Indian Journal of Pure & Applied Physics
Vol. 60, May 2022, pp. 407-414

Implementation of Quantum Support Vector Machine Algorithm Using a Benchmarking Dataset

Gurmohan Singh[a*], Manjit Kaur[a], Mandeep Singh[a] & Yadwinder Kumar[b]

[a]Centre for Development of Advanced Computing, Mohali, Punjab - 160 071 India

[b]Yadwindra College of Engineering, Talwandi Sab, Punjabi University, Patiala, Punjab - 151 302 India

Received 15 February 2022; accepted 19 April 2022

The evolution of quantum computers and quantum machine learning (QML) algorithms have started demonstrating exponential speed-ups. In machine learning problems, the efficient handling and manipulation of linear algebra subroutines defines the complexity of the task to be performed. Quantum computers handle big datasets in the form of vectors and matrix operations very efficiently. In this paper, quantum support vector machine (QSVM) algorithm is used to solve a classification problem using a benchmarking MNIST dataset of handwritten images of digits. Quantum SVM variational and kernel matrix algorithms are implemented to analyze quantum speedup on quantum simulator and physical quantum processor back-ends. The study compared classical and quantum SVM algorithms in terms of execution time and accuracy. The results explicitly prove quantum speed-up achieved by quantum classifiers on quantum back-ends for machine learning applications.

Keywords: Dirac notation; Hilbert space; Inner product; Machine Learning; Quantum bit; Principal Component Analysis; Support Vector Machine

1 Introduction

Continued developments in patternability of silicon transistors to nanoscale empowered modern computers to handle and manipulate large amounts of data. The contemporary complex applications require big data sets to be efficiently handled and manipulated by classical computers[1]. The growing size of data has started creating big challenges for classical computers in terms of performance and computing resources. The growing trend of big data has driven the need for a new computing architecture or approach to handle complicated big data problems[2]. The advancements in development of quantum computer and quantum inspired machine learning algorithms are promising to offer quantum speed-up over their classical counterparts[3]. In fact, even before the arrival of actual quantum computers researchers have started coming up with quantum machine learning algorithms that offer considerable time speed-up over the corresponding classical machine learning algorithms[4-6]. All these examples clearly bring out the fact that the quantum machine learning algorithms have the potential to offer considerable speedups over the corresponding classical algorithms. It is anticipated that quantum computing paradigm will facilitate handling of big datasets and offer solution for many intractable problems. It is also predicted that quantum computers are capable of searching unsorted big datasets[7], factorizing integers[8] and rapidly extracting the desired patterns. They have capability of searching for multiple data items concurrently and discovering pattern of importance only. The machine learning, artificial intelligence, big data analytics, financial modelling, molecular modelling *etc*. applications would get immensely benefitted from quantum computing revolution much before fully quantum solutions came into reality[9-10]. Quantum inspired algorithms are enabling a boost to machine learning and data analytics[11]. It is anticipated that machine learning is going to benefit most from developments in quantum computing field[12]. The way to success hides behind mapping the real-world problems to quantum space.

Artificial intelligence systems produce precise results provided machine learning algorithms employed for training supplied with bigger datasets. AI systems perform efficiently based upon how accurately the data is classified according to its particular attributes or features[13]. Quantum computers have capability to extract computationally complicated attributes of data which could reveal new conceptions hidden till now. The researchers have demonstrated that quantum supremacy is arriving faster than anticipated[14]. Machine learning is generally used in instances when there is no solution or formula for solving intricate

*Corresponding author: (E-mail: gurmohan@cdac.in)

problems and big datasets with multiple variables are involved. Machine learning evolved as a key tool to handle larger datasets for solving problems in different areas such as computational finance, computational biology, computer vision and image processing, and natural language processing *etc*. The ML algorithms learn natural patterns from data to generate greater perception facilitating in improved decision making and predictions. The use of quantum computation in machine learning is just not limited to the academic community but also the industry is looking forward towards it with hopes and aspirations. The day is not far when the quantum machine learning applications will be used to provide more efficient solutions to the common machine learning problems.

The paper is structured in following sections as: Section 2 presents preliminaries of quantum computing. Section 3 enumerates quantum machine learning and quantum support vector machine algorithm. Section 4 presents details of experiments conducted and a comparison of classical and quantum machine learning approaches. Section 5 concludes the paper.

2 Preliminaries

Quantum computers manipulate information encoded in form of quantum bits (Qubit). A qubit is most basic entity of quantum information or a quantum counterpart of a binary bit[12]. A qubit can be described as a two-state quantum-mechanical device which strictly follows rules of quantum mechanics with two-states labelled as $|0\rangle$ and $|1\rangle$ described by a two-dimensional (2D) vector space over the complex numbers C^2.

A qubit can be either in $|0\rangle$ and $|1\rangle$ state or in a random quantum state generally denoted as$|\Psi\rangle$. The random quantum state $|\Psi\rangle$ may be any superposition of $|0\rangle$ and $|1\rangle$ basis vectors computed using (1) as

$$|\Psi\rangle = \alpha\ |0\rangle + \beta\ |1\rangle \tag{1}$$

where, α and β are two complex numbers having probability of $|\alpha|^2 + |\beta|^2 = 1$. On measuring a qubit always collapses to classical values of 0 or 1 with probabilities of $|\alpha|^2$ or $|\beta|^2$; respectively. A qubit with phase can be represented using (2) as

$$|\Psi\rangle = \sqrt{p}\ |0\rangle + e^{i\varphi}\sqrt{1-p}\ |1\rangle \tag{2}$$

where, p is probability of a bit being in 0 state with limits $0 \leq p \leq 1$, and quantum phase is $0 \leq \varphi < 2\pi$. The unitary transformations cause a change in qubit or qubit-based systems. The unitary transformation of a qubit is equivalent to a quantum gate (Q-gate) operation[12]. The Hadamard (H) gate is a fully quantum gate extensively used in all quantum computations. It performs superposition operation which is one of the most basic requirements of all quantum computations transforming finite quantum state of qubits to superposition state to leverage their full quantum capability. The controlled-NOT (C-NOT) gate is used to realize entanglement operation in quantum computing.

Researchers are exploring various physical implementations of qubits. The photon polarization, an ion's discrete energy levels, an electron's spin, nuclear spin states of an atom, and superconducting Josephson junction *etc*. are being investigated for physical qubit realizations. The qubits in quantum computers need to be coupled amongst them-selves for a meaningful quantum operation. The stringent requirements are that qubits must be disengaged from outer environment with exception of only control, readout and writing accessibility. Most of existing qubit implementations are microscopic like nuclei or electron spin, atoms or ions *etc*. However, the superconducting qubits are macroscopic. The two most important parameters are coherence and quantum noise associated with any qubit implementation and present several new research areas for further exploration.

3 Quantum Machine Leaning

Quantum computers process the information using sub-atomic level particles based on quantum mechanical principles[12]. The intersection of machine learning and quantum computing gives birth to a field popularly known as quantum machine learning (QML). In QML, the data is processed on a quantum machine taking advantage of quantum mechanical properties. The quantum algorithms may offer great enhancement in computing speed for many intractable problems still remaining unrealistic for classical supercomputing machines till date. It is important to note that quantum advantage is not applicable in all cases. The quantum computing leverages from the immense data encoding capability of quantum bits due to physical properties of sub-atomic particles and their interactions i.e. superposition property. Quantum computations experiences slowdown as qubits are entangled and extracting their final state is a complex task. But, due to super-dense coding property,

quantum computers are excellent in handling and executing big data sets and significantly minimizing the space/time contemplations especially in case of machine learning.

A quantum algorithm generally involves encoding of classical or quantum data in quantum space, applying unitary transformation operations on input qubit sets, and finally measuring the qubit(s) state to extract a classical output. Many researchers have demonstrated experimental and theoretical implementation of quantum support vector machine (QSVM) machines using qubits to find solution of machine learning problems. J. Biamonte *et al.*[4] exemplified relationship between quantum computation and machine learning. Machine learning can harness advantages offered by quantum algorithms using quantum principles to overcome computational complexity problems. They have highlighted quantum speedup that could be achieved using Bayesian inference, online perceptron, quantum PCA, and quantum SVM. Ciliberto *et al.*[15], discussed that increase in computational complexity and data availability have transformed machine learning algorithm leading to remarkable results. They have also discussed about computational costs associated with use of linear algebra, neural networks, sampling and optimization. Dunjko *et al.*[10] discussed different quantum algorithms such as quantum SVM and quantum PCA which have been mathematically proved to be providing quantum speed-up in machine learning and artificial intelligence applications. Schuld *et al.*[16] explained quantum machine learning algorithms for handling big data in machine learning. They have further discussed classical machine learning types, quantum gates, and various quantum machine learning algorithms. Havlicek *et al.*[17] demonstrated implementation of two quantum machine learning algorithms for classification problems on real time noisy intermediate scale quantum (NISQ) superconducting processors. Kerenidis & Prakash[18] have proposed a quantum machine learning algorithm for the recommendation system and have achieved polylogarithmic speedup. Inspired by their work on recommendation systems E. Tang[6] designed a QML based recommendation algorithm that can achieve an exponential improvement.

Support vector machine is the most widely used algorithm in supervised machine learning due to its simplicity, high accuracy, and lesser computational resource requirements[19]. In this algorithm, the main aim is to identify a hyper plane in n-dimensional space which explicitly segregates feature vectors into two classes. This hyper plane acts as delineating line between two classes as depicted in Fig. 1. The finding of a plane with maximum margin (distance) among feature vectors of two classes is the main target. The reason behind choosing maximum margin is to keep some space for classifying future feature vectors with higher accuracy. The number of input feature vectors decides the dimensions of the hyperplane. For a 2-dimensional feature space, a hyperplane is a line and for a 3-dimesional feature space, a hyperplane becomes a 2-dimensional plane. The feature vectors close to the hyperplane are called support vectors as shown in Fig. 1. The orientation and position of hyperplane is mainly decided by the support vectors. The maximum margin can be optimized using support vectors. The classes in SVM are labelled with output '1' and '-1' in accordance with output of a linear function. The margin is determined by threshold values of '-1' and '1' in SVM.

A SVM algorithm learns from a similarly scattered and independent training data$\{ (x_1, v_1) \ldots\ldots\ldots\ldots\ldots (x_n, v_n) \}$, here $v \in \{-1, 1\}$ are two classes with respective threshold values of '-1' and '1'. A hyperplane is mathematically represented using (3) as

$$w^T x - b = 0 \tag{3}$$

where, $\vec{w}$ represents vector normal to hyperplane and b is bias parameter which decides hyperplane offset from origin. It is assumed that data classes are linearly separable. The margin between two planes is given by $\left\| \frac{2}{w} \right\|$.

By minimizing w, maximum margin could be achieved. Mathematically, the output of linear SVM is determined using (4) as

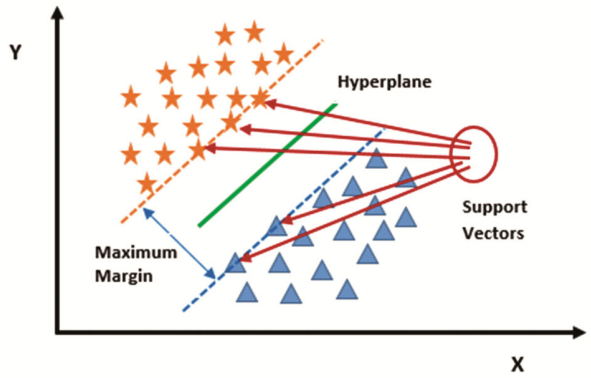

Fig. 1 — Concept of hyperplane and support vectors.

$$\hat{v}_i \;=\; sgn\;\;(w\;x_i + b) \tag{4}$$

where, x_i is the i^{th} instance of training data. The data items falling out of margin are $w^T x_i - b \geq 1$ for $v_i = 1$ and $w^T x - b < 1$ for $v_i = -1$. Both the above conditions can be written jointly as (5)

$$v_i\;(w^T x_i - b) \geq 1 \text{ for} 1 \ldots\ldots N \tag{5}$$

The hyperplane with maximum margin can be attained within these conditions is $\underset{w.\,b}{argmin}\frac{1}{2}\|w^2\|$.

This is valid for linear formulation of linear time complexity solution. In dual formulation [20], the aim is to maximize the function using Kuhn-Tucker multipliers (α_i), the expression modifies to (6) as

$$L(\vec{\alpha}) \;=\; \sum_{i=1}^{N} v_i\;\alpha_i \;=\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\;\alpha_j\;K_{i\;j} \tag{6}$$

with conditions $\sum_{i=1}^{N}\alpha_i \;= 0$ and $v_i\;\alpha_i \;\geq\; 0$. The hyperplane parameters b and $\vec{w}$ can be computed as $b = \;v_i\; - \vec{w}\;x_i$ and $\vec{w}\; = \sum_{i=1}^{N}\alpha_i\;x_i$. Here, the $K_{i\;j}(\vec{x}_i\,,\;\vec{x}_j) = \vec{x}_i\;*\;\vec{x}_j$ introduces a concept of kernel matrix[21] with kernel function$k\;(x\;,\;x')$. The classical support vector machine algorithm produces binary classifier[9] for new data vector $\vec{x}_b$ as described in (7) as

$$v\;(\vec{x}_b) = sgn\;\left(\sum_{i=1}^{n}\alpha_i\;k\;(\vec{x}_i\;,\;\vec{x}_b)\;+b\right) \tag{7}$$

where, the sgn () function is

$$sgn\;\begin{cases}1, & for\;\vec{x}\;\geq 0\\ -1 & for\;\vec{x}\;< 0\end{cases}$$

In quantum SVM with kernel matrix, it is assumed that oracles to train the data return quantum vectors $|\vec{x}_i\rangle = {}^{1}\!/_{|\vec{x}_i|}\sum_{j=1}^{N}(\vec{x}_i)_j|k\rangle$. The normalized kernel matrix is computed[9-10] using (8) as

$$\widehat{\mathrm{K}} = \frac{\mathrm{K}}{\mathrm{tr\;K}} = \frac{1}{\mathrm{N}_\chi}\sum_{\mathrm{i,j=1}}^{\mathrm{N}}\langle\vec{\mathrm{x}}_\mathrm{j}|\vec{\mathrm{x}}_\mathrm{i}\rangle\;|\vec{\mathrm{x}}_\mathrm{i}|\big|\vec{\mathrm{x}}_\mathrm{j}\big||\mathrm{i}\rangle\langle\mathrm{j}| \tag{8}$$

where,

$$N_\chi = \;\sum_{i=1}^{N}|\vec{x}^2|and|\chi\rangle = \;{}^{1}\!\Big/_{\sqrt{N_\chi}}\sum_{i=1}^{N}|\vec{x}_i|\;|i\rangle|\vec{x}_i\rangle$$

To compute matrix inverse $(\widehat{K}^{-1})$ quantum mechanically, the operation $e^{-i\,\hat{K}\,\Delta t}$ must be computed very efficiently. It is observed that operation $e^{-i\,\hat{K}\,\Delta t}$ computationally correct with an error of $O(\Delta t^2)$[10, 21]. With introducing slack variable e_i and replacing inequality constraint with equality constraints as described by (9) as

$$v_i(\vec{w}\;\vec{x}_i + b) \geq 1 \;\rightarrow (\vec{w}\;\vec{x}_i + b) = v_i - v_i\;e_i \tag{9}$$

Further, the application of implied Lagrange function produces an additional term $\frac{\gamma}{2}\sum_{n=1}^{N}e_i^2$.

This term is defined by user and calculates training error relative weightage and responsible for overall aim of SVM. The least square approximation of problem could be solved using (10) as

$$F\;\begin{pmatrix}b\\ \vec{\alpha}\end{pmatrix}\;\equiv\;\begin{pmatrix}0 & \vec{I}^T\\ \vec{I} & K\;+\;\gamma^{-1}I\end{pmatrix}\begin{pmatrix}b\\ \vec{\alpha}\end{pmatrix}\;=\begin{pmatrix}0\\ \vec{\mathrm{v}}\end{pmatrix} \tag{10}$$

where, I is unit matrix and $\vec{v}$ denotes training data labels. The parameters b and $\vec{\alpha}$ decides the value of SVM classifier. A new data point $\vec{x}_b$ can be categorized with the help of (11) as

$$v\;(\vec{x}_b) = \;\mathrm{sgn}\qquad(\vec{w}\;\vec{x}_b + b) = \mathrm{sgn}\;\left(\sum_{i=1}^{N}\alpha_i\;k\;(\vec{x}_i\;\vec{x}_b) + b\right) \tag{11}$$

The classical SVM is generally framed as a quadratic programming problem solvable in time $O\;(log\;(\varepsilon^{-1})\;poly\;(N,M)$. Here, ε is accuracy, N is dimensionality of feature space, and M is number of training vectors. The major difference between classical SVM and quantum SVM algorithm is time taken to solve the problem. The quantum SVM take time logarithmic while classical SVM take time polynomial in handling dimensionality of vector space and the number of vectors [3, 9]. Thus, quantum SVM is capable of providing exponential speed advantage over its classical counterpart. The machine learning is basically about manipulation and classification of large datasets. Quantum computers are exceptionally better in handling and execution of large vectors and matrices of their inner products in high dimensionality vector spaces. In quantum SVM, firstly the classical data is transformed to quantum states over $log_2 N$ quantum bits taking $O\;(log_2 N)$ mapping steps. Then, the data in quantum form is processed using various quantum operations such as matrix inversion and Quantum Fourier Transform (QFT) *etc*. The superiority of quantum operations lies in anticipating distances and inner products among the post-processed vectors in exponential lesser time as compared to classical operations. As compared to

classical SVM's problem solving time of $O\,(poly\,(N\,M)$, quantum SVM take a problem-solving time of $O(log_2(NM))$ only. The other major advantage of processing classification problems on quantum computers is that quantum states stored in quantum random access memory (QRAM) can be accessed in parallel[3,9]. The Table 1 lists computational complexity of classical and quantum SVM algorithms used for classification of MNIST dataset in this study.

4 Experiments and Results

The benchmarking MNIST dataset[22-23] of handwritten digits comprising of 60000 images of training data and 10000 images of test data has been used in this experiment. There are two files named train.csv and test.csv which contain grey-scale images of handwritten digits, from 0 to 9. Each handwritten black and white image comprise of digits of 28 x 28 pixels. The digits are centered and size-normalized in a fixed-size image. The training dataset file train.csv has 785 columns. It contains label in the 1st column and remaining columns contain pixel value of that image.

The test dataset file test.csv contain same columns except label column. The accuracy of MNIST dataset is defined as percentage of correctly classified images in the test set. The whole process of applying quantum machine learning is divided into various steps such as dataset selection, data pre-processing & visualization[24], exploratory data analysis (EDA)[25], algorithm selection, principal component analysis (PCA), classification using quantum support vector machine algorithm (variational/kernel based approach)[9,21,26], quantum circuit generation, and result readout and visualization. The selection of dataset is critical factor for finding an optimal solution to a classification problem. The selected dataset must have sufficient training and data vectors. Many data pre-processing steps such as rescaling, normalization, formatting, binarizing, and cleaning along with different visualization plots are generally required for datasets to develop robust machine learning models. Data pre-processing is mainly performed to identify null and missing values which further help to identify corrupted images within the dataset. Before selecting a machine learning approach to solve a problem, one need to find answer to questions like suitability of algorithms for a dataset and feature variable selection of data. EDA can make sure that results are valid, correct, and applicable to the problem. It is performed after validation of raw data, anomaly checking, and ensuring error free dataset. The algorithm must be capable of providing very high accuracy and faster processing. The principal component analysis (PCA) is an expedient statistical way of discovering patterns in high dimensionality data[27-28].

It is usually used when the number of features/variables in datasets become very high or in simple words it is to scale down the dimensionality while preserving the variations in the dataset. The features are mapped to a new set called principal components (PCs). They are eigen vectors of the covariance matrix and are ordered and orthogonal. The Table 2 lists the steps taken to perform PCA in this experiment.

For classical algorithms, PCA executes in $O(d^2)$ in terms of query and computational complexity. In case of quantum PCA[28], the data vector $\vec{a}$ is mapped to a quantum state $|a_j\rangle$ having $log(\,d)$ qubits with QRAM requiring only $O(d)$ operations divided by $log(\,d)$ steps which can be executed in parallel. The density matrix equivalent of co-variance matrix for the chosen quantum state $|a_j\rangle$ described using (12) as

$$\rho = \left({}^{1}/_{n}\right) \sum_j |a_j\rangle\langle a_j| \tag{12}$$

where, n are data vectors. Then, quantum data is sampled repeatedly, followed by density matrix exponentiation and quantum phase estimation operations[28] resulting in eigen vector and eigen values of matrices. These operations permits decomposing $|a_j\rangle$ to its principal components $|c_k\rangle$, the eigen value of co-variance matrix is computed with help of (13) as

Table 1 — Computational complexity of the algorithms

Algorithms	Time
Classical SVM	O (poly (NM))
Quantum SVM	O (log (NM))

Table 2 — Principal Component Analysis Steps

a) Choosing data-points: Shape of sample data = (15000, 784)

b) Data standardizing: (15000, 784)

c) Computing the co-variance matrix ($C = \sum_j \vec{a}\,\vec{a}^T$), where, $\vec{a}$is data vector and $\vec{a}^T$is it's transpose The shape of co-variance matrix is (784, 784)

d) Diagonalizing of co-variance matrix ($C = \sum_k e_k \vec{c_k}\,\vec{c_k}^T$), where $\vec{c_k}$ is eigen vector and $e_k$its eigen value. The shape of eigen vector is (2, 784)

e) Projection of original data sample on the plane formed by principal eigen vectors by vector-vector multiplication i.e. resultant new data point's shape becomes (2, 784) x (784, 15000) = (2, 15000)

$$|a_j\rangle \rightarrow \sum_k a_k |c_k\rangle \left|\vec{e_k}\right\rangle \quad (13)$$

As compared to its classical counterpart, the quantum PCA executes in $O((log\, d)^2)$ in terms of query and computational complexity.

In this experiment, the accuracy and execution time of classical and quantum support vector machine classifiers have been computed. To solve machine learning and problems in other application areas, IBM has developed IBM® Quantum Experience[29] and IBM® application programming interfaces (APIs)[30] to access its real time quantum devices and simulators. In this study, the real quantum processors (IBMQ_16_ Melbourne and IBMQX2) and IBMQ_QASM simulator have been deployed as back-ends. The quantum SVM's variational and kernel-based approaches are used in our machine learning classification problem.

In QSVM variational approach, hyperplane(s) are computed to classify new test data. It can handle classification problems having even more than two classes. But, this approach uses two quantum algorithms making it computationally more intensive. The first algorithm calculates hyperplane(s) from available training data whereas the second algorithm perform the classification of new test data. In QSVM kernel-based approach, only one algorithm is used and it is primarily used for binary classification problems.

At the onset, a kernel matrix is computed from training data using a quantum machine, then a classical machine computes support vectors from it. Thereafter, the classification of the test data is performed.

The Table 3 illustrates the list of the algorithms, computational back-ends, datasets, and performance metrics used in this study.

Table 3 — Experiment set-up details

Algorithm used	Computational Back-end	Dataset	Targeted Metrics
Classical SVM	Local CPU environment	MNIST	Accuracy, execution time
QSVM Variational Approach	Local CPU environment, IBMQX2, IBMQ_16_ Melbourne, IBMQ_QASM simulator	MNIST	Accuracy, execution time
QSVM Kernel Approach	Local CPU environment, IBMQX2, IBMQ_16_ Melbourne, IBMQ_QASM simulator	MNIST	Accuracy, execution time

The Table 4 describes the execution time results for accuracy achieved using different back-ends and MNIST dataset of handwritten digits.

The results in Table 4 reveal that QSVM variational algorithm is too much time consuming as it uses two different quantum algorithms making it computationally more intensive. Compared to classical SVM run on local CPU environment, the QSVM variational algorithm run on QASM simulator is 81.37 % more time consuming. QSVM kernel matrix algorithm run on IBM QASM simulator back-end is computationally least intensive as compared to both classical SVM and QSVM variational algorithms run on local CPU and QASM simulator, respectively. It took only 139 millisecond (ms) to produce the classification results. It is 99.44 and 99.98 % computationally more efficient than classical SVM and QSVM variational algorithms run on local CPU and QASM simulator, respectively. In this study, QSVM kernel matrix algorithm is also simulated on two real time quantum processor back-ends IBMQX2 and IBMQ_16_Melbourne to obtain the classification results for MNIST dataset. The QSVM kernel matrix algorithm run on real quantum devices IBMQX2 and IBMQ_16_Melbourne is 39.64 and 81.62 % computationally more efficient than classical SVM run on local CPU machine.

The results in Table 5 disclose that accuracy of classical SVM algorithm run on local CPU is 91.26 %. The QSVM algorithm executed on QASM simulator exhibits accuracy of 95 %.

The highest accuracy of 97.5 % is exhibited by QSVM kernel matrix algorithm run on IBMQX2, IBMQ_16_Melbourne, and IBM QASM simulator. As compared to classical SVM, the quantum SVM variational algorithm display 3.9 % improvement in

Table 4 — Execution time results

Algorithm used	Computational Back-end	Execution Time (s)
Classical SVM	Local CPU environment	248.64
QSVM Variational	QASM Simulator	1335
QSVM Kernel Matrix	IBMQX2	89.6
	IBMQ_16_Melbourne	45.7
	IBM QASM Simulator	0.139

Table 5 — Accuracy results

Algorithm used	Computational Back-end	Accuracy (%)
Classical SVM	Local CPU environment	91.26
QSVM Variational	QASM Simulator	95
QSVM Kernel Matrix	IBMQX2	97.5
	IBMQ_16_Melbourne	97.5
	IBM QASM Simulator	97.5

Fig. 2 — (a) Labelling of handwritten digits, and (b) Images after separating the training pictures of the numbers from their label and after reshaping to have digit image of 28×28 pixels.

accuracy whereas quantum SVM kernel matrix algorithm exhibit 6.4 % improvement. In comparison to quantum SVM variational algorithm, the quantum SVM kernel matrix algorithm show 2.56 % improvement in accuracy.

Labelling of data is a very crucial step in data pre-processing helping in identification of features and characteristics in the dataset. To develop an efficient machine learning model for regression, classification, and pattern recognition problems, it is foremost requirement to select only explanatory, unique, and astute features within the dataset. Precisely labelled data makes the basic foundation for validation of the model. Fig. 2 (a) enumerates the labelling of handwritten digits performed during data pre-processing steps. The Fig. 2(b) depicts the few handwritten images obtained after separating the training pictures of the numbers from their labels and after reshaping to have digit image of 28×28 pixels.

After pre-processing, visualization, PCA, classical and quantum SVM classifiers are applied to solve the classification problem on different back-ends to produce accuracy and execution time results. Fig. 3 (a), (b) & (c) enumerates visualization of handwritten image of digit 9 with its pixel values after separating the training images from their labels and after reshaping to make it a 28×28 pixels, final readout image obtained after applying classical and quantum SVM algorithms.

Fig. 3 — (a) Visualization of handwritten image of digit 9 with its pixel values after separating the training images from their labels and after reshaping to make it of 28×28 pixels, (b) final readout image obtained after applying classical SVM algorithm, and (c) final readout image obtained after applying quantum SVM algorithm.

5 Conclusion

This study demonstrated that machine learning problems with large datasets could be solved taking quantum advantage offered by quantum machine learning algorithms. A benchmarking MNIST dataset comprising of handwritten images has been used for pattern classification problem. The classical and quantum SVM classifiers are used to solve the classification problem. The quantum classifiers explicitly demonstrate speed advantage over their classical counterpart. The quantum simulator and superconducting quantum processors are used as back-ends to run the quantum algorithms. The QSVM kernel matrix algorithm implemented on 5 and 14 qubit real quantum processors is 39.64 and 81.62 % computationally more efficient than classical SVM run on local CPU machine. Further, the quantum SVM kernel matrix algorithm exhibit 6.4 % better accuracy as compared to classical SVM. The results clearly indicate quantum algorithms can accelerate the execution time to solve many complex machine learning problems as compared the classical machines. Based on results, it can be concluded that very soon quantum computers capable of executing feature mapping or classification on big data sets would even surpass existing classical supercomputing machines.

Acknowledgments

Authors are highly grateful to IBM Inc. for providing access to their software and hardware computational resources through cloud network.

Declaration

Authors declare that there are no conflict of interests.

References

1 Tsai C, Lai C & Chao H, *J Big Data*, 2 (2015) 1.
2 Fisher D, DeLine R, Czerwinski M & Drucker S, *Interactions*, 19 (2012) 50.
3 Saini S, Khosla PK, Kaur M & Singh G, *Int J Theor Phys*, 59 (2020) 4013.
4 Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N & Lloyd S, *Nature*, 549 (2017) 195.
5 Du Y, Hsieh M H, Liu T, & Tao D, *Phys Rev Research*, 2(3) (2020) 33199.
6 Tang E, *In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, (2019) https://doiorg/101145/33132763316310.
7 Grover L K, *Phys Rev Lett*, 79 (1997) 325.
8 Shor PW, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, (1994) https://doiorg/101109/SFCS1994365700.
9 Rebentrost P, Mohseni S & Lloyd S, *Phys Rev Lett*, 113 (13) (2014) 30503.
10 Dunjko V & Briegel H J, *Rep Prog Phys,* 81(7) (2018) 074001.
11 Chen D & Tian Y B & He-Liang H, arXiv preprint arXiv:190608902 (2019).
12 Nielsen M A & Chuang I L, *Cambridge: Cambridge University Press*, (2000).
13 Ying M, *Artificial Intel*, 174 (2) (2010) 162.
14 Arute F, Arya K & Babbush R, *Nature,* 574 (2019) 505.
15 Ciliberto C, Herbster M, Ialongo A D, Pontil M, Rocchetto A, Severini S & Wossnig L, *Proc R Soc A*, 474 (2018) 20170551.
16 Schuld M, Sinayskiy I & Petruccione F, *Contemporary Phys*, 56 (2014) 172.
17 Havlíček V, Córcoles A D & Temme K, *Nature*, 567 (2019) 209.
18 Kerenidis I & Prakash A, *8th Innovations Theoretical Computer Sci Conf, ser Leibniz International Proceedings in Informatics (LIPIcs), 67, Berkeley, CA, USA*, 49 (2017) 1.
19 Hearst M A, Schölkopf B, Dumais S, Osuna E & Platt J, *IEEE Intelligent Systems*, 13 (1998) 18.
20 Wittek P, *Academic Press*, (2014) 73.
21 Muller K-R, Mika S, Ratsch G, Tsuda K & Scholkopf B, *IEEE Transactions on Neural Networks*, 12(2) (2001) 181.
22 Lecun Y & Cortes C, *The MNIST database of handwritten digits* [Online] *Available: http://yannlecuncom/exdb/mnist/* 1998.
23 Lecun Y, Bottou L, Bengio Y & Haffner P, *Proc IEEE*, 86 (1998) 2278.
24 Famili A, Shen W, Weber R & Simoudis E, *Intel Data Anal*, 1 (1997) 3.
25 Komorowski M, Marshall D C, Salciccioli J D & Crutain Y, *Secondary Analysis of Electronic Health Records Springer, Cham https://doiorg/101007/978-3-319-43742-2_15.*
26 Kandala A, Mezzacapo A & Temme K, *Nature*, 549 (2017) 242.
27 Yan S, Xu D, Zhang B, Zhang H J, Yang Q & Lin, S, *IEEE Trans Pattern Anal Mach Intel*, 29 (2007) 40.
28 Lloyd S, Mohseni M & Rebentrost P, *Nature Phys*, 10 (2014), 631.
29 *IBM Inc, https://quantumexperiencengbluemixnet/qx/experience.*
30 *IBM Inc*, *https://qiskit.org/aqua.*