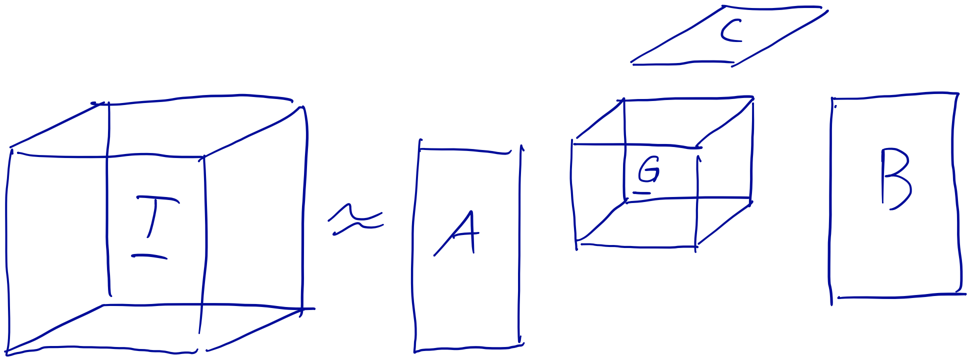
The Tucker Decompositions

Let $\underline{T} \in \mathbb{R}^{I \times J \times K}$. Its Tucker3- or just Tucker-decomposition has a core tensor $\underline{G} \in \mathbb{R}^{P \times Q \times R}$ and thee factor matrices $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$, and $C \in \mathbb{R}^{K \times R}$, and it is defined as

$$\begin{aligned} \underline{T} &\approx \underline{G} \times_1 A \times_2 B \times_3 C \\ &= \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \, \vec{a}_p \circ \vec{b}_q \circ \vec{c}_r \\ &= [\![\underline{G}; A, B, C]\!]. \end{aligned}$$

Hence, elementwise Tucker is

$$t_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \, a_{ip} b_{jq} c_{kr}.$$



Tucker decomposition is usually applied only to 3-way tensors, but N-way version is straight forward to define:

$$\underline{T} \approx \underline{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)}$$

$$t_{i_1 i_2 \cdots i_N} \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N}\, a^{(1)}_{i_1 r_1}\, a^{(2)}_{i_2 r_2} \cdots a^{(N)}_{i_N r_N}$$

The core tensor $\underline{G}$ can be considered as a compressed version of the original tensor $\underline{T}$ if $P<I$, $Q<J$, and $R<K$. If $P=Q=R$ and $\underline{G}$ is (hyper-) diagonal, then Tucker3 reduces to the CP decomposition. In particular, with hyper-diagonal $\underline{G}$ where all diagonal entries are 1, we have that $[\![\underline{G}; A, B, C]\!] = [\![A, B, C]\!]$.

The Tucker decomposition can be expressed in a matricized form:

$$T_{(1)} \approx A\, G_{(1)} (C \otimes B)^T$$
$$T_{(2)} \approx B\, G_{(2)} (C \otimes A)^T$$
$$T_{(3)} \approx C\, G_{(3)} (B \otimes A)^T .$$

To gain some intuition, let's consider the first frontal slice of $[\![\underline{G}; A, B, C]\!]$.

$$[\![\underline{G}; A, B, C]\!]_1 = A D^{(1)} B^T, \quad \text{with } D^{(1)} = \sum_{r=1}^{R} c_{1r} G_r .$$

Now we have

$$
\begin{aligned}
T_{(1)} &= A \left[D^{(1)} B^T \; D^{(2)} B^T \; \cdots \; D^{(K)} B^T \right] \\
&= A \left[\left(\sum_r c_{1r} G_r\right) B^T \cdots \left(\sum_r c_{kr} G_r\right) B^T \right] \\
&= A \left[G_1 \; G_2 \; \cdots \; G_K \right] \begin{pmatrix} c_{11} B^T & c_{21} B^T & \cdots & c_{K1} B^T \\ c_{12} B^T & c_{22} B^T & \cdots & c_{K2} B^T \\ \vdots & \vdots & \ddots & \vdots \\ c_{1R} B^T & c_{2R} B^T & \cdots & c_{KR} B^T \end{pmatrix} \\
&= A\, G_{(1)} (C \otimes B)^T
\end{aligned}
$$

In N-way case, the mode-n matricized version becomes

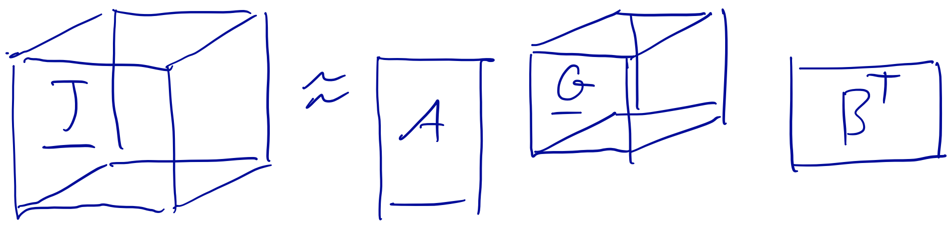
$$T_{(n)} \approx A^{(n)} G_{(n)} \left(A^{(N)} \otimes \cdots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)}\right)^T$$

Tucker1 and Tucker2

The Tucker2 decomposition leaves one factor matrix as an identity matrix, e.g.

$$\underline{T} \approx \underline{G} \times_1 A \times_2 B = [\![\underline{G}; A, B, I]\!],$$

where $\underline{T} \in \mathbb{R}^{I \times J \times K}$ and $\underline{G} \in \mathbb{R}^{P \times Q \times K}$



The Tucker1 decomposition leaves two factor matrices as identity, for instance

$$\underline{T} \approx \underline{G} \times_1 A = [\![\underline{G}; A, I, I]\!].$$

This is equivalent to standard least-squares matrix factorization.

The n-rank

Let $\underline{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_N}$. The n-rank of $\underline{T}$, $\text{rank}_n(\underline{T})$, is the column rank of $T_{(n)}$, i.e., the number of linearly independent columns in $T_{(n)}$. If we set $R_n = \text{rank}_n(\underline{T})$, then $\underline{T}$ is rank-$(R_1, R_2, \ldots, R_N)$, though note that this definition is not compatible with the usual tensor rank.

Clearly, $R_n \leq I_n$ for all $n \in [N]$. Finding a Tucker decomposition of size $(R_1, R_2, \ldots, R_N)$ is easy if $\text{rank}_n(\underline{T}) = R_n$ for all n.

Computing Tucker

From now on, we enforce that the columns of the factor matrices are mutually orthogonal, that is,

$$A^T A = I_P, \quad B^T B = I_Q, \text{ and } C^T C = I_R.$$

Higher-order SVD (HOSVD)

HOSVD is a simple method to calculate the Tucker decomposition using SVD:

$$
\begin{aligned}
&\underline{\text{for}}\ n = 1, \dots, N\ \underline{\text{do}} \\
&\quad (U, \Sigma, V) \leftarrow \text{SVD}(T_{(n)}) \\
&\quad A^{(n)} \leftarrow U(i, 1{:}R_n) \\
&\underline{\text{end}} \\
&\underline{G} \leftarrow \underline{T} \times_1 A^{(1)T} \times_2 A^{(2)T} \times_3 \cdots \times_N A^{(N)T}
\end{aligned}
$$

In the last row above, we use the fact that the factor matrices are column-orthogonal:

$$
\underline{T} = \underline{G} \times_1 A \iff T_{(1)} = A\, G_{(1)} \iff A^{-1} T_{(1)} = G_{(1)}
$$

$$
\overset{*}{\iff} A^T T_{(1)} = G_{(1)} \iff \underline{T} \times_1 A^T = G_{(1)},
$$

where * is due to the column orthogonality.

HOSVD is not very good for finding a low-error decomposition, as it doesn't take into account the higher-order structure in the tensor. It is, however, useful for finding an initial solution for other algorithms.

Higher-order Orthogonal Iteration (HOOI)

We know that

$$\underline{G} = \underline{T} \times_1 A^T \times_2 B^T \times_3 C^T \qquad (x)$$

is optimal for column-orthogonal A, B, and C.

Let us re-write the objective:

$$\|\underline{T} - [\![\underline{G}; A, B, C]\!]\|^2 = \|\underline{T}\|^2 - 2\langle \underline{T}, [\![\underline{G}; A, B, C]\!]\rangle + \|[\![\underline{G}; A, B, C]\!]\|^2$$

$$\overset{*}{=} \|\underline{T}\|^2 - 2\langle \underline{T}, [\![\underline{G}; A, B, C]\!]\rangle + \|\underline{G}\|^2$$

$$\overset{**}{=} \|\underline{T}\|^2 - 2\langle \underline{T} \times_1 A^T \times_2 B^T \times_3 C^T, \underline{G}\rangle + \|\underline{G}\|^2$$

$$\overset{***}{=} \|\underline{T}\|^2 - 2\langle \underline{G}, \underline{G}\rangle + \|\underline{G}\|^2$$

$$\overset{****}{=} \|\underline{T}\|^2 - \|\underline{G}\|^2$$

$$= \|\underline{T}\|^2 - \|\underline{T} \times_1 A^T \times_2 B^T \times_3 C^T\|^2.$$

* We use the column orthogonality

** We multiply the factors out from $[\![\underline{G}; A, B, C]\!]$.

*** We use (x)

**** We use $\langle \underline{G}, \underline{G}\rangle = \|\underline{G}\|^2$

As $\|\underline{T}\|$ is constant, we learn that

$$\|\underline{T} - [\![\underline{G}; A, B, C]\!]\|^2 \propto -\|\underline{T} \times_1 A^T \times_2 B^T \times_3 C^T\|^2$$

Hence, we want to maximize

$$\max_{A,B,C} \| \underline{T} \times_1 A^T \times_2 B^T \times_3 C^T \|,$$

or equivalently

$$\max_{A,B,C} \| A^T T_{(1)} (C \otimes B)^T \|$$

$$\max_{A,B,C} \| B^T T_{(2)} (C \otimes A)^T \|$$

$$\max_{A,B,C} \| C^T T_{(3)} (B \otimes A)^T \|.$$

Essentially, $\underline{G}$ is fully determined by the factor matrices, and we do not have to take it into account.

To compute HOOI, we again use SVD:

Initialize $A^{(1)}, A^{(2)}, \dots$ using HOSVD
repeat
  for $n = 1, \dots, N$ **do**
    $\underline{Y} \leftarrow \underline{T} \times_1 A^{(1)T} \times_2 \cdots \times_N A^{(N)T}$
    $U, \Sigma, V \leftarrow \mathrm{SVD}(Y_{(n)})$
    $A^{(n)} \leftarrow U(:, 1{:}R_n)$
  end
until convergence
$\underline{G} \leftarrow \underline{T} \times_1 A^{(1)T} \times_2 \cdots \times_N A^{(N)T}$

Naive ALS for Tucker

We can also use the standard ALS approach to solve Tucker decomposition. This does not, generally yield to orthogonal factor matrices, though. In some cases, this is what we want. To update the factor matrices, we have:

$$A \leftarrow T_{(1)} \left(G_{(1)} (C \otimes B)^T \right)^+$$
$$B \leftarrow T_{(2)} \left(G_{(2)} (C \otimes A)^T \right)^+$$
$$C \leftarrow T_{(3)} \left(G_{(3)} (B \otimes A)^T \right)^+$$

To update the core, we can use a vectorized format of Tucker and solve

$$\underline{G} = \arg\min_{\underline{G}} \| \mathrm{vec}(\underline{T}) - (C \otimes B \otimes A)\,\mathrm{vec}(\underline{G}) \|,$$

which is just a standard least-squares problem.

The ALS algorithm is rarely used due to the large pseudo-inverse.

It can be useful for computing the Tucker2 decomposition, though. For Tucker2, we replace C with the identity matrix, and solve

$$A \leftarrow T_{(1)} \left(G_{(1)} (I \otimes B)^T \right)^+$$
$$B \leftarrow T_{(2)} \left(G_{(2)} (I \otimes A)^T \right)^+,$$

where we can use the fact that

$$(I \otimes B)^T = \begin{pmatrix} B^T & & & \\ & B^T & & \\ & & \ddots & \\ & & & B^T \end{pmatrix}$$

and that

$$G_{(1)} (I \otimes B)^T = [G_1 \, G_2 \cdots G_k] (I \otimes B)^T = \begin{pmatrix} G_1 B^T & & & \\ & G_2 B^T & & \\ & & \ddots & \\ & & & G_k B^T \end{pmatrix}$$

The core can be updated separately for each frontal slice:

$$G_k \leftarrow B^+ T_k A^+.$$

Nonnegative Tucker3

Similarly to NCP, we can consider the nonnegative variant of the Tucker3 decomposition. We use the unfolded versions of Tucker

$$T_{(1)} \approx A\, G_{(1)} (C \otimes B)^T$$
$$T_{(2)} \approx B\, G_{(2)} (C \otimes A)^T$$
$$T_{(3)} \approx C\, G_{(3)} (B \otimes A)^T,$$

and update the factors using multiplicative rules similar to those in NCP.

To initialize A, B, C, and $\underline{G}$ we can run HOSVD on $\underline{T}$ and truncate the negative values in the result to zero. We can also use random initialization, but this can yield very bad initial solutions.

The nonnegative Tucker3 algorithm is

Input: $\underline{T} \in \mathbb{R}_{\geq 0}^{I_1 \times I_2 \times \cdots \times I_N}$, $(R_1, R_2, \ldots, R_N)$

$(\underline{G}, A^{(1)}, \ldots, A^{(N)}) \leftarrow \mathrm{HOSVD}(\underline{T}, R_1, R_2, \ldots, R_N)$

$\underline{G} \leftarrow [\underline{G}]_+$; $A^{(n)} \leftarrow [A^{(n)}]_+$ for all $n \in [N]$

<u>repeat</u>

$\underline{Q} \leftarrow \underline{G} \times_1 A^{(1)} \times_2 \cdots \times_N A^{(N)}$

<u>for</u> $n = 1 \ldots N$ <u>do</u>

$A^{(n)} \leftarrow A^{(n)} * \left(T_{(n)} A^{\otimes_{-n}} G_n^T \right) \oslash \left(Q_{(n)} A^{\otimes_{-n}} G_{(n)}^T \right)$

$\vec{a}_i^{(n)} \leftarrow \vec{a}_i^{(n)} / \| \vec{a}_i^{(n)} \|$

<u>end</u>

$\underline{G} \leftarrow \underline{G} * \left(\underline{T} \times_1 A^{(1)} \times_2 \cdots \times_N A^{(N)} \right) \oslash \left(Q_{(n)} \times_1 A^{(1)} \times_2 \cdots \times_N A^{(N)} \right)$

<u>until</u> convergence

Here, $A^{\otimes_{-n}} = A^{(N)} \otimes \cdots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \times A^{(1)}$.

Applications of Tucker

TensorFaces

We can use Tucker decomposition to separate the effects of different illuminations, expressions, poses, etc from pictures, provided that we have training data with all variations for all subjects.

In TensorFaces, we have photos of subjects in different views (front, left, right, ...), under different illuminations and having different expressions. This gives a people-by-views-by-illuminations-by-expressions-by-pixels tensor. See the next page for some examples (image from Vasilescu & Terzopoulos: Multilinear analysis of image ensembles: TensorFaces, ECCV '02).

Applying the Tucker decomposition with no reduction in the dimensions yields

$$\underline{G} \times_1 A_{people} \times_2 A_{views} \times_3 A_{illums} \times_4 A_{expres} \times_5 A_{pixels}$$

The $A_{(n)}$ factor matrices encode the variations in these modes, while the core tensor governs the interactions.

This model generalizes Eigenfaces: we can write

$$T_{(pixels)} = A_{pixels}\, G_{(pixels)} \left(A_{express} \otimes A_{illums} \otimes A_{views} \otimes A_{people}\right)^T$$

to get a standard Eigenfaces setting. Multiplying $\underline{G} \times_5 A_{pixels}$ gives us a tensor that shows the primary variations along the modes. Some examples are in the next page (Vasilescu & Terzopoulos, 2002). On the other hand, if we multiply

$$\underline{G} \times_2 A_{views} \times_5 A_{pixels}$$

we get the variations with the different viewpoints, as depicted in the picture in the previous page (Vasilescu & Terzopoulos, 2002).

Finding facts in open IR

In open information retrieval our goal is to extract structured knowledge from unstructured data using unsupervised methods. Typically, we want to extract subject-predicate-object (s, p, o) triples with disambiguated entities and relations. To do that, we can first run standard natural language parsers to obtain noun phrase-verbal phrase-noun phrase (np, vp, np) triples, e.g

Donald J. Trump, is, POTUS
Donald Trump, is the president of, USA
The Donald, is the prez of, 'Murica
Donald Trump, is the son of, POTUS

These triples encode two (s, p, o) triples

donald_j_trump, isPresidentOf, USA

donald_j_trump_jr, isSonOf, donald_j_trump

To extract these, we need to handle synonyms (president vs. prez) and homonyms (Donald Trump [Jr]), among others. We can model this with mappings from noun phrases to entities and from verbal phrases to relations. Considering subjects and objects separately, we have $A: np \to s$, $B: np \to o$, and $C: vp \to p$. We model these as matrices. If they are column orthogonal, and if our original (np, vp, np) triples are stored in a tensor $\underline{T}$, we can get the true (s, p, o) triples as

$$\underline{T} \times_1 A^T \times_2 B^T \times_3 C^T.$$

That is, if we do Tucker decomposition to the tensor containing the surface triples, we find the latent entities and relations (though they might not contain any "real" entities or relations).

A practical problem with this approach is that the core tensor has to be very big, making it hard to work with. The core should also be sparse, which is not the case if we obtain it with the multiplication.