# Application of IBM Quantum Computing on LHC High Energy Physics data analysis

**Speaker: Wen Guan**

**Jay Chan, Wen Guan, Shaojun Sun, Alex Wang, Sau Lan Wu, Chen Zhou**
**Physics Department, University of Wisconsin-Madison**
**and**
**Miron Livny**
**Computing Sciences Department and Wisconsin Institute for Discovery, University of Wisconsin-Madison**
**and**
**Alberto Di Meglio, Federico Carminati**
**CERN Openlab, IT Department, CERN**
**and**
**Panagiotis Barkoutsos, Ivano Tavernelli, Stefan Woerner, Christa Zoufal**
**IBM Research Zurich**

**June 19, 2019, CERN**
**CERN Graph Net::work::shop**

# 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 machine learning
  - 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 perform machine learning.

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

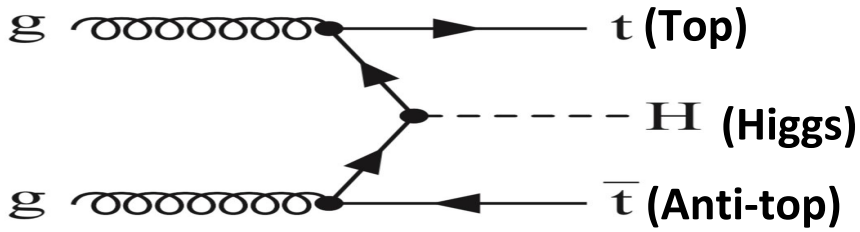
# Our program with IBM Qiskit

**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 → $\gamma\gamma$), Higgs coupling to two top quarks analysis.**




* SVM = Support Vector Machine
* IBM Qiskit = IBM Quantum Information Science Kit

# Our program with IBM Qiskit

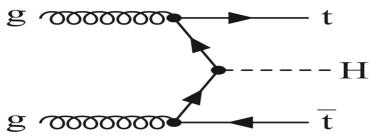
Our preliminary program can be divided into three parts with the Environment of IBM Qiskit:

**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

ttH: number of features(variables) = 45

PCA
(convert features to less features based on available qubits)

Feature Map
(map classical features to large dimensional quantum features)

- FeatureMap: Each feature(variable) of input event is encoded in the amplitude of one separate qubit, but we have many 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 match the number of qubits.

Quantum Machine Learning
(training and evaluating)

Support Vector Machine (SVM) quantum variational, for example

End

# 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 machine learning methods with different numbers of qubits, different number of events, different parameters and optimizers**

# Part 2: Employing QSVM Variational with Q simulators

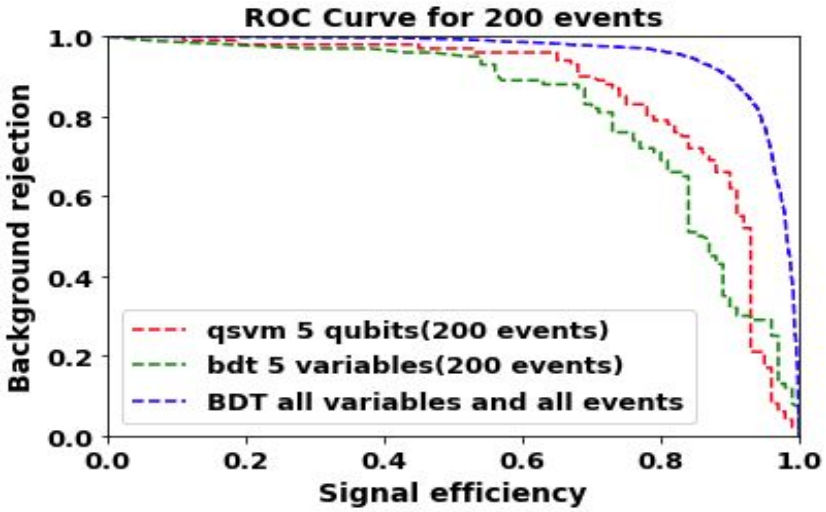
- With 5 qubits, we successfully finished training and testing with 200 events, 800 events and 3200 events with IBM Qiskit qasm simulator (where '200' events means 200 training events and 200 test events; same for others).
  - For QSVM, SPSA optimizer is used with 3000 iterations.
  - BDT and QSVM are using exactly the same inputs for comparison.
  - Q simulator: Here Qiskit Qasm simulator is used.

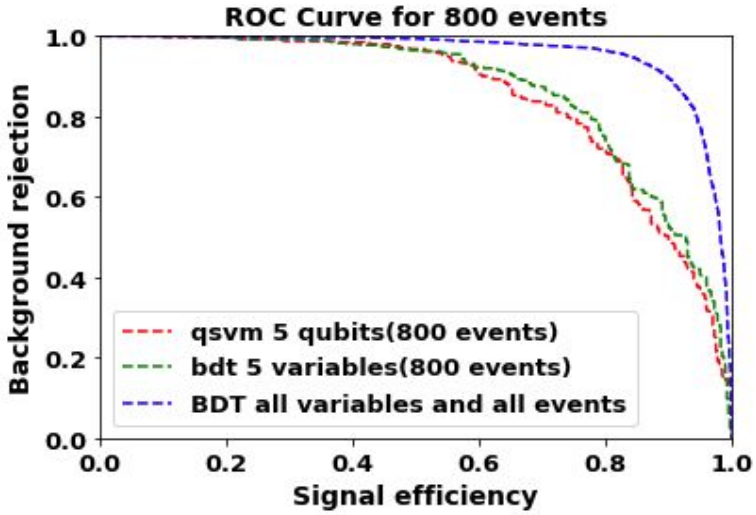
| ttH(H->$\gamma\gamma$) ACCURACY | 200 | 800 | 3200 |
|---|---|---|---|
| QSVM | 0.795 | 0.802 | 0.768 |
| BDT | 0.75 | 0.785 | 0.780 |

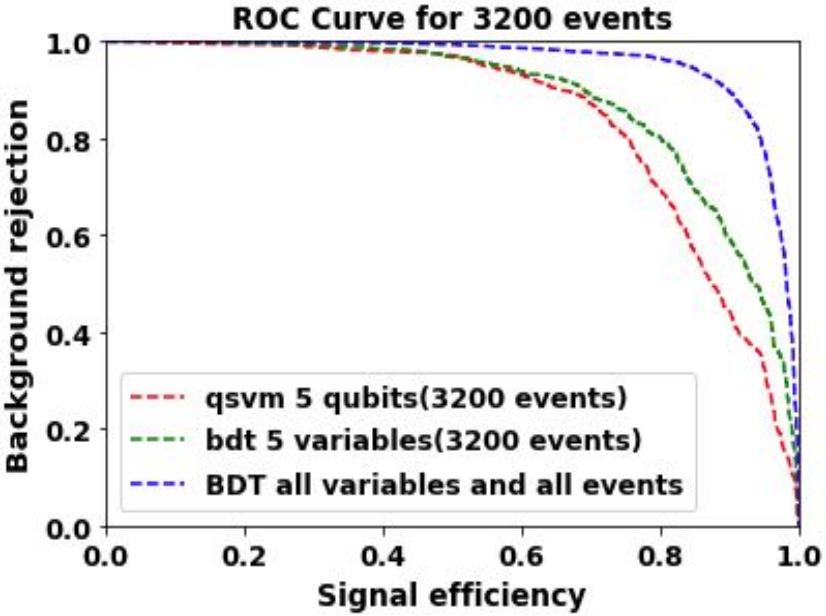
| ttH(H->$\gamma\gamma$) AUC | 200 | 800 | 3200 |
|---|---|---|---|
| QSVM | 0.865 | 0.859 | 0.837 |
| BDT | 0.821 | 0.869 | 0.863 |

# Part 2: Employing QSVM Variational with Q simulators

- Here BDT(green) and QSVM(red) are using exactly the same inputs for comparison.



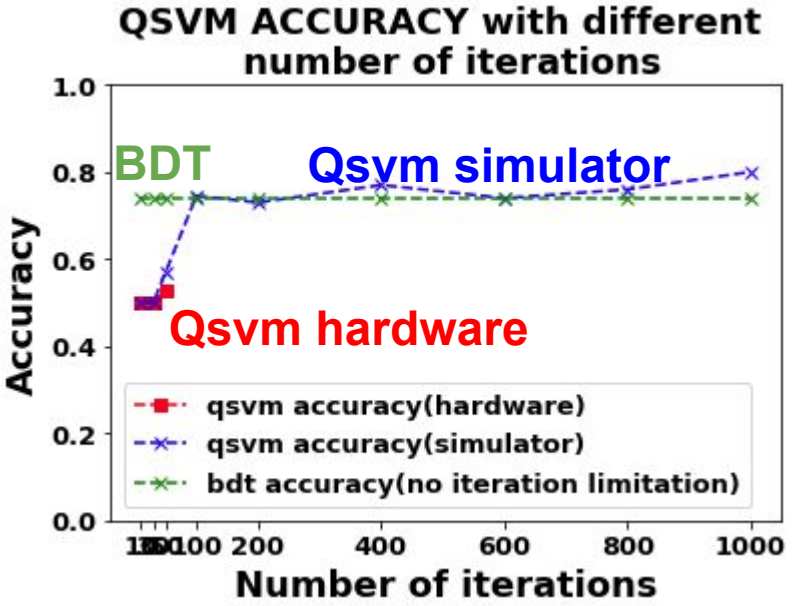



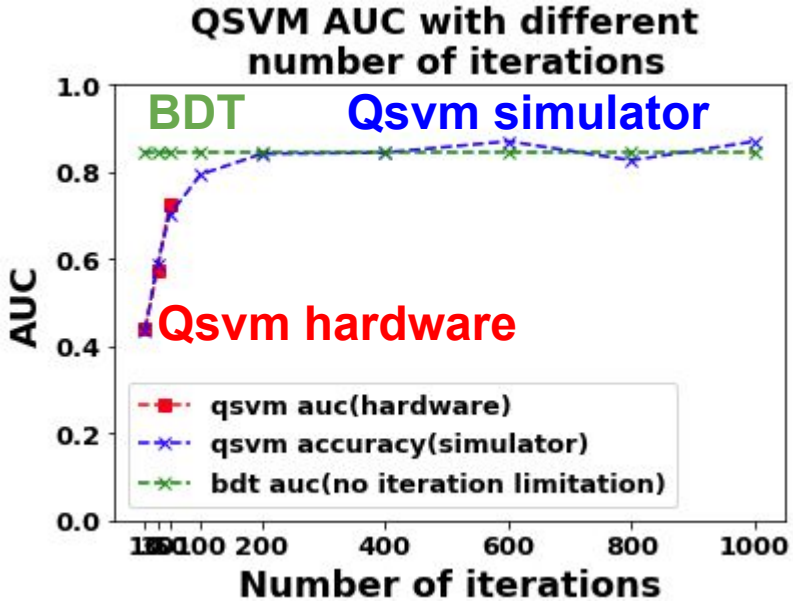

- Here are the ROC Curve plots with QSVM(red) and BDT(green), for 200 events, 800 events and 3200 events, with 5 qubits.
- ROC curve(Blue): classical machine learning BDT with all 45 variables and all simulated events in LHC ttH analysis.
- ROC curve(Red): QSVM 5 qubits
- ROC curve(Green): classical machine learning BDT with the same inputs of 5 variables per event as QSVM for comparison.

# 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



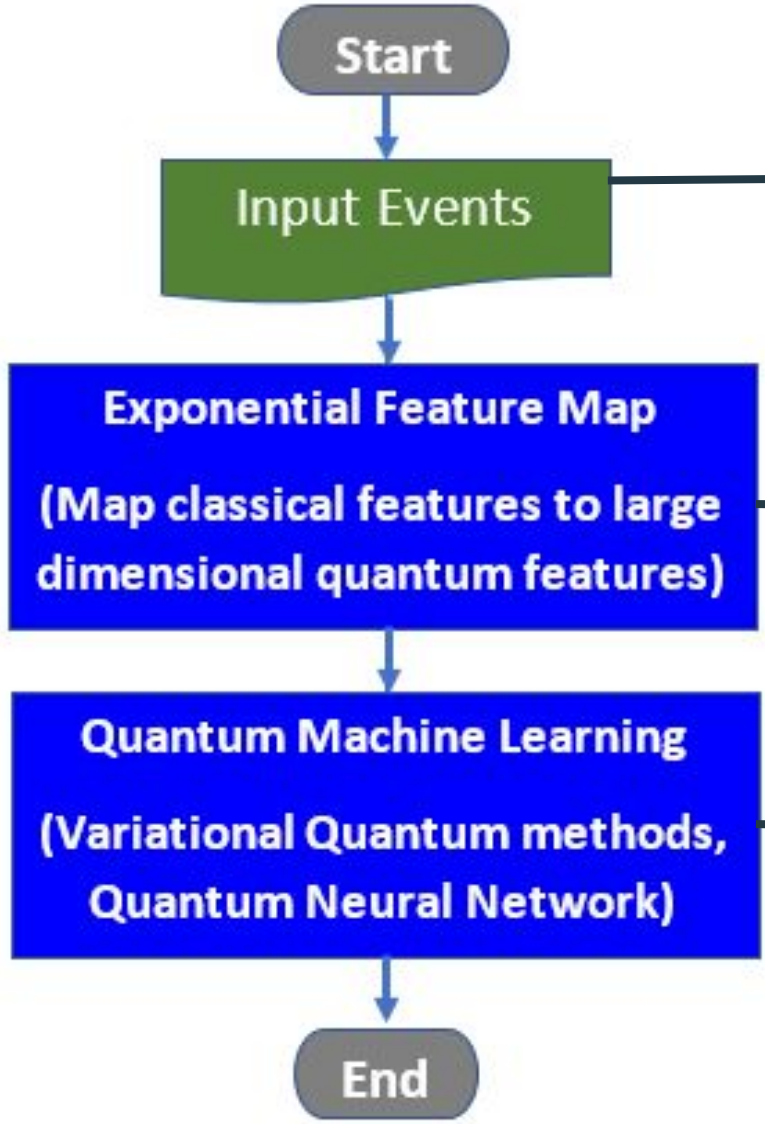




- Here are the ACCURACY and AUC plots with different number of iterations.
  - Because of access time limitation, on the hardware we only finished 10,30 and 50 iterations.
- Within limited iterations, the result from hardware(red) is compatible with the result from simulator(blue) in tested iterations.
- The result from simulator(blue) reached similar result with classical BDT(green) method with enough iterations.

# 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**
  - **working to extend it to more iterations**

# New: New Workflow for Quantum Machine Learning process




ttH: number of features(variables) = 45

- FeatureMap: With N qubits to encode $2^N$ variables
- Without PCA

- Support Vector Machine (SVM) quantum variational, to improve the variational training circuits
- Quantum Neural Networks

## Summary

- We have successfully applied variational quantum ML method Variatiaonal QSVM on hep analysis and have achieved close performance, comparing classical ML method BDT.
- We have also applied our analysis on IBMQ hardware with limited iterations. We are working to extend it.
- We are also applying quantum neural networks for hep analysis. It's in progress.

# BACKUP SLIDES

# Quantum algorithm running flow

- Quantum algorithm running flow, for example IBM Qiskit


- Issues:
  - The quantum compiling process compiles codes and input data together, while classical compiling separates codes and input data.
    - With more data, the compiling process will use more time and more memory.
    - With different data, a new compiling is required.

* Qasm = Quantum assembly language

# 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

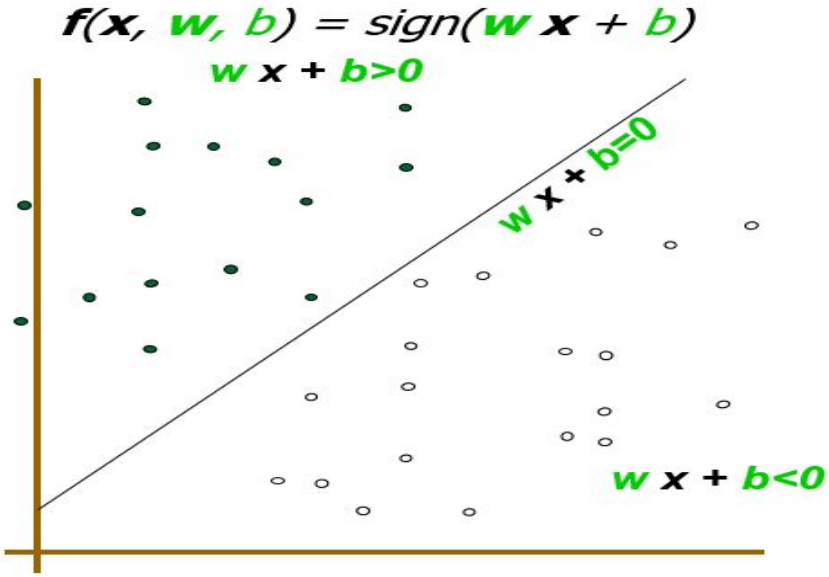
- Quantum feature map: Map bit info non-linearly to quantum 'feature Hilbert space'
  - Tensor product encoding
    - Each feature(variable) of input event is encoded in the amplitude of one separate qubit
    - All features of one event is the tensor product of corresponding qubits
  - Entanglement 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 "$|01\rangle$"
- 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 "$|i\rangle$" 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

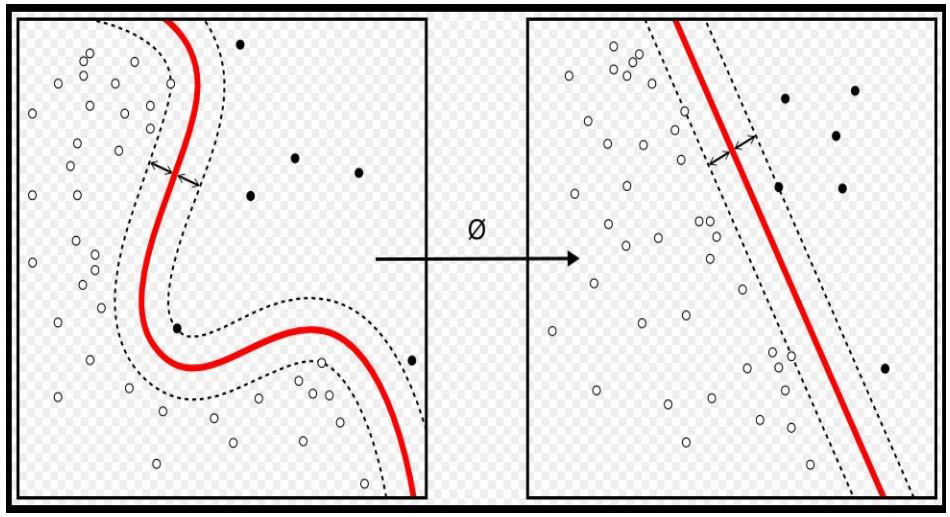



**Ref: Support Vector Machine and Its Application(Mingyue Tan, 2004)**

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