Received: 17 December 2019 | Revised: 29 April 2020 | Accepted: 28 May 2020

DOI: 10.1002/gamm.202000013

ORIGINAL PAPER

GAMM Mitteilungen

A literature survey of matrix methods for data science

Martin Stoll

Chair of Scientific Computing, Department of Mathematics, TU Chemnitz, Chemnitz, Germany

Correspondence
Martin Stoll, Chair of Scientific Computing, Department of Mathematics, TU Chemnitz, Reichenhainer Str. 41, 09126 Chemnitz, Germany. Email: martin.stoll@mathematik.tu-chemnitz.de

Open access funding enabled and organized by Projekt DEAL.

Abstract

Efficient numerical linear algebra is a core ingredient in many applications across almost all scientific and industrial disciplines. With this survey we want to illustrate that numerical linear algebra has played and is playing a crucial role in enabling and improving data science computations with many new developments being fueled by the availability of data and computing resources. We highlight the role of various different factorizations and the power of changing the representation of the data as well as discussing topics such as randomized algorithms, functions of matrices, and high-dimensional problems. We briefly touch upon the role of techniques from numerical linear algebra used within deep learning.

KEYWORDS

Data Science, Numerical linear algebra

1 | INTRODUCTION

The study of extracting information from data has become crucial in many field ranging from business, engineering, fundamental research, or culture. We here assume that data science intends to analyze and understand actual phenomena with *data* according to [135]. To achieve this, we follow [80] in that data science draws on elements of machine learning, data mining and many other mathematical fields such as optimization or statistics . Also, we want to point out that in order to obtain information from data it is not necessarily implied that the amount of data is *big* but often it is.

The multitude of applications where such data occur and need to be studied goes beyond the scope of this paper. Naturally, matrices arise as part of spatial data analysis [28,116,205] or time-series data analysis [2,40,153,232,290] but can also be found in many more disciplines [279].

We here view the data as being represented in a matrix

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \in \mathbb{R}^{n,m} \tag{1}$$

with the dimensions m, n related to the underlying data. Here m is the dimension of the feature space with feature vectors $a_i \in \mathrm{R}^m$ viewed here as the rows of A. The dimension n is the number of data points and is thus possibly very large. We often assume that $n > m$. Alternatively, the data can also arise in the form of a tensor of order d

$$A \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \tag{2}$$

with dimensions $n_i \in \mathrm{N}\ \forall i$.

The tasks of extracting meaningful information from the collected data vary between application areas. In the process of extracting information from data we often encounter tasks such as *unsupervised learning*, *semi-supervised learning*, and *supervised learning* (cf [134,152]). The difference between these can roughly be summarized by the availability and usage of training data with none for unsupervised, only for a subset for semi-supervised, and for all of the data in supervised learning.

Before starting the detailed discussion, we would like to point out a particular example that is a core problem in data science, statistics, numerical linear algebra, and computer science alike. This is the least squares problem [109], where in brief one wants to fit a linear model to labeled data. Let us assume that we are given n data points $a_i \in \mathrm{R}^m$ and typically outputs y_i, for example, a value of either -1 or 1 for a classification problem. We are then interested in finding a weight vector $w \in \mathrm{R}^m$ such that the 2-norm of the residual $r_i = a_i^T w - y_i$ is minimized. It is well known that this would lead to a linear regression problem

$$\min_w \|Aw - y\|_2^2 + \lambda \|w\|_2^2 \tag{3}$$

with A as above. Here, we also introduce $\lambda > 0$ as a regularization or ridge parameter for the regularization term $\|w\|_2^2$. The regularization term could also be measured in several different norms such as the l_1-norm [273] or the total variation norm [284]. The solution to this problem is then given by

$$w_* = (A^\top A + \lambda I)^{-1} A^T y, \tag{4}$$

a prototypical linear system of equations with a symmetric and positive definite matrix as long as $\lambda > 0$. This example illustrates that the different disciplines are intertwined. Problem (3) arises in data mining relying on techniques from numerical linear algebra to make the evaluation robust and efficient. On the other hand, the development of sophisticated numerical methods is driven by studying problems with *real data*. The goal of this survey is to show how information extraction from data, data modeling, and pattern finding relies on efficient techniques from numerical linear algebra that not only enable computations but also reveal hidden information.

The paper is structured as follows. We first illustrate the use of classical factorizations such as the QR and singular value decomposition (SVD) and then introduce interpretable factorizations such as the nonnegative matrix factorization (NMF) or the CUR decomposition. We additionally discuss literature devoted to kernel methods with special attention given to the graph Laplacian. Randomization and functions of matrices are discussed next. We close with a brief discussion of recent results for high-dimensional problems and deep learning applications.

As a word of caution, we want to remark that the field of numerical linear algebra is vast and the analysis of data via techniques from numerical linear algebra is not new. The goal of this survey is to point to recent trends and we apologize to the authors whose results we missed while writing this. In particular we want to refer to the beautiful books by Eldén [92] and Strang [265] that provide general introductions to linear algebra for data science applications.

2 | DATA MATRICES AND FACTORIZATIONS

The decompositional approach to matrix computations has been named one of the top 10 algorithms of the 20th century [79]. Matrix factorizations are a ubiquitous tool in data science and have received much attention over the last years. A great example is the use of matrix factorization techniques for recommender systems such as the Netflix challenge[1] [168]. We here review some important matrix factorizations, their applications as well as tailored factorizations used for data science. We split the discussion into classical factorizations, which have been the workhorse of many applications such as engineering or fluid mechanics, and factorizations that are designed to more closely resemble the nature of the data.

[1] https://en.wikipedia.org/wiki/Netflix_Prize.

2.1 | Classical factorizations

2.1.1 | The singular value decomposition

Given a data matrix $A \in \mathbb{R}^{n,m}$, assuming $n \geq m$, a SVD [109] is given as

$$A = USV^\top$$

with $U \in \mathbb{R}^{n,n}$ and $V \in \mathbb{R}^{m,m}$ being orthogonal matrices. The matrix $S \in \mathbb{R}^{n,m}$ is of the following form

$$S = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & & \\ & & \ddots & \\ \vdots & & & \sigma_m \\ 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & 0 \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

with singular values $\sigma_1 \geq \sigma_2 \, \dots \, \geq \sigma_m \geq 0$. An equivalent and often very useful representation is the outer product form of the SVD as

$$A = \sum_{i=1}^{m} \sigma_i u_i v_i^T,$$

where u_i and v_i are the columns of U and V, respectively. This allows for a natural interpretation of U as providing a basis for the column space of A and V for its row-space. A crucial task is to find a good rank-k approximation to the matrix A. The Eckart-Young-Mirsky theorem [110] states that the best rank-k approximation in any unitarily invariant matrix norm is given as

$$A \approx A_k := \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

hereby ignoring the smaller singular values in the summation. This is known as the *truncated SVD* and is written in matrix form as

$$A_k = U_k S_k V_k^\top. \tag{5}$$

The SVD is one of the most crucial tools in the complexity reduction of large-scale problems. The SVD is naturally well suited to solve the least squares problem (3) of the form

$$w_* = (A^\top A + \lambda I)^{-1} A^\top y = (VS^\top U^T USV^\top + \lambda I)^{-1} VS^\top U^\top y = V(S^\top S + \lambda I)^{-1} S^\top U^\top y. \tag{6}$$

This is expensive due to the cost of computing the full SVD but the truncated SVD can be exploited for solving least squares problems as discussed in [132]. There have been many algorithmic updates in the computation of the (truncated) SVD that deal with the numerical difficulties of large-scale problems, rounding errors, locking of wanted singular vectors, and purging of unwanted information [14,139,146,263]. At the heart often lies the Lanczos bidiagonalization and for large scale problems incorporating implicit restarts is mandatory (cf [14,30,140,263]). The algorithmic foundations for computing the truncated SVD, namely the Lanczos bidiagonalization [105] was shown to be equivalent [30,91] to a well-known method in statistics, namely the *NIPALS* (nonlinear iterative partial least squares) method. To the best of our knowledge, the algorithmic improvements developed for the Lanczos-bidiagonalization have not been exploited within

NIPALS implementations (cf [30]). Nevertheless, NIPALS has become a crucial tool in the analysis of problems from economics applications [99,127,149].

The SVD is also a key ingredient in model order reduction[2] [19,50] classically used for reducing the dimensionality of physics-related models based on differential equations. Recently, model order reduction has found more and more applications in machine learning [170,274]. One of the most important applications that directly mirrors the use of the SVD in computational science and engineering is the creation of reduced representations. Here, applications include text mining [3], face recognition [308], medicine [305], and many more.

The SVD also comes in many disguises among the different disciplines of statistics, engineering, and applied mathematics. Given a data matrix A applying principal component analysis (PCA), which is equivalent to performing the SVD, has been a key tool for understanding the structure of the data. In PCA, the column means within A is zero and then the right singular vectors v_i are called the *principal component directions* of A and the left singular vectors are the *principal components of* A. More information and applications are given in [154,155,202,299].

In order to compute the SVD for data matrices that originate from massive datasets one often has to resort to techniques from high performance computing [8,211,266]. Additionally, it is possible to rely on randomized algorithms that we discuss in Section 4.

The SVD is also a crucial ingredient in many algorithms for high-dimensional data analysis, see Section 6 on tensor factorizations.

2.1.2 | The QR factorization

Given a set of vectors collected in the matrix A and considering the span of the columns of A it is clear that the vectors themselves can potentially be a terribly conditioned basis for further numerical computations. Hence, one wants to find a well-conditioned basis, ideally a basis with orthogonal vectors. This task is achieved by computing the QR factorization, that is,

$$A = QR$$

with $Q \in \mathrm{R}^{n,n}$ an orthogonal matrix and $R \in \mathrm{R}^{n,m}$ an upper triangular matrix of the form

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix},$$

where the matrix $\hat{R} \in \mathbb{R}^{m,m}$ is invertible if only if the matrix A has full column rank m. From the representation it is clear that one can also work with $A = \hat{Q}\hat{R}$ where $\hat{Q}$ only contains the first m columns of Q. The QR factorization is another ubiquitous factorization in applied mathematics. Its computation is typically rather expensive and for the details we refer to [109]. In particular, the reduction of A to triangular form via so-called householder reflectors is a de facto standard. The solution of the least squares problem (3) without regularization [109] via the QR factorization is well-known

$$w_* = \underset{w}{\mathrm{argmin}} \|QRw - y\|_2 = \underset{w}{\mathrm{argmin}} \|Rw - Q^\top y\|_2.$$

Truncated QR decompositions in disguise are also at the heart of the Lanczos [171] and Arnoldi [9] methods, which are the key algorithms of Krylov subspace methods. Many algorithms in numerical linear algebra rely on Krylov subspaces, that is

$$\mathcal{K}_\ell(M, r) := \mathrm{span}\{r, Mr, M^2r, M^3r, \ldots, M^{\ell-1}r\}$$

being the space of dimension ℓ, where M is the system matrix of the underlying problem, for example, $M = (A^\top A + \lambda I)$ for the ridge regression problem (4), and r is a vector associated with a right-hand side of (4). In order to obtain robust methods we rely on a well-conditioned basis of $\mathcal{K}_\ell(M, r)$. The Lanczos method computes an orthonormal basis of

[2] A prototypical example is the drastic reduction of the dimensionality of the system matrices defining a dynamical system.

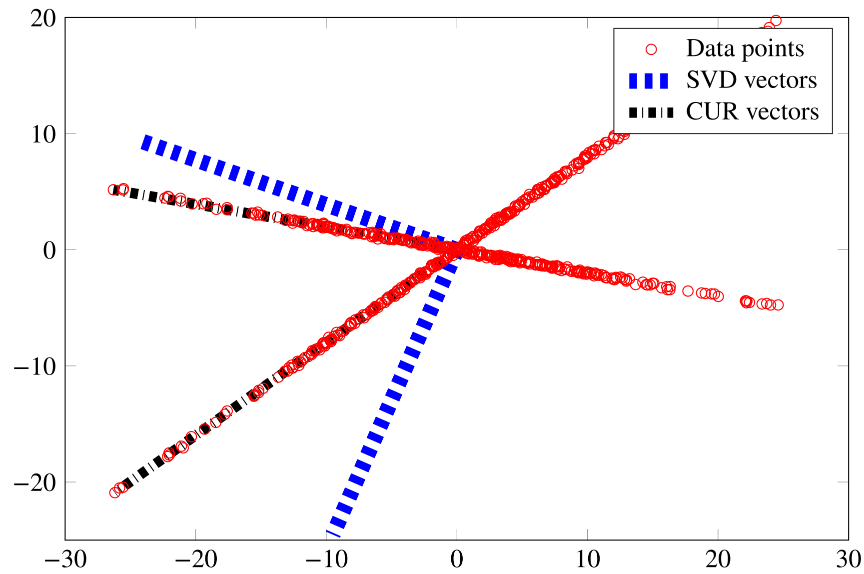

FIGURE 1 We here illustrate a set of data points (red) that are stored in the matrix A. We then show the two dominating rows (black) of A obtained from a CUR decomposition and the two dominating singular vectors (blue)

increasing dimensionality with great efficiency only requiring one matrix vector product per iteration. For more details, we refer to [110,236]. Truncated QR decompositions are based on computing the columns of a matrix Q spanning the basis of the range of the Krylov matrix.

In addition, the pivoted QR factorization [260,261] computes the factorization

$$AP = QR$$

with P a permutation matrix. This QR factorization can be computed maintaining the sparsity of the matrix A (cf [260]) and it is also closely related to interpretable factorizations, which we discuss next.

2.2 | Interpretable factorizations

Despite the beautiful mathematical properties of both the QR and the SVD they sometimes do not provide a representation that allows an easy interpretation of the results for practitioners. For example, in many applications the data are nonnegative while the singular vectors can contain negative values. Mathematically this means that a given data matrix $A \in \mathbb{R}^{n,m}$ is well approximated by the truncated SVD $A \approx U_k S_k V_k^\top$ but the singular vectors contained in $U_k(:,l)$, are in general not sparse or nonnegative even though this holds for the data. We adopt MATLAB notation addressing columns and rows of matrices by using $A(:,k)$ for the kth column of A and $A(k,:)$ for the kth row. Analogously, this can be defined for several rows or columns.

To preserve the properties inherent in the application while maintaining a good approximation of the original data with reduced complexity, many interpretable factorizations have been developed over recent years (cf [38,81,84,103,175,192,292] for some of them). Here, Figure 1 illustrates how the CUR approximation, which we introduce later in this section, naturally represents the original data.

We now briefly introduce these methods and comment on applications and new developments. We here follow the notation of [286] and start with the CX decomposition defined by

$$A \approx CX,$$

where $C \in \mathbb{R}^{n,k}$ and $X \in \mathbb{R}^{k,m}$. Here the matrix C is a matrix consisting of k columns of the data matrix A, that is,

$$C = A(:,J)$$

for an index set J of order k. The important feature of the matrix C is that its columns are interpretable as they are taken from the original data. Since the data matrix is often sparse this is inherited by C and as a result storing

C requires less memory than storing the singular vector matrix U_k. The computation of C and X is done via the minimization of

$$\min \|A - CX\|,$$

where $\|\cdot\|$ may be the 2-norm or the Frobenius norm. This problem is known as the column subset selection problem [37] and its NP-completeness is discussed in [251]. Typically, more structure is required than just a general matrix X and one quickly moves to the interpolative decomposition (ID) [286]

$$A \approx CV^{\top}, \tag{7}$$

where $C \in \mathbb{R}^{n,k}$ is as before but the matrix $V \in \mathbb{R}^{m,k}$ is constructed such that it contains a $k \times k$ identity matrix and $\max_{i,j} |v_{ij}| \leq 1$. A procedure to obtain a low-rank ID via a pivoted QR is given in [286] returning a $V^T = [I_k \; T_l]P^T$ with P a permutation matrix and T_l a solution related to the upper triangular factors of the pivoted QR. This approach works analogously when a one-sided representation in terms of matrix rows is desired. One can also obtain a two-sided ID

$$A \approx WA(I, J)V^T. \tag{8}$$

We start its computation by using a one-sided ID $A \approx CV^{\top}$, which already provides us with the set of crucial column indices J. In order to compute the row indices I and the matrix W, we now compute a one-sided decomposition for the matrix C^T. Following (7) we then obtain

$$C^T \approx \tilde{C}\tilde{V}^T,$$

where $\tilde{C}$ contains by design columns of C^T, that is, rows of C and thus elements of the rows of A along with the row index set I. The matrix W is then given as $W = \tilde{V}$. Note that W and V do not contain columns and rows of the original matrix A, respectively, and hence properties such as sparsity and nonnegativity found in the data are typically not carried over.

As a result, we can consider a factorization that avoids this pitfall and bears a lot of similarity to the two-sided ID. This is achieved by the CUR decomposition

$$A \approx CUR,$$

where $C \in \mathbb{R}^{n,k}$, $U \in \mathbb{R}^{k,k}$, and $R \in \mathbb{R}^{k,m}$. Here, C contains columns of the original matrix and R represents a subset of its columns. Both matrices inherit properties such as nonnegativity, sparsity, and finally interpretability. The CUR decomposition is also known as the skeleton decomposition [102,113,163,219,276] and is closely related to a rank-revealing QR factorization [118,286]. The point of departure for computing the CUR decomposition is typically a low-rank factorization of the matrix A. Both the truncated SVD (5) and the two-sided ID (8) are the typical initial factorizations. Given a truncated SVD, the crucial algorithmic step is to select the index sets I and J. For this one typically computes *leverage scores* as sums over the rows of the matrices U_k and V_k coming from the truncated SVD $A \approx U_k S_k V_k^{\top}$, for example, $\ell_j = \sum_{i=1}^{k} U_{j,i}$ for $j = 1, \ldots, n$ for the column selection and analogously for V_k. In [192] the authors provide a statistical interpretation of the leverage scores as a probability distribution. More recently, Embree and Sorensen [257] have introduced a procedure based on the discrete empirical interpolation method [50] where the selection of the column and row indices is based on a greedy projection technique resembling a pivoting strategy within the LU factorization. In [73] the authors compute the column and row subset using conditional expectations with a more efficient numerical realization being recently introduced in [64]. It remains to compute the intersection matrix U as $U = A(I, J)^{-1}$ with the other possibility being $U = C^{\dagger}AR^{\dagger}$, where $\dagger$ indicates the Moore-Penrose inverse. Recently, a perturbation analysis of the CUR decomposition was presented in [131].

Another important interpretable factorization of the matrix A is the so-called NMF [24,77,92,103]

$$A^T \approx WH, \tag{9}$$

which is a low-rank approximation using $W \in \mathrm{R}^{m,k}$ and $H \in \mathrm{R}^{k,n}$ with component-wise nonnegativity written as $W \geq 0$, $H \geq 0$. The interpretation of the columns of $W(:, j) \in \mathrm{R}^m$ is that they form a basis of order k that best approximates the data points a_j, that is, the columns of A^T via

$$a_l = WH(:,l) = \sum_{j=1}^{k} W_{:,j} H_{j,l}.$$

The coefficients of how the data are expanded in the basis defined by W are stored in H. Such linear dimension reduction frameworks are found in various data tasks within image processing or text mining. Here again the nonnegativity of the elements in W means that the resulting matrix is more sparse than an SVD-based approach and provides an interpretable feature representation [184,186]. The computation of a NMF is an NP-hard ill-posed problem [282] and it is typically based on solving the minimization problem

$$\min_{W,H\geq 0} \|A^T - WH\|_F.$$

Alternating minimization procedures, which consist of an alternating update of the factors W and H, are typically employed and we refer to [103,128,189] for more details. It is also possible to include further constraints such as sparsity as was done in [228,229]. In [75] the authors analyze the relationship between the NMF factorization of a *kernel matrix* and spectral clustering discussed later. For further improving the performance the authors in [47] include a kernel matrix as a regularization term for the objective function. This shows that kernel matrices, which are matrices changing the representation of the data, are often very useful and we discuss these next.

3 | CHANGING THE DATA REPRESENTATION

So far we focused on methods that directly utilize the data A as a matrix. Often it is necessary to transform the data to a different representation. The goal is that for the transformed data the learning task is easier and we now describe several approaches designed for that purpose.

3.1 | The graph Laplacian operator

The data encoded in $A \in \mathbb{R}^{n,m}$ either have a natural representation as a graph with the nodes v_j representing the associated feature vectors $a_j\ \forall j$ or they can be modeled that way. The result is a graph $G = (V, E)$ consisting of the nodes $v_j \in V$ and edges $e \in E$, where an edge e consists of a pair of nodes. We here consider undirected graphs where an edge is typically equipped with an edge weight representing the *strength* of the connection between the corresponding nodes. Practically, the most relevant weight function is the Gaussian weight function

$$w(v_i, v_j) = w_{ij} = \exp(-\|a_i - a_j\|_2^2/\sigma^2). \tag{10}$$

Many applications naturally have a graph structure but one can also convert data to graph form and we refer to [[285], Section 2.2] where different techniques are presented. Given a graph the weights are collected into a matrix $W \in \mathbb{R}^{n,n}$ where the diagonal is set to zero. If two nodes are not connected in the graph the associated matrix entry is set to zero. A particularly interesting and challenging example is a fully connected graph where all data points are compared pairwise also resulting in a dense matrix W. As a second ingredient we compute the diagonal degree matrix D where $d_{ii} = \sum_{j=1}^{n} w(v_i, v_j)$. We then obtain the *graph Laplacian* $L = D - W$, which is often used in a normalized form either as the symmetric normalized Laplacian $L_{sym} = I - D^{-1/2}WD^{-1/2}$ or as the random walk Laplacian $L_{rw} = D^{-1}L$. The properties of the graph Laplacian are discussed in [55,285]. In more detail, we can see that L is symmetric and its positive semi-definiteness follows from

$$u^\top Lu = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2. \tag{11}$$

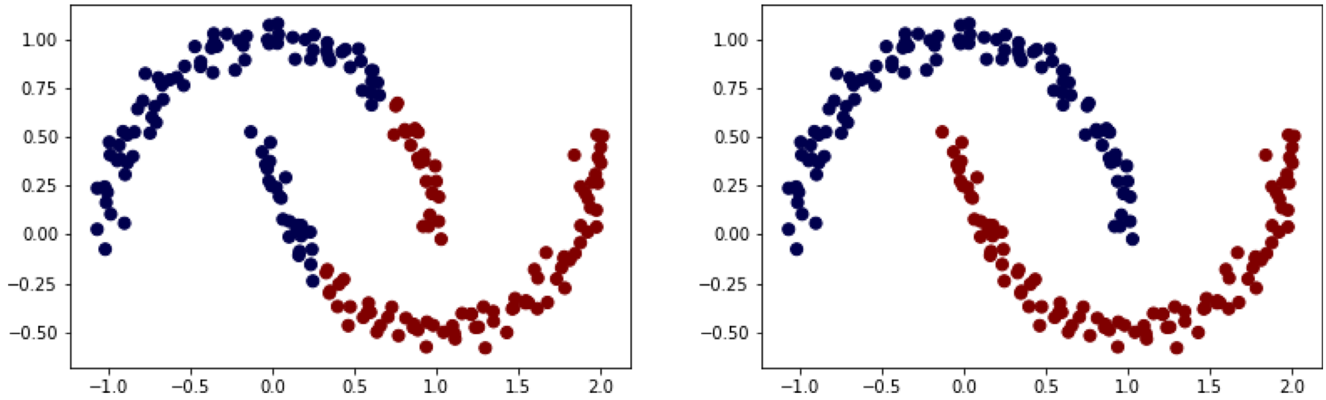

FIGURE 2 The typical *two-moons* data set with desired unsupervised classification into two-classes. A standard *scikit-learn* (https://scikit-learn.org) k-means clustering applied on the left vs *scikit-learn* spectral clustering (right). The curvature of the data is only correctly identified with the spectral clustering approach

This relation only changes slightly when L_{sym} is used. In the context of data science many of the properties of the graph Laplacian can be utilized for tasks such as clustering. In particular, the eigeninformation of L and L_{sym} is crucial. For example, the number of zero eigenvalues gives information about the number of connected components of the underlying graph. The eigenvector corresponding to the first nonzero eigenvalue is known as the Fiedler vector. As the eigenvector corresponding to the zero eigenvalue is the constant vector $c[1, \dots, 1]^T$ with $c \in \mathbb{R}$ and since the Fiedler vector is orthogonal to it, we must have sign changes in the Fiedler vector. This makes the Fiedler a first candidate to perform clustering simply by the sign of its entries[3] [27,49,76,278].

In fact, one of the classical tasks that is performed using the eigeninformation of the graph Laplacian is spectral clustering [285], where one computes the first[4] k eigenvectors $\phi_1, \dots, \phi_k$. As the graph Laplacian translates the original data A into an alternative space encoded into new matrices W and D, its eigeninformation will lead to a different clustering behavior than traditional methods such as k-means [259]. It can be seen from Figure 2 that standard k-means clustering based on A [133,157] shows poorer performance when compared against spectral clustering [285]. In more detail, the most common spectral clustering methods proceed by using k-means on the rows of the eigenvector matrix

$$\Phi_k = [\phi_1, \dots, \phi_k]. \tag{12}$$

In [285] the author illustrates that performing spectral clustering solves relaxed versions of known graph cut problems, which aim at partitioning the vertices of a graph into disjoint subsets. If instead of the graph Laplacian the two normalized versions are used, one obtains different results, for L_{rw} see [250] and for L_{sym} [215]. The hyperparameter[5] σ in the denominator of the weight function can be replaced by a local scaling with an additional hyperparameter describing the locality [306]. Often the parameter is chosen in a heuristic way according to the performance of the algorithm using the graph Laplacian [215]. When interpreted in terms of Gaussian processes the parameter σ is related to the characteristic length-scale of the process [230].

As the foundation of the spectral clustering method is the computation of k eigenvectors of the (normalized) graph Laplacian, it is important to be able to compute these eigenvectors efficiently. Be reminded that the matrices L, L_{rw}, and $L_{sym} = I - D^{-1/2}WD^{-1/2}$ are singular and the multiplicity of the zero-eigenvalue corresponds to the number of connected components in the graph. We are interested in computing the smallest eigenvalues of L. Iterative eigenvalue algorithms typically converge towards the largest eigenvalues and as a result we would need to invert the matrix L, which is not possible. Focusing on L_{sym} it is obvious that the smallest eigenvalues of L_{sym} can be computed from the largest eigenvalues of $D^{-1/2}WD^{-1/2}$. Methods for efficiently computing the eigeninformation of such a matrix often rely on Krylov subspaces [110,176,237,262], where it is crucial to perform the matrix vector products with $D^{-1/2}WD^{-1/2}$ efficiently. If we assume

[3] In https://people.eecs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html a connection to vibrating strings and standing waves is made that beautifully illustrates this property.
[4] The eigenvalues of the Laplacian are given as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
[5] As hyperparameter we understand a parameter whose value is set before the learning process starts.

that the graph consists of one connected component D is diagonal and invertible. The main cost of multiplying with $D^{-1/2}WD^{-1/2}$ comes from computing matrix vector products with W. For sparse graphs the matrix W will itself be sparse and matrix vector products will be inexpensive. The main computational challenge is then encountered for nonsparse or fully connected graphs. For large data sets matrix vector products with W are often infeasible and hence more sophisticated techniques are needed. Recently, methods based on the nonequispaced Fourier transform [5], the (improved) fast Gauss transform [208,298], or algebraic fast multipole methods [194,302] have shown great potential resulting in a complexity of $\mathcal{O}(n \log n)$ for the matrix vector products with a fixed number of columns m. While these methods provide great speed-ups the dimensionality m of the feature vectors still provides a significant challenge in computations and we return to this point in the next section.

The basis Φ given in (12) is not only important for spectral clustering but also for a group of methods recently introduced for semi-supervised learning, that is, methods that, given a small set of labeled data, classify the remaining unlabeled points simultaneously. The methods are based on partial differential equation techniques from material science modeling [7,46], namely diffuse-interface methods, and have also been used in image inpainting [25]. The graph Laplacian then replaces the classical Laplacian operator resulting in a differential equation based on the graph data via

$$u_t = \varepsilon L_{sym} u - \frac{1}{\varepsilon}\psi'(u) + \omega(f-u) \tag{13}$$

with $\psi(u)$ being a potential defined on the graph nodes enforcing two classes and ω incorporating penalization for deviation from the training data stored in f. The variable ε is a hyperparameter related to the thickness of the interface region. Due to the large number of vertices the dimensionality of (13) is vast. A projection using Φ_k reduces the PDE to a k-dimensional equation. Similar to model order reduction the nonlinearity still needs to be evaluated in the large-dimensional space [50].

We briefly want to comment on the use of the graph Laplacian in image processing, where the pixels are often represented as the nodes in a graph, which would then lead to a fully connected graph. In this field the graph Laplacian is often used as a regularizer for denoising [158,188,231] or image restoration [159]. In [204] the construction of the Laplacian is performed patch-wise in both a local and a nonlocal fashion. A more in-depth discussion for applications in image processing can also be found in [53].

The graph Laplacian has also recently enjoyed wide applicability within deep learning, namely, as an essential ingredient within so-called *graph convolutional networks* [42,138,161] where the equation at layer l for semi-supervised learning becomes

$$X^{(l+1)} = \sigma_l\left(\sum_{k=1}^{N_K} K^{(k)} X^{(l)} \Theta^{(l,k)}\right),$$

with σ_l an activation function and weights Θ that need to be learned. The crucial filter matrices $K^{(k)}$ are computed using the eigeninformation of the graph Laplacian associated with the input $X^{(0)}$. The matrices $K^{(k)}$ are composed from a k-dimensional filter space $\text{span}\{\phi_1, \ldots, \phi_k\}$, typically of the form $K^{(k)} = U\phi_k(\Lambda)U^T$, where U and Λ are the eigenvector and eigenvalue matrix of the graph Laplacian or a slight modification of it. The name convolutional network stems from the fact that the transformation $U\phi_k(\Lambda)U^T$ can be interpreted as graph Fourier transform [70,252]. Similarly, classical convolutional networks apply a filter/convolution to the data to detect more structure within the data [173]. In more detail, for a signal $x \in \mathrm{R}^n$, with n the number of nodes in the graph, $\hat{x} = U^T x$ is the graph Fourier transform and its inverse is given by $x = U\hat{x}$. It is clear that polynomial filters ϕ_j are easy to apply either directly, by multiplying with the matrices, or via an (approximate) eigendecomposition of the Laplacian. Many filters and efficient methods for their computations have been suggested and we refer to [6,138,161,180,183,253,293,312] for some of them.

There have also been generalizations of the graph Laplacian to other settings. We in particular want to mention the case of *hypergraphs* [310,311], where an edge is now a collection of possibly many nodes. Again, one can obtain a normalized Laplacian operator of the form $L = I - \Xi$ and perform spectral clustering based on the eigeninformation of this matrix. Hypergraphs are encountered in many applications such as biological networks [165], image processing [304], social networks [309], or music recommendation [43]. Hypergraphs have also been used in the context of semi-supervised learning [34,177,227] and particular in convolutional neural networks based on hypergraphs [6,297].

Graph Laplacians have also been used to analyze multilayer networks [31,164] where a set of nodes can be connected in various layers (cf Figure 3 for an illustration). The connections between the nodes and the various layers can be

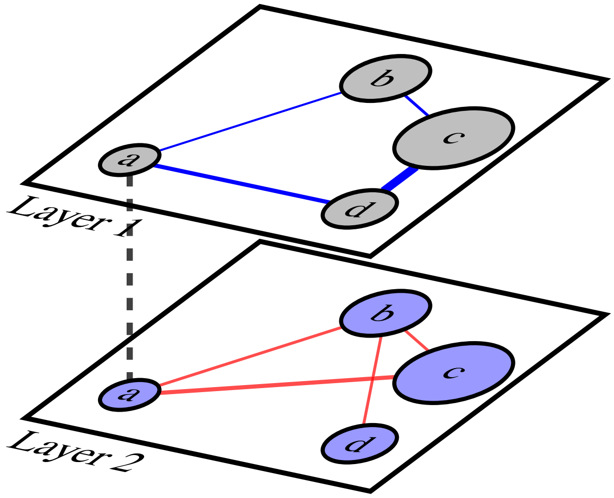

FIGURE 3 A simple multilayer graph

represented as a tensor but also using a (supra)-Laplacian [256]

$$L = L^{(L)} + L^{(I)}$$

with the intra-layer-supra Laplacian $L^{(L)} = \mathrm{blkdiag}(L_1, \ldots, L_K)$ and the interlayer-supra Laplacian $L^{(I)} = L_I \otimes I$ with L_I the interlayer Laplacian. Again, the eigeninformation of the supra-Laplacian provides rich information about the network [225,256,270]. One can also obtain networks where the nodes are not connected across layers but rather only have intra-layer connections (cf [200]). We do not discuss the full details of the various different network structures here but identify this as a very exciting area of future research.

In the context of analyzing social relationships we want to mention *signed networks*, which are graphs with positive and negative edge weights. These networks are used to model friend and foe type relationships and we refer to [126,178,179,247,267] and the references mentioned therein for an overview of some of the crucial applications. Again techniques such as spectral clustering [199,201,247], semi-supervised learning [197], convolutional networks [72] are available to extract further information from the data. The difficulty for signed networks is that the classical graph Laplacian is not feasible as for example the sum of the weights could be zero resulting in a noninvertible degree matrix. As a result several competing Laplacians are possible (see [101,267] for an overview).

The graph Laplacian is also an essential tool analyzing complex networks via network motifs [20,220], graph centralities [96,98,224], or community detection [213,214]. It has also been suggested to replace the graph Laplacian by a *deformed Laplacian* or *Bethe Laplacian* [41,209,238] as

$$H(s) = (s^2 - 1)I - sW + D$$

for a parameter $s \in \mathbb{R}$ and W the adjacency matrix of the graph. It has been shown that $H(s)$ corresponds to a non-backtracking random walk, which is a simple random walk that is conditioned not to jump back along the edge it has just traversed [169] and shows better performance when community detection is desired in very sparse graphs generated by the stochastic block model.

In the next section, we turn our attention to the case when the kernel, that is (10) is not only part of the graph Laplacian, but is viewed as the defining element of embedding the data into a high-dimensional space.

3.2 | Kernel methods

The transformation of the data via a so-called *kernel function*, like the Gaussian encountered for the graph Laplacian, is a technique that has been successfully applied in many data science tasks [147,210,246,249]. In Figure 4 we illustrate a dataset that is difficult to linearly separate in two dimensions on the left. When the data are transformed via kernelization

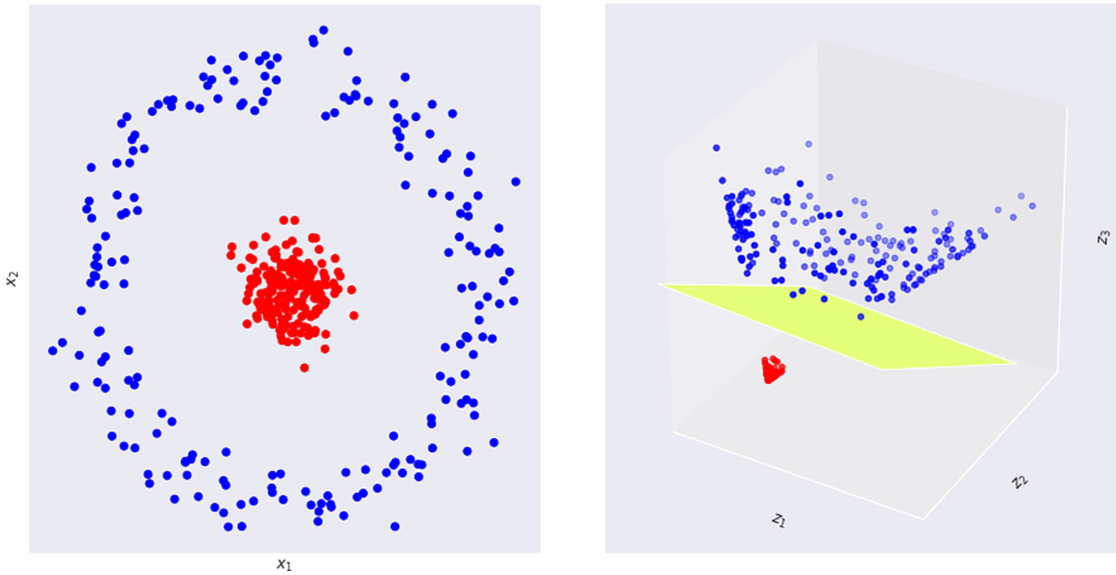

FIGURE 4 A typical two-dimensional dataset (left) difficult to separate linearly vs the embedding of the data into three-dimensional space (right) with the decision boundary shown in grey

to three dimensional space it is easily separable. Kernel methods can be motivated from the evaluation of the function $f(x) = w^T x$ for a new data point x by using the computed weights w as minimizers of a least squares problem (3). By employing Lagrangian duality [246] we can write the weights as $w = \frac{1}{\lambda}\sum_i \alpha_i a_i \in \mathbb{R}^m$ where λ is a regularization parameter, $a_i \in \mathrm{R}^m$ are the vectors associated with the data and the α_i are the Lagrange multipliers. Inserting the new point x gives $f(x) = \frac{1}{\lambda}\sum_i \alpha_i a_i^T x$, which relies on the evaluation of inner products $a_i^T x$. It turns out that the evaluation of this and other inner products is replaced by a more general *kernel function* $k(\cdot, \cdot)$. We then write the above as

$$f(x) = \frac{1}{\lambda}\sum_i \alpha_i k(a_i, x)$$

using the kernel instead of the inner product. The goal within kernel methods is now to find a kernel that allows for better separability then the trivial kernel $k(a_i, x) = a_i^T x$. To understand the role of the kernel function let us look at the kernel [246]

$$k(v, x) = (x^T v)^2$$

and consider the feature mapping ϕ that maps the two-dimensional data into three-dimensional space via

$$\phi(x) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{bmatrix}$$

and we can see that

$$k(x, v) = \phi(x)^T \phi(v).$$

This even comes at a computational advantage as we do not need to compute the higher-dimensional ϕ vectors. This technique of avoiding the direct computation of the higher-dimensional nonlinear relation via the evaluation of a kernel function is known as the *kernel trick* [242,243,246]. It is clear now that the adjacency matrix of the graph Laplacian is also a kernel matrix for the particular choice of the Gaussian kernel. The assembly of the kernel for all data points a_i into a matrix leads to a positive semi-definite Gram/kernel matrix K. In fact, under the name of *kernel PCA* the leading (now largest) eigenvectors of K are computed to obtain principal components/directions in the data (cf [244,245]). The use of the kernel trick in machine learning is omnipresent. One of the simplest but powerful methods is the so-called kernel ridge regression (KRR), where the core problem is to minimize the function

$$\frac{1}{2}\|b - Au\| + \frac{\lambda}{2}\|u\|$$

with b a vector encoding the training data and u a vector of weights. This problem is then solved using a dual formulation resulting in a formulation based on the matrix AA^T, which consists of inner products of the feature vectors. A kernelization of AA^T then leads to the kernel matrix K for which we need to solve the linear system [5]

$$(K + \lambda I)w = b.$$

The system matrix is symmetric and positive definite for a positive regularization or ridge parameter $\lambda \in \mathbb{R}$. Such a system is typically solved numerically with the use of a preconditioned iterative solver such as the conjugate gradient method [110,141,236]. The key ingredients are the matrix vector products with $K + \lambda I$ and developing a preconditioner $P \approx K + \lambda I$. For the matrix vector product the nonequispaced fast Fourier transform can be used for a variety of different kernels [5] and in the case of Gaussian kernels many bespoke methods exist such as [194,298,302]. A method based on sketching and the random feature method [226] is introduced in [12], which can also be applied to various different kernel functions. A flexible preconditioner, which changes in every iteration, combined with a suitable Krylov solver was presented in [258] whereas the authors in [248] construct a low-rank approximation preconditioner based on an ID and fast matrix vector products. A more difficult scenario arises when m, the dimensionality of the feature vectors $a_i\mathrm{R}^m$ gets larger. In this case, fast matrix vector multiplication becomes more difficult and suffers from the curse of dimensionality unless certain decay rates are imposed on the matrix entries [208]. In [300] the authors consider an example with $m = 90$ where they use a divide-and-conquer parallel algorithm that also comes with communication avoidance. For more details we refer to [300] and also the mentioned literature for competing methods. In [233] the authors use a technique based on randomization (cf Section 4) for both the matrix vector product and the preconditioner in KRR while a similar problem is solved in [303] via high performance computing approaches.

The power of the kernelization has been exploited as an essential ingredient of support vector machines (SVMs) [63,246,280]. In more detail, SVMs are derived from maximizing the margin of the separating hyperplane. The support vectors are the closest data points to this hyperplane. In order to be able to obtain a nonlinear hyperplane kernelization of the inner products is employed. The resulting kernel matrix is then at the heart of the quadratic program that needs to be solved for determining the support vectors [246] but the computation suffers from having to deal with dense kernel matrices. In [221] the sequential minimal optimization method is introduced, which breaks the problem into smaller, analytically solvable problems. Kernel matrices are also crucial when solving the closely related support vector regression (SVR) problem [88,255]. While both SVM and SVR remain popular methods, deep learning techniques have recently gained more popularity and have been combined with kernel techniques via graph convolution networks [138,161] or as loss functions in convolutional networks [191,268].

Often the linear algebra tasks associated with large scale kernel methods will involve randomization, which we discuss next.

4 | RANDOMIZATION

In their seminal review paper [130] the authors discuss the importance of the decompositional approach to matrix computations [79] (cf Section 2). Due to the computational complexity it is not always straightforward to efficiently compute matrix factorizations, such as the SVD or QR, since in many data science applications the matrix A is so large that computing approximate factorizations is too costly. Additionally, the data might be corrupted or it might be desirable to avoid many passes over the data so that classical methods need further thought.

One of the key ingredients in allowing efficient numerical linear algebra is the process of *randomization.* Randomization has proven to be a valuable tool in various matrix computation tasks such as matrix vector products [62,82] or low-rank approximations [29,83]. These techniques typically follow the scheme of producing a skinny matrix $Q \in \mathrm{R}^{n,k+p}$, with k the desired approximation rank and p an oversampling parameter that is needed for theoretical guarantees, such that we obtain the orthogonal-projection-approximation onto the subspace spanned by Q via

$$A \approx QQ^T A.$$

From the information contained in Q one then computes standard factorizations such as QR or SVD decompositions. In more detail, we form the matrix $B = Q^T A \in \mathbb{R}^{k+p,m}$, which then gives the low-rank approximation $A \approx QB$. Replacing B by its SVD results in an approximate SVD of A via

$$A \approx QB = Q(U\Sigma V^T) = (QU)\Sigma V^T.$$

The first stage, that is, the computation of the matrix Q, follows from the prototype algorithm illustrated in Algorithm 1.

Algorithm 1. Prototype algorithm from [130]

Draw a random $n \times (k+p)$ test matrix G[6].
Compute $Y = AG$.
Construct orthogonal basis for range(Y), for example, $[Q, R] = qr(Y)$.

We are typically only interested in a decomposition of order k when proceeding to the second stage of the method but that G is of dimension $k+p$ where p is the oversampling parameter. The dimensionality of G is crucial as our aim is to reduce the complexity as much as possible. Theoretical bounds for the approximation quality are given in [130] and depend on the parameters k and p as well as the matrix size and the $(k+1)$st singular value of A. Such a randomized SVD has become an essential tool in many scientific computing and data science applications in areas such as uncertainty quantification [181], optimal experimental design [4], or computer vision [93].

One of the key applications of randomized methods within data science is the use for approximation of kernel matrices [26,85,148,182,195,307] where the desire is to avoid the computation of the full kernel matrix since the storage demand can be too large. In particular, the kernel matrix is approximated via

$$A \approx (AQ)(Q^T AQ)^{-1} Q^T A^T,$$

where in the traditional setup the matrix Q contains columns of the identity matrix drawing columns from the matrix A and we obtain the so-called *Nyström scheme*

$$\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} A_{11}^{-1} \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} = \begin{bmatrix} I \\ A_{21}A_{11}^{-1} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}A_{11}^{-1}A_{12} \end{bmatrix} \approx A.$$

This means in order to approximate a kernel matrix A one only needs to compute a small number of columns and rows, which has been shown to be very successful (cf [85,223]). The authors in [85] give an approximation bound using the diagonal entries of the kernel matrix and a probabilistic interpretation whereas the authors in [219] provide a bound for the Chebyshev-norm of the pseudo-skeleton approximation and the $k+1$st singular value of the kernel matrix. In [195] the author suggests to use a Q obtained via Algorithm 1 and thus obtain a variant of the traditional Nyström method.

We already pointed out in Section 2.2 that interpretable decompositions can be obtained via QR or SVD approximations to A. As the author in [195] points out computing a matrix that holds a basis for the column space of A is crucial. For this the author in [195] computes $Y = AG$ and via a subspace iteration proceeds to iteratively refine the matrix Y using the update $Y \leftarrow AA^T Y$. The resulting Y is then used to identify the relevant row-indices for the ID. In the same way one can obtain the two-sided ID and also a randomized CUR decomposition, where the details are given in [195]. Note that algorithms for directly computing the CUR decomposition compute some sort of statistical score for the importance of the columns/rows via the truncated SVD [37,192,288]. When obtaining the CUR from the two-sided ID we get the index

[6]Typically, this is drawn as a Gaussian random matrix.

sets from the ID matrices. The usefulness of the importance sample or sampling statistics is illustrated in [39] where the author shows that operations from numerical linear algebra such as inner products or matrix vector products can be performed using randomized numerical linear algebra. The key ingredients when multiplying AB or a^Tb is the sampling strategy for producing the approximations. The strategy is often based on an *importance sampling* strategy [84,86] to draw elements or rows/columns of a matrix for further processing. For example, the authors in [94] use the vector q

$$q_j = \frac{\|A(:,j)\|_2\|B(j,:)\|_2}{\sum_{i=1}^{n}\|A(:,i)\|_2\|B(i,:)\|_2}, \quad j = 1, \dots, n$$

of probabilities[7] to produce an approximation $C \approx AB$ based on rows of A and columns of B. For the solution of a least squares problem such as (3) [39,87] the (random) selection process is decoupled from a deterministic computation via a traditional numerical linear algebra method. In particular, the first stage computes a score based on for example, Euclidean norms, a truncated SVD matrix, or columns of a truncated Hadamard matrix. This means that the original problem is separated into a random sampling procedure leading to a reduced formulation that is then solved via a deterministic NLA approach. Due to the success in data science applications randomized methods have also penetrated classical problems in scientific computing such as solving linear systems of equations [114,212,275], eigenvalue problems [117,241], or inverse problems [281,294,295]. A recent survey can be found in [196].

One area where randomization has helped greatly with the evaluation of complex mathematical expressions is the computations of functions of matrices, which we want to describe now.

5 | FUNCTIONS OF MATRICES

The concept of evaluating a function of a matrix

$$f : R^{n,n} \to R^{n,n},$$

where f is not meant to be evaluated element-wise, is an old but still very relevant one and we refer to [143] for an excellent introduction to the topic. Matrix functions appear in a variety of applications with a particularly important example given by Gaussian processes [235]; a ubiquitous tool in machine learning and statistics or in the analysis of complex networks [22,97]. One can define the matrix function via the Cauchy integral theorem as

$$f(A) = \frac{1}{2\pi i}\int_{\Gamma} f(z)(zI - A)^{-1}dz \tag{14}$$

provided f is analytic on and inside a closed contour $\Gamma \subset \mathbb{C}$ that encloses the spectrum of A. There are other definitions that are equivalent for analytic functions (cf [143]). Matrix functions provide a large number of challenges for numerical methods and an early discussion with 19 *dubious methods* for the matrix exponential can be found in [206] with an update provided in [207] adding Krylov subspace methods as the 20th method. One typically considers two different setups, the first is the computation or approximation of the matrix $f(A)$ explicitly and the other is the evaluation of $f(A)b$ where the explicit computation of $f(A)$ and subsequent application to b is avoided. For a recent survey on evaluating $f(A)b$ we refer to [124]. In this case the computation of $f(A)$ and then applying it to b is obviously avoided. Efficient techniques typically depend on the size of the data matrix A. The approximation of $f(A)b$ employing the Cauchy integral formula (14) via contour integration was given in [129]. Often the main work goes into solving linear systems with $(zI - A)$ and once direct solvers become infeasible approaches based on Krylov subspace methods have shown very good performance for approximating the evaluation of $f(A)b$ [89,123,145,166,166]. For the more complex case of computing $f(A)$ we again refer to [143] for an overview of suitable methods. We here want to mention certain scenarios in data science applications that lead us to such matrix functions.

Throughout scientific computing and engineering fractional Laplacians, where instead of second order derivatives one considers derivatives of arbitrary orders, have gained much popularity in recent years [222] and one way to evaluate this efficiently is using matrix functions, for example, [45]. Fractional powers of the Laplacian have recently become

[7]This is obtained from the minimization of the expected value of the variance of the inner product/matrix vector product.

more popular for semi-supervised learning [17,69], where to the best of our knowledge techniques based on the full eigendecomposition have been employed, which for large graphs quickly becomes infeasible. In [198] a multilayer graph is considered where arbitrary matrix powers of the layer Laplacians are evaluated using the polynomial Krylov subspace method [143] and using contour integrals [200].

We return again to the study of complex networks and are now interested in the network centrality [97] as this helps in identifying the most important nodes, such as the most influential people in a social network. We assume that the network is understood as an undirected graph and describes some of the most important centrality measures. One measure of centrality is the degree matrix D of the graph Laplacian assigning the degree d_{ii} to every node. Other centrality measures are of great use for understanding the underlying data. For example, the eigenvector centrality [32] is defined as

$$b_i = \frac{1}{\lambda_1} W\phi_1$$

with W the adjacency matrix of the network and (λ_1, ϕ_1) the Perron-Frobenius eigenvalue and eigenvector, respectively. The authors in [97] illustrate that powers of the adjacency matrix W provide essential information about the graph. In particular, studying the (i,j)th entry of W^n allows to count the number of different walks of length n getting from node i to node j. For example the diagonal entries of W^2 give the degrees of the nodes in the network. Note that powers of the graph Laplacian are considered in [1] for the purpose of spectral clustering and within graph convolutional networks for semi-supervised learning in [187]. The use of W^2 suggests to consider higher powers of the adjacency matrix for longer walks and one obtains

$$\left(I + \frac{W^2}{2!} + \frac{W^3}{3!} + \frac{W^4}{4!} + \dots\right)_{ii} = \exp(W)_{ii} = e_i^T \exp(W) e_i$$

as the so-called subgraph centrality for node i (cf [95,98]) with e_i the corresponding unit vector. The quantity trace(exp(W)) is defined as the Estrada index of a network [66,71,104]. Due to the high complexity the computation of the matrix function exp(W) should be avoided, especially if complex networks are considered. It is well known that expressions of the form $u^T f(A)u$ can be well approximated by $e_1^T f(T_k)e_1$, where T_k is the tridiagonal matrix coming from the Lanczos process applied to A and u is assumed to have norm one. In particular, for our example we have $u = e_i$, $f(\cdot) = \exp(\cdot)$, and $A = W$. As the authors in [106,107] show there is a beautiful relation between $u^T f(A)u$ and Gauss-quadrature with first results going back to [111]. In particular, the authors in [21] use the error estimates that stem from studying the relation between the Lanczos process and Gauss quadrature. In [33] the authors also rely on the power of Gauss quadrature to compute Katz scores and commute times between nodes in a network. Using the Lanczos process or other Krylov subspace methods [107,108,264] to approximate quantities of the form $u^T f(A)u$ also proves essential for many applications (cf [10,11,150]) when the trace of a matrix (function) has to be estimated via

$$\text{trace}(f(A)) \approx \frac{1}{s}\sum_{i=1}^{s} v_i^T f(A) v_i,$$

where we rely on s carefully chosen random vectors v_i. One such estimator is the so-called Hutchinson estimator [13] where $v_i = \pm 1$ and the numerical computation again relies on the Lanczos process and its efficient approximation of the spectrum of A [18,203,277]. Gaussian processes [190,230,235] are another essential tool within statistics and data science. Their evaluation poses a significant numerical challenge as it relies on evaluating matrix functions as we will illustrate now. Following [78], a Gaussian process is a collection of random variables with a joint probability distribution. Let us consider $X = \{x_1, \dots, x_n\}$ with all the $x_i \in \mathbb{R}^d$. The Gaussian process can then define a distribution over functions $f(x) \sim \mathcal{GP}(\mu(x), k(x,x'))$ with mean $\mu(x) : \mathbb{R}^d \to \mathbb{R}$ and covariance function $k(x,x')$: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Now $f_X \in \mathbb{R}^n$ represents the vector of the function values for $f(x_i)$, $\mu_X \in \mathbb{R}^n$ the evaluation of $\mu(x_i)$, and the matrix $K_{XX} \in \mathbb{R}^{n,n}$ represents the evaluation of $k(x_i, x_j)$ $\forall i,j$. Then it holds that $f_X \sim \mathcal{N}(\mu_X, K_{XX})$. The chosen covariance kernel depends on hyperparameters θ and we assume that we are given a set of noisy function values $y \in \mathrm{R}^n$ with variance σ^2. Assuming a Gaussian prior distribution depending on θ one obtains the log marginal likelihood as

$$\mathcal{L}(\theta|y) = -\frac{1}{2}[(y-\mu_X)^T(K_{XX}+\sigma^2 I)^{-1}(y-\mu_X) + \log(\det(K_{XX}+\sigma^2 I)) + n\log(2\pi)].$$

It is obvious that for the numerical solution of the minimization of the log marginal likelihood the evaluation of the log-determinant is crucial. If the matrix K_{XX} is small then we can afford the computation via the Cholesky decomposition, which is an $\mathcal{O}(n^3)$ computation. An efficient computation is then given via

$$\log(\det(W)) = 2\sum_{i=1}^{n} L_{ii},$$

where $W = LL^T$ is the Cholesky decomposition of the symmetric positive definite matrix W. In [234,235] the authors show that the methods for approximating the log-determinant become more efficient if one works with Markovian fields (cf [254] for approaches beyond Cholesky for Markovian fields). In general the Cholesky decomposition will be too expensive and again Krylov subspace methods such as the Lanczos process and its connection to Gauss-quadrature are exploited. From the relation

$$\exp(\mathrm{trace}(\log(W))) = \exp(\mathrm{trace}(X\log(\Lambda)X^{-1}) = \exp\left(\sum_i \log(\lambda_i)\right) = \prod_i \lambda_i = \det(W)$$

we obtain the following beautiful relationship

$$\log(\det(W)) = \mathrm{trace}(\log(W)).$$

This shows that in order to approximate the log-determinant, it is again crucial to efficiently evaluate a matrix function, namely

$$\mathrm{trace}(\log(A)) \approx \frac{1}{s}\sum_{i=1}^{s} v_i^T \log(A) v_i.$$

The Lanczos-based approximation has been used in [15,185,277] and the authors in [78] combine these ideas with fast matrix vector products by modified structured kernel interpolation [291] allowing a good approximation of the log-determinant and also its derivatives.

We refer to [144] for an overview of software for the computation of matrix functions in various programming languages.

6 | HIGH-DIMENSIONAL PROBLEMS

All of the above problems are tailored to when the data are given as a matrix $A \in \mathrm{R}^{n,m}$ but often it is also natural for the data to be given as a tensor

$$A \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \tag{15}$$

with the dimensions $n_i \in \mathrm{N}$ for all d modes. It is obvious that the storage requirements for a full tensor A quickly surpass the available resources in many applications. This exponential increase in relation to the parameter d is often referred to as the *curse of dimensionality.* Tensors and their approximations are closely related to the representation of a nonlinear function in a high-dimensional space [74]

$$f(x_1, \ldots, x_d) \approx \sum_{j=1}^{l} \alpha_j \phi(x_1, \ldots, x_d), \tag{16}$$

where the choice of functions ϕ is crucial and these could be a basis, a frame, or a dictionary of functions [44,61,90,156,271]. In this case it is desirable to find an approximation to $f(x_1, \ldots, x_d)$ with some form of sparsity, that is, as small as possible number of terms l. Alternatively, one can interpret the function as being defined on a tensor product space and aiming at designing a low-rank approximation to f (cf eg, [216]). The approximation of f in (16) is then performed by relying on similar formats and tool as the ones that are used for the approximation of the given tensor A in (15), which we briefly describe now.

For this reason approximating the tensor has a long tradition in computational mathematics, physics, chemistry, and other disciplines. While being already 10 years old we would like to point to [167] for a seminal overview of numerical tensor methods due to the introductory character and the broad overview. There are several low-rank approximations to the full tensor A that correspond to different tensor formats such as the classical CANDECOMP/PARAFAC(CP) formulation [160] where

$$A = \sum_{j=1}^{R} \lambda_j u_j^{(1)} \circ \cdots \circ u_j^{(d)}, \tag{17}$$

with $\circ$ being the dyadic product. While this format is the most natural, it has properties that make it not ideal for (many) applications, for example, the set of rank R tensors is not closed, the computation ill-posed, and so on (cf [115] for more details). Alternatively, one can consider the Tucker format/HOSVD [67,68], which somewhat generalizes the SVD to the multidimensional case but still suffers from the curse of dimensionality due to the d-dimensional core tensor needed. This is overcome by the tensor-train format [218], which consists of d core tensors of maximal dimension 3. In the context of data science tensors have long been an essential tool and to list all applications and techniques goes beyond the scope of this paper. We here only list a few results that address similar questions to the above mentioned matrix cases: nonnegative factorizations [54,59,60], ID [240], CUR [64,193], kernel methods [136,137,269], semi-supervised learning [119], and randomization [16,289].

This list is far from complete and for more detailed reviews on tensor methods for data science we refer to the recent surveys [56-58] and the references given therein. For a very broad overview of recent techniques and literature for numerical tensor methods beyond data science we refer to [115].

One of the key areas in which tensor prove very powerful tools is the field of deep learning, which we now briefly review.

7 | NUMERICAL LINEAR ALGEBRA IN DEEP LEARNING

The study of computational and mathematical aspects of deep learning is glowing white hot with activity and we only want to point to number of places where linear algebra is used for better understanding or improving performance of deep learning methods.

In the most generic setup, the task in an artificial neural network is to determine the weight matrices $W^{(j)}$ and bias vectors $b^{(j)}$ from minimizing a loss function to obtain the function

$$F(x) := W^{(L)}(\ldots\, \sigma(W^{(2)}(\sigma(W^{(1)}x + b^{(1)})) + b^{(2)}))\, \ldots + b^{(L)} \tag{18}$$

see [112,142,174] for more information and further references. The loss function typically consists of a sum of many terms due to the large number of training data and as a result the optimization is based on gradient descent schemes [36]. For classical optimization problems it is often desirable to use second order methods due to their superior convergence properties. These typically require the use of direct or iterative solvers to compute the step towards optimality, for example, solving with the Hessian matrix in a Newton method. As a result the applicability of Newton-type schemes for the computation of $W^{(j)}$ and $b^{(j)}$ has received more attention with a strong focus on exploiting the structure of the Hessian matrix [35,51,65,239,283,287].

The computational complexity of neural networks is challenging on many levels. In order to reduce the numerical effort the authors in [272,296] use the SVD to analyze the weight matrices and then restructure the neural network accordingly, while the authors in [301] use the condition numbers of the weight matrices for such a restructuring. The weight matrices have received further attention as in [217] the authors compress the weight matrices of fully connected layers using a low-rank tensor approximation, namely, the tensor train format [218]. Additionally, low-rank approximations are also useful for speeding up the evaluation of convolutional neural networks [151] by using a low-rank representation of the filters, which are used for detecting image features. For a very similar task the authors in [121,122,172] rely on optimized tensor decompositions. Note that recently these techniques have also been applied to adversarial networks [48].

The connection of neural network architectures to optimal control problems as introduced in [120,125] makes heavy use of PDE-constrained optimization techniques including efficient matrix vector products. This topic has also recently received more attention from within the machine learning community [52] and promises to be a very interesting field for combining traditional methods from numerical analysis with deep learning.

As already pointed out in Section 3.1 graph convolutional networks (GCNs) have shown great potential and received much popularity recently as tools for semi-supervised learning [161]. The expressive power of representing data as graphs that is the basis of GCNs has led to using their methodology for adversarial networks [100], matrix completion [23], or within graph auto encoders [162].

This is only a brief glimpse where numerical linear algebra techniques have enhanced deep learning methodology. We believe that this will be an area of intense research in the future.

8 | CONCLUSION

We have illustrated with this literature review that numerical linear algebra is alive and kicking. In a sense the rise of machine learning, big data, and data science shows that *old methods* are still very much in fashion but also that new techniques are required to either meet the demands of practitioners or account for the complexity and size of the data.

ACKNOWLEDGEMENTS

The author would like to thank Dominik Alfke, Peter Benner, Pedro Mercado, and Daniel Potts for helpful comments on an earlier version of this manuscript. He is also greatly indebted to the two anonymous referees who greatly helped to improve the presentation. Open access funding enabled and organized by Projekt DEAL.

REFERENCES

[1] E. Abbe, E. Boix, P. Ralli, and C. Sandon, *Graph powering and spectral robustness*, (2018), arXiv preprint arXiv:1809.04818.

[2] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, Time-series clustering – a decade review, Inf. Syst. **53** (2015), 16–38.

[3] R. Albright, *Taming text with the SVD*, SAS Institute Inc, Cary, NC, 2004.

[4] A. Alexanderian et al., A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized l_0 sparsification, SIAM J. Sci. Comput. **36** (2014), A2122–A2148.

[5] D. Alfke et al., NFFT meets Krylov methods: Fast matrix-vector products for the graph Laplacian of fully connected networks, Front. Appl. Math. Stat. **4** (2018), 61.

[6] D. Alfke and M. Stoll, Semi-supervised classification on non-sparse graphs using low-rank graph convolutional networks, (2019), arXiv preprint arXiv:1905.10224.

[7] S. M. Allen and J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall. **27** (1979), 1085–1095.

[8] E. Angerson, et al, LAPACK: A portable linear algebra library for high-performance computers, Proceedings SUPERCOMPUTING '90, IEEE Computer Society Press, IEEE, 1990, pp. 2–11.

[9] W. E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Quart. Appl. Math. **9** (1951), 17–29.

[10] M. J. Atallah, F. Chyzak, and P. Dumas, A randomized algorithm for approximate string matching, Algorithmica **29** (2001), 468–486.

[11] H. Avron, Counting triangles in large graphs using randomized matrix trace estimation, Workshop on Large-scale Data Mining: Theory and Applications, vol. 10, 2010, pp. 10–9.

[12] H. Avron, K. L. Clarkson, and D. P. Woodruff, Faster kernel ridge regression using sketching and preconditioning, SIAM J. Matrix Anal. Appl. **38** (2017), 1116–1138.

[13] H. Avron and S. Toledo, Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix, J. ACM **58** (2011), 1–34.

[14] J. Baglama and L. Reichel, Augmented implicitly restarted Lanczos bidiagonalization methods, SIAM J. Sci. Comput. **27** (2005), 19–42.

[15] Z. Bai et al., *Computing partial eigenvalue sum in electronic structure calculations, technical report SCCM-98-03*, Stanford University, Stanford, CA, 1998.

[16] C. Battaglino, G. Ballard, and T. G. Kolda, A practical randomized CP tensor decomposition, SIAM J. Matrix Anal. Appl. **39** (2018), 876–901.

[17] E. Bautista, P. Abry, and P. Gonçalves, L^γ PageRank for semi-supervised learning, (2019), arXiv preprint arXiv:1903.06007.

[18] M. Bellalij et al., Bounding matrix functionals via partial global block Lanczos decomposition, Appl. Numer. Math. **94** (2015), 127–139.
[19] P. Benner, S. Gugercin, and K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. **57** (2015), 483–531.
[20] A. R. Benson, D. F. Gleich, and J. Leskovec, Higher-order organization of complex networks, Science **353** (2016), 163–166.
[21] M. Benzi and P. Boito, Quadrature rule-based bounds for functions of adjacency matrices, Linear Algebra Appl. **433** (2010), 637–652.
[22] M. Benzi and P. Boito, Matrix functions in network analysis, GAMM Mitteilungen (2020).
[23] R. V. D. Berg, T. N. Kipf, and M. Welling, Graph convolutional matrix completion, (2017), arXiv preprint arXiv:1706.02263.
[24] M. W. Berry et al., Algorithms and applications for approximate nonnegative matrix factorization, Comput. Stat. Data Anal. **52** (2007), 155–173.
[25] A. L. Bertozzi, S. Esedoglu, and A. Gillette, Inpainting of binary images using the Cahn–Hilliard equation, IEEE Trans. Image Process. **16** (2007), 285–291.
[26] A. L. Bertozzi and A. Flenner, Diffuse interface models on graphs for classification of high dimensional data, Multiscale Model. Simul. **10** (2012), 1090–1118.
[27] A. Bertrand and M. Moonen, Seeing the bigger picture: How nodes can learn their place within a complex ad hoc network topology, IEEE Signal Process. Mag. **30** (2013), 71–82.
[28] F. Berzal and N. Matín, Data mining, SIGMOD Rec. **31** (2002), 66.
[29] E. Bingham, H. Mannila, Random projection in dimensionality reduction, Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '01, ACM, ACM Press, 2001, pp. 245–250.
[30] A. Björck, Stability of two direct methods for bidiagonalization and partial least squares, SIAM J. Matrix Anal. Appl. **35** (2014), 279–291.
[31] S. Boccaletti et al., The structure and dynamics of multilayer networks, Phys. Rep. **544** (2014), 1–122.
[32] P. Bonacich, Power and centrality: A family of measures, Am. J. Sociol. **92** (1987), 1170–1182.
[33] F. Bonchi et al., Fast matrix computations for pairwise and columnwise commute times and Katz scores, Internet Math. **8** (2012), 73–112.
[34] J. Bosch, S. Klamt, and M. Stoll, Generalizing diffuse interface methods on graphs: Nonsmooth potentials and hypergraphs, SIAM J. Appl. Math. **78** (2018), 1350–1377.
[35] A. Botev, H. Ritter, D. Barber, Practical Gauss–Newton optimisation for deep learning, Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR, 2017, pp. 557–565.
[36] L. Bottou, Large-scale machine learning with stochastic gradient descent, Proceedings of COMPSTAT'2010, Physica-Verlag HD, 2010, pp. 177–186.
[37] C. Boutsidis, M. W. Mahoney, and P. Drineas, An improved approximation algorithm for the column subset selection problem, Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Society for Industrial and Applied Mathematics, 2009, pp. 968–977.
[38] C. Boutsidis and D. P. Woodruff, Optimal CUR matrix decompositions, SIAM J. Comput. **46** (2017), 543–589.
[39] M. W. M. Boyd, Randomized algorithms for matrices and data, Found. Trends Mach. Learn. **3** (2010), 123–224.
[40] R. G. Brown, *Smoothing, forecasting and prediction of discrete time series*, Courier Corporation, Chelmsford, MA, 2004.
[41] J. Bruna and X. Li, Community detection with graph neural networks, Stat **1050** (2017), 27.
[42] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, Spectral networks and locally connected networks on graphs, (2013), arXiv preprint arXiv:1312.6203.
[43] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He, Music recommendation by unified hypergraph: combining social media information and music content, Proceedings of the 18th ACM International Conference on Multimedia, ACM, 2010, pp. 391–400.
[44] H.-J. Bungartz and M. Griebel, Sparse grids, Acta Numer. **13** (2004), 147–269.
[45] K. Burrage, N. Hale, and D. Kay, An efficient implicit FEM scheme for fractional-in-space reaction-diffusion equations, SIAM J. Sci. Comput. **34** (2012), A2145–A2172.
[46] J. W. Cahn and J. E. Hilliard, Free Energy of a Nonuniform System. I. Interfacial Free Energy, J. Chem. Phys. **28** (1958), 258–267.
[47] D. Cai et al., Graph regularized nonnegative matrix factorization for data representation, IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011), 1548–1560.
[48] X. Cao, X. Zhao, and Q. Zhao, Tensorizing generative adversarial nets, Proceedings of the 2018 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Asia, IEEE2018, pp. 206–212.
[49] T. F. Chan, P. Ciarlet, and W. K. Szeto, On the optimality of the median cut spectral bisection graph partitioning method, SIAM J. Sci. Comput. **18** (1997), 943–948.
[50] S. Chaturantabut and D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. **32** (2010), 2737–2764.
[51] C. Chen, S. Reiz, C. Yu, H.-J. Bungartz, and G. Biros, Fast evaluation and approximation of the Gauss-Newton Hessian matrix for the multilayer perceptron, (2019), arXiv preprint arXiv:1910.12184.
[52] T. Q. Chen et al., Neural ordinary differential equations, Adv. Neural Inf Process Syst. (2018), 6571–6583.
[53] G. Cheung et al., Graph spectral image processing, Proc. IEEE **106** (2018), 907–930.
[54] E. C. Chi and T. G. Kolda, On tensors, sparsity, and nonnegative factorizations, SIAM J. Matrix Anal. Appl. **33** (2012), 1272–1299.
[55] F. Chung, *Spectral graph theory*, Vol **92**, American Mathematical Society, Providence, Rhode Island, 1996.
[56] A. Cichocki, Tensor networks for big data analytics and large-scale optimization problems, (2014), arXiv preprint arXiv:1407.3124.
[57] A. Cichocki et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends Mach. Learn. **9** (2016), 249–429.

[58] A. Cichocki et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives, Found. Trends Mach. Learn. **9** (2017), 249–429.

[59] A. Cichocki, R. Zdunek, and S.-i. Amari, Nonnegative matrix and tensor factorization, IEEE Signal Process. Mag. **25** (2008), 142–145.

[60] A. Cichocki et al., *Nonnegative matrix and tensor factorizations*, John Wiley & Sons, Hoboken, NJ, Ltd, 2009.

[61] A. Cohen and R. DeVore, Approximation of high-dimensional parametric PDEs, Acta Numer. **24** (2015), 1–159.

[62] E. Cohen and D. D. Lewis, Approximating matrix multiplication for pattern recognition tasks, J. Algor. **30** (1999), 211–252.

[63] C. Cortes and V. Vapnik, Support-vector networks, Mach. Learn. **20** (1995), 273–297.

[64] A. Cortinovis and D. Kressner, Low-rank approximation in the Frobenius norm by column and row subset selection, (2019), arXiv preprint arXiv:1908.06059.

[65] F. Dangel and P. Hennig, A modular approach to block–diagonal Hessian approximations for second-order optimization methods, (2019), arXiv preprint arXiv:1902.01813.

[66] J. A. de la Peña, I. Gutman, and J. Rada, Estimating the estrada index, Linear Algebra Appl. **427** (2007), 70–76.

[67] L. De Lathauwer, *Signal processing based on multilinear algebra*, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[68] L. De Lathauwer, B. De Moor, and J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. **21** (2000), 1253–1278.

[69] S. De Nigris, E. Bautista, P. Abry, K. Avrachenkov, and P. Gonçalves, Fractional graph-based semi-supervised learning, Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 356–360.

[70] M. Defferrard, X. Bresson, and P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process Syst. (2016), 3844–3852.

[71] H. Deng, S. Radenkovic, and I. Gutman, *The Estrada index*, in *Applications graph spectra*, Mathematical Institute, Belgrade, 2009, 123–140.

[72] T. Derr, Y. Ma, and J. Tang, Signed graph convolutional networks, Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 929–934.

[73] A. Deshpande and L. Rademacher, Efficient volume sampling for row/column subset selection, Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, IEEE, 2010, pp. 329–338.

[74] R. A. DeVore, Nonlinear approximation, Acta Numer. **7** (1998), 51–150.

[75] C. Ding, X. He, and H. D. Simon, On the equivalence of nonnegative matrix factorization and spectral clustering, Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM, Society for Industrial and Applied Mathematics, 2005, pp. 606–610.

[76] C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, A min-max cut algorithm for graph partitioning and data clustering, Proceedings of the 2001 IEEE International Conference on Data Mining, IEEE, IEEE Computer Society 2001, pp. 107–114.

[77] C. H. Ding, T. Li, and M. I. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Trans. Pattern Anal. Mach. Intell. **32** (2008), 45–55.

[78] K. Dong et al., *Scalable log determinants for Gaussian process kernel learning*, in *Advances in neural information processing systems*, Curran Associates, New York, 2017, 6327–6337.

[79] J. Dongarra and F. Sullivan, Guest editors introduction to the top 10 algorithms, Comput. Sci. Eng. **2** (2000), 22–23.

[80] D. Donoho, 50 years of data science, J Comput Graph Stat **26** (2017), 745–766.

[81] D. Donoho and V. Stodden, *When does non-negative matrix factorization give a correct decomposition into parts?* in *Advances in neural information processing systems*, Curran Associates, New York, 2004, 1141–1148.

[82] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices i: Approximating matrix multiplication, SIAM J. Comput. **36** (2006a), 132–157.

[83] P. Drineas, R. Kannan, and M. W. Mahoney, Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix, SIAM J. Comput. **36** (2006b), 158–183.

[84] P. Drineas et al., Fast approximation of matrix coherence and statistical leverage, J. Mach. Learn. Res. **13** (2012), 3475–3506.

[85] P. Drineas and M. W. Mahoney, On the Nyström method for approximating a gram matrix for improved kernel-based learning, J Mach Learn Res. **6** (2005), 2153–2175.

[86] P. Drineas and M. W. Mahoney, RandNLA, Commun. ACM **59** (2016), 80–90.

[87] P. Drineas et al., Faster least squares approximation, Numer. Math. **117** (2010), 219–249.

[88] H. Drucker et al., *Support vector regression machines*, in *Advances in neural information processing systems*, Curran Associates, New York, 1997, 155–161.

[89] V. Druskin and L. Knizhnerman, Extended Krylov subspaces: Approximation of the matrix square root and related functions, SIAM J. Matrix Anal. Appl. **19** (1998), 755–771.

[90] D. Dũng, V. Temlyakov, and T. Ullrich, *Hyperbolic cross approximation*, Springer International Publishing, New York, NY, 2018.

[91] L. Eldén, Partial least-squares vs. Lanczos bidiagonalization—I: Analysis of a projection method for multiple regression, Comput. Stat. Data Anal. **46** (2004), 11–31.

[92] L. Eldén, *Matrix methods in data mining and pattern recognition*, Vol **15**, SIAM, Philadelphia, 2019.

[93] N. B. Erichson and C. Donovan, Randomized low-rank dynamic mode decomposition for motion detection, Comput. Vis. Image Underst. **146** (2016), 40–50.

[94] S. Eriksson-Bique et al., Importance sampling for a Monte Carlo matrix multiplication algorithm, with application to information retrieval, SIAM J. Sci. Comput. **33** (2011), 1689–1706.

[95] E. Estrada, Characterization of 3D molecular structure, Chem. Phys. Lett. **319** (2000), 713–718.

[96] E. Estrada, N. Hatano, and M. Benzi, The physics of communicability in complex networks, Phys. Rep. **514** (2012), 89–119.

[97] E. Estrada and D. J. Higham, Network properties revealed through matrix functions, SIAM Rev. **52** (2010), 696–714.

[98] E. Estrada and J. A. Rodríguez-Velázquez, Subgraph centrality in complex networks, Phys. Rev. E **71** (2005), 056103.

[99] J. F. Hair Jr. et al., Partial least squares structural equation modeling (PLS-SEM), Europ. Bus. Rev. **26** (2014), 106–121.

[100] S. Fan and B. Huang, Labeled graph generative adversarial networks, CoRR (2019), abs/1906.03220.

[101] J. Gallier, Spectral theory of unsigned and signed graphs. applications to graph clustering: A survey, (2016), arXiv preprint arXiv:1601.04692.

[102] F. Gantmacher, *The theory of matrices*, Vol **I**, American Mathematical Society, Providence, Rhode Island, 1964, 95–103.

[103] N. Gillis, *The why and how of nonnegative matrix factorization*, in *Regularization, optimization, kernels, and support vector machines*, Vol **12**, Chapman & Hall/CRC, London, 2014.

[104] Y. Ginosar et al., Estrada index and Chebyshev polynomials, Chem. Phys. Lett. **454** (2008), 145–147.

[105] G. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, J. Soc. Ind. Appl Math Ser B Numer Anal **2** (1965), 205–224.

[106] G. H. Golub and G. Meurant, *Matrices, moments and quadrature*, Pitman research notes in mathematics series, Longman Scientic & Technical Harlow, Essex. 1994, 105–105.

[107] G. H. Golub and G. Meurant, *Matrices, moments and quadrature with applications*, Vol **30**, Princeton University Press, Princeton, NJ, 2009.

[108] G. H. Golub, M. Stoll, and A. Wathen, Approximation of the scattering amplitude and linear systems, Electron. Trans. Numer. Anal. **31** (2008), 178–203.

[109] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[110] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013.

[111] G. H. Golub and J. H. Welsch, Calculation of Gauss quadrature rules, Math. Comput. **23** (1969), 221.

[112] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, Cambridge, MA, 2016.

[113] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, Pseudo-skeleton approximations by matrices of maximal volume, Math. Notes **62** (1997), 515–519.

[114] R. M. Gower and P. Richtárik, Randomized iterative methods for linear systems, SIAM J. Matrix Anal. Appl **36** (2015), 1660–1690.

[115] L. Grasedyck, D. Kressner, and C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen **36** (2013), 53–78.

[116] E. Grilli, F. Menna, and F. Remondino, A review of point clouds segmentation and classification algorithms, Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. **42** (2017), 339.

[117] M. Gu, Subspace iteration randomization and singular value problems, SIAM J. Sci. Comput. **37** (2015), A1139–A1173.

[118] M. Gu and S. C. Eisenstat, Efficient algorithms for computing a strong rank-revealing QR factorization, SIAM J. Sci. Comput. **17** (1996), 848–869.

[119] E. Gujral and E. E. Papalexakis, SMACD: semi-supervised multi-aspect community detection, Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 702–710.

[120] S. Günther, L. Ruthotto, J. B. Schroder, E. Cyr, and N. R. Gauger, Layer-parallel training of deep residual neural networks, (2018), arXiv preprint arXiv:1812.04352.

[121] J. Gusak, M. Kholiavchenko, E. Ponomarev, L. Markeeva, P. Blagoveschensky, A. Cichocki, and I. Oseledets, Automated multi-stage compression of neural networks, Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019a.

[122] J. Gusak, M. Kholyavchenko, E. Ponomarev, L. Markeeva, I. Oseledets, and A. Cichocki, MUSCO: Multi-stage compression of neural networks, (2019b), arXiv preprint arXiv:1903.09973.

[123] S. Güttel, Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection, GAMM-Mitteilungen **36** (2013), 8–31.

[124] S. Güttel, D. Kressner, and K. Lund, Limited-memory polynomial methods for large-scale matrix functions, (2020), arXiv preprint arXiv:2002.01682.

[125] E. Haber and L. Ruthotto, Stable architectures for deep neural networks, Inverse Prob. **34** (2017), 014004.

[126] P. Hage, A graph theoretic approach to the analysis of alliance structure and local grouping in highland New Guinea, Anthropol. Forum **3** (1973), 280–294.

[127] J. F. Hair Jr. et al., *A primer on partial least squares structural equation modeling (PLS-SEM)*, Sage Publications, Thousand Oaks, California, 2016.

[128] D. Hajinezhad, T.-H. Chang, X. Wang, Q. Shi, and M. Hong, Nonnegative matrix factorization using ADMM: Algorithm and convergence analysis. Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016, pp. 4742–4746.

[129] N. Hale, N. J. Higham, and L. N. Trefethen, Computing Aα, log(A) and related matrix functions by contour integrals, SIAM J. Numer. Anal. **46** (2008), 2505–2523.

[130] N. Halko, P. G. Martinsson, and J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. **53** (2011), 217–288.

[131] K. Hamm and L. Huang, Perturbations of CUR decompositions, (2019), arXiv preprint arXiv:1908.08101.

[132] P. C. Hansen, The truncated SVD as a method for regularization, BIT **27** (1987), 534–553.

[133] J. A. Hartigan and M. A. Wong, Algorithm as 136: A k-means clustering algorithm, J. R. Stat. Soc. C-Appl. **28** (1979), 100–108.

[134] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Springer, New York, NY, 2009.

[135] C. Hayashi, *What is data science ? fundamental concepts and a heuristic example*, in *Studies in classification, data analysis, and knowledge organization*, Springer, Japan, 1998, 40–51.

[136] L. He, X. Kong, P. S. Yu, X. Yang, A. B. Ragin, and Z. Hao, DuSK: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages, Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM, Society for Industrial and Applied Mathematics, 2014, pp. 127–135.

[137] L. He, C.-T. Lu, G. Ma, S. Wang, L. Shen, P. S. Yu, and A. B. Ragin, Kernelized support tensor machines, Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR, 2017, pp. 1442–1451.

[138] M. Henaff, J. Bruna, and Y. LeCun, Deep convolutional networks on graph-structured data, (2015), arXiv preprint arXiv:1506.05163.

[139] V. Hernández, J. E. Román, and A. Tomás, A robust and efficient parallel SVD solver based on restarted Lanczos bidiagonalization, Electron. Trans. Numer. Anal. **31** (2008), 68–85.

[140] V. Hernández et al., *Restarted Lanczos bidiagonalization for the SVD in SLEPc*, in *STR-8 technical report*, Springer, New York, NY, 2007.

[141] M. R. Hestenes, *Conjugate direction methods in optimization*, Vol **49**, Springer, New York, NY, 1980.

[142] C. F. Higham and D. J. Higham, Deep learning: An introduction for applied mathematicians, SIAM Rev. **61** (2019), 860–891.

[143] N. J. Higham, *Functions of matrices*, Vol **104**, Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[144] N. J. Higham and E. Deadman, *A catalogue of software for matrix functions. Version 2.0*, (2016).

[145] M. Hochbruck and C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. **34** (1997), 1911–1925.

[146] M. E. Hochstenbach, Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems, BIT **44** (2004), 721–754.

[147] T. Hofmann, B. Schölkopf, and A. J. Smola, Kernel methods in machine learning, Ann. Stat. **36** (2008), 1171–1220.

[148] C.-J. Hsieh, S. Si, and I. S. Dhillon, *Fast prediction for large-scale kernel machines*, in *Advances in neural information processing systems*, Curran Associates, New York, 2014, 3689–3697.

[149] J. Hulland, Use of partial least squares (PLS) in strategic management research: A review of four recent studies, Strat. Mgmt. J. **20** (1999), 195–204.

[150] M. Hutchinson, A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines, Commun Stat Simulat Comput **19** (1990), 433–450.

[151] M. Jaderberg, A. Vedaldi, and A. Zisserman, Speeding up convolutional neural networks with low rank expansions, (2014), arXiv preprint arXiv:1405.3866.

[152] G. James et al., *An introduction to statistical learning*, Springer, New York, NY, 2013.

[153] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognit **44** (2011), 2231–2240.

[154] I. T. Jolliffe, *Principal component analysis*, Springer, New York, NY, 1986.

[155] I. T. Jolliffe and J. Cadima, Principal component analysis: A review and recent developments, Phil. Trans. R. Soc. A **374** (2016), 20150202.

[156] L. Kämmerer, D. Potts, and T. Volkmer, Approximation of multivariate periodic functions by trigonometric polynomials based on rank-1 lattice sampling, J Complex. **31** (2015), 543–576.

[157] T. Kanungo et al., An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002), 881–892.

[158] A. Kheradmand and P. Milanfar, A general framework for kernel similarity-based image denoising, Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 415–418.

[159] A. Kheradmand and P. Milanfar, A general framework for regularized, similarity-based image restoration, IEEE Trans. Image Process **23** (2014), 5136–5151.

[160] H. A. L. Kiers, Towards a standardized notation and terminology in multiway analysis, J. Chemometrics **14** (2000), 105–122.

[161] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, (2016a), arXiv preprint arXiv:1609.02907.

[162] T. N. Kipf and M. Welling, Variational graph auto-encoders, (2016b), arXiv preprint arXiv:1611.07308.

[163] N. Kishore Kumar and J. Schneider, Literature survey on low rank approximation of matrices, Linear Multilinear A **65** (2016), 2212–2244.

[164] M. Kivela et al., Multilayer networks, SSRN J. **2** (2013), 203–271.

[165] S. Klamt, U.-U. Haus, and F. Theis, Hypergraphs and cellular networks, PLoS Comput Biol **5** (2009), e1000385.

[166] L. Knizhnerman and V. Simoncini, A new investigation of the extended Krylov subspace method for matrix function evaluations, Numer. Linear Algebra Appl. **17** (2009), 615–638.

[167] T. G. Kolda and B. W. Bader, Tensor decompositions and applications, SIAM Rev. **51** (2009), 455–500.

[168] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, Computer (2009), 42(8), 30–37.

[169] F. Krzakala et al., Spectral redemption in clustering sparse networks, Proc. Nat. Acad. Sci. **110** (2013), 20935–20940.

[170] J. N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. **814** (2017), 1–4.

[171] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, United States Government Press Office, Los Angeles, CA, 1950.

[172] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, Speeding-up convolutional neural networks using fine-tuned CP-decomposition, (2014), arXiv preprint arXiv:1412.6553.

[173] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, Handbook Brain Theory Neural Netw. **3361** (1995), no. 10, 1995.
[174] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature **521** (2015), 436–444.
[175] D. D. Lee and H. S. Seung, Algorithms for non-negative matrix factorization, Adv Neural Inf Process Syst (2001), 556–562.
[176] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide*, Vol **6**, Society for Industrial and Applied Mathematics, Philadelphia, 1998.
[177] M. Leordeanu, A. Zanfir, C. Sminchisescu, Semi-supervised learning and optimization for hypergraph matching, Proceedings of the 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2274–2281.
[178] J. Leskovec, D. Huttenlocher, and J. Kleinberg, Predicting positive and negative links in online social networks, Proceedings of the 19th International Conference on World Wide Web - WWW '10, ACM Press, 2010a, pp. 641–650.
[179] J. Leskovec, D. Huttenlocher, and J. Kleinberg, Signed networks in social media, Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10, ACM Press, 2010b, pp. 1361–1370.
[180] R. Levie et al., CayleyNets: Graph convolutional neural networks with complex rational spectral filters, IEEE Trans. Signal Process. **67** (2019), 97–109.
[181] C. Li and G. Stadler, Sparse solutions in optimal control of PDEs with uncertain parameters: The linear case, SIAM J. Control Optim. **57** (2019), 633–658.
[182] M. Li et al., Large-scale nyström kernel matrix approximation using randomized SVD, IEEE Trans. Neural Netw. Learn. Syst. **26** (2015), 152–164.
[183] S. Li, Y. Jin, and D. I. Shuman, Scalable M channel critically sampled filter banks for graph signals, IEEE Trans. Signal Process. **67** (2019), 3954–3969.
[184] X. Li, G. Cui, and Y. Dong, Graph regularized non-negative low-rank matrix factorization for image clustering, IEEE Trans. Cybern. **47** (2017), 3840–3853.
[185] L. Lin, Y. Saad, and C. Yang, Approximating spectral densities of large matrices, SIAM Rev. **58** (2016), 34–65.
[186] H. Liu et al., Constrained nonnegative matrix factorization for image representation, IEEE Trans. Pattern Anal. Mach. Intell. **34** (2012), 1299–1311.
[187] S. Liu, L. Chen, H. Dong, Z. Wang, D. Wu, and Z. Huang, Higher-order weighted graph convolutional networks, (2019), arXiv preprint arXiv:1911.04129.
[188] X. Liu et al., Progressive image denoising through hybrid graph Laplacian regularization: A unified framework, IEEE Trans. Image Process. **23** (2014), 1491–1503.
[189] X. Luo et al., A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Netw. Learn. Syst. **27** (2016), 579–592.
[190] D. J. MacKay, Introduction to Gaussian processes, NATO ASI Ser. F Comput Syst. Sci **168** (1998), 133–166.
[191] G. Madjarov et al., An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. **45** (2012), 3084–3104.
[192] M. W. Mahoney and P. Drineas, CUR matrix decompositions for improved data analysis, PNAS **106** (2009), 697–702.
[193] M. W. Mahoney, M. Maggioni, and P. Drineas, Tensor-CUR decompositions for tensor-based data, SIAM J. Matrix Anal. Appl. **30** (2008), 957–987.
[194] W. B. March, B. Xiao, S. Tharakan, C. D. Yu, and G. Biros, A kernel-independent FMM in general dimensions, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15, IEEE, ACM Press, 2015, pp. 1–12.
[195] P.-G. Martinsson, Randomized methods for matrix computations, Math. Data **25** (2018), 187–231.
[196] P.-G. Martinsson and J. Tropp, Randomized numerical linear algebra: Foundations and algorithms, (2020), arXiv preprint arXiv:2002.01387.
[197] P. Mercado, J. Bosch, and M. Stoll, Node classification for signed social networks using diffuse interface methods, Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Cham, 2019a, pp. 524-540.
[198] P. Mercado, A. Gautier, F. Tudisco, and M. Hein, The power mean Laplacian for multilayer graph clustering, (2018), arXiv preprint arXiv:1803.00491.
[199] P. Mercado, F. Tudisco, and M. Hein, *Clustering signed networks with the geometric mean of Laplacians*, in *Advances in neural information processing systems*, Curran Associates, New York, 2016, 4421–4429.
[200] P. Mercado, F. Tudisco, and M. Hein, *Generalized matrix means for semi-supervised learning with multilayer graphs*, in *Advances in neural information processing systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., **32**, 2019b, pp. 14877–14886.
[201] P. Mercado, F. Tudisco, and M. Hein, *Spectral clustering of signed graphs via matrix power means*, in *Proceedings of the 36th International Conference on Machine Learning*, Vol **97**, K. Chaudhuri and R. Salakhutdinov, Eds., PMLR, 2019c, 4526–4536.
[202] T. Metsalu and J. Vilo, ClustVis: A web tool for visualizing clustering of multivariate data using principal component analysis and heatmap, Nucleic Acids Res. **43** (2015), W566–W570.
[203] G. Meurant, Estimates of the trace of the inverse of a symmetric matrix using the modified Chebyshev algorithm, Numer Algor. **51** (2008), 309–318.
[204] P. Milanfar, A tour of modern image filtering: New insights and methods, both practical and theoretical, IEEE Signal Process. Mag. **30** (2013), 106–128.
[205] H. J. Miller and J. Han, *Geographic data mining and knowledge discovery*, CRC Press, Boca Raton, FL, 2009.

[206] C. Moler and C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Rev. **20** (1978), 801–836.

[207] C. Moler and C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Rev. **45** (2003), 3–49.

[208] V. I. Morariu et al., *Automatic online tuning for fast Gaussian summation*, in *Advances in neural information processing systems*, Curran Associates, New York, 2009, 1113–1120.

[209] F. Morbidi, The deformed consensus protocol, Automatica **49** (2013), 3049–3055.

[210] K.-R. Muller et al., An introduction to kernel-based learning algorithms, IEEE Trans. Neural Netw. **12** (2001), 181–201.

[211] Y. Nakatsukasa and N. J. Higham, Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD, SIAM J. Sci. Comput. **35** (2013), A1325–A1349.

[212] D. Needell, Randomized Kaczmarz solver for noisy linear systems, BIT **50** (2010), 395–403.

[213] M. E. J. Newman, Detecting community structure in networks, Europ. Phys. J. B Condens Matter **38** (2004), 321–330.

[214] M. E. J. Newman, Modularity and community structure in networks, Proc. Nat. Acad. Sci. **103** (2006), 8577–8582.

[215] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, in *Advances in neural information processing systems*, Curran Associates, New York, 2002, 849–856.

[216] A. Nouy, Higher-order principal component analysis for the approximation of tensors in tree-based low-rank formats, Numer. Math. **141** (2019), 743–789.

[217] A. Novikov et al., Tensorizing neural networks, Adv Neural Inf Process Syst (2015), 442–450.

[218] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. **33** (2011), 2295–2317.

[219] A. Osinsky and N. Zamarashkin, Pseudo-skeleton approximations with better accuracy estimates, Linear Algebra Appl. **537** (2018), 221–249.

[220] A. Paranjape, A. R. Benson, and J. Leskovec, Motifs in temporal networks, Proceedings of the 10th ACM International Conference on Web Search and Data Mining - WSDM '17, ACM Press, 2017, pp. 601–610.

[221] J. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, Microsoft, Albuquerque, New Mexico, (1998).

[222] I. Podlubny, *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, Vol **198**, Elsevier, Amsterdam, 1998.

[223] F. Pourkamali-Anaraki, S. Becker, and M. B. Wakin, Randomized clustered Nyström for large-scale kernel machines, Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018.

[224] X. Qi et al., Laplacian centrality: A new centrality measure for weighted networks, Inf. Sci. **194** (2012), 240–253.

[225] F. Radicchi, Driving interconnected networks to supercriticality, Phys. Rev. X **4** (2014), 021014.

[226] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, in *Advances in neural information processing systems*, Curran Associates, New York, 2008, 1177–1184.

[227] S. S. Rangapuram, T. Bühler, and M. Hein, Towards realistic team formation in social networks based on densest subgraphs, Proceedings of the 22nd International Conference on World Wide Web - WWW '13, ACM Press, 2013, pp. 2427–2435.

[228] J. Rapin et al., Sparse and non-negative BSS for noisy data, IEEE Trans. Signal Process. **61** (2013), 5620–5632.

[229] J. Rapin et al., NMF with sparse regularizations in transformed domains, SIAM J. Imag. Sci. **7** (2014), 2020–2047.

[230] C. E. Rasmussen, *Gaussian processes in machine learning*, in *Summer school on machine learning*, Springer, New York, NY, 2003, 63–71.

[231] Y. Romano, M. Elad, and P. Milanfar, The little engine that could: Regularization by denoising (RED), SIAM J. Imag. Sci. **10** (2017), 1804–1844.

[232] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*, Vol **589**, John Wiley & Sons, Inc, Hoboken, NJ, 1987.

[233] A. Rudi, L. Carratino, and L. Rosasco, *Falkon: An optimal large scale kernel method*, in *Advances in neural information processing systems*, Curran Associates, New York, 2017, 3888–3898.

[234] H. Rue, Fast sampling of Gaussian Markov random fields, J. R. Stat. Soc. B **63** (2001), 325–338.

[235] H. Rue and L. Held, *Gaussian Markov random fields*, Chapman & Hall/CRC Press, Boca Raton, FL, 2005.

[236] Y. Saad, *Iterative methods for sparse linear systems*, Vol **82**, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[237] Y. Saad, *Numerical methods for large eigenvalue problems*, Vol **66**, Society for Industrial and Applied Mathematics, Philadelphia, 2011.

[238] A. Saade, F. Krzakala, and L. Zdeborová, *Spectral clustering of graphs with the Bethe Hessian*, in *Advances in neural information processing systems*, Curran Associates, New York, 2014, 406–414.

[239] L. Sagun, L. Bottou, and Y. LeCun, Singularity of the Hessian in deep learning, (2016), arXiv preprint arXiv:1611.07476.

[240] A. K. Saibaba, HOID: higher order interpolatory decomposition for tensors based on tucker representation, SIAM J. Matrix Anal. Appl. **37** (2016), 1223–1249.

[241] A. K. Saibaba, J. Lee, and P. K. Kitanidis, Randomized algorithms for generalized Hermitian eigenvalue problems with application to computing karhunen-loève expansion, Numer. Linear Algebra Appl. **23** (2015), 314–339.

[242] R. Schaback and H. Wendland, Kernel techniques: From machine learning to meshless methods, Acta Numerica **15** (2006), 543–639.

[243] B. Schölkopf, *The kernel trick for distances*, in *Advances in neural information processing systems*, Curran Associates, New York, 2001, pp. 301–307.

[244] B. Schölkopf, A. Smola, and K.-R. Müller, Kernel principal component analysis, Proceedings of the International Conference on Artificial Neural Networks, New York, NY, Springer, 1997, pp. 583–588.

[245] B. Schölkopf, A. Smola, and K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. **10** (1998), 1299–1319.

[246] B. Schölkopf and A. J. Smola, *Learning with kernels*, The MIT Press, Cambridge, MA, 2018.
[247] J. Sedoc, J. Gallier, D. Foster, and L. Ungar, Semantic word clusters using signed spectral clustering, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2017, pp. 939–949.
[248] G. Shabat, E. Choshen, D. Ben-Or, and N. Carmel, Fast and accurate Gaussian kernel ridge regression using matrix decompositions for preconditioning, (2019), arXiv preprint arXiv:1905.10587.
[249] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge University Press, Cambridge, MA, 2004.
[250] J. Shi and J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Machine Intell. **22** (2000), 888–905.
[251] Y. Shitov, Column subset selection is NP-complete, (2017), arXiv preprint arXiv:1701.02764.
[252] D. I. Shuman et al., The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. **30** (2013), 83–98.
[253] D. I. Shuman et al., Distributed signal processing via Chebyshev polynomial approximation, IEEE Trans. Signal Inf. Process. over Netw **4** (2018), 736–751.
[254] D. P. Simpson, I. W. Turner, A. N. Pettitt, Fast sampling from a Gaussian Markov random field using Krylov subspace approaches, (2008).
[255] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, Stat Comput **14** (2004), 199–222.
[256] A. Solé-Ribalta et al., Spectral properties of the Laplacian of multiplex networks, Phys. Rev. E **88** (2013), 032807.
[257] D. C. Sorensen and M. Embree, A DEIM induced CUR factorization, SIAM J. Sci. Comput. **38** (2016), A1454–A1482.
[258] B. V. Srinivasan, Q. Hu, N. A. Gumerov, R. Murtugudde, and R. Duraiswami, Preconditioned Krylov solvers for kernel regression, (2014), arXiv preprint arXiv:1408.1237.
[259] D. Steinley, K-means clustering: A half-century synthesis, Br. J. Math. Stat. Psychol. **59** (2006), 1–34.
[260] G. Stewart, Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix, Numer Math **83** (1999), 313–323.
[261] G. W. Stewart, *Matrix algorithms*, Vol **1**, Society for Industrial and Applied Mathematics, Philadelphia, 1998.
[262] G. W. Stewart, *Matrix algorithms*, Vol **2**, Society for Industrial and Applied Mathematics, Philadelphia, 2001.
[263] M. Stoll, A Krylov–Schur approach to the truncated SVD, Linear Algebra Appl. **436** (2012), 2795–2806.
[264] Z. Strakoš and P. Tichý, On efficient numerical approximation of the bilinear form c*A−1b, SIAM J. Sci. Comput. **33** (2011), 565–587.
[265] G. Strang, *Linear algebra and learning from data*, Cambridge Press, Wellesley, MA, 2019.
[266] D. Sukkari, H. Ltaief, and D. Keyes, A high performance QDWH-SVD solver using hardware accelerators, ACM Trans. Math. Softw. **43** (2016), 1–25.
[267] J. Tang et al., A survey of signed network mining in social media, ACM Comput. Surv. **49** (2016), 1–37.
[268] Y. Tang, Deep learning using linear support vector machines, (2013), arXiv preprint arXiv:1306.0239.
[269] D. Tao, X. Li, W. Hu, S. Maybank, X. Wu, Supervised tensor learning, Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05), IEEE, 2005, p. 8.
[270] D. Taylor et al., Eigenvector-based centrality measures for temporal networks, Multiscale Model. Simul. **15** (2017), 537–574.
[271] V. Temlyakov, *Greedy approximation*, Vol **20**, Cambridge University Press, Cambridge, MA, 2009.
[272] E. Teoh, K. Tan, and C. Xiang, Estimating the number of hidden neurons in a feedforward network using the singular value decomposition, IEEE Trans. Neural Netw. **17** (2006), 1623–1629.
[273] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Royal Stat. Soc. Ser. B (Methodol.) **58** (1996), 267–288.
[274] S. Trehan, K. T. Carlberg, and L. J. Durlofsky, Error modeling for surrogates of dynamical systems using machine learning, Int. J. Numer Methods Eng. **112** (2017), 1801–1827.
[275] S. Tu, S. Venkataraman, A. C. Wilson, A. Gittens, M. I. Jordan, and B. Recht, Breaking locality accelerates block Gauss-seidel, Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR, 2017, pp. 3482–3491.
[276] E. Tyrtyshnikov, Mosaic-skeleton approximations, Calcolo **33** (1996), 47–57.
[277] S. Ubaru, J. Chen, and Y. Saad, Fast estimation of tr(f(A)) via stochastic Lanczos quadrature, SIAM J. Matrix Anal. Appl. **38** (2017), 1075–1099.
[278] J. C. Urschel, Nodal decompositions of graphs, Linear Algebra Appl. **539** (2018), 60–71.
[279] W. van der Aalst, *Data science in action*, Springer, Berlin Heidelberg/Germany, 2016, 3–23.
[280] V. Vapnik, *Estimation of dependences based on empirical data: springer series in statistics (Springer series in statistics)*, Springer-Verlag, Berlin, Heidelberg/Germany, 1982.
[281] S. Vatankhah, R. A. Renaut, and V. E. Ardestani, Total variation regularization of the 3-d gravity inverse problem using a randomized generalized singular value decomposition, Geophys. J. Int. **213** (2018), 695–705.
[282] S. A. Vavasis, On the complexity of nonnegative matrix factorization, SIAM J. Optim. **20** (2010), 1364–1377.
[283] O. Vinyals and D. Povey, Krylov subspace descent for deep learning, Artif. Intell. Stat. **22** (2012), 1261–1268.
[284] C. R. Vogel and M. E. Oman, Iterative methods for total variation denoising, SIAM J. Sci. Comput. **17** (1996), 227–238.
[285] U. von Luxburg, A tutorial on spectral clustering, Stat Comput **17** (2007), 395–416.
[286] S. Voronin and P.-G. Martinsson, Efficient algorithms for CUR and interpolative matrix decompositions, Adv Comput Math **43** (2016), 495–516.
[287] C.-C. Wang, K. L. Tan, and C.-J. Lin, Newton methods for convolutional neural networks, (2018), arXiv preprint arXiv:1811.06100.
[288] S. Wang and Z. Zhang, Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling, J Mach Learn Res. **14** (2013), 2729–2769.

[289] Y. Wang et al., *Fast and guaranteed tensor decomposition via sketching*, in *Advances in neural information processing systems*, Curran Associates, New York, 2015, 991–999.
[290] T. Warren Liao, Clustering of time series data–a survey, Pattern Recognit. **38** (2005), 1857–1874.
[291] A. Wilson and H. Nickisch, Kernel interpolation for scalable structured Gaussian processes (KISS-GP), Proceedings of the International Conference on Machine Learning, 2015, pp. 1775–1784.
[292] D. P. Woodruff et al., Sketching as a tool for numerical linear algebra, Found Trends Theor Comput Sci **10** (2014), 1–157.
[293] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, (2019), arXiv preprint arXiv:1901.00596.
[294] H. Xiang and J. Zou, Regularization with randomized SVD for large-scale discrete inverse problems, Inverse Prob. **29** (2013), 085008.
[295] H. Xiang and J. Zou, Randomized algorithms for large-scale inverse problems with general Tikhonov regularizations, Inverse Prob. **31** (2015), 085008.
[296] J. Xue, J. Li, and Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, Interspeech (2013), 2365–2369.
[297] N. Yadati, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, HyperGCN: Hypergraph convolutional networks for semi-supervised classification, (2018), arXiv preprint arXiv:1809.02589.
[298] C. Yang, R. Duraiswami, and L. S. Davis, *Efficient kernel machines using the improved fast Gauss transform*, in *Advances in neural information processing systems*, Curran Associates, New York, 2005, 1561–1568.
[299] S. Yi et al., Joint sparse principal component analysis, Pattern Recognit. **61** (2017), 524–536.
[300] Y. You, J. Demmel, C.-J. Hsieh, and R. Vuduc, Accurate, fast and scalable kernel ridge regression on parallel and distributed systems, Proceedings of the 2018 International Conference on Supercomputing - ICS '18, ACM Press, 2018, pp. 307–317.
[301] R. Yousefzadeh and D. P. O'Leary, Refining the structure of neural networks using matrix conditioning, (2019), arXiv preprint arXiv:1908.02400.
[302] C. D. Yu, J. Levitt, S. Reiz, and G. Biros, Geometry-oblivious FMM for compressing dense SPD matrices, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '17, ACM Press, 2017, p. 53.
[303] C. D. Yu, W. B. March, B. Xiao, and G. Biros, INV-ASKIT: A parallel fast direct solver for kernel matrices, Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2016, pp. 161–171.
[304] J. Yu, D. Tao, and M. Wang, Adaptive hypergraph learning and its application in image classification, IEEE Trans. Image Process. **21** (2012), 3262–3272.
[305] A. Zear, A. K. Singh, and P. Kumar, A proposed secure multiple watermarking technique based on DWT, DCT and SVD for application in medicine, Multimed. Tools Appl. **77** (2016), 4863–4882.
[306] L. Zelnik-Manor and P. Perona, *Self-tuning spectral clustering*, in *Advances in neural information processing systems*, Curran Associates, New York, 2005, 1601–1608.
[307] K. Zhang, I. W. Tsang, and J. T. Kwok, Improved Nyström low-rank approximation and error analysis, Proceedings of the 25th International Conference on Machine Learning - ICML '08, ACM Press, 2008, pp. 1232–1239.
[308] Q. Zhang and B. Li, Discriminative k-SVD for dictionary learning in face recognition, Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2691–2698.
[309] Z.-K. Zhang and C. Liu, A hypergraph model of social tagging networks, J. Stat. Mech. **2010** (2010), P10005.
[310] D. Zhou, J. Huang, and B. Schölkopf, Beyond pairwise classification and clustering using hypergraphs, (2005).
[311] D. Zhou, J. Huang, and B. Schölkopf, *Learning with hypergraphs: Clustering, classification, and embedding*, in *Advances in neural information processing systems*, Curran Associates, New York, 2007, 1601–1608.
[312] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, Graph neural networks: A review of methods and applications, (2018), arXiv preprint arXiv:1812.08434.

How to cite this article: Stoll M. A literature survey of matrix methods for data science. *GAMM-Mitteilungen*. 2020;43:e202000013. https://doi.org/10.1002/gamm.202000013