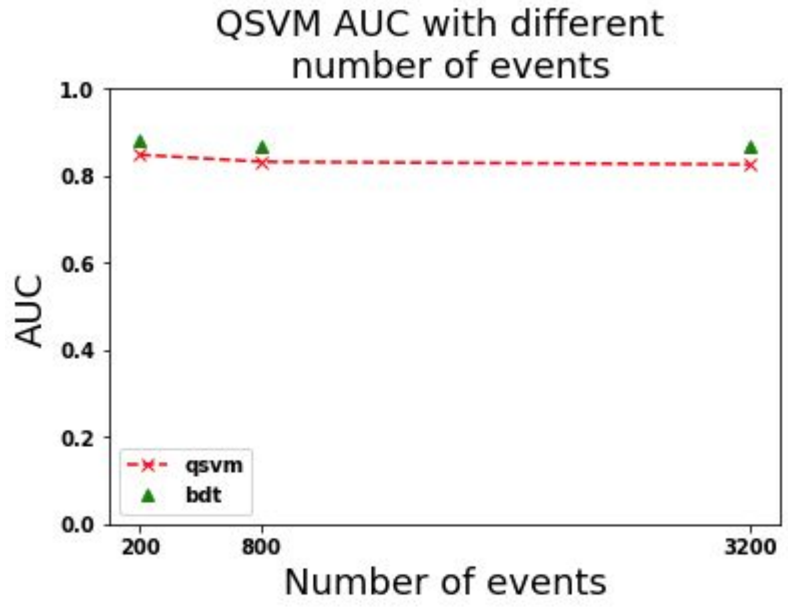




- **Here are the ROC Curve plots with QSVM and BDT, for 200 events, 800 events and 3200 events.**
  - **QSVM and BDT has very close ROC Curves.**
  - **BDT ROC Curve is a little better**

Part 2: Employing QSVM Variational with Q simulators

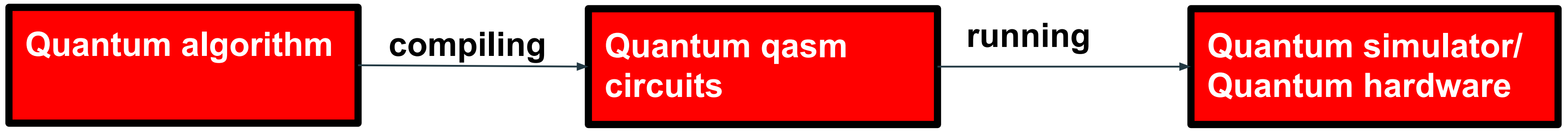




- Here are the accuracy and auc between QSVM and BDT, with 200 events, 800 events and 3200 events.
  - QSVM method got similar accuracy as BDT.
  - BDT got a litter better auc.
    - QSVM training is based on accuracy, not considering auc currently.
  - Note: With less than 3k events, the result varies when selecting different bunches of events. Here running the same bunch of 200 or 800 events is just for method comparing, not real analysis.

Part 2: Employing QSVM Variational with Q simulators

- Quantum algorithm running flow, for example IBM Qiskit



- Quantum algorithm can be prepared with python, with circuits defined with python too.
- Compiling process will compile the python codes to a dictionary with qasm codes(json serializable)
- Quantum simulator or quantum hardware will then execute the qasm codes

* Qasm = Quantum assembly language

Part 2: Employing QSVM Variational with Q simulators

- To simulate 200 events with 5 qubits, 10 qubits and 20 qubits, here are the memory usage.

200 events	5 qubits	10 qubits	20 qubits
Compiling	446 M	4.8 G	21 G
Simulation	64 M	0.98 G	4.5 G

- The python compiling process consumed much more memory than the quantum simulator.
- Here the compiling process is using only one process
  - If more than one compiling process, more memory will be used.
- With more events, the compiling process will use more memory.

10 qubits	200 events	800 events
Compiling	4.8G	12.7G
Simulation	0.98G	3.1G

Part 2: Employing QSVM Variational with Q simulators

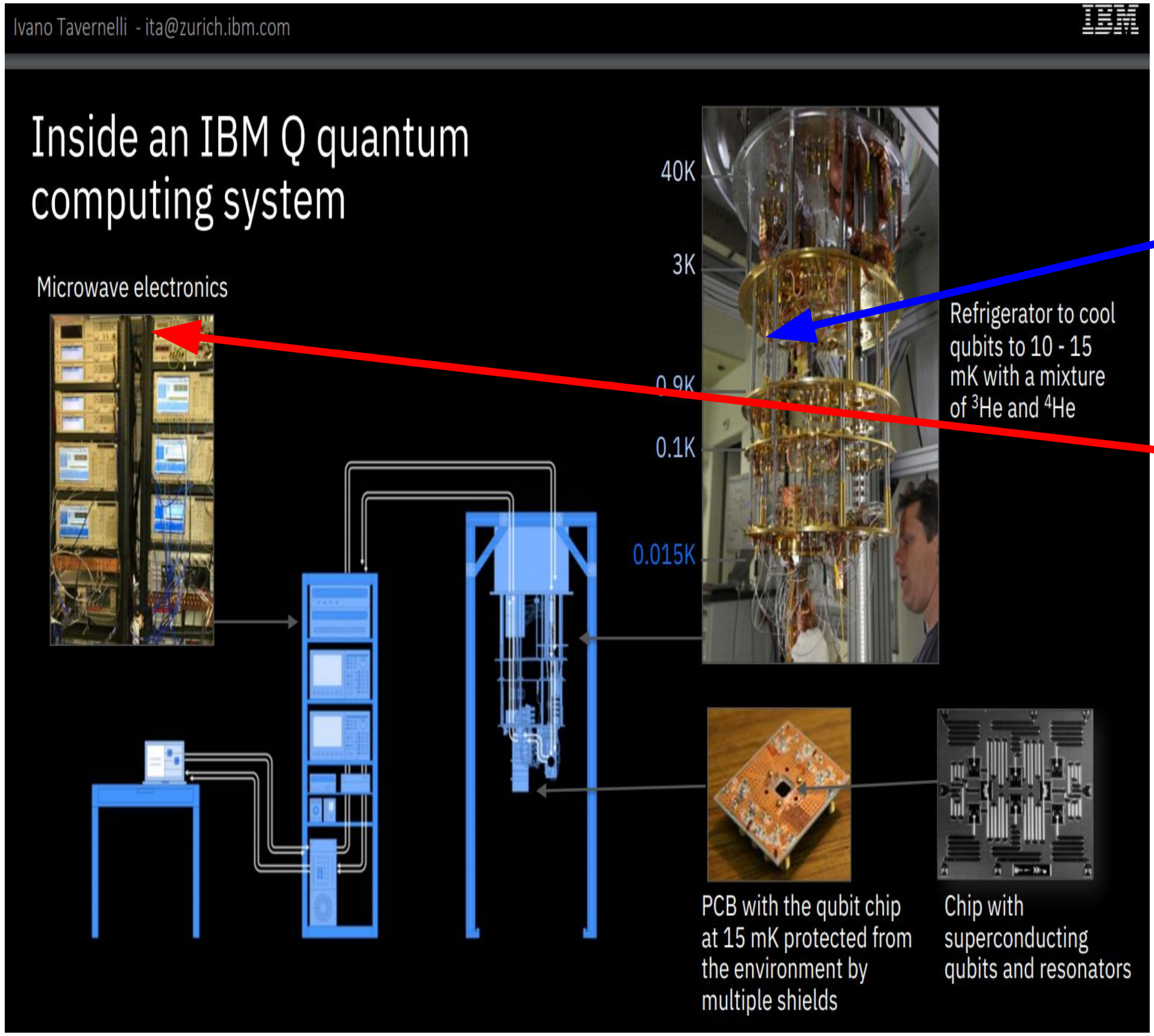
- To simulate 200 events with 5 qubits, 10 qubits and 20 qubits, here are the time consumption.

200 events	5 qubits	10 qubits	20 qubits
Compiling time(seconds)	6	87	403
Simulation time(seconds)	2	27	3334

- Quantum compiling time is also a big part of the total running time.
- Simulation CPU time increase exponentially O(2^n), where n is the number of qubits. That’s why it takes much longer time to simulate 20 qubits.
- Here the simulation process is using one process too, with qasm simulator; The time is for one iteration
- With more events, the compiling time will increase too.

10 qubits	200 events	800 events
Compiling(seconds)	87	398
Simulation(seconds)	27	88

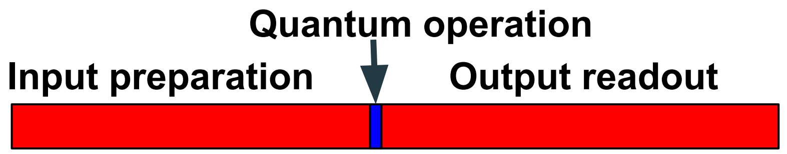
Part 3: Employing QSVM Variational with IBMQ hardware



Ivano Tavernelli - ita@zurich.ibm.com
IBM
Inside an IBM Q quantum computing system
Microwave electronics
40K
3K
0.9K
0.1K
0.015K
Refrigerator to cool qubits to 10 - 15 mK with a mixture of ³He and ⁴He
PCB with the qubit chip at 15 mK protected from the environment by multiple shields
Chip with superconducting qubits and resonators

“Quantum Computing at IBM”, Ivano Tavernelli, Quantum Computing for High Energy Physics, CERN-Geneva, November 5-6,2018

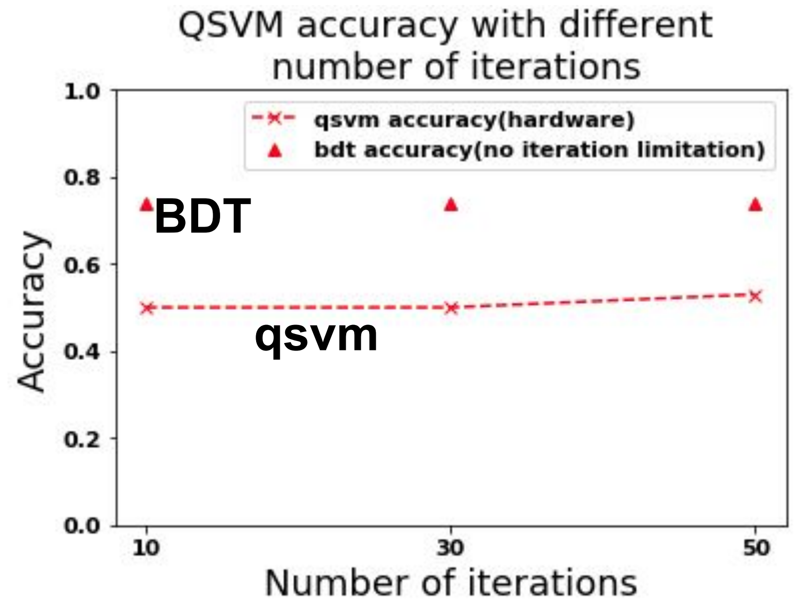
- **Quantum hardware limitation**
  - **Operations can reach 5MHz with Quantum chips(see backup slides 22)**
  - **But Quantum input preparation and output reading is not optimized yet (Microwave electronics). As a result, the total quantum execution time is not optimized.**

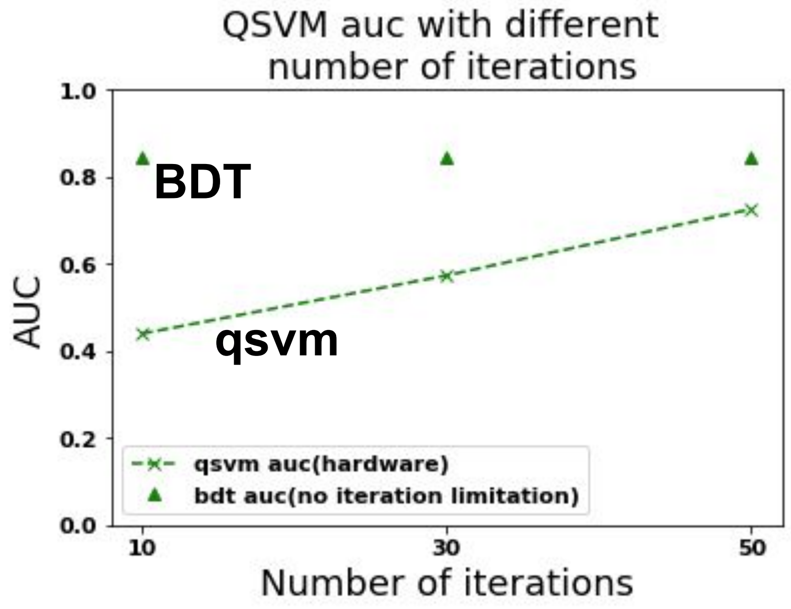


Part 3: Employing QSVM Variational with IBMQ hardware

- With the help of IBM Research Zurich, we finished some training on the IBMQ hardware with 100 training events and 100 test events, 5 qubits.
- Because of hardware access time and timeout limitation, we only finished very few iterations (for example 10,30,50) on the hardware, instead of several thousands of iterations on the simulators.

Part 3: Employing QSVM Variational with IBMQ hardware





- Here are the accuracy and auc plots with different number of iterations.
  - With increasing iterations, the qsvm accuracy increases slowly.
  - With increasing iterations, the qsvm auc increases very fast.
- We need to run much more iterations on the hardware if we want to get similar results like BDT.

Part 3: Employing QSVM Variational with IBMQ hardware

- **Limitation with IBMQ hardware**
  - **Only few iterations are tested currently**
    - **Limited access time**
      - **Long queue time**
  - **Input preparation and output reading is not optimized**

Summary

Referring to Part 1 of this presentation:

- We introduced our workflow to employ quantum methods for LHC High Energy Physics analysis.

Summary

Referring to Part 2 of this presentation:

- Using IBM Qiskit simulator, we have successfully employed Quantum Support Vector Machine method for ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis. We have measured the accuracy and auc with different number of events.
- At current stage, with 5 qubits, we reached similar accuracy of 0.79 and very close auc of 0.83, comparing with the classical machine learning method(BDT) with accuracy 0.79 and auc 0.87. It also means work is needed to improve QSVM algorithm.
- We also measured the memory usage and execution time between quantum compiling and quantum simulation. We found that quantum compiling is a big part for both memory usage and time consumption. With few qubits, quantum compiling consumes much more memory and takes much longer time than for the actual quantum simulation.

Summary

Referring to Part 3 of this presentation:

- **Using IBM Q Experience hardware, we have successfully employed Quantum Support Vector Machine method for ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis.**
- **Again, the accuracy and auc is limited by the iterations. But the accuracy and auc is increasing as increasing iterations.**

BACKUP SLIDES

Quantum measurement

- **Quantum state is a superposition which contains the probabilities of possible positions.**
- **When the final state is measured, they will only be found in one of the possible positions.**
  - **The quantum state ‘collapses’ to a classical state as a result of making the measurement.**
- **“No-cloning theorem”**
  - **Impossible to create an identical copy of an arbitrary unknown quantum state.**
- **To obtain the probability of a possible position, some number of shots are needed.**

Hardware Information

- **Hardware status currently**
  - **Classical computer:**
    - **3~4 GHz**
    - **Millions of circuits with many cores, GPU can have thousands of cores**
  - **Quantum computer**
    - **200 ns per operation**
    - **5M Hz**
    - **Not many parallel channels or threads**
    - https://quantumcomputing.stackexchange.com/questions/2402/how-many-operations-can-a-quantum-computer-perform-per-second

How to use quantum computer

- How to use quantum computers
    a. Convert classical features to be able to be processed to quantum computers
        - Feature map
    b. Using quantum algorithms to process the data
        - Algorithms developed based on quantum computers, such as Quantum Support Vector Machine, Quantum annealing, Grover Search and so on

Tensor product feature map

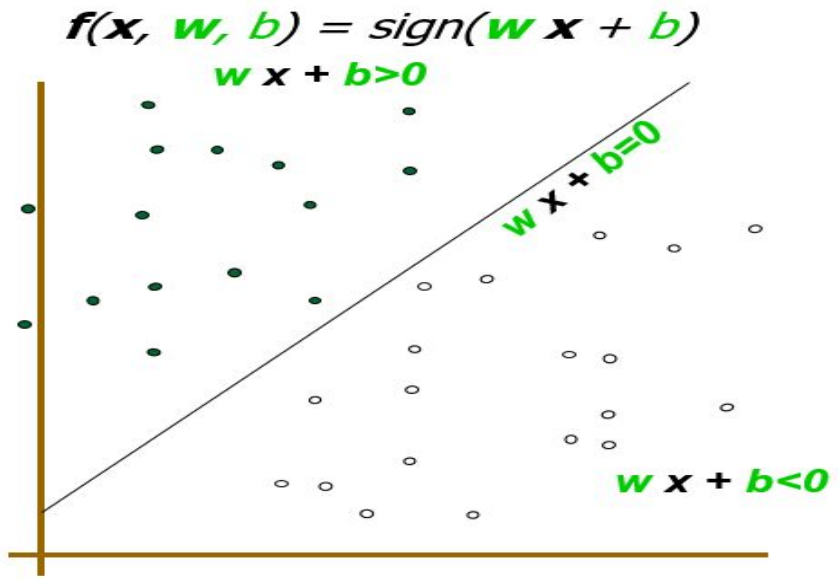
- Quantum feature map: Map bit info non-linearly to quantum ‘feature Hilbert space’
  - Tensor product encoding
    - Each feature(variable) of input event is encoded in the amplitude of one separate qubit
    - All features of one event is the tensor product of corresponding qubits
  - Entanglement between features
    - Without entanglement
    - Between next one feature(linear entanglement)
    - Between all of the next features(full entanglement)

Other feature map methods

- Basic encoding
  - One bit maps to one qubit
  - For example, two bits "01" maps to two qubits "|01>"
- Amplitude encoding
  - N classical features maps to $\log_2 N$ qubits
  - $X = (x_0, \ldots, x_{N-1})$, $N=2^n$
  - $|\varphi_x\rangle = \Sigma \; X_i * |i\rangle$ (qubit "|i>" is the i'th computational basis state)
  - Looking whether it's possible and how to do it

Support Vector Machine

- Support Vector Machine (SVM)
  - a supervised ML that draws a decision boundary between two classes to classify data points
  - Originally it's constructed as a linear classifier
  - Maximize the distance from the line or hyperplane to the nearest data point on each side

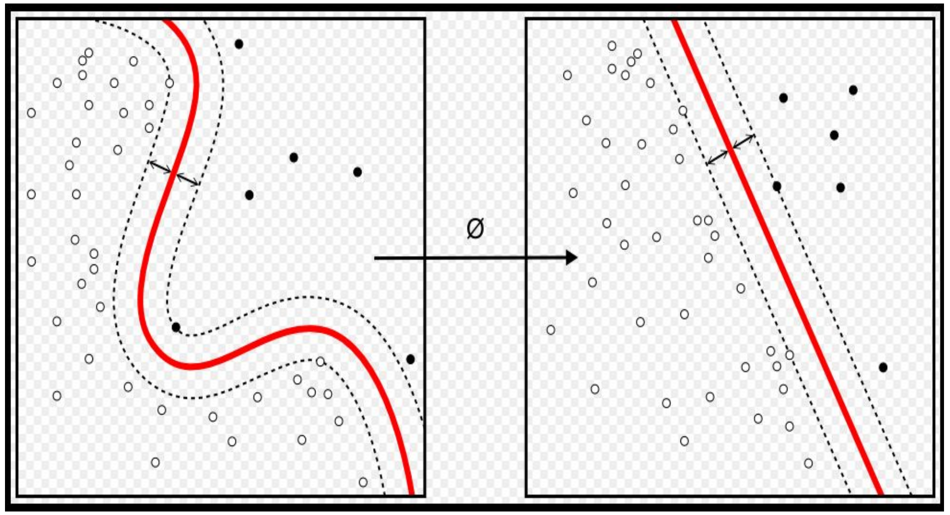


Ref: Support Vector Machine and Its Application(Mingyue Tan, 2004)

Ref: Support vector machine(Wikipedia)

SVM kernel function

- **Kernel function**
  - Often the sets of data points are not linearly separable
  - **Map data points to a much higher dimensional space which presumably making the separation easier**



Ref: Support vector machine(Wikipedia)

- Performance depends on different kernel functions
- Limitation to successful solutions when feature space becomes large
- Computationally expensive to estimate the kernel

Quantum SVM

- Quantum SVM
  - Take advantage of the large dimensionality of quantum Hilbert space
    - Non-linearly maps input data into a very large dimensional feature Hilbert space
    - **Exploiting an exponentially large quantum state space**
  - Take advantage of the quantum speedup
  - Estimate the kernel and optimize the classifier