The CP and TUCKER model

Learning objective: *The aim of this exercise is to understand how the alternating least squares algorithms (ALS) for the CP and TUCKER model works as well as to be able to analyze and interpret multi-way data by the models.*

Notation: We first introduce some tensor notation:

$\boldsymbol{X}_{(n)}$	The n-mode matricizing: $\mathcal{X}^{I_1 \times I_2 \times \ldots \times I_N} \rightarrow \boldsymbol{X}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$ The inverse operation is denoted un-matricizing, i.e. $\boldsymbol{X}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N} \rightarrow \mathcal{X}^{I_1 \times I_2 \times \ldots \times I_N}$
$\boldsymbol{A} \otimes \boldsymbol{B}$	The Kronecker product, i.e. $\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & \cdots & a_{1,J}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{I,1}\boldsymbol{B} & \cdots & a_{I,J}\boldsymbol{B} \end{bmatrix}$
$\boldsymbol{A} \odot \boldsymbol{B}$	The Khatri-Rao product $\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 \ \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 \ \ldots \ \boldsymbol{a}_J \otimes \boldsymbol{b}_J]$
$\times_n$	$(\mathcal{Q} \times_n \mathbf{P})_{i_1,i_2,\ldots,j_n,\ldots i_N} = \sum_{i_n} \mathcal{Q}_{i_1,i_2,\ldots,i_n,\ldots i_N} \mathbf{P}_{j_n,i_n}$

These operations are given in the Matlab scripts:

- *matricizing.m*
- *unmatricizing.m*
- *kron.m*
- *krprod.m*
- *tmult.m*

The following relation will turn useful:
$(\boldsymbol{A} \odot \boldsymbol{B})^\top (\boldsymbol{A} \odot \boldsymbol{B}) = \boldsymbol{A}^\top \boldsymbol{A} * \boldsymbol{B}^\top \boldsymbol{B}$ where $*$ denotes element-wise multiplication.

Alternating Least Squares (ALS): Alternating Least Squares is a general framework for optimizing least squares problems where some variables are found keeping other variables fixed. For instance the general problem, $\mathbf{X} \approx \mathbf{AB}$ can be solved by minimizing $\|\mathbf{X} - \mathbf{AB}\|_F^2$ with respect to $\mathbf{A}$ and $\mathbf{B}$. The solution is found by alternatingly updating $\mathbf{A}$ and $\mathbf{B}$ according to:

$$\mathbf{A} \leftarrow \mathbf{X}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1} = \mathbf{X}\mathbf{B}^\dagger = \mathbf{X}/\mathbf{B} \tag{1}$$

$$\mathbf{B} \leftarrow (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{X} = \mathbf{A}^\dagger\mathbf{X} = \mathbf{A}\backslash\mathbf{X} \tag{2}$$

Where $\mathbf{A}^\dagger = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$ and $\mathbf{B}^\dagger = \mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}$. This is implemented in the matlab function *pinv.m* while the Matlab operators / and \ efficiently solve for $\mathbf{A}$ and $\mathbf{B}$ respectively.

ALS implementation of the CP model

The CandeComp/PARAFAC (CP) model for a three way array $\mathcal{X}^{I\times J\times K}$ reads

$$x_{i,j,k} \approx \sum_d a_{i,d} b_{j,d} c_{k,d},$$

One of the most common ways to estimate the model is by alternating least squares. Here the loadings of one mode is estimated keeping the loadings of all the remaining modes fixed. Iteratively updating all modes in this way the algorithm is terminated when some convergence criterion is reached, most often given by a relative change in the sum of squares error less than some specified tolerance. Using the matricizing operation and Khatri-Rao product the CP model can be written as

$$\begin{aligned}
\boldsymbol{X}_{(1)} &\approx \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^\top \\
\boldsymbol{X}_{(2)} &\approx \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^\top \\
\boldsymbol{X}_{(3)} &\approx \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^\top
\end{aligned}$$

For the least squares objective we thus find

$$\begin{aligned}
\boldsymbol{A}^{(1)} &\leftarrow \boldsymbol{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\boldsymbol{C}^\top\boldsymbol{C} * \boldsymbol{B}^\top\boldsymbol{B})^{-1} \\
\boldsymbol{B}^{(1)} &\leftarrow \boldsymbol{X}_{(2)}(\boldsymbol{C} \odot \boldsymbol{A})(\boldsymbol{C}^\top\boldsymbol{C} * \boldsymbol{A}^\top\boldsymbol{A})^{-1} \\
\boldsymbol{C}^{(1)} &\leftarrow \boldsymbol{X}_{(3)}(\boldsymbol{B} \odot \boldsymbol{A})(\boldsymbol{B}^\top\boldsymbol{B} * \boldsymbol{A}^\top\boldsymbol{A})^{-1}
\end{aligned}$$

Task 1: Using the script *CP.m* implement the missing parts of the CP algorithm denoted by ???.

ALS implementation of the TUCKER model

The Tucker model reads for a third order tensor $\mathcal{X}^{I\times J\times K}$

$$x_{i,j,k} \approx \sum_{lmn} g_{l,m,n} a_{i,l} b_{j,m} c_{k,n},$$

where the so-called core array $\mathcal{G}^{L\times M\times N}$ with elements $g_{l,m,n}$ accounts for all possible linear interactions between the components of each mode. To indicate how many vectors pertain to each modality it is customary also to denote the model a Tucker(L,M,N) model. Using the n-mode tensor product $\times_n$ the model can be written as

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}.$$

As such, each mode of the array is spanned by given loading matrices for that mode such that the vectors of each modality interact with the vectors of all remaining modalities with strengths given by the core tensor $\mathcal{G}$.

One of the most common approaches to estimating the Tucker model is also based on alternating least squares. Using the n-mode matricizing and kronecker product operation the Tucker model can be written as

$$\begin{aligned} \boldsymbol{X}_{(1)} &\approx \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^\top \\ \boldsymbol{X}_{(2)} &\approx \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^\top \\ \boldsymbol{X}_{(3)} &\approx \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^\top. \end{aligned}$$

Leading to the following parameter updates by alternating least squares

$$\begin{aligned} \boldsymbol{A} &\leftarrow \boldsymbol{X}_{(1)}(\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^\top)^\dagger \\ \boldsymbol{B} &\leftarrow \boldsymbol{X}_{(2)}(\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^\top)^\dagger \\ \boldsymbol{C} &\leftarrow \boldsymbol{X}_{(3)}(\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A})^\top)^\dagger \\ \mathcal{G} &\leftarrow \mathcal{X} \times_1 \boldsymbol{A}^\dagger \times_2 \boldsymbol{B}^\dagger \times_3 \boldsymbol{C}^\dagger. \end{aligned}$$

Without loss of expressive power in the model we can impose orthonormality on the loadings of each mode. This results in the following alternating updates for the model parameters using the singular value decomposition (svd)

$$\begin{aligned} \boldsymbol{A}\boldsymbol{S}^{(1)}{\boldsymbol{V}^{(1)}}^\top &= \boldsymbol{X}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B}), \\ \boldsymbol{B}\boldsymbol{S}^{(2)}{\boldsymbol{V}^{(2)}}^\top &= \boldsymbol{X}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A}), \\ \boldsymbol{C}\boldsymbol{S}^{(3)}{\boldsymbol{V}^{(3)}}^\top &= \boldsymbol{X}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{A}). \end{aligned}$$

Upon convergence the core can be estimated according to

$$\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}^\dagger \times_2 \boldsymbol{B}^\dagger \times_3 \boldsymbol{C}^\dagger.$$

while the sum of square error (SSE) due to the orthonormality of the loadings is given by

$$SSE = \|\mathcal{X}\|_F^2 - \|\mathcal{G}\|_F^2 \tag{3}$$

Task 2: Using the script *Tucker.m* implement the missing parts of the Tucker algorithm based on imposing orthonormality indicated by ???.

Core Consistency Diagnostic

Since the CP model is a special case of the Tucker model where the core array $\mathcal{G} = \mathcal{I}$, i.e., is diagonal with ones along the diagonal, the Tucker model can be used to evaluate the cross-talk between components of the CP model. A measure of this is the core consistency diagnostic (CCD)

$$\text{CCD} = 100 \cdot (1 - \frac{\|\mathcal{G} - \mathcal{I}\|_F^2}{\|\mathcal{I}\|_F^2})$$

Where $\mathcal{G}$ is estimated as

$$\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}_{CP}^\dagger \times_2 \boldsymbol{B}_{CP}^\dagger \times_3 \boldsymbol{C}_{CP}^\dagger.$$

where $\boldsymbol{A}_{CP}$, $\boldsymbol{B}_{CP}$ and $\boldsymbol{C}_{CP}$ are the loadings of the CP solution. The CCD is often used to estimate the adequate number of components, D, in the CP model. Too many components will result in a strong degree of cross talk across the loadings of the modes thus will yield a low value of the CCD. Too few components will not have any cross-talk at all. Thus, the "correct" number of components is taken to be just before a major drop-off in the curve of $\{d, \text{CCD}\}$. As written in the original paper on core consistency diagnostics:

> *"As a rule of thumb, a core consistency above 90% can be interpreted as 'very trilinear', whereas a core consistency in the neighborhood of 50% would mean a problematic model with signs of both trilinear variation and variation which is not trilinear. A core consistency close to zero or even negative implies an invalid model, because the space covered by the component matrices is then not primarily describing trilinear variation." - Bro and Kiers, 2003*

Task 3: Using the script *CorConDiag.m* implement the missing parts of the Core Consistency Diagnostic.

CP and Tucker analysis of Chemistry data

One common application of the CP and Tucker model is the analysis of fluorescence spectra of Chemistry data. The *claus.mat* data was part of an investigation conducted by Claus A. Andersson at the Chemometrics Group at KU-life. Here five samples containing different amounts of tyrosine, tryptophane and phenylalanine were measured by fluorescence. An approach where the samples are lit by light of different wavelengths and the corresponding emission wavelengths of the sample recorded. (excitation 250-300 nm, emission 250-450 nm, 1 nm intervals). As such, the array to be analyzed is $\mathcal{X}^{5\times 51\times 201}$. The true concentrations of the three compounds is given in the variable y.

Task 4: Analyze the data by regular matrix decomposition using the singular value decomposition of the matricized array

$$\mathbf{X}_{(1)}^{Samples\times Emission-Excitation\ spectra} = \mathbf{X}_{(1)}^{5\times 51\cdot 201}. \tag{4}$$

Use the function *plotSVD.m* to visualize the results. Does SVD correctly separate the compounds into separate components?

Task 5: Fit your CP algorithm developed in Task 1 to the *claus.mat* data and use your *CoreConDiag.m* script from Task 3 to identify what the optimal number of components are as well as the spectral profiles and relative concentrations of the estimated compounds. Use plotCP.m to visualize and interpret the components. Have the underlying compounds been correctly separated into separate components?

Task 6: Fit also the corresponding TUCKER models using the algorithm created in Task 2 to the data and visualize the results using the script *plotTUCKER.m*. Have the underlying compounds been correctly separated into separate components? Explain how the CP and TUCKER model differ.

(Extra challenge): CP and Tucker for general N order arrays

For a general N-way array, i.e. $\mathcal{X}^{I_1 \times I_2 \times \ldots \times I_N}$ the CP model is given by

$$x_{i_1,i_2,\ldots,i_N} \approx \sum_d \prod_{n=1}^{N} a^{(n)}_{i_n,d},$$

$\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)} \boldsymbol{Z}^\top$, where $\boldsymbol{Z} = \boldsymbol{A^{(N)}} \odot \boldsymbol{A^{(N-1)}} \odot \ldots \odot \boldsymbol{A^{(n+1)}} \odot \boldsymbol{A^{(n-1)}} \odot \ldots \odot \boldsymbol{A^{(1)}}$

Hence for a general N-way array the update for the n^{th} mode is given by

$\boldsymbol{X}_{(n)} \approx \boldsymbol{A}^{(n)} \boldsymbol{Z}^\top$, *where* $\boldsymbol{Z} = \boldsymbol{A^{(N)}} \odot \boldsymbol{A^{(N-1)}} \odot \ldots \odot \boldsymbol{A^{(n+1)}} \odot \boldsymbol{A^{(n-1)}} \odot \ldots \odot \boldsymbol{A^{(1)}}$
$\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{X}_{(n)} \boldsymbol{Z} (\boldsymbol{Z}^\top \boldsymbol{Z})^{-1}$

The TUCKER model for a general N-way array reads

$$\mathcal{X}_{i_1,i_2,\ldots,i_N} \approx \sum_{j_1 j_2 \ldots j_N} \mathcal{G}_{j_1,j_2,\ldots,j_N} \boldsymbol{A}^{(1)}_{i_1,j_1} \boldsymbol{A}^{(2)}_{i_2,j_2} \cdot \ldots \cdot \boldsymbol{A}^{(N)}_{i_N,j_N},$$

where $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$ and $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$. To indicate how many vectors pertain to each modality it is customary also to denote the model a Tucker$(J_1, J_2, \cdots, J_N)$.

Using the n-mode matricizing and Kronecker product operation the Tucker model can be written as

$$\begin{aligned} \boldsymbol{X}_{(n)} &\approx \boldsymbol{R}_{(n)} = \boldsymbol{A}^{(n)} \boldsymbol{Z}^{(n)} \quad \textit{where} \\ \boldsymbol{Z}_{(n)} &= \mathbf{G}_{(n)} (\mathbf{A}^{(N)} \otimes \ldots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \ldots \otimes \mathbf{A}^{(1)}) = (\mathcal{R} \times_n \boldsymbol{A}^{(n)^\dagger})_{(n)}. \end{aligned}$$

Again without loss of expressive power in the model we can impose orthonormality on the loadings of each mode. This results in the following alternating updates for the model parameters using the singular value decomposition (svd)

$$\boldsymbol{A}^{(n)} \boldsymbol{S}^{(n)} \boldsymbol{V}^{(n)^\top} = \boldsymbol{X}_{(n)} (\boldsymbol{A}^{(N)} \otimes \cdots \otimes \boldsymbol{A}^{n+1} \otimes \boldsymbol{A}^{n-1} \cdots \boldsymbol{A}^{1}),$$

Upon convergence the core can be estimated according to

$$\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}^{(1)^\top} \times_2 \boldsymbol{A}^{(2)^\top} \cdots \times_N \boldsymbol{A}^{(N)^\top}.$$

while the sum of square error (SSE) due to the orthonormality of the loadings again is given by

$$SSE = \|\mathcal{X}\|_F^2 - \|\mathcal{G}\|_F^2 \tag{5}$$

Task 7 Open the script TuckerN.m and CPN.m and fill in the missing parts denoted by ???. The scripts implements the Tucker and CP methods for general N-way array and makes use of the Matlab cell array structure (type *help cell* in matlab to learn more about the cell structure).