Quantum Machine Learning

QSVM, QNN and other methods

9 July 2026 I Lourens van Niekerk

Member of the Helmholtz Association

JÜLICH
Forschungszentrum
Shaping Change

Motivation for QML

Quantum is cool, what more do you need?

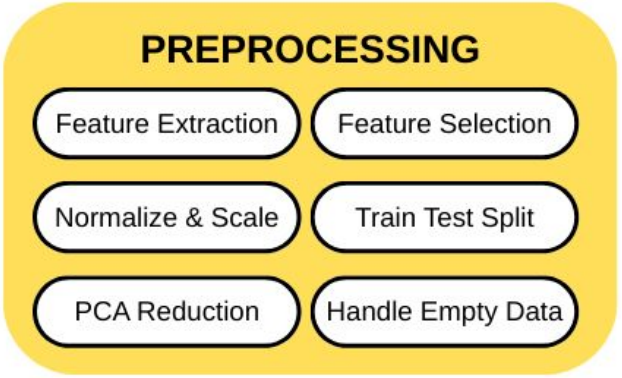
- What are the **potentials**?
  - **Faster** ML
  - More **accurate** models
  - **Cheaper** ML
  - **Innovative** thinking
- What is **here**?
  - Expanding **hardware & software** resources
  - Accelerating **research & funding**
- What is on the **horizon**?
  - Quantum **utility** in tested areas
  - **Accessibility** to the public



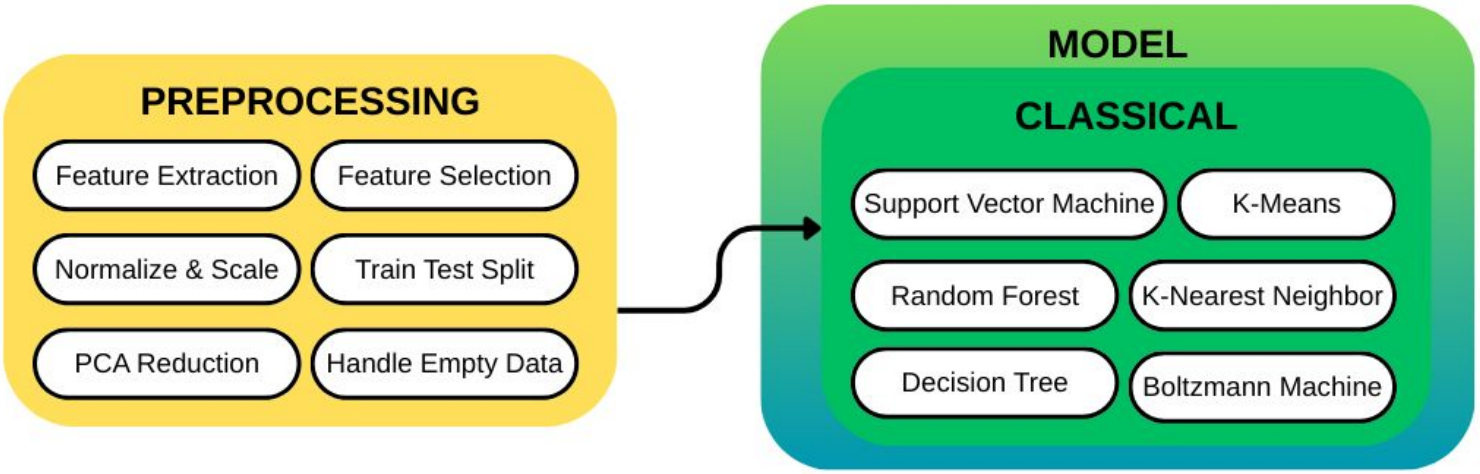




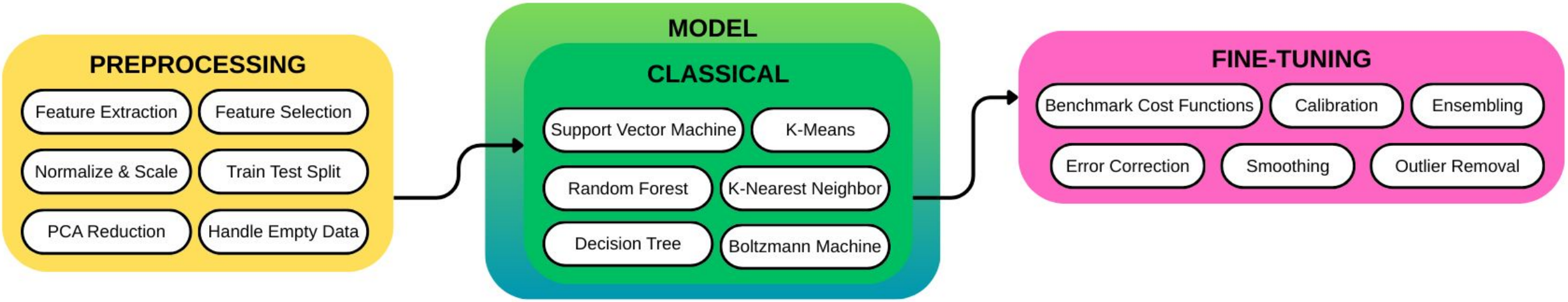

The Machine Learning Pipeline



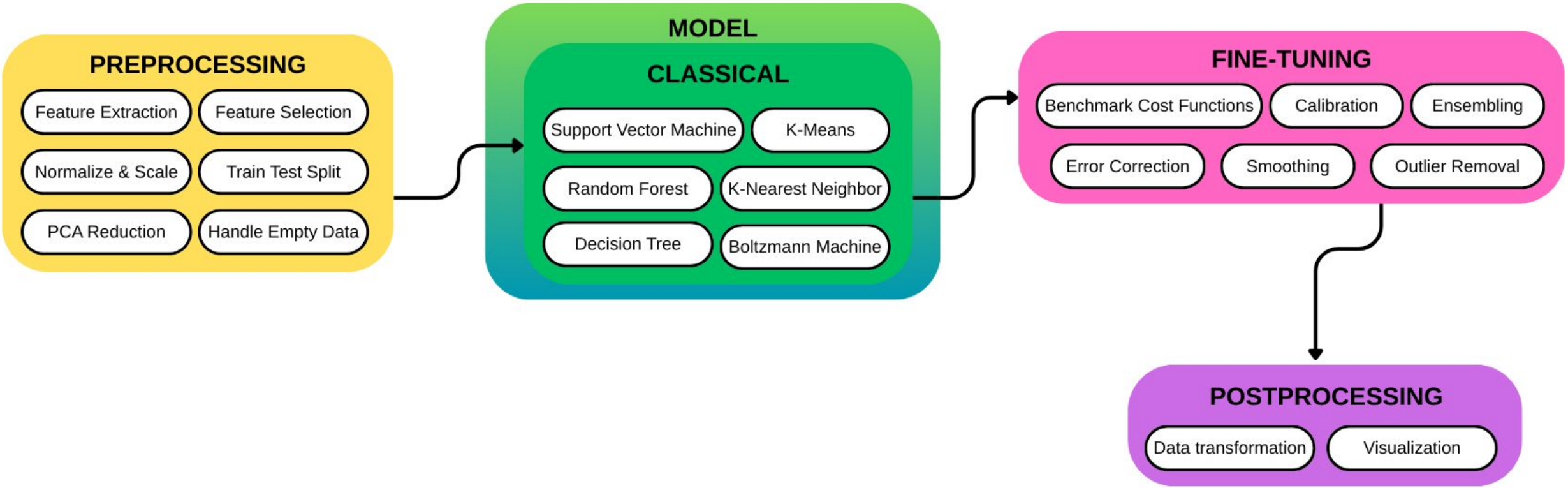
The Machine Learning Pipeline



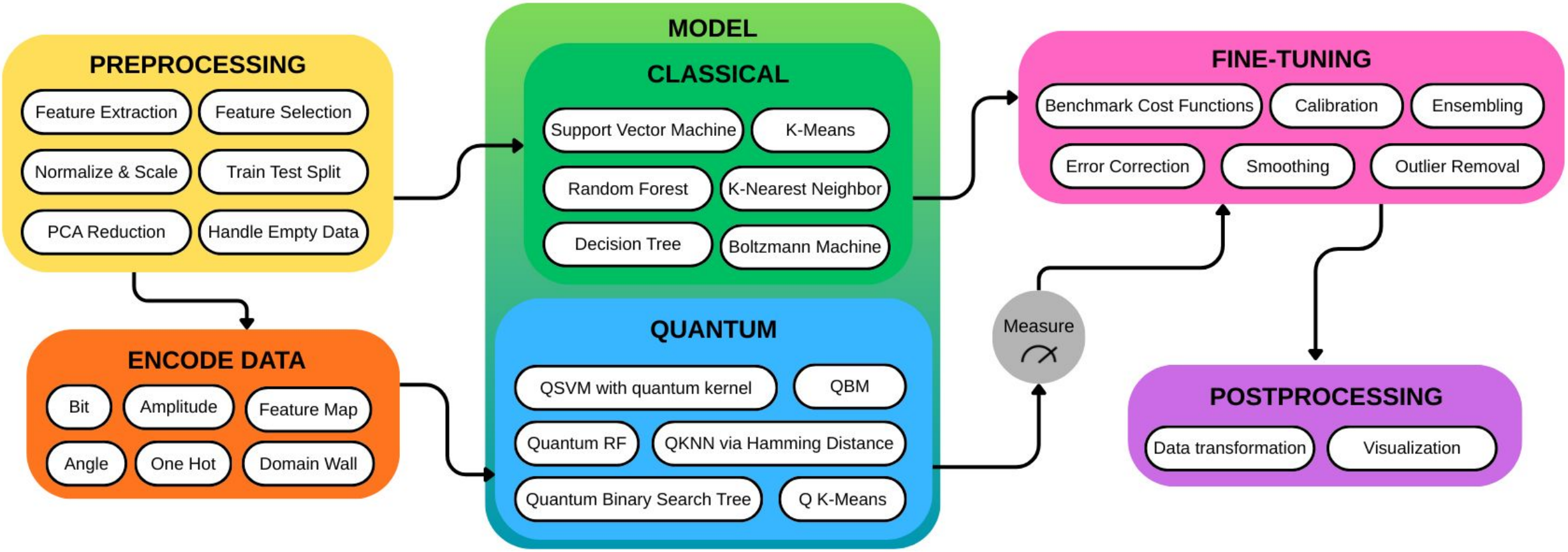
The Machine Learning Pipeline



The Machine Learning Pipeline



The Machine Learning Pipeline

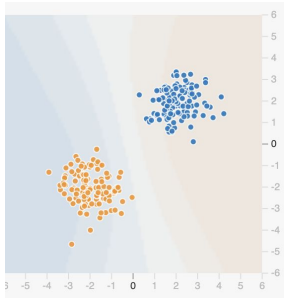


Types of Learning

Unsupervised Learning

Model discovers info independently.
Suitable for unpredictable data.
e.g. K-Means clustering.
μ_i are k-many centroids.

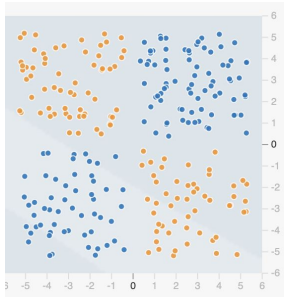
$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$



No color?
No problem!

Supervised Learning

Model knows what the labels are.
Usually better results than unsupervised.
e.g. Support Vector Machines.



No color?
Problem!

Semi-Supervised Learning

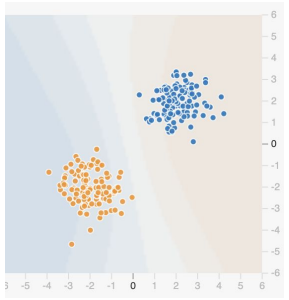
Reinforcement Learning

Types of Learning

Unsupervised Learning


Model discovers info independently.
Suitable for unpredictable data.
e.g. K-Means clustering.
μ_i are k-many centroids.

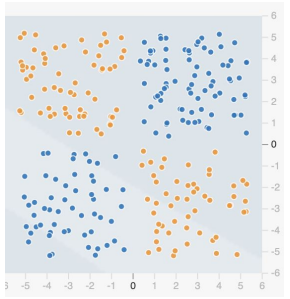
$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$



No color?
No problem!

Supervised Learning

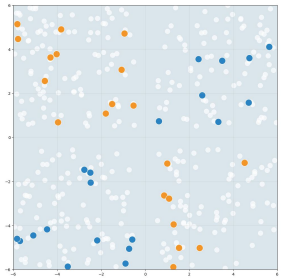

Model knows what the labels are.
Usually better results than unsupervised.
e.g. Support Vector Machines.



No color?
Problem!

Semi-Supervised Learning


Labelled & unlabelled (20/80).
Common since labelling difficult.
e.g. semi-supervised SVM (S3VM).
Additional cost to avoid dense unlabelled data.



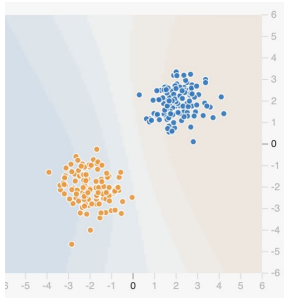
Reinforcement Learning

Types of Learning

Unsupervised Learning

Model discovers info independently.
Suitable for unpredictable data.
e.g. K-Means clustering.
μ_i are k-many centroids.

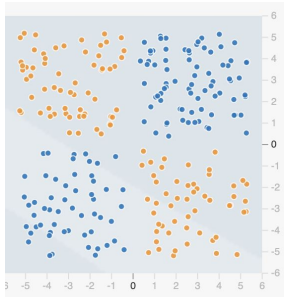
$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$



No color?
No problem!

Supervised Learning

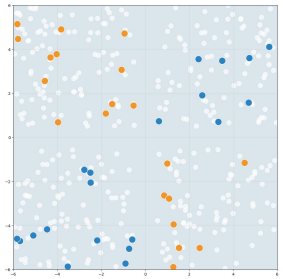
Model knows what the labels are.
Usually better results than unsupervised.
e.g. Support Vector Machines.



No color?
Problem!

Semi-Supervised Learning

Labelled & unlabelled (20/80).
Common since labelling difficult.
e.g. semi-supervised SVM (S3VM).
Additional cost to avoid dense unlabelled data.



Reinforcement Learning

Model incentivized to do a task.
Evaluate actions on states iteratively.
e.g. Q-Learning.

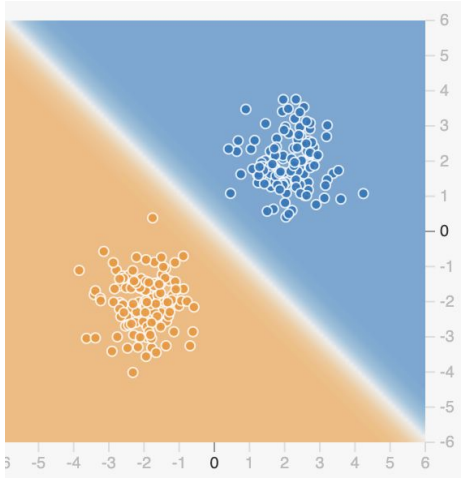
teach a dog to give paw

Support Vector Machines (SVMs)

Classifier vs Regressor

Classifier

Discrete label assignment.
False prediction more costly.
Thus easier to identify and correct.
Learns decision boundary
to separate classes.



Regressor

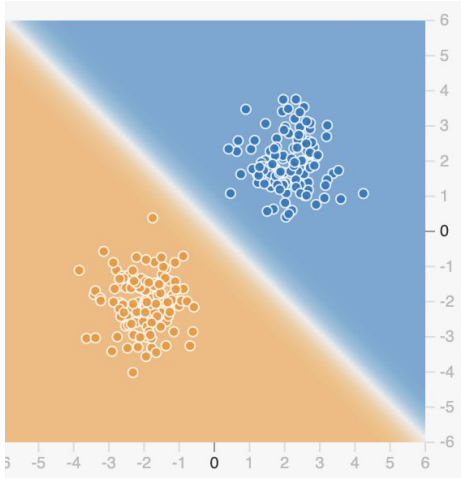
Support Vector Machines (SVMs)

Classifier vs Regressor

Classifier

Discrete label assignment.
False prediction more costly.
Thus easier to identify and correct.
Learns decision boundary
to separate classes.
α_i determines decision boundary.

α_i multipliers
$\vec{d}_i$ data point
t_i labels in $\{-1, 1\}$
$k(\cdot)$ kernel function
C regularization constant



$$\min_{\{\alpha_i\}} \frac{1}{2} \sum_{n,m=0}^{N-1} \alpha_n \alpha_m t_n t_m k(\vec{d}_n, \vec{d}_m) - \sum_{n=0}^{N-1} \alpha_n$$

$$\text{subject to } 0 \leq \alpha_n \leq C$$

$$\text{and } \sum_{n=0}^{N-1} \alpha_n t_n = 0$$

Regressor

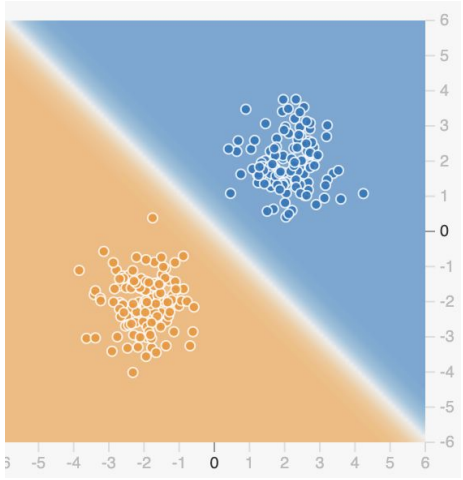
Support Vector Machines (SVMs)

Classifier vs Regressor

Classifier

Discrete label assignment.
False prediction more costly.
Thus easier to identify and correct.
Learns decision boundary
to separate classes.
α_i determines decision boundary.

α_i multipliers
$\vec{d}_i$ data point
t_i labels in $\{-1, 1\}$
$k(\cdot)$ kernel function
C regularization constant



$$\min_{\{\alpha_i\}} \frac{1}{2} \sum_{n,m=0}^{N-1} \alpha_n \alpha_m t_n t_m k(\vec{d}_n, \vec{d}_m) - \sum_{n=0}^{N-1} \alpha_n$$

$$\text{subject to } 0 \leq \alpha_n \leq C$$

$$\text{and } \sum_{n=0}^{N-1} \alpha_n t_n = 0$$

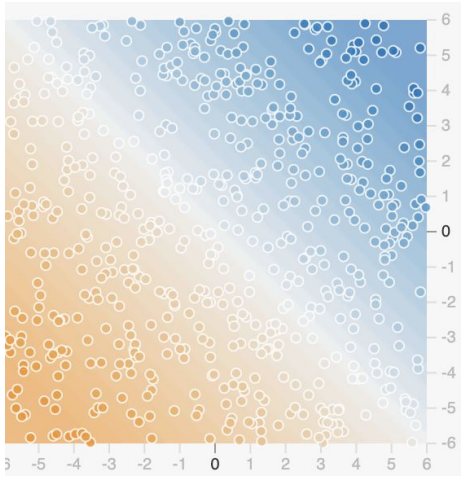
Regressor

Continuous label assignment
Double the parameters to optimize.
Exposes relationships of labels.
Learns target function
to assign probabilities.

α_i, α_i^* multipliers
C, ϵ regularization constants

$$\min_{\{\alpha_n\},\{\alpha_n^*\}} \frac{1}{2} \sum_{n,m=0}^{N-1} (\alpha_n - \alpha_n^*)(\alpha_m - \alpha_m^*) k(\vec{d}_n, \vec{d}_m)$$

$$+ \epsilon \sum_{n=0}^{N-1} (\alpha_n + \alpha_n^*) - \sum_{n=0}^{N-1} t_n (\alpha_n - \alpha_n^*)$$



$$\text{subject to } \sum_{n=0}^{N-1} (\alpha_n - \alpha_n^* = 0)$$

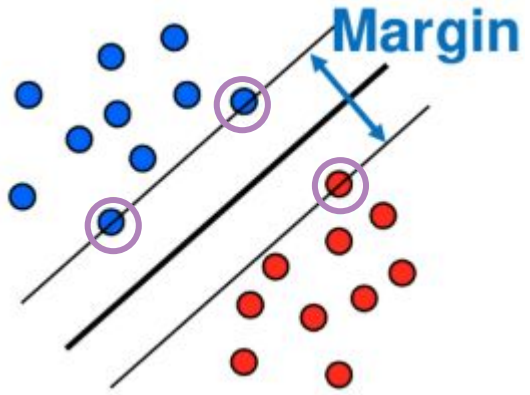
$$0 \leq \alpha_n \leq C$$

$$0 \leq \alpha_n^* \leq C$$

Kernel Trick in SVM

When a 2D line won't cut it

Easy!

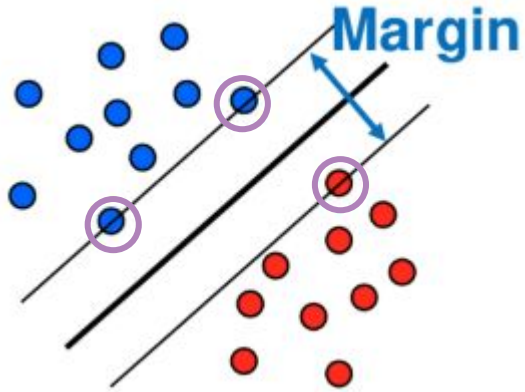


- Decision boundary cuts through plane, based on support vectors

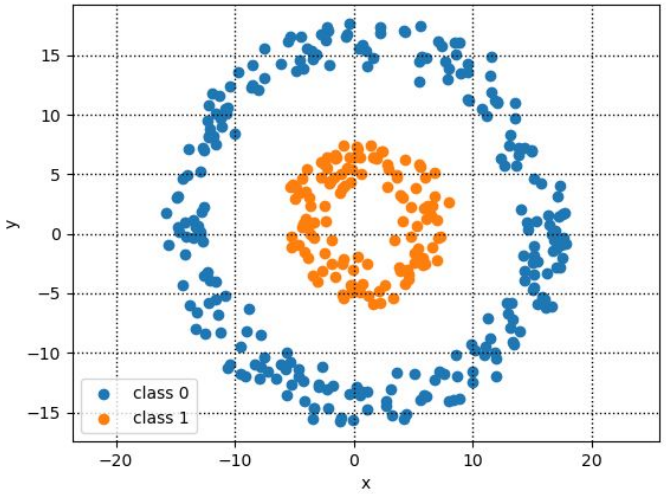
Kernel Trick in SVM

When a 2D line won't cut it

Easy!



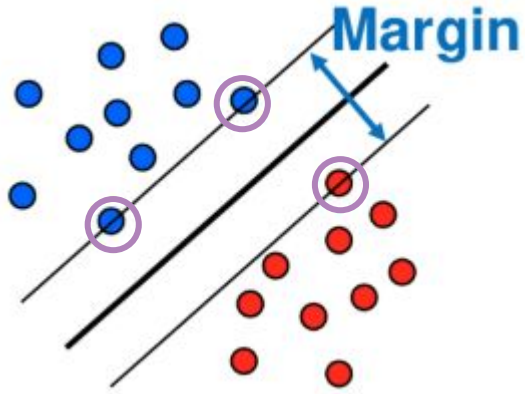
- Decision boundary cuts through plane, based on support vectors
- Sometimes not so straight-forward



Kernel Trick in SVM

When a 2D line won't cut it

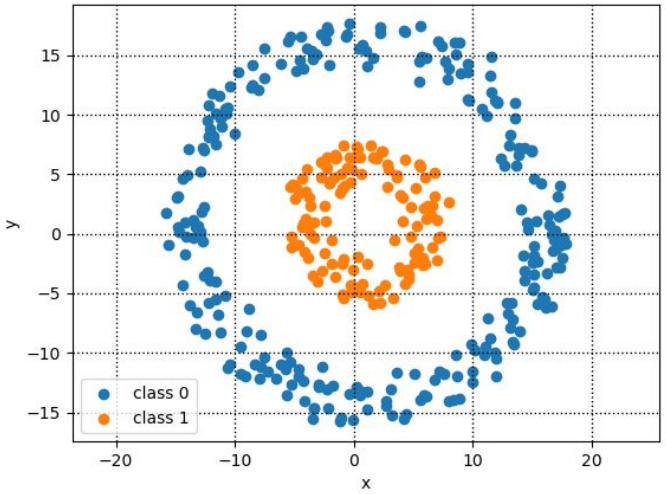
Easy!



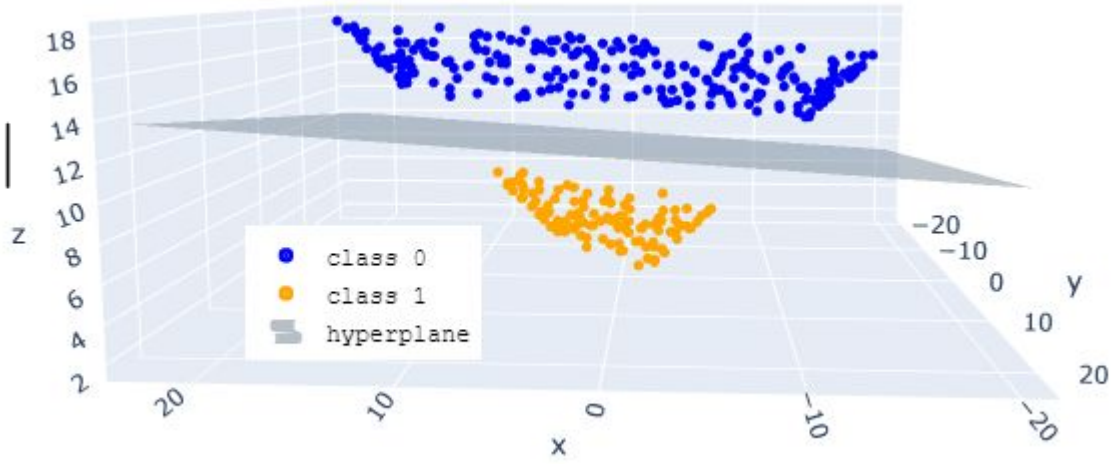
- Decision boundary cuts through plane, based on support vectors
- Sometimes not so straight-forward
- Kernel trick

Raise dimension ➡ hyperplane cut ➡ Lower dimension

no need for additional storage!



$$z = |x_i| + |y_i|$$

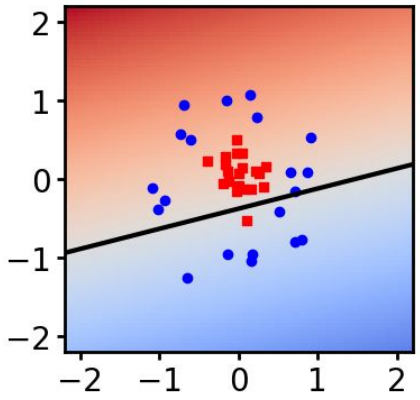


Types of Kernels

Kernel: function comparing data points along higher dimension

Linear

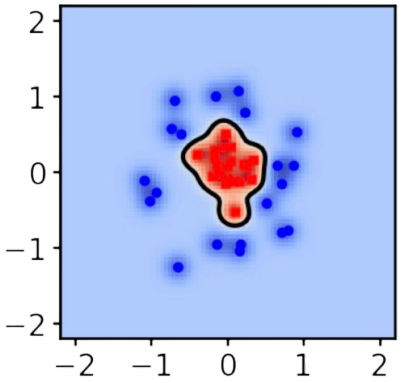
- Linear separability
- Faster calculation
- Often least applicable



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

Radial Basis Function (RBF)

- Most popular (default)
- γ handles range of point influence
- Suitable for (unknown) complexity



$$K(\vec{x}_i, \vec{x}_j) = e^{-\gamma||\vec{x}_i - \vec{x}_j||^2}$$

Polynomial

Sigmoid

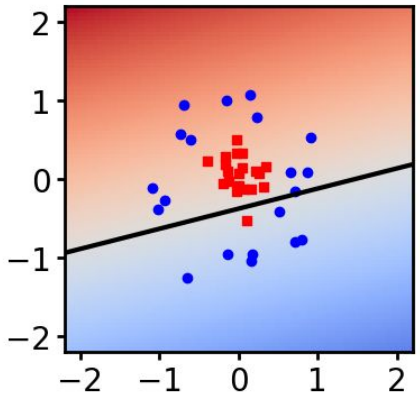
Types of Kernels

Kernel: function comparing data points along higher dimension

Linear

- Linear separability
- Faster calculation
- Often least applicable

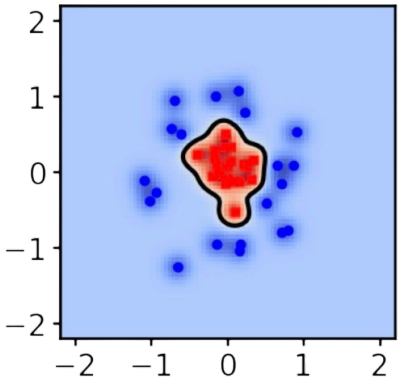
$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$



Radial Basis Function (RBF)

- Most popular (default)
- γ handles range of point influence
- Suitable for (unknown) complexity

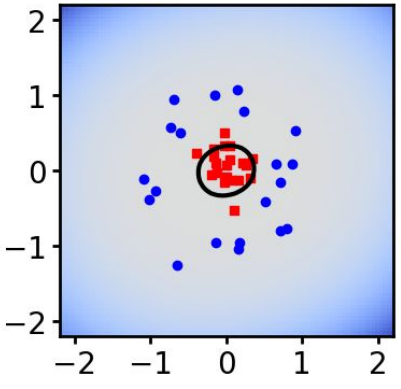
$$K(\vec{x}_i, \vec{x}_j) = e^{-\gamma||\vec{x}_i - \vec{x}_j||^2}$$



Polynomial

- Has additional bias & degree
- Thus adjustable to fit
- Smooth curves in higher dimension

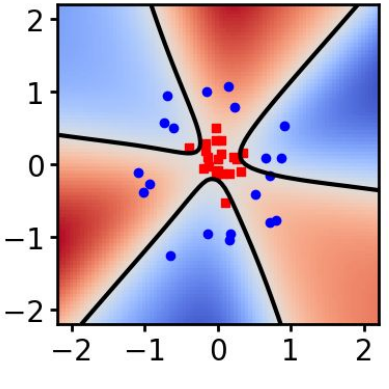
$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + r)^d$$



Sigmoid

- γ controls slope, and *r* bias term
- Resembles neural network activation
- Need a lot of tuning potentially

$$K(\vec{x}_i, \vec{x}_j) = tanh(\gamma(\vec{x}_i \cdot \vec{x}_j) + r)$$



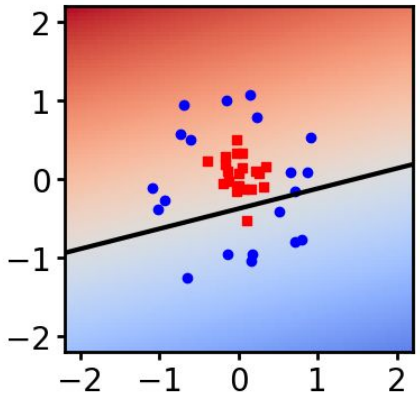
Types of Kernels

Kernel: function comparing data points along higher dimension

Linear

- Linear separability
- Faster calculation
- Often least applicable

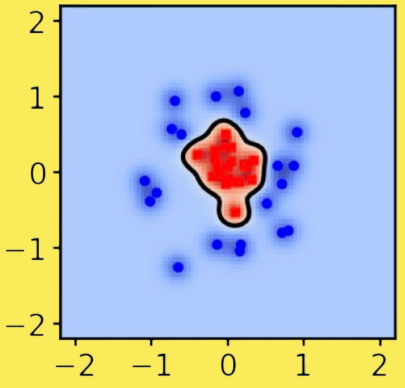
$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$



Radial Basis Function (RBF)

- Most popular (default)
- γ handles range of point influence
- Suitable for (unknown) complexity

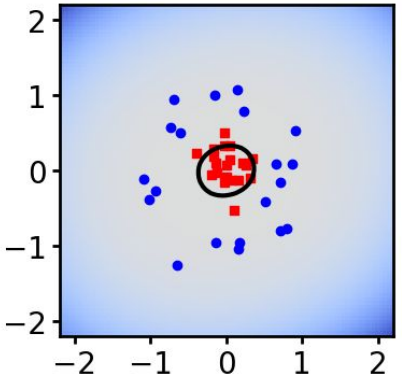
$$K(\vec{x}_i, \vec{x}_j) = e^{-\gamma||\vec{x}_i - \vec{x}_j||^2}$$



Polynomial

- Has additional bias & degree
- Thus adjustable to fit
- Smooth curves in higher dimension

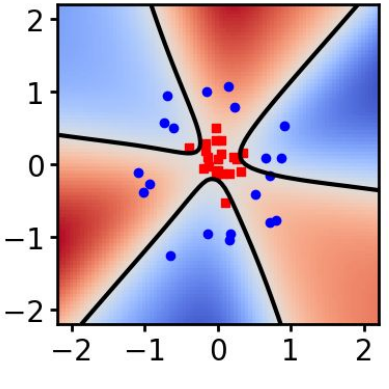
$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + r)^d$$



Sigmoid

- γ controls slope, and *r* bias term
- Resembles neural network activation
- Need a lot of tuning potentially

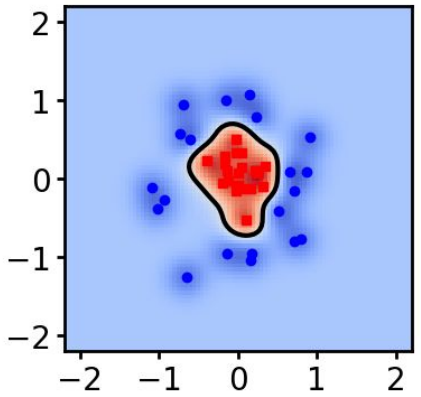
$$K(\vec{x}_i, \vec{x}_j) = tanh(\gamma(\vec{x}_i \cdot \vec{x}_j) + r)$$



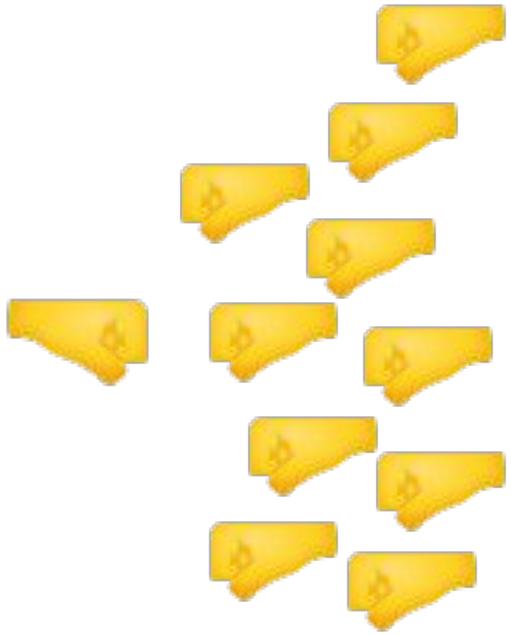
A Glimpse at QSVM

One solution vs many low-energy solutions

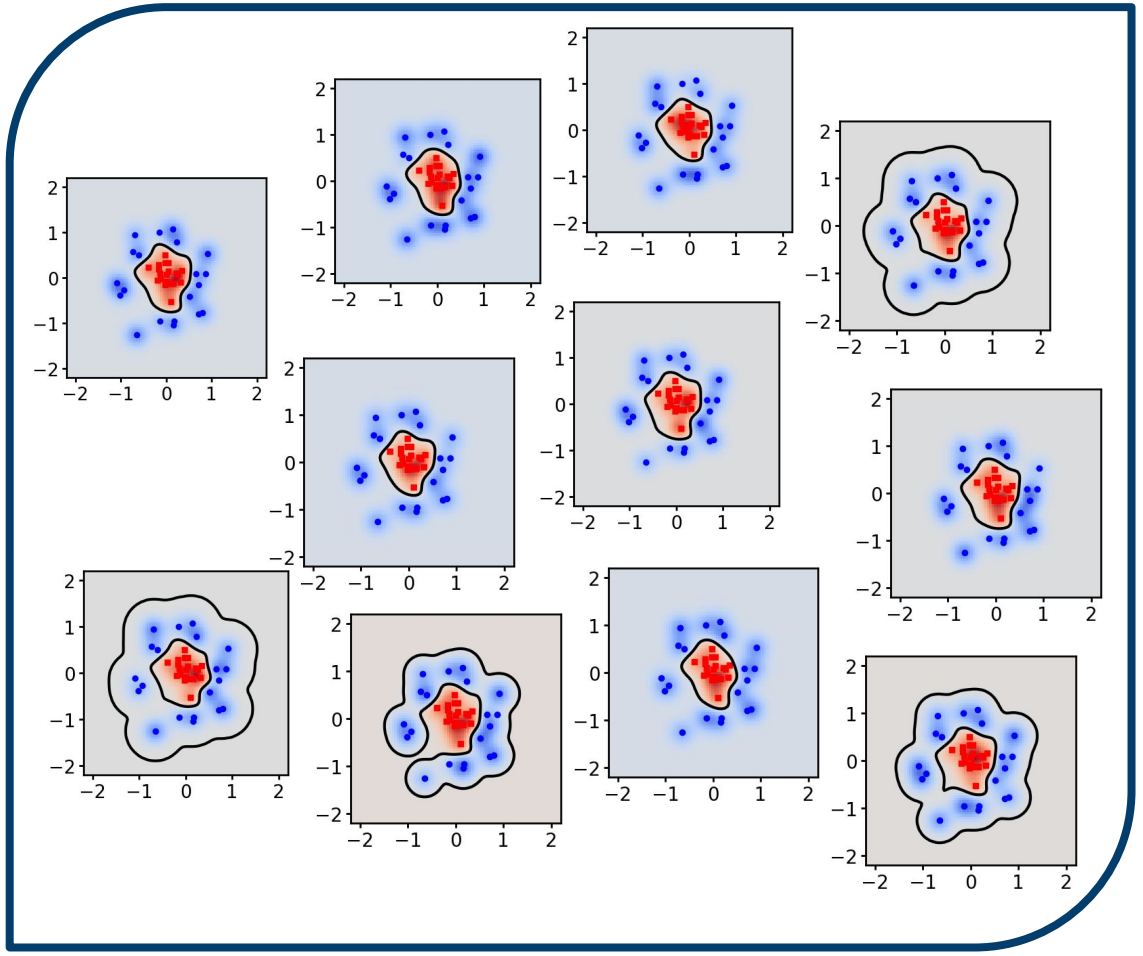
Generally, at least one quantum solution from the ensemble punches harder, when looking at unseen data



Classical



Get the ensemble for free; would need additional planning in classical case



Quantum (annealing)

Pros & Cons

of QSVM and QML in general

PROS	CONS
Potential speedup	Encoding/Decoding bottleneck

Pros & Cons

of QSVM and QML in general

PROS	CONS
Potential speedup	Encoding/Decoding bottleneck
Entanglement opportunity	No guarantee entanglement helps

Pros & Cons

of QSVM and QML in general

PROS	CONS
Potential speedup	Encoding/Decoding bottleneck
Entanglement opportunity	No guarantee entanglement helps
Promising theoretical scaling	Uncertain practical scaling

Pros & Cons

of QSVM and QML in general

PROS	CONS
Potential speedup	Encoding/Decoding bottleneck
Entanglement opportunity	No guarantee entanglement helps
Promising theoretical scaling	Uncertain practical scaling
Novel techniques	Lack of maturity

Pros & Cons

of QSVM and QML in general

PROS	CONS
Potential speedup	Encoding/Decoding bottleneck
Entanglement opportunity	No guarantee entanglement helps
Promising theoretical scaling	Uncertain practical scaling
Novel techniques	Lack of maturity
Higher complexity solving capacity	Harder to tune & understand

Annealing SVC vs Gate-based SVC

Annealing

- Access 4500+ qubits on D-Wave
- Encode as QUBO problem
- Only optimization on quantum device
- Enough physical qubits to make logical qubits

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]}$$

$$\min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

Q is $NK \times NK$ QUBO-matrix.

Ref: D-Wave with qubits allotted for QSVM

Gate-based

Annealing SVC vs Gate-based SVC

Annealing

- Access 4500+ qubits on D-Wave
- Encode as QUBO problem
- Only optimization on quantum device
- Enough physical qubits to make logical qubits

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]}$$

$$\min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

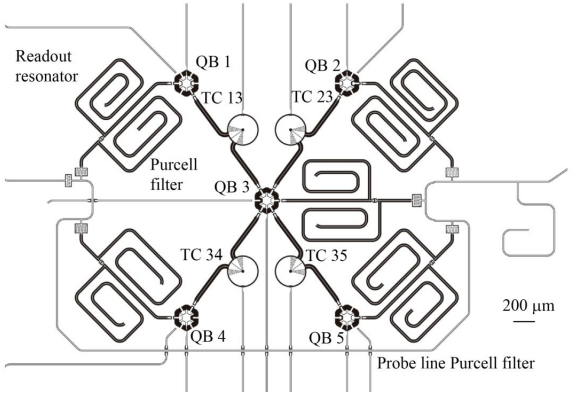
Q is $NK \times NK$ QUBO-matrix.

Ref: D-Wave with qubits allotted for QSVM

Gate-based

- Limited in (quality) qubits
- Can simulate on laptop up to ~30 qubits
- Optimization is (potentially) only classical part

What could we do?



Ref: IQM Spark 5-qubit machine

Annealing SVC vs Gate-based SVC

Annealing

- Access 4500+ qubits on D-Wave
- Encode as QUBO problem
- Only optimization on quantum device
- Enough physical qubits to make logical qubits

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]}$$

$$\min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

Q is $NK \times NK$ QUBO-matrix.

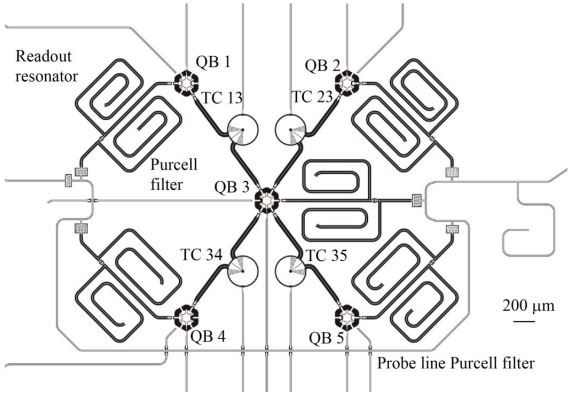
Ref: D-Wave with qubits allotted for QSVM

Gate-based

- Limited in (quality) qubits
- Can simulate on laptop up to ~30 qubits
- Optimization is (potentially) only classical part

What could we do?

Quantum kernel trick	*Quantum kernel itself*
Encode classical data into quantum data on a high dimensional Hilbert space	Use classical-like kernels or try out quantum kernels that use entanglement



Ref: IQM Spark 5-qubit machine

Quantum SVC

QUBO implementation

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]} \quad \& \quad \min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

$$\& \quad Q_{[n,k],[m,j]} = \frac{1}{2} B^{k+j-2P} t_n t_m (k(\vec{d}_n, \vec{d}_m) + \xi) - \delta_{nm}\delta_{kj} B^{k-P}$$

1. Reformulate as QUBO
2. (Optional) reduce ≈ 0 values of Q to just 0

opt(Q)

Quantum SVC

QUBO implementation

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]} \quad \& \quad \min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

$$\& \quad Q_{[n,k],[m,j]} = \frac{1}{2} B^{k+j-2P} t_n t_m (k(\vec{d}_n, \vec{d}_m) + \xi) - \delta_{nm}\delta_{kj} B^{k-P}$$

1. Reformulate as QUBO
2. (Optional) reduce ≈ 0 values of Q to just 0
3. Place QUBO in Binary Quadratic Model (BQM)
4. Embed on D-Wave

opt(Q)

BQM

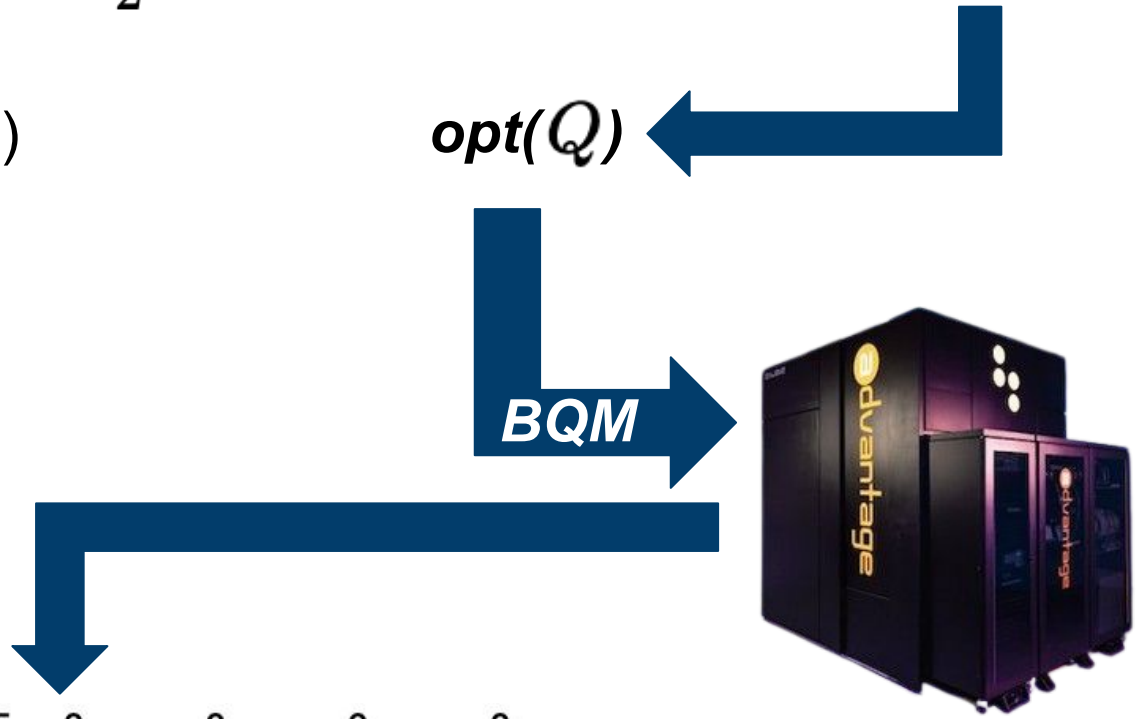
Quantum SVC

QUBO implementation

$$\alpha_n = \sum_{k=0}^{K-1} B^{k-P} x_{[n,k]} \quad \& \quad \min_{\{x_i=0,1\}} \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} x_{[n,k]} Q_{[n,k],[m,j]} x_{[m,j]}$$

$$\& \quad Q_{[n,k],[m,j]} = \frac{1}{2} B^{k+j-2P} t_n t_m (k(\vec{d}_n, \vec{d}_m) + \xi) - \delta_{nm}\delta_{kj} B^{k-P}$$

1. Reformulate as QUBO
2. (Optional) reduce ≈ 0 values of Q to just 0
3. Place QUBO in Binary Quadratic Model (BQM)
4. Embed on D-Wave
5. Run optimization
6. Retrieve lowest energies



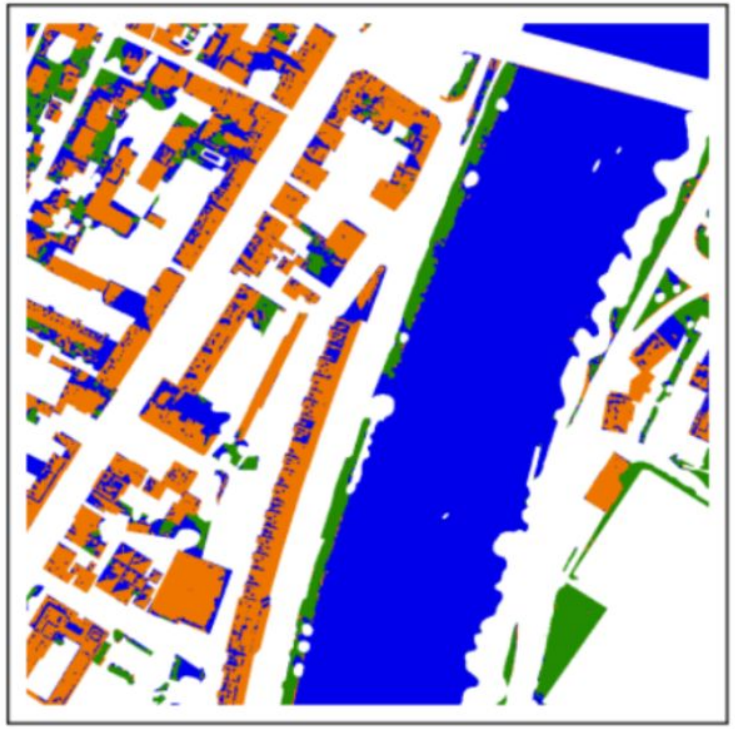

```
alphas=array([1.  , 0.  , 0.  , 1.  , 1.  , 0.25, 0.75, 0.  , 0.  , 0.  , 0.  ,
       0.  , 0.5 , 0.5 , 0.  , 0.75, 0.  , 0.5 , 0.75, 0.5 , 1.  , 0.5 ,
       0.75, 1.25, 0.  , 0.75, 0.75, 0.25, 0.  , 1.  , 0.  , 0.  , 0.75,
       0.5 , 0.25, 0.75, 0.75, 0.  , 1.25, 0.75])
```

Quantum SVC via Annealing

Result comparison with classical SVC

Willsch et al., CPC 248, 107006 (2020)
Cavallaro et al., IGARSS 2020, 1973 (2020)
Delilbasic et al., IGARSS 2021, 2608 (2021)
Pasetto et al., LGRS 19, 1505705 (2022)
Delilbasic et al., JSTARS 17, 1434 (2023)
Pasetto et al., JSTARS 17, 3263 (2024)
Zardini et al., TQE 5, 3103512 (2024)

Ground truth



Classical SVM

Quantum SVM

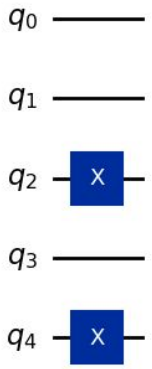
Gates-based Encoding

from classical data to quantum data

Bit encoding

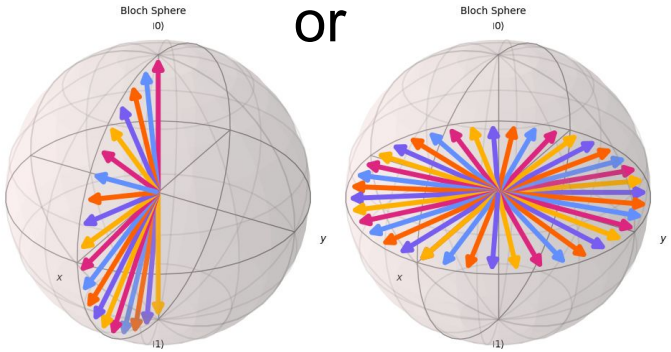
Only need X gates
Only (really) useful for bit data
Else num_qubits = 8 * num_features

$$00101 \longrightarrow |00101\rangle$$



Angle encoding

Encode along an axis
Choose based on data
Can combine both



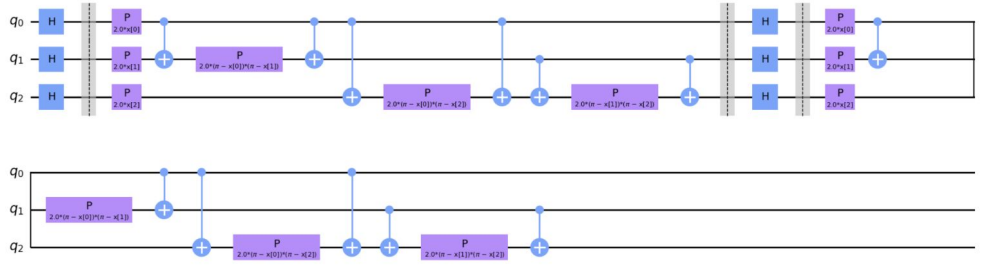
Amplitude encoding

Most qubit efficient $\#qubits \leq log_2(N)$
Least time efficient $TIME = \mathcal{O}(N)$
Need evermore CX gates

$$\begin{bmatrix} \frac{x_0}{||\vec{x}||} \\ \frac{x_1}{||\vec{x}||} \\ \frac{x_2}{||\vec{x}||} \\ \frac{x_3}{||\vec{x}||} \end{bmatrix}$$

Feature map encoding

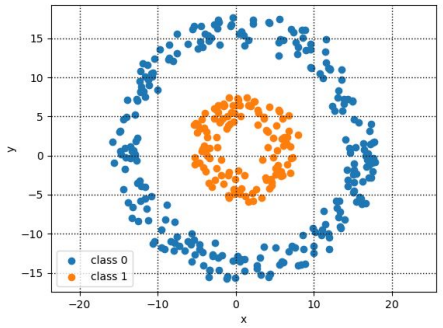
N parameters in M param.-gates for N features
e.g. ZZFeatureMap (2 repetitions)



Gate-based Kernels

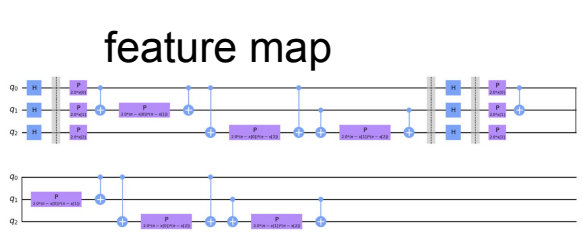
from classical to quantum kernels & kernel tricks

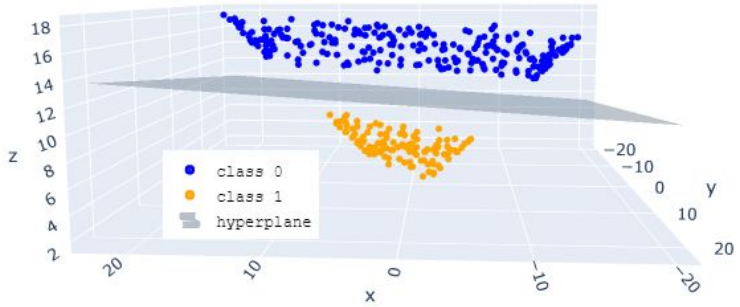
Quantum Kernel Tricks



classical data

feature map





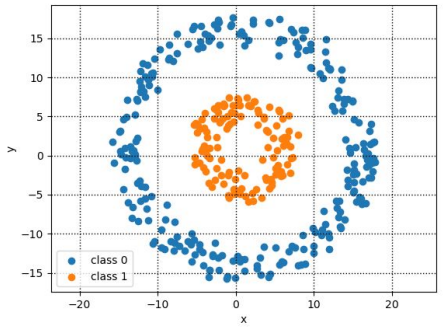
higher dimension
quantum data

Quantum Kernels

Gate-based Kernels

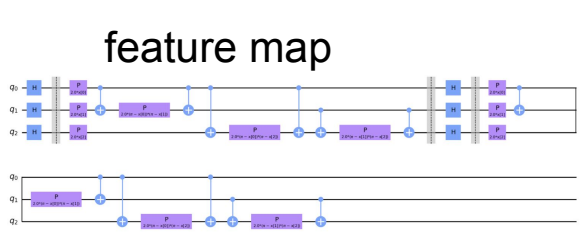
from classical to quantum kernels & kernel tricks

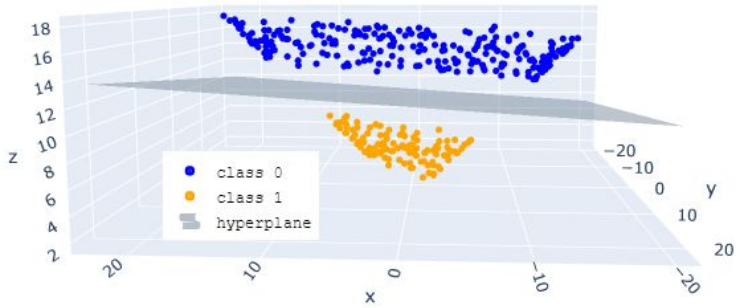
Quantum Kernel Tricks



classical data

feature map



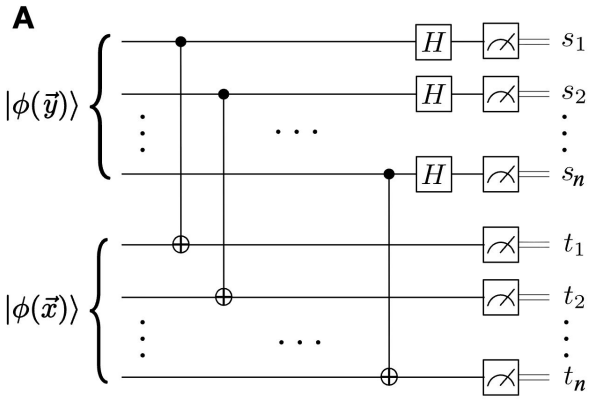


higher dimension quantum data

Quantum Kernels

linear kernel inner product	**linear quantum kernel** fidelity
$\langle \vec{d}_n, \vec{d}_m \rangle$	$\lvert\langle \psi_{\vec{d}_n} \lvert \psi_{\vec{d}_m} \rangle\rvert^2$

perform via SWAP test (below special version)

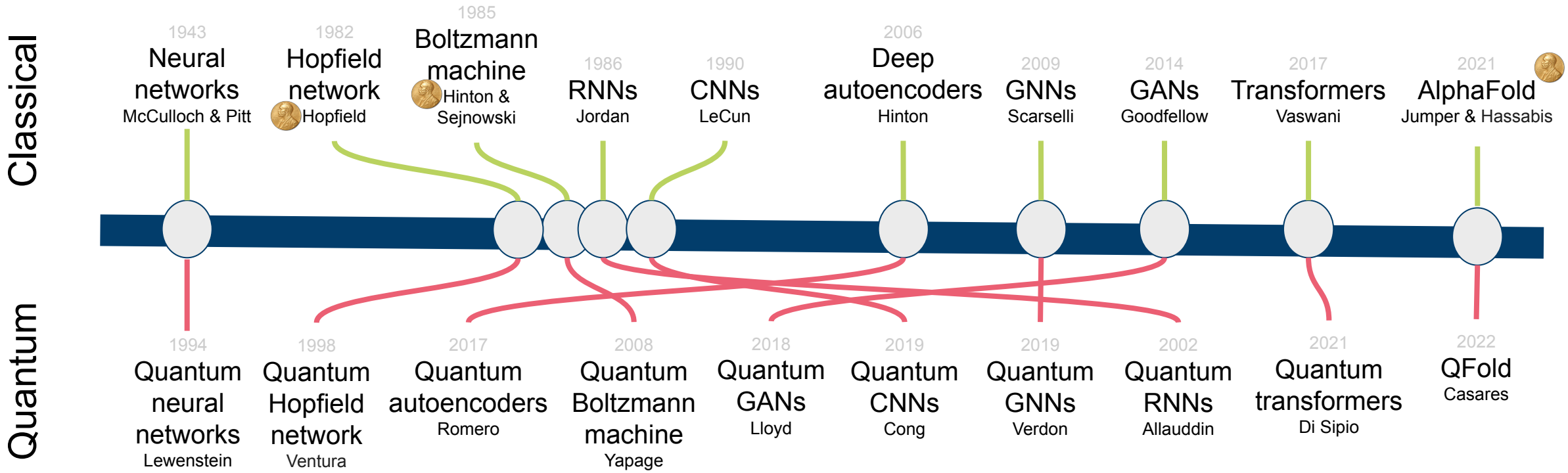


other experimental kernels too

Havlíček et al. Nature 567.7747 (2019): 209-212

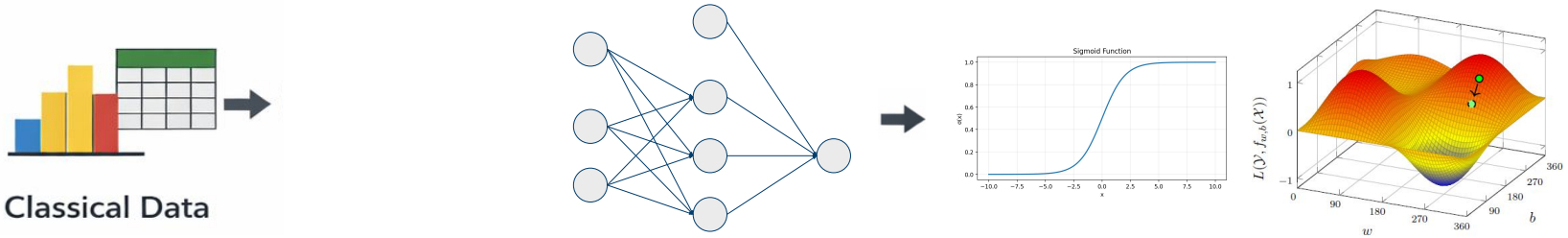
Quantum Deep Learning

(Quantum) Deep Learning Models

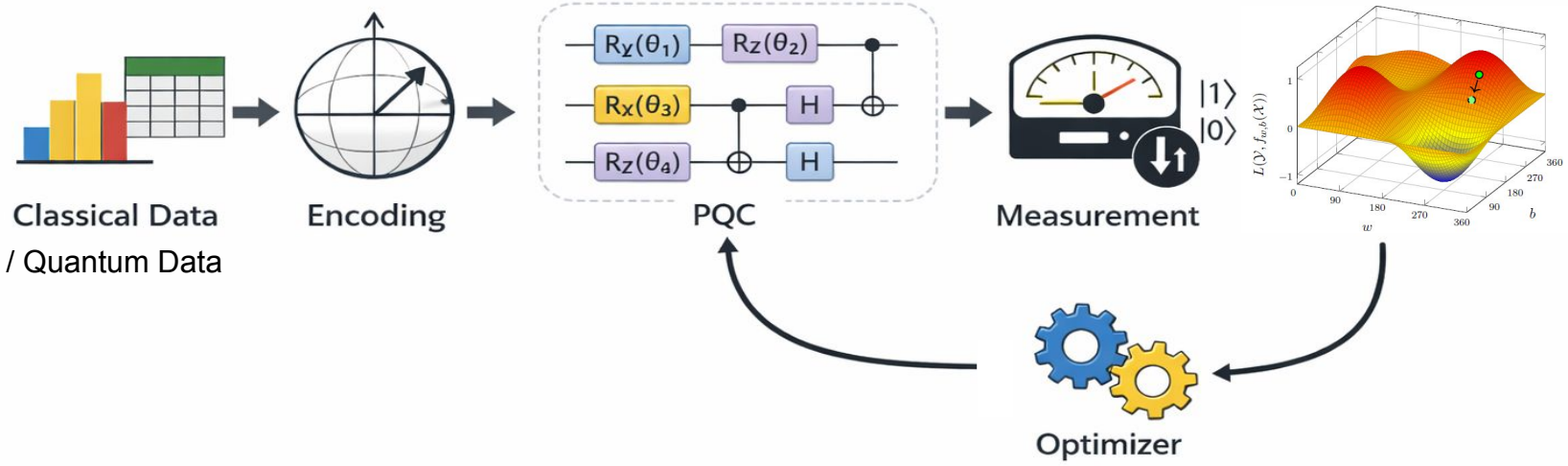


What is (Quantum) Deep Learning?

- Classical deep learning: multi-layer artificial neural networks

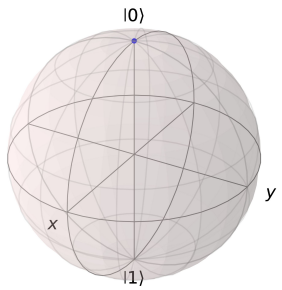


- Quantum deep learning: multi-layer parameterized quantum circuits (PQCs)

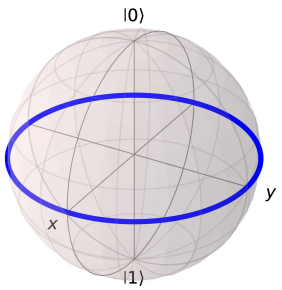


PQCs as Building Blocks for Learnable Models

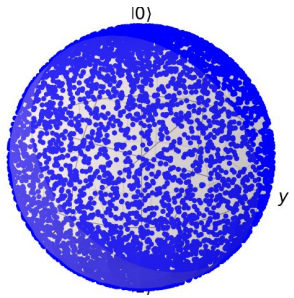
- PQC: a quantum circuit with parameterized gates
- PQCs vary in properties
  - Expressibility:



$R_z(\theta)$

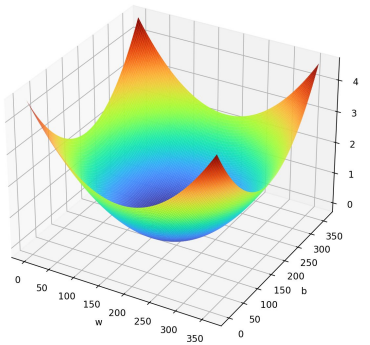


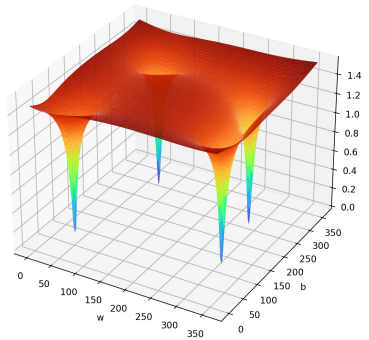
$R_z(\theta)H$

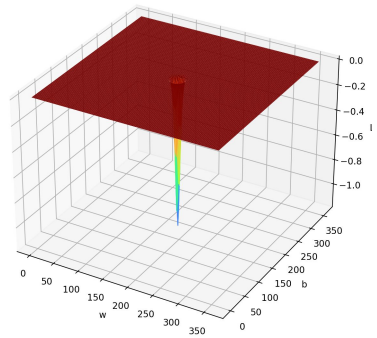


$R_z(\theta_0)R_x(\theta_1)$

  - Trainability:







Gradients and Optimizer I

[1] Schuld *et al.*, *Phys. Rev. A* 99, 032331 (2019)
[2] Henderson *et al.*, *Quantum Mach. Intell.* 2, 2 (2020)

- Gradient computation options:

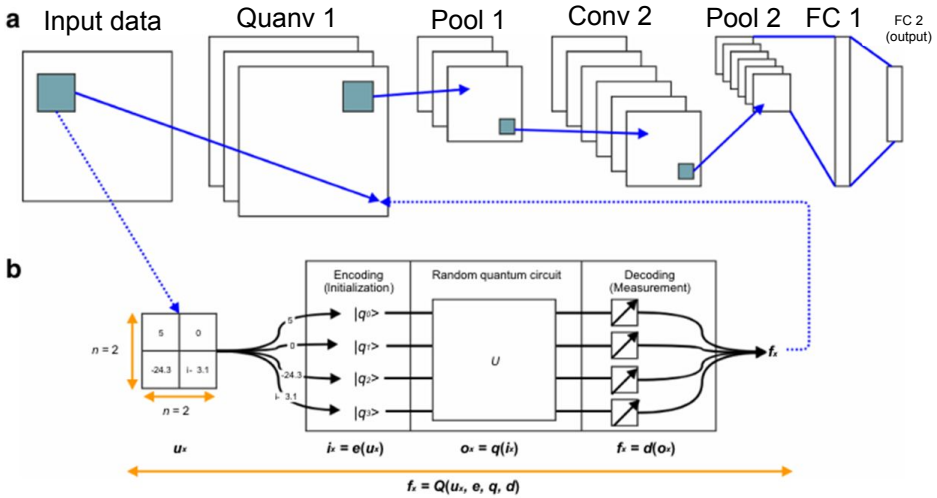
$$\frac{\partial f}{\partial \theta} \approx \frac{f(\theta+\varepsilon) - f(\theta-\varepsilon)}{2\varepsilon}$$

Finite differences

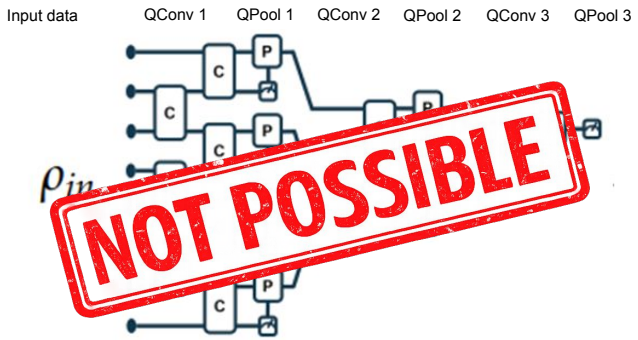
$$\frac{\partial f}{\partial \theta} = \frac{1}{2}\left[f\left(\theta+\frac{\pi}{2}\right) - f\left(\theta-\frac{\pi}{2}\right)\right]$$

Parameter-shift rule [1]

- Backpropagation:



Chain rule in hybrid models [2]



Multi-layer PQCs on QPU

Gradients and Optimizer II

[1] Arrasmith *et al.*, *Quantum* 5, 558 (2021)

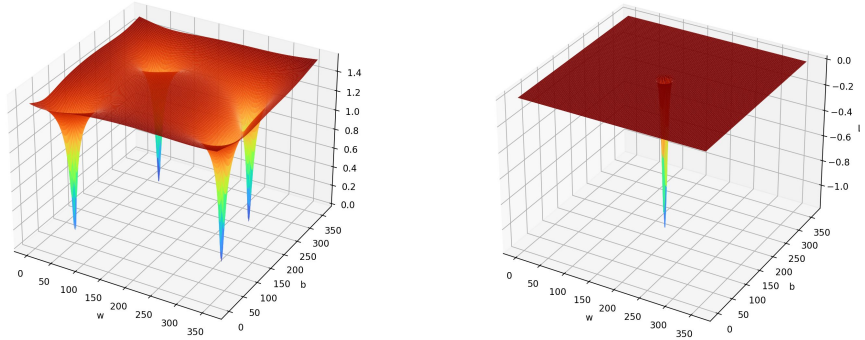
- Gradient computation cost depends on:
  - Number of parameters $|\theta|$
  - Number of shots
- Classical optimizer updates parameters via
  - Gradient-based methods:
    - SGD, Adam: $\mathcal{O}(|\theta| * \#shots)$
    - L-BFGS: $\mathcal{O}(m|\theta| * \#shots)$
  - Gradient-free methods:
    - COBYLA, Nelder-Mead: $\mathcal{O}(|\theta| * \#shots)$
    - SPSA: $\mathcal{O}(\#shots)$
- Additional challenge: vanishing-gradient landscapes affect gradient-based and gradient-free optimization [1]

Barren Plateau Problem

[1] McClean *et al.*, *Nat. Commun.* 9, 4812 (2018)
[2] Holmes *et al.*, *PRX Quantum* 3, 010313 (2022)
[3] Mele, *Quantum* 8, 1340 (2024)
[4] Cerezo *et al.*, *Nat. Commun.* 12, 1791 (2021)
[5] Grimsley *et al.*, *Nat. Commun.* 10, 3007 (2019)
[6] Pesah *et al.*, *Phys. Rev. X* 11, 041011 (2021)

- Barren plateaus (BPs) [1]: cost landscape becomes exponentially flat

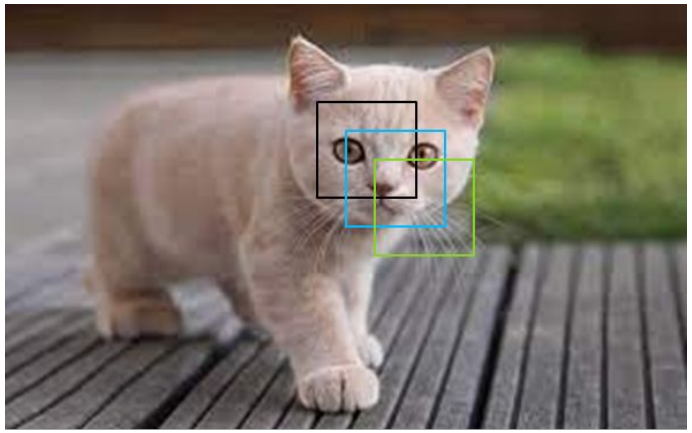
$$\mathrm{Var}_\theta[\partial_\mu C(\theta)] \leq F(n), \quad F(n) \in \mathcal{O}\left(\frac{1}{b^n}\right), \quad b > 1.$$

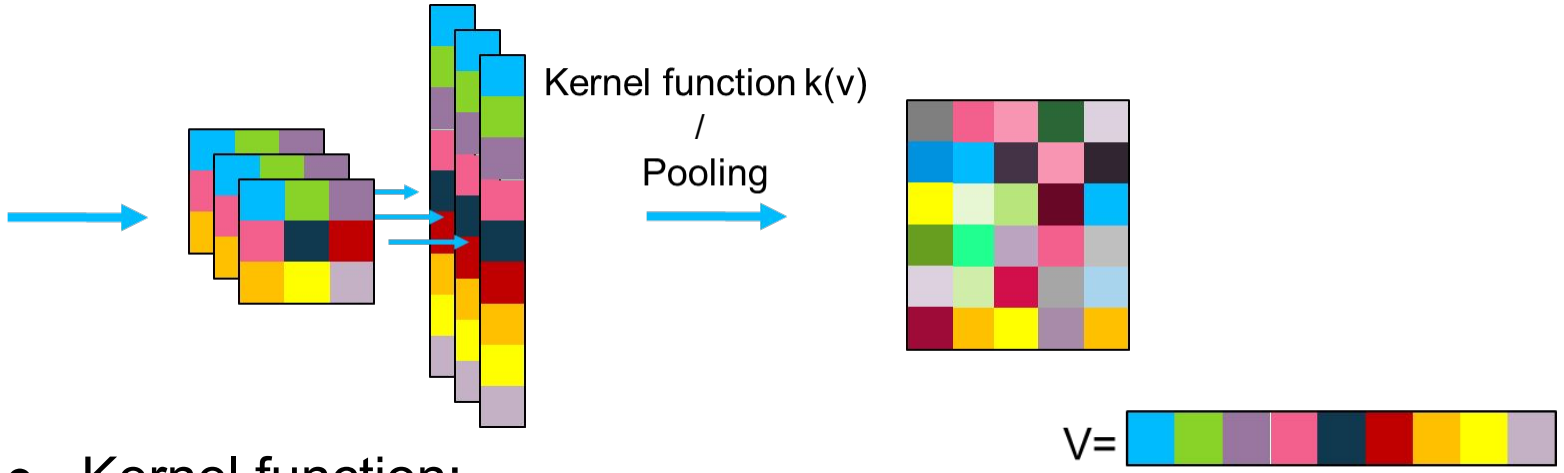


- Common PQCs [2] and 1-designs [1]: $\mathbb{E}\left[\partial_\mu C(\boldsymbol{\theta})\right] = 0$
- Occurrence:
  - BPs occur in highly expressible circuits (approx. 2-designs [3]) [2]
- Mitigation techniques:
  - Local cost [4], variational structure [5], structured PQC layers such as in QCNNs [6], …

How Do Convolutional Neural Networks (CNNs) Work?

- Classical CNN:



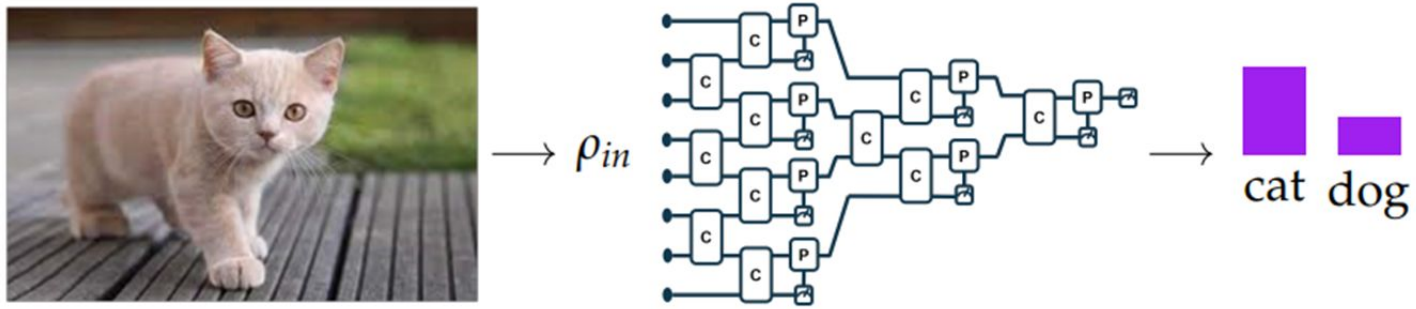


- Kernel function:
  - Classical matrix multiplication
- Pooling
  - Aggregate

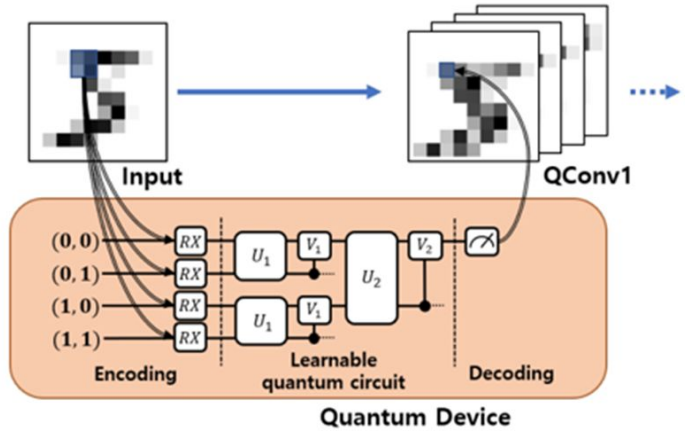
- Advantages: locality + translation equivariance + weight sharing + hierarchical features

How Do Quantum CNNs (QCNNs) Work?

- Quantum CNN (BP free [1]):



Hierachical [2,3,4,8]



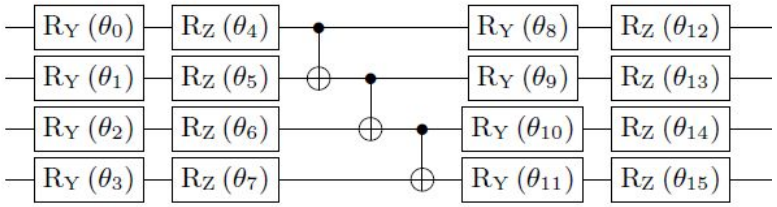
Hybrid [5,6,7,8]

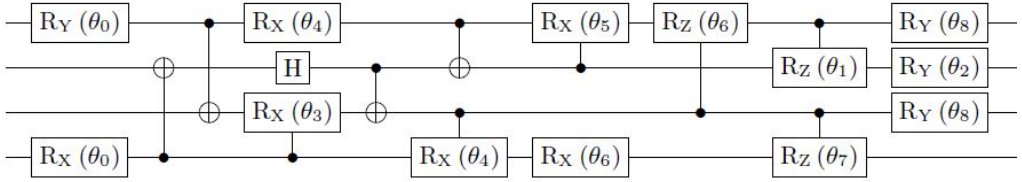
- Kernel function:
  - Local PQCs
- Pooling (qubit reduction)
  - Measurement or control gates

[1] Pesah *et al.*, *Phys. Rev. X* **11**, 041011 (2021)
[2] Grant *et al.*, *npj Quantum Inf.* 4, 65 (2018)
[3] Cong, Choi & Lukin, *Nat. Phys.* 15, 1273–1278 (2019)
[4] Hur, Kim & Park, *Quantum Mach. Intell.* 4, 3 (2022)
[5] Oh, Choi & Kim, *Proc. ICTC 2020*, pp. 236–239 (2020)
[6] Liu *et al.*, *Sci. China Phys. Mech. Astron.* 64, 290311 (2021)
[7] Chen *et al.*, *Phys. Rev. Research* 4, 013231 (2022)
[8] Röseler *et al.*, arXiv:2505.05957v2 (2025)

QCNNs — Real Hardware & Theoretical Problems

- Real hardware issues
  - Depth / gate overhead after transpilation
  - Gradient noise / shot noise
  - Encoding / decoding bottleneck
    - Preprocessing or infeasible encodings (end-to-end)
    - Measurement overhead (hybrid)
- Theoretical issues
  - Suboptimal PQC design
  - Non-optimal CNN comparisons
  - Some QCNNs are efficiently simulable [1]





[1] Bermejo *et al.*, arXiv:2408.12739 (2024)

Classical Simulability

- Small system size (classical simulation)

Classical Simulability

- Small system size (classical simulation)
- Tensor-network (MPS) simulation [1]:
  - Efficient when entanglement is low (small bond dimension)

[1] Vidal, *Phys. Rev. Lett.* 91, 147902 (2003)

Classical Simulability

- Small system size (classical simulation)
- Tensor-network (MPS) simulation [1]:
  - Efficient when entanglement is low (small bond dimension)
- Gottesman–Knill theorem [2]:
  - Clifford-only quantum circuits are classically efficiently simulable

[1] Vidal, *Phys. Rev. Lett.* 91, 147902 (2003)
[2] Gottesman, arXiv:quant-ph/9807006 (1998)

Classical Simulability

- Small system size (classical simulation)
- Tensor-network (MPS) simulation [1]:
  - Efficient when entanglement is low (small bond dimension)
- Gottesman–Knill theorem [2]:
  - Clifford-only quantum circuits are classically efficiently simulable
- Pauli shadows [3]:
  - Random local Pauli measurements → “shadow” estimator for many observables with fewer shots

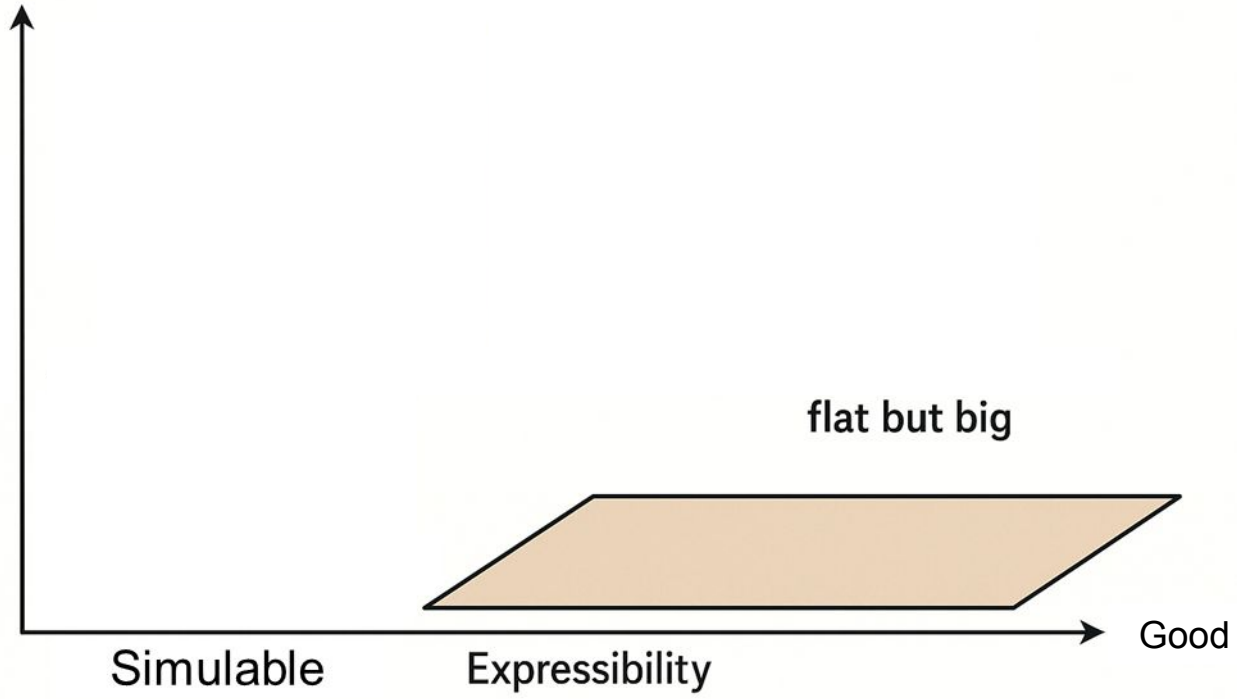
$$S(\rho; N) = \{\hat{\rho}_1 = \mathcal{M}^{-1}(U_1^\dagger \left|\hat{b}_1\right\rangle \left\langle\hat{b}_1\right| U_1),\ \ldots,\ \hat{\rho}_N = \mathcal{M}^{-1}(U_N^\dagger \left|\hat{b}_N\right\rangle \left\langle\hat{b}_N\right| U_N)\}$$

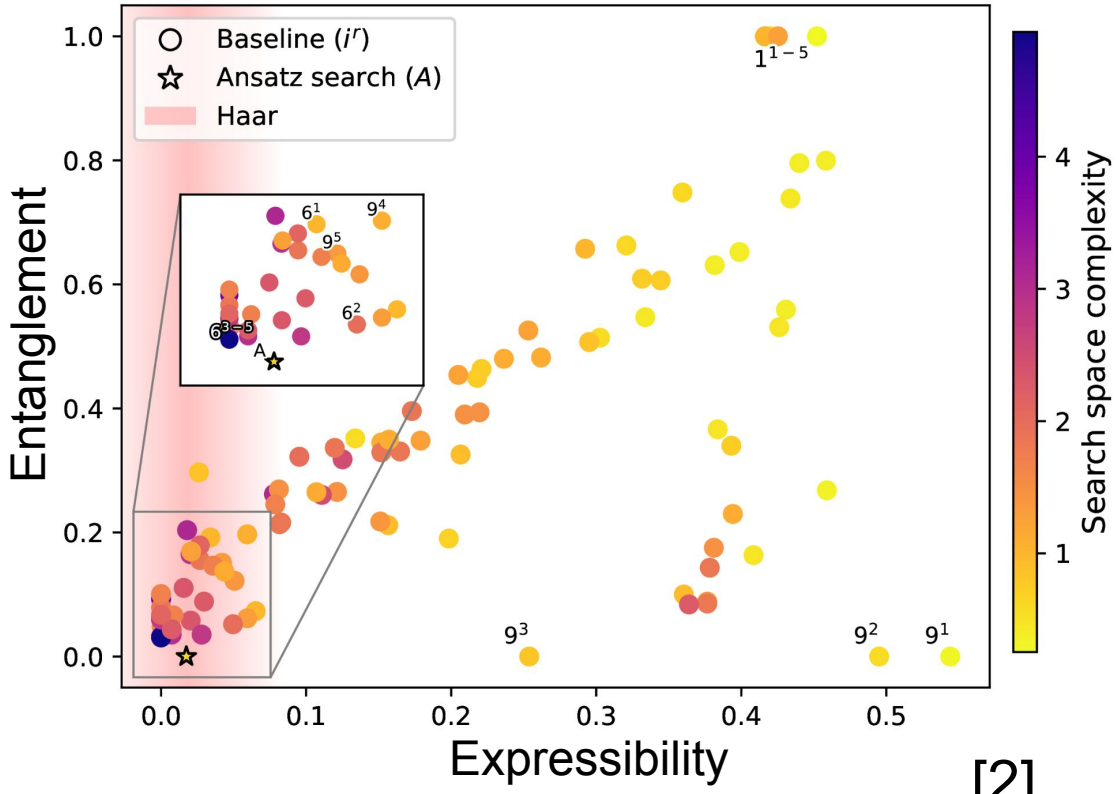
[1] Vidal, *Phys. Rev. Lett.* 91, 147902 (2003)
[2] Gottesman, arXiv:quant-ph/9807006 (1998)
[3] Huang, Kueng & Preskill, *Nat. Phys.* 16, 1050–1057 (2020)

Outlook — How to Choose a PQC I

[1] Sim, Johnson & Aspuru-Guzik, *Adv. Quantum Technol.* 2, 1900070 (2019)
[2] Röseler, Willsch & Michielsen, arXiv:2603.14451v1 (2026)

- Expressibility vs entanglement PQC search compared to common circuits [1]



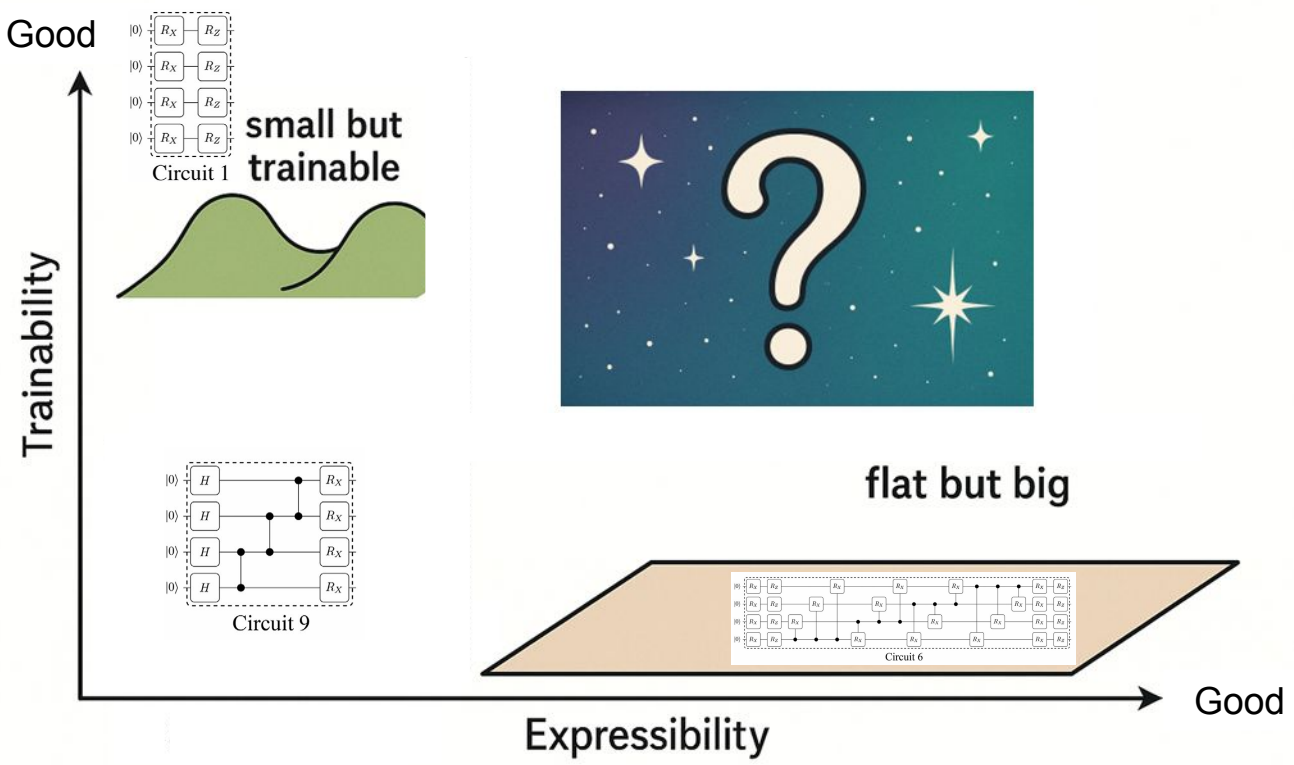


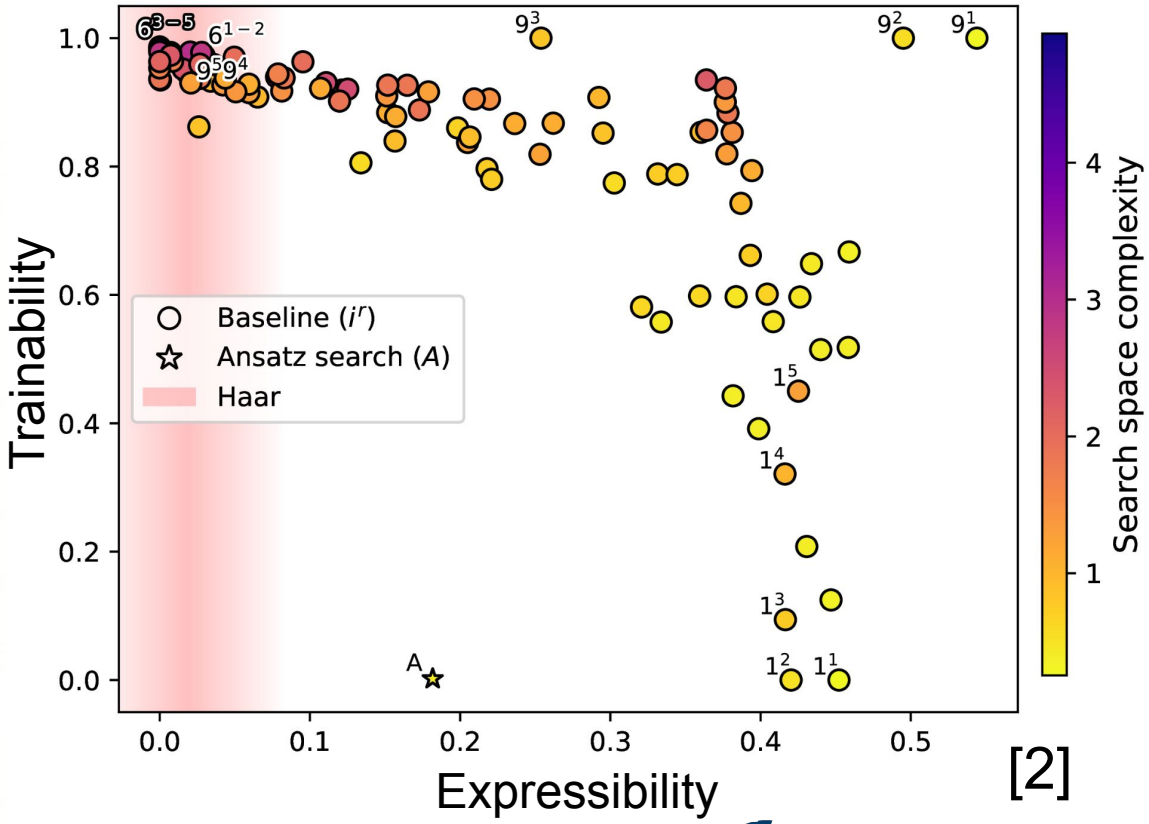
[2]

Outlook — How to Choose a PQC II

[1] Sim, Johnson & Aspuru-Guzik, *Adv. Quantum Technol.* 2, 1900070 (2019)
[2] Röseler, Willsch & Michielsen, arXiv:2603.14451v1 (2026)

- Expressibility vs trainability PQC search compared to common circuits [1]

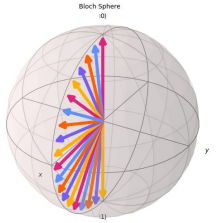


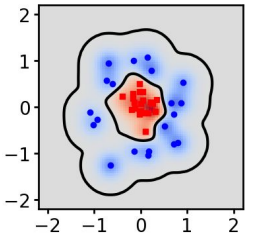


Summary

What we have learned

- Quantum machine learning
  - Data encoding (How can data be encoded efficiently?)
  - QSVM (How should one choose a good kernel for QSVMs?)
  - QBM (What is a good Hamiltonian choice for a quantum Boltzmann machine?)

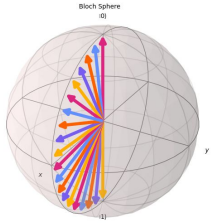


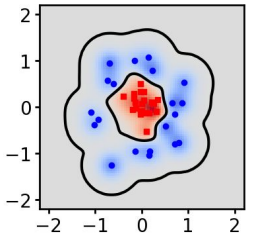


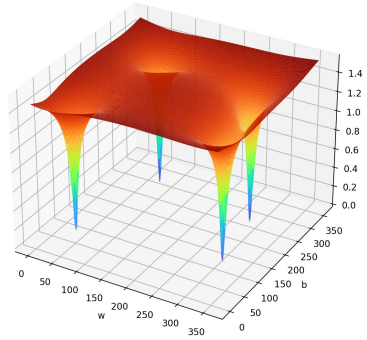
Summary

What we have learned

- Quantum machine learning
  - Data encoding (How can data be encoded efficiently?)
  - QSVM (How should one choose a good kernel for QSVMs?)
- Quantum deep learning
  - Barren plateaus (Can VQAs affected by barren plateaus still be useful?)
  - QCNNs (Does there exist a non-simulable, barren-plateau-free QCNN?)
  - Classical simulable (Can Pauli shadows be utilized for quantum models?)
  - PQC choice (Can PQCs be used to explain physical phenomena?)







Marketing

Opportunities for theses

Theses offers for **Masters** and **Bachelors** students

Focuses: **Computer Science**, **Physics**, or **Mathematics**

Resources: HPC and **real** quantum computers

Large **variety** of quantum devices

General information about our Quantum Information Processing (QIP) group

Contacts:

Gabriel Fazito - g.fazito@fz-juelich.de

Kristel Michielsen - k.michielsen@fz-juelich.de

Thank you

Questions?