ALGEBRAIC AND OPTIMIZATION BASED ALGORITHMS FOR MULTIVARIATE REGRESSION USING SYMMETRIC TENSOR DECOMPOSITION

Stijn Hendrikx[*†], *Martijn Bousse*[*], *Nico Vervliet*[*], *Lieven De Lathauwer*[*†]

[*]Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium.

[†]Group Science, Engineering and Technology, KU Leuven Kulak, E. Sabbelaan 53, B-8500 Kortrijk, Belgium.

StijnHendrikx1@gmail.com, {Martijn.Bousse, Nico.Vervliet, Lieven.DeLathauwer}@kuleuven.be

ABSTRACT

Multivariate regression is an important task in domains such as machine learning and statistics. We cast this regression problem as a linear system with a solution that is a vectorized symmetric low-rank tensor. We show that the structure of the data and the decomposition can be exploited to obtain efficient optimization methods. Furthermore, we show that an algebraic algorithm can be derived even when the number of given data points is low. We illustrate the performance of our regression model using a real-life dataset from materials sciences.

Index Terms— tensor, regression, polynomial, polyadic decomposition

1. INTRODUCTION

Nowadays large amounts of data are available in various domains, resulting in an increased interest in machine learning methods to extract useful information from this data. Data often also has a higher-order structure, which can be represented using higher-order tensors in a straightforward way. Tensor decompositions are important tools in machine learning and signal processing for extracting hidden components and for compact representations [1]–[3]. In this paper, we combine the advantages of both tensors and machine learning methods by applying tensor mathematics to such a method. This combination has been made before in the literature [4], [5].

The model in this paper is a polynomial that is compactly represented by a polyadic decomposition. Other polynomial models that are represented by a tensor decomposition are exponential machines [6], tensor train polynomial classifiers [7], higher-order factorization machines [8], [9] and tensor machines [10]. The optimization problem of our model is formulated as an LS-CPD [11], a linear system with a polyadic decomposition constrained solution, where the Khatri–Rao structure of the coefficient matrix is exploited to efficiently optimize the variables of the model. We discuss an algebraic and an optimization-based algorithm to solve this problem and perform a computational complexity analysis of the optimization-based method. We test our model on a dataset from materials sciences to show that few data points are required to achieve high accuracy.

In the remainder of this section we discuss notations and basic definitions, the canonical polyadic decomposition and the connection between multivariate polynomials and symmetric tensors. In Section 2 we explain how a polyadic decomposition is used to obtain a compact polynomial model. In Section 3 we discuss the Gauss–Newton algorithm and in Section 4 the algebraic algorithm that can be used to optimize the variables of the model. In Section 5 we test the model and we conclude this paper in Section 6.

1.1. Notations and basic definitions

A tensor is a higher-order generalization of a vector (first-order) and a matrix (second-order). We denote vectors, matrices, and tensors by bold lowercase ($\mathbf{a}$), bold uppercase ($\mathbf{A}$), and calligraphic letters ($\mathcal{A}$), respectively. An Nth-order symmetric tensor is invariant to all permutations of its N indices. The vectorization $\text{vec}(\cdot)$ of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ maps its entries to a vector $\mathbf{a} \in \mathbb{R}^{I_1 \cdots I_N}$. The Kronecker, Hadamard, column- and row-wise Khatri–Rao, outer, mode-n tensor-matrix product and element-wise Nth power are denoted by $\otimes$, $*$, $\odot$, $\odot^{\mathsf{T}}$, $\circledast$, $\bullet_n$, and $\cdot^{\circledast N}$, respectively. A sequence of elements is denoted with a superscript between brackets, e.g., $\{b^{(n)}\}_{n=1}^{N}$. A rank-1 tensor of order N is defined as the outer product of N nonzero vectors. The rank of a tensor is equal to the minimal number of rank-1 terms that produce the tensor as their sum.

1.2. Canonical Polyadic Decomposition (CPD)

The polyadic decomposition (PD) is a powerful tool for signal processing and machine learning [1], [3], [5] with rather mild uniqueness conditions [12], [13]. The PD decomposes a

Research supported by: (1) Flemish Government: This work was supported by the Fonds de la Recherche Scientifique — FNRS and the Fonds Wetenschappelijk Onderzoek — Vlaanderen under EOS Project no 30468160 (SeLMA); (2) KU Leuven Internal Funds C16/15/059; (3) Internal Funds KU Leuven PDM/18/146.

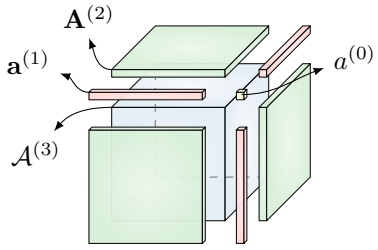


Fig. 1. By applying a process called homogenization, symmetric tensors can also represent the coefficients of non–homogeneous polynomials. For example, by stacking the coefficients $a^{(0)}, \mathbf{a}^{(1)}, \mathbf{A}^{(2)}$ and $\mathcal{A}^{(3)}$ of a degree-3 polynomial into a tensor as shown above, we can represent it with a symmetric third-order tensor. Image reproduced from [14].

tensor into a sum of R rank-1 tensors and is called canonical (CPD) if R equals the rank of the tensor. Hence, we have

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{b}_r^{(1)} \otimes \cdots \otimes \mathbf{b}_r^{(N)} \stackrel{\text{def}}{=} \left[\!\!\left[\mathbf{B}^{(1)}, \cdots, \mathbf{B}^{(N)} \right]\!\!\right], \quad (1)$$

with the columns of factor matrix $\mathbf{B}^{(n)} \in \mathbb{R}^{I \times R}$ for $n = 1, \ldots, N$ equal to the factor vectors $\mathbf{b}_1^{(n)}, \ldots, \mathbf{b}_R^{(n)}$. In this paper, we focus on the symmetric PD, i.e., $\mathbf{B}^{(n)} = \mathbf{B}$ for $n = 1, \ldots, N$.

1.3. Polynomials and symmetric tensors

A multivariate polynomial can be represented by a tensor. For example, consider a *homogeneous* polynomial f of degree $N = 2$ with variables $\mathbf{x} \in \mathbb{R}^I$. The polynomial can then be written using a *symmetric* matrix $\mathbf{T} \in \mathbb{R}^{I \times I}$ as follows[1]:

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} \mathbf{T} \mathbf{x} = \sum_{i=1}^{I} \left(t_{ii} x_i + \sum_{j=1}^{I} t_{ij} x_i x_j \right). \quad (2)$$

Note that Eq. (2) is equivalent with $f(\mathbf{x}) = \mathbf{T} \cdot_1 \mathbf{x}^{\mathrm{T}} \cdot_2 \mathbf{x}^{\mathrm{T}}$, allowing us to generalize in a straightforward way to homogeneous polynomials of degree N using the mode-n product and a tensor $\mathcal{T} \in \mathbb{R}^{I \times \cdots \times I}$ of order N:

$$f(\mathbf{x}) = \mathcal{T} \cdot_1 \mathbf{x}^{\mathrm{T}} \cdot_2 \mathbf{x}^{\mathrm{T}} \cdots \cdot_N \mathbf{x}^{\mathrm{T}}. \quad (3)$$

We can also represent non–homogeneous polynomials by a symmetric tensor by means of homogenization and extending the variables $\mathbf{x}$ to $\mathbf{v} = \left[1, \mathbf{x}^{\mathrm{T}}\right]^{\mathrm{T}}$; see [14].

[1]Note that only the symmetric part of $\mathbf{T}$ is relevant anyway in order to represent the polynomial f as can be seen by writing $\mathbf{T}$ as the sum of its symmetric and asymmetric part.

2. MULTIVARIATE REGRESSION USING PD

Assuming we have M data points $y_m = f(\mathbf{x}_m)$, the model in (3) can be reformulated as the following linear system:

$$\mathbf{y} = \mathbf{A}\mathrm{vec}\left(\mathcal{T}\right) \quad \text{with} \quad \mathbf{A} = \underbrace{\left(\mathbf{X} \odot^{\mathrm{T}} \mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X}\right)}_{N}, \quad (4)$$

in which $\mathbf{y} \in \mathbb{R}^M$ and $\mathbf{X} \in \mathbb{R}^{M \times I}$ contain the data points and the features $\mathbf{x}_m$ as its rows, respectively. In this paper, we assume that the coefficients of the underlying polynomial can be represented by a symmetric tensor that admits a low-rank (symmetric) PD:

$$\mathbf{y} = \left(\mathbf{X} \odot^{\mathrm{T}} \mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X}\right) \mathrm{vec}\left(\left[\!\!\left[\mathbf{B}, \mathbf{B}, \cdots, \mathbf{B}, \mathbf{c}^{\mathrm{T}} \right]\!\!\right]\right). \quad (5)$$

Additional weights $\mathbf{c} \in \mathbb{R}^R$ are included for each rank-1 term, allowing negative coefficients for even-degree polynomials. Eq. (5) is a linear system with a symmetric PD constrained solution *and* a (row-wise) Khatri–Rao structured coefficient matrix $\mathbf{A}$, allowing us to use the LS-CPD framework [11]. While a multivariate polynomial has $\binom{I+N-1}{N} = \frac{(I+N-1)!}{N!(I-1)!}$ coefficients, which equals the number of unique tensor entries, we need only $R(I+1)$ variables thanks to the compact representation by a symmetric PD, allowing us to efficiently fit a polynomial.

3. GAUSS–NEWTON ALGORITHM

Given data $\mathbf{X}$ and $\mathbf{y}$, a multivariate polynomial approximating this data is found by solving the optimization problem

$$\min_{\mathbf{B},\mathbf{c}} f = \frac{1}{2} ||\mathbf{r}||_{\mathrm{F}}^2 \quad (6)$$

in which the residuals $\mathbf{r}$ are given by

$$\mathbf{r} = \left(\mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X}\right) \mathrm{vec}\left(\left[\!\!\left[\mathbf{B}, \cdots, \mathbf{B}, \mathbf{c}^{\mathrm{T}} \right]\!\!\right]\right) - \mathbf{y}. \quad (7)$$

We solve (6) for $\mathbf{B}$ and $\mathbf{c}$ via a standard GN algorithm with dogleg trust region. While this problem can be written and solved as an LS-CPD (Eq. (5)), the key to deriving an efficient algorithm is to exploit the Khatri–Rao and multilinear structure when computing the objective function, gradient, and Gramian. We focus on a GN algorithm, but these expressions can also be used for other quasi–Newton and nonlinear least–squares algorithms.

The GN algorithm solves (6) by linearizing the residuals in (7) in each iteration k and then solving the resulting linear system [15]. The solution to the latter problem is given by the linear system $\mathbf{H}_k \mathbf{p}_k = -\mathbf{g}_k$ with $\mathbf{H}_k$ the Gramian of the Jacobian $\mathbf{J}_k^{\mathrm{T}} \mathbf{J}_k$, GN step $\mathbf{p}_k$ with $\mathbf{z}_k = \left[\mathrm{vec}(\mathbf{B})^{\mathrm{T}}, \mathbf{c}^{\mathrm{T}}\right]^{\mathrm{T}}$, and the gradient $\mathbf{g}_k = (\partial f / \partial \mathbf{z}_k)^{\mathrm{T}}$ [15]. This system can be solved exactly using the pseudoinverse of $\mathbf{H}_k$ or inexactly using conjugated gradients. The variables can then be updated using a dogleg trust region using $\mathbf{p}_k$ and $\mathbf{g}_k$, or using line search

[15]. While a random factor matrix $\mathbf{B}$ and coefficients $\mathbf{c}$ can be used to initialize the algorithm, starting from an algebraic solution often provides a better initialization; see Section 4.

In the remainder of this section, we derive the efficient expressions required for the algorithm as well as their complexity. The complete algorithm and its computational complexity are shown in Algorithm 1 and Table 1, respectively.

Algorithm 1: Symmetric PD-constrained multivariate regression using GN and dogleg trust region.

Input: $\mathbf{X}$, $\mathbf{y}$, and initial values for $\mathbf{B}$ and $\mathbf{c}$
Output: $\mathbf{B}$ and $\mathbf{c}$
while *not converged* **do**
  Compute Jacobian $\mathbf{J}$ and gradient $\mathbf{g}$ using (10)-(11) and (15), respectively.
  Solve $\mathbf{J}^{\mathsf{T}}\mathbf{J}\mathbf{p} = -\mathbf{g}$ for $\mathbf{p}$.
  Update $\mathbf{B}$ and $\mathbf{c}$, using dogleg trust region from $\mathbf{p}$, $\mathbf{g}$, and objective function evaluation via (9).

3.1. Expressions

Objective function By taking the sum of squared entries of the residuals $\mathbf{r} \in \mathbb{R}^M$, we obtain the objective function value. The residuals can be computed more efficiently by using multilinear identities:

$$\mathbf{r} = \left(\mathbf{X} \odot^{\mathsf{T}} \cdots \odot^{\mathsf{T}} \mathbf{X}\right)\left(\mathbf{B} \odot \cdots \odot \mathbf{B}\right)\mathbf{c} - \mathbf{y} \tag{8}$$

$$= \left(\overset{D}{\underset{d=1}{*}} \mathbf{X}\mathbf{B}\right)\mathbf{c} - \mathbf{y} = \left(\mathbf{X}\mathbf{B}\right)^{\cdot D}\mathbf{c} - \mathbf{y}. \tag{9}$$

Jacobian The Jacobian $\mathbf{J} \in \mathbb{R}^{M \times R(I+1)}$ can be partitioned into two parts $\left[\mathbf{J}_{\mathbf{B}}, \mathbf{J}_{\mathbf{c}}\right]$ corresponding to the variables $\mathbf{B}$ and $\mathbf{c}$ with:

$$\mathbf{J}_{\mathbf{B}} = \frac{\partial \mathbf{r}}{\partial \mathrm{vec}(\mathbf{B})} = \left(N(\mathbf{X}\mathbf{B})^{\cdot N-1} \odot \mathbf{c}^{\mathsf{T}}\right) \odot^{\mathsf{T}} \mathbf{X}, \tag{10}$$

$$\mathbf{J}_{\mathbf{c}} = \frac{\partial \mathbf{r}}{\partial \mathbf{c}} = (\mathbf{X}\mathbf{B})^{\cdot N}. \tag{11}$$

We can obtain (10) by computing the derivative row-wise:

$$(\mathbf{J}_{\mathbf{B}})_m = \frac{\partial r_m}{\mathrm{vec}(\mathbf{B})} \tag{12}$$

$$= \left(\frac{\partial \left(\mathbf{x}_m^{\mathsf{T}}\mathbf{B}\right)^{\cdot N}}{\partial \mathrm{vec}(\mathbf{B})} * \mathbf{c}^{\mathsf{T}}\right)\frac{\partial \left(\mathbf{x}_m^{\mathsf{T}}\mathbf{B}\right)}{\partial \mathrm{vec}(\mathbf{B})} \tag{13}$$

$$= \left(N\left(\mathbf{x}_m^{\mathsf{T}}\mathbf{B}\right)^{\cdot N-1} * \mathbf{c}^{\mathsf{T}}\right)\left(\mathbf{I} \otimes \mathbf{x}_m^{\mathsf{T}}\right) \tag{14}$$

using the chain rule and a multilinear identity. By applying (14) for every row, we obtain (10).

Gradient and Gramian The gradient $\mathbf{g}$ and the Gramian $\mathbf{H}$ are given by:

$$\mathbf{g} = \mathbf{J}^{\mathsf{T}}\mathbf{r} \in \mathbb{R}^{R(I+1)} \tag{15}$$

$$\mathbf{H} = \mathbf{J}^{\mathsf{T}}\mathbf{J} \in \mathbb{R}^{R(I+1) \times R(I+1)}. \tag{16}$$

Table 1. By carefully exploiting the Khatri–Rao structure and the symmetry in (5), we can obtain a significant improvement in the computational complexity.

		Complexity	
	Calls/iteration	Our algorithm	LS-CPD [11]
f	$\mathrm{it}_{\mathrm{TR}}$	$\mathcal{O}(MR(I+N))$	$\mathcal{O}(MRI^N)$
$\mathbf{J}$	1	$\mathcal{O}(MR(I+N))$	$\mathcal{O}\left(MRNI^N\right)$
$\mathbf{g}$	1	$\mathcal{O}(MRI)$	$\mathcal{O}\left(MRIN\right)$
$\mathbf{J}^{\mathsf{T}}\mathbf{J}$	1	$\mathcal{O}(MR^2I^2)$	$\mathcal{O}\left(MRI^2N\right)$

3.2. Computational complexity

If the system $\mathbf{H}_k\mathbf{p}_k = -\mathbf{g}_k$ is solved using the pseudoinverse of the Gramian, the GN algorithm has a computational complexity of $\mathcal{O}\left(\mathrm{it}_{\mathrm{GN}}(MR^2I^2 + R^3I^3)\right)$ where $\mathrm{it}_{\mathrm{GN}}$ is the number of GN iterations. In Table 1, we compare the computational complexity of the expressions of our dedicated algorithm with the expressions of the LS–CPD algorithm of [11] that ignores the Khatri–Rao structure. The number of trust region (TR) iterations is denoted by $\mathrm{it}_{\mathrm{TR}}$, which is typically 1. Exploiting the structure clearly reduces the complexity; see Table 1.

4. ALGEBRAIC ALGORITHM

If $\mathbf{A}$ as full column rank, a naive approach for finding a low-rank solution $\mathbf{t} = \mathrm{vec}\left(\mathcal{T}\right)$ from $\mathbf{A}\mathbf{t} = \mathbf{y}$, is to first solve for $\mathbf{x}$, then reshape $\mathbf{t}$ into $\mathcal{T}$ and finally decompose $\mathcal{T}$ using standard algorithms for computing a CPD [11]. Because of the transposed Kathri–Rao products of $\mathbf{X}$ (Eq. (4)), $\mathbf{A}$ has repeated columns and does not have full column rank: $\mathrm{rank}(\mathbf{A}) \leq F < I^N$, in which $F = \binom{I+N-1}{N}$ which coincides with the number of unique coefficients. However, by restricting the solution $\mathcal{T}$ to the set of symmetric tensors, we show that $\mathcal{T}$ can be found if $\mathrm{rank}(\mathbf{A}) = F$ and therefore M can be as low as F.

Let $\mathbf{U} \in \mathbb{R}^{M \times F}$ be the matrix containing the unique columns of $\mathbf{A} \in \mathbb{R}^{M \times I^N}$ and $\mathbf{P} \in \{0,1\}^{F \times I^N}$ the binary matrix placing the columns of $\mathbf{U}$ at the right position, then

$$\mathbf{A} = \mathbf{U}\mathbf{P}. \tag{17}$$

More formally, let $\pi(i_1, i_2, \ldots, i_N)$ be the set of permutations of elements of tuple $(i_1, i_2, \ldots, i_N)$. The column-wise, resp., element-wise definitions of $\mathbf{U}$ and $\mathbf{P}$ are given by

$$\mathbf{u}_j = \mathbf{x}_{j_1}\mathbf{x}_{j_2}\cdots\mathbf{x}_{j_N} \tag{18}$$

$$p_{ji} = \begin{cases} 1 & \text{if } (i_1, i_2, \ldots, i_N) \in \pi(j_1, j_2, \ldots, j_N) \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

where j is the index corresponding to entry $(j_1, j_2, \ldots, j_N)$ in the colexicographically ordered set $\{(j_1, j_2, \ldots, j_N) \,|\, 1 \leq j_1 \leq j_2 \leq \cdots \leq j_N \leq I\}$ and $i = \sum_{n=1}^{N}(i_n - 1)I^{(n-1)} + 1$.

The solution $\operatorname{vec}(\mathcal{T})$ can be computed using the pseudoinverse, denoted by $\cdot^\dagger$, as

$$\operatorname{vec}(\mathcal{T}) = \mathbf{A}^\dagger \mathbf{y} = \mathbf{P}^\intercal \mathbf{U}^\dagger \mathbf{y} = \mathbf{P}^\intercal (\mathbf{U}^\intercal \mathbf{U})^{-1} \mathbf{U}^\intercal \mathbf{y}. \quad (20)$$

$\mathcal{T}$ is unique if $\mathbf{U}^\intercal\mathbf{U}$ has full rank, hence $M \geq F$. Using the definition of symmetric tensors, it is easy to show that range$(\mathbf{P}^\intercal)$ is the set of (vectorized) symmetric tensors. Therefore, $\mathcal{T}$ is a symmetric tensor and its decomposition can be recovered using standard CPD algorithms. (Note that uniqueness of $\mathcal{T}$ does not imply uniqueness of its decomposition.)

5. EXPERIMENTS

We illustrate that few samples are required for multivariate regression using symmetric tensor decompositions on a materials sciences dataset. Both algorithms are implemented in Matlab 2016b using Tensorlab 3.0 [16].

The Gibbs free energy of the liquid phase of an Ag-Cu-Ni-Sn alloy can be modeled by

$$G(x_1, x_2, x_3) = G_\mathrm{p}(x_1, x_2, x_3) + \bar{R}T \sum_{i=1}^{4} x_i \log x_i, \quad (21)$$

in which G_p is an unknown polynomial, $x_i \in [0, 1]$ for $i = 1, \ldots, N$ are $N = 3$ independent molar fractions, $x_4 = 1 - x_1 - x_2 - x_3$, and $\bar{R}$ and T are known constants [17]–[19]. While only G_p is modeled in [17] as the logarithmic terms are known and can be subtracted first, we model both G_p and G.

First, to model G_p we use the feature vector $\mathbf{x} = \begin{bmatrix}1, x_1, x_2, x_3\end{bmatrix}^\intercal$. The degree N and rank R have been determined by grid search and a validation set as $N = 4$ and $R = 12$. Using these parameters, we perform 50 experiments in which we randomly sample M data points to compute the $\mathbf{B}$ and $\mathbf{c}$ using the algebraic and optimization algorithms. For the latter, we keep only best result in terms of objective function value for five random initializations. We then compute the 99% quantile of the point–wise relative error for 166K validation points. In Fig. 5, we report the median over all experiments. Both methods achieve a very high accuracy, even for a very low number of samples M. (There are $IR = 48$ free variables.) For $M < 35 = \binom{4+4-1}{3}$, the algebraic method fails as $\mathbf{U}$ (Eq. (17)) is not of full column rank. If $M > 35$, the algebraic method is typically a bit more accurate, but also more expensive.

Second, we also model G without subtracting the logarithmic terms and use $\mathbf{x} = \begin{bmatrix}1, x_i, \log x_i, \log(1 - x_1 - x_2 - x_3)\end{bmatrix}^\intercal$, $i = 1, 2, 3$, as feature vector. As before, a maximal degree $N = 3$ and $R = 20$ are determined via validation sets. In Fig. 5, we show the error for two sampling strategies: unbiased and biased by taking at least 25% samples for which $x_i < 0.01$ for some $i = 1, 2, 3$. Biased sampling typically improves the accuracy, as can be seen in Fig. 5, which is expected as the logarithmic terms are large near $x = 0$. Compared to Fig. 5, the accuracy is worse, which is expected as the functions x and $\log x$ behave similarly in for $x \in (0, 1)$, resulting in an ill-conditioned problem.

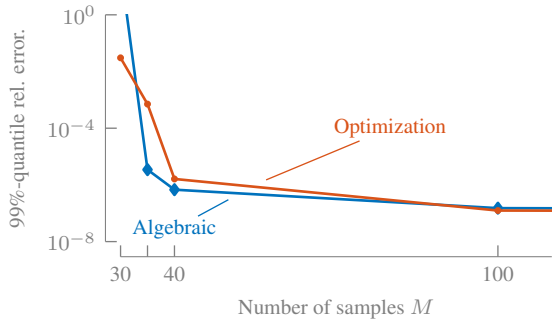


Fig. 2. A high accuracy is achieved by both methods when modeling G_p, even for few samples. The algebraic method fails for $M < 35$. Median over 50 experiments.

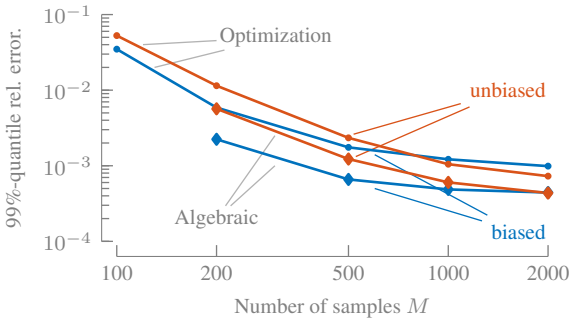


Fig. 3. Biased sampling improves accuracy when modeling G. The algebraic method is a bit more accurate, but it fails for $M < 120$. Median over 50 experiments.

6. CONCLUSION AND FURTHER WORK

A multivariate polynomial can be represented by a *low-rank* tensor, allowing efficient multivariate regression using few samples as illustrated for a materials sciences problem. By casting the problem as a linear system with a low-rank tensor constrained solution and exploiting the available structure, an efficient Gauss–Newton algorithm with a low time complexity can be derived. By exploiting the structure, the problem can be solved algebraically while lowering the number of required data points. While more expensive, it is often a good starting point for the optimization algorithm.

Future work includes the derivation of lower bounds on the number of samples via generic uniqueness conditions building upon [11]. Representing the coefficient tensor by other tensor decompositions which are often more suitable for compression, e.g., the multilinear singular value decomposition, is an other interesting future direction.

References

[1] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[2] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, 2013.

[5] N. D. Sidiropoulos, L. De Lathauwer, E. E. Xiao Fu, C. Kejun Huang, C. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[6] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 66, no. 6, pp. 789–797, 2018.

[7] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong, "Parallelized tensor train learning of polynomial classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4621–4632, 2018.

[8] S. Rendle, "Factorization machines," in *ICDM '10 Proceedings of the 2010 IEEE International Conference on Data Mining*, 2010, pp. 995–1000.

[9] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *Advances in Neural Information Processing Systems*, 2016, pp. 3351–3359.

[10] J. Yang and A. Gittens, "Tensor machines for learning target-specific polynomial features," 2015, [Online]. Available: `https://arxiv.org/abs/1504.01697`.

[11] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications," *Numerical Linear Algebra with Applications*, vol. 25, no. 6, e2190, 2018.

[12] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors—Part II: Uniqueness of the overall decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 876–903, 2013.

[13] ——, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm," *Linear Algebra and its Applications*, vol. 513, pp. 342–375, 2017.

[14] O. Debals, "Tensorization and applications in blind source separation," PhD thesis, KULeuven, 2017.

[15] J. Nocedal and S. Wright, *Numerical optimization*. Springer New York, Jul. 2006.

[16] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, *Tensorlab 3.0*, Available online at `https://www.tensorlab.net`, Mar. 2016.

[17] N. Vervliet, O. Debals, and L. De Lathauwer, "Canonical polyadic decomposition of incomplete tensors with linearly constrained factors," *Technical Report 16-172, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2017.

[18] A. Kroupa, "Modelling of phase diagrams and thermodynamic properties using Calphad method — Development of thermodynamic databases," *Computational Materials Science*, vol. 66, pp. 3–13, 2013.

[19] H. L. Lukas, S. G. Fries, and B. Sundman, *Computational thermodynamics: The Calphad method*. Cambridge university press, 2007.