SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Density peak clustering using tensor network

Xiao Shi[1,2] & Yun Shang[1,3*]

[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*

Appendix A Related works

Appendix A.1 Density-based clustering algorithm

In the field of clustering, density-based algorithms offer an alternative approach compared to traditional K-means methods that are limited in their ability to handle non-convex data sets. The fundamental principle of density clustering is rooted in considering data point density and the interconnectivity between samples, and expanding clusters accordingly. A prominent instance of such an algorithm is the DBSCAN algorithm [1]. The DBSCAN method involves identifying high-density samples and linking adjacent high-density points to form clusters. Its widespread use has sparked the creation of various advanced versions, including GDBSCAN [2], Recon-DBSCAN [3], RNN-DBSCAN [4], and OPTICS [5].

The DPC algorithm [6], introduced in 2014, presents a novel approach to density-based clustering. This algorithm operates on the assumption that the local density of data points near the cluster center is relatively low, and the distance between a cluster center and high-density points is substantial. To accomplish the clustering task, the DPC algorithm computes two values for each data point x_i: the local density ρ_i and the distance to the closest high-density point δ_i. Let d_{ij} represents the distance between points x_i and x_j, d_c is a predefined cut-off distance. The definition of ρ_i is given as:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{A1}$$

where

$$\chi(x) = \begin{cases} 1 & x < 0 \\ 0 & otherwise \end{cases} \tag{A2}$$

In the context of clustering, when the sample size is limited, the Gaussian kernel is often utilized as an alternative to the density method (as defined in Equation 1) for calculating the density of data points. The Gaussian kernel function is defined as follows:

$$\rho_i = \sum_j exp(-\frac{d_{ij}^2}{d_c^2}) \tag{A3}$$

δ_i is then defined by

$$\delta_i = \begin{cases} \underset{j:\rho_i<\rho_j}{min}(d_{ij}) & \rho_i \neq max(\rho) \\ \\ \underset{j}{max}(d_{ij}) & otherwise \end{cases} \tag{A4}$$

Following the computation of ρ_i and δ_i, the points exhibiting high values of both parameters are selected as cluster centers by means of a decision graph. The remaining data points are then assigned to the nearest neighbor cluster center. While the DPC algorithm has demonstrated efficacy as a simple and effective approach to clustering, it is not without limitations. For instance, there exists no objective criterion for defining the size of a data set and determining whether the Gaussian kernel or the formula 1 method should be utilized. Furthermore, the DPC algorithm only considers the Euclidean distance in its classification procedure, which can result in "chain reaction" errors. To address these shortcomings, various advancements have been made in recent years, such as the optimization of the density function in KNN-DPC [7] and the introduction of a graph-based connectivity estimation strategy in DPC-CE [8].

* Corresponding author (email: shangyun@amss.ac.cn)

Appendix A.2 Tensor Network Machine Learning Algorithms

Recently, the tensor network method has become an important theoretical and computational tool in the fields of classical statistics and quantum many-body physics [9–11]. At the same time, machine learning inspired by tensor networks becomes an emerging topic. Tensor networks are able to capture the underlying structures and patterns in complex data by efficiently representing high-dimensional tensors using a much lower number of parameters. This makes them well-suited for tasks such as data compression, feature extraction, and dimensionality reduction, which are all important tasks in machine learning. Since tensor networks can map precisely to quantum circuits, one exciting line of research in tensor network machine learning is to deploy and even train such models on quantum hardware [12, 13]. At the same time, since tensor networks can learn probability distributions for given data, this also naturally bridges quantum many-body physics and machine learning.

Tensor network can be combined with machine learning in two ways. One is to directly use tensor network as machine learning model architecture. The other is to use tensor networks to compress layers or other auxiliary tasks in the neural network architecture. In [14], MPS have been used as proof-of-principle for tensor networks in supervised learning. In [15], MPS have been used to parameterize generative models, and update the tensors in MPS by gradient descent. Sun et al. [16] implemented the classification task with MPS, and its results exceeded a series of baseline models such as naive Bayes classifiers, SVMs, etc. Alexander et al. [17] transformed the weight matrices of fully connected layers into the Tensor Train format, significantly reducing the number of parameters and memory footprint, while preserving the network's expressive power. Gao et al. [18] replaced the fully connected layers in the neural network with MPO, which greatly compresses the parameters without affecting the accuracy. By combining with some good initialization methods, Shi et al. [19] proposed to use tensor network to solve the clustering problem, and they achieved the best clustering results on some data sets. However, in contrast to K-means algorithm, here our quantum clustering algorithm is based on the density peak modeling method. Therefore, unlike K-means, we do not require prior knowledge of the number of clusters during the clustering process. This feature is closer to real-world scenarios and correspondingly increases the complexity of the algorithm design.

Appendix B Algorithms

Appendix B.1 Training MPS

In order to train an MPS with all the data, the first thing is to map all the data into quantum states Suppose we have data set $S = \{\vec{x}^1, \vec{x}^2, .., \vec{x}^n\}$. The i-th element $\vec{x}_i^k$ of the input vector $\vec{x}^k = (x_1^k, x_2^k, ..., x_m^k)$ of length m is mapped to a superposition of quantum states $|0\rangle$ and $|1\rangle$, which can be described as

$$|\psi_i^k\rangle = cos(\frac{\pi}{2}x_i^k)|0\rangle + sin(\frac{\pi}{2}x_i^k)|1\rangle \tag{B1}$$

Therefore, the input vector $\vec{x}^k$ can be written as the tensor product of $|\psi_i^k\rangle$

$$|\Psi(\vec{x}^k)\rangle = |\psi_1^k\rangle \otimes |\psi_2^k\rangle \otimes ... \otimes |\psi_m^k\rangle \tag{B2}$$

It is expressible in the form of a tensor network state, as

$$|\Psi_k^{\vec{\sigma}}\rangle = \sum_{\alpha_0,\alpha_1,...,\alpha_m} X_{k,\alpha_0,\alpha_1}^{\sigma_0} X_{k,\alpha_1,\alpha_2}^{\sigma_1} ... X_{k,\alpha_{m-1},\alpha_m}^{\sigma_{m-1}} |\sigma\rangle \tag{B3}$$

where σ_i is its physical indices, α_i is its auxiliary indices with $\alpha_0 = \alpha_1 = ... = \alpha_n = 1$. Each X_i represents a $1 \times 2 \times 1$ third-order tensor whose elements are

$$X_{i,1,1}^1 = cos(\frac{\pi}{2}x_i^k), X_{i,1,1}^2 = sin(\frac{\pi}{2}x_i^k) \tag{B4}$$

Analogously, we randomly generate a quantum state $\Phi^{\vec{\tau}}$ of length m and bond dimension equals to D in the form of MPS:

$$|\Phi^{\vec{\tau}}\rangle = \sum_{\beta_0,\beta_1,...,\beta_m} Y_{\beta_0,\beta_1}^{\tau_0} Y_{\beta_1,\beta_2}^{\tau_1} ... Y_{\beta_{m-1},\beta_m}^{\tau_{m-1}} |\tau\rangle \tag{B5}$$

Where β is an auxiliary indices, which determines the upper limit of the entanglement entropy that this MPS state can accommodate, and $1 \leqslant \beta_i \leqslant D$. After completing the above steps, a variational matrix product states algorithm will be used to update the parameters in the MPS. We consider the utilization of MPS to generate a probability distribution that represents the probability distribution of the target data. In this context, we adopt the Kullback-Leibler (KL) divergence [20] as a metric to quantify the proximity between two probability distributions. Given the empirical data distribution $P(x)$ and the model distribution $Q(x;\theta)$, where θ denotes the parameters of the model, the KL divergence is expressed as follows:

$$\begin{aligned} D_{KL}(P(x)||Q(x;\theta)) &= \sum_x P(x) \ln \left(\frac{P(x)}{Q(x;\theta)}\right) && \text{(B6)} \\ &= \sum_x P(x) \ln P(x) - \sum_x P(x) \ln Q(x;\theta) && \text{(B7)} \end{aligned}$$

This expression encapsulates the comparison between the empirical data distribution and the model distribution, facilitating the assessment of their relative similarities. The first term on the right-hand side corresponds to the entropy of the empirical data distribution, capturing the level of uncertainty or disorder within the data. The second term can be understood as the expectation of $Q(x;\theta)$ relative to $P(x)$. Thus, the entire KL divergence serves as a measure to quantify the disparities between the two distributions, $P(x)$ and $Q(x;\theta)$. In the case where the data set contains no duplicate entries, the entropy of $P(x)$ is equivalent to $-\ln|\Gamma|$, where $|\Gamma|$ represents the number of elements in the data set. By substituting $P(x) = \frac{1}{|\Gamma|}\sum_{x'\in\Gamma}\delta(x-x')$ into the expression, the second term transforms into the negative log-likelihood (NLL) function:

$$\mathcal{L} = -\sum_x P(x)\ln Q(x;\theta) \tag{B8}$$

$$= -\frac{1}{|\Gamma|}\sum_{x\in\Gamma}\ln Q(x;\theta) \tag{B9}$$

Due to the non-negativity of the KL divergence, we establish that $\mathcal{L} \geqslant -\sum_x P(x)\ln P(x)$. This noteworthy observation signifies that the entropy of the empirical data distribution sets a lower limit for the NLL. When $\mathcal{L} = -\sum_x P(x)\ln P(x)$, the model's distribution precisely coincides with the empirical data distribution. This remarkable alignment indicates that the model achieves an exact replication of the empirical data, capturing its intricate characteristics and faithfully representing its underlying distribution.

Specifically, the randomly initialized quantum state $\Phi^{\vec{\tau}}$ is converted into canonical forms, and the NLL function is adopted as the loss function

$$f(\Phi^{\vec{\tau}}) = -\frac{1}{|\Gamma|}\sum_i \ln|\langle\Psi_i^{\vec{\sigma}}|\Phi^{\vec{\tau}\dagger}\rangle|^2 \tag{B10}$$

Gradient descent is used to update each tensor when the quantum state $\Phi^{\vec{\tau}}$ satisfies the normalization condition:

$$Y^{\tau_i} - \eta\frac{\partial f}{\partial Y^{\tau_i}} \rightarrow Y^{\tau_i} \tag{B11}$$

where η is the learning rate. Tensors will be updated from the first to the last, and then back, this process is called a sweep. The iteration process will stop when the sweep reaches the maximum number of iterations or the loss function converges. After this, the quantum state $\Phi^{\vec{\tau}}$ will give the joint distribution probability of pixels.

Appendix B.2 Generation of clusters

After training through the algorithm in the last section, a quantum state $\Phi^{\vec{\tau}}$ with bond dimension equal to D is obtained. Similar to the DPC algorithm, we define ρ and δ by fidelity measure instead of distance measure, and replace the Gaussian kernel function with the Sigmoid kernel function:

$$\rho_i = tanh(f_i/(10 * f_c)) \tag{B12}$$

where $f_i = |\langle\Psi_i^{\vec{\sigma}}|\Phi^{\vec{\tau}}\rangle|$ represents the fidelity between quantum state $\Psi_i^{\vec{\sigma}}$ and $\Phi^{\vec{\tau}}$, f_c is a predefined cutoff distance. Similarly, computing the fidelity $f_{ij} = |\langle\Psi_i^{\vec{\sigma}}|\Psi_j^{\vec{\sigma}}\rangle|$ of the quantum states of the data points x_i and x_j, we get the definition of δ_i as follows

$$\delta_i = \min_{j:\rho_j>\rho_i}(f_{ij}) \tag{B13}$$

Different from the DPC algorithm, we adopt a method similar to Ref [21] here, and select the point with relatively large rho and delta as the local cluster center. The definition of local clusters is given in Def 1.

Def 1: (Local Cluster Center)
A data point x_i is defined as a local cluster center if it satisfies the conditions $\delta_i > f_c$ and $\rho_i > \overline{\rho}$, where $\overline{\rho}$ is the average of all point density.

After the local cluster centers have been identified, the remaining points can be assigned to their nearest higher-density neighbor to generate a set of local clusters. Therefore, L local clusters $(C^{(1)}, C^{(2)}, ..., C^{(L)})$ are obtained.

Similar to the DBSCAN algorithm, for each local cluster $C^{(k)}$, core point and border point need to be defined. Therefore, for the local cluster $C^{(k)}$, an MPS representation $\Phi_k^{\vec{\tau}}$ is trained using the method in Section 3.1. The definitions of core point and border point are given in Def 2.

Def 2: (Definition of core point and border point)
Assuming that the point c_i in the local cluster C^k satisfies $f'_i > \overline{f}'_k$, where $f'_i = |\langle\Psi_i^{\vec{\sigma}}|\Phi_k^{\vec{\tau}}\rangle|, \overline{f}'_k = \frac{1}{n_k}\sum\limits_{c_j\in C^{(k)}}|\langle\Psi_j^{\vec{\sigma}}|\Phi_k^{\vec{\tau}}\rangle|$, and n_k is the number of points in the local cluster C^k. Then the point c_i is called the core point of the local cluster C^k, otherwise c_i is the border point.

In the following, in order to determine the connectivity between local clusters, we will give the definitions of Density Directly-connectable and Density Connectable respectively.

Def 3: (Density Directly-connectable of Clusters)
If there are core points $c^i \in C^i$, $c^j \in C^j$ in local clusters C^i and C^j, and $f_{ij} < f_d$, where f_d is a predefined parameter, then the local clusters C^i and C^j is directly-connectable.

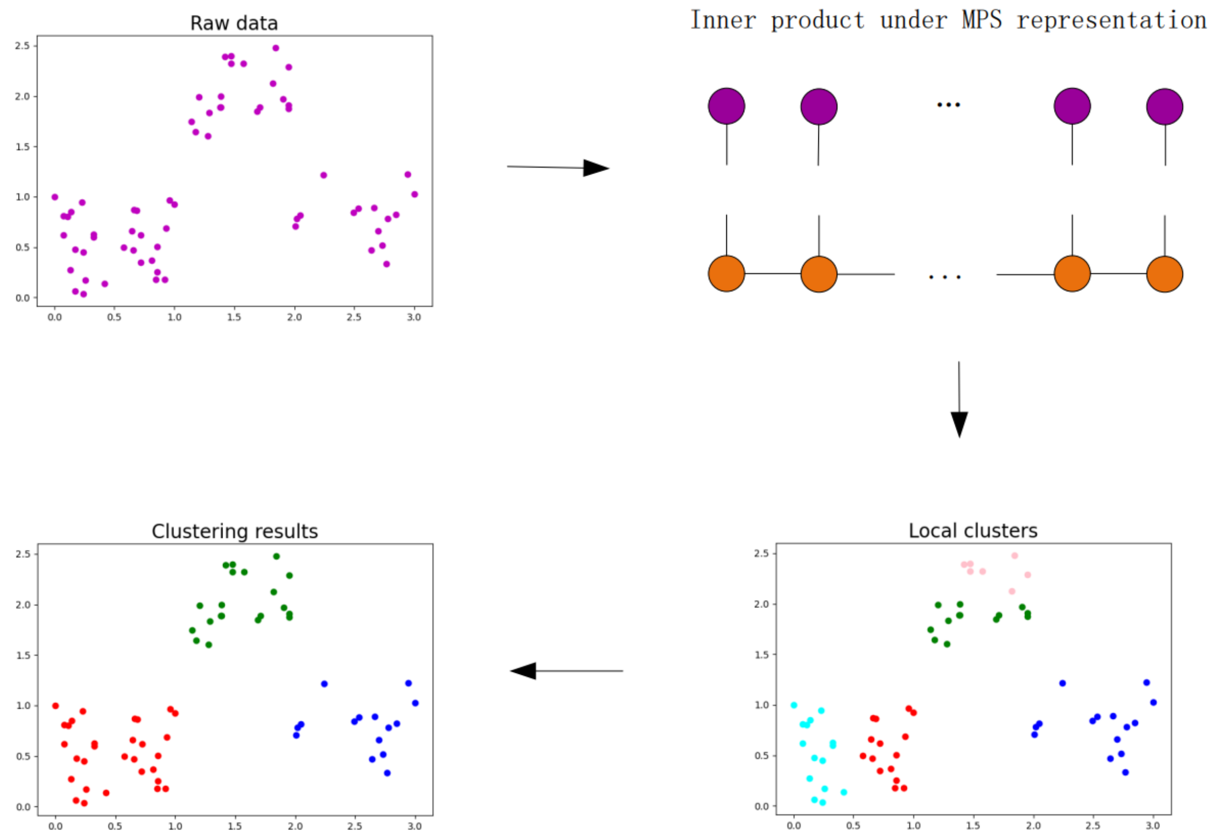

Figure B1 A flowchart of the structure tensor network-based density peak clustering algorithm. The first step involves mapping the raw data into a high-dimensional Hilbert space and training a MPS to capture its probability distribution. This results in the identification of local clusters based on the fidelity of the MPS representation. Finally, the merging of local clusters that satisfy the Density Connectable criterion yields the ultimate clustering outcome.

Def 4: ((Density Connectable of Clusters)

If there are paths $C^i, C^1, C^2, ..., C^n, C^j$, where C^i and C^1, C^k and $C^{(k+1)}$, C^n and C^j are all Density Directly-connectable, then C^i and C^j are called Density Connectable.

Finally, all local clusters with density connectable are merged to get the final clustering result. It is easy to verify that our algorithm does not require an input number of classes and can find clusters of arbitrary shapes. The flowchart can be found in Fig.B1.

Appendix C Experimental results

Appendix C.1 Experimental results on synthetic data sets

Our first experiment is to apply the algorithm to six commonly used synthetic data sets: Twomoons, Jain, Threecircles, Smile, Fourlines, and Unbalance. Both Twomoons and Jain data sets consist of two moon-shaped clusters, but the size of the two data sets is different, and the density of the clusters is not the same. Some manifold data sets, such as Threecircles and Smile, can be used to evaluate the performance of the algorithm on non-spherical clusters. Fourline is represented as a linearly separable data set consisting of 4 linearly non-uniform density clusters. Unbalance is a large-scale synthetic data set consisting of multiple spherical clusters. Their main characteristics are summarized in Table C1. Note that in the process of mapping data into quantum states, since the mapping function is a trigonometric function, we first scale the data between 0 and 1 using max-min normalization to avoid problems with periodicity. In comparative experiments, the proposed algorithm is compared with other methods such as K-means [22], DPC [6], DBSCAN [1], SNN-DPC [23], DGDPC [24], DPC-CE [8]. Among them, K-means and DBSCAN are commonly used as classical clustering algorithms. The DPC algorithm is used as the benchmark algorithm. SNN-DPC, DGDPC and DPC-CE are three better revisions of DPC algorithm. The parameters that need to be pre-specified in all these algorithms are listed in Table C2.

Fig. C1- C6 visualizes the differences between the DPC algorithm, our algorithm, and its true label. We use three kinds of popular external evaluation index of clustering algorithms called FMI, ARI and NMI to evaluate all the clustering results. Table C3 compares the effectiveness of these methods numerically. And in the iterative process of MPS, the upper limit of its sweeps is set to 30. It can be seen from the results that our method can achieve 100% accuracy on 5 of the data sets. Significantly better than DBSCAN and DPC algorithms on Threecircle and Jain data sets. Our algorithm also outperforms DPC on the Twomoons, Smile and Fourlines data sets. Only on the Unbalance data set, our algorithm is slightly lower than DPC and DBSCAN.

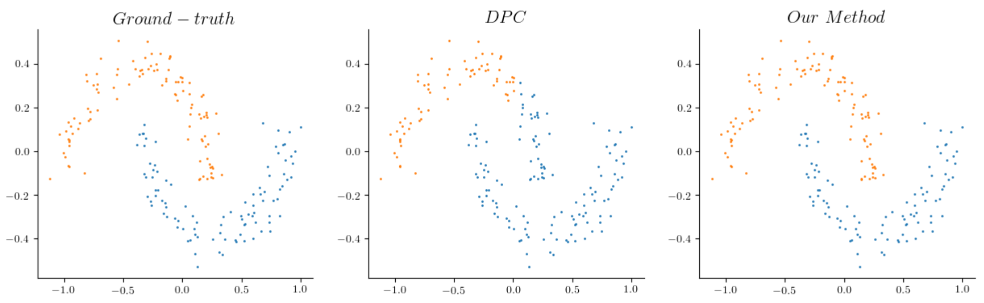

Figure C1 Compared results of DPC and our method on Twomoons data set.

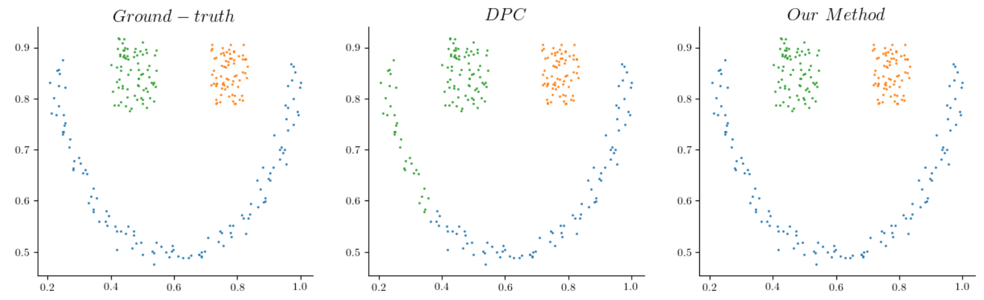

Figure C2 Compared results of DPC and our method on Smile data set.

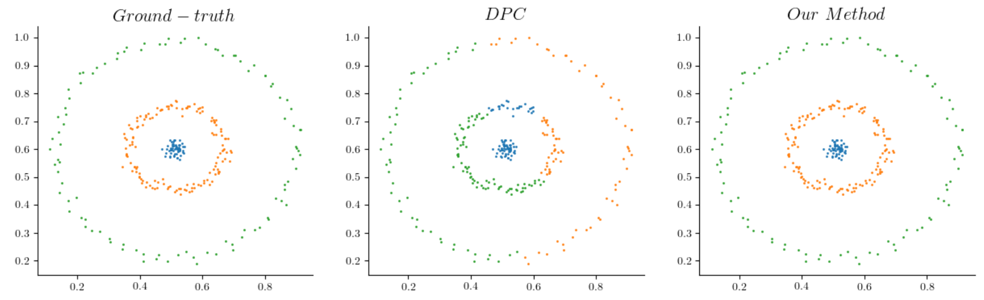

Figure C3 Compared results of DPC and our method on Threecircles data set.

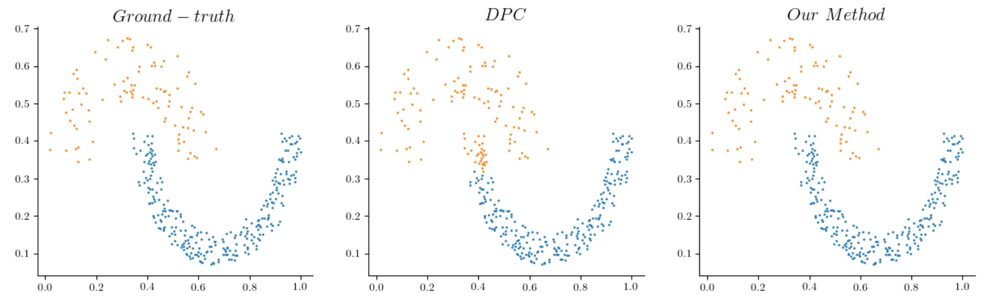

Figure C4 Compared results of DPC and our method on Jain data set.

Table C1 Description of the synthetic data sets.

Data set	Point	Attribute	Clusters
Twomoons	200	2	2
Smile	266	2	3
Threecircles	299	2	3
Jain	373	2	2
Fourlines	512	2	4
Unbalance	6500	2	8

Table C2 Parameters configurations of compared algorithms and our method.

Algorithms	Parameters setting
K-means	*cluster number k*
DPC	$d_c = 1\% \sim 5\%$
DBSCAN	$Eps = 0.5 \sim 3, MinPts = 4$
SNN-DPC	$d_c = 2\% \ \sim 3\%, 3 \leqslant K \leqslant 50$
DGDPC	$d_c = 1\% \sim 5\%, m = 0.1 \sim 1$
DPC-CE	$d_c = 2\%, T_r = 0.25, P_r = 0.3$
Our method	$d_c = 0.05\% \sim 0.6\%, f_d = 0.93 \sim 0.999, D = 8$

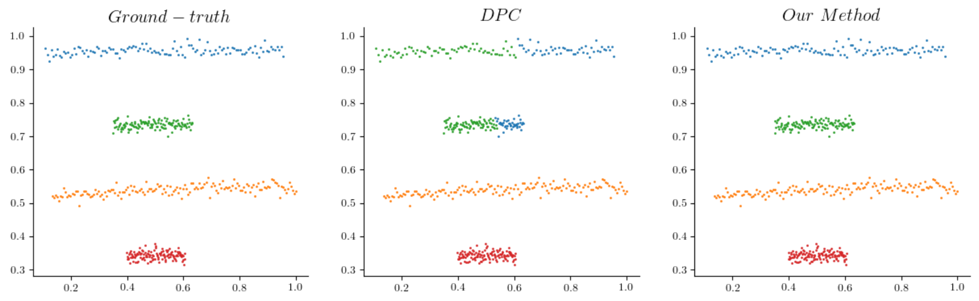

Figure C5 Compared results of DPC and our method on Fourlines data set.

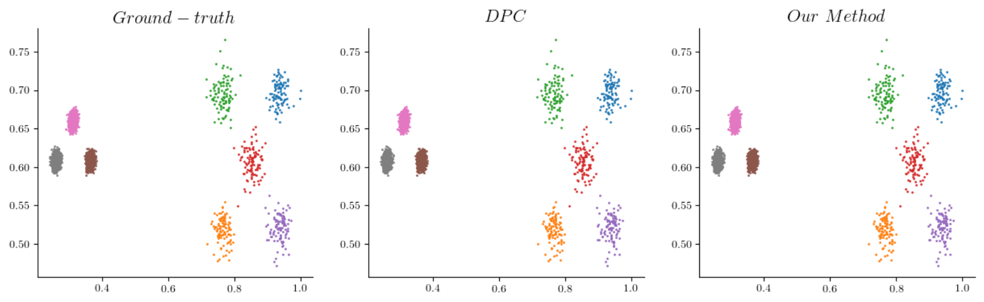

Figure C6 Compared results of DPC and our method on Unbalance data set.

Table C3 Clustering results of all methods in real world data set, the bond dimension in MPS is set to be 8 and the best results are stressed in bold.

Method	Twomoons			Smile		
	FMI	ARI	NMI	FMI	ARI	NMI
K-means	0.5683	0.1401	0.1077	0.6155	0.4022	0.5318
DPC	0.7175	0.4068	0.4584	0.7826	0.6683	0.7623
DBSCAN	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
SNN-DPC	0.7175	0.4068	0.4587	**1.0**	**1.0**	**1.0**
DGDPC	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DPC-CE	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Our Method	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Method	Threecircles			Jain		
	FMI	ARI	NMI	FMI	ARI	NMI
K-means	0.4045	0.0555	0.1637	0.7005	0.3241	0.3690
DPC	0.5161	0.2514	0.3703	0.8819	0.7146	0.6522
DBSCAN	0.9193	0.8739	0.8647	0.9767	0.9473	0.8930
SNN-DPC	0.7160	0.5310	0.6860	**1.0**	**1.0**	**1.0**
DGDPC	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DPC-CE	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Our Method	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Method	Fourlines			Unbalance		
	FMI	ARI	NMI	FMI	ARI	NMI
K-means	0.6462	0.5024	0.6725	0.8142	0.8463	0.8107
DPC	0.7850	0.7115	0.7698	**1.0**	**1.0**	**1.0**
DBSCAN	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
SNN-DPC	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DGDPC	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DPC-CE	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Our Method	**1.0**	**1.0**	**1.0**	0.9999	0.9999	0.9994

Table C4 Description of the real world data set.

Data set	Point	Attribute	Clusters
Wine	178	13	3
Vehicle	846	18	4
Yeast	1484	9	10
Abalone	4177	8	29

Appendix C.2 Experimental results on real world data sets

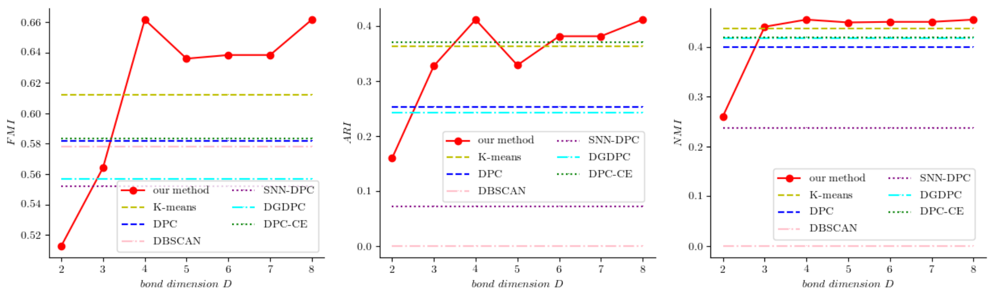

Figure C7 Our algorithm's performance on the Wine data set, as well as other top-performing algorithms, was evaluated under various bond dimensions. A higher bond dimension D allows for more detailed representations, but also increases the computational complexity of the algorithm.

In this section, experiments are validated on 4 real world data sets. Details for these data sets are given in Table C4, and parameter information is also listed in Table C2. According to the entanglement entropy area law, in the quantum state represented by MPS, the bond dimension D determines the upper limit of the entanglement entropy it can accommodate, and the parameter space of the MPS grows at the scale of D^2. Therefore, taking the Wine data set as an example, we compared the clustering performance under different bond dimension D, which is shown in FIG. C7. As can be seen from the figure, when $D \geqslant 4$, our algorithm has a significant improvement in the value of FMI compared to other algorithms. And at $D \geqslant 3$, our algorithm already exceeds the performance of other algorithms on NMI. We believe that the quality of the clustering results has a great relationship with the entanglement entropy of the trained MPS. The larger the entanglement entropy means the smaller the local entanglement of the data, the stronger the expressability of the MPS to the data. More details can be found in Sec. 4.4

We still use the three metrics of NMI, ARI, and FMI to compare the clustering results with other methods. In Table C5 we present the results of our algorithm when $D = 8$. It can be seen that the performance of our three indicators is currently the best on the Wine data set, and our FMI outperforms other methods on the Vehicle and Yeast data sets. On the Abalone data set, its FMI and NMI are the highest among these methods, and our algorithm are only 0.0210, 0.0103 and 0.0660 lower than SNN-DPC in FMI, ARI, and NMI, respectively. In terms of ARI, the result obtained by our algorithm is 0.0416, close to the largest one (DPC-CE's ARI = 0.0613).

All in all, the clustering results are encouraging, and they show that the tensor network clustering algorithm can achieve better results than other existing algorithms even when the number of clusters is not known in advance. It demonstrates the excellent ability of our algorithm to handle real-world data sets.

Appendix C.3 Experimental results on image data sets

Finally, we compare our algorithm with other good clustering algorithms on computer vision benchmark data sets to demonstrate the effectiveness of our algorithm. Their statistical information is shown in Table C6. Both the MNIST and Fashion data sets have 70,000 images, each containing 28*28 grayscale pixels. The USPS data set has a relatively small number of images, with 9298 images, each containing 16*16 grayscale pixels. Like the method we deal with in the above, we use max-min normalization to preprocess the data, which performs a linear transformation on the original data.

When data with high-dimensional feature space is involved in practical applications, a series of preprocessing steps are needed in order to obtain better clustering, considering the time cost. Here we first use the autoencoder to reduce the dimension of the data. The autoencoder consists of two parts. The first part is the encoder E, which compresses the initial data x to the latent space through a learned feature vector $e = f(x)$, and the second part is called decoding D, which learns a new function g that maps the compressed data into the original feature space. The training process of the autoencoder can be expressed as:

$$\underset{(e,d)\in E\times D}{argmin} \quad \epsilon(x, g(f(x))) \tag{C1}$$

where $\epsilon(x, g(f(x)))$ is the reconstruction error between the input data x and $g(f(x))$.

Although autoencoders are a common and efficient way to compress data. But it does not preserve the distance information between data well enough. With this in mind, we need to take a further approach to the data obtained from the autoencoder.

UMAP (Uniform Manifold Approximation and Projection) [25] is a common dimensionality reduction technique that can be used for general nonlinear dimensionality reduction, but also for t-SNE-like visualizations. UMAP is a dimensionality reduction algorithm based on manifold learning technology and topological data analysis ideas. It is divided into two steps, the first step is to learn the manifold structure of the data in the high-dimensional space, and the second step is to find the low-dimensional

Table C5 Clustering results of all methods in real world data set, like before, the bond dimension in MPS is set to be 8 and the best results are stressed in bold.

Method	Wine			Vehicle		
	FMI	ARI	NMI	FMI	ARI	NMI
K-means	0.5835	0.3711	0.4288	0.3590	0.1216	0.1867
DPC	0.5817	0.2535	0.3997	0.3973	0.0829	0.1136
DBSCAN	0.5782	0	0	0.4873	0	0
SNN-DPC	0.5520	0.0728	0.2375	0.3620	0.0532	0.0589
DGDPC	0.5570	0.2436	0.4169	0.4361	**0.4982**	**0.4733**
DPC-CE	0.5834	0.3715	0.4193	0.4254	0.4020	0.4020
Our Method	**0.6616**	**0.4125**	**0.4550**	**0.4951**	0.0006	0.0156

Method	Yeast			Abalone		
	FMI	ARI	NMI	FMI	ARI	NMI
K-means	0.2980	**0.1331**	**0.2436**	0.1119	0.0433	0.1611
DPC	0.4703	0.0107	0.1224	0.1356	0.0348	0.0432
DBSCAN	0.4037	0.0254	0.0296	0.2248	0.0385	0.0980
SNN-DPC	0.4422	0.0121	0.1182	**0.2459**	0.0519	**0.1685**
DGDPC	0.4397	0.0987	0.1226	0.1937	0.0529	0.1066
DPC-CE	0.4705	0.1185	0.1277	0.2250	**0.0613**	0.1373
Our Method	**0.4710**	0.0125	0.0653	0.2249	0.0416	0.1025

Table C6 Description of the common used image data set.

Data set	Point	Attribute	Clusters
MNIST	70000	784	10
Fashion	70000	784	10
USPS	9298	256	10

representation of this manifold. Compared with t-SNE, it significantly improves the speed and better preserves the global structure of the data. Therefore, in practice, we will first use the autoencoder to act on the original data to learn an initial representation, then we relearn the data from the autoencoder by searching for a more clustered manifold using a local distance-preserving manifold learning method. Here, the structure of the autoencoder we use is $FC_{512} \rightarrow FC_{256} \rightarrow FC_{64} \rightarrow FC_{16} \rightarrow FC_{64} \rightarrow FC_{256} \rightarrow FC_{512}$. Where FC_{512} indicates that it is a fully connected layer with 512 neurons. This means that with this autoencoder, we compress the original data into a 16-dimensional latent space. Next, the 16-dimensional data is further reduced to 10-dimensional using the UMAP method, which is then fed into our tensor network clustering algorithm.

Following the above works, we use two metrics, ACC (accuracy) and NMI (normalized mutual information), to evaluate the performance of the algorithm. In Table C7, we present a comparison of our method with other top-performing algorithms. Among all the algorithms, only DBSCAN, DPC, DDC and our method do not need to know the number of clusters in advance, While other algorithms take real clusters numbers as known conditional inputs.

It can be seen that in the algorithm with unknown number of clusters, our algorithm demonstrates the current state-of the-art result on the MNIST and USPS data sets. Although the ACC is a little lower than the DDC algorithm on the Fashion data set, but our results are better than the DDC algorithm on the NMI indicator. Even compared with the state-of-the-art clustering algorithms [19], our algorithm is only 1.24% and 3.22% lower in ACC and NMI on the MNIST data set, which provides a competitive scheme.

Appendix D Analysis

Appendix D.1 Rationality analysis

From the Equation A1 and A3, it can be seen that Gaussian kernel function is used to reflect the distance and density relationship between data points. From Fig. D1(a), it can be observed that the function $y = e^{-x}$ is monotonically decreasing for $x \geqslant 0$. Therefore, in general, when a data point is closer to the rest of the points, i.e. distance d_{ij} is small, then its density ρ_i is high. For our approach, the trained MPS $|\Phi^{\vec{\tau}}\rangle$ describes the probability distribution of the entire data set. Therefore, for data points lying in a high data distribution region, after being transformed into the quantum state $|\Psi_i^{\vec{\sigma}}\rangle$, they will exhibit relatively higher fidelity with $|\Phi^{\vec{\tau}}\rangle$. As shown in Fig. D1(a), the Sigmoid kernel function $y = tanh(x)$ is a monotonically increasing function for $x \geqslant 0$. Therefore, these points also have higher densities. This is the same as the DPC algorithm which means that our algorithm aligns with the DPC algorithm in terms of relative density estimation.

To test this, we took the wine data set as an example and set the bond dimension to 8. We computed distance sum $D_i = \sum_j d_{ij}$ and the fidelity $F_i = |\langle\Phi^{\vec{\tau}}|\Psi_i^{\vec{\sigma}}\rangle|$ for each point. The results are shown in Fig. D1(b). It can be observed that points with higher fidelity will exhibit relatively smaller distance sums to other points, which is consistent with our observations. Furthermore, in the DPC algorithm, δ_i represents the maximum distance between sample point i and other points with higher densities. Since fidelity characterizes the similarity between two quantum states, a higher similarity leads to a larger fidelity. In our method, when transforming the sample points into quantum states as shown in Equation B3, if two sample points are farther apart, their similarity will be lower. Therefore, from Equation B13, it can be concluded that δ_i represents the minimum fidelity between sample point i and all the sample points with higher densities, which is similar to the concept in the DPC algorithm.

Table C7 ACC and NMI of our method compared to other top-performing image clustering models. The algorithm's best results with known and unknown numbers of clusters are stressed in Italian and bold, respectively.

Method	MNIST		USPS		Fashion	
	ACC	NMI	ACC	NMI	ACC	NMI
K-means	53.91	49.04	65.76	60.98	52.22	51.1
DBSCAN	-	-	16.7	0	10.0	0
DPC	-	-	39.0	43.3	34.4	39.8
DEC [26]	84.3	83.4	76.2	76.7	51.8	54.6
IDEC [27]	88.06	86.72	76.05	78.46	52.9	55.7
JULE [28]	96.4	91.3	95.0	91.3	-	-
DEPICT [29]	96.5	91.7	96.4	92.7	39.2	39.2
EnSC [30]	96.3	91.5	61.0	68.4	62.9	63.6
InfoGAN [31]	87.0	84.0	-	-	61.0	59.0
ClusterGAN [32]	95.0	89.0	-	-	63.0	64.0
DualAE [33]	97.8	94.1	86.9	85.7	*66.2*	64.5
ConvDEC [34]	94.0	91.6	78.4	82.0	51.4	58.8
DDC [21]	96.5	93.2	96.7	91.8	**61.9**	68.2
ADSSC-MPS-8 [19]	*99.04*	*97.22*	*98.82*	*96.62*	65.61	*72.15*
MPS-8(ours)	**97.8**	**94.0**	**97.1**	**92.6**	59.99	**68.49**

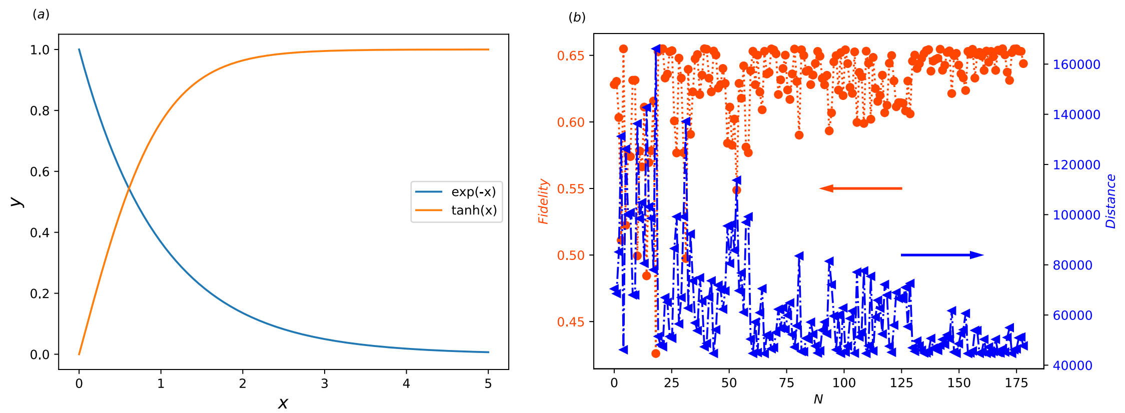

Figure D1 (a) The schematic plot below illustrates the behavior of the functions y=exp(-x) and y=tanh(x) for x ∈ [0,5]. As x increases, it can be observed that y=exp(-x) gradually decreases, while y=tanh(x) gradually increases.(b) In the Wine dataset, the blue dots represent the distance sums between each data point and all other data points. The red dots represent the fidelity between the data points transformed into quantum states and the trained quantum state. Generally, regions with smaller blue values correspond to larger red values.These visual representations provide a concise overview of the behavior and relationships described in the given statements, allowing for a better understanding of the trends and patterns exhibited by the functions and the Wine dataset.

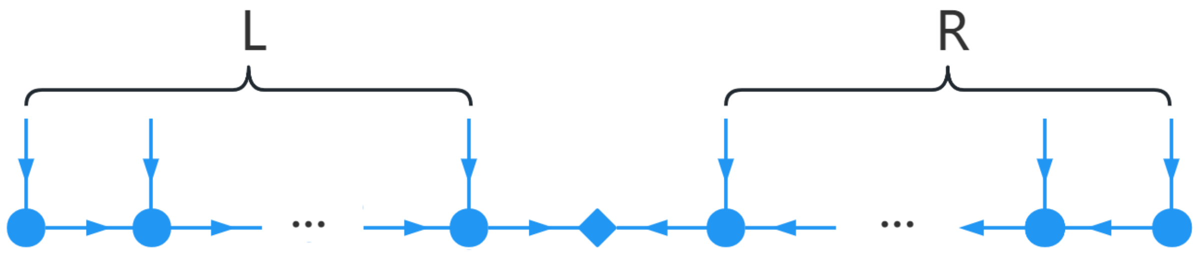

Figure D2 The specific structural form of MPS. The MPS is divided into two subsystems L and R from the middle to calculate the entanglement entropy between them. Among them, the subsystem L satisfies the left canonical form, and the subsystem R satisfies the right canonical form.

Obviously, these observations assert that the definitions of density and core points in our approach are well-defined and reasonable. From the point of research and applications, these may aid in deepening the understanding and interpretation of data density distribution characteristics.

Appendix D.2 Entanglement entropy of the MPS

Entanglement entropy is a fundamental concept in quantum physics that characterizes the entanglement present in a quantum system. It is obtained by dividing the system into two subsystems and measuring the amount of entanglement between them. The larger the entanglement entropy, the smaller the local correlations within the data and vice versa. One important property of MPS is that they obey an area law for entanglement entropy, which means that the entanglement entropy of an MPS with bond dimension χ and length m is upper bounded by a value that is independent of the system length. This property has significant implications for the computational complexity of simulating and manipulating MPS, as well as for the behavior of quantum many-body systems in general.

For a center-normalized MPS, the entanglement entropy reaches its maximum value when the Schmidt coefficients are all equal to $1/\sqrt{\chi}$ [35]. This can be seen by dividing the MPS into two subsystems at its orthogonal center and using Schmidt decomposition to determine the entanglement entropy.

Take the Wine data set as an example. We consider the canonical form of the MPS trained on all data in the data set to obtain the normalized entanglement spectrum. We convert it to the center canonical form at the $\lfloor m/2 \rfloor$ bond and split it into left and right parts using Schmidt decomposition

$$\Phi^{\vec{\sigma}} = \sum_{i=1}^{r} \lambda_i |\phi_i^L\rangle \otimes |\phi_i^R\rangle \tag{D1}$$

Where r is the Schmidt rank, λ_i is the Schmidt coefficient, which is a non-negative real number and satisfies $\sum_{i=1}^{r} \lambda_i^2 = 1$. As shown in Fig.D2. Therefore, its entanglement entropy S can be obtained in the following

$$S = -\sum_{i=1}^{r} \lambda_i^2 ln(\lambda_i^2) \tag{D2}$$

The result is summarized in Fig. D3. It shows that the entanglement entropy is relatively large when bond dimension equals to 4 and 8, which is consistent with the clustering results that can work better under the corresponding bond. And its trend is basically the same as that of FMI, ARI and NMI. It can be seen that the final clustering result has an important relationship with the entanglement entropy of the trained MPS. Also, compared with other classical algorithms, it can be seen that establishing entanglement through quantum methods is more conducive to finding hidden relationships between data. The strength of nonlocality has been shown to impact the performance of the clustering algorithm in this example, with stronger nonlocality leading to improved results.

Appendix D.3 Complexity analysis

We investigate the time complexity of our structure tensor network-based density peak clustering algorithm, applied to a data set $S = \{x^1, x^2, ..., x^n\}$ of length L with virtual bond dimension D, physical index dimension d, and generate k local clusters. The time complexity of our algorithm includes the following parts: (1) training the MPS for the entire data with complexity $O(LD^3)$; (2) Compute the densities ρ_i and δ_i for all data points:Consider the inner product of two quantum states $|\langle \Psi_i^{\vec{\sigma}} | \Phi^{\vec{\sigma}\dagger} \rangle| = \sum\limits_{\sigma} X^{\sigma_L^\dagger} ... X^{\sigma_2^\dagger} X^{\sigma_1^\dagger} Y^{\sigma_1} Y^{\sigma_2} ... Y^{\sigma_L}$.As each X^{σ_i} is a tensor of size $1 \times 2 \times 1$, the inner product expression can be re-written as $|\langle \Psi_i^{\vec{\sigma}} | \Phi^{\vec{\sigma}\dagger} \rangle| = (\sum\limits_{\sigma_1} X^{\sigma_1^\dagger} Y^{\sigma_1})(\sum\limits_{\sigma} X^{\sigma_2^\dagger} Y^{\sigma_2})...(\sum\limits_{\sigma} X^{\sigma_L^\dagger} Y^{\sigma_L})$. And the time complexity is $O(LdD^2)$. Thus, the time complexity for computing all point densities ρ_i is $O(nLdD^2)$, and the time complexity for calculating δ_i is $O(Ln^2)$; (3) Similar to the above, the time complexity for finding the core and border points is $O(kLD^3 + nLdD^2)$. Therefore, the overall time complexity of our algorithm is $O(kLD^3 + nLdD^2 + Ln^2)$.

References

1 M. Ester, H.-P. Kriegel, J. Sander, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

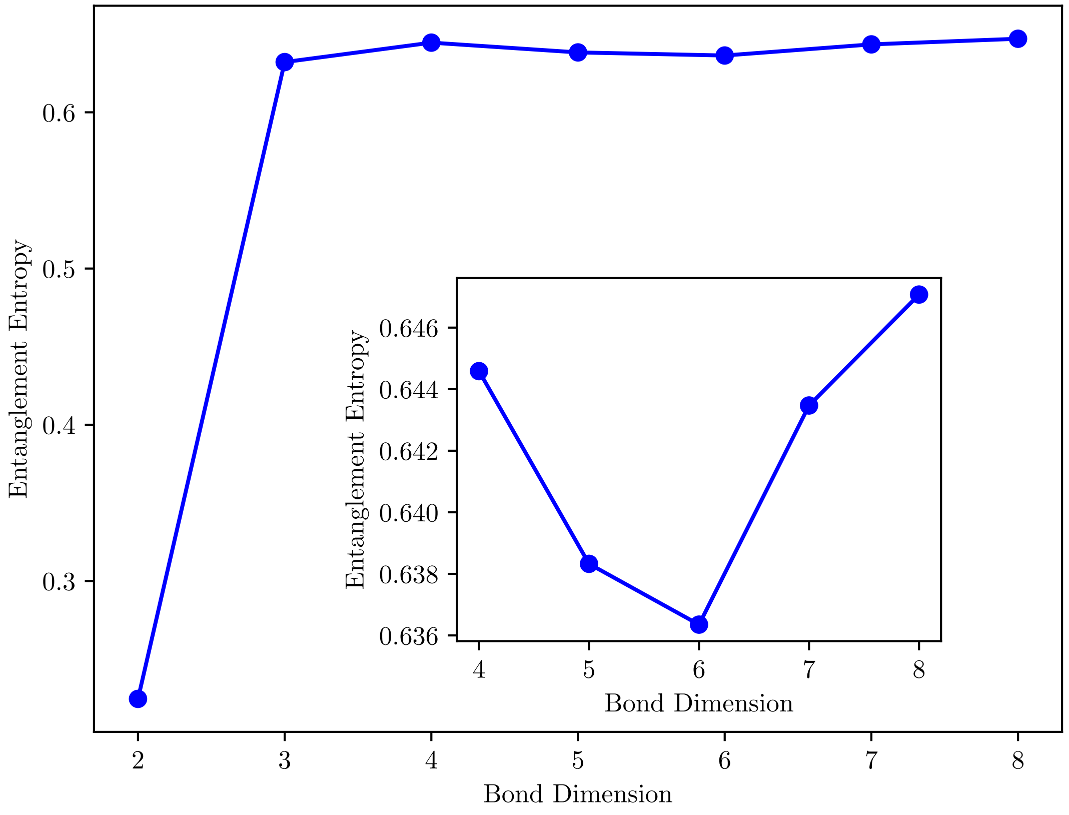

Figure D3 Taking the Wine data set as an example, the entanglement entropy measured at the $\lfloor m/2 \rfloor$ bond. It compares the entanglement entropy MPS can accommodate under different bond dimensions.

2 J. Sander, M. Ester, H.-P. Kriegel, et al. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2(2):169–194, 1998.
3 Y. Zhu, K. Ming Ting, M. J. Carman. Density-ratio based clustering for discovering clusters with varying densities. *Pattern Recognition*, 60:983–997, 2016.
4 A. Bryant, K. Cios. Rnn-dbscan: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1109–1121, 2018.
5 M. Ankerst, M. M. Breunig, H.-P. Kriegel, et al. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.
6 A. Rodriguez, A. Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.
7 M. Du, S. Ding, H. Jia. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Systems*, 99:135–145, 2016.
8 W. Guo, W. Wang, S. Zhao, et al. Density peak clustering with connectivity estimation. *Knowledge-Based Systems*, 243:108501, 2022.
9 S. R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992.
10 U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011. January 2011 Special Issue.
11 J. I. Cirac, D. Pérez-García, N. Schuch, et al. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Rev. Mod. Phys.*, 93:045003, Dec 2021.
12 W. Huggins, P. Patil, B. Mitchell, et al. Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, 4(2):024001, jan 2019.
13 I. Cong, S. Choi, M. D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.
14 E. Stoudenmire, D. J. Schwab. Supervised learning with tensor networks. *Advances in Neural Information Processing Systems*, 29, 2016.
15 Z.-Y. Han, J. Wang, H. Fan, et al. Unsupervised generative modeling using matrix product states. *Phys. Rev. X*, 8:031012, Jul 2018.
16 Z.-Z. Sun, C. Peng, D. Liu, et al. Generative tensor network classification model for supervised machine learning. *Phys. Rev. B*, 101:075135, Feb 2020.
17 Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.
18 Z.-F. Gao, S. Cheng, R.-Q. He, et al. Compressing deep neural networks by matrix product operators. *Phys. Rev. Research*, 2:023300, Jun 2020.
19 X. Shi, Y. Shang, C. Guo. Clustering using matrix product states. *Phys. Rev. A*, 105:052424, May 2022.

20 Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

21 Y. Ren, N. Wang, M. Li, et al. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020.

22 J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

23 R. Liu, H. Wang, X. Yu. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. *Information Sciences*, 450:200–226, 2018.

24 Z. Zhang, Q. Zhu, F. Zhu, et al. Density decay graph-based density peak clustering. *Knowledge-Based Systems*, 224:107075, 2021.

25 L. McInnes, J. Healy, J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

26 J. Xie, R. Girshick, A. Farhadi. Unsupervised deep embedding for clustering analysis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR.

27 X. Guo, L. Gao, X. Liu, et al. Improved deep embedded clustering with local structure preservation. In *Ijcai*, pages 1753–1759, 2017.

28 J. Yang, D. Parikh, D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5147–5156, 2016.

29 K. Ghasedi Dizaji, A. Herandi, C. Deng, et al. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

30 C. You, C.-G. Li, D. P. Robinson, et al. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3928–3937, 2016.

31 X. Chen, Y. Duan, R. Houthooft, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

32 S. Mukherjee, H. Asnani, E. Lin, et al. Clustergan: Latent space clustering in generative adversarial networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4610–4617, Jul. 2019.

33 X. Yang, C. Deng, F. Zheng, et al. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

34 X. Guo, E. Zhu, X. Liu, et al. Deep embedded clustering with data augmentation. In *Asian conference on machine learning*, pages 550–565. PMLR, 2018.

35 J. Eisert, M. Cramer, M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Rev. Mod. Phys.*, 82:277–306, Feb 2010.