Implementing Quantum-Kernel-Based Classifiers in the NISQ Era: Supplementary Material

Shivani Mahashakti Pillay[1], Ilya Sinayskiy[2], Edgar Jembere[1], and Francesco Petruccione[2]

[1] School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, South Africa
[2] School of Chemistry and Physics, University of KwaZulu-Natal, South Africa

A A Brief Introduction to Quantum Computing

Quantum computers make use of quantum systems to perform computations. While classical computers store classical information in bits, quantum computers store quantum information in qubits, where each qubit is a 2-level quantum system [1]. Unlike classical bits, the qubit can exist in a superposition of two states: $|0\rangle$ and $|1\rangle$, until some measurement is performed on the qubit.

A qubit in a superposition is usually represented as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The probability that the qubit is in the state $|0\rangle$ is $|\alpha|^2$ and the probability that the qubit is in the state $|1\rangle$ is $|\beta|^2$.

Information can be obtained from the qubit through a measurement of the quantum system. This collapses the superposition of the qubit, the outcome of the measurement is then either 0 or 1. Due to the probabilistic nature of quantum mechanics, different outcomes can be obtained in each measurement. Thus, many measurements are made to obtain the final result of the computation.

Qubits are manipulated through unitary transformations, since unitary transformations preserve the normalization of the state of the qubit. Quantum circuits, which are analogous to classical logical circuits, provide a visual representation for the unitary transformations applied to qubits [1]. The gates used in a quantum circuit transform the state of the qubit/s thus manipulating the information contained in the qubit/s.

Multiple qubits can also be manipulated in the same system. In a multi-qubit system with no entanglement, the system can be represented as the tensor product of the individual qubit states. This is known as a separable state. For example, consider two qubit states $|\psi\rangle_A$ and $|\psi\rangle_B$, the state $|\Psi\rangle = |\psi\rangle_A \otimes |\psi\rangle_B$ is a separable state.

For an entangled multi-qubit state, the system cannot be represented as the tensor product of the individual qubit states. A circuit to entangle two qubits is shown in Figure A 0. This circuit generates a quantum state known as a Bell State. This Bell State can be generated by applying a Hadamard

gate, represented by H followed by a Controlled-NOT operation, represented by U_{CNOT}, on two qubits in the ground state $|0\rangle \otimes |0\rangle$ as follows:

$$U_{\text{CNOT}}(H \otimes I(|0\rangle \otimes |0\rangle)) = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$$

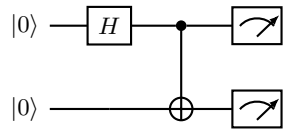

Fig. A 0: A circuit to generate the Bell Pair. A Hadamard gate and C-NOT operation is used.

A more detailed introduction to quantum computing can be found in [1].

B A Brief Introduction to Kernels

Intuitively, a kernel is some similarity measure defined over the input space. If $\mathcal{X}$ denotes the input space, a kernel is some positive semi-definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$, $(\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}')$.

Kernels are closely linked to feature maps. A kernel can be derived from the inner product of two data vectors that have been transformed by some feature map. If $\phi : \mathcal{X} \to \mathcal{F}$ is a feature map that maps the input data to a feature space $\mathcal{F}$ then a kernel can be derived as

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}) | \phi(\mathbf{x}') \rangle$$

This is useful for non-linearly separable data; the feature map can be chosen such that the data is mapped to a higher dimensional feature space in which it becomes linearly separable. This idea follows from Cover's Theorem on separability of patterns [2]. In this way, kernels allow a linear classifiers like the SVM to classify non-linearly separable data.

Classically, instead of explicitly applying the feature map to each data vector and evaluating the inner product between pairs of the transformed data vectors, the kernel is evaluated directly as a function of the input data vectors [3]. This is known as the *kernel trick* and the function that is evaluated is referred to as a *kernel function*. The kernel trick relies on the assumption that the kernel function is simpler to evaluate than the inner product of two explicitly mapped input data vectors. However, for some large feature spaces, it is possible that even the kernel function is computationally expensive to evaluate.

C Parameter Settings

	Encoding Functions				
Dataset	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5
Circles	1.9467	10.0	1.0155	0.8950	3.1976
Moons	0.9394	9.36148	3.9259	9.8137	8.7687
XOR	3.3324	9.7023	5.8498	3.1795	9.2719
Exp	7.4154	8.2168	7.2296	10.0	8.6547

Table C 1: For each non-linearly separable dataset, the regularization parameters used for every classifier using the same set of encoding functions

The number of shots used to execute the circuits on the real device was 8192.

D Results

Kernel Matrix	Encoding Functions				
	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5
$K^{(\text{ideal})}$	1.0	1.0	1.0	1.0	1.0
$K^{(\text{raw})}$	0.98	0.95	1.0	1.0	1.0
$K^{(\text{post})}$	0.98	0.99	1.0	1.0	1.0

Table C 1: Classification accuracies obtained using the different kernel matrices for the Circles dataset

Kernel Matrix	Encoding Functions				
	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5
$K^{(\text{ideal})}$	0.79	0.98	0.89	0.99	0.99
$K^{(\text{raw})}$	0.76	0.94	0.85	0.89	0.91
$K^{(\text{post})}$	0.77	0.96	0.90	1.0	0.96

Table C 2: Classification accuracies obtained using the different kernel matrices for the Moons dataset

Kernel Matrix	Encoding Functions				
	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5
$K^{(\text{ideal})}$	0.88	0.92	0.99	0.98	0.97
$K^{(\text{raw})}$	0.86	0.83	0.97	0.92	0.96
$K^{(\text{post})}$	0.85	0.90	0.98	0.98	0.96

Table C 3: Classification accuracies obtained using the different kernel matrices for the Exp dataset

Kernel Matrix	Encoding Functions				
	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5
$K^{(\text{ideal})}$	1.0	0.97	1.0	0.94	0.97
$K^{(\text{raw})}$	1.0	0.91	0.99	0.89	0.92
$K^{(\text{post})}$	1.0	0.95	0.99	0.94	0.95

Table C 4: Classification accuracies obtained using the different kernel matrices for the XOR dataset

Dataset	Encoding Function	Post-Processing Strategy	Relative Improvement in Alignment
Moons	Φ_2	`Id`, `R-TIK`, `M-MEAN`, `Id/R-TIK/R-THR/R-FLP/R-SDP`	-8.583229
Moons	Φ_2	`Id`, `R-TIK`, `M-SINGLE`, `Id/R-TIK/R-THR/R-FLP/R-SDP`	-8.534183
Moons	Φ_2	`M-READ`, `R-TIK`, `M-MEAN`, `Id/R-TIK/R-THR/R-FLP/R-SDP`	-8.267544
Moons	Φ_2	`M-READ`, `R-TIK`, `M-SINGLE`, `Id/R-TIK/R-THR/R-FLP/R-SDP`	-8.219666
Exp	Φ_2	`Id`, `R-TIK`, `M-SINGLE`, `Id/R-TIK/R-THR/R-FLP/R-SDP`	-8.107733

Table C 5: The five worst relative improvements in alignment achieved

The Relative Improvement in Accuracy (RIAcc) is a measure that is introduced to describe how a post-processing strategy can increase/decrease the accuracy of the classifier using the kernel matrix it yields. It is given by:

$$RIAcc(K^{\text{post}}) = \frac{\text{Accuracy}(K^{\text{post}}) - \text{Accuracy}(K^{\text{raw}})}{1 - \text{Accuracy}(K^{\text{raw}})}$$

Dataset	Encoding Function	Post-Processing Strategy	Relative Improvement in Alignment	Relative Improvement in Accuracy

Circles	Φ_1	`Id`
		`Id,`
		`M-SINGLE,`
		`R-FLP`
			0.078560	-0.5

Circles	Φ_1	`M-READ,`
		`Id,`
		`M-SINGLE,`
		`R-FLP`
			0.102308	-0.5

Circles	Φ_1	`M-READ,`
		`R-FLP,`
		`Id,`
		`Id/R-SDP`
			0.100556	-0.5

Circles	Φ_1	`M-READ,`
		`R-FLP,`
		`M-SPLIT,`
		`Id/R-TIK,`
		`R-THR/R-FLP/R-SDP`
			0.100556	-0.5

Circles	Φ_1	`M-READ`
		`Id`
		`M-MEAN`
		`R-FLP`
			0.103245	-0.5

Table C 6: Five examples where an increase in alignment lead to a decrease in accuracy

References

1. M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
2. Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, (3):326–334, 1965.
3. Bernhard Schölkopf. The kernel trick for distances. In *Advances in Neural Information Processing Systems*, pages 301–307, 2001.