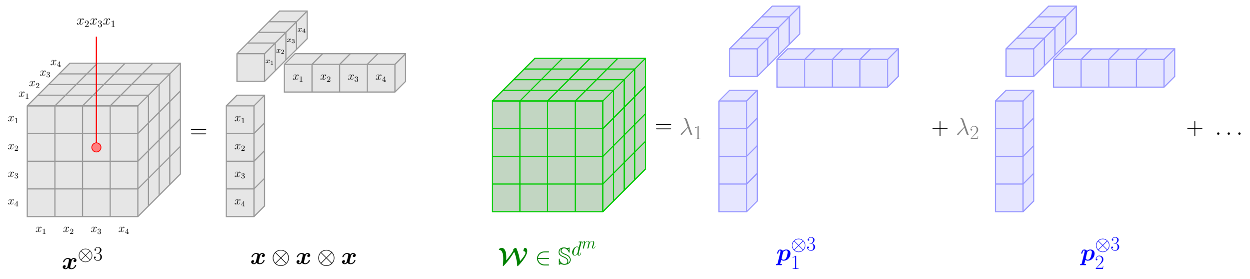
$x_2 x_3 x_1$
x_1 x_2 x_3 x_4
$=$
$\boldsymbol{x}^{\otimes 3}$
$\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x}$
$\boldsymbol{\mathcal{W}} \in \mathbb{S}^{d^m}$
$= \lambda_1$
$+ \lambda_2$
$+ \dots$
$\boldsymbol{p}_1^{\otimes 3}$
$\boldsymbol{p}_2^{\otimes 3}$

Figure 5. Illustration of symmetric rank-one tensor (left) and symmetric outer product decomposition (right).

Supplementary material

A. Symmetric tensors

A.1. Background

Let $\mathbb{R}^{d_1 \times \dots \times d_m}$ be the set of $d_1 \times \dots \times d_m$ real m-order tensors. In this paper, we focus on cubical tensors, i.e., $d_1 = \dots = d_m = d$. We denote the set of m-order cubical tensors by $\mathbb{R}^{d^m}$. We denote the elements of $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d^m}$ by $\boldsymbol{\mathcal{M}}_{j_1,\dots,j_m}$, where $j_1, \dots, j_m \in [d]$.

Let $\boldsymbol{\sigma} = [\sigma_1, \dots, \sigma_m]$ be a permutation of $\{1, \dots, m\}$. Given $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d^m}$, we define $\boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}} \in \mathbb{R}^{d^m}$ as the tensor such that

$$(\boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}})_{j_1,\dots,j_m} := \boldsymbol{\mathcal{M}}_{j_{\sigma_1},\dots,j_{\sigma_m}} \quad \forall j_1, \dots, j_m \in [d].$$

In other words $\boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}}$ is a copy of $\boldsymbol{\mathcal{M}}$ with its axes permuted. This generalizes the concept of transpose to tensors.

Let P_m be the set of all permutations of $\{1, \dots, m\}$. We say that a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d^m}$ is symmetric if and only if

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{\sigma}} = \boldsymbol{\mathcal{X}} \quad \forall \boldsymbol{\sigma} \in P_m.$$

We denote the set of symmetric tensors by $\mathbb{S}^{d^m}$.

Given $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d^m}$, we define the symmetrization of $\boldsymbol{\mathcal{M}}$ by

$$\mathcal{S}(\boldsymbol{\mathcal{M}}) = \frac{1}{m!} \sum_{\boldsymbol{\sigma} \in P_m} \boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}}.$$

Note that when $m = 2$, then $\mathcal{S}(\boldsymbol{M}) = \frac{1}{2}(\boldsymbol{M} + \boldsymbol{M}^{\mathrm{T}})$.

Given $\boldsymbol{x} \in \mathbb{R}^d$, we define a symmetric rank-one tensor by $\boldsymbol{x}^{\otimes m} := \underbrace{\boldsymbol{x} \otimes \dots \otimes \boldsymbol{x}}_{m \text{ times}} \in \mathbb{S}^{d^m}$, i.e., $(\boldsymbol{x}^{\otimes m})_{j_1, j_2, \dots, j_m} = x_{j_1} x_{j_2} \dots x_{j_m}$. We denote the symmetric outer product decomposition (Comon et al., 2008) of $\boldsymbol{\mathcal{W}} \in \mathbb{S}^{d^m}$ by

$$\boldsymbol{\mathcal{W}} = \sum_{s=1}^{k} \lambda_s \boldsymbol{p}_s^{\otimes m},$$

where k is called the symmetric rank of $\boldsymbol{\mathcal{W}}$. This generalizes the concept of eigendecomposition to tensors. These two concepts are illustrated in Figure 5.

A.2. Proof of Lemma 6

Assume $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d^m}$ and $\boldsymbol{\mathcal{X}} \in \mathbb{S}^{d^m}$. Then,

$$\begin{aligned}
\langle \mathcal{S}(\boldsymbol{\mathcal{M}}), \boldsymbol{\mathcal{X}} \rangle &= \frac{1}{m!} \sum_{\boldsymbol{\sigma} \in P_m} \langle \boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}}, \boldsymbol{\mathcal{X}} \rangle && \text{by definition of } \mathcal{S}(\boldsymbol{\mathcal{M}}) \text{ and by linearity} \\
&= \frac{1}{m!} \sum_{\boldsymbol{\sigma} \in P_m} \langle (\boldsymbol{\mathcal{M}}_{\boldsymbol{\sigma}})_{\boldsymbol{\sigma}^{-1}}, \boldsymbol{\mathcal{X}}_{\boldsymbol{\sigma}^{-1}} \rangle && \text{since } \langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \langle \boldsymbol{\mathcal{A}}_{\boldsymbol{\sigma}}, \boldsymbol{\mathcal{B}}_{\boldsymbol{\sigma}} \rangle \; \forall \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{d^m}, \forall \boldsymbol{\sigma} \in P_m \\
&= \frac{1}{m!} \sum_{\boldsymbol{\sigma} \in P_m} \langle \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{X}}_{\boldsymbol{\sigma}^{-1}} \rangle && \text{by definition of inverse permutation} \\
&= \frac{1}{m!} \sum_{\boldsymbol{\sigma} \in P_m} \langle \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{X}} \rangle && \text{since } \boldsymbol{\mathcal{X}} \in \mathbb{S}^{d^m} \\
&= \langle \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{X}} \rangle.
\end{aligned}$$

B. Proofs related to ANOVA kernels

B.1. Proof of multi-linearity (Lemma 2)

For $m = 1$, we have

$$\begin{aligned}
\mathcal{A}^1(\boldsymbol{p}, \boldsymbol{x}) &= \sum_{j=1}^{d} p_j x_j \\
&= \sum_{k \neq j} p_k x_k + p_j x_j \\
&= \mathcal{A}^1(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j}) + \; p_j x_j \; \mathcal{A}^0(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j})
\end{aligned}$$

where we used $\mathcal{A}^0(\boldsymbol{p}, \boldsymbol{x}) = 1$.

For $1 < m \leq d$, first notice that we can rewrite (3) as

$$\begin{aligned}
\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) &= \sum_{j_m > \cdots > j_1} p_{j_1} x_{j_1} \dots p_{j_m} x_{j_m} \quad j_k \in [d], k \in [m] \\
&= \sum_{j_1=1}^{d-m+1} \sum_{j_2=j_1+1}^{d-m+2} \cdots \sum_{j_m=j_{m-1}+1}^{d} p_{j_1} x_{j_1} \dots p_{j_m} x_{j_m}.
\end{aligned}$$

Then,

$$\begin{aligned}
\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) &= \sum_{j_1=1}^{d-m+1} \sum_{j_2=j_1+1}^{d-m+2} \cdots \sum_{j_m=j_{m-1}+1}^{d} p_{j_1} x_{j_1} p_{j_2} x_{j_2} \dots p_{j_m} x_{j_m} \\
&= \sum_{j_2=j_1+1}^{d-m+2} \cdots \sum_{j_m=j_{m-1}+1}^{d} p_1 x_1 p_{j_2} x_{j_2} \dots p_{j_m} x_{j_m} + \\
&\quad \sum_{j_1=2}^{d-m+1} \sum_{j_2=j_1+1}^{d-m+2} \cdots \sum_{j_m=j_{m-1}+1}^{d} p_{j_1} x_{j_1} p_{j_2} x_{j_2} \dots p_{j_m} x_{j_m} \\
&= p_1 x_1 \mathcal{A}^{m-1}(\boldsymbol{p}_{\neg 1}, \boldsymbol{x}_{\neg 1}) + \mathcal{A}^m(\boldsymbol{p}_{\neg 1}, \boldsymbol{x}_{\neg 1}).
\end{aligned}$$

We can always permute the elements of $\boldsymbol{p}$ and $\boldsymbol{x}$ without changing $\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x})$. It follows that

$$\mathcal{A}^m(\boldsymbol{p}, \boldsymbol{x}) = p_j x_j \mathcal{A}^{m-1}(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j}) + \mathcal{A}^m(\boldsymbol{p}_{\neg j}, \boldsymbol{x}_{\neg j}) \quad \forall j \in [d].$$

B.2. Efficient computation when $m \in \{2, 3\}$

Using the multinomial theorem, we can expand the homogeneous polynomial kernel as

$$\mathcal{H}^m(\boldsymbol{p}, \boldsymbol{x}) = \langle \boldsymbol{p}, \boldsymbol{x} \rangle^m = \sum_{k_1 + \cdots + k_d = m} \binom{m}{k_1, \ldots, k_d} \prod_{j=1}^{d} (p_j x_j)^{k_j} \tag{16}$$

where

$$\binom{m}{k_1, \ldots, k_d} \coloneqq \frac{m!}{k_1! \ldots k_d!}$$

is the multinomial coefficient and $k_j \in \{0, 1, \ldots, m\}$. Intuitively, $\binom{m}{k_1, \ldots, k_d}$ is the weight of the monomial $(p_1 x_1)^{k_1} \ldots (p_d x_d)^{k_d}$ in the expansion. For instance, if $\boldsymbol{p}, \boldsymbol{x} \in \mathbb{R}^3$, then the weight of $p_1 x_1 p_3^2 x_3^2$ is $\binom{3}{1,0,2} = 3$. The main observation is that monomials where all $k_1, \ldots, k_d$ are in $\{0, 1\}$ correspond to monomials of (3). If we can compute all other monomials efficiently, then we just need to subtract them from the homogeneous kernel in order to obtain (3).

To simplify notation, we define the shorthands

$$\rho_j \coloneqq p_j x_j, \quad \mathcal{D}^m(\boldsymbol{p}, \boldsymbol{x}) \coloneqq \sum_{j=1}^{d} \rho_j^m \quad \text{and} \quad \mathcal{D}^{m,n}(\boldsymbol{p}, \boldsymbol{x}) \coloneqq \mathcal{D}^m(\boldsymbol{p}, \boldsymbol{x}) \mathcal{D}^n(\boldsymbol{p}, \boldsymbol{x}).$$

Case $m = 2$

For $m = 2$, the possible monomials are of the form ρ_j^2 for all j and $\rho_i \rho_j$ for $j > i$. Applying (16), we obtain

$$\begin{aligned}
\mathcal{H}^2(\boldsymbol{p}, \boldsymbol{x}) &= \sum_{j=1}^{d} \rho_j^2 + 2 \sum_{j > i} \rho_i \rho_j \\
&= \mathcal{D}^2(\boldsymbol{p}, \boldsymbol{x}) + 2\mathcal{A}^2(\boldsymbol{p}, \boldsymbol{x})
\end{aligned}$$

and therefore

$$\mathcal{A}^2(\boldsymbol{p}, \boldsymbol{x}) = \frac{1}{2} \left[\mathcal{H}^2(\boldsymbol{p}, \boldsymbol{x}) - \mathcal{D}^2(\boldsymbol{p}, \boldsymbol{x}) \right].$$

This formula was already mentioned in (Stitson et al., 1997). It was also rediscovered in (Rendle, 2010; 2012), although the connection with the ANOVA kernel was not identified.

Case $m = 3$

For $m = 3$, the possible monomials are of the form ρ_j^3 for all j, $\rho_i \rho_j^2$ for $i \neq j$ and $\rho_i \rho_j \rho_k$ for $k > j > i$. Applying (16), we obtain

$$\begin{aligned}
\mathcal{H}^3(\boldsymbol{p}, \boldsymbol{x}) &= \sum_{j=1}^{d} \rho_j^3 + 3 \sum_{i \neq j} \rho_i \rho_j^2 + 6 \sum_{k > i > i} \rho_i \rho_j \rho_k \\
&= \mathcal{D}^3(\boldsymbol{p}, \boldsymbol{x}) + 3 \sum_{i \neq j} \rho_i \rho_j^2 + 6\mathcal{A}^3(\boldsymbol{p}, \boldsymbol{x}).
\end{aligned}$$

We can compute the second term efficiently by using

$$\begin{aligned}
\sum_{i \neq j} \rho_i \rho_j^2 &= \sum_{i,j=1}^{d} \rho_i \rho_j^2 - \sum_{j=1}^{d} \rho_j^3 \\
&= \mathcal{D}^{2,1}(\boldsymbol{p}, \boldsymbol{x}) - \mathcal{D}^3(\boldsymbol{p}, \boldsymbol{x}).
\end{aligned}$$

We therefore obtain

$$\begin{aligned}
\mathcal{A}^3(\boldsymbol{p}, \boldsymbol{x}) &= \frac{1}{6} \left[\mathcal{H}^3(\boldsymbol{p}, \boldsymbol{x}) - \mathcal{D}^3(\boldsymbol{p}, \boldsymbol{x}) - 3 \left(\mathcal{D}^{2,1}(\boldsymbol{p}, \boldsymbol{x}) - \mathcal{D}^3(\boldsymbol{p}, \boldsymbol{x}) \right) \right] \\
&= \frac{1}{6} \left[\mathcal{H}^3(\boldsymbol{p}, \boldsymbol{x}) - 3\mathcal{D}^{2,1}(\boldsymbol{p}, \boldsymbol{x}) + 2\mathcal{D}^3(\boldsymbol{p}, \boldsymbol{x}) \right].
\end{aligned}$$

B.3. Proof of multi-convexity (Theorem 1)

Let us denote the rows of $\boldsymbol{P}$ by $\bar{\boldsymbol{p}}_1, \ldots, \bar{\boldsymbol{p}}_d \in \mathbb{R}^k$. Using Lemma 2, we know that there exists constants a_s and b_s such that for all $j \in [d]$

$$\begin{aligned}
\hat{y}_{\mathcal{A}^m}(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{P}) &= \sum_{s=1}^{k} \lambda_s \mathcal{A}^m(\boldsymbol{p}_s, \boldsymbol{x}) \\
&= \sum_{s=1}^{k} \lambda_s (p_{js} x_j a_s + b_s) \\
&= \sum_{s=1}^{k} p_{js} \lambda_s x_j a_s + \text{const} \\
&= \langle \bar{\boldsymbol{p}}_j, \bar{\boldsymbol{\mu}}_j \rangle + \text{const} \quad \text{where} \quad \bar{\boldsymbol{\mu}}_j := [\lambda_1 x_j a_1, \ldots, \lambda_k x_j a_k]^{\mathrm{T}}.
\end{aligned}$$

Hence $\hat{y}_{\mathcal{A}^m}(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{P})$ is an affine function of $\bar{\boldsymbol{p}}_1, \ldots, \bar{\boldsymbol{p}}_d$. The composition of a convex loss function and an affine function is convex. Therefore, (4) is convex in $\bar{\boldsymbol{p}}_j \ \forall j \in [d]$. Convexity w.r.t. $\boldsymbol{\lambda}$ is obvious.

C. Proof of equivalence between regularized problems (Theorem 2)

First, we are going to prove that the optimal solution of the nuclear norm penalized problem is a symmetric matrix. For that, we need the following lemma.

Lemma 7 *Upper-bound on nuclear norm of symmetrized matrix*

$$\|\mathcal{S}(\boldsymbol{M})\|_* \leq \|\boldsymbol{M}\|_* \quad \forall \boldsymbol{M} \in \mathbb{R}^{d^2}$$

Proof.

$$\begin{aligned}
\|\mathcal{S}(\boldsymbol{M})\|_* &= \|\frac{1}{2}(\boldsymbol{M} + \boldsymbol{M}^{\mathrm{T}})\|_* \\
&= \frac{1}{2}(\|\boldsymbol{M} + \boldsymbol{M}^{\mathrm{T}}\|_*) \\
&\leq \frac{1}{2}(\|\boldsymbol{M}\|_* + \|\boldsymbol{M}^{\mathrm{T}}\|_*) \\
&= \|\boldsymbol{M}\|_*,
\end{aligned}$$

with equality in the third line holding if and only if $\boldsymbol{M} = \boldsymbol{M}^{\mathrm{T}}$. The second and third lines use absolute homogeneity and subadditivity, two properties that matrix norms satisfy. The last line uses the fact that $\|\boldsymbol{M}\|_* = \|\boldsymbol{M}^{\mathrm{T}}\|_*$. □

Lemma 8 *Symmetry of optimal solution of nuclear norm penalized problem*

$$\operatorname*{argmin}_{\boldsymbol{M} \in \mathbb{R}^{d^2}} \bar{L}_{\mathcal{K}}(\boldsymbol{M}) := L_{\mathcal{K}}(\mathcal{S}(\boldsymbol{M})) + \beta \|\boldsymbol{M}\|_* \in \mathbb{S}^{d^2}$$

Proof. From any (possibly asymmetric) square matrix $\boldsymbol{A} \in \mathbb{R}^{d^2}$, we can construct $\boldsymbol{M} = \mathcal{S}(\boldsymbol{A})$. We obviously have $L_{\mathcal{K}}(\mathcal{S}(\boldsymbol{A})) = L_{\mathcal{K}}(\mathcal{S}(\boldsymbol{M}))$. Combining this with Lemma 7, we have that $\bar{L}_{\mathcal{K}}(\boldsymbol{M}) \leq \bar{L}_{\mathcal{K}}(\boldsymbol{A})$. Therefore we can always achieve the smallest objective value by choosing a symmetric matrix. □

Next, we recall the variational formulation of the nuclear norm based on the SVD.

Lemma 9 *Variational formulation of nuclear norm based on SVD*

$$\|\boldsymbol{M}\|_* = \min_{\substack{\boldsymbol{U}, \boldsymbol{V} \\ \boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}}} \frac{1}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) \quad \forall \boldsymbol{M} \in \mathbb{R}^{d^2} \tag{17}$$

Table 1. Examples of convex loss functions. We defined $\tau = \frac{1}{1+e^{-y\hat{y}}}$.

Loss	Domain of y	$\ell(y, \hat{y})$	$\ell'(y, \hat{y})$	$\ell''(y, \hat{y})$	μ
Squared	$\mathbb{R}$	$\frac{1}{2}(\hat{y} - y)^2$	$\hat{y} - y$	1	1
Squared hinge	$\{-1, 1\}$	$\max(1 - y\hat{y}, 0)^2$	$-2y \max(1 - y\hat{y}, 0)$	$2\delta_{[\hat{y}y<1]}$	2
Logistic	$\{-1, 1\}$	$\log(\tau^{-1})$	$y(\tau - 1)$	$\tau(1 - \tau)$	$\frac{1}{4}$

The minimum above is attained at $\|\boldsymbol{M}\|_* = \frac{1}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2)$, *where* $\boldsymbol{U} \in \mathbb{R}^{d\times r}$ *and* $\boldsymbol{V} \in \mathbb{R}^{d\times r}$, $r = \operatorname{rank}(\boldsymbol{M})$, *are formed from the reduced SVD of* $\boldsymbol{M}$, *i.e.,* $\boldsymbol{U} = \boldsymbol{A} \operatorname{diag}(\boldsymbol{\sigma})^{\frac{1}{2}}$ *and* $\boldsymbol{V} = \boldsymbol{B} \operatorname{diag}(\boldsymbol{\sigma})^{\frac{1}{2}}$ *where* $\boldsymbol{M} = \boldsymbol{A} \operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{B}^{\mathrm{T}}$.

For a proof, see for instance (Mazumder et al., 2010, Section A.5).

Now, we give a specialization of the above for symmetric matrices, based on the eigendecomposition instead of SVD.

Lemma 10 *Variational formulation of nuclear norm based on eigendecomposition*

$$\|\boldsymbol{M}\|_* = \min_{\substack{\boldsymbol{\lambda}, \boldsymbol{P} \\ \boldsymbol{M}=\boldsymbol{P}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{P}^{\mathrm{T}}}} \sum_{s=1}^{k} |\lambda_s| \, \|\boldsymbol{p}_s\|^2 \quad \forall \boldsymbol{M} \in \mathbb{S}^{d^2}, \tag{18}$$

where $k = \operatorname{rank}(\boldsymbol{M})$. *The minimum above is attained by the reduced eigendecomposition* $\boldsymbol{M} = \boldsymbol{P} \operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{P}^{\mathrm{T}}$ *and* $\|\boldsymbol{M}\|_* = \|\boldsymbol{\lambda}\|_1$.

Proof. Let $\boldsymbol{A} \operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{B}^{\mathrm{T}}$ and $\boldsymbol{P} \operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{P}^{\mathrm{T}}$ be the reduced SVD and eigendecomposition of $\boldsymbol{M} \in \mathbb{S}^{d^2}$, respectively. The relation between the SVD and the eigendecomposition is given by

$$\begin{aligned} \sigma_s &= |\lambda_s| \\ \boldsymbol{a}_s &= \operatorname{sign}(\lambda_s)\boldsymbol{p}_s \\ \boldsymbol{b}_s &= \boldsymbol{p}_s. \end{aligned}$$

From Lemma 9, we therefore obtain

$$\begin{aligned} \boldsymbol{u}_s &= \sqrt{\sigma_s}\boldsymbol{a}_s = \sqrt{|\lambda_s|}\operatorname{sign}(\lambda_s)\boldsymbol{p}_s \\ \boldsymbol{v}_s &= \sqrt{\sigma_s}\boldsymbol{b}_s = \sqrt{|\lambda_s|}\boldsymbol{p}_s. \end{aligned}$$

Now, computing $\frac{1}{2}(\sum_s \|\boldsymbol{u}_s\|^2 + \|\boldsymbol{v}_s\|^2)$ gives $\sum_{s=1}^k |\lambda_s| \, \|\boldsymbol{p}_s\|^2$. The minimum value $\|\boldsymbol{M}\|_* = \|\boldsymbol{\lambda}\|_1$ follows from the fact that $\boldsymbol{P}$ is orthonormal and hence $\|\boldsymbol{p}_s\|^2 = 1 \; \forall s \in [k]$. $\square$

We now have all the tools to prove our result. The equivalence between (12) and (14) when $r = \operatorname{rank}(\boldsymbol{M}^*)$ is a special case of (Mazumder et al., 2010, Theorem 3). From Lemma 8, we know that the optimal solution of (14) is symmetric. This allows us to substitute (17) with (18), and therefore, (13) is equivalent to (14) with $k = \operatorname{rank}(\boldsymbol{M}^*)$. As discussed in (Mazumder et al., 2010), the result also holds when $r = k$ is larger than $\operatorname{rank}(\boldsymbol{M}^*)$.

D. Efficient coordinate descent algorithms

D.1. Direct approach, $\mathcal{K} = \mathcal{A}^m$ for $m \in \{2, 3\}$

As stated in Theorem 1, the direct optimization objective is multi-convex when $\mathcal{K} = \mathcal{A}^m$. This allows us to easily minimize the objective by solving a succession of coordinate-wise convex problems. In this section, we develop an efficient algorithm for minimizing (13) with $m \in \{2, 3\}$. It is easy to see that minimization w.r.t. $\boldsymbol{\lambda}$ can be reduced to a standard ℓ_1-regularized convex objective via a simple change of variable. We therefore focus our attention to minimization w.r.t. $\boldsymbol{P}$.

As a reminder, we want to minimize

$$f \coloneqq \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \beta \sum_{s=1}^{k} |\lambda_s| \|\boldsymbol{p}_s\|^2$$

where

$$\hat{y}_i \coloneqq \sum_{s=1}^{k} \lambda_s \mathcal{A}^m(\boldsymbol{p}_s, \boldsymbol{x}_i).$$

After routine calculation, we obtain

$$\begin{aligned}
\frac{\partial \mathcal{A}^2(\boldsymbol{p}_s, \boldsymbol{x}_i)}{\partial p_{js}} &= \langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle x_{ji} - p_{js} x_{ji}^2 = (\langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle - p_{js} x_{ji}) x_{ji} \\
\frac{\partial \mathcal{A}^3(\boldsymbol{p}_s, \boldsymbol{x}_i)}{\partial p_{js}} &= \frac{1}{2} \langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle^2 x_{ji} - p_{js} x_{ji}^2 \langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle - \frac{1}{2} x_{ji} \mathcal{D}^2(\boldsymbol{p}_s, \boldsymbol{x}_i) + p_{js}^2 x_{ji}^3 \\
&= \mathcal{A}^2(\boldsymbol{p}_s, \boldsymbol{x}_i) x_{ji} - p_{js} x_{ji}^2 \langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle + p_{js}^2 x_{ji}^3 \\
\frac{\partial \mathcal{A}^m(\boldsymbol{p}_s, \boldsymbol{x}_i)}{\partial p_{js}^2} &= 0 \quad \forall m \in \mathbb{N} \\
\frac{\partial \hat{y}_i}{\partial p_{js}} &= \lambda_s \frac{\partial \mathcal{A}^m(\boldsymbol{p}_s, \boldsymbol{x}_i)}{\partial p_{js}} \\
\frac{\partial \hat{y}_i}{\partial p_{js}^2} &= 0 \quad \forall j \in [d], s \in [k].
\end{aligned}$$

The fact that the second derivative is null is a consequence of the multi-linearity of $\mathcal{A}^m$.

Using the chain rule, we then obtain

$$\begin{aligned}
\frac{\partial f}{\partial p_{js}} &= \sum_{i=1}^{n} \ell'(y_i, \hat{y}_i) \frac{\partial \hat{y}_i}{\partial p_{js}} + 2\beta |\lambda_s| p_{js} \\
\frac{\partial f}{\partial p_{js}^2} &= \sum_{i=1}^{n} \left[\ell''(y_i, \hat{y}_i) \left(\frac{\partial \hat{y}_i}{\partial p_{js}} \right)^2 + \ell'(\hat{y}_i, y_i) \frac{\partial \hat{y}_i}{\partial p_{js}^2} \right] + 2\beta |\lambda_s| \\
&= \sum_{i=1}^{n} \ell''(y_i, \hat{y}_i) \left(\frac{\partial \hat{y}_i}{\partial p_{js}} \right)^2 + 2\beta |\lambda_s|.
\end{aligned}$$

Assuming that ℓ is μ-smooth, its second derivative is upper-bounded by μ and therefore we have

$$\frac{\partial f}{\partial p_{js}^2} \leq \eta_{js} \quad \text{where} \quad \eta_{js} \coloneqq \mu \sum_{i=1}^{n} \left(\frac{\partial \hat{y}_i}{\partial p_{js}} \right)^2 + 2\beta |\lambda_s|.$$

Then the update

$$p_{js} \leftarrow p_{js} - \eta_{js}^{-1} \frac{\partial f}{\partial p_{js}}$$

guarantees that the objective value is monotonically decreasing except at the coordinate-wise minimum. Note that in the case of the squared loss $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$, the above update is equivalent to a Newton step and is the exact minimizer of the coordinate-wise objective. An epoch consists in updating all variables once, for instance in cyclic order.

For an efficient implementation, we need to maintain $\hat{y}_i \; \forall i \in [n]$ and statistics that depend on $\boldsymbol{p}_s$. For the former, we need $O(n)$ memory. For the latter, we need $O(kmn)$ memory for an implementation with full cache. However, this requirement is not realistic for a large training set. In practice, the memory requirement can be reduced to $O(mn)$ if we recompute the quantities then sweep through $p_{1s}, \ldots, p_{ds}$ for s fixed. Overall the cost of one epoch is $O(k n_z(\boldsymbol{X}))$. A similar implementation technique is described for factorization machines with $m = 2$ in (Rendle, 2012).

D.2. Lifted approach, $\mathcal{K} = \mathcal{H}^m$

We present an efficient coordinate descent solver for the lifted approach with $\mathcal{K} = \mathcal{H}^m$, for arbitrary integer $m \geq 2$. Recall that our goal is to learn $\boldsymbol{\mathcal{W}} = \mathcal{S}(\boldsymbol{\mathcal{M}}) \in \mathbb{S}^{d^m}$ by factorizing $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{d^m}$ using m matrices of size $d \times r$. Let us call these

Algorithm 1 CD algorithm for direct obj. with $\mathcal{K} = \mathcal{A}^{\{2,3\}}$

```
Input: $\boldsymbol{\lambda}$, initial $\boldsymbol{P}$, $\mu$-smooth loss function $\ell$, regularization parameter $\beta$, number of bases $k$, degree $m$, tolerance $\epsilon$
Pre-compute $\hat{y}_i := \hat{y}_{\mathcal{A}^m}(\boldsymbol{x}_i; \boldsymbol{\lambda}, \boldsymbol{P})\ \forall i \in [n]$
Set $\Delta \leftarrow 0$
for $s := 1, \ldots, k$ do
  Pre-compute $\langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle$ and $\mathcal{A}^2(\boldsymbol{p}_s, \boldsymbol{x}_i)\ \forall i \in [n]$
  for $j := 1, \ldots, d$ do
    Compute inv. step size $\eta := \mu \sum_{i=1}^n \left(\frac{\partial \hat{y}_i}{\partial p_{js}}\right)^2 + 2\beta|\lambda_s|$
    Compute $\delta := \eta^{-1} \left[\sum_{i=1}^n \ell'(y_i, \hat{y}_i) \frac{\partial \hat{y}_i}{\partial p_{js}} + 2\beta|\lambda_s| p_{js}\right]$
    Update $p_{js} \leftarrow p_{js} - \delta$; Set $\Delta \leftarrow \Delta + |\delta|$
    Synchronize $\hat{y}_i$, $\langle \boldsymbol{p}_s, \boldsymbol{x}_i \rangle$ and $\mathcal{A}^2(\boldsymbol{p}_s, \boldsymbol{x}_i)\ \forall i$ s.t. $x_{ji} \neq 0$
  end for
end for
If $\Delta \leq \epsilon$ stop, otherwise repeat
Output: $\boldsymbol{P}$
```

Algorithm 2 CD algorithm for lifted objective with $\mathcal{K} = \mathcal{H}^m$

```
Input: initial $\{\boldsymbol{U}^t\}_{t=1}^m$, $\mu$-smooth loss function $\ell$, regularization parameter $\beta$, rank $r$, degree $m$, tolerance $\epsilon$
Pre-compute $\hat{y}_i := \sum_{s=1}^r \prod_{t=1}^m \langle \boldsymbol{u}_s^t, \boldsymbol{x}_i \rangle\ \forall i \in [n]$
Set $\Delta \leftarrow 0$
for $t := 1, \ldots, m$ and $s := 1, \ldots, r$ do
  Pre-compute $\xi_i := \prod_{t' \neq t} \langle \boldsymbol{u}_s^{t'}, \boldsymbol{x}_i \rangle\ \forall i \in [n]$
  for $j := 1, \ldots, d$ do
    Compute inv. step size $\eta := \mu \sum_{i=1}^n \xi_i^2 x_{ji}^2 + \beta$
    Compute $\delta := \eta^{-1} \left[\sum_{i=1}^n \ell'(y_i, \hat{y}_i) \xi_i x_{ji} + \beta u_{js}^t\right]$
    Update $u_{js}^t \leftarrow u_{js}^t - \delta$; Set $\Delta \leftarrow \Delta + |\delta|$
    Synchronize $\hat{y}_i\ \forall i$ s.t. $x_{ji} \neq 0$
  end for
end for
If $\Delta \leq \epsilon$ stop, otherwise repeat
Output: $\{\boldsymbol{U}^t\}_{t=1}^m$
```

matrices $\boldsymbol{U}^1, \ldots, \boldsymbol{U}^m$ and their columns $\boldsymbol{u}_s^t = [u_{1s}^t, \ldots, u_{ds}^t]^{\mathrm{T}}$ with $t \in [m]$ and $s \in [r]$. The decomposition of $\boldsymbol{\mathcal{M}}$ can be expressed as a sum of rank-one tensors

$$\boldsymbol{\mathcal{M}} = \sum_{s=1}^r \boldsymbol{u}_s^1 \otimes \cdots \otimes \boldsymbol{u}_s^m.$$

Using (11) we obtain

$$\hat{y}_i := \langle \boldsymbol{\mathcal{W}}, \boldsymbol{x}_i^{\otimes m} \rangle = \langle \boldsymbol{\mathcal{M}}, \boldsymbol{x}_i^{\otimes m} \rangle = \sum_{s=1}^r \prod_{t=1}^m \langle \boldsymbol{u}_s^t, \boldsymbol{x}_i \rangle.$$

The first and second coordinate-wise derivatives are given by

$$\frac{\partial \hat{y}_i}{\partial u_{js}^t} = \prod_{t' \neq t} \langle \boldsymbol{u}_s^{t'}, \boldsymbol{x}_i \rangle x_{ji} \quad \text{and} \quad \frac{\partial \hat{y}_i}{\partial (u_{js}^t)^2} = 0.$$

We consider the following regularized objective function

$$f := \sum_{i=1}^n \ell(y_i, \hat{y}_i) + \frac{\beta}{2} \sum_{t=1}^m \sum_{s=1}^r \|\boldsymbol{u}_s^t\|^2.$$

Using the chain rule, we obtain

$$\frac{\partial f}{\partial u_{js}^t} = \sum_{i=1}^n \ell'(y_i, \hat{y}_i) \frac{\partial \hat{y}_i}{\partial u_{js}^t} + \beta u_{js}^t \quad \text{and} \quad \frac{\partial f}{\partial (u_{js}^t)^2} = \sum_{i=1}^n \ell''(y_i, \hat{y}_i) \left(\frac{\partial \hat{y}_i}{\partial u_{js}^t}\right)^2 + \beta.$$

Assuming that ℓ is μ-smooth, its second derivative is upper-bounded by μ and therefore we have

$$\frac{\partial f}{\partial (u_{js}^t)^2} \leq \eta_{js}^t \quad \text{where} \quad \eta_{js}^t := \mu \sum_{i=1}^n \left(\frac{\partial \hat{y}_i}{\partial u_{js}^t}\right)^2 + \beta.$$

Then the update

$$u_{js}^t \leftarrow u_{js}^t - (\eta_{js}^t)^{-1} \frac{\partial f}{\partial u_{js}^t}$$

guarantees that the objective value is monotonically decreasing, except at the coordinate-wise minimum. Note that in the case of the squared loss $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$, the above update is equivalent to a Newton step and is the exact minimizer of the coordinate-wise objective. An epoch consists in updating all variables once, for instance in cyclic order.

For an efficient implementation, the two quantities we need to maintain are $\hat{y}_i \; \forall i \in [n]$ and $\prod_{t' \neq t} \langle \boldsymbol{u}_s^{t'}, \boldsymbol{x}_i \rangle \; \forall i \in [n], s \in [r], t \in [m]$. For the former, we need $O(n)$ memory. For the latter, we need $O(rmn)$ memory for an implementation with full cache. However, this requirement is not realistic for a large training set. In practice, the memory requirement can be reduced to $O(mn)$ if we recompute the quantity then sweep through $u_{1s}^t, \dots, u_{ds}^t$ for t and s fixed. Overall the cost of one epoch is $O(mrn_z(\boldsymbol{X}))$.

D.3. Lifted approach, $\mathcal{K} = \mathcal{A}^2$

For $\langle \cdot, \cdot \rangle_>$, efficient computations are more involved since we need to ignore irrelevant monomials. Nevertheless, we can also compute the predictions directly without explicitly symmetrizing the model. For $m = 2$, it suffices to subtract the effect of squared features. It is easy to verify that we then obtain

$$\langle \mathcal{S}(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}), \boldsymbol{x}^{\otimes 2} \rangle_> = \frac{1}{2} \left[\langle \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}, \boldsymbol{V}^{\mathrm{T}} \boldsymbol{x} \rangle - \sum_{s=1}^{r} \langle \boldsymbol{u}_s \circ \boldsymbol{x}, \boldsymbol{v}_s \circ \boldsymbol{x} \rangle \right],$$

where $\circ$ indicates element-wise product. The coordinate-wise derivatives are given by

$$\frac{\partial y_i}{\partial u_{js}} = \frac{1}{2} \left[\langle \boldsymbol{v}_s, \boldsymbol{x} \rangle x_{ji} - v_{js} x_{ji}^2 \right] \quad \text{and} \quad \frac{\partial y_i}{\partial v_{js}} = \frac{1}{2} \left[\langle \boldsymbol{u}_s, \boldsymbol{x} \rangle x_{ji} - u_{js} x_{ji}^2 \right].$$

Generalizing this to arbitrary m is a future work.

E. Datasets

For regression experiments, we used the following public datasets.

Dataset	n (train)	n (test)	d	Description
abalone	3,132	1,045	8	Predict the age of abalones from physical measurements
cadata	15,480	5,160	8	Predict housing prices from economic covariates
cpusmall	6,144	2,048	12	Predict a computer system activity from system performance measures
diabetes	331	111	10	Predict disease progression from baseline measurements
E2006-tfidf	16,087	3,308	150,360	Predict volatility of stock returns from company financial reports

The *diabetes* dataset is available in scikit-learn (Pedregosa et al., 2011). Other datasets are available from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

For recommender system experiments, we used the following two public datasets.

Dataset	n	d
Movielens 1M	1,000,209 (ratings)	9,940 = 6,040 (users) + 3,900 (movies)
Last.fm	108,437 (tag counts)	24,078 = 12,133 (artists) + 11,945 (tags)

For *Movielens 1M*, the task is to predict ratings between 1 and 5 given by users to movies, i.e., $y \in \{1, \dots, 5\}$. For *Last.fm*, the task is to predict the number of times a tag was assigned to an artist, i.e., $y \in \mathbb{N}$.

The design matrix $\boldsymbol{X}$ was constructed following (Rendle, 2010; 2012). Namely, for each rating y_i, the corresponding $\boldsymbol{x}_i$ is set to the concatenation of the one-hot encodings of the user and item indices. Hence the number of samples n is the number of ratings and the number of features is equal to the sum of the number of users and items. Each sample contains exactly two non-zero features. It is known that factorization machines are equivalent to matrix factorization when using this representation (Rendle, 2010; 2012).

We split samples uniformly at random between 75% for training and 25% for testing.