

# RANDOMIZED ALGORITHMS FOR SYMMETRIC NONNEGATIVE MATRIX FACTORIZATION \*

KOBY HAYASHI<sup>†</sup>, SINAN G. AKSOY<sup>‡</sup>, GREY BALLARD<sup>§</sup>, AND HAESUN PARK<sup>†</sup>

**Abstract.** Symmetric Nonnegative Matrix Factorization (SymNMF) is a technique in data analysis and machine learning that approximates a symmetric matrix with a product of a nonnegative, low-rank matrix and its transpose. To design faster and more scalable algorithms for SymNMF we develop two randomized algorithms for its computation. The first algorithm uses randomized matrix sketching to compute an initial low-rank approximation to the input matrix and proceeds to rapidly compute a SymNMF of the approximation. The second algorithm uses randomized leverage score sampling to approximately solve constrained least squares problems. Many successful methods for SymNMF rely on (approximately) solving sequences of constrained least squares problems. We prove theoretically that leverage score sampling can approximately solve nonnegative least squares problems to a chosen accuracy with high probability. Additionally, we prove sampling complexity results for previously proposed hybrid sampling techniques which deterministically include high leverage score rows. This hybrid scheme is crucial for obtaining speeds ups in practice. Finally we demonstrate that both methods work well in practice by applying them to graph clustering tasks on large real world data sets. These experiments show that our methods approximately maintain solution quality and achieve significant speed ups for both large dense and large sparse problems.

**Key words.** Nonnegative Matrix Factorization, Symmetric Nonnegative Matrix Factorization, Leverage Scores, matrix sketching, randomized numerical linear algebra (RandNLA)

**AMS subject classifications.** 05C50, 65F20, 65F55, 65F50, 90C20

**1. Introduction.** We propose the first randomized algorithms for Symmetric Nonnegative Matrix Factorization (SymNMF). Nonnegative Matrix Factorization (NMF) is an important method in data analysis with applications to data visualization, text mining, feature learning, information fusion and more [38, 35, 57, 31, 16]. SymNMF is a variant of NMF where the input matrix is symmetric and the output low-rank approximation is also constrained to be symmetric [35, 61]. Applications of SymNMF include (hyper)graph clustering, image segmentation, and information fusion [55, 15, 27, 7, 8]. Several randomized algorithms for nonsymmetric NMF have been previously proposed and shown to be effective for dense and small sparse problems [54, 51, 19], but as far as we are aware there is no prior work on randomized algorithms for SymNMF. Along the way we also prove two interesting results generalizing existing guarantees for leverage score sampling for overdetermined least squares problems to all convex overdetermined least squares problems (e.g. nonnegative least squares) and prove theoretical guarantees for hybrid leverage score sampling schemes applied to such problems. Our contributions include:

- a randomized algorithm for SymNMF we call “Low-rank Approximated Input SymNMF” (LAI-SymNMF),
- a randomized algorithm based on leverage score sampling for least squares

---

\*Created on 08/12/2024.

**Funding:** Koby Hayashi acknowledges support from the United States Department of Energy through the Computational Sciences Graduate Fellowship (DOE CSGF) under grant number: DE-SC0020347. The authors would like to acknowledge the support provided by the National Science Foundation through grants OAC-2106920 and CCF-1942892. Information Release PNNL-SA-193926.

<sup>†</sup>School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA ([khayashi9@gatech.edu](mailto:khayashi9@gatech.edu), [hpark@cc.gatech.edu](mailto:hpark@cc.gatech.edu)).

<sup>‡</sup>Pacific Northwest National Laboratory, Seattle, WA, USA ([sinan.aksoy@pnnl.gov](mailto:sinan.aksoy@pnnl.gov))

<sup>§</sup>Dept. of Computer Science, Wake Forest University, Winston-Salem, NC, USA ([ballard@wfu.edu](mailto:ballard@wfu.edu))

Symbols	Meaning	Symbols	Meaning
$\mathbf{I}$	Identity Matrix	$\mathbf{a}_i$	$i$ th column vector of $\mathbf{A}$
$\Omega$	Gaussian Matrix	$l_i(\mathbf{A})$	the $i$ th leverage score of $\mathbf{A}$
$\otimes$	Kronecker Product	$p_i$	$i$ th leverage score sampling probability
$\mathbf{S}$	Sketching Matrix	$[n]$	index set 1 to $n$
$[\mathbf{A}]_+$	Proj. to nonnegative orthant	$\mathcal{A}$	A set, Euler script
$\mathbf{X}$	NMF Data Matrix	$\mathbf{A}$	A matrix, bold-uppercase
$\mathbf{W}$	Left NMF Factor	$\mathbf{a}$	A vector, bold-lowercase
$\mathbf{H}$	Right NMF Factor	$ \cdot $	Absolute value or cardinality
$\mathbf{A}^\top$	Transposition	$\mathbb{R}_+$	Nonnegative Real Numbers
$\mathbf{Q}_A$	Orthonormal basis for range of $\mathbf{A}$	$\kappa(\mathbf{A})$	Condition Number of $\mathbf{A}$
$\ \cdot\ _F$	Frobenius norm	$\ \cdot\ _2$	2-norm
$\sigma_i(\mathbf{A})$	$i$ th largest singular value of $\mathbf{A}$	$\sigma_{\max}(\mathbf{A}), \sigma_{\min}(\mathbf{A})$	largest and smallest singular value

Table 1: Notation

problems we call LvS-SymNMF,

- novel theoretical analysis of leverage score sampling for the *Nonnegative Least Squares* problem and more generally convex least squares problems,
- theoretical analysis of a hybrid leverage score sampling scheme for convex least squares problem,
- and experiments on two large, real world clustering tasks.

The rest of the paper is organized as follows. Section 2, which discusses background material including non-randomized SymNMF algorithms, reviews existing randomized NMF methods and other related work such as randomized methods for other low-rank matrix and tensor decompositions. Section 3 introduces our first proposed algorithm LAI-SymNMF. LAI-SymNMF uses randomized methods to rapidly compute an initial, unconstrained low-rank approximation before proceeding to compute an NMF of this LAI. Section 4 presents a row-sampling algorithm called LvS-SymNMF. This method solves a sequence of nonnegative least squares (NLS) problems using a technique called leverage score sampling to accelerate the solver. Additionally, we use a hybrid approach from [36] that involves both deterministic and randomized sampling based on the leverage scores. Our novel theoretical analysis of this hybrid strategy gives the sample complexity needed to achieve an accuracy guarantee relative to the NLS residual, with high probability, and we empirically show its advantage over purely randomized sampling. Section 5 presents experimental results for the proposed algorithms on two real world data sets. Each data set is represented as a graph and the SymNMF output is used to cluster the vertices. Both methods achieve significant speed ups over deterministic methods ranging from 5 to  $7.5\times$  and are able to maintain accuracy in terms of normalized residual norms and cluster quality.

**2. Preliminaries.** We begin by briefly discussing the NMF and SymNMF problems followed by an introduction to various methods for computing NMF’s and SymNMF’s. Then we discuss the “Randomized Range Finder” and leverage score sampling for least squares problems. The section also contains a statement of Theorem 2.1 which provides novel bounds for Leverage Score sampling applied to Nonnegative Least Squares Problems. Various notation is given in Table 1.

Standard NMF is formulated as

$$(2.1) \quad \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{X} - \mathbf{WH}^\top\|_F$$

where  $\mathbf{X} \in \mathbb{R}^{m \times n}$ ,  $\mathbf{W} \in \mathbb{R}_+^{m \times k}$ , and  $\mathbf{H} \in \mathbb{R}_+^{n \times k}$ . The notation  $\mathbb{R}_+$  denotes the

nonnegative orthant and  $k$  is the desired reduced rank, given as an input, and usually  $k \ll \min(m, n)$ . While  $\mathbf{X}$  is often also nonnegative, it is not strictly required. Since NMF is a non-convex optimization problem, optimization algorithms often find a local minimum at best. The works [23, 32] give a comprehensive discussion of NMF.

When  $\mathbf{X}$  is symmetric,  $\mathbf{X} = \mathbf{X}^\top$ , it is often desired the two low-rank factors in [Equation \(2.1\)](#) be the same [35, 52, 61, 34]. This problem is called Symmetric NMF (SymNMF) and its objective function is expressed as

$$(2.2) \quad \min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{H}\mathbf{H}^\top\|_F.$$

SymNMF has found applications to (hyper)graph clustering, image segmentation, and community detection in social networks [35, 27, 17]. For a relationship between SymNMF and spectral clustering see [15].

**2.1. Algorithms for SymNMF.** Many algorithms for solving SymNMF shown in [Equation \(2.2\)](#) have been proposed. Broadly these methods can be put into two categories: Alternating Updating (AU) methods and all-at-once optimization methods. AU methods alternate between updating a subset of the variables while holding others fixed, eventually iterating through the entire subset of variables. All-at-once methods update all of the variables simultaneously.

AU methods for SymNMF include symmetrically regularized Alternating Nonnegative Least Squares (ANLS) [35], symmetrically regularized Hierarchical Least Squares (HALS) [61], Cyclic Coordinate Descent (CCD) [52], and Progressive Hierarchical Alternating Least Squares (PHALS) [28]. All-at-once methods include Projected Gradient Descent (PGD), Projected Newton-like update [35], and Projected Gauss-Newton with Conjugate Gradients (PGNCG) [22]. We focus on methods based on regularized ANLS and the PGNCG algorithm. This is because the ANLS method is generally superior to both PGD and the Newton-like method [35], and the CCD method tends to be unsuitable for large data sets as it sequentially iterates over the elements of  $\mathbf{H}$ . We now present the methods based on regularized ANLS and the PGNCG algorithm.

### 2.1.1. Symmetrically Regularized Alternating Nonnegative Least

**Squares for SymNMF.** The regularized ANLS and HALS methods for SymNMF are based on solving a surrogate problem of the form

$$(2.3) \quad \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{X} - \mathbf{WH}^\top\|_F^2 + \alpha \|\mathbf{W} - \mathbf{H}\|_F^2,$$

where  $\mathbf{W}$  and  $\mathbf{H}$  are forced to be close to each other in Frobenius norm using a large value of  $\alpha$ , thus guiding the iterates towards a symmetric approximation [35]. The authors of [61] show that under mild assumptions the critical points of [Equation \(2.2\)](#) and [Equation \(2.3\)](#) are the same and argue that this makes [Equation \(2.3\)](#) an appropriate surrogate for [Equation \(2.2\)](#). [Equation \(2.3\)](#) can be iteratively approximated by using the ANLS method for updating  $\mathbf{W}$  and  $\mathbf{H}$  as in the following two equations :

$$(2.4) \quad \min_{\mathbf{W} \geq 0} \left\| \begin{bmatrix} \mathbf{H} \\ \sqrt{\alpha} \mathbf{I} \end{bmatrix} \mathbf{W}^\top - \begin{bmatrix} \mathbf{X} \\ \sqrt{\alpha} \mathbf{H}^\top \end{bmatrix} \right\|_F \text{ and } \min_{\mathbf{H} \geq 0} \left\| \begin{bmatrix} \mathbf{W} \\ \sqrt{\alpha} \mathbf{I} \end{bmatrix} \mathbf{H}^\top - \begin{bmatrix} \mathbf{X} \\ \sqrt{\alpha} \mathbf{W}^\top \end{bmatrix} \right\|_F.$$

This approach enables using many of the tools for standard NMF for the SymNMF problem. For example, the two equations shown in [Equation \(2.4\)](#) can be solved as Nonnegative Least Squares (NLS) problems.

The ANLS method is generally superior to PGD and the Newton-like method [35]. This is mainly because PGD suffers from slow convergence and the Newton-like method is expensive for even moderately sized problems due to the need for approximate Hessian inversion. To solve the ANLS formulation we use the Block Principle Pivoting (BPP) solver from [33].

**2.1.2. Hierarchical Alternating Least Squares for SymNMF.** The HALS method for NMF was developed in [24, 9]. A method that uses the HALS framework for optimizing Equation (2.3) was proposed in [61]. The update rules are given by the following two equations

$$(2.5) \quad \mathbf{w}_i = \max \left( \frac{(\mathbf{R}_i + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha}, 0 \right) \text{ and } \mathbf{h}_i = \max \left( \frac{(\mathbf{R}_i + \alpha \mathbf{I}) \mathbf{w}_i}{\|\mathbf{w}_i\|_2^2 + \alpha}, 0 \right).$$

Here  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k]$ ,  $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_k]$  are the columns of  $\mathbf{W}$  and  $\mathbf{H}$  and  $\mathbf{R}_i = \mathbf{X} - \sum_{j \neq i, j=1}^k \mathbf{w}_j \mathbf{h}_j^\top$ . Following these rules, the columns of  $\mathbf{W}$  and  $\mathbf{H}$  are updated as pairs in sequence as  $\mathbf{w}_1, \mathbf{h}_1, \dots, \mathbf{w}_i, \mathbf{h}_i, \dots, \mathbf{w}_k, \mathbf{h}_k$ . This method is less efficient as it relies on the matrix  $\mathbf{R}_i$ , as in Equation (2.5). To make the updates efficient we modify the update rule following algorithms from [24] for more efficient HALS updates to

$$(2.6) \quad \mathbf{w}_i \leftarrow \left( \frac{(\mathbf{X} - \mathbf{W}\mathbf{H}^\top + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} + \frac{\|\mathbf{h}_i\|_2^2}{\|\mathbf{h}_i\|_2^2 + \alpha} \mathbf{w}_i \right)_+,$$

$$(2.7) \quad \mathbf{h}_i \leftarrow \left( \frac{(\mathbf{X} - \mathbf{H}\mathbf{W}^\top + \alpha \mathbf{I}) \mathbf{w}_i}{\|\mathbf{w}_i\|_2^2 + \alpha} + \frac{\|\mathbf{w}_i\|_2^2}{\|\mathbf{w}_i\|_2^2 + \alpha} \mathbf{h}_i \right)_+$$

which are mathematically equivalent to those in Equation (2.5). These rules can be seen as combining the update order for HALS from [9] and regularization, such as used in [10]. A derivation can be found in the Appendix A. This formulation allows for the updates of all the columns of  $\mathbf{W}$  then all the columns of  $\mathbf{H}$  or vice versa. As discussed in [24], choosing to update all  $\mathbf{w}_i$ 's followed by all  $\mathbf{h}_i$ 's allows for the products  $\mathbf{W}^\top \mathbf{X}$  and  $\mathbf{X}\mathbf{H}$  to be computed and reused through a single sweep of updates over all columns of  $\mathbf{W}$  and  $\mathbf{H}$  resulting in better computational efficiency in practice. To illustrate this, consider the update for  $\mathbf{w}_i$ . The bulk of the computation is needed for computing  $(\mathbf{X} - \mathbf{W}\mathbf{H}^\top + \alpha \mathbf{I}) \mathbf{h}_i$  which is the  $i$ th column of  $(\mathbf{X}\mathbf{H} - \mathbf{W}\mathbf{H}^\top \mathbf{H} + \alpha \mathbf{H})$ , where the product  $\mathbf{X}\mathbf{H}$  will not change as each  $\mathbf{w}_i$  is updated. The same applies for the columns of  $\mathbf{H}$ . Overall our proposed updates shown in Equation (2.6) are more memory efficient and more computationally efficient by a factor of 2.

**2.1.3. Projected Gauss-Newton with Conjugate Gradients for SymNMF.** The algorithm PGNCG-SymNMF was proposed for efficiently computing SymNMF in highly parallel computing environments [22]. It is an all-at-once method and uses the Projected Gauss-Newton method to directly optimize the SymNMF objective, Equation (2.2). The main computational load lies in solving a least squares problem of the form

$$\min_{\mathbf{p}} \|\mathbf{J}\mathbf{p} + \mathbf{r}\|_2$$

for a search direction  $\mathbf{p}$  at every iteration. The matrix  $\mathbf{J}$  and the vector  $\mathbf{r}$  are the Jacobian and residual of Equation (2.2) respectively. A solution to this LS problem is then approximated using the Conjugate Gradient (CG) method on the Normal Equations :  $\mathbf{J}^\top \mathbf{J}\mathbf{p} = -\mathbf{J}^\top \mathbf{r}$ . A core computational kernel of the CG method is computing

**Algorithm RRF** : Randomized Range Finder

---

**Require:** : data matrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$ , target rank  $r$ , oversampling parameter  $\rho$ , and an exponent  $q$

**Ensure:**  $\mathbf{Q}_Y \in \mathbb{R}^{m \times (r+\rho)}$ , is an approximate orthonormal basis for the leading column span of  $\mathbf{X}$ :

- 1: **function**  $[\mathbf{Q}_Y] = \text{RRF}(\mathbf{X}, r, \rho, q)$
  - 2:    $l := r + \rho$                        $\triangleright l$  is the rank of the approximation being computed
  - 3:   Draw a Gaussian Random matrix  $\Omega \in \mathbb{R}^{n \times l}$
  - 4:   Compute  $\mathbf{Y} := (\mathbf{X}\mathbf{X}^T)^q \mathbf{X}\Omega \in \mathbb{R}^{n \times l}$
  - 5:   Compute  $\mathbf{Y} = \mathbf{Q}_Y \mathbf{R}_Y$  a thin-QR Decomposition of  $\mathbf{Y}$  where  $\mathbf{Q}_Y \in \mathbb{R}^{n \times l}$  and  $\mathbf{R}_Y \in \mathbb{R}^{l \times l}$
  - 6: **end function**
- 

matrix vector products with the matrix  $\mathbf{J}^T \mathbf{J}$ . Fortunately, the Jacobian  $\mathbf{J}$  has the Kronecker product form  $\mathbf{J} = -(\mathbf{H} \otimes \mathbf{I}_m) - \mathbf{P}_{m,m}(\mathbf{H} \otimes \mathbf{I}_m)$ , where  $\mathbf{P}_{m,m}$  is the perfect shuffle or “vec” permutation, which can be used to efficiently apply  $\mathbf{J}$  to a vector. Additionally the vector  $\mathbf{g} = \mathbf{J}^T \mathbf{r}$  has the form  $\mathbf{g} = -2\text{vec}(\mathbf{X}\mathbf{H} - \mathbf{H}(\mathbf{H}^T \mathbf{H}))$ , which is typically the main computational bottleneck and requires the matrix multiplications  $\mathbf{X}\mathbf{H}$  and  $\mathbf{H}^T \mathbf{H}$ . See [22] for details.

The PGNCG method is competitive with the ANLS and CCD method for Sym-NMF [22]. The PGNCG method generally converges much faster than PGD as it approximates second-order derivatives and does not suffer from large computational complexity, as the Newton-like algorithm does, due to the exploitation of the Jacobian’s structure for use in the CG iterations.

**2.2. Sketching in Numerical Linear Algebra.** Randomized Numerical Linear Algebra (RndNLA) is an important area of research with practical applications in finding fast approximate solutions to linear systems, least squares problems, eigenvalue problems, among others. Surveys on this topic include [26, 44]. There are two main tools we will use from the RndNLA literature. The first is the Randomized Range Finder (RRF) [26] which has many applications in RandNLA such as computing approximate, truncated Singular Value Decompositions (SVD’s) and Symmetric Eigenvalue Decompositions (EVD’s). The second is leverage score sampling for approximately solving least squares problems [56, 42].

**2.2.1. Randomized Range Finder.** The RRF is a method for finding an approximate orthonormal basis for the range space of a matrix. It is the foundation for many randomized methods in RandNLA, such as computing an approximate, truncated SVD in a randomized way [26]. An algorithm outline for the RRF is given in Algorithm RRF. Parameters of the RRF are the target rank  $r$ , a column oversampling parameter  $\rho$ , and  $q$ , the number of power iterations to perform. The computational complexity of the RRF is  $O(qmn + ml^2)$  where  $l = r + \rho$ . The approximate output from the RRF is often used to compute a “QB-Decomposition”. If a matrix  $\mathbf{X}$  is input to the RRF and a matrix  $\mathbf{Q}_X$  is output, then  $\mathbf{Q}_X \mathbf{Q}_X^T \mathbf{X} = \mathbf{Q}_X \mathbf{B}_X$ , where  $\mathbf{B}_X = \mathbf{Q}_X^T \mathbf{X}$ , is called a QB-Decomposition of  $\mathbf{X}$ .

For a symmetric input, an approximate eigenvalue decomposition can also be obtained by using the approximate basis from the RRF. This procedure is shown in Algorithm Apx-EVD, which stands for approximate eigenvalue decomposition. More details and references can be found in [26].

---

**Algorithm Apx-EVD** : Approximate Truncated Eigenvalue Decomposition of Symmetric Matrix

---

**Require:** : symmetric matrix  $\mathbf{X} \in \mathbb{R}^{m \times m}$ , target rank  $r$ , oversampling parameter  $\rho$ , and exponent  $q$   
**Ensure:**  $\mathbf{U} \in \mathbb{R}^{m \times (r+\rho)}$  and  $\mathbf{\Lambda} \in \mathbb{R}^{(r+\rho) \times (r+\rho)}$  where  $\mathbf{X} \approx \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$  is an approximate EVD of  $\mathbf{X}$ .

```

1: function  $[\mathbf{U}, \mathbf{\Lambda}] = \text{APX-EVD}(\mathbf{X}, r, \rho, q)$ 
2:    $l := r + \rho$             $\triangleright l$  is the rank of the approximation being computed
3:    $\mathbf{Q}_X := \text{RRF}(\mathbf{X}, r, \rho, q)$ , where  $\mathbf{Q}_X \in \mathbb{R}^{m \times l}$ 
4:   Compute  $\mathbf{T} := \mathbf{Q}_X^\top \mathbf{X} \mathbf{Q}_X \in \mathbb{R}^{l \times l}$ 
5:   Compute  $\mathbf{T} = \mathbf{Q}_T \mathbf{\Lambda} \mathbf{Q}_T^\top$  an eigenvalue decomposition of  $\mathbf{T}$  where  $\mathbf{Q}_T \in \mathbb{R}^{l \times l}$ 
   and  $\mathbf{\Lambda} \in \mathbb{R}^{l \times l}$ 
6:   Compute  $\mathbf{U} := \mathbf{Q}_X \mathbf{Q}_T \in \mathbb{R}^{m \times l}$ 
7: end function
```

---

**2.2.2. Leverage Score Sampling for Ordinary Least Squares.** Another approach that we will use from RandNLA is sketching for ordinary least squares (OLS) problems, specifically, leverage score sampling for OLS problems. A standard OLS or  $l_2$ -regression problem is

$$(2.8) \quad \min_{\mathbf{x} \in \mathbb{R}^k} \|\mathbf{Ax} - \mathbf{b}\|_2$$

where  $\mathbf{A} \in \mathbb{R}^{m \times k}$ ,  $\mathbf{x} \in \mathbb{R}^k$ , and  $\mathbf{b} \in \mathbb{R}^m$ . Our focus will be on overdetermined OLS problems, where  $m \gg k$ , and  $\mathbf{A}$  has full rank.

The sketch and solve paradigm [44] for OLS methods takes the form

$$(2.9) \quad \hat{\mathbf{x}}_{ols} = \arg \min_{\mathbf{x}} \|\mathbf{S}\mathbf{Ax} - \mathbf{S}\mathbf{b}\|_2,$$

where  $\mathbf{S} \in \mathbb{R}^{s \times m}$ , with  $s \ll m$ , is called a *sketching* matrix. Computational savings come from the fact that one can now solve the smaller problem in Equation (2.9) as opposed to the full sized problem in Equation (2.8).

There are many ways to generate the sketching matrix  $\mathbf{S}$ . We focus on when  $\mathbf{S}$  is a row-sampling matrix generated according to the leverage score distribution of  $\mathbf{A}$ . Leverage score sampling is a well-studied method for sketching OLS problems [42]. In this method the leverage scores (see Equation (2.10)) are used to define a probability distribution over the rows of the matrix  $\mathbf{A}$ . That is, some number of rows, say  $s$  rows, of the matrix  $\mathbf{A}$  are sampled with replacement with probability proportional to the value of their leverage scores. The leverage score of the  $i$ th row of a matrix  $\mathbf{A}$  is defined as

$$(2.10) \quad l_i(\mathbf{A}) = \|\mathbf{Q}_A[i, :] \|_2^2$$

where the matrix  $\mathbf{Q}_A$  is any orthonormal basis for the column space of  $\mathbf{A}$  and  $\mathbf{Q}_A[i, :]$  is the  $i$ th row of  $\mathbf{Q}_A$ . For example the matrix  $\mathbf{U}_A$ , where  $\mathbf{U}_A \mathbf{\Sigma}_A \mathbf{V}_A^\top = \mathbf{A}$  is a thin SVD of  $\mathbf{A}$ , can be used to calculate the leverage scores. These values are normalized into probabilities  $p_i = \frac{l_i(\mathbf{A})}{\|\mathbf{Q}_A\|_F^2}$ . Using the  $p_i$ 's,  $s$  samples are drawn with replacement and the matrix  $\mathbf{S} \in \mathbb{R}^{s \times m}$  is formed as

$$(2.11) \quad \mathbf{S}_{ji} = \begin{cases} \frac{1}{\sqrt{sp_i}}, & \text{if row } i \text{ was drawn as the } j\text{th sample} \\ 0 & \end{cases} .$$

Due to the special form of  $\mathbf{S}$ ,  $\mathbf{SA}$  does not require matrix-matrix multiplication but only row selection and scaling. Computing the leverage scores of  $\mathbf{A}$  via a full matrix factorization, such as QR or SVD, costs  $O(mk^2)$ . This makes solving the smaller problem in [Equation \(2.9\)](#) just as expensive as the original LS problem in [Equation \(2.8\)](#), in the case of a single right hand side (RHS). To deal with this, schemes for quickly computing approximate leverage scores have been proposed [\[13\]](#). Additionally, sometimes special structures in the coefficient matrix,  $\mathbf{A}$ , can be exploited to obtain fast leverage score estimates [\[36, 6\]](#).

When leverage score sampling is used, with an appropriate number of samples  $s$ , the solution to the sampled problem in Eq. [\(2.9\)](#) satisfies the bound

$$(2.12) \quad \|\hat{\mathbf{x}}_{ols} - \mathbf{x}_{ols}\|_2 \leq \sqrt{\epsilon_r} \frac{\|\mathbf{r}_{ols}\|_2}{\sigma_{min}(\mathbf{A})}$$

with probability  $1 - \delta$  where  $\mathbf{r}_{ols} = \mathbf{Ax}_{ols} - \mathbf{b}$  and  $\mathbf{x}_{ols}$  is the minimizer of Eq. [\(2.8\)](#). Additionally,  $\delta$  and  $\epsilon_r$  are values between 0 and 1 which control the failure probability and approximation error respectively. The number of samples  $s$  is a function of these values and lower values of  $\delta$  and  $\epsilon_r$  incur a higher value of  $s$ . That is for lower failure probability and/or smaller error one must take more samples. Details can be found in [\[37\]](#).

**2.2.3. Leverage Score Sampling for Nonnegative Least Squares.** In order to apply leverage score sampling for Least Squares problems to NMF we extend results for Leverage Score Sampling for OLS problems to Nonnegative Least Squares Problems. The Nonnegative Least Squares problem has the general form

$$(2.13) \quad \min_{\mathbf{x} \geq 0} \|\mathbf{Ax} - \mathbf{b}\|_2.$$

NLS problems are quite different from OLS problems as they do not yield a closed form solution. Despite this we are able to prove an analogous bound for sampling NLS problems with leverage scores. The statement of this result is given by the following theorem

**THEOREM 2.1.** *Let  $\mathbf{x}_{nls} = \arg \min_{\mathbf{x} \geq 0} \|\mathbf{Ax} - \mathbf{b}\|_2$  be a NLS solution where  $\mathbf{A} \in \mathbb{R}^{m \times k}$ ,  $m > k$ , and  $\text{rank}(\mathbf{A}) = k$ . Also let  $\mathbf{S} \in \mathbb{R}^{s \times m}$  be a leverage score sampling matrix for  $\mathbf{A}$  as in Eq. [\(2.11\)](#) with  $s$  samples satisfying*

$$s \geq k \max(C \log(k/\delta), 1/(\delta\epsilon_r)) \quad \text{where } C = 144/(1 - \sqrt{2})^2$$

for some  $\epsilon_r, \delta \in (0, 1)$ . Also let  $\hat{\mathbf{x}}_{nls} = \arg \min_{\mathbf{x} \geq 0} \|\mathbf{SAx} - \mathbf{Sb}\|_2$  be the sampled NLS solution. Then with probability  $1 - \delta$ , the following holds :

$$\|\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls}\|_2 \leq \sqrt{\epsilon_r} \frac{\|\mathbf{r}_{nls}\|_2}{\sigma_{min}(\mathbf{A})}$$

where  $\mathbf{r}_{nls} = \mathbf{Ax}_{nls} - \mathbf{b}$  and  $\sigma_{min}(\mathbf{A})$  is the minimum singular value of  $\mathbf{A}$ <sup>1</sup>.

---

<sup>1</sup>We note that [Theorem 2.1](#) does not include a leverage score ‘‘misestimation factor’’ ( $\beta$  in [\[56\]](#)) which is often included in works concerning leverage score sketching for OLS problems [\[37, 43, 56\]](#). When inexact leverage scores are used for sampling, the misestimation factor gives a measure of how close the inexact leverage scores are to the true leverage scores. We do not thoroughly discuss misestimation factors because we do not use the concept in this work. However, [Theorem 2.1](#) can be easily generalized to incorporate such a factor.

This result immediately begets the idea to use randomized NLS methods to rapidly solve the NLS subproblems in Alternating Nonnegative Least Squares methods for NMF and SymNMF previously described in Subsection 2.1.1. We described such a randomized algorithm in Section 4. A full proof of Theorem 2.1 is given in Subsection 4.3.1. We note that Theorem 2.1 holds for all convex Least Squares problems, this fact is made apparent in the proof given in Subsection 4.3.1.

**2.3. Related Work.** There has been much work on speeding up and scaling algorithms for low-rank methods mostly focusing on parallel algorithms and, more recently, randomization. We now give a brief review of related work with a focus on randomized methods.

**2.3.1. Randomized Methods for other Low-rank Approximations.** One of the first applications of leverage score sampling to matrix low-rank approximation was for computing a CUR decomposition [53]. The CUR decomposition selects actual rows and columns of the matrix to produce a low-rank approximation. Since the CUR decomposition is composed of sampled rows and columns it is said to be interpretable in the “space” of the original data.

There has been a lot of work related to randomized algorithms for computing the Canonical Polyadic (CP) decomposition for tensors. The CP Decomposition decomposes a  $N$ -way tensor into a sum of rank one tensors. Various constraints can be imposed similarly to low-rank matrix approximations, for example the Nonnegative CP decomposition. Two of the first methods for computing a randomized CP decomposition are by Battaglino et al. [1] and Zhou et al. [60]. Erichson et al. [18] proposed a randomized method for computing a CP decomposition based on using a tensor version of the RRF. The main idea is to first compress the tensor, via the RRF, compute a CP of the compressed tensor, and then lift the compressed CP back to the uncompressed space. Leverage scores have also been used in randomized CP algorithms [36, 6, 3]. There has also been work on scaling low-rank approximation methods to distributed computing environments [21, 29, 30] and combining such methods with randomized techniques [2]. These methods are applicable to sparse input tensors as they perform explicit sampling of tensor elements, thus preserving the sparsity pattern.

**2.3.2. Randomized NMF Algorithms.** Existing randomized algorithms for NMF focus on compressing the input matrix  $\mathbf{X}$ . Speed up is obtained from the fact that iteratively updating the low-rank factors for a compressed version of  $\mathbf{X}$  is cheaper. The first proposed method we are aware of used a random Gaussian matrix to sketch, once from each side of  $\mathbf{X}$ , resulting in two sketched matrices with smaller dimensions [54]. Tepper and Sapiro [51] proposed a similar method but used the RRF to compute an approximate basis for the row and column spans of  $\mathbf{X}$ . Their results showed that using the RRF significantly increased the accuracy of the randomized NMF results in terms of final residual. Erichson et al. [19] proposed a randomized HALS algorithm also based on the RRF. This method computes a single RRF of the input matrix  $\mathbf{X}$  and fits NMF to the resulting QB-decomposition. This method has the advantage of calling the RRF once and avoids sketching the factor matrices of NMF at each iteration. However, due to the way the problem size is reduced, imposing nonnegativity to the factors of the original matrix becomes an issue.

**3. NMF with Low-rank Approximate Input.** Our first proposed algorithm is a method called Low-rank Approximate-Input NMF (LAI-SymNMF). LAI-NMF computes an NMF of a low-rank approximation of the initial data matrix  $\mathbf{X}$ . The

**Algorithm LAI-SymNMF** : SymNMF of a Low-Rank Approximated  $\mathbf{X}$ 


---

**Require:** : a symmetric matrix  $\mathbf{X} \in \mathbb{R}^{m \times m}$ , target rank  $k$ , oversampling parameter  $\rho$ , exponent  $q$ , and regularization parameter  $\alpha$

**Ensure:**  $\mathbf{H}$ , the factor for an approximate rank- $k$  SymNMF of  $\mathbf{X}$ .

```

1: function  $[\mathbf{H}] = \text{LAI-SYMNMF}(\mathbf{X}, k, \rho, q)$ 
2:    $l := k + \rho$ 
3:    $[\mathbf{U}, \mathbf{\Lambda}] := \text{Apx-EVD}(\mathbf{X}, k, \rho, q)$ ,  $\triangleright$  Obtain approximate, truncated EVD of  $\mathbf{X}$ ,
    $\mathbf{U} \in \mathbb{R}^{m \times l}$  and  $\mathbf{\Lambda} \in \mathbb{R}^{l \times l}$ 
4:    $\mathbf{V} := \mathbf{U}\mathbf{\Lambda}$ 
5:   Initialize  $\mathbf{H} \in \mathbb{R}^{n \times k}$ 
6:   while Convergence Crit. Not Met do
7:      $\mathbf{Y}_H := (\mathbf{H}^\top \mathbf{V})\mathbf{U}^\top + \alpha \mathbf{H}^\top$   $\triangleright$  This replaces  $\mathbf{H}^\top \mathbf{X}$ 
8:      $\mathbf{G}_H := \mathbf{H}^\top \mathbf{H} + \alpha \mathbf{I}$ 
9:      $\mathbf{W} := \text{Update}(\mathbf{G}_H, \mathbf{Y}_H)$   $\triangleright$  See Appendix E for description of Update()
10:     $\mathbf{Y}_W := (\mathbf{W}^\top \mathbf{U})\mathbf{V}^\top + \alpha \mathbf{W}^\top$   $\triangleright$  This replaces  $\mathbf{W}^\top \mathbf{X}$ 
11:     $\mathbf{G}_W := \mathbf{W}^\top \mathbf{W} + \alpha \mathbf{I}$ 
12:     $\mathbf{H} := \text{Update}(\mathbf{G}_W, \mathbf{Y}_W)$ 
13:   end while
14: end function

```

---

objective function for LAI-NMF is

$$(3.1) \quad \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{U}_X \mathbf{V}_X - \mathbf{W} \mathbf{H}^\top\|_F,$$

where  $\mathbf{U}_X \in \mathbb{R}^{m \times l}$ ,  $\mathbf{V}_X \in \mathbb{R}^{l \times n}$  with  $k \leq l \ll \min(m, n)$  and  $\mathbf{U}_X \mathbf{V}_X \approx \mathbf{X}$  is a low-rank approximation of  $\mathbf{X}$ . The primary idea is that an approximate solution to [Equation \(3.1\)](#) can be quickly computed by exploiting the product form of  $\mathbf{U}_X \mathbf{V}_X$  to compute matrix vector products. That is,  $\mathbf{U}_X(\mathbf{V}_X \mathbf{v})$  is cheaper to compute and approximates the product  $\mathbf{X}\mathbf{v}$  (for an arbitrary vector  $\mathbf{v}$ ). Computing matrix products with the data matrix  $\mathbf{X}$  is the main computational bottleneck for many NMF and SymNMF algorithms. This idea has been explored before in [59] where the authors used low-rank approximations such as the truncated SVD, and in [19] where the QB-decomposition was used.

**3.1. SymNMF with Low-rank Approximate Input.** We now present **LAI-SymNMF**, an instantiation of the low-rank approximate input method. Since the data matrix  $\mathbf{X}$  is symmetric we also require that our low-rank approximation be symmetric and not compress only one side of the matrix  $\mathbf{X}$  (which would destroy symmetry) as in [19]. This is accomplished by using the approximate EVD of a symmetric matrix, using [Algorithm Apx-EVD](#), which gives an approximate truncated EVD of  $\mathbf{X}$ . The formulation for **LAI-SymNMF** is

$$(3.2) \quad \min_{\mathbf{H} \geq 0} \|\mathbf{U}_X \mathbf{\Lambda}_X \mathbf{U}_X^\top - \mathbf{H} \mathbf{H}^\top\|_F,$$

where  $\mathbf{U}_X \mathbf{\Lambda}_X \mathbf{U}_X^\top \approx \mathbf{X}$  is an approximate truncated EVD of  $\mathbf{X}$ .

**LAI-SymNMF** is flexible. Overall if the method for computing SymNMF requires computing the products  $\mathbf{X}\mathbf{H}$  and  $\mathbf{H}^\top \mathbf{H}$ , and potentially  $\mathbf{W}^\top \mathbf{X}$  and  $\mathbf{W}^\top \mathbf{W}$  as in [Equation \(2.3\)](#), then **LAI-SymNMF** can be efficient. Not all methods for computing

SymNMF rely on these products, such as Cyclic Coordinate Descent (CCD) [52]. As previously mentioned, in [35] it was shown that SymNMF via ANLS applied to Equation (2.3) is superior to PGD and a Newton-like method. Similarly, it was shown in [22] that SymNMF via BPP was competitive with a Projected Gauss-Newton based method for SymNMF and CCD. CCD is relatively inefficient for large problems as it iterates sequentially over every element of  $\mathbf{H}$ . Therefore we use methods based on Equation (2.3) with BPP and HALS, and the PGNCG's method as our base line for comparison.

To simplify pseudocode and emphasize the flexibility of the method we introduce the  $\text{Update}(\mathbf{G}, \mathbf{Y})$  function which takes in the Gram matrix  $\mathbf{G} \in \mathbb{R}^{k \times k}$  and the product between  $\mathbf{X}$  and either  $\mathbf{W}$  or  $\mathbf{H}$  denoted as  $\mathbf{Y} \in \mathbb{R}^{k \times m}$ , and performs an update using methods such as HALS or BPP. For a more in-depth discussion of  $\text{Update}()$  see Appendix E. We note that the  $\text{Update}()$  function abstraction is useful for Alternating Updating (AU) based methods. However, one of the advantages of our LAI method is that it is applicable to more algorithms. Existing randomized methods can be effectively used for the NMF update rules such as BPP, MU, or HALS but cannot or have not been used for all-at-once methods such as PGNCG. The algorithm outline showing how LAI can be used in conjunction with the PGNCG method is shown in Algorithm LAI-PGNCG-SymNMF in Appendix B.2.

*Computational Complexity.* The major part of computational complexity of Algorithm LAI-SymNMF is due to the RRF and iteratively updating the factors. Again, the cost of the RRF is  $O(qm^2l)$ . Then computing  $\mathbf{V} = \mathbf{X}^\top \mathbf{U}$ , where  $\mathbf{U}$  is the output of the RRF,  $\mathbf{V}$  and  $\mathbf{U}$  are  $m \times l$  and  $\mathbf{X} = \mathbf{X}^\top$  is  $m \times m$ , costs  $O(m^2l)$ . Additionally, each iteration requires forming two Gram matrices costing  $O(mk^2)$  each and applying the LAI to the factor matrices costing  $O(mkl)$ . If the algorithm runs for  $t$  iterations then the overall cost is  $O(qm^2l + tmkl)$  and  $l \geq k$ . So if  $tk \ll qm$  then we expect that computing the low-rank approximate input via the RRF will dominate the run time. Naturally, the choice of update function will determine the update cost.

**3.2. Approximation Errors for LAI-NMF.** The authors of [59] presented a simple error bound applicable to LAI-NMF that we can use to reason about LAI-NMF's performance. Proposition 1 from [59] states the following:

**PROPOSITION 3.1.** *Given a matrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$  and a low rank approximation  $\mathbf{X} \approx \mathbf{U}_X \mathbf{V}_X$ , where  $\mathbf{U}_X \in \mathbb{R}^{m \times l}$  and  $\mathbf{V}_X \in \mathbb{R}^{l \times n}$ , with error  $\mu = \|\mathbf{X} - \mathbf{U}_X \mathbf{V}_X\|_F$ , define  $\{\mathbf{W}^*, \mathbf{H}^*\}$  as the minimizers of Equation (3.1) and let  $v^* = \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}^\top\|_F$  with low-rank parameter  $k \leq l$ , then*

$$(3.3) \quad v^* \leq \|\mathbf{X} - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F \leq 2\mu + v^*$$

*Proof.* Define  $\{\mathbf{W}_+, \mathbf{H}_+\} = \arg \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}^\top\|$

$$\begin{aligned} \|\mathbf{X} - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F &= \|\mathbf{X} - \mathbf{U}_X \mathbf{V}_X + \mathbf{U}_X \mathbf{V}_X - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F \\ &\leq \|\mathbf{X} - \mathbf{U}_X \mathbf{V}_X\|_F + \|\mathbf{U}_X \mathbf{V}_X - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F \leq \mu + \|\mathbf{U}_X \mathbf{V}_X - \mathbf{W}_+(\mathbf{H}_+)^\top\|_F \\ &\leq \mu + \|\mathbf{U}_X \mathbf{V}_X - \mathbf{X}\|_F + \|\mathbf{X} - \mathbf{W}_+(\mathbf{H}_+)^\top\|_F \\ &= 2\mu + v^* \quad \square \end{aligned}$$

Proposition 3.1 allows us to reason about the achievable quality of approximation of LAI-NMF. Choosing a larger  $l$  can help decrease  $\mu$  but will also result in higher computational complexity. A natural choice for computing  $\mathbf{U}_X$  and  $\mathbf{V}_X$  is the truncated SVD (which would minimize  $\mu$ ) or, as we use, an approximate truncated SVD or

EVD computed using the **RRF**. As an alternative, the intermediate inequality, from the proof of [Proposition 3.1](#),

$$(3.4) \quad \|\mathbf{X} - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F \leq \mu + \|\mathbf{U}_X \mathbf{V}_X - \mathbf{W}_+(\mathbf{H}_+)^\top\|_F$$

provides an intuitive way to reason about LAI-NMF. This inequality says that LAI-NMF residual depends on  $\mu$ , which measures the quality of the low-rank input, and the term  $\|\mathbf{U}_X \mathbf{V}_X - \mathbf{W}_+(\mathbf{H}_+)^\top\|_F$ . This second term can be thought of as quantifying how much of the optimal NMF solution is captured in the low-rank input.

[Proposition 3.1](#) can give an error bound for [Algorithm LAI-SymNMF](#). For this, one needs an error bound for the QB-decomposition or the approximate EVD. That is, given a decomposition from the **RRF** as  $\mathbf{Q}_X \mathbf{B}_X$ , we seek a value  $\|\mathbf{Q}_X \mathbf{B}_X - \mathbf{X}\|_F \leq \mu_{RRF}$ . [Theorem 3.2](#), from Gu [25], provides such a bound. [Theorem 3.2](#) is a partial statement of Theorem 5.8 from [25] that we include for reference and completeness.

**THEOREM 3.2.** *Let  $\mathbf{Q}_X \mathbf{B}_X$  be a low-rank approximation of  $\mathbf{X} \in \mathbb{R}^{m \times n}$ , with  $n \leq m$ , obtained from the **RRF** with desired low-rank  $r$ , power iteration parameter  $q$ , column over sampling parameter  $\rho = l - r$  and a parameter  $0 < \delta \ll 1$ . Define*

$$C_\delta = \frac{e\sqrt{l}}{\rho+1} \left( \frac{2}{\delta} \right)^{\frac{1}{\rho+1}} \left( \sqrt{n-l+\rho} + \sqrt{l} + \sqrt{2\log(2/\delta)} \right).$$

*Then with probability  $1 - \delta$  the following holds:*

$$\|\mathbf{Q}_X \mathbf{B}_X - \mathbf{X}\|_F \leq \sqrt{\left( \sum_{j=r+1}^n \sigma_j^2(\mathbf{X}) \right) + r C_\delta^2 \sigma_{r+1}^2(\mathbf{X}) \left( \frac{\sigma_{r+1}(\mathbf{X})}{\sigma_r(\mathbf{X})} \right)^{4q}}.$$

This Theorem says that  $\|\mathbf{Q}_X \mathbf{B}_X - \mathbf{X}\|_F \leq \mu_{RRF}$  holds with some chosen probability  $1 - \delta$ , where  $\mu_{RRF}$  depends on  $\delta$  and other parameters of the **RRF** such as  $l$  and  $q$ .

[Proposition 3.3](#) explicitly combines [Proposition 3.1](#) and [Theorem 3.2](#) to give a probabilistic error bound for LAI-NMF with a randomized low-rank input from the **RRF**.

**PROPOSITION 3.3.** *Given a matrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$  compute a low-rank approximation  $\mathbf{X} \approx \mathbf{Q}_X \mathbf{B}_X$  where  $\mathbf{Q}_X \in \mathbb{R}^{m \times l}$  and  $\mathbf{B}_X \in \mathbb{R}^{l \times n}$  from the **RRF**. Then by [Theorem 3.2](#), for any  $0 < \delta \ll 1$ , we have that*

$$\|\mathbf{X} - \mathbf{Q}_X \mathbf{B}_X\|_F \leq \sqrt{\left( \sum_{j=k+1}^n \sigma_j^2(\mathbf{X}) \right) + k C_\delta^2 \sigma_{k+1}^2(\mathbf{X}) \left( \frac{\sigma_{k+1}(\mathbf{X})}{\sigma_k(\mathbf{X})} \right)^{4q}} = \mu_{RRF}$$

*holds with probability  $1 - \delta$ . Define  $\{\mathbf{W}^*, \mathbf{H}^*\} = \arg \min_{\mathbf{W}, \mathbf{H} \geq 0} \|\mathbf{Q}_X \mathbf{B}_X - \mathbf{W} \mathbf{H}^\top\|_F$  and the optimal NMF error  $v^* = \min_{\{\mathbf{W}, \mathbf{H}\} \geq 0} \|\mathbf{X} - \mathbf{W} \mathbf{H}^\top\|_F$ . Then with probability  $1 - \delta$*

$$(3.5) \quad v^* \leq \|\mathbf{X} - \mathbf{W}^*(\mathbf{H}^*)^\top\|_F \leq 2\mu_{RRF} + v^*$$

*as in [Proposition 3.1](#).*

In the case that  $\mathbf{X}$  is symmetric and the **RRF** is used to compute an approximate EVD it is simple to extend [Proposition 3.3](#). Given  $\mu_{RRF}$ , one can obtain an error bound for the approximate, truncated EVD produced by [Algorithm Apx-EVD](#). We

use a fact from [26]. Given a low-rank approximation  $\mathbf{X} \approx \mathbf{Q}_X \mathbf{B}_X$ , where  $\mathbf{X}$  is symmetric, from [Algorithm RRF](#) and defining  $\mathbf{P}_X = \mathbf{Q}_X \mathbf{Q}_X^T$ , observe that

$$\begin{aligned} \|\mathbf{X} - \mathbf{P}_X \mathbf{X} \mathbf{P}_X\|_F &= \|\mathbf{X} - \mathbf{P}_X \mathbf{X} + \mathbf{P}_X \mathbf{X} - \mathbf{P}_X \mathbf{X} \mathbf{P}_X\|_F \\ &\leq \|\mathbf{X} - \mathbf{P}_X \mathbf{X}\|_F + \|\mathbf{P}_X \mathbf{X} - \mathbf{P}_X \mathbf{X} \mathbf{P}_X\|_F = \|\mathbf{X} - \mathbf{Q}_X \mathbf{B}_X\|_F + \|\mathbf{P}_X (\mathbf{X} - \mathbf{X} \mathbf{P}_X)\|_F \\ &\leq \mu_{RRF} + \|\mathbf{X} - \mathbf{P}_X \mathbf{X}\|_F \leq 2\mu_{RRF}, \end{aligned}$$

where the last equality uses the symmetry of  $\mathbf{X}$  and  $\mathbf{P}_X$ . Therefore the low-rank approximation produced by [Algorithm Apx-EVD](#) achieves a residual of no more than  $2\mu_{RRF}$  if the [RRF](#) achieves an residual of no more than  $\mu_{RRF}$  (with high probability).

**3.3. Practical Considerations for LAI-SymNMF.** The quality of the Sym-NMF approximate solution found by [LAI-SymNMF](#) is dependant on the LAI. In the proposed [Algorithm LAI-SymNMF](#), the LAI is a truncated EVD. We propose two methods for ensuring that a high-quality factorization is produced by [LAI-SymNMF](#). Each one deals with a separate component of [Equation \(3.4\)](#). The first is to post-process the output from [LAI-SymNMF](#) by running a few iterations of the full NMF method. The second is to test and improve the quality of the approximate truncated EVD before starting the NMF iterations. We now discuss these two methods in more detail.

*Iterative Refinement.* Iterative Refinement (IR) runs some number of NMF iterations using the full matrix  $\mathbf{X}$  instead of the LAI. That is, after the iterations of [LAI-SymNMF](#) are finished, the algorithm switches over to using the full input matrix  $\mathbf{X}$ , therefore capturing information possibly lost in the low-rank approximation of  $\mathbf{X}$ . This helps in cases where the right side of [Equation \(3.4\)](#) is large. In practice our experimental results show that this method is effective in improving the Sym-NMF approximations attained by [LAI-SymNMF](#) while running faster than standard SymNMF methods.

*Adaptive RRF.* The [RRF](#) has two main hyperparameters 1) column oversampling parameter  $\rho$  and 2) the power iteration parameter  $q$ . There exists work on adaptive methods for selecting  $\rho$  [26]. For our algorithms, where  $k$  is usually considered a static input to NMF methods, we find that choosing  $\rho$  in the range of  $2k$  to  $3k$  is satisfactory. Empirically we find that determining a good  $q$  is more difficult. Prior works recommend a choice of  $q = 2$  [19, 26]. However we find that this choice can be inadequate and negatively impact performance. To remedy this we propose an Adaptive RRF algorithm that automatically chooses  $q$ . This method checks the residual of the QB-Decomposition after each power iteration and stops once a certain stopping criteria is met (e.g. lack of reduction in residual), similar to NMF. The residual check is cheap. Checking the residual of the QB-Decomposition after each power iteration requires only one extra multiplication against  $\mathbf{X}$  when calling the [RRF](#) by use of a standard ‘trick’ for computing the residual. That is if  $q$  power iterations are performed we only apply  $\mathbf{X}$ ,  $q + 1$  times. The algorithm outline can be found in the Appendix in [Algorithm Ada-RRF](#). This approach ensures we achieve a good value of  $\mu_{RRF}$  as in [Equation \(3.4\)](#).

**3.4. Discussion of LAI-SymNMF.** Compared to existing randomized methods, such as those in [19, 51], the LAI method is more general in that it can work for any NMF method that relies on matrix vector products  $\mathbf{X}\mathbf{v}$ , where  $\mathbf{v}$  is an arbitrary vector, for performance. For example, the Compressed-NMF method from [51], which we compare against in [Section 5](#), is only applicable for Alternating Updating methods. The PGNCG method for SymNMF from [22] can be used for [LAI-SymNMF](#)

but not for Compressed-NMF. Finally, unlike the randomized method in [19], the LAI method decouples the randomization from the NMF algorithm and accordingly preserves the convergence properties, such as convergence to a stationary point [32], of existing NMF algorithms applied to the low-rank input and can be reasoned about via [Proposition 3.3](#). A more detailed discussion of existing randomized NMF methods and LAI-NMF is given in [Appendix B.1](#).

**4. SymNMF via Leverage Score Sampling.** This section presents the algorithm for randomized SymNMF based on using leverage scores sampling to sketch the NLS problems in [Equation \(2.4\)](#). In the context of low-rank approximations leverage score sampling has been successfully used for computing CP decompositions [36, 6], especially of large sparse tensors. We propose this method as suitable for large, sparse data sets such as graph data. Though we focus on SymNMF we expect that this method would work well for standard NMF as well. Leverage score sampling preserves not only sparsity but nonnegativity as well.

**4.1. Leverage Score Sampling for Multiple Right Hand Sides.** Unlike in the CP decomposition, the coefficient matrix in the LS problem for low-rank matrix approximation has, in general, no special structure we can exploit to obtain fast leverage score estimates. However, for many methods the products  $\mathbf{W}^\top \mathbf{X}$  and  $\mathbf{X}\mathbf{H}$  are the most expensive part of an NMF iteration [29]. By computing a thin QR factorization of the matrices  $\mathbf{W}$  and  $\mathbf{H}$  at each iteration we can obtain exact leverage scores for use in sampling and avoid the expensive full matrix products involving the data matrix  $\mathbf{X}$ . The algorithm outline is given in [Algorithm LvS-SymNMF](#).

To formalize this idea, consider the NLS problem for updating  $\mathbf{H}$ :

$$(4.1) \quad \min_{\mathbf{H} \geq 0} \|\mathbf{W}\mathbf{H}^\top - \mathbf{X}\|_F,$$

where the coefficient matrix  $\mathbf{W} \in \mathbb{R}^{m \times k}$  is much smaller than the right hand side matrix  $\mathbf{X} \in \mathbb{R}^{m \times m}$  if  $k \ll n$ . Consider (approximately) solving the problem by the update as  $\mathbf{H} := \text{Update}(\mathbf{W}^\top \mathbf{W}, \mathbf{W}^\top \mathbf{X})$ . Recall  $\text{Update}()$  was introduced for [Algorithm LAI-SymNMF](#) and its details can be found in [Appendix E](#). Computing  $\mathbf{W}^\top \mathbf{W}$  and  $\mathbf{W}^\top \mathbf{X}$  costs  $O(mk^2)$  and  $O(m^2k)$  flops. The cost of the  $\text{Update}()$  will be denoted as  $O(T(m, k))$  and is dependent on the method used. In light of this we suggest the following randomized approach:

1. Compute a thin QR-decomposition of  $\mathbf{W} = \mathbf{Q}_W \mathbf{R}_W$  for  $O(mk^2)$  flops.
2. Compute the leverage scores exactly using  $\mathbf{Q}_W$  and generate the sampling matrix  $\mathbf{S}_W \in \mathbb{R}^{s \times m}$  as in [Equation \(2.11\)](#) drawing  $s$  samples.
3. Perform an  $\text{Update}()$  for the reduced problem  $\min_{\mathbf{H} \geq 0} \|\mathbf{S}_W \mathbf{W}\mathbf{H}^\top - \mathbf{S}_W \mathbf{X}\|_F^2$  as  $\mathbf{H} := \text{Update}(\mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{W}, \mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{X})$ .  
(As opposed to  $\mathbf{H} := \text{Update}(\mathbf{W}^\top \mathbf{W}, \mathbf{W}^\top \mathbf{X})$ .)

The conditions for this scheme to provide speed up are roughly that  $s \ll m$  and that the cost  $T(m, k)$  does not dominate the overall complexity. The key observation here is that computing the thin QR-decomposition costs only  $O(mk^2)$  flops and so when a large number of right hand side (RHS) vectors is present, computing the leverage scores is not the dominating cost. Note that this observation is relevant for problems with a similar structure to [Equation \(4.1\)](#). For example one can approximately solve an OLS problem with many RHS vectors and small coefficient matrix using this scheme.

Since the NLS problems given by [Equation \(2.4\)](#) are regularized, we propose the

**Algorithm LvS-SymNMF** Randomized SymNMF via Leverage Score Sampling

---

**Require:** Symmetric data matrix  $\mathbf{X} \in \mathbb{R}^{m \times m}$ , target rank  $k$ , threshold  $\tau$

**Ensure:**  $\{\mathbf{H}\}$  as the factors for an approximate rank- $k$  SymNMF of  $\mathbf{X}$ .

```

1: function  $[\mathbf{H}] = \text{LvS-SYMNMF}(\mathbf{X}, k, s, \tau)$ 
2:   Randomly initialize  $\mathbf{H} \in \mathbb{R}^{m \times k}$ 
3:   while Convergence Crit. Not Met do
4:      $\mathbf{R}_H := \text{chol}(\mathbf{H}^\top \mathbf{H})$             $\triangleright$  Compute upper triangular Cholesky factor  $\mathbf{R}$ 
5:     Solve  $\mathbf{H} \mathbf{R}_H^{-1} = \mathbf{Q}_H$             $\triangleright$  Triangular solve for  $\mathbf{Q}_H$ 
6:     Compute  $\mathbf{p}_H[i] = \|\mathbf{Q}_H[i, :] \|_2^2$  for  $i = 1 : m$   $\triangleright$  Compute leverage scores of  $\mathbf{H}$ 
7:     Construct  $\mathbf{S}_H$  using  $\mathbf{p}_H$  according to Eqn. (2.11)
8:      $\mathbf{Y}_H := \mathbf{H}^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{X}^\top + \alpha \mathbf{H}^\top$ 
9:      $\mathbf{G}_H := \mathbf{H}^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{H} + \alpha \mathbf{I}$ 
10:     $\mathbf{W} := \text{Update}(\mathbf{G}_H, \mathbf{Y}_H)$        $\triangleright$  See Appendix E for description of Update()
11:     $\mathbf{R}_W := \text{chol}(\mathbf{W}^\top \mathbf{W})$        $\triangleright$  Compute upper triangular Cholesky factor  $\mathbf{R}_W$ 
12:    Solve  $\mathbf{W} \mathbf{R}_W^{-1} = \mathbf{Q}_W$             $\triangleright$  Triangular solve for  $\mathbf{Q}_W$ 
13:    Compute  $\mathbf{p}_W[i] = \|\mathbf{Q}_W[i, :] \|_2^2$  for  $i = 1 : m$ 
14:    Construct  $\mathbf{S}_W$  using  $\mathbf{p}_W$  according to Eqn. (2.11)
15:     $\mathbf{Y}_W := \mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{X}^\top + \alpha \mathbf{W}^\top$ 
16:     $\mathbf{G}_W := \mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{W} + \alpha \mathbf{I}$ 
17:     $\mathbf{H} := \text{Update}(\mathbf{G}_W, \mathbf{Y}_W)$ 
18:   end while
19: end function

```

---

scheme given below for leverage score sampling:

$$\begin{aligned} \left\| \begin{bmatrix} \mathbf{S} & 0 \\ 0 & \mathbf{I}_k \end{bmatrix} \left( \begin{bmatrix} \mathbf{H} \\ \sqrt{\alpha} \mathbf{I}_k \end{bmatrix} \mathbf{W}^\top - \begin{bmatrix} \mathbf{X} \\ \sqrt{\alpha} \mathbf{H}^\top \end{bmatrix} \right) \right\|_F^2 &= \left\| \begin{bmatrix} \mathbf{S} \mathbf{H} \\ \sqrt{\alpha} \mathbf{I}_k \end{bmatrix} \mathbf{W}^\top - \begin{bmatrix} \mathbf{S} \mathbf{X} \\ \sqrt{\alpha} \mathbf{H}^\top \end{bmatrix} \right\|_F^2 \\ &= \left\| \mathbf{S} \mathbf{H} \mathbf{W}^\top - \mathbf{S} \mathbf{X} \right\|_F^2 + \alpha \left\| \mathbf{W} - \mathbf{H} \right\|_F^2, \end{aligned}$$

where the leverage score sampling matrix denoted by  $\mathbf{S}$  samples only rows of  $\mathbf{H}$  and the regularization portion is deterministically included. A similar technique is used for the sampling of  $\mathbf{W}$  when  $\mathbf{H}$  is being updated.

**4.1.1. Complexity.** At a high level the main computational kernels of SymNMF via regularization include  $\mathbf{XH}$ ,  $\mathbf{W}^\top \mathbf{X}$ ,  $\mathbf{W}^\top \mathbf{W}$ , and  $\mathbf{H}^\top \mathbf{H}$ , which cost  $O(m^2k)$  flops for the products with  $\mathbf{X}$  and  $O(mk^2)$  for the Gramians. Once this is done these matrix products are used to perform an update, e.g. via (approximately) solving the NLS problem.

Leverage score sampling replaces these products with  $\mathbf{XS}_H^\top \mathbf{S}_H \mathbf{H}$ ,  $\mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{X}$ ,  $\mathbf{W}^\top \mathbf{S}_W^\top \mathbf{S}_W \mathbf{W}$ , and  $\mathbf{H}^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{H}$ , which cost  $O(msk)$  and  $O(sk^2)$ . The number of samples  $s$  will be discussed in more detail later in Subsection 4.3.1. Additionally, computing the thin QR-Decomposition to obtain the leverage scores costs  $O(mk^2)$ . The discrepancy between asymptotic flop costs of the deterministic method and the leverage score based method comes primarily from the difference between  $m$  and  $s$ .

As previously stated, sampling does not generally affect the cost of the Update Rule which costs  $O(T(m, k))$ . The two update rules we use are the HALS and BPP methods. For a discussion of these rules and their properties see [32, 23].

**4.2. Practical Considerations for SymNMF via Leverage Score Sampling.** This section describes the implementation of [Algorithm LvS-SymNMF](#). We discuss how the leverage scores are computed and the use of a hybrid leverage score sampling scheme which was introduced in [36]. A theoretical analysis of Hybrid Sampling is given later in [Subsection 4.3.2](#).

For computing the thin QR-decomposition of a full rank matrix  $\mathbf{F} \in \mathbb{R}^{m \times k}$  where  $\mathbf{F} = \mathbf{Q}\mathbf{R}$ ,  $\mathbf{Q} \in \mathbb{R}^{m \times k}$  and  $\mathbf{R} \in \mathbb{R}^{k \times k}$ , we use the CholeskyQR algorithm. CholeskyQR computes  $\mathbf{F}^\top \mathbf{F}$ , then computes the Cholesky Decomposition  $\mathbf{F}^\top \mathbf{F} = \mathbf{R}^\top \mathbf{R}$  where  $\mathbf{R}$  is  $k \times k$  and upper triangular, and lastly solves the triangular linear system  $\mathbf{F} = \mathbf{Q}\mathbf{R}$  to obtain  $\mathbf{Q}$ . CholeskyQR is numerically less stable than Householder QR but faster and empirically we find that it works well for computing leverage scores.

Hybrid leverage score sampling samples a subset of rows deterministically and then randomly samples from the remaining rows. We find that Hybrid Sampling is crucial for good performance in our empirical results and we offer a rigorous analysis of its theoretical performance in [Subsection 4.3.2](#). Hybrid Sampling was proposed and shown to be effective for computing CP decompositions of sparse tensors in [36]. A similar method has been used and theoretically analyzed before for the column subset selection problem [46].

In Hybrid Sampling, a threshold  $\tau \in [0, 1]$  is used as a hyperparameter. When sampling according to the leverage score distribution, all rows that satisfy  $p_i \geq \tau$  are deterministically selected. Let the full set of row indices be  $\mathcal{I}$ , the set that is deterministically included be  $\mathcal{I}_D$  and  $s_D = |\mathcal{I}_D|$ , and the rest be  $\mathcal{I}_R = \mathcal{I} \setminus \mathcal{I}_D$  with  $s_R = |\mathcal{I}_R|$ , the remaining indices from which random samples are drawn.

Let  $\mathbf{Q}_A$  be a  $m \times k$  matrix with orthonormal columns that is being sampled. Assume without loss of generality that the rows of  $\mathbf{Q}_A$  permuted conformally to the sets  $\mathcal{I}_D$  and  $\mathcal{I}_R$  then the hybrid sampling matrix takes the form

$$(4.2) \quad \mathbf{S}_H = \begin{bmatrix} \mathbf{S}_D & \mathbf{0}_{s_D \times (m-s_D)} \\ \mathbf{0}_{(s_R) \times s_D} & \mathbf{S}_R \end{bmatrix} \in \mathbb{R}^{s_H \times m}$$

where  $\mathbf{S}_R \in \mathbb{R}^{s_R \times (m-s_D)}$  and  $\mathbf{S}_D \in \mathbb{R}^{s_D \times s_D}$  is a permutation matrix for the deterministically included portion defined as

$$(4.3) \quad (\mathbf{S}_D)_{ji} = \begin{cases} 1, & \text{if row } i \text{ is the } j\text{th deterministic sample} \\ 0, & \text{otherwise.} \end{cases}$$

$\mathbf{S}_D$  is often included in notation to make the deterministic inclusion aspect of the equations explicit. The submatrix  $\mathbf{S}_R$  is a leverage score sampling matrix as defined in [Equation \(2.11\)](#) but just of the indices in  $\mathcal{I}_R$ . When sampling for  $\mathbf{S}_R$ , rows that were sampled during the deterministic phase are not considered and the leverage scores are renormalized appropriately. The new leverage scores probabilities are  $\tilde{p}_i = \frac{l_i}{k-\theta}$  where  $\theta = \sum_{i \in \mathcal{I}_D} l_i(\mathbf{A})$ .

**4.3. Analysis of LvS-SymNMF.** We now explore some theoretical questions relevant to [Algorithm LvS-SymNMF](#). Specifically we seek to answer two questions: 1) Can results for leverage score sampling for OLS problems be extended to NLS problems? and 2) What is the sample complexity of the Hybrid Sampling method from [36]?

**4.3.1. Leverage Score Sampling for NLS Problems.** Error bounds and corresponding sampling complexities for sketching the ordinary least squares (OLS)

problem have been previously shown in a number of works [14, 48]. Larsen and Kolda gave a bound and proof structure in their work on computing a randomized CP decomposition [36]. Boutsidis and Drineas [4] considered using the randomized Hadamard transform for solving the NLS problem. Our proof structure generally follows that in [14] for OLS problems.

This section provides a proof of [Theorem 2.1](#) which provides generalized error bounds and sampling complexities for the Nonnegative Least Squares problem (NLS). We note that our results hold for all constrained least squares (CLS) problems as far as the problem remains convex. The NLS problem in [Equation \(2.13\)](#) is a convex optimization problem but unlike the case of OLS, does not yield a closed form solution. We are concerned only with the case where  $\mathbf{A}$  is full rank and overdetermined. The NLS error bound and sample complexity can be derived based on results by Daniels for the perturbation of Convex Quadratic Programs [11] and leverage score sampling for the OLS problem [14, 43, 37, 56].

The proof uses two *Structural Conditions* (SC's) such that if both are true then the error bound in [Theorem 2.1](#) for the NLS problem holds. We first discuss the sampling complexity and probability conditions under which these SC's hold.

Let  $\mathbf{A} = \mathbf{U}_A \boldsymbol{\Sigma}_A \mathbf{V}_A^\top$  be the thin SVD of the coefficient matrix in [Equation \(2.13\)](#) where  $\mathbf{U}_A \in \mathbb{R}^{m \times n}$ ,  $\boldsymbol{\Sigma}_A \in \mathbb{R}^{n \times n}$ , and  $\mathbf{V}_A \in \mathbb{R}^{n \times n}$ . Let  $\mathbf{S}$  be a leverage score sampling matrix for  $\mathbf{A}$ . The first Structural Condition (SC1) is

$$(4.4) \quad 1 - \epsilon_s \leq \sigma_i(\mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{U}_A) \leq 1 + \epsilon_s$$

for all  $i \in [k]$ , some  $\epsilon_s \in (0, 1)$ , and where  $\sigma_i(\mathbf{A})$  is the  $i$ th singular value of  $\mathbf{A}$ . The second SC (SC2) is

$$(4.5) \quad \|\mathbf{U}_A^\top \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{r}_{nls}\|_2 \leq \sqrt{\epsilon_r} \frac{\|\mathbf{r}_{nls}\|_2}{\sqrt{2}}$$

for some  $\epsilon_r \in (0, 1)$ . The leverage score sketching matrix  $\mathbf{S}$  satisfies SC1 and SC2 with high probability (given sufficiently many samples). For SC1 this is shown in [56] and for SC2 it can be shown by using [Theorem F.2](#) on the product  $\mathbf{U}_A^\top \mathbf{r}_{nls}$  [43]. We include the associated theorems for these statements in [Appendix F.1](#).

The second result we make use of is a bound on the perturbation of convex Quadratic Programs (QP's). Convex QP's have the general form

$$(4.6) \quad \min_{\mathbf{x} \in \mathcal{C}} Q(\mathbf{x}) = \min_{\mathbf{x} \in \mathcal{C}} \frac{1}{2} \mathbf{x}^\top \mathbf{K} \mathbf{x} - \mathbf{x}^\top \mathbf{d}$$

where  $\mathbf{K}$  is a square positive semi-definite matrix,  $\mathbf{d}$  is a vector, and  $\mathcal{C}$  is a convex set. Consider the NLS problems in [Equation \(2.13\)](#) and

$$(4.7) \quad \min_{\mathbf{x} \geq 0} \|\mathbf{S} \mathbf{A} \mathbf{x} - \mathbf{S} \mathbf{b}\|_2$$

and the equivalent QP's, respectively,

$$(4.8) \quad \min_{\mathbf{x} \geq 0} Q(\mathbf{x}) = \min_{\mathbf{x} \geq 0} \frac{1}{2} \mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{A}^\top \mathbf{b},$$

$$(4.9) \quad \min_{\mathbf{x} \geq 0} \hat{Q}(\mathbf{x}) = \min_{\mathbf{x} \geq 0} \frac{1}{2} \mathbf{x}^\top \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{A}^\top \mathbf{S}^\top \mathbf{b}.$$

The objective function given by 4.9 can be interpreted as a perturbed version of the objective function given by 4.8 with perturbed parameters  $\hat{\mathbf{K}} = \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{A}$  and  $\hat{\mathbf{d}} = \mathbf{A}^\top \mathbf{S}^\top \mathbf{b}$ . We make use of the following, which is a straightforward consequence of the fact that  $\nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) \geq 0$  for all  $\mathbf{x} \in \mathcal{C}$ , which is the characterization of a minima (at  $\mathbf{x}^*$ ) of a differentiable convex function  $f(\mathbf{x})$ ,

LEMMA 4.1 ([11, Equation 2.4]). *Convex Quadratic Program Inequality:  $Q(\mathbf{x})$  and  $\hat{Q}(\mathbf{x})$  from Equation (4.8) and Equation (4.9) satisfy*

$$(4.10) \quad (\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls})^\top [\nabla \hat{Q}(\hat{\mathbf{x}}_{nls}) - \nabla \hat{Q}(\mathbf{x}_{nls})] \leq (\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls})^\top [\nabla Q(\mathbf{x}_{nls}) - \nabla \hat{Q}(\mathbf{x}_{nls})]$$

where  $\nabla Q(\mathbf{x}) = \mathbf{K}\mathbf{x} - \mathbf{d}$  is the gradient of  $Q(\mathbf{x})$  at  $\mathbf{x}$ ,  $\nabla \hat{Q}(\mathbf{x}) = \hat{\mathbf{K}}\mathbf{x} - \hat{\mathbf{d}}$  is the gradient of  $\hat{Q}(\mathbf{x})$  at  $\mathbf{x}$ , and  $\mathbf{x}_{nls}$  and  $\hat{\mathbf{x}}_{nls}$  are the minimizers of Eq. (4.8) and Eq. (4.9) respectively.

All the tools needed for the proof of [Theorem 2.1](#) have now been established.

*Proof of Theorem 2.1.* Substituting the QP formulations of the original and sampled NLS problems into [Equation \(4.10\)](#), denoting  $\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls} = \mathbf{y}$  and parameterizing in terms of the matrix  $\mathbf{U}_A$  by writing  $\mathbf{Z} = \Sigma_A \mathbf{V}_A^\top$  and  $\mathbf{Z}\mathbf{y} = \mathbf{z}$ , the right hand side becomes

$$\begin{aligned} \mathbf{y}^\top [\nabla \hat{Q}(\hat{\mathbf{x}}_{nls}) - \nabla \hat{Q}(\mathbf{x}_{nls})] &= \mathbf{y}^\top [\hat{\mathbf{K}}\hat{\mathbf{x}}_{nls} - \hat{\mathbf{d}} - \hat{\mathbf{K}}\mathbf{x}_{nls} + \hat{\mathbf{d}}] = \mathbf{y}^\top \hat{\mathbf{K}}\mathbf{y} = \\ \mathbf{y}^\top \mathbf{Z}^\top \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{U}_A \mathbf{Z} \mathbf{y} &= \mathbf{z}^\top (\mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{U}_A) \mathbf{z} = \|\mathbf{S} \mathbf{U}_A \mathbf{z}\|_2^2 \geq \sigma_{min}(\mathbf{S} \mathbf{U}_A)^2 \|\mathbf{z}\|_2^2. \end{aligned}$$

The left hand side of [Equation \(4.10\)](#) yields

$$\begin{aligned} \mathbf{y}^\top [\nabla Q(\mathbf{x}_{nls}) - \nabla \hat{Q}(\mathbf{x}_{nls})] &= \mathbf{y}^\top [\mathbf{K}\mathbf{x}_{nls} - \mathbf{d} - \hat{\mathbf{K}}\mathbf{x}_{nls} + \hat{\mathbf{d}}] \\ &= \mathbf{z}^\top [\mathbf{U}_A^\top \mathbf{U}_A \mathbf{Z} \mathbf{x}_{nls} - (\mathbf{U}_A^\top \mathbf{S}^\top)(\mathbf{S} \mathbf{U}_A \mathbf{Z}) \mathbf{x}_{nls} - \mathbf{U}_A^\top \mathbf{b} + (\mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S}) \mathbf{b}] \\ &= \mathbf{z}^\top [\mathbf{U}_A^\top (\mathbf{A}\mathbf{x}_{nls} - \mathbf{b}) - \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} (\mathbf{A}\mathbf{x}_{nls} - \mathbf{b})] \\ &= \mathbf{z}^\top [\mathbf{U}_A^\top \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{r}_{nls}]. \end{aligned}$$

Recall that  $\mathbf{r}_{nls} = \mathbf{A}\mathbf{x}_{nls} - \mathbf{b}$  is the NLS residual vector. Combining the previous two equations back into [Lemma 4.1](#) we have

$$\mathbf{z}^\top [\mathbf{U}_A^\top \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{r}_{nls}] \geq \sigma_{min}(\mathbf{S} \mathbf{U}_A)^2 \|\mathbf{z}\|_2^2,$$

which implies

$$(4.11) \quad \|\mathbf{z}\|_2 \left\| \mathbf{U}_A^\top \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}^\top \mathbf{S} \mathbf{r}_{nls} \right\|_2 \geq \sigma_{min}(\mathbf{S} \mathbf{U}_A)^2 \|\Sigma_A \mathbf{V}_A (\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls})\|_2^2.$$

We now invoke the SC's with sufficient samples

$$s \geq k \max(C \log(k/\delta), 1/(\delta \epsilon_r)) \text{ where } C = 144/(1 - \sqrt{2})^2$$

so that both SC1, as in [Theorem F.1](#) with  $\epsilon_s = 1 - 1/\sqrt{2}$  meaning that  $\sigma_{min}(\mathbf{S} \mathbf{U}_A)^2 \geq 1/\sqrt{2}$ , and SC2, as in [Equation \(4.5\)](#) as in [Lemma F.3](#) with  $\mathbf{U}_A^\top \mathbf{r}_{nls}$  being the matrix product to approximate, both hold with high probability. With these parameters each

SC holds with at least probability  $1 - \delta/2$  so they both hold with at least probability  $1 - \delta$ . Thus, from [Equation \(4.11\)](#) we have that if both SC's hold, then

$$\frac{1}{\sqrt{2}}\sigma_{min}(\mathbf{A})\|(\hat{\mathbf{x}}_{nls} - \mathbf{x}_{nls})\|_2 \leq \frac{\sqrt{\epsilon_r}\|\mathbf{r}_{nls}\|_2}{\sqrt{2}},$$

from which the claimed upper bound immediately follows.  $\square$

*Discussion.* As stated previously [Theorem 2.1](#) is applicable to all problems of the form

$$(4.12) \quad \min_{\mathbf{x} \in \mathcal{C}} \|\mathbf{Ax} - \mathbf{b}\|_2.$$

with overdetermined, full rank  $\mathbf{A}$ , i.e., all convex LS problems. This is due to the fact that Eq. (4.10) is a result of convexity. Additionally, there are other types of sketches which can be used. For example, the Subsampled Randomized Hadamard Transform discussed in [14] and the ‘sparse embedding matrices’ from [56] can be applied. This is because these sketches can satisfy the two SC's with an appropriate number of samples.

**4.3.2. Analysis of Hybrid Sampling.** This section presents our theoretical results for Hybrid Sampling. According to the proof of [Theorem 2.1](#) to show that Hybrid Sampling works for OLS and NLS problems we need only show that the Hybrid Sampling matrix satisfies the two SC's in [Equation \(4.4\)](#) and [Equation \(4.5\)](#). Applying these results in the proof structure used for [Theorem 2.1](#) will yield sampling complexities, theoretical guarantees, and algorithms for Hybrid Sampling and solving NLS problems.

The first Structural Condition for Hybrid Sampling is given by [Lemma 4.2](#).

**LEMMA 4.2.** *Given  $\mathbf{A} \in \mathbb{R}^{m \times k}$  consider its thin SVD  $\mathbf{U}_A \Sigma_A \mathbf{V}_A^\top$  and its row leverage scores  $l_i(\mathbf{A})$  for each row  $i \in [m]$ , where  $[m]$  denotes the set of integers from 1 to  $m$ . Let  $\mathcal{I}_D$  be the set of deterministically included rows and define  $\mathcal{I}_R = [m] \setminus \mathcal{I}_D$  as the set of rows to be sampled from. Let  $\mathbf{S}_H \in \mathbb{R}^{s \times m}$  be a row sampling and rescaling matrix constructed via sampling with replacement on  $\mathcal{I}_R$  with  $s_R$  samples drawn according to the renormalized leverage scores  $l_i(\mathbf{A})$ ,  $s_D$  deterministic samples taken from  $\mathcal{I}_D$  where  $|\mathcal{I}_D| = s_D$ , and  $s = s_D + s_R$ . Define  $\theta = \sum_{i \in \mathcal{I}_D} l_i(\mathbf{A})$  and  $\xi = k - \theta$ . If  $\frac{144\xi \log(2k/\delta)}{\epsilon_s^2} < s_R$  then the following equation holds with probability at least  $1 - \delta$*

$$1 - \epsilon_s \leq \sigma_i^2(\mathbf{S}_H \mathbf{U}_A) \leq 1 + \epsilon_s$$

for all  $i \in [m]$  and  $\epsilon_s, \delta \in (0, 1)$ .

This proof is a modification of the proofs in [56, 41] for proving the analogous statement for standard leverage score sampling, which apply a Matrix Chernoff Bound. A statement of the Matrix Chernoff Bound is available in the Appendix as [Theorem F.4](#). This theorem allows us to reason about the quantity  $\Pr[\|\mathbf{W}\|_2 > \epsilon]$  where  $\mathbf{W} = \frac{1}{s_R} \sum_{j=1}^{s_R} \mathbf{X}_j$  and  $\mathbf{X}_j$  are draws of an independent, symmetric random matrix  $\mathbf{X} \in \mathbb{R}^{k \times k}$ . Additionally let  $\mathcal{S}_R$  be the set of samples or draws of  $\mathbf{X}$  such that  $|\mathcal{S}_R| = s_R$ . To prove [Lemma 4.2](#) we set  $\mathbf{W} = \mathbf{I} - \mathbf{U}_A^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{U}_A$ , where  $\mathbf{S}_H$  is given by Eq. (4.2).

*Proof.* We assume an appropriate row permutation conformal with Eq. (4.2) such that  $\mathbf{U}_A = \begin{bmatrix} \mathbf{U}_D \\ \mathbf{U}_R \end{bmatrix}$  where  $\mathbf{U}_D \in \mathbb{R}^{(s_D \times k)}$  and  $\mathbf{U}_R \in \mathbb{R}^{(m-s_D) \times k}$ . Define  $\mathbf{G}_D = (\mathbf{S}_D \mathbf{U}_D)^T \mathbf{S}_D \mathbf{U}_D = \mathbf{U}_D^T \mathbf{U}_D$  and  $\mathbf{G}_R = \mathbf{U}_R^T \mathbf{U}_R$ . Then we have  $(\mathbf{S}_H \mathbf{U}_A)^T (\mathbf{S}_H \mathbf{U}_A) = \mathbf{G}_D + (\mathbf{S}_R \mathbf{U}_R)^T (\mathbf{S}_R \mathbf{U}_R)$ . Let  $\mathbf{q}$  be a random row sample drawn from  $\mathbf{U}_R$  according to the distribution given by  $\tilde{p}_i = \|\mathbf{u}_i\|_2^2 / \|\mathbf{U}_R\|_F^2 = \|\mathbf{u}_i\|_2^2 / \xi$ , the corresponding leverage score probability for  $i \in \mathcal{I}_R$ , and let  $\tilde{\pi}$  be  $\mathbf{q}$ 's leverage score. Define  $\mathbf{q}_j$  for  $j \in [s_R]$  to be a sequence of independent draws of  $\mathbf{q}$ , so that  $\mathbf{q}_j = \mathbf{u}_i$  and  $\tilde{\pi}_j = \tilde{p}_i$  if the  $i$ th row of  $\mathbf{U}$  is the  $j$ th sample. Let  $\mathbf{X}_j = \mathbf{I}_k - \mathbf{G}_D - \frac{1}{\tilde{\pi}_j} \mathbf{q}_j \mathbf{q}_j^T$ , which is an independent, symmetric random matrix drawn from  $\mathbf{X} = \mathbf{I}_k - \mathbf{G}_D - \frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T$ . Note that for any  $i$  we have  $\frac{\|\mathbf{u}_i\|_2^2}{\tilde{p}_i} = \frac{\|\mathbf{u}_i\|_2^2}{\|\mathbf{u}_i\|_2^2} \xi = \xi$ .

Then

$$\begin{aligned} \mathbf{W} &= \frac{1}{s_R} \sum_{j=1}^{s_R} \mathbf{X}_j = \frac{1}{s_R} \sum_{j=1}^{s_R} \left( \mathbf{I}_k - \mathbf{G}_D - \frac{1}{\tilde{\pi}_j} \mathbf{q}_j \mathbf{q}_j^T \right) = \mathbf{I}_k - \mathbf{G}_D - \sum_{j=1}^{s_R} \frac{\mathbf{q}_j}{\sqrt{s_R \tilde{\pi}_j}} \frac{\mathbf{q}_j^T}{\sqrt{s_R \tilde{\pi}_j}} \\ &= \mathbf{I}_k - \mathbf{G}_D - (\mathbf{S}_R \mathbf{U}_R)^T \mathbf{S}_R \mathbf{U}_R = \mathbf{I}_k - (\mathbf{S}_H \mathbf{U}_A)^T \mathbf{S}_H \mathbf{U}_A. \end{aligned}$$

So  $\mathbf{I}_k - (\mathbf{S}_H \mathbf{U}_A)^T \mathbf{S}_H \mathbf{U}_A$  is a sum of symmetric, independent random matrices. Next, to apply Theorem F.4 we need to verify three conditions:  $\mathbb{E}[\mathbf{X}] = 0$ ,  $\|\mathbf{X}\|_2 \leq \gamma$ , and  $\|\mathbb{E}[\mathbf{X}^T \mathbf{X}]\|_2 \leq \nu^2$ , where  $\gamma$  and  $\nu^2$  are bounds used in Theorem F.4 to be derived.

First, we prove  $\mathbb{E}[\mathbf{X}] = \mathbf{0}$ . Observe that  $\mathbb{E}[\frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T] = \mathbf{G}_R$ , then

$$\mathbb{E}[\mathbf{X}] = \mathbb{E}[\mathbf{I}_k - (\mathbf{S}_D \mathbf{U}_D)^T (\mathbf{S}_D \mathbf{U}_D) - \frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T] = \mathbf{G}_R - \mathbb{E}[\frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T] = \mathbf{0}_{k \times k}$$

Second, bound  $\|\mathbf{X}\|_2 \leq \gamma$ . Observe  $\frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T$  is a rank-1 matrix with its 2-norm bounded by  $\max_{i \in \mathcal{I}_R} \frac{\|\mathbf{u}_i\|_2^2}{\tilde{p}_i} = \xi$ . Then

$$\|\mathbf{X}\|_2 \leq \|\mathbf{I}_k - (\mathbf{S}_D \mathbf{U}_D)^T (\mathbf{S}_D \mathbf{U}_D)\|_2 + \|\frac{1}{\tilde{\pi}} \mathbf{q} \mathbf{q}^T\|_2 \leq 1 + \xi.$$

So  $\gamma = 1 + \xi$  gives the needed bound.

Finally, we show  $\|\mathbb{E}[\mathbf{X}^T \mathbf{X}]\| \leq \nu^2$ . By expanding  $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$  and using the fact that

$$\mathbb{E}[(\mathbf{q} \mathbf{q}^T)(\mathbf{q} \mathbf{q}^T)/\tilde{\pi}^2] = \sum_{i \in \mathcal{I}_R} \frac{1}{\tilde{p}_i} (\mathbf{u}_i \mathbf{u}_i^T)^T (\mathbf{u}_i \mathbf{u}_i^T) = \xi \sum_{i \in \mathcal{I}_R} \mathbf{u}_i \mathbf{u}_i^T = \xi \mathbf{U}_R^T \mathbf{U}_R,$$

we obtain

$$\begin{aligned} \|\mathbb{E}[\mathbf{X}^T \mathbf{X}]\|_2 &= \|\xi \mathbf{U}_R^T \mathbf{U}_R - \mathbf{G}_R \mathbf{G}_R\|_2 = \|\mathbf{U}_R^T (\xi \mathbf{I} - \mathbf{U}_R \mathbf{U}_R^T) \mathbf{U}_R\|_2 \\ &\leq \|\xi \mathbf{I} - \mathbf{U}_R \mathbf{U}_R^T\|_2 \leq |\xi - \sigma_{\max}(\mathbf{U}_R \mathbf{U}_R^T)| \leq |\xi - 1|. \end{aligned}$$

So  $\nu^2 = |\xi - 1|$  gives the desired bound.

Substituting these into Equation (F.2) leads to

(4.13)

$$\Pr[\|\mathbf{I} - \mathbf{U}_A^T \mathbf{S}_H^T \mathbf{S}_H \mathbf{U}_A\|_2^2 \geq \epsilon_s] = \delta \leq 2k \exp \left[ -s \epsilon_s^2 / \left( 2(|\xi - 1|) + 2(1 + \xi) \epsilon_s / 3 \right) \right]$$

for some desired probability  $\delta$  and desired error tolerance  $\epsilon_s$ . By choosing  $\frac{144\xi \log(2k/\delta)}{\epsilon_s^2} \leq s_R$  we have with probability at least  $1 - \delta$

$$\|\mathbf{I} - \mathbf{U}_A^T \mathbf{S}_H^T \mathbf{S}_H \mathbf{U}_A\|_2^2 \leq \epsilon_s$$

which proves Lemma 4.2.  $\square$

The second Structural Condition for Hybrid Sampling is given by [Lemma 4.3](#).

**LEMMA 4.3.** *Let the thin SVD of  $\mathbf{A} \in \mathbb{R}^{m \times k}$  be  $\mathbf{U}_A \Sigma_A \mathbf{V}_A^\top$  and define the non-negative least squares problem  $\mathbf{x}_{nls} = \arg \min_{\mathbf{x} \geq 0} \|\mathbf{Ax} - \mathbf{b}\|_2^2$  with  $\mathbf{b} \in \mathbb{R}^{m \times n}$ . Let  $\theta = \sum_{i \in \mathcal{J}_D} l_i(\mathbf{A})$  be the sum of the leverage scores corresponding to rows in the deterministically included set. Let  $\mathbf{S}_H$  be the Hybrid Sampling matrix for  $\mathbf{A}$  with  $\frac{2\xi}{\delta\epsilon_r} \leq s_R$ . Then the following holds with probability at least  $1 - \delta$*

$$\|\mathbf{U}_A \mathbf{r}_{nls} - \mathbf{U}_A \mathbf{S}_H^\top \mathbf{S}_H \mathbf{r}_{nls}\|_2^2 \leq \epsilon_r \mathcal{R}_{nls}^2 / 2$$

where  $\mathcal{R}_{nls}^2 = \|\mathbf{r}_{nls}\|_2^2$ ,  $\mathbf{r}_{nls} = \mathbf{Ax}_{nls} - \mathbf{b}$ , and  $\epsilon_r, \delta \in (0, 1)$ .

*Proof.* Define  $\mathbf{S}_{\bar{D}}$  to be the  $(m - s_H) \times m$  matrix that picks out the rows of  $\mathbf{U}_A$  that are not deterministically sampled by  $\mathbf{S}_D$ . Row-partition the vector  $\mathbf{r}_{nls}$  conformally to  $\mathbf{U}_A$  as  $\mathbf{r}_{nls} = \begin{bmatrix} \mathbf{r}_D \\ \mathbf{r}_R \end{bmatrix}$ . Then we have

$$\begin{aligned} \|\mathbf{U}_A \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{r}_{nls}\|_F^2 &= \|\mathbf{U}_A \mathbf{r}_{nls} - \mathbf{U}_D^\top \mathbf{S}_D^\top \mathbf{S}_D \mathbf{r}_D - \mathbf{U}_R^\top \mathbf{S}_R^\top \mathbf{S}_R \mathbf{r}_R\|_F^2 \\ &= \|\mathbf{U}_D^\top \mathbf{S}_D^\top \mathbf{S}_D \mathbf{r}_D + \mathbf{U}_R^\top \mathbf{S}_R^\top \mathbf{S}_R \mathbf{r}_R - \mathbf{U}_D^\top \mathbf{S}_D^\top \mathbf{S}_D \mathbf{r}_D - \mathbf{U}_R^\top \mathbf{S}_R^\top \mathbf{S}_R \mathbf{r}_R\|_F^2 \\ &= \|\mathbf{U}_R^\top \mathbf{S}_{\bar{D}}^\top \mathbf{S}_{\bar{D}} \mathbf{r}_R - \mathbf{U}_R^\top \mathbf{S}_R^\top \mathbf{S}_R \mathbf{r}_R\|_F^2. \end{aligned}$$

From this form we can apply [Lemma F.3](#) to obtain

$$\begin{aligned} \mathbb{E}[\|\mathbf{U}_A \mathbf{r}_{nls} - \mathbf{U}_A^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{r}_{nls}\|_F^2] &= \mathbb{E}[\|\mathbf{U}_R^\top \mathbf{r}_R - \mathbf{U}_R^\top \mathbf{S}_R^\top \mathbf{S}_R \mathbf{r}_R\|_F^2] \\ &\leq \frac{1}{s_R} \|\mathbf{U}_R\|_F^2 \|\mathbf{r}_R\|_F^2 \leq \frac{\xi}{s_R} \|\mathbf{r}_{nls}\|_F^2 \leq \frac{\xi}{s_R} \mathcal{R}_{nls}^2. \end{aligned}$$

Finally using Markov's Inequality we have

$$Pr\left[\|\mathbf{U}^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{r}_{nls}\|_F^2 \geq \frac{\epsilon_r \|\mathbf{r}_{nls}\|_F^2}{2}\right] \leq 2 \frac{\mathbb{E}[\|\mathbf{U}^\top \mathbf{S}_H^\top \mathbf{S}_H \mathbf{r}_{nls}\|_F^2]}{\epsilon_r \|\mathbf{r}_{nls}\|_F^2} \leq 2 \frac{\frac{\xi}{s_R} \mathcal{R}_{nls}^2}{\epsilon_r \|\mathbf{r}_{nls}\|_F^2} = \frac{2\xi}{\epsilon_r s_R}$$

To succeed with probability at least  $\delta$ , we need  $\frac{2\xi}{\epsilon_r \delta} \leq s_R$ .  $\square$

*Discussion of [Lemma 4.2](#) and [Lemma 4.3](#).* [Lemma 4.2](#) and [Lemma 4.3](#) tell us that if  $\theta = \sum_{i \in \mathcal{J}_D} l_i(\mathbf{A})$  proportion of the  $k$  total leverage score ‘mass’ is taken deterministically with  $s_D$  rows, one needs only to take an additional  $\xi\phi$  random samples, where  $\xi = k - \theta$ , instead of  $k\phi$  to achieve the same theoretical guarantees for NLS or OLS problems, where  $\phi = \max(C \log(k/\delta), 1/(\delta\epsilon_r))$  and  $C = 144/(1 - \sqrt{2})^2$ . Overall hybrid sampling requires  $s_D + \xi\phi$  samples whereas standard leverage score sampling requires  $k\phi$  samples. For hybrid sampling to result in a lower sample complexity we must have  $s_D < (k - \xi)\phi = \theta\phi$ . This relies on the deterministic samples accounting for a sufficiently large  $\theta$ . This helps to explain why the hybrid method typically outperforms the deterministic method when the same number of samples are taken and supports the experimental results for the CP decomposition of sparse tensors in [36].

To determine the deterministic inclusion set  $\mathcal{J}_D$  we use the thresholding technique from [36]. This method chooses a parameter  $0 < \tau \leq 1$  and then the deterministic set includes all rows such that  $\tau \leq l_i(\mathbf{A})/k$ . This means that all rows with a leverage score value greater or equal to the threshold  $\tau$  are deterministically included. [Lemma 4.2](#) and [Lemma 4.3](#) do not strictly rely on using a thresholding technique to determine which rows should be deterministically included. Both results easily hold in the case

that a different method is used to determine the set  $\mathcal{I}_D$ . Any such technique for determining  $\mathcal{I}_D$ , like the thresholding one, will likely want to keep  $|s_D|$  small while making  $\theta$  as large as possible.

**5. Experimental Results.** This section presents empirical findings on the proposed SymNMF algorithms. The section focuses on two data sets, one dense and one sparse. Both data sets are represented as graphs and the proposed SymNMF algorithms are used to cluster the vertices of the graphs following approaches given in previous work [27, 35]. Namely, clustering via [27] proceeds by constructing a hypergraph random walk from the data, which is then used to construct a symmetric adjacency matrix that serves as the input to SymNMF clustering. For all experiments we use the initialization strategy from [35], though we acknowledge other initialization techniques such as those based on the SVD [5]. The strategy in [35] proceeds by generating a initial matrix for  $\mathbf{H} \in \mathbb{R}_+^{m \times k}$  with elements drawn uniformly from the interval on 0 to 1, and scaling them by the value  $2 * \sqrt{\zeta/k}$  where  $\zeta$  is the average of all elements of  $\mathbf{X}$ . Intuitively this ensures that the norm of the initial factor matrices is not too large or small in comparison to  $\mathbf{X}$ . Given an output factor  $\mathbf{H}$  the cluster membership for the  $i$ th vertex  $v$  is determined by finding the column index with the maximum value in the  $i$ th row of  $\mathbf{H}$  [35].

For checking convergence we use the normalized residual and the norm of the projected gradient. These are two standard metrics for checking the convergence of NMF and its variants [32, 19]. Definitions and some more discussion of these two metrics can be found in [Appendix C](#).

**5.1. Web of Science Text Data.** For the dense, symmetric case we utilize the Web of Science (WoS) Text<sup>2</sup> data set. The collection contains 46985 documents and 58120 terms. The number of term-document relationships is roughly 0.0013% of all possible connections. Each document is assigned one of 7 categories based on its content which we use as ground truth labels.

We cluster the data set using SymNMF via the Hypergraph with Edge Dependent Vertex Weights (EDVW) methodology [27]. This turns the matrix into a symmetric adjacency matrix where documents are vertices and words are considered as hyperedges. The matrix is likely dense as each hyperedge is expanded into a weighted clique in the obtained adjacency matrix. To assess clustering quality we compute the Adjusted Rand Index (ARI) for the clusterings produced by each SymNMF algorithm. To cluster with SymNMF we follow the methodology in [15].

Once the algorithms are unable to reduce the normalized residual by more than  $1e-4$  for four consecutive iterations the methods stop. If Iterative Refinement (IR), described in [Subsection 3.3](#), is being used the method will switch over to using the full input data matrix and apply the same stopping criteria to determine when to stop fitting to the full input matrix. All methods use [Ada-RRF](#) to determine how many power iterations to perform. [Ada-RRF](#) iterates until the normalized residual ceases to decrease by  $1e-3$  per power iteration. Each method is run 10 times.

We use a labeling system to name the algorithms. The algorithms we consider here vary in terms of Update rules among 1) HALS, 2) BPP, 3) PGNCG, if they use LAI, Compression (Comp) as in [51], or neither, and finally if Iterative Refinement is used, denoted with IR. A combination of these labels indicates the method used. For example LAI-BPP-IR is the [Algorithm LAI-SymNMF](#) with BPP as the update rule and iterative refinement run at the end. LAI-BPP indicates the same technique but

---

<sup>2</sup><https://data.mendeley.com/datasets/9rw3vkcfy4/6>

without IR at the end. Finally BPP indicates standard SymNMF method with BPP as the update rule.

The regularized version of SymNMF in [Equation \(2.3\)](#) requires an input hyperparameter  $\alpha$  to balance the two objectives. The compressed problem also needs this hyperparameter. In practice we find that using the value  $\alpha = \max(\mathbf{X})$  recommended for the uncompressed problem works well [35].

**5.1.1. Results.** The convergence results for these experiments are shown in [Figure 1](#). These show that the [LAI-SymNMF](#) method results in significant speed up. [LAI-SymNMF](#) without Iterative Refinement (IR) achieves about a  $4\times$  speed up over standard SymNMF with BPP. For [LAI-SymNMF](#) with HALS we observe a speed up of  $7.5\times$ . For both BPP and HALS, IR does not appear to be needed when [Ada-RRF](#) finds a good starting low-rank approximation. For comparison we include the SymNMF variant of the ‘Compressed NMF’ method, from [51], and see that it performs nearly identically to [LAI-SymNMF](#). See [Appendix B.1](#) for a discussion of the similarity of the two methods. In terms of the PGNCG method in [Figure 4c](#) and [Table 2](#) we see that randomization greatly benefits PGNCG resulting in about a  $6\times$  speed up. Additionally, IR in conjunction with PGNCG achieves the lowest normalized residual norm. Overall we see that randomized methods for SymNMF result in significant speed up while preserving, or even improving, solution quality. The run time of the randomized methods included the time needed to compute the LAI. This is why all the randomized methods appear to ‘start’ later than the non-randomized methods.

Note that we do not report results for [LvS-SymNMF](#) on this data set as the code took too long to execute. This is due to the fact that copying large portions (sampled rows) of a large dense data matrix  $\mathbf{X}$  at each iteration takes a long time.

Additional experimental data can be found in [Appendix G.1](#). [Figure 4](#) and [Tables 4](#) and [5](#) show results for different values of the oversampling parameter  $\rho$ . We find that varying  $\rho$  does not have much effect on the solution quality but can negatively impact run time. Finally, [Appendix G.1](#) includes plots and tables showing statistics for using a static choice of  $q = 2$  with IR. Comparing these results with those from using [Algorithm Ada-RRF](#) shows the superiority of [Algorithm Ada-RRF](#) versus a static choice of  $q$ .

We briefly compare against Spectral Clustering as described in [45, 27] to validate SymNMF-based clustering results. Spectral clustering achieves an average ARI of 0.293 over 20 runs. This is a worse ARI score than all of the SymNMF methods we try, see [Table 2](#). Spectral clustering using the MATLAB functions `eigs()` and `kmeans()` takes about 35 seconds to execute. We also note that the normalized residual achieved by a rank = 7 SVD is 0.9340.

**5.2. Microsoft Open Academic Graph.** We run our methods for SymNMF on the Microsoft Open Academic Graph (OAG) [58]. This is a data set that combines the Microsoft Academic Graph (MAG) [49] and the Arnet-Miner (Aminer) academic graph [50]. From the OAG<sup>3</sup> we take a subset of 37,732,477 papers and use their citation information to form a sparse graph with 966,206,008 non-zeros as in [20]. This adjacency matrix is symmetrically normalized and the diagonal is zeroed out following the methodology in [35]. The rank is set to  $k = 16$  for all experiments. The parameter  $s$  is set to  $\lceil 0.05 * m \rceil$  where  $m$  is the dimension of the square input matrix.

The regularized version of SymNMF in [Equation \(2.3\)](#) requires an input hyperparameter  $\alpha$  to balance the two objectives. The sampled problem will also need this hy-

---

<sup>3</sup><https://github.com/ramkikannan/planc>

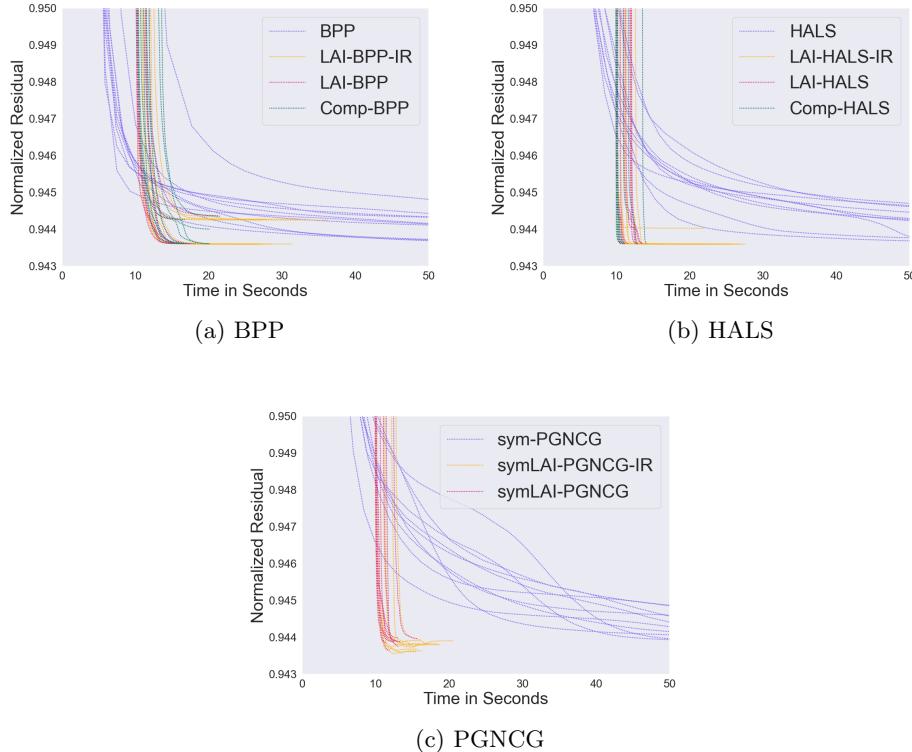


Fig. 1: Normalized Residual Norm Plots for the WoS data set. Three different update rules are shown: BPP, HALS, and PGNCG.

Alg.	Iters	Time	Avg. Min-Res	Min-Res	Mean-ARI
PGNCG	50.8	80.311	0.944	0.9437	0.3063
LAI-PGNCG	92.1	12.834	0.9439	0.9437	0.3107
PGNCG-IR	93.8	16.984	0.9437	0.9435	0.3161
BPP	38.1	66.95	0.9439	0.9436	0.3224
LAI-BPP	71.5	16.734	0.9438	0.9436	0.3264
LAI-BPP-IR	79.0	29.062	0.9439	0.9436	0.3148
Comp-BPP	79.3	18.266	0.9438	0.9436	0.3207
HALS	58.1	92.461	0.9437	0.9436	0.3201
LAI-HALS	97.0	12.124	0.9436	0.9436	0.3269
LAI-HALS-IR	90.5	23.799	0.9436	0.9436	0.3205
Comp-HALS	75.0	11.454	0.9437	0.9436	0.3237

Table 2: Metrics for the various run shown in Figure 1. Iters is mean number of iterations taken, Time is mean run time in seconds, Avg. Min-Res is average minimum achieved residual, Min-res is overall lowest achieved residual, and Mean-ARI is the mean ARI score. Averages are taken over 20 runs.

perparameter. We propose that using the same values of  $\alpha$  for the leverage score sampling method and the deterministic method is reasonable since  $\mathbb{E}[\|\mathbf{S}_H \mathbf{H}\|_F^2] = \|\mathbf{H}\|_F^2$  and  $\mathbb{E}[\|\mathbf{S}_W \mathbf{W}\|_F^2] = \|\mathbf{W}\|_F^2$ .

**Algorithm LvS-SymNMF** is run with BPP and HALS as the update rules. The normalized residual norms and the projected gradient values are shown in [Figure 2](#). See [Appendix C.3](#) for details on projected gradient values. First we note that the LAI method with BPP update rule is generally unsuccessful at reducing the error. We hypothesize that this is because the RRF produces a LAI with a large value of  $\|\mathbf{U}_X \mathbf{V}_X - \mathbf{W}_+ (\mathbf{H}_+)^T\|_F$  as in [Equation \(3.4\)](#).

Next we note that using the Hybrid Sampling method is important to achieve good speed-up. In the case of the BPP update rule, in [Figure 2a](#) one can observe that the purely random method ( $\tau = 1$ ) does not provide any speed up over the standard BPP method. However when the hybrid method with  $\tau = 1/s$  is used, one can see that the leverage score sampling algorithm becomes competitive. When the BPP update rule is used, a significant amount of the time per iteration goes into solving the  $2m$  NLS problems. So while leverage score sampling effectively eliminates the cost of multiplying by  $\mathbf{X}$ , only about a 50% speed up is achieved as leverage score sampling has no effect on the cost of the update function. In the case of HALS, we observe significantly better speed up when leverage score sampling is used. [Figure 2a](#) clearly shows that hybrid leverage score sampling outperforms both leverage score sampling and the standard HALS method. The hybrid leverage score sampling method in this case is able to achieve about a  $5.5 \times$  speed up per iteration over standard HALS. We note that the use of the modified HALS update rules, as in Equations [\(2.5\)](#) and [\(2.6\)](#), are crucial for obtaining good performance.

[Figure 3](#) shows the time break down for 3 algorithms: leverage score sampling with HALS and BPP as the update rule and a standard NMF method with HALS as the update rule. The timings are divided into 3 categories: 1) “Matrix Multiplication” (MM) for computing the four main matrix products, 2) “Solve” for the time spent in the update rule (e.g. time spent in the BPP solver), and 3) “Sampling” time spent performing leverage score sampling which includes computing the thin QR decomposition. The sampling time also includes time to form the needed subsampled matrices. The OAG data set is stored using MATLAB’s sparse matrix representation which allows for fast row/column slicing (because the matrix is symmetric) via the Compressed Sparse Column or (CSC) format. This is in juxtaposition to the hypergraph representation of the WoS data set which is dense and so does not benefit from sparse matrix formats. Additionally, the MM time is greatly reduced by using leverage score sampling while incurring an acceptable overhead in terms of sampling time. Lastly we see that this data confirms that the BPP solver has limited potential computational gains from leverage score sampling.

[Figure 2](#) also shows the projected gradient values for the OAG runs. As discussed in the next section, the solutions found by the randomized and non-randomized methods are quite different. The fact that the randomized methods does a good job of reducing both the Residual and the Projected Gradient helps increase confidence that **Algorithm LvS-SymNMF** is performing well. See [Appendix C.3](#) for details on the Projected Gradient.

**5.2.1. Results for the OAG.** We now analyze the output of running the **LvS-SymNMF** algorithms on the OAG. The factors with  $k = 16$  are used to form  $k$  clusters, as in the WoS hypergraph, following the methodology in [35]. Since this data set does not come with ground truth, assessing the quality of the **LvS-SymNMF**

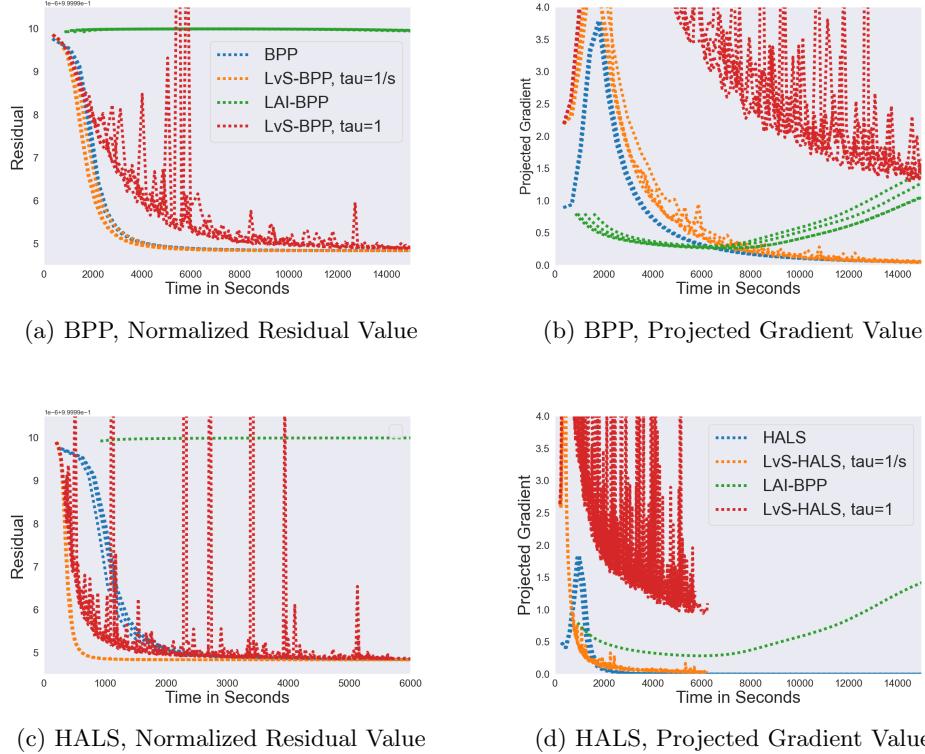


Fig. 2: Normalized residual and projected gradient values for OAG using HALS and BPP update rules.  $k = 16$ ,  $\tau = 1/s$  or  $\tau = 1$  as indicated in the legend.

outputs is more difficult. The residuals output by both the HALS and the LvS-HALS method differ by a tiny amount, on the order of  $1e-12$ , with the leverage score method achieving the lower residual.

We use a similar labeling scheme as for the WoS experiments. LvS represents the use of leverage scores sampling and the considered update rules are HALS and BPP. So LvS-BPP represents Algorithm LvS-SymNMF with BPP. Additionally for leverage score sampling methods the value of  $\tau$  used is also indicated.

The two sets of clusters produced by the methods are different, as can be immediately deduced from the difference in cluster sizes. Standard HALS produces one huge cluster containing all but around 100,000 of the vertices. These 100,000 vertices are then split into an additional 15 clusters whose sizes range from around 5000 to 8000 vertices. The LvS-HALS method finds 3 large clusters of sizes about 20 million, 17 million, and 150,000. The remaining 13 clusters have between one to six thousand vertices each. Cluster-0 is the largest cluster for both methods. Cluster-15 and Cluster-10 are the 17 million and 150 thousand sized cluster for LVS-HALS. The smaller clusters are all somewhat connected to Cluster-0 and sparsely, if at all, connected to each other.

To analyze these clusters more rigorously we compute the Silhouette Scores for each vertex [47]. For a set of clusters  $\{C_j\}$  for  $j \in [k]$  we define the following quantities

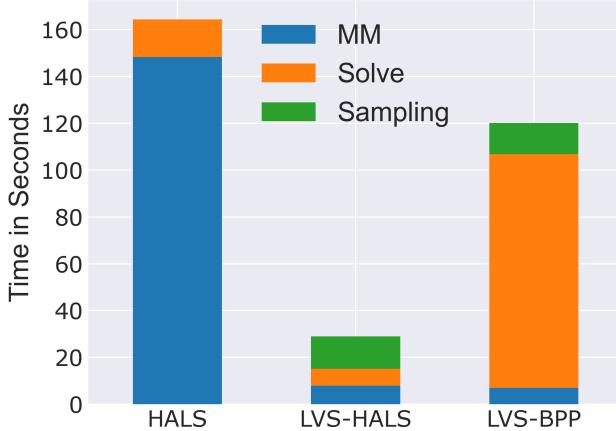


Fig. 3: Time Breakdown per iteration for three algorithms. Standard SymNMF with HALS update rule (HALS), SymNMF with leverage score sampling and HALS (LVS-HALS), and SymNMF with leverage score sampling and BPP (LVS-BPP).

for a vertex  $v$  in cluster  $C_l$

$$a(v) = \frac{1}{|C_l| - 1} \sum_{j \in C_l, j \neq v} \mathbf{A}_{vj} \quad \text{and} \quad b(v) = \max_{t, t \neq l} \frac{1}{|C_t|} \sum_{j \in C_t} \mathbf{A}_{vj}.$$

Here  $a(v)$  measures a vertex's similarity to its own cluster,  $b(v)$  measures a vertex's similarity to the next most similar cluster and  $\mathbf{A}$  is the graph adjacency matrix input to the algorithm SymNMF. The Silhouette Score for  $v$  is then given by

$$s(v) = \frac{a(v) - b(v)}{\max(a(v), b(v))}.$$

Silhouette Scores range from 1 to  $-1$ , with 1 meaning a vertex is very similar to its own cluster and  $-1$  meaning it should be moved to its next closest cluster. Individual vertex scores are then averaged over their cluster memberships to obtain cluster level Silhouette Scores. Note that this definition of Silhouette Score is for similarity metrics and so is different than the typical Silhouette Score equation which is defined for dissimilarity metrics.

The Silhouette Scores are computed for each cluster. The Silhouette Scores for the HALS methods are perfect with the exception of Cluster-0, which still achieves a high score of 0.78. The Silhouette Scores for the LVS-HALS algorithm are not as high as the ones for HALS but are still well above 0. The exception is Cluster-0 (from LVS-HALS), which has a near 0 Silhouette Score of  $-0.04$ , meaning that it does not form a good cluster but also that its vertices do not necessarily belong in one of the other clusters. Other than Cluster-0 the LVS-HALS method finds clusters with mean Silhouette Score  $0.92 \pm 0.1$  ( $\pm$  Standard Deviation). This indicates that the LVS-HALS method is able to find well structured clusters in terms of Silhouette Score. To further validate that LVS-HALS is finding meaningful clusters we show

<b>T10</b>	<b>T11</b>	<b>T12</b>	<b>T13</b>	<b>T14</b>	<b>T15</b>
matter	optical	determination	flow	equivalent	signaling
observations	voltage	organic	devices	constant	transcription
emission	gain	summary	fluid	weight	kinase
cluster	electron	block	integrated	forms	receptor
ray	strain	absorption	driven	coefficient	genes
infrared	layers	iii	electric	half	receptors
mass	laser	compounds	film	proved	pathways
evolution	films	gel	device	plane	proteins
physics	crystal	ions	voltage	establish	mediated
simulations	layer	chemistry	liquid	form	pathway

Table 3: Top key words for Leverage Score HALS output. Topics 10-15 from the Microsoft Open Academic Graph run using HALS as the update rule with Leverage Score sampling ( $\tau = 1/s$ ) and  $k = 16$ . The 10 top key words in terms of tf-idf association are shown in the table. We can see many of the top key words per topic seem to form a coherent subject matter.

the top 10 words associated with Clusters 10 to 15 in [Table 3](#). More data about this experiment can be found in [Appendix G.2](#).

**6. Conclusion.** We have presented two novel algorithms for computing Sym-NMF in randomized ways. These methods are suitable for large problems as demonstrated by our findings on two data sets. Our proposed methods achieve significant speed ups of  $5\times$  or more and produce quality solutions on downstream clustering tasks. Additionally our techniques are applicable to standard NMF formulations as well. Finally we presented a number of theoretical results that justify why and when our proposed algorithms perform well in practice. These results include an analysis of leverage score sampling for approximately solving Nonnegative Least Squares problems, sample complexity analysis for the Hybrid Sampling scheme from [37], and some guarantees for [Algorithm LAI-SymNMF](#).

Comparing these two methods we believe that the [Algorithm LAI-SymNMF](#) method is appropriate when a high quality LAI can be computed quickly and captures the underlying NMF signal. This is highlighted by [Equation \(3.4\)](#) which bounds the potential quality of the [Algorithm LAI-SymNMF](#) in terms of the quality of the Low-rank Approximate Input (LAI). In other cases [Algorithm LvS-SymNMF](#) can be used, particularly in the case of sparse input data, as it does not compute an upfront LAI but instead samples new subsets of rows at every iteration. Comparing these methods computationally, [Algorithm LAI-SymNMF](#) costs roughly  $O(qm^2l)$  for computing the LAI and then  $O(mkl)$  per iteration, where  $l = O(k)$  is the rank of the intermediate approximation, and [Algorithm LvS-SymNMF](#) costs roughly  $O(mks) = O(mk^2 \max(\log(\frac{k}{\delta}), \frac{1}{\delta\epsilon_r}))$  per iteration, where  $s$  is the number of row samples. The upfront cost of [Algorithm LAI-SymNMF](#) is expensive but its per iteration cost is lower than that of [Algorithm LvS-SymNMF](#). This is assuming one stays in the regime of  $k \ll m$ . Additionally it is worth noting that both methods still require operations costing  $O(mk^2)$  for the products  $\mathbf{H}^\top \mathbf{H}$  and  $\mathbf{W}^\top \mathbf{W}$  or computing a thin-QR of  $\mathbf{W}$  and  $\mathbf{H}$ . This again emphasizes the reliance of  $k \ll m$  for computational efficiency. It is also important to note that the computational complexity costs of

each algorithm do not tell the whole story. For example on large dense problems [Algorithm LvS-SymNMF](#)'s need to copy rows of the data matrix can be expensive as mentioned in [Section 5](#).

**Appendix A. Derivation of Symmetrically Regularized HALS.** This section presents the derivation of the regularized HALS method for SymNMF and its efficient updating. For this derivation  $\mathbf{X} = \mathbf{X}^\top$  is symmetric throughout. Define

$$\mathbf{R}_{\neq i} = \mathbf{A} - \mathbf{W}_{\neq i} \mathbf{H}_{\neq i}^\top = \mathbf{A} - \sum_{j \neq i, j=1}^k \mathbf{w}_j \mathbf{h}_j^\top$$

Where  $\mathbf{W}_{\neq i}$  is  $\mathbf{W}$  without the  $i$ th column and similar for  $\mathbf{H}_{\neq i}$ . Then the rank one update equation can be posed as

$$\begin{aligned} \left\| \begin{bmatrix} \mathbf{H} \\ \sqrt{\alpha} \mathbf{I}_k \end{bmatrix} \mathbf{W}^\top - \begin{bmatrix} \mathbf{X} \\ \sqrt{\alpha} \mathbf{H}^\top \end{bmatrix} \right\|_F^2 &\rightarrow \left\| \begin{bmatrix} \mathbf{h}_i \\ \sqrt{\alpha} \mathbf{e}_i \end{bmatrix} \mathbf{w}_i^\top + \begin{bmatrix} \mathbf{H}_{\neq i} \\ \sqrt{\alpha} \mathbf{I}_{\neq i} \end{bmatrix} \mathbf{W}_{\neq i}^\top - \begin{bmatrix} \mathbf{A} \\ \sqrt{\alpha} \mathbf{H}^\top \end{bmatrix} \right\|_F^2 \\ &= \left\| \begin{bmatrix} \mathbf{h}_i \\ \sqrt{\alpha} \mathbf{e}_i \end{bmatrix} \mathbf{w}_i^\top - \begin{bmatrix} \mathbf{R}_{\neq i}^\top \\ \sqrt{\alpha} (\mathbf{I}_{\neq i} \mathbf{W}_{\neq i}^\top - \mathbf{H}^\top) \end{bmatrix} \right\|_F^2. \end{aligned}$$

Next denote the matrix  $\tilde{\mathbf{R}}_{\neq i}$  and the vector  $\mathbf{u}_i$  as (Note that we have flipped sign here)

$$\tilde{\mathbf{R}}_{\neq i} = \begin{bmatrix} \mathbf{R}_{\neq i}^\top \\ \sqrt{\alpha} (\mathbf{H}^\top - \mathbf{I}_{\neq i} \mathbf{W}_{\neq i}^\top) \end{bmatrix}, \mathbf{u}_i = \begin{bmatrix} \mathbf{h}_i \\ \sqrt{\alpha} \mathbf{e}_i \end{bmatrix}$$

Now apply the standard HALS update

$$\begin{aligned} \mathbf{w}_i^\top &= \left[ \frac{\tilde{\mathbf{R}}_{\neq i}^\top \mathbf{u}_i}{\|\mathbf{u}_i\|_2^2} \right]_+ = \left[ \left( \begin{bmatrix} \mathbf{R}_{\neq i}^\top \\ \sqrt{\alpha} (\mathbf{H}^\top - \mathbf{I}_{\neq i} \mathbf{W}_{\neq i}^\top) \end{bmatrix}^\top \begin{bmatrix} \mathbf{h}_i \\ \sqrt{\alpha} \mathbf{e}_i \end{bmatrix} \right) / (\|\mathbf{h}_i\|_2^2 + \alpha) \right]_+ \\ &= \frac{\mathbf{R}_{\neq i} \mathbf{h}_i + \alpha \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} = \frac{(\mathbf{R}_{\neq i} + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} \end{aligned}$$

Finally the efficient updating method is derived as

$$\begin{aligned} \frac{(\mathbf{R}_{\neq i} + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} &= \frac{(\mathbf{A} - \mathbf{W}_{\neq i} \mathbf{H}_{\neq i}^\top + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} \\ &= \frac{(\mathbf{A} - \mathbf{W}_{\neq i} \mathbf{H}_{\neq i}^\top + \mathbf{w}_i \mathbf{h}_i^\top - \mathbf{w}_i \mathbf{h}_i^\top + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} = \frac{(\mathbf{A} - \mathbf{W} \mathbf{H}^\top + \mathbf{w}_i \mathbf{h}_i^\top + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} \\ &= \frac{(\mathbf{A} - \mathbf{W} \mathbf{H}^\top + \alpha \mathbf{I}) \mathbf{h}_i + (\mathbf{w}_i \mathbf{h}_i^\top) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} = \frac{(\mathbf{A} - \mathbf{W} \mathbf{H}^\top + \alpha \mathbf{I}) \mathbf{h}_i + \|\mathbf{h}_i\|_2^2 \mathbf{w}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} \end{aligned}$$

The final update rule is written as

$$\mathbf{w}_i \leftarrow \left( \frac{(\mathbf{A} - \mathbf{W} \mathbf{H}^\top + \alpha \mathbf{I}) \mathbf{h}_i}{\|\mathbf{h}_i\|_2^2 + \alpha} + \frac{\|\mathbf{h}_i\|_2^2}{\|\mathbf{h}_i\|_2^2 + \alpha} \mathbf{w}_i \right)_+$$

for the columns of  $\mathbf{W}$  and this can be done similarly for the columns of  $\mathbf{H}$ .

## Appendix B. Additional Material for LAI-SymNMF.

**B.1. Comparison between LAI-NMF and Compressed-NMF.** **LAI-SymNMF** is similar to the randomized NMF algorithms proposed in [19, 51]. The ‘‘Randomized HALS’’ method proposed in [19] which destroys symmetry is inherently unsymmetric, as it compresses only one dimension of the matrix, and so we do not consider comparing against it. In fact **LAI-SymNMF** could be viewed as a natural extension of this

“Randomized HALS” method to randomized SymNMF. The so called “Compressed-NMF” algorithm proposed in [51] can be straightforwardly extended to the SymNMF problem. We do so and compare against it later. Generally, Compressed-NMF operates as follows: 1) compute two approximate orthonormal bases for  $\mathbf{X}$  and  $\mathbf{X}^\top$  as  $\mathbf{Q}_\mathbf{X} = \text{RRF}(\mathbf{X}, k, \rho, \omega)$  and  $\mathbf{Q}_{\mathbf{X}^\top} = \text{RRF}(\mathbf{X}^\top, k, \rho, \omega)$  respectively and 2) alternatingly solve the two problems

$$\min_{\mathbf{H} \geq 0} \|\mathbf{Q}_\mathbf{X}^\top (\mathbf{W}\mathbf{H}^\top - \mathbf{X})\|_F \quad \text{and} \quad \min_{\mathbf{W} \geq 0} \|\mathbf{Q}_{\mathbf{X}^\top}^\top (\mathbf{H}\mathbf{W}^\top - \mathbf{X}^\top)\|_F$$

The products  $\mathbf{Q}_\mathbf{X}^\top \mathbf{X}$  and  $\mathbf{Q}_{\mathbf{X}^\top}^\top \mathbf{X}^\top$  can be computed once but  $\mathbf{Q}_\mathbf{X}^\top \mathbf{H}$  and  $\mathbf{Q}_\mathbf{X}^\top \mathbf{W}$  need be recomputed each iteration. In this way a smaller problem is solved at each iteration resulting in computational savings. This method can be used for SymNMF by applying it to Eq. (2.3). Additionally, in the symmetric case one need call the RRF only once.

We relate Compressed-NMF to LAI-NMF in the following way. Consider a QB-decomposition computed using the RRF where  $\mathbf{Q}_\mathbf{X} = \text{RRF}(\mathbf{X}, k, \rho, \omega)$  and  $\mathbf{B}_\mathbf{X} = \mathbf{Q}_\mathbf{X}^\top \mathbf{X}$ . Using the QB decomposition for LAI-NMF in Eq. (3.1), the update to  $\mathbf{H}$  and its Quadratic Program (QP) form are

$$\min_{\mathbf{H} \geq 0} \|\mathbf{W}\mathbf{H}^\top - \mathbf{Q}_\mathbf{X}\mathbf{B}_\mathbf{X}\|_F \leftrightarrow \min_{\mathbf{H} \geq 0} \text{tr}\left(\frac{1}{2}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^\top - \mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{B}_\mathbf{X}\right).$$

Taking the QP formulation for the Compressed NMF update for  $\mathbf{H}$ ,  $\min_{\mathbf{H} \geq 0} \|\mathbf{Q}_\mathbf{X}^\top (\mathbf{W}\mathbf{H}^\top - \mathbf{X})\|_F^2$ , yields

$$\begin{aligned} & \min_{\mathbf{H} \geq 0} \text{tr}\left(\frac{1}{2}\mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{Q}_\mathbf{X}^\top\mathbf{W}\mathbf{H}^\top - \mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{Q}_\mathbf{X}^\top\mathbf{X}\right) \\ &= \min_{\mathbf{H} \geq 0} \text{tr}\left(\frac{1}{2}\mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{Q}_\mathbf{X}^\top\mathbf{W}\mathbf{H}^\top - \mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{B}_\mathbf{X}\right). \end{aligned}$$

The term on the right,  $\mathbf{H}\mathbf{W}^\top\mathbf{Q}_\mathbf{X}\mathbf{B}_\mathbf{X}$ , is the same for both problems. The difference between the two methods comes from the fact that the Gram matrix in LAI-NMF is  $\mathbf{W}^\top\mathbf{W}$  but the Gram in Compressed-NMF contains the projection  $\mathbf{Q}_\mathbf{X}\mathbf{Q}_\mathbf{X}^\top$ . Empirically, for SymNMF, we find that these methods perform nearly identically. From this perspective we argue the effect of  $\mathbf{Q}_\mathbf{X}\mathbf{Q}_\mathbf{X}^\top$  in the Gram matrix is minimal. Overall we propose that the LAI approach is easier to reason about, as it simply fits an NMF model of the low-rank approximation of  $\mathbf{X}$ . As previously stated, using the basis  $\mathbf{Q}_\mathbf{X}$  as a sketch for NLS problems is not justified theoretically.

**B.2. Low-rank Approximate Input Projected Gauss-Newton with Conjugate Gradients for SymNMF.** This section contains the pseudo code for the PGNCG method with low-rank approximate inputs. The original algorithm was developed for high performance distributed computing environments [22]. The Pseudocode is given in Algorithm LAI-PGNCG-SymNMF. The only alterations to this pseudo code are lines 3 and 7. This highlights how simple the idea of LAI-SymNMF is in practice.

**Appendix C. Stopping Criteria.** Having a stopping criteria is often useful for NMF algorithms. In this work we use two measures the Normalized Residual and the Projected Gradient to measure convergence.

---

**Algorithm LAI-PGNCG-SymNMF** : SymNMF of a Low-Rank Approximation  
 $[\mathbf{H}]$ 


---

**Require:** **input:** a symmetric matrix  $\mathbf{X} \in \mathbb{R}^{m \times m}$ , target rank  $k$ , column oversampling parameter  $\rho$ , power iteration exponent  $q$ , and max number of CG iterations  $s_{max}$

**Ensure:**  $\{\mathbf{H}\}$  as the factors for an approximate rank- $k$  SymNMF of  $\mathbf{X}$ .

```

1: function  $[\mathbf{H}] = \text{LAI-SYMNMF}(\mathbf{X}, k, \rho, \omega)$ 
2:    $l := k + \rho$ 
3:    $[\mathbf{U}, \mathbf{\Lambda}] := \text{Apx-EVD}(\mathbf{X}, k, \rho, q)$ , % Obtain approximate basis for range of  $\mathbf{X}$ ,
    $\mathbf{U} \in \mathbb{R}^{m \times l}$  and  $\mathbf{T} \in \mathbb{R}^{l \times l}$ 
4:   Initialize  $\mathbf{H} \in \mathbb{R}^{n \times k}$ 
5:   while Convergence Crit. Not Met do
6:      $\mathbf{Z} = \mathbf{0}_{n \times k}$ 
7:      $\mathbf{R} = -2(\mathbf{U}\mathbf{\Lambda}(\mathbf{U}^\top\mathbf{H}) - \mathbf{H}(\mathbf{H}^\top\mathbf{H}))$            ▷ This replaces  $\mathbf{X}\mathbf{H} - \mathbf{H}(\mathbf{H}^\top\mathbf{H})$ 
8:      $\mathbf{P} = \mathbf{R}$ 
9:      $e_{old} = \|\mathbf{R}\|_F^2$ 
10:    for  $s = 1 : s_{max}$  do
11:       $\mathbf{Y} = 2(\mathbf{P}\mathbf{H}^\top\mathbf{H} + \mathbf{H}\mathbf{P}^\top\mathbf{H})$ 
12:       $\alpha = e_{old}/(\sum_{i,j} \mathbf{P}_{ij}\mathbf{Y}_{ij})$ 
13:       $\mathbf{Z} = \mathbf{Z} + \alpha\mathbf{P}$ 
14:       $\mathbf{R} = \mathbf{R} - \alpha\mathbf{Y}$ 
15:       $e_{new} = \|\mathbf{R}\|_F^2$ 
16:       $e_{old} = e_{new}$ 
17:    end for
18:     $\mathbf{H} = \max(\mathbf{H} - \mathbf{Z}, 0)$ 
19:  end while
20: end function

```

---

**C.1. Residual Checks.** The NMF algorithms often require computation of the residual  $\|\mathbf{X} - \mathbf{WH}^\top\|_F$  or the normalized residual

$$(C.1) \quad \frac{\|\mathbf{X} - \mathbf{WH}^\top\|_F}{\|\mathbf{X}\|_F}.$$

However, checking the residual working with the full matrix  $\mathbf{X}$  can be computationally expensive. Since checking the residual requires working with the full data matrix  $\mathbf{X}$  it can potentially dominate the run time of a randomized algorithm. Therefore, it is important to have an appropriate method for estimating the residual at each iteration [36, 1].

*Residual Computation for LAI-NMF.* For **LAI-SymNMF** the issue is easily remedied as we can simply check the residual against the factored form of  $\mathbf{X} \approx \mathbf{UV}^\top$  as

$$(C.2) \quad \frac{\|\mathbf{UV}^\top - \mathbf{WH}^\top\|_F}{\|\mathbf{UV}^\top\|_F}.$$

This allows for rapid evaluation of an approximate residual using techniques similar to that used for NMF. Our experiments show this is often a reasonable approach to use.

**C.2. Fast Residual Evaluation in NMF.** The NMF residual is given by  $r^2 = \|\mathbf{X} - \mathbf{WH}^\top\|_F^2$ . This quantity can be computed cheaply by reusing already computed quantities from the previous update. To see this observe that

$$\begin{aligned}\|\mathbf{X} - \mathbf{WH}^\top\|_F^2 &= \text{tr}\left((\mathbf{X} - \mathbf{WH}^\top)^\top(\mathbf{X} - \mathbf{WH}^\top)\right) \\ &= \text{tr}(\mathbf{X}^\top \mathbf{X}) + \text{tr}(\mathbf{HW}^\top \mathbf{WH}^\top) - 2\text{tr}(\mathbf{W}^\top \mathbf{XH}).\end{aligned}$$

The  $\text{tr}(\mathbf{X}^\top \mathbf{X})$  need only be computed one time, computing  $\text{tr}(\mathbf{HW}^\top \mathbf{WH}^\top)$  is cheap as long as  $k << \min(m, n)$  and can reuse the most recent Gram matrix (either  $\mathbf{W}^\top \mathbf{W}$  or  $\mathbf{H}^\top \mathbf{H}$ ), and  $2\text{tr}(\mathbf{W}^\top \mathbf{XH})$  can be computed cheaply by reusing the more recently computed product with  $\mathbf{X}$  depending on the update order of  $\mathbf{W}$  and  $\mathbf{H}$ . This method can easily be used with LAI-SymNMF to compute  $\|\mathbf{U}_X \mathbf{\Lambda}_X \mathbf{U}_X^\top - \mathbf{HH}^\top\|_F^2$  at each iteration. Therefore we are able to evaluate the residual for almost free at each iteration of the algorithm. This is important for determining when to stop iterating.

**C.3. Projected Gradients.** The norm of the projected gradient is often used as a stopping criteria for NMF algorithms in place of the residual. The idea is that when the NMF objective is at a stationary point, according to the Karush-Kuhn-Tucker (KKT) conditions, the projected gradients with respect to  $\mathbf{W}$  and  $\mathbf{H}$  will be 0. Therefore we assume that if the projected gradients are small then we are close to a stationary point.

The projected gradient norm is defined as

$$(C.3) \quad \sqrt{\|\nabla f_W\|_F^2 + \|\nabla f_H\|_F^2}$$

where the partial gradients are

$$(C.4) \quad \nabla f_W = 2(\mathbf{WH}^\top \mathbf{H} - \mathbf{AH})$$

$$(C.5) \quad \nabla f_H = 2(\mathbf{HW}^\top \mathbf{W} - \mathbf{A}^\top \mathbf{W})$$

and the projected gradient is

$$(C.6) \quad (\nabla^p f_W)_{ij} = \begin{cases} (\nabla f_W)_{ij} & \text{if } (\nabla f_W)_{ij} < 0 \text{ or } \mathbf{W}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

similar for the partial gradient with respect to  $\mathbf{H}$ . One issue with using the projected gradient as a comparison measure between algorithms is that different diagonal scalings of  $\mathbf{W}$  and  $\mathbf{H}$  can give different projected gradient values [32, 33]. However this is not an issue with the SymNMF objective. The gradient of Eq. (2.2) is given by

$$(C.7) \quad \nabla f_H = 4(\mathbf{HH}^\top - \mathbf{X})\mathbf{H}$$

**Appendix D. Adaptive Randomized Range Finder.** Using the standard “trick” for efficiently checking the residual we can derive the following formula which we use in Algorithm Ada-RRF :

$$\begin{aligned}\|\mathbf{QB} - \mathbf{X}\|_F^2 &= \|\mathbf{X}\|_F^2 - 2\text{tr}(\mathbf{Q}^\top \mathbf{XB}^\top) + \text{tr}(\mathbf{B}^\top \mathbf{Q}^\top \mathbf{QB}) \\ &= \|\mathbf{X}\|_F^2 - 2\text{tr}(\mathbf{BB}^\top) + \text{tr}(\mathbf{B}^\top \mathbf{B}) = \|\mathbf{X}\|_F^2 - \text{tr}(\mathbf{BB}^\top)\end{aligned}$$

To check the residual of our LAI we need only compute the matrix  $\mathbf{B}$ . This  $\mathbf{B}$  can then be used in the next power iteration if needed.

**Algorithm Ada-RRF** Adaptive Randomized Range Finder

---

**Require:** **input:** data matrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$ , target rank  $r$ , oversampling parameter  $\rho$ , maximum number of power iterations  $q_{max}$

**Ensure:**  $\mathbf{Q}_Y \in \mathbb{R}^{m \times l}$

```

1: function  $[\mathbf{U}, \mathbf{\Lambda}] = \text{APX-EVD}(X, r, \rho, \omega)$ 
2:    $l := r + \rho$ 
3:   Draw a Gaussian Random matrix  $\mathbf{\Omega} \in \mathbb{R}^{n \times l}$ 
4:   Compute  $\mathbf{Q} = qr(\mathbf{X}\mathbf{\Omega}) \in \mathbb{R}^{m \times l}$ 
5:   while  $j \leq q_{max}$  do
6:      $\mathbf{B}^\top = \mathbf{X}^\top \mathbf{Q} \in \mathbb{R}^{n \times l}$ 
7:     Evaluate  $\|\mathbf{QB} - \mathbf{X}\|_F^2 = \|\mathbf{X}\|_F^2 - \text{tr}(\mathbf{BB}^\top)$ 
8:      $\mathbf{Q} = qr(\mathbf{B}^\top) \in \mathbb{R}^{n \times l}$ 
9:     if Stopping Criteria satisfied then
10:      break
11:    end if
12:     $\mathbf{Y} = \mathbf{X}\mathbf{Q} \in \mathbb{R}^{m \times l}$ 
13:     $\mathbf{Q} = qr(\mathbf{Y}, 0)$ 
14:  end while
15:  Return  $\mathbf{Q}$ 
16: end function

```

---

**Appendix E. Update() Function.** In this section we thoroughly explain the  $\text{Update}(\mathbf{G}, \mathbf{Y})$  function we use to simplify our pseudocode. This function takes in two matrices  $\mathbf{G}$  which is  $k \times k$  and  $\mathbf{Y}$  which is  $k \times m$  or  $k \times n$ . The Framework for Alternating Updating NMF (FAUN) was proposed for the design of a massively parallel NMF code [29]. In the FAUN the matrix products  $\mathbf{X}^\top \mathbf{W}$ ,  $\mathbf{XH}$ ,  $\mathbf{W}^\top \mathbf{W}$ , and  $\mathbf{H}^\top \mathbf{H}$  are computed and used to perform updates. Many of the most effective NMF update rules require the computation of these 4 matrices as they appear in the gradient of the NMF NLS subproblems

$$(E.1) \quad \min_{\mathbf{W} \geq 0} \|\mathbf{X} - \mathbf{WH}^\top\|_F$$

and

$$(E.2) \quad \min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{WH}^\top\|_F.$$

Many methods can be implemented using the FAUN. We now briefly discuss a few of the more popular methods.

*Multiplicative Updates.* is one of the most popular methods for performing NMF updates. Proposed in [39], it is guaranteed to non increase the objective function and uses the rules

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij} \frac{(\mathbf{XH})_{ij}}{(\mathbf{WH}^\top \mathbf{H})_{ij}} \quad \text{and} \quad \mathbf{H}_{ij} \leftarrow \mathbf{H}_{ij} \frac{(\mathbf{X}^\top \mathbf{W})_{ij}}{(\mathbf{HW}^\top \mathbf{W})_{ij}}.$$

*Hierarchical Least Squares.* (HALS) uses the following update rules

$$\mathbf{w}_i \leftarrow \left[ \mathbf{w}_i + \frac{(\mathbf{XH})_i - (\mathbf{WH}^\top \mathbf{H})_i}{(\mathbf{H}^\top \mathbf{H})_{ii}} \right]_+ \quad \text{and} \quad \mathbf{h}_i \leftarrow \left[ \mathbf{h}_i + \frac{(\mathbf{X}^\top \mathbf{W})_i - (\mathbf{HW}^\top \mathbf{W})_i}{(\mathbf{W}^\top \mathbf{W})_{ii}} \right]_+.$$

All the columns of  $\mathbf{W}$  and  $\mathbf{H}$  are updated in sequence. This method is popular for its good convergence properties and its simplicity to implement.

*Alternating Nonnegative Least Squares.* ANLS updates the full  $\mathbf{W}$  and  $\mathbf{H}$  matrices in an alternating fashion by computing the optimal solutions for the NLS problems Eqns. (E.2) and (E.1). These NLS problems are equivalent to solving the following  $m + n$  quadratic programs (QPs)

$$\begin{aligned} \min_{\hat{\mathbf{w}}_j \geq 0} & \left( \hat{\mathbf{w}}_j^\top \mathbf{H}^\top \mathbf{H} \hat{\mathbf{w}}_j - \hat{\mathbf{w}}_j^\top \mathbf{H}^\top \hat{\mathbf{x}}_j \right) \text{ for } j = 1 : m \\ \min_{\hat{\mathbf{h}}_t \geq 0} & \left( \hat{\mathbf{h}}_t^\top \mathbf{W}^\top \mathbf{W} \hat{\mathbf{h}}_t - \hat{\mathbf{h}}_t^\top \mathbf{W}^\top \mathbf{x}_t \right) \text{ for } t = 1 : n \end{aligned}$$

for every row vector  $\hat{\mathbf{w}}_j$  and  $\hat{\mathbf{h}}_t$  of  $\mathbf{W}$  and  $\mathbf{H}$ , respectively.

It is simple to see that all of these rules rely on the aforementioned four matrix products. Our proposed algorithms LAI-SymNMF and LvS-SymNMF can use any of these updates rules, and more e.g. projected gradient methods for solving NLS problems [40].

## Appendix F. Theorems for Randomized Numerical Linear Algebra.

This section contains a number of theorems that we use from the Randomized NLA literature.

**F.1. Structural Condition Theorems.** We include statements of the two Structural Conditions taken from the work by Larsen and Kolda [37]. Prior versions and similar statements of this result are originally from [56, 41]. These theorems are included for completeness and reference.

**THEOREM F.1.** *Given  $\mathbf{A} \in \mathbb{R}^{m \times k}$  consider its SVD  $\mathbf{U}\Sigma\mathbf{V}^\top$  and its row leverage scores  $l_i(\mathbf{A})$  for each row  $i \in [m]$ , where  $[m]$  denotes the set of integers from 1 to  $m$ . Let  $\hat{l}_i(\mathbf{A})$  be an overestimate of the leverage score such that for some  $\beta \leq 1$  it holds that  $p_i(\hat{l}_i(\mathbf{A})) \geq \beta * p_i(l_i(\mathbf{A}))$  for all  $i \in [m]$ , where  $p_i(l_i(\mathbf{A}))$  denotes the probability corresponding to the  $i$ th leverage score. Construct a row sampling and rescaling matrix  $\mathbf{S} \in \mathbb{R}^{s \times m}$  via importance sampling according to the leverage score overestimates  $\hat{l}_i(\mathbf{A})$ . If  $s > \frac{144k \log(2k/\delta)}{\beta\epsilon^2}$  then the following equation holds with at least probability  $1 - \delta$*

$$1 - \epsilon \leq \sigma_i^2(\mathbf{SU}) \leq 1 + \epsilon$$

for all  $i \in [m]$  and  $\epsilon, \delta \in (0, 1)$ .

This theorem tells us that all the singular values of  $\mathbf{SU}$  are close to 1 if we take enough samples  $s$ . Note that this implies

$$(F.1) \quad \|\mathbf{I} - \mathbf{U}^\top \mathbf{S}^\top \mathbf{S} \mathbf{U}\|_2 < \epsilon.$$

**F.2. Randomized Matrix Multiply.** For reference we state a result originally from [12].

**THEOREM F.2.** *Consider two matrices  $\mathbf{A} \in \mathbb{R}^{k \times m}$  and  $\mathbf{B} \in \mathbb{R}^{k \times n}$  and their product  $\mathbf{A}^\top \mathbf{B} = \mathbf{C}$ . Construct a sampling matrix  $\mathbf{S}$  with  $s$  rows that are chosen according to the probability distribution  $\mathbf{p} \in [0, 1]^n$  with a  $\beta > 0$  such that  $\mathbf{p}_i \geq \beta \|\mathbf{A}[i, :] \|_2^2 / \|\mathbf{A}\|_F^2$  for all  $i \in [m]$ . Define  $\mathbf{t}$  as vector of length  $s$  of sampled indices such that  $\mathbf{t}_i$  is the index of the  $i$ th sampled row. Consider the approximate matrix product*

$$\frac{1}{s} \sum_i^s \mathbf{A}[\mathbf{t}_i, :]^\top \mathbf{B}[\mathbf{t}_i, :] = \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{B}$$

Then the approximate matrix product satisfies

$$\mathbb{E} \left[ \|\mathbf{A}^\top \mathbf{B} - \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{B}\|_F^2 \right] \leq \frac{1}{\beta s} \|\mathbf{A}\|_F^2 \|\mathbf{B}\|_F^2$$

Applying Theorem (F.2) in conjunction with Markov's Inequality we obtain the following Lemma

LEMMA F.3. Taking  $s \geq \frac{2\|\mathbf{A}\|_F^2}{\beta\delta\epsilon_r}$  samples where  $\delta, \epsilon_r \in (0, 1)$  results in

$$\|\mathbf{A}^\top \mathbf{B} - \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{B}\|_F^2 \geq \frac{\epsilon_r \|\mathbf{B}\|}{2}$$

with probability  $\delta$ .

*Proof.* Applying Markov's Inequality we have

$$\begin{aligned} \Pr \left[ \|\mathbf{A}^\top \mathbf{B} - \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{B}\|_F^2 \geq \frac{\epsilon_r \|\mathbf{B}\|_F^2}{2} \right] &\leq \frac{2\mathbb{E}[\|\mathbf{A}^\top \mathbf{B} - \mathbf{A}^\top \mathbf{S}^\top \mathbf{S} \mathbf{B}\|_F^2]}{\epsilon_r \|\mathbf{B}\|_F^2} \\ &\leq \frac{\|\mathbf{A}\|_F^2 \|\mathbf{B}\|_F^2}{\beta s \epsilon_r \|\mathbf{B}\|_F^2} = \frac{\|\mathbf{A}\|_F^2}{\beta s \epsilon_r} \end{aligned}$$

For the bound to hold with probability  $\delta$  we must have  $s \geq \frac{2\|\mathbf{A}\|_F^2}{\beta\delta\epsilon_r}$ .  $\square$

**F.3. Matrix Chernoff Bounds.** This is a Matrix Chernoff Bound taken from [56] which is used to show the validity of the Hybrid Sampling leverage scores matrix.

**THEOREM F.4. Matrix Chernoff Bound:** Let  $\mathbf{X}_1, \dots, \mathbf{X}_s$  be independent copies of a symmetric random matrix  $\mathbf{X} \in \mathbb{R}^{k \times k}$  with  $\mathbb{E}[\mathbf{X}] = 0$ ,  $\|\mathbf{X}\|_2 \leq \gamma$ , and  $\|\mathbb{E}[\mathbf{X}^\top \mathbf{X}]\|_2 \leq \nu^2$ . Let  $\mathbf{W} = \frac{1}{s} \sum_{i=1}^s \mathbf{X}_i$ . Then for any  $\epsilon > 0$  we have

$$(F.2) \quad \Pr[\|\mathbf{W}\|_2 > \epsilon] \leq 2k \exp(-s\epsilon^2/(2\nu^2 + 2\gamma\epsilon/3))$$

**Appendix G. Additional Experimental Data.** This section contains some additional data and experiments related to the experiments on the WoS and OAG data sets.

**G.1. World of Science Data Set.** This section contains some additional experiments on the World of Science data set. The first set of experiments are concerned with varying  $\rho$  or the column over sampling parameter in the RRF. The observation here is that increasing  $\rho$  does not seem to have a beneficial impact on clustering quality or final residual. Figure 1 shows the runs for the various algorithms with  $\rho = 40$  and  $\rho = 80$ . Table 4 and Table 5 give various metrics associated with these runs. Each algorithm was run 10 times. “Iters” is average number of iterations, “Time” is the average time in seconds, “Avg. Min-Res” is the average minimum residual achieved, “Min-Res” is the overall minimum residual achieved, and “Mean-ARI” is the mean of the ARI scores for each algorithm.

The second set of experiments are run with  $q = 2$  and without the use of Algorithm Ada-RRF. Table 6 gives data for this run. One can observe that method without IR do not achieve high quality residual or ARI scores. While IR can help fix these issues it does so at a higher computational cost. The convergence plots for these experiments are in Figures 5a to 5c. Contrasting these with the results in Table 2 we conclude that using Ada-RRF to obtain an appropriate  $q$  is generally more efficient in terms of reducing residual and overall run time than using a static choice of  $q = 2$ .

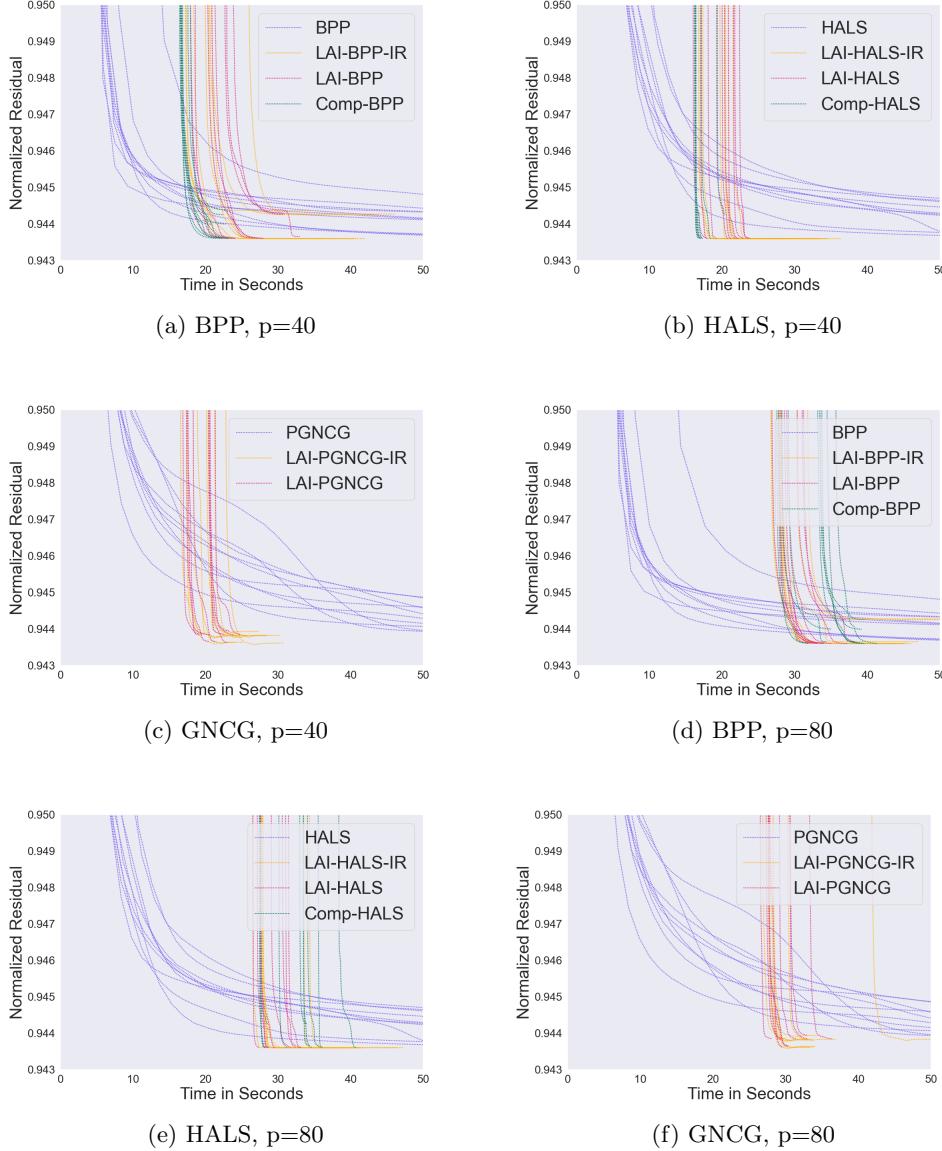


Fig. 4: Normalized Residual error value for various SymNMF Algorithms on the WoS data set using an EDVW Hypergraph Representation.

*WoS Experiments System Details.* The WoS experiments were run on a MacBook Pro with MATLAB 2021a. The MacBook Pro has a 2.3 GHz Quad-Core Intel Core i7 processor and 16 GBs of RAM. MATLAB was given access to all 4 cores.

Alg.	Iters	Time	Avg. Min-Res	Min-Res	Mean-ARI
BPP	38.1	66.95	0.9439	0.9436	0.3224
LAI-BPP	70.7	27.25	0.9437	0.9436	0.3234
LAI-BPP-IR	76.7	39.366	0.9438	0.9436	0.3242
LAI-HALS	85.2	21.214	0.9436	0.9436	0.3292
LAI-HALS-IR	89.2	32.429	0.9436	0.9436	0.3284
HALS	58.1	92.461	0.9437	0.9436	0.3201
GNCG	50.8	80.311	0.944	0.9437	0.3063
LAI-GNCG	90.4	21.877	0.9438	0.9436	0.3116
Comp-BPP	69.5	22.472	0.9438	0.9436	0.3163
LAI-GNCG-IR	97.0	27.053	0.9437	0.9436	0.3107
Comp-HALS	75.7	17.533	0.9437	0.9436	0.3254

Table 4: Data Table for the WoS data set. Each algorithm was run 10 times. For LIA methods auto- $q$  and  $p = 40$ . The columns record average number of iterations until the stopping criteria is met, average time in seconds, average minimum residual, and minimum residual achieved over all runs.

Alg.	Iters	Time	Avg. Min-Res	Min-Res	Mean-ARI
BPP	38.1	66.95	0.9439	0.9436	0.3224
LAI-BPP	66.9	34.824	0.9437	0.9436	0.3271
LAI-BPP-IR	72.3	46.136	0.9437	0.9436	0.3225
LAI-HALS	82.8	30.167	0.9436	0.9436	0.3298
LAI-HALS-IR	87.0	41.765	0.9436	0.9436	0.329
HALS	58.1	92.461	0.9437	0.9436	0.3201
GNCG	50.8	80.311	0.944	0.9437	0.3063
LAI-GNCG	85.5	31.1	0.9438	0.9436	0.3111
BComp-PP	61.8	37.139	0.9438	0.9436	0.3149
GNCG-IR	89.4	35.702	0.9437	0.9436	0.3117
Comp-HALS	74.5	33.205	0.9437	0.9436	0.3257

Table 5: Data Table for the WoS data set. Each algorithm was run 10 times. For LIA methods auto- $q$  and  $p = 80$ . The columns record average number of iterations until the stopping criteria is met, average time in seconds, average minimum residual, and minimum residual achieved over all runs.

**G.2. Microsoft Open Academic Graph Experiments.** This section contains additional data related to the Microsoft OAG experiments. Table 7 shows the top 10 words for each cluster found by the HALS algorithm and Table 8 shows the top 10 words for each cluster found by the LVS-HALS algorithm.

*System Details.* Experiments on the OAG dataset were run on the Hive Cluster at the Georgia Institute of Technology. The runs were given access to eight Xeon 6226 CPU @ 2.70GHz on a single shared memory node. Experiments were run in MATLAB version 2019a.

**G.2.1. Leverage Scores and Deterministic Sampling.** Figures 6a and 6b show two sets of statistics about the LvS-HALS method when applied to the OAG

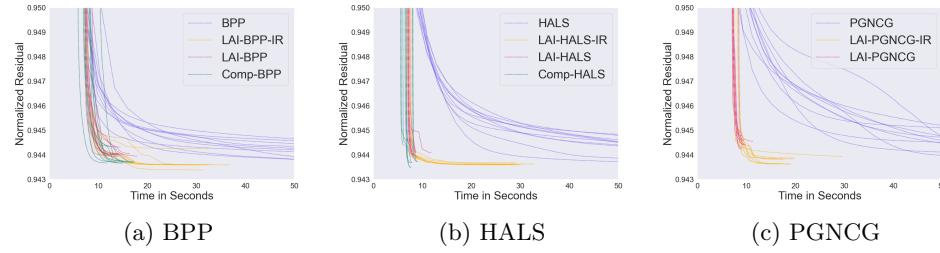


Fig. 5: Normalized Residual Norm Plots for the WoS data set. Three different update rules are shown: BPP, HALS, and PGNCG. These experiments use  $q = 2$  and do not make use of [Algorithm Ada-RRF](#).

Alg.	Iters	Time	Avg. Min-Res	Min-Res	Mean-ARI
GNCG	50.8	99.55	0.944	0.9437	0.3063
LAI-GNCG	72.6	9.5	0.9445	0.9443	0.2993
LAI-GNCG-IR	77.2	19.336	0.9438	0.9436	0.3072
BPP	38.1	83.603	0.9439	0.9436	0.3224
LAI-BPP	73.7	14.52	0.944	0.9439	0.3162
LAI-BPP-IR	83.5	32.027	0.9436	0.9434	0.3227
Comp-BPP	80.0	15.079	0.9437	0.9437	0.3254
LAI-HALS	95.3	8.699	0.944	0.9439	0.3242
LAI-HALS-IR	79.7	29.024	0.9436	0.9436	0.3297
HALS	58.1	111.332	0.9437	0.9436	0.3201
Comp-HALS	74.0	7.806	0.9439	0.9435	0.314

Table 6: Data Table for the WoS data set. Each algorithm was run 10 times. For LAI methods  $q = 2$ , no Ada-RRF is used. The columns record average number of iterations until the stopping criteria is met, average time in seconds, average minimum residual, and minimum residual achieved over all runs. Contrasting this with Figure 1 and Table 2 shows that using Ada-RRF is beneficial for the WoS data set.

Topic	TW1	TW2	TW3	TW4	TW5	TW6	TW7	TW8	TW9	TW10
0	zone	mRNA	receptor	apoptosis	kinase	receptors	peptide	PCR	gene	intracellular
1	teaching	article	calculation	mathematical	summary	finite	engineering	boundary	modern	beam
2	summary	hand	engineering	industry	law	mathematical	article	electric	science	calculation
3	teaching	engineering	integration	fast	summary	boundary	calculation	china	principle	principles
4	summary	equations	article	gas	modern	calculation	teaching	principle	mathematical	reliability
5	beam	teaching	integration	hand	article	summary	equations	mathematical	project	equation
6	hand	teaching	summary	engineering	project	article	mathematical	calculation	electric	technical
7	situation	summary	article	calculation	finite	teaching	engineering	hand	industrial	software
8	teaching	summary	calculation	actual	article	hand	principle	engineering	mathematical	law
9	entire	teaching	summary	project	principle	article	engineering	china	electric	mathematical
10	teaching	hand	article	calculation	summary	engineering	international	beam	finite	industrial
11	summary	teaching	china	article	calculation	construction	species	mathematical	separation	engineering
12	summary	principle	article	china	teaching	beam	calculation	law	technical	equation
13	china	summary	teaching	project	modern	hand	computer	industry	technical	principle
14	finite	teaching	mathematical	engineering	numbers	summary	beam	construction	principle	american
15	industry	summary	calculation	teaching	situation	engineering	equation	industrial	law	supply

Table 7: Top key words for HALS output. Run on the Microsoft Open Academic Graph.

Topic	TW1	TW2	TW3	TW4	TW5	TW6	TW7	TW8	TW9	TW10
0	zone	teaching	construction	forward	situation	law	education	chinese	china	industry
1	teaching	industrial	service	supply	actual	summary	industry	calculation	international	simulation
2	thermal	limits	equilibrium	summary	geometry	regional	heat	hand	fast	units
3	transform	signals	machine	voltage	power	parallel	signal	feature	classification	space
4	beam	storage	hand	particle	operation	modes	coupled	calculation	integration	china
5	beam	project	teaching	existence	mathematical	prove	proved	tool	modern	plane
6	equation	stability	functional	prove	physics	beam	fixed	space	weak	equations
7	fast	teaching	summary	taking	supply	form	situation	finite	article	international
8	forms	fundamental	machine	teaching	mathematical	form	measuring	social	past	operating
9	summary	minimal	hand	methyl	industry	operation	maintenance	technical	integration	project
10	matter	observations	emission	cluster	ray	infrared	mass	evolution	physics	simulations
11	optical	voltage	gain	electron	strain	layers	laser	films	crystal	layer
12	determination	organic	summary	block	absorption	iii	compounds	gel	ions	chemistry
13	flow	devices	fluid	integrated	driven	electric	film	device	voltage	liquid
14	equivalent	constant	weight	forms	coefficient	half	proved	plane	establish	form
15	signaling	transcription	kinase	receptor	genes	receptors	pathways	proteins	mediated	pathway

Table 8: Topwords for Leverage Score HALS output. Selected topics from the Microsoft Open Academic Graph run using HALS as the update rule with Leverage Score sampling ( $\tau = 1/s$ ). The 10 top words in terms of tf-idf association are shown in the table. We can see many of the top key words per topic seem to form a coherent subject matter.

data set. The two different bars distinguished by L and R denote the Left and Right factors of the SymNMF problem. Figure 6a shows the percentage (y-axis) of the total fraction of samples that are being taken deterministically at each iteration ( $\frac{s_D}{s_R+s_D}$ ). Note that values are only plotted for every 5th iteration (x-axis). This shows a clear trend towards taking fewer and fewer samples as the iterations progress. Figure 6b shows that fraction of leverage score mass or  $\frac{\ell}{k}$  that is accounted for by the deterministic samples at each iteration. The amount of leverage score mass being accounted for quickly approaches 1. This means that a small number of deterministic samples are accounting for nearly all of the leverage score mass in the computed factor  $\mathbf{H}$ .

## REFERENCES

- [1] C. BATTAGLINO, G. BALLARD, AND T. G. KOLDA, *A practical randomized CP tensor decomposition*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 876–901, <https://doi.org/10.1137/17M1112303>.
- [2] V. BHARADWAJ, O. A. MALIK, R. MURRAY, A. BULUÇ, AND J. DEMMEL, *Distributed-memory randomized algorithms for sparse tensor CP decomposition*, Oct 2023, <https://arxiv.org/abs/2210.05105>.
- [3] V. BHARADWAJ, O. A. MALIK, R. MURRAY, L. GRIGORI, A. BULUC, AND J. DEMMEL, *Fast exact leverage score sampling from Khatri-Rao products with applications to tensor decomposition*, 2023, <https://arxiv.org/abs/2301.12584>.
- [4] C. BOUTSIDIS AND P. DRINEAS, *Random projections for the nonnegative least-squares problem*, Linear Algebra and its Applications, 431 (2009), pp. 760–771, <https://doi.org/https://doi.org/10.1016/j.laa.2009.03.026>.
- [5] C. BOUTSIDIS AND E. GALLOPOULOS, *SVD based initialization: A head start for nonnegative matrix factorization*, Pattern Recognition, 41 (2008), pp. 1350–1362, <https://doi.org/https://doi.org/10.1016/j.patcog.2007.09.010>.
- [6] D. CHENG, R. PENG, Y. LIU, AND I. PERROS, *SPALS: Fast alternating least squares via implicit leverage scores sampling*, in Advances in Neural Information Processing Systems, vol. 29, Curran Associates, Inc., 2016, <https://proceedings.neurips.cc/paper/2016/file/f4f6dce2f3a0f9dada0c2b5b66452017-Paper.pdf>.
- [7] D. CHOI, B. DRAKE, AND H. PARK, *Co-embedding multi-type data for information fusion and visual analytics*, in 2023 26th International Conference on Information Fusion (FUSION), 2023, pp. 1–8, <https://doi.org/10.23919/FUSION52260.2023.10224157>.
- [8] D. CHOI, A. XIANG, O. OZTURK, D. SHRESTHA, B. DRAKE, H. HAIDARIAN, F. JAVED, AND H. PARK, *Wellfactor: Patient profiling using integrative embedding of healthcare data*,

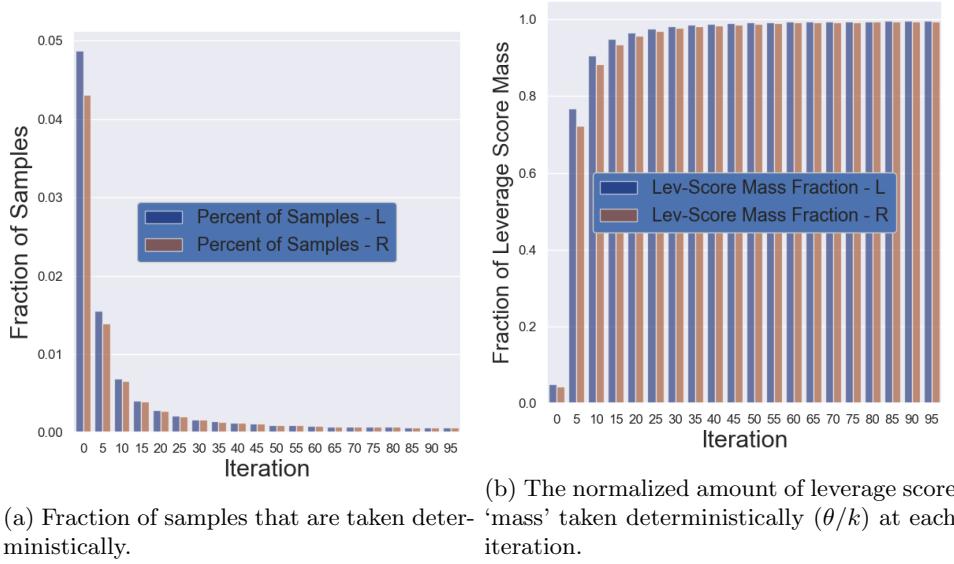


Fig. 6: Two statistics for the hybrid sampling approach in the LvS-HALS algorithm on the OAG data set.

- arXiv preprint arXiv:2312.14129, (2023).
- [9] A. CICHOCKI AND A.-H. PHAN, *Fast local algorithms for large scale nonnegative matrix and tensor factorizations*, IEICE Transactions, 92-A (2009), pp. 708–721, <https://doi.org/10.1587/transfun.E92.A.708>.
  - [10] J. E. COHEN AND R. BRO, *Nonnegative PARAFAC2: a flexible coupling approach*, 2018, <https://arxiv.org/abs/1802.05035>.
  - [11] J. W. DANIEL, *Stability of the solution of definite quadratic programs*, Mathematical Programming, 5 (1973), <https://doi.org/10.1007/BF01580110>.
  - [12] P. DRINEAS, R. KANNAN, AND M. W. MAHONEY, *Fast monte carlo algorithms for matrices I: Approximating matrix multiplication*, SIAM Journal on Computing, 36 (2006), pp. 132–157, <https://doi.org/10.1137/S0097539704442684>.
  - [13] P. DRINEAS, M. MAGDON-ISMAIL, M. W. MAHONEY, AND D. P. WOODRUFF, *Fast approximation of matrix coherence and statistical leverage*, CoRR, abs/1109.3843 (2011), <https://arxiv.org/abs/1109.3843>.
  - [14] P. DRINEAS, M. W. MAHONEY, S. MUTHUKRISHNAN, AND T. SARLÓS, *Faster least squares approximation*, CoRR, (2007), <https://arxiv.org/abs/0710.1435>.
  - [15] R. DU, B. DRAKE, AND H. PARK, *Hybrid clustering based on content and connection structure using joint nonnegative matrix factorization*, Journal of Global Optimization, 74 (2019), pp. 1 – 17, <https://doi.org/10.1007/s10898-017-0578-x>.
  - [16] R. DU, D. KUANG, B. DRAKE, AND H. PARK, *DC-NMF: nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling*, Journal of Global Optimization, 68 (2017), pp. 777–798, <https://www.proquest.com/scholarly-journals/dc-nmf-nonnegative-matrix-factorization-based-on/docview/1918796586/se-2>.
  - [17] R. DU, D. KUANG, B. DRAKE, AND H. PARK, *Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization*, Computational Social Networks, 4 (2017), pp. 1 – 26, <https://doi.org/10.1186/s40649-017-0043-5>.
  - [18] N. B. ERICHSON, K. MANOHAR, S. L. BRUNTON, AND J. N. KUTZ, *Randomized CP tensor decomposition*, CoRR, (2017), <https://arxiv.org/abs/1703.09074>.
  - [19] N. B. ERICHSON, A. MENDIBLE, S. WIHLBORN, AND J. N. KUTZ, *Randomized nonnegative matrix factorization*, Pattern Recognition Letters, 104 (2018), pp. 1–7, <https://doi.org/10.1016/j.patrec.2018.01.007>.
  - [20] S. ESWAR, B. COBB, K. HAYASHI, R. KANNAN, G. BALLARD, R. VUDUC, AND H. PARK,

- Distributed-memory parallel jointnmf*, in Proceedings of the 37th International Conference on Supercomputing, ICS '23, New York, NY, USA, 2023, Association for Computing Machinery, p. 301–312, <https://doi.org/10.1145/3577193.3593733>.
- [21] S. ESWAR, K. HAYASHI, G. BALLARD, R. KANNAN, M. A. MATHESON, AND H. PARK, *PLAN<sub>C</sub>: Parallel low-rank approximation with nonnegativity constraints*, ACM Trans. Math. Softw., 47 (2021), <https://doi.org/10.1145/3432185>.
  - [22] S. ESWAR, K. HAYASHI, G. BALLARD, R. KANNAN, R. VUDUC, AND H. PARK, *Distributed-memory parallel symmetric nonnegative matrix factorization*, in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1–14, <https://doi.org/10.1109/SC41405.2020.00078>.
  - [23] N. GILLIS, *Nonnegative Matrix Factorization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020, <https://doi.org/10.1137/1.9781611976410>.
  - [24] N. GILLIS AND F. GLINEUR, *Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization*, Neural Comput., 24 (2012), p. 1085–1105, [https://doi.org/10.1162/NECO\\_a\\_00256](https://doi.org/10.1162/NECO_a_00256).
  - [25] M. GU, *Subspace iteration randomization and singular value problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1139–A1173, <https://doi.org/10.1137/130938700>, <https://arxiv.org/abs/https://doi.org/10.1137/130938700>.
  - [26] N. HALKO, P.-G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, (2009), <https://doi.org/10.48550/ARXIV.0909.4061>.
  - [27] K. HAYASHI, S. G. AKSOY, C. H. PARK, AND H. PARK, *Hypergraph random walks, Laplacians, and clustering*, in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, New York, NY, USA, 2020, Association for Computing Machinery, p. 495–504, <https://doi.org/10.1145/3340531.3412034>.
  - [28] L. HOU, D. CHU, AND L.-Z. LIAO, *A progressive hierarchical alternating least squares method for symmetric nonnegative matrix factorization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 45 (2023), pp. 5355–5369, <https://doi.org/10.1109/TPAMI.2022.3206465>.
  - [29] R. KANNAN, G. BALLARD, AND H. PARK, *A high-performance parallel algorithm for nonnegative matrix factorization*, in Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '16, New York, NY, USA, 2016, Association for Computing Machinery, <https://doi.org/10.1145/2851141.2851152>.
  - [30] R. KANNAN, G. BALLARD, AND H. PARK, *MPI-FAUN: An MPI-based framework for alternating-updating nonnegative matrix factorization*, IEEE Transactions on Knowledge and Data Engineering, 30 (2018), pp. 544–558, <https://doi.org/10.1109/TKDE.2017.2767592>.
  - [31] H. KIM, J. CHOO, J. KIM, C. K. REDDY, AND H. PARK, *Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization*, in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, New York, NY, USA, 2015, Association for Computing Machinery, p. 567–576, <https://doi.org/10.1145/2783258.2783338>.
  - [32] J. KIM, Y. HE, AND H. PARK, *Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework*, J. of Global Optimization, 58 (2014), p. 285–319, <https://doi.org/10.1007/s10898-013-0035-4>.
  - [33] J. KIM AND H. PARK, *Fast nonnegative matrix factorization: An active-set-like method and comparisons*, SIAM Journal on Scientific Computing, 33 (2011), pp. 3261–3281, <https://doi.org/10.1137/110821172>.
  - [34] D. KUANG, C. DING, AND H. PARK, *Symmetric Nonnegative Matrix Factorization for Graph Clustering*, pp. 106–117, <https://doi.org/10.1137/1.9781611972825.10>, <https://epubs.siam.org/doi/abs/10.1137/1.9781611972825.10>, <https://arxiv.org/abs/https://epubs.siam.org/doi/pdf/10.1137/1.9781611972825.10>.
  - [35] D. KUANG, S. YUN, AND H. PARK, *Symnmf: Nonnegative low-rank approximation of a similarity matrix for graph clustering*, Journal of Global Optimization, 62 (2015), <https://doi.org/10.1007/s10898-014-0247-2>.
  - [36] B. W. LARSEN AND T. G. KOLDA, *Practical leverage-based sampling for low-rank tensor decomposition*, 2020, <https://arxiv.org/abs/2006.16438>.
  - [37] B. W. LARSEN AND T. G. KOLDA, *Sketching matrix least squares via leverage scores estimates*, 2022, <https://arxiv.org/abs/2201.10638>.
  - [38] D. LEE AND H. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–91, <https://doi.org/10.1038/44565>.
  - [39] D. LEE AND H. S. SEUNG, *Algorithms for non-negative matrix factorization*, in Ad-

- vances in Neural Information Processing Systems, T. Leen, T. Dietterich, and V. Tresp, eds., vol. 13, MIT Press, 2000, <https://proceedings.neurips.cc/paper/2000/file/f9d1152547c0bde01830b7e8bd60024c-Paper.pdf>.
- [40] C.-J. LIN, *Projected gradient methods for nonnegative matrix factorization*, Neural Computation, 19 (2007), pp. 2756–2779, <https://doi.org/10.1162/neco.2007.19.10.2756>.
  - [41] M. MAGDON-ISMAIL, *Row sampling for matrix algorithms via a non-commutative bernstein bound*, ArXiv, abs/1008.0587 (2010).
  - [42] M. W. MAHONEY, *Randomized algorithms for matrices and data*, CoRR, abs/1104.5557 (2011), <https://arxiv.org/abs/1104.5557>.
  - [43] M. W. MAHONEY, *Lecture notes on randomized linear algebra*, CoRR, (2016), <https://arxiv.org/abs/1608.04481>.
  - [44] P.-G. MARTINSSON AND J. TROPP, *Randomized numerical linear algebra: Foundations & algorithms*, 02 2020.
  - [45] A. Y. NG, M. I. JORDAN, AND Y. WEISS, *On spectral clustering: Analysis and an algorithm*, in Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS’01, Cambridge, MA, USA, 2001, MIT Press, p. 849–856.
  - [46] D. PAPALIOPOULOS, A. KYRILLIDIS, AND C. BOUTSIDIS, *Provable deterministic leverage score sampling*, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA, 2014, Association for Computing Machinery, p. 997–1006, <https://doi.org/10.1145/2623330.2623698>.
  - [47] P. ROUSSEEUW, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.*, Journal of Computational and Applied Mathematics, 20 (1987), pp. 53–65, [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7).
  - [48] T. SARLOS, *Improved approximation algorithms for large matrices via random projections*, in 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), 2006, pp. 143–152, <https://doi.org/10.1109/FOCS.2006.37>.
  - [49] A. SINHA, Z. SHEN, Y. SONG, H. MA, D. EIDE, B.-J. P. HSU, AND K. WANG, *An overview of microsoft academic service (mas) and applications*, in Proceedings of the 24th International Conference on World Wide Web, WWW ’15 Companion, New York, NY, USA, 2015, Association for Computing Machinery, p. 243–246, <https://doi.org/10.1145/2740908.2742839>.
  - [50] J. TANG, J. ZHANG, L. YAO, J. LI, L. ZHANG, AND Z. SU, *ArnetMiner: Extraction and mining of academic social networks*, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’08, New York, NY, USA, 2008, Association for Computing Machinery, p. 990–998, <https://doi.org/10.1145/1401890.1402008>.
  - [51] M. TEPPER AND G. SAPIRO, *Compressed nonnegative matrix factorization is fast and accurate*, IEEE Transactions on Signal Processing, 64 (2016), pp. 2269–2283, <https://doi.org/10.1109/TSP.2016.2516971>.
  - [52] A. VANDAELE, N. GILLIS, Q. LEI, K. ZHONG, AND I. S. DHILLON, *Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization*, IEEE Transactions on Signal Processing, 64 (2016).
  - [53] M. M. W., *Cur matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 697–702, <https://cir.nii.ac.jp/crid/1570009751570164352>.
  - [54] F. WANG AND P. LI, *Efficient nonnegative matrix factorization with random projections*, in SDM, 2010.
  - [55] J. J. WHANG, R. DU, S. JUNG, G. LEE, B. DRAKE, Q. LIU, S. KANG, AND H. PARK, *MEGA: Multi-view semi-supervised clustering of hypergraphs*, Proc. VLDB Endow., 13 (2020), p. 698–711, <https://doi.org/10.14778/3377369.3377378>.
  - [56] D. P. WOODRUFF, *Sketching as a tool for numerical linear algebra*, CoRR, (2014), <https://arxiv.org/abs/1411.4357>.
  - [57] W. XU, X. LIU, AND Y. GONG, *Document clustering based on non-negative matrix factorization*, in Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’03, New York, NY, USA, 2003, Association for Computing Machinery, p. 267–273, <https://doi.org/10.1145/860435.860485>.
  - [58] F. ZHANG, X. LIU, J. TANG, Y. DONG, P. YAO, J. ZHANG, X. GU, Y. WANG, B. SHAO, R. LI, AND K. WANG, *OAG: Toward linking large-scale heterogeneous entity graphs*, in ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), August 2019, <https://www.microsoft.com/en-us/research/publication/oag-toward-linking-large-scale-heterogeneous-entity-graphs/>.
  - [59] G. ZHOU, A. CICHOCKI, AND S. XIE, *Fast nonnegative matrix/tensor factorization based on low-rank approximation*, IEEE Transactions on Signal Processing, 60 (2012), pp. 2928–2940,

<https://doi.org/10.1109/TSP.2012.2190410>.

- [60] G. ZHOU, A. CICHOCKI, AND S. XIE, *Decomposition of big tensors with low multilinear rank*, ArXiv, abs/1412.1885 (2014).
- [61] Z. ZHU, X. LI, K. LIU, AND Q. LI, *Dropping symmetry for fast symmetric nonnegative matrix factorization*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Red Hook, NY, USA, 2018, Curran Associates Inc., p. 5160–5170.