Clustering using matrix product states

Xiao Shi,[1,2] Yun Shang,[1,3,*] and Chu Guo[4,5,†]
[1]*Institute of Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[2]*School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China*
[3]*NCMIS, MDIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[4]*Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, Henan 450000, China*
[5]*Key Laboratory of Low-Dimensional Quantum Structures and Quantum Control of Ministry of Education, Department of Physics and Synergetic Innovation Center for Quantum Effects and Applications, Hunan Normal University, Changsha 410081, China*

(Received 20 November 2021; accepted 2 May 2022; published 18 May 2022)

The matrix product state has been demonstrated to be able to explore the most relevant portion of the exponentially large quantum Hilbert space and find accurate solutions for one-dimensional interacting quantum many-body systems. Inspired by this success, here we propose a clustering algorithm based on the matrix product state, which first maps the classical data into quantum states represented as matrix product states, and then minimizes the loss function using a variational matrix product states algorithm in the enlarged space. We demonstrate this algorithm by applying it to several commonly used machine learning data sets, showing that this algorithm could reach higher learning precision and that it is less likely to be trapped in local minima compared to the standard K-means algorithm. We also show that this algorithm can achieve state-of-the-art learning precision on popular computer vision data sets when used in combination with better initialization schemes.

DOI: 10.1103/PhysRevA.105.052424

I. INTRODUCTION

The past few years have witnessed a growing interest in the intersection between quantum physics and machine learning. On the one side, machine learning tools have been used to solve quantum problems, such as phase recognition [1–8], quantum state tomography [9–13], solving quantum many-body problems [14–24], and non-Markovian quantum dynamics [25]. On the other side, tools from quantum many-body physics, especially tensor network states algorithms, were used to solve classical machine learning problems. The tensor network states (TNS) algorithm has become one of the most important algorithms to study quantum many-body systems in the last two decades. In particular, the matrix product state (MPS), a one-dimensional variant of TNS, solves one-dimensional strongly correlated quantum many-body systems with unprecedented efficiency and accuracy and has thus become the algorithm of choice in this field [26–28]. In the last decade, MPS has also begun to find applications in classical fields, such as representing the high-dimensional probability distribution [29,30] where it is known as tensor train, and solving classical machine learning tasks including classification problems [31–33], generative modeling [34], and sequence-to-sequence modeling [35].

Despite the various applications of MPS to classical problems, it is, however, extremely difficult to find a practical instance where MPS could achieve a comparable performance to state-of-the-art algorithms already existing in those fields. The reverse is also true when applying machine learning or deep learning algorithms to solve quantum many-body physics problems. Clustering is an elementary machine learning task to separate unlabeled data into distinct and nonoverlapping clusters. It has important applications in bioinformatics [36], image processing [37], and social networks [38]. However, for data processing and analysis, traditional clustering algorithms still face great challenges. With the development of quantum computing, some quantum clustering algorithms were proposed [39–41]. In this work, we propose an MPS-based algorithm for the clustering task and demonstrate that it can achieve comparable or even higher learning accuracies than the best known existing algorithms on three popular image clustering data sets from computer vision.

A standard approach for clustering is the K-means clustering, which divides the data into k different classes by minimizing the variance of the data in each class and can be formulated as a minimization problem with the loss function [42]

$$\text{loss}(\{\vec{\mathcal{O}}_1, \vec{\mathcal{O}}_2, \dots, \vec{\mathcal{O}}_k\}) = \sum_{n=1}^{N} \min_{1\leqslant j\leqslant k} ||\vec{x}_n - \vec{\mathcal{O}}_j||^2, \quad (1)$$

with $\vec{\mathcal{O}}_j$ the jth centroid, $\vec{x}_n$ the nth input data, and $||\cdot||$ denotes the Euclidean norm. However, this computational task is NP-hard [43]. In 1982, Lloyd proposed a herustic algorithm which works by choosing k random centroids, and then iteratively dividing the data into k classes according to their distances with the centroids and recomputing the center within each cluster [44]. This algorithm is still widely used till now due to its simplicity and efficiency, which will be referred to as the K-means algorithm in the following. The K-means

*shangyun602@163.com
†guochu604b@gmail.com

algorithm has a well-known drawback that it is prone to be trapped at local minima, which would result in bad clustering with high probability, especially for high-dimensional data. Various efforts have been paid to improve the accuracy of the K-means algorithm, such as better ways of initialization [45–50], or smoother objective loss functions [51–54]. Recently, a deep neural network was also applied to clustering [55–58] (deep clustering), as a completely different approach from the standard K-means algorithm. At the time of writing, the authors of Ref. [58] reported the most accurate image clustering model termed as the selective pseudolabel clustering.

Compared to deep clustering, our MPS clustering algorithm is still along the lines of the standard K-means algorithm in that the elementary structure of the K-means algorithm is kept, while the centeroids are searched within a MPS manifold instead of the original plain vector space. The MPS manifold is fully characterized by a single interger, termed as the bond dimension D, in comparison with deep neural networks, which usually contain a large number of hyperparameters. In the following, we will first introduce our MPS clustering algorithm and then demonstrate its performance on both small-scale machine learning data sets and several popular computer vision data sets.

II. MPS CLUSTERING ALGORITHM

The reason why the standard K-means algorithm results in poor accuracy for high-dimensional data is two-fold: (1) the existence of a huge number of local minima, such that with a naive initialization strategy one would easily fall into one of them and (2) the centeroids themselves are not expressive enough, for example, different centeroids are not orthogonal enough to each other. While a vast amount of the literature focused on improving the first issue by generating more reasonable inputs for the K-means algorithm (either more knowledged initial centroids or better behaved input data after certain transformations), using, for example, spectral clustering or subspace clustering [45,50], our MPS clustering directly aims to improve the second one, namely, increasing the expressivity of the centroids by representing them as MPSs. By MPS representation, the vector elements of centroids may have an entanglement relation and then they will have a stronger expression than the K-means case. Furthermore, we can see that the larger the inner product of the center points, the less obvious the difference between the center points. From this we can find the MPS algorithm can have a better performance than the K-means algorithm. As a result, our algorithm can be easily used as a substitution for the K-means algorithm and used in combination with better initialization strategies.

The first step of our algorithm is to map each input vector into an MPS, which is mathematically represented as a one-dimensional chain of rank-3 tensors. This can be done as follows. For the nth input vector $\vec{x}_n$ with L elements $(x_{n,1}, x_{n,2}, \dots, x_{n,L})$, one can map the lth element $x_{n,l}$ into a rank-3 tensor $X_{n,l}$ of size $1\times2\times1$ as [35,59]

$$(X_{n,l})^0_{0,0} = \cos\left(\frac{\pi}{2}x_{n,l}\right), \quad (X_{n,l})^1_{0,0} = \sin\left(\frac{\pi}{2}x_{n,l}\right). \tag{2}$$

As a result, $\vec{x}_n$ is mapped into an MPS denoted as $X_n^{\vec{\sigma}}$

$$X_n^{\vec{\sigma}} = \sum_{a_1,\dots,a_{L+1}} X^{\sigma_1}_{n,a_1,a_2} X^{\sigma_2}_{n,a_2,a_3} \dots X^{\sigma_L}_{n,a_L,a_{L+1}}, \tag{3}$$

where σ_l denotes the *physical indices* and a_l denotes the *auxiliary indices* of MPS, and we use $X^{\sigma_l}_{n,a_l,a_{l+1}} = (X_{n,l})^{\sigma_l}_{a_l,a_{l+1}}$ for short. The bond dimension D is defined as the largest size of the auxiliary indices, namely,

$$D = \max_{2\leqslant l\leqslant L} \dim(a_l). \tag{4}$$

We note that for each $X_n^{\vec{\sigma}}$ we have $\dim(\sigma_l) = 2$ and $D = 1$. The centroids $\mathcal{O}_j^{\vec{\sigma}}$, in comparison, are initialized to be MPSs with a fixed bond dimension $D \geqslant 1$, and can be written as

$$\mathcal{O}_j^{\vec{\sigma}} = \sum_{b_1,\dots,b_{L+1}} M^{\sigma_1}_{j,b_1,b_2} M^{\sigma_2}_{j,b_2,b_3} \dots M^{\sigma_L}_{j,b_L,b_{L+1}}, \tag{5}$$

with $1 \leqslant j \leqslant k$. Here the larger D, the more entanglement in the centeroid and then it has better expression. With the data structures defined for the inputs and the centeroids, our MPS clustering algorithm works as follows. First we randomly initialize k centeroids $\mathcal{O}_j^{\vec{\sigma}}$ in *right-canonical* forms with a fixed bond dimension D. Then we iterate the following two steps: (1) dividing all the data into k disadjoint classes according to their distances with the centroids, defined as

$$d^M_{n,j} = \left(X_n^{\vec{\sigma},\dagger} - \mathcal{O}_j^{\vec{\sigma},\dagger}\right)\left(X_n^{\vec{\sigma}} - \mathcal{O}_j^{\vec{\sigma}}\right), \tag{6}$$

that is, an input data $X_n^{\vec{\sigma}}$ is categorized into the mth class with $m = \operatorname{argmin}_{j=1}^k d^M_{n,j}$; (2) recomputing the centeriod for each class independently by minimizing the loss function

$$\text{loss}^M\left(\mathcal{O}_j^{\vec{\sigma}}\right) = -\frac{1}{N_j}\sum_{n_j=1}^{N_j} \ln\left|\mathcal{O}_j^{\vec{\sigma},\dagger} X_{n_j}^{\vec{\sigma}}\right|^2, \tag{7}$$

under the normalization condition $\mathcal{O}_j^{\vec{\sigma},\dagger}\mathcal{O}_j^{\vec{\sigma}} = 1$. Here N_j is the size of the jth class, $X_{n_j}^{\vec{\sigma},\dagger}$ is the n_jth data inside the jth class and $|\cdot|$ means the absolute value. We note that Eq. (7) is the same as the loss function used in Ref. [34], except that it is used as a loss function for each centroid in our case. The loss function in Eq. (7) is minimized using a variational MPS algorithm, where one performs a gradient descent algorithm for each tensor $M^{\sigma_l}_{j,b_l,b_{l+1}}$ locally instead of for all these tensors at the same time, that is,

$$M^{\sigma_l}_{j,b_l,b_{l+1}} \leftarrow M^{\sigma_l}_{j,b_l,b_{l+1}} - \eta\frac{\partial\text{loss}^M\left(\mathcal{O}_j^{\vec{\sigma}}\right)}{\partial M^{\sigma_l}_{j,b_l,b_{l+1}}}, \tag{8}$$

and η the learning rate (details of the variational MPS algorithm can be found in the Appendix A). In the prediction stage, given a new input data, we compute the distances between it and all the centeriods using Eq. (6) and label it with the one for which the distance is minimal.

The K-means algorithm performs well on linearly separable data sets, but not on nonlinear data sets. However, MPS-based algorithms have significantly improved performance on nonlinear data because it maps the data to a larger Hilbert space through MPS expressions, and in this larger Hilbert space, the discrimination between the data will become stronger. As can be seen from the results in Table I, our algorithm improves the accuracy by about 4 to 18 percentage points compared to the K-means algorithm on these three nonlinear data sets. This also embodies that the bond dimension determines the upper bound of entanglement entropy that MPS can capture [60] and the accuracy improved.

III. APPLICATIONS

A. Benchmarking on small clustering data sets

The higher expressivity of MPS clustering is best demonstrated on relatively small data sets where the algorithm suffers less from being trapped in local minima. Here we benchmark the learning accuracy of MPS clustering against K-means algorithm without particular optimization of the initialzation stage, namely, both using random initialization. The numerical benchmarking results on three small-scale clustering data sets, namely Ionosphere, Yeast, and Lymphography, which are show in Table I. Since the K-means algorithm is based on the distance measurement, if the difference between variables of different dimensions is too large, it may cause a small number of variables to exert an excessively high influence on the whole cost, thus eliminating the effect of the rest variables. To avoid this effect, we divide each dimension of the data by its largest value in the data set, namely $x_{n,l} \leftarrow x_{n,l}/x_l^{\max}$ with $x_l^{\max} = \max_{1\leqslant n\leqslant N} x_{n,l}$, thus ensuring $0 \leqslant x_{n,l} \leqslant 1$ for all $1 \leqslant n \leqslant N$ and $1 \leqslant l \leqslant L$. The learning accuracy (ACC) is defined as the number of correctly predicted labels over the whole number of data, namely N (we use the whole data set to train and to evaluate the learning accuracy as is commonly done in the literature). Each numerical experiment is repeated 100 times with random initialization, and then the final learning accuracies are taken as the average over the ten best instances. For the K-means algorithm we use the function sklearn.cluster.KMeans for the package sklearn. For MPS clustering we do 30 iterations and inside each iteration we do three sweeps. As we can see from the last three columns of Table I, MPS clustering generally outperforms the K-means algorithm for every data set in terms of ACC. Moreover, ACC goes higher when increasing the bond dimension D of MPS, which is expected since the parameter space grows as D^2, and MPS is more expressive with larger D as long as there is enough data to avoid overfitting. (More details can be found in the Appendix B.)

TABLE I. Comparison between K-means algorithm and our MPS clustering algorithm. The first column lists the names of the data sets. "N" is the total number of data in each data set. "Dim" denotes the number of features for each data. "Classes" denotes the number of clusters. The last three columns show the learning accuracies (in percentages) for the K-means algorithm and MPS clustering algorithm with $D = 8$ and $D = 16$, respectively. The learning accuracy is evaluated as the average of the best ten instances from 100 trials with random initialization.

Name	N	Dim	Classes	K-means	MPS-8	MPS-16
Ionosphere	351	34	2	71.23	86.81	87.26
Yeast	1484	8	10	39.87	43.23	43.75
Lymphography	148	18	4	55.81	72.77	73.04

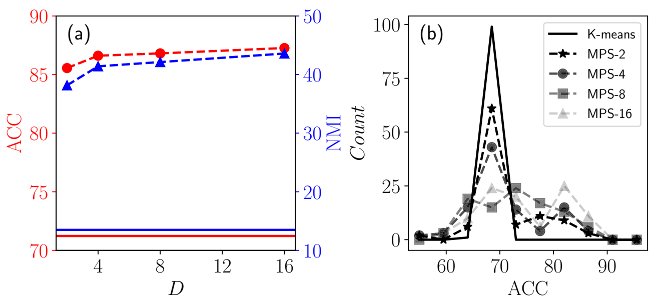


FIG. 1. (a) Learning accuracy (blue dashed line with triangle) and NMI (red dashed line with circle) as a function of the bond dimension D for our MPS clustering algorithm. The blue solid and red solid lines are the corresponding values of the K-means algorithm. (b) Distribution of learning accuracy for 100 trials for the K-means algorithm (black solid line) and our MPS clustering algorithms with different bond dimensions (dashed lines).

In the next we perform a more detailed analysis of the performance of our MPS clustering algorithm as a function of the only hyperparameter in the algorithm, namely, the bond dimension D. In particular, we take the data set Ionosphere as an example and consider two performance measures, ACC and normalized mutual information (NMI), which is defined as [61]

$$\mathrm{NMI}(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}. \tag{9}$$

Here $\Omega = \{\omega_1, \omega_2, \ldots, \omega_k\}$ denotes all the predicted clusters and $C = \{c_1, c_2, \ldots, c_j\}$ denotes the ground truth clusters (ω_l and c_l are sets with the same labels). I and H are the mutual information and Shannon entropy, respectively.

In Fig. 1(a) we plot ACC and NMI as functions of D, with their corresponding values from the K-means algorithm as references. Again the results are averaged over the ten best instances out of a total of 100 trials. We can see that the values for both measures increase with D until becoming more or less flat. In Fig. 1(b), we show the likelihood of MPS clustering to be trapped in bad minima by ploting the distribution of ACC for the 100 trials. We can see that the ACCs for the K-means algorithm are mostly centered around the value 71%, while for MPS clustering it is possible to reach a much higher ACC (close to 90%) and it is more likely to result in a higher ACC with a larger D. Larger D means more entanglement among vector elements, which means better expressiveness to the data points. So we can see from Fig. 1 that as D increases, the clustering accuracy are getting higher and the number of times that the algorithm gets stuck in the local minima are getting less.

B. Benchmarking on computer vision data sets

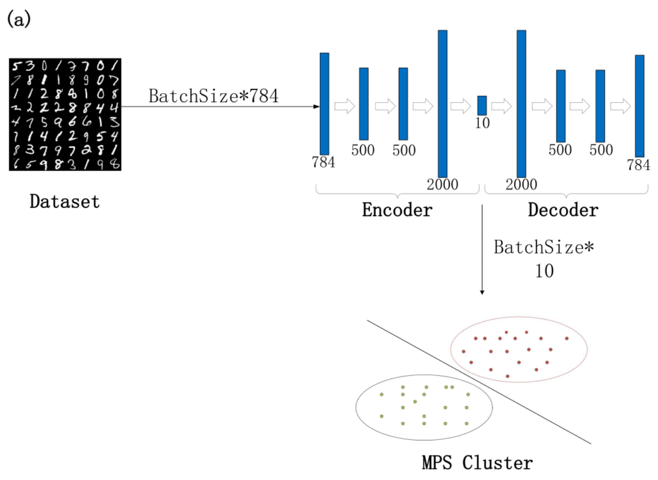
Real applications in general involve data with high-dimensional feature spaces. The straightforward application of the K-means algorithm could easily be trapped in local minima and result in bad clustering. Various spectrum clustering and subspace clustering techniques were proposed which aimed to provide better starting points for the K-means algorithm. In practice, we find that a direct application of

TABLE II. Learning accuracy (ACC) and NMI of our method compared to other top-performing image clustering models. The best results are stressed in bold and the second best results are stressed in italics.

Method	MNIST		USPS		FashionMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI
K-means	53.91	49.04	65.76	60.98	52.22	51.1
AE-K-means	77.08	77.13	68.29	66.14	58.91	62.15
DEC [62]	84.3	83.4	76.2	76.7	51.8	54.6
IDEC [63]	88.06	86.72	76.05	78.46	52.9	55.7
DEPICT [64]	96.5	91.7	96.4	92.7	39.2	39.2
EnSC [65]	96.3	91.5	61.0	68.4	62.9	63.6
DynAE [66]	98.7	96.4	98.1	94.8	59.1	64.2
InfoGAN [67]	87.0	84.0	–	–	61.0	59.0
ClusterGAN [57]	95.0	89.0	–	–	63.0	64.0
DualAE [68]	97.8	94.1	86.9	85.7	**66.2**	64.5
ADSSC [50]	99.0	*97.1*	96.94	*96.52*	60.29	70.3
SPC [58]	*99.03*	97.04	*98.4*	95.42	65.58	*72.09*
MPS-8	62.28	58.10	66.86	69.78	55.08	53.67
AE-MPS-8	88.92	80.94	70.8	69.1	62.12	62.29
ADSSC-MPS-8	**99.04**	**97.22**	**98.82**	**96.62**	*65.61*	**72.15**

MPS clustering with random initialization would suffer from the same problem of bad clustering for high-dimensional data. Nevertheless, the MPS clustering algorithm can simply be used as a drop-in replacement of the K-means algorithm and thus can be used in combination with those better initialization schemes to solve real-world problems. Particularly, we introduce two kinds of improved MPS clustering algorithms, one with an autoencoder and the other with sequential-approximation doubly stochastic subspace cClustering (ADSSC) [50], which we entitle as AE-MPS and ADSSC-MPS, respectively. In AE-MPS an autoencoder is first used to compress the dimensionality of the input data, which is then fed into MPS culstering. With ADSSC-MPS one first learns a doubly stochastic affinity matrix from the input data and then perform spectrum clustering technique to find the clusters, which is further fed into an autoencoder. (The usage of autoencoder in this case is only because the spectrum clustering produces negative numbers which do not comply with our encoding in Eq. (2), and it may be removed with a modified encoding scheme. For the autoencoder we choose the activation function to be the sigmoid function, such that the output only consists of numbers between 0 and 1.) The output of the autoencoder is finally fed into MPS clustering. These two algorithms are also shown in Fig. 2.

We benchmark our algorithm on three computer vision data sets, namely MNIST, USPS, and FashionMNIST. MNIST and USPS are handwritten digits' data sets, and FashionMNIST contains clothing items. Both MNIST and FashionMNIST have 70 000 images, and USPS has 9298 images in total. In TABLE. II we show the results of our numerical experiments on those three data sets. We can see that our AE-MPS and ADSSC-MPS outperforms AE-K-means and bare ADSSC in terms of ACC and NMI in most situations. Moreover, for the MNIST and USPS data set, our ADSSC-MPS score is the best out of all the candidates. For the USPS data set, our



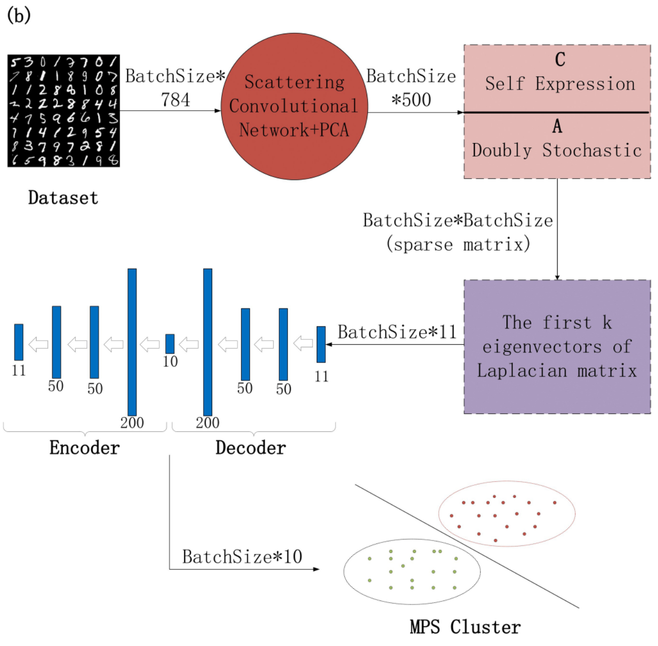


FIG. 2. Structures for (a) our AE-MPS clustering algorithm and (b) ADSSC-MPS clustering algorithm. The fully connected autoencoder is used here, and the number of neurons in each layer is written below each layer.

ADSSC-MPS scores best in terms of the NMI and second in terms of learning accuracy. For all three cases considered here, we choose an output dimension of 10 for the autoencoder used in AE-MPS, while for ADSSC-MPS we choose 11 features for the spectrum clustering similar to that found in Ref. [50], and then an output dimension of 10 for the subsequent autoencoder (detailed architectures can be found in Fig. 2). In the classic K-means algorithm, randomly initializing the center point can easily cause the algorithm to fall into a local minimum. Generally, the K-means algorithm with different centroid seeds will be run many times to solve the problem. The final result is the one with the smallest Euclidean distance from each data point to its center point. Here we replace the Euclidean distance by fidelity measure since the MPSs are normalized for each data and the center; we can filter the results similarly. In detail, for both the ACC and NMI of AE-MPS and ADSSC-MPS, we run 100 trials and use total

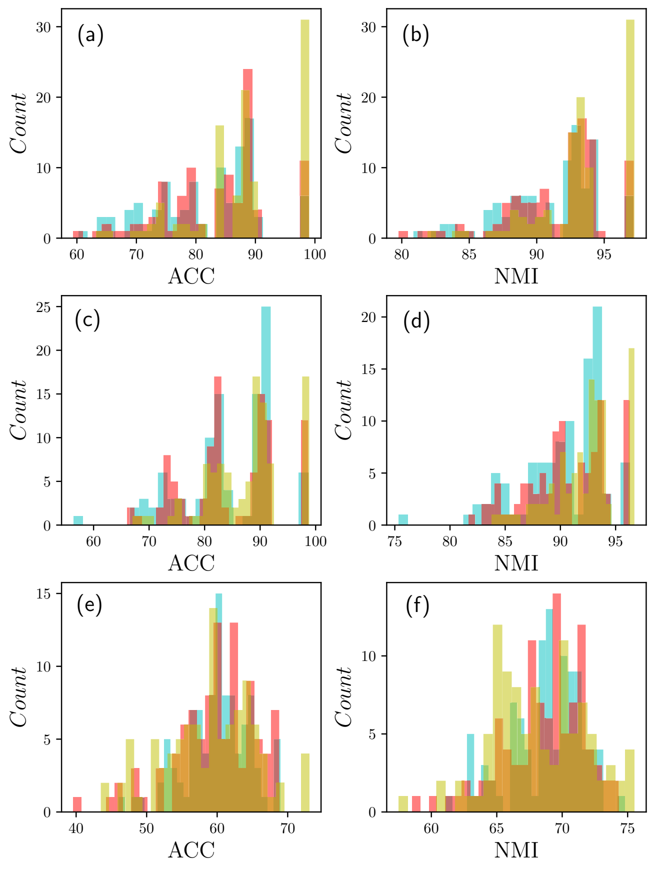


FIG. 3. (a), (b) Histograms of learning accuracies and NMIs for MNIST. (c), (d) Histograms of learning accuracies and NMIs for USPS. (e), (f) Histograms of learning accuracies and NMIs for FashionMNIST. The cyan, red, and yellow bars stand for $D = 1, 4, 16$, respectively.

fidelity to filter out the results, which is defined as

$$f_{\text{total}} = \sum_j n_j - \sum_{j,n_j} \left| \mathcal{O}_j^{\vec{\sigma},\dagger} X_{n_j}^{\vec{\sigma}} \right|. \tag{10}$$

We select 10 of the 100 results with the smallest f_{total} and calculate the average to get the final result.

From Table II, we can see MPS algorithms have better clustering effects compared with the existing algorithms since MPS adds the coherent relation between vector elements and has a better expression. For the MNIST data set, it can be seen from Table II that, although the bond dimension of MPS is only 8, its accuracy reaches 62.28, which is almost a 10 percent increase compared with that of the K-means algorithm (53.91). Since different data sets have different image size and sharpness of image, they do not have significant improvement on the other two data sets. However, by some preprocessing tricks, the MPS-based methods are all more accurate than the previous methods since preprocessing can reduce the dimensionality of the data set and shorten the running time of the program. After processing the data set with AE, the MPS-based algorithm shows more than a 10 percent increase compared with that of the AE-K-means on the MNIST data set. In particular, the ADSSC-MPS-8 method outperforms all exisiting algorithms on the MNIST and USPS data sets. Although our accuracy on the Fashion data set is a bit lower than the existing state-of-the-art algorithm. However, compared with the ADSSC algorithm, the accuracy of ADSSC-MPS is more than a 5 percent increase compared with that of ADSSC algorithm.

We also consider the effect of different bond dimensions by taking the ADSSC-MPS algorithm as an example. The results are shown in Fig. 3. Interestingly, we can see that the effect of a larger D is more of an increasing of the probability to find good clustering than of an increasing of the absolute values of ACC or NMI. More concretely, taking the MNIST data set, for example, the largest values for $D = 1, 4, 16$ are more or less the same, which for $D = 16$ there is a much higher probability (more than 30%) to get the largest ACC and NMI, while for $D = 1$ this probability is only around 10%.

IV. CONCLUSION

In summary, we propose a matrix-product-states-based clustering algorithm, where the input data are mapped to separable states in an exponentially large Hilbert space and then MPS is used as the ansatz for each centroid. The benchmark on small-scale clustering data sets with the K-means algorithm shows that our MPS clustering algorithm can reach a higher learning precision and that it is less likely to be trapped in bad minima. We also demonstrate on three large-scale image data sets that MPS clustering can be taken as a drop-in replacement of the K-means algorithm to be used in combination with more clever initialization schemes and produce state-of-the-art learning precisions.

To this end, we point out that MPS is a special type of tensor network work which allows most common arithmetic operations and other types of tensor networks, such as tree tensor networks or multiscale entanglement renormalizationan ansatz (MERA), may be explored to study clustering problems [32]. This algorithm could also be straightforwardly extended to a quantum algorithm with the MPS ansatz replaced by parameteric quantum circuits, for example. Perhaps, more interestingly, a better initialization scheme such as spectrum clustering which directly incorporates MPS formalism may also be explored in future works.

ACKNOWLEDGMENTS

Y. Shang thanks the support of National Natural Science Foundation of China (Grant No. 61872352) and the Program for Creative Research Group of the National Natural Science Foundation of China (Grant No. 61621003). C. Guo acknowledges support from the National Natural Science Foundation of China under Grants No. 11805279, No. 61833010, No. 12074117, and No. 12061131011.

APPENDIX A: VARIATIONAL MATRIX PRODUCT STATES FOR EACH CENTROID

To find the optimal MPS representation for each centroid, we use a variational MPS algorithm similar to that of Ref. [34]. The main idea is to optimize the tensors on each site one by one instead of optimizing them at the same time. The major steps of the algorithms are summarized as follows.

First the MPS corresponding to each centroid is initialized in right-canonical form, which can be done by randomly initializing all the tensors of each centroid and then preparing the resulting MPS into right-canonical form, namely, each tensor $M^{\sigma_l}_{j,b_l,b_{l+1}}$ satisfies [28,69]

$$\sum_{\sigma_l,b_{l+1}} M^{\sigma_l}_{j,b_l,b_{l+1}} M^{\sigma_l}_{j,b'_l,b_{l+1}} = \delta_{b_l,b'_l}, \tag{A1}$$

where j is the index of the centroid and δ_{b_l,b'_l} is the Kronecker delta function. This will also ensure that each centroid has Euclidean norm 1. Then, similar to the ground-state searching algorithm, one optimizes the tensors of the MPS one by one, that is, optimizing the tensors from the 1th site to the Lth site, and then from the Lth site to the 1th site (L is the size of each input data). A full iteration like this is referred to as a sweep. The difference compared to ground-state searching is that, for each local optimization, we perform a gradient descent algorithm instead of eigenvalue decomposition. For example, to optimize the tensor $M^{\sigma_l}_{j,b_l,b_{l+1}}$ during the left to right iteration on site l, one first computes the gradient of the loss function against this tensor as

$$\frac{\partial \mathrm{loss}^M\left(\mathcal{O}^{\vec{\sigma}}_j\right)}{\partial M^{\sigma_l}_{j,b_l,b_{l+1}}} = -\frac{2}{N}\sum_{n_j=1}^{N_j} \frac{L_{j,n_j,b_l} X^{\sigma_l}_{n_j} R_{j,n_j,b_{l+1}}}{|\mathcal{O}^{\vec{\sigma},\dagger}_j X^{\vec{\sigma}}_{n_j}|}, \tag{A2}$$

with

$$L_{j,n_j,b_l} = \sum_{b_k,\sigma_k}\prod_{k<l} M^{\sigma_k}_{j,b_k,b_{k+1}} X^{\sigma_k}_{n_j}, \tag{A3}$$

$$R_{j,n_j,b_{l+1}} = \sum_{b_k,\sigma_k}\prod_{k>l} M^{\sigma_k}_{j,b_k,b_{k+1}} X^{\sigma_k}_{n_j}, \tag{A4}$$

with $L_{j,n_j,b_1} = R_{j,n_j,b_{L+1}} = 1$. Here N_j denotes the number of data in the jth cluster. After that, the tensor $M^{\sigma_l}_{j,b_l,b_{l+1}}$ is updated as

$$M^{\sigma_l}_{j,b_l,b_{l+1}} \leftarrow M^{\sigma_l}_{j,b_l,b_{l+1}} - \eta \frac{\partial \mathrm{loss}^M\left(\mathcal{O}^{\vec{\sigma}}_j\right)}{\partial M^{\sigma_l}_{j,b_l,b_{l+1}}}, \tag{A5}$$

with η the learning rate. Then one prepares the resulting tensor into left-canonical form using SVD, namely,

$$\mathrm{SVD}(M_{j,(\sigma_l,b_l),b_{l+1}}) = \sum_{b'_{l+1}} U^{\sigma_l}_{j,b_l,b'_{l+1}} S_{j,b'_{l+1},b'_{l+1}} V_{j,b'_{l+1},b_{l+1}}, \tag{A6}$$

where we assumed grouping the two tensor indices (σ_l, b_l) into a single index during the SVD decomposition (we do not distinguish between upper and lower tensor indices). After that $U^{\sigma_l}_{j,b_l,b'_{l+1}}$ is used to substitute $M^{\sigma_l}_{j,b_l,b_{l+1}}$ while $S_{j,b'_{l+1},b'_{l+1}}$ and $V_{j,b'_{l+1},b_{l+1}}$ are absorbed into the next tensor $M^{\sigma_{l+1}}_{j,b_{l+1},b_{l+2}}$. After that one moves to the $l+1$th site. During the right to left iteration at site l, one group the two tensor indices (σ_l, b_{l+1}) and perform the SVD

$$\mathrm{SVD}(M_{j,b_l,(\sigma_l,b_{l+1})}) = \sum_{b'_l} U_{j,b_l,b'_l} S_{j,b'_l,b'_l} V^{\sigma_l}_{j,b'_l,b_{l+1}}, \tag{A7}$$

then one uses $V^{\sigma_l}_{j,b'_l,b_{l+1}}$ to substitute $M^{\sigma_l}_{j,b_l,b_{l+1}}$ while U_{j,b_l,b'_l} and S_{j,b'_l,b'_l} are absorbed into with the tensor on the $l-1$th site. After that one moves to the $l-1$th site. In this way the resulting MPS will automatically be right-canonical after a full sweep.

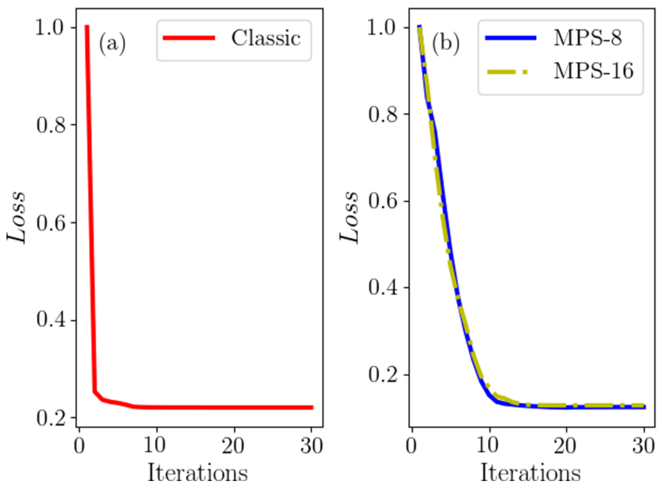


FIG. 4. (a) The loss value as a function of the number of K-means iterations for the K-means algorithm. (b) The loss value as a function of the number of sweeps for MPS clustering with $D = 8$ (blue line) and $D = 16$ (yellow line), respectively. For MPS clustering we have used three sweeps in each K-means like the iteration. Here the Yeast data set is used. The loss values in both cases are divided by their initial values.

APPENDIX B: DETAILS OF THE NUMERICAL EXPERIMENTS

We first show the convergence of K-means and MPS clustering as the number of iterations in Fig. 4, where we used the Yeast data set. The loss values are defined as the sum of the losses of all the different clusters. For the K-means algorithm, we plot the loss values as a function of the K-means iteration. While for MPS clustering, we plot the loss values as a function of K-means-like iteration. The loss values are all divided by their initial values. We can see that for the K-means algorithm the loss values quickly converge to a minimum. In comparison, MPS clustering the converges slower, but to smaller values.

It is also insightful to look at the distances between the learned centroids for both algorithms. We thus show the inner products between the learned centroids in Fig. 5, where we also used the Yeast data set. Ideally, if the diagonal terms are

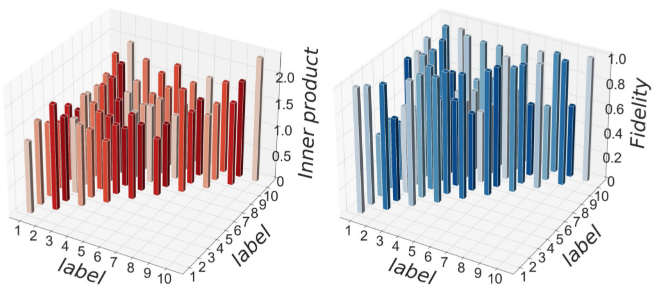


FIG. 5. Inner product between the learned centroids for the K-means algorithm (left panel) and for MPS clustering with $D = 8$ (right panel). Here the Yeast data set is used.

much larger than the off-diagonal terms, then it means the centroids are much better separated. We can see that for the K-means algorithm, centroids cannot be well distinguished with each other at all. Furthermore, we can infer that how centroids are not orthogonal enough will affect the clustering effect. While for MPS clustering, the situation is slightly better (the learning accuracies in both cases are lower than 50%).

[1] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).
[2] P. Zhang, H. Shen, and H. Zhai, Phys. Rev. Lett. **120**, 066401 (2018).
[3] K. Ch'ng, N. Vazquez, and E. Khatami, Phys. Rev. E **97**, 013306 (2018).
[4] E. P. L. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nat. Phys. **13**, 435 (2017).
[5] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Sci. Rep. **7**, 8823 (2017).
[6] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Phys. Rev. X **7**, 031038 (2017).
[7] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. **118**, 216401 (2017).
[8] X.-Y. Dong, F. Pollmann, and X.-F. Zhang, Phys. Rev. B **99**, 121104(R) (2019).
[9] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Nat. Phys. **14**, 447 (2018).
[10] G. Torlai and R. G. Melko, Phys. Rev. Lett. **120**, 240503 (2018).
[11] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo, and S. Severini, npj Quantum Inf. **4**, 28 (2018).
[12] Y. Quek, S. Fort, and H. K. Ng, npj Quantum Inf. **7**, 105 (2021).
[13] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, Nat. Mach. Intell. **1**, 155 (2019).
[14] L.-F. Arsenault, A. Lopez-Bezanilla, O. A. von Lilienfeld, and A. J. Millis, Phys. Rev. B **90**, 155136 (2014).
[15] L.-F. Arsenault, O. A. von Lilienfeld, and A. J. Millis, arXiv:1506.08858.
[16] G. Torlai and R. G. Melko, Phys. Rev. B **94**, 165134 (2016).
[17] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Phys. Rev. X **8**, 021050 (2018).
[18] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101(R) (2017).
[19] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).
[20] K.-I. Aoki and T. Kobayashi, Mod. Phys. Lett. B **30**, 1650401 (2016).
[21] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[22] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Phys. Rev. B **96**, 205152 (2017).
[23] S. Czischek, M. Gärttner, and T. Gasenzer, Phys. Rev. B **98**, 024311 (2018).
[24] C. Guo and D. Poletti, Phys. Rev. E **103**, 013309 (2021).
[25] I. A. Luchnikov, S. V. Vintskevich, D. A. Grigoriev, and S. N. Filippov, Phys. Rev. Lett. **124**, 140502 (2020).
[26] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992).
[27] S. R. White, Phys. Rev. B **48**, 10345 (1993).
[28] U. Schollwöck, Ann. Phys. (NY) **326**, 96 (2011).
[29] I. Oseledets and E. Tyrtyshnikov, Lin. Alg. Applic. **432**, 70 (2010).
[30] D. Savostyanov and I. Oseledets, in *Proceedings of the 2011 International Workshop on Multidimensional (nD) Systems* (IEEE, New York, 2011), pp. 1–8.
[31] E. Stoudenmire and D. J. Schwab, in *Advances in Neural Information Processing Systems*, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Proceedings of NIPS (NeurIPS, San Diego, CA, 2016), pp. 4799–4807.
[32] E. M. Stoudenmire, Quantum Sci. Technol. **3**, 034003 (2018).
[33] Z.-Z. Sun, C. Peng, D. Liu, S.-J. Ran, and G. Su, Phys. Rev. B **101**, 075135 (2020).
[34] Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, Phys. Rev. X **8**, 031012 (2018).
[35] C. Guo, Z. Jie, W. Lu, and D. Poletti, Phys. Rev. E **98**, 042114 (2018).
[36] M. R. Karim, O. Beyan, A. Zappa, I. G. Costa, D. Rebholz-Schuhmann, M. Cochez, and S. Decker, Brief. Bioinform. **22**, 393 (2021).
[37] C. Niu, H. Shan, and G. Wang, arXiv:2103.09382.
[38] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan, in *International Workshop on Algorithms and Models for the Web-Graph* (Springer, New York, 2007), pp. 56–67.
[39] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv:1307.0411.
[40] N. Wiebe, A. Kapoor, and K. Svore, Quantum Inf. Comput. **15**, 0318 (2015).
[41] X. Xie, L. Duan, T. Qiu, and J. Li, Sci. Rep. **11**, 1 (2021).
[42] J. B. Macqueen, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (University of California Press, Berkeley, CA, 1967).
[43] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, Mach. Learn. **75**, 245 (2009).
[44] S. Lloyd, IEEE Trans. Inf. Theory **28**, 129 (1982).
[45] A. Y. Ng, M. I. Jordan, and Y. Weiss, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2002), pp. 849–856.
[46] U. von Luxburg, Stat. Comput. **17**, 395 (2007).
[47] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy, J. ACM **59**, 1 (2012).
[48] M. E. Celebi, H. A. Kingravi, and P. A. Vela, Expert Syst. Applic. **40**, 200 (2013).
[49] O. Bachem, M. Lucic, H. Hassani, and A. Krause, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2016), pp. 55–63.
[50] D. Lim, R. Vidal, and B. D. Haeffele, arXiv:2011.14859.
[51] B. Zhang, M. Hsu, and U. Dayal, Hewlett-Packard Labs, Technical Report No. HPL-1999-124 55 (1999).
[52] J. Xu and K. Lange, in *International Conference on Machine Learning* (PMLR, Maastricht, The Netherlands, 2019), pp. 6921–6931.
[53] R. C. De Amorim and B. Mirkin, Pattern Recog. **45**, 1061 (2012).
[54] S. Chakraborty and S. Das, Pattern Recog. Lett. **100**, 67 (2017).
[55] P. Huang, Y. Huang, W. Wang, and L. Wang, in *2014 22nd International Conference on Pattern Recognition* (IEEE, New York, 2014), pp. 1532–1537.
[56] F. Ding, F. Luo, and Y. Yang, arXiv:1911.05210.
[57] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Menlo Park, CA, 2019), Vol. 33, pp. 4610–4617.
[58] L. Mahon and T. Lukasiewicz, in *German Conference on Artificial Intelligence (Künstliche Intelligenz)* (Springer, Berlin, 2021), pp. 158–178.

[59] C. Guo, K. Modi, and D. Poletti, Phys. Rev. A **102**, 062414 (2020).

[60] L. Tagliacozzo, T. R. de Oliveira, S. Iblisdir, and J. I. Latorre, Phys. Rev. B **78**, 024410 (2008).

[61] T. O. Kvalseth, IEEE Trans. Syst., Man, Cyberne. **17**, 517 (1987).

[62] J. Xie, R. Girshick, and A. Farhadi, in *International Conference on Machine Learning* (PMLR, Maastricht, The Netherlands, 2016), pp. 478–487.

[63] X. Guo, L. Gao, X. Liu, and J. Yin, in *Ijcai* (International Joint Conferences on Artificial Intelligence, Melbourne, Australia, 2017), pp. 1753–1759.

[64] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, New York, 2017), pp. 5736–5745.

[65] C. You, C.-G. Li, D. P. Robinson, and R. Vidal, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2016), pp. 3928–3937.

[66] N. Mrabah, N. M. Khan, R. Ksantini, and Z. Lachiri, Neural Networks **130**, 206 (2020).

[67] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, in *Proceedings of the 30th International Conference on Neural Information Processing Systems* (NIPS, Berkeley, CA, 2016), pp. 2180–2188.

[68] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2019), pp. 4066–4075.

[69] H.-L. Huang, W.-S. Bao, and C. Guo, Phys. Rev. A **100**, 032305 (2019).