Rank

Stephen Boyd and Sanjay Lall
Revision by Amirhossein Afsharrad

EE263
Stanford University

Rank of a matrix

range $(A) \subseteq \mathbb{R}^m$

$\neq$

range $(A^T) \subseteq \mathbb{R}^n$

we define the *rank* of $A \in \mathbb{R}^{m \times n}$ as

$$\textbf{rank}(A) = \textbf{dim range}(A)$$

$A \in \mathbb{R}^{m \times n}$

$A = [a_1 \ \dots \ a_n]$

$\text{rank}(A) = r < \min(m, n)$

100 50

$\text{rank}(A) = r \qquad r = 30$

(nontrivial) **facts:**

- $\textbf{rank}(A) = \textbf{rank}(A^T)$
- $\textbf{rank}(A)$ is maximum number of independent columns (or rows) of A
  hence $\textbf{rank}(A) \leq \textbf{min}(m, n)$

* remove any column a such that the rank will not change

keep doing that until you have a basis

Rank of a matrix

- **rank**(A) is maximum number of independent columns of A
  - to see this, notice that if the columns of A are independent, then the number of columns r is the rank, since the columns are a basis for the range
  - and if not, then there must be one column in the span of the others, so remove it, and repeat if necessary
  - all other independent sets of columns must have no more than r elements.
- proof of **rank**$(A) =$ **rank**(A^{T}) uses QR, to come

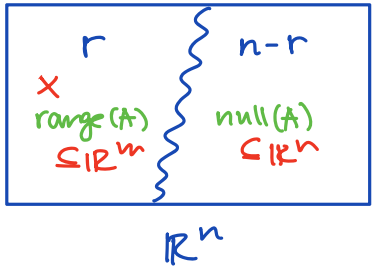
Conservation of dimension

$A \in \mathbb{R}^{m \times n}$

• WRONG intuition

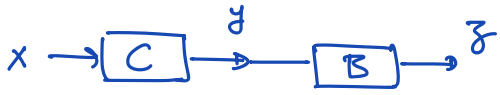
input dimension

$$\underbrace{\textbf{dim range}(A)}_{\text{rank}(A)} + \textbf{dim null}(A) = n$$



- **rank**(A) is dimension of set 'hit' by the mapping $y = Ax$
- **dim null**(A) is dimension of set of x 'crushed' to zero by $y = Ax$
- 'conservation of dimension': each dimension of input is either crushed to zero or ends up in output
- roughly speaking:
  - n is number of degrees of freedom in input x
  - **dim null**(A) is number of degrees of freedom lost in the mapping from x to $y = Ax$
  - **rank**(A) is number of degrees of freedom in output y
- proof using QR

Coding interpretation of rank



$A \in \mathbb{R}^{20\times 10}$

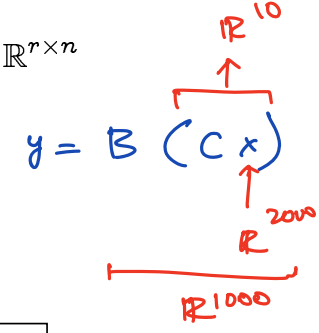
$B \in \mathbb{R}^{10\times 20}$

$BA = I$

$$\textbf{rank}(BC) \leq \min\{\textbf{rank}(B), \textbf{rank}(C)\}$$

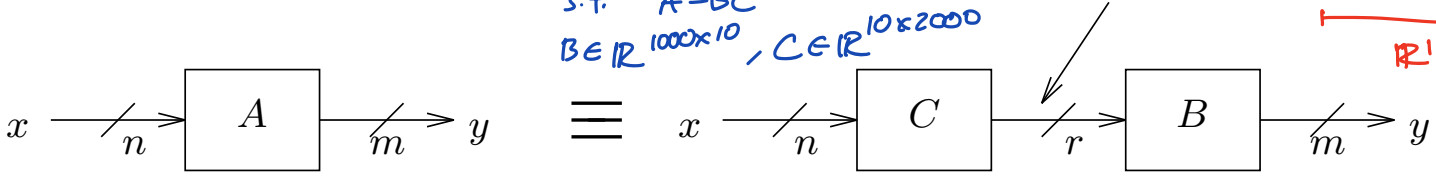
AB is never the identity $\quad AB = I_{20\times 20} \rightarrow \text{rank}(AB) = 20 < \text{rank}(A) = 10$ contradiction

- hence if $A = BC$ with $B \in \mathbb{R}^{m\times r}$, $C \in \mathbb{R}^{r\times n}$, then $\textbf{rank}(A) \leq r$
- converse: if $\textbf{rank}(A) = r$ then $A \in \mathbb{R}^{m\times n}$ factors as $A = BC$ with $B \in \mathbb{R}^{m\times r}$, $C \in \mathbb{R}^{r\times n}$
- $\textbf{rank}(A) = r$ is minimum size vector needed to faithfully reconstruct y from x



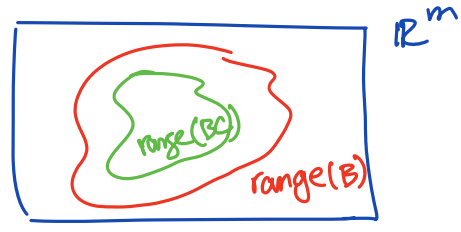
$A \in \mathbb{R}^{1000\times 2000}$, $\text{rank}(A) = 10$ $\rightarrow$ B, C exist s.t. $A = BC$

$B \in \mathbb{R}^{1000\times 10}$, $C \in \mathbb{R}^{10\times 2000}$



Coding interpretation of rank

$y \in \text{range}(BC) \rightarrow y = BCx \rightarrow y = Bz \rightarrow y \in \text{range}(B)$, where $z = Cx$



- $\textbf{rank}(BC) \leq \textbf{rank}(B)$ because $\textbf{range}(BC) \subset \textbf{range}(B)$
- transpose implies $\textbf{rank}(BC) \leq \textbf{rank}(C)$
- factorization converse comes from QR

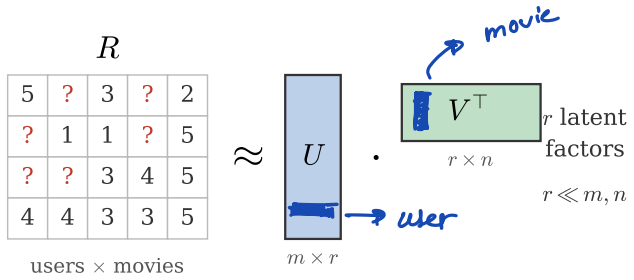
Low-rank factorization: latent factors

if $A = BC$ with inner dimension r, then r numbers (the *latent factors*) mediate every input-output pair

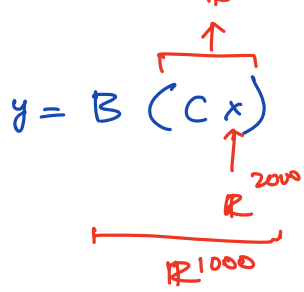
recommender systems: a ratings matrix factors as $R \approx UV^\top$

- a row of U is a user's taste, a row of V a movie's genre mix
- r hidden factors explain all ratings, and fill in the missing ones

the same idea represents users, items, and words as short *embedding* vectors in $\mathbb{R}^r$; store $(m+n)r$ numbers instead of mn



Application: fast matrix-vector multiplication



$A \in \mathbb{R}^{1000 \times 2000}$, $\text{rank}(A) = 10$ → B, C exist s.t. $A = BC$, $B \in \mathbb{R}^{1000 \times 10}$, $C \in \mathbb{R}^{10 \times 2000}$

- need to compute matrix-vector product $y = Ax$, $A \in \mathbb{R}^{m \times n}$
- A has known factorization $A = BC$, $B \in \mathbb{R}^{m \times r}$
- computing $y = Ax$ directly: mn operations → 2×10^6 3000×10
- computing $y = Ax$ as $y = B(Cx)$ (compute $z = Cx$ first, then $y = Bz$): $rn + mr = (m + n)r$ operations
- savings can be considerable if $r \ll \mathbf{min}\{m, n\}$

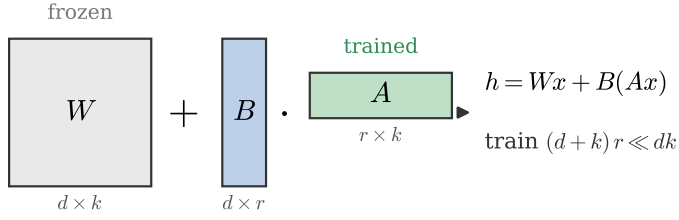
$$\frac{3 \times 10^4}{2 \times 10^6} = 1.5 \times 10^{-2}$$

Application: low-rank adaptation (LoRA)

a pretrained weight matrix $W \in \mathbb{R}^{d \times k}$ is huge, and fine-tuning all of it is expensive

LoRA freezes W and learns a *low-rank* update $\Delta W = BA$, with $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, $r \ll d, k$

- forward pass $h = Wx + B(Ax)$ (the same fast-product trick)
- trains $(d + k)r$ parameters instead of dk, often well under 1%
- many small adapters can share one frozen base model



Low-rank approximation: compression

real matrices (images, data tables) are often close to a *low-rank* matrix: a rank-r approximation needs only $r(m+n)$ numbers instead of mn

rank 5

2% of the data

rank 20

7% of the data

rank 50

18% of the data

full (rank 512)

100%

rank measures *redundancy*: a few independent patterns already capture most of the data (the best rank-r approximation comes from the SVD, later)

Full rank matrices

for $A \in \mathbb{R}^{m \times n}$ we always have $\mathbf{rank}(A) \leq \mathbf{min}(m, n)$

we say A is *full rank* if $\mathbf{rank}(A) = \mathbf{min}(m, n)$

- for **square** matrices, full rank means nonsingular
- for **skinny** matrices ($m \geq n$), full rank means columns are independent
- for **fat** matrices ($m \leq n$), full rank means rows are independent