Class 4: PWF 2021 - QML 12/02/2021

Quantum Ansatz design:

Ansatz for $U(\vec{\theta})$: what are we using? where are we placing them?

General for $U(\vec{\theta})$:

$$U(\vec{\theta}) = \prod_{j=1}^{L} e^{-i\theta_j \sigma_j} W_j$$

($\theta_j \sigma_j$: parametrized gates; W_j: unparametrized gate(s))

- Problem-inspired ansatz. we take information about the problem and we embed it into the design of $U(\vec{\theta})$.

  UCC $e^{T-\hat{T}} = U(\vec{\theta})$ $\quad T = T_1 + T_2$

  QAOA $U(\vec{\theta}) = \prod_{i=1}^{p} e^{-i\beta_i H_M} e^{-i\gamma_i H_P}$

- Problem-agnostic ansatzes we have no information to encode into the design of $U(\vec{\theta})$.

Hardware Efficients Ansatzes

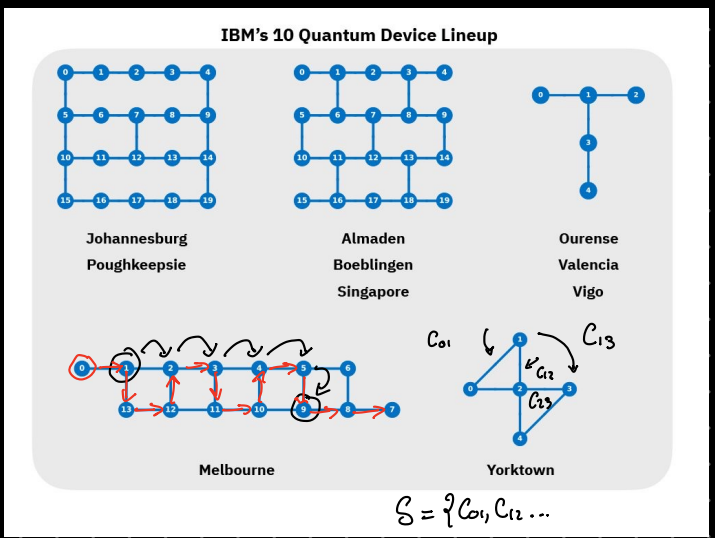
Kandala et al Nature 549, 242-246 (2017)

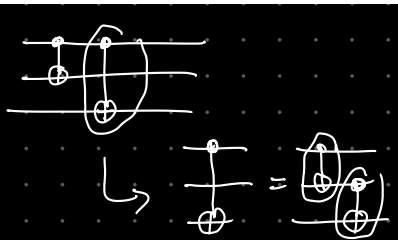
arxiv 1704.05018

We are going to build a quantum circuit from the Native set of a QC.

Native set of gates:

- arise from the native interactions in the device
- arise from the native connectivity



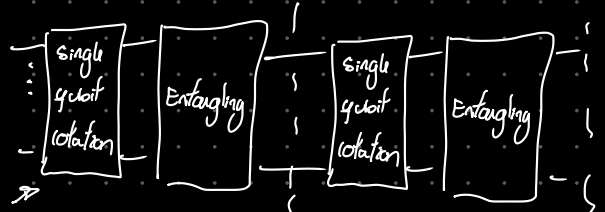


Set of Native gates is compos

$\{ R_\mu, P(\varphi), CNOT \} \longrightarrow$ device orientes $\{ R_\mu, P(\varphi), CNOT_{i,i+1} \}$

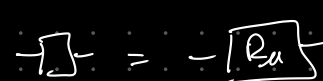
$\mu = x, y, z$

Hardware Efficient ansatz: reduce the depth and use native native



we assume a layered structure of single-qubit rotations and entangling gates

1) 

 → n parameters

 3n parameters

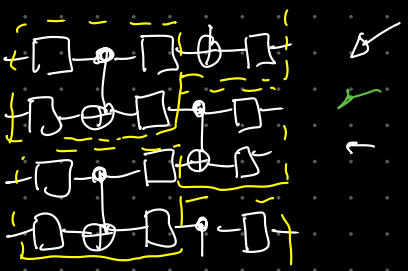
TIP #7

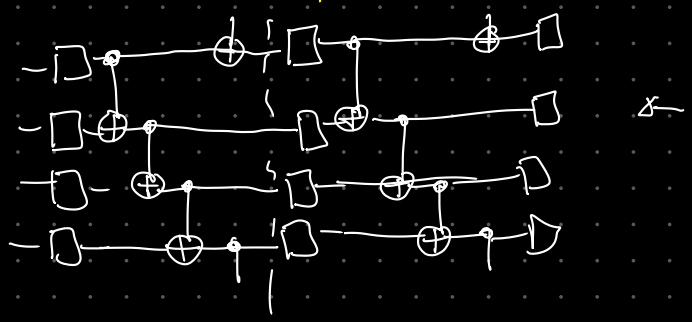
This is the most general 1-qubit gate

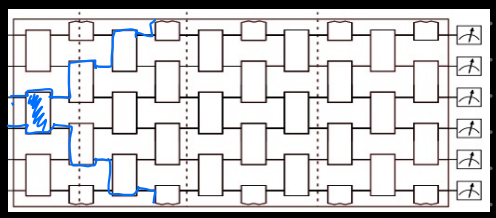
2) 

a) Entangling alternating pairs of qubits

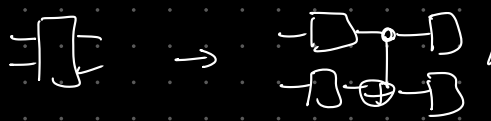
b) Entangle all qubits

a) $U(\alpha) =$ 

b) 

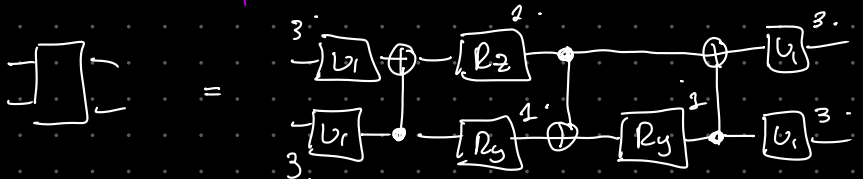


Hardware efficient ansatz with alternating 2-qubit unitaries



what is the most general 2-qubit gate

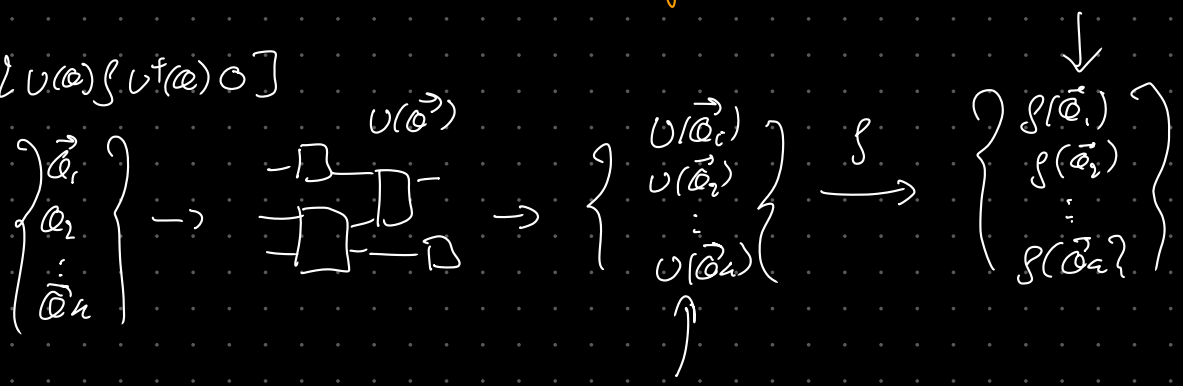
Vatan PRA 69, 032315 (2004) arxiv 0308006 (2003)

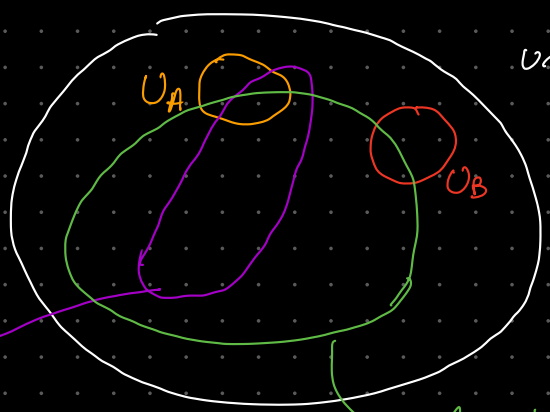


U_1 = the general 1-qubit gate (3 parameters)

Difference between problem inspired & agnostic ansatzes

$Tr[U(\vec{\theta}) O U^\dagger(\vec{\theta}) O]$





group

Unitary space of degree d

$U(d)$

Higher volume in the group

= high expressitivity

Accesible space by $U(\vec{\theta})$ Problem

Accesible space by a problem-agnostic

- Variable structure ansatz $U(\vec{\theta}, \vec{k})$

Bilkis

arXiv 2103.06712

$\vec{\theta}$: rotation angles — continuous variable

$\vec{k}$: position or type of gates — discrete variable

Optimizer Choice

Classical methods

SPSA
Nelder-Mead
COBYLA
Powell

Gradient-free

Gradient descent
stochastic GD
Newton GD

Gradient-based

- Computing Gradients on a QC

Mitari PRA 98, 032309 (2018) arXiv 1803.00745

Schuld PRA 99, 032331 (2019) arXiv 1811.11184

- $C(\vec{\theta}) = Tr[U(\vec{\theta}) S U^{\dagger}(\vec{\theta}) O]$; $U(\vec{\theta}) = \prod_{j=1}^{L} e^{-i\theta_j/2 \sigma_j} W_j$

$\sigma_j = ?X, Y, Z?$

$$\frac{\partial C(\vec{\theta})}{\partial \theta_k}$$

1) $$\frac{\partial}{\partial x} Tr[A(x)] = Tr\left[\frac{\partial}{\partial x} A(x)\right]$$

$$\frac{\partial C(\vec{\theta})}{\partial \theta_k} = Tr\left[\frac{\partial}{\partial \theta_k} \underbrace{\left(U(\vec{\theta}) S U^{\dagger}(\vec{\theta}) O\right)}_{?}\right]$$

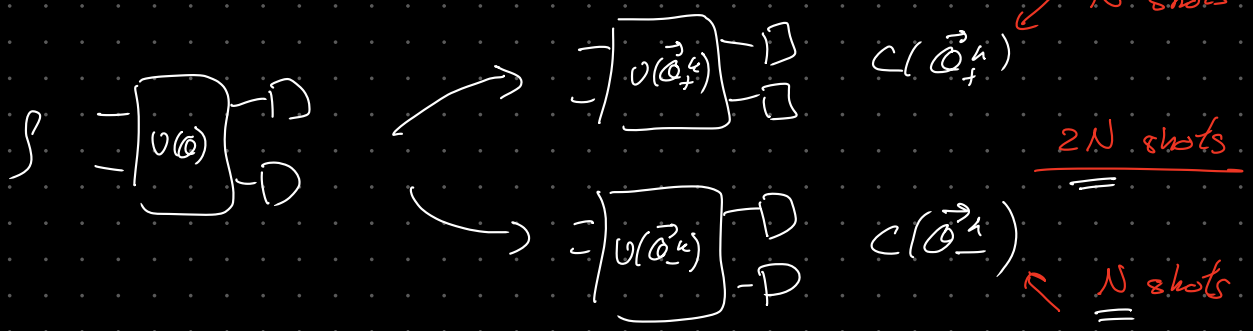
2) $$\frac{\partial}{\partial x} A(x) B C(x) = \left(\frac{\partial}{\partial x} A(x)\right) B C(x) + A(x) B \left(\frac{\partial C(x)}{\partial x}\right)$$

3) $$\frac{\partial}{\partial \theta_k} U(\vec{\theta})$$

$$\frac{\partial C(\vec{\theta})}{\partial \theta_k} = \frac{1}{2}\left(C(\vec{\theta}_+^k) - C(\vec{\theta}_-^k) \right)$$

Parameter Shift Rule

$$\vec{\theta}_\pm^k = (\theta_1, \theta_2, \dots, \theta_k \pm \tfrac{\pi}{2}, \dots \theta_L)$$



$$\vec{\nabla} C(\theta) = \left(\frac{\partial C(\vec{\theta})}{\partial \theta_i}, \dots \right)$$

2LN shots

$$\left| \frac{\partial C(\vec{\theta})}{\partial \theta_k} \right| = \frac{1}{2} \left| C(\vec{\theta}_+^k) - C(\vec{\theta}_-^k) \right| \leq \frac{1}{2} \left(|C(\vec{\theta}_+^k)| + |C(\vec{\theta}_-^k)| \right)$$

$|C(\vec{\theta}_+^k)| \leq |C_{max}|$, $|C(\vec{\theta}_-^k)| \leq |C_{max}|$

$$\leq |C_{max}| \leq 1$$

The landscape of VQA is not rugged.

Quantum-aware optimizer

Arrasmith arXiv 2004.06252

Kulber Quantum 4, 263 (2020) arXiv 1909.09083.

$$O = \sum_i c_i \sigma_i$$

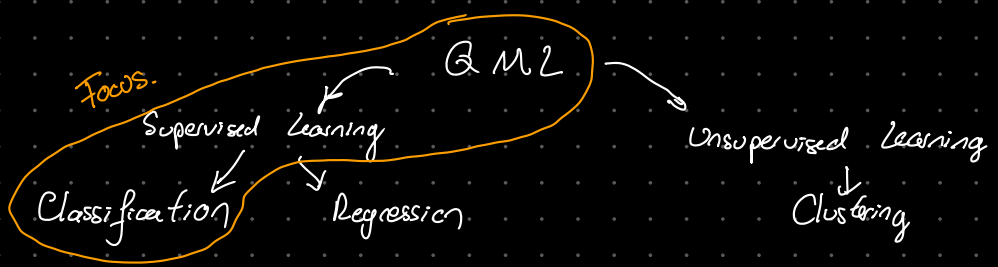
$\langle O \rangle$

Quantum Machine Learning

Use Quantum Computers to learn from data.

VQA : Input ⊕ program = Output

QML : Input ⊕ Output = program



Framework for Supervised Classification :

- we denote by X the set of all possible examples or instances. X = input space

  we denote by Y the set of labels or target values

We limit ourselves to the case where $Y = \{-1, 1\}$ = Binary classification

- we assume there is an unknown map $h: X \to Y$ what we want to learn
- we have a way to draw examples independently and identically distributed (i.i.d) according to some (unknown) distribution D.

Learning Problem :

Draw iid according to D

- train set S
- test set T

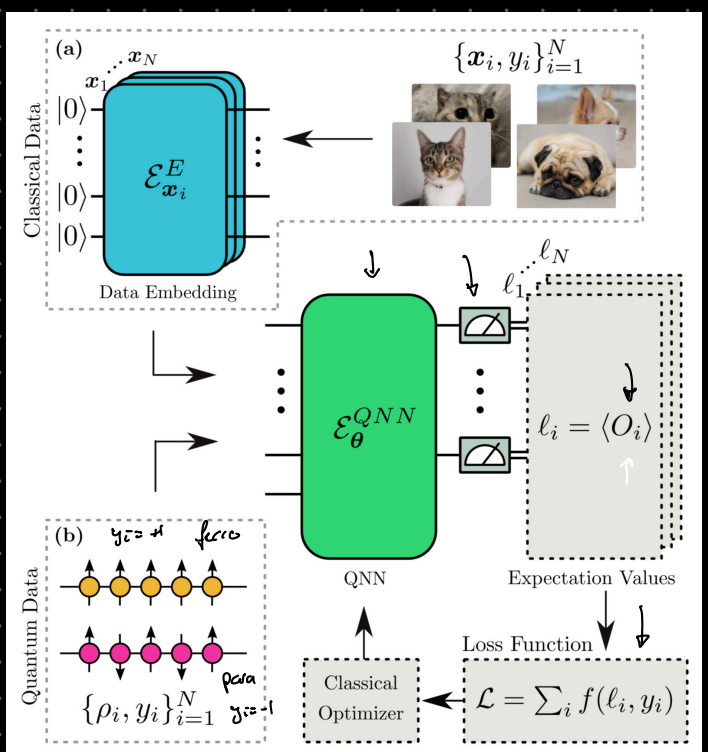
given a possible set of maps $\mathcal{H}$, a hypothesis set, we want to train a QML model $\tilde{h} \in \mathcal{H}$ over the training data so that the predictions of $\tilde{h}$ match those of h with high probability in

- the training set low training error
- the test set, previously unseen cases, low generalization error.

why low training error $\neq$ low generalization $\quad \tilde{h}(x_i) = y_i \quad x_i \in S$

$\tilde{h}(x_i) = \begin{cases} -1 & 50\% \\ +1 & 50\% \end{cases} \quad x_i \notin S$

What does the QML model looks like? Main goal: use a QC to give our hypothesis $\tilde{h}$ access to the exponentially large Hilbert space.



$\{\rho(\vec{x}_i), y_i\}$

y_i ↕ labels

$\{\rho_i, y_i\}$

↝

for classical data we need to map the inputs onto quantum states

$\vec{x}_i \Rightarrow \rho(\vec{x}_i)$

Cost function = VQA

Loss function = QML

Classical Data the dataset is of the form $\{\vec{x}_i, y_i\}$

$\vec{x}_i$: bitstrings, real-valued vectors; y_i: labels

The first step is to embed the classical data points $\vec{x}_i$ onto quantum states that a QC can use for information processing

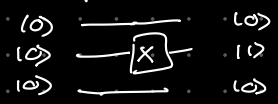
1) **Basis encoding**: real numbers, arithmetic operations

Take a bitstring, or convert real valued number into bitstring representation

$\vec{x} \in \{0,1\}^{\otimes m}$

we promote $\vec{x} \rightarrow |\vec{x}\rangle$ if $m = n$

$010 \rightarrow |010\rangle$



The whole dataset can be expressed in a superposition

$$|S\rangle = \frac{1}{|S|} \sum_{i=1}^{|S|} |\vec{x}_i\rangle$$

PRO: easy

CON: wasteful, QM is linear

2) Amplitude encoding The data is embedded into the amplitudes of a quantum state

$$\vec{x} = (x_1, x_2 \dots x_m) \rightarrow \vec{x} \updownarrow |2\rangle = \begin{pmatrix} \\ \\ \end{pmatrix}$$

$$|\vec{x}\rangle = \sum_{j=1}^{m} \frac{x_j}{\sqrt{\sum_{j=1}^{m} x_j^2}} |z_j\rangle$$

$|z_j\rangle :: \{|0\rangle, |1\rangle\}^{\otimes n}$

$$|S\rangle = \frac{1}{|S|} \sum_{i=1}^{|S|} |\vec{x}_i\rangle$$

CON: widely used in many QML algorithms, qubit efficient

$\log_2(m)$ qubits to encode a bitstring of length m

$n = \log_2(m)$

$2^n = 2^{\log_2(m)} = m$ $\quad O(\text{poly}(n)) = O(\text{poly}(\log(m)))$

PRO: not obvious how to prepare $|\vec{x}\rangle$, we have to fine-tune or optimize circuits that prepare $|\vec{x}\rangle$, Also linear.

3) Quantum Circuit encoding use the values of $\vec{x}$ as parameters in a embedding/encoding circuit.

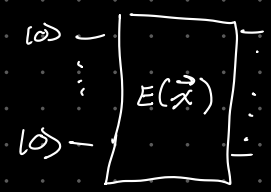
Lloyd arxiv 2001.03622 (2020)

Thanwarit arxiv 2105.02276 (2021)

Hubregtsen arxiv 2105.02276 (2021)

Havlicek Nature 567, 209-212 (2019) arxiv 1804.11326

Initialise qubits to a fiducial state $|\vec{0}\rangle = |0\rangle^{\otimes n}$



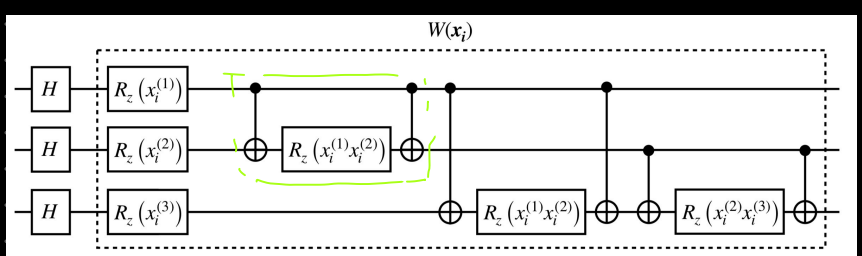
Fixed structure or a variable structure

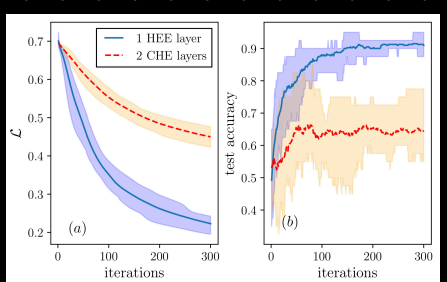
$\vec{x}_i = (x_i^{(1)}, x_i^{(2)}, \dots)$

Parametrized Quantum Circuit

$$\textcircled{*} \quad E(\vec{x}_i) = \left(\prod_{j<k} e^{-i x_i^{(j)} x_i^{(k)} Z_j Z_k} \right) \left(\prod_{j=1}^{n} e^{-i x_i^{(j)} Z_j} \right)$$

$|\vec{x}_i| = m = n$



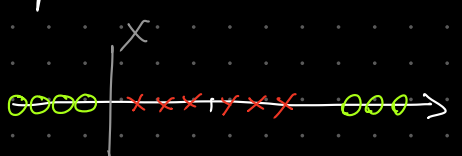


PRO: easy to obtain non-linearities, Quantum kernel

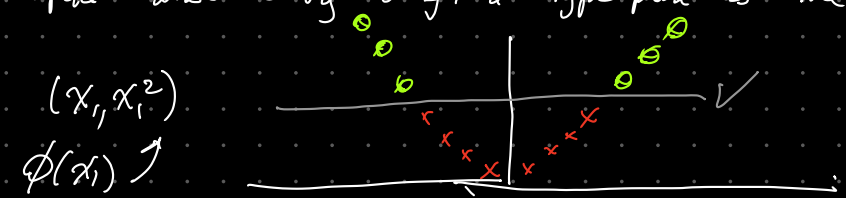
CON: not obvious what ansatz to use for $E(\vec{x})$

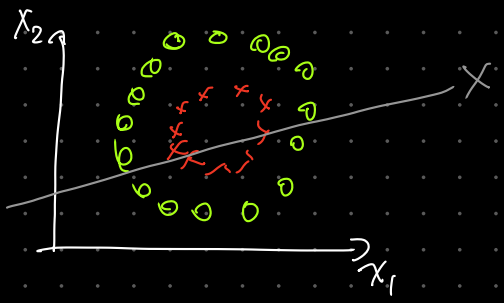
Kernel Trick in Machine Learning

We have data that is not linearly separable in some original input data space.

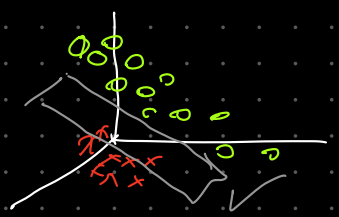


We apply a transformation to the data and map it to a higher-dimensional space where we try to fit a hyper-plane as the decision boundary.





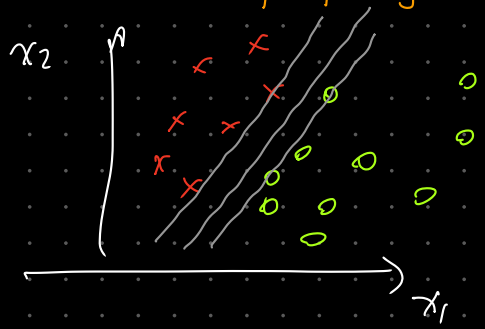
$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$



The kernel trick used to analyze data in the higher-dimensional space. Take the vectors in the original space, and return the inner product in the higher-dimension.

$\vec{x}_1, \vec{x}_2$ and a map ϕ $\quad K(\vec{x}_1, \vec{x}_2) = \langle \phi(x_1), \phi(x_2) \rangle$

Task Goal: of finding the decision boundary:



Support Vector Machines (SVM)

We solve the optimization problem of determining the hyperplane that separates the data by class.

Data points with minimum distance to the hyperplane are called the support vectors.

Due to their closeness to the boundary, their influence on the exact position of hyperplane is larger

Can be solved via the

Kernel $K(\vec{x}_i, \vec{x}_j)$

$\vec{x} \xrightarrow{\phi} \rho(\vec{x}) = E(\vec{x}) \vec{10} \times \vec{0} | E^+(\vec{x})$

what's the kernel

$$K(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}\left[\rho(x_i)\, \rho(x_j)\right]$$

SVM for classification

Classification → Kernel methods, SVM

→ Neural Networks

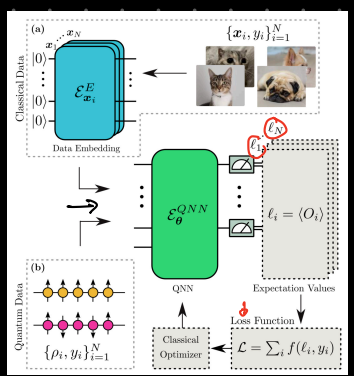
Kernel CQN: kernel might not properly bifurcate the space, hyperplane is not enough

~ o ~

Quantum Neural Networks

Framework

QNN = generalize classical NN for information processing based on the laws of Q.M.



$\{\rho_i, y_i\}$ these can be quantum data or encoded classical data

ρ_i → QNN → ρ_i

y_i labels

QNN = parametrized quantum circuit

Take a quantum state ρ_i from the dataset, send it QNN (i.e. parametrized quantum circuit) and compute some expectation value via measurements

$$\ell_i(\vec{\theta}, y_i) = \mathrm{Tr}\left[U(\vec{\theta})\, \rho_i\, U^\dagger(\vec{\theta})\, O_{y_i} \right]$$

And we are going to compute the loss function

$$\mathcal{L}(\vec{\theta}) = \sum_{i=1}^{|S|} f\left(\ell_i(\vec{\theta}, y_i), y_i\right)$$

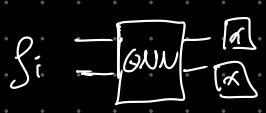
Example

Binary classification on quantum data

X = subspace of $\mathcal{H}$ $\quad Y = \{-1, 1\}$

(1)

- We can assign labels via measuring k qubits ($k \leq n$) and estimate the number of bitstring with even / odd parity
  label 1 / label -1



00, $N_{00} \to p_{00} = N_{00}/N$ — even
01, $N_{01} \to p_{01} = N_{01}/N$ — odd
10, $N_{10} \to p_{10} = N_{10}/N$ — odd
11, $N_{11} \to p_{11} = N_{11}/N$ — even

parity of (z) = number of 1s in z (mod 2)

How many outcomes had even parity $\quad N_{00} + N_{11}$, $p_{00} + p_{11}$

" " " odd " " $\quad N_{10} + N_{01}$ / $p_{10} + p_{01}$

The probability of assigning label y_i

$$p(y_i|\vec{0}) = \mathrm{Tr}\left[U(\vec{0}) \rho_i U^\dagger(\vec{0}) O_{y_i} \right]$$

$$O_1 = \sum_{z:\text{even}} |z\rangle\langle z|$$

$$O_{-1} = \sum_{z:\text{odd}} |z\rangle\langle z|$$

(2)

- Instead of the measurement outcome being a probability, the measurement outcome can be in itself the assigned label

$$\tilde{y}_i(\vec{0}) = \mathrm{Tr}\left[U(\vec{0}) \rho_i U^\dagger(\vec{0}) Z^{\otimes n} \right] \quad \in [-1, 1]$$

continuous label

Approach (1) and (2) are equivalent

$$Z^{\otimes n} = \sum_{z:\text{even}} |z\rangle\langle z| - \sum_{z:\text{odd}} |z\rangle\langle z|$$

$$Z^{\otimes n} = O_1 - O_{-1}$$

$$\tilde{y}_i(\vec{0}) = p_i(1|\vec{0}) - p_i(-1|\vec{0})$$

$$p_i(y_i|\vec{0}) = \frac{1 + y_i \tilde{y}_i(\vec{0})}{2}$$

Labels

00	01	10	11
$\rho =$ [ ]			
$p(00)$	$p(01)$	$p(10)$	$p(11)$

For the purpose of accuracy testing we need a discrete value

Assign label $\hat{y}_i$ if $p_i(y_i|\vec{0}) \geq p_i(-y_i|\vec{0})$

$$\hat{y}_i \in Y$$

Common loss functions

$\in [-1, 1]$

Mean-squared error (MSE)

$$L_{MSE} = \frac{1}{|S|} \sum_{i=1}^{|S|} \left(\tilde{y}_i(\vec{\theta}) - y_i \right)^2$$

Negative log-likelihood

$$L_{log} = -\frac{1}{|S|} \sum_{i=1}^{|S|} \log\left(p_i(y_i | \vec{\theta}) \right)$$

(y_i: correct label)