Low-Rank Adaptation

CSE 240D

Spring 2026

Hanyang Xu

What is a rank of a matrix?

The rank is how many of the rows are "unique": not made of other rows. (Same for columns.)

Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}$$

The second row is just 3 times the first row. Just a useless copycat. Doesn't count.

So even though there are 2 rows, the rank is only 1.

SVD Computes the Exact Mathematical Rank

Theorem 1 (Singular Value Decomposition (SVD)). *Consider* $A \in \mathbb{R}^{d \times n}$ *and let* $r = \min(d, n)$. *A can always be written as the product of three matrices,* $A = U\Sigma V^T$, *where:*

- $U \in \mathbb{R}^{d \times r}$ *is a matrix with orthonormal columns,*
- $\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$ *is a non-negative diagonal matrix with entries* $\sigma_1 \geq \ldots \geq \sigma_r \geq 0$,
- $V \in \mathbb{R}^{n \times r}$ *is a matrix with orthonormal columns.*

U's columns are called the "left singular vectors" of A, V's columns are its "right singular vectors", and $\sigma_1, \ldots, \sigma_r$ *are its "singular values". When the SVD is computed after mean centering A's columns or rows, the singular vectors are sometimes call "principal components".*

Low-Rank Structure

In particular, we are interested in datasets where most of our vectors $a_1, \ldots, a_n$ can be well approximated as a linear combination of a small ground set of vectors in $\mathbb{R}^d$, $\{b_1, \ldots, b_k\}$. I.e. for some set of k coefficients $\{C_{1i}, C_{2i}, \ldots, C_{ki}\}$ we approximate a_i by:

$$a_i \approx \sum_{j=1}^{k} C_{ji} b_j.$$

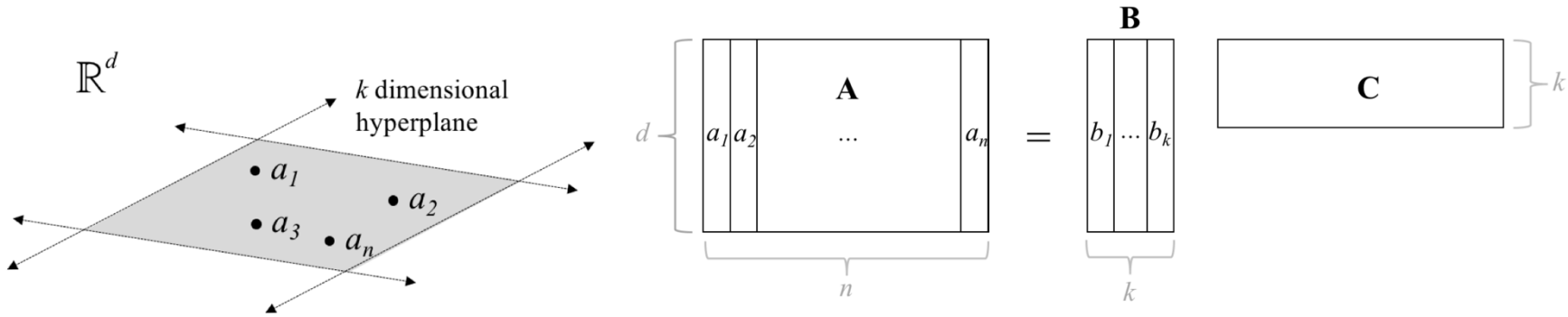


Figure 1: If $a_1, \ldots, a_n$ lie on a low dimensional hyperplane, then $A = [a_1, \ldots, a_n]$ is exactly low rank, so it can be written as the product of two matrices, $B \in \mathbb{R}^{d \times k}$ and $C \in \mathbb{R}^{k \times n}$

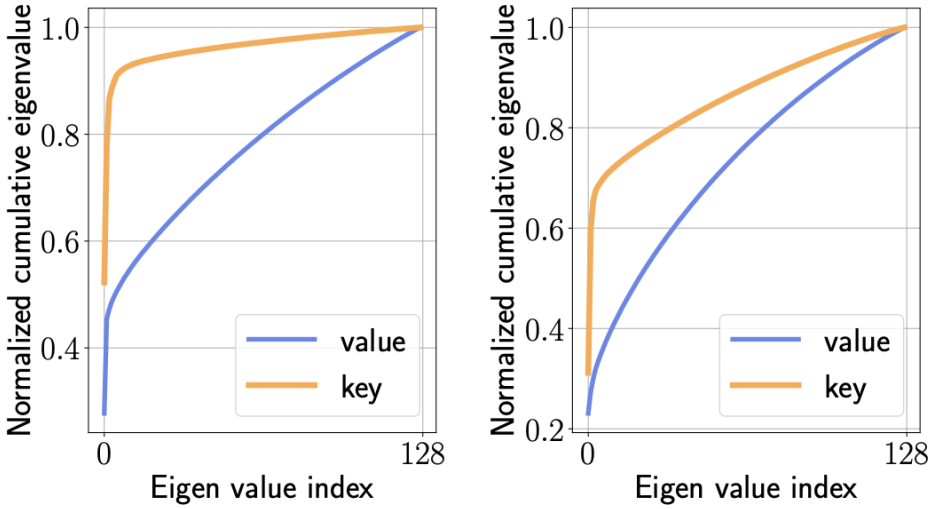
Low-Rank Structure in K and V Matrix

1. ***Heads are redundant***: Same observation in GQA

2. **The key** only needs to match **the effective rank of the query**, not more (dot product will zero out the additional dimensions)

3. **Empirical Confirmed:** Running SVD on models, the singular values cliff off fast — most of the energy is in the top ~512 directions out of ~16,000.

Eigen Attention: Attention in Low-Rank Space for KV Cache Compression – ACL 2024



a Layer #15 Head #16 b Layer #25 Head #16

Vanilla Low-Rank Adaptation on a Trained Model

$$W = \begin{bmatrix} 4 & 3 & 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 5 & 4 & 2 & 1 & 0 & 0 & 0 \\ 2 & 4 & 6 & 3 & 2 & 1 & 0 & 0 \\ 1 & 2 & 3 & 4 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 2 & 3 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

SVD on W

$$U = \begin{bmatrix} -0.34 & 0.50 & 0.43 & -0.47 & -0.02 & -0.46 & -0.08 & 0.06 \\ -0.53 & 0.36 & 0.08 & 0.17 & -0.26 & 0.63 & 0.31 & 0.00 \\ -0.61 & -0.07 & -0.20 & 0.56 & 0.25 & -0.38 & -0.24 & -0.07 \\ -0.40 & -0.33 & -0.30 & -0.63 & 0.35 & 0.29 & -0.16 & -0.13 \\ -0.26 & -0.50 & -0.05 & -0.13 & -0.71 & -0.21 & 0.05 & 0.34 \\ -0.11 & -0.41 & 0.46 & 0.05 & 0.37 & -0.11 & 0.68 & 0.00 \\ -0.03 & -0.25 & 0.47 & 0.06 & -0.26 & 0.10 & -0.32 & -0.73 \\ -0.01 & -0.16 & 0.49 & 0.13 & 0.20 & 0.32 & -0.50 & 0.57 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 13.55 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.21 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.90 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.84 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.54 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.28 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.26 \end{bmatrix}$$

$$V^\top = \begin{bmatrix} -0.34 & -0.53 & -0.61 & -0.40 & -0.26 & -0.11 & -0.03 & -0.01 \\ 0.50 & 0.36 & -0.07 & -0.33 & -0.50 & -0.41 & -0.25 & -0.16 \\ 0.43 & 0.08 & -0.20 & -0.30 & -0.05 & 0.46 & 0.47 & 0.49 \\ -0.47 & 0.17 & 0.56 & -0.63 & -0.13 & 0.05 & 0.06 & 0.13 \\ -0.02 & -0.26 & 0.25 & 0.35 & -0.71 & 0.37 & -0.26 & 0.20 \\ -0.46 & 0.63 & -0.38 & 0.29 & -0.21 & -0.11 & 0.10 & 0.32 \\ -0.08 & 0.31 & -0.24 & -0.16 & 0.05 & 0.68 & -0.32 & -0.50 \\ -0.06 & 0.00 & 0.07 & 0.13 & -0.34 & 0.00 & 0.73 & -0.57 \end{bmatrix}$$

Vanilla Low-Rank Adaptation on a Trained Model

$$U = \begin{bmatrix} -0.34 & 0.50 & 0.43 & -0.47 & -0.02 & -0.46 & -0.08 & 0.06 \\ -0.53 & 0.36 & 0.08 & 0.17 & -0.26 & 0.63 & 0.31 & 0.00 \\ -0.61 & -0.07 & -0.20 & 0.56 & 0.25 & -0.38 & -0.24 & -0.07 \\ -0.40 & -0.33 & -0.30 & -0.63 & 0.35 & 0.29 & -0.16 & -0.13 \\ -0.26 & -0.50 & -0.05 & -0.13 & -0.71 & -0.21 & 0.05 & 0.34 \\ -0.11 & -0.41 & 0.46 & 0.05 & 0.37 & -0.11 & 0.68 & 0.00 \\ -0.03 & -0.25 & 0.47 & 0.06 & -0.26 & 0.10 & -0.32 & -0.73 \\ -0.01 & -0.16 & 0.49 & 0.13 & 0.20 & 0.32 & -0.50 & 0.57 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 13.55 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.21 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.90 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.84 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.54 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.28 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.26 \end{bmatrix}$$

$$V^\top = \begin{bmatrix} -0.34 & -0.53 & -0.61 & -0.40 & -0.26 & -0.11 & -0.03 & -0.01 \\ 0.50 & 0.36 & -0.07 & -0.33 & -0.50 & -0.41 & -0.25 & -0.16 \\ 0.43 & 0.08 & -0.20 & -0.30 & -0.05 & 0.46 & 0.47 & 0.49 \\ -0.47 & 0.17 & 0.56 & -0.63 & -0.13 & 0.05 & 0.06 & 0.13 \\ -0.02 & -0.26 & 0.25 & 0.35 & -0.71 & 0.37 & -0.26 & 0.20 \\ -0.46 & 0.63 & -0.38 & 0.29 & -0.21 & -0.11 & 0.10 & 0.32 \\ -0.08 & 0.31 & -0.24 & -0.16 & 0.05 & 0.68 & -0.32 & -0.50 \\ -0.06 & 0.00 & 0.07 & 0.13 & -0.34 & 0.00 & 0.73 & -0.57 \end{bmatrix}$$

Rank = 3 Adaptation

$$U_3 = \begin{bmatrix} -0.34 & 0.50 & 0.43 \\ -0.53 & 0.36 & 0.08 \\ -0.61 & -0.07 & -0.20 \\ -0.40 & -0.33 & -0.30 \\ -0.26 & -0.50 & -0.05 \\ -0.11 & -0.41 & 0.46 \\ -0.03 & -0.25 & 0.47 \\ -0.01 & -0.16 & 0.49 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 13.55 & 0 & 0 \\ 0 & 5.21 & 0 \\ 0 & 0 & 2.90 \end{bmatrix}$$

$$V_3^\top = \begin{bmatrix} -0.34 & -0.53 & -0.61 & -0.40 & -0.26 & -0.11 & -0.03 & -0.01 \\ 0.50 & 0.36 & -0.07 & -0.33 & -0.50 & -0.41 & -0.25 & -0.16 \\ 0.43 & 0.08 & -0.20 & -0.30 & -0.05 & 0.46 & 0.47 & 0.49 \end{bmatrix}$$

A=U3 @ Σ3

$$A = \begin{bmatrix} -4.55 & 2.63 & 1.25 \\ -7.13 & 1.85 & 0.22 \\ -8.25 & -0.37 & -0.59 \\ -5.45 & -1.71 & -0.86 \\ -3.45 & -2.60 & -0.15 \\ -1.53 & -2.12 & 1.33 \\ -0.40 & -1.32 & 1.37 \\ -0.15 & -0.82 & 1.42 \end{bmatrix}$$

B=V3T

$$B = V_3^\top = \begin{bmatrix} -0.34 & -0.53 & -0.61 & -0.40 & -0.26 & -0.11 & -0.03 & -0.01 \\ 0.50 & 0.36 & -0.07 & -0.33 & -0.50 & -0.41 & -0.25 & -0.16 \\ 0.43 & 0.08 & -0.20 & -0.30 & -0.05 & 0.46 & 0.47 & 0.49 \end{bmatrix}$$

LoRA: Low-Rank Adaptation Applied to Fine-Tuning

Goal: Reduce requirements in fine tuning

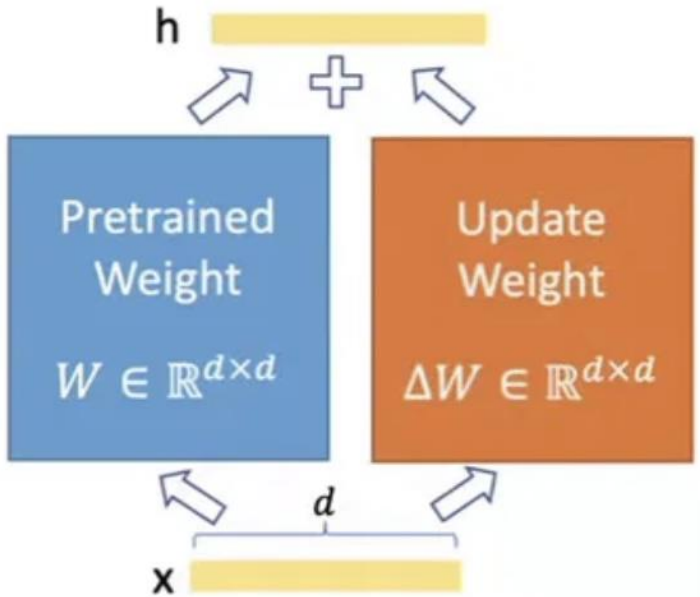
Premise: Prio work show that when adapting to a specific task, pre-trained language models can still learn efficiently despite a random projection to a smaller subspace.

Hypothesis: The updates to the weights also have a low "intrinsic rank" during adaptation.

Intuition: low rank updates make sense for *fine-tuning*:

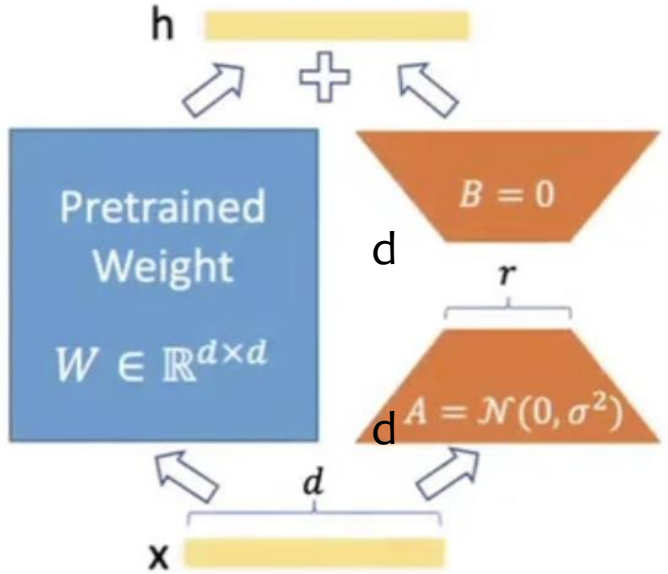
1. The task is much smaller than the model
2. Fine-tune only steers the model (which capability to emphasize)
3. One example is rank-1 by definition, a bunch of examples have similar gradient directions

LoRA: Low-Rank Adaptation Applied to ***Fine-Tuning***



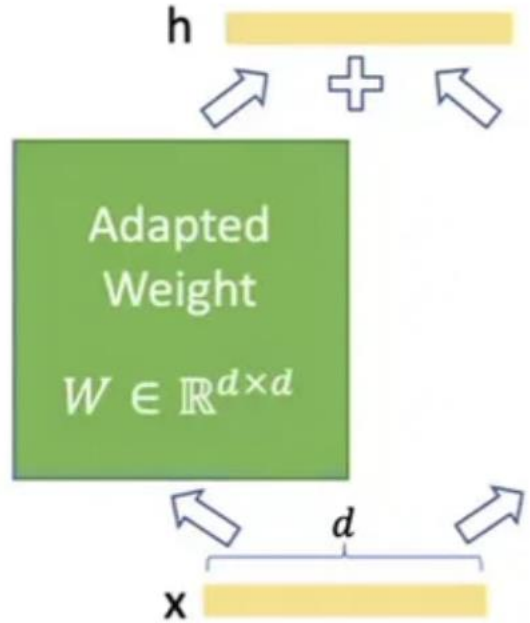
Vanilla Fine-Tuning:

Pretrained Weight Frozen
Only change ΔW



Replace ΔW with Its **Low-Rank Counterpart A and B**

Backpropagate to **learn** A and B, which combined to be smaller than Update Weights



Before Inference:

Reconstruct an **approximation** of Update Weights $\Delta W \approx A @ B$
Then Adapted Weight=Weight + ΔW

Multi-Head Latent Attention: Low-Rank Adaptation Natively Applied to Attention

The core of MLA is the low-rank joint compression for keys and values to reduce KV cache:

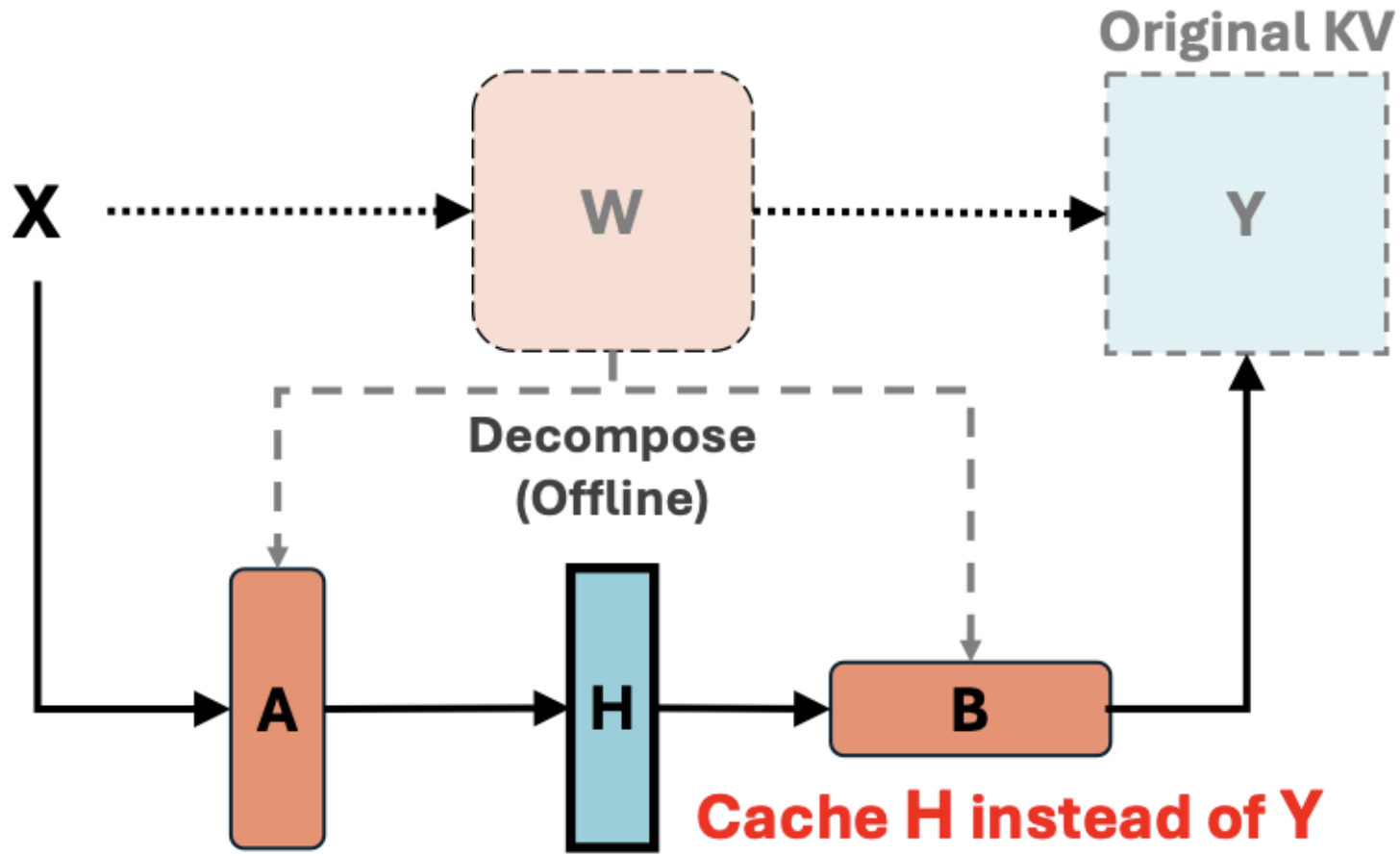
$$\mathbf{c}_t^{KV} = W^{DKV}\mathbf{h}_t, \tag{9}$$

$$\mathbf{k}_t^{C} = W^{UK}\mathbf{c}_t^{KV}, \tag{10}$$

$$\mathbf{v}_t^{C} = W^{UV}\mathbf{c}_t^{KV}, \tag{11}$$

where $\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ is the compressed latent vector for keys and values; $d_c (\ll d_h n_h)$ denotes the KV compression dimension; $W^{DKV} \in \mathbb{R}^{d_c \times d}$ is the down-projection matrix; and $W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ are the up-projection matrices for keys and values, respectively. During inference, MLA only needs to cache $\mathbf{c}_t^{KV}$, so its KV cache has only $d_c l$ elements, where l denotes the number of layers.

Palu: Low-Rank Adaptation Apply to ***KV-Cache Compression***



If I want to build a compression accelerator, where should it be located?