Compositional optimization of quantum circuits

Optimizing quantum support vector machines

Raghav Juyal & Noa Aarts

Planning

- Introduction
- Background information
- The simulation
- Results
- Conclusion
- Outlook

The problem

- Embedding Machine Learning kernels in gates
- Small data problems
- Sensitivity to ansatz
- Quantum Support Vector Machine

Finding an optimal ansatz

1. Search all permutations
2. Optimize gate parameters

- Exponential scaling

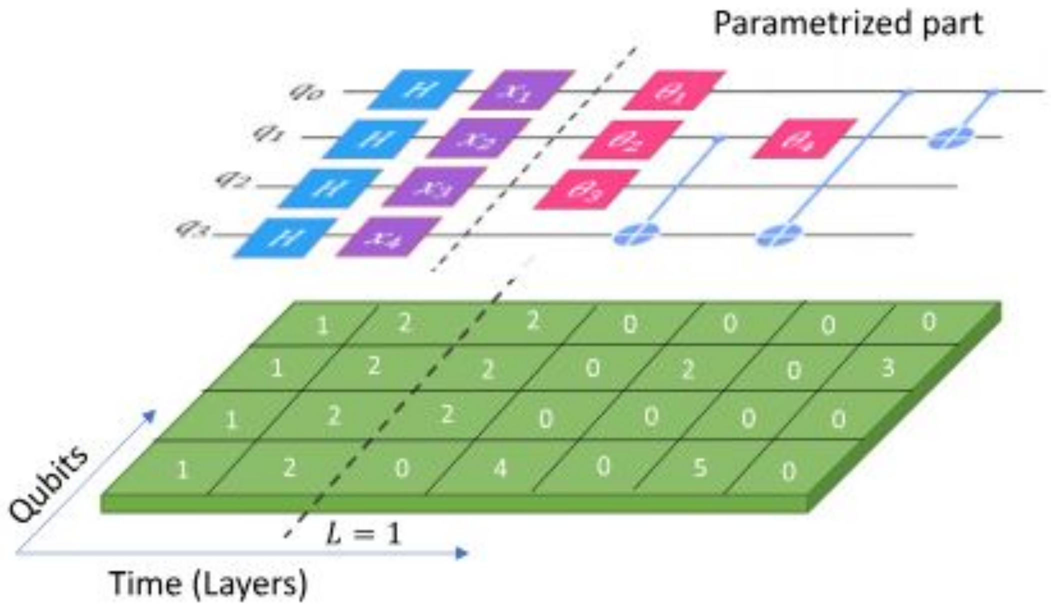


Image Source: [Paper]

A solution

- Bayesian approach
- Marginal likelihood
- Bayesian Information Criterion (BIC)
- Pick K best circuits
- Optimize parameters of M best circuits

Background

Support Vector Machine(SVM)

- Supervised Learning Algorithm
- Maximize Margin
- Linear Classification

$$\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\mathbf{x}_i^\mathsf{T} \mathbf{x}_j) y_j c_j,$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \le c_i \le \frac{1}{2n\lambda} \text{ for all } i.$$

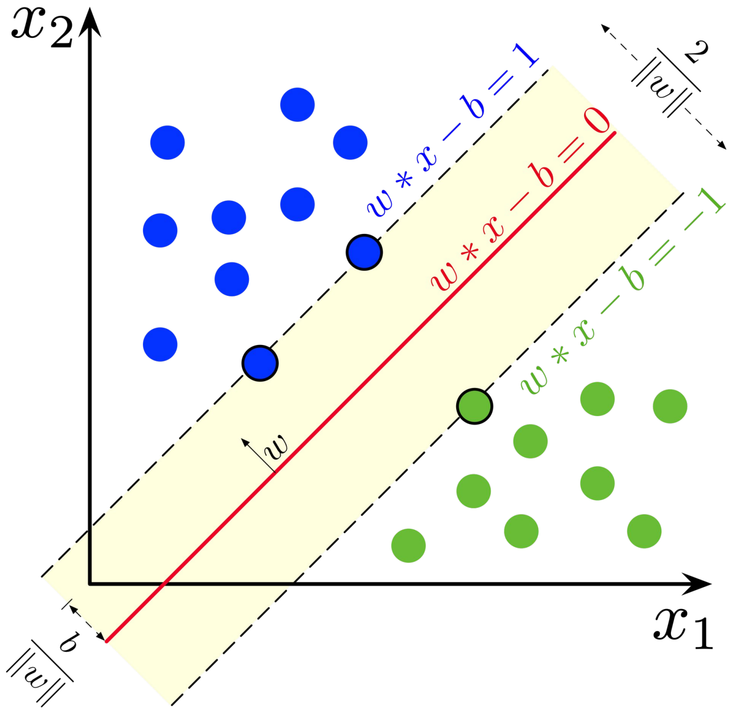
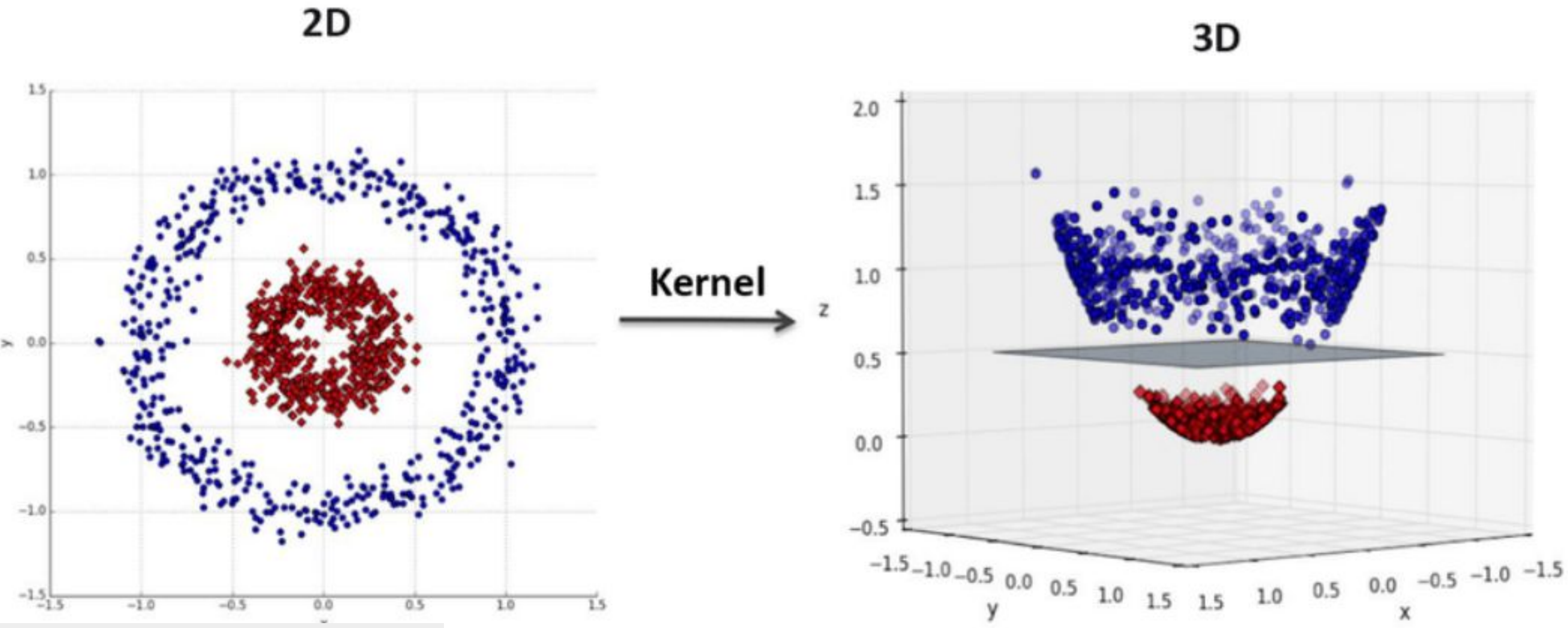


Image source: [Wikipedia]

Feature Maps and Kernels

Image Source: [Medium]

$K(x, z) = |\langle \Phi(x), \Phi(z) \rangle|^2$



Quantum SVM

- Use quantum circuit to get the kernel then use classical computer for SVM

$$k(x, x') = | \langle \phi(x') | S^{\dagger}(x') S(x) | \phi(x) \rangle |^2$$

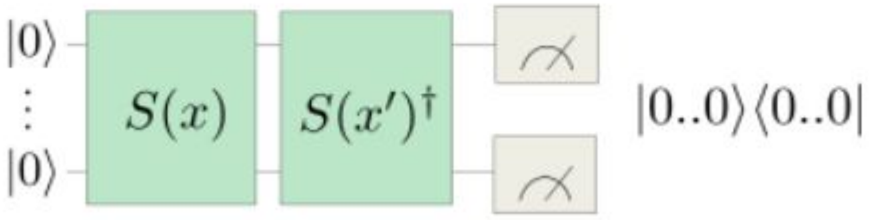


Image Source: [Pennylane]

Error Metrics and Optimization

- Bayesian Information Criterion(BIC) = k ln(n) - 2 ln(L)
    - k: number of parameters, n: number of observations, L: Likelihood Function
- Akaike Information Criterion(AIC) = 2k - 2 ln(L)
- Validation Accuracy
- F1 Score = (2*TP)/(2*TP + FP + FN)
    - TP: True Positive, FP: False Positive, FN: False Negative

Choosing Parameters

- Bayesian Optimization using skopt library
- Functions are expensive to compute and noisy
- Number of calls = 15

Dataset: Qiskit ad hoc dataset

- ZZ Feature Map
- Separation Gap
- 100 Train Data
- 100 Validation Data
- 4100 Test Data

Image Source: [Qiskit]

```
ad_hoc_data(training_size, test_size, n, gap, plot_data=False, one_hot=True,
    include_sample_total=False)  [source]
```

Generates a toy dataset that can be fully separated with `ZZFeatureMap` according to the procedure outlined in [1]. To construct the dataset, we first sample uniformly distributed vectors $\vec{x} \in (0, 2\pi]^n$ and apply the feature map

$$|\Phi(\vec{x})\rangle = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} |0^{\otimes n}\rangle$$

where

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i\right)$$

and

$$\begin{cases} \phi_{\{i,j\}} = (\pi - x_i)(\pi - x_j) \\ \phi_{\{i\}} = x_i \end{cases}$$

We then attribute labels to the vectors according to the rule

$$m(\vec{x}) = \begin{cases} 1 & \langle \Phi(\vec{x}) | V^\dagger \prod_i Z_i V | \Phi(\vec{x}) \rangle > \Delta \\ -1 & \langle \Phi(\vec{x}) | V^\dagger \prod_i Z_i V | \Phi(\vec{x}) \rangle < -\Delta \end{cases}$$

where Δ is the separation gap, and $V \in \mathrm{SU}(4)$ is a random unitary.

The current implementation only works with n = 2 or 3.

Implementation

The Simulation

- Initialisation
- Generating candidates
- Optimizing candidates
- Our additions

Image Source: Supplementary Material

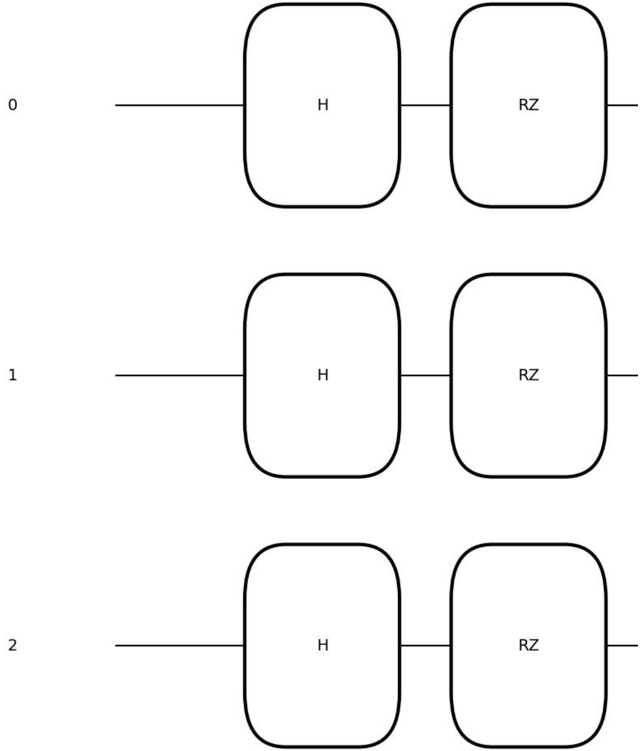
Algorithm 1 Quantum kernel optimization for SVM

Input:
(i) Classification data set $\{\boldsymbol{x}, y\}$: including training, validation;
(ii) A set of quantum gates;
(iii) $L_{\max}$: number of quantum circuit (QC) layers;
(iv) K: number of QCs for local search;
(v) M: number QCs with optimized parameters;
(vi) $\mathcal{N}$: maximum number of iterations in Bayesian optimization (BO).
Output: An optimized QC architecture with optimized gate parameters $\boldsymbol{\theta}^*$

Build a QC that encodes inputs into gate parameters
Initialize list OptimalQC with K identical QCs
for All $1 < L \leq L_{\max}$ **do**
  for Each element in OptimalQC **do**
    for All possible gate combinations in layer L **do**
      Append a layer of gates to QC
      Compute the quantum kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
      Train an SVM model with $k(\boldsymbol{x}, \boldsymbol{x}')$
      Convert outputs of SVM to probabilistic predictions
      Calculate BIC with the validation set
    end for
  end for
  Sort all resulting quantum kernels by BIC in increasing order
  Replace list OptimalQC with K lowest BIC entries
  for $M \leq K$ QCs with lowest BIC $\in$ OptimalQC **do**
    while iteration $\leq \mathcal{N}$ **do**
      Determine the gate parameters $\boldsymbol{\theta}$ using the acquisition function of BO
    end while
    Assign the resulting gate parameters to $\boldsymbol{\theta}^*$
  end for
end for

Initialisation

- Initialise to |+>
- Data Rotations



Generating candidates

- Recursive function
- Checks for CNOT collisions
- No repeated gates
- Calculate BIC and AIC

```python
def gate_combinations_sub(
    qubits: int, previous_layer: tuple[int]
) -> Generator[tuple[int, ...], None, None]:
    if qubits == 0:
        yield ()
    else:
        for combination in gate_combinations_sub(qubits - 1, previous_layer):
            yield combination + (0,)  # Identity
            if previous_layer[qubits - 1] != 1:
                yield combination + (1,)  # Hadamard
            if previous_layer[qubits - 1] != 2:
                yield combination + (2,)  # R_z
            for offset in range(1, len(combination) + 1):  # CNOT gates
                if (
                    combination[-offset] == 0
                    and all(
                        combination[-offset + o] != o + 2 for o in range(1, offset + 1)
                    )
                    and previous_layer[qubits - 1] != offset + 2
                ):
                    yield combination + (offset + 2,)
```

Optimizing candidates

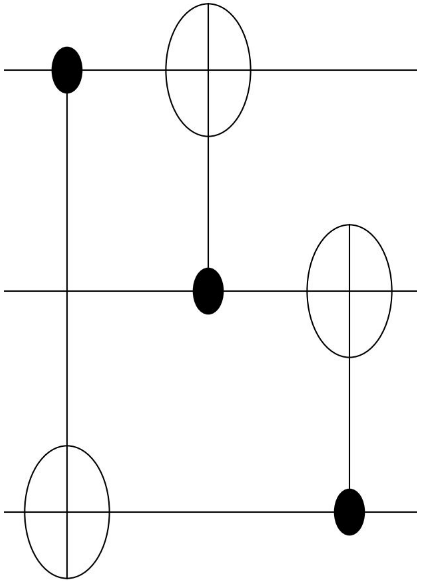
- Sort on criterion
- Take top M
- Optimize R_Z parameters
- Bayesian optimization
    - 10 starting points
    - 15 steps

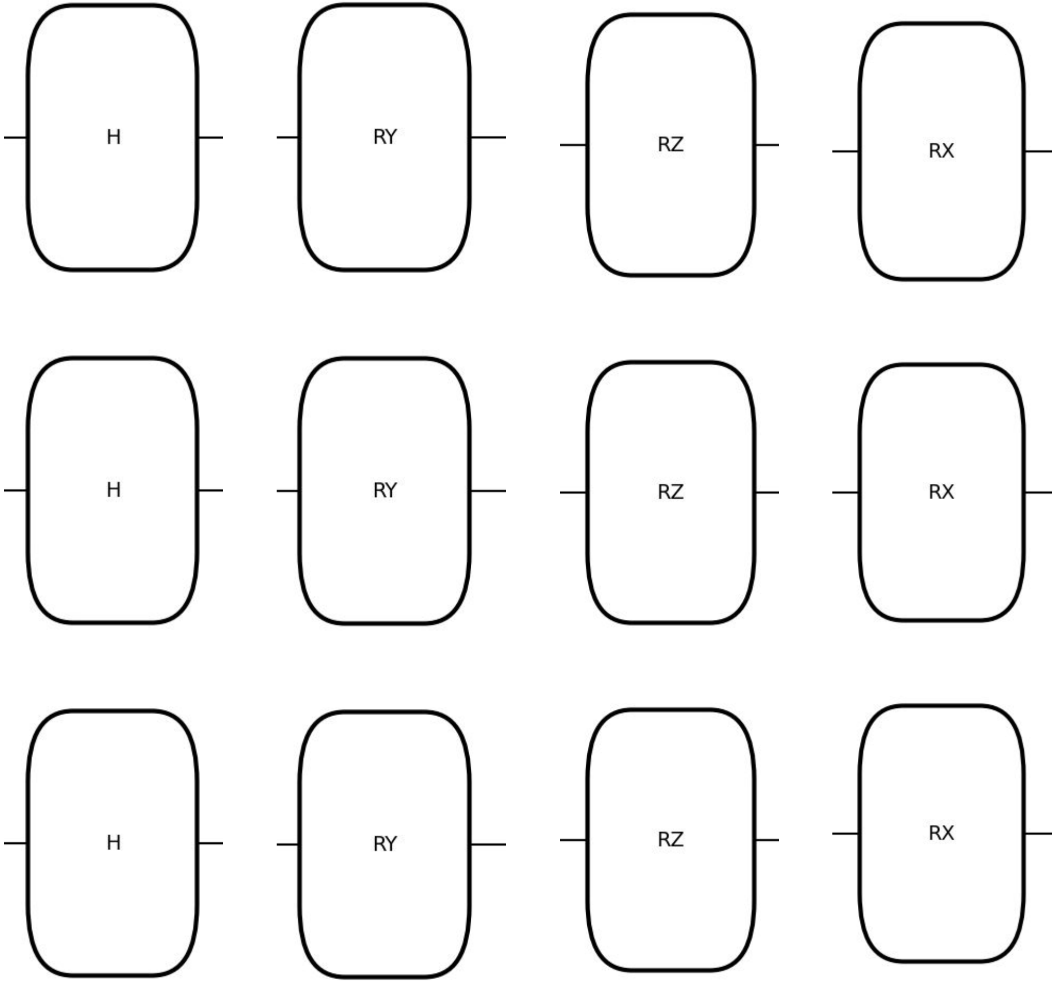
Testing the outcomes

- Test accuracy
- Separate dataset
- Much larger than training/validation
- Tests the BIC validity

Fixed Layers

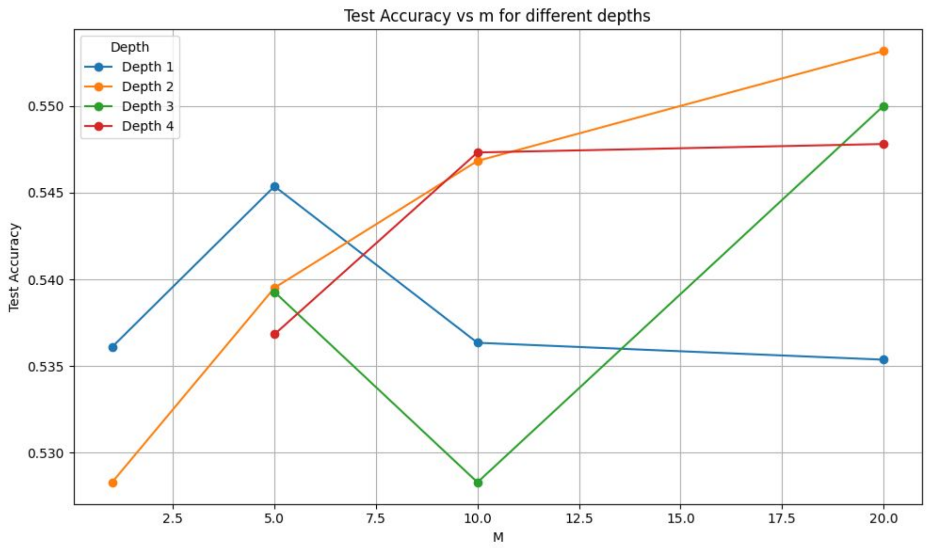
- No per-qubit generation
- 5 types of layers

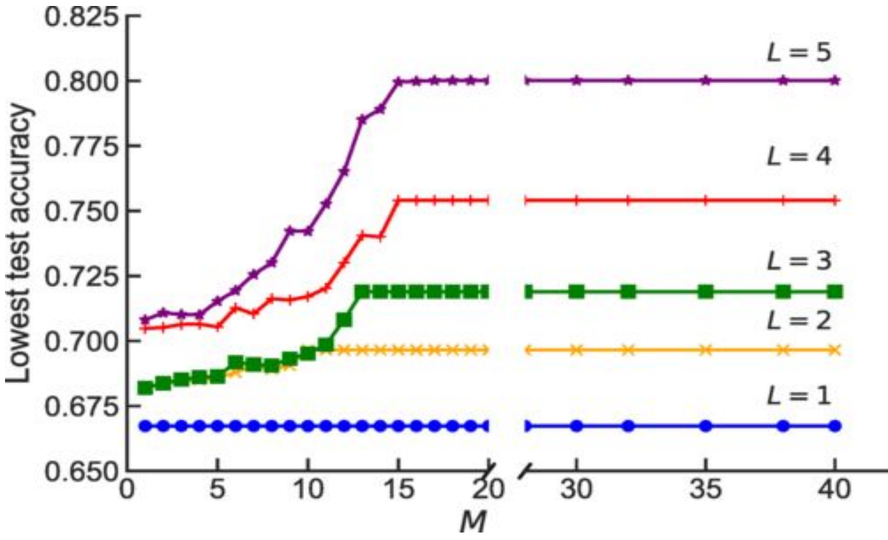




Results

Accuracy vs M

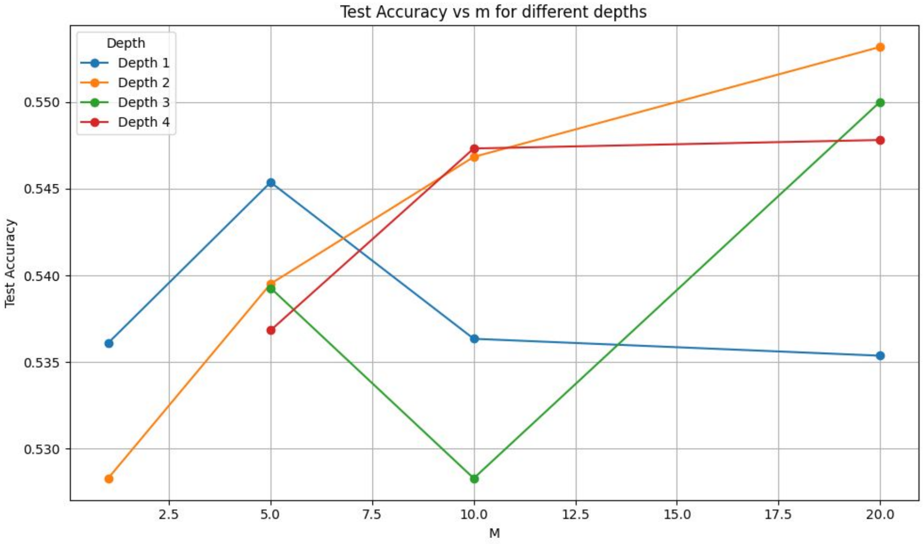




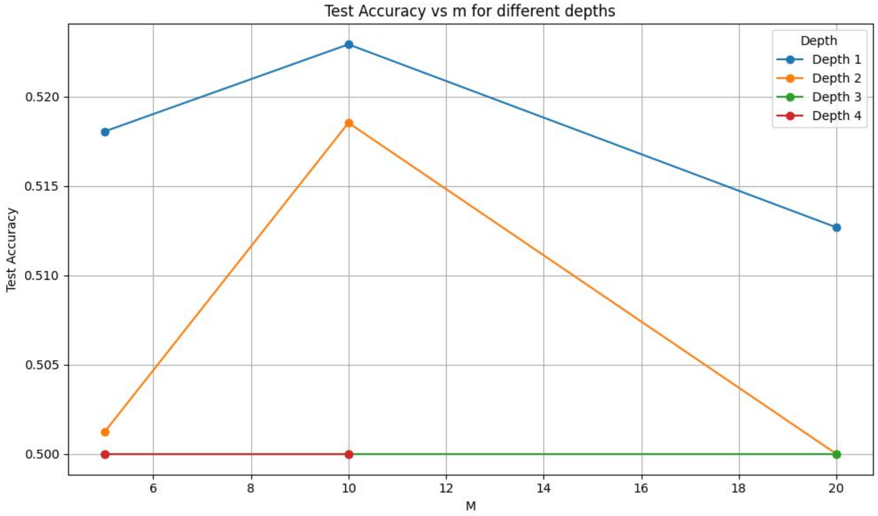
Our simulation

The [Paper]

Accuracy vs M for fixed set of layers

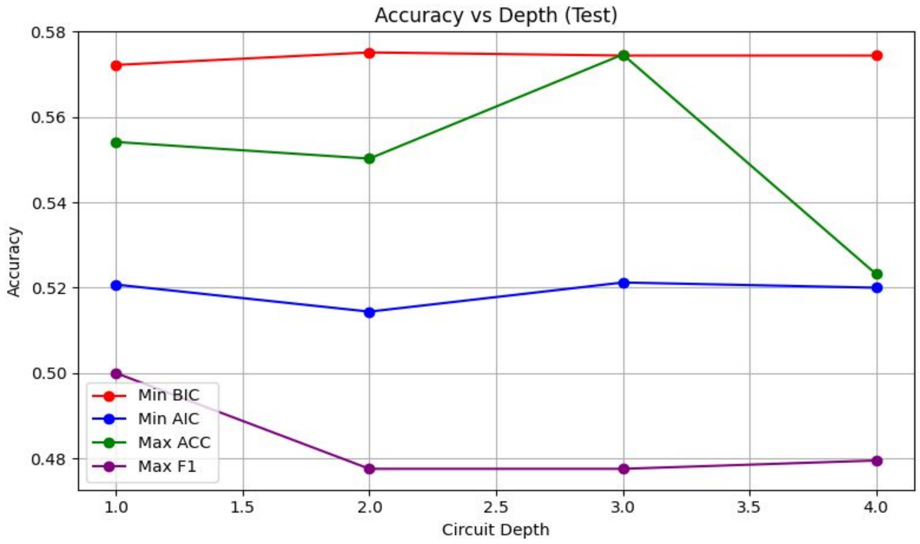


Original Gate Set

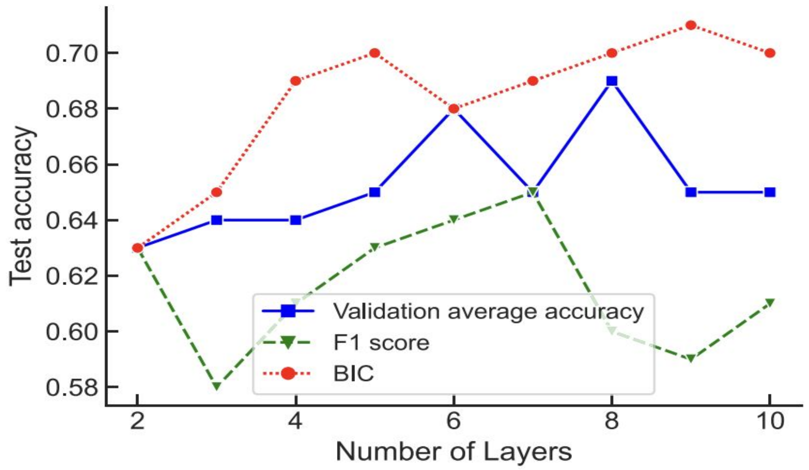


Fixed set of layers

Kernel Selection Metrics

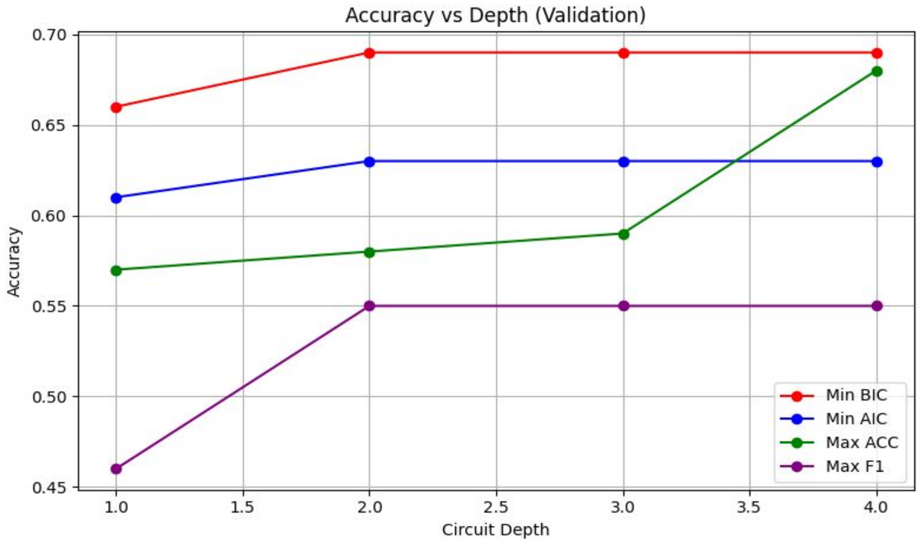


Our simulation

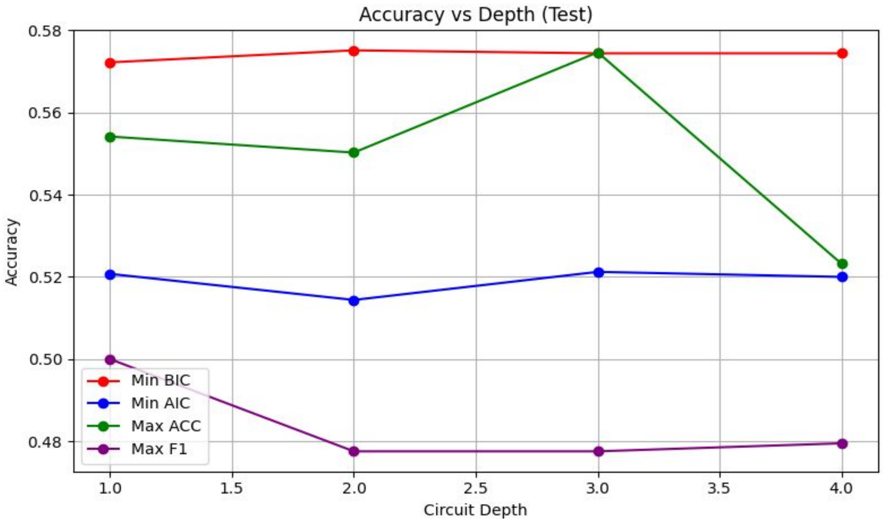


The [Paper]

Kernel Selection Metrics: Validation vs Test

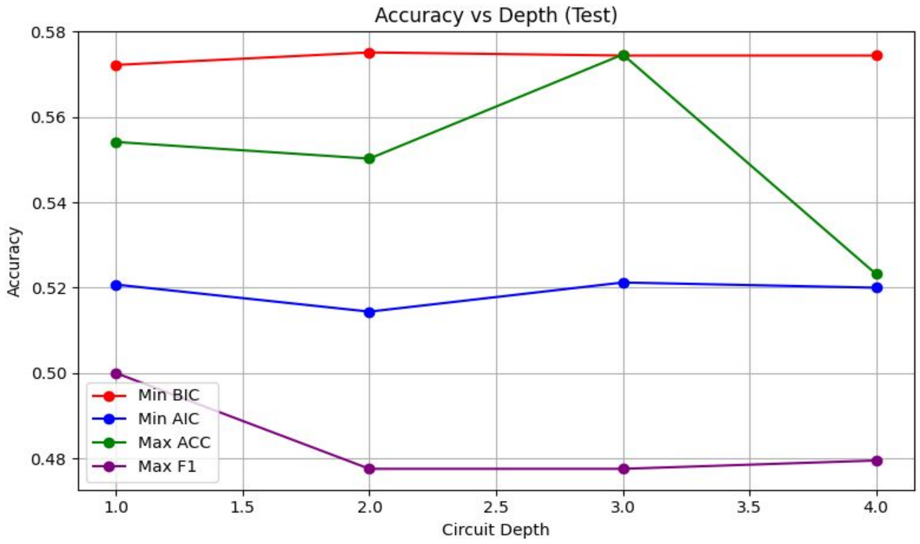


Validation Accuracies

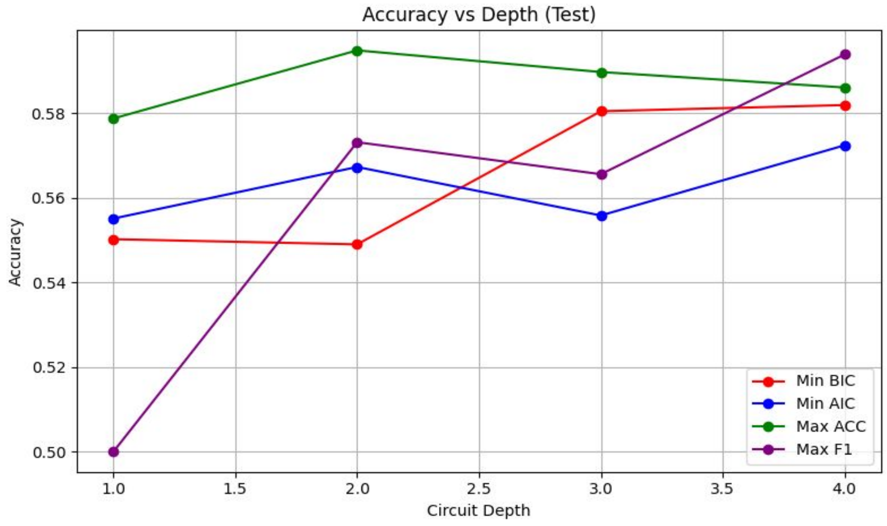


Test Accuracies

Kernel Selection Metrics: Varying Dataset

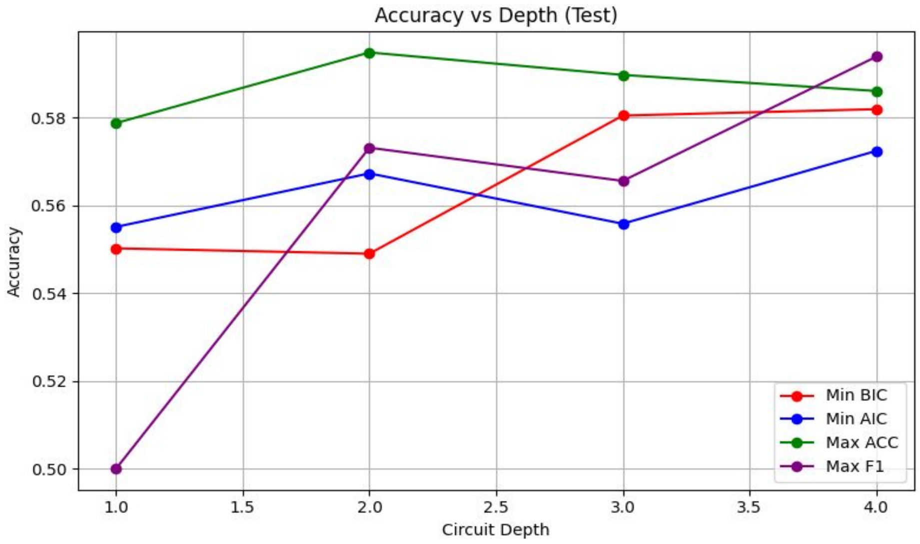


Gap = 0.3

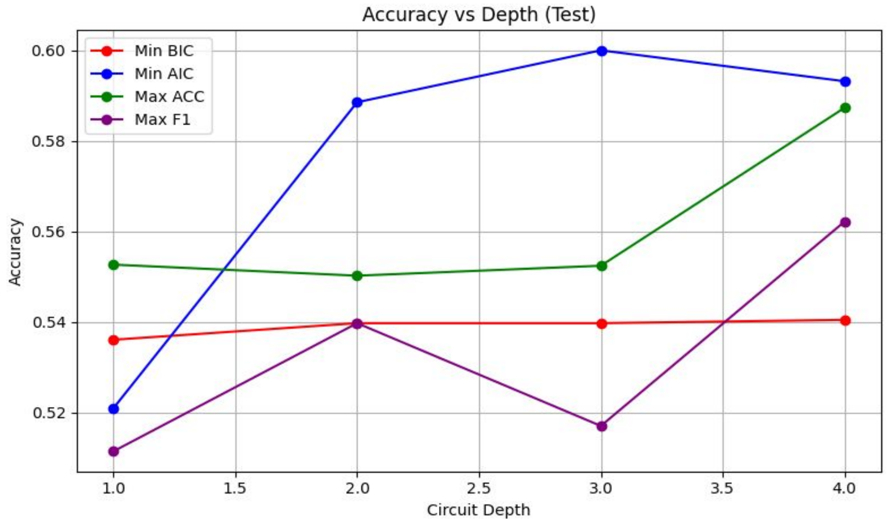


Gap = 0.5

Kernel Selection Metrics: Varying Initial Points



Initial Points = 10



Initial Points = 5

Conclusion and Outlook

- While unable to reproduce all results
- Kernel selection metrics
- AIC between F1 and accuracy, worse than BIC

- More datapoints for comparisons
- Continue trying to replicate results
- Try AICc as well

Paper Followed: [Compositional optimization of quantum circuits for quantum kernels of support vector machines]

Pitfalls we found

- Initially using R_X gates instead of R_Z for initialisation
- We didn't initially include x_k in the R_Z gates
- Initial points versus number of calls in Bayesian Optimization
- Direction of CNOT gates

Questions

Our code

https://github.com/itepastra/aqa contains all the code used to generate our images