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 (rank of a random tensor)
  - Typical rank for $\mathbb{C}$
  - By d.f. arguments $R = n^{d-1}$
  - Each factor is of size $n \times R$
Typical Rank and Generic Rank

- When there is only a single typical tensor rank, it is the *generic rank*.
Unlike the low-rank matrix case, the CP decomposition can be unique up to permuting of $R$ rank-1 terms and scaling of vectors in each. Uniqueness depends on rank of factors. Also $k$-rank.

If $A, B, C$ are full rank, then $R \leq 3n/2$. 

$R = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{ij}B_{jk}C_{ki}$
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 of $T$
as smallest $R$, s.t. $\exists$ (continuous) sequence

$T_i$,...
s.t.

$\lim\|T_i\| = 0$

$\begin{array}{l}
\text{b-root 2, } \text{root 3 means}
\end{array}$
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

\[
\min \frac{1}{2} \| T - \bigotimes_{i,j,k} u_{ijk} w_i \|_F^2
\]

\[
c_e(u,v,w) = \frac{1}{2} \sum_{i,j,k} \left( \sum_{s} u_{ijk} w_i v_{s} - \right.
\]

\[
\frac{\partial c_e}{\partial u_{uvw}} = 0
\]

\[
\frac{\partial c_e}{\partial u_{ijk}} = -\sum_{s} (i,j,k,v_{s} w_i u_{s} - )
\]

\[
+ u_{s} w_i v_{s} (\sum_{s} u_{ijk} v_{s} w_i )
\]
\[
\sum_{j,k} \sum_{i,r} w_{ir} w_{jr} = \sum_{j,k} \sum_{i,r} w_{ir} w_{jr} \sum_{s,t} w_{ts} w_{st} \quad \text{MTAIRP}
\]

\[
\left( \sum_{j,k} v_{ir} v_{jr} \right) \left( \sum_{k,l} w_{kl} w_{kl} \right) \quad \text{u,i,r}
\]

\[
T_{\text{CL}} (W_{\theta U}) = \left( \frac{1}{2} \left( (V^T V) + (W^T W) \right) u^T \right)
\]

\[
\varphi_{\frac{1}{2}} (\alpha - \frac{1}{\sqrt{C}}) \left( \alpha - \frac{C}{\rho_0} \right)
\]

\[
\frac{\partial e}{\partial u} = - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = 0 \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0
\]
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*.
Properties of Alternating Least Squares for CP

- egs are defined by Chehdi - Rao products
  amenable to optimizations

- convergence
  - locally (near local minima of \( e \))

- linear convergence

- no global minima

- convergence can stagnate due to factors becoming ill-conditioned

- stationary point of ALS is a local minima of \( e \) or saddle point
Alternating Least Squares for Tucker Decomposition

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

- The cost of ALS can be reduced by amortizing computation common terms
d-i sweeps
with d TTM3
Fast Residual Norm Calculation

- Calculating the norm of the residual has cost $2d s^d R$, but can be done more cheaply within ALS
\[ f_T(u, v) = T_{x_u, y_u} \]
Pairwise Perturbation Algorithm

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

```python
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])
```
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 \textit{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 \textit{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 $i$th row of $A$

```python
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 $f^{(T)}(x, y) = T^{(1)}(x \odot y)$ can be implemented as

```python
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 $f(T)$, but would then require $3mr$ operations if $m$ is the number of nonzeros in $T$, can reduce to $2mr$ by amortization

```python
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 $\mathbf{T} \succeq 0$ and $\mathbf{U}, \mathbf{V}, \mathbf{W} \succeq 0$ 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
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.