Variable projection applied to block term decomposition of higher-order tensors

Guillaume Olikier[1(✉)], P.-A. Absil[1], and Lieven De Lathauwer[2,3] ⋆

[1] ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium
guillaume.olikier@uclouvain.be

[2] Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium

[3] KU Leuven Campus Kortrijk, Kortrijk, Belgium

Abstract. Higher-order tensors have become popular in many areas of applied mathematics such as statistics, scientific computing, signal processing or machine learning, notably thanks to the many possible ways of decomposing a tensor. In this paper, we focus on the best approximation in the least-squares sense of a higher-order tensor by a block term decomposition. Using variable projection, we express the tensor approximation problem as a minimization of a cost function on a Cartesian product of Stiefel manifolds. The effect of variable projection on the Riemannian gradient algorithm is studied through numerical experiments.

Keywords: numerical multilinear algebra, higher-order tensor, block term decomposition, variable projection method, Riemannian manifold, Riemannian optimization.

1 Introduction

Higher-order tensors have found numerous applications in signal processing and machine learning thanks to the many tensor decompositions available [1,2,3,4]. In this paper, we focus on a recently introduced tensor decomposition called block term decomposition (BTD) [5,6,7]. The usefulness of BTD in blind source separation was outlined in [8,9] and further examples are discussed in [10,11,12,13,14].

The BTD unifies the two most well known tensor decompositions which are the Tucker decomposition and the canonical polyadic decomposition (CPD). It

⋆ This work was supported by (1) "Communauté française de Belgique - Actions de Recherche Concertées" (contract ARC 14/19-060), (2) Research Council KU Leuven: C1 project C16/15/059-nD, (3) F.W.O.: project G.0830.14N, G.0881.14N, (4) Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no. 30468160 (SeLMA), (5) EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Advanced Grant: BIOTENSORS (no. 339804). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information.

also gives a unified view on how the basic concept of rank can be generalized from matrices to tensors. While in CPD, as well as in classical matrix decompositions, the components are rank-one terms, i.e., "atoms" of data, the terms in a BTD have "low" (multilinear) rank and can be thought of as "molecules" (consisting of several atoms) of data. Rank-one terms can only model data components that are proportional along columns, rows, ... and this assumption may not be realistic. On the other hand, block terms can model multidimensional sources, variations around mean activity, mildly nonlinear phenomena, drifts of setting points, frequency shifts, mildly convolutive mixtures, and so on. Such a molecular analysis is not possible in the matrix setting. Furthermore, it turns out that, like CPDs, BTDs are still unique under mild conditions [6,10].

In practice, it is more frequent to approximate a tensor by a BTD than to compute an exact BTD. More precisely, the problem of interest is to compute the best approximation in the least-squares sense of a higher-order tensor by a BTD. Only a few algorithms are currently available for this task. The Matlab toolbox Tensorlab [15] proposes the two following functions: (i) `btd_minf` uses L-BFGS with dogleg trust region (a quasi-Newton method), (ii) `btd_nls` uses nonlinear least squares by Gauss–Newton with dogleg trust region. Another available algorithm is the alternating least squares algorithm introduced in [7]. This algorithm is not included in Tensorlab and does not work better than `btd_nls` in general.

In this paper, we show that the performance of numerical methods can be improved using variable projection. Variable projection consists in exploiting the fact that, when the optimal value of some of the optimization variables is easy to find when the others are fixed, this optimal value can be injected in the objective function, yielding a new optimization problem where only the other variables appear. This technique has already been applied to the Tucker decomposition in [16] and exploited in [17,18]. Here we extend it to the BTD approximation problem which is then expressed as a minimization of a cost function on a Cartesian product of Stiefel manifolds. Numerical experiments show that variable projection modifies the performance of the Riemannian gradient algorithm for BTDs of two terms by either increasing or decreasing its running time and/or its reliability. Preliminary results can be found in the short conference paper [19]. The present paper gives a detailed derivation of the variable projection technique and presents numerical experiments for noised BTDs. We focus on third-order tensors for simplicity but the generalization to tensors of any order is straightforward.

2 Preliminaries and notation

We let $\mathbb{R}^{I_1 \times I_2 \times I_3}$ denote the set of real third-order tensors of size (I_1, I_2, I_3). In order to improve readability, vectors are written in bold-face lower-case (e.g., $\mathbf{a}$), matrices in bold-face capitals (e.g., $\mathbf{A}$), and higher-order tensors in calligraphic letters (e.g., $\mathcal{A}$). For $n \in \{1, 2, 3\}$, the mode-n vectors of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ are obtained by varying the nth index while keeping the other indices fixed. The mode-n rank of $\mathcal{A}$, denoted $\operatorname{rank}_n(\mathcal{A})$, is the dimension of the linear space

spanned by its mode-n vectors. The multilinear rank of $\mathcal{A}$ is the triple of the mode-n ranks. The mode-n product of $\mathcal{A}$ by $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$, denoted $\mathcal{A} \cdot_n \mathbf{B}$, is obtained by multiplying all the mode-n vectors of $\mathcal{A}$ by $\mathbf{B}$. We endow $\mathbb{R}^{I_1 \times I_2 \times I_3}$ with the standard inner product, defined by

$$\langle \mathcal{A}, \mathcal{B} \rangle := \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} \mathcal{A}(i_1, i_2, i_3)\mathcal{B}(i_1, i_2, i_3),$$

and we let $\|\cdot\|$ denote the induced norm, i.e., the Frobenius norm. It is sometimes convenient to represent a tensor as a vector (vectorization) or as a matrix (matricization). The vectorization of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, denoted $\mathrm{vec}(\mathcal{A})$, is the vector of length $I_1 I_2 I_3$ defined as follows:

$$(\mathrm{vec}(\mathcal{A}))\,((i_1 - 1)I_2 I_3 + (i_2 - 1)I_3 + i_3) := \mathcal{A}(i_1, i_2, i_3).$$

We define the following matrix representations of $\mathcal{A}$:

$$\begin{aligned} \mathcal{A}(i_1, i_2, i_3) &= (\mathbf{A}_{(1)})(i_1, I_3(i_2 - 1) + i_3) \\ &= (\mathbf{A}_{(2)})(i_2, I_1(i_3 - 1) + i_1) \\ &= (\mathbf{A}_{(3)})(i_3, I_2(i_1 - 1) + i_2). \end{aligned}$$

One can check that if $\mathcal{A} = \mathcal{S} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V} \cdot_3 \mathbf{W}$, then

$$\mathrm{vec}(\mathcal{A}) = (\mathbf{U} \otimes \mathbf{V} \otimes \mathbf{W})\,\mathrm{vec}(\mathcal{S}), \tag{1}$$

$$\mathbf{A}_{(1)} = \mathbf{U}\mathbf{S}_{(1)}(\mathbf{V} \otimes \mathbf{W})^{\mathrm{T}}, \tag{2}$$

$$\mathbf{A}_{(2)} = \mathbf{V}\mathbf{S}_{(2)}(\mathbf{W} \otimes \mathbf{U})^{\mathrm{T}}, \tag{3}$$

$$\mathbf{A}_{(3)} = \mathbf{W}\mathbf{S}_{(3)}(\mathbf{U} \otimes \mathbf{V})^{\mathrm{T}}. \tag{4}$$

Vectorization and matricization are linear mappings which preserve the norm.

3 Variable projection

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Consider positive integers R and R_i such that $R_i \leq \mathrm{rank}_i(\mathcal{A})$ for each $i \in \{1, 2, 3\}$ and $m := I_1 I_2 I_3 \geq R R_1 R_2 R_3 =: n$. The approximation of $\mathcal{A}$ by a BTD of R terms of multilinear rank (R_1, R_2, R_3) is a nonconvex minimization problem which can be expressed using variable projection as

$$\min_{\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W}} \underbrace{\left\| \mathcal{A} - \sum_{r=1}^{R} \mathcal{S}_r \cdot_1 \mathbf{U}_r \cdot_2 \mathbf{V}_r \cdot_3 \mathbf{W}_r \right\|^2}_{=: f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})} = \min_{\mathbf{U},\mathbf{V},\mathbf{W}} \underbrace{\min_{\mathcal{S}} f_{\mathcal{A}}(\mathcal{S}, \mathbf{U}, \mathbf{V}, \mathbf{W})}_{=: g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})}$$

for the variables $\mathcal{S} \in (\mathbb{R}^{R_1 \times R_2 \times R_3})^R$, $\mathbf{U} \in (\mathbb{R}^{I_1 \times R_1})^R$, $\mathbf{V} \in (\mathbb{R}^{I_2 \times R_2})^R$ and $\mathbf{W} \in (\mathbb{R}^{I_3 \times R_3})^R$ subject to the constraints $\mathbf{U} \in \mathrm{St}(R_1, I_1)^R$, $\mathbf{V} \in \mathrm{St}(R_2, I_2)^R$

and $\mathbf{W} \in \mathrm{St}(R_3, I_3)^R$, where given integers $p \geq q \geq 1$ we let $\mathrm{St}(q,p)$ denote the *Stiefel manifold*, i.e.,

$$\mathrm{St}(q,p) := \{\mathbf{X} \in \mathbb{R}^{p\times q} : \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I}_q\}.$$

A schematic representation of the BTD approximation problem is given in Fig. 1. Each term in a BTD is a Tucker term. The tensors $\mathcal{S}_r \in \mathbb{R}^{R_1\times R_2\times R_3}$ are called the core tensors while the matrices $\mathbf{U}_r, \mathbf{V}_r, \mathbf{W}_r$, which can be assumed to be in the Stiefel manifold without loss of generality, are referred to as the factor matrices.

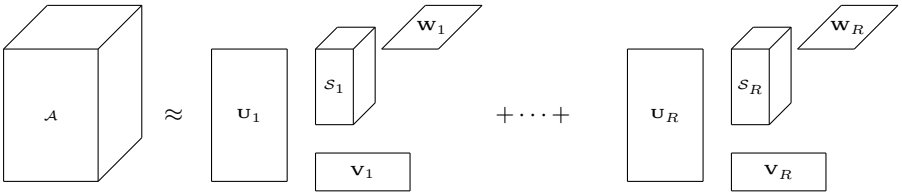

Fig. 1. Schematic representation of the BTD approximation problem.

Computing $g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})$ is a least squares problem. Indeed, using (1), if we define $\mathbf{a} := \mathrm{vec}(\mathcal{A}) \in \mathbb{R}^m$, $\mathbf{P}(\mathbf{U},\mathbf{V},\mathbf{W}) := [\mathbf{U}_j \otimes \mathbf{V}_j \otimes \mathbf{W}_j]_{i,j=1}^{1,R} \in \mathbb{R}^{m\times n}$ and $\mathbf{s} := [\mathrm{vec}(\mathcal{S}_i)]_{i,j=1}^{R,1} \in \mathbb{R}^n$, then

$$g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W}) = \min_{\mathbf{s}\in\mathbb{R}^n} \|\mathbf{a} - \mathbf{P}(\mathbf{U},\mathbf{V},\mathbf{W})\mathbf{s}\|^2.$$

We let $\mathcal{S}^*(\mathbf{U},\mathbf{V},\mathbf{W})$ denote the minimizer of this least squares problem.[1] Thus,

$$g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W}) = f_{\mathcal{A}}(\mathcal{S}^*(\mathbf{U},\mathbf{V},\mathbf{W}),\mathbf{U},\mathbf{V},\mathbf{W}).$$

Computing the partial derivatives of $g_{\mathcal{A}}$ reduces to the computation of partial derivatives of $f_{\mathcal{A}}$. Indeed, using the first-order optimality condition

$$\left.\frac{\partial f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathcal{S}}\right|_{\mathcal{S}=\mathcal{S}^*(\mathbf{U},\mathbf{V},\mathbf{W})} = \mathbf{0} \tag{5}$$

and the chain rule yields

$$\frac{\partial g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})}{\partial(\mathbf{U},\mathbf{V},\mathbf{W})} = \left.\frac{\partial f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})}{\partial(\mathbf{U},\mathbf{V},\mathbf{W})}\right|_{\mathcal{S}=\mathcal{S}^*(\mathbf{U},\mathbf{V},\mathbf{W})}. \tag{6}$$

It remains to compute those partial derivatives of $f_{\mathcal{A}}$. In order to make the derivation convenient, we first recall some basic facts on differentiation. Given two vector spaces X and Y over a same field, we let $\mathrm{Lin}(X,Y)$ denote the vector space of linear mappings from X to Y.

[1] The minimizer is unique if and only if the matrix $\mathbf{P}(\mathbf{U},\mathbf{V},\mathbf{W})$ has full column rank which is the case almost everywhere (with respect to the Lebesgue measure) since $m \geq n$.

Total derivative and gradient. Let $(X, \langle\cdot,\cdot\rangle)$ be a pre-Hilbert space and let $\|\cdot\|$ denote the norm induced by the inner product $\langle\cdot,\cdot\rangle$. A function $f : X \to \mathbb{R}$ is *differentiable* at $x \in X$ if and only if there is $L \in \mathrm{Lin}(X, \mathbb{R})$ such that

$$\lim_{h\to 0} \frac{f(x+h) - f(x) - L(h)}{\|h\|} = 0,$$

which means that for every $\epsilon > 0$, there is $\delta > 0$ such that for any $h \in X$, $\|h\| \leq \delta$ implies

$$\frac{|f(x+h) - f(x) - L(h)|}{\|h\|} \leq \epsilon.$$

If such a L exists, it is unique, denoted by $\mathrm{D}\, f(x)$, and called the *total derivative* of f at x. The *gradient* of f at x is the only $g \in X$ such that

$$\mathrm{D}\, f(x)[h] = \langle g, h\rangle$$

for all $h \in X$; it is denoted by $\operatorname{grad} f(x)$. If f is differentiable at $x \in X$, then

$$\mathrm{D}\, f(x)[h] = \lim_{t\to 0} \frac{f(x+th) - f(x)}{t}.$$

for every $h \in X$.

Gradient of the squared norm. Let $f : X \to \mathbb{R} : x \mapsto f(x) := \|x\|^2$. For any $x, h \in X$ and any real $t \neq 0$,

$$\frac{f(x+th) - f(x)}{t} = \frac{2t\langle x, h\rangle + t^2 \|h\|^2}{t} = 2\langle x, h\rangle + t\|h\|^2.$$

It follows that $\mathrm{D}\, f(x)[h] = 2\langle x, h\rangle$ and so that $\operatorname{grad} f(x) = 2x$.

Affine transformation. Let $(X, \langle\cdot,\cdot\rangle_X)$ and $(Y, \langle\cdot,\cdot\rangle_Y)$ be two pre-Hilbert spaces, $g : Y \to \mathbb{R}$ be differentiable, $L \in \mathrm{Lin}(X, Y)$, $b \in Y$, $A : X \to Y : x \mapsto A(x) := L(x) + b$, and $f := g \circ A$. For any $x, h \in X$,

$$\begin{aligned}
\langle \operatorname{grad} f(x), h\rangle_X &= \lim_{t\to 0} \frac{f(x+th) - f(x)}{t} \\
&= \lim_{t\to 0} \frac{g(L(x) + b + tL(h)) - g(L(x) + b)}{t} \\
&= \langle \operatorname{grad} g(L(x) + b), L(h)\rangle_Y.
\end{aligned}$$

From now on, let us assume that X and Y have finite dimension so that L has an *adjoint*, which means that there is a (unique) $L^* \in \mathrm{Lin}(Y, X)$ such that

$$\langle y, L(x)\rangle_Y = \langle L^*(y), x\rangle_X$$

for any $x \in X$ and $y \in Y$. This allows us to conclude that for any $x \in X$,

$$\operatorname{grad} f(x) = L^*(\operatorname{grad} g(L(x) + b)).$$

Adjoint of the matrix product. Let $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\mathbf{B} \in \mathbb{R}^{q\times n}$. The adjoint of

$$L : \mathbb{R}^{p\times q} \to \mathbb{R}^{m\times n} : \mathbf{X} \mapsto \mathbf{AXB}$$

is

$$L^* : \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q} : \mathbf{Y} \mapsto \mathbf{A}^{\mathrm{T}}\mathbf{Y}\mathbf{B}^{\mathrm{T}}.$$

Partial derivatives of $f_{\mathcal{A}}$. Using the matricization formulas (2)-(4) yields

$$\begin{aligned} f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W}) &= \left\| \sum_{r=1}^{R} \mathbf{U}_r (\mathbf{S}_r)_{(1)} (\mathbf{V}_r \otimes \mathbf{W}_r)^{\mathrm{T}} - \mathbf{A}_{(1)} \right\|^2 \\ &= \left\| \sum_{r=1}^{R} \mathbf{V}_r (\mathbf{S}_r)_{(2)} (\mathbf{W}_r \otimes \mathbf{U}_r)^{\mathrm{T}} - \mathbf{A}_{(2)} \right\|^2 \\ &= \left\| \sum_{r=1}^{R} \mathbf{W}_r (\mathbf{S}_r)_{(3)} (\mathbf{U}_r \otimes \mathbf{V}_r)^{\mathrm{T}} - \mathbf{A}_{(3)} \right\|^2. \end{aligned}$$

Applying the results of the preceding paragraphs to these three equations gives the three following ones for every $i \in \{1,\dots,R\}$:

$$\frac{\partial f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{U}_i} = 2\left(\sum_{j=1}^{R} \mathbf{U}_j (\mathbf{S}_j)_{(1)} (\mathbf{V}_j \otimes \mathbf{W}_j)^{\mathrm{T}} - \mathbf{A}_{(1)} \right) (\mathbf{V}_i \otimes \mathbf{W}_i)(\mathbf{S}_i)_{(1)}^{\mathrm{T}},$$

$$\frac{\partial f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{V}_i} = 2\left(\sum_{j=1}^{R} \mathbf{V}_j (\mathbf{S}_j)_{(2)} (\mathbf{W}_j \otimes \mathbf{U}_j)^{\mathrm{T}} - \mathbf{A}_{(2)} \right) (\mathbf{W}_i \otimes \mathbf{U}_i)(\mathbf{S}_i)_{(2)}^{\mathrm{T}},$$

$$\frac{\partial f_{\mathcal{A}}(\mathcal{S},\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{W}_i} = 2\left(\sum_{j=1}^{R} \mathbf{W}_j (\mathbf{S}_j)_{(3)} (\mathbf{U}_j \otimes \mathbf{V}_j)^{\mathrm{T}} - \mathbf{A}_{(3)} \right) (\mathbf{U}_i \otimes \mathbf{V}_i)(\mathbf{S}_i)_{(3)}^{\mathrm{T}}.$$

4 Riemannian gradient algorithm

We have shown in the preceding section that the approximation of $\mathcal{A}$ by a BTD reduces to the minimization of a real-valued function defined on a Riemannian manifold, namely, the restriction of $g_{\mathcal{A}}$ on $\prod_{i=1}^{3} \mathrm{St}(R_i, I_i)^R$. In this section, we briefly introduce the Riemannian gradient algorithm which we shall use to solve our problem; our reference is [20].

Line-search methods to minimize a real-valued function F defined on a Riemannian manifold $\mathcal{M}$ are based on the update formula

$$x_{k+1} = R_{x_k}(t_k \eta_k),$$

where η_k is selected in the tangent space to $\mathcal{M}$ at x_k, denoted $\mathrm{T}_{x_k}\mathcal{M}$, R_{x_k} is a retraction on $\mathcal{M}$ at x_k, and $t_k \in \mathbb{R}$. The algorithm is defined by the choice of three ingredients: the retraction R_{x_k}, the search direction η_k and the step size t_k.

The gradient method consists of choosing $\eta_k := -\operatorname{grad} F(x_k)$ where $\operatorname{grad} F$ is the Riemannian gradient of F. In the case where $\mathcal{M}$ is an embedded submanifold of a linear space $\mathcal{E}$ and F is the restriction on $\mathcal{M}$ of some function $\bar{F} : \mathcal{E} \to \mathbb{R}$, $\operatorname{grad} F(x)$ is simply the projection of the usual gradient of $\bar{F}$ at x on $\mathrm{T}_x\mathcal{M}$. For instance, $\mathrm{St}(q,p)$ is an embedded submanifold of $\mathbb{R}^{p\times q}$ and the projection of $\mathbf{Y} \in \mathbb{R}^{p\times q}$ on $\mathrm{T}_{\mathbf{X}}\mathrm{St}(q,p)$ is given by [20, equation (3.35)]

$$(\mathbf{I}_p - \mathbf{X}\mathbf{X}^{\mathrm{T}})\mathbf{Y} + \mathbf{X}\operatorname{skew}(\mathbf{X}^{\mathrm{T}}\mathbf{Y}) \tag{7}$$

where $\operatorname{skew}(\mathbf{A}) := \frac{1}{2}(\mathbf{A} - \mathbf{A}^{\mathrm{T}})$ is the skew-symmetric part of $\mathbf{A}$. Our cost function, the restriction of $g_{\mathcal{A}}$ on $\prod_{i=1}^{3} \mathrm{St}(R_i, I_i)^R$, is defined on a Cartesian product of Stiefel manifolds; this is not an issue since the tangent space of a Cartesian product is the Cartesian product of the tangent spaces and the projection can be performed componentwise. We are now able to compute the Riemannian gradient of the restriction of $g_{\mathcal{A}}$. Starting from the first-order optimality condition (5) written in matrix forms (2)-(4), we can show that for each $i \in \{1, \ldots, R\}$,

$$\mathbf{U}_i^{\mathrm{T}} \frac{\partial g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{U}_i} = \mathbf{V}_i^{\mathrm{T}} \frac{\partial g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{V}_i} = \mathbf{W}_i^{\mathrm{T}} \frac{\partial g_{\mathcal{A}}(\mathbf{U},\mathbf{V},\mathbf{W})}{\partial \mathbf{W}_i} = \mathbf{0}.$$

Therefore, in view of the projection formula (7), the Riemannian gradient of the restriction of $g_{\mathcal{A}}$ is equal to the (usual) gradient of $g_{\mathcal{A}}$ given by (6).

A popular retraction on $\mathrm{St}(q,p)$, which we shall use in our problem, is the qf retraction [20, equation (4.8)]:

$$R_{\mathbf{X}}(\mathbf{Y}) := \operatorname{qf}(\mathbf{X} + \mathbf{Y})$$

where $\operatorname{qf}(\mathbf{A})$ is the $\mathbf{Q}$ factor of the decomposition of $\mathbf{A} \in \mathbb{R}^{p\times q}$ with $\operatorname{rank}(\mathbf{A}) = q$ as $\mathbf{A} = \mathbf{Q}\mathbf{R}$ where $\mathbf{Q} \in \mathrm{St}(q,p)$ and $\mathbf{R}$ is an upper triangular $q \times q$ matrix with positive diagonal elements. Again, the manifold in our problem is a Cartesian product of Stiefel manifolds and in this case the retraction can be performed componentwise.

At this point, it remains to specify the step size t_k. For that purpose, we will use the backtracking strategy presented in [20, section 4.2]. Assume we are at the kth iteration. We want to find $t_k > 0$ such that $F(R_{x_k}(-t_k \operatorname{grad} F(x_k)))$ is sufficiently small compared to $F(x_k)$. This can be achieved by the Armijo rule: given $\bar{\alpha} > 0$, $\beta, \sigma \in (0,1)$ and $\tau_0 := \bar{\alpha}$, we iterate $\tau_i := \beta\tau_{i-1}$ until

$$F(R_{x_k}(-\tau_i \operatorname{grad} F(x_k))) \leq F(x_k) - \sigma\tau_i \left\|\operatorname{grad} F(x_k)\right\|^2$$

and then set $t_k := \tau_i$. In our implementation, we set $\bar{\alpha} := 0.2$, $\sigma := 10^{-3}$, $\beta := 0.2$ and we perform at most 10 iterations in the backtracking loop.

The procedure described in the preceding paragraph corresponds to [20, Algorithm 1] with $c := 1$ and equality in [20, equation (4.12)], except that the number of iterations in the backtracking loop is limited. In our problem, the domain of the cost function is compact since it is a Cartesian product of Stiefel manifolds. Therefore, [20, Corollary 4.3.2] applies and ensures that

$$\lim_{k\to\infty} \left\|\operatorname{grad} F(x_k)\right\| = 0,$$

except if at some iteration the backtracking loop needs more than 10 iterations. In view of this result, it seems natural to stop the algorithm as soon as the norm of the Riemannian gradient becomes smaller than a given quantity $\epsilon > 0$.

5 Numerical results

In this section, we perform numerical experiments to study the effect of variable projection on the Riemannian gradient algorithm applied to the BTD problem. To this end, we evaluate the ability of this algorithm, both with and without variable projection, to recover known BTDs possibly corrupted by some noise. Thus, in this experiment, we try to recover a structure that is really present.

First, we explain how we build BTDs for this test. We set $R := 2$ and we select the parameters (I_1, I_2, I_3) and (R_1, R_2, R_3). Then, for each $r \in \{1, \dots, R\}$, we select $\mathcal{S}_r \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, $\mathbf{U}_r \in \mathrm{St}(R_1, I_1)$, $\mathbf{V}_r \in \mathrm{St}(R_2, I_2)$ and $\mathbf{W}_r \in \mathrm{St}(R_3, I_3)$ according to the standard normal distribution, i.e., $\mathcal{S}_r :=$ `randn(`R_1`,`R_2`,`R_3`)` and $\mathbf{U}_r :=$ `qf(randn(`I_1`,`R_1`))` in Matlab. Then, we set

$$\mathcal{A} := \sum_{r=1}^{R} \mathcal{S}_r \cdot_1 \mathbf{U}_r \cdot_2 \mathbf{V}_r \cdot_3 \mathbf{W}_r. \tag{8}$$

Finally, we select $\mathcal{N} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ according to the standard normal distribution, i.e., $\mathcal{N} :=$ `randn(`I_1`,`I_2`,`I_3`)` in Matlab, and define

$$A_\sigma := \frac{\mathcal{A}}{\|\mathcal{A}\|} + \sigma \frac{\mathcal{N}}{\|\mathcal{N}\|} \tag{9}$$

for some real value of the parameter σ which controls the noise level on the BTD.

Now, we describe the test itself. For 100 different $\mathcal{A}_\sigma$ as in (9), we ran the Riemannian gradient algorithm with variable projection (i.e., on the cost function $g_{\mathcal{A}_\sigma}$) and without variable projection (i.e., on the cost function $f_{\mathcal{A}_\sigma}$) using for each $\mathcal{A}_\sigma$ a randomly selected starting iterate. Representative results are given in Table 1 for $\sigma := 0$ and $\sigma := 0.3$, which corresponds to a signal-to-noise ratio of about 10 dB, both for $(I_1, I_2, I_3) := (5, 5, 5)$ and $(R_1, R_2, R_3) := (2, 2, 2)$.[2]

The success ratios are not equal to one because the number of iterations that can be performed by the algorithm was (arbitrarily) limited to 10^4. When variable projection is used, on one hand, the mean running time is multiplied by about 0.86 for $\sigma := 0$ and 0.78 for $\sigma := 0.3$, and on the other hand, the success ratio is multiplied by about 0.89 for both $\sigma := 0$ and $\sigma := 0.3$.

The same test with $(I_1, I_2, I_3) := (10, 10, 10)$ and $(R_1, R_2, R_3) := (2, 2, 3)$, still with $\sigma := 0$ and $\sigma := 0.3$, has been conducted.[3] For both values of σ, we observed that variable projection multiplies the running time by about 1.1 on one hand, and multiplies the success ratio by about 1.4 on the other hand.

[2] The Matlab code that produced the results is available at `https://sites.uclouvain.be/absil/2018.01`.

[3] With these parameters, the BTD $\mathcal{A}$ in (8) is *essentially unique* by [6, Theorem 5.3].

	$\sigma := 0$		$\sigma := 0.3$	
	with VP	without VP	with VP	without VP
successes	39	44	41	46
min(iter)	2047	2069	995	891
mean(iter)	5644	5966	4119	4740
max(iter)	9509	9960	9498	9958
mean(backtracking iter)	1	1	1.004	1
min(time)	2.11	2.36	1.05	1.02
mean(time)	5.85	6.83	4.25	5.44
max(time)	9.79	11.35	9.77	11.35

Table 1. By "success", we mean for $\sigma = 0$ that the norm of the (Riemannian) gradient is brought below $5 \cdot 10^{-14}$ and that the objective function is brought below 10^{-25} within 10^4 iterations; for $\sigma = 0.3$, we mean that the norm of the gradient is brought below 10^{-7} still within 10^4 iterations; the algorithm was not able to bring the norm of the gradient as low as in the noise-free case. Notation: "iter" refers to the number of iterations performed by the gradient algorithm while "backtracking iter" refers to the number of iterations performed in the backtracking loops. Running times are given in seconds. The information in each column is computed based only on the successful runs.

6 Conclusion

In this paper, we applied variable projection to the BTD problem and discussed its effect on the Riemannian gradient algorithm. Our numerical experiments showed that variable projection may either increase or decrease the running time and/or the reliability of the algorithm depending on the particular data tensor considered.

References

1. A. Cichocki, D. Mandic, A. H. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, March 2015.
2. 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.
3. A. Cichocki, N. Lee, I. Oseledets, A. H. Phan, Q. Zhao, D. Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.
4. A. Cichocki, A. H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

5. L. De Lathauwer. Decompositions of a higher-order tensor in block terms—Part I: Lemmas for partitioned matrices. *SIAM J. Matrix Anal. Appl.*, 30(3):1022–1032, 2008.
6. L. De Lathauwer. Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness. *SIAM J. Matrix Anal. Appl.*, 30(3):1033–1066, 2008.
7. L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms—Part III: Alternating least squares algorithms. *SIAM J. Matrix Anal. Appl.*, 30(3):1067–1083, 2008.
8. L. De Lathauwer. Block component analysis, a new concept for blind source separation. In F. Theis, A. Cichocki, A. Yeredor, and M. Zibulevsky, editors, *Latent Variable Analysis and Signal Separation: 10th International Conference, LVA/ICA 2012, Tel Aviv, Israel, March 12-15, 2012. Proceedings*, pages 1–8. Springer Berlin Heidelberg, 2012.
9. M. Yang, Z. Kang, C. Peng, W. Liu, and Q. Cheng. On block term tensor decompositions and its applications in blind signal separation. URL: `http://archive.ymsc.tsinghua.edu.cn/pacm_paperurl/20160105102343471889031`.
10. L. De Lathauwer. Blind separation of exponential polynomials and the decomposition of a tensor in rank-$(L_r, L_r, 1)$ terms. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1451–1474, December 2011.
11. O. Debals, M. Van Barel, and L. De Lathauwer. Löwner-based blind signal separation of rational functions with applications. *IEEE Transactions on Signal Processing*, 64(8):1909–1918, April 2016.
12. B. Hunyadi, D. Camps, L. Sorber, W. Van Paesschen, M. De Vos, S. Van Huffel, and L. De Lathauwer. Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, 2014(1):139, September 2014.
13. C. Chatzichristos, E. Kofidis, Y. Kopsinis, M. M. Moreno, and S. Theodoridis. Higher-order block term decomposition for spatially folded fMRI data. In P. Tichavský, M. Babaie-Zadeh, O. J. J. Michel, and N. Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation*, pages 3–15, Cham, 2017. Springer International Publishing.
14. C. Chatzichristos, E. Kofidis, and S. Theodoridis. PARAFAC2 and its block term decomposition analog for blind fMRI source unmixing. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2081–2085, Aug 2017.
15. N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0, Mar. 2016. Available online. URL: `https://www.tensorlab.net`.
16. L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.
17. M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer. Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme. *SIAM J. Matrix Anal. Appl.*, 32(1):115–135, 2011.
18. B. Savas and L.-H. Lim. Quasi-Newton Methods on Grassmannians and Multilinear Approximations of Tensors. *SIAM J. on Scientific Computing*, 32(6):3352–3393, 2010.
19. G. Olikier, P.-A. Absil, and L. De Lathauwer. A variable projection method for block term decomposition of higher-order tensors. *Accepted for ESANN 2018*.
20. P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA, 2008.