CMSC 5743
Efficient Computing of Deep Neural Networks

Model 02: Low Rank Decomposition

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

(Latest update: October 14, 2024)

2024 Fall

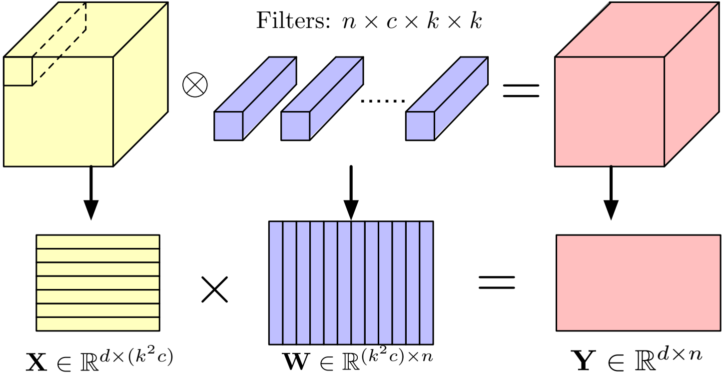
Overview

Overview

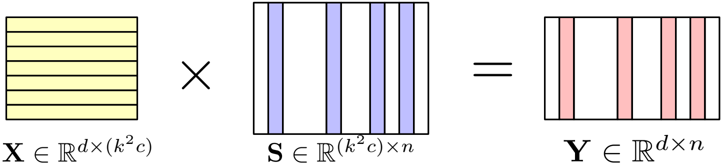
1. Re-visit DNN Pruning
2. Singular Value Decomposition (SVD)
3. How to Compute SVD?
4. Applications to DNN Decomposition
5. Tensor Decomposition
6. Matrix Regression Approach

Re-visit DNN Pruning

`Im2col` (Image2Column) Convolution



- Transform convolution to matrix multiplication
- Unified calculation for both convolution and fully-connected layers

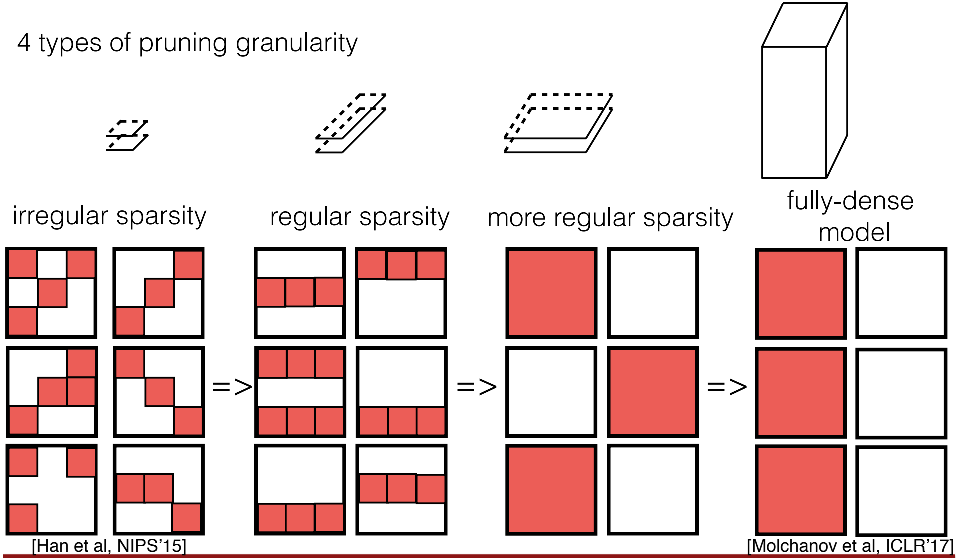


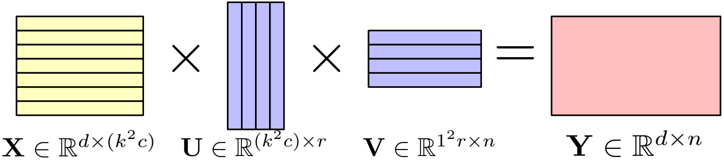
Sparse DNN

- *Sparsification*: weight pruning;
- *Compression*: compressed sparse format for storage;
- *Potential acceleration*: sparse matrix multiplication algorithm.

Exploring the Granularity of Sparsity that is Hardware-friendly

4 types of pruning granularity





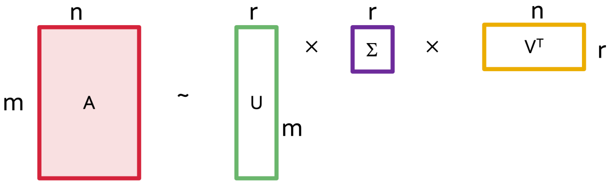
Low-rank DNN

- *Low-rank approximation*: matrix decomposition or tensor decomposition.
- *Compression and acceleration*: less storage required and less FLOP in computation.

Overview

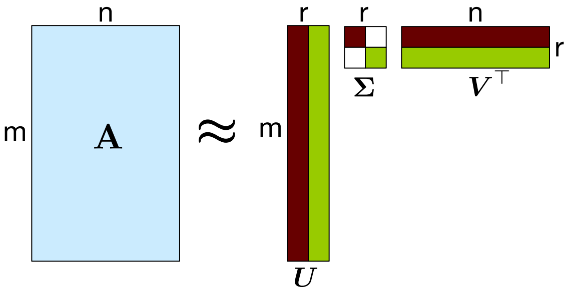
Singular Value Decomposition (SVD)

Reducing Matrix Dimension



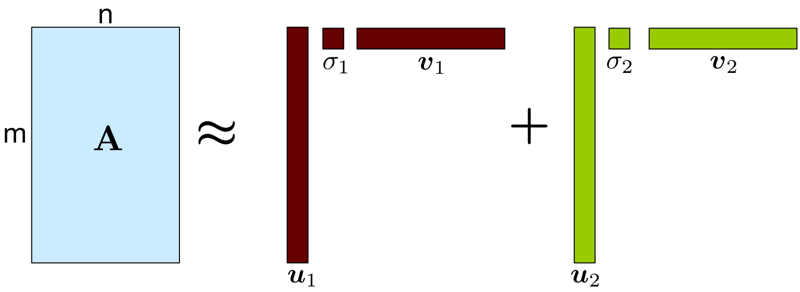
- Gives a decomposition of any matrix into a product of three matrices.
- There are strong constraints on the form of each of these matrices
- Results in a unique decomposition
- From this decomposition, you can choose any number r of intermediate concepts (latent factors)
- In a way that minimizes the reconstruction error

$$A \approx U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i \otimes v_i^\top$$



- A: input data matrix ($m \times n$ matrix)
- U: left singular vectors ($m \times r$ matrix)
- Σ: Singular values ($r \times r$ diagonal matrix)
- V: right singular vectors ($n \times r$ matrix)

$$A \approx U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i \otimes v_i^\top$$



- σ_i: scalar
- u_i, v_i: vector
- If we set $\sigma_2 = 0$, then the gree columns may as well not exist.

SVD: Properties

Property

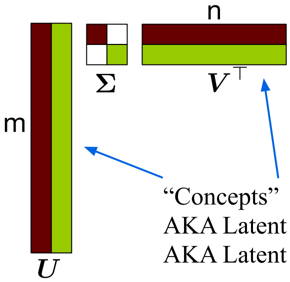
It is always possible to decompose a real matrix $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$

- $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$: unique
- $\boldsymbol{U}, \boldsymbol{V}$: column orthonormal
  - $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}; \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$ ($\boldsymbol{I}$: identity matrix)
  - Columns are orthogonal unit vectors
- $\boldsymbol{\Sigma}$: diagonal
  - Entries (singular values) are non-negative
  - Sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$)

https://www.cs.cornell.edu/courses/cs322/2008sp/stuff/TrefethenBau_Lec4_SVD.pdf

Consider a matrix. What does SVD do?

$$
\begin{array}{c}
 \\ \text{SciFi} \\ \\ \\ \\ \text{Romance} \\ \\
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
$$



“Concepts”
AKA Latent dimensions
AKA Latent factors

- Ratings matrix where each column corresponds to a movie and each row to a user.
- First 4 users prefer SciFi, while others prefer Romance.

Example: Users-to-Movies

$$
\begin{array}{c}
\\ \text{SciFi} \\ \\ \\ \text{Romance} \\ \\ \\
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & -\mathbf{0.59} & \mathbf{0.65} \\
0.07 & -\mathbf{0.73} & -\mathbf{0.67} \\
0.07 & -\mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
$$

$$
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69} \\
0.40 & -\mathbf{0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

- $\boldsymbol{U}$: "user-to-concept" factor matrix
- $\boldsymbol{V}$: "movie-to-concept" factor matrix
- $\boldsymbol{\Sigma}$: its diagonal elements: "strength" of each concept

Example: Users-to-Movies

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{array}{c} \text{SciFi} \\ \\ \\ \\ \text{Romance} \\ \\ \\ \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{array}{c}
\begin{array}{cc} \text{SciFi-concept} & \text{Romance-concept} \end{array} \\
\begin{bmatrix}
\mathbf{0.13} & 0.02 \\
\mathbf{0.41} & 0.07 \\
\mathbf{0.55} & 0.09 \\
\mathbf{0.68} & 0.11 \\
0.15 & -\mathbf{0.59} \\
0.07 & -\mathbf{0.73} \\
0.07 & -\mathbf{0.29}
\end{bmatrix}
\end{array}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 \\
0 & \mathbf{9.5} \\
0 & 0
\end{bmatrix}
$$

$$
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69}
\end{bmatrix}
$$

- $\boldsymbol{U}$: "user-to-concept" factor matrix
- $\boldsymbol{V}$: "movie-to-concept" factor matrix
- $\boldsymbol{\Sigma}$: its diagonal elements: "strength" of each concept

Example: Users-to-Movies

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{array}{c} \\ \text{SciFi} \\ \\ \\ \text{Romance} \\ \\ \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\overset{\text{“strength” of SciFi-concept}}{\mathbf{12.4}} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
$$

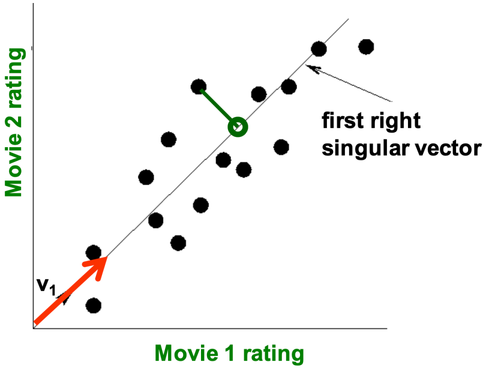
$$
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

- $\boldsymbol{U}$: “user-to-concept” factor matrix
- $\boldsymbol{V}$: “movie-to-concept” factor matrix
- $\boldsymbol{\Sigma}$: its diagonal elements: “strength” of each concept

$$\begin{array}{c|ccccc|} & \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \\ \hline & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \text{SciFi} & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ & 0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\ \text{Romance} & 0 & 0 & 0 & 5 & 5 \\ & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{array} = \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & -\mathbf{0.59} & \mathbf{0.65} \\ 0.07 & -\mathbf{0.73} & -\mathbf{0.67} \\ 0.07 & -\mathbf{0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix}$$

$$\times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69} \\ 0.40 & -\mathbf{0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

- $\boldsymbol{U}$: “user-to-concept” factor matrix
- $\boldsymbol{V}$: “movie-to-concept” factor matrix
- $\boldsymbol{\Sigma}$: its diagonal elements: “strength” of each concept

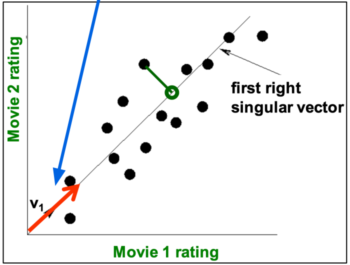


- Instead of using two coordinates (x, y) to describe points
- Let's use only one coordinate
- Point't position is its location along vector v_1

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{array}{c} \uparrow \\ \text{SciFi} \\ \downarrow \\ \\ \uparrow \\ \text{Romance} \\ \downarrow \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & -\mathbf{0.59} & \mathbf{0.65} \\
0.07 & -\mathbf{0.73} & -\mathbf{0.67} \\
0.07 & -\mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69} \\
0.40 & -\mathbf{0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SVD: Dimensionality Reduction

$$
\begin{array}{c}
\begin{array}{cc}
 & \begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{array}{c} \text{SciFi} \\ \\ \\ \\ \text{Romance} \\ \\ \\ \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & -\mathbf{0.59} & \mathbf{0.65} \\
0.07 & -\mathbf{0.73} & -\mathbf{0.67} \\
0.07 & -\mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\\
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69} \\
0.40 & -\mathbf{0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
\end{array}
$$

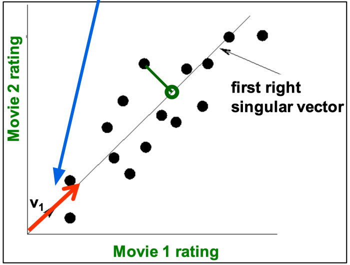


SVD: Dimensionality Reduction

$$
\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \\
\hline
\text{SciFi} & & & & & \\
\text{Romance} & & & & &
\end{array}
$$

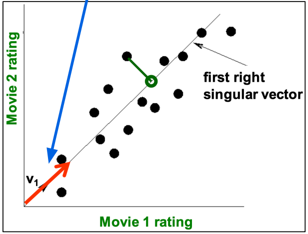
$$
\left[\begin{array}{ccccc}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{array}\right]
=
\left[\begin{array}{ccc}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{array}\right]
\times
\left[\begin{array}{ccc}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{array}\right]
\times
\left[\begin{array}{ccccc}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{array}\right]
$$

Variance on the axis



$$
\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \\
\hline
\text{SciFi} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\text{SciFi} & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\text{SciFi} & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\text{SciFi} & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
\text{Romance} & 0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
\text{Romance} & 0 & 0 & 0 & 5 & 5 \\
\text{Romance} & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{array}
$$

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\boxed{\mathbf{0.56} \quad \mathbf{0.59} \quad \mathbf{0.56} \quad 0.09 \quad 0.09} \\
0.12 \quad -0.02 \quad 0.12 \quad \mathbf{-0.69} \quad \mathbf{-0.69} \\
0.40 \quad \mathbf{-0.80} \quad 0.40 \quad 0.09 \quad 0.09
\end{bmatrix}
$$



Q: How to coordinates of the points in the projection axis?

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{array}{c} \text{SciFi} \\ \\ \\ \\ \text{Romance} \\ \\ \\ \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & -\mathbf{0.59} & \mathbf{0.65} \\
0.07 & -\mathbf{0.73} & -\mathbf{0.67} \\
0.07 & -\mathbf{0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
$$

$$
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -\mathbf{0.69} & -\mathbf{0.69} \\
0.40 & -\mathbf{0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

- Set small singular values to zero
- The above example is rank-2 approximation
- We could also do rank-1 approximation.
- The larger the rank, the more accurate the approximation

SVD: Another Interpretation

$$
\begin{array}{c}
 \\
\text{SciFi} \\
 \\
 \\
 \\
\text{Romance} \\
 \\
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Avatar} & \text{Dune} & \text{Family Man} & \text{5cm per second} \end{array} \\
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & 2 & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & 5 & 5 \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
\approx
\begin{bmatrix}
\mathbf{0.13} & 0.02 & \cancel{-0.01} \\
\mathbf{0.41} & 0.07 & \cancel{-0.03} \\
\mathbf{0.55} & 0.09 & \cancel{-0.04} \\
\mathbf{0.68} & 0.11 & \cancel{-0.05} \\
0.15 & \mathbf{-0.59} & \cancel{\mathbf{0.65}} \\
0.07 & \mathbf{-0.73} & \cancel{\mathbf{-0.67}} \\
0.07 & \mathbf{-0.29} & \cancel{\mathbf{0.32}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
\cancel{0.40} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{0.09} & \cancel{0.09}
\end{bmatrix}
$$

- Set small singular values to zero
- The above example is rank-2 approximation
- We could also do rank-1 approximation.
- The larger the rank, the more accurate the approximation

SVD Reconstruction Error

- Reconstruction error is quantified by the Frobenius norm:

$$||\boldsymbol{M}||_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$||\boldsymbol{A} - \boldsymbol{B}||_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2}$$

SVD Reconstruction Error

- Reconstruction error is quantified by the Frobenius norm:

$$||\boldsymbol{M}||_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$||\boldsymbol{A} - \boldsymbol{B}||_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2}$$

Theorem:

The best rank-1 approximation to $\boldsymbol{A}$ is $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top$, where σ_1 is the largest sngular value, $\boldsymbol{u}_1$ is the first left singular vector, and $\boldsymbol{v}_1$ is the first right singular vector of $\boldsymbol{A}$.

What's a good value for r?

- Let the energy of a set of singular values be the sum of their squares.
- Pick r so the retained singular values have at least 90% of the total energy

What's a good value for r?

- Let the energy of a set of singular values be the sum of their squares.
- Pick r so the retained singular values have at least 90% of the total energy

Example:

- With singular values 12.4, 9.5, and 1.3, total energy = 245.7
- If we drop 1.3, whose square is only 1.7
- We are left with energy 244, or over 99% of the total

Please prove: $AA^\top$ and $A^\top A$ are with the same eigenvalues.

Proof:

Overview

How to Compute SVD?

Finding Eigenpairs

- First we need finding the principal eigenvalue (the largest one)
- and the corresponding eigenvector of a symmetric matrix
- Note: $\boldsymbol{M}$ is symmetric if $M_{ij} = M_{ji}$ for all i and j

Method[1]:

1. Start with any "guess eigenvecor" $\boldsymbol{x}_0$
2. Construct $\boldsymbol{x}_{k+1} = \frac{\boldsymbol{M}\boldsymbol{x}_k}{||\boldsymbol{M}\boldsymbol{x}_k||}$, for $k = 0, 1, \ldots$ [2]
3. Stop when consecutive $\boldsymbol{x}_i$ show little change

[1] Power iteration: https://en.wikipedia.org/wiki/Power_iteration
[2] $||..||$ denotes the Frobenius norm

$$\boldsymbol{M} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Finding Eigenpairs: Example

$$\boldsymbol{M} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\frac{\boldsymbol{M}\boldsymbol{x}_0}{\|\boldsymbol{M}\boldsymbol{x}_0\|} = \begin{bmatrix} 3 \\ 5 \end{bmatrix} / \sqrt{34} = \begin{bmatrix} 0.51 \\ 0.86 \end{bmatrix} = \boldsymbol{x}_1$$

$$\boldsymbol{M} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\frac{\boldsymbol{M}\boldsymbol{x}_0}{\|\boldsymbol{M}\boldsymbol{x}_0\|} = \begin{bmatrix} 3 \\ 5 \end{bmatrix} / \sqrt{34} = \begin{bmatrix} 0.51 \\ 0.86 \end{bmatrix} = \boldsymbol{x}_1$$

$$\frac{\boldsymbol{M}\boldsymbol{x}_1}{\|\boldsymbol{M}\boldsymbol{x}_1\|} = \begin{bmatrix} 2.23 \\ 3.60 \end{bmatrix} / \sqrt{17.93} = \begin{bmatrix} 0.53 \\ 0.85 \end{bmatrix} = \boldsymbol{x}_2$$

Finding the Principal Eigenvalue

With principal eigenvector $\boldsymbol{x}$, we can find eigenvalue λ by

$$\lambda = \boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x}$$

Finding the Principal Eigenvalue

With principal eigenvector $\boldsymbol{x}$, we can find eigenvalue λ by

$$\lambda = \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}$$

Prove:

1. We know $\boldsymbol{x} \cdot \lambda = \boldsymbol{M}\boldsymbol{x}$ if λ is the eigenvalue
2. Multiply both sides by $\boldsymbol{x}^\top$ on the left
3. Since $\boldsymbol{x}^\top \boldsymbol{x} = 1$, we have $\lambda = \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}$

Finding the Principal Eigenvalue

With principal eigenvector $\boldsymbol{x}$, we can find eigenvalue λ by

$$\lambda = \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}$$

Prove:

1. We know $\boldsymbol{x} \cdot \lambda = \boldsymbol{M}\boldsymbol{x}$ if λ is the eigenvalue
2. Multiply both sides by $\boldsymbol{x}^\top$ on the left
3. Since $\boldsymbol{x}^\top \boldsymbol{x} = 1$, we have $\lambda = \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}$

Example: if we take $\boldsymbol{x}^\top = [0.53, 0.85]$, then

$$\lambda = [0, 530.85] \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 0.53 \\ 0.85 \end{bmatrix} = 4.25$$

Finding More Eigenpairs

1. Eliminate the portion of the matrix $\boldsymbol{M}$ that can be generated by the first eigenpair, λ and $\boldsymbol{x}$:

$$\boldsymbol{M}* = \boldsymbol{M} - \lambda \boldsymbol{x}\boldsymbol{x}^\top$$

2. Recursively find the principal eigenpair for $\boldsymbol{M}*$
3. Eliminate the effect of that pair, and so on

Finding More Eigenpairs

1. Eliminate the portion of the matrix $\boldsymbol{M}$ that can be generated by the first eigenpair, λ and $\boldsymbol{x}$:

$$\boldsymbol{M}* = \boldsymbol{M} - \lambda \boldsymbol{x}\boldsymbol{x}^\top$$

2. Recursively find the principal eigenpair for $\boldsymbol{M}*$
3. Eliminate the effect of that pair, and so on

Example:

$$\boldsymbol{M}* = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} - 4.25 \begin{bmatrix} 0.53 \\ 0.85 \end{bmatrix} [0.530.85] = \begin{bmatrix} -0.19 & 0.09 \\ 0.09 & 0.07 \end{bmatrix}$$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = (\boldsymbol{V}^\top)^\top\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U\Sigma V}^\top)^\top = (\boldsymbol{V}^\top)^\top \boldsymbol{\Sigma}^\top \boldsymbol{U}^\top = \boldsymbol{V\Sigma U}^\top$
3. $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{V\Sigma U}^\top \boldsymbol{U\Sigma V}^\top = \boldsymbol{V\Sigma}^2 \boldsymbol{V}^\top$
   - Note: $\boldsymbol{U}$ is orthonormal, so $\boldsymbol{U}^\top \boldsymbol{U}$ is an identity matrix
   - Note: $\boldsymbol{\Sigma}^2$ is a diagonal matrix whose i-th element is the square of the i-th element of $\boldsymbol{\Sigma}$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = (\boldsymbol{V}^\top)^\top\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top$
3. $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$
   - Note: $\boldsymbol{U}$ is orthonormal, so $\boldsymbol{U}^\top\boldsymbol{U}$ is an identity matrix
   - Note: $\boldsymbol{\Sigma}^2$ is a diagonal matrix whose i-th element is the square of the i-th element of $\boldsymbol{\Sigma}$
4. $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2$
   - Note: $\boldsymbol{V}$ is also orthonormal

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = (\boldsymbol{V}^\top)^\top\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top$
3. $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$
   - Note: $\boldsymbol{U}$ is orthonormal, so $\boldsymbol{U}^\top\boldsymbol{U}$ is an identity matrix
   - Note: $\boldsymbol{\Sigma}^2$ is a diagonal matrix whose i-th element is the square of the i-th element of $\boldsymbol{\Sigma}$
4. $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2$
   - Note: $\boldsymbol{V}$ is also orthonormal
5. We can find $\boldsymbol{V}\&\boldsymbol{\Sigma}$ by finding the eigenpairs for $\boldsymbol{A}^\top\boldsymbol{A}$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = (\boldsymbol{V}^\top)^\top\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top$
3. $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$
   - Note: $\boldsymbol{U}$ is orthonormal, so $\boldsymbol{U}^\top\boldsymbol{U}$ is an identity matrix
   - Note: $\boldsymbol{\Sigma}^2$ is a diagonal matrix whose i-th element is the square of the i-th element of $\boldsymbol{\Sigma}$
4. $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2$
   - Note: $\boldsymbol{V}$ is also orthonormal
5. We can find $\boldsymbol{V}\&\boldsymbol{\Sigma}$ by finding the eigenpairs for $\boldsymbol{A}^\top\boldsymbol{A}$
6. Symmetric argument: $\boldsymbol{A}\boldsymbol{A}^\top$ gives us $\boldsymbol{U}$

How to Compute the SVD

1. Start by supposing $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
2. $\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = (\boldsymbol{V}^\top)^\top\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top$
3. $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top$
   - Note: $\boldsymbol{U}$ is orthonormal, so $\boldsymbol{U}^\top\boldsymbol{U}$ is an identity matrix
   - Note: $\boldsymbol{\Sigma}^2$ is a diagonal matrix whose i-th element is the square of the i-th element of $\boldsymbol{\Sigma}$
4. $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2$
   - Note: $\boldsymbol{V}$ is also orthonormal
5. We can find $\boldsymbol{V}\&\boldsymbol{\Sigma}$ by finding the eigenpairs for $\boldsymbol{A}^\top\boldsymbol{A}$
6. Symmetric argument: $\boldsymbol{A}\boldsymbol{A}^\top$ gives us $\boldsymbol{U}$

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Solution:

$$\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Solution:

$$\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Solution:

$$\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

$\rightarrow \lambda_1 = \boldsymbol{v}_1^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{v}_1 = 10$; therefore $\sigma_1 = \sqrt{\lambda_1} = \sqrt{10}$

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Solution:

$$\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

$\rightarrow \lambda_1 = \boldsymbol{v}_1^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{v}_1 = 10$; therefore $\sigma_1 = \sqrt{\lambda_1} = \sqrt{10}$

$\rightarrow \lambda_2 = \boldsymbol{v}_2^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{v}_2 = 0$; therefore $\sigma_2 = \sqrt{\lambda_2} = 0$

Example:

Calculate SVD of matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix}$.

Solution:

$$\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

$$\boldsymbol{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

$\rightarrow \lambda_1 = \boldsymbol{v}_1^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{v}_1 = 10$; therefore $\sigma_1 = \sqrt{\lambda_1} = \sqrt{10}$

$\rightarrow \lambda_2 = \boldsymbol{v}_2^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{v}_2 = 0$; therefore $\sigma_2 = \sqrt{\lambda_2} = 0$

...

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \begin{bmatrix} 1/\sqrt{5} & -2/\sqrt{5} & 0 \\ 2/\sqrt{5} & 1/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

SVD: Complexity

To compute the full SVD using specialized methods:

- $\mathcal{O}(nm^2)$ or $\mathcal{O}(n^2m)$ (whichever is less)

But:

- Less work, if we just want singular values
- Or if we want the first k singular vectors
- Or if the matrix is sparse

Implemented in linear algebra packages link

- LINPACK, Matlab, SPlus, Mathematica, ...

Applications to DNN Decomposition

Low Rank Approximation for Conv[3]

This CVPR2015 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

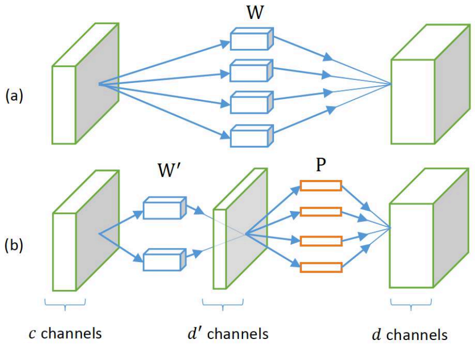
Efficient and Accurate Approximations of Nonlinear Convolutional Networks

Xiangyu Zhang[1*] Jianhua Zou[1] Xiang Ming[1*] Kaiming He[2] Jian Sun[2]
[1]Xi'an Jiaotong University [2]Microsoft Research

[3]Xiangyu Zhang et al. (2015). "Efficient and accurate approximations of nonlinear convolutional networks". In: *Proc. CVPR*, pp. 1984–1992.

Low Rank Approximation for Conv

- Layer responses lie in a low-rank subspace
- Decompose a convolutional layer with d filters with filter size $k \times k \times c$ to
  - A layer with d' filters ($k \times k \times c$)
  - A layer with d filter ($1 \times 1 \times d'$)



Zhang et al Efficient and Accurate Approximations of Nonlinear Convolutional Networks CVPR'15

Low Rank Approximation for Conv

speedup	rank sel.	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	err. ↑ %
2×	no	32	110	199	219	219	219	219	1.18
2×	**yes**	32	83	182	211	239	237	253	**0.93**
2.4×	no	32	96	174	191	191	191	191	1.77
2.4×	**yes**	32	74	162	187	207	205	219	**1.35**
3×	no	32	77	139	153	153	153	153	2.56
3×	**yes**	32	62	138	149	166	162	167	**2.34**
4×	no	32	57	104	115	115	115	115	4.32
4×	**yes**	32	50	112	114	122	117	119	**4.20**
5×	no	32	46	83	92	92	92	92	6.53
5×	**yes**	32	41	94	93	98	92	90	**6.47**

Zhang et al Efficient and Accurate Approximations of Nonlinear Convolutional Networks CVPR'15

Singular Value Decomposition[4]

Published as a conference paper at ICLR 2016

Convolutional Neural Networks With Low-rank Regularization

Cheng Tai[1], **Tong Xiao**[2], **Yi Zhang**[3], **Xiaogang Wang**[2], **Weinan E**[1]
[1]The Program in Applied and Computational Mathematics, Princeton University
[2]Department of Electronic Engineering, The Chinese University of Hong Kong
[3]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor
{chengt,weinan}@math.princeton.edu; yeezhang@umich.edu
{xiaotong,xgwang}@ee.cuhk.edu.hk

- A new algorithm for computing the low-rank tensor decomposition
- A new method for training low-rank constrained CNNs from scratch

[4]Cheng Tai et al. (2016). "Convolutional neural networks with low-rank regularization". In: *Proc. ICLR*.

Pretrained CNN Approximation

- Convolution Calculation

$$\mathcal{F}_n(x,y) = \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} \mathcal{Z}^c\left(x', y'\right) \mathcal{W}_n^c\left(x - x', y - y'\right)$$

- $\mathcal{W}_n \in \mathbb{R}^{d \times d \times C}$ to represent the n -th filter.$\mathcal{Z} \in \mathbb{R}^{X \times Y \times U}$ be the input feature map.
- An approximation of W

$$\tilde{\mathcal{W}}_n^c = \sum_{k=1}^{K} \mathcal{H}_n^k \left(\mathcal{V}_k^c\right)^T$$

  where K is a hyper-parameter controlling the rank, $\mathcal{H} \in \mathbb{R}^{N \times 1 \times d \times R}$ is the horizontal filter, $\mathcal{V} \in \mathbb{R}^{K \times d \times 1 \times C}$ is the vertical filter (Notes: $\mathcal{H}_k^c$ and $\mathcal{V}_k^c$ are both vectors in $\mathbb{R}^d$). Both $\mathcal{H}$ and $\mathcal{V}$ are learnable parameters.
- Then the convolution becomes

$$\tilde{W}_n * \mathcal{Z} = \sum_{c=1}^{C} \sum_{k=1}^{K} \mathcal{H}_n^k \left(\mathcal{V}_k^c\right)^T * \mathcal{Z}^c = \sum_{k=1}^{K} \mathcal{H}_n^k * \left(\sum_{c=1}^{C} \mathcal{V}_k^c * \mathcal{Z}^c\right)$$

Complexity Analysis

- Standard Convolution Complexity: $O(d^2NCXY)$ operations
- Approximation Scheme Complexity
  The computational cost associated with the vertical filters is $O(dKCXY)$ and with horizational fileters is $O(dNKXY)$, a total computational cost is $O(dK(N+C)XY)$
- If $K < \frac{dNC}{N+C}$, acceleration can be achieved

Approximate Parameters H and V

- Minimizing the objective function

$$E_1(\mathcal{H},\mathcal{V}) := \sum_{n,c} \left\| \mathcal{W}_n^c - \sum_{k=1}^{K} \mathcal{H}_n^k \left(\mathcal{V}_k^c\right)^T \right\|_F^2$$

- Theorem: Define the following bijection that maps a tensor to a matrix $\mathcal{T} : \mathbb{R}^{C\times d\times d\times N} \mapsto \mathbb{R}^{Cd\times dN}$, tensor element (i_1, i_2, i_3, i_4) maps to (j_1, j_2) , where

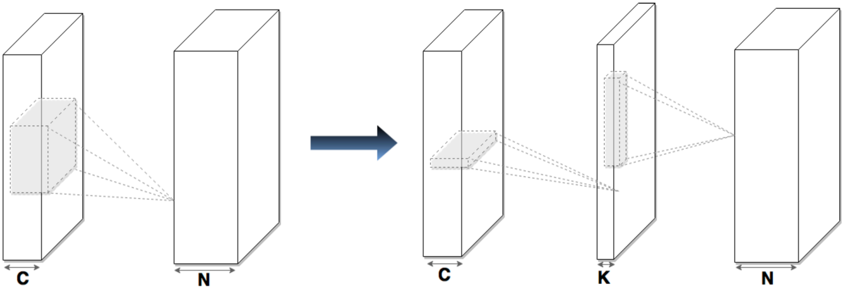
$$j_1 = (i_1 - 1)\,d + i_2, \quad j_2 = (i_4 - 1)\,d + i_3$$

Define $W := \mathcal{T}[\mathcal{W}]$. Let $W = UDQ^T$ be the singular Value Decomposition (SVD) of W. Let

$$\hat{\mathcal{V}}_k^c(j) = U_{(c-1)d+j,k}\sqrt{D_{k,k}}$$
$$\hat{\mathcal{H}}_n^k(j) = Q_{(n-1)d+j,k}\sqrt{D_{k,k}}$$

then $(\hat{\mathcal{H}}, \hat{\mathcal{V}})$ is a solution to minimizing the object function

- The proposed parametrization for low-rank regularization.

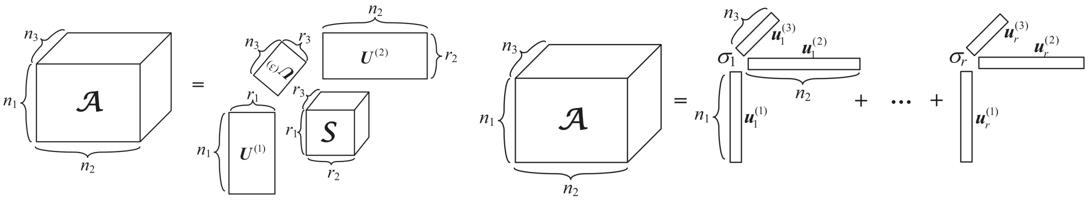


Left: The original convolutional layer. Right: low-rank constraint convolutional layer with rank-K.

Training Low-rank Constrained CNN From Scratch

- The effect of SVD Decomposition
  Each convolutional layer is parameterized as the composition of two convolutional layers,
- Exploding and vanishing gradients expecially for large networks
- Batch Normalition can handle this problem
  (Recall the theory of Batch Normalization)

Tensor Decomposition



(a) Tucker-decomposition (b) CP-decomposition

Tucker Decomposition[5]

Published as a conference paper at ICLR 2016

COMPRESSION OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR FAST AND LOW POWER MOBILE APPLICATIONS

Yong-Deok Kim[1], **Eunhyeok Park**[2], **Sungjoo Yoo**[2], **Taelim Choi**[1], **Lu Yang**[1] **& Dongjun Shin**[1]

[1]Software R&D Center, Device Solutions, Samsung Electronics, South Korea
{yd.mlg.kim, tl.choi, lu2014.yang, d.j.shin}@samsung.com

[2]Department of Computer Science and Engineering, Seoul National University, South Korea
{canusglow, sungjoo.yoo}@gmail.com

- Propose a one-shot whole network compression scheme which consists of simple three steps: (1) rank selection, (2) low-rank tensor decomposition, and (3) fine-tuning.
- Tucker decomposition (Tucker, 1966) with the rank determined by a global analytic solution of variational Bayesian matrix factorization is applied on each kernel tensor.

[5]Yong-Deok Kim et al. (2016). "Compression of deep convolutional neural networks for fast and low power mobile applications". In: *Proc. ICLR*.

Kernel Tensor Approximation

- Convolution Calculation

$$\mathcal{Y}_{h',w',t} = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{s=1}^{S}\mathcal{K}_{i,j,s,t}\mathcal{X}_{h_i,w_j,s}$$

$$h_i = (h'-1)\,\Delta + i - P \text{ and } w_j = (w'-1)\,\Delta + j - P$$

where $\mathcal{K}$ is a 4-way kernel tensor of size $D \times D \times S \times T$, δ is stride, and P is zero-padding size

- Tucker Decomposition:The rank-$(R_1; R_2; R_3; R_4)$ Tucker decomposition of 4-way kernel tensor K has the form:

$$\mathcal{K}_{i,j,s,t} = \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_9=1}^{R_3}\sum_{r_4=1}^{R_4}\mathcal{C}'_{r_1,r_2,r_3,r_4}U^{(1)}_{i,r_1}U^{(2)}_{j,r_2}U^{(3)}_{s,r_3}U^{(4)}_{t,r_4}$$

where $\mathcal{C}'$ is a core tensor of size $R_1 \times R_2 \times R_3 \times R_4$ and $U^{(1)}, U^{(2)}, U^{(3)}$, and $U^{(4)}$ are factor matrices of sizes $D \times R_1, D \times R_2, S \times R_3$, and $T \times R_4$, respectively.

Tucker Decomposition

- Every mode does not have to be decomposed(e.g. For example, we do not decompose mode-1 and mode-2 which are associated with spatial dimensions because they are already quite small).
- Under this variant called Tucker-2 decomposition, the kernel tensor is decomposed to:

$$\mathcal{K}_{i,j,s,t} = \sum_{r_0=1}^{R_0} \sum_{r_4=1}^{R_4} \mathcal{C}_{i,j,r_3,r_4} U^{(3)}_{s,r_0} U^{(4)}_{t,r_4}$$

  where $\mathcal{C}$ is a core tensor of size $D \times D \times R_3 \times R_4$
- With the approximation of kernel, the convolution is as following:

$$\mathcal{Z}_{h,w,r_3} = \sum_{s=1}^{S} U^{(3)}_{s,r_3} \mathcal{X}_{h,w,s}$$

$$\mathcal{Z}'_{h',w',r_4} = \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{r_0=1}^{R_0} \mathcal{C}_{i,j,r_3,r_4} \mathcal{Z}_{h_t,w_j,r_9}$$

R_4

- Tucker-2 decompositions for speeding-up a convolution

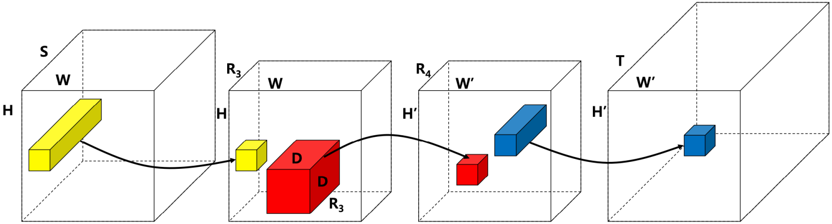


Figure 3: Tucker-2 decompositions for speeding-up a convolution. Each transparent box corresponds to 3-way tensor $\mathcal{X}$, $\mathcal{Z}$, $\mathcal{Z}'$, and $\mathcal{Y}$ in (3-5), with two frontal sides corresponding to spatial dimensions. Arrows represent linear mappings and illustrate how scalar values on the right are computed. Yellow tube, red box, and blue tube correspond to 1×1, $D \times D$, and 1×1 convolution in (3), (4), and (5) respectively.

- Complexity Analysis

$$M = \frac{D^2 ST}{SR_3 + D^2 R_3 R_4 + TR_4} \text{ and } E = \frac{D^2 STH'W'}{SR_3 HW + D^2 R_3 R_4 H'W' + TR_4 H'W'}$$

M represents the compression ratio, E represents the speed-up ratio

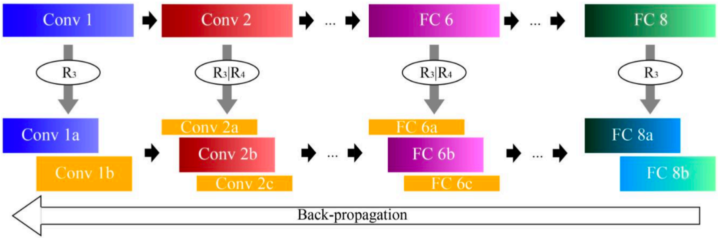
Rank Selection With Global Analytic VBMF

- Motivation: The rank-$(R_3; R_4)$ control the trade-off between performance (memory, speed, energy) improvement and accuracy loss.
- Method: variational Bayesian atrix factorization[6]
- Advantages: VBMF can automatically find noise variance, rank and even provide theoretical condition for perfect rank recovery

[6] Shinichi Nakajima et al. (2013). "Global analytic solution of fully-observed variational Bayesian matrix factorization". In: *Journal of Machine Learning Research* 14.Jan, pp. 1–37.

Tucker Decomposition

- One-shot whole network compression scheme



Three parts: (1) rank selection with VBMF; (2) Tucker decomposition on kernel tensor; (3) fine-tuning of entire network.

- Notes:Tucker-2 decomposition is applied from the second convolutional layer to the first fully connected layers, and Tucker-1 decomposition to the other layers.

CP-Decomposition[7]

Published as a conference paper at ICLR 2015

SPEEDING-UP CONVOLUTIONAL NEURAL NETWORKS USING FINE-TUNED CP-DECOMPOSITION

Vadim Lebedev[1,2], Yaroslav Ganin[1], Maksim Rakhuba[1,3], Ivan Oseledets[1,4], and Victor Lempitsky[1]

[1]Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia
[2]Yandex, Moscow, Russia
[3]Moscow Institute of Physics and Technology, Moscow Region, Russia
[4]Institute of Numerical Mathematics RAS, Moscow, Russia

- Take a convolutional layer and decompose its kernel using CP-decomposition
- Fine-tune the entire network using backpropagation.

[7]Vadim Lebedev et al. (2015). "Speeding-up convolutional neural networks using fine-tuned CP-decomposition". In: *Proc. ICLR*.

Advantages

- Ease of the decomposition implementation
- Ease of the CNN implementation
- Ease of fine-tuning
- Efficiency

CP-Decomposition

Principle

- A low-rank decomposition of a matrix A of size $n \times m$ with rank R is given by

$$A(i,j) = \sum_{r=1}^{R} A_1(i,r)A_2(j,r), \quad i = \overline{1,n}, \quad j = \overline{1,m}$$

- For a d-dimensional array A of size $n_1 \times \cdots \times n_d$ a CP-decomposition has the following form

$$A\left(i_1, \ldots, i_d\right) = \sum_{r=1}^{R} A_1\left(i_1, r\right) \ldots A_d\left(i_d, r\right)$$

where the minimal possible R is called canonical rank.

- Profit we need to store only $\left(n_1 + \cdots + n_d\right) R$ elements instead of the whole tensor with $n_1 \ldots n_d$ elements.

Notes:

- There is no finite algorithm for determining canonical rank of a tensor when $d > 2$
- Non-linear least squares (NLS) method minimizes the L2-norm of the approximation residual (for a user-defined fixed R) using Gauss-Newton optimization.

Kernel Tensor Approximation

- Convolution Calculation

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)U(i,j,s)$$

- $K(\cdot,\cdot,\cdot,\cdot)$ is a 4D kernel tensor of size $d \times d \times S \times T$ d is the spatial dimensions, S is input channels, T is output channels, while δ denotes "half-width" $(d-1)/2$
- Kernel Approximation

$$K(i,j,s,t) = \sum_{r=1}^{R} K^x(i-x+\delta, r)K^y(j-y+\delta, r)K^s(s,r)K^t(t,r)$$

- where $K^x(\cdot,\cdot), K^y(\cdot,\cdot), K^s(\cdot,\cdot), K^t(\cdot,\cdot)$ are the four components of the composition representing 2D tensors (matrices) of sizes $d \times R, d \times R, S \times R$, and $T \times R$ respectively.

CP-Decomposition

Convolution Approximation

- Substitue the Kernel Approx to Conv

$$V(x,y,t) = \sum_{r=1}^{R} K^t(t,r) \left(\sum_{i=x-\delta}^{x+\delta} K^x(i-x+\delta,r) \left(\sum_{j=y-\delta}^{y+\delta} K^y(j-y+\delta,r) \left(\sum_{s=1}^{S} K^s(s,r)U(i,j,s) \right) \right) \right)$$

- Step by Step Calculation

$$U^s(i,j,r) = \sum_{s=1}^{S} K^s(s,r)U(i,j,s)$$

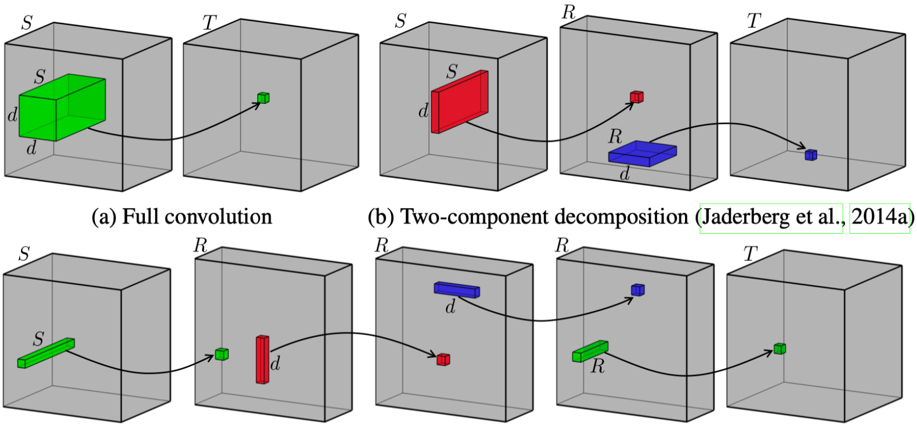
$$U^{sy}(i,y,r) = \sum_{j=y-\delta}^{y+\delta} K^y(j-y+\delta,r)U^s(i,j,r)$$

$$U^{\mathrm{syx}}(x,y,r) = \sum_{i=x-\delta}^{x+\delta} K^x(i-x+\delta,r)U^{sy}(i,y,r)$$

$$V(x,y,t) = \sum_{r=1}^{R} K^t(t,r)U^{\mathrm{syx}}(x,y,r)$$

CP-Decomposition

Complexity Comparison:



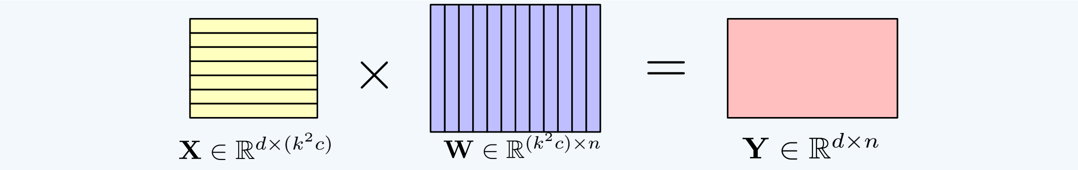
(a) Full convolution (b) Two-component decomposition (Jaderberg et al., 2014a)

(c) CP-decomposition

Overview

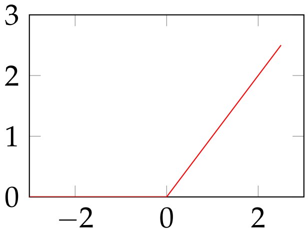
Matrix Regression Approach

Matrix Approximation or Matrix Regression?



- Matrix approximation: $\boldsymbol{W} \approx \boldsymbol{W}'$
- Matrix regression: $\boldsymbol{Y} = \boldsymbol{W} \cdot \boldsymbol{X} \approx \boldsymbol{W}' \cdot \boldsymbol{X}$

Non-linearity Approximation



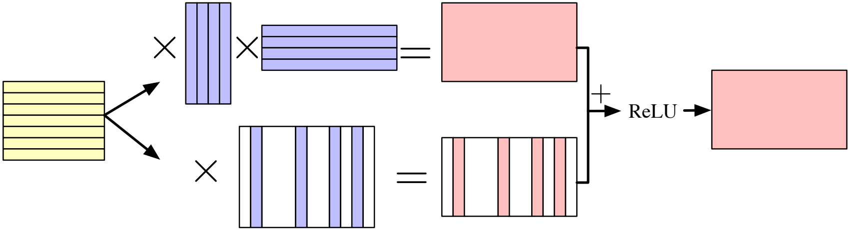
ReLU

- Activation unit: `ReLU`
- Error more sensitive to positive response;
- Enlarge the solution space.

$$\min_{\boldsymbol{W}} \sum_{i=1}^{N} \|\boldsymbol{W}\boldsymbol{X}_i - \boldsymbol{Y}_i\|_F \rightarrow \min_{\boldsymbol{W}} \sum_{i=1}^{N} \|r(\boldsymbol{W}\boldsymbol{X}_i) - \boldsymbol{Y}_i\|_F$$

- $\boldsymbol{X}$: input feature map
- $\boldsymbol{Y}$: output feature map

Proposed Unified Structure[8]



- Simultaneous low-rank approximation and network sparsification;
- Non-linearity is taken into account.
- Acceleration is achieved with structured sparsity.

[8] Yuzhe Ma et al. (2019). "A Unified Approximation Framework for Non-Linear Deep Neural Networks". In: *Proc. ICTAI*.

Formulation

Given a pre-trained network, the goal is to minimize the reconstruction error of the response in each layer after activation, using sparse component and low-rank component.

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{B}} \; & \sum_{i=1}^{N} \left\| \boldsymbol{Y}_i - r((\boldsymbol{A}+\boldsymbol{B})\boldsymbol{X}_i) \right\|_F, \\ \text{s.t.} \; & \|\boldsymbol{A}\|_0 \le S, \\ & \operatorname{rank}(\boldsymbol{B}) \le L. \end{aligned}$$

- $\boldsymbol{X}$: input feature map
- $\boldsymbol{Y}$: output feature map

Not easy to solve: l_0 minimization and rank minimization are NP-hard.

Relaxation

$$\min_{\boldsymbol{A},\boldsymbol{B}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r((\boldsymbol{A}+\boldsymbol{B})\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_*$$

- The l_0 constraint is relaxed by $l_{2,1}$ norm such that the zero elements in $\boldsymbol{A}$ appear column-wise;
- The rank constraint on $\boldsymbol{B}$ is relaxed by nuclear norm of $\boldsymbol{B}$, which is the sum of the singular values;
- Apply alternating direction method of multipliers (ADMM) to solve it;

Reference: https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/admm.pdf

Reformulating the problem with an auxiliary variable $\boldsymbol{M}$,

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{M}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_* ,$$
$$\text{s.t. } \boldsymbol{A} + \boldsymbol{B} = \boldsymbol{M}.$$

Then the augmented Lagrangian function is

$$\begin{aligned} &L_t(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{M}, \boldsymbol{\Lambda}) \\ =& \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_* + \langle \boldsymbol{\Lambda}, \boldsymbol{A} + \boldsymbol{B} - \boldsymbol{M} \rangle + \frac{t}{2} \|\boldsymbol{A} + \boldsymbol{B} - \boldsymbol{M}\|_F^2 , \end{aligned}$$

Iteratively solve with following rules. All of them can be solved efficiently.

$$
\begin{cases}
\boldsymbol{A}_{k+1} = \underset{\boldsymbol{A}}{\operatorname{argmin}}\ \lambda_1 \|\boldsymbol{A}\|_{2,1} + \frac{t}{2} \left\| \boldsymbol{A} + \boldsymbol{B}_k - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t} \right\|_F^2, \\
\boldsymbol{B}_{k+1} = \underset{\boldsymbol{B}}{\operatorname{argmin}}\ \lambda_2 \|\boldsymbol{B}\|_* + \frac{t}{2} \left\| \boldsymbol{B} + \boldsymbol{A}_{k+1} - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t} \right\|_F^2, \\
\boldsymbol{M}_{k+1} = \underset{\boldsymbol{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \langle \boldsymbol{\Lambda}_k, \boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M} \rangle + \frac{t}{2} \|\boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M}\|_F^2, \\
\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + t(\boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M}_{k+1}).
\end{cases}
$$

Solving $l_{2,1}$-norm

$$\min_{\boldsymbol{A}} \lambda_1 \|\boldsymbol{A}\|_{2,1} + \frac{t}{2}\left\|\boldsymbol{A} + \boldsymbol{B}_k - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t}\right\|_F^2$$

Closed Form Update Rule[9]

$$\boldsymbol{A}_{k+1} = \mathrm{prox}_{\frac{\lambda_1}{t}\|\cdot\|_{2,1}}(\boldsymbol{M}_k - \boldsymbol{B}_k - \frac{\boldsymbol{\Lambda}_k}{t}),$$

$$\boldsymbol{C} = \boldsymbol{M}_k - \boldsymbol{B}_k - \frac{\boldsymbol{\Lambda}_k}{t},$$

$$[\boldsymbol{A}_{k+1}]_{:,i} = \begin{cases} \dfrac{\|[\boldsymbol{C}]_{:,i}\|_2 - \frac{\lambda_1}{t}}{\|[\boldsymbol{C}]_{:,i}\|_2}[\boldsymbol{C}]_{:,i}, & \text{if } \|[\boldsymbol{C}]_{:,i}\|_2 > \dfrac{\lambda_1}{t}; \\ 0, & \text{otherwise.} \end{cases}$$

[9] G. Liu et al., *"Robust recovery of subspace structures by low-rank representation"*, *TPAMI, 2013*.

Solving nuclear norm

$$\min_{\boldsymbol{B}} \lambda_2 \|\boldsymbol{B}\|_* + \frac{t}{2} \left\| \boldsymbol{B} + \boldsymbol{A}_{k+1} - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t} \right\|_F^2$$

Closed Form Update Rule[10]

$$\boldsymbol{B}_{k+1} = \mathrm{prox}_{\frac{\lambda_2}{t}\|\cdot\|_*}(\boldsymbol{M}_k - \boldsymbol{A}_{k+1} - \frac{\boldsymbol{\Lambda}_k}{t}),$$

$$\boldsymbol{D} = \boldsymbol{M}_k - \boldsymbol{A}_{k+1} - \frac{\boldsymbol{\Lambda}_k}{t},$$

$$\boldsymbol{B}_{k+1} = \boldsymbol{U}\mathcal{D}_{\frac{\lambda_2}{t}}(\boldsymbol{\Sigma})\boldsymbol{V}, \quad \text{where } \mathcal{D}_{\frac{\lambda_2}{t}}(\boldsymbol{\Sigma}) = \mathrm{diag}(\{(\sigma_i - \frac{\lambda_2}{t})_+\}).$$

[10] J-F. Cai et al., *"A singular value thresholding algorithm for matrix completion"*, *SIOPT, 2010.*

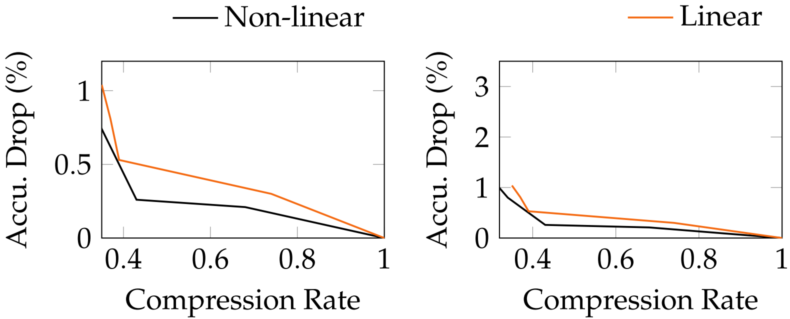
Comparison on *CIFAR-10* dataset

Model	Method	Accuracy ↓	CR	Speed-up
VGG-16	Original	0.00%	1.00	1.00
	ICLR'17[11]	**0.06%**	2.70	1.80
	Ours	0.40%	**4.44**	**2.20**
NIN	Original	0.00%	1.00	1.00
	ICLR'16[12]	1.43%	1.54	1.50
	IJCAI'18[13]	1.43%	1.45	-
	Ours	**0.41%**	**2.77**	**1.70**

[11] Hao Li et al. (2017). "Pruning filters for efficient convnets". In: *Proc. ICLR*.

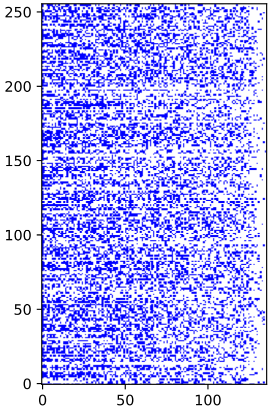
[12] Cheng Tai et al. (2016). "Convolutional neural networks with low-rank regularization". In: *Proc. ICLR*.

[13] Shiva Prasad Kasiviswanathan, Nina Narodytska, and Hongxia Jin (2018). "Network Approximation using Tensor Sketching". In: *Proc. IJCAI*, pp. 2319–2325.

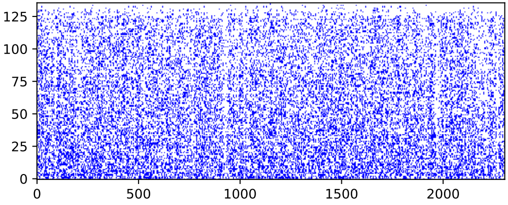


(a) (b)

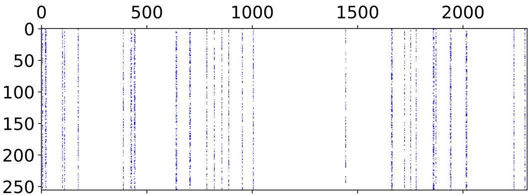
Comparison of reconstructing linear response and non-linear response: (a) layer `conv2-1`; (b) layer `conv3-1`.



(a)



(b)



(c)

Approximated filters of `conv3-1`. Blue dots have non-zero values. Low-rank filter $\boldsymbol{B}$ with rank 136 is decomposed into $\boldsymbol{UV}$, both of which have rank 136. (a) Matrix $\boldsymbol{U}$; (b) Matrix $\boldsymbol{V}$. (c) Column-wise sparse filter $\boldsymbol{A}$.

Comparison on *ImageNet* dataset

Model	Method	Top-5 Accu.↓	CR	Speed-up
AlexNet	Original	0.00%	1.00	1.00
	ICLR'16[14]	**0.37%**	5.00	**1.82**
	ICLR'16[15]	1.70%	5.46	1.81
	CVPR'18[16]	1.43%	1.50	-
	Ours	1.27%	**5.56**	1.10
GoogleNet	Original	0.00%	1.00	1.00
	ICLR'16[10]	0.42%	2.84	1.20
	ICLR'16[11]	0.24%	1.28	1.23
	CVPR'18[12]	0.21%	1.50	-
	Ours	**0.00%**	**2.87**	**1.35**

[14] Cheng Tai et al. (2016). "Convolutional neural networks with low-rank regularization". In: *Proc. ICLR*.

[15] Yong-Deok Kim et al. (2016). "Compression of deep convolutional neural networks for fast and low power mobile applications". In: *Proc. ICLR*.

[16] Ruichi Yu et al. (2018). "NISP: Pruning networks using neuron importance score propagation". In: *Proc. CVPR*, pp. 9194–9203.