CS 598 EVS: Tensor Computations

Tensor Decomposition

Edgar Solomonik

University of Illinois at Urbana-Champaign

CP Decomposition Rank

- The *canonical polyadic or CANDECOMP/PARAFAC (CP) decomposition* expresses an order d tensor in terms of d factor matrices

Tensor Rank Properties

- Tensor rank does not satisfy many of the properties of matrix rank

Typical Rank and Generic Rank

- When there is only a single typical tensor rank, it is the *generic rank*

Uniqueness Sufficient Conditions

- Unlike the low-rank matrix case, the CP decomposition can be unique

Uniqueness Necessary Conditions

- Necessary conditions for uniqueness of the CP decomposition also exist

Degeneracy

- The best rank-k approximation may not exist, a problem known as *degeneracy* of a tensor

Border Rank

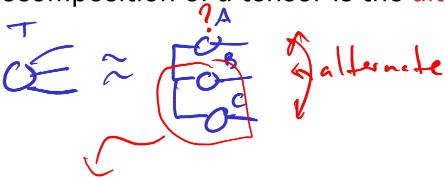
- Degeneracy motivates an approximate notion of rank, namely *border rank*

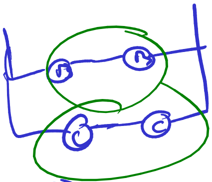
Approximation by CP Decomposition

- Approximation via CP decomposition is a nonlinear optimization problem

Alternating Least Squares Algorithm

- The standard approach for finding an approximate or exact CP decomposition of a tensor is the *alternating least squares (ALS) algorithm*





$$(B \odot C) A^T \cong T_{(1)}^T$$

$$\min_A \left\| (B \odot C) A^T - T_{(1)}^T \right\|_F$$

$O(nR^2 + R^3)$

$T \in \mathbb{R}^{n \times n \times n}$

CP rank R

$O(nR^2)$

$O(n^3 R)$

$$\underbrace{(B \odot C)^T (B \odot C)}_{B^T B \,*\, C^T C} A^T = \underbrace{(B \odot C)^T T_{(1)}^T}_{MTTKRP}$$

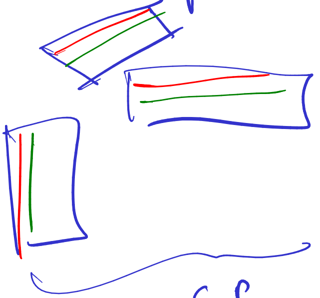
Properties of Alternating Least Squares for CP

+ monotonic decrease of objective

+ stationary point is local minima

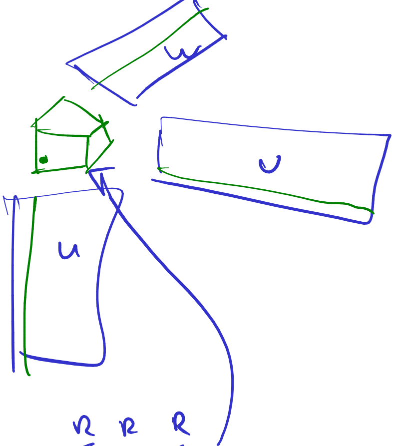
- linear (local) convergence rate

Tucker decomposition



CP

$\rightarrow$



$$t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

$$t_{ijl} = \sum_{r}^{R} \sum_{p}^{R} \sum_{q}^{R} z_{rpq} u_{ir} v_{jp} w_{lq}$$

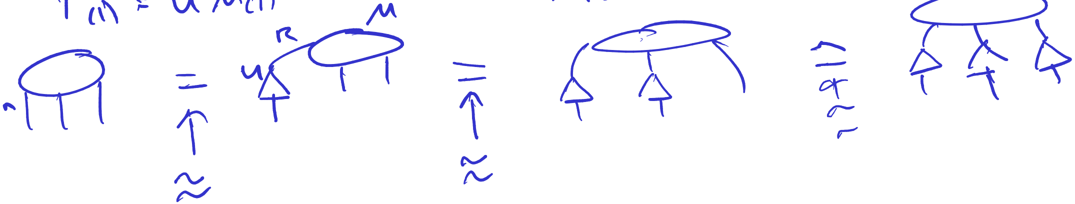
$$U^T U = I$$

HoSVD

+ direct / one-shot

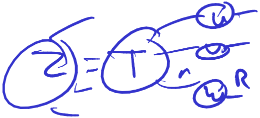
+ exact if Tucker is exact

$T_{(1)}^T = U M_{(1)}^T$ $\qquad\qquad$ $M_{(2)}^T = V N_{(2)}^T$



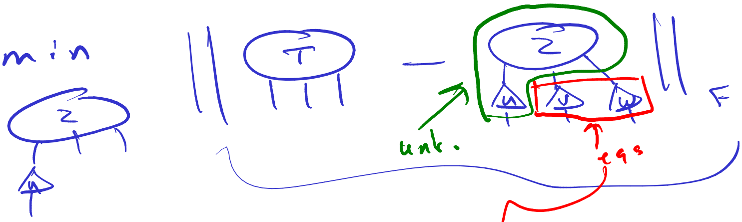
sequentially truncated HoSVD

$- \; T_{(1)}^T = U M_{(1)}^T$ $\qquad\qquad$ $\tilde{T}_{(2)}^T = V N_{(2)}^T$



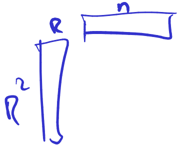
Alternating Least Squares for Tucker Decomposition

- For Tucker decomposition, an analogous optimization procedure to ALS is referred to as *high-order orthogonal iteration (HOOI)*



solve for core tensor + 1 factor at a time

$$\min_{Z^T_{(1)}U^T} \left\| T^T_{(1)} - \underbrace{(V \otimes W)}_{n^2 \quad R^2}(Z^T_{(1)}U^T) \right\|_2$$



$$\min_{X,\ \mathrm{rank}(X)=R} \left\| T_{(1)}^T - \boxed{(V \otimes W)} X \right\|_F$$

orthogonal

$$(V \otimes W)^T (V \otimes W) = I$$

$$V^T V \otimes W^T W$$

$$(V \otimes W)^T T_{(1)}^T$$

$$\min_X \left\| G^T B - X \right\|_F$$

$$\left\| B - \underline{Q} X \right\|_F =$$

HOOI Algorithm

• monotonic progress

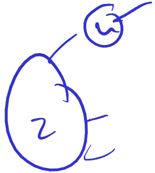
Iterate

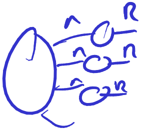
Alternate

rank R

$$T_{(1)}(V \otimes W) \approx U Z_{(1)}$$

$\approx$





$s^3 R$

$s^2 R^2$

TTMc

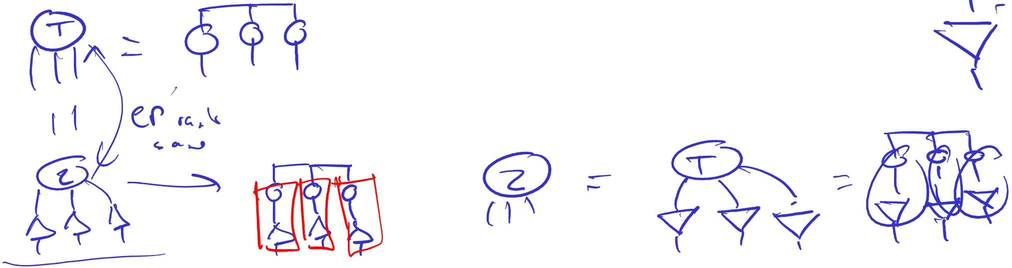
tensor-times matrix chain

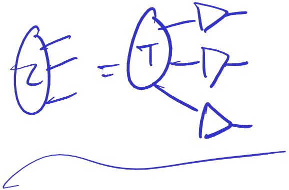
Tucker $\Longleftrightarrow$ CP

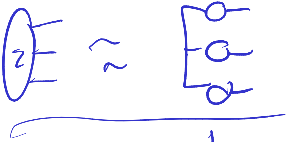
if CP rank $R < n$

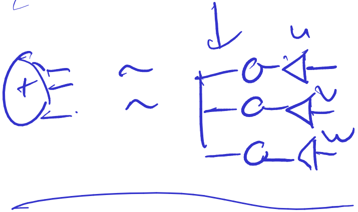
CP rank of Z is R

core tensor for rank R Tucker

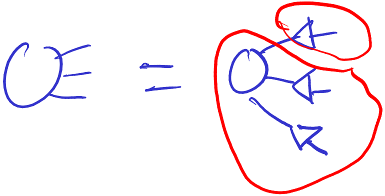


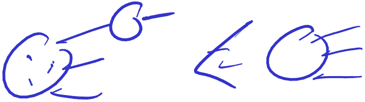

z
T


z


t
u
v
w


$f_T(x,y)$

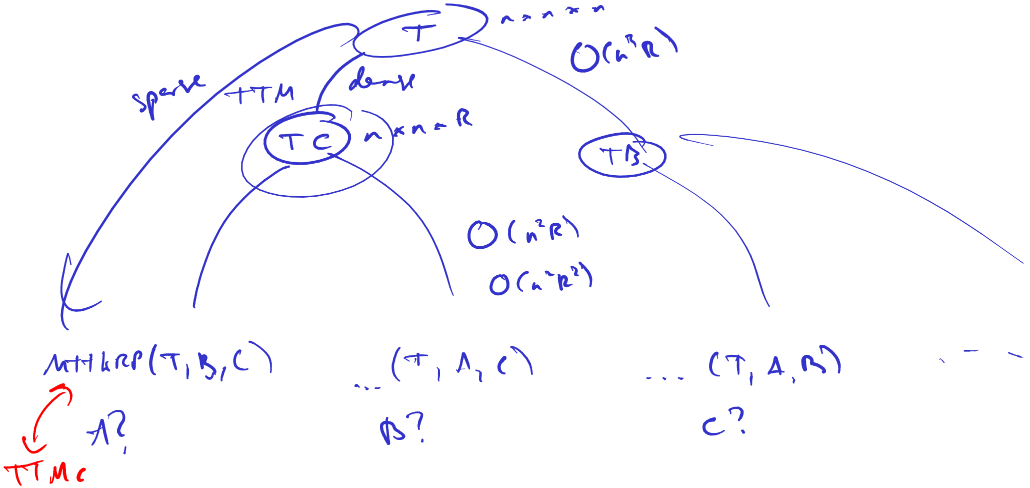




Dimension Trees for ALS

- The cost of ALS can be reduced by amortizing computation common terms

For both CP and Tucker



Fast Residual Norm Calculation

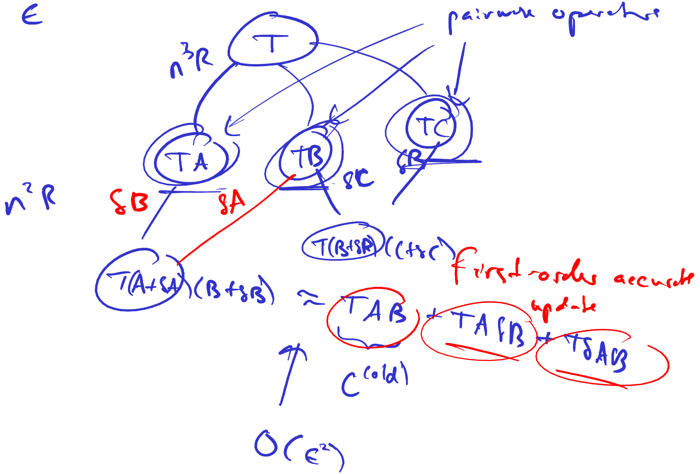
- Calculating the norm of the residual has cost $2ds^dR$, but can be done more cheaply within ALS

Pairwise Perturbation Algorithm

- A route to further reducing the cost of ALS is to perform it approximately via *pairwise perturbation*

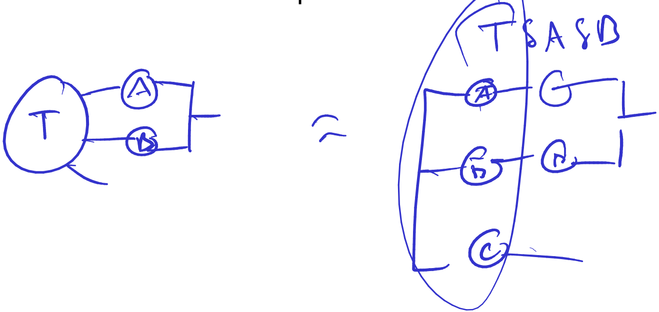
$$\|A_4^{(i+1)} - A_4^{(i)}\|_F \le \epsilon$$

iteration index



Pairwise Perturbation Second Order Correction

- When approximating a tensor using CP, the partially converged CP factors can sometimes be used in place of the tensor to accelerate cost



Approximate CP ALS using Random Sampling

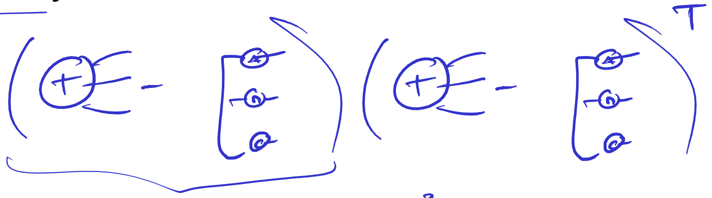
- Another approach to approximating ALS is to sample the least-squares equations[1]

[1] C. Battaglino, G. Ballard, T. G. Kolda, 2018

Gauss-Newton Algorithm

- ALS generally achieves linear convergence, while Newton-based methods can converge quadratically

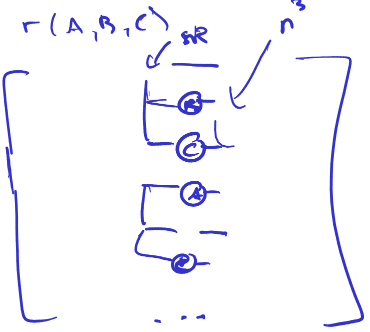
$f(A,B,C) = \frac{1}{2}$



$f = \frac{1}{2} r^T r$

$J_r x = r$

$J_r =$



$\frac{\partial r}{\partial A}$

$\frac{\partial r}{\partial B}$

$\frac{\partial r}{\partial C}$

$$J_r^T J_r x = J_r^T r \qquad J_r = \begin{bmatrix} & & & & \end{bmatrix}$$

$$I \otimes (B \odot C)$$

$$x = \begin{bmatrix} \mathrm{vec}(A) \\ \mathrm{vec}(B) \\ \mathrm{vec}(C) \end{bmatrix} \in \mathbb{R}^{3nR}$$

$$J_r^T J_r \in \mathbb{R}^{3nR \times 3nR} \qquad n^2 R^2$$

direct method has cost $O(n^3 R^3)$

$$I \otimes \left(B^T B * C^T C\right)$$

$$J_r^T J_r = \frac{\partial f}{\partial A} = \begin{bmatrix} \underline{K_1} & \underline{--} & \underline{--} \\ \underline{--} & \underline{K_2} & \underline{--} \\ \underline{--} & \underline{--} & \underline{K_3} \end{bmatrix}$$

$$J_r \in \mathbb{R}^{n^3 \times 3nR}$$

Gauss-Newton for CP Decomposition

- CP decomposition for order $d = 3$ tensors ($d > 3$ is similar) minimizes

Gauss-Newton for CP Decomposition

- A step of Gauss-Newton requires solving a linear system with $\boldsymbol{H}$

```
u = []
for q in range(d):
  u.append(zeros((n,R)))
  for p in range(d):
    if q == p:
      u[q] += einsum("rz,kz->kr",G[q,p],v[p])
    else:
      u[q] += einsum("kz,lr,rz,lz->kr", \
                     U[q],U[p],G[q,p],v[p])
```

Tensor Completion

- The *tensor completion* problem seeks to build a model (e.g., CP decomposition) for a partially-observed tensor

- The problem was partially popularized by the Netflix prize collaborative filtering problem

CP Tensor Completion Gradient and Hessian

- The gradient of the tensor completion objective function is sparsified according to the set of observed entries
- ALS for tensor decomposition solves quadratic optimization problem for each row of each factor matrix, in the completion case, Newton's method on these subproblems yields different Hessians

Methods for CP Tensor Completion

- ALS for tensor completion with CP decomposition incurs additional cost

- Alternative methods for tensor completion include coordinate descent and stochastic gradient descent

Coordinate Descent for CP Tensor Completion

- Coordinate descent avoids the need to solve linear systems of equations

Sparse Tensor Contractions

- Tensor completion and sparse tensor decomposition require operations on sparse tensors

- Sparse tensor contractions often correspond to products of *hypersparse* matrices, i.e., matrices with mostly zero rows

Sparse Tensor Formats

- The overhead of transposition, and non-standard nature of the arising sparse matrix products, motivates sparse data structures for tensors that are suitable for tensor contractions of interest

- The *compressed sparse fiber (CSF)* format provides an effective representation for sparse tensors

Operations in Compressed Format

- CSF permits efficient execution of important sparse tensor kernels
  - Analogous to CSR format, which enables efficient implementation of the sparse matrix vector product
  - where `row[i]` stores a list of column indices and nonzeros in the ith row of $\boldsymbol{A}$

```
for i in range(n):
  for (a_ij,j) in row[i]:
    y[i] += a_ij * x[j]
```

  - In CSF format, a multilinear function evaluation $\boldsymbol{f}^{(\mathcal{T})}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{T}_{(1)}(\boldsymbol{x} \odot \boldsymbol{y})$ can be implemented as

```
for (i,T_i) in T_CSF:
  for (j,T_ij) in T_i:
    for (k,t_ijk) in T_ij:
      z[i] += t_ijk * x[j] * y[k]
```

MTTKRP in Compressed Format

- MTTKRP and CSF pose additional implementation opportunities and challenges
  - MTTKRP $u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$ can be implemented by adding a loop over r to our code for $\boldsymbol{f}^{(\mathcal{T})}$, but would then require $3mr$ operations if m is the number of nonzeros in $\mathcal{T}$, can reduce to $2mr$ by amortization

```
for (i,T_i) in T_CSF:
  for (j,T_ij) in T_i:
    for r in range(R):
      f_ij = 0
      for (k,t_ijk) in T_ij:
        f_ij += t_ijk * w[k,r]
      u[i,r] = f_ij * v[j,r]
```

  - However, this amortization is harder (requires storage or iteration overheads) if the index i is a leaf node in the CSF tree
  - Similar challenges in achieving good reuse and obtaining good arithmetic intensity arise in implementation of other kernels, such as TTMc

All-at-once Contraction

- When working with sparse tensors, it is often more efficient to contract multiple operands in an all-at-once fashion

Constrained Tensor Decomposition

- Many applications of tensor decomposition in data science, feature additional structure, which can be enforced by constraints

Nonnegative Tensor Factorization

- *Nonnegative tensor factorization (NTF)*, such as CP decomposition with $\boldsymbol{\mathcal{T}} \geqslant 0$ and $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W} \geqslant 0$ are widespread and a few classes of algorithms have been developed

Nonnegative Matrix Factorization

- NTF algorithms with alternating updates have a close correspondence with alternating update algorithms for *Nonnegative matrix factorization (NMF)*

Coordinate Descent for NMF and NTF

- Coordinate descent gives optimal closed-form updates for variables in NMF and NTF

Generalized Tensor Decomposition

- Aside from addition of constraints, the objective function may be modified by using different elementwise loss functions
- Some loss function admit ALS-like algorithms, while others may require gradient-based optimization