$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}_{m\times n}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1}}_{b}$$

CX1104: **Linear Algebra for Computing**

Chap. No : **8.4.2**

Lecture : **Eigen and Singular Values**

Topic : **SVD & Pseudoinverse**

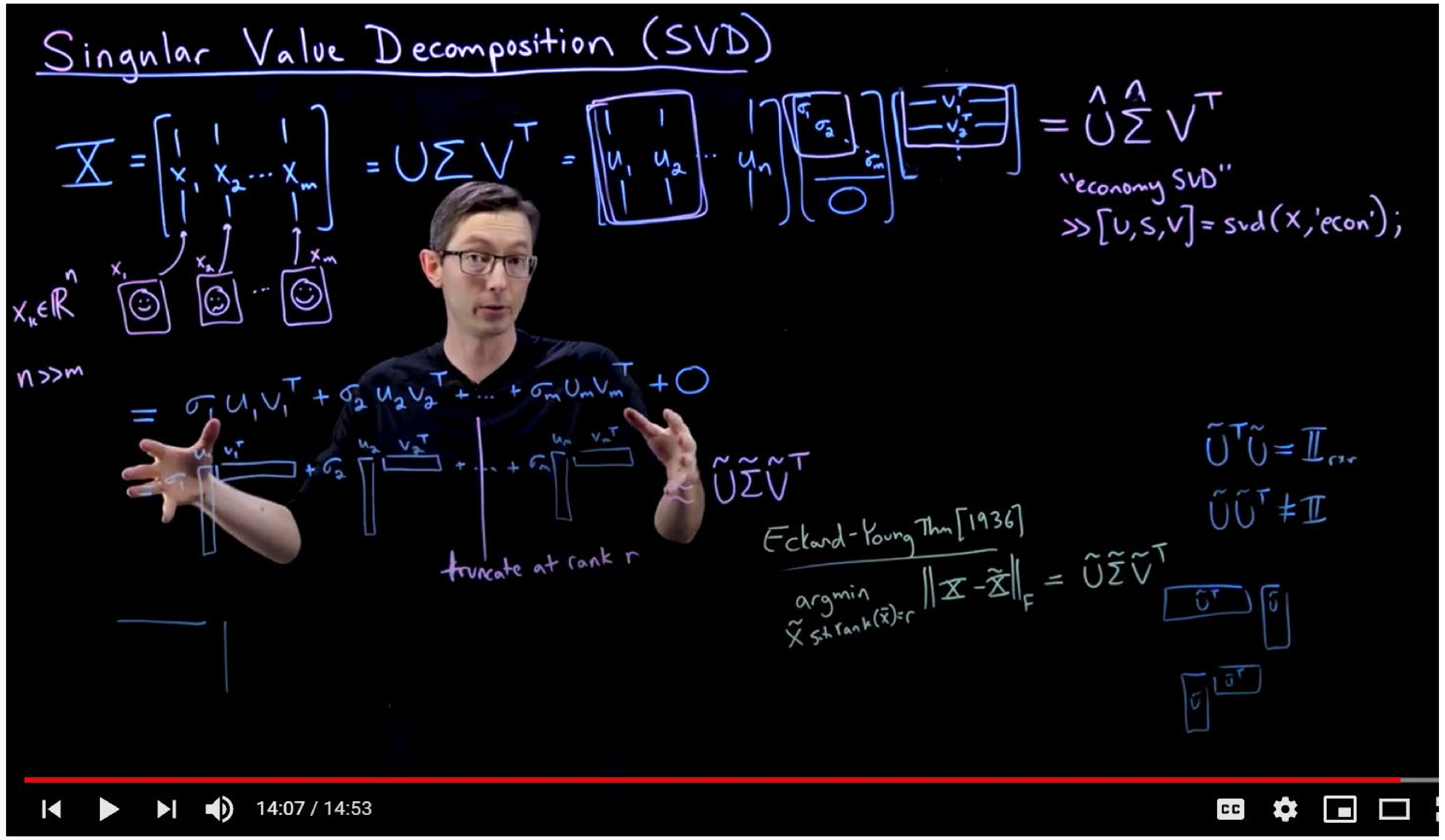
Concept : **Matrix Approximation and Image Compression**

Instructor: **A/P Chng Eng Siong**

TAs: **Zhang Su, Vishal Choudhari**

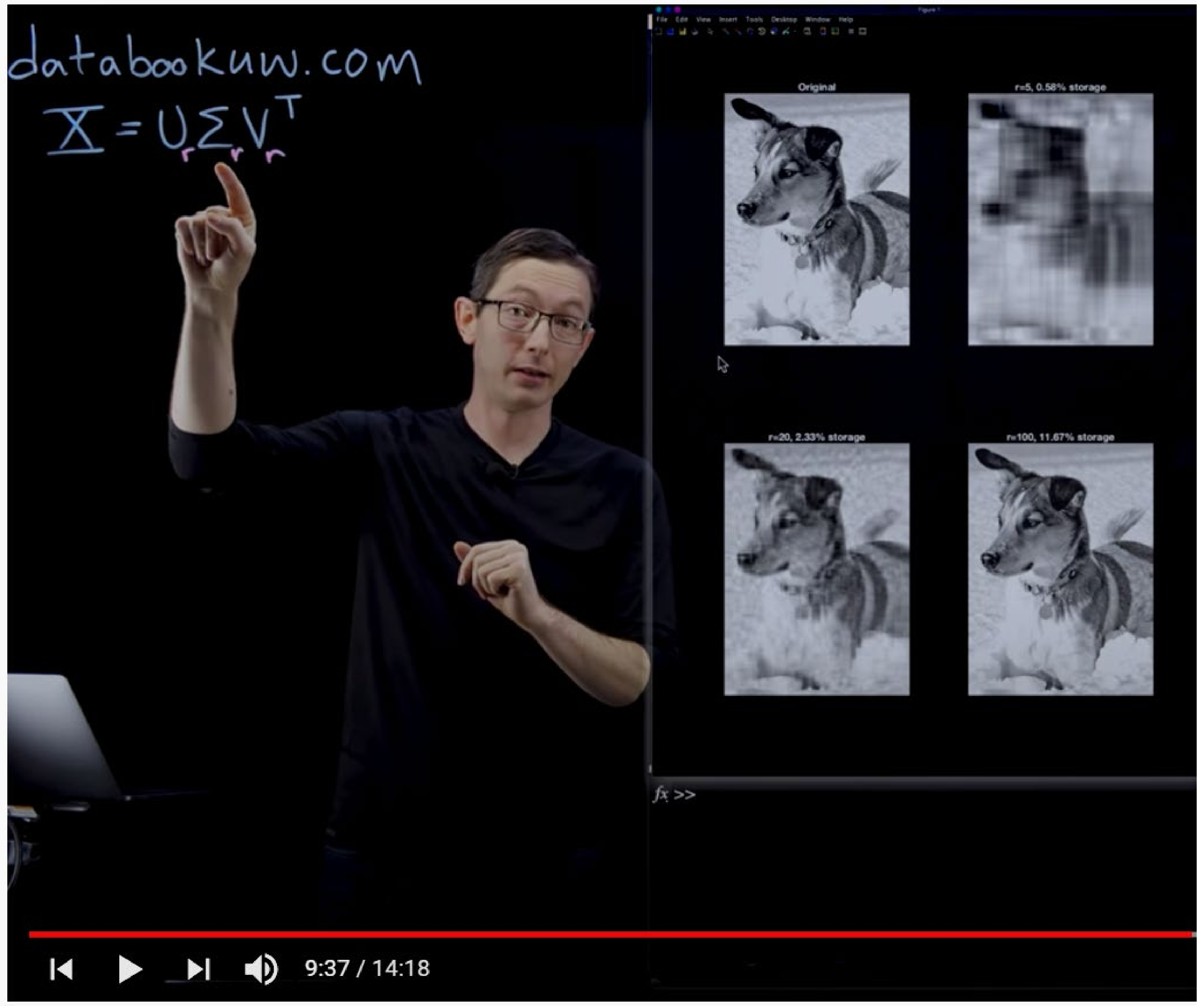
Rev: 23rd July 2020

Brunton & Kutz: Data Science and Engineering



Singular Value Decomposition (SVD): Matrix Approximation

21,424 views • Jan 20, 2020 600 3 SHARE SAVE



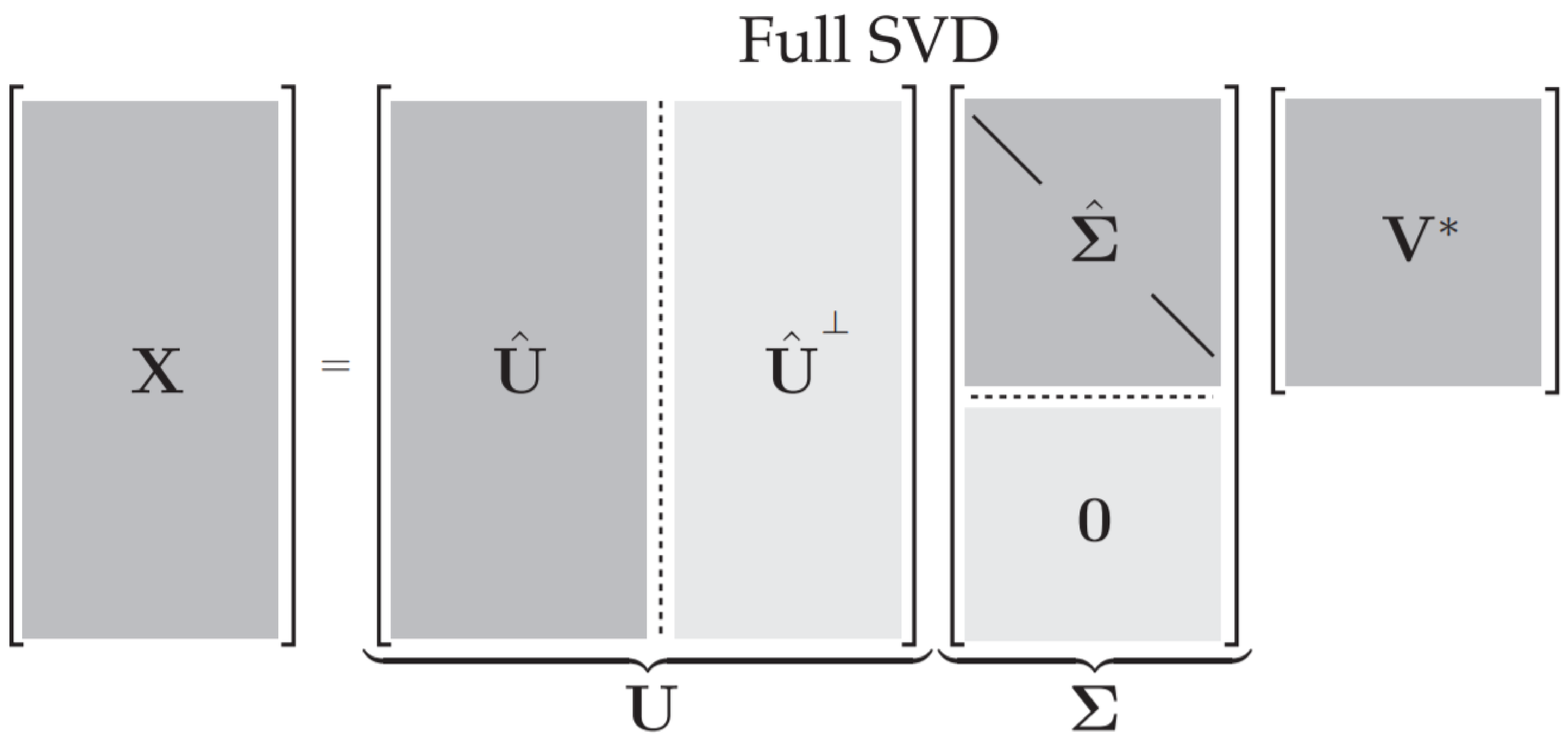
SVD: Image Compression [Matlab]

8,445 views • Feb 1, 2020

Ref: Matrix Approximation
https://www.youtube.com/watch?v=xy3QyyhiuY4

Ref: Image Compression
https://www.youtube.com/watch?v=QQ8vxj-9OfQ

Singular Value Decomposition (SVD)



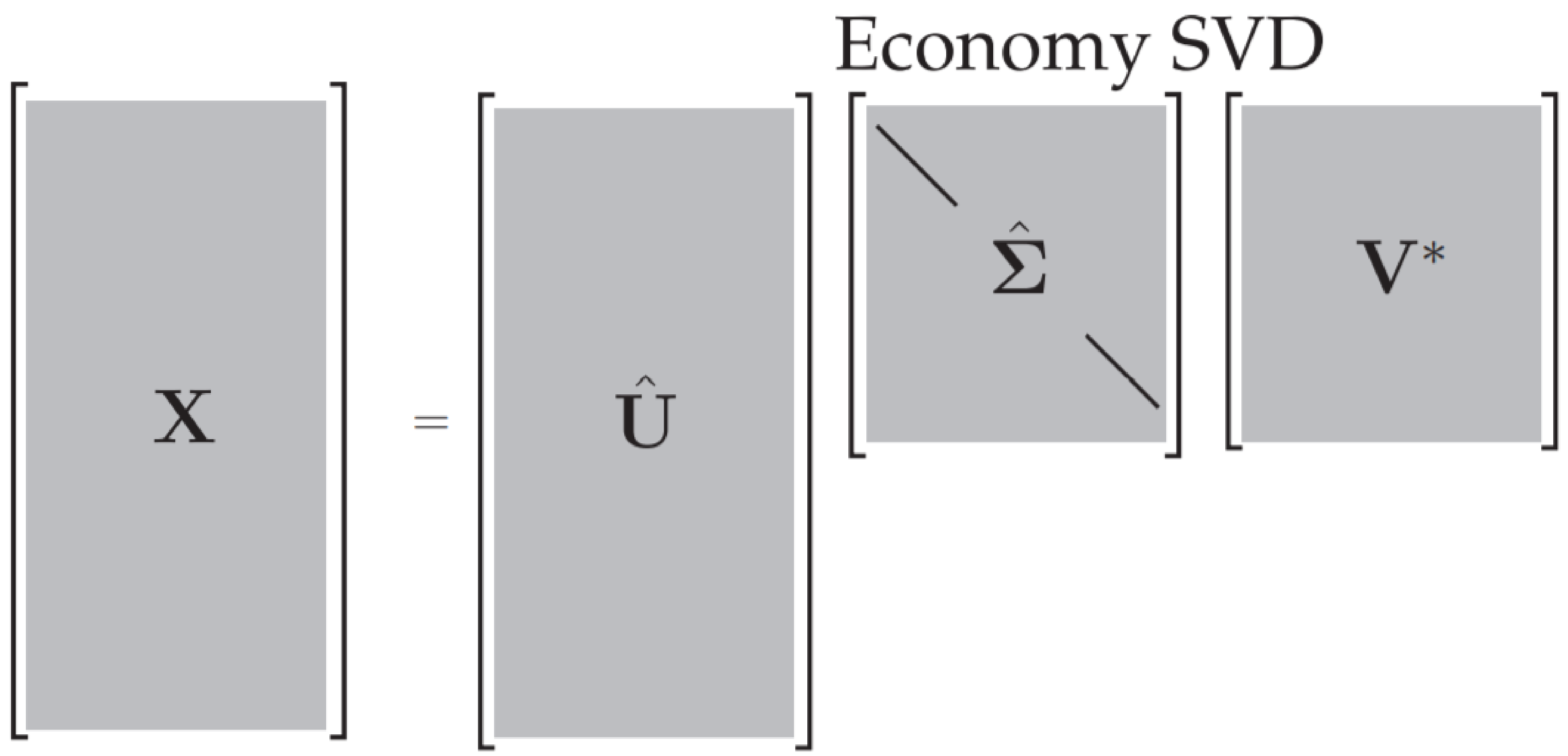
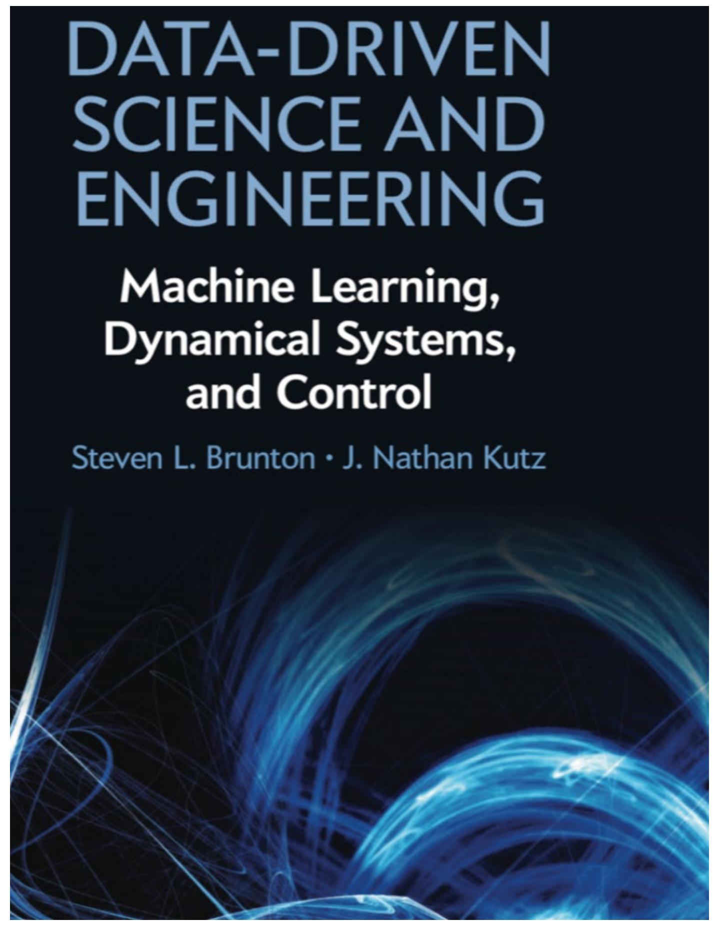


Figure 1.1 Schematic of matrices in the full and economy SVD.

Ref: Ch 1
http://www.databookuw.com/



Matrix Approximation using SVD

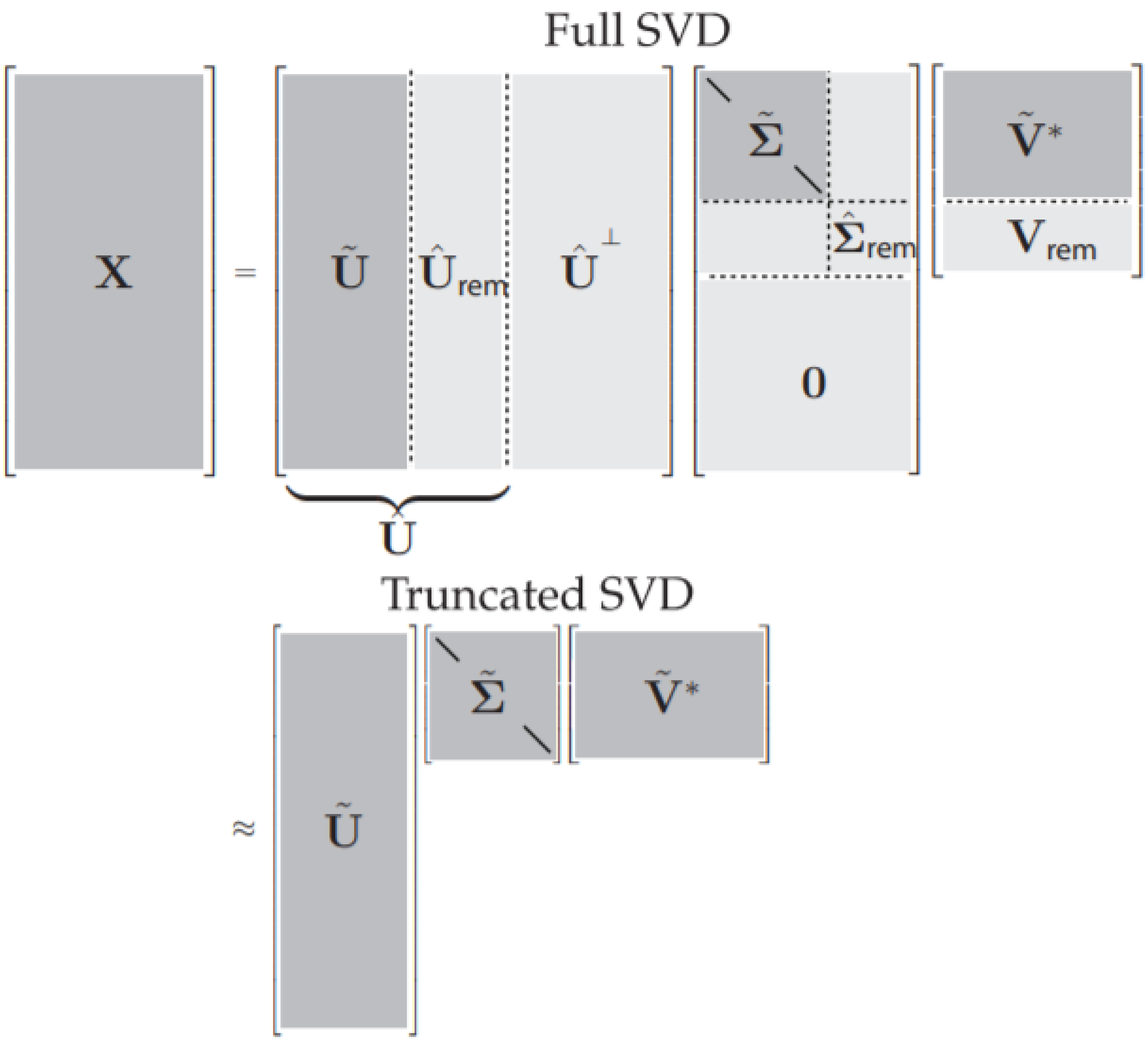


Figure 1.2 Schematic of truncated SVD. The subscript '$_{\mathrm{rem}}$' denotes the remainder of $\hat{\mathbf{U}}$, $\hat{\boldsymbol{\Sigma}}$ or $\mathbf{V}$ after truncation.

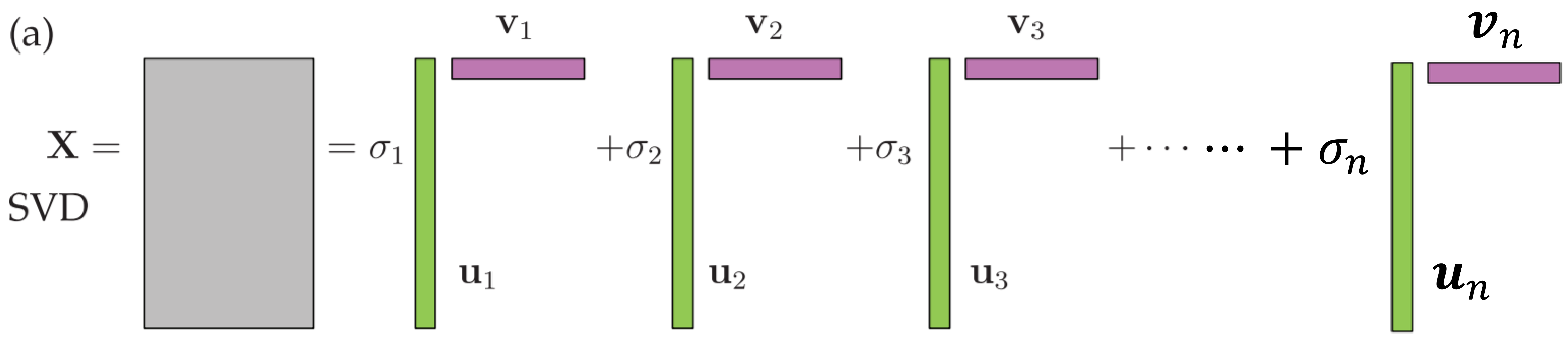
Truncation

The truncated SVD is illustrated in Fig. 1.2, with $\tilde{\mathbf{U}}$, $\tilde{\boldsymbol{\Sigma}}$ and $\tilde{\mathbf{V}}$ denoting the truncated matrices. If $\mathbf{X}$ does not have full rank, then some of the singular values in $\hat{\boldsymbol{\Sigma}}$ may be zero, and the truncated SVD may still be exact. However, for truncation values r that are smaller than the number of nonzero singular values (i.e., the rank of $\mathbf{X}$), the truncated SVD only approximates $\mathbf{X}$:

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.6}$$

There are numerous choices for the truncation rank r, and they are discussed in Sec. 1.7. If we choose the truncation value to keep all non-zero singular values, then $\mathbf{X} = \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^*$ is exact.

Viewing Approximation as sum of Outer Product: Dyadic summation



$$X = \sum \sigma_k u_k v_k \quad for\ k = 1..n$$

$$\tilde{\mathbf{X}} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^* .$$

Matrix Approximation using SVD

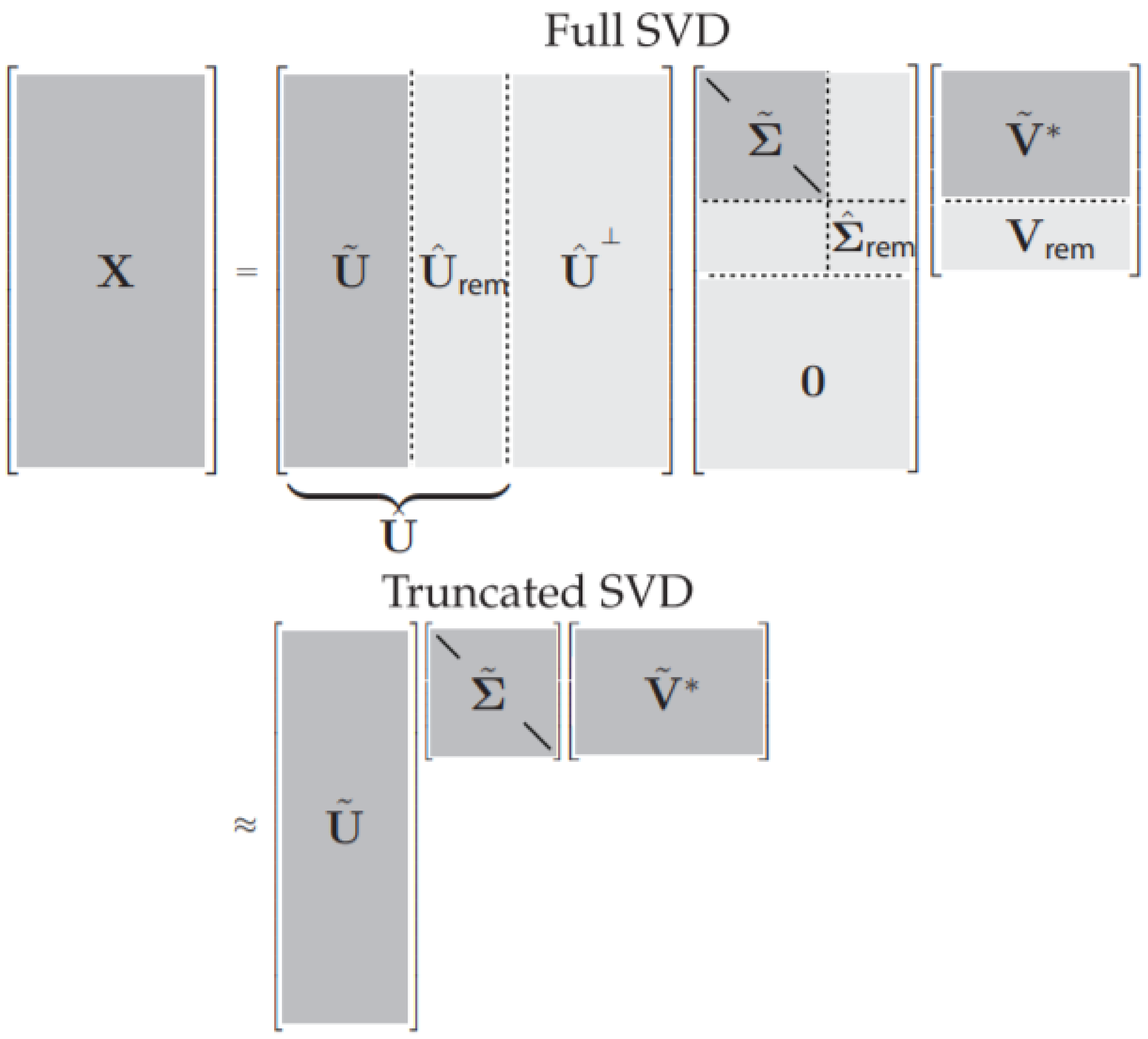


Figure 1.2 Schematic of truncated SVD. The subscript '$_{\text{rem}}$' denotes the remainder of $\hat{\mathbf{U}}$, $\hat{\boldsymbol{\Sigma}}$ or $\mathbf{V}$ after truncation.

Theorem 1 (Eckart-Young [170]) *The optimal rank-r approximation to $\mathbf{X}$, in a least-squares sense, is given by the rank-r SVD truncation $\tilde{\mathbf{X}}$:*

$$\underset{\tilde{\mathbf{X}},\ s.t.\ \operatorname{rank}(\tilde{\mathbf{X}})=r}{\operatorname{argmin}} \|\mathbf{X}-\tilde{\mathbf{X}}\|_F = \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.4}$$

Here, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ denote the first r leading columns of $\mathbf{U}$ and $\mathbf{V}$, and $\tilde{\boldsymbol{\Sigma}}$ contains the leading $r \times r$ sub-block of $\boldsymbol{\Sigma}$. $\|\cdot\|_F$ is the Frobenius norm.

Here, we establish the notation that a truncated SVD basis (and the resulting approximated matrix $\tilde{\mathbf{X}}$) will be denoted by $\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^*$. Because $\boldsymbol{\Sigma}$ is diagonal, the rank-r SVD approximation is given by the sum of r distinct rank-1 matrices:

$$\tilde{\mathbf{X}} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*. \tag{1.5}$$

Frobenius Norm

DOWNLOAD
Wolfram Notebook

The Frobenius norm, sometimes also called the Euclidean norm (a term unfortunately also used for the vector L^2-norm), is matrix norm of an $m \times n$ matrix $\mathbf{A}$ defined as the square root of the sum of the absolute squares of its elements,

$$\|\mathbf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

(Golub and van Loan 1996, p. 55).

Brunton: The Frobenius Norm of a Matrix
https://www.youtube.com/watch?v=Gt56YxMBlVA

Matrix Approximation: example picture

First, we load the image:

```
A=imread('../DATA/dog.jpg');
X=double(rgb2gray(A)); % Convert RBG->gray, 256 bit->double.
nx = size(X,1); ny = size(X,2);
imagesc(X), axis off, colormap gray
```

and take the SVD:

```
[U,S,V] = svd(X);
```

Next, we compute the approximate matrix using the truncated SVD for various ranks ($r = 5$, 20, and 100):

```
for r=[5 20 100];  % Truncation value
    Xapprox = U(:,1:r)*S(1:r,1:r)*V(:,1:r)'; % Approx. image
    figure, imagesc(Xapprox), axis off
    title(['r=',num2str(r,'%d'),']);
end
```

Finally, we plot the singular values and cumulative energy in Fig. 1.4:

```
subplot(1,2,1), semilogy(diag(S),'k')
subplot(1,2,2), plot(cumsum(diag(S))/sum(diag(S)),'k')
```

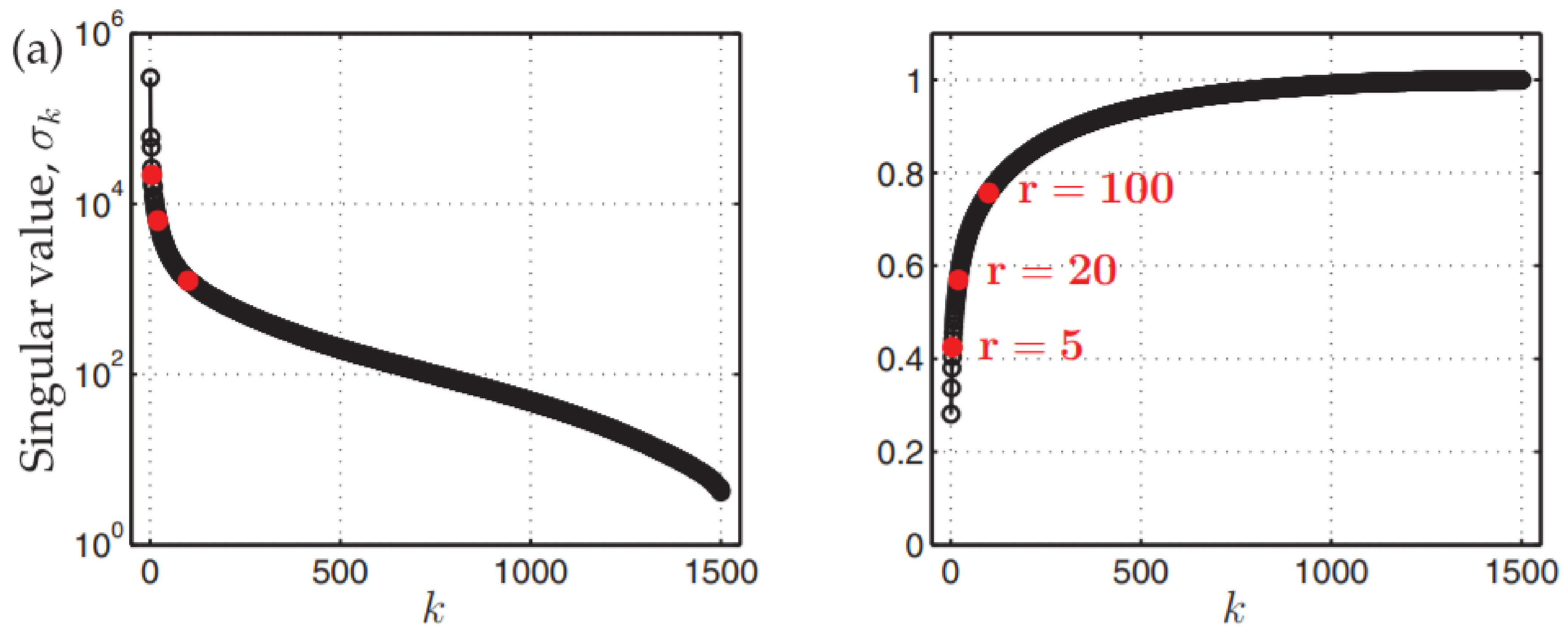


Figure 1.4 (a) Singular values σ_k. (b) Cumulative energy in the first k modes.

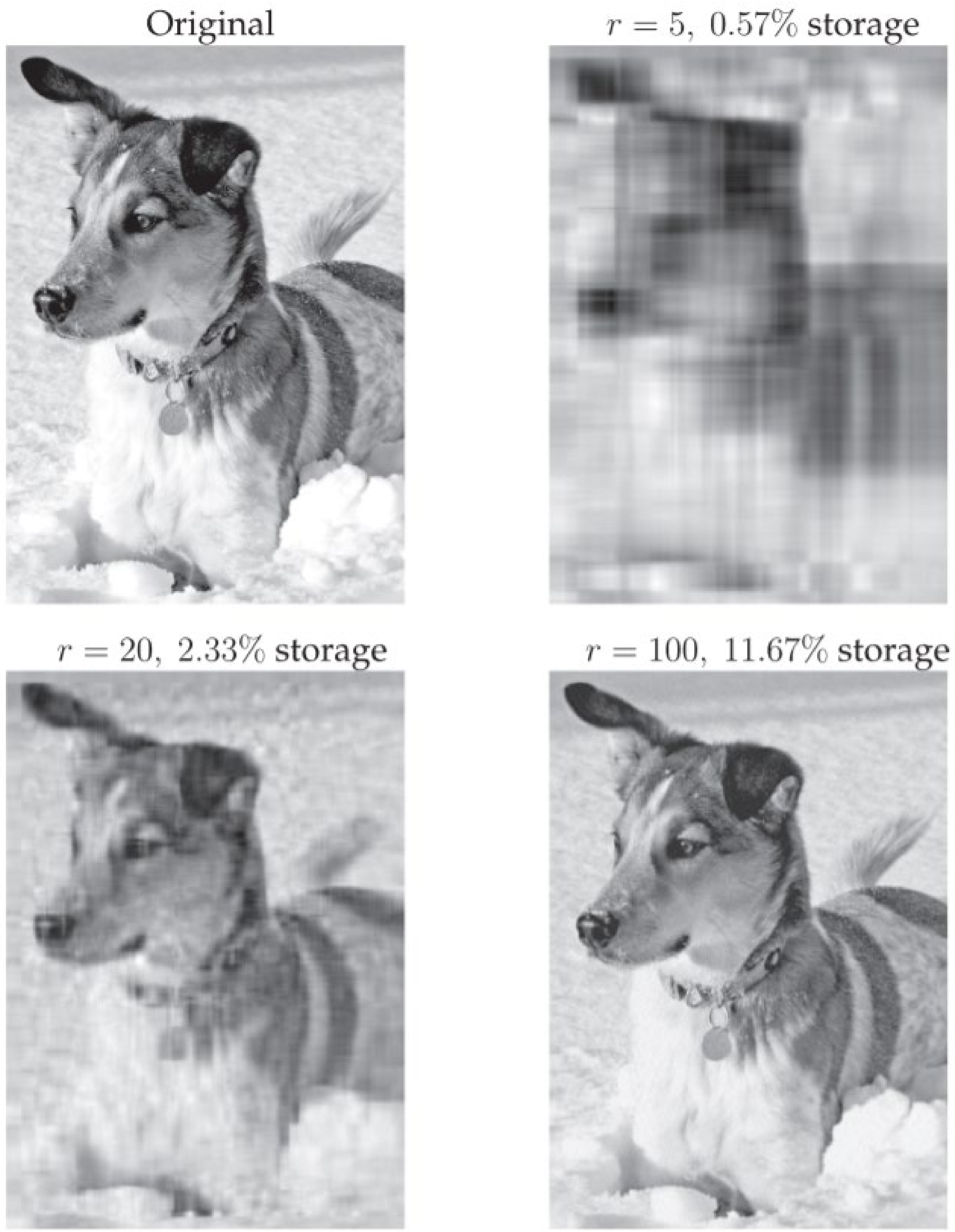


Figure 1.3 Image compression of Mordecai the snow dog, truncating the SVD at various ranks r. Original image resolution is 2000×1500.

Approximating Lena

.\Code\test_lena.m

```
close all; clear all

load lena512
imshow(lena512,[0 255])
[U,D,V] = svd(lena512);
diagD = diag(D);
[m,n] = size(lena512);

subplot(1,2,1); semilogy(diag(D),'k'); ylabel('singular value \sigma_k'); xlabel('k');
subplot(1,2,2); plot(cumsum(diag(D))/sum(diag(D)),'k'); ylabel('cumulative energy'); xlabel('k')


subplot(3,1,1);imshow(lena512,[0 255])
pause


for r = 1:1:100
    opStr=sprintf('Using %d singular values',r);
    subplot(3,1,1);imshow(lena512,[0 255])
    approxLena_r = U(:,1:r)*D(1:r,1:r)*((V(:,1:r))');

    subplot(3,1,2);imshow(approxLena_r,[0 255])
    diffImage =(lena512-approxLena_r);
    maxDiffVal = max(max(diffImage));
    subplot(3,1,3); imshow(diffImage,[0 maxDiffVal])
    title(opStr);

    pause(0.1);
end
```