CS 598 EVS: Tensor Computations

Tensor Decomposition

Edgar Solomonik

University of Illinois, 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

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

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 $\boldsymbol{H}$
  - Cholesky of $\boldsymbol{H}$ requires $O(d^2n^2R^2)$ memory and cost $O(d^3n^3R^3)$
  - Matrix-vector product with $\boldsymbol{H}$ can be computed with cost $O(d^2nR^2)$
  - Can use CG method with implicit matrix-vector product[1]
  - Each product $\boldsymbol{u} = \boldsymbol{H}\boldsymbol{v}$ can be performed using tensor contractions each with cost $O(nR^2)$
  - $\boldsymbol{H}$ admits an effective block-diagonal preconditioner (inverse of each block applies step of ALS)

[1] P. Tichavsky, A. H. Phan, and A. Cichocki, 2013

Alternating Mahalanobis Distance Minimization (AMDM)

- High-order convergence can be achieved for low-rank exact CP using the AMDM algorithm

Computing the CP Rank

Exact algorithms for bounding the CP rank of a tensor can be phrased via methods for polynomial systems of equations[2]

[2] Aliabadi, Mohsen, and Shmuel Friedland. "On the complexity of finding tensor ranks." Communications on Applied Mathematics and Computation 3.2 (2021): 281-289.

Effective Nullstellensatz

Hilbert's weak Nullstellensatz is a characterization of polynomial equations[3] This

characterization reduces polynomial systems to linear equations

[3] Following Terence Tao's formulation of this theorem
`https://terrytao.wordpress.com/2007/11/26/hilberts-nullstellensatz/` (accessed Oct. 2024).

Tensor Completion

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

- The problem generalized matrix completion, a problem partly 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

Complexity of Sparse Tensor Contractions

- The cost of a contraction of two sparse tensors depends on the position of the nonzeros

- The cost of a contraction of a single sparse tensor with dense tensors can be quantified more directly

Complexity of Contractions with a Single Sparse Tensor

In general, when contracting a single sparse tensor with many dense tensors partial sums can be amortized

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)*[4]

[4] Gillis, Nicolas. "The why and how of nonnegative matrix factorization." Regularization, optimization, kernels, and support vector machines 12.257 (2014): 257-291.

Optimality Conditions for NMF

- The optimality conditions for NMF are
- These follow from the KKT conditions, including complementarity slackness

Coordinate Descent for NMF and NTF

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

Alternating Optimization for NMF and NTF

- If all except one factor is fixed, the resulting subproblem is an inequality-constrained convex optimization problem

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