Matrix Product State–Based Quantum Classifier

Amandeep Singh Bhatia
amandeepbhatia.singh@gmail.com
Mandeep Kaur Saggi
mandeepsaggi90@gmail.com
Ajay Kumar
ajayloura@gmail.com
Sushma Jain
sjain@thapar.edu
Department of Computer Science, Thapar Institute of Engineering and Technology, Patiala 147004, India

Interest in quantum computing has increased significantly. Tensor network theory has become increasingly popular and widely used to simulate strongly entangled correlated systems. Matrix product state (MPS) is a well-designed class of tensor network states that plays an important role in processing quantum information. In this letter, we show that MPS, as a one-dimensional array of tensors, can be used to classify classical and quantum data. We have performed binary classification of the classical machine learning data set Iris encoded in a quantum state. We have also investigated its performance by considering different parameters on the ibmqx4 quantum computer and proved that MPS circuits can be used to attain better accuracy. Furthermore the learning ability of an MPS quantum classifier is tested to classify evapotranspiration (ET_0) for the Patiala meteorological station located in northern Punjab (India), using three years of a historical data set (Agri). We have used different performance metrics of classification to measure its capability. Finally, the results are plotted and the degree of correspondence among values of each sample is shown.

1 Introduction

Quantum computing deals with theoretical computational systems, combining visionary ideas of computer science, physics, and mathematics. It concerns the behavior and nature of energy at the quantum level to improve the efficiency of computation. Feynman (1982) initially proposed the idea of quantum computing in 1982 after performing a quantum mechanics simulation on a classical computer. Quantum computing relies on the quantum phenomena of entanglement, superposition, and interference to perform operations, which are generally considered resources for its speed.

Although the full influence of quantum computing is probably more than a decade away, it has the potential to transform information processing and promises a wide range of applications in the areas of quantum chemistry, high-energy physics, and condensed matter, which are not tractable on classical computers.

In the past decade, the simulation of open and closed quantum systems has had an overwhelming response, with the study of tensor network theory taking a central role in quantum physics, simulation, and machine learning. Tensor network states are a new language, based on entanglement, for quantum many-body systems (Gao & Duan, 2017). Tensor network states are classified on the basis of dimensions along which the tensors are traversed. It is widely used to simulate strongly entangled correlated systems and to represent quantum states and circuits (Schuld, Bocharov, Svore, & Wiebe, 2018). *Tensor network methods* is the term associated with the tools, which are widely employed in experimental and quantum theoretical applications of machine learning.

The matrix product state (MPS) is the most prominent example of tensor network states. It can be observed by maximum entanglement entropy without forfeiting one-dimensional distributions. Matrix product tensor networks have the ability to surround the entire input or output state space. By using classical resources, tensor networks have shown impressive results for supervised and unsupervised learning tasks (Pestun & Vlassopoulos, 2017). For large dimensions, tree tensor networks (TTNs) and multiscale entanglement renormalization ansatz (MERA) have shown to be equivalent to neural networks. These methods have seen many breakthroughs and transformations in different areas of physics, mathematics, and computer science.

Matrix product states have a wide range of practical applications: supervised learning (Stoudenmire & Schwab, 2016), quantum dynamics (Bhatia & Kumar, 2018b), simulating quantum computation (Grant et al., 2018), quantum finite state machines (Bhatia & Kumar, 2018a), unsupervised learning (Han, Wang, Fan, Wang, & Zhang, 2018), simulating MPS on a quantum computer (Bhatia & Saggi, 2018), quantum machine learning MPS (Biamonte, 2018), and many more. These states are complete, with low entangled states represented efficiently, which is not possible with large-dimension tensor network states. MPS can also be employed in various emerging technologies, such as quantum cryptography, optical computing, dynamics quantum clustering, and image recognition. It obeys one-dimensional area law, is dense in nature, offers finitely correlated tensors and translational invariance, and is suited for describing higher-dimensional systems too. Recently, MPS methods have been introduced to compress the weights of neural network layers and classify images.

Through the effective deployment of information and communication technologies (ICTs), India's agriculture sector has been transformed. The combination of ICTs and analytics can provide novel ways and ideas to do socially accepted and profitable agriculture. It will be also beneficial for the environment (e.g., soil, water, climate). There is now a need for innovation in technological, economical, and social agriculture. As the technology rapidly spreads, big data analytics and quantum machine learning will be the key to fostering a new revolution in agriculture. The technology of quantum computers has evolved to solve real-world problems based on historical data, machine-generated data, and real-time streaming data. Quantum computing can also bring a revolution through its ability to handle experimental data, which can produce numerous solutions in various areas such as health care, smart cities, and smart agriculture etc (Saggi & Jain, 2018). In developing countries, farmers with limited knowledge and skills are significant factors. Some tough questions need to be addressed. Can farmers manage their farms and grow and harvest the crops too? Can they build smart networks for field-testing the next generation of quantum computers, which promise to revolutionize complicated data processing problems. They will be able to handle more data efficiently and can surpass conventional systems. Quantum machine learning techniques are also closely tied to a variety of application areas.

In this letter, we consider the supervised machine learning task of classifying the Iris and climatic data sets on quantum computers using and MPS quantum classifier. Section 2 examines related work. In section 3, we described encoding of data and qubit efficient MPS classifiers. In section 4, we look at model development and experimental results. Section 5 concludes.

2 Prior Work

The combination of neural networks and classical machine learning models with the efficiency of quantum computing has experienced incredible responses in the past few years. Initially, Harrow, Hassidim, and Lloyd (2009) designed a quantum algorithm to approximate the features of solving a set of N linear equations that runs in polynomial time. It is exponentially faster than the best classical algorithm. Rebentrost, Mohseni, and Lloyd (2014) presented a quantum support vector machine that can be implemented on a quantum computer with the complexity $O(\log NM)$, logarithmic in the size of training and classification stages. It has been observed that in contrast to classical algorithms, exponential speed is gained.

Accurately estimating evapotranspiration (ET_o) is a crucial issue for agricultural planning because it plays a significant role in irrigation water scheduling designed to use water efficiently. Evapotranspiration is a vital component of the hydrological cycle, and there are a number of alternative models for representing ET_o processes. It can be measured directly by experimental techniques—for example, eddy covariance systems, lysimeters, and Bowen ratio energy balance (Kool et al., 2014; Martí, González-Altozano, López-Urrea, Mancha, & Shiri, 2015), but these methods are complex and unavailable in many regions (Allen, Pereira, Raes, & Smith,

1998) because of their high cost. Therefore, mathematical models for ET_o estimation are needed. To further support a range of ET_o modeling studies, there is a need to facilitate the implementation of different ET models conveniently, consistently, and efficiently. Some software packages focus on ET_o modeling. Recently, Saggi and Jain (2019) developed a water model framework based on deep learning for predicting the ET_o of stations in Patiala and Hoshiarpur.

Otterbach et al. (2017) investigated a hybrid quantum algorithm for an unsupervised learning task known as clustering on a 19-qubit quantum computer. They showed that noise is enabled by using gradient-free Bayesian optimization and offer an optimal solution for all random problem instances. Recently, Farhi and Neven (2018) introduced the concept of classification with quantum neural networks (QNNs). It presents a general framework for supervised learning to represent labeled classical or quantum data. It consists of a sequence of unitary transformations on the input encoded quantum state, and the Pauli operator is measured on the final output qubit. Furthermore, they considered real-world data containing images of two distinct sets of handwritten digits. They showed QNNs learning ability to exactly determine the two data sets.

Initially, Stoudenmire and Schwab (2016) proposed a framework for a quantum-inspired tensor network for multiclass supervised learning. The MPS-based model is used to classify the images (MNIST) data set and had less than a 0.01 testing error. Han et al. (2018) introduced MPS-based generative modeling for unsupervised learning tasks. They considered bars and stripes random binary patterns and the MNIST handwritten digits to investigate their features, abilities, and shortcomings. They showed that MPS exhibits much stronger learning ability compared to the inverse Ising and Hopfield models. Liu, Zhang, Lewenstein, and Ran (2018) implemented entanglement-guided architectures to classify images, where quantum states are written in MPS. Liu et al. (2017) proposed two-dimensional training tree-like tensor networks as classifiers for image recognition problems and tested them on MNIST and CIFAR data sets. They showed that the proposed algorithm encodes classes of images into a tensor network state optimally and can be characterized by quantum entanglement. Glasser, Pancotti, August, Rodriguez, and Cirac (2018) introduced generalized tensor networks and discussed the relationship between restricted Boltzmann machines and string-bond states. They showed that generalized tensor networks can be associated exactly and can classify images accurately with smaller bond dimension.

Gardas, Rams, and Dziarmaga (2018) simulated many body quantum systems using a hybrid classical quantum algorithm, where the wave function of quantum Ising models is represented using a Boltzmann machine. The neural network is trained using a D-wave quantum annealer, and the ground-state energy is calculated. Huggins, Patel, Whaley, and Stoudenmire (2019) proposed tensor network–based quantum computing

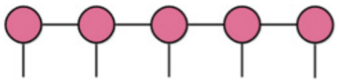


Figure 1: Representation of MPS with five sites.

approaches for generative and discriminative tasks designed to generate samples from a probability distribution and assign labels to images. The experiments are executed on quantum hardware using an optimization procedure for handwritten classes of images, and the training model's noise resilience is tested. Grant et al. (2018) introduced the concept of hierarchical quantum classifiers and executed binary classification for classical and quantum data. They considered two classical machine learning data sets, Iris and MNIST, and deployed the classifiers on a quantum computer. They showed impressive results and better results by considering different unitary parameters.

In this letter, we offer the following contributions:

- We demonstrate that an MPS as a one-dimensional array of tensors can be used to classify quantum mechanical data in addition to classical data sets.
- We encode classical data sets (Iris and Agri) into a quantum entangled state, which is given as an input to an MPS tensor network quantum circuit.
- Four and six qubit inputs are taken for the Iris and Agri data sets, and measurement is performed on a quantum circuit.
- To investigate performance, we deployed an MPS classifier on a real-time quantum device (ibmqx4).

3 Matrix Product State

The matrix product state encodes the extent of entanglement in bond dimensions. It is a method of a tensor network, where the tensors are connected in a one-dimensional geometry. Figure 1 shows the MPS as a one-dimensional array of tensors and a finite system of five sites. It provides an efficient approximation of realistic local Hamiltonians and can be produced sequentially by tensors. In fact, any pure quantum state can be described by substituting the coefficients—rank-N tensor by N-rank 3 tensors and 2-rank by 2 tensors. In MPS, a pure quantum state $|\phi\rangle$ is represented as

$$|\phi\rangle = \sum_{\sigma_1,\sigma_2,\ldots\sigma_L}^{d} Tr[M_1^{\sigma_1} M_2^{\sigma_2} \ldots M_L^{\sigma_L}]|\sigma_1, \sigma_2, \ldots \sigma_L\rangle, \tag{3.1}$$

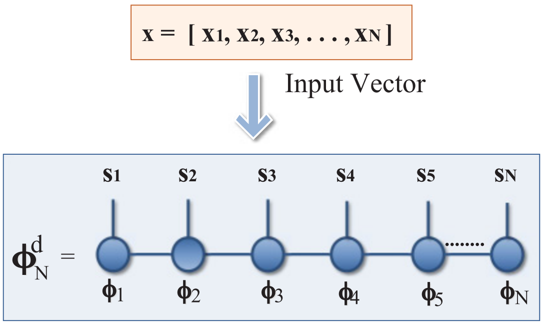


Figure 2: Mapping of input vector to order N tensor.

where $M_i^{\sigma_i}$ are complex square matrices, d is dimension, σ_i represents the indices ({0, 1} for qubits), and $Tr()$ denotes trace of matrix (Bhatia & Kumar, 2018a).

3.1 Encoding of Classical Data. In quantum mechanics, the N independent systems can be combine by performing tensor product operation on their respective state vectors (Stoudenmire & Schwab, 2016; Huggins, Patel, Whaley, & Stoudenmire, 2019). Consider a feature map

$$\phi^d(x) = \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \ldots \otimes \phi^{s_N}(x_N), \tag{3.2}$$

where s_j are indices run over the local dimension d such that $d = \{s_1, s_2, \ldots, s_N\}$. Therefore, each state vector x_j is mapped to a full feature map $\phi(x)$ in a d-dimensional space. Figure 2 shows the tensor diagram of full feature map $\phi(x)$.

Before illustrating the MPS tensor network, it is crucial to encode a classical machine learning data set into a quantum state. Consider a classical data set $S = \{(x^d, y^d)\}_{d=1}^{D}$ for binary classification, where $y^d \in \{0, 1\}$ are class labels for N-dimensional input vectors such that $x^d \in \mathbb{R}^N$. We have normalized the input vectors to lie in $[-\pi, \pi]$. Thus, the qubit ϕ is represented as

$$\phi_n^d = \cos(x_n^d)|0\rangle + \sin(x_n^d)|1\rangle, \tag{3.3}$$

$$\phi_n^d = \begin{bmatrix} \cos(x_1^d) \\ \sin(x_1^d) \end{bmatrix} \otimes \begin{bmatrix} \cos(x_2^d) \\ \sin(x_2^d) \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} \cos(x_N^d) \\ \sin(x_N^d) \end{bmatrix}. \tag{3.4}$$

We map the N-dimensional input vectors $x^d \in \mathbb{R}^N$ to a product state on N qubits by using the feature map, equation 3.2. The full quantum data are represented as tensor product $\phi^d = \otimes_{n=1}^{N} \phi_n^d$ (Huggins et al., 2019; Grant et al., 2018). Thus, the preparation of a quantum state is efficient, as it

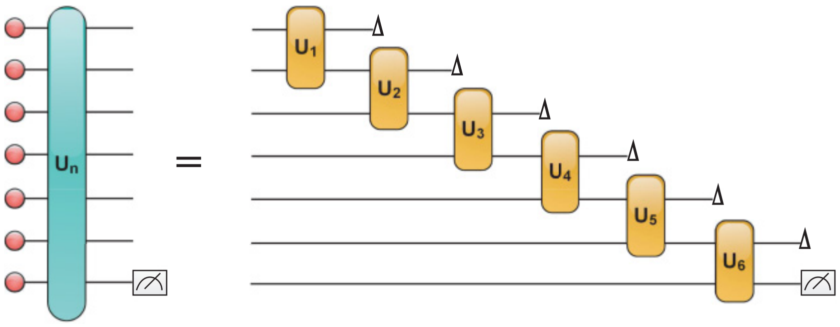


Figure 3: Matrix product state quantum classifier.

needs only single-qubit rotations to encode each segment of classical data set $n = \{1, 2, \ldots N\}$ in the amplitude of a qubit. Overall, there is no relevant cost for such encoding. Similar to a classical data set for binary classification, a quantum data set for binary classification is denoted as a set $S_q = \{(\phi^d, y^d)\}_{d=1}^{D}$, where $y^d \in \{0, 1\}$ are class labels for 2^N-dimensional input vectors such that $\phi^d \in \mathbb{C}^{2^N}$. It can be easily checked that the quantum data as outputs of a quantum circuit are in a superposition state.

3.2 Quantum Circuit Classifier. We now discuss MPS quantum circuit classifiers for classification of quantum data, which is made up of unitary operations. We followed an iterative approach by keeping positive trace values from the N-qubit input space to output qubits. We apply unitary operations composed of single qubit rotations around the y-axis and a CNOT gate to the input set and discard one of the qubits (unobserved) from each unitary operation. Therefore, we split the qubit into two parts for the next layer of the circuit. This process continues until the last qubit is left to be measured. Note that at each stage of the circuit, we keep one of the qubits resulting from one of the unitary operations of the earlier stage, and eventually unitary transformation occurs on two qubits from another subpart of the circuit.

The unitary blocks in Figure 3 consist of an input data set with an ancilla qubit that is initialized to zero. It can be easily traced out. Using the ancilla qubit, we can execute a large class of nonlinear operations. In Figure 3, circles denote inputs prepared in a product state, and triangles indicate unobserved qubits. When all unitary operations interpreted in the circuit have been executed, then one qubit is observed and used as the output qubit. The measurement on a particular qubit is carried out by applying Pauli operators in a particular direction. The output qubit determines the predicted value of the input, that is, the class label values assigned. In order to calculate the most likely state of the output qubit, the quantum circuit can be evaluated for a number of iterations considering the same input to determine the probability distribution among the computational basis states. The MPS quantum circuit for seven qubits is shown in Figure 3. It consists of inputs represented by circles, unitary blocks $\{U_i\}_{i=1}^{6}$, and a measurement operator on the last qubit. Here, single-qubit rotations in the y-direction are followed by a CNOT gate and discard a qubit for the next layer of the quantum circuit.

In order to assess the quality of actual and predicted values of the data set, we need to calculate the cost function to measure the difference between actual and predicted values of the data set. It is given as

$$J_\theta = \frac{1}{D}\sum_{d=1}^{D}(M_\theta(x^d) - y^d)^2, \tag{3.5}$$

where x^d and y^d are the input and class labels, respectively, M is a qubit operator, θ represents the set of parameters to use, and D is the total number of data points. The equation calculates the average amount that the model's predictions differ from the actual values. The goal is to minimize the cost function, which must be close to zero. Although there exist various procedures to carry out optimization, we have employed the conjugate gradient (CG) method optimize the large data sets. It is an iterative and effective method to optimize the results, but it can break down over multiple iterations when the function to be optimized is noisy. Alternatively, the stochastic gradient descent method, which is resilient to noise, can be applied.

Different parameters are used to measure the performance of the MPS quantum classifier, such as accuracy (ACC), sensitivity (Sens), specificity (Spec), and Gini coefficient. ACC is computed to measure the correctness of the classifier; Spec refers to the ability of the classifier to identify negative results; Sens defines the true positive rate (i.e., correctly identified by the classifier); and the Gini coefficient determines the inequality in the distribution, which should be between 0 and 1, where N is the total number of data points, TP is true positive, TN defines true negative, and FP and FN represent false positive and false negative, respectively:

- Accuracy:

$$ACC = \frac{TP + TN}{N} \times 100 \tag{3.6}$$

- Sensitivity:

$$Sens = \frac{TP}{TP + FN} \tag{3.7}$$

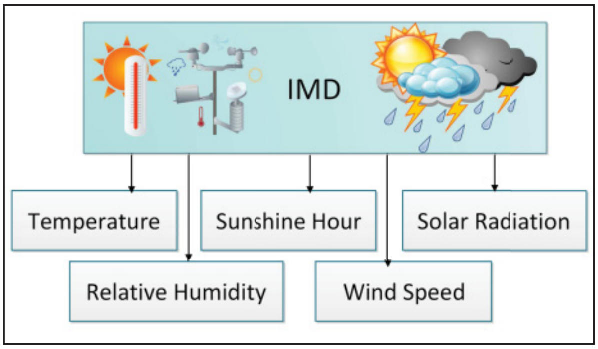


Figure 4: Parameters of the IMD weather data set.

- Specificity:

$$Spec = \frac{TN}{TN + FP} \tag{3.8}$$

4 Model Development

Here, we present the preprocessing, encoding, and management of quantum data that are implemented on a real-time quantum device ibmqx4, using open source software and the programming language Python 3.6.5 with installed Qiskit (i.e., an open-source software development kit (SDK) for working with the IBM Q quantum processors).

4.1 Data Collection and Preprocessing for the Study. First, we develop the MPS-based model for classification. At the start, we collected the Iris sample data sets from the UCI machine learning web portal. The data set consists of 150 examples in three varieties of Iris (setosa, versicolor, virginica). We rescaled the input vectors to lie in $[-\pi, \pi]$ and applied binary classification to class labels. After the normalization process, we formed pairwise combinations of samples (Iris_1, Iris_2, and Iris_3) on the basis of classes; for example, Iris_1 consists of data belonging to class labels 1 and 2 (now encoded as 0 and 1). Iris_2 and Iris_3 consist of data belonging to class labels 2, 3 and 1, 3, respectively. Figure 4 shows the parameters of the IMD weather data set.

Climatic data from the Patiala station have been retrieved from India is Meteorological Department (IMD) in Pune. The station is located at 30.33°E latitude and 76.38°S longitude. The study area includes the Patiala station, located in northern Punjab (India). The elevation is 351 m above sea level. The daily meteorological data for Patiala during 2014, 2015, and 2016 were used. The data consist of the following parameters: maximum and

Table 1: Statistical Parameters of Meteorological Variables and ET_o of Patiala Station.

Parameters	Max	Min	Mean	SD	SK	K
T_{min} (°C)	30.7	2.3	18.71	7.50	− 0.27	− 1.28
T_{max} (°C)	44.4	9.8	30.39	7.10	− 0.47	− 0.36
R_H(%)	100	0	73.30	17.64	− 0.77	− 0.02
u_2 (km h^{-1} day^{-1})	16	0	3.23	2.18	1.46	3.16
I_s (h)	12.2	0	6.24	3.53	− 0.52	− 0.98
R_s (MJ m^{-2} day^{-2})	28.2	4.9	16.15	6.14	− 0.01	− 0.98
ET_o (mm)	6	0	2.49	1.48	0.17	1.01

Table 2: Performance Comparison of MPS for Each Sample of the Agriculture and Iris Data Sets.

ET_o	Categories	Classes	Samples
0	(0–1)→ LOW	C_1	$Agri_1$ (C_1 (0) and C_2 (1))
1			
2	(2–3)→ MEDIUM	C_2	$Agri_2$ (C_2 (0) and C_3 (1))
3			
4	(4–6)→ HIGH	C_3	$Agri_3$ (C_1 (0) and C_3(1))
5			
6			

minimum air temperature (T_{max}) (T_{min}), relative humidity (R_H), wind speed (u_2), solar radiation (R_s), sunshine hours (I_s), evapotranspiration (ET_o), standard-deviation (SD), skewness (SK), and kurtosis (K). The statistical parameters of meteorological variables at Patiala are in Table 1.

In the agriculture data set, the ET_o varies from 0 to 6. In order to perform binary classification, we divided the ET_o into three categories: set {0, 1} comes under LOW, {2, 3} is MEDIUM, and {4, 5, 6} is categorized as HIGH, as shown in Table 2. From this data set, we generated three binary classification tasks: C_1, C_2, and C_3. We rescaled the input vectors element-wise to lie in $[-\pi, \pi]$ and applied binary classification to the class labels.

After the normalization process, we formed a pairwise combination of samples ($Agri_1$, $Agri_2$, and $Agri_3$) on the basis of classes; for example, $Agri_1$ consists of data belonging to class labels C_1 and C_2 (now encoded as 0 and 1). Similarly, $Agri_2$ and $Agri_3$ consist of data belonging to class labels C_2, C_3 and C_1, C_3, respectively. Furthermore, each sample is divided into training and testing sets. The training set consists of 80% of the original data sets, and the testing set consists of 20% of it. Also, the MPS quantum classifier is applied to the training data set. The MPS quantum classifier is applied to the data set for a number of iterations, considering the same input repeatedly.

After achieving best accuracy, the trained model is applied to the testing data set (unseen) and the results are analyzed.

4.2 Development Phases. After partitioning the full data set in the third phase, the training and testing sets were mapped into a tensor network state using equation 3.2. The input vectors are encoded into a quantum state for classifying the classical data on a quantum computer using equation 2.3. Finally, we take the tensor product of each input quantum state to form complete quantum data to use in a quantum circuit, using equation 3.4. Figure 5 shows the seven stages of our methodology for model development.

In the fifth stage, a qubit-efficient MPS quantum classifier is trained using unitary parameters and qubit rotations in the chosen direction. Finally, one or more qubits are measured using Pauli operators.

In the final stage, we determined the predicted value of the input, that is, the class label values assigned for the training set after assigning it to the quantum circuit. In order to calculate the most probable state of the output qubit, we repeated the fifth stage for a number of iterations considering the same input. Finally, we had classification results for each sample. The experimental results are plotted in the next section for each sample.

4.3 Experimental Results: Iris Data Set. In this section, we test the ability of the MPS quantum classifier to classify the Iris data set. For the experiment, we used qubit rotations in the y-direction to yield real values and parameterized the unitary operations with the ancilla qubit. It is represented as a four-qubit input gate consisting of an ancilla qubit. It can be traced out in order to execute nonlinear operations on the data set.

In order to analyze the performance of the MPS quantum classifier, we divided the data set into three samples (Iris$_1$, Iris$_2$, and Iris$_3$) on the basis of the pairwise combination of class labels. Each sample consists of two-thirds of the data set. We also split each sample into a training set and a testing test with an 80:20 ratio to compute accuracy. We classified the Iris data set, using an MPS quantum classifier on the basis of accuracy and computed cost during the training and testing periods. On executing the MPS quantum classifier on each sample for classes 1 and 2, 2 and 3, and 1 and 3 of Iris data set, we found an 85% accuracy rate while differentiating classes 1 and 2, an 80% accuracy rate for classes 2 and 3, and a 90% accuracy rate differentiating classes 1 and 3. Therefore, we verified that the results from the largest values and cost with the lowest values provide the basis of higher model efficiency.

We analyzed the output of the MPS classifier and compared it for different Iris samples on the basis of such parameters as binary classification accuracy, cost, Spec, Sens, and Gini coefficient, as shown in Table 3. The MPS quantum circuit gives results of 88.75% and 85% and a cost of 0.11 and 0.15 for Iris$_1$ training and testing sets, respectively. Iris$_2$ shows an accuracy of 83.75% and 80% and a cost of 0.16 and 0.2 for training and testing sets.

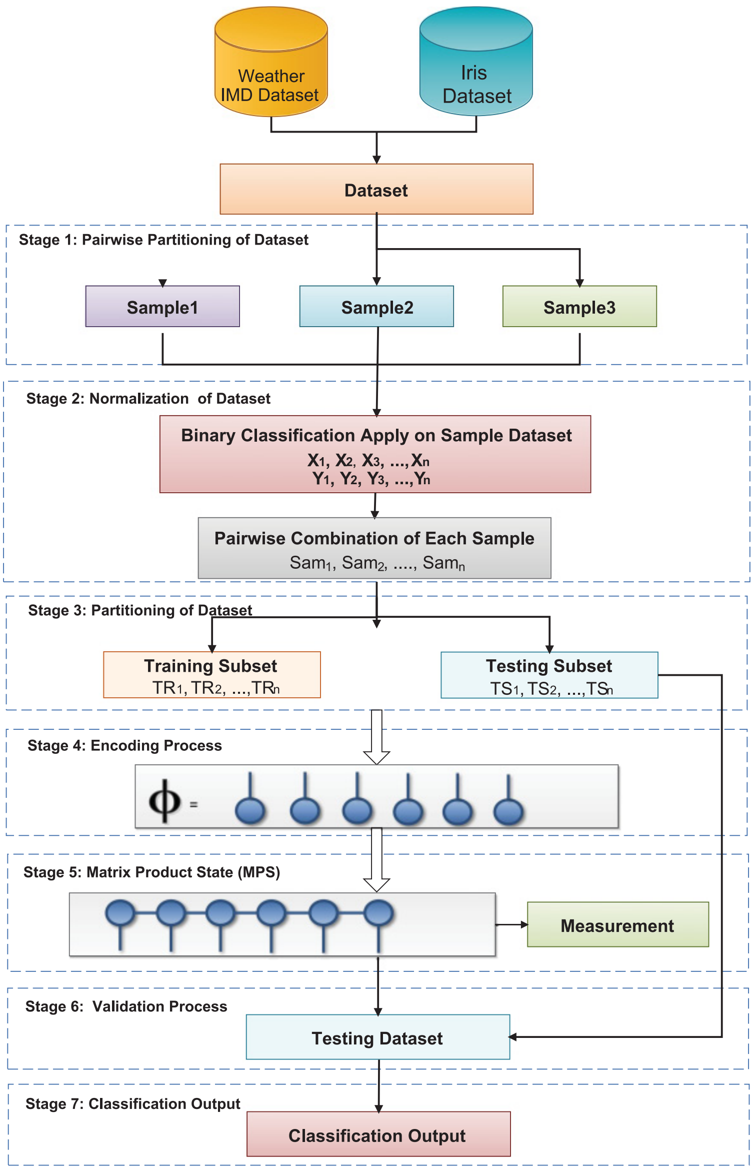

Weather IMD Dataset
Iris Dataset
Dataset
Stage 1: Pairwise Partitioning of Dataset
Sample1
Sample2
Sample3
Stage 2: Normalization of Dataset
Binary Classification Apply on Sample Dataset
X1, X2, X3, ...,Xn
Y1, Y2, Y3, ...,Yn
Pairwise Combination of Each Sample
Sam1, Sam2, ..., Samn
Stage 3: Partitioning of Dataset
Training Subset
TR1, TR2, ...,TRn
Testing Subset
TS1, TS2, ...,TSn
Stage 4: Encoding Process
ϕ =
Stage 5: Matrix Product State (MPS)
Measurement
Stage 6: Validation Process
Testing Dataset
Stage 7: Classification Output
Classification Output

Figure 5: Model development phases for classification.

Iris$_3$ results are slightly better than those of Iris$_1$, with an accuracy of 95% and 90% for training and testing sets. The MPS quantum classifier correctly classified the Iris data set and achieved a cost function value of 0.05 and 0.1

Table 3: Performance Comparison of MPS for Each Sample of the Iris Data Set.

	Training					Testing				
Sample	Cost	ACC	Spec	Sens	Gini	Cost	ACC	Spec	Sens	Gini
Iris_1	0.11	88.75	0.86	0.90	0.80	0.15	85	0.85	0.83	0.67
Iris_2	0.16	83.75	0.82	0.84	0.80	0.2	80	0.71	1.0	0.50
Iris_3	0.05	95	0.90	1.0	0.92	0.1	90	0.84	1.0	0.77

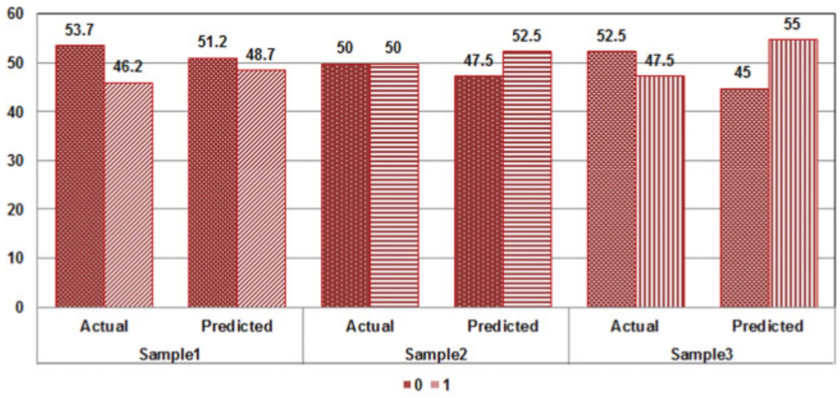
for the training and testing sets of the Iris_3 sample, respectively (see equation 3.5).

We performed the experiments with quantum circuits of $N = 4$ qubits. We have produced data sets consisting of 2500 quantum states for each of the classes $y \in \{1, 2, 3\}$. We gave each quantum state as input into the quantum computer where the MPS quantum classifier is implemented. We considered four-qubit input gates, including an ancilla qubit set to $|0\rangle$. The comparisons between the actual and predicted classification values in percentages for training and testing sets of Iris_1 (sample1), Iris_2 (sample2), and Iris_3 (sample3) are shown in Figure 6.

4.4 Experimental Results: Agri Data Set. In this section, we determine the performance of the proposed model for the agriculture domain using larger historical and streaming data sets. Each sample of the Agri data set consists of statistical parameters, given in Table 1, formed by a pairwise combination of classes. In order to test the ability of the MPS classifier, we trained it with the training set as an input and repeated the process considering the same input. Further, the testing set is given to a quantum classifier. The performance of the proposed model during the training and testing periods for each sample is given in Table 4.

Compared with each sample of the training data sets, the accuracy of the testing data set of Agri_1 is just slightly greater. It can be easily checked that training accuracy of the Agri_2 and Agri_3 samples is marginally higher than the testing samples, respectively. In the case of the training data set of the Agri_1 sample, the specificity is approximately 0.98, that is, the MPS classifier identifies more negative results compared to the testing set's 0.76. Therefore, the true positive value of the training set is less than that of the testing set for Agri_1. The estimated ET_o actual and predicted values (in %) for the training and testing sets of Agri_1 (Sample1), Agri_2 (Sample2) and Agri_3 (Sample3) are plotted in Figure 7.

The main advantage of the MPS quantum classifier is that training can be implemented with high efficiency. The mapping of classical data into MPS form (i.e., a highly dimensional Hilbert space) is beneficial for generating high-order correlations between classes. MPS's bond dimension manages the parameters of the machine learning model. It is easy to compute and



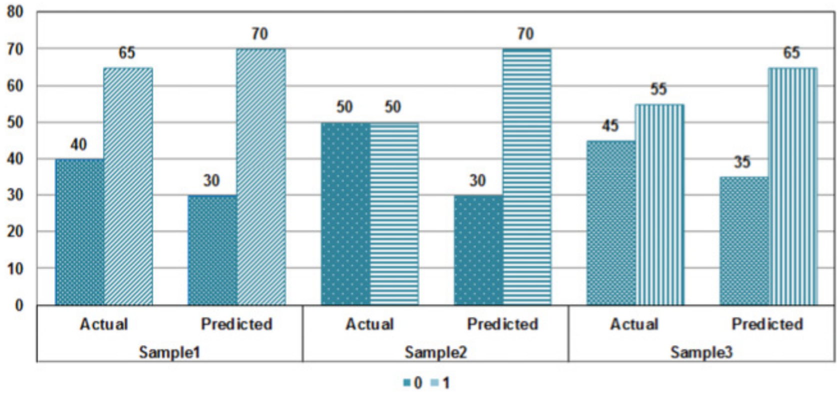


Figure 6: Estimated IRIS$_{species}$ actual and predicted values (%) for training (in red) and testing (in blue) data sets of Iris$_1$, Iris$_2$, and Iris$_3$.

Table 4: Performance Comparison of MPS for Each Sample of the Agri Data Set.

	Training					Testing				
Sample	Cost	ACC	Spec	Sens	Gini	Cost	ACC	Spec	Sens	Gini
Agri$_1$	0.20	79.03	0.98	0.72	0.52	0.19	80.65	0.76	0.83	0.53
Agri$_2$	0.24	75.34	0.68	0.82	0.51	0.26	73.33	0.67	0.79	0.50
Agri$_3$	0.21	78.73	0.73	0.89	0.61	0.22	77.04	0.77	0.75	0.53

can be selected adaptively. Usually the bond dimension exists between 10 and 10,000 or more quantum states. It follows that larger bond dimensions result in higher accuracy. In fact, on selecting an extremely large bond dimension, the model can also result in overfitting. MPS quantum classifiers

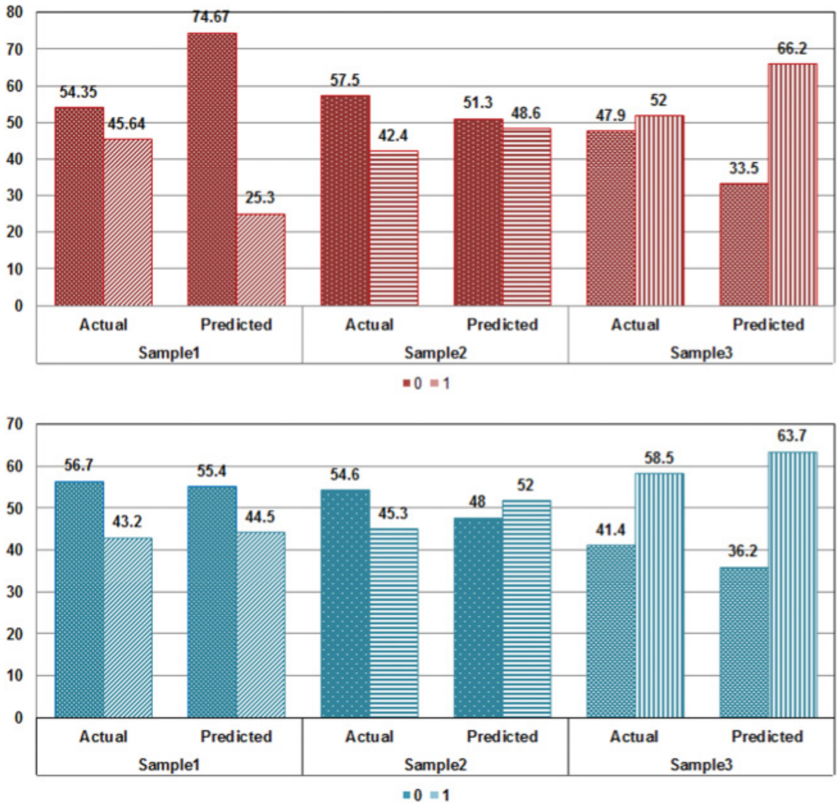


Figure 7: Estimated ET$_o$ Actual and predicted values (%) for the training (in red) and testing (in blue) data sets of Agri$_1$, Agri$_2$, and Agri$_3$.

have been used to avoid overfitting as well as underfitting, deal with corrected predictors, and reduce the variance of prediction error. It performed effectively and efficiently handled large data sets. We believe that it can be adapted to many other machine learning tasks to scrutinize their power. It showed great learning capability for Iris species and ET$_o$ estimations in the Agri data set. Figure 8 describes the results of training and testing set accuracy, cost, Spec, Sens, and Gini coefficient for each sample and shows consistent accuracy of the MPS quantum classifier.

The similarity between actual and predicted values is expressed on the basis of centered root-mean-square difference, correlation, and their variations in amplitude (i.e., standard deviation). We have used Taylor diagrams to graphically outline the degree of correspondence among values (see Figure 9). It has been used to investigate the performance of models to study aspects of climatic environment. The colors indicate the actual and predicted

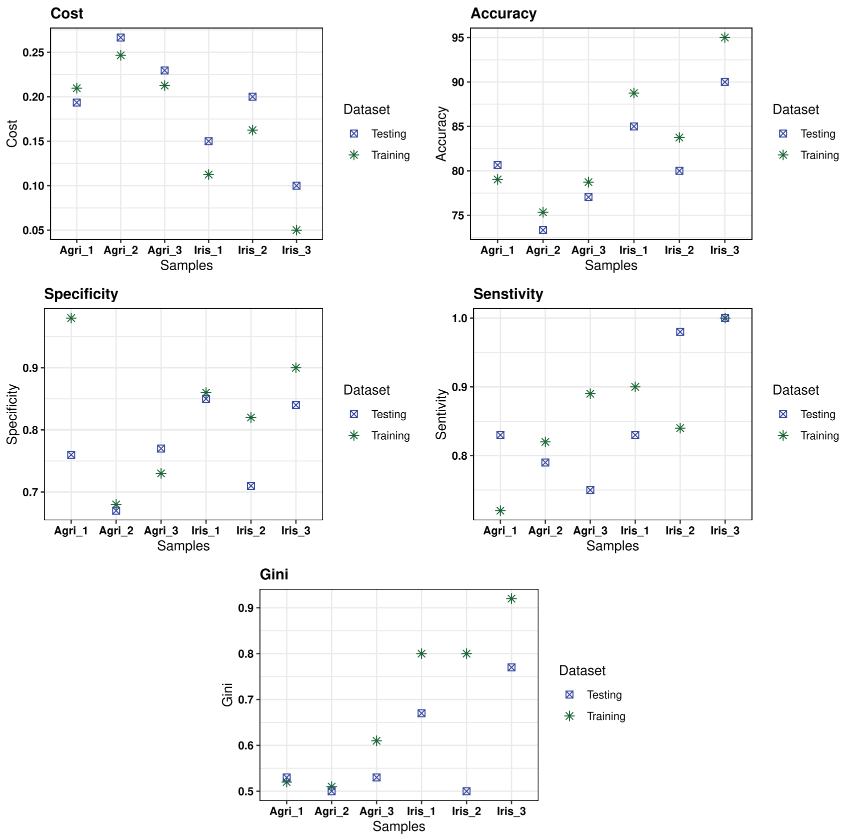


Figure 8: Comparison of forecasting ET$_o$ and Iris species results with the MPS model. (a) Cost. (b) ACC. (c) Spec. (d) Sens. (e) Gini.

values of different samples for the Iris and Agri data sets. Here, Iris$_1$(actual) and Iris$_1$(predicted) depict the actual and predicted values of the training and testing sets of the Iris data set, respectively.

5 Conclusion

In this letter, we have illustrated that a matrix product state quantum classifier can be used to classify quantum data efficiently. We have focused on an MPS quantum circuit augmented with ancilla qubit that is implemented on a quantum computer with the restrictions on of using only actual qubit rotations. The key advantage of classification with an MPS quantum circuit is that it can be executed efficiently with a small number of qubits. The MPS

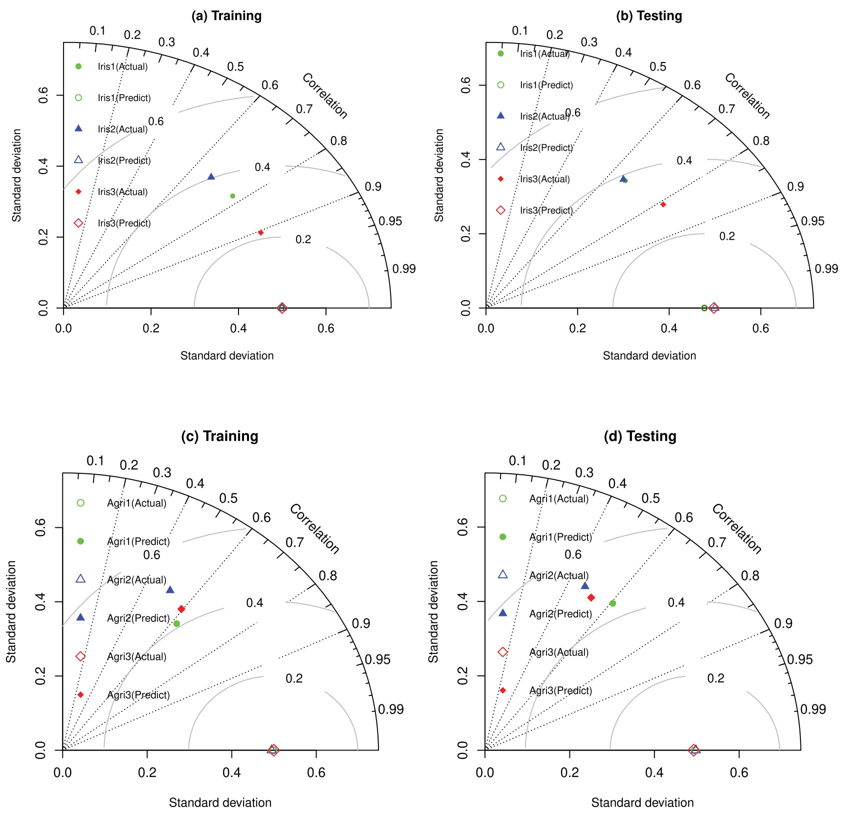


Figure 9: Representation of degree of correspondence between each sample of Iris and Agri data sets using Taylor diagrams. (a) Training$_{Iris}$. (b) Testing$_{Iris}$. (c) Training$_{Agri}$. (d) Testing$_{Agri}$.

quantum classifier has shown great learning capability for the Iris and Agri data sets. We analyzed the different classification performance metrics for each sample, and have shown the degree of correspondence among values. The deployment of an MPS quantum model to evaluate other machine learning tasks is promising. In the future, we will investigate the performance of other tensor networks for large, real-time quantum data.

Acknowledgments

A.S.B. was supported by Maulana Azad National Fellowship (MANF), funded by the Ministry of Minority Affairs, Government of India. M.K.S.

was supported by the Council of Scientific and Industrial Research, India, funded by its R&D organization.

References

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). *Crop evapotranspiration: Guidelines for computing crop water requirements* (FAO Irrigation and Drainage paper 56). Rome: FAO.

Bhatia, A. S., & Kumar, A. (2018a). Quantifying matrix product state. *Quantum Information Processing*, *17*(3), 41.

Bhatia, A. S., & Kumar, A. (2018b). Neurocomputing approach to matrix product state using quantum dynamics. *Quantum Information Processing*, *17*(10), 278.

Bhatia, A. S., & Saggi, M. K. (2018). *Simulation of matrix product state on a quantum computer*. arXiv:1811.09833.

Biamonte, J. (2018). *Quantum machine learning matrix product states*. arXiv:1804.02398.

Farhi, E., & Neven, H. (2018). *Classification with quantum neural networks on near term processors*. arXiv:1802.06002.

Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics*, *21*(6–7), 467–488.

Gao, X., & Duan, L.-M. (2017). Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, *8*(1), 662.

Gardas, B., Rams, M. M., & Dziarmaga, J. (2018). Quantum neural networks to simulate many-body quantum systems. *Physical Review B*, *98*(18), 184304.

Glasser, I., Pancotti, N., August, M., Rodriguez, I. D., & Cirac, J. I. (2018). Neural-network quantum states, string-bond states, and chiral topological states. *Physical Review X*, *8*(1), 011006.

Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., . . . Severini, S. (2018). Hierarchical quantum classifiers. *Quantum Information*, *4*(1), (65).

Han, Z.-Y., Wang, J., Fan, H., Wang, L., & Zhang, P. (2018). Unsupervised generative modeling using matrix product states. *Physical Review X*, *8*(3), 031012.

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, *103*(15), 150502.

Huggins, W., Patel, P., Whaley, K. B., & Stoudenmire, E. M. (2019). Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, *4*(2), 024011.

Kool, D., Agam, N., Lazarovitch, N., Heitman, J., Sauer, T., & Ben-Gal, A. (2014). A review of approaches for evapotranspiration partitioning. *Agricultural and Forest Meteorology*, *184*, 56–70.

Liu, D., Ran, S.-J., Wittek, P., Peng, C., García, R. B., Su, G., & Lewenstein, M. (2017). *Machine learning by two-dimensional hierarchical tensor networks: A quantum information theoretic perspective on deep architectures*. arXiv:1710.04833.

Liu, Y., Zhang, X., Lewenstein, M., & Ran, S.-J. (2018). *Entanglement-guided architectures of machine learning by quantum tensor network*. arXiv:1803.09111.

Martí, P., González-Altozano, P., López-Urrea, R., Mancha, L. A., & Shiri, J. (2015). Modeling reference evapotranspiration with calculated targets: Assessment and implications. *Agricultural Water Management*, *149*, 81–90.

Otterbach, J., Manenti, R., Alidoust, N., Bestwick, A., Block, M., Bloom, B., . . . Rigetti, C. (2017). *Unsupervised machine learning on a hybrid quantum computer*. arXiv:1712.05771.

Pestun, V., & Vlassopoulos, Y. (2017). *Tensor network language model*. arXiv:1710.10248.

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical Review Letters*, *113*(13), 130503.

Saggi, M. K., & Jain, S. (2018). A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing and Management*, *54*(5), 758–790.

Saggi, M. K., & Jain, S. (2019). Reference evapotranspiration estimation and modeling of the punjab northern india using deep learning. *Computers and Electronics in Agriculture*, *156*, 387–398.

Schuld, M., Bocharov, A., Svore, K., & Wiebe, N. (2018). *Circuit-centric quantum classifiers*. arXiv:1804.00633.

Stoudenmire, E. M., & Schwab, D. J. (2016). *Supervised learning with quantum-inspired tensor networks*. arXiv:1605.05775.

Received September 8, 2018; accepted March 9, 2019.