Qlearnkit: a Python toolkit for quantum machine learning

Giulio Corallo, Massimiliano Pronesti, Federico Tiblias
Institut EURECOM

January 20, 2022

Abstract

Quantum hardware offers new interesting possibilities for training machine learning models given its significant speedup. Plenty of quantum algorithms exist in theory but actual implementation are few and far between. In this paper we describe a possible implementation of the K-NN, K-Means, SVM and Ridge Regression models using quantum circuits. This cohesive implementation is based on Qiskit and collected inside a Python package named Qlearnkit. We have shown that these models perform well in terms of accuracy and various other metrics compared to classical counterparts. Time performances on the other hand were relatively low since simulating quantum hardware adds a significant computation overhead.

I. Introduction

Machine learning and deep learning techniques are gaining more and more attention given the breakthrough results of recent years [1] [2]. Nonetheless, the ever-increasing complexity of such models is growing side by side with the computational resources needed to train them. This led researchers to investigate further computational solutions [3]. Among the many promising alternatives, quantum computing stands out, given its potential in terms of execution speedup for large-scale problems. In fact, it has been shown that quantum algorithms exhibit rather significant speedups in commonly used subroutines such as Fourier transforms, finding eigenvectors and eigenvalues, solving linear equations, some even attaining an exponential advantage over their classical counterparts. These speedups translate very well into an increase in performance for commonly used techniques in machine learning and convex optimization such as Principal Component Analysis, kernel methods, Support Vector Machines, least-squares fitting and gradient descent [4]. In this paper we describe our implementation of the following well-known supervised and unsupervised machine learning algorithms for a gated quantum computer: Least-Squares Support Vector Machine, K-Nearest Neighbors, K-Means clustering and Ridge Regression. These implementations have been assembled into a Python library named qlearnkit, built on top of Qiskit and available on GitHub and PyPI.

II. Related Work

Building a proper, physically scalable and stable quantum computer is a rather difficult task. Nevertheless, research in the field and its application to different domains, such as cryptography and machine learning has been carried out since the late '80s, providing theoretical implementations and insights on their benefits with respect to the classical versions. An excellent summary of near future quantum computing applications to machine learning is provided in [5], where the authors show how purely quantum or hybrid version of machine learning techniques can perform better in several directions, such as runtime, capacity or learning itself. [6] provides insights into the state-of-the-art quantum machine learning implementations. In section 4, it is pointed out that quantum computing will be able to speedup learning techniques in different degrees, since the subroutines such as HHL and amplitude amplification can provide exponential speedup and quadratic speedup separately, e.g. exponential speedup to Quantum LDA, Quantum PCA and Quantum SVM, Quadratic speedup to Quantum K-Means clustering.

From the software point of view, being quantum machine learning a nascent field, not many toolkits exist in the market or among the open-source community. Qiskit ([7]) and IBM are definitely leading the path and qiskit-machine-learning, born from the now deprecated qiskit-aqua, provides some pure and hybrid quantum machine learning implementations. Differently from Qiskit, qlearnkit looks at the ease of usage and implements different classification, regression and clustering techniques.

Another relevant work is Pennylane ([8]), a cross-platform Python library for differentiable programming of quantum computers, providing tools for quantum computing and machine learning though offering a different approach towards usage. In fact, it seamlessly integrates classical machine learning libraries with quantum simulators and hardware, giving users the power to train quantum circuits.

Finally, [9] is a fancy implementation of a pure quantum K-Nearest-Neighbors classifier built on top of qiskit and developed as a work of thesis at Radboud University Nijmegen in collaboration with ING Quantum Technology. Differently from it, qlearnkit employs a simpler quantum architecture and reduces the constraints on the number of required qubits, the speed of execution and avoids race conditions appearing due to an incorrect use of parallelization with bigger datasets, where [6] seems to fail.

III. Datasets

The Datasets we used to test our implementations of the quantum algorithms are those provided by scikit-learn[10], in particular iris, wine, diabetes, make_regression, make_classification and make_blobs.

The Results section clarifies which datasets are used on which models.

IV. Methods

i. Quantum Encodings

Quantum computers expect data in quantum state for processing. The first step in quantum machine learning is to load classical data by encoding it into the state of the qubits. This process is also known as quantum data encoding [11] or embedding and is an important step in Quantum state preparation. Classical data encoding for quantum computation plays a critical role in the overall design and performance of the quantum machine learning algorithm.

In the specific case of our library, we implemented amplitude and angle encodings. Amplitude encoding represents classical data as amplitudes of quantum states. Angle encoding, contrarily, represents classical features as angles of a multi-qubit system.

ii. Quantum K-Nearest Neighbors

The quantum version of the K-Nearest Neighbors algorithm estimates the distance of two vectors in the following way: given a zero initialized ancillary qubit and a quantum state $|\Psi\rangle$ which stores vectors we want to estimate

$$|0\rangle|\Psi\rangle, |\Psi\rangle = |\psi\rangle|\phi\rangle$$

First we apply a Hadamard gate to the ancillary qubit, pushing it into superposition

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\Psi\rangle$$

To entangle the state Ψ with our ancillary qubit, we apply a swap gate to Ψ controlled on the ancillary qubit

$$\frac{1}{\sqrt{2}}(|0\rangle|\Psi\rangle + |1\rangle F(|\Psi\rangle))|\Psi\rangle$$

where $F(|\Psi\rangle)$ represent the swapped version of $|\Psi\rangle$

$$F(|\Psi\rangle) = |\phi\rangle|\psi\rangle$$

Finally we apply another Hadamard gate to the ancillary qubit

$$\frac{1}{2}(|0\rangle|(I+F)\Psi\rangle + |1\rangle|(I-F)\Psi\rangle)$$

where I is the identity operator. Thus, the probability of the ancillary qubit being measured as in the $|1\rangle$ state is

$$P(|1\rangle) = |\Psi\rangle\frac{I-F}{2}|\Psi\rangle = \langle\psi\phi|\frac{I-F}{2}|\psi\phi\rangle$$

Expanding the inner product we get

$$P(|1\rangle) = \frac{1}{2}\langle\psi\phi, (|\psi\phi\rangle - |\psi\phi\rangle)\rangle = \frac{1-\langle\psi,\phi\rangle^2}{2}$$

The last term is proportional to the square distance between two normalized vectors. This proves rather interesting from a quantum point of view as we exploit an inner property of states in quantum mechanics instead of directly measuring it [12]. This approach, called SWAP test, is the key for the quantum distance measurement.

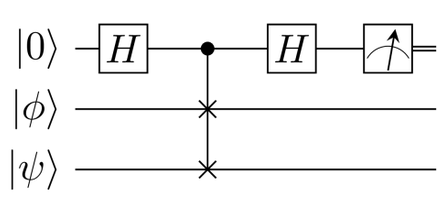


Figure 1: *Swap Test Circuit*

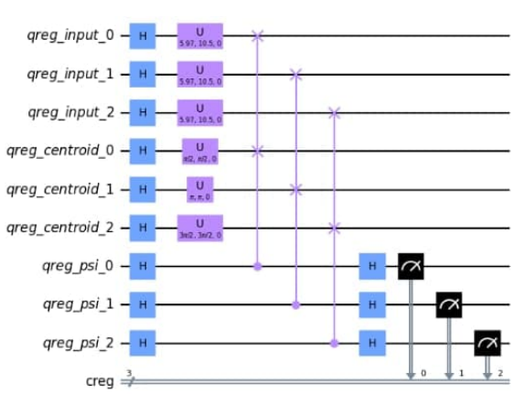
There are other explorable metrics proposed in scientific literature, such as the Hamming distance metric [13], nevertheless the first one sticks to the traditional idea of the classical K-NN algorithm. In fact, after constructing the quantum circuits, their results are used to compute the fidelities and eventually to do a majority voting to predict the class.

iii. Quantum K-Means

In a classical K-Means algorithm, given a dataset, at each iteration, the data points are assigned to a cluster according to the Euclidean distance. The centroids are then updated so that the new centroid is the average of all points that have been assigned to the cluster in this iteration. The loss function that this algorithm is trying to minimize is the RSS (residual sums of squares). A single iteration has complexity of $O(kNd)$ and the algorithm can be super-polynomial in the worst case [14]. The quantum version of the K-Means algorithm has performance similar to the classical K-Means and provides a running time with substantial savings, especially for large datasets, in fact the running time is poly-logarithmic in the number of elements of the dataset [15]. In order to run the QKMeans algorithm we firstly map data feature values to a quantum state:

$$\phi = \frac{(x+1)\pi}{2}$$
$$\theta = \frac{(x+1)\pi}{2}$$

(where ϕ is the phase and θ the angle) and then we perform encoding using the U gate provided by the Qiskit library. The algorithm is then run with a similar logic to the classical one. First, we pick some random initial points using some heuristic, for example K-Means++[16]. At each step of our iteration, we need to compute distances of our new data point with respect to the centroids of our cluster, which is easily doable in a quantum context, in a similar way as explained for the Quantum K-NN. Alternatively, instead of constructing a 1-level SWAP test circuit we can construct a k-level circuit, where k is the number of clusters, in order to compute in one-shot the distances of a state vector with respect to the k cluster centroids. This can get better performances and high parallelism. Below a figure of the circuit assuming $k = 3$.



However, in this case the number of qubit needed increases. In fact, for each level added, we need 3 more qubits.

iv. Quantum Support Vector Machines

Quantum SVMs offer an exponential speedup in training and predicting compared to classical ones. An SVM convex optimization problem can be reformulated into a system of linear equations in the form $Ax = b$, the solution of which is a trained SVM model:

$$\begin{bmatrix} 0 & 1_M^T \\ 1_M & K + \gamma^{-1} I_M \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ Y \end{bmatrix}$$

with M number of training samples, $K = K(x_i, x_j)$ kernel matrix, X training set, Y classification labels, γ regularization hyperparameter, $1_M = [1, ...1]^T$, I_M identity matrix, α and β respectively weights and bias used in prediction. Model prediction on new data can be obtained as

$$f(\cdot) = K(\cdot, X) \cdot \alpha + \beta$$

Building A requires computing the kernel matrix K of distances from each sample to each sample, and solving the system requires a matrix inversion [17]. The kernel matrix can be obtained exponentially faster using quantum computers: a kernel function can be evaluated with a $O(logN)$ complexity against the $O(N)$ of classical computers, where N is the number of features [18]. Taking advantage of these two properties, we implemented an algorithm capable of constructing a kernel matrix and computing the solution of a linear system in time $O(M^3 + M^2 logN)$ [19]. First we build the kernel matrix using Qiskit's FeatureMaps, then we prepare the $Ax = b$ system and finally we solve it with a matrix inversion. A further refined version of the algorithm was formulated that can enhance the matrix inversion and provide a total runtime of $O(logMN)$. However, it was not implemented due to time and complexity constraints.

v. Quantum Kernel Ridge Regression

We know a closed-form solution of the Ridge Regression problem exists that requires computing a kernel matrix of points [20]. In a way almost identical to the Quantum SVM described above, we start from the linear system formulation of the problem but delegate the computing of the kernel matrix to a quantum subroutine, thus obtaining a speedup.

V. Results

i. Quantum K-Nearest Neighbors

The Quantum K-Nearest Neighbors Classifier was compared against a classical version employing the euclidean metric for distance measurement, in terms of accuracy, F1-score and Jaccard index. The plots below were generated training the models using $k = 3$ and the full Iris dataset normalized via amplitude encoding. The classical data has been reduced via PCA with 2 components before plotting the results.

Both the qasm_simulator and the statevector_simulator provided by Qiskit produce the same results, classifying the data well enough with respect to the classical version provided by scikit-learn. It should be noted, however, that in this example the classification process was affected by the precision of the measurement.

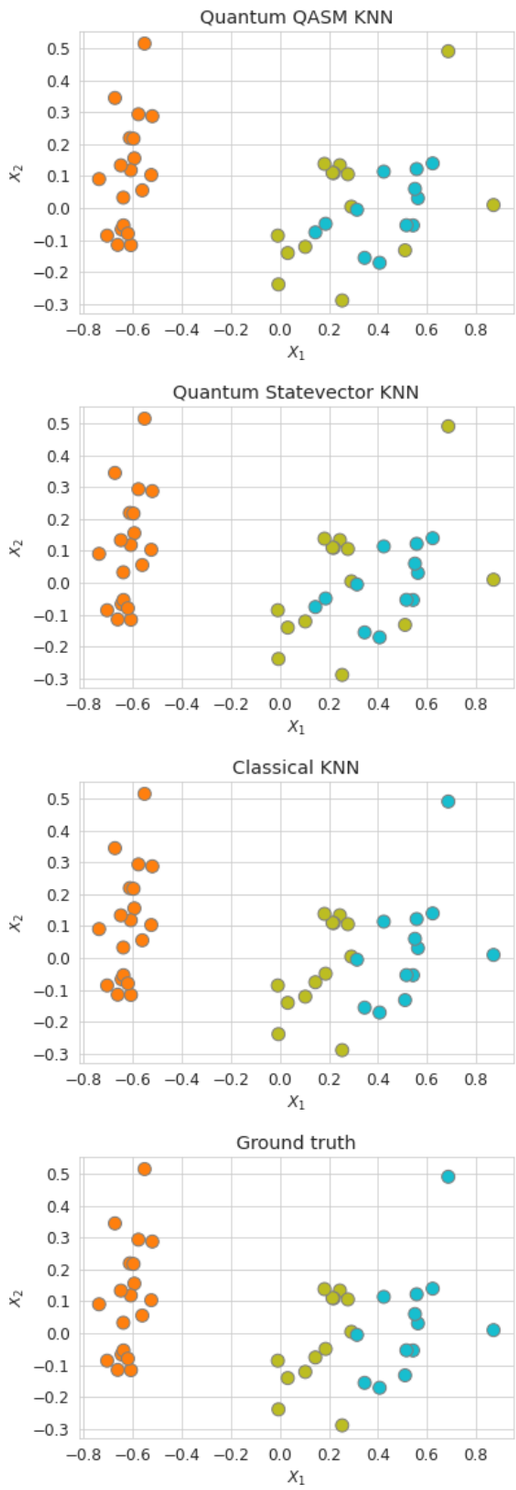
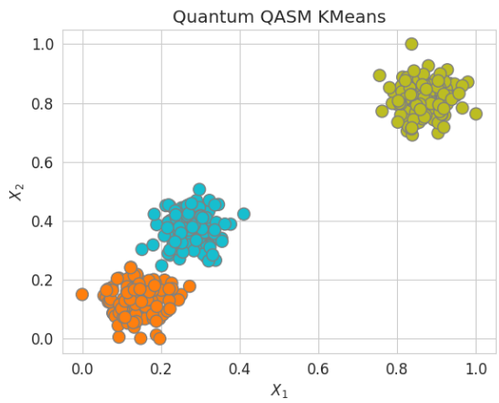


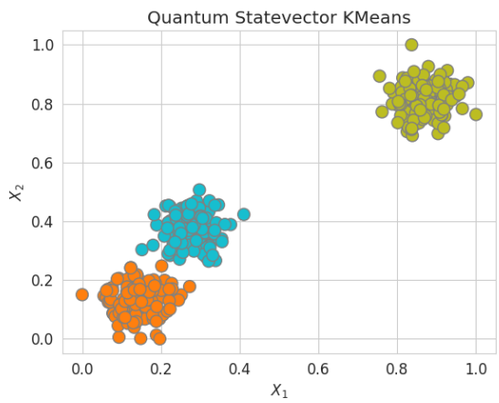
Table 1: *K-NN performance results*

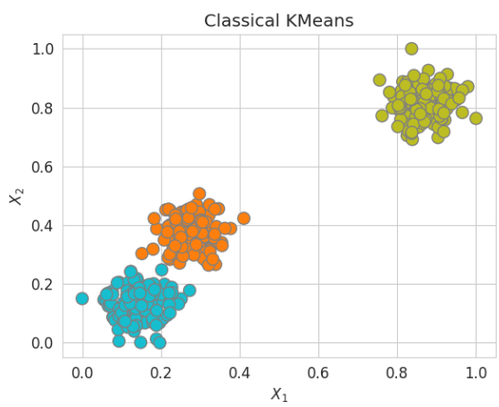
	ACC	F1	JAC
QASM K-NN	0.930	0.930	0.850
Statevector K-NN	0.930	0.930	0.850
Classical K-NN	1.000	1.000	1.000

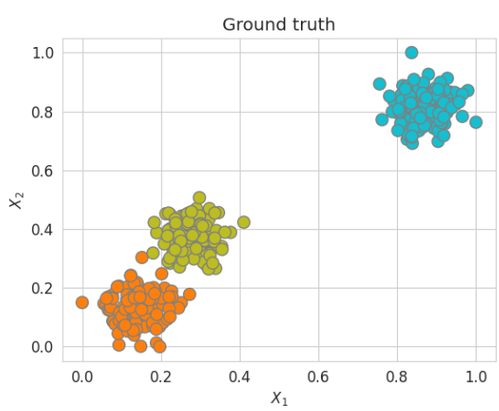
ii. Quantum K-Means

The Quantum K-Means clustering was tested on a synthetic dataset generated using scikit-learn's make_blobs. make_blobs generates isotropic Gaussian blobs for clustering. From this data we rescaled each feature into the $[0, 1]$ range. We tested the algorithm with both a noisy and a non-noisy backend, respectively qasm_simulator and statevector_simulator.









On top of the Silhouette score, we evaluated the performance of Quantum K-Means using Adjusted Rand Index and Completeness. The results of this analysis are shown in Table 3. More detailed informations can be found in [21].

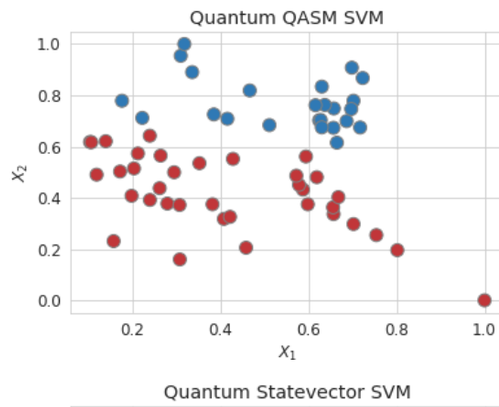
Table 2: *KMeans performance results*

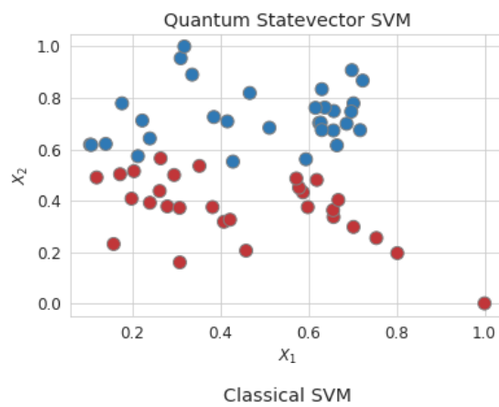
	SIL	ARI	COMP
QASM K-Means	0.747	0.978	0.968
Statevector K-Means	0.750	0.993	0.987
Classical K-Means	0.750	0.993	0.987

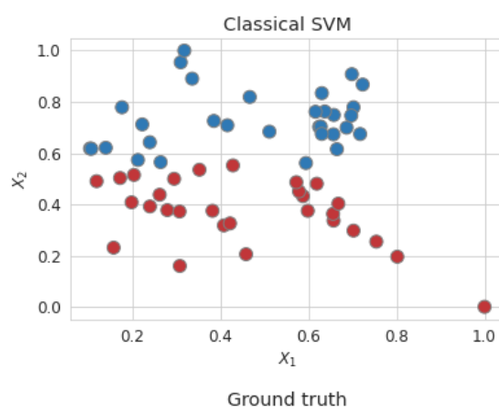
Our simulations shows that Quantum K-Means achieves very similar clusters to those of the classical K-Means algorithm. In particular, using a non-noisy backend, the Quantum K-Means achieves the exact same clusters of its classical counterpart. Even using different dataset, such as Iris and Wine, we got similar results.

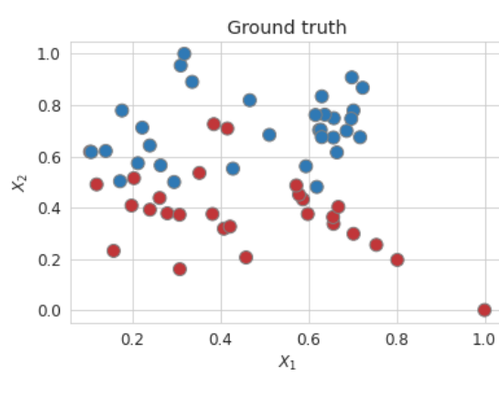
iii. Quantum Support Vector Machines

Quantum SVM was compared against a classical RBF SVM in terms of accuracy, F1-score and Jaccard index. The synthetic dataset used for this comparison was generated with scikit-learn's make_classification method, with each feature rescaled into the $[0, 1]$ range.









Using the qasm_simulator provided by Qiskit, the quantum model was able to fit the data well enough, correctly identifying the curved decision boundary

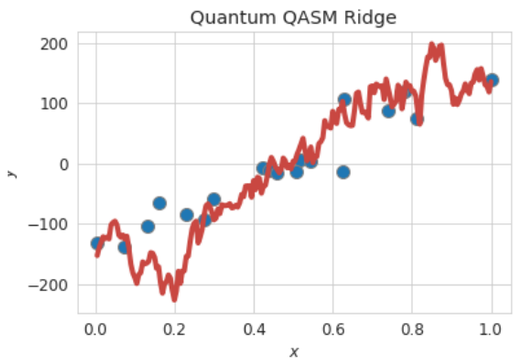
and assigning the right label to most of the data. Plenty of classification errors can be attributed to the noisy nature of said simulator. By comparison, a non-noisy backend such as statevector_simulator was able to obtain results almost identical to the classical SVM. Another example of how important noise reduction will be in the development of future quantum computers.

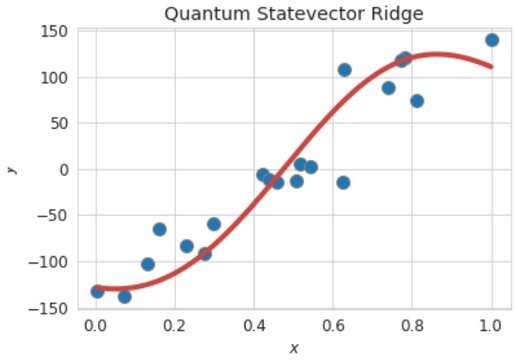
Table 3: *SVM performance results*

	ACC	F1	JAC
QASM SVM	0.800	0.786	0.647
Statevector SVM	0.883	0.889	0.800
Classical SVM	0.883	0.889	0.800

iv. Quantum Kernel Ridge Regression

Regression was tested on a synthetic dataset generated using scikit-learn's make_regression. The quantum model attained an high R^2 score, while was greatly outperformed by a classical one in terms of Mean Squared Error. The noise introduced by the qasm_simulator used is clearly visible and has skewed the results. statevector_simulator appears smoother. It's also worth noting that the kernel we chose, Qiskit's ZFeatureMap, is non-linear in nature while the data we tried to fit was. We preferred not to design an ad-hoc nonlinear dataset in order to show the interesting properties of quantum feature maps. We believe such an algorithm could still be employed successfully given efficient noise reduction and a suitable dataset.





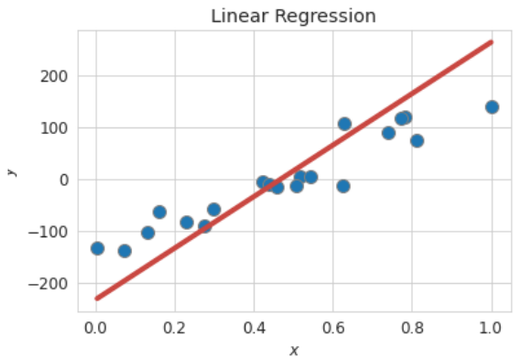


Table 4: *Ridge performance results*

	MSE	R^2
QASM Ridge	$4.999 * 10^2$	0.930
Statevector Ridge	$4.265 * 10^2$	0.940
Linear Regression	$1.375 * 10^{-27}$	1.000

VI. Conclusion

We showed that quantum-based algorithms can compete against optimized well-established classical ones in terms of accuracy. With the notable exception of Qiskit, no cohesive work exists that groups in a familiar and ready-to-use way quantum-based learning algorithms. Running our models on Qiskit proved efficient albeit limiting: as expected, simulating a quantum system requires a significant amount of overhead computation. This prevented our work from competing against classical algorithms in terms of execution time. It's also worth noting that the scope of our work was bound by the total number of qubits made available by simulators. Qlearnkit can be expanded with other algorithms for which a quantum formulation exists. NNs, PCA and Linear Regression are good candidates, as well as the more efficient SVM formulation mentioned above. An alternative to LS-SVM based on variational circuits is also being considered.

VII. Contributions

Giulio Corallo: Quantum K-Means, testing

Massimiliano Pronesti Quantum K-Nearest Neighbors Classifier and Regressor, repository management, continuous integration and deployment, documentation, testing

Federico Tiblias: Quantum Support Vector Classifier, Quantum Kernel Ridge Regressor, testing, data visualization

References

[1] Ashish Vaswani et al. "Attention is all you need". In: (June 2017). arXiv: 1706.03762 [cs.CL].

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[3] Lorenzo Buffoni and Filippo Caruso. "New Trends in Quantum Machine Learning". In: (Aug. 2021). arXiv: 2108.09664 [quant-ph].

[4] Jacob Biamonte et al. "Quantum Machine Learning". In: (Nov. 2016). arXiv: 1611.09347 [quant-ph].

[5] Frank Phillipson. "Quantum Machine Learning: Benefits and Practical Examples". In: (2020).

[6] Yao Zhang and Qiang Ni. "Recent Advances in Quantum Machine Learning". In: (2020). DOI: https://doi.org/10.1002/que2.34.

[7] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[8] Ville Bergholm et al. "arXiv:1811.04968". In: (2018).

[9] Daniël Kok. *Qiskit Quantum kNN: A pure quantum knn classifier for a gated quantum computer*. URL: https://qiskit-quantum-knn.readthedocs.io.

[10] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[11] MANUELA WEIGOLD et al. "Data Encoding Patterns for Quantum Computing". In: (2020). URL: https://hillside.net/plop/2020/papers/weigold.pdf.

[12] Afrad Basheer, A Afham, and Sandeep K Goyal. "Quantum *k*-nearest neighbors algorithm". In: (Mar. 2020). arXiv: 2003.09187 [quant-ph].

[13] Jing Li et al. "Quantum K-nearest neighbor classification algorithm based on Hamming distance". In: (Mar. 2021). arXiv: 2103.04253 [quant-ph].

[14] David Arthur and Sergei Vassilvitskii. "How Slow is the k-Means Method?" In: *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. SCG '06. Sedona, Arizona, USA: Association for Computing Machinery, 2006, pp. 144–153. ISBN: 1595933409. DOI: 10.1145/1137856.1137880. URL: https://doi.org/10.1145/1137856.1137880.

[15] Iordanis Kerenidis et al. "q-means: A quantum algorithm for unsupervised machine learning". In: (Dec. 2018). arXiv: 1812.03584 [quant-ph].

[16] David Arthur and Sergei Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245.

[17] Johan Suykens and Joos Vandewalle. "Least Squares Support Vector Machine Classifiers". In: *Neural Processing Letters* 9 (June 1999), pp. 293–300. DOI: 10.1023/A:1018628609742.

[18] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: (July 2013). arXiv: 1307.0471 [quant-ph].

[19] Vojtech Havlicek et al. "Supervised learning with quantum enhanced feature spaces". In: (Apr. 2018). arXiv: 1804.11326 [quant-ph].

[20] *Lecture 6 — February 9 6.1 Recap 6.2 Kernel Ridge Regression.* URL: https://people.eecs.berkeley.edu/~wainwrig/stat241b/lec6.pdf.

[21] *A demo of K-Means clustering on the handwritten digits data*. URL: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html.