Supervised learning with quantum enhanced feature spaces

BY: IBM TEAM

PRESENTED BY: RAPHAEL BLANKSON

Hello!

“

Machine Learning + Quantum Computing
= Amazing future.

1 Introduction

Let’s do a little prep

Q + ML

- Both changing the way computation is being done.
- Both helping to solve previously untenable problems

MACHINE LEARNING

- Kernel methods for ML are ubiquitous for pattern recognition
- SVMs - well known for classification problems

LIMITATIONS - ML

What happens if:

- Feature space becomes very large
- Kernel functions become expensive to estimate

SOLUTION BY QUANTUM alg.

Quantum algorithms offers

- Exponential speed-ups by taking advantage of the exponentially large quantum space - Hilbert space
    - through controllable entanglement and interference

🔑 keywords

- **Kernel** - a set of mathematical functions that take data as input and transform it to required form.

- **Entanglement** - states of two or more objects described with reference to each other, eventhough the individual objects may be spatially separated

- I**nterference** - interfere with other particles
    - Byproduct of **superposition**

🔨Proposed Method

The authors:

- Propose and implement two novel methods on superconducting processor
- Both methods take advantage of large dimension of quantum hilbert space to obtain enhanced solution
- The data used were created artificially

🔨Proposed Method

The authors:

- Present a quantum algorithm that has potential to run on **near-term quantum devices**
    - **SHORT DEPTH CIRCUITS** are the natural class of algorithms for such noisy devices
- The proposed algorithm takes on the original problem of supervised learning: **construction of a classifier.**

🔨Proposed Method

- **Method One:** Use variational Circuit to generate a separating hyperplane in the quantum feature space - very much like linear binary classifier

- **Method Two:** Use quantum computer to estimate the kernel function of the quantum feature space directly

Necessary Conditions:

- To obtain quantum advantage in both methods, the Kernel must be very **Hard**

🔑 keywords

- **Near - Term**
  - Noisy devices without full error correction
  - Decoherence, gate errors and measurement errors limit the usefulness
  - Algorithms will need to be designed with noisy hardware in mind

CLASSICAL SVM

A little on SVMs

Classical SVM

Consider classifying data set S with unknown labels

$$S = (\vec{x}_i, y_i)\; i = 1, \ldots, l \quad \vec{x}_i \in \mathbb{R}^n$$

$$y \in \{+1, -1\}$$

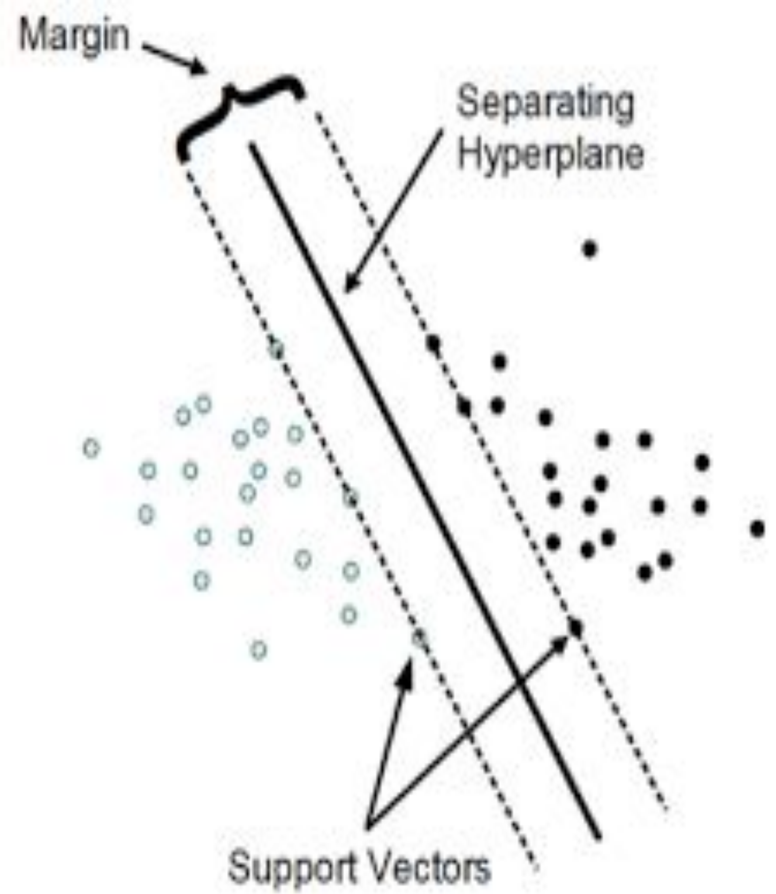
Have access to labeled Training set T

$$T = (\vec{x}_i, yi)\; i = 1...., k$$

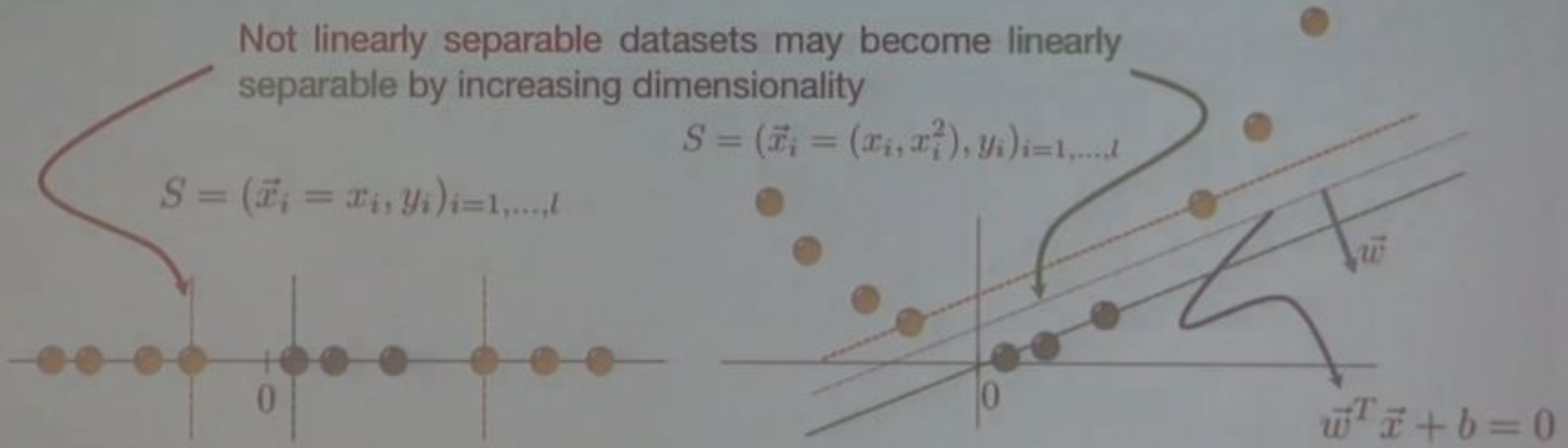
Need to find $m : T \cup S \to C$

$$\tilde{m}(\vec{x}) = sign\left(\sum_\alpha w_\alpha x_\alpha + b\right)$$

Success rate: $v_{succ} = \frac{|\{s \in S|\ \tilde{m}(s) = m(s)\}|}{\{S\}}$



Choosing the right feature map



Classifying the dataset means solving the following problem:

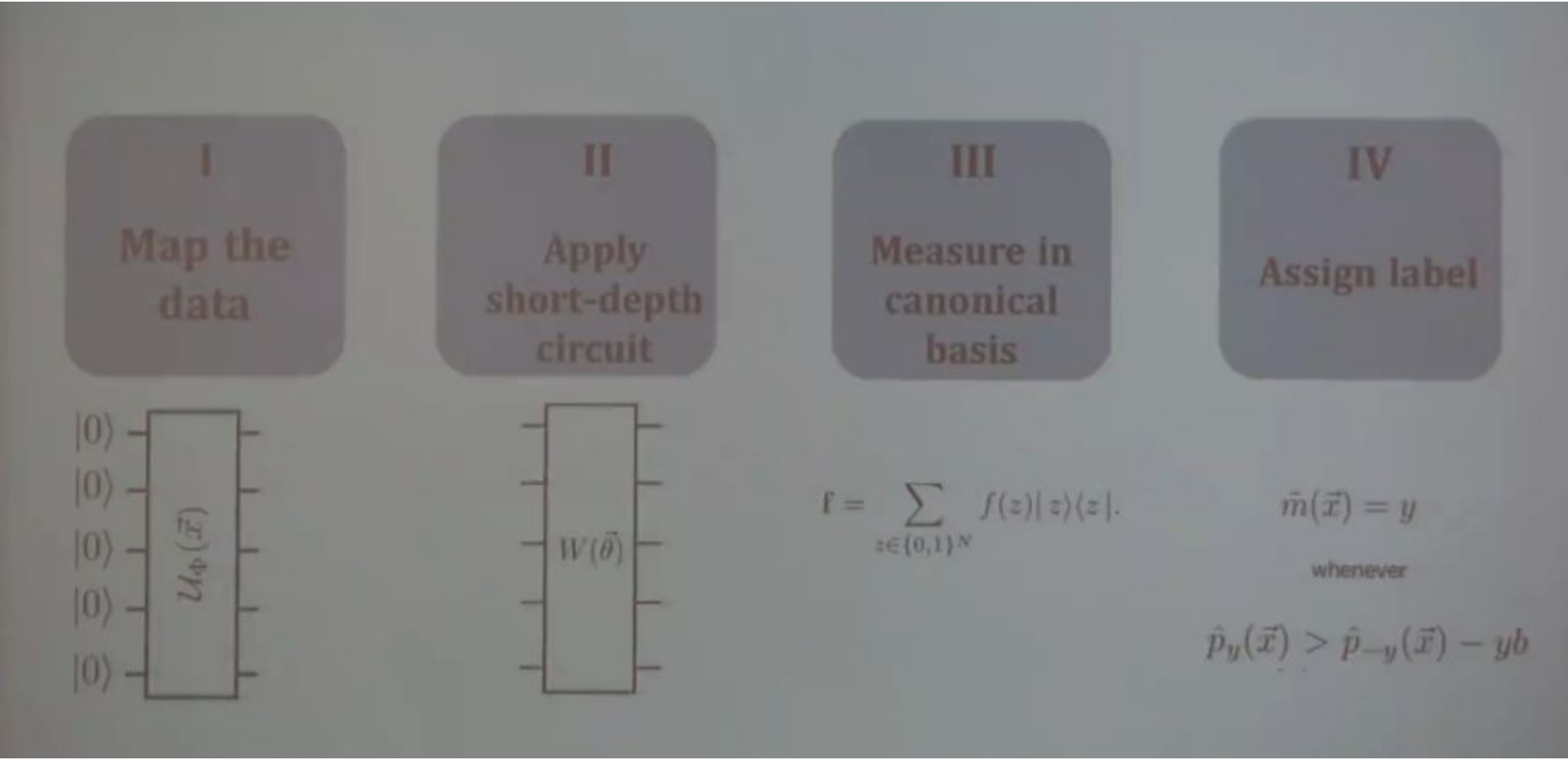
$$\min_{S,w,b} \frac{1}{2}||w||^2 \text{ s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1,\ i = 1, \ldots, l$$

The proposed Methods

Variational Circuit

This is directly related to linear binary classifiers (SVMs)

VC METHOD


I
Map the data
II
Apply short-depth circuit
III
Measure in canonical basis
IV
Assign label
|0⟩
|0⟩
|0⟩
|0⟩
|0⟩
$\mathcal{U}_{\Phi}(\vec{x})$
$W(\vec{\theta})$
$f = \sum_{z\in\{0,1\}^N} f(z)|z\rangle\langle z|.$
$\bar{m}(\vec{x}) = y$
whenever
$\hat{p}_y(\vec{x}) > \hat{p}_{-y}(\vec{x}) - yb$

VC METHOD

- **Decision rule** **(using the empirical output distribution)**

$$\tilde{m}(\vec{x}) = y \quad \text{whenever} \quad \hat{p}_y(\vec{x}) > \hat{p}_{-y}(\vec{x}) - yb$$

The probability of measuring label y $\quad p_y = \frac{1}{2}\left(1 + y\langle\Phi(\vec{x})\,|W^\dagger(\theta)\ \mathbf{f}\ W(\theta)|\,\Phi(\vec{x})\rangle\right)$

Choose operator basis $\quad \mathcal{P}_n = \langle X_i, Y_i, Z_i\rangle_{i=1,\ldots,n}$ and expand $\quad W^\dagger(\theta,\varphi)\ \mathbf{f}\ W(\theta,\varphi) = \frac{1}{2^n}\sum_\alpha w_\alpha(\theta,\varphi)P_\alpha$

then $\quad \tilde{m}(x) = \text{sign}\left(2^{-n}\sum_\alpha w_\alpha(\vec{\theta})\Phi_\alpha(\vec{x}) + b\right) \qquad |\Phi(\vec{x})\rangle\langle\Phi(\vec{x})| = \frac{1}{2^n}\sum_\alpha \Phi_\alpha(\vec{x})P_\alpha$

- **Construction becomes classically efficient for a “trivial” Kernel**

$$K(\vec{x},\vec{y}) = \prod_{i=1}^n |\langle\,\phi_i(\vec{x})\,|\,\phi_i(\vec{y})\,\rangle|^2$$

VC METHOD

- Cost function

$$R_{\mathrm{emp}}(\vec{\theta}) = \frac{1}{|T|} \sum_{\vec{x} \in T} \Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right) \qquad \Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right) \approx \mathrm{sig}\left(\frac{\sqrt{R}\left(\frac{1}{2} - \left(\hat{p}_y(\vec{x}) - \frac{yb}{2}\right)\right)}{\sqrt{2(1 - \hat{p}_y(\vec{x}))\hat{p}_y(\vec{x})}}\right)$$

- Minimize likelihood of predicting a wrong label
- Sigmoid focuses on the really bad cases of the data

VC METHOD

- **Cost function**

$$R_{\mathrm{emp}}(\vec{\theta}) = \frac{1}{|T|} \sum_{\vec{x} \in T} \Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right)$$

$$\Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right) \approx \mathrm{sig}\left(\frac{\sqrt{R}\left(\frac{1}{2} - \left(\hat{p}_y(\vec{x}) - \frac{yb}{2}\right)\right)}{\sqrt{2(1 - \hat{p}_y(\vec{x}))\hat{p}_y(\vec{x})}}\right)$$

- Minimize likelihood of predicting a wrong label
- Sigmoid focuses on the really bad cases of the data

VC METHOD

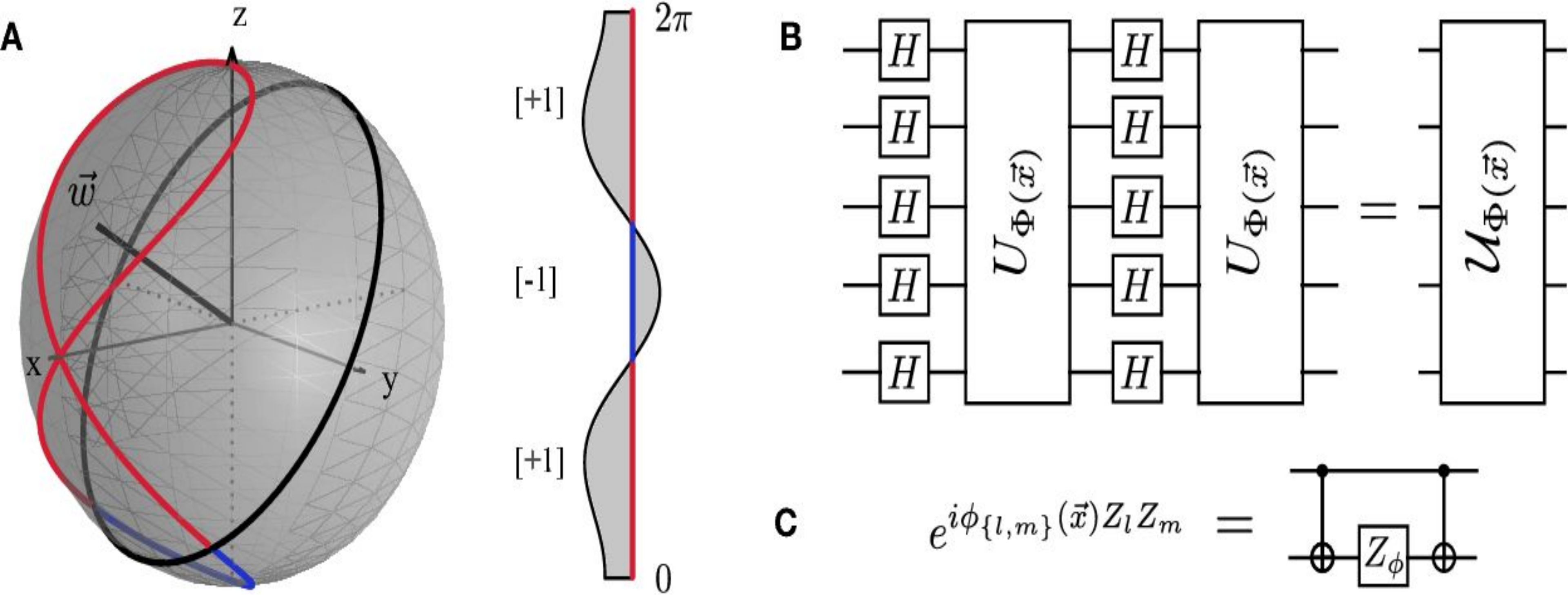
- **Cost function**

$$R_{\text{emp}}(\vec{\theta}) = \frac{1}{|T|}\sum_{\vec{x}\in T} \Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right)$$

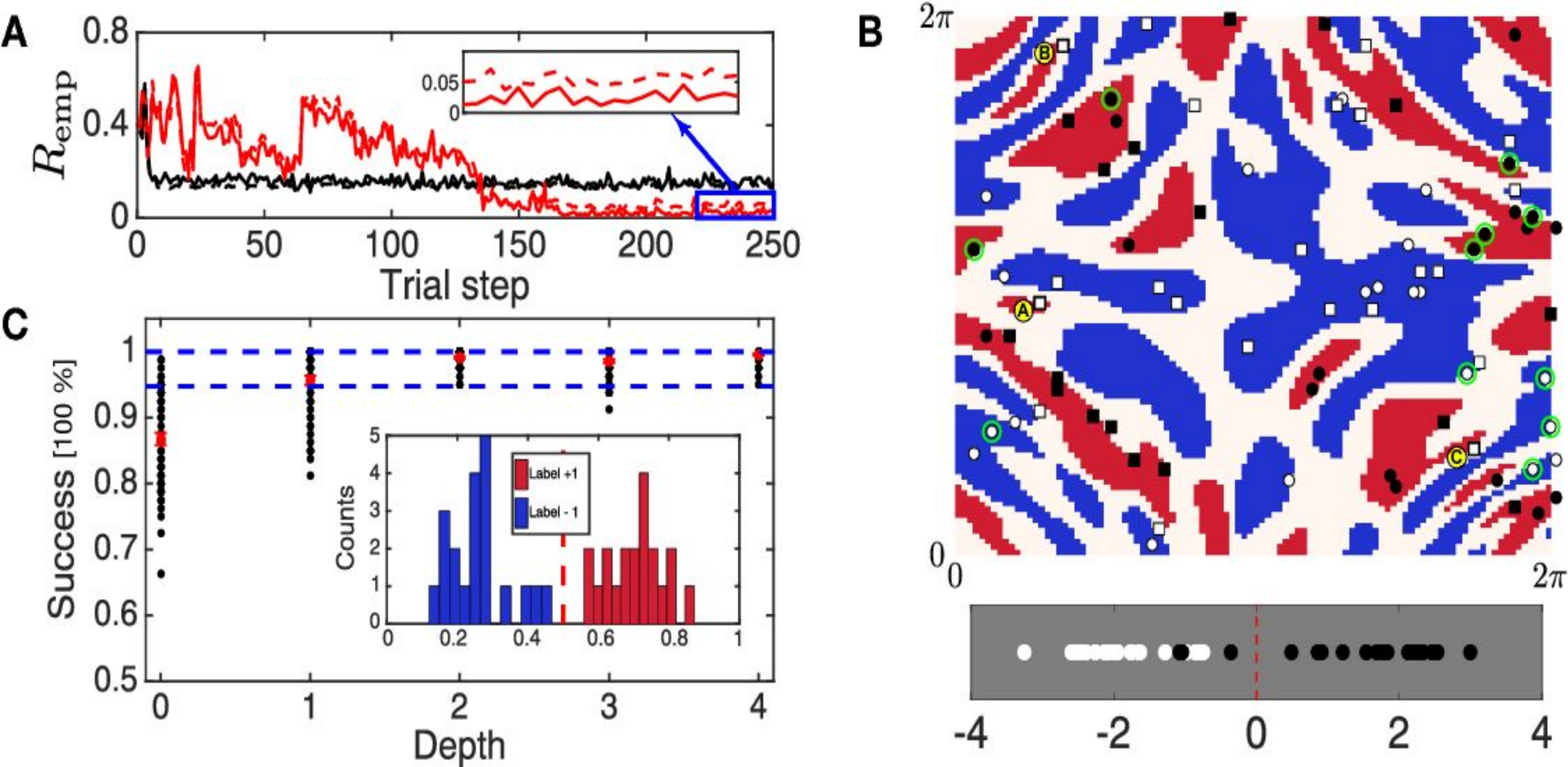
$$\Pr\left(\tilde{m}(\vec{x}) \neq m(\vec{x})\right) \approx \text{sig}\left(\frac{\sqrt{R}\left(\frac{1}{2} - \left(\hat{p}_y(\vec{x}) - \frac{yb}{2}\right)\right)}{\sqrt{2(1-\hat{p}_y(\vec{x}))\hat{p}_y(\vec{x})}}\right)$$

- Minimize likelihood of predicting a wrong label
- Sigmoid focuses on the really bad cases of the data

VC METHOD


A
z
$\vec{w}$
x
y
[+1]
[-1]
[+1]
2π
0
B
H
$U_{\Phi(\vec{x})}$
$=$
$\mathcal{U}_{\Phi(\vec{x})}$
C
$e^{i\phi_{\{l,m\}}(\vec{x}) Z_l Z_m} =$
Z_ϕ

VC METHOD


A
0.8
0.4
0
R_{emp}
0.05
0
0
50
100
150
200
250
Trial step
B
2π
0
0
2π
-4
-2
0
2
4
C
1
0.9
0.8
0.7
0.6
0.5
Success [100 %]
0
1
2
3
4
Depth
Counts
Label +1
Label - 1
0.2
0.4
0.6
0.8

Direct Kernel

Second Approach

Method 2: Direct Kernel method

- **SVM training (Dual of the original quadratic program)**

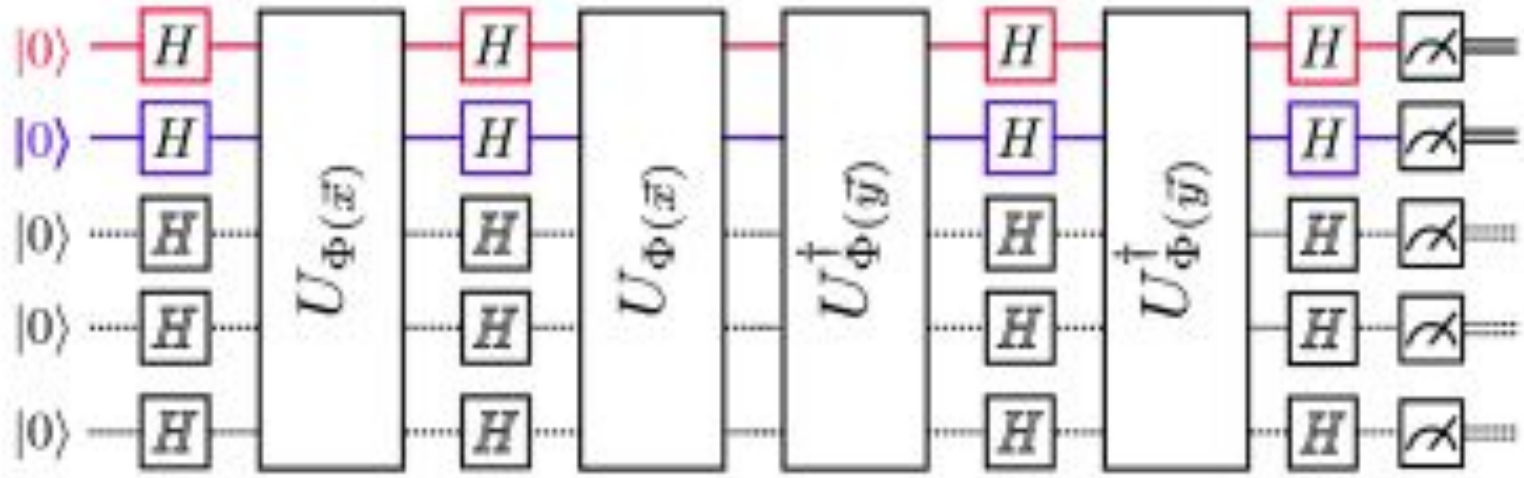
$$L_D(\alpha) = \sum_{i=1}^{t} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{t} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$

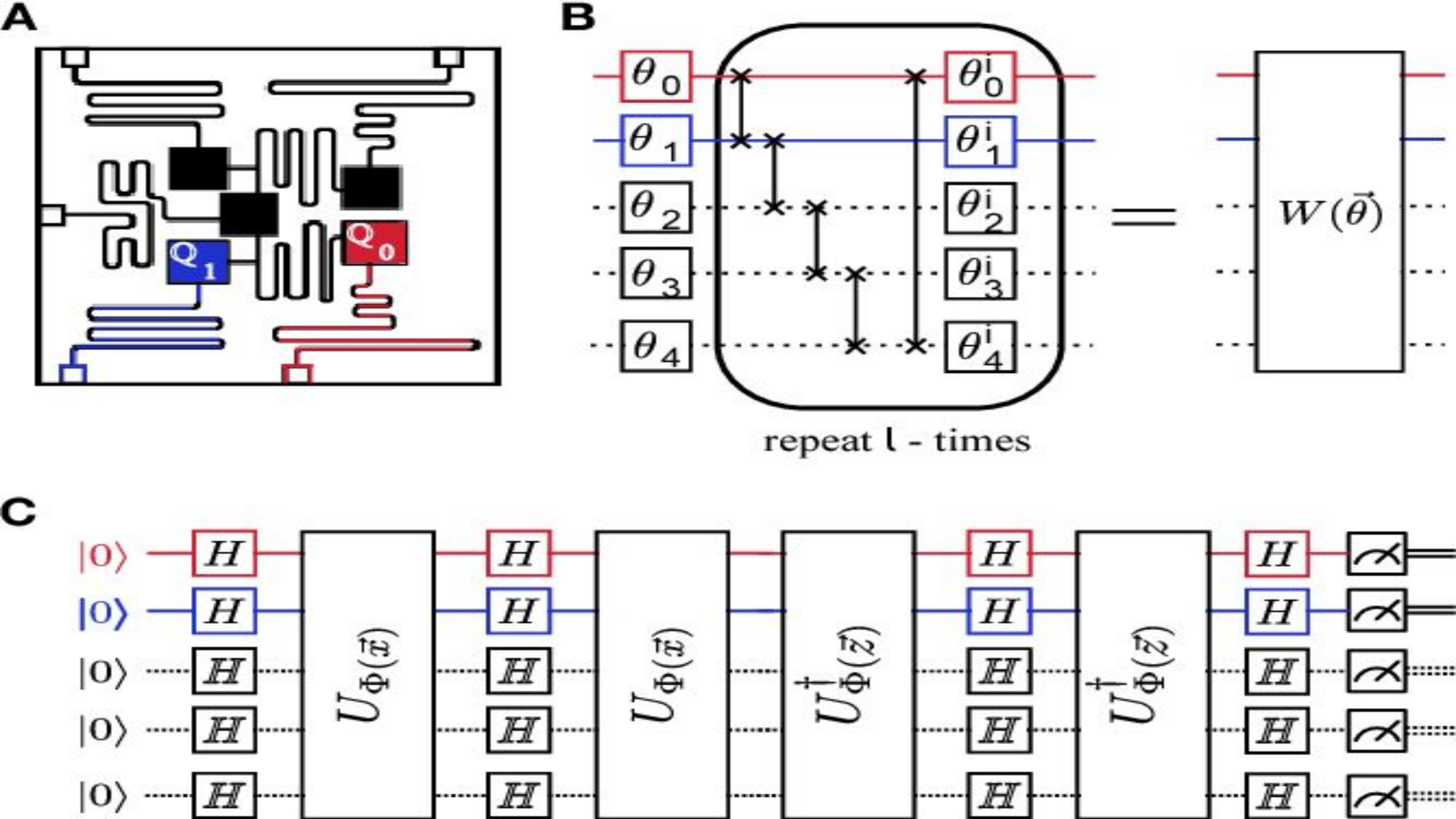
- SVM classification $\tilde{m}(\vec{s}) = \text{sign}\left(\sum_{i=1}^{t} y_i \alpha_i^* K(\vec{x}_i, \vec{s}) + b \right)$

1. The algorithm needs to call the quantum computer $|T|^2$ times in training and N_S times for classificatio
2. This is a concave optimization problem, and therefore efficient

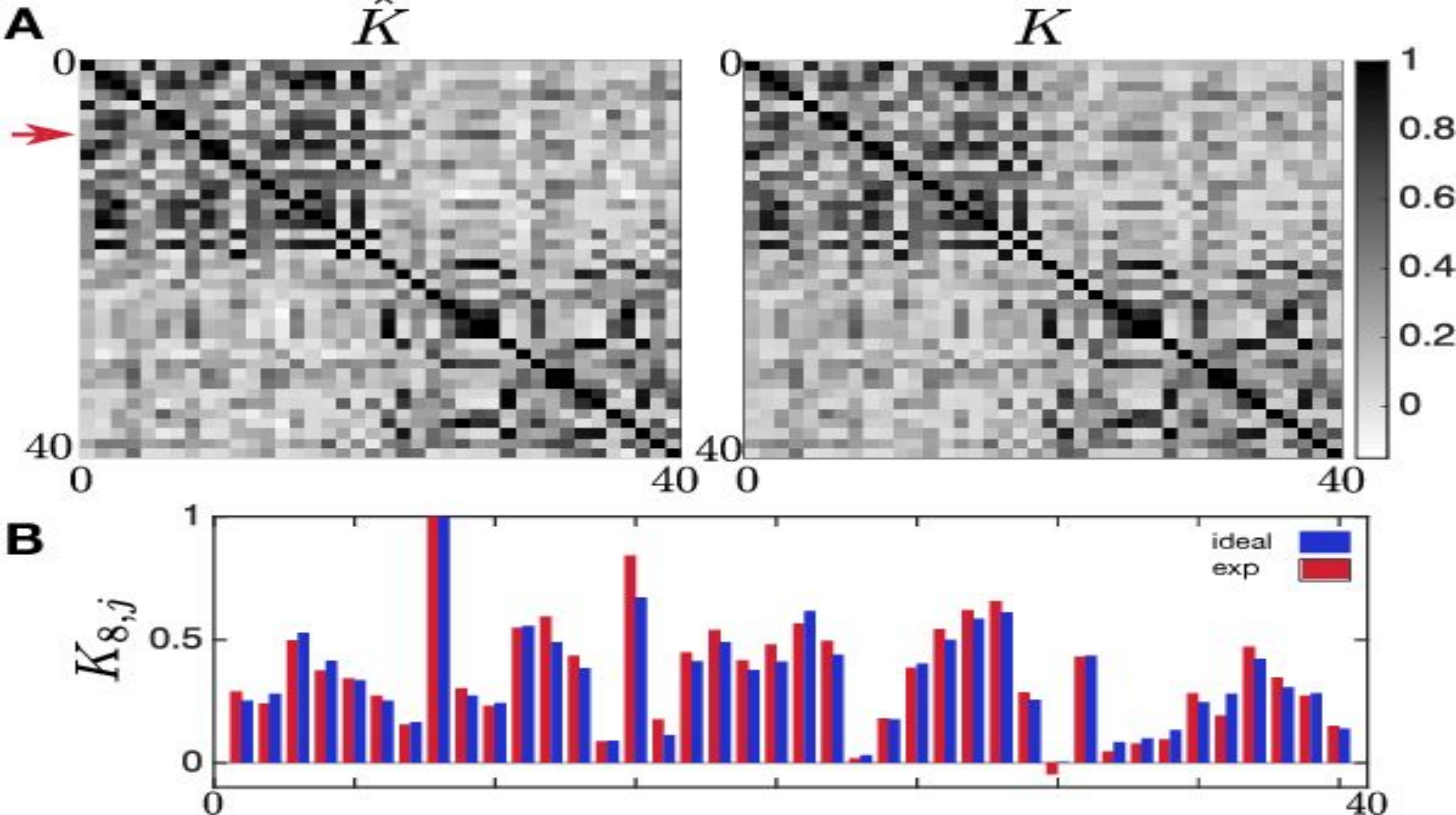
ESTIMATING THE QUANTUM KERNEL

$$K(\vec{x}, \vec{z}) = \left| \langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle \right|^2 = \left| \langle 0^n | \mathcal{U}^\dagger_{\Phi(\vec{x})} \mathcal{U}_{\Phi(\vec{z})} | 0^n \rangle \right|^2$$




A
Q0
Q1
B
θ0
θ1
θ2
θ3
θ4
θi0
θi1
θi2
θi3
θi4
repeat l - times
W(θ)
C
|0⟩
H
UΦ(x)
U†Φ(z)

Experimental Results



A
$\hat{K}$
K
0
40
0
40
0
40
0
40
1
0.8
0.6
0.4
0.2
0
B
$K_{8,j}$
1
0.5
0
0
40
ideal
exp

🔑 LIVE TEST

LETS US SEE EVERYTHING LIVE AND COLORED

https://ibm-q4ai.mybluemix.net/

OPEN PROBLEMS:

How do we find interesting Quantum kernels and feature spaces

CONCLUSION

The authors have experimentally have showed a classifier that exploits quantum feature space and proved to achieve 100% success despite the presence of noise

Thanks!

Any questions?

You can find me at @username & user@mail.me