Quantum-inspired algorithm for general minimum conical hull problems

Yuxuan Du,[1,*] Min-Hsiu Hsieh,[2,†] Tongliang Liu,[1,‡] and Dacheng Tao[1,§]

[1]*UBTECH Sydney AI Centre, School of Computer Science, Faculty of Engineering, The University of Sydney, Australia*

[2]*Centre for Quantum Software and Information, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia*

(Received 30 March 2020; accepted 15 July 2020; published 5 August 2020)

A wide range of fundamental machine learning tasks that are addressed by the maximum *a posteriori* estimation can be reduced to a general minimum conical hull problem. The best-known solution to tackle general minimum conical hull problems is the divide-and-conquer anchoring learning scheme (DCA), whose runtime complexity is polynomial in size. However, big data are pushing these polynomial algorithms to their performance limits. In this paper, we propose a sublinear classical algorithm to tackle general minimum conical hull problems when the input has stored in a sample-based low-overhead data structure. The algorithm's runtime complexity is polynomial in the rank and polylogarithmic in size. The proposed algorithm achieves the exponential speedup over DCA and, therefore, provides advantages for high-dimensional problems.

DOI: 10.1103/PhysRevResearch.2.033199

I. INTRODUCTION

Maximum *a posteriori* (MAP) estimation is a central problem in machine and statistical learning [1,2]. The general MAP problem has been proven to be NP hard [3]. Despite the hardness in the general case, there are two fundamental learning models, the matrix factorization and the latent variable model, that enable MAP problem to be solved in polynomial runtime under certain constraints [4–8]. The algorithms that have been developed for these learning models have been used extensively in machine learning with competitive performance, particularly on tasks such as subspace clustering, topic modeling, collaborative filtering, signal processing, structure prediction, feature engineering, motion segmentation, sequential data analysis, and recommender systems [5,7,9]. A recent study demonstrates that MAP problems addressed by matrix factorization and the latent variable models can be reduced to the general minimum conical hull problem [10]. In particular, the general minimum conical hull problem transforms problems resolved by these two learning models into a geometric problem, whose goal is to identify a set of extreme data points with the smallest cardinality in data set $\mathbf{Y}$ such that every data point in data set $\mathbf{X}$ can be expressed as a conical combination of the identified extreme data points (see Sec. II B). Unlike the matrix factorization and the latent variable models that their optimizations generally suffer from the local minima, a unique global solution is guaranteed for the general minimum conical hull problem [10]. Driven by the promise of a global solution and the broad applicability, it is imperative to seek algorithms that can efficiently resolve the general minimum conical hull problem with theoretical guarantees.

The divide-and-conquer anchoring (DCA) scheme is among the best currently known solutions for addressing general minimum conical hull problems. The idea is to identify all k extreme rays (i.e., k extreme data points) of a conical hull from a finite set of real data points with high probability [10]. The discovered extreme rays form the global solution for the problem with explainability and, thus, the scheme generalizes better than conventional algorithms, such as expectation-maximization [11]. DCA's strategy is to decompose the original problem into distinct subproblems. Specifically, the original conical hull problem is randomly projected on different low-dimensional hyperplanes to ease computation. Such a decomposition is guaranteed by the fact that the geometry of the original conical hull is partially preserved after a random projection. However, a weakness of DCA is that it requires a polynomial runtime complexity with respect to the size of the input. This complexity heavily limits DCA's use in many practical situations given the number of massive-scale data sets that are now ubiquitous [12]. Hence, more effective methods for solving general minimum conical hull problems are highly desired.

To address the above issue, we propose an efficient classical algorithm that tackles general minimum conical hull problems in polylogarithmic time with respect to the input size. Consider two data sets $\mathbf{X}$ and $\mathbf{Y}$ that have stored in a specific low-overhead data structure, i.e., a sampled-based data structure supports the length-square sampling operations [13]. Let the maximum rank, the Frobenius norm, and the condition number of the given two data sets be k, $\|\mathbf{H}\|_F$, and κ, respectively. We prove that the runtime complexity of the our algorithm is $\tilde{\mathcal{O}}(k^6\kappa^{12}\|\mathbf{H}\|_F^6/\epsilon^6)$ with the tolerable level of error ϵ. The achieved sublinear runtime complexity indicates

*yudu5543@uni.sydney.edu.au
†min-hsiu.hsieh@uts.edu.au
‡tongliang.liu@sydney.edu.au
§dacheng.tao@sydney.edu.au

that our algorithm has capability to benefit numerous learning tasks that can be mapped to the general minimum conical hull problem, e.g., the MAP problems addressed by latent variable models and matrix factorization.

Two core ingredients of the proposed algorithm are the "divide-and-conquer" strategy and the reformulation of the minimum conical hull problem as a sampling problem. We adopt the "divide-and-conquer" strategy to acquire a favorable property from DCA. In particular, all subproblems, i.e., the general minimum conical hull problems on different low-dimensional random hyperplanes, are independent of each other. Therefore, they can be processed in parallel. In addition, the total number of subproblems is only polynomially proportional to the rank of the given data set [10,14]. An immediate observation is that the efficiency of solving each subproblem governs the efficiency of tackling the general minimum conical hull problem. To this end, our algorithm converts each subproblem into an approximated sampling problem and obtains the solution in sublinear runtime complexity. Through advanced sampling techniques [13,15], the runtime complexity to prepare the approximated sampling distribution that corresponds to each subproblem is polylogarithmic in the size of input. To enable our algorithm has an end-to-end sublinear runtime, we propose a general heuristic postselection method to efficiently sample the solution from the approximated distribution, whose computation cost is also polylogarithmic in size.

Our work creates an intriguing aftermath for the *quantum machine learning* community [16]. The overarching goal of quantum machine learning is to develop quantum algorithms that quadratically or even exponentially reduce the runtime complexity of classical algorithms [16]. Numerous quantum machine learning algorithms with quantum speedups have been proposed since the early 2010s [17–22]. However, the polylogarithmic runtime complexity in our algorithm implies that a rich class of quantum machine learning algorithms do not, in fact, achieve these claimed quantum speedups. More specifically, if a quantum algorithm aims to solve a learning task that can be mapped to the general minimum conical hull problem, then its quantum speedup will collapse. As a result, we show that quantum speedups collapse for these quantum algorithms: recommendation system [23], matrix factorization [24], and clustering [25–27]. Furthermore, our algorithm, accompanied by other quantum-inspired algorithms [13,28,29], improves the threshold to devise quantum machine learning algorithms with provable runtime speedups. Recently, the quantum-inspired classical algorithms have been broadly employed as the criterion to justify the quantum advantages achieved by the proposed quantum machine learning algorithms, e.g., the proposals [30–32] have carefully compared their results with the corresponding quantum-inspired results.

Moreover, our proposal and other quantum-inspired algorithms also motivate the quantum machine learning community to explore other advanced quantum techniques that are difficult to be dequantized. A representative example is Ref. [33], which proposed a sublinear quantum algorithm for training linear and kernel-based classifiers. In light of the quantum-inspired classical machine learning algorithms, Ref. [33] points out that the exploitation of too powerful input data structures such as the quantum random access memory might render any finding of quantum speedup inconclusive. Driven by this observation, their algorithm leveraged a very mild quantum input model to earn the quadratic runtime speedups compared with its classical counterparts. The employed quantum input model guarantees that their proposal cannot be dequantized by the quantum-inspired algorithms. In summary, the quantum-inspired algorithms contribute to devising quantum machine learning algorithms with more solid quantum advantages.

A. Related work

We make the following comparisons with previous studies in maximum *a posteriori* estimation and quantum machine learning.

(i) The mainstream methods to tackle MAP problems prior to the study [10] can be separated into two groups. The first group includes expectation-maximization, sampling methods, and matrix factorization [11,34,35], where the learning procedure has severely suffered from local optima. The second group contains the method of moments [36], which has suffered from the large variance and may lead to the failure of final estimation. The study [10] effectively alleviates the difficulties encountered by the above two groups. However, a common weakness possessed by all existing methods is the polynomial runtime with respect to the input size.

(ii) There are several studies that collapse the quantum speedups by proposing quantum-inspired classical algorithms [15,37,38]. For example, the study in Ref. [13] removes the quantum speedup for recommendation systems; the study in Ref. [39] eliminates the quantum speedup for principal component analysis problems; and the study in Ref. [40] collapses the quantum speedup for support vector machine problem. The correctness of a branch of quantum-inspired algorithms is validated by the study in Ref. [41]. The studies in Refs. [13] and [40] can be treated as a special case of our result, since both recommendation systems and support vector machine can be efficiently reduced to the general conical hull problems [14,42]. In other words, our work is a more general methodology to collapse the speedups achieved by quantum machine learning algorithms.

The rest of this paper proceeds as follows. A formal outline of the general minimum conical hull problem is given in Sec. II. The runtime complexity of our algorithm is analyzed in Sec. III. Section IV discusses the algorithm's correctness. Numerical simulations are conducted in Sec. V. We conclude the paper in Sec. VI.

II. PROBLEM SETUP

A. Notations

Throughout this paper, we use the following notations. We denote $\{1, 2, \ldots, n\}$ as $[n]$. Given a vector $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{v}_i$ or $\mathbf{v}(i)$ represent the ith entry of $\mathbf{v}$ with $i \in [n]$ and $\|\mathbf{v}\|$ refers to the ℓ_2 norm of $\mathbf{v}$ with $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^n \mathbf{v}_i^2}$. The notation $\mathbf{e}_i$ always refers to the ith unit basis vector. Suppose that $\mathbf{v}$ is nonzero, we define $\mathcal{P}_\mathbf{v}$ as a probability distribution in which the index i of $\mathbf{v}$ will be chosen with the probability $\mathcal{P}_\mathbf{v}(i) = (|\mathbf{v}_i|/\|\mathbf{v}\|)^2$ for any $i \in [n]$. A sample from $\mathcal{P}_\mathbf{v}$ refers to an index number

i of $\mathbf{v}$, which will be sampled with the probability $\mathcal{P}_{\mathbf{v}}(i)$. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, $\mathbf{X}_{ij}$ represents the (i, j) entry of $\mathbf{X}$ with $i \in [n]$ and $j \in [m]$. $\mathbf{X}(i;:)$ and $\mathbf{X}(:;j)$ represent the ith row and the jth column of the matrix $\mathbf{X}$, respectively. The transpose of a matrix $\mathbf{X}$ (a vector $\mathbf{v}$) is denoted by $\mathbf{X}^\top$ ($\mathbf{v}^\top$). The Frobenius and spectral norm of $\mathbf{X}$ is denoted as $\|\mathbf{X}\|_F$ and $\|\mathbf{X}\|_2$, respectively. The condition number κ_X of a positive semidefinite $\mathbf{X}$ is $\kappa_X = \|\mathbf{X}\|_2/\sigma_{\min}(\mathbf{X})$, where $\sigma_{\min}(\mathbf{X})$ refers to the minimum singular of $\mathbf{X}$. $\mathbb{R}_+$ refers to the real positive numbers. Given two sets $\mathcal{A}$ and $\mathcal{B}$, we denote $\mathcal{A}$ minus $\mathcal{B}$ as $\mathcal{A} \setminus \mathcal{B}$. The cardinality of a set $\mathcal{A}$ is denoted as $|\mathcal{A}|$. We use the notation $\tilde{\mathcal{O}}(k)$ as shorthand for $\mathcal{O}(k\log(n))$.

B. General minimum conical hull problem

A cone is a nonempty convex set that is closed under the conical combination of its elements. Mathematically, given a set $\mathcal{R} = \{r_i\}_{i=1}^k$ with r_i being the ray, a cone formed by $\mathcal{R}$ is defined as $\text{cone}(\mathcal{R}) = \{\sum_{i=1}^k \alpha_i r_i : \forall i \in [k],\ r_i \in \mathcal{R},\ \alpha_i \in \mathbb{R}_+\}$, and $\text{cone}(\mathcal{R})$ is the conical hull of $\mathcal{R}$. A ray is a part of a line that has one endpoint that lies at the origin and goes on infinitely in only one direction. A ray r_i is an extreme ray (or an *anchor*) if it cannot be expressed as the conical combination of elements in $\mathcal{R} \setminus r_i$. A fundamental property of the cone and conical hull is its *separability*, namely whether a point in $\text{cone}(\mathcal{R})$ can be represented as a conical combination of certain subsets of rays that define $\text{cone}(\mathcal{R})$. The above separability can be generalized to the matrix form as follows [10].

Definition 1 (General separability condition [10]). Let $\mathbf{X} \in \mathbb{R}^{n_X\times m}$ be a matrix with n_X rows. Let $\mathbf{Y} \in \mathbb{R}^{n_Y\times m}$ be a matrix with n_Y rows, and let $\mathbf{Y}_\mathcal{A}$ be a submatrix of $\mathbf{Y}$ with row indexes $\mathcal{A} \subset [n_Y]$. We say that $\mathbf{X}$ is separable with respect to $\mathbf{Y}_\mathcal{A}$ if $\mathbf{X} = \mathbf{F}\mathbf{Y}_\mathcal{A}$, where there exists $\mathbf{F}$ satisfying $rank(\mathbf{F}) \geqslant |\mathcal{A}|$.

In other words, the general separability condition states that, $\forall i \in [n_X]$, we have $\mathbf{X}_i \in \text{cone}(\mathbf{Y}_\mathcal{A})$, where the set $\mathbf{Y}_\mathcal{A} = \{\mathbf{Y}(i;:)\}_{i\in\mathcal{A}}$. Under this definition, the general minimum conical hull problem aims to find the minimal set $\mathcal{A}$, as the so-called anchor set, from the rows of $\mathbf{Y}$.

Definition 2 (General minimum conical hull problem [10]). Given two matrices $\mathbf{X} \in \mathbb{R}^{n_X\times m}$ and $\mathbf{Y} \in \mathbb{R}^{n_Y\times m}$ with n_X and n_Y rows, respectively, the general minimum conical hull problem $\text{MCH}(\mathbf{Y}, \mathbf{X})$ finds a minimal subset of rows in $\mathbf{Y}$ whose conical hull contains $\text{cone}(\mathbf{X})$:

$$\text{MCH}(\mathbf{Y}, \mathbf{X}) := \arg\min\{|\mathcal{A}| : \text{cone}(\mathbf{Y}_\mathcal{A}) \supset \text{cone}(\mathbf{X})\}, \quad (1)$$

where $\text{cone}(\mathbf{Y}_\mathcal{A})$ is induced by rows $\mathcal{A}$ of $\mathbf{Y}$.

We remark that the general separability condition is reasonable for many learning tasks, i.e., any learning task can be solved by the matrix factorization model or the latent variable model possessing general separability [10].

For ease of understanding, we provide an intuitive example about how to treat the separable nonnegative matrix factorization as the general minimum conical hull problem (Further examples can be founded in Ref. [10]). Recall that a given data matrix $\mathbf{X}$ admits a separable nonnegative matrix factorization if it can be expressed as $\mathbf{X} = \mathbf{W}\mathbf{X}_\mathcal{A}$, where $\mathbf{W} \in \mathbb{R}^{n_X\times k}$ and $\mathbf{X}_\mathcal{A} \in \mathbb{R}^{k\times m}$ are two nonnegative matrices, k refers to the rank of $\mathbf{X}$ with $k \sim O[\text{poly}\log(n_X, m)]$, and $\mathbf{X}_\mathcal{A}$ is a submatrix of $\mathbf{X}$. From the view of the conical hull problem as formulated in Definition 2, we have $\mathbf{Y} := \mathbf{X}$ [43,44]. Geometrically, all rows of $\mathbf{X}$ reside in a cone generated by a small subset of rows of $\mathbf{X}$, i.e., $\text{cone}(\mathbf{X}) \subseteq \text{cone}(\mathbf{X}_\mathcal{A})$ with $|\mathcal{A}| = k$. The goal of the general minimum conical hull problem is finding the index set $\mathcal{A}$ that describes $\mathbf{X}_\mathcal{A}$.

C. Divide-and-conquer anchoring scheme for the general minimum conical hull problem

Efficiently locating the anchor set $\mathcal{A}$ of a general minimum conical hull problem is challenging. To resolve this issue, the DCA learning scheme is proposed [10,14]. This scheme adopts the following strategy. The original minimum conical hull problem is first reduced to a small number of subproblems by projecting the original cone into low-dimensional hyperplanes. Then these subproblems on low-dimensional hyperplanes are then tackled in parallel. The anchor set of the original problem can be obtained by combining the anchor sets of the subproblems because the geometric information of the original conical hull is partially preserved after projection. Moreover, the efficiency of DCA schemes can be guaranteed because anchors in these subproblems can be effectively determined in parallel and the total number of subproblems is modest.

DCA is composed of two steps, i.e., the divide step and the conquer step. Following the notations of Definition 2, given two sets of points (rows) with $\mathbf{X} \in \mathbb{R}^{n_X\times m}$ and $\mathbf{Y} \in \mathbb{R}^{n_Y\times m}$, the goal of DCA is to output an anchor set $\mathcal{A}$ such that $\mathbf{X}(i;:) \in \text{cone}(\mathbf{Y}_\mathcal{A})$ with $\mathcal{A} \subset [n_Y]$ and $|\mathcal{A}| = k$, $\forall i \in [n_X]$. Note that $k \sim O[\text{poly}\log(n_X, n_Y)]$ refers to the number of latent factors or variables. Following the result in Ref. [10], Table I summarizes the physical meaning of k in different models.

1. Divide step

A set of projection matrices $\{\mathbf{B}_t\}_{t=1}^p$ with $\mathbf{B}_t \in \mathbb{R}^{m\times d}$ is sampled from a random ensemble $\mathbb{M}$, e.g., Gaussian random matrix ensemble, the ensemble composed of standard unit vectors $\mathbf{e}_i$ or real data vectors, and various sparse random matrix ensembles [10]. The general minimum conical hull problem for the tth subproblem is to find an anchor set $\mathcal{A}_t$ for the projected matrices $\mathbf{Y}_t := \mathbf{Y}\mathbf{B}_t \in \mathbb{R}^{n_Y\times d}$ and $\mathbf{X}_t := \mathbf{X}\mathbf{B}_t \in \mathbb{R}^{n_X\times d}$:

$$\text{MCH}(\mathbf{Y}_t, \mathbf{X}_t) := \arg\min\{|\mathcal{A}_t| : \text{cone}(\mathbf{Y}_{\mathcal{A}_t}) \supset \text{cone}(\mathbf{X}_t)\}, \quad (2)$$

TABLE I. The abbreviations of NMF, SC, GMM, HMM, and LDA represent nonnegative matrix factorization, spectral clustering, Gaussian mixture models, hidden Markov models, and latent Dirichlet allocation, respectively.

Learning model	k in conical hull problem
NMF	No. of factors
SC	No. of basis from all clusters
GMM	No. of components or clusters
HMM	No. of hidden states
LDA	No. of topics

where $\mathbf{Y}_{\mathcal{A}_t}$ is the submatrix of $\mathbf{Y}$ whose rows are indexed by $\mathcal{A}_t \subset [n_Y]$. Attributed to the flexible mechanism of DCA, several studies [10,14,45] provide distinct strategies to solve Eq. (2) with varied projected dimension d.

2. Conquer step

This step yields the anchor set $\mathcal{A}$ by employing the following selection rule to manipulate the collected $\{\mathcal{A}_t\}_{t=1}^{p}$. First, we compute, $\forall i \in [n_Y]$,

$$\hat{g}_i := \frac{1}{p}\sum_{t=1}^{p} \mathbb{1}_{\mathcal{A}_t}[\mathbf{Y}(i;:)], \tag{3}$$

where $\mathbf{Y}(i;:)$ is the ith row of $\mathbf{Y}$, and $\mathbb{1}_{\mathcal{A}_t}[\mathbf{Y}(i;:)]$ is the indicator function that outputs "1" when the index i is in $\mathcal{A}_t$, and zero otherwise. The anchor set $\mathcal{A}$ of size k is constructed by selecting k indexes with the largest $\hat{g}_i$.

The following proposition demonstrates the theoretical guarantee that employs DCA to solve the general minimum conical hull problem.

Proposition 3 (Corollary 1 [10]). With probability $1-\delta$, DCA can recover the anchor set $\mathcal{A}$ by solving $p = \Omega(k\log(k/\delta))$ subproblems, where $k = |\mathcal{A}|$.

The total runtime complexity of DCA is $\tilde{\mathcal{O}}(\max\{\text{poly}(n_X), \text{poly}(n_Y)\})$ when parallel processing is allowed [10].

D. Reformulation as a sampling problem

Interestingly, following the result in Ref. [45], each subproblem in Eq. (2) can be reduced to a sampling problem when $d = 1$. This observation is one of the crucial components that make our algorithm exponentially faster than the conventional DCA schemes [10].

Fix $d = 1$. $\mathcal{A}_t$ defined in Eq. (2) becomes

$$\mathcal{A}_t := \left\{\arg\min_{i\in[n_Y]} \left[\mathbf{Y}_t(i;:) - \max_{j\in[n_X]} \mathbf{X}_t(j;:)\right]_+\right\}, \tag{4}$$

where $\mathbf{Y}_t \in \mathbb{R}^{n_Y\times 1}$, $\mathbf{X}_t \in \mathbb{R}^{n_X\times 1}$, $(x)_+ = x$ if $x \geqslant 0$ and ∞ otherwise. We give an intuitive explanation of Eq. (4) in Fig. 1.

Note that, $\forall t \in [p]$, Eq. (4) can then be written as

$$\mathcal{A}_t := \left\{\arg\min_{i\in[n_Y]} \left[\mathcal{P}_{\mathbf{Y}_t}(i) - \xi_t \max_{j\in[n_X]} [\mathcal{P}_{\mathbf{X}_t}(j)]\right]_+\right\}, \tag{5}$$

where $\mathcal{P}_{\mathbf{X}_t}$ and $\mathcal{P}_{\mathbf{Y}_t}$ refer to the distributions of $\mathbf{X}_t$ and $\mathbf{Y}_t$ and $\xi_t = \|\mathbf{X}_t\|/\|\mathbf{Y}_t\|$ is a constant at tth random projection to rescale the value $\max_{j\in[n_X]}[\mathcal{P}_{\mathbf{X}_t}(j)]$. We will explain how to efficiently approximate ξ_t in Sec. III C.

III. MAIN ALGORITHM

The core of our algorithm consists of the following major steps. In the preprocessing step, we reconstruct two matrices, $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$, to approximate the original matrices $\mathbf{X}$ and $\mathbf{Y}$ so that the tth projected subproblem can be replaced with $\tilde{\mathbf{X}}_t = \tilde{\mathbf{X}}\mathbf{B}_t$ and $\tilde{\mathbf{Y}}_t = \tilde{\mathbf{Y}}\mathbf{B}_t$ with little disturbance. The mathematical representation of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ is $\tilde{\mathbf{X}} = \tilde{\mathbf{V}}_X\tilde{\mathbf{V}}_X^\top\mathbf{X}$ and $\tilde{\mathbf{Y}} = \tilde{\mathbf{V}}_Y\tilde{\mathbf{V}}_Y^\top\mathbf{Y}$, respectively, where $\tilde{\mathbf{V}}$ approximates the left singular matrix $\mathbf{V}$ of the given input with $\|\mathbf{V}-\tilde{\mathbf{V}}\| \leqslant \epsilon$. The motivation to rebuild the low-rank approximations of input is to exploit the advanced sampling techniques [13] to approximate the projected result $\tilde{\mathbf{X}}_t$ and $\tilde{\mathbf{Y}}_t$ in polylogarithmic runtime with respect to the input size. Considering that the precondition to use such sampling methods is recasting the given task to a sampling task with the support of the square-length sampling operations [13], we adopt the sampling version of the general minimum conical hull problem defined in Eq. (5). The divide step employs two sampling subroutines that allow us to efficiently approximate two distributions $\mathcal{P}_{\tilde{\mathbf{X}}_t}$ and $\mathcal{P}_{\tilde{\mathbf{Y}}_t}$ in polylogarithmic runtime with respect to the input size. All subproblems are processed in parallel, following the divide-and-conquer principle. We then propose the general heuristic postselection rule to identify the target index $\mathcal{A}_t$ by substituting $\{\mathcal{P}_{\mathbf{Y}_t}(i) - \xi_t \max_{j\in[n_X]}[\mathcal{P}_{\mathbf{X}_t}(j)]_+\}$ in Eq. (5) with $\{\mathcal{P}_{\tilde{\mathbf{Y}}_t}(i) - \xi_t \max_{j\in[n_X]}[\mathcal{P}_{\tilde{\mathbf{X}}_t}(j)]_+\}$. Last, we employ the selection rule in Eq. (3) to form the anchor set $\mathcal{A}$ for the original minimum conical hull problem.

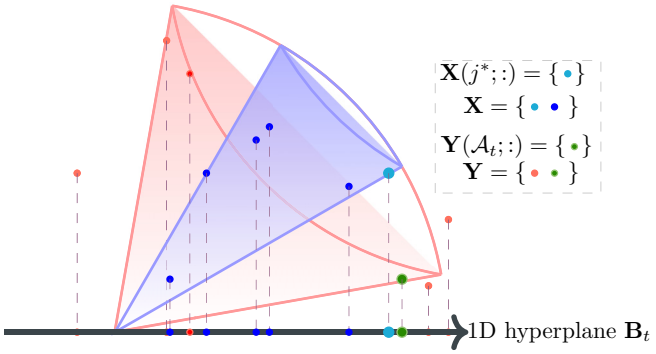

FIG. 1. The projection on a one-dimensional hyperplane. we show how to obtain the anchor $\mathcal{A}_t$ as the solution in Eq. (4). The data set $\mathbf{Y}$ is composed of red and green nodes. The data set $\mathbf{X}$ is a cone that is composed of blue and light blue nodes. $\mathbf{B}_t$ represents a random one-dimensional projection hyperplane. On hyperplane $\mathbf{B}_t$, the light blue node $\mathbf{X}(j^*;:)$ refers to the result of $\max_{j\in[n_X]}\mathbf{X}_t(j;:)$ in Eq. (4). The green node $\mathbf{Y}(\mathcal{A}_t;:)$ corresponds to the solution of Eq. (4), where $\mathcal{A}_t$ refers to the anchor at the tth random projection. Geometrically, on the hyperplane $\mathbf{B}_t$, the green node is the nearest node among all nodes in $\mathbf{Y}$ relative to the light blue node and has a larger magnitude.

Before elaborating on the details of the main algorithm, we emphasize the innovations of this work, i.e., the reformulation of the general conical hull problem as a sampling problem and the general heuristic postselection rule. The sampling version of the general conical hull problem is the precondition to introduce advanced sampling techniques to reduce the computational complexity. The general heuristic postselection rule is the central component to guarantee that the solution of the general conical hull problem can be obtained in sublinear runtime. In particular, the intrinsic mechanism of the general conical hull problem enables us to employ the general heuristic postselection rule to query a specific element from the output in polylogarithmic runtime.

In this section, we introduce the length-square sampling operations in Sec. III A. Two sampling subroutines developed in Ref. [13] are introduced in Sec. III B. The implementation of our algorithm is shown in Sec. III C. The computation analysis is given in Sec. III D. We give the proof of Theorems 8 and 9 in Appendices B and D, respectively.

A. Length-square sampling operations

The promising performance of various sampling algorithms for machine learning tasks is guaranteed when the given data set supports length-square sampling operations [13,15,38]. We first give the definition of the access cost to facilitate the description of such sampling operations,

Definition 4 (Access cost). Given a matrix $\mathbf{W} \in \mathbb{R}^{n\times m}$, we denote that the cost of querying an entry $\mathbf{W}(i;j)$ or querying the Frobenius norm $\|\mathbf{W}\|_F$ is $Q(\mathbf{W})$, the cost of querying the ℓ_2 norm of $\|\mathbf{W}(i;:)\|$ is $N(\mathbf{W})$, and the cost of sampling a row index $i \in [n]$ of $\mathbf{H}$ with the probability $\mathcal{P}_{\mathbf{W}}(i) = \|\mathbf{W}(i;:)\|^2/\|\mathbf{W}\|_F^2$ or sampling an index $j \in [m]$ with the probability $\mathcal{P}_{\mathbf{W}(i;:)}(j)$ is $S(\mathbf{W})$. We denote the overall access cost of $\mathbf{W}$ as $L(\mathbf{W}) := S(\mathbf{W}) + Q(\mathbf{W}) + N(\mathbf{W})$.

We use an example to address the difference between query access and sampling access. Given a vector $\mathbf{v} \in \mathbb{R}^n$, the problem is to find a hidden large entry $\mathbf{v}(i)$. It takes $\Omega(n)$ cost to find $\mathbf{v}(i)$ with just query access, while the computation cost is $\mathcal{O}(1)$ with query and sample access.

The length-square sampling operations, *as so-called ℓ_2 norm sampling operations*, are defined as:

Proposition 5 (ℓ_2 norm sampling operation). Given an input matrix $\mathbf{H} \in \mathbb{R}^{n\times m}$ with s nonzero entries, there exists a data structure storing $\mathbf{H}$ in space $\mathcal{O}(s\log^2 m)$, with the following properties:

(i) The query cost $Q(\mathbf{H})$ is at most $\mathcal{O}(\log(nm))$;

(ii) The query cost $N(\mathbf{H})$ is at most $\mathcal{O}(\log^2(n))$;

(iii) The sampling cost $S(\mathbf{H})$ is at most $\mathcal{O}(\log^2(nm))$.

The overall cost of accessing $\mathbf{H}$ is therefore $L(\mathbf{H}) = \mathcal{O}(poly[\log(mn)])$.

Remark. The ℓ_2 norm sampling operation can be efficiently fulfilled if the input data are stored in a low-overhead data structure, e.g., the binary tree structure (BNS) [23] (more details about BNS are given in Appendix A).

B. Two sampling subroutines

Here we introduce two sampling subroutines, the inner product subroutine and the thin matrix-vector multiplication subroutine [13], used in the proposed algorithm.

1. Inner product subroutine

In our algorithm, the inner product subroutine is employed to obtain each entry of $\hat{\mathbf{q}}_t$ in parallel, i.e., $\hat{\mathbf{q}}_t(i)$ estimates $\tilde{\mathbf{q}}_t(i) \equiv \tilde{\mathbf{v}}^{(i)\top}\mathbf{H}_t$, where $\tilde{\mathbf{v}}^{(i)} \in \mathbb{R}^{n\times 1}$, $\mathbf{H}_t \equiv \mathbf{H}\mathbf{B}_t$, $\mathbf{H} \in \mathbb{R}^{n\times m}$, and $\mathbf{B}_t \in \mathbb{R}^{m\times 1}$. Let Z be a random variable that, for $j \in [n]$, $l \in [m]$, takes value

$$Z(j,l) = \frac{\|\tilde{\mathbf{v}}^{(i)}\|\|\mathbf{B}_t\|\mathbf{H}(j,l)}{\tilde{\mathbf{v}}^{(i)}(j)\mathbf{B}_t(l)}$$

with probability

$$\mathcal{P}_Z(Z(j,l)) = \frac{|\tilde{\mathbf{v}}^{(i)}(j)|^2|\mathbf{B}_t(l)|^2}{\|\tilde{\mathbf{v}}^{(i)}\|^2\|\mathbf{B}_t\|^2}. \tag{6}$$

We can estimate $\tilde{\mathbf{q}}_t(i)$ using Z as follows [38]. Fix $\epsilon, \delta > 0$. Let

$$N_Z = N_p \times N_q \sim \mathcal{O}\left(\frac{\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right). \tag{7}$$

We first sample the distribution $\mathcal{P}_Z$ with N_Z times to obtain a set of samples $\{z_i\}_{i=1}^{N_Z}$, followed by dividing them into N_p groups, $\{S_1,\ldots,S_{N_p}\}$, where each group S_i contains N_q samples. Let $\bar{z}_i = \sum_{i=1}^{N_q} z_i/N_q$ be the empirical mean of the ith group S_i, and let $\bar{z}^*$ be the median of $\{\bar{z}_i\}_{i=1}^{N_p}$. Then Ref. [15, Lemma 12] and Ref. [38, Lemma 7] guarantee that, with probability at least $1-\delta$, the following holds:

$$|\bar{z}^* - \tilde{\mathbf{q}}_t(i)| \leqslant \epsilon. \tag{8}$$

The computational complexity of the inner product subroutine is as follows:

Lemma 6 (Ref. [15, Lemma 12] and Ref. [38, Lemma 7]). Assume that the overall access to $\mathbf{H}$ is $L(\mathbf{H})$ and the query access to $\mathbf{B}_t$ and $\tilde{\mathbf{V}}$ is $Q(\mathbf{B}_t)$ and $Q(\tilde{\mathbf{V}})$, respectively. The runtime complexity to yield Eq. (8) is

$$\mathcal{O}\left(\frac{\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{\epsilon^2}[L(\mathbf{H}) + Q(\mathbf{B}_t) + Q(\tilde{\mathbf{V}})]\log\left(\frac{1}{\delta}\right)\right).$$

Proof. We first recall the main result of Lemma 12 in Ref. [15]. Given the overall access to $\mathbf{H}$ and query access to the matrix $\tilde{\mathbf{V}}$ and $\mathbf{B}_t$ with complexity $Q(\tilde{\mathbf{V}})$ and $Q(\mathbf{B}_t)$, the inner product $\tilde{\mathbf{q}}_t(i) \equiv \tilde{\mathbf{v}}^{(i)\top}\mathbf{H}_t$ can be estimated to precision $\epsilon\|\mathbf{H}\|_F\|\tilde{\mathbf{v}}^{(i)}\|\|\mathbf{B}_t\|_F$ with probability at least $1-\delta$ in time

$$\mathcal{O}\left(\frac{[L(\mathbf{H}) + Q(\mathbf{B}_t) + Q(\tilde{\mathbf{V}})]}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

With setting the precision to ϵ instead of $\epsilon\|\mathbf{H}\|_F\|\tilde{\mathbf{v}}^{(i)}\|\mathbf{B}_t\|_F$, it can be easily inferred that the runtime complexity to estimate $\tilde{\mathbf{q}}_t(i) \equiv \tilde{\mathbf{v}}^{(i)\top}\mathbf{H}_t$ with probability at least $1-\delta$ is

$$\mathcal{O}\left(\frac{\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{\epsilon^2}[L(\mathbf{H}) + Q(\mathbf{B}_t) + Q(\tilde{\mathbf{V}})]\log\left(\frac{1}{\delta}\right)\right). \quad \blacksquare$$

2. Thin matrix-vector multiplication subroutine

Given a matrix $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ and $\hat{\mathbf{q}}_t \in \mathbb{R}^k$ with the ℓ_2 norm sampling access, the thin matrix-vector multiplication subroutine aims to output a sample from $\mathcal{P}_{\tilde{\mathbf{V}}\hat{\mathbf{q}}_t}$. The implementation of the thin matrix-vector multiplication subroutine is as follows [15].

For each loop

(i) Sample a column index $j \in [k]$ uniformly.

(ii) Sample a row index $l \in [n]$ from distribution $\mathcal{P}_{\tilde{\mathbf{V}}^\top(:;j)}(l) = \frac{|\tilde{\mathbf{V}}^\top(l;j)|^2}{\|\tilde{\mathbf{V}}^\top(:;j)\|^2}$.

(iii) Compute $|\tilde{\mathbf{V}}(l;:)\hat{\mathbf{q}}_t|^2$.

(iv) Output l with probability $\frac{|\tilde{\mathbf{V}}(l;:)\hat{\mathbf{q}}_t|^2}{\|\tilde{\mathbf{V}}(l;:)\|^2\|\hat{\mathbf{q}}_t\|^2}$ or restart to sample (j,l) again (back to step 1).

We execute the above loop until a sample l is successfully output.

The complexity of the thin matrix-vector multiplication subroutine, namely the required number of loops, is as follows:

Algorithm 1 Subsampling method [13]

Data: $\mathbf{H} \in \mathbb{R}^{n\times m}$ with supporting ℓ_2 norm sampling operations, parameters s, ϵ, κ.
Result: The singular value decomposition of $\mathbf{C}$.
1 Independently sample s columns indices $[i_s]$ according to the probability distribution $\mathcal{P}_{\mathbf{H}}$;
2 Set $\mathbf{R} \in \mathbb{R}^{n\times s}$ as the matrix formed by $\mathbf{H}(:, i_t)/\sqrt{s\mathcal{P}_{\mathbf{H}(:,i_t)}}$ with $i_t \in [i_s]$;
3 Sample a column index t with $t \in [s]$ uniformly and then sample a row index $j \in [n]$ distributed as $\mathcal{P}_{\mathbf{R}(j;t)}$. Sample a total number of s row indexes $[j_s]$ in this way;
4 Let $\mathbf{C} \in \mathbb{R}^{s\times s}$ be a matrix whose tth row is $\mathbf{C}(t;:) = \mathbf{R}(j_t;:)/\sqrt{s\mathcal{P}_{\mathbf{R}(j_t;:)}}$.
5 Apply SVD to obtain right singular vector $\{\omega^{(i)}\}_{i=1}^k$ and singular values $\{\sigma^{(i)}\}_{i=1}^k$ of $\mathbf{C}$;
6 Output the decomposition results $\{\sigma^{(i)}, \omega^{(i)}\}_{i=1}^k$ of $\mathbf{C}$.

Lemma 7 (Ref. [15, Lemma 12] and Ref. [13, Proposition 6.4]). Let $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ and $\hat{\mathbf{q}}_t \in \mathbb{R}^k$. Given ℓ_2 norm sampling access to $\tilde{\mathbf{V}}$, we can length-square sample from $\mathcal{P}_{\tilde{\mathbf{V}}\hat{\mathbf{q}}_t}$ in the expected runtime complexity

$$\mathcal{O}\left(\frac{k\|\hat{\mathbf{q}}_t\|^2}{\|\tilde{\mathbf{V}}\hat{\mathbf{q}}_t\|^2}[S(\tilde{\mathbf{V}}) + kQ(\tilde{\mathbf{V}})]\right).$$

C. The implementation of the algorithm

Our algorithm consists of three steps, the preprocessing step, the divide step, and the conquer step. The first step prepares an efficient description for $\mathbf{X}$ and $\mathbf{Y}$. The second step locates anchors $\{\mathcal{A}_t\}_{t=1}^p$ by solving p subproblems in parallel. The last step obtains the anchor set $\mathcal{A}$.

1. Preprocessing step

The preprocessing step aims to efficiently construct $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ such that the matrix norm $\|\tilde{\mathbf{X}} - \mathbf{X}\|_2$ and $\|\tilde{\mathbf{Y}} - \mathbf{Y}\|_2$ are small. This step employs the subsampling method [13] to construct an approximated left singular matrix $\tilde{\mathbf{V}}_H$ of $\mathbf{H}$, where $\mathbf{H} \in \mathbb{R}^{n_H\times m_H}$ can be either $\mathbf{X}$ or $\mathbf{Y}$, so that $\tilde{\mathbf{H}} = \tilde{\mathbf{V}}_H \tilde{\mathbf{V}}_H{}^\top \mathbf{H}$. If no confusion arises, then the subscript $\mathbf{H}$ can be disregarded.

We summarize the subsampling method inAlgorithm 1 to detail the acquisition of $\tilde{\mathbf{V}}$. We build the matrix $\mathbf{R} \in \mathbb{R}^{n\times s}$ by sampling s columns from $\mathbf{H}$ and then build the matrix $\mathbf{C} \in \mathbb{R}^{s\times s}$ by sampling s rows from $\mathbf{R}$. After obtaining $\mathbf{R}$ and $\mathbf{C}$, we implicitly define the approximated left singular matrix $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ as

$$\tilde{\mathbf{V}}(:; i) \equiv \tilde{\mathbf{v}}^{(i)} := \frac{\mathbf{R}\omega^{(i)}}{\sigma^{(i)}}, \tag{9}$$

where $\{\sigma^{(i)}\}_{i=1}^k$ and $\{\omega^{(i)}\}_{i=1}^k$ refer to k singular values and right singular vectors of $\mathbf{C}$.[1]

2. Divide step

The obtained approximated left singular matrices $\tilde{\mathbf{V}}_X \in \mathbb{R}^{n_X\times k_X}$ and $\tilde{\mathbf{V}}_Y \in \mathbb{R}^{n_Y\times k_Y}$ enable us to employ advanced sampling techniques to locate potential anchors $\{\mathcal{A}_t\}_{t=1}^p$. Here we only focus on locating the anchor $\mathcal{A}_t$ for the tth subproblem, since each subproblem is independent and can be solved in the same way. The divide step employs Eq. (5) to locate $\mathcal{A}_t$. In particular, we first prepare two distributions $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ to approximate $\mathcal{P}_{\mathbf{X}_t}$ and $\mathcal{P}_{\mathbf{Y}_t}$ and then sample these two distributions to locate $\mathcal{A}_t$.

The preparation of two distributions $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ is achieved by exploiting two sampling subroutines, the inner product subroutine and the thin matrix-vector multiplication subroutine [13], as introduced in Sec. III B. Recall that the two approximated matrices at the tth subproblem are $\tilde{\mathbf{X}}_t = \tilde{\mathbf{V}}_X(\tilde{\mathbf{V}}_X^\top \mathbf{X}_t)$ and $\tilde{\mathbf{Y}}_t = \tilde{\mathbf{V}}_Y(\tilde{\mathbf{V}}_Y^\top \mathbf{Y}_t)$. Denote $\tilde{\mathbf{q}}_{X,t} \equiv \tilde{\mathbf{V}}_X^\top \mathbf{X}_t \in \mathbb{R}^{k_X\times 1}$ and $\tilde{\mathbf{q}}_{Y,t} \equiv \tilde{\mathbf{V}}_Y^\top \mathbf{Y}_t \in \mathbb{R}^{k_Y\times 1}$. Instead of directly computing $\tilde{\mathbf{q}}_{X,t}$ and $\tilde{\mathbf{q}}_{Y,t}$, we construct their approximated vectors $\hat{\mathbf{q}}_{X,t}$ and $\hat{\mathbf{q}}_{Y,t}$ using the inner product subroutine to ensure the low computational cost, followed by the thin matrix-vector multiplication subroutine to prepare probability distributions $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$, where $\hat{\mathbf{X}}_t \equiv \tilde{\mathbf{V}}_X \hat{\mathbf{q}}_{X,t}$ and $\hat{\mathbf{Y}}_t \equiv \tilde{\mathbf{V}}_Y \hat{\mathbf{q}}_{Y,t}$. The closeness between $\mathcal{P}_{\hat{\mathbf{X}}_t}$ (respectively, $\mathcal{P}_{\hat{\mathbf{Y}}_t}$) and $\mathcal{P}_{\mathbf{X}_t}$ (respectively, $\mathcal{P}_{\mathbf{Y}_t}$) is controlled by the number of samplings s, as analyzed in Sec. IV.

The rescale parameter ξ_t defined in Eq. (5) can be efficiently approximated by employing the inner product subroutine. Recall that $\xi_t = \|\mathbf{Y}_t\|/\|\mathbf{X}_t\|$. We can approximate ξ_t by $\hat{\xi}_t = \|\hat{\mathbf{Y}}_t\|/\|\hat{\mathbf{X}}_t\|$. Alternatively, an efficient method of approximating $\|\hat{\mathbf{Y}}_t\|$ and $\|\hat{\mathbf{X}}_t\|$ is sufficient to acquire ξ_t. Let $\mathbf{H}$ be the general setting that can either be $\mathbf{X}$ or $\mathbf{Y}$. The ℓ_2 norm of $\|\hat{\mathbf{H}}_t\|$ can be efficiently estimated by using the inner product subroutine. Intuitively, the ℓ_2 norm of $\|\hat{\mathbf{H}}_t\|$ can be expressed by the inner product of $\hat{\mathbf{H}}_t$, i.e., $\|\hat{\mathbf{H}}_t\|^2 = \hat{\mathbf{H}}_t^\top \hat{\mathbf{H}}_t$. Recall that $\hat{\mathbf{H}}_t$ has an explicit representation $\hat{\mathbf{H}}_t = \tilde{\mathbf{V}}_H \hat{\mathbf{q}}_{H,t}$, and the efficient access cost for $\tilde{\mathbf{V}}_H$ and $\tilde{\mathbf{q}}_{H,t}$ enables us to use the inner product subroutine to efficiently obtain $\|\tilde{\mathbf{H}}_t\|$.

Given $\mathcal{P}_{\hat{\mathbf{X}}_t}$, $\mathcal{P}_{\hat{\mathbf{Y}}_t}$, and $\hat{\xi}_t$, we propose the general heuristic postselection method to determine $\mathcal{A}_t$. Following the sampling version of the general minimum conical hull problem defined in Eq. (5), we first sample the distribution $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with N_X times to obtain a value $C^*_{\hat{\mathbf{X}}_t}$ such that $C^*_{\hat{\mathbf{X}}_t} = \max_{j\in[n_X]}[\mathcal{P}_{\hat{\mathbf{X}}_t}(j)]$. We next sample the distribution $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with N_Y times to find an index $\hat{\mathcal{A}}_t$ approximating $\mathcal{A}_t = \arg\min_{i\in[n_Y]}(\mathcal{P}_{\hat{\mathbf{Y}}_t}(i) - \hat{\xi}_t C^*_{\hat{\mathbf{X}}_t})_+$. The following theorem quantifies the required number of samplings to guarantee $\hat{\mathcal{A}}_t = \mathcal{A}_t$, where $\mathcal{A}_t$ is defined in Eq. (5).

Theorem 8 (General heuristic postselection (Informal)). Assume that $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ are multinomial distributions. Denote that $C^*_{\hat{\mathbf{X}}_t} \geqslant \varepsilon_T$. If $|\mathcal{P}_{\hat{\mathbf{X}}_t}(i) - \mathcal{P}_{\hat{\mathbf{X}}_t}(j)| > \varepsilon$ for $C^*_{\hat{\mathbf{X}}_t} = \mathcal{P}_{\hat{\mathbf{X}}_t}(i)$ and $\forall j \neq i$, and $|\mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{A}_t) - \mathcal{P}_{\hat{\mathbf{Y}}_t}(j)| > \varepsilon$ for $\forall j \neq \mathcal{A}_t$ and a small constant ε, then for any $\delta > 0$ with a probability at least $1 - \delta$, we have $\hat{\mathcal{A}}_t = \mathcal{A}_t$ with $N_X, N_Y \sim \mathcal{O}(\kappa_X, \text{polylog}[\max\{n_X, n_Y\}])$.

[1] The "implicitly define $\tilde{\mathbf{V}}$" means that only the index array of $\mathbf{R}$ and $\mathbf{C}$, and the SVD result of $\mathbf{C}$ are required to be stored in the memory.

Algorithm 2 A sublinear runtime algorithm for the general minimum conical hull problem

Data: Given $\mathbf{X} \in \mathbb{R}^{n_X \times m}$, $\mathbf{Y} \in \mathbb{R}^{n_Y \times m}$, and $\{\mathbf{B}_t\}_{t=1}^{p}$ with $\mathbf{B}_t \in \mathbb{R}^{m\times 1}$. $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{B}_t$ support ℓ_2 sampling operations. The number of anchors k.

Result: Output the set of anchors $\mathcal{A}$ with $|\mathcal{A}| = k$.

1 *Preprocessing step:*
2 Set $s_X \leftarrow \frac{85^2 k_X^2 \kappa_X^4 \ln(8n_X/\eta)\|\mathbf{X}\|_F^2}{9\epsilon^2}$ and $s_Y \leftarrow \frac{85^2 k_Y^2 \kappa_Y^4 \ln(8n_Y/\eta)\|\mathbf{Y}\|_F^2}{9\epsilon^2}$ (see Theorem 19);
3 Sample $\mathbf{R}_X \in \mathbb{R}^{n_X \times s_X}$ from $\mathbf{X}$ and sample $\mathbf{C}_X \in \mathbb{R}^{s_X \times s_X}$ from $\mathbf{R}_X$ using Algorithm 1;
4 Sample $\mathbf{R}_Y \in \mathbb{R}^{n_Y \times s_Y}$ from $\mathbf{Y}$ and sample $\mathbf{C}_Y \in \mathbb{R}^{s_Y \times s_Y}$ from $\mathbf{R}_Y$ using Algorithm 1;
5 Implicitly define $\tilde{\mathbf{V}}_X$ and $\tilde{\mathbf{V}}_Y$ by employing Eq. (9);
6 (*Divide Step*) Set $t = 0$;
7 **while** $t < p$ (Computing in parallel.) **do**
8 Estimate $\tilde{\mathbf{q}}_X^t = \tilde{\mathbf{V}}_X^\top \mathbf{X}_t$ and $\tilde{\mathbf{q}}_Y^t = \tilde{\mathbf{V}}_Y^\top \mathbf{Y}_t$ by $\hat{\mathbf{q}}_X^t$ and $\hat{\mathbf{q}}_Y^t$ via inner product subroutine;
9 Prepare $\mathcal{P}_{\hat{\mathbf{X}}_t}$ ($\mathcal{P}_{\hat{\mathbf{Y}}_t}$) with $\hat{\mathbf{X}}_t = \tilde{\mathbf{V}}_X \hat{\mathbf{q}}_X^t$ ($\hat{\mathbf{Y}}_t = \tilde{\mathbf{V}}_Y \hat{\mathbf{q}}_Y^t$) via matrix-vector multiplication subroutine;
10 Collect $\hat{\mathcal{A}}^t$ by sampling $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ via the general heuristic postselection method;
11 $t \leftarrow t + 1$;
12 **end**
13 (*Conquer Step*) Output the anchor set $\mathcal{A}$ using the selection rule defined in Eq. (3);

3. Conquer step

After the divide step, a set of potential anchors $\{\mathcal{A}_t\}_{t=1}^{p}$ with $p = \mathcal{O}[k\log(k)]$, where k refers to the number of anchors with $k \sim \min\{\log(n_X), \log(n_Y)\}$, is collected [10,24,45]. We adopt the selection rule defined in Eq. (3) to determine the anchor set $\mathcal{A}$. This step can be accomplished by various sorting algorithms with runtime $\mathcal{O}\{\text{poly}[k\log(k)]\}$ [10].

We summarize our algorithm as follows.

D. Computation complexity of the algorithm

The complexity of the proposed algorithm is dominated by the preprocessing step and the divide step. Specifically, the computational complexity is occupied by four elements: finding the left singular matrix $\tilde{\mathbf{V}}$ (Line 5 in Algorithm 2), estimating $\tilde{\mathbf{q}}_t$ by $\hat{\mathbf{q}}_t$ (Line 7 in Algorithm 2), preparing the approximated probability distribution $\mathcal{P}_{\hat{\mathbf{H}}}$ (Line 8 in Algorithm 2), and estimating the rescale factor $\hat{\xi}_t$ (Line 10 in Algorithm 2). We evaluate the computation complexity of these four operations separately and obtain the following theorem.

Theorem 9. Given two data sets $\mathbf{X} \in \mathbb{R}^{n_X \times m}$ and $\mathbf{Y} \in \mathbb{R}^{n_Y \times m}$ that satisfy the general separability condition and support the length-square sampling operations, the rank and condition number for $\mathbf{X}$ (respectively, $\mathbf{Y}$) are denoted as k_X (respectively, k_Y) and κ_X (respectively, κ_Y). The tolerable error is denoted as ϵ. The runtime complexity of the proposed algorithm for solving the general minimum conical hull problem is

$$\max\left(\tilde{\mathcal{O}}\left(\frac{85^6 k_X^6 \kappa_X^{12}\|\mathbf{X}\|_F^6}{9^3\epsilon^6}\right), \tilde{\mathcal{O}}\left(\frac{85^6 k_{Y,t}^6 \kappa_Y^{12}\|\mathbf{Y}\|_F^6}{9^3\epsilon^6}\right)\right).$$

Recall that the input data sets $\mathbf{X}$ and $\mathbf{Y}$ of the general minimum conical hull problem are low rank, where $k_X, k_Y \sim O[\text{poly}\log(n_X, n_Y)]$. Following Theorem 9 and the polylogarithmic dependence between the rank and the input dimensions, the proposed algorithm achieves the exponential speedup in terms of n_X, n_Y, and m.

IV. CORRECTNESS OF THE ALGORITHM

In this section, we present the correctness of our algorithm. We also briefly explain how the results are derived and validate our algorithm by numerical simulations. We provide the detailed proofs of Theorem 10 in the Appendix C.

The correctness of our algorithm is determined by the closeness between the approximated distribution and the analytic distribution. The closeness is evaluated by the total variation distance [13], i.e., Given two vectors $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{w} \in \mathbb{R}^n$, the total variation distance of two distributions $\mathcal{P}_\mathbf{v}$ and $\mathcal{P}_\mathbf{w}$ is $\|\mathcal{P}_\mathbf{v}, \mathcal{P}_\mathbf{w}\|_{\text{TV}} := \frac{1}{2}\sum_{i=1}^{n} |\mathcal{P}_\mathbf{v}(i) - \mathcal{P}_\mathbf{w}(i)|$. The following theorem states that $\|\mathcal{P}_{\hat{\mathbf{X}}_t} - \mathcal{P}_{\mathbf{X}_t}\|_{\text{TV}}$ and $\|\mathcal{P}_{\hat{\mathbf{Y}}_t} - \mathcal{P}_{\mathbf{Y}_t}\|_{\text{TV}}$ are controlled by the number of samplings s:

Theorem 10. Given a matrix $\mathbf{H} \in \mathbb{R}^{n\times m}$ with ℓ_2 norm sampling operations, let $\mathbf{R} \in \mathbb{R}^{n\times s}$, $\mathbf{C} \in \mathbb{R}^{s\times s}$ be the sampled matrices from $\mathbf{H}$ following Algorithm 1. The distribution prepared by the proposed algorithm is denoted as $\mathcal{P}_{\hat{\mathbf{H}}_t}$. Set

$$s = \frac{85^2 k^2 \kappa^4 \ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^2},$$

with probability at least $(1-\eta)$, we always have $\|\mathcal{P}_{\hat{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\text{TV}} \leqslant \epsilon$.

V. EXPERIMENTS

We apply the proposed algorithm to accomplish the near separable nonnegative matrix factorization (SNMF) to validate the correctness of Theorem 10 [14,46] and compare the runtime complexity with original DCA. SNMF, which has been extensive applied to hyperspectral imaging, signal processing, and text mining, can be treated as a special case of the general minimum conical hull problem [10]. Given a nonnegative matrix $\mathbf{X}$, SNMF aims to find a decomposition such that $\mathbf{X} = \mathbf{F}\mathbf{X}(\mathcal{R}, :) + \mathbf{N}$, where the basis matrix $\mathbf{X}(\mathcal{R}, :)$ is composited by r rows from $\mathbf{X}$, $\mathbf{F}$ is the nonnegative encoding matrix, and $\mathbf{N}$ is a noise matrix [43].

The synthetic data matrix used in the experiments is generated in the form of $\mathbf{Y} \equiv \mathbf{X} = \mathbf{F}\mathbf{X}_\mathcal{A} + \mathbf{N}_G$, where $\mathbf{F} = [\mathbf{I}_k; \mathcal{U}]$, the entries of noise $\mathbf{N}_G$ are generated by sampling Gaussian distribution $\mathcal{N}(0, \mu)$ with noise level μ, the entries of $\mathbf{X}_\mathcal{A}$ and

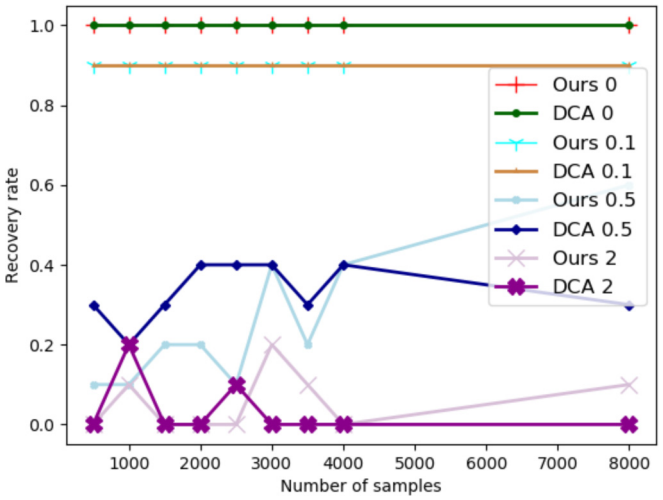

FIG. 2. Finding anchor set $\mathcal{A}$ of a 500×500 matrix of rank 10 on five noise levels and six different settings of the number of samples. The label "Ours x" refers to the noise level $\mu = x$, e.g., "DCA 2" represents $\mu = 2$.

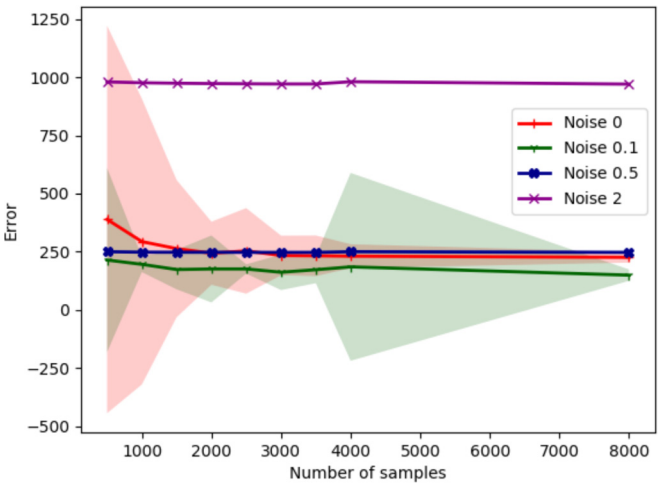

FIG. 3. Finding anchor set $\mathcal{A}$ of a 500×500 matrix of rank 10 on five noise levels and six different settings of the number of samples. The label "Noise x" refers to noise level $\mu = x$. The shadow region refers to the variance of the error, e.g., the green region refers to the variance of reconstruction error with $\mu = 0.5$.

$\mathcal{U}$ are generated by independent and identically distributed uniform distribution in $[0, 1]$ at first and then their rows are normalized to have unit ℓ_1 norm. Analogously to DCA, we use the recovery rate as the metric to evaluate the precision of anchor recovery. The anchor index recovery rate ρ is defined as $\rho = |\mathcal{A} \cap \hat{\mathcal{A}}|/|\mathcal{A}|$, where $\hat{\mathcal{A}}$ refers to the anchor set obtained by our algorithm or DCA.

We first set the dimensions of $\mathbf{X}$ as 500×500 and set rank k as 10 to validate the correctness of our algorithm. We generate four data sets with different noise levels, which are $\mu = 0, 0.1, 0.5, 2$. The number of subproblems is set as $p = 100$. We give nine different settings for the number of sampling s for our algorithm, ranging from 500 to 8000. We compare our algorithm with DCA to determine the anchor set $\mathcal{A}$. The simulation results are shown in Figs. 2 and 3. For the case of $\mu = 0$, both our algorithm and DCA can accurately locate all anchors. With increased noise, the recovery rate continuously decreases both for our algorithm and DCA, since the separability assumption is not preserved. As shown in Fig. 3, the reconstruction error, which is evaluated by the Frobenius norm of $\|\hat{\mathbf{X}} - \mathbf{X}\|_F$, continuously decreases with increased s for the noiseless case. In addition, the variance of the reconstructed error, illustrated by the shadow region, continuously shrinks with increased s. For the high noise level case, the collapsed separability assumption arises that the reconstruction error is unrelated to s.

We then generate a set of matrix with the rank $k = 10$, noise level $\mu = 0$, and size $n \in [5000, 70000]$ with interval 5000, to explore when our algorithm has lower runtime over original DCA. The number of sampling s and the subproblems is set as $s = 10 * k^2 \log(n)$ and $p = k \log(k)$, respectively. We compute the averaged runtime over p subproblems. As shown in Fig. 4, the runtime of our algorithm (highlight by green line) and original DCA (highlight by red line) is logarithmically and linearly increased with the linearly increased input size, respectively. The simulation results indicate that our algorithm outperforms classical DCA when n is sufficiently large.

VI. CONCLUSION

In this paper, we have proposed a sublinear runtime classical algorithm to resolve the general minimum conical hull problem. We first reformulated the general minimum conical hull problem as a sampling problem. We then exploited two sampling subroutines and proposed the general heuristic post-selection method to achieve low computational cost. We theoretically analyzed the correctness and the computation cost of our algorithm. The proposed algorithm benefit numerous learning tasks that can be mapped to the general minimum conical hull problem, especially for tasks that need to process data sets on a huge scale. The numerical simulation results indicate that the high exponents in the complexity bound can be further improved via advanced sampling techniques. There

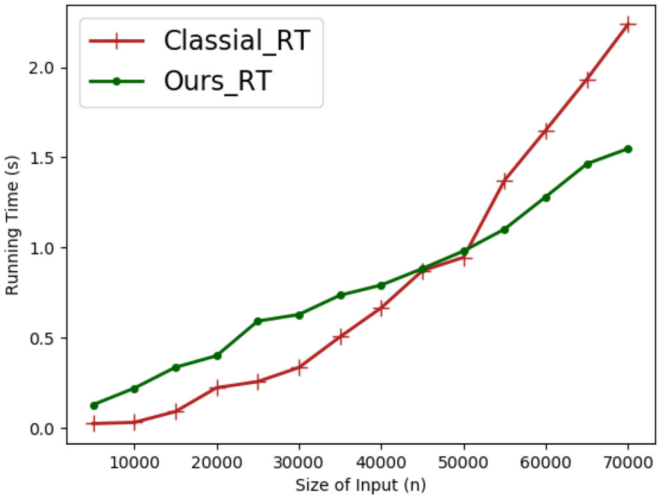

FIG. 4. The averaged runtime complexity for each subproblem of our algorithm and original DCA. The labels "Classical_RT" and "Ours_RT" refer to the runtime to accomplishing classical DCA and our algorithm, respectively.

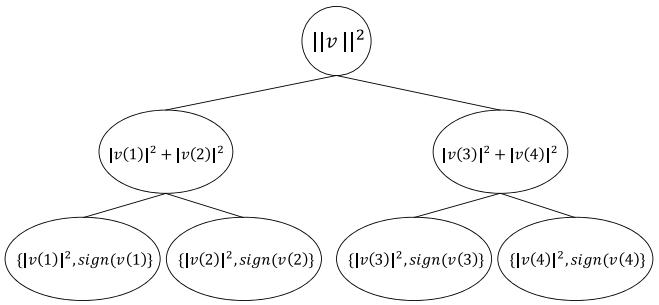

FIG. 5. The BNS for $\mathbf{v} \in \mathbb{R}^4$.

are three promising future directions. First, we will explore other advanced sampling techniques to further improve the polynomial dependence in the computation complexity. Second, we will investigate whether there exist other fundamental learning models that can be reduced to the general minimum conical hull problem. One of the most strongest candidates is the semidefinite programming solver. Third, for the quantum machine learning community, it is intriguing to explore other quantum machine learning algorithms that cannot be dequantied by quantum inspired algorithms.

We organize the Appendices as follows. In Appendix A, we detail the binary tree structure to support length-square sampling operations. We then provide the proof of Theorem 8 in Appendix B. Because the proof of Theorem 9 cost employs the results of Theorem 10, we give the proof of Theorem 10 in Appendix C and present the proof of Theorem 9 in Appendix D.

ACKNOWLEDGMENTS

This work received support from Australian Research Council (Project FL-170100117 and Future Fellowship under Grant No. FT140100574), and the Faculty of Engineering and Information Technologies at the University of Sydney (the Engineering and Information Technologies Research Scholarship).

APPENDIX A: THE BINARY TREE STRUCTURE FOR LENGTH-SQUARE SAMPLING OPERATIONS

As mentioned in the main text, a feasible solution to fulfill ℓ_2 norm sampling operations is the BNS to store data [23]. Here we give the intuition about how BNS constructed for a vector. For ease of notations, we assume the given vector has size 4 with $\mathbf{v} \in \mathbb{R}^4$. As demonstrated in Fig. 5, the root node records the square ℓ_2 norm of $\mathbf{v}$. The ith leaf node records the ith entry of $\mathbf{v}(i)$ and its square value, e.g., $\{|\mathbf{v}(i)|^2, \mathrm{sgn}[\mathbf{v}(i)]\}$. Each internal node contains the sum of the values of its two immediate children. Such an architecture ensures the ℓ_2 norm sampling operations.

APPENDIX B: GENERAL HEURISTIC POST-SELECTION (PROOF OF THEOREM 8)

Recall that the anchor $\mathcal{A}_t$ is defined as

$$\mathcal{A}_t = \arg\min_{i\in[n_Y]} \left\{ \mathcal{P}_{\hat{\mathbf{Y}}_t}(i) - \hat{\xi}_t \max_{j\in[n_X]} [\mathcal{P}_{\hat{\mathbf{X}}_t}(j)] \right\}_+.$$

The goal of the general heuristic postselection is approximating $\mathcal{A}_t$ by sampling distributions $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with $\mathcal{O}(\mathrm{polylog}[\max\{n_X, n_Y\}])$ times, since the acquisition of the explicit form of $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ requires $\mathcal{O}[\mathrm{poly}(n_X, n_Y)]$ computation complexity and collapses the desired speedup. Let $\{\mathbf{x}_i\}_{i=1}^{N_X}$ be N_X examples independently sampled from $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with $\mathcal{P}_{\hat{\mathbf{X}}_t}(\mathbf{x}=z) = |\hat{\mathbf{X}}_t(z)|^2/\|\hat{\mathbf{X}}_t\|^2$ and $N_{X,z}$ be the number of examples taking value of $z \in [n_X]$ with $\sum_{z=1}^{n_X} N_{X,z} = N_X$. Similarly, let $\{\mathbf{y}_i\}_{i=1}^{N_Y}$ be N_Y examples independently sampled from $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with $\mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathbf{y}=z) = |\hat{\mathbf{Y}}_t(z)|^2/\|\hat{\mathbf{Y}}_t\|^2$ and $N_{Y,z}$ be the number of examples taking value of $z \in [n_Y]$ with $\sum_{z=1}^{n_Y} N_{Y,z} = N_Y$. Denote that w_X is the total number of different indexes after sampling $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with N_X times, $I_{X,v}$ and $J_{X,v}$ are two indexes corresponding to the vth largest value among $\{N_{X,z}\}_{z=1}^{w_X}$ and $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with $w_X \leqslant n_X$ and $v \in [w_X]$, respectively.[2] Similarly, denote that w_Y is the total number of distinguished indexes after sampling $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with N_Y times, and the indexes $I_{Y,v}$ and $J_{Y,v}$ are vth largest value among $\{N_{Y,z}\}_{z=1}^{w_Y}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with $w_Y \leqslant n_Y$ and $v \in [w_Y]$, respectively. In particular, we have

$$I_{X,1} = \arg\max_z N_{X,z}, \quad I_{Y,1} = \arg\max_z N_{Y,z},$$

$$J_{X,1} = \arg\max_z \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathbf{x}=z), \quad \text{and } J_{Y,1} = \arg\max_z \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathbf{y}=z).$$

The procedure of the general heuristic postselection is as follows:

(i) Sample $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with N_X times and order the sampled items to obtain $\mathcal{X} = \{I_{X,1}, I_{X,2}, \ldots, I_{X,w_X}\}$ with $w_X \leqslant n_X$;

(ii) Query the distribution $\mathcal{P}_{\hat{\mathbf{X}}_t}$ to obtain the value $C^*_{\hat{\mathbf{X}}_t}$ with $C^*_{\hat{\mathbf{X}}_t} = \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathbf{x} = I_{X,1})$;

(iii) Sample $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with N_Y times and order the sampled items to obtain $\mathcal{Y} = \{I_{Y,1}, I_{Y,2}, \ldots, I_{Y,w_Y}\}$ with $w_Y \leqslant n_Y$.

(iv) Locate $\hat{\mathcal{A}}_t$ with $\hat{\mathcal{A}}_t = I_{Y,v^*}$ and $I_{Y,v^*} = \arg\min_{z\in\mathcal{Y}} (\frac{N_{Y,z}}{N_Y} - \hat{\xi}_t C^*_{\hat{\mathbf{X}}_t})_+$.

An immediate observation is that, with N_X, $N_Y \to \infty$, we have $\hat{\mathcal{A}}_t = \mathcal{A}_t$, where

$$\mathcal{P}_{\hat{\mathbf{X}}_t}(\mathbf{x} = I_{X,1}) = \max_{J_{X,v}\in[w_X]} [\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,v})],$$

and

$$\frac{N_{Y,I_{Y,v^*}}}{N_Y} = \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathbf{y} = \mathcal{A}_t), \quad w_X = n_X, \quad w_Y = n_Y.$$

The general heuristic postselection is guaranteed by the following theorem (the formal description of Theorem 5).

Theorem 11 ((Formal) General heuristic postselection). Assume that $\mathcal{P}_{\hat{\mathbf{X}}_t}$ and $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ are multinomial distributions. If $\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,1}) \geqslant \varepsilon_T$, $|\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,1}) - \mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,2})| > \varepsilon$, and $|\mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*\pm1}) - \mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*})| > \varepsilon$ for the constants ε_T and ε with $\mathcal{A}_t = J_{Y,v^*}$, then for any $\delta > 0$, $N_X \sim \mathcal{O}(\max\{1/\varepsilon, 1/\varepsilon_T\})$ and $N_Y \sim \mathcal{O}(\max\{1/\varepsilon, 1/(\hat{\xi}_t\varepsilon_T)\})$, we have $\hat{\mathcal{A}}_t = \mathcal{A}_t$ with a probability at least $1-\delta$.

Remark. The physical meaning of ε can be treated as the threshold of the "near-anchor," that is, when the distance of a

[2]Due to the limited sampling times, the sampled results $\{N_{X,1}, N_{X,2}, \ldots, N_{X,w_X}\}$ (or $\{N_{Y,1}, N_{Y,2}, \ldots, N_{Y,w_X}\}$) may occupy a small portion of the all n_X (or n_Y) possible results.

data point and the anchor point after projection is within the threshold ε, we say the data point can be treated as anchors. The real anchor set $\mathcal{A}$ therefore should be expanded and include these near anchors. In other words, ε and ε_T can be manually controllable. The parameter $\hat{\xi}_t$ is bounded as follows:

$$\hat{\xi}_t = \frac{\|\hat{\mathbf{Y}}_t\|}{\|\hat{\mathbf{X}}_t\|} \leqslant \kappa_X \|\hat{\mathbf{Y}}\|_F.$$

In this work, we set $1/\varepsilon \sim \mathcal{O}[\text{polylog}(n_X, n_Y)]$ and $1/\varepsilon_T \sim \mathcal{O}[\text{polylog}(n_X, n_Y)]$, which gives $N_X, N_Y \to \mathcal{O}(\kappa_X, \text{polylog}(\max\{n_X, n_Y\}))$.

An equivalent statement of Theorem 11 is as follows:

Problem 12. How many samples, N_X and N_Y, are required to guarantee $I_{X,1} = J_{X,1}$ and $I_{Y,v^*} = J_{Y,v^*}$, where $J_{X,1} = \arg\max_{J_{X,v} \in [n_X]} [\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,v^*})]$ and $\mathcal{A}_t = J_{Y,v^*}$.

We use Bretegnolle-Huber-Carol inequality [47] to prove Theorem 11, i.e.,

Lemma 13 (Bretegnolle-Huber-Carol inequality [47]). Let $\mathcal{P}_D$ be a multinomial distribution with l event probabilities $\{\mathcal{P}_D(i)\}_{i=1}^{l}$. We randomly sample N events from $\mathcal{P}_D$ and let N_i be the number of event i appeared. Then, the following holds with a probability at least $1 - \delta$ for any $\delta > 0$,

$$\mathcal{P}\left(\sum_{i=1}^{l} \left|\frac{N_i}{N} - p_i\right| \geqslant \lambda\right) \leqslant 2^l \exp\left(\frac{-N\lambda^2}{2}\right). \tag{B1}$$

Proof of Theorem 11. The proof is composed of two parts. The first part is to prove that the index $I_{X,1}$ with $I_{X,1} = J_{X,1}$ can be determined with sampling complexity $\mathcal{O}[1/(\varepsilon^2 \varepsilon_T^2)]$. The second part is to prove that the index I_{Y,w_Y} with $I_{Y,w_Y} = \mathcal{A}_t$ can be determined with sampling complexity $\mathcal{O}[1/(\varepsilon^2 \varepsilon_T^2)]$.

For the first part, we split the set $\mathcal{X}$ into two subsets $\mathcal{X}_1$ and $\mathcal{X}_2$, i.e., $\mathcal{X}_1 = \{I_{X,1}\}$ and $\mathcal{X}_2 = \{I_{X,2}, \ldots, I_{X,w_X}\}$. The decomposition of $\mathcal{X}$ into two subsets is equivalent to setting $l = 2$ in Eq. (B1). In particular, we have $N_1 = N_{X,I_{X,1}}$ and $N_2 = \sum_{v=2}^{w_X} N_{X,I_{X,v}}$. The Bretegnolle-Huber-Carol inequality yields

$$\mathcal{P}\left(\left|\frac{N_1}{N_X} - \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathcal{X}_1)\right| + \left|\left[\frac{N_2}{N_X} - \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathcal{X}_2)\right]\right| \geqslant \lambda\right) \leqslant 4\exp\left(\frac{-N_X\lambda^2}{2}\right). \tag{B2}$$

The above inequality implies that, when $\delta = 4\exp(\frac{-N_X\lambda^2}{2})$, we have

$$\left|\frac{N_{X,I_{X,1}}}{N_X} - \mathcal{P}_{\hat{\mathbf{X}}_t}(I_{X,1})\right| \leqslant \left|\frac{N_1}{N_X} - \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathcal{X}_1)\right| + \left|\left[\frac{N_2}{N_X} - \mathcal{P}_{\hat{\mathbf{X}}_t}(\mathcal{X}_2)\right]\right| \leqslant \sqrt{\frac{2\log(4/\delta)}{N_X}}, \tag{B3}$$

with probability at least $1 - \delta$.

The assumption $\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,1}) \geqslant \varepsilon_T$ guarantees that $J_{X,1} \in \mathcal{X}$ by sampling $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with $\mathcal{O}(1/\varepsilon_T)$ times. In addition, since we have assumed that $|\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,1}) - \mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,2})| > \varepsilon$, it can be easily inferred that, when $N_X \geqslant \sqrt{\frac{2\log(4/\delta)}{\varepsilon^2}}$ with a probability at least $1 - \delta$, there is one and only one value $\frac{N_{X,I_{X,1}}}{N_X}$ that is in the ε-neighborhood of $\mathcal{P}_{\hat{\mathbf{X}}_t}(I_{X,1})$. We therefore conclude that $I_{X,1} = J_{X,1}$ can be guaranteed by sampling $\mathcal{P}_{\hat{\mathbf{X}}_t}$ with $N_X \geqslant \max\{\sqrt{\frac{2\log(4/\delta)}{\varepsilon^2}}, \frac{1}{\varepsilon_T}\}$.

For the second part, we split the set $\mathcal{Y}$ into three subsets $\mathcal{Y}_1$, $\mathcal{Y}_2$, and $\mathcal{Y}_3$, i.e., $\mathcal{Y}_1 = \{I_{Y,1}, \ldots, I_{Y,v^*-1}\}$, $\mathcal{Y}_2 = \{I_{Y,v^*}\}$, and $\mathcal{Y}_3 = \{I_{Y,v^*+1}, \ldots, I_{Y,w_Y}\}$. Analogous to the above case, the decomposition of $\mathcal{Y}$ into three subsets for the case of sampling $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ is equivalent to setting $l = 3$ in Eq. (B1). In particular, we have $N_1 = \sum_{v=1}^{v^*} N_{Y,I_{Y,v}}$, $N_2 = N_{Y,I_{Y,v^*}}$, and $N_3 = \sum_{v=v^*+1}^{w_Y} N_{Y,I_{Y,v}}$. The Bretegnolle-Huber-Carol inequality yields

$$\mathcal{P}\left(\left|\frac{N_1}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_1)\right| + \left|\left[\frac{N_2}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_2)\right]\right| + \left|\left[\frac{N_3}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_3)\right]\right| \geqslant \lambda\right) \leqslant 8\exp\left(\frac{-N_Y\lambda^2}{2}\right). \tag{B4}$$

The above inequality implies that, when $\delta = 4\exp(\frac{-N_Y\lambda^2}{2})$, we have

$$\left|\frac{N_{Y,I_{Y,v^*}}}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(I_{Y,v^*})\right| \leqslant \left|\frac{N_1}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_1)\right| + \left|\left[\frac{N_2}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_2)\right]\right| + \left|\left[\frac{N_3}{N_Y} - \mathcal{P}_{\hat{\mathbf{Y}}_t}(\mathcal{Y}_3)\right]\right| \leqslant \sqrt{\frac{2\log(8/\delta)}{N_Y}}, \tag{B5}$$

with probability at least $1 - \delta$.

Since we have assumed $\mathcal{P}_{\hat{\mathbf{X}}_t}(J_{X,1}) \geqslant \varepsilon_T$, we have $\mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*}) \geqslant \hat{\xi}_t \varepsilon_T$. In other words, $\mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*}) \geqslant \hat{\xi}_t \varepsilon_T$ guarantees that $I_{Y,v^*} = J_{Y,v^*}$ by sampling $\mathcal{P}_{\hat{\mathbf{Y}}_t}$ with $\mathcal{O}[1/(\hat{\xi}_t \varepsilon_T)]$ times. In addition, the assumption $|\mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*\pm1}) - \mathcal{P}_{\hat{\mathbf{Y}}_t}(J_{Y,v^*})| > \varepsilon$ leads to that, when $N_Y \geqslant \max\{\sqrt{\frac{2\log(8/\delta)}{\varepsilon^2}}, \frac{1}{\hat{\xi}_t \varepsilon_T}\}$ with a probability at least $1 - \delta$, there is one and only one value $\frac{N_{Y,I_{Y,v^*}}}{N_Y}$ that is in the ε neighborhood of $\mathcal{P}_{\hat{\mathbf{Y}}_t}(I_{Y,v^*})$. We therefore conclude that $I_{Y,v^*} = J_{Y,v^*}$ with $J_{Y,v^*} = \hat{\mathcal{A}}_t$.

Combing the results of two parts together, it can be easily inferred that, with the sampling complexity $\tilde{\mathcal{O}}(\max\{\frac{1}{\varepsilon}, \frac{1}{\hat{\xi}_t \varepsilon_T}\})$, we have $\mathcal{A}_t = \hat{\mathcal{A}}_t$ with probability $1 - \delta$. ■

APPENDIX C: PROOF OF THEOREM 10

In this Appendix, we give the proof of Theorem 10. Detailed proofs of Theorems 15 and 16 are in subsections C 3 and C 4, respectively.

Proof of Theorem 10. Recall that $\hat{\mathbf{H}}_t = \tilde{\mathbf{V}}\hat{\mathbf{q}}_{H,t}$, and $\hat{\mathbf{q}}_{H,t}$ is an approximation of $\tilde{\mathbf{q}}_{H,t} = \tilde{\mathbf{V}}^\top \mathbf{H}_t$. The triangle inequality yields

$$\|\mathcal{P}_{\hat{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\text{TV}} \leqslant \|\mathcal{P}_{\hat{\mathbf{H}}_t} - \mathcal{P}_{\tilde{\mathbf{H}}_t}\|_{\text{TV}} + \|\mathcal{P}_{\tilde{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\text{TV}}, \tag{C1}$$

where $\|\mathcal{Q}\|_{\text{TV}}$ is the total variation distance of $\mathcal{Q}$. In the following, we bound the two terms on the right-hand side of Eq. (C1) respectively.

1. Correctness of $\|\mathcal{P}_{\tilde{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\mathrm{TV}}$.

The goal here is to prove that

$$\|\mathcal{P}_{\tilde{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\mathrm{TV}} \leqslant \frac{\epsilon}{2}. \tag{C2}$$

By Lemma 14 below, Eq. (C2) follows if the following inequality holds:

$$\left\|\tilde{\mathbf{H}}_t - \mathbf{H}_t\right\| \leqslant \frac{\epsilon}{4}\|\mathbf{H}_t\|. \tag{C3}$$

Finally, the inequality in Eq. (C3) is guaranteed to hold because of Theorem 15.

Lemma 14 (Lemma 6.1 [13]). For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ satisfying $\|\mathbf{x} - \mathbf{y}\| \leqslant \epsilon$, the corresponding distributions $\mathcal{P}_{\mathbf{x}}$ and $\mathcal{P}_{\mathbf{y}}$ satisfy $\|\mathcal{P}_{\mathbf{x}}, \mathcal{P}_{\mathbf{y}}\|_{\mathrm{TV}} \leqslant \frac{2\epsilon}{\|\mathbf{x}\|}$.

Theorem 15. Let the rank and the condition number of $\mathbf{H} \in \mathbb{R}^{n\times m}$ be k and κ, respectively. Fix

$$s = \frac{85^2 k^3 \kappa^4 \ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^2}.$$

Then, Algorithm 2 yields $\|\tilde{\mathbf{H}}_t - \mathbf{H}_t\| \leqslant \frac{\epsilon\|\mathbf{H}_t\|}{4}$ with probability at least $(1-\eta)$.

2. Correctness of $\|\mathcal{P}_{\hat{\mathbf{H}}_t} - \mathcal{P}_{\tilde{\mathbf{H}}_t}\|_{\mathrm{TV}}$.

Analogously to the above part, we bound

$$\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\| \leqslant \frac{\epsilon}{4}\|\tilde{\mathbf{H}}_t\|, \tag{C4}$$

to yield

$$\|\mathcal{P}_{\hat{\mathbf{H}}_t} - \mathcal{P}_{\tilde{\mathbf{H}}_t}\|_{\mathrm{TV}} \leqslant \frac{\epsilon}{2}. \tag{C5}$$

And Eq. (C4) can be obtained by the following theorem.

Theorem 16. Let the rank of $\mathbf{H} \in \mathbb{R}^{n\times m}$ be k. Set the number of samplings in the inner product subroutine as

$$N_Z \sim \mathcal{O}\left(\frac{(4+\epsilon)\sqrt{k}\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{4\epsilon}\log(1/\delta)\right).$$

Then Algorithm 2 yields $\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\| \leqslant \frac{\epsilon}{4}\|\tilde{\mathbf{H}}_t\|$ with at least $1-\delta$ success probability.

Finally, Eq. (C1) holds combining Eq. (C2) with Eq. (C5). ■

3. Proof of Theorem 15

Due to $\mathbf{H}_t = \mathbf{V}\mathbf{V}^\top\mathbf{H}\mathbf{B}_t$, we have

$$\begin{aligned}\|\tilde{\mathbf{H}}_t - \mathbf{H}_t\| &= \|\tilde{\mathbf{V}}\tilde{\mathbf{V}}^\top\mathbf{H}\mathbf{B}_t - \mathbf{V}\mathbf{V}^\top\mathbf{H}\mathbf{B}_t\| \\ &\leqslant \left\|\left(\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right)\right\|_2 \|\mathbf{H}\mathbf{B}_t\| \\ &\leqslant \left\|\left(\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right)\right\|_2 \|\mathbf{H}\|_2, \end{aligned} \tag{C6}$$

where $\Pi_{(\mathbf{H})} = \sum_i \mathbf{v}^{(i)}\mathbf{v}^{(i)\top}$ and $\mathbf{v}^{(i)}$ is the left singular vectors of $\mathbf{H}$. The first inequality of Eq. (C6) is obtained by exploiting the submultiplicative property of spectral norm [48], i.e., for any matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ and any vector $\mathbf{z} \in \mathbb{R}^m$, we have $\|\mathbf{M}\mathbf{z}\| \leqslant \|\mathbf{M}\|_2\|\mathbf{z}\|$. The second inequality of Eq. (C6) comes from the submultiplicative property of spectral norm and $\|\mathbf{B}_t\| = 1$. To achieve $\|\tilde{\mathbf{H}}_t - \mathbf{H}_t\| \leqslant \frac{\epsilon}{4}\|\mathbf{H}_t\|$ in Eq. (C3), Eq. (C6) indicates that the approximated left singular matrix $\tilde{\mathbf{V}}$ should satisfy

$$\left\|\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right\|_2 \leqslant \frac{\epsilon}{4}. \tag{C7}$$

The spectral norm $\|\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\|_2$ can be quantified as follows.

Theorem 17. Suppose that the rank of $\mathbf{H}$ is k and $\tilde{\mathbf{v}}^{(i)}$ refers to a approximated singular vector of $\mathbf{H}$ such that

$$\left|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| \leqslant \alpha \leqslant \frac{1}{4k}. \tag{C8}$$

Then, we have

$$\left\|\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right\|_2 \leqslant \frac{17k\alpha}{3}. \tag{C9}$$

The proof of Theorem 17 is given in subsection D 4.

Theorem 17 implies that to achieve Eq. (C7) [or, equivalently, Eq. (C3)], we should bound α as

$$\left\|\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right\|_2 \leqslant \frac{17k\alpha}{3} \leqslant \frac{\epsilon}{4}. \tag{C10}$$

We use the following lemma to give an explicit representation of α by the sampled matrix $\mathbf{R}$ and $\mathbf{C}$,

Lemma 18. Suppose that $\omega^{(l)}$ refers to the right singular vector of $\mathbf{C}$ such that $\Pi_{(\mathbf{C})} = \sum_l \omega^{(l)}\omega^{(l)\top}$ and $\omega^{(i)\top}\mathbf{C}^\top\mathbf{C}\omega^{(j)} = \delta_{ij}(\sigma^{(i)})^2$, where $(\sigma^{(i)})^2 \geqslant 4/(5\kappa^2)$. Suppose that the rank of both R and C is k and

$$\|\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C}\|_2 \leqslant \gamma.$$

Let $\tilde{\mathbf{v}}^{(l)} := \mathbf{R}\omega^{(l)}/\sigma^{(l)}$, and then we have

$$\left|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| \leqslant \frac{5\kappa^2\gamma}{4}. \tag{C11}$$

The proof of Lemma 18 is presented in subsection C 3 a.

In conjunction with Eq. (C8) and Eq. (C11), we set $\alpha = (5\gamma\kappa^2)/4$ and rewrite Eq. (C10) as

$$\left\|\sum_{i=1}^k \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\right\|_2 \leqslant \frac{85k\gamma\kappa^2}{12} \leqslant \frac{\epsilon}{4}. \tag{C12}$$

In other words, when $\gamma \leqslant \frac{3\epsilon}{85k\kappa^2}$, Eq. (C3) is achieved so that $\|\tilde{\mathbf{H}}_t - \mathbf{H}_t\| \leqslant \frac{\epsilon}{4}\|\mathbf{H}_t\|$. Recall that γ is quantified by $\|\mathbf{R}^\top\mathbf{R} - \mathbf{C}\mathbf{C}^\top\|_2$ as defined in Eq. (C11), we use the following theorem to bound γ.

Theorem 19. Given a nonnegative matrix $\mathbf{H} \in \mathbb{R}^{n\times m}$, let $\mathbf{R} \in \mathbb{R}^{n\times s}$, $\mathbf{C} \in \mathbb{R}^{s\times s}$ be the sampled matrix following Algorithm 1. Setting s as $s = \frac{85^2k^2\kappa^4\ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^2}$, with probability at least $(1-\eta)$, we always have $\|\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C}\| \leqslant \gamma$,

The proof of Theorem 19 is given in subsection C 3 b.

Combining the result of Theorem 17 and Lemma 18, we know that with sampling s rows of $\mathbf{H}$, the approximated distribution is $\frac{\epsilon}{2}$ close to the desired result, i.e.,

$$\|\mathcal{P}_{\tilde{\mathbf{H}}_t} - \mathcal{P}_{\mathbf{H}_t}\|_{\mathrm{TV}} \leqslant \frac{\epsilon}{2}.$$

4. Proof of Theorem 15

Proof of Theorem 17. We first introduce a lemma to facilitate the proof of Theorem 17.

Lemma 20 (Adapted from Lemma 5 [15]). Let $\mathbf{A}$ be a matrix of rank at most k, and suppose that $\mathbf{W}$ has k columns that span the row and column spaces of $\mathbf{A}$. Then $\|\mathbf{A}\|_2 \leqslant \|(\mathbf{W}^\top\mathbf{W})^{-1}\|_2\|\mathbf{W}^\top\mathbf{A}\mathbf{W}\|_2$.

Proof of Theorem 17. The main procedure to prove this theorem is as follows. By employing the Lemma 20, we can set $\mathbf{A}$ and $\mathbf{W}$ as

$$\mathbf{A} := \sum_{i=1}^{k} \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}, \quad \mathbf{W} \equiv \mathbf{V} = \sum_{i=1}^{k} \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top},$$

and then bound $\|\mathbf{W}^\top\mathbf{A}\mathbf{W}\|_2$ and $\|(\mathbf{W}^\top\mathbf{W})^{-1}\|_2$ separately. Last, we combine the two results to obtain the bound $\|\sum_{i=1}^{k} \tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top} - \Pi_{(\mathbf{H})}\|_2 \leqslant \frac{17k\alpha}{3}$ in Eq. (C9).

Following the above observation, we first bound the term $\|\mathbf{W}^\top\mathbf{A}\mathbf{W}\|_2$. We rewrite $\mathbf{W}^\top\mathbf{A}\mathbf{W}$ as

$$\mathbf{W}^\top\mathbf{A}\mathbf{W} = \sum_{i,j=1}^{k} \tilde{\mathbf{v}}^{(i)}(\tilde{\mathbf{v}}^{(i)\top}A\tilde{\mathbf{v}}^{(j)})\tilde{\mathbf{v}}^{(j)\top}. \tag{C13}$$

The entry $\mathbf{A}(i, j)$ of $\mathbf{A}$ with $\mathbf{A}(i, j) = (\tilde{\mathbf{v}}^{(i)\top}A\tilde{\mathbf{v}}^{(j)})$ is bounded by α, i.e.,

$$\begin{aligned}
&\left|\tilde{\mathbf{v}}^{(i)\top}\mathbf{A}\tilde{\mathbf{v}}^{(j)}\right| \\
&= \left|\tilde{\mathbf{v}}^{(i)\top}\left(\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top} - \Pi_{(\mathbf{H})}\right)\tilde{\mathbf{v}}^{(j)}\right| \\
&= \left|\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)} - \tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}\right| \\
&\leqslant \left|\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| + \alpha \\
&\leqslant \left|\sum_{t=1,t\neq\{i,j\}}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)}\right| \\
&\quad + \left|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| + \alpha \\
&\leqslant \left|\sum_{t=1,t\neq\{i,j\}}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)}\right| + 4\alpha \leqslant \frac{17\alpha}{4}.
\end{aligned} \tag{C14}$$

The first equivalence of Eq. (C14) comes from the definition of $\mathbf{A}$, and the second equivalence employs $\tilde{\mathbf{v}}^{(i)\top}\Pi_{(\mathbf{H})}\tilde{\mathbf{v}}^{(j)} = \tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}$. The first inequality of Eq. (C14) exploits triangle inequality and Eq. (C8), i.e.,

$$\begin{aligned}
&\left|\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)} - \tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}\right| \\
&\leqslant \left|\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| + |\delta_{ij} - \tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}| \\
&\leqslant \left|\sum_{t=1}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| + \alpha.
\end{aligned} \tag{C15}$$

The second inequality of Eq. (C14) directly comes from the triangle inequality. The last second inequality of Eq. (C14) employs the inequality $|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}| \leqslant 3\alpha$ for both the case $i = j$ and $i \neq j$, guaranteed by Eq. (C8) and $\alpha^2 < \alpha$. Specifically, for the case $i \neq j$, we bound $|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}|$ as

$$\begin{aligned}
&\left|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)}\right| \\
&\leqslant |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}\tilde{\mathbf{v}}^{(j)\top}\tilde{\mathbf{v}}^{(j)}| + |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}| \\
&= |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}||\tilde{\mathbf{v}}^{(j)\top}\tilde{\mathbf{v}}^{(j)}| + |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)}||\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)}| \\
&\leqslant \alpha(|\tilde{\mathbf{v}}^{(j)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{jj} + \delta_{jj}|) + (|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)} - \delta_{ii} + \delta_{ii}|)\alpha \\
&\leqslant 2\alpha(\alpha + 1) \leqslant 3\alpha.
\end{aligned}$$

For the case $i = j$, we bound $|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}|$ as

$$\begin{aligned}
&\left|\sum_{t'=\{i,j\}}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t')}\tilde{\mathbf{v}}^{(t')\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}\right| \\
&= |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)} - \delta_{ii}| \\
&\leqslant |\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)} - \delta_{ii}||\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(i)} + \delta_{ii}| \\
&\leqslant \alpha(\alpha + 1) \leqslant 3\alpha.
\end{aligned}$$

The last inequality of Eq. (C14) comes from

$$\left|\sum_{t=1,t\neq\{i,j\}}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)}\right| \leqslant \frac{\alpha}{4},$$

since

$$\begin{aligned}
&\left|\sum_{t=1,t\neq\{i,j\}}^{k}\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}\tilde{\mathbf{v}}^{(t)\top}\tilde{\mathbf{v}}^{(j)}\right| \\
&= \sum_{t=1,t\neq\{i,j\}}^{k}|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(t)}|^2 \\
&\leqslant \sum_{t=1,t\neq\{i,j\}}^{k}\alpha^2 \leqslant k\alpha^2 \leqslant \alpha/4.
\end{aligned}$$

Combing Eq. (C13) and Eq. (C14), we immediately have

$$\|\mathbf{W}^\top\mathbf{A}\mathbf{W}\|_2 \leqslant \frac{17k\alpha}{4}.$$

In addition, from Eq. (C8) and the definition of $\mathbf{W}$, we can obtain

$$\|\mathbf{W}^\top\mathbf{W} - \mathbf{I}\|_2 \leqslant k\alpha \leqslant 1/4$$

and then $\|(\mathbf{W}^\top\mathbf{W})^{-1}\|_2 \leqslant 4/3$. Combining the two result, we have

$$\|\mathbf{A}\|_2 \leqslant \|(\mathbf{W}^\top\mathbf{W})^{-1}\|_2\|\mathbf{W}^\top\mathbf{A}\mathbf{W}\|_2 \leqslant \frac{17k\alpha}{3}. \tag{C16}$$

■

a. Proof of Lemma 18

Proof of Lemma 18. The inequality $|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}| \leqslant \frac{5\kappa^2\gamma}{4}$ in Eq. (C11) can be proved following the definition of $\tilde{\mathbf{v}}^{(i)}$. Mathematically, we have

$$\begin{aligned}|\tilde{\mathbf{v}}^{(i)\top}\tilde{\mathbf{v}}^{(j)} - \delta_{ij}| &= \left|\frac{\omega^{(i)\top}\mathbf{R}^\top\mathbf{R}\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right| \\ &\leqslant \left|\frac{\omega^{(i)\top}\mathbf{C}^\top\mathbf{C}\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right| + \frac{\gamma}{\sigma^{(i)}\sigma^{(j)}} \\ &= \frac{\gamma}{\sigma^{(i)}\sigma^{(j)}} \leqslant \frac{5\kappa^2\gamma}{4}. \end{aligned} \tag{C17}$$

The first equivalence of Eq. (C17) comes from the definition of $\tilde{\mathbf{v}}^{(i)}$. The first inequality of Eq. (C17) is derived by employing $\|\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C}\| \leqslant \gamma$, i.e.,

$$\begin{aligned}&\left|\frac{\omega^{(i)\top}\mathbf{R}^\top\mathbf{R}\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right| \\ &= \left|\frac{\omega^{(i)\top}(\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C} + \mathbf{C}^\top\mathbf{C})\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right| \\ &\leqslant \left|\frac{\omega^{(i)\top}(\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C})\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}}\right| + \left|\frac{\omega^{(i)\top}\mathbf{C}^\top\mathbf{C}\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right| \\ &\leqslant \frac{\gamma}{\sigma^{(i)}\sigma^{(j)}} + \left|\frac{\omega^{(i)\top}\mathbf{C}^\top\mathbf{C}\omega^{(j)}}{\sigma^{(i)}\sigma^{(j)}} - \delta_{ij}\right|. \end{aligned} \tag{C18}$$

The last second equivalence of Eq. (C17) employs

$$\omega^{(i)\top}\mathbf{C}^\top\mathbf{C}\omega^{(j)} = \delta_{ij}(\sigma^{(i)})^2.$$

The last inequality of Eq. (C17) uses

$$(\sigma^{(i)})^2 \geqslant 4/(5\kappa^2).$$

■

b. Proof of Theorem 19

We introduce the following lemma to facilitate the proof.

Lemma 21 (Adapted from Theorem 4.4 [49]). Given any matrix $\mathbf{R} \in \mathbb{R}^{n\times s}$. Let $\mathbf{C} \in \mathbb{R}^{s\times s}$ be obtained by length-squared sampling with $\mathbb{E}(\mathbf{C}^\top\mathbf{C}) = \mathbf{R}^\top\mathbf{R}$. Then, for all $\epsilon \in [0, \|\mathbf{R}\|_2/\|\mathbf{R}\|_F]$, we have

$$\Pr(\|\mathbf{C}^\top\mathbf{C} - \mathbf{R}^\top\mathbf{R}\|_2 \geqslant \epsilon\|\mathbf{R}\|_2\|\mathbf{R}\|_F) \leqslant 2ne^{-\epsilon^2 s/4}. \tag{C19}$$

Hence, for $s \geqslant 4\ln(2n/\eta)/\epsilon^2$, with probability at least $(1-\eta)$ we have $\|\mathbf{C}^\top\mathbf{C} - \mathbf{R}^\top\mathbf{R}\|_2 \leqslant \epsilon\|\mathbf{R}\|\|\mathbf{R}\|_F$.

Proof of Theorem 19. The Lemma 21 indicates that the sample complexity of s determines $\|\mathbf{C}^\top\mathbf{C} - \mathbf{R}^\top\mathbf{R}\|_2$. With setting $\gamma = \epsilon\|\mathbf{R}\|_2\|\mathbf{R}\|_F$, we have

$$\Pr(\|\mathbf{C}^\top\mathbf{C} - \mathbf{R}^\top\mathbf{R}\|_2 \geqslant \gamma) \leqslant 2ne^{-\gamma^2 s/4(\|\mathbf{R}\|_2\|\mathbf{R}\|_F)}. \tag{C20}$$

Let the right-hand side of the above inequality be η, i.e.,

$$\begin{aligned}&2ne^{-\gamma^2 s/4(\|\mathbf{R}\|_2\|\mathbf{R}\|_F)} = \eta \\ &\xrightarrow{\log} 4\|\mathbf{R}\|_2\|\mathbf{R}\|_F\ln(2n/\eta) = \gamma^2 s \\ &\to s = \frac{4\|\mathbf{R}\|_2\|\mathbf{R}\|_F\ln(2n/\eta)}{\gamma^2} \\ &\to s = \frac{85^2k^2\kappa^4\|\mathbf{R}\|_2\|\mathbf{R}\|_F\ln(8n/\eta)}{9\epsilon^2} \\ &\leqslant \frac{85^2k^2\kappa^4\|\mathbf{H}\|_F^2\ln(8n/\eta)}{9\epsilon^2}, \end{aligned} \tag{C21}$$

where the inequality comes from $\|\mathbf{R}\|_F \leqslant \|\mathbf{R}\|_F$ and $\|\mathbf{R}\|_F = \|\mathbf{H}\|_F$. Therefore, with setting s as

$$s = \frac{85^2k^2\kappa^4\ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^2},$$

we have $\|\mathbf{R}^\top\mathbf{R} - \mathbf{C}^\top\mathbf{C}\|_2 \leqslant \gamma$ with probability at least $(1-\eta)$. ■

5. Proof of Theorem 16

Proof of Theorem 16. We first give the upper bound of the term $\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\|$, i.e.,

$$\begin{aligned}\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\| &= \|\tilde{\mathbf{V}}\hat{\mathbf{q}}_{H,t} - \tilde{\mathbf{V}}\tilde{\mathbf{q}}_{H,t}\| \\ &\leqslant \|\tilde{\mathbf{V}}\|_2\|\hat{\mathbf{q}}_{H,t} - \tilde{\mathbf{q}}_{H,t}\| \leqslant \frac{4+\epsilon}{4}\|\hat{\mathbf{q}}_{H,t} - \tilde{\mathbf{q}}_{H,t}\|. \end{aligned} \tag{C22}$$

The first inequality comes from the the submultiplicative property of spectral norm. The second inequality supported by Eq. (C12) with

$$\|\tilde{\mathbf{V}}\|_2 \leqslant 1 + \frac{\epsilon}{4}.$$

Following the definition of ℓ_2 norm, we have

$$\|\hat{\mathbf{q}}_{H,t} - \tilde{\mathbf{q}}_{H,t}\|^2 = \sum_{i=1}^{k}[\hat{\mathbf{q}}_{H,t}(i) - \tilde{\mathbf{q}}_{H,t}(i)]^2.$$

Denote the additive error

$$\epsilon' = \max_{i\in[k]}|\hat{\mathbf{q}}_{H,t}(i) - \tilde{\mathbf{q}}_{H,t}(i)|,$$

we rewrite Eq. (C22) as

$$\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\| \leqslant \frac{4+\epsilon}{4}\sqrt{k}\epsilon'. \tag{C23}$$

An observation of the above equation is that we have $\|\hat{\mathbf{H}}_t - \tilde{\mathbf{H}}_t\| \leqslant \epsilon/4$ if

$$\epsilon' \leqslant \frac{4\epsilon}{(4+\epsilon)\sqrt{k}}. \tag{C24}$$

We use the result of the inner product subroutine to quantify the required number of samplings to achieve Eq. (C24). The conclusion of Lemma 6 is that when

$$N_Z \sim \mathcal{O}\left(\frac{\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{\epsilon}\log(1/\delta)\right),$$

we have $|\tilde{\mathbf{q}}_{H,t}(i) - \hat{\mathbf{q}}_{H,t}(i)| \leqslant \epsilon$ with at least $1-\delta$ success probability. With substituting ϵ by ϵ', we immediately obtain $|\tilde{\mathbf{q}}_{H,t}(i) - \hat{\mathbf{q}}_{H,t}(i)| \leqslant \epsilon'$, where the required number of samplings is

$$N_Z \sim \mathcal{O}\left(\frac{(4+\epsilon)\sqrt{k}\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{4\epsilon}\log(1/\delta)\right). \tag{C25}$$

■

APPENDIX D: THE COMPLEXITY OF THE ALGORITHM (PROOF OF THEOREM 9)

Proof of Theorem 9. As analyzed in the main text, the complexity of the proposed algorithm is dominated by four operations in the preprocessing step and the divide step, i.e., finding the left singular vectors $\tilde{\mathbf{V}}$, estimating the inner product to build $\hat{\mathbf{q}}_t$, preparing the approximated probability distribution $\mathcal{P}_{\hat{\mathbf{H}}}$, and estimating the rescale factor $\hat{\xi}_t$. We evaluate the computation complexity of these four operations separately and then give the overall computation complexity of our algorithm.

In this subsection, we first evaluate the computation complexity of these four parts separately and then combine the results to give the computation complexity of our algorithm. Due to same reconstruction rule, we use a general setting $\mathbf{H} \in \mathbb{R}^{n\times m}$ that can either be $\mathbf{X}$ or $\mathbf{Y}$ to evaluate the computation complexity for the four parts.

1. Complexity of finding $\tilde{\mathbf{V}}$

Supported by the ℓ_2 norm sampling operations, the matrix $\mathbf{C}$ can be efficiently constructed following Algorithm 1, where $\mathcal{O}[2s\log^2(mn)]$ query complexity is sufficient. Applying SVD onto $\mathbf{C} \in \mathbb{R}^{s\times s}$ with $s = \frac{85^2k^2\kappa^4\ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^2}$ generally costs

$$\mathcal{O}(s^3) = \tilde{\mathcal{O}}\left(\frac{85^6k^6\kappa^{12}\|\mathbf{H}\|_F^6}{9^3\epsilon^6}\right)$$

runtime complexity. Once we obtain such the SVD result of $\mathbf{C}$, the approximated left singular vectors $\tilde{\mathbf{V}}$ can be implicitly represented, guaranteed by the following Lemma:

Lemma 22 (Adapted from Ref. [13]). Let the given data set support the ℓ_2 norm sampling operations along with the description of $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times s}$, We can sample from any $\tilde{\mathbf{v}}^{(t)}$ in $\mathcal{O}(Ks^2)$ expected queries with $K = \kappa\|\mathbf{H}\|_F^2$ and query for any particular entry $\tilde{\mathbf{V}}(i,j)$ in $\mathcal{O}(s)$ queries.

2. Complexity of estimating $\tilde{\mathbf{q}}_t$ by $\hat{\mathbf{q}}_t$

The runtime complexity to estimate $\tilde{\mathbf{q}}_t$ by $\hat{\mathbf{q}}_t$ obeys the following corollary, i.e.,

Corollary 23. Let $\mathbf{B}_t \in \mathbb{R}^{m\times 1}$ be the input vector, $\mathbf{H} \in \mathbb{R}^{n\times m}$ be the input matrix with rank k, and $\tilde{\mathbf{V}} \in \mathbb{R}^{n\times k}$ be the approximated left singular matrix. We can estimate $\tilde{\mathbf{q}}_t = \tilde{\mathbf{V}}^\top\mathbf{H}\mathbf{B}_t$ by $\hat{\mathbf{q}}_t$ to precision ϵ with probability at least $1-\delta$ using

$$\tilde{\mathcal{O}}\left(\frac{4k(1+\epsilon)^{1.5}\|\mathbf{H}\|_F}{\epsilon}\left[\frac{85^2k^2\kappa^4\|\mathbf{H}\|_F^2\ln(8n/\eta)}{9\epsilon^2}\right]\right)$$

runtime complexity.

Proof of Corollary 23. The proof of Corollary 23 employs the result of Theorem 16 and Lemma 6.

Theorem 16 indicates that, to estimate $\tilde{\mathbf{q}}_t(i)$ by $\hat{\mathbf{q}}_t(i)$, the required number of samplings is

$$N_Z \sim \mathcal{O}\left(\frac{(4+\epsilon)\sqrt{k}\|\mathbf{H}\|_F\|\mathbf{B}_t\|\|\tilde{\mathbf{v}}^{(i)}\|}{4\epsilon}\log(1/\delta)\right).$$

Following the result of Lemma 6, the runtime complexity to obtain $\hat{\mathbf{q}}_t(i)$ is

$$N_Z[L(\mathbf{H}) + Q(\tilde{\mathbf{V}}) + Q(\mathbf{B}_t)]. \tag{D1}$$

Since $\|\tilde{\mathbf{v}}^{(i)}\| \leqslant \sqrt{1+\epsilon}$ for any $\tilde{\mathbf{v}}^{(i)}$ indicated by Eq. (C22), we rewrite Eq. (D1) as

$$N_Z[L(\mathbf{H}) + Q(\tilde{\mathbf{V}}_t) + Q(\mathbf{B}_t)]$$
$$\leqslant \tilde{\mathcal{O}}\left(\frac{\|\mathbf{H}\|_F(1+\epsilon)^{1.5}\sqrt{k}}{\epsilon}[L(\mathbf{H}) + Q(\tilde{\mathbf{V}}_t) + Q(\mathbf{B}_t)]\right). \tag{D2}$$

We now quantify the access cost of $\mathbf{H}$, $\mathbf{B}_t$, and $\tilde{\mathbf{V}}$ to give an explicit bound of Eq. (D1). We have $L(\mathbf{H}) = \mathcal{O}[\log^2(nm)]$ and $Q(\mathbf{B}_t) = \mathcal{O}[\log(m)]$, since $\mathbf{H}$ and $\mathbf{B}_t$ are stored in BNS data structure. We have $Q(\tilde{\mathbf{V}}) = \mathcal{O}(s)$ supported by Lemma 22. Combing the above access cost and Eq. (D1), the runtime complexity to estimate $\hat{\mathbf{q}}_t(i)$ is

$$\tilde{\mathcal{O}}\left(\frac{(1+\epsilon)^{1.5}\sqrt{k}\|\mathbf{H}\|_F}{\epsilon}\frac{85^2k^2\kappa^4\|\mathbf{H}\|_F^2\ln(8n/\eta)}{9\epsilon^2}\right).$$

Since each entry can be computed in parallel, the runtime complexity to obtain $\hat{\mathbf{q}}_t$ is also

$$\tilde{\mathcal{O}}\left(\frac{(1+\epsilon)^{1.5}\sqrt{k}\|\mathbf{H}\|_F}{\epsilon}\frac{85^2k^2\kappa^4\|\mathbf{H}\|_F^2\ln(8n/\eta)}{9\epsilon^2}\right).$$

■

3. Complexity of sampling from $\mathcal{P}_{\hat{\mathbf{H}}_t}$

Recall that the definition of $\tilde{\mathbf{H}}_t$ is $\hat{\mathbf{H}}_t = \tilde{V}\hat{\mathbf{q}}_t$. We first evaluate the computation complexity to obtain one sample from $\mathcal{P}_{\hat{\mathbf{H}}_t}$. From Lemma 7, we know the expected runtime complexity to sample from $\mathcal{P}_{\hat{\mathbf{H}}_t}$ is $\mathcal{O}\{\frac{k\|\hat{\mathbf{q}}_t\|^2}{\|\tilde{\mathbf{V}}\hat{\mathbf{q}}_t\|^2}[S(\tilde{\mathbf{V}}) + kQ(\tilde{\mathbf{V}})]\}$. Specifically, we have

$$\|\hat{\mathbf{q}}_t\| \leqslant \|\mathbf{q}_t\| + \frac{\epsilon}{2} = \|\tilde{\mathbf{V}}^\top\mathbf{H}_t\| + \frac{\epsilon}{2} \leqslant \|\tilde{\mathbf{V}}^\top\|_2\|\mathbf{H}_t\| + \frac{\epsilon}{2}$$
$$\leqslant \sqrt{(1+\epsilon)}\|\mathbf{H}\|_F + \frac{\epsilon}{2}, \tag{D3}$$

where the first inequality employs the triangle inequality, the second inequality employs the submultiplicative property of spectral norm, and the third inequality utilizes $\|\mathbf{H}_t\| \leqslant \|\mathbf{H}\|_2\|\mathbf{B}_t\| \leqslant \|\mathbf{H}\|_F\|\mathbf{B}_t\|$ with $\|\mathbf{B}_t\| = 1$. Concurrently, we have $\|\tilde{\mathbf{V}}\mathbf{q}_t\| = \Omega(1)$. Employing Lemma 22 to quantify $S(\tilde{\mathbf{V}})$ and $Q(\tilde{\mathbf{V}})$, the complexity to obtain a sample from $\mathcal{P}_{\hat{\mathbf{H}}_t}$ is

$$\mathcal{O}\big(k(1+\epsilon)(\|\mathbf{H}\|_F^2\kappa s^2 + ks)\big).$$

With substituting s with its explicit representation in Theorem 10, the complexity is

$$\mathcal{O}\big(k(1+\epsilon)(\|\mathbf{H}\|_F^2\kappa s^2 + ks)\big) \approx \mathcal{O}\left(\frac{85^4k^5\kappa^9\ln^2(8n/\eta)\|\mathbf{H}\|_F^6}{9^2\epsilon^4}\right).$$

Following the result of the general heuristic postselection method in Theorem 11, we sample the distribution $\mathcal{P}_{\hat{\mathbf{H}}_t}$ with $N \sim \tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ times in parallel, which gives the runtime complexity

$$\mathcal{O}\left(\frac{85^4k^5\kappa^9\ln^2(8n/\eta)\|\mathbf{H}\|_F^6}{9^2\epsilon^5}\log^2(nm)\right).$$

4. Complexity of estimating ξ_t by $\hat{\xi}_t$

For the general case with $\mathbf{X} \neq \mathbf{Y}$, we should estimate the ℓ_2 norm of their project results, i.e., $\|\mathbf{H}_t\|^2 = \mathbf{H}_t^\top \mathbf{H}_t$ that $\mathbf{H}$ can either be $\mathbf{X}$ or $\mathbf{Y}$. Recall that the explicit representation of the approximated result is $\|\hat{\mathbf{H}}_t\|$ with $\|\mathbf{H}_t\| = \hat{\mathbf{q}}_t^\top \tilde{\mathbf{V}}^\top \tilde{\mathbf{V}} \hat{\mathbf{q}}_t$, where the (j, j)th entry of $\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}$ is $\|\tilde{\mathbf{v}}^{(j)}\|^2$ and the else entries are zero. An immediate observation is that $\|\mathbf{H}_t\|$ can be obtained by using the inner product subroutine in Lemma 6.

We first calculate the sample and query complexity to query the (j, j)th entry of $\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}$, namely the query and sample complexity to obtain the inner product of $\tilde{\mathbf{v}}^{(j)}$. By employing the result of the inner product subroutine, with removing $\mathbf{H}$ and setting both $\mathbf{B}_t$ and $\tilde{\mathbf{v}}^{(i)}$ as $\tilde{\mathbf{v}}^{(j)}$, we can estimate $\tilde{\mathbf{v}}^{(j)\top} \tilde{\mathbf{v}}^{(j)}$ to precision ϵ with probability at least $1 - \delta$ in time

$$\begin{aligned} &\mathcal{O}\left(\frac{\|\tilde{\mathbf{v}}^{(j)}\|^2}{\epsilon^2}[Q(\tilde{\mathbf{v}}^{(i)})] \log\left(\frac{1}{\delta}\right)\right) \\ &\leqslant \mathcal{O}\left(\left[\frac{(1+\epsilon)}{\epsilon^2}(s)\right] \log\left(\frac{1}{\delta}\right)\right) \\ &\approx \mathcal{O}\left(\frac{85^2 k^2 \kappa^4 \ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^4} \log\left(\frac{1}{\delta}\right)\right), \end{aligned} \tag{D4}$$

where we use Lemma 22 to get $Q(\tilde{\mathbf{v}}^{(i)}) = \mathcal{O}(s)$ and the inequality comes from $\|\tilde{\mathbf{v}}^{(i)}\| \leqslant (1+\epsilon)$ in Eq. (C6). We store k nonzero entries of $\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}$ in memory.

We next use the inner product subroutine to obtain $\|\hat{\mathbf{H}}_t\|$ with $\|\hat{\mathbf{H}}_t\| = \hat{\mathbf{q}}_t^\top \tilde{\mathbf{V}}^\top \tilde{\mathbf{V}} \hat{\mathbf{q}}_t$. Since both $\hat{\mathbf{q}}_t$ and $\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}$ are stored in memory, we have $L(\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}) = \mathcal{O}(k)$ and $Q(\hat{\mathbf{q}}_t) = \mathcal{O}(1)$. Following the result of Lemma 6, we can estimate $\|\hat{\mathbf{H}}_t\|$ to precision ϵ with probability at least $1 - \delta$ with runtime complexity

$$\begin{aligned} &\mathcal{O}\left(\frac{\|\tilde{\mathbf{V}}^\top \tilde{\mathbf{V}}\|_F \|\mathbf{q}_t\|^2}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right) \\ &\leqslant \mathcal{O}\left(\frac{\sqrt{k(1+\epsilon)^2}\sqrt{k(1+\epsilon)\|\mathbf{H}\|_F}}{\epsilon^2}(k) \log\left(\frac{1}{\delta}\right)\right), \end{aligned} \tag{D5}$$

where the inequality comes from $\|\tilde{\mathbf{v}}^{(i)}\| \leqslant (1+\epsilon)$ in Eq. (C6) and Eq. (D3).

Combining Eq. (D4) and Eq. (D5), the computation complexity to obtain the ℓ_2 norm of $\|\hat{\mathbf{H}}_t\|$ is

$$\mathcal{O}\left(\frac{85^2 k^2 \kappa^4 \ln(8n/\eta)\|\mathbf{H}\|_F^2}{9\epsilon^4} \log\left(\frac{1}{\delta}\right)\right). \tag{D6}$$

5. The overall complexity of our algorithm

An immediate observation of the above four parts is that the query complexity and runtime complexity of our algorithm is denominated by the complexity of finding $\tilde{V}$, i.e., $\mathcal{O}(s^3) = \tilde{\mathcal{O}}(\frac{85^6 k^6 \kappa^{12}\|\mathbf{H}\|_F^6}{9^3\epsilon^6})$. Since $\mathbf{H}$ is a general setting that can either be $\mathbf{X}$ or $\mathbf{Y}$, the runtime complexity for our algorithm is

$$\max\left\{\tilde{\mathcal{O}}\left(\frac{85^6 k_X^6 \kappa_X^{12}\|\mathbf{X}\|_F^6}{9^3\epsilon^6}\right), \tilde{\mathcal{O}}\left(\frac{85^6 k_{X,t}^6 \kappa_Y^{12}\|\mathbf{Y}\|_F^6}{9^3\epsilon^6}\right)\right\}.$$

■

[1] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, Berlin, 2006).

[2] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence* (CRC Press, Boca Raton, FL, 2010).

[3] S. E. Shimony, Finding maps for belief networks is np-hard, Artif. Intell. **68**, 399 (1994).

[4] N. D. Lawrence, Gaussian process latent variable models for visualisation of high dimensional data, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2004), pp. 329–336.

[5] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature **401**, 788 (1999).

[6] J. C. Loehlin, *Latent Variable Models: An Introduction to Factor, Path, and Structural Analysis* (Lawrence Erlbaum Associates, Mahwah, NJ, 1987).

[7] R. Salakhutdinov and A. Mnih, Probabilistic matrix factorization, in *Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07* (Curran Associates Inc., Red Hook, NY, 2007), pp. 1257–1264.

[8] M. N. Schmidt, O. Winther, and L. K. Hansen, Bayesian non-negative matrix factorization, in *Proceedings of the International Conference on Independent Component Analysis and Signal Separation* (Springer, Berlin, 2009), pp. 540–547.

[9] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, in *Readings in Computer Vision* (Elsevier, Amsterdam, 1987), pp. 564–584.

[10] T. Zhou, J. A. Bilmes, and C. Guestrin, Divide-and-conquer learning by anchoring a conical hull, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2014), pp. 1242–1250.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc. Ser. B **39**, 1 (1977).

[12] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. **26**, 97 (2013).

[13] E. Tang, A quantum-inspired classical algorithm for recommendation systems, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM Press, New York, 2019), pp. 217–228.

[14] T. Zhou, W. Bian, and D. Tao, Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions, in *Proceedings of the 2013 IEEE 13th International Conference on Data Mining (ICDM'13)*, (IEEE, Los Alamitos, CA, 2013), pp. 917–926.

[15] A. Gilyén, S. Lloyd, and E. Tang, Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension, arXiv:1811.04909.

[16] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).

[17] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu, Quantum algorithms and lower bounds for convex optimization, Quantum **4**, 221 (2020).

[18] Y. Du, M.-H. Hsieh, and D. Tao, Efficient online quantum generative adversarial learning algorithms with applications, arXiv:1904.09602.

[19] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM Press, New York, 2019), pp. 193–204.

[20] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).

[21] A. Kapoor, N. Wiebe, and K. Svore, Quantum perceptron models, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2016), pp. 3999–4007.

[22] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. **10**, 631 (2014).

[23] I. Kerenidis and A. Prakash, Quantum recommendation systems, in *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS'17)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018).

[24] Y. Du, T. Liu, Y. Li, R. Duan, and D. Tao, Quantum divide-and-conquer anchoring for separable non-negative matrix factorization, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (AAAI Press, Stockholm, 2020), pp. 2093–2099.

[25] E. Aïmeur, G. Brassard, and S. Gambs, Quantum clustering algorithms, in *Proceedings of the 24th International Conference on Machine Learning* (ACM Press, New York, 2007), pp. 1–8.

[26] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411.

[27] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning, Quantum Inf. Comput. **15**, 316 (2015).

[28] N.-H. Chia, A. Gilyén, T. Li, H.-H. Lin, E. Tang, and C. Wang, Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning, in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (ACM Press, New York, NY, 2020), pp. 387–400.

[29] D. Jethwani, F. Le Gall, and S. K. Singh, Quantum-inspired classical algorithms for singular value transformation, arXiv:1910.05699.

[30] I. Kerenidis and A. Prakash, Quantum gradient descent for linear systems and least squares, Phys. Rev. A **101**, 022316 (2020).

[31] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, q-means: A quantum algorithm for unsupervised machine learning, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2019), pp. 4134–4144.

[32] F. G. S. L. Brandão, A. Kalev, T. Li, C. Y.-Y. Lin, K. M. Svore, and X. Wu, Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP'19)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019).

[33] T. Li, S. Chakrabarti, and X. Wu, Sublinear quantum algorithms for training linear and kernel-based classifiers, in *Proceedings of the International Conference on Machine Learning* (ACM Press, New York, 2019), pp. 3815–3824.

[34] S. Geman and D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-6**, 721 (1984).

[35] R. Salakhutdinov and A. Mnih, Probabilistic matrix factorization, in *Proceedings of the 20th International Conference on Neural Information Processing Systems* (*NIPS'07*) (Curran Associates Inc., Red Hook, NY, 2007), pp. 1257–1264.

[36] M. Belkin and K. Sinha, Polynomial learning of distribution families, in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA, 2010), pp. 103–112.

[37] N.-H. Chia, T. Li, H.-H. Lin, and C. Wang, Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches, arXiv:1901.03254.

[38] N.-H. Chia, H.-H. Lin, and C. Wang, Quantum-inspired sublinear classical algorithms for solving low-rank linear systems, arXiv:1811.04852.

[39] E. Tang, Quantum-inspired classical algorithms for principal component analysis and supervised clustering, arXiv:1811.00414.

[40] C. Ding, T.-Y. Bao, and H.-L. Huang, Quantum-inspired support vector machine, arXiv:1906.08902.

[41] J. M. Arrazola, A. Delgado, B. R. Bardhan, and S. Lloyd, Quantum-inspired algorithms in practice, arXiv:1905.10415.

[42] Y. Liu and S. Zhang, Fast quantum algorithms for least squares regression and statistic leverage scores, Theor. Comput. Sci. **657**, 38 (2017).

[43] D. Donoho and V. Stodden, When does non-negative matrix factorization give a correct decomposition into parts?, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2004), pp. 1141–1148.

[44] S. Arora, R. Ge, R. Kannan, and A. Moitra, Computing a nonnegative matrix factorization—provably, SIAM J. Comput. **45**, 1582 (2016).

[45] X. Yu, W. Bian, and D. Tao, Scalable completion of nonnegative matrix with separable structure, in *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (AAAI Press, Phoenix, Arizona, 2016), pp. 2279–2285.

[46] A. Kumar, V. Sindhwani, and P. Kambadur, Fast conical hull algorithms for near-separable non-negative matrix factorization, in *Proceedings of the International Conference on Machine Learning* (ACM Press, New York, 2013), pp. 231–239.

[47] A. W. Van Der Vaart and J. A. Wellner, Weak convergence, in *Weak Convergence and Empirical Processes* (Springer, Berlin, 1996), pp. 16–28.

[48] R. A. Horn and C. R. Johnson, *Matrix Analysis* (Cambridge University Press, Cambridge, UK, 2012).

[49] R. Kannan and S. Vempala, Randomized algorithms in numerical linear algebra, Acta Numer. **26**, 95 (2017).