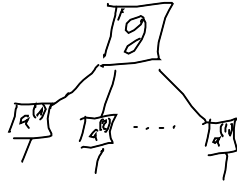


Tucker-Decomposition

Nächste Idee für hochdimensionale Schmidt:

$$T_{n_1 n_2 n_3 \dots n_k} = \sum_{k_1 \dots k_r} g_{k_1 \dots k_r} a_{k_1 n_1}^{(1)} a_{k_2 n_2}^{(2)} \dots a_{k_r n_r}^{(r)}$$



- 1.) Der wesentlichste Punkt für die Kompression ist das der Tensor g geringste Dimension hat als der ursprüngliche Tensor.
- 2.) Jeder Tensor k_n in diese Form gebracht werden, aber die Dimension g muss nicht kleiner sein.

Berechnung der Tucker-Decomposition mit HOSVD

HOSVD ($X_{k_1 \dots k_N}$)

for $n = 1, \dots, N$ do

 Fasse alle Indizes k_i ($i \neq n$)

 zu einem Index zusammen

 und führe sie SVD durch

 (Mehre nur die wichtigsten

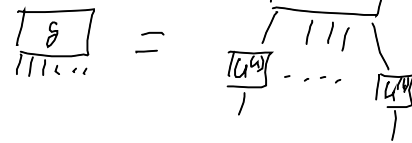
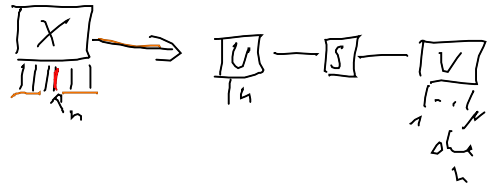
 Singular Werte mit)

 Speichere die linken Singulärvektoren

$U^{(n)}$

end

$$g_{k_1 \dots k_r} = \sum_{k_{n+1} \dots k_N} U_{k_{n+1} n_1}^{(1)} \dots U_{k_{n+1} n_r}^{(r)} X_{k_1 \dots k_N}$$



Bemerkung

- Meist ist die Approximation nicht die effektivste, meist wird noch ein ALS Fitting algorithm

- Oder es wird ein iteratives Verfahren verwendet:

HOOI ($X_{n_1 \dots n_N}$)

 initialisiere $U^{(n)}$ for $n = 1, \dots, N$ mit HOSVD

repeat

 for $n = 1, \dots, N$ do

$$\begin{matrix} \boxed{Y} \\ \hline \text{|||||} \end{matrix} = \begin{matrix} \boxed{X} \\ \hline \text{---} \end{matrix}$$

Führe SVD für Y als beschränkten Dimension, wie bei HOSVD
 Erzeuge $U^{(1)}$

und
 mit Hilfe Lösung wie die Approximation nicht bestanden

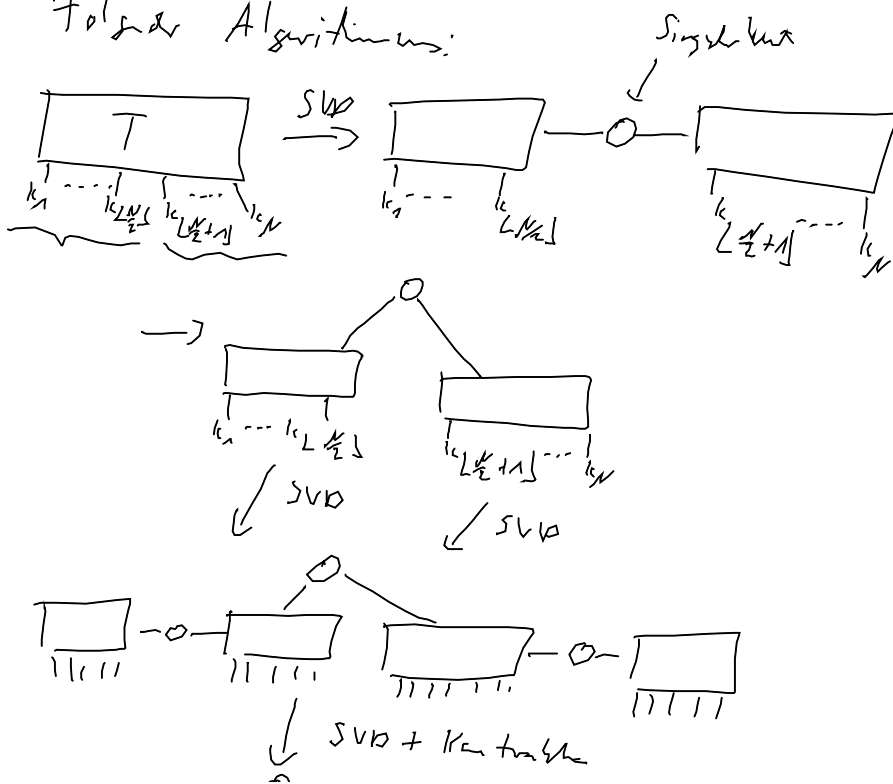
$$\begin{matrix} \boxed{g} \\ \hline \text{|||||} \end{matrix} = \begin{matrix} \boxed{X} \\ \hline \text{---} \end{matrix} \quad \text{und } \text{---}$$

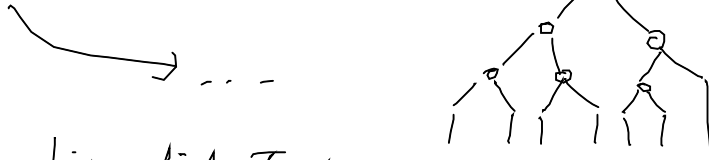
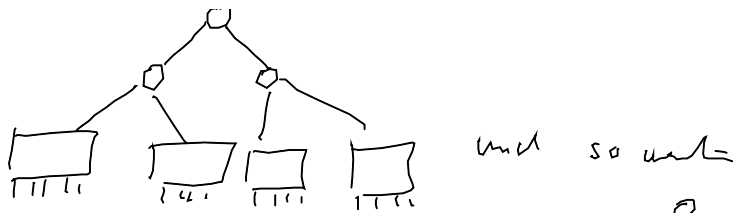
Problem: - Keine Möglichkeit dem Tensor leicht zu updaten
 immer komplette Teile notwendig (kei Dynamik,
 kein Fixwert)

Bemerkung: Es gibt noch weitere Variablen die Zerlegen
Hierarchical Tucker

Die Idee hinter Hierarchial Tucker, ist zuerst die
 Dekomposition in einem Tensor dreifachfähig sind hier wieder
 Kruskal.

Folgender Algorithmus:



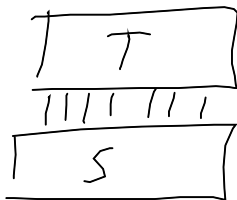


Das hierarchisch Tada Fund ist ein Art
Binär Tree Baum

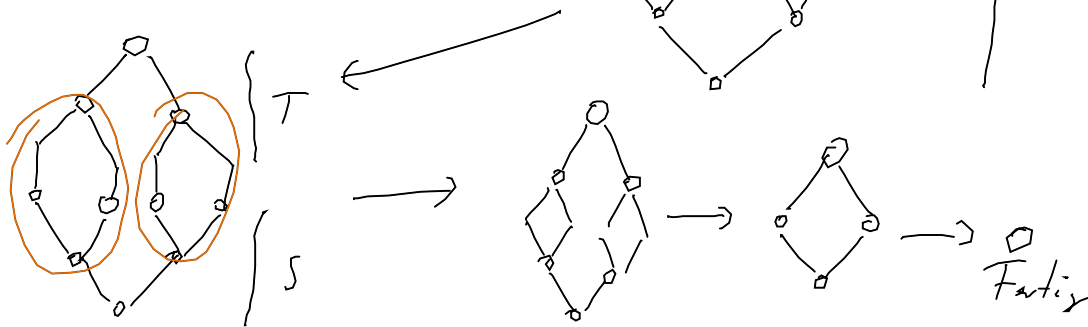
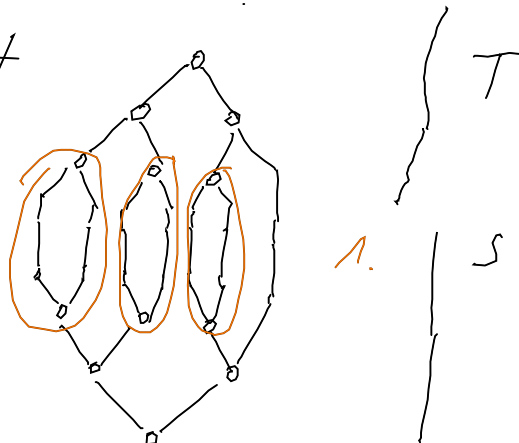
Können wir mit dem HTada Fund rechnen?

Ja, klar!

1.) Norm und Skalarprodukt



ist die
Fan für
den vollen
Tree



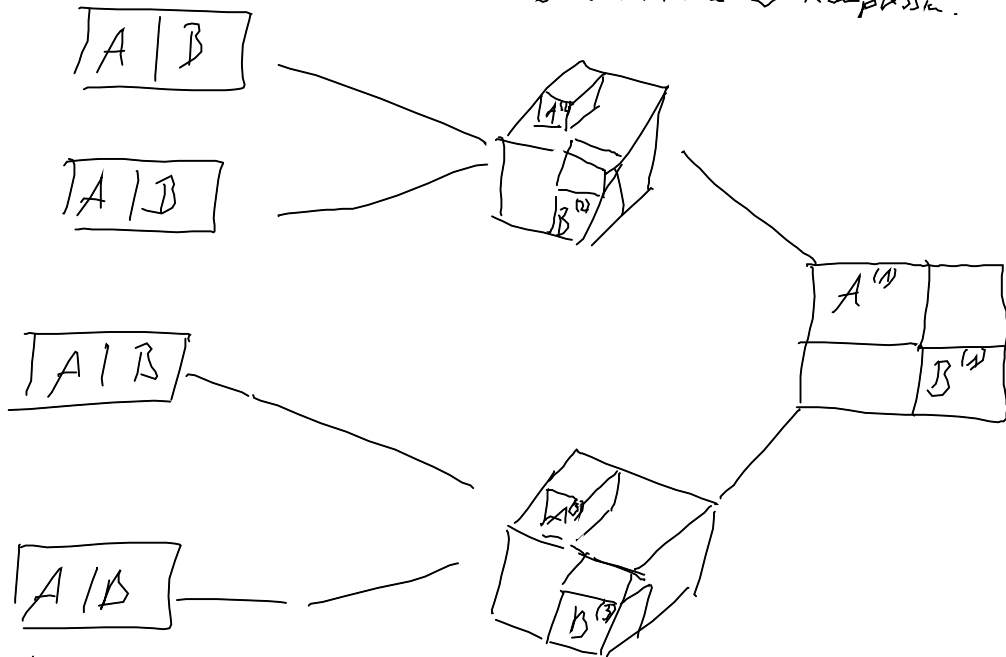
2.) Addition

Sehen wir zunächst von zwei Matrizen aus in SVD

$$A = U_A S_A V_A \quad B = U_B S_B V_B \quad \text{damit}$$

$$A+B = [U_A \ U_B] \cdot \begin{pmatrix} S_A & 0 \\ 0 & S_B \end{pmatrix} \cdot [V_A \ V_B]^T$$

\Rightarrow Das sollte man wieder an SVD durchführen zur Kompression.



3) Orthogonalisierung

Es gibt für jede Zerlegung Regeln wie die Tensoren orthogonalisiert werden können und hat zu Einflüssen
 (i) transkription, (ii) Teilweise kein Name und Skalarprodukt schnell berechnet wird.

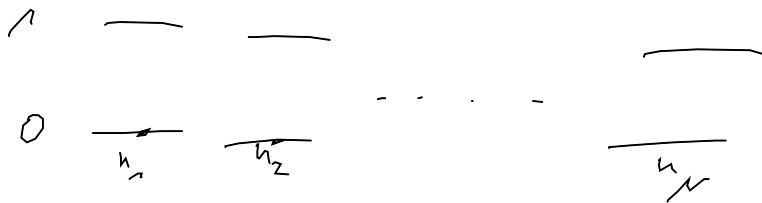
S. Bestätigung MATHICSE Technical report 4 2012
 "htucker - a matlab toolbox for tensors in the hierarchical Tucker format".

Hier besteht das Problem die Tensoren zu manipulieren.

VIII.21

Matrix Product States und Matrix Product Operator

Quanteninformation ist ein großes Forschungsgebiet,
 Grundbausteine sind meist Qubits (aka Zweiniveausystem)



in einem Kontext kann man das auch Spin-Kette
 oder Bose Hubbard model

Wenn n_i der Zustandsindex des i Qubits ist, die Gesamtwelt
 $\mathcal{Z}_{n_1, n_2, \dots, n_N}$ ein Tensor!

Zur Simulation muss die Schrödingergleichung gelöst werden:

$$i\hbar \partial_t |\psi\rangle = H|\psi\rangle \Leftrightarrow i\hbar \partial_t \mathcal{Z}_{n_1, \dots, n_N} = \sum_{n'_1, \dots, n'_N} H_{n_1, \dots, n_N; n'_1, \dots, n'_N} \mathcal{Z}_{n'_1, \dots, n'_N}$$

mit $|\psi\rangle = \sum_{n_1, \dots, n_N} \mathcal{Z}_{n_1, \dots, n_N} |n_1, \dots, n_N\rangle$

\Rightarrow TDD Tensorrechnung für \mathcal{Z} und für H findet!

\Rightarrow Matrix Product States (Vidal PRL 81 (14) 147502 (2003)
 (siehe zu verstehen bzw Schlüsselwort
 arxiv! 1008.3477v2 (2011))