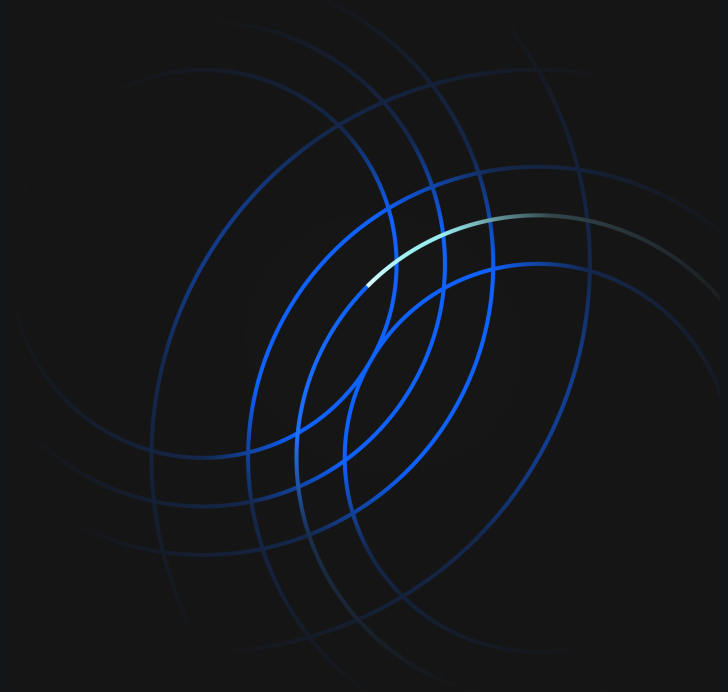
Quantum Support Vector Machines
QML Summer School: Week 2, Day 1

Bryce Fuller
Quantum Computing Applications Researcher



Today's Objectives

Qiskit

Lecture 1

(Part 1) Linear Classifiers

- The "Kernel Trick"

(Part 2) Quantum feature Maps

- Variational Quantum Classifier
- Quantum Kernel Estimator

Lecture 2 (Kristan)

- What is our quantum feature space?
- What are some necessary conditions for advantage?

Supplementary Information

Qiskit

CS229 Lecture notes

Andrew Ng

Part V
Support Vector Machines

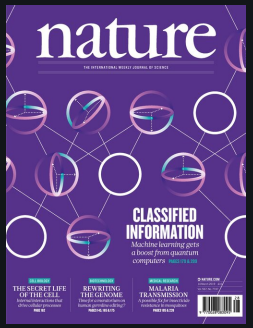
This set of notes presents the Support Vector Machine (SVM) learning algorithm. SVMs are among the best (and many believe is indeed the best) "off-the-shelf" supervised learning algorithm. To tell the SVM story, we'll need to first talk about margins and the idea of separating data with a large "gap." Next, we'll talk about the optimal margin classifier, which will lead us into a digression on Lagrange duality. We'll also see kernels, which give a way to apply SVMs efficiently in very high dimensional (such as infinite-dimensional) feature spaces, and finally, we'll close off the story with the SMO algorithm, which gives an efficient implementation of SVMs.

Letter | Published: 13 March 2019

Supervised learning with quantum-enhanced feature spaces

Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow & Jay M. Gambetta

Nature **567**, 209–212 (2019) | Cite this article



Part One

- Linear Classifiers
- The “Kernel Trick”

Linear Classifiers

Qiskit

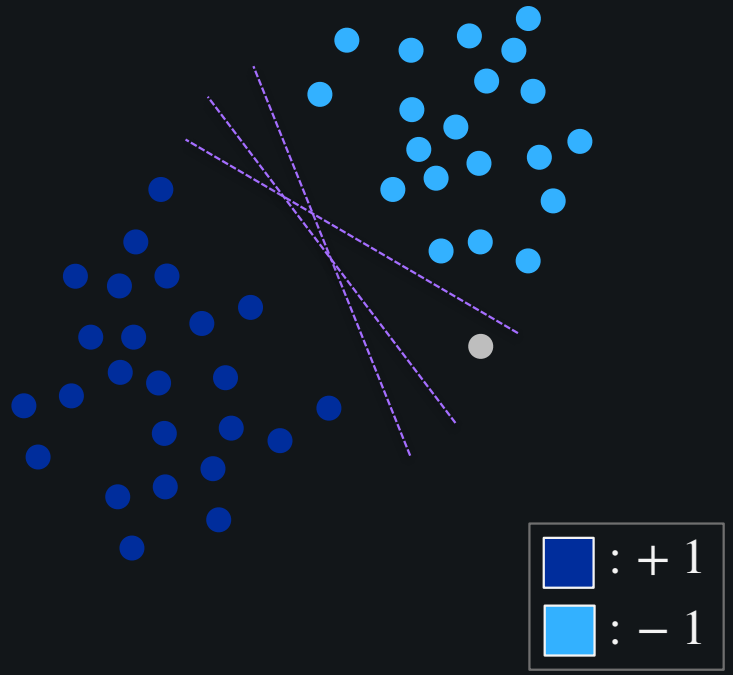
Given a set of labeled points

$$(x_i, y_i),\ y_i \in \{-1, +1\}$$

We want to predict the label of an unseen point

• • •

→ Find a hyperplane which separates our labeled data. Use this as our decision rule.



Linear Classifiers

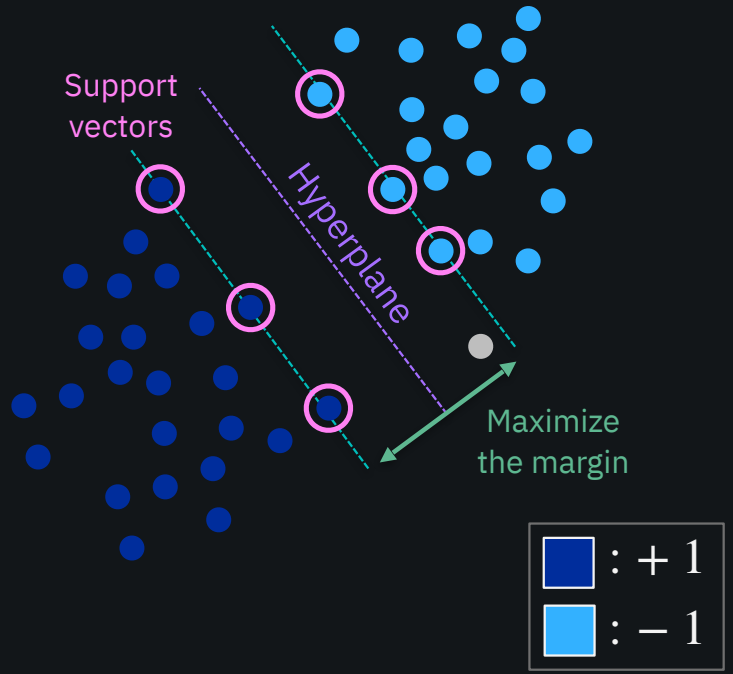
Given a set of labeled points

$$(x_i, y_i),\ y_i \in \{-1, +1\}$$

We want to predict the label of an unseen point

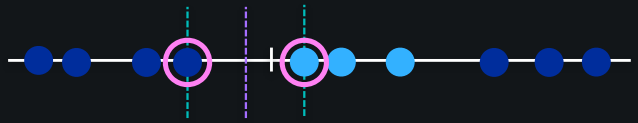
• • •

→ Find a hyperplane which separates our labeled data. Use this as our decision rule.



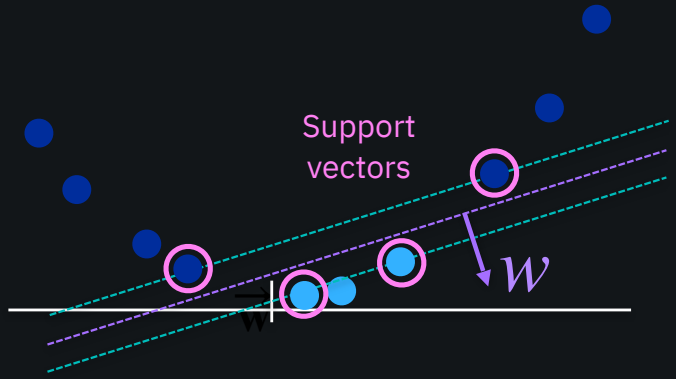
Linear Classifiers

Non-linearly separable datasets may become linearly separable by **including new features**.



Hyperplane: $w^T x + b = 0$

This transformation is called a **feature map**.



Hyperplane: $w^T \Phi(x) + b = 0$

Linear Classifiers

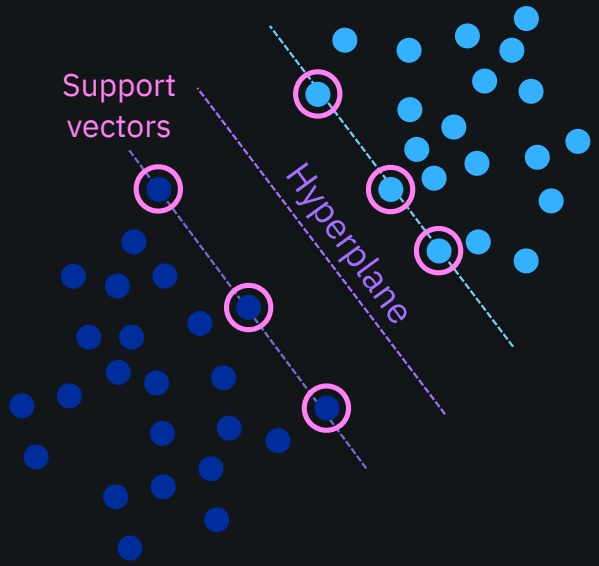
Our decision rule can be written as:

$$\text{label}(x) = \text{sign}(w^T x + b)$$

If we use a feature map Φ, this becomes

$$\text{label}(x) = \text{sign}(w^T \Phi(x) + b)$$

How do we find the hyperplane w ?



Solving for the optimal separating hyperplane

Qiskit

Penalize for each datapoint which falls on the wrong side of w

$$\min_{\alpha,w,b} L_P = \frac{\|w\|^2}{2} - \sum_{i \in T} \alpha_i \left[y_i \left(w^\intercal \Phi(x_i) + b \right) - 1 \right]$$

Lagrange multipliers

Class labels

We can recast this into an alternative, dual form by solving:

$$\frac{\partial L_p}{\partial w} = 0, \quad \frac{\partial L_p}{\partial b} = 0,$$

Recasting our optimization problem

Qiskit

Primal Problem

$$\min_{\alpha,w,b} \; L_P = \frac{\|w\|^2}{2} - \sum_{i \in T} \alpha_i \left[y_i \left(w^\intercal \Phi(x_i) + b \right) - 1 \right]$$

Lagrange multipliers

Dual Problem

$$\max_{\alpha} \; L_D(\alpha) = \sum_{i \in T} \alpha_i - \frac{1}{2} \sum_{i,j \in T} y_i y_j \alpha_i \alpha_j \; \Phi(x_i)^\intercal \Phi(x_j)$$

See https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf for a more detailed treatment

Advantages of the Dual Form

Qiskit

$$\max_{\alpha} \; L_D(\alpha) = \sum_{i \in T} \alpha_i - \frac{1}{2} \sum_{i,j \in T} y_i y_j \alpha_i \alpha_j \underbrace{\Phi(x_i)^\intercal \Phi(x_j)}_{K_{i,j}}$$

Class labels

Lagrange multipliers

The dual form is useful because **dot products** can be replaced by a **kernel** function that implicitly encodes the feature map.

This is called the “kernel trick.”

The Kernel Trick

$$\max_{\alpha} \; L_D(\alpha) = \sum_{i \in T} \alpha_i - \frac{1}{2} \sum_{i,j \in T} y_i y_j \alpha_i \alpha_j \; K_{i,j}$$

$$K_{i,j} = K(x_i, x_j) = \Phi(x_i)^{\intercal} \Phi(x_j)$$

Solution:

Set of $\{\alpha_i\}$ corresponding to the **support vectors** in the training set.

A new data point is labeled using :

$$\text{label}(s) = \text{sign}\left(\sum_{i \in N_s} \alpha_i y_i K\left(x_i, s\right) + b \right)$$

Why is this useful?

Qiskit

Consider the Radial Basis Function Kernel

$$K_{i,j}^{RBF} = e^{\frac{\|x_i - x_j\|}{2\sigma^2}} = \Phi^{RBF}(x_i)^\intercal \Phi^{RBF}(x_i)$$

This can be computed in $\mathcal{O}(\dim(x_i))$

The corresponding feature vectors are **infinite dimensional.**

What should I take away from this?

Qiskit

For a linear classifier of the form:

label(x) = sign($w^{\intercal}\Phi(x) + b$)

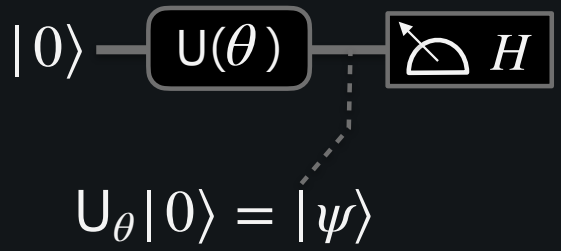
If $K_{i,j}$ is efficiently computable, we can efficiently solve the dual form of our problem.

Part Two

- Quantum Feature Maps
- Support Vector Classifiers
- Quantum Kernel Estimators

Qiskit

Review: Measuring Observables on a QC



Review: Measuring Observables on a QC

If we know the eigenvalues of H, we can assign an energy to each bit string we measure, and then take statistics!

Review: Measuring Observables on a QC

Qiskit

Problem:

In general, implementing the W rotation takes exponential time.

We typically do not have information about the eigenvalues of H

Solution:

Decompose H into a sum of simple observables! (Paulis)

$$H = \sum_i c_i P_i \qquad P_i \in \{I, X, Y, Z\}^{\otimes n}$$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Review: Measuring Observables on a QC

Qiskit

Problem:

In general, implementing the W rotation takes exponential time.

We typically do not have information about the eigenvalues of H

Solution:

Decompose H into a sum of simple observables! (Paulis)

$$H = \sum_i c_i P_i \qquad P_i \in \{I, X, Y, Z\}^{\otimes n}$$

$$\langle\psi|H|\psi\rangle = \sum_i c_i \langle\psi|P_i|\psi\rangle$$

These are easy to compute!

Qiskit

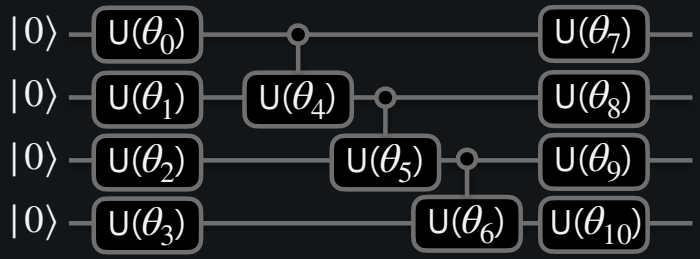
What is Quantum Feature Map?

Variational Quantum Classifier

Qiskit

We need:

- A way to feed our data into the circuit.
- A way to assign {+1, -1} labels



Variational Quantum Classifier

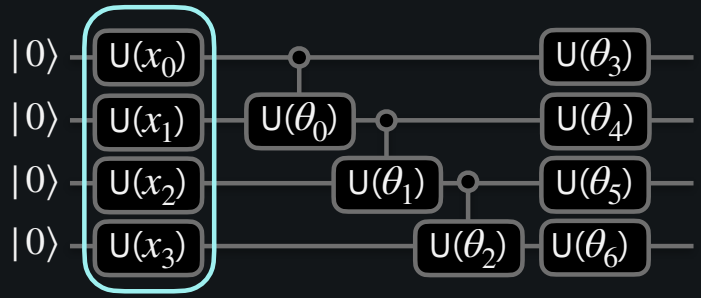
Qiskit

We need:

- A way to feed our data into the circuit.
- A way to assign {+1, -1} labels

First Idea:

- Feed our data into some of the parameterized gates



Variational Quantum Classifier

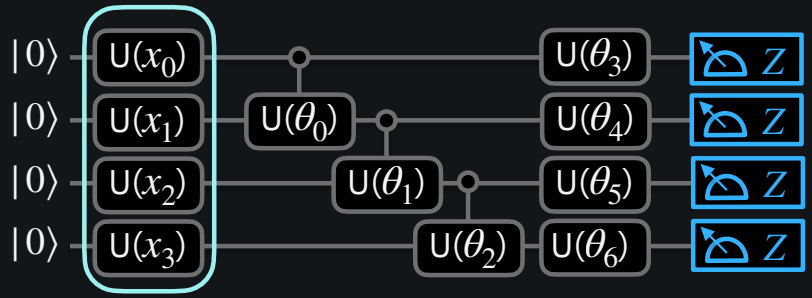
Qiskit

We need:

- A way to feed our data into the circuit.
- A way to assign {+1, -1} labels

First Idea:

- Feed our data into some of the parameterized gates
- Measure a Pauli observable to assign our label



Variational Quantum Classifier

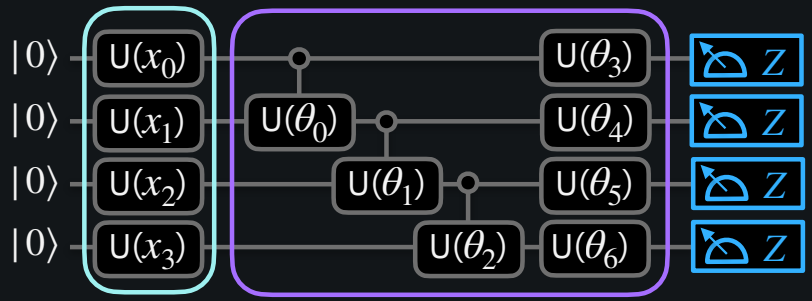
Qiskit

We need:

- A way to feed our data into the circuit.
- A way to assign {+1, -1} labels

First Idea:

- Feed our data into some of the parameterized gates
- Measure a Pauli observable to assign our label
- Optimize the remaining parameters to improve performance



Variational Quantum Classifier

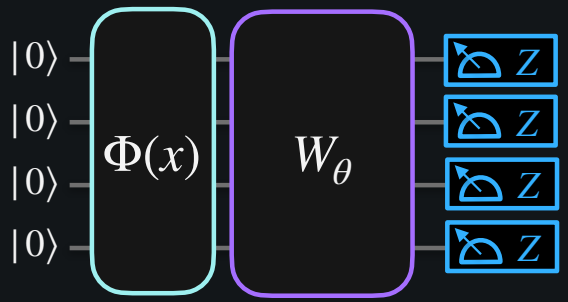
Qiskit

We need:

- A way to feed our data into the circuit.
- A way to assign {+1, -1} labels

First Idea:

- Feed our data into some of the parameterized gates
- Measure a Pauli observable to assign our label
- Optimize the remaining parameters to improve performance



$$f_\theta(x) = \langle \Phi(x) | W_\theta^\dagger Z W_\theta | \Phi(x) \rangle$$

$$\in [-1,1]$$

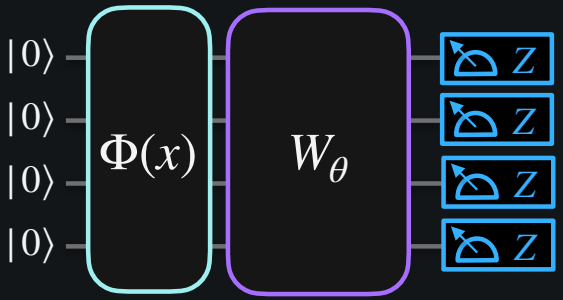
Choose a threshold $b \in [-1,1]$

$$\text{label}(x) = \begin{cases} +1, & \text{if } f_\theta(x) \geq b \\ -1, & \text{if } f_\theta(x) < b \end{cases}$$

What is a quantum feature map?

Qiskit

Variational Quantum Classifier



$$f_\theta(x) = \langle \Phi(x) | W_\theta^\dagger Z W_\theta | \Phi(x) \rangle$$

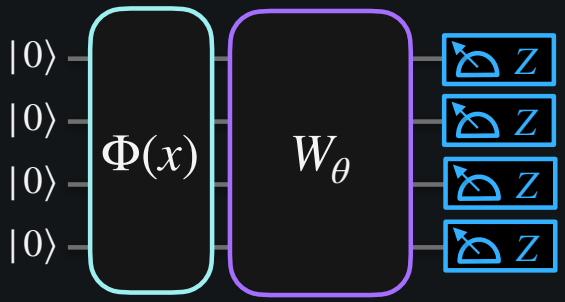
Choose a threshold $b \in [-1,1]$

$$\text{label}(x) = \begin{cases} +1, & \text{if } f_\theta(x) \geq b \\ -1, & \text{if } f_\theta(x) < b \end{cases}$$

What is a quantum feature map?

Qiskit

Variational Quantum Classifier



This is our feature map

What exactly is W_θ doing?

$$f_\theta(x) = \langle \Phi(x) | \underbrace{\left(W_\theta^\dagger Z W_\theta \right)}_{H_\theta} | \Phi(x) \rangle$$

Choose a threshold $b \in [-1,1]$

$$\text{label}(x) = \text{sign}\left(f_\theta(x) + b \right)$$

CLAIM:
This model is just a linear classifier!

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \langle \Phi(x) | W_\theta^\dagger Z W_\theta | \Phi(x) \rangle$$ Define: $H_\theta = W_\theta^\dagger Z W_\theta$

$$= \langle \Phi(x) | H_\theta | \Phi(x) \rangle$$ For a scalar c, $\text{Tr}[c] = c$

$$= \text{Tr}\left[\langle \Phi(x) | H_\theta | \Phi(x) \rangle \right]$$ $\text{Tr}[AB] = \text{Tr}[BA]$

$$= \text{Tr}\left[H_\theta | \Phi(x) \rangle \langle \Phi(x) | \right]$$ Define: $\Phi(x) = | \Phi(x) \rangle \langle \Phi(x) |$

$\in \mathbb{C}^{2^n, 2^n}$

$$= \text{Tr}\left[H_\theta \Phi(x) \right]$$

Next, let's rewrite H_θ, $\Phi(x)$

Proof: VQC is a Linear Classifier

Qiskit

$$H_\theta = \frac{1}{2^n} \sum_{\alpha \in 4^n} \langle H_\theta, P_\alpha \rangle \, P_\alpha$$

Pauli Matrices

Hilbert-Schmidt inner product

Proof: VQC is a Linear Classifier

Qiskit

$$H_\theta = \frac{1}{2^n} \sum_{\alpha \in 4^n} \text{Tr}\left[H_\theta P_\alpha\right] P_\alpha$$

Define: $h_\alpha(\theta) = \text{Tr}\left[H_\theta P_\alpha\right]$

$$H_\theta = \frac{1}{2^n} \sum_{\alpha \in 4^n} h_\alpha(\theta) P_\alpha$$

$$\Phi(x) = \frac{1}{2^n} \sum_{\alpha \in 4^n} \text{Tr}\left[\Phi(x) P_\alpha\right] P_\alpha$$

Define: $\Phi_\alpha(x) = \text{Tr}\left[\Phi(x) P_\alpha\right]$

$$\Phi(x) = \frac{1}{2^n} \sum_{\alpha \in 4^n} \Phi_\alpha(x) P_\alpha$$

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \mathrm{Tr}\left[\left(\frac{1}{2^n}\sum_{\alpha\in 4^n} h_\alpha(\theta) P_\alpha\right)\left(\frac{1}{2^n}\sum_{\beta\in 4^n} \Phi_\beta(x) P_\beta\right)\right]$$

Decompose H_θ, $\Phi(x)$ into the Pauli basis

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{4^n}\mathrm{Tr}\left[\left(\sum_{\alpha \in 4^n} h_\alpha(\theta) P_\alpha\right)\left(\sum_{\beta \in 4^n} \Phi_\beta(x) P_\beta\right)\right]$$

Bring normalization out front

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{4^n}\mathrm{Tr}\left[\sum_{\alpha,\beta \in 4^n} h_\alpha(\theta)\Phi_\beta(x)\ P_\alpha P_\beta\right]$$

Simplify double sum

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \text{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{4^n} \sum_{\alpha,\beta \in 4^n} h_\alpha(\theta)\Phi_\beta(x)\ \text{Tr}\left[P_\alpha P_\beta\right]$$

Distribute Trace operation

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \text{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{4^n} \sum_{\alpha,\beta \in 4^n} h_\alpha(\theta)\Phi_\beta(x)\ \text{Tr}\left[P_\alpha P_\beta\right]$$

Observe that: $\text{Tr}\left[P_\alpha, P_\beta\right] = \begin{cases} 2^n & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \end{cases}$

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \text{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{4^n} \sum_{\alpha,\beta \in 4^n} h_\alpha(\theta)\Phi_\beta(x)\ \text{Tr}\left[P_\alpha P_\beta\right]$$

$$= \frac{1}{2^n} \sum_{\alpha \in 4^n} h_\alpha(\theta)\Phi_\alpha(x)$$

$$\text{Tr}\left[P_\alpha, P_\beta\right] = \begin{cases} 2^n & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \end{cases}$$

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

$$= \frac{1}{2^n} \sum_{\alpha \in 4^n} h_\alpha(\theta) \Phi_\alpha(x)$$

Proof: VQC is a Linear Classifier

Qiskit

$$f_\theta(x) = \mathrm{Tr}\left[H_\theta \Phi(x)\right]$$

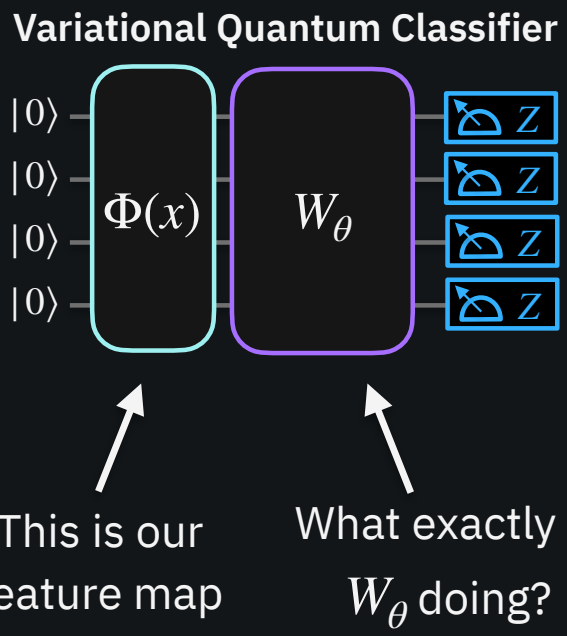
$$= \frac{1}{2^n} \sum_{\alpha \in 4^n} h_\alpha(\theta) \Phi_\alpha(x) \quad \in [-1,1]$$

Choose a threshold $b \in [-1,1]$

$$\text{label}(x) = \text{sign}\left(\frac{1}{2^n} \sum_{\alpha \in 4^n} h_\alpha(\theta) \Phi_\alpha(x) \; + \; b \right)$$

Implications: VQC is a Linear Classifier

Qiskit



W_θ parameterizes only a subset of possible hyperplanes.

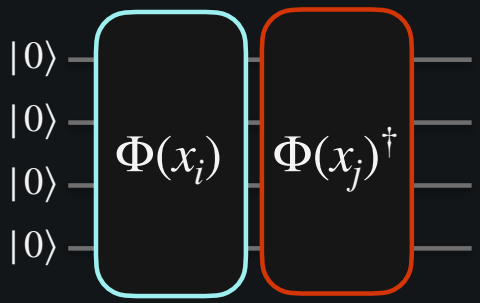
Reaching an arbitrary hyperplane is equivalent to measuring an arbitrary Observable
→ requires an exp-depth change-of-basis circuit.

☆ If we can directly compute our Kernel function, we can use the "Kernel Trick" to efficiently solve for the optimal hyperplane!

Estimating $K_{i,j}$ on a Quantum Computer

Qiskit

Quantum Kernel Estimator



For $i,j \in$ Training set:

- Prepare $\Phi(x_j)^\dagger \Phi(x_i)|0\rangle$
- Let $K_{ij} = \Pr[\text{measure } |0\rangle]$
- Plug K_{ij} into Dual form $L_D(\alpha)$ & solve.
- Return $\{\alpha_i\}$

$$\Pr[\text{measure } |0\rangle] = |\langle 0|\Phi(x_j)^\dagger \Phi(x_i)|0\rangle|^2$$

$$K_{ij} = |\langle \Phi(x_j)|\Phi(x_i)\rangle|^2$$

$$\text{Label}(s) = \text{Sign}\left(\sum_{i \in N_s} \alpha_i K(x_i, s) + b\right)$$

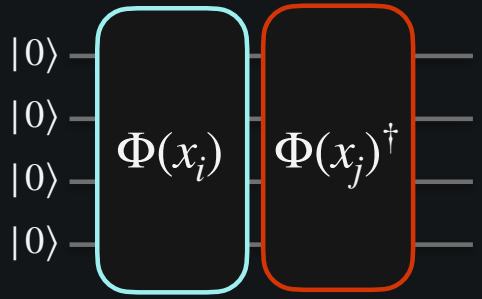
Take-Aways on QKE

Qiskit

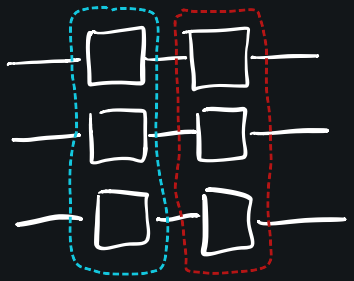
VQC ⊂ QKE

QKE := VQC with an optimally solved, ∞-depth W_θ

Quantum Kernel Estimator



$\Phi(x)$ should be hard to compute classically.



efficiently simulable.