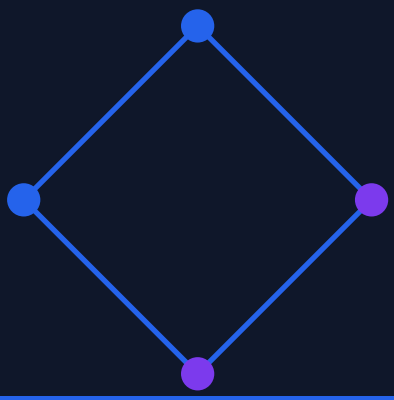
PMAI LAB

Mathematics • Python • Artificial Intelligence



LINEAR ALGEBRA FOR MACHINE LEARNING

Lecture 08

Singular Value Decomposition, PCA, Low-Rank Approximation, and Data Compression

Lecture focus Decomposing any data matrix into orthogonal directions, identifying informative components, and building compact approximations for dimensionality reduction and compression.

Dr. Hira Benish

Associate Professor of Mathematics

Student Learning Handout | Includes NumPy practice and review exercises

1 Lecture Overview

Eigenvalue decomposition is powerful but applies directly only to square matrices and may not provide an orthogonal basis. Singular value decomposition works for every real or complex matrix and exposes its most important input and output directions.

SVD supports least squares, pseudoinverses, matrix rank, PCA, recommendation systems, latent semantic analysis, denoising, and image compression. It is one of the most useful tools in applied linear algebra and machine learning.

Learning Outcomes

After completing this lecture, students should be able to:

- state the full and compact forms of the SVD;
- interpret left and right singular vectors geometrically;
- connect singular values with eigenvalues of A^TA and AA^T;
- determine rank, condition number, and pseudoinverse from the SVD;
- construct a truncated SVD and low-rank approximation;
- explain the Eckart-Young optimality result;
- derive PCA from the SVD of centered data;
- calculate explained-variance ratios; and
- implement PCA and image compression using Python.

1.1 Why SVD Is So General

Property	SVD advantage
Matrix shape	Works for square and rectangular matrices
Existence	Every matrix has an SVD
Geometry	Uses orthonormal input and output directions
Rank	Number of positive singular values
Compression	Orders components by decreasing strength
Numerical analysis	Reveals ill-conditioning and near dependence

2 The Singular Value Decomposition

For $A \in \mathbb{R}^{m\times n}$, the full SVD is

$$A = U\Sigma V^T,$$

where

$$U \in \mathbb{R}^{m\times m}, \quad \Sigma \in \mathbb{R}^{m\times n}, \quad V \in \mathbb{R}^{n\times n}.$$

The columns of U and V are orthonormal:

$$U^TU = I, \qquad V^TV = I.$$

The diagonal entries of Σ are the singular values

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0,$$

where $r = \text{rank}(A)$.

2.1 Compact SVD

Keeping only the r positive singular values gives

$$A = U_r\Sigma_r V_r^T,$$

where

$$U_r \in \mathbb{R}^{m\times r}, \quad \Sigma_r \in \mathbb{R}^{r\times r}, \quad V_r \in \mathbb{R}^{n\times r}.$$

2.2 Outer-Product Form

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

Each term is a rank-one matrix. The singular value σ_i measures the strength of that component.

Three-Stage Transformation

For $A\mathbf{x} = U\Sigma V^T\mathbf{x}$:

1. V^T expresses the input in right-singular-vector coordinates;
2. Σ scales those coordinates by the singular values; and
3. U places the result in the output space.

3 Connection with Eigenvalues and Subspaces

From $A = U\Sigma V^T$,

$$A^TA = V\Sigma^T\Sigma V^T, \qquad AA^T = U\Sigma\Sigma^T U^T.$$

Therefore,

$$A^TA\mathbf{v}_i = \sigma_i^2\mathbf{v}_i, \qquad AA^T\mathbf{u}_i = \sigma_i^2\mathbf{u}_i.$$

The right singular vectors are eigenvectors of A^TA, the left singular vectors are eigenvectors of AA^T, and singular values are square roots of the non-zero eigenvalues.

3.1 Fundamental Subspaces

- $\mathbf{v}_1, \ldots, \mathbf{v}_r$ form an orthonormal basis for Row(A).
- $\mathbf{u}_1, \ldots, \mathbf{u}_r$ form an orthonormal basis for Col(A).
- Remaining right singular vectors span Null(A).
- Remaining left singular vectors span Null(A^T).

3.2 Condition Number

For a full-rank matrix,

$$\kappa_2(A) = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}.$$

A large ratio indicates that some directions are transformed much more weakly than others, making inversion sensitive to noise.

3.3 Moore-Penrose Pseudoinverse

$$A^+ = V\Sigma^+U^T,$$

where each positive singular value is replaced by its reciprocal. The pseudoinverse extends inverse-like computation to rectangular and singular matrices.

Machine-Learning Connection

The pseudoinverse returns a minimum-norm least-squares solution. Small singular values reveal unstable directions and motivate truncation or regularization.

4 Worked Rank-One Example

Consider

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Both columns are identical, so rank$(A) = 1$.

One compact SVD is

$$\mathbf{u}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \sigma_1 = 2, \qquad \mathbf{v}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Therefore,

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T = 2 \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix} \right) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

The second singular value is zero. The direction

$$\mathbf{v}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

lies in the null space because $A\mathbf{v}_2 = \mathbf{0}$.

4.1 Geometric Interpretation

- The direction $[1, 1]^T$ is scaled by 2.
- The perpendicular direction $[1, -1]^T$ is completely removed.
- Every output lies on the line spanned by $[1, 1]^T$.

Sign Ambiguity

If $\mathbf{u}_i$ and $\mathbf{v}_i$ are both multiplied by -1, their outer product is unchanged. Software may therefore return singular vectors with different signs while representing the same SVD.

5 Low-Rank Approximation

The rank-k truncated SVD is

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = U_k \Sigma_k V_k^T, \qquad k < r.$$

It keeps the strongest components and discards weaker directions.

5.1 Eckart-Young Theorem

Among all matrices B of rank at most k, A_k is a best approximation to A under both the spectral and Frobenius norms:

$$\|A - A_k\|_2 = \sigma_{k+1},$$
$$\|A - A_k\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}.$$

5.2 Retained Energy

A common component-selection measure is

$$\text{retained energy}(k) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

5.3 Storage Comparison

An $m \times n$ matrix stores mn values. A rank-k SVD representation stores approximately

$$k(m + n + 1)$$

values: U_k, the k singular values, and V_k.

Compression is beneficial when $k(m + n + 1) \ll mn$.

Compression versus Denoising

Discarded components may contain both noise and useful detail. Choosing k balances compactness against reconstruction quality and should be guided by the task, not only a fixed percentage.

Machine-Learning Connection

Low-rank models capture latent structure in images, user-item matrices, text-term matrices, and embeddings. The approximation replaces many original variables with a smaller set of learned factors.

6 Principal Component Analysis through SVD

Let $X \in \mathbb{R}^{m \times n}$ contain observations as rows and features as columns. First center every feature:

$$X_c = X - \mathbf{1}\boldsymbol{\mu}^T.$$

Compute

$$X_c = U\Sigma V^T.$$

6.1 PCA Quantities

- Principal directions: columns of V.
- Component scores: $Z = X_c V = U\Sigma$.
- Sample covariance: $C = X_c^T X_c/(m-1)$.
- Explained variance: $\lambda_i = \sigma_i^2/(m-1)$.
- Explained-variance ratio:

$$\frac{\sigma_i^2}{\sum_j \sigma_j^2}.$$

Keeping the first k components gives

$$Z_k = X_c V_k$$

and the reconstruction

$$\widehat{X} = Z_k V_k^T + \mathbf{1}\boldsymbol{\mu}^T.$$

6.2 Centering and Scaling

- Centering is essential because PCA studies variation around the mean.
- Standardization is often appropriate when features use different units or scales.
- Standardization is not automatic; it changes the question from covariance structure to correlation-like structure.

> PCA is unsupervised: it preserves directions of large input variance, not necessarily directions most useful for predicting a target.

Machine-Learning Connection

PCA supports visualization, compression, noise reduction, preprocessing, and multicollinearity management. Component meaning must still be interpreted using the original feature loadings.

7 Python and NumPy Practice

7.1 Computing and Verifying an SVD

```python
import numpy as np

A = np.array([
    [3.0, 1.0, 1.0],
    [-1.0, 3.0, 1.0]
])

U, singular_values, Vt = np.linalg.svd(
    A, full_matrices=False
)

A_reconstructed = (U * singular_values) @ Vt

print("Singular values:", singular_values)
print("Reconstruction correct:",
      np.allclose(A, A_reconstructed))
print("Rank:", np.linalg.matrix_rank(A))
print("Condition number:", np.linalg.cond(A))
```

7.2 Truncated SVD Function

```python
def truncated_svd(A, k):
    U, s, Vt = np.linalg.svd(A, full_matrices=False)
    if not 1 <= k <= len(s):
        raise ValueError("k is outside the valid range.")
    return (U[:, :k] * s[:k]) @ Vt[:k, :]

A_rank1 = truncated_svd(A, k=1)
relative_error = (
    np.linalg.norm(A - A_rank1, "fro") /
    np.linalg.norm(A, "fro")
)

print("Rank-1 approximation:\n", A_rank1)
print("Relative error:", relative_error)
```

7.3 Pseudoinverse Check

```python
A_plus = np.linalg.pinv(A)
print(np.allclose(A @ A_plus @ A, A))
```

8 Applications: PCA and Image Compression

8.1 PCA with NumPy

```
X = np.array([
    [2.0, 0.0], [3.0, 1.0], [4.0, 1.0],
    [5.0, 3.0], [6.0, 4.0]
])

mean = X.mean(axis=0)
X_centered = X - mean
U, s, Vt = np.linalg.svd(X_centered, full_matrices=False)

explained_ratio = s ** 2 / np.sum(s ** 2)
scores_1d = X_centered @ Vt[:1, :].T
X_reconstructed = scores_1d @ Vt[:1, :] + mean

print("Explained ratio:", explained_ratio)
print("Reduced shape:", scores_1d.shape)
print("Reconstructed data:\n", X_reconstructed)
```

8.2 Compressing a Handwritten-Digit Image

```
from sklearn.datasets import load_digits

digits = load_digits()
image = digits.images[0]

U, s, Vt = np.linalg.svd(image, full_matrices=False)
k = 2
compressed = (U[:, :k] * s[:k]) @ Vt[:k, :]

original_values = image.size
svd_values = k * (image.shape[0] + image.shape[1] + 1)
relative_error = (
    np.linalg.norm(image - compressed, "fro") /
    np.linalg.norm(image, "fro")
)

print("Original storage:", original_values)
print("SVD storage:", svd_values)
print("Relative error:", relative_error)
```

Machine-Learning Connection

The compressed image keeps the strongest spatial patterns while discarding weaker components. Larger k improves visual quality but reduces the storage advantage.

9 Review and Independent Practice

Practice Tasks

1. State the dimensions of U, Σ, and V for the full SVD of a 5×3 matrix.
2. Explain how singular vectors relate to the four fundamental subspaces.
3. Find a compact SVD of $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ and verify its rank-one reconstruction.
4. Use singular values to calculate rank and condition number for a matrix of your choice.
5. Construct rank-1 and rank-2 approximations of a matrix and compare their Frobenius errors.
6. Apply PCA to a dataset with at least three features. Report the number of components needed for 90% explained variance.
7. Compress a grayscale image for several values of k and compare storage with reconstruction error.

9.1 Key Takeaways

- Every matrix has an SVD with orthonormal input and output directions.
- Singular values measure the strength of independent matrix components.
- SVD reveals rank, subspaces, conditioning, and pseudoinverse structure.
- Truncated SVD gives an optimal low-rank approximation.
- PCA is obtained from the SVD of a centered data matrix.
- Explained variance helps select components but must be balanced with task performance.
- Compression replaces the original matrix with a smaller factor representation.

9.2 Key Terms

Singular value	Left singular vector	Right singular vector
Compact SVD	Pseudoinverse	Condition number
Truncated SVD	Low-rank approximation	Frobenius norm
PCA	Explained variance	Reconstruction error

Next lecture: Linear algebra in neural networks, embeddings, attention, and gradient-based learning.