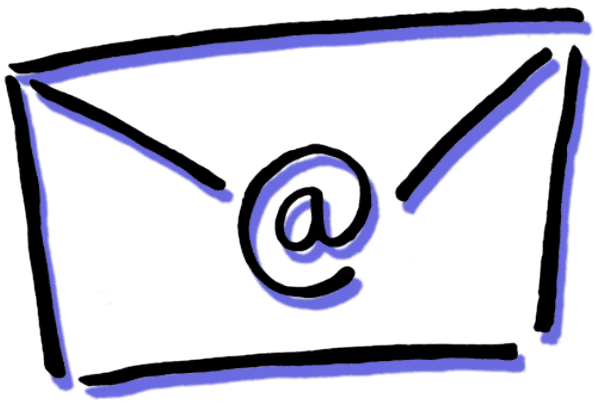
SIAG/OPT Views and News

A Forum for the SIAM Activity Group on Optimization

Volume 27 Number 1 January 2019

Contents

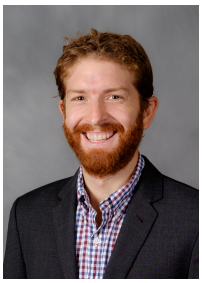
Are you receiving this by postal mail?
Do you prefer electronic delivery?



Email siagoptnews@lists.mcs.anl.gov!

Articles

CP and Tucker Tensor Decompositions

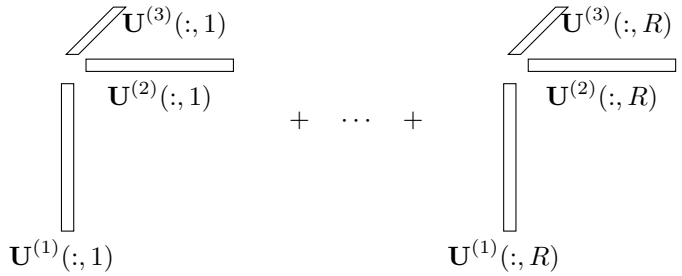


Grey Ballard
Department of Computer Science
Wake Forest University
Winston-Salem, NC, USA
Email: ballard@wfu.edu
http://users.wfu.edu/ballard

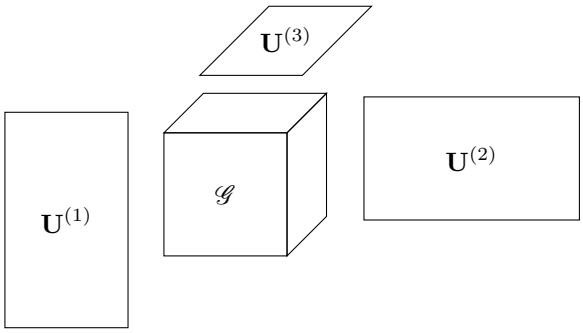
1 Introduction

Tensor decompositions are generalizations of low-rank matrix approximations to higher dimensional data. They have become popular for their utility in applications—including blind source separation, dimensionality reduction, compression, anomaly detection—when the original data is represented as a multidimensional array. As tensors and tensor computations have developed over time in fields such as physics [29], machine learning [2], algebraic geometry [22], and chemistry [8], differences have evolved in terminology and connotations. Moreover, while there are many flavors of low-rank matrix approximations, the generalizations to tensors are even more diverse. We note that the term tensor *decompositions* can be misleading, as they are rarely analogues of matrix decompositions like LU or QR that can be computed using direct methods, which are exact factorizations in exact arithmetic. Rather, the tensor decompositions we typically seek are *approximations*: solutions to optimization problems that are computed using iterative methods. In some cases, general-purpose optimization techniques are the best approach, but often the multilinear structure of tensor decompositions can be exploited to improve convergence or the quality of solution.

Here we'll consider two of the most popular tensor decompositions (CP and Tucker), and we'll highlight sample applications to illustrate how tensor decompositions are computed and utilized. In Section 2 we contrast two analyses of the famous Enron email data set, one using matrix-based analysis and the other using a tensor decomposition, to see the possi-



(a) CP



(b) Tucker

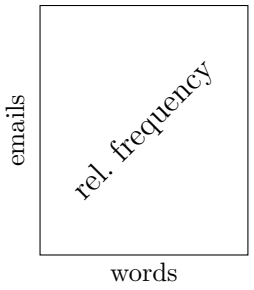
Figure 1: Decomposition structures

ble advantages of multiway analysis. Sections 3 and 4 focus on the CP and Tucker decompositions, respectively, presenting one popular algorithm and application for each. For more complete details of these and other decompositions, we recommend the classic survey of Kolda and Bader [17]. We also recommend several recent surveys with more up-to-date references to applications [11, 14, 26, 28, 30].

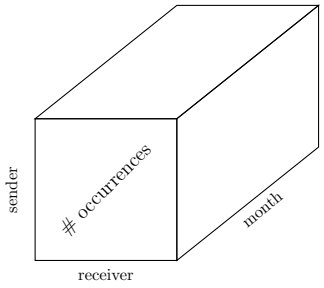
The CP and Tucker decompositions are related, and each can be considered a generalization of the matrix singular value decomposition (SVD). Each decomposes (or approximates) a tensor as a sum of rank-one tensors, where a rank-one tensor is an outer product of vectors. The truncated SVD can be expressed as $\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T} = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^\mathsf{T}$, where the sets of left and right singular vectors correspond to the two modes of the matrix (columns and rows). For both CP and Tucker, there exist analogues of the matrices $\mathbf{U}$ and $\mathbf{V}$, with a factor matrix $\mathbf{U}^{(n)}$ for each mode n of the tensor. In the case of CP, there are analogues of the singular values that weight each rank-one component, but CP differs from the SVD in that the factor matrices are not orthonormal (the columns are not necessarily pairwise orthogonal). In the case of Tucker, the factor matrices typically maintain the orthogonality property, but Tucker differs from the SVD in that the analogue of the singular value matrix $\mathbf{\Sigma}$ is a *dense* tensor instead of a diagonal one. Fig. 1 depicts the structure of the CP and Tucker decompositions, and we use tensor notation to define them explicitly in Sections 3 and 4.

2 Matrix-Based vs. Tensor-Based Analysis

As an example, we'll consider two different analyses of the Enron email data set. Enron was an American company at



(a) Matrix-based analysis



(b) Tensor-based analysis

Figure 2: Shapes of Enron data analyzed by two approaches

the center of an accounting fraud scandal and subsequently declared bankruptcy in 2001. The Federal Energy Regulatory Commission made a huge trove of company emails publicly available, and the data set has become a popular test bed for multiple research communities, used by as many as 30 papers a year [24]. Here we'll contrast two methods applied to the data set, one based on matrix analysis and the other on tensor analysis.

In a survey of algorithms and applications for nonnegative matrix factorization (NMF), Berry et al. [6] use the Enron data set as a sample application of NMF as a tool for document clustering. They shape the email data into a message-by-noun matrix $\mathbf{A}$, where $\mathbf{A}(i, j)$ is nonzero if email message i contains noun j. The precise numerical value of the entry is determined by a term-weighting scheme, but for our purposes we need to know only that every nonzero entry is positive.

Assuming $\mathbf{A}$ is $M \times N$, NMF determines nonnegative $M \times R$ and $N \times R$ matrices $\mathbf{W}$ and $\mathbf{H}$, such that $\mathbf{A} \approx \mathbf{W}\mathbf{H}^\mathsf{T} = \sum_{r=1}^{R} \mathbf{W}(:, r) \circ \mathbf{H}(:, r)$, where $\circ$ denotes an outer product (see [12] for a nice introduction to NMF and its algorithms and applications). We note that Berry et al. use a least-squares error function and additional constraint penalty terms in their NMF formulation to improve the quality of their results. Matrices $\mathbf{W}$ and $\mathbf{H}$ can be interpreted as discriminating among R concepts or topics. For the rth concept, $\mathbf{W}(:, r)$ encodes what email messages discussed the topic, and $\mathbf{H}(:, r)$ encodes the keywords of the topic; large entries signify high relevance. The keywords give clues to the meaning of each concept, and the most relevant emails of the most important concepts can be further studied.

Thus, this matrix-based approach is aimed at discovering patterns in the data to answer the exploratory question, "What were the main concepts discussed in the emails?" As expected, Berry et al. report discovering topics of email conversation related to business ventures around the world (discussion of Scotland and India were distinct concepts), the California blackout (which was exacerbated by Enron's trading practices), and the accounting scandal itself. They also note a particularly strong concept with keywords including American football player names, corresponding to email discussion surrounding fantasy football leagues among employees.

In separate work, Chi and Kolda [10] also use the Enron data set as a sample application of a nonnegative *tensor* decomposition. They shape the data into a person-by-person-by-month tensor $\mathscr{T}$, where $\mathscr{T}(i,j,k)$ is the number of emails person i sent to person j in month k, so that every entry in the tensor is a nonnegative integer. Similar to using NMF, Chi and Kolda use a CP decomposition to find underlying patterns in the data. However, by shaping the data as a tensor, their exploratory question involves a multiway relationship: "Who was emailing with whom *and when*?" This question is especially relevant in comparing the timestamps of emails to the start of the various investigations into the company and when details were made public.

In a CP decomposition, the tensor is approximated by a sum of rank-one tensors, much like NMF. However, in this case, a rank-one tensor is an outer product of three vectors. Because $\mathscr{T}$ represents count data in this application, Chi and Kolda constrain the factor matrices to be nonnegative and use Kullback-Liebler (KL) divergence as the error function. Each rank-one tensor represents a latent pattern, or component, in the data, and the entries of the corresponding columns of the factor matrices encode the relationship between the entities in each mode to the particular pattern. For each component, extra metadata on the employees and the timing of significant events can be used as clues to ascribe meaning to the patterns.

Chi and Kolda find one component with strong representation by senior employees as both senders and receivers that peaks after the time of the Security and Exchange Commission (SEC) investigation. Another component involves senders categorized as "legal" staff with receivers from other categories. This component shows greatest relevance right at the time of the SEC investigation, possibly corresponding to internal lawyers advising employees on interactions with investigators.

3 CP Decomposition

The CP decomposition approximates the input tensor with a sum of rank-one tensors, or components, where each rank-one tensor is an outer product of vectors [9, 15]. The R vectors in mode n of all components are typically collected as columns in a factor matrix $\mathbf{U}^{(n)}$. In tensor notation, the usual optimization problem for computing a CP decomposition with a specified rank R can be written as

$$\arg\min_{\{\mathbf{U}^{(n)}\}} \left\| \mathscr{X} - \sum_{r=1}^{R} \mathbf{U}^{(1)}(:,r) \circ \cdots \circ \mathbf{U}^{(N)}(:,r) \right\|^2,$$

where the tensor norm is the square root of the sum of squares of the entries (like the ℓ_2 vector norm). In element-wise notation, the objective function is

$$\sum_{i_1,\ldots,i_N} \left(\mathscr{X}(i_1,\ldots,i_N) - \sum_{r=1}^{R} \mathbf{U}^{(1)}(i_1,r) \cdots \mathbf{U}^{(N)}(i_N,r) \right)^2. \quad (1)$$

However, the objective function need not be based on the ℓ_2 norm; for applications like the example in Section 2, alternatives such as KL divergence are more appropriate.

3.1 Algorithms

One of the most popular algorithms for computing a CP decomposition is known as Alternating Least Squares (CP-ALS), which is given in Alg. 1. It alternates among the factor matrices, holding all but one fixed and solving the linear least-squares subproblem via the normal equations. Line 4 is known as the Matricized-Tensor Times Khatri-Rao Product (MTTKRP) and requires the bulk of the computation. The symbol $\odot$ denotes the Khatri-Rao product. A matricized tensor (denoted by $\mathbf{X}_{(n)}$, for mode n) is a reshaping of a tensor into a matrix, used to express computations in more familiar matrix notation. In this reshaping, mode-n "fibers" of $\mathscr{X}$ become columns of $\mathbf{X}_{(n)}$ with a specified ordering. Line 5 computes the Gram matrix of the Khatri-Rao product of matrices but uses properties of the Khatri-Rao and Hadamard (denoted by $*$) products to make it computationally more efficient. The normal equations are solved in Line 6 via Cholesky decomposition, for example. Many details of the algorithm are omitted, including initialization strategies, convergence criteria, and normalization of the factor matrix columns.

Algorithm 1 $\{\mathbf{U}^{(n)}\} = \text{CP-ALS}(\mathscr{X}, R)$

1: Initialize $\mathbf{U}^{(n)}$ to be $I_n \times R$ for $2 \le n \le N$, where $\mathscr{X}$ is $I_1 \times \cdots \times I_N$
2: **while** not converged **do**
3: **for** $n = 1$ to N **do**
4: $\mathbf{M}^{(n)} = \mathbf{X}_{(n)}(\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \cdots \odot \mathbf{U}^{(1)})$
5: $\mathbf{S}^{(n)} = {\mathbf{U}^{(N)}}^T\mathbf{U}^{(N)} * \cdots * {\mathbf{U}^{(n+1)}}^T\mathbf{U}^{(n+1)} * {\mathbf{U}^{(n-1)}}^T\mathbf{U}^{(n-1)} * \cdots * {\mathbf{U}^{(1)}}^T\mathbf{U}^{(1)}$
6: Solve $\mathbf{U}^{(n)}\mathbf{S}^{(n)} = \mathbf{M}^{(n)}$ for $\mathbf{U}^{(n)}$
7: **end for**
8: **end while**

The objective function given in Equation (1) is nonlinear and nonconvex. Alg. 1 guarantees monotonic decrease in objective function value, but the computed solution is sensitive to initial values and not guaranteed to be at or near a global minimum. The algorithm is not guaranteed to converge to a stationary point because the set of rank-R tensors is not closed. Further, CP-ALS can suffer from slow convergence and so-called swamps [25], where the algorithm slogs through a plateau in objective function value before converging to a much a better solution. Alternative approaches that avoid some of these problems include using general gradient-based optimization algorithms like nonlinear conjugate gradient or L-BFGS and nonlinear least-squares methods such as Gauss-Newton or Levenberg-Marquardt [33, 1]. Nonetheless, CP-ALS remains popular because of its simplicity and general effectiveness, and considerable effort has been put towards reducing the time per iteration, for example by avoiding redundant computation [27] and efficient parallelization [32].

3.2 Example Application: Fast Matrix Multiplication

The most common applications of CP involve discovering latent low-rank signals in noisy data; in these cases we seek only an approximation of the data. In this section, we describe an application that requires an exact CP decomposition: computationally discovering fast matrix multiplication algorithms. Fast algorithms for matrix multiplication are those that perform fewer than n^3 scalar multiplications in multiplying two square $n \times n$ matrices; Strassen's $O(n^{2.81})$ algorithm is the most well known among such algorithms [34]. By exploiting a connection between bilinear operators (including matrix multiplication) and tensor computations, we can discover a new algorithm by computing an exact CP decomposition of a particular tensor.

Just as every (finite-dimensional) linear operator can be represented by a matrix, multilinear operators can be represented by a tensor. If a matrix is low rank, then its corresponding linear operator can be applied to a vector more cheaply via a low rank decomposition. If a tensor is low rank, then its corresponding multilinear operator can be applied to a set of vectors more cheaply. Because matrix multiplication is a bilinear operator, a low-rank CP decomposition of the 3D tensor that represents matrix multiplication (of matrices of fixed dimension) yields a cheaper algorithm than the dot-product-based classical algorithm [7, 16]. Furthermore, because matrix multiplication can be performed recursively, discovering a better algorithm for matrices of very small dimension leads to a reduction in the *exponent* of the computational complexity of the recursive algorithm applied to matrices of arbitrary dimension.

To be concrete, consider multiplication of 2×2 matrices:

$$\mathbf{A} \cdot \mathbf{B} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \mathbf{C}.$$

If we organize the elements of the three matrices in vector form, we can express matrix multiplication as a tensor operation

$$\mathscr{T} \times_1 \begin{pmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{pmatrix}^{\mathsf{T}} \times_2 \begin{pmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} c_{11} \\ c_{12} \\ c_{21} \\ c_{22} \end{pmatrix}^{\mathsf{T}},$$

where $\mathscr{T}$ is a $4 \times 4 \times 4$ tensor with the following slices:

$$\mathbf{T}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{T}_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{T}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{T}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and $\times_1$ and $\times_2$ denote the tensor-times-vector (TTV) operation in modes 1 and 2, respectively. TTV corresponds to dot products between the vector and all fibers in a given mode of the tensor. Because there are 8 nonzeros in the tensor, there is an obvious rank-8 CP decomposition that corresponds to the 8 scalar multiplications performed by the classical algorithm. Strassen's algorithm corresponds to a rank-7 decomposition of $\mathscr{T}$. That is, there exist rank-one tensors $\mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$ such that

$$\begin{aligned} \mathscr{T} \times_1 \mathbf{a}^{\mathsf{T}} \times_2 \mathbf{b}^{\mathsf{T}} &= \sum_{r=1}^{7} (\mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r) \times_1 \mathbf{a}^{\mathsf{T}} \times_2 \mathbf{b}^{\mathsf{T}} \\ &= \sum_{r=1}^{7} (\mathbf{a}^T \mathbf{u}_r) \cdot (\mathbf{b}^T \mathbf{v}_r)\, \mathbf{w}_r. \end{aligned}$$

When the entries of $\mathbf{a}$ and $\mathbf{b}$ are themselves submatrices (the entries of $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are always scalars), then the most expensive computation in the expression on the right-hand side is the multiplication between $\mathbf{a}^T\mathbf{u}_r$ and $\mathbf{b}^T\mathbf{v}_r$, and there are 7 of these. Applied recursively, Strassen's algorithm yields an exponent of $\log_2 7 \approx 2.81$, where the 7 comes from the rank of the decomposition and the 2 comes from the dimension of the matrix multiplication.

While it is known that no better decomposition exists for 2×2 multiplication [21], there exist many exciting open problems in the application of fast matrix multiplication. Classical 3×3 multiplication requires 27 multiplications, there exist known rank-23 decompositions of the corresponding tensor (yielding an exponent of $\log_3 23 \approx 2.85$) [20], but it is an open question whether there exists an exact decomposition with lower rank. A rank-21 decomposition would yield a better exponent than Strassen's exponent for the 2×2 case, as $\log_3 21 \approx 2.77$. We are also not limited to square matrices; better algorithms for multiplying rectangular matrices will also yield asymptotic improvements. Numerical optimization techniques (along with today's computational power) have proved the most effective in discovering new algorithms [31, 5], but the search goes on, begging for insightful combinatorial and/or numerical optimization approaches.

4 Tucker Decomposition

The Tucker decomposition approximates the input tensor with a set of factor matrices along with a core tensor with as many modes as the input tensor [35]. Like CP, the approximation is a sum of rank-one components, but now every combination of columns from the N factor matrices are included in the approximation and weighted by the corresponding entry in the core tensor $\mathscr{G}$.

The optimization problem is expressed as

$$\underset{\mathscr{G}, \{\mathbf{U}^{(n)}\}}{\arg\min} \left\| \mathscr{X} - \mathscr{G} \times_1 \mathbf{U}^{(1)} \cdots \times_N \mathbf{U}^{(N)} \right\|,$$

where $\times_n$ denotes the nth-mode tensor-times-matrix (TTM) operation. TTM generalizes TTV and is defined to be all-pairs dot-products between the rows of a matrix and the fibers of a tensor, equivalent to matrix multiplication with the right reshaping of the tensor. In element-wise notation, the objective function is

$$\sum_{i_1,\ldots,i_N} \left(\mathcal{X}(i_1,\ldots,i_N) - \sum_{r_1,\ldots,r_N} \mathcal{G}(r_1,\ldots,r_N) \cdot \mathbf{U}^{(1)}(i_1,r_1) \cdots \mathbf{U}^{(N)}(i_N,r_N) \right)^2 .$$

4.1 Algorithms

The Higher-Order Singular Value Decomposition (HOSVD) is a particular Tucker decomposition of a tensor, and the algorithm used for computing it is also referred to as HOSVD [23, 35]. It computes the factor matrices as the leading left singular vectors of each tensor matricization $\mathbf{X}_{(n)}$ and then computes the core tensor $\mathcal{G}$ from those factor matrices. Computing the leading left singular vectors can be done in various ways (e.g., using direct, iterative, or randomized SVD algorithms), but one common approach is to compute the eigenvalue decomposition of the Gram matrix $\mathbf{X}_{(n)}\mathbf{X}_{(n)}^T$, whose eigenvectors correspond to the left singular vectors of $\mathbf{X}_{(n)}$.

The Sequentially Truncated HOSVD (ST-HOSVD), shown as Alg. 2, is a variant that truncates the tensor after each factor matrix is computed, which yields a computational advantage but adds dependencies to the algorithm. The HOSVD and ST-HOSVD (with different mode orders) yield slightly different results, but all are within a factor of $\sqrt{N}$ of the optimal Tucker decomposition for a fixed set of ranks [13]. Iterative algorithms, such as Higher-Order Orthogonal Iteration [23, 19], can improve the objective function value obtained by ST-HOSVD, but the cost per iteration is nearly the same as all of Alg. 2, so these approaches are typically seen as more effort than they are worth.

Algorithm 2 $[\mathcal{G}, \{\mathbf{U}^{(n)}\}] = \text{ST-HOSVD}(\mathcal{X}, \{R_n\})$

1: $\mathcal{Y} = \mathcal{X}$
2: **for** $n = 1$ to N **do**
3: Set $\mathbf{U}^{(n)}$ to be the leading R_n left singular vectors of $\mathbf{Y}_{(n)}$
4: $\mathcal{Y} = \mathcal{Y} \times_n \mathbf{U}^{(n)T}$
5: **end for**
6: $\mathcal{G} = \mathcal{Y}$

An alternative use of these algorithms is to specify a norm-wise error tolerance instead of specifying the dimensions of the core tensor. Using the singular values of $\mathbf{X}_{(n)}$ or $\mathbf{Y}_{(n)}$, the dimension in each mode can be determined to satisfy the error requirement. This guaranteed error is an important advantage of the Tucker decomposition over the CP decomposition.

4.2 Example Application: Numerical Simulation Data Compression

One of the most compelling applications of Tucker decomposition is multidimensional data compression. In this section we'll describe a particular application that involves terabyte-sized multidimensional data sets and the effectiveness of Tucker-based compression. Numerical simulation of scientific phenomena enables approximation of physical experiments that might be too difficult or expensive to conduct, and it also has led to a data deluge, often producing more data than can be effectively analyzed.

For example, direct numerical simulation of a combustion engine involves tracking multiple variable values on discrete grid points in 3D space over time; a numerical value is produced for every variable, x-coordinate, y-coordinate, z-coordinate, and time step. Dimensions of this 5D tensor can be $11 \times 500 \times 500 \times 500 \times 400$, for 11 variables, 500 grid points in each spatial dimension, and 400 time steps [18], which is 4.4 TB (with double-precision values). While such data sets can be produced in reasonable time on supercomputers, the sheer size of the data means that visualization, analysis, and even moving the data to other file systems requires high-end hardware. The Tucker decomposition has been shown to compress this data set by tremendous factors, by 200,000× for a norm-wise relative error of 1e-2 and 400× for a norm-wise relative error of 1e-4 [4].

Computing a Tucker decomposition of a tensor that large is difficult to do without parallel hardware (it takes tens of nodes on most clusters to even load the tensor into memory). However, the compressed version of the data can fit onto a single workstation; for the example given above, the more accurate Tucker representations requires only about 11 GB. The structure of a Tucker representation also allows for efficient processing, as linear operations (such as subtensor selection) on the tensor can be computed without reconstructing the entire data set. For example, an analyst may wish to select a single variable (mode 1) and downsample one of the spatial dimensions (mode 2) at a single time step (mode 5) for the purposes of visualization. This can be accomplished through the following series of TTMs:

$$\mathcal{Z} = \mathcal{G} \times_1 \left(\mathbf{C}^{(1)T}\mathbf{U}^{(1)}\right) \times_2 \left(\mathbf{C}^{(2)T}\mathbf{U}^{(2)}\right) \times_3 \mathbf{U}^{(3)} \times_4 \mathbf{U}^{(4)} \times_5 \left(\mathbf{C}^{(5)T}\mathbf{U}^{(5)}\right),$$

where

$$\mathbf{C}^{(1)} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \mathbf{C}^{(2)} = \begin{bmatrix} .5 & 0 & \cdots & 0 \\ .5 & 0 & \cdots & 0 \\ 0 & .5 & \cdots & 0 \\ 0 & .5 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}, \mathbf{C}^{(5)} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix}.$$

Fig. 3 depicts such a visualization for this particular application: a temperature isosurface in three dimensions, which captures the threshold of a flame at a certain point in time. Here, the original data is shown with data reconstructed from the Tucker representations from the two different error tolerances; there are no visual differences in the reconstructions.

We note that the amount of compression offered by a Tucker decomposition is highly data dependent. The trade-off between accuracy of approximation and compression ratio is governed by the singular values of the matricized tensors within Alg. 2. Just like the truncated SVD for matrices, a quick drop off in the singular values implies great potential for compression.

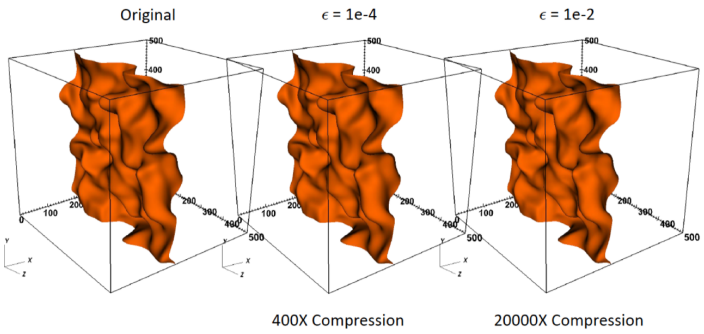


Figure 3: Visualization of a temperature isosurface at one time point, comparing partial reconstruction of compressed Tucker representations with the original data.

5 Conclusion

The jump from 2D to higher dimensional arrays does not seem like a significant generalization. However, because matrix to tensor decomposition corresponds to switching from the world of linear to multilinear (i.e., nonlinear) operations, the mathematical and computational difficulties take a mighty leap. CP-ALS (Alg. 1) is simple and often effective, but it has many shortfalls and can be bested by more sophisticated optimization techniques. For variants of CP and many other tensor decompositions, the optimization community has much to offer the field of tensor computations. The diversity of tensors (in terms of number of modes, dimensions of modes, underlying low-rank structure, sparsity structure, etc.) ensures that there is not a single decomposition or algorithm that is always the best choice. Constraints on factor matrices are often useful or necessary for interpretation of results, and there are many more open questions regarding the most effective constrained optimization methods for those applications.

It is not hard to start toying around with tensors, tensor decompositions, and optimization techniques. While the list of user-friendly software packages is growing, we recommend two MATLAB libraries for getting started: Tensor Toolbox [3] and TensorLab [36]. We also note that development of more efficient libraries, written in lower-level languages, is also underway. For two examples, the SPLATT software package [32] is written for computing CP decompositions of sparse tensors on shared- and distributed-memory parallel machines, and TuckerMPI [4] is a library for computing Tucker decompositions of dense tensors in parallel, which was used for the example given in Section 4.2. We expect more development as the demand for both productivity- and efficiency-oriented tensor software continues to grow in future years.

REFERENCES

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[2] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[3] B. W. Bader, T. G. Kolda, et al. MATLAB Tensor Toolbox version 3.0-dev. Available online, October 2017.

[4] G. Ballard, A. Klinvex, and T. G. Kolda. TuckerMPI: Efficient parallel software for Tucker decompositions of dense tensors. Technical report 1901.06043, arXiv, 2019.

[5] A. R. Benson and G. Ballard. A framework for practical parallel fast matrix multiplication. In *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP 2015, pages 42–53, New York, NY, USA, February 2015. ACM.

[6] M. W. Berry, M. Browne, A. N. Langville, V. Paul Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155 – 173, 2007.

[7] R. P. Brent. Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity. *Numerische Mathematik*, 16(2):145–156, 1970.

[8] R. Bro. PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2):149 – 171, 1997.

[9] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, Sep 1970.

[10] E. C. Chi and T. G. Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[11] P. Comon. Tensors : A brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, May 2014.

[12] N. Gillis. Introduction to nonnegative matrix factorization. *SIAG/OPT Views and News*, 25(1):7–16, 2017.

[13] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*, volume 42 of *Springer Series in Computational Mathematics*. Springer-Verlag Berlin Heidelberg, 2012.

[14] W. Hackbusch. Numerical tensor calculus. *Acta Numerica*, 23:651–742, 2014.

[15] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *Working Papers in Phonetics*, 16(10,085):1 – 84, 1970.

[16] D. E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981.

[17] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[18] H. Kolla, X.-Y. Zhao, J. H. Chen, and N. Swaminathan. Velocity and reactive scalar dissipation spectra in turbulent premixed flames. *Combustion Science and Technology*, 188(9):1424–1439, 2016.

[19] P. M. Kroonenberg and J. De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[20] J. D. Laderman. A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications. *Bulletin of the American Mathematical Society*, 82(1):126–128, 01 1976.

[21] J. M. Landsberg. The border rank of the multiplication of 2×2 matrices is seven. *Journal of the American Mathematical Society*, 19:447–459, 2006.

[22] J. M. Landsberg. *Tensors: Geometry and Applications*. Graduate studies in mathematics. American Mathematical Society, 2012.

[23] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[24] J. Leber. The immortal life of the Enron e-mails. *MIT Technology Review*, Jul 2013.

[25] B. C. Mitchell and D. S. Burdick. Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies. *Journal of Chemometrics*, 8(2):155–168, 1994.

[26] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology*, 8(2):16:1–16:44, October 2016.

[27] A.-H. Phan, P. Tichavsky, and A. Cichocki. Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, Oct 2013.

[28] S. Rabanser, O. Shchur, and S. Günnemann. Introduction to Tensor Decompositions and their Applications in Machine Learning. Technical Report 1711.10781, arXiv, November 2017.

[29] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96 – 192, 2011.

[30] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, July 2017.

[31] A. V. Smirnov. The bilinear complexity and practical algorithms for matrix multiplication. *Computational Mathematics and Mathematical Physics*, 53(12):1781–1795, 2013.

[32] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis. SPLATT: Efficient and parallel sparse tensor-matrix multiplication. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, IPDPS '15, pages 61–70, Washington, DC, 2015.

[33] L. Sorber, M. Van Barel, and L. de Lathauwer. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(Lr,Lr,1) terms, and a new generalization. *SIAM Journal on Optimization*, 23(2):695–720, 2013.

[34] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[35] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[36] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0, Mar. 2016. Available online.

Big Data is Low Rank

Madeleine Udell
Operations Research and Information Engineering
Cornell University udell@cornell.edu
http://www.people.orie.cornell.edu/~mru8

This column defends the assertion that big data is low rank and considers implications for data scientists and opportunities for optimizers.

1 Introduction

Low rank models have demonstrated effective performance in a wide range of data science applications. In this column, we survey a few of the more surprising applications of low rank models in data science, introduce a mathematical explanation for their effectiveness, and survey optimization approaches and challenges in fitting these models.

We'll start by giving a flavor of the challenges in these data science applications. Suppose you have collected information (*features*) on a number of distinct entities (*examples*). Often these examples are people. Features might include

- the opinions of every respondent to a survey,
- the purchase and browsing history for each customer who has visited an e-commerce website,
- the financial history of a credit card applicant, or
- the medical record of every patient at a hospital,

in addition to demographic characteristics.

In other applications, examples might be the following:

- Securities in a financial model. Here, features might include stock performance, accounting metrics, and indicators of environmental stewardship and of sound corporate governance.
- Samples of tumors from different individuals. Here, features might include immunological markers, size, location, vascularization, and indicators of key mutations.
- Geolocations. Here, features might include local demographics or daily weather over the past year.
- Datasets or problem instances. Here, features might include performance of various algorithms and heuristics to fit the dataset or solve the problem.

In the applications above, the features can be numeric, Boolean, ordinal, or categorical. Even among numeric features, the data can be on wildly different scales or follow very different statistical distributions. Some data may be corrupted with gross errors: survey respondents may lie or misunderstand the question, data may be improperly coded, or doctors inputting their patients' data may make a mistake in haste. Medical records contain instances of babies

born weighing hundreds of pounds, and credit card records contain applicants whose credit score was 999 (out of 800).

Furthermore, many entries may be missing: questions skipped on surveys, patients who died or dropped out of a panel study, new questions added several years into the study, medical tests deemed unnecessary, sensors that failed, concentrations below a machine's sensitivity threshold, locations covered by clouds, eyes covered by sunglasses, algorithms that took too long to run. Notice that *whether or not* each entry is missing is always observed. The pattern of missingness may itself be informative about the value of the entry, or it may not be.

Other information is sometimes available. For example, the data may have internal structure (vector, matrix, tensor), Perhaps each observation is associated with some relevant covariates or is known to have been recorded at a particular time. The data may not be available all at the same time, but rather as a stream of observations.

Data analysts may be interested in a variety of related questions. Can we impute missing data and denoise observed data? Which features are correlated? Which examples are similar? How many *effective* features are present, and how many are just noise? How can this dataset best be used to predict some other quantity of interest for each example? If the dataset spans a long time period, have the statistics of the data changed during that period and, if so, when? Can we learn from the dataset without snooping into the future? When the dataset is large, developing efficient algorithms to answer these questions is important.

This column will discuss optimization methods to answer these questions (and more) by identifying low rank structure in the dataset. These techniques have been studied for over a century, and the literature is correspondingly large. We cannot hope to provide a comprehensive survey here. Instead, we present a few surprising applications of these methods, introduce the mathematics of low rank models, discuss optimization methods to fit these models, and examine why these techniques are effective for such a wide variety of problems.

2 Model

Suppose that the data is collected into a table A with m rows (one for each example) and n columns (one for each feature). The value of the jth feature for the ith example is written as A_{ij}. Some entries may also be *missing* or unobserved. Define $\Omega \subseteq [m] \times [n]$ to be the set of observed entries.

Low rank models make one simple—and seemingly strong—assumption about the data. They posit that every example $i = 1, \dots, m$ can be represented by a low-dimensional vector $x_i \in \mathbb{R}^k$ and that every feature $j = 1, \dots, n$ can be represented by a low-dimensional vector $y_j \in \mathbb{R}^k$ so that

$$x_i^T y_j \approx A_{ij} \tag{1}$$

for every observation $(i, j) \in \Omega$. We call these vectors the low-dimensional *representations* of each example and feature.

Notice that the left-hand side of eq. (1) is a number, while the right-hand side can be Boolean, ordinal, or categorical. What can "$\approx$" mean in this case? The solution we adopt here is to choose a loss function ℓ and to define $\approx$ so that

$$\ell(x_i^T y_j, A_{ij}) \text{ is small } \iff x_i^T y_j \approx A_{ij}.$$

We'll discuss a few common loss functions in Section 6; for a more extensive review, see [32] or the software package `LowRankModels.jl`.

Collecting the representations into matrices

$$\begin{bmatrix} \\ X \\ \\ \end{bmatrix} = \begin{bmatrix} —x_1— \\ \vdots \\ —x_m— \end{bmatrix}, \quad \begin{bmatrix} \quad Y \quad \end{bmatrix} = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_n \\ | & & | \end{bmatrix},$$

we see the parameter k controls the rank of the matrix XY.

To fit a low rank model, we seek representations $x_i \in \mathbb{R}^k$ for each example $i = 1, \dots, m$ and $y_j \in \mathbb{R}^k$ for each feature $j = 1, \dots, n$ to minimize the sum of the losses over the observed entries,

$$\sum_{(i,j) \in \Omega} \ell(x_i^T y_j, A_{ij}). \tag{2}$$

Sometimes this loss is minimized together with a regularizer that controls the complexity of the learned representations. We'll discuss algorithms for this problem in Section 5.

Let's first remark on how a low rank model can address the data science challenges posed above:

- To accommodate data tables with heterogeneous entries, use different loss functions for each kind of observation.
- To impute or denoise observations, use the inner product $x_i^T y_j$ to predict the observed value. When the observations come from some restricted domain $\mathcal{A}$, the loss function $\ell : \mathbb{R} \times \mathcal{A} \to \mathbb{R}$ maps the number $x_i^T y_j$ to a prediction $\hat{A}_{ij}$ via $\hat{A}_{ij} = \arg\min_a \ell(x_i^T y_j, a)$. Notice that the domain of ℓ ensures $\hat{A}_{ij} \in \mathcal{A}$.
- To determine which features are correlated or which examples are similar, compare their low-dimensional representations.
- To assess the effective dimension (rank) k, use cross-validation: leave out some of the observations as a validation set, fit a model to the remaining observations for each value of k under consideration, and choose the value of k that minimizes the loss on the validation set.
- When items are not missing at random, treating each observation equally can lead to a biased estimate. Instead, estimate the propensity of observing an entry and weight observations by the inverse propensity score to achieve a consistent estimate [29].
- Low rank models use a flexible optimization formulation that easily accommodates covariates [9, 26, 28]
- It's straightforward to design an optimization procedure to avoid snooping from time series data. Suppose that each example i is observed at time i. Use a

block coordinate descent or block coordinate minimization algorithm. The key to prevent snooping is to order the blocks so that the representation learned for example i never depends on observations from future times $i+1, i+2, \ldots$ For more detail, see the recent review [7].

- The representations x_i of each example $i = 1, \ldots, n$ can be used as features in other learning models. The main advantages are that these representations are lower dimensional, real-valued, and fully observed, in contrast with the original features. It is also possible to learn representations that simultaneously fit the observed features well and perform well for a supervised task; see, for example, [27].

3 Applications

We first review four different applications of low rank models, drawn from the author's research. These applications are meant to give a flavor of the wide variety of problem domains in which low-rank structure appears and to indicate the kinds of challenges these techniques can address.

Medical informatics. Medical treatments succeed when they correctly identify which patient would benefit from a given treatment. In order to learn personalized treatments from observational data, one necessary first step is to identify patients with similar "phenotypes": those with similar symptoms, similar comorbidities, and (we hope) similar responses to each treatment. Identifying clusters of similar patients from medical records is difficult: observations are heterogeneous and may be very sparse. Nevertheless, low rank models have been used successfully to impute missing data and to identify groups of similar patients [30].

Automated machine learning. In automated machine learning (AutoML), the goal is to quickly identify an algorithm (together with its hyperparameters) that will perform well on a new dataset. Yang et al. [35] propose to learn which algorithms will accurately fit a new dataset by using a low rank model. Here, examples are datasets, and each feature is the performance of a particular algorithm. Observations are made by running an algorithm on a dataset. The first (slow) step is to collect observations by running many algorithms on many datasets. Surprisingly, the resulting table of observations has a spectrum that decays rapidly. The second (fast) step determines the best algorithm for a new dataset: Yang et al. [35] suggest running a small number of (fast, informative) algorithms on the new dataset. These observations can be used to impute the performance of all other algorithms and to choose the algorithm(s) with the best predicted performance. The resulting AutoML method is competitive with the state of the art in automated machine learning.

Understanding categorical variables. High-dimensional categorical variables often stymie data analysis: using a standard one-hot encoding inflates the number of variables and can result in overfitting. Low rank models can be used to to embed these high-dimensional categoricals into a low-dimensional vector space. Fu and Udell [10] show how to use this approach to reduce the dimension of the feature "zip code"(with 32,989 nominal values) to a ten-dimensional vector. Substituting the zip code feature by these low-rank representations of the zip code allows for better predictions of labor code violations by businesses in each zip code compared with standard approaches.

Causal inference. To correctly identify the causal effect of a treatment on an outcome from observational data, one must control for possible confounders: other covariates that may influence both the treatment and the outcome. However, controlling for more and more (noisy) covariates increases the variance of the model; worse, some covariates may not be observed for all examples. Instead, Kallus et al. [20] suggest controlling for *latent* confounders by fitting a low rank model to the covariates. The low rank representations of the covariates are identified as the latent confounders. Empirically, controlling for these latent confounders improved the accuracy of *every* causal inference method tested [20].

4 Why Low Rank?

Why do low rank models perform so well in such a wide variety of problems? A data table can be well approximated by a low rank model if there are

- a small number of latent features for each row and
- a small number of latent features for each column
- so that entries of the data table are approximately (functions of) the inner product of the row latent features with the column latent features.

Two elements of this story are surprising. Why should row and column latent features be low dimensional? And why should they interact linearly to form the data?

A more general (and perhaps more plausible) model is to choose some high-dimensional row and column latent features $\alpha_i \in \mathcal{A} \subseteq \mathbb{R}^N$ and $\beta_j \in \mathcal{B} \subseteq \mathbb{R}^N$ (where N may be large), and some arbitrary function $g : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$, and to model entries as $g(\alpha_i, \beta_j)$. We call such a model a latent variable model.

Now we can say why the effectiveness of low rank models should not be a surprise. Under very general conditions on g, $\mathcal{A}$, and $\mathcal{B}$,[1] the matrix with entries $g(\alpha_i, \beta_j)$ is approximately low rank. More concretely, consider the problem of approximating a matrix $A \in \mathbb{R}^{m \times n}$ by a lower-rank matrix X so that the difference between X and A is no greater than ϵ on each entry. How does rank of this ϵ-approximation to A change with m and n? Udell and Townsend [33] show that the optimal value of the problem

$$\begin{array}{ll} \text{minimize} & \text{Rank}(X) \\ \text{subject to} & \|X - A\|_\infty \leq \epsilon \end{array}$$

[1] It suffices for the sets $\mathcal{A}$ and $\mathcal{B}$ to be bounded and for g to be analytic with bounded derivatives. Udell and Townsend [33] show that the same result holds under more general conditions.

grows as $\mathcal{O}(\log(m+n)/\epsilon^2)$. That is, the rank of the matrix A grows much less quickly than its dimension. Hence large enough datasets have low relative rank: big data is low rank.

The idea of the proof is simple. For each α, expand g around $\beta = 0$ by its Taylor series

$$\begin{aligned} & g(\alpha,\beta) - g(\alpha,0) \\ = \; & \langle \nabla g(\alpha,0), \beta\rangle + \langle \nabla^2 g(\alpha,0), \beta\beta^\top\rangle + \dots \\ = \; & \begin{bmatrix} \nabla g(\alpha,0) \\ \mathrm{vec}(\nabla^2 g(\alpha,0)) \\ \vdots \end{bmatrix}^\top \begin{bmatrix} \beta \\ \mathrm{vec}(\beta\beta^\top) \\ \vdots \end{bmatrix}, \end{aligned}$$

where we have collected terms depending on α and on β into two vectors. Notice that we have approximated $g(\alpha,\beta)$ by an inner product. Since g is analytic, we can achieve an approximation with error ϵ by truncating the expansion after $O(\log(1/\epsilon))$ terms.

If α and β are themselves low dimensional (for example, univariate), this immediately gives a low rank factorization of A. Otherwise, apply the Johnson-Lindenstrauss lemma [17] to reduce the dimension of the vectors. Udell and Townsend [33] use a variant of the lemma that bounds the error in the inner product:

Lemma 1 (Variant of the Johnson–Lindenstrauss lemma [33]). Consider $x_1, \dots, x_n \in \mathbb{R}^N$. Pick $0 < \epsilon < 1$ and set $r = \lceil 8(\log n)/\epsilon^2 \rceil$. A linear map $Q : \mathbb{R}^N \to \mathbb{R}^r$ exists such that for all $1 \le i, j \le n$,

$$\left| x_i^T x_j - x_i^T Q^T Q x_j \right| \le \epsilon \left(\|x_i\|^2 + \|x_j\|^2 - x_i^T x_j \right). \qquad (3)$$

The technical conditions on the function g and the sets $\mathcal{A}$ and $\mathcal{B}$ guarantee that the right-hand side of eq. (3) is bounded by a constant, finishing the proof.

5 Fitting Low Rank Models

5.1 PCA

When observations are numeric, it's traditional to measure error with the quadratic loss $\ell(u,a) = (u-a)^2$, which makes the optimization problem eq. (2) particularly easy to solve when every entry is observed. In this case, eq. (2) is known as *principal components analysis* (PCA):

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij} - Z_{ij})^2 = \|A - Z\|_F^2 \\ \text{subject to} & \mathrm{Rank}(Z) \le k. \end{array} \qquad (4)$$

Here, we introduce the variable $Z = XY \in \mathbb{R}^{m\times n}$; representations $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{k\times n}$ can be recovered by using any (rank-revealing) factorization of the matrix Z.

Optimization problems with rank constraints are in general challenging. This problem is the one exception: it can be solved easily by using the singular value decomposition (SVD). Let $\sigma_1, \dots, \sigma_k$ be the first k singular values of A, and let $u_1, \dots, u_k$ and $v_1, \dots, v_k$ be the first k left and right singular vectors, respectively. A solution to eq. (4) is

$$Z^\star = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

which is unique so long as the singular values are. The top few singular values are particularly easy to compute: indeed, in Hotelling's 1933 paper introducing PCA, he computes them by hand using power iteration [15].

5.2 Missing Entries

The solution is not as straightforward when some entries are missing. One simple heuristic is to guess a value for each of the missing entries and then solve the resulting (completely observed) problem with PCA. Often, missing values are replaced by the column mean. This heuristic is easy to understand or to code, but it does not work when a large fraction of the entries are missing.

Two general strategies—convex and nonconvex—are used for fitting low rank models with many missing entries. These methods are often described for quadratic losses, but they work well for any convex differentiable loss.

Convex methods. Convex methods model the product $Z = XY \in \mathbb{R}^{m\times n}$ rather than modeling the low-dimensional representations explicitly, and they do not explicitly constrain the rank of the model. Instead, they introduce a regularizer or constraint on Z—usually involving the nuclear norm $\|Z\|_*$, which is the 1-norm of the singular values of Z—to encourage a low-rank solution. The resulting problem is a large-scale semidefinite program:

$$\text{minimize} \quad \sum_{(i,j)\in\Omega} (A_{ij} - Z_{ij})^2 + \lambda \|Z\|_* \qquad (5)$$

with variable $Z \in \mathbb{R}^n$. Algorithms include interior point methods (for small problems), proximal gradient methods, and conditional gradient methods.

While interior point methods scale poorly, first-order methods are more promising: the gradient of the loss term in eq. (5) is sparse, and so it is easy to manipulate even for very large problems. Proximal gradient methods [24, 23, 4] compute the SVD of an $m \times n$ matrix at each iteration: this step can be prohibitively slow in high dimensions. The conditional gradient method (CGM) for eq. (5) avoids the SVD [16]; instead, it moves in the direction of the top singular vector of the gradient. As an added benefit, the rank of the iterate is bounded by the number of iterations. The storage requirements of CGM can be further reduced by sketching the decision variable, which gives an optimal memory algorithm for eq. (5) [36].

Nonconvex methods. Nonconvex methods search over the matrices X and Y and thereby (implicitly) constrain the rank of the product XY. The resulting problem is nonconvex and may have local minima and other suboptimal stationary points. Algorithms include gradient descent, alternating

minimization, alternating proximal gradient methods, and manifold optimization.

Some variants allow the dimension k to vary in order to avoid or escape bad stationary points [18]. This strategy exploits the connection between the convex and nonconvex problem formulation. Let's describe one such method. When progress of the nonconvex method slows, map the nonconvex iterate $(X, Y) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{k\times n}$ to the matrix $Z = X^T Y \in \mathbb{R}^{m\times n}$. Consider Z as an iterate of CGM, and take one step of the CGM method to form Z'. This step increases the rank of the iterate by at most one: $\mathrm{Rank}(Z') \leq \mathrm{Rank}(Z)+1$. Factor Z' as $Z' = X'Y'$ to obtain a new iterate $(X', Y') \in \mathbb{R}^{m\times k+1} \times \mathbb{R}^{k+1\times n}$ to use in the nonconvex method, and continue.

Methods such as manifold optimization [2] guarantee convergence to a second-order stationary point. Furthermore, all such points are optimal for eq. (5) if $k = O(\sqrt{n})$ [3]. Unfortunately this result is tight: for smaller k, second-order critical points are not generically optimal [34].

Statistical guarantees. When observations are generated from a true low rank matrix via a simple statistical model (e.g., with entrywise Gaussian noise), one can prove that both convex and nonconvex optimization methods recover the true matrix as the number of observations increases, for appropriate choices of the parameters. Examples of this approach include [5, 6, 11, 13, 19, 21, 25]; see [8] for a recent review of nonconvex recovery results. Convex methods and manifold optimization (for large enough k) provide guaranteed solutions for the convex relaxation eq. (5). When the data generating distribution is unknown, however, all methods should be regarded as heuristics for minimizing eq. (2).

6 Loss Functions

Choosing the right loss function can substantively improve the imputation error of the model [1, 30]. In fact, this approach can even result in smaller squared error! The loss function induces a nonlinear mapping from the parameter $z = x_i^T y_j$ to the imputed value $\hat{A}_{ij}$ via $\hat{A}_{ij} = \arg\min_a \ell(z, a)$, so the resulting matrix need not be low rank. Anderson-Bergman et al. [1] show examples of real datasets for which a low rank model of rank k, fit by using a data-driven loss function, induces imputations $\hat{A}$ that improve on the square error of the best rank-k model:

$$\sum_{(i,j)\in\Omega} (\hat{A}_{ij} - A_{ij})^2 \leq \inf_{\mathrm{Rank}(Z)\leq k} \sum_{(i,j)\in\Omega} (Z_{ij} - A_{ij})^2.$$

What loss function to pick? A common choice in the theoretical literature is to consider a parametric noise distribution, often in the exponential family, and choose as a loss function the negative log likelihood of the noise distribution. This approach has the advantage of offering provable guarantees, but it can be difficult to validate the choice of noise model for real data. Instead, a recent suggestion is to learn the noise distribution from the data [1, 14]. Unfortunately, these loss functions are often significantly more challenging to optimize.

We may also pick a loss function by considering our qualitative goals in fitting the model. When the goal is to predict individual entries of a matrix, it's often more natural to measure the mean absolute error of a model, rather than the mean square error, and so we'd measure error in the (entrywise) 1-norm. Or we may want a model that fits *every* entry of the matrix well; in this case, we'd want to minimize the maximum absolute error.

If the data takes values from a discrete set (e.g., $\{0, 1\}$ or $\{1, 2, 3, 4, 5\}$), it's natural to round the entries from our low rank model so that imputed values have the same domain. In this case, we may want to know how often the (rounded) model gets the answer *right* or *wrong* and so measure the *misclassification error*: the number of entries $x_i^T y_j$ for $(i, j) \in \Omega$ that don't round to A_{ij}.

When the loss functions are not differentiable or are not convex, the corresponding methods (using, for example, subgradients in place of gradients) lack guarantees and tend to work much more poorly. When the loss functions are not continuous (like the misclassification error) or are not separable entrywise (like the maximum absolute error), the problem is even more severe. For example, finding a rank-1 matrix that minimizes the maximum absolute error to a set of observations is NP hard [12].

Various heuristics to solve these problems have been proposed [12, 33]. In practice, a common strategy is to replace these error metrics by functions that are easier to optimize: in place of the infinity norm, quadratic loss [33]; in place of the 1-norm, elementwise Huber loss [32]; in place of misclassification loss, hinge loss or ordinal hinge loss [31, 32]. The best model is chosen by fitting the surrogate loss function for a range of different parameters and choosing the one that produces the lowest error with respect to the original loss function. Interestingly, under strong statistical assumptions, even simple algorithms like gradient descent applied to the nonconvex formulation—still with quadratic objective!—can be proven to control entrywise error [22].

7 Conclusion

Low rank models can be used to answer a wide variety of questions in data science. They can adapt to perversities in the data including missing and heterogeneous entries, and they perform well across a diverse array of problems. Numerous optimization methods are available to fit low rank models, some with provable statistical recovery guarantees and others that guarantee convergence to the solution of a particular relaxation, eq. (5). These methods yield useful results in practice. Minimizing the original rank-constrained objective, eq. (2), for general data distributions, is more challenging. Moreover, substantial room exists to improve optimization heuristics for fitting low rank models involving nonsmooth or discontinuous loss functions.

REFERENCES

[1] C. Anderson-Bergman, T. G. Kolda, and K. Kincher-Winoto. XPCA: Extending PCA for a combination of discrete and continuous variables. *ArXiv preprint 1808.07510*, 2018.

[2] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[3] N. Boumal, V. Voroninski, and A. S. Bandeira. Deterministic guarantees for Burer-Monteiro factorizations of smooth semidefinite programs. *ArXiv preprint 1804.02008*, 2018.

[4] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[5] E. J. Candès and Y. Plan. Matrix completion with noise. *CoRR*, abs/0903.3131, 2009.

[6] E.J. Candès and B. Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.

[7] Y. Chi, L. Balzano, and Y. M. Lu. A modern perspective on streaming PCA and subspace tracking: The missing data case. *Proceedings of the IEEE*, 2018.

[8] Y. Chi, Y. M. Lu, and Y. Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *ArXiv preprint 1809.09573*, 2018.

[9] W. Fithian and R. Mazumder. Flexible low-rank statistical modeling with side information. *ArXiv preprint 1308.4211*, 2013.

[10] A. Fu and M. Udell. Census labor law violations. https://github.com/h2oai/h2o-3/blob/master/h2o-r/demos/rdemo.census.labor.violations.large.R. Accessed: 2019-01-13.

[11] R. Ge, J. D. Lee, and T. Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

[12] N. Gillis and Y. Shitov. Low-rank matrix approximation in the infinity norm. *ArXiv preprint 1706.00078*, 2017.

[13] S. Gunasekar, A. Acharya, N. Gaur, and J. Ghosh. Noisy matrix completion using alternating minimization. In *Machine Learning and Knowledge Discovery in Databases*, pages 194–209. Springer, 2013.

[14] F. Han and H. Liu. Semiparametric principal component analysis. In *Advances in Neural Information Processing Systems*, pages 171–179, 2012.

[15] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[16] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435, 2013.

[17] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.

[18] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[19] N. Kallus and M. Udell. Revealed preference at scale: Learning personalized preferences from assortment choices. In *The 2016 ACM Conference on Economics and Computation*, New York, NY, 2016.

[20] N. Kallus, X. Mao, and M. Udell. Causal inference with noisy and missing covariates via matrix factorization. In *Advances in Neural Information Processing Systems*, 2018.

[21] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[22] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution. *ArXiv preprint 1711.10467*, 2017.

[23] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[24] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

[25] S. Negahban and M. Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097, 2011.

[26] M. Paradkar and M. Udell. Graph-regularized generalized low rank models. In *CVPR Workshop on Tensor Methods in Computer Vision*, 2017.

[27] I. Rish, G. Grabarnik, G. Cecchi, F. Pereira, and G. J. Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In *Proceedings of the 25th International Conference on Machine Learning*, pages 832–839. ACM, 2008.

[28] G. Robin, H.-T. Wai, J. Josse, O. Klopp, and É. Moulines. Low-rank interaction with sparse additive effects model for large data frames. In *Advances in Neural Information Processing Systems*, pages 5501–5511, 2018.

[29] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims. Recommendations as treatments: Debiasing learning and evaluation. *ArXiv preprint 1602.05352*, 2016.

[30] A. Schuler, V. Liu, J. Wan, A. Callahan, M. Udell, D. Stark, and N. Shah. Discovering patient phenotypes using generalized low rank models. *Pacific Symposium on Biocomputing (PSB)*, 2016.

[31] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, volume 17, pages 1329–1336, 2004.

[32] M. Udell, C. Horn, R. Zadeh, and S. Boyd. Generalized low rank models. *Foundations and Trends in Machine Learning*, 9(1):1–118, 2016.

[33] M. Udell and A. Townsend. Why are big data matrices approximately low rank? *SIAM Mathematics of Data Science (SIMODS), to appear*, 2018.

[34] I. Waldspurger and A. Waters. Rank optimality for the Burer-Monteiro factorization. *ArXiv preprint 1812.03046*, 2018.

[35] C. Yang, Y. Akimoto, D. W. Kim, and M. Udell. OBOE: Collaborative filtering for automl initialization. In *NeurIPS Workshop on Automated Machine Learning*, 2018.

[36] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Greetings from the NSF

Juan C. Meza
Division of Mathematical Sciences
National Science Foundation
Email: jcmeza@nsf.gov

Let me start by thanking the editors for the opportunity to write some of my thoughts on my new position. As you may have heard, I took on the position of Director for the Division of Mathematical Sciences (DMS) at the National Science Foundation early this spring. Let me also thank all of the people who wrote to me with congratulations and well wishes. I truly appreciated the tremendous level of support I received.

While it's less than a year since I started, it's been a great experience so far. My first goal was to learn as much about the various research programs as I could. I had always been aware of the many wonderful programs that DMS supports, but I was truly impressed by both the depth and breadth of the research portfolio. As I learned early on, DMS funds about 63% of all basic mathematics research in the US and with a budget of $233M in FY17, in turns out that there's a lot of programs to learn about. In fact, DMS supports close to 5000 researchers including over 2200 senior researchers, close to 1900 graduate students, 319 postdocs, and 376 undergraduate students.

The second goal I had in mind was to talk to as many people as I could, to hear what their impressions of DMS were and how DMS might be able to help the math community. Luckily, the timing was good, and I was able to meet with many of the professional societies. Overall, I ended up giving talks to the American Mathematical Society and SIAM Science Policy Committees, the Combined Board of Mathematics Societies, the American Statistics Association Board, and the Joint Policy Board for Mathematics. All of the meetings were informative, and the discussions yielded numerous suggestions that I will be following up on. If you're interested, we have posted all of the talks on the DMS web pages (https://www.nsf.gov/mps/dms/presentations.jsp) – this is a new link that you will find on our main DMS web pages. My plan is to maintain a list of all the public presentations we give so that the community can learn what we're up to.

As I've learned, there are always new programs and initiatives ongoing, but here I'd like to highlight a few that the optimization community might want to hear about.

The first new program is called TRIPODS, which stands for Transdisciplinary Research in Principles of Data Science. This program aims to bring mathematicians, statisticians, and computer scientists to develop the foundations of data science. We anticipate having two phases for this program. I was glad to see that there was already great involvement from the optimization community in the first phase, with awards to Stephen Wright at the University of Wisconsin; John Wright at Columbia; Katya Schienberg at Lehigh, and Dmitriy Drusvyatskiy at the University of Washington, who was featured in the previous issue of Views and News. You can find a full list of the awards at (https://nsf.gov/funding/pgm_summ.jsp?pims_id=505347). A phase II is anticipated in the future (subject to availability of funds) so I hope that this trend continues.

The second new development was the creation of four new centers for mathematics of complex biological systems (https://www.nsf.gov/news/news_summ.jsp?cntn_id=245523&org=DMS&from=news). These centers came about through collaborations between DMS and the Biology Directorate at NSF and in close partnership with the Simons Foundation. The goal of these centers is to combine expertise from mathematicians, molecular cell biologists, and organismal biologists to study how genes translate into all of the diverse phenotypes we see today. Here again, optimization is playing a key role in some of these research projects.

Let me also add a few words on the NSF's 10 Big Ideas. The Big Ideas represent cutting-edge research areas that will drive NSF's long-term research agenda. You can find more information on these at: https://www.nsf.gov/news/special_reports/big_ideas/.

While all of these are in the process of being more clearly defined, I would like to encourage you take a quick look at the web pages because I believe that several of these offer great opportunities, which the mathematical community could contribute to. In particular, Harnessing the Data Revolution, Quantum Leap, and Understanding the Rules of Life, all have interesting and challenging mathematical problems that will need to be solved, and many of these include optimization problems. For example, in the Quantum Leap program, I would imagine that numerical algorithms in general and optimization in particular would need to be revisited in light of a quantum computer. Some early work in these areas is intriguing, but clearly we still have a long way to go. For Understanding the Rules of Life, there are many biological processes that remain unclear as to how they work. For example, how do cells know when to differentiate into one of the many specific types of cells we have in our bodies and how do they know to stop. There are a lot of theories, but not many mathematical models that could be used to predict the observed behavior. There are many more exciting activities going on at NSF, but I'm afraid that I won't be able to do justice to all of it in the space we have here. What I will say is that I hope to be able to communicate often with the math community and I'm always looking to hear from you if you have any good ideas on how DMS can help your community.

Bulletin

Email items to siagoptnews@lists.mcs.anl.gov *for consideration in the bulletin of forthcoming issues.*

1 Event Announcements

1.1 Sixth International Conference on Continuous Optimization (ICCOPT)

ICCOPT 2019 BERLIN
INTERNATIONAL CONFERENCE ON CONTINUOUS OPTIMIZATION

The ICCOPT 2019 will take place on the campus of the Technical University (TU) of Berlin, August 3-8, 2019. The ICCOPT is a flagship conference of the Mathematical Optimization Society (MOS), and preceding editions were organized in Tokyo (2016), Lisbon (2013), Santiago, Chile (2010), Hamilton, Canada (2007), Troy, USA (2004).

ICCOPT 2019 is hosted by the Weierstrass Institute for Applied Analysis and Stochastics (WIAS) Berlin. It will include a Summer School (August 3-4) and a Conference (August 5-8) with a series of plenary and semi-plenary talks, organized and contributed sessions, and poster sessions.

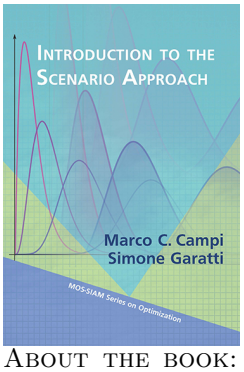
For more details and online registration, please visit our webpage https://iccopt2019.berlin.

Program Committee:
M. Hintermüller, H. Frankowska, D. Sun, J. Gondzio, D. Ralph, C. Sagastizábal, M. Teboulle, S. Ulbrich, S. Wright, T. Luo, J.-S. Pang

Deadlines:
Mar 22, 2019: Abstract submission for talks
Apr 15, 2019: Abstract submission for posters
May 15, 2019: Early bird registration

2 Book Announcements

2.1 Introduction to the Scenario Approach



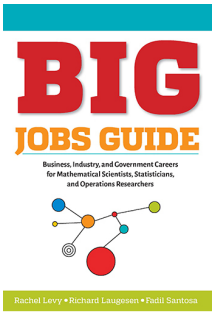
By Marco Campi and Simone Garatti
Publisher: SIAM
ISBN: 978-1-611975-43-7, viii + 114 pages
Published: 2018
http://bookstore.siam.org/mo26/

About the book: This book is about making decisions driven by experience. In this context, a scenario is an observation that comes from the environment, and scenario optimization refers to optimizing decisions over a set of available scenarios. Scenario optimization can be applied across a variety of fields, including machine learning, quantitative finance, control, and identification.

The scenario approach has been given a solid mathematical foundation in recent years, addressing fundamental questions such as: How should experience be incorporated in the decision process to optimize the result? How well will the decision perform in a new case that has not been seen before in the scenario sample? And how robust will results be when using this approach?

This concise, practical book provides readers with an easy access point to make the scenario approach understandable to nonexperts, and offers an overview of various decision frameworks in which the method can be used. It contains numerous examples and diverse applications from a broad range of domains, including systems theory, control, biomedical engineering, economics, and finance. Practitioners can find "easy-to-use recipes," while theoreticians will benefit from a rigorous treatment of the theoretical foundations of the method, making it an excellent starting point for scientists interested in doing research in this field.
Audience: Introduction to the Scenario Approach will appeal to scientists working in optimization, practitioners working in myriad fields involving decision-making, and anyone interested in data-driven decision-making.

2.2 BIG Jobs Guide: Business, Industry, and Government Careers for Mathematical Scientists, Statisticians, and Operations Researchers



By Rachel Levy, Richard Laugesen, and Fadil Santosa
Publisher: SIAM
ISBN: 978-1-611975-28-4, xii + 141 pages
Published: 2018
http://bookstore.siam.org/ot158/

About the book: Jobs using mathematics, statistics, and operations research are projected to grow by almost 30% over the next decade. BIG Jobs Guide helps job seekers at every stage of their careers in these fields explore opportunities in business, industry, and government (BIG).

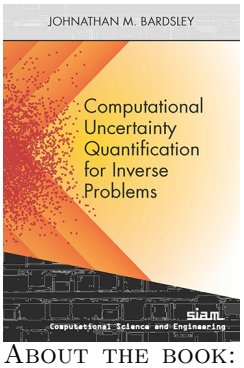
Written in a conversational and practical tone, BIG Jobs Guide offers insight on topics such as:

- What skills can I offer employers?
- How do I write a high-impact résumé?
- Where can I find a rewarding internship?
- What kinds of jobs are out there for me?

The Guide also offers insights to advisors and mentors on topics such as how departments can help students get BIG jobs and how faculty members and internship mentors can build institutional relationships.

AUDIENCE: Whether you're an undergraduate or graduate student or a job seeker in mathematics, statistics, or operations research, this hands-on book will help you reach your goal—landing an internship, getting your first job or transitioning to a new one.

2.3 Computational Uncertainty Quantification for Inverse Problems



By Johnathan M. Bardsley
Publisher: SIAM
ISBN: 978-1-611975-37-6, viii + 135 pages
Published: 2018
http://bookstore.siam.org/cs19/

ABOUT THE BOOK: This book is an introduction to both computational inverse problems and uncertainty quantification (UQ) for inverse problems. The book also presents more advanced material on Bayesian methods and UQ, including Markov chain Monte Carlo sampling methods for UQ in inverse problems. Each chapter contains MATLAB code that implements the algorithms and generates the figures, as well as a large number of exercises accessible to both graduate students and researchers.

AUDIENCE: *Computational Uncertainty Quantification for Inverse Problems* is intended for graduate students, researchers, and applied scientists. It is appropriate for courses on computational inverse problems, Bayesian methods for inverse problems, and UQ methods for inverse problems.

3 Other Announcements

3.1 Announcing the SIAG/OPT Early Career Prize

The SIAM Activity Group on Optimization Early Career Prize (SIAG/OPT Early Career Prize) will be awarded for the first time at the 2020 SIAM Conference on Optimization (OP20), May 26-29, 2020 at Hong Kong Polytechnic University. The prize will be awarded every three years to an outstanding early career researcher in the field of optimization for distinguished contributions to the field in the six calendar years prior to the award year.

Start thinking about your nomination now! The SIAG/OPT Early Career Prize will be open for nominations May 1, 2019 – October 15, 2019.

Eligibility Criteria: The award will recognize an individual who has made outstanding, influential, and potentially long-lasting contributions to the field of optimization within six years of receiving the PhD or equivalent degree as of January 1 of the award year. The committee may consider exceptions to the six–year rule for career interruptions or delays occurring, e.g., for child bearing, child rearing, or elder care. The award can be received only once in a lifetime.

The contribution(s) for which the award is given must be publicly available and may belong to any aspect of optimization in its broadest sense. The contribution(s) may include paper(s) published in English in peer-reviewed journal(s) or conference proceedings, or high quality freely available open source software. The committee expects evidence of independent contribution by the candidate.

For the 2020 award, the candidate must have received their PhD no earlier than January 1, 2014.

Required Materials for a Nomination:

- Nominator's letter of recommendation for candidate
- Candidate's CV
- Two or three letters of support from experts in the field

Learn more about our prize program and view all prizes with open calls. Contact prizeadmin@siam.org with questions regarding nomination procedure.

3.2 2020 SIAG/OPT Best Paper Prize

The SIAM Activity Group on Optimization Best Paper Prize (SIAG/OPT Best Paper Prize) is awarded every three years to the author(s) of the most outstanding paper, as determined by the prize committee, on a topic in optimization published in the four calendar years preceding the award year.

The prize will next be awarded in 2020 at the SIAM Conference on Optimization.

Eligibility Criteria: The qualifying paper must contain significant research contributions to the field of optimization, as commonly defined in the mathematical literature, with direct or potential applications. The paper must have been published in English in a peer-reviewed journal within the four calendar years preceding the award year.

For the 2020 award, the paper must have been published between the dates of January 1, 2016 – December 31, 2019.

The SIAG/OPT Best Paper Prize will be open for nominations May 1, 2019.

For additional details, please visit the SIAG/OPT Best Paper Prize page.

3.3 MPC "Best Paper of the Year" 2018

Mathematical Programming Computation (MPC) announces its "Best Paper of the Year" and an "Honorable Mention' both for 2018. The selections were made from among all papers that appeared in print in MPC in 2018.

The winner is: "Parallelizing the dual revised simplex method," by Qi Huangfu and Julian A.J. Hall, March 2018, Volume 10, Issue 1, pp 119–142.

Honorable mention: "Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods," by Pierre Bonami, Oktay Günlük, and Jeff Linderoth, September 2018, Volume 10, Issue 3, pp 333–382.

Congratulations to these authors!

Chair's Column

Welcome to the January 2019 edition of our SIAG/OPT's Views and News. This short note provides an update about the most important developments since our last newsletter, and drafts our emerging to do list for the new year.

As discussed in the last Chair's Column, the SIAG/OPT had a strong representation at the SIAM Annual Meeting in Portland, Oregon. Thanks and appreciation goes to our program director, Michael Friedlander, for organizing a series of featured minisymposia; our members Rekha R. Thomas (U. Washington), Charles F. Van Loan, (Cornell U.), and Éva Tardos (Cornell U.) delivered plenary lectures; and four of our members, Helen Moore, Pablo Parrilo, Tamás Terlaky, and Kim-Chuan Toh, were inducted in the 2018 class of SIAM Fellows.

During the summer and early fall of 2018, we finalized the definition, eligibility, and selection rules of the SIAM Optimization Early Career Prize. We have the approval of SIAM's Major Award Committee, and you may have seen the initial announcement (if not, see this issue's bulletin). We certainly will present the inaugural Early Career Prize in Hong Kong at Opt'20. Consequently, SIAG/OPT has now two major awards to recognize the exceptional research our members conduct.

In 2019 we accelerate preparations for our signature conference SIAM/Opt'20. The organizing committee started its work to select the plenary and tutorial speakers. Later in the year, the call for minisymposium proposals, abstract submissions, and call for participation will follow. Our Vice Chair, Andreas Waechter (Northwestern U.) will be responsible for forming the prize committees. We will need to prepare for electing the next leadership team. Please let Andreas and me know if you have suggestions for prize committee members, as well as candidates for any of the leadership positions.

This newsletter is the last one that our Views and News Editor Stefan Wild (Argonne National Lab.) assembles together with Jennifer Erway (Wake Forest U.). After 5 years of dedicated service, Stefan is taking on other responsibilities. Our thanks go out to both Stefan and Jennifer for producing highly informative, excellent newsletters year after year. They did an outstanding job. Jennifer is continuing as Editor, and I am pleased to announce that Pietro Belotti (FICO-EXPRESS) is appointed as co-editor of Views and News. I have no any doubt that Jennifer and Pietro will create highly informative editions of SIAG/Opt Views and News in the coming years. Please join me to welcome Pietro in this important function in our SIAG/Opt!

Finally, I'd like to remind all of you that the time arrived to renew your SIAM, and SIAG/Opt, membership to ensure your continued membership in our great community. With this in mind, I wish all of you a prosperous new year! Looking forward to have another great year for Optimization!

Tamás Terlaky, SIAG/OPT Chair
Department of Industrial and Systems Engineering, P.C. Rossin College of Engineering and Applied Science, Lehigh University, Bethlehem, PA 18015-1582, USA,
terlaky@lehigh.edu, http://www.lehigh.edu/~tat208

Comments from the Editors

Our activity group likely needs no convincing of the vital role that optimization plays in data science. This issue features three articles on connections between big data and optimization. In case a trigger warning is needed for the usage of the term "big data," to our knowledge the sole previous occurrence in *Views and News* was in Katya Scheinberg's forward-looking issue 24-1 article. This is a good time to remind readers that all 27 volumes of *Views and News* are available at the online archive.

Grey Ballard's article on tensors highlights the role of and challenges associated with multidimensional arrays beyond the 2D matrices that our community is comfortable with. Approximations built on these techniques have both uses in optimization and opportunities for research in combinatorial and continuous optimization.

Madeleine Udell's article illustrates the breadth of applications that involve finding a low-rank representation from large-scale data. A variety of models and optimization methods can be employed to address such problems.

Juan Meza, SIAG/OPT's previous chair, returns to *Views and News* with a perspective from a key sponsor of mathematical research (including recent awards in data science) in the United States.

After seven *Views and News* issues together, we are excited to welcome Pietro Belotti as a new editor. Pietro is an optimization developer at FICO in Birmingham, UK; many of you likely know his work on mixed-integer (nonlinear) optimization problems. Pietro makes a fine addition to the number of continents and backgrounds represented by the *Views and News* editors.

Starting with the next issue, Pietro will be taking over for Stefan, who has recently begun a term as program director for the SIAG on computational science and engineering. Here's a pitch to follow up SIAM Opt'20 with SIAM CSE'21; optimization and SIAG/OPT members have had an increasing footprint at the CSE meeting, and there are 22 sessions at CSE'19 with optimization in the title.

As always, the editors welcome your feedback at siagoptnews@lists.mcs.anl.gov. Suggestions for new issues, comments, and papers are always welcome!

Stefan M. Wild, Editor
Mathematics & Computer Science Div., Argonne National Lab, USA, wild@anl.gov, http://www.mcs.anl.gov/~wild
Jennifer Erway, Editor
Department of Mathematics, Wake Forest University, USA, erwayjb@wfu.edu, http://www.wfu.edu/~erwayjb