Low rank approximations of tensors

Suraj Kumar

Inria & ENS Lyon

Email:*suraj.kumar@ens-lyon.fr*

CR12: October 2024

https://surakuma.github.io/courses/daamtc.html

Properties of matrix Frobenius norm for real matrices

$$||A||_F^2 = \sum_{i,j} A^2(i,j) = Trace(AA^T) = Trace(A^TA)$$

$$||A+B||_F^2 = ||A||_F^2 + ||B||_F^2 + 2\langle A, B\rangle_F$$

Here $\langle A, B\rangle_F$ is known as Frobenius inner product and defined as $\langle A, B\rangle_F = Trace(A^TB) = Trace(B^TA)$.

If Q is an orthonormal matrix then,

$$||A||_F^2 = ||QQ^TA||_F^2 + ||(I - QQ^T)A||_F^2,$$
$$||QC||_F = ||C||_F,$$
$$||Q^TA||_F = ||QQ^TA||_F \leq ||A||_F,$$
$$\langle A - QQ^TA, QQ^TA\rangle_F = 0.$$

Tensor norm

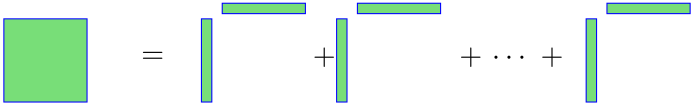
- The norm of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is analogous to the matrix Frobenius norm, and defined as

$$||\mathcal{A}||_F = \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \mathcal{A}^2(i_1, i_2, \cdots, i_d)}$$

We will only focus on Frobenius norm in this course.

Singular Value Decomposition (SVD)

- It decomposes a matrix $A \in \mathbb{R}^{m \times n}$ to the form $U \Sigma V^T$
  - U is an $m \times m$ orthogonal matrix
  - V is an $n \times n$ orthogonal matrix
  - Σ is an $m \times n$ rectangular diagonal matrix
- The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are called singular values
  - $\sigma_i \geq 0$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)}$
- The largest r such that $\sigma_r \neq 0$ is called the rank of the matrix
- SVD represents a matrix as the sum of r rank one matrices



Low rank approximations of matrices using SVD

SVD decomposition: $A = U\Sigma V^T$

Let u_i and v_i be the column vectors of U and V, respectively.

r'-rank approximation

If $\tilde{A} = \sum_{i=1}^{r'} \sigma_i u_i v_i^T$, then $\tilde{A}$ is an r'-rank approximation of A.

$$||A - \tilde{A}||_F^2 = \sum_{i=r'+1}^{\min(m,n)} \sigma_i^2$$

SVD gives the best r'-rank approximation of any matrix.

Approximation for ϵ accuracy

We select minimum r' such that $\sum_{i=r'+1}^{\min(m,n)} \sigma_i^2 \leq \epsilon^2$. The approximation is $\tilde{A} = \sum_{i=1}^{r'} \sigma_i u_i v_i^T$.

$$||A - \tilde{A}||_F^2 = \sum_{i=r'+1}^{\min(m,n)} \sigma_i^2 \leq \epsilon^2$$

Properties of SVD

The SVD of $A \in \mathbb{R}^{m \times n}$ can be written as $A = U\Sigma V^T$. Here $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m \times n}$ is a rectangular diagonal matrix.

- Columns of U are also eigen vectors of AA^T
- Similarly, columns of V are eigen vectors of A^TA
- If $\sigma_i > 0$ is a singular value of A then σ_i^2 is an eigen value of AA^T and A^TA

$\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal matrices. Their diagonal entries are the eigen values of AA^T and A^TA, respectively.

We can also express SVD as

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1 V_2 \end{pmatrix}^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T.$$

This is equivalent to

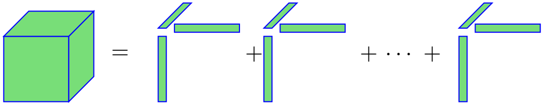
$$A = U_1U_1^TA + U_2U_2^TA = AV_1V_1^T + AV_2V_2^T.$$

Table of Contents

1. CP decomposition
2. Tucker decomposition
3. Tensor Train decomposition
4. Compact representations of tensor operations
5. Miscellaneous

CP decomposition of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$

It factorizes a tensor into a sum of rank one tensors.



CP decomposition of a 3-dimensional tensor.

$$\mathcal{A} = \sum_{\alpha=1}^{r} U_1(:,\alpha) \circ U_2(:,\alpha) \circ \cdots \circ U_d(:,\alpha)$$

It can be concisely expressed as $\mathcal{A} = [\![U_1, U_2, \cdots, U_d]\!]$. CP decomposition for a 3-dimensional tensor in matricized form can be written as:

$$A_{(1)} = U_1(U_3 \odot U_2)^T,\ A_{(2)} = U_2(U_3 \odot U_1)^T\ A_{(3)} = U_3(U_2 \odot U_1)^T.$$

It is useful to assume that $U_1, U_2 \cdots U_d$ are normalized to length one with the weights given in a vector $\lambda \in \mathbb{R}^r$.

$$\mathcal{A} = [\![\lambda; U_1, U_2, \cdots, U_d]\!] = \sum_{\alpha=1}^{r} \lambda_\alpha U_1(:,\alpha) \circ U_2(:,\alpha) \circ \cdots \circ U_d(:,\alpha)$$

Tensor rank

$$\mathcal{A} = \sum_{\alpha=1}^{r} \lambda_\alpha U_1(:,\alpha) \circ U_2(:,\alpha) \circ \cdots \circ U_d(:,\alpha)$$

- The minimum r required to express $\mathcal{A}$ is called the rank of $\mathcal{A}$

The rank of a real-valued tensor may be different over $\mathbb{R}$ and $\mathbb{C}$. For example, consider the frontal slices of $\mathcal{A} \in \mathbb{R}^{2\times 2\times 2}$

$$\mathcal{A}(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \mathcal{A}(:,:,2) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

This has rank three over $\mathbb{R}$ and two over $\mathbb{C}$. The CP decomposition over $\mathbb{R}$ has the following factor matrices:

$$U_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}, U_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \text{ and } U_3 = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}.$$

The CP decomposition over $\mathbb{C}$ has the following factor matrices:

$$U_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}, U_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}, \text{ and } U_3 = \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}.$$

Rank and low-rank approximations

- Determining the rank of a tensor is an NP-complete problem

- If $\mathcal{A} = \sum_{\alpha=1}^{r} \lambda_\alpha U_1(:,\alpha) \circ U_2(:,\alpha) \circ \cdots \circ U_d(:,\alpha)$, summing $k < r$ terms may not yield a best rank-k approximation

- Possible that the best rank-k approximation of a tensor may not exist

CP decomposition: example

Let $\mathcal{A} \in \mathbb{R}^{2\times 4\times 3}$ and $A = [\![U_1, U_2, U_3]\!]$. The rank of $\mathcal{A}$ is 2.

$$U_1 = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 4 & 6 \\ 3 & 7 \end{pmatrix}, \quad U_3 = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Computation of $\mathcal{A}(2,3,1)$,

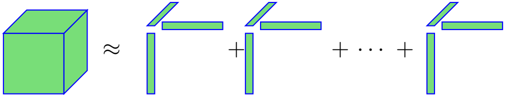
$$\begin{aligned} \mathcal{A}(2,3,1) =& \sum_{\alpha=1}^{2} U_1(2,\alpha)U_2(3,\alpha)U_3(1,\alpha) \\ =& 2\cdot 4\cdot 1 + 4\cdot 6\cdot 4 = 104 \end{aligned}$$

$\mathcal{A}$ has total 24 elements, while the CP representation has 18 elements.

Table of Contents

1. CP decomposition
   - Computing CP with Alternating Least Squares
2. Tucker decomposition
   - Computing Tucker decomposition
3. Tensor Train decomposition
   - Computing Tensor Train decomposition
4. Compact representations of tensor operations
5. Miscellaneous
   - Research topics/articles for the project
   - Strassen's algorithm: application of CP-decomposition

CP optimization problem for a 3-dimensional tensor



For fixed rank k, we want to solve

$$\min_{U_1, U_2 U_3} ||\mathcal{A} - \sum_{\alpha=1}^{k} \lambda_\alpha U_1(:,\alpha) \circ U_2(:,\alpha) \circ U_3(:,\alpha)||_F.$$

- It is a nonlinear, nonconvex optimization problem
- In the matrix case, the SVD provides us the optimal solution
- In the tensor case, convergence to optimum not guaranteed

Alternating Least Squares (ALS) method

Fixing all but one factor matrix, we have a linear least squares problem:

$$\min_{\hat{U}_1} ||\mathcal{A} - \sum_{\alpha=1}^{k} \hat{U}_1(:,\alpha) \circ U_2(:,\alpha) \circ U_3(:,\alpha)||_F$$

or equivalently

$$\min_{\hat{U}_1} ||A_{(1)} - \hat{U}_1(U_3 \odot U_2)^T||_F$$

ALS works by alternating over factor matrices, updating one at a time.

CP-ALS algorithm

Repeat until maximum iterations reached or no further improvement obtained

1. Solve $U_1(U_3 \odot U_2)^T = A_{(1)}$ for $U_1 \Rightarrow U_1 = A_{(1)}(U_3 \odot U_2)(U_3^T U_3 * U_2^T U_2)^\dagger$
2. Normalize columns of U_1
3. Solve $U_2(U_3 \odot U_1)^T = A_{(2)}$ for $U_2 \Rightarrow U_2 = A_{(2)}(U_3 \odot U_1)(U_3^T U_3 * U_1^T U_1)^\dagger$
4. Normalize columns of U_2
5. Solve $U_3(U_2 \odot U_1)^T = A_{(3)}$ for $U_3 \Rightarrow U_3 = A_{(3)}(U_2 \odot U_1)(U_2^T U_2 * U_1^T U_1)^\dagger$
6. Normalize columns of U_3

Here $A^\dagger$ denotes the Moore–Penrose pseudoinverse of A. We use the following identity to get expressions for U_1, U_2 and U_3:

$$(A \odot B)^T (A \odot B) = A^T A * B^T B$$

ALS for computing a CP decomposition

Algorithm 1 CP-ALS method to compute CP decomposition

```
Require: input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, desired rank $k$, initial factor matrices
  $U_j \in \mathbb{R}^{n_j \times k}$ for $1 \leq j \leq d$
Ensure: $[\![\lambda; U_1, \cdots, U_d]\!]$ : a rank-$k$ CP decomposition of $\mathcal{A}$
  repeat
    for $i = 1$ to $d$ do
      $V \leftarrow U_1^\mathsf{T} U_1 * \cdots * U_{i-1}^\mathsf{T} U_{i-1} U_{i+1}^\mathsf{T} U_{i+1} * \cdots * U_d^\mathsf{T} U_d$
      $U_i \leftarrow A_{(i)}(U_d \odot \cdots \odot U_{i+1} \odot U_{i-1} \odot U_1)$
      $U_i \leftarrow U_i V^\dagger$
      $\lambda \leftarrow$ normalize colums of $U_i$
    end for
  until converge or the maximum number of iterations
```

- The collective operation $A_{(i)}(U_d \odot \cdots \odot U_{i+1} \odot U_{i-1} \odot U_1)$ is known as Matricized tensor times Khatri-Rao product (MTTKRP) computation
- U_j can be chosen randomly or by setting k left singular vectors of $A_{(j)}$ for $1 \leq j \leq d$

Table of Contents

1 CP decomposition

2 Tucker decomposition

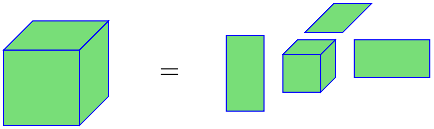
3 Tensor Train decomposition

4 Compact representations of tensor operations

5 Miscellaneous

Tucker decomposition of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$

It represents a tensor with d matrices (usually orthonormal) and a small core tensor.



Tucker decomposition of a 3-dimensional tensor.

$$\mathcal{A} = \mathcal{G} \times_1 U_1 \cdots \times_d U_d$$

$$\mathcal{A}(i_1, \cdots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} \mathcal{G}(\alpha_1, \cdots, \alpha_d) U_1(i_1, \alpha_1) \cdots U_d(i_d, \alpha_d)$$

It can be concisely expressed as $\mathcal{A} = [\![\mathcal{G}; U_1, \cdots, U_d]\!]$.

Here r_j for $1 \leq j \leq d$ denote a set of ranks. Matrices $U_j \in \mathbb{R}^{n_j \times r_j}$ for $1 \leq j \leq d$ are usually orthonormal and known as factor matrices. The tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ is called the core tensor.

Tucker decomposition: an example

Let $\mathcal{A} \in \mathbb{R}^{3\times 3\times 3}$, $\mathcal{G} \in \mathbb{R}^{2\times 2\times 2}$ and $\mathcal{A} = [\![\mathcal{G}; U_1, U_2, U_3]\!]$.

$$U_1 = \frac{1}{3}\begin{pmatrix} 2 & -2 \\ 1 & 2 \\ 2 & 1 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad U_3 = \frac{1}{5}\begin{pmatrix} 0 & 4 \\ 3 & 3 \\ 4 & 0 \end{pmatrix}$$

$$\mathcal{G}(:,:,1) = \begin{pmatrix} 1 & 4 \\ 2 & 5 \end{pmatrix}, \qquad \mathcal{G}(:,:,2) = \begin{pmatrix} 7 & 10 \\ 8 & 11 \end{pmatrix}$$

$$\begin{aligned}
\mathcal{A}(3,2,1) = &\sum_{\alpha_1=1}^{2}\sum_{\alpha_2=1}^{2}\sum_{\alpha_3=1}^{2} \mathcal{G}(\alpha_1,\alpha_2,\alpha_3)U_1(3,\alpha_1)U_2(2,\alpha_2)U_3(1,\alpha_3) \\
=&\mathcal{G}(1,1,1)U_1(3,1)U_2(2,1)U_3(1,1) + \mathcal{G}(1,1,2)U_1(3,1)U_2(2,1)U_3(1,2) \\
&+ \mathcal{G}(1,2,1)U_1(3,1)U_2(2,2)U_3(1,1) + \mathcal{G}(1,2,2)U_1(3,1)U_2(2,2)U_3(1,2) \\
&+ \mathcal{G}(2,1,1)U_1(3,2)U_2(2,1)U_3(1,1) + \mathcal{G}(2,1,2)U_1(3,2)U_2(2,1)U_3(1,2) \\
&+ \mathcal{G}(2,2,1)U_1(3,2)U_2(2,2)U_3(1,1) + \mathcal{G}(2,2,2)U_1(3,2)U_2(2,2)U_3(1,2) \\
=&1\cdot\frac{2}{3}\cdot 0\cdot 0 + 7\cdot\frac{2}{3}\cdot 0\cdot\frac{4}{5} + 4\cdot\frac{2}{3}\cdot 1\cdot 0 + 10\cdot\frac{2}{3}\cdot 1\cdot\frac{4}{5} \\
&+ 2\cdot\frac{1}{3}\cdot 0\cdot 0 + 8\cdot\frac{1}{3}\cdot 0\cdot\frac{4}{5} + 5\cdot\frac{1}{3}\cdot 1\cdot 0 + 11\cdot\frac{1}{3}\cdot 1\cdot\frac{4}{5} = \frac{124}{15}.
\end{aligned}$$

Table of Contents

1 CP decomposition
- Computing CP with Alternating Least Squares

2 Tucker decomposition
- Computing Tucker decomposition

3 Tensor Train decomposition
- Computing Tensor Train decomposition

4 Compact representations of tensor operations

5 Miscellaneous
- Research topics/articles for the project
- Strassen's algorithm: application of CP-decomposition

High Order SVD (HOSVD) for computing a Tucker decomposition

Algorithm 2 HOSVD method to compute a Tucker decomposition

```
Require: input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, desired rank $(r_1, \cdots, r_d)$
Ensure: $\mathcal{A} = \mathcal{G} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d$
  for $k = 1$ to $d$ do
      $U_k \leftarrow r_k$ leading left singular vectors of $A_{(k)}$
  end for
  $\mathcal{G} = \mathcal{A} \times_1 U_1^\mathsf{T} \times_2 U_2^\mathsf{T} \cdots \times_d U_d^\mathsf{T}$
```

- When $r_i < rank(A_{(i)})$ for one or more i, the decomposition is called the truncated-HOSVD (T-HOSVD)
- Output of T-HOSVD can be used as a starting point for an ALS algorithm
- The collective operation $\mathcal{A} \times_1 U_1^\mathsf{T} \times_2 U_2^\mathsf{T} \cdots \times_d U_d^\mathsf{T}$ is known as Multiple Tensor-Times-Matrix (Multi-TTM) computation

Quasi-optimality of T-HOSVD

Let $\tilde{\mathcal{A}} = \mathcal{G} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d$ be the tensor obtained from T-HOSVD.

$$\begin{aligned}
||\mathcal{A} - \tilde{\mathcal{A}}||_F^2 =& ||\mathcal{A} - \mathcal{G} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d||_F^2 = ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \\
=& ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T + \mathcal{A} \times_1 U_1 U_1^T - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \\
=& ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T||_F^2 + ||\mathcal{A} \times_1 U_1 U_1^T - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \\
=& ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T||_F^2 + ||\mathcal{A} \times_1 U_1 U_1^T - \mathcal{A} \times_1 U_1 U_1^T \times_2 U_2 U_2^T||_F^2 + \cdots \\
& \cdots + ||\mathcal{A} \times_1 U_1 U_1^T \cdots \times_{d-1} U_{d-1} U_{d-1}^T - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \\
\leq& ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T||_F^2 + ||\mathcal{A} - \mathcal{A} \times_2 U_2 U_2^T||_F^2 + \cdots + ||\mathcal{A} - \mathcal{A} \times_d U_d U_d^T||_F^2
\end{aligned}$$

Theorem

Tensor $\tilde{\mathcal{A}}$ obtained from T-HOSVD satisfies quasi-optimality condition

$$||A - \tilde{\mathcal{A}}||_F \leq \sqrt{d} ||\mathcal{A} - \mathcal{A}_{best}||_F,$$

where $\mathcal{A}_{best}$ is the best approximation of $\mathcal{A}$ with ranks $(r_1, \cdots, r_d)$.

Proof: $||\mathcal{A} - \mathcal{A} \times_i U_i U_i^T||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F$ for $1 \leq i \leq d$. Substituting these in the previous result yields the specified inequality.

Sequentially T-HOSVD (ST-HOSVD) for Tucker decomposition

- This method is more work efficient than T-HOSVD
- In each step, it reduces the size of one dimension of the tensor

Algorithm 3 ST-HOSVD method to compute a Tucker decomposition

```
**Require:** input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, desired rank $(r_1, \cdots, r_d)$
**Ensure:** $[\![\mathcal{G}; U_1, \cdots, U_d]\!]$ : a $(r_1, \cdots, r_d)$-rank Tucker decomposition of $\mathcal{A}$
  $\mathcal{B} \leftarrow \mathcal{A}$
  **for** $k = 1$ to $d$ **do**
      $S \leftarrow B_{(k)} B_{(k)}^T$
      $U_k \leftarrow r_k$ leading eigen vectors of $S$
      $\mathcal{B} \leftarrow \mathcal{B} \times_k U_k$
  **end for**
  $\mathcal{G} = \mathcal{B}$
```

Quasi-optimality of ST-HOSVD

Let $\tilde{\mathcal{A}} = \mathcal{G} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d$ be the tensor obtained from ST-HOSVD.

$$\begin{aligned} ||\mathcal{A} - \tilde{\mathcal{A}}||_F^2 =& ||\mathcal{A} - \mathcal{G} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d||_F^2 = ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \\ =& ||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T||_F^2 + ||\mathcal{A} \times_1 U_1 U_1^T - \mathcal{A} \times_1 U_1 U_1^T \times_2 U_2 U_2^T||_F^2 + \cdots \\ & \cdots + ||\mathcal{A} \times_1 U_1 U_1^T \cdots \times_{d-1} U_{d-1} U_{d-1}^T - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F^2 \end{aligned}$$

Theorem

Tensor $\tilde{\mathcal{A}}$ obtained from ST-HOSVD satisfies quasi-optimality condition

$$||A - \tilde{\mathcal{A}}||_F \leq \sqrt{d}||\mathcal{A} - \mathcal{A}_{best}||_F,$$

where $\mathcal{A}_{best}$ is the best approximation of $\mathcal{A}$ with ranks $(r_1, \cdots, r_d)$.

Proof: We know that $||\mathcal{A} - \mathcal{A} \times_i U_i U_i^T||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F$ for $1 \leq i \leq d$.

$$||\mathcal{A} - \mathcal{A} \times_1 U_1 U_1^T||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F$$

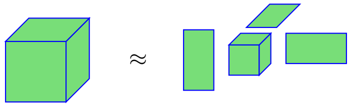
$$||\mathcal{A} \times_1 U_1 U_1^T - \mathcal{A} \times_1 U_1 U_1^T \times_2 U_2 U_2^T||_F \leq ||\mathcal{A} - \mathcal{A} \times_2 U_2 U_2^T||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F$$

$$\vdots$$

$$||\mathcal{A} \times_1 U_1 U_1^T \cdots \times_{d-1} U_{d-1} U_{d-1}^T - \mathcal{A} \times_1 U_1 U_1^T \cdots \times_d U_d U_d^T||_F \leq ||\mathcal{A} - \mathcal{A} \times_d U_d U_d^T||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F$$

Summing the above terms yields the specified inequality.

Tucker decomposition optimization problem for a 3-dimensional tensor



For fixed ranks orthonormal matrices U_1, U_2, U_3, we want to solve

$$\min_{U_1,U_2,U_3} ||\mathcal{A} - \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3||_F, \text{ where } \mathcal{G} = \mathcal{A} \times_1 U_1^T \times_2 U_2^T \times_3 U_3^T.$$

This is equivalent to

$$\max_{U_1,U_2,U_3} ||\mathcal{A} \times_1 U_1^T \times_2 U_2^T \times_3 U_3^T||_F.$$

It is a nonlinear, nonconvex optimization problem.

Higher-order orthogonal iteration (HOOI) method

Fixing all but one factor matrix, we have a matrix problem:

$$\max_{\hat{U}_1} ||\mathcal{A} \times_1 \hat{U}_1{}^T \times_2 U_2^T \times_3 U_3^T||_F$$

HOOI works by alternating over factor matrices, updating one by computing left singular vectors

HOOI method for computing a Tucker decomposition

Algorithm 4 HOOI method to compute Tucker decomposition

```
Require: input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, desired ranks $(r_1, \cdots, r_d)$, initial
  factor matrices $U_j \in \mathbb{R}^{n_j \times r_j}$ for $1 \le j \le d$
Ensure: $[\![\mathcal{G}; U_1, \cdots, U_d]\!]$ : a $(r_1, \cdots, r_d)$-rank Tucker decomposition of $\mathcal{A}$
  repeat
    for $i = 1$ to $d$ do
      $\mathcal{B} \leftarrow \mathcal{A} \times_1 U_1^T \cdots \times_{i-1} U_{i-1}^T \times_{i+1} U_{i+1}^T \cdots \times_d U_d^T$
      $U_i \leftarrow r_i$ left singular vectors of $B_{(i)}$
    end for
  until converge or the maximum number of iterations
  $\mathcal{G} \leftarrow \mathcal{A} \times_1 U_1^T \times_2 U_2^T \cdots \times_d U_d^T$
```

- Outputs of HOSVD (U_j for $1 \le j \le d$) can be used as a starting point for this method

Table of Contents

1 CP decomposition

2 Tucker decomposition

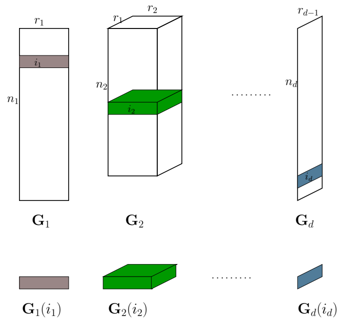
3 Tensor Train decomposition

4 Compact representations of tensor operations

5 Miscellaneous

Tensor Train (TT) decomposition: Product of matrices view

- A *d*-dimensional tensor is represented with 2 matrices and *d*-2 3-dimensional tensors.



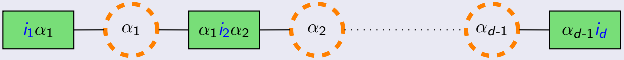
$$\mathbf{A}(i_1, i_2, \cdots, i_d) = \mathbf{G}_1(i_1)\mathbf{G}_2(i_2)\cdots\mathbf{G}_d(i_d)$$

An entry of $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is computed by multiplying corresponding matrix (or row/column) of each matrix/tensor.

Tensor Train decomposition

$\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is represented with cores $\mathcal{G}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$, $k=1,2,\cdots d$, $r_0=r_d=1$ and its elements satisfy the following expression:

$$\begin{aligned}\mathcal{A}(i_1,\cdots,i_d) &= \sum_{\alpha_0=1}^{r_0}\cdots\sum_{\alpha_d=1}^{r_d} \mathcal{G}_1(\alpha_0,i_1,\alpha_1)\cdots\mathcal{G}_d(\alpha_{d-1},i_d,\alpha_d)\\ &= \sum_{\alpha_1=1}^{r_1}\cdots\sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathcal{G}_1(1,i_1,\alpha_1)\cdots\mathcal{G}_d(\alpha_{d-1},i_d,1)\end{aligned}$$



The ranks r_k are called TT-ranks.

- The number of entries in this decomposition = $\mathcal{O}(n_1r_1 + n_2r_1r_2 + n_3r_2r_3 + \cdots + n_{d-1}r_{d-2}r_{d-1} + n_dr_{d-1})$

TT-decomposition: an example

Let $\mathcal{A} \in \mathbb{R}^{3\times4\times5}$. $\mathcal{G}_1 \in \mathbb{R}^{3\times2}, \mathcal{G}_2 \in \mathbb{R}^{2\times4\times2}$, and $\mathcal{G}_3 \in \mathbb{R}^{2\times5}$ are the cores of a TT-decomposition.

$$\mathcal{G}_1 = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}, \quad \mathcal{G}_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$\mathcal{G}_2(:,1,:) = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}, \mathcal{G}_2(:,2,:) = \begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}, \mathcal{G}_2(:,3,:) = \begin{pmatrix} 1 & 1 \\ 4 & 1 \end{pmatrix}, \mathcal{G}_2(:,4,:) = \begin{pmatrix} 1 & 1 \\ 5 & 1 \end{pmatrix}$$

Computation of $\mathcal{A}(2,3,4)$,

$$\begin{aligned} \mathcal{A}(2,3,4) =& \mathcal{G}_1(2,:)\mathcal{G}_2(:,3,:)\mathcal{G}_3(:,4) \\ =& \begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} = 27 \end{aligned}$$

Another representation of unfolding matrices of a tensor

A_k denotes k-th unfolding matrix of tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

$$A_k = [A_k(i_1, i_2, \cdots, i_k; i_{k+1}, \cdots, i_d)]$$

- Size of A_k is $(\prod_{\ell=1}^{k} n_\ell) \times (\prod_{\ell=k+1}^{d} n_\ell)$

Table of Contents

1. CP decomposition
   - Computing CP with Alternating Least Squares
2. Tucker decomposition
   - Computing Tucker decomposition
3. Tensor Train decomposition
   - Computing Tensor Train decomposition
4. Compact representations of tensor operations
5. Miscellaneous
   - Research topics/articles for the project
   - Strassen's algorithm: application of CP-decomposition

TT-SVD algorithm for TT approximation [Oseledets, 2011]

Algorithm 5 TT-SVD algorithm

```
Require: $d$-dimensional tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and desired ranks ($r_0 = 1$,
    $r_1, r_2, \cdots r_{d-1}, r_d = 1$)
Ensure: Cores $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ for $1 \leq k \leq d$ of a TT representation
 1: Temporary tensor: $\mathfrak{C} = \mathcal{A}$
 2: for $k = 1 : d-1$ do
 3:     $A_k = reshape(\mathfrak{C}, r_{k-1}n_k, \frac{numel(\mathfrak{C})}{r_{k-1}n_k})$
 4:     Compute SVD: $A_k = U\Sigma V^T$
 5:     New core: $\mathcal{G}_k := reshape(U(;1:r_k), r_{k-1}, n_k, r_k)$
 6:     $\mathfrak{C} = \Sigma(1:r_k; 1:r_k)V^T(1:r_k;)$
 7: end for
 8: $\mathcal{G}_d = \mathfrak{C}$
 9: return $\mathcal{G}_1, \cdots, \mathcal{G}_d$
```

- $reshape(A, m_1, \cdots, m_\ell)$: rearranges array A into a $m_1 \times \cdots \times m_\ell$ array
- $numel(A)$: number of elements of array A

Error with TT-SVD approximation

Suppose the unfolding matrices of $\mathcal{A}$ satisfy the following:
$A_k = R_k + E_k, \quad R_k$ is the best r_k- rank approximation of $A_k, \quad$ for $1 \leq k \leq d-1$.

The accuracy analysis of TT-SVD is similar to that of ST-HOSVD method (see [Oseledets, 2011]).
Tensor $\mathcal{B}$ obtained from the TT-SVD algorithm satisfies

$$||\mathcal{A} - \mathcal{B}||_F^2 \leq \sum_{k=1}^{d-1} ||E_k||_F^2.$$

Theorem

Tensor $\mathcal{B}$ obtained from TT-SVD satisfies quasi-optimality condition

$$||A - \mathcal{B}||_F \leq \sqrt{d-1}||\mathcal{A} - \mathcal{A}_{best}||_F,$$

where $\mathcal{A}_{best}$ is the best $(r_1, \cdots, r_{d-1})$-ranks approximation of $\mathcal{A}$ in TT-format.

Proof: As SVD gives the best r_k rank approximation for A_k, we have

$$||E_k||_F \leq ||\mathcal{A} - \mathcal{A}_{best}||_F \text{ for } 1 \leq k \leq d.$$

Putting the values of $||E_k||_F$ in the error expression of TT-SVD algorithm completes the proof.

Why TT representation is good for high dimension tensors?

This representation allows one to perform various basic linear algebra operations in its own structure.

- *Addition*: The addition of two tensors in the TT-representation ,

$$\mathcal{A} = \mathcal{A}_1(i_1)\cdots\mathcal{A}_d(i_d), \quad \mathcal{B} = \mathcal{B}_1(i_1)\cdots\mathcal{B}_d(i_d),$$

  requires to merge cores for each mode. Auxiliary dimensions are added. The cores $\mathcal{C}_k(i_k)$ of $\mathcal{C} = \mathcal{A} + \mathcal{B}$ are defined as

$$\mathcal{C}_k(i_k) = \begin{pmatrix} \mathcal{A}_k(i_k) & 0 \\ 0 & \mathcal{B}_k(i_k) \end{pmatrix}, \quad \text{for } 2 \leq k \leq d-1, \text{ and}$$

$$\mathcal{C}_1(i_1) = \begin{pmatrix} \mathcal{A}_1(i_1) & \mathcal{B}_1(i_1) \end{pmatrix}, \quad \mathcal{C}_d(i_d) = \begin{pmatrix} \mathcal{A}_d(i_d) \\ \mathcal{B}_d(i_d) \end{pmatrix}.$$

- *Multiplication by a number*: requires to scale one of the cores
- Multidimensional contraction, Hadamard product and scalar product can also be performed
- Further approximation (or compression) can also be obtained

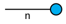
Table of Contents

1. CP decomposition
2. Tucker decomposition
3. Tensor Train decomposition
4. Compact representations of tensor operations
5. Miscellaneous

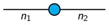
Tensor network representations

Notation: Tensors are denoted by solid shapes and number of lines denote the dimensions of the tensors. Connecting two lines implies summation (or contraction) over the connected dimensions.

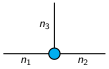
Vector :



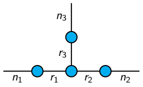
Matrix :



3-dimensional tensor :



Tucker decomposition of a 3-dimensional tensor :



TT decomposition of of a 4-dimensional tensor

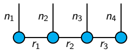


Table of Contents

1 CP decomposition
- Computing CP with Alternating Least Squares

2 Tucker decomposition
- Computing Tucker decomposition

3 Tensor Train decomposition
- Computing Tensor Train decomposition

4 Compact representations of tensor operations

5 Miscellaneous
- Research topics/articles for the project
- Strassen's algorithm: application of CP-decomposition

Course project

- A list of topics/articles is given
- Each student or a group of two students will prepare a 5-6 pages report for the chosen topic/article
- Deadline for submitting the report: Nov 5
- Presentation would be after Nov 5
- Email me your or your group topic/article choices (atleast two) in preference order by Oct 15

If you want to propose another topic or article, your are more than welcome to discuss it with me.

Research topics

- Communication costs of a specific matrix factorization
- Extending a specific matrix factorization for tensors
- Use of tensors in a particular domain
  - Neuroscience, data analysis, molecular simulations, quantum computing, face recognition

What do I expect from you in the report?

- State-of-the-art of the field
- Bottleneck part of the operation
- Your idea of improvement and preliminary work on why it will be effective

Research articles

- Obtain lower bounds on data transfers for various computations on a sequential machine: Automated Derivation of Parametric Data Movement Lower Bounds for Affine Programs
- Performance optimizations for TSQR algorithm: Reconstructing Householder Vectors from Tall-Skinny QR
- Low rank approximation for stencil computations: LoRAStencil: Low-Rank Adaptation of Stencil Computation on Tensor Cores
- Sequential lower bounds and optimal algorithms for symmetric computations: I/O-Optimal Algorithms for Symmetric Linear Algebra Kernels
- Hypergraph partitioning-based methods to improve MTTKRP performance: Scalable Sparse Tensor Decompositions in Distributed Memory Systems
- A parallel method to perform MTTKRP on a parallel shared memory machine: SPLATT: Efficient and Parallel Sparse Tensor-Matrix Multiplication
- Randomization based parallel HOSVD and ST-HOSVD methods: Parallel Randomized Tucker Decomposition Algorithms
- Tucker decomposition to improve performance of convolution kernels: Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network
- Tensor train representation for the weight matrices of the fully connected layers: Tensorizing Neural Networks

Contents of the report for a research article

- The general idea of the work
- A detailed analysis of some parts
- Overview of the state of the art
- Mention why the work of this paper is important
- Your feedback on the work (possible extensions, limitations of the work, ...)
- What challenges you faced while reading the paper (which parts are not clear, explanation is not appropriate, missing information, ...)

Each group (or person) will do a presentation of the selected topic/article for 30-45 minutes, followed by 5-10 minutes of questions/comments.

Table of Contents

1. CP decomposition
   - Computing CP with Alternating Least Squares
2. Tucker decomposition
   - Computing Tucker decomposition
3. Tensor Train decomposition
   - Computing Tensor Train decomposition
4. Compact representations of tensor operations
5. Miscellaneous
   - Research topics/articles for the project
   - Strassen's algorithm: application of CP-decomposition

Strassen's algorithm for matrix multiplication ($C = AB$)

- Matrix is divided into 2×2 blocks

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}$$

$$\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

2×2 Matrix multiplication as a tensor operation

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

We can write this multiplication as a tensor operation,

$$\mathfrak{T} \times_1 \begin{pmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{pmatrix} \times_2 \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} \\ C_{12} \\ C_{21} \\ C_{22} \end{pmatrix}$$

Where $\mathfrak{T}$ is a $4 \times 4 \times 4$ tensor with the following frontal slices:

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} T_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad T_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} T_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2×2 Matrix multiplication as a tensor operation

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

We can write this multiplication as a tensor operation,

$$\mathfrak{T} \times_1 \begin{pmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{pmatrix} \times_2 \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} \\ C_{12} \\ C_{21} \\ C_{22} \end{pmatrix}$$

For example,

$$T_2 \times_1 \begin{pmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{pmatrix} \times_2 \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} = (A_{11} \;\; A_{12} \;\; A_{21} \;\; A_{22}) \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} = A_{11}B_{12} + A_{12}B_{22} = C_{12}$$

Matrix multiplication with CP decomposition

CP decomposition of $\mathfrak{T}$, $\mathfrak{T} = [\![U, V, W]\!]$ can be written as,

$$\mathfrak{T} = \sum_{r=1}^{R} u_r \circ v_r \circ w_r$$

Here u_r, v_r and w_r are the columns of U, V and W, respectively. R is the rank of $\mathfrak{T}$. We can write matrix multiplication as,

$$\begin{aligned}\mathfrak{T} \times_1 \begin{pmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{pmatrix} \times_2 \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} &= \sum_{r=1}^{R} (u_r \circ v_r \circ w_r) \times_1 \begin{pmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{pmatrix} \times_2 \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix} \\ &= \sum_{r=1}^{R} \Big[(A_{11} \;\; A_{12} \;\; A_{21} \;\; A_{22}) u_r (B_{11} \;\; B_{12} \;\; B_{21} \;\; B_{22}) v_r \Big] w_r = \begin{pmatrix} C_{11} \\ C_{12} \\ C_{21} \\ C_{22} \end{pmatrix}\end{aligned}$$

Factor matrices and Strassen's algorithm

Factor matrices,

$$U = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$W = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Strassen's algorithm,

$$\begin{aligned} M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22})B_{11} \\ M_3 &= A_{11}(B_{12} - B_{22}) \\ M_4 &= A_{22}(B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12})B_{22} \\ M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Factor matrices U, V and W construct the algorithm.