# Application of Quantum Machine Learning to High Energy Physics Analysis at LHC using IBM Quantum Computer Simulators and IBM Quantum Computer Hardware

**Speaker: Alex Wang**

**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**
**Federico Carminati, Alberto Di Meglio**
**CERN Openlab, IT Department, CERN**
**and**
**Panagiotis Barkoutsos, Ivano Tavernelli, Stefan Woerner, Christa Zoufal**
**IBM Research Zurich**


**July 29 - August 02, 2019 - Boston, USA**
**Meeting of the Division of Particles & Fields of the American Physical Society**

# 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

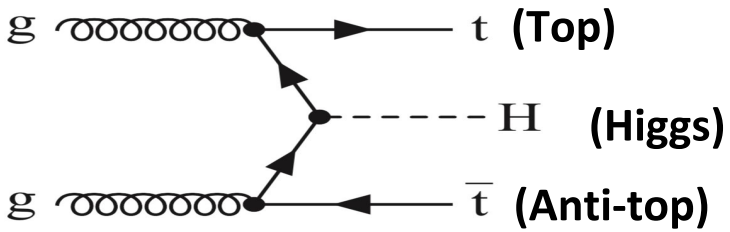
# Our program with IBM Qiskit

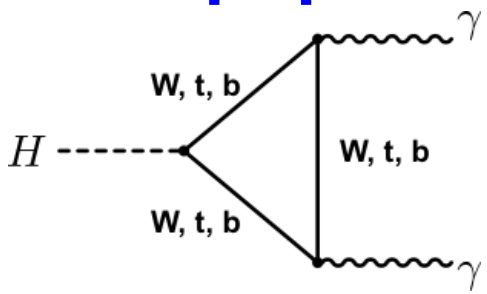
**Our Goal:**

**Perform LHC High Energy Physics analysis with Quantum computing**

**Our preliminary program is to:**

**Employ the 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 production in association with two top quarks analysis.**




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

# An example of classical machine learning: ttH ($H \rightarrow \gamma\gamma$) analysis by the ATLAS Collaboration

## (ttH: Higgs production in association with two top quarks)

*Phys. Lett. B 784 (2018) 173*

*ATLAS-CONF-2019-004*

Select events with **two photons**

➔Separate into a **hadronic channel** ($n_{lep}$ = 0) and a **leptonic channel** ($n_{lep}$ >= 1)

Background: continuum bkg. (γγ, etc.) and resonant bkg. from other Higgs production modes (ggH, etc.)

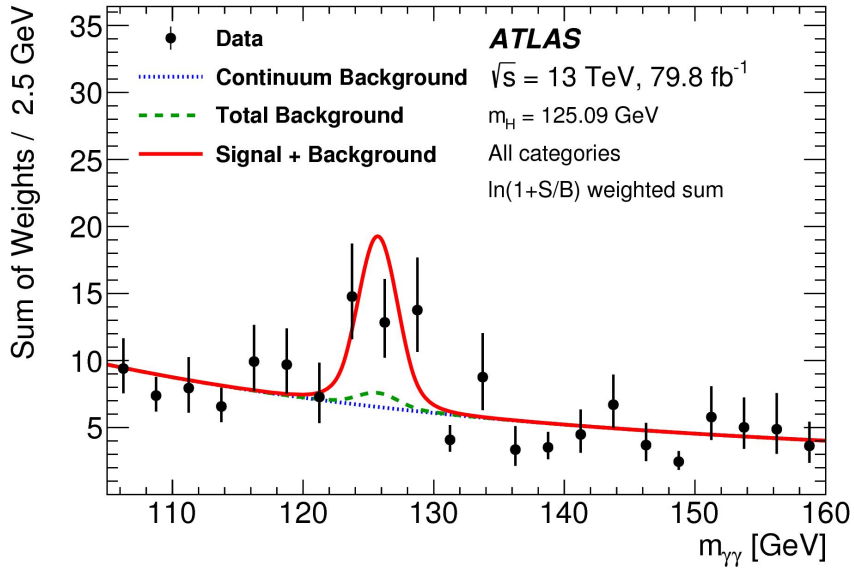
➔In each channel, train a **Boosted Decision Tree** (BDT, a classical machine learning technique) with XGBoost

create **categories based on BDT output**

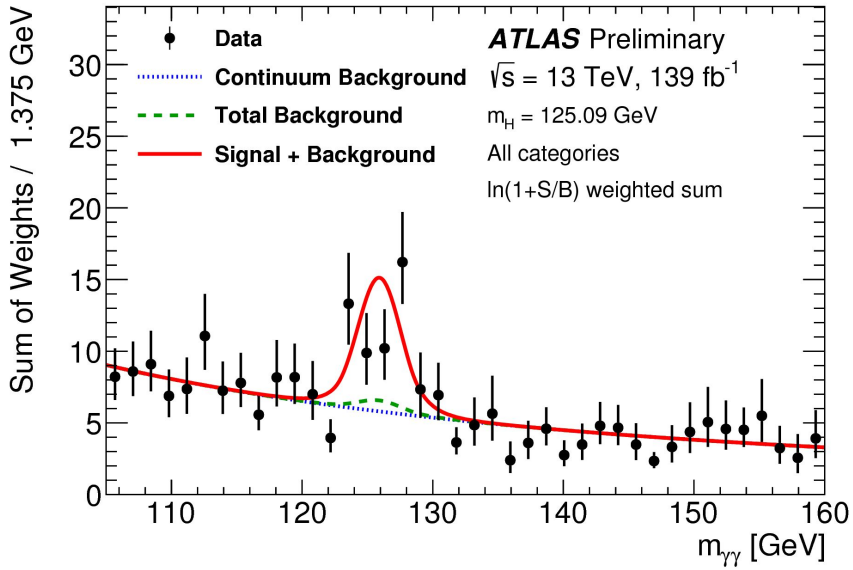
➔Fit **diphoton mass** over 7 categories

Measure ttH production signal strength, etc.

# An example of classical machine learning: ttH (H → $\gamma\gamma$) analysis by the ATLAS Collaboration




*Phys. Lett. B 784 (2018) 173*, **80 fb$^{-1}$**




*ATLAS-CONF-2019-004*, **139 fb$^{-1}$**

- The observed significance is 4.1σ (4.9σ) in the ATLAS ttH (H→γγ) analysis using 80 fb$^{-1}$ (139 fb$^{-1}$) of 13 TeV data
- This talk will show the machine learning step of the ATLAS ttH (H→γγ) analysis with Delphes simulation events **using quantum machine learning**

# Our program with IBM Qiskit

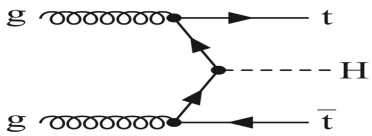
Our preliminary program can be divided into three parts with the Environment of IBM Qiskit:

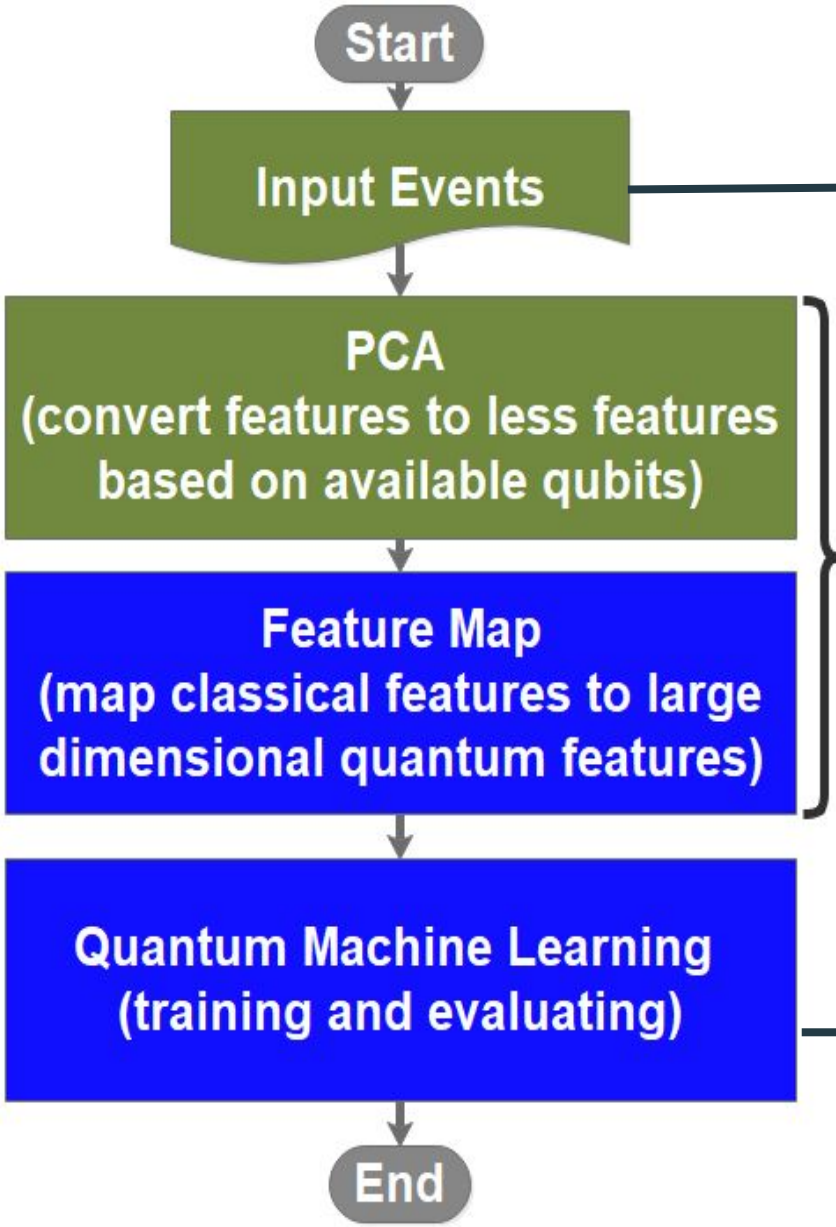
**Part 1.** Our workflow for quantum machine learning.

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

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

# Part 1: Our Workflow for Quantum Machine Learning




ttH: number of features(variables) = 45

- FeatureMap: Each feature(variable) of the 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 into fewer features to match the number of qubits.

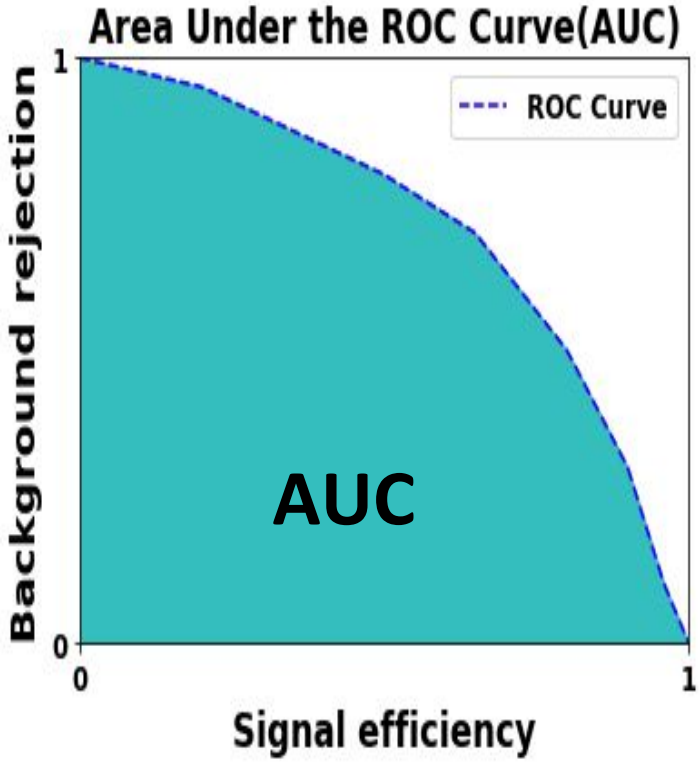
Support Vector Machine (SVM) quantum variational method, for example

# Part 2: Employing QSVM Variational with Q simulators

- **Employing SVM Quantum Variational for LHC HEP analyses**
  - **For example, a ttH ($H \rightarrow \gamma\gamma$), Higgs production in association with two top quarks analysis**
  - **Exploring different feature maps and entanglement methods**
  - **Training and evaluating quantum machine learning methods with different numbers of qubits, different numbers of events, different parameters and optimizers**

# Part 2: Employing QSVM Variational with Q simulators

- **Definitions**
  - A **BDT**(Boosted Decision Tree) is a classical machine learning method. Here we are using XGBoost.
  - **Q simulator:** Quantum circuits simulator, such as Qasm simulator.
  - **Accuracy**: The ratio of correct predictions to total predictions.
  - **ROC Curve**: a graph showing background rejection vs signal efficiency.
  - **AUC**: Area Under the ROC Curve




Ref: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

# Part 2: Employing QSVM Variational with Q simulators

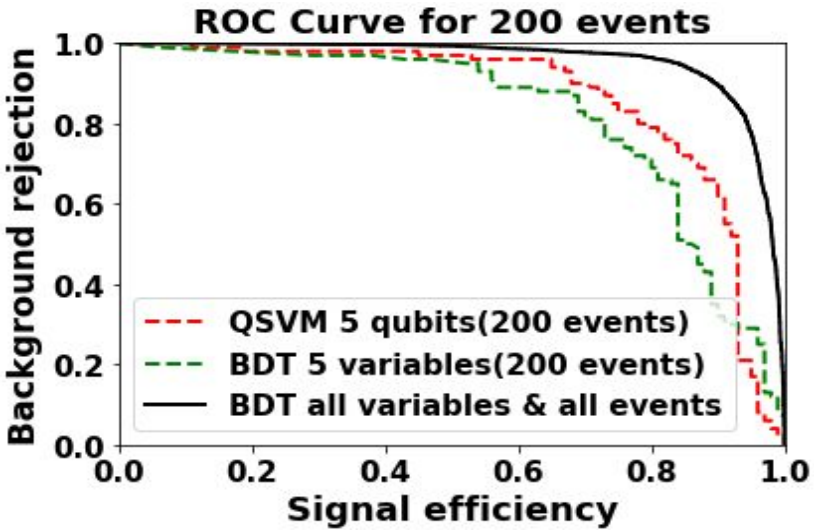
- With 5 qubits, we successfully finished training and testing with 200 events, 800 events and 3200 events with IBM Qiskit qasm simulator (where '200' events means 200 training events and 200 test events; same for others. Events are simulated with Delphes).
  - For QSVM, SPSA optimizer is used with 3000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

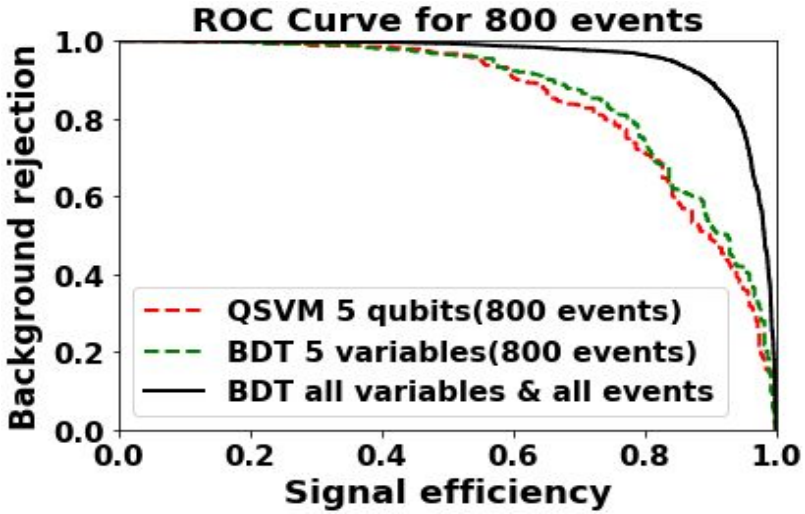
| ttH(H->$\gamma\gamma$) ACCURACY | 200 | 800 | 3200 |
|---|---|---|---|
| QSVM | 0.795 | 0.802 | 0.768 |
| BDT | 0.75 | 0.785 | 0.780 |

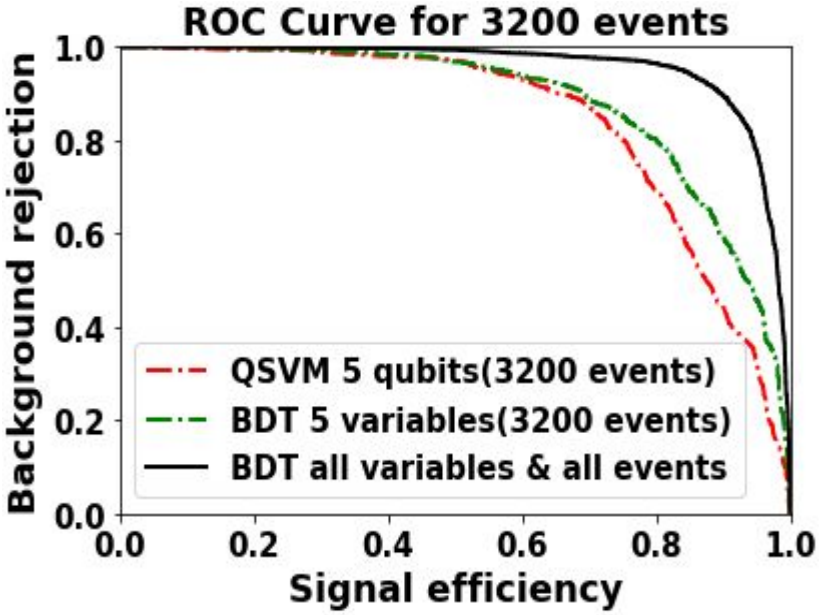
| ttH(H->$\gamma\gamma$) AUC | 200 | 800 | 3200 |
|---|---|---|---|
| QSVM | 0.865 | 0.859 | 0.837 |
| BDT | 0.821 | 0.869 | 0.863 |

# Part 2: Employing QSVM Variational with Q simulators

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

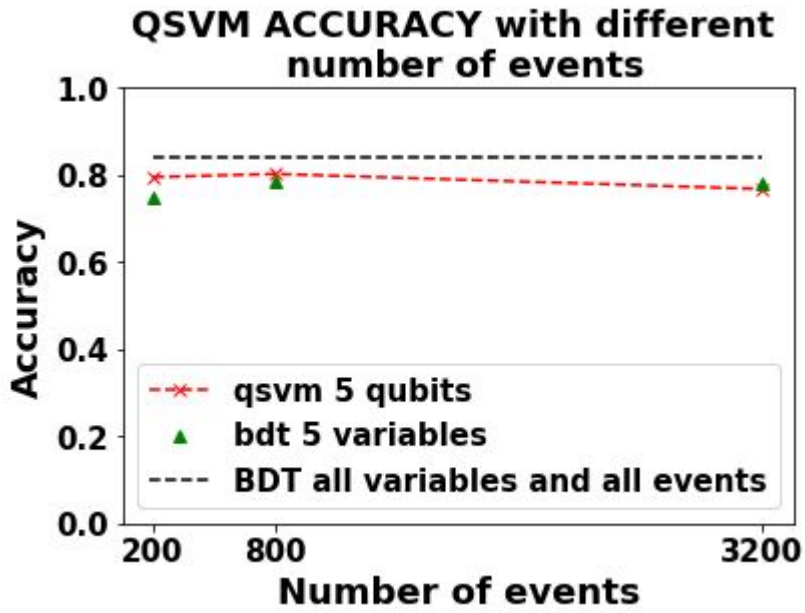


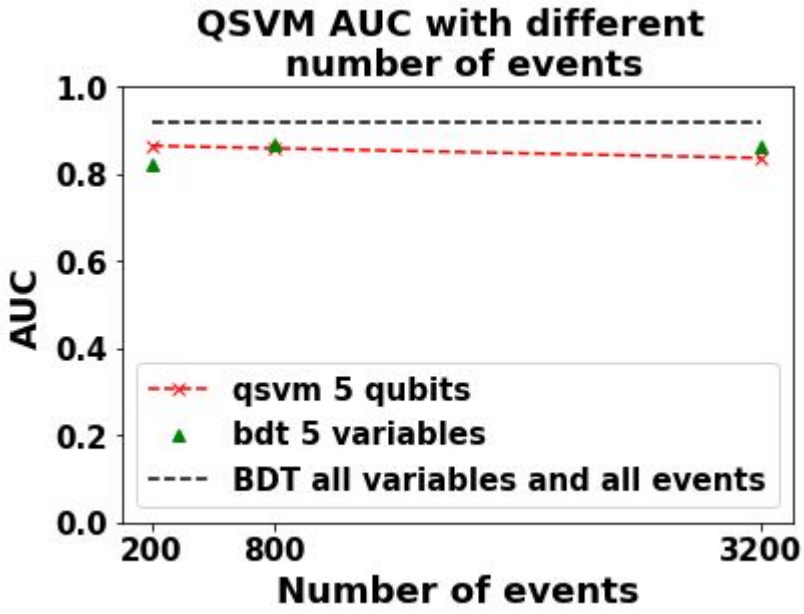




- Here are ROC Curve plots with QSVM(red) and BDT(green), for 200 events, 800 events and 3200 events, with 5 qubits.
- ROC curve(Red): QSVM, 5 qubits
- ROC curve(Green): classical machine learning BDT with the same inputs of 5 variables per event as QSVM for comparison.
- ROC curve(Black): classical machine learning BDT with all 45 variables, all (~ 25k) simulated events.
- For 800 events, the red curve is close to the green curve

# Part 2: Employing QSVM Variational with Q simulators




- Here are ACCURACY and AUC for the QSVM(red) and BDT(green), with 200 events, 800 events and 3200 events.
  - The QSVM(red) method has a similar accuracy and AUC to the BDT(green) method with 5 qubits and a limited number of events. But it's still far from the BDT with all variables and all events(black).
    - Working on various ways (e.g. optimizer, loss function) to improve qsvm AUC.

# Part 2 Bonus: Another example of classical machine learning:
# $H \rightarrow \mu\mu$ analysis by the ATLAS Collaboration

Select events with **two muons** *ATLAS-CONF-2019-028*

➔Separate into **0-jet channel**, **1-jet channel** and **2-jet channel**

Background: $Z \rightarrow \mu\mu$ production

➔Train **Boosted Decision Trees** with XGBoost

3 training variables in the 0-jet channel, 6 in the 1-jet channel, 14 in the 2-jet channel

create **categories based on BDT output**

➔Fit **dimuon mass** over all categories

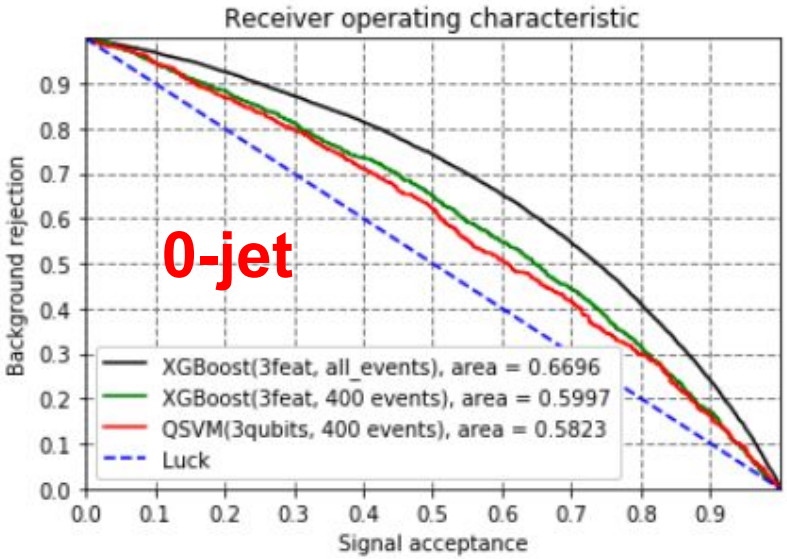
The observed(expected) significance is $0.8\sigma(1.5\sigma)$ using 139 $fb^{-1}$ of ATLAS data

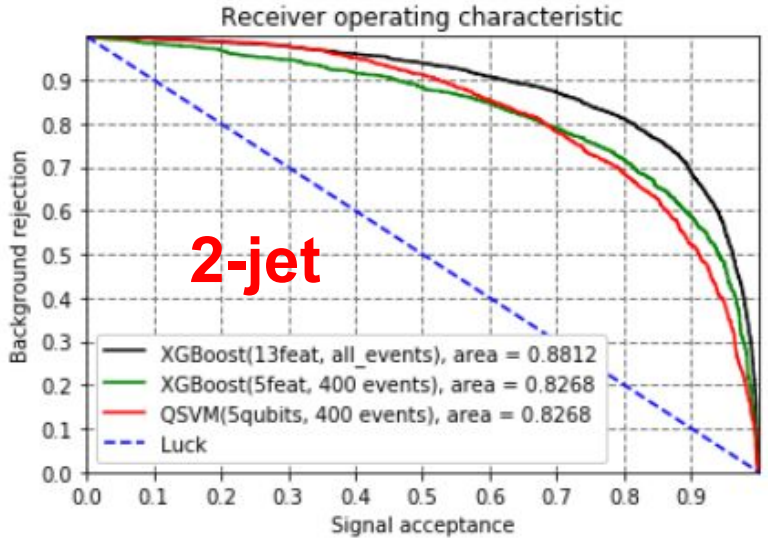
# Part 2 Bonus (Very preliminary): Applying Quantum SVM to the H → μμ analysis

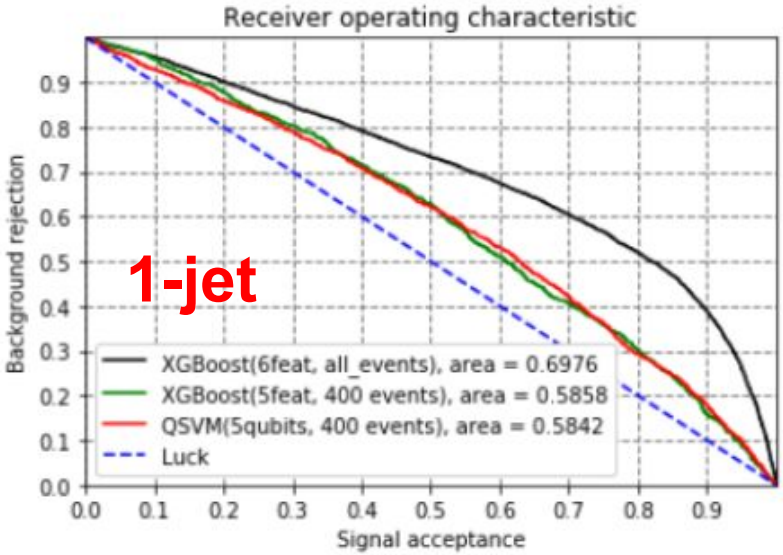
- We are also performing the machine learning step of the H→μμ analysis with Delphes simulation events using quantum SVM
- We successfully finished training and testing for 0-jet (3 qubits), 1-jet (5 qubits) and 2-jet (5 qubits) channels with 400 events using the IBM Qiskit qasm simulator (where 400 events means 400 training events and 400 test events; same for others).
  - In the 1-jet and 2-jets channels, we need to use PCA to reduce the number of training variables to 5

| H->μμ ROC AUC<br>400 events | 0-jet (3 qubits)<br>3 variables | 1-jet (5 qubits)<br>5 variables | 2-jet (5 qubits)<br>5 variables |
|---|---|---|---|
| QSVM | 0.585 | 0.586 | 0.827 |
| BDT | 0.598 | 0.584 | 0.827 |

# Part 2 Bonus (Very preliminary): Applying Quantum SVM to the H → μμ analysis



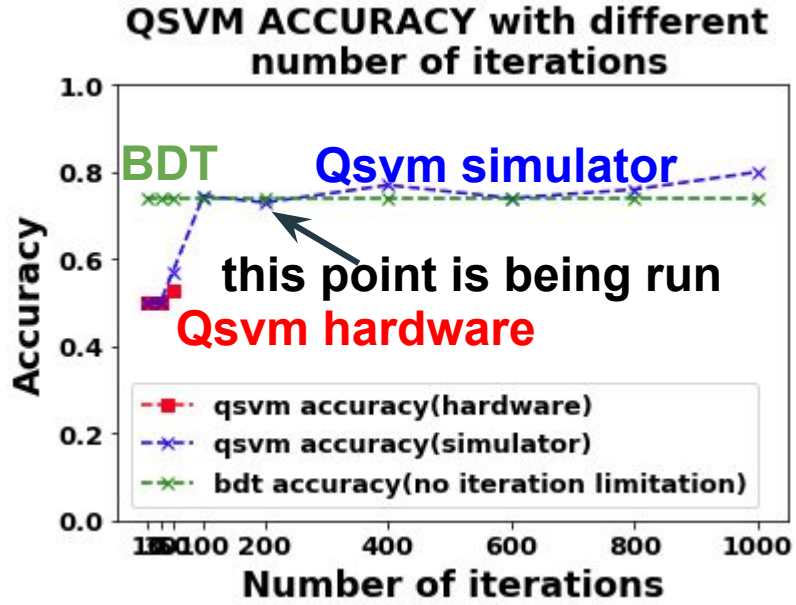



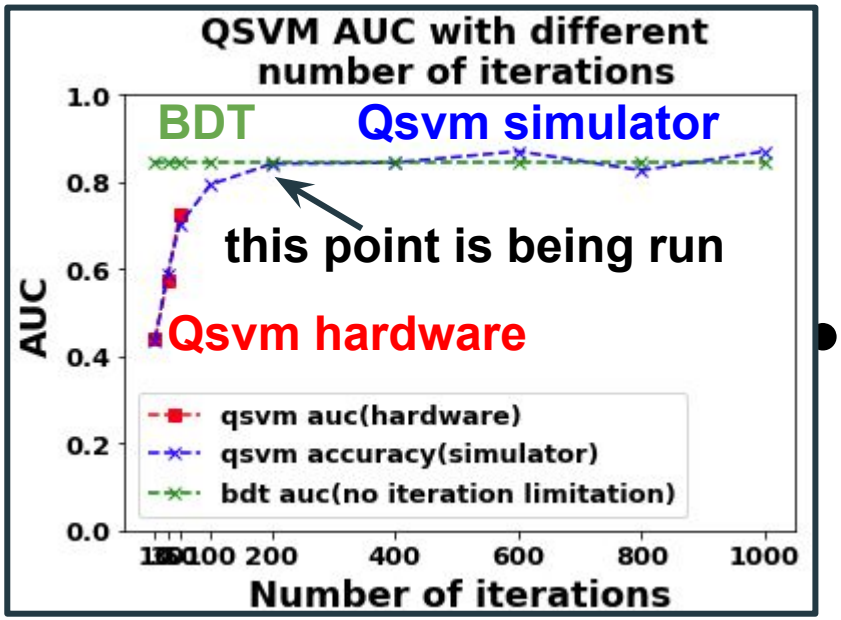

- Here are ROC Curve plots with QSVM(red) and BDT(green) for 400 events, with 3 qubits (for 0-jet) or 5 qubits (for 1jet and 2-jet)
- ROC curve(black): classical machine learning BDT with all variables and all (57k for 0 jet, 20k for 1 jet, 9k for 2 jet) simulated events.
- For 400 events, the AUC of the red curve is close to the AUC of the green curve
- Now back to the ttH example…

# Part 3: Employing QSVM Variational with IBM Q hardware

- With the help of IBM Research Zurich, we finished some training on the IBM Q hardware with 100 training events and 100 test events with 5 qubits.
- Because of hardware access time and timeout limitations, 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 IBM Q hardware




- Here are ACCURACY and AUC plots with different numbers of iterations with 100 Delphes events.
  - Due to access time limitations, on the hardware we only finished 10, 30 and 50 iterations.
- With limited iterations, the result from the hardware(red) is compatible with the result from the simulator(blue) in tested iterations.
- The result from the simulator(blue) reached a similar result as the classical BDT(green) method with enough iterations.

# Part 3: Employing QSVM Variational with IBM Q hardware

- **Temporary limitations with IBM Q hardware**
  - **Only a few iterations are tested currently**
    - **Limited access time**
      - **Long queue time**
  - **Input preparation and output reading is not optimized**
  - **We are working on running hardware with a larger number of iterations**

# Summary

**Referring to Part 1 of this presentation:**

- **We introduced our workflow to employ quantum machine learning methods for LHC High Energy Physics analyses.**

# Summary

## Referring to Part 2 of this presentation:

- **Using IBM Qiskit simulator, we have successfully employed the Quantum Support Vector Machine method for a ttH ($H \rightarrow \gamma\gamma$), Higgs production in association with two top quarks analysis at the LHC with Delphes simulation events. We have measured the accuracy and AUC with different numbers of events.**

- **At the current stage, with a 5 qubit QSVM, we have reached an accuracy of 0.77 and AUC of 0.84, very close to the accuracy of 0.78 and AUC of 0.88 from a classical machine learning method (BDT). (Here the BDT and QSVM are using exactly the same inputs for comparison). At the same time, we are working on various ways (e.g. different optimizers, loss functions) to improve AUC.**

# Summary

**Referring to Part 2 of this presentation:**

- **We are also performing the machine learning step of the H→μμ analysis with Delphes simulation events using quantum SVM**

# Summary

**Referring to Part 3 of this presentation:**

- **Using the IBM Q Experience hardware, we have successfully employed a Quantum Support Vector Machine method(5 qubits) for a ttH ($H \rightarrow \gamma\gamma$), Higgs production in association with two top quarks analysis at the LHC with Delphes simulation events.**

- **Again, the accuracy and AUC are limited by the number of iterations. But the hardware result is compatible with the simulator result, which is itself similar to the result from a classical machine learning BDT with enough iterations. We are working on running hardware with a larger number of iterations**

# 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

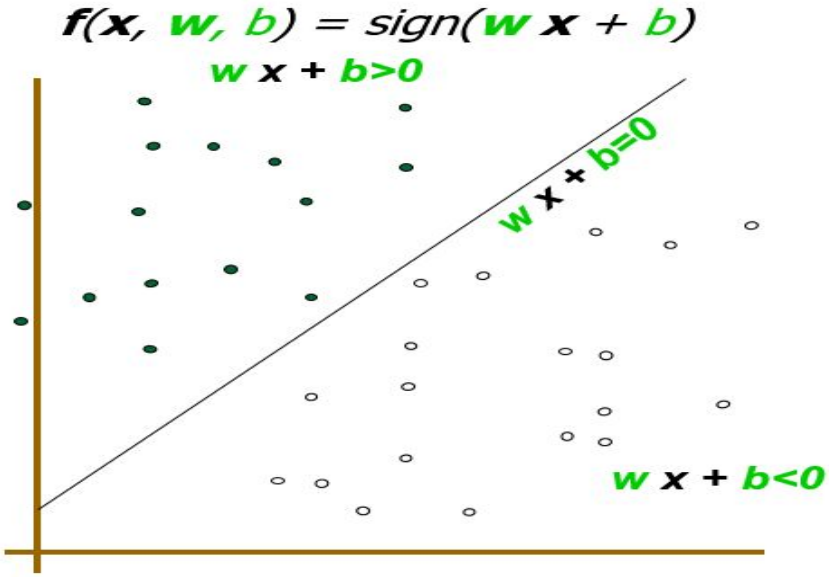
- Quantum feature map: Map bit info non-linearly to quantum 'feature Hilbert space'
  - Tensor product encoding
    - Each feature(variable) of input event is encoded in the amplitude of one separate qubit
    - All features of one event is the tensor product of corresponding qubits
  - Entanglement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

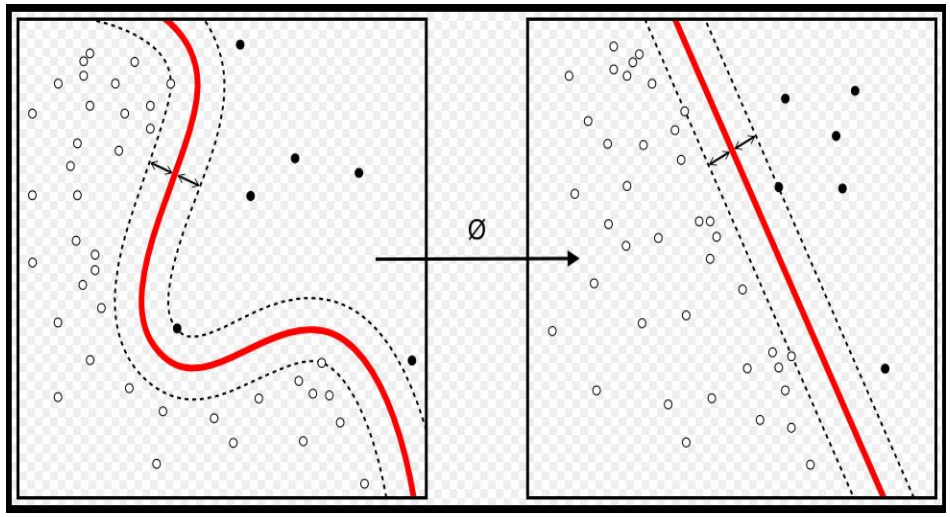



**Ref: Support Vector Machine and Its Application(Mingyue Tan, 2004)**

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