Matrix and tensor methods in ML
Lecture 2

Maxim Rakhuba

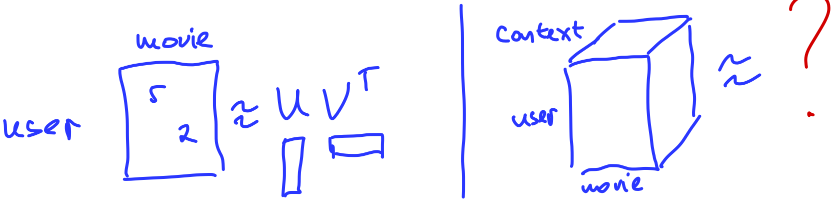
CS department
Higher School of Economics

June 24, 2022

Tensors

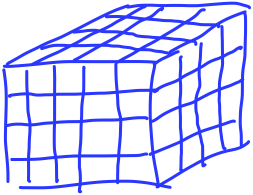
We call $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *a tensor*, d – *dimensionality.*

1)



2) $C \in \mathbb{R}^{k \times k \times m_{in} \times m_{out}}$

3) $\psi(r_1, \dots, r_d)$



$\approx \psi_{i_1 \dots i_d}$

Tensors

We call $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *a tensor*, d – *dimensionality*.

The curse of dimensionality

- Impossible to store n^d entries ($n_1 = \cdots = n_d = n$) for large d.
- **Example:** for $n = 2, d = 300$ number of entries is $2^{300} \gg 10^{80}$ (estimate of the number of atoms in the Universe).

Tensors

We call $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *a tensor*, d – *dimensionality*.

The curse of dimensionality

- Impossible to store n^d entries ($n_1 = \cdots = n_d = n$) for large d.
- **Example:** for $n = 2$, $d = 300$ number of entries is $2^{300} \gg 10^{80}$ (estimate of the number of atoms in the Universe).

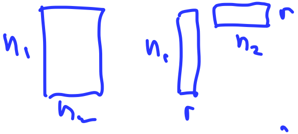
The blessing of dimensionality

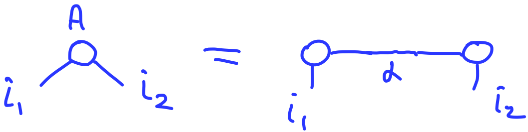
- $a \in \mathbb{R}^N$, $N = n_1 \dots n_d$ can be reshaped to $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.
- By breaking the curse, we can use extreme N, e.g., $N = 10^{30}$.

Tensor decompositions ($d = 2$)

Skeleton decomposition of $A \in \mathbb{R}^{n_1 \times n_2}$:

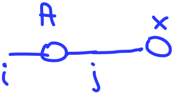
$$A = UV^\top \quad \Longleftrightarrow \quad a(i_1, i_2) = \sum_{\alpha=1}^{r} u_\alpha(i_1)\, v_\alpha(i_2).$$



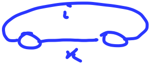


Examples

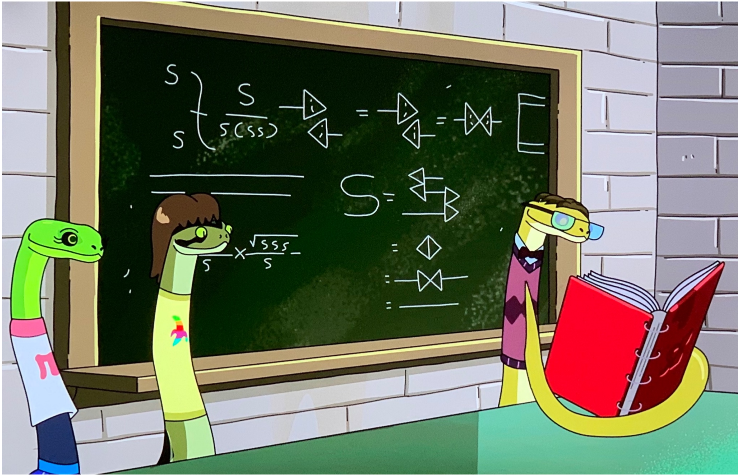
$(Ax)_i = \sum_j a_{ij} x_j$



$\mathrm{Tr}(AB) = \sum a_{ik} b_{ki}$



Tensor decompositions



Tensor decompositions ($d = 2$)

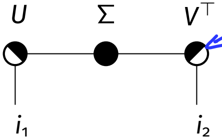
Singular value decomposition (SVD):

$$a(i_1, i_2) = \sum_{\alpha=1}^{r} \sigma_\alpha \, u_\alpha(i_1) \, v_\alpha(i_2).$$

where $U^\intercal U = I_r$, $V^\intercal V = I_r$ and $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$.

- Explicit construction of the best rank-k approximation.
- Robust algorithms for computing SVD.

In tensor diagram notation:



denotes orthogonality

Tensor decompositions

TT-decomposition of $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$:
(Oseledets, Tyrtyshnikov, 2009)

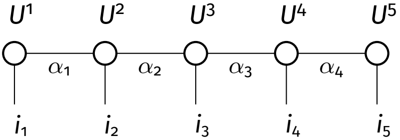
$$\mathcal{X}_{i_1 \ldots i_d} = \sum_{\alpha_1, \ldots, \alpha_{d-1}=1}^{r_1, \ldots, r_{d-1}} U^1_{\alpha_1}(i_1)\, U^2_{\alpha_1 \alpha_2}(i_2)\, U^3_{\alpha_2 \alpha_3}(i_3) \ldots U^d_{\alpha_{d-1}}(i_d).$$

$n^d \quad nr \quad nr^2 \quad nr^2 \ldots \quad nr$

- TT-rank: $\mathbf{r} = (r_1, \ldots, r_{d-1})$
- Storage: $O(dnr^2) \ll n^d$ for small r

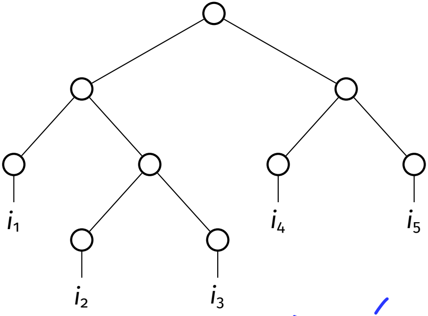
$r = r_1 = \ldots = r_d, \quad n_1 = \ldots = n_d = n$

Tensor diagram representation of TT



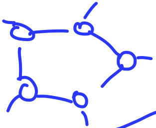
Tensor networks

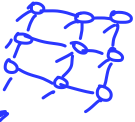
Hierachical Tucker decomposition



Popular tensor networks:

- tensor ring;
- 2D lattice (PEPS);
- ...





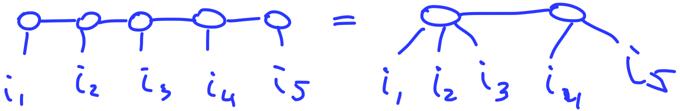
What is the TT rank?

Define k-th unfolding matrix of $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$:

$$A_k := \texttt{reshape}\left(\mathcal{A},\ [n_1 \dots n_k,\ n_{k+1} \dots n_d]\right). \in \mathbb{R}^{(n_1 \cdots n_k) \times (n_{k+1} \cdots n_d)}$$

For TT-rank we have:

$$\text{TT-rank}(\mathcal{A}) = (\mathrm{rank}(A_1), \dots, \mathrm{rank}(A_{d-1}))\,.$$



How to compute the TT decomposition?

$$a(i_1, i_2, i_3) = \sum_{\alpha_1=1}^{r_1} U^1_{\alpha_1}(i_1)\, V^1_{\alpha_1}(i_2, i_3) =$$

$$= U^1 (V^1)^T$$

$$= \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} U^1_{\alpha_1}(i_1)\, U^2_{\alpha_1 \alpha_2}(i_2)\, U^3_{\alpha_2}(i_3)$$

TT-SVD is skeleton decomp.

are approximated using SVD

TT-SVD error bound

Theorem (I. Oseledets, 2011)

Suppose that

$$A_k = R_k + E_k, \quad \operatorname{rank} R_k = r_k, \quad \|E_k\|_F = \varepsilon_k, \quad k = 1, \dots, d-1.$$

Then TT-SVD computes $\mathcal{B}$ with the TT-rank $\{r_1, \dots, r_{d-1}\}$:

$$\|\mathcal{A} - \mathcal{B}\|_F \le \sqrt{\sum_{k=1}^{d-1} \varepsilon_k^2}.$$

Compressing neural network

Recall a convolutional layer

Convolutional layer:

$$\mathcal{Y}(x,y,t) = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1}\sum_{s=1}^{m_{in}} \mathcal{C}(i,j,s,t)\mathcal{X}(x+i,y+j,s).$$

Apply a tensor decomposition to the kernel $\mathcal{C} \in \mathbb{R}^{k\times k\times m_{in}\times m_{out}}$ [1].
Multiple tensor decompositions are applied to $\mathcal{C}$ in [2].

---

[1] Vadim Lebedev et al. "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition". In: *3rd ICLR*. 2015.
[2] Kohei Hayashi et al. "Exploring unexplored tensor network decompositions for convolutional neural networks". In: *NeurIPS* (2019).

Compression via tensorization

$y = Wx + b$

Matrices depend on 2 indices: $\{W(i,j)\}_{i,j=1}^{2^d} \in \mathbb{R}^{2^d \times 2^d}$

Compression via tensorization

Matrices depend on 2 indices: $\{W(i,j)\}_{i,j=1}^{2^d} \in \mathbb{R}^{2^d \times 2^d}$

To apply TT decomposition

1. Reshape to a multidimensional array:

$$\mathcal{W} := \texttt{reshape}(W, \underbrace{[2,\ldots,2]}_{2d}). \in \mathbb{R}^{2\times 2 \cdots \times 2}$$

$$W(i,j) = \mathcal{W}(i_1,\ldots,i_d, j_1,\ldots,j_d)$$

$$\mathcal{W}_d = W \Rightarrow (\text{TT-rank}(\mathcal{W}))_d = \text{rank}(W)$$

identity map has full rank

Compression via tensorization

Matrices depend on 2 indices: $\{W(i,j)\}_{i,j=1}^{2^d} \in \mathbb{R}^{2^d \times 2^d}$

To apply TT decomposition

1. Reshape to a multidimensional array:

$$\mathcal{W} := \texttt{reshape}(W, \underbrace{[2, \dots, 2]}_{2d}).$$

2. Permute indices:

$$\mathcal{W}(i_1, j_1, \dots, i_d, j_d) := \mathcal{W}(i_1, \dots, i_d, j_1, \dots, j_d).$$

Compression via tensorization

Matrices depend on 2 indices: $\{W(i, j)\}_{i,j=1}^{2^d} \in \mathbb{R}^{2^d \times 2^d}$

To apply TT decomposition

1. Reshape to a multidimensional array:

$$\mathcal{W} := \texttt{reshape}(W, \underbrace{[2, \dots, 2]}_{2d}).$$

2. Permute indices:

$$\mathcal{W}(i_1, j_1, \dots, i_d, j_d) := \mathcal{W}(i_1, \dots, i_d, j_1, \dots, j_d).$$

3. Apply a tensor decomposition to $\mathcal{W}$.

Compression via tensorization

Matrices depend on 2 indices: $\{W(i, j)\}_{i,j=1}^{2^d} \in \mathbb{R}^{2^d \times 2^d}$

To apply TT decomposition

1. Reshape to a multidimensional array:

$$\mathcal{W} := \mathtt{reshape}(W, \underbrace{[2, \ldots, 2]}_{2d}).$$

2. Permute indices:

$$\mathcal{W}(i_1, j_1, \ldots, i_d, j_d) := \mathcal{W}(i_1, \ldots, i_d, j_1, \ldots, j_d).$$

3. Apply a tensor decomposition to $\mathcal{W}$.

Used for FC [3] and conv. layers [4].

---

[3] Alexander Novikov et al. "Tensorizing neural networks". In: *NIPS*. 2015, pp. 442–450.
[4] Timur Garipov et al. "Ultimate tensorization: compressing convolutional and fc layers alike". In: *arXiv preprint arXiv:1611.03214* (2016).

Are there more tensor decompositions?

Separation of variables ($d > 2$)

Canonical decomposition (Hitchcock, 1927)

$$a(i_1, i_2, i_3) = \sum_{\alpha=1}^{r} u_\alpha(i_1) v_\alpha(i_2)\, w_\alpha(i_3)$$

- Minimal possible r is called *rank*.

Separation of variables ($d > 2$)

Canonical decomposition (Hitchcock, 1927)

$$a(i_1, \dots, i_d) = \sum_{\alpha=1}^{r} u_{\alpha}^{(1)}(i_1) u_{\alpha}^{(2)}(i_2) \dots u_{\alpha}^{(d)}(i_d).$$

- Minimal possible r is called *rank*.

$A = [\![U^1, U^2, \dots, U^d]\!]$

короткая запись CP

Separation of variables ($d > 2$)

Canonical decomposition (Hitchcock, 1927)

$$a(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} u_\alpha^{(1)}(i_1) u_\alpha^{(2)}(i_2) \ldots u_\alpha^{(d)}(i_d).$$

$S\,S^{-1}$

$A = U V^T =$

$= (US)(VS^{-T})^T$

- Minimal possible r is called *rank*.
- Decomposition is unique under mild conditions.

Separation of variables ($d > 2$)

Canonical decomposition (Hitchcock, 1927)

$$a(i_1, \dots, i_d) = \sum_{\alpha=1}^{r} u_{\alpha}^{(1)}(i_1) u_{\alpha}^{(2)}(i_2) \dots u_{\alpha}^{(d)}(i_d).$$

- Minimal possible r is called *rank*.
- Decomposition is unique under mild conditions.
- Set of tensors with rank $\leq r$ is not closed.

Uniqueness

Definition

Kruskal rank $k(A)$ is maximum value of k such that any k columns of a matrix A are linearly independent.

Theorem

Let $A = [\![U, V, W]\!]$ with (U, V, W have R columns)

$$k(U) + k(V) + k(W) \geq 2R + 2,$$

then the decomposition is unique up to column permutation and diagonal scaling of U, V, W.

$U \to U D_1$
$V \to V D_2$
$W \to W D_3$

diagonal matr.

$: \quad D_1 \cdot D_2 \cdot D_3 = I$

How to compute canonical decomposition?

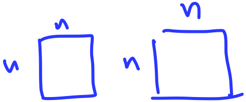
$$J(U, V, W) \equiv \|A - [\![U, V, W]\!]\|_F \to \min_{U,V,W}$$

Alternating least squares

1. Optimize over U with fixed V, W
2. Optimize over V with fixed U, W
3. Optimize over W with fixed U, V

Proceed iteratively.

Complexity of matrix multiplication



$C = A\ B$

$O(n^3)$

Complexity of matrix multiplication [5]

$$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

"Row-by-column":

$$\begin{aligned} C_1 &= A_1 B_1 + A_2 B_3 \\ C_2 &= A_1 B_2 + A_2 B_4 \\ C_3 &= A_3 B_1 + A_4 B_3 \\ C_4 &= A_3 B_2 + A_4 B_4 \end{aligned}$$

8 multiplications and 4 additions

[5] Strassen, V. (1969). Gaussian elimination is not optimal. Numerische mathematik, 13(4), 354-356.

Complexity of matrix multiplication [5]

$$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

$O\left(n^{\log_2 7}\right)$, $\log_2 7 \approx 2.81\ldots < 3$

"Row-by-column":

$$\begin{aligned} C_1 &= A_1B_1 + A_2B_3 \\ C_2 &= A_1B_2 + A_2B_4 \\ C_3 &= A_3B_1 + A_4B_3 \\ C_4 &= A_3B_2 + A_4B_4 \end{aligned}$$

8 multiplications and 4 additions

Strassen:

$$\begin{aligned} M_1 &= (A_1 + A_4)(B_1 + B_4) \\ M_2 &= (A_3 + A_4)B_1 \\ M_3 &= A_1(B_2 - B_4) \\ M_4 &= A_4(B_3 - B_1) \\ M_5 &= (A_1 + A_2)B_4 \\ M_6 &= (A_3 - A_1)(B_1 + B_2) \\ M_7 &= (A_2 - A_4)(B_3 + B_4) \\ C_1 &= M_1 + M_4 - M_5 + M_7 \\ C_2 &= M_3 + M_5 \\ C_3 &= M_2 + M_4 \\ C_4 &= M_1 + M_3 - M_2 + M_6 \end{aligned}$$

7 multiplications and 18 additions

[5] Strassen, V. (1969). Gaussian elimination is not optimal. Numerische mathematik, 13(4), 354-356.

Complexity of matrix multiplication [5]

$$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

"Row-by-column":

$$C_1 = A_1 B_1 + A_2 B_3$$
$$C_2 = A_1 B_2 + A_2 B_4$$
$$C_3 = A_3 B_1 + A_4 B_3$$
$$C_4 = A_3 B_2 + A_4 B_4$$

8 multiplications and 4 additions

$$C_i = \sum_{j,k=1}^{4} x_{ijk} A_j B_k \qquad i = 1, \dots, 4$$

$$x_{ijk} = \sum_{\alpha=1}^{R} u_{i\alpha} v_{j\alpha} w_{k\alpha}$$

(CP decomp.)

$$\Downarrow$$

$$C_i = \sum_{\alpha=1}^{R=7} u_{i\alpha} \Big(\sum_j v_{j\alpha} A_j \Big) \Big(\sum_k w_{k\alpha} B_k \Big)$$

[5] Strassen, V. (1969). Gaussian elimination is not optimal. Numerische mathematik, 13(4), 354-356.