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*
Properties of Alternating Least Squares for CP
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
Fast Residual Norm Calculation

- Calculating the norm of the residual has cost $2ds^d R$, 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*
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.
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
A step of Gauss-Newton requires solving a linear system with $H$

```
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])
```
Matrix Sketching

Randomized methods provide accurate approximate solutions to linear least squares problems, which can be applied to accelerate ALS, as well as more basic problems.
Random Projections

Accuracy of sketching techniques is theoretically characterized by statistical analysis
Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma is a powerful tool for obtaining error bounds in a projected vector space

$$SA\hat{x} \simeq Sb$$
Matrix Sketching

The best choice of sketch matrix depends on the desired accuracy and the structure of $A$.
Matrix Sketching via Sampling

Uniform sampling of rows is insufficient to obtain good accuracy in general.

Leverage score sampling provides better accuracy guarantees.
Mixing Techniques

To circumvent leverage score sampling, we can mix rows randomly. Instead of choosing elements of $S$ randomly, pseudo-random distributions allow $S$ to be applied more rapidly.
Another approach to approximating ALS is to sample the least-squares equations\textsuperscript{1}

\textsuperscript{1}C. Battaglino, G. Ballard, T. G. Kolda, 2018
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.
Matrix Completion

Given a partially observed matrix

\[ A \rightarrow \text{movies} \]

\[ \mathcal{R} - \text{set of pairs of indices} \]

\[ a_{ij} \in \mathcal{R} \]
\[ f(U, V) = \frac{1}{|I|} \sum_{(i,j) \in I} \left( a_{ij} - \sum_{r \in R} u_{ir} v_{jr} \right)^2 + \lambda \|U\|_F^2 + \gamma \|U\|_F^2 \]

- as closed form solution
- but \( f(U, V) \) is convex if we fix \( U \) or \( V \)
CP Tensor Completion Gradient and Hessian

- The gradient of the tensor completion objective function is sparsified according to the set of observed entries.

\[
\nabla f(u, \nu, \omega) = \sum_{\ell, j, k \in \mathcal{E}} \left( \frac{1}{\|\mathbf{c}_{j, k}\|} \right) \left( d_{\ell, j, k} - \langle u_{\ell, \nu_{j}, \omega_{k}} \rangle \right)^2
\]

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.

\[
H_f(u, \nu, \omega) = \sum_{\ell, j, k \in \mathcal{E}} \frac{1}{\|\mathbf{c}_{j, k}\|} \left( \mathbf{v}_j \mathbf{v}_j^T + \mathbf{w}_k \mathbf{w}_k^T \right) + 2\lambda \mathbf{I}
\]
\[ n_1 = 1_j \quad \forall j, i \in \Xi_1 \wedge q^2 \]

\[ u = \frac{1}{\overline{u}} \sum_j x_j u_j v_j^* w_{j*} \]

\[ u = \frac{1}{\overline{u}} \left( \sum_j (u_j v_j^*) (w_{j*} w_{j*}^T) \right) \]

\[ = \frac{1}{\overline{u}} \underbrace{\sum_j v_j v_j^T \cdot w_j w_j^T} \in (U \cdot 0 \cdot U)^T (U \cdot 0 \cdot U) \]
Methods for CP Tensor Completion

- ALS for tensor completion with CP decomposition incurs additional cost

1. To optimize $U$
   - need to form Hessians, $H^{(n)} \sim \sum_{i,j} \nabla^2 H^{(n)}$ and solve

2. for RHS need MTTKR P with $T$ (observed entries of $X$

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

   - Coord. desc. minimize for some $u_i$, instead of $u_i$
     - at a time (less progress/subproblem), but updates cheaper
   - SGD: consider subgradient
     
     \[ u_{i}^{(n+1)} = u_{i}^{(n)} - \frac{1}{1 + 2T} \cdot g \cdot \langle u_{i}, u_{j}, u_{k} \rangle (u_{j} + w_{k}) \]
Coordinate Descent for CP Tensor Completion

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

\[
\frac{\partial f}{\partial u_{ir}} = \frac{1}{\lVert \mathbf{U} \rVert^2} \sum_{j=1}^{n} \sum_{k=1}^{r} \left( \left( p_{ijk} - \sum_{l=1}^{d} \langle u_{il}, v_{lj}, w_{kl} \rangle \right)^2 + \lambda \sum_{l=1}^{d} \| u_{il} \|_F^2 \right) \nabla_{u_{ir}} f_{ij} \nabla_{u_{en}} f_{en} - \sum_{l=1}^{d} \langle u_{il}, v_{lj}, w_{kl} \rangle (u_{il} v_{lj} w_{kl}) + 2 \lambda u_{ir}
\]
\[ u_{ir} = \frac{1}{|\mathbf{M}|} \sum_{c_{ir}, c_{ur}, c_{ur} = 1}^{C} \mathbf{P}_{c_{ir}} \mathbf{v}_{ir} + \frac{1}{|\mathbf{M}|} \sum_{c_{ir}, c_{ur}, c_{ur} = 1}^{C} \mathbf{v}_{ir} \mathbf{v}_{ur} \]

residual-like term

\[ \frac{1}{1 + \frac{1}{\sum_{c_{ir}, c_{ur}, c_{ur} = 1}^{C} \mathbf{v}_{ir} \mathbf{v}_{ur}}} \]

diff. ord. closer

cyclic coord. descent

CCD

CCD++
Suppose we've updated \( a_i \) (one column) with \( \hat{r}_{ij}^{(n)} = (d_{ijk} - \langle z_{i,j}, u_{r}, v_{r} \rangle) \).

Next, we may consider \( r_{ijk}^{(n+1)} = \hat{r}_{ijk} - u_{ir}v_{jr} + u_{ir}v_{j} - u_{i}v_{jr} \).

**CP cost per col-update** is \( O(1/\lambda_2) \), so for all factors/cols, \( O(1/\lambda_2 R) \).
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 i\textsuperscript{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 $f^\tau(x, y) = T_1(x \odot y)$ 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 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.
Many applications of tensor decomposition in data science, feature additional structure, which can be enforced by constraints.

\[
\begin{align*}
\min_{u,v,w} & \quad \| T - [u,v,w] \|_F \\
\text{subject to} & \quad u,v,w \geq 0
\end{align*}
\]

- Non-negativity
- Constrained to a certain structure or normality

\[
\begin{align*}
\| u_i - u_{i+1} \| \leq C \quad (\text{continuity})
\end{align*}
\]
Nonnegative tensor factorization (NTF), such as CP decomposition with \( T \geq 0 \) and \( U, V, W \geq 0 \) 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 \textit{Nonnegative matrix factorization (NMF)}. 
Coordinate Descent for NMF and NTF

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

\[ f(u,v) = \| T - u v^T \|_2^2 \]

\[ \frac{df}{du_i} = (T - u v^T) v_i = 0 = \rho u_i - u_i v_i^T v_i \implies \]

\[ u_i = \frac{\rho v_i}{v_i^T v_i} \quad u_i = \frac{1}{\sigma_{v_i}} \left[ \begin{array}{c} \rho v_i \\ \sigma_{v_i} \end{array} \right] \quad \text{for} \quad 1 \leq i \leq n \]

\[(x_1 x_2)_{(1)} = x_1 14 \times 20\]
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.