Professor‘s Project 3 SVD

June 22, 2025

[1]:
```python
import numpy as np
import pandas as pd

def demonstrate_svd():
    """
    Demonstrates SVD using a movie ratings matrix where:
    - Rows represent users
    - Columns represent movies
    - Values are ratings (1-5 scale)
    """

    print("=== Singular Value Decomposition (SVD) for Movie Ratings ===\n")

    # Create a toy movie ratings matrix (users × movies)
    # This is a rectangular matrix with complete ratings (no missing values)
    users = ['Alice', 'Bob', 'Carol', 'Dave', 'Eve']
    movies = ['The Matrix', 'Titanic', 'Star Wars', 'The Godfather']

    # 5 users × 4 movies matrix with ratings 1-5
    ratings_matrix = np.array([
        [5, 2, 4, 1],  # Alice
        [4, 1, 5, 2],  # Bob
        [1, 5, 2, 4],  # Carol
        [2, 4, 1, 5],  # Dave
        [3, 3, 3, 3]   # Eve
    ], dtype=float)

    print("Original Movie Ratings Matrix (A):")
    print("Shape:", ratings_matrix.shape, "(5 users × 4 movies)")
    print("Note: This is a rectangular matrix with all values filled in\n")

    # Display as a nice table
    df = pd.DataFrame(ratings_matrix, index=users, columns=movies)
    print(df)
    print("\n" + "="*50 + "\n")

    # Perform SVD: A = U × Σ × V^T
```

```python
U, sigma, Vt = np.linalg.svd(ratings_matrix, full_matrices=False)

print("SVD Decomposition: A = U × Σ × V^T\n")

# U matrix (users × features)
print(f"U Matrix (Left Singular Vectors) - Shape: {U.shape}")
print("Represents users in the latent feature space:")
print("Each row is a user, each column is a latent feature")
print(np.round(U, 3))
print()

# Sigma (diagonal values only)
print(f"Singular Values ( ) - Shape: {sigma.shape}")
print("These are the diagonal elements of the Σ matrix:")
print("They represent the importance/strength of each latent feature")
print(np.round(sigma, 3))
print()

# Create full Sigma matrix for visualization
Sigma = np.zeros((U.shape[1], Vt.shape[0]))
np.fill_diagonal(Sigma, sigma)
print(f"Full Σ Matrix - Shape: {Sigma.shape}")
print(np.round(Sigma, 3))
print()

# V^T matrix (features × movies)
print(f"V^T Matrix (Right Singular Vectors) - Shape: {Vt.shape}")
print("Represents movies in the latent feature space:")
print("Each row is a latent feature, each column is a movie")
print(np.round(Vt, 3))
print()

print("="*50 + "\n")

# Verify reconstruction
print("Verification: Reconstructing the original matrix")
reconstructed = U @ Sigma @ Vt
print("U × Σ × V^T =")
print(np.round(reconstructed, 3))
print()

# Check if reconstruction is accurate
reconstruction_error = np.linalg.norm(ratings_matrix - reconstructed)
print(f"Reconstruction error (should be ~0): {reconstruction_error:.10f}")
print()

print("="*50 + "\n")
```

```
    # Demonstrate dimensionality reduction
    print("Dimensionality Reduction Example:")
    print("Using only the top 2 singular values (rank-2 approximation)")

    # Keep only top 2 components
    k = 2
    U_reduced = U[:, :k]
    sigma_reduced = sigma[:k]
    Vt_reduced = Vt[:k, :]

    print(f"U_reduced shape: {U_reduced.shape}")
    print(f"sigma_reduced shape: {sigma_reduced.shape}")
    print(f"Vt_reduced shape: {Vt_reduced.shape}")

    # Reconstruct with reduced dimensions
    Sigma_reduced = np.diag(sigma_reduced)
    approximated = U_reduced @ Sigma_reduced @ Vt_reduced

    print("\nRank-2 Approximation:")
    df_approx = pd.DataFrame(np.round(approximated, 2), index=users,␣
 ↪columns=movies)
    print(df_approx)

    approximation_error = np.linalg.norm(ratings_matrix - approximated)
    print(f"\nApproximation error: {approximation_error:.3f}")

    # Show which singular values we kept vs discarded
    print(f"\nSingular values kept: {np.round(sigma_reduced, 3)}")
    print(f"Singular values discarded: {np.round(sigma[k:], 3)}")
    print(f"Energy retained: {np.sum(sigma_reduced**2) / np.sum(sigma**2):.1%}")

def explain_interpretation():
    """
    Explains the interpretation of SVD components in the user-movie context
    """
    print("\n" + "="*60)
    print("INTERPRETATION IN USER-MOVIE CONTEXT:")
    print("="*60)

    interpretations = [
        "MATRIX LAYOUT (Users × Movies):",
        "• Rows = Users (Alice, Bob, Carol, Dave, Eve)",
        "• Columns = Movies (The Matrix, Titanic, Star Wars, The Godfather)",
        "• Values = Ratings (1-5 scale)",
        "",
        "U MATRIX (Users × Latent Features):",
        "• Each row represents a user's preferences across latent features",
        "• Latent features might represent genres like 'Action', 'Romance', etc.
 ↪",
        "• U[i,j] = how much user i likes latent feature j",
        "",
        "Σ MATRIX (Feature Importance):",
        "• Diagonal values show importance of each latent feature",
        "• Larger values = more important patterns in the data",
        "• Helps identify dominant preferences in the dataset",
        "",
        "V^T MATRIX (Features × Movies):",
        "• Each column represents a movie's characteristics",
        "• Each row represents a latent feature (e.g., genre)",
        "• V^T[i,j] = how much movie j belongs to latent feature i",
        "",
        "PRACTICAL APPLICATIONS:",
        "• Recommend movies: Find similar users (rows in U)",
        "• Find similar movies: Find similar columns in V^T",
        "• Reduce storage: Use only top k features",
        "• Handle missing ratings: SVD can help fill gaps"
    ]

    for interpretation in interpretations:
        print(interpretation)

def explain_svd_properties():
    """
    Explains key properties and requirements of SVD
    """
    print("\n" + "="*60)
    print("KEY PROPERTIES OF SVD:")
    print("="*60)

    properties = [
        "1. MATRIX REQUIREMENTS:",
        "   • Can be applied to ANY matrix (square or rectangular)",
        "   • Must have ALL values filled in (no missing entries)",
        "   • In our example: 5×4 matrix (5 users, 4 movies)",
        "",
        "2. DECOMPOSITION COMPONENTS:",
        "   • U: Left singular vectors (users → latent features)",
        "   • Σ: Singular values (feature importance, always   0)",
        "   • V^T: Right singular vectors (latent features → movies)",
        "",
        "3. MATHEMATICAL PROPERTIES:",
        "   • U columns are orthonormal (U^T × U = I)",
        "   • V^T rows are orthonormal (V^T × V = I)",
        "   • Singular values are in descending order",
        "",
        "4. APPLICATIONS:",
        "   • Dimensionality reduction (keep top k components)",
        "   • Collaborative filtering for recommendations",
        "   • Data compression and noise reduction",
        "   • Matrix completion (filling missing values)",
        "   • Principal Component Analysis (PCA)"
    ]

    for prop in properties:
        print(prop)

if __name__ == "__main__":
    demonstrate_svd()
    explain_interpretation()
    explain_svd_properties()
```

```
=== Singular Value Decomposition (SVD) for Movie Ratings ===

Original Movie Ratings Matrix (A):
Shape: (5, 4) (5 users × 4 movies)
Note: This is a rectangular matrix with all values filled in

       The Matrix  Titanic  Star Wars  The Godfather
Alice         5.0      2.0        4.0            1.0
Bob           4.0      1.0        5.0            2.0
Carol         1.0      5.0        2.0            4.0
Dave          2.0      4.0        1.0            5.0
Eve           3.0      3.0        3.0            3.0

==================================================

SVD Decomposition: A = U × Σ × V^T

U Matrix (Left Singular Vectors) - Shape: (5, 4)
Represents users in the latent feature space:
Each row is a user, each column is a latent feature
[[-0.447  0.5   -0.5    0.5  ]
 [-0.447  0.5    0.5   -0.5  ]
 [-0.447 -0.5    0.5    0.5  ]
 [-0.447 -0.5   -0.5   -0.5  ]
 [-0.447  0.     0.     0.   ]]

Singular Values ( ) - Shape: (4,)
These are the diagonal elements of the Σ matrix:
They represent the importance/strength of each latent feature
```

```
[13.416  6.     1.414  1.414]

Full Σ Matrix - Shape: (4, 4)
[[13.416  0.     0.     0.   ]
 [ 0.     6.     0.     0.   ]
 [ 0.     0.     1.414  0.   ]
 [ 0.     0.     0.     1.414]]

V^T Matrix (Right Singular Vectors) - Shape: (4, 4)
Represents movies in the latent feature space:
Each row is a latent feature, each column is a movie
[[-0.5   -0.5   -0.5   -0.5  ]
 [ 0.5   -0.5    0.5   -0.5  ]
 [-0.707  0.     0.707 -0.   ]
 [-0.     0.707 -0.    -0.707]]

==================================================

Verification: Reconstructing the original matrix
U × Σ × V^T =
[[5. 2. 4. 1.]
 [4. 1. 5. 2.]
 [1. 5. 2. 4.]
 [2. 4. 1. 5.]
 [3. 3. 3. 3.]]

Reconstruction error (should be ~0): 0.0000000000

==================================================

Dimensionality Reduction Example:
Using only the top 2 singular values (rank-2 approximation)
U_reduced shape: (5, 2)
sigma_reduced shape: (2,)
Vt_reduced shape: (2, 4)

Rank-2 Approximation:
       The Matrix  Titanic  Star Wars  The Godfather
Alice         4.5      1.5        4.5            1.5
Bob           4.5      1.5        4.5            1.5
Carol         1.5      4.5        1.5            4.5
Dave          1.5      4.5        1.5            4.5
Eve           3.0      3.0        3.0            3.0

Approximation error: 2.000

Singular values kept: [13.416  6.   ]
Singular values discarded: [1.414 1.414]
```

```
Energy retained: 98.2%

============================================================
INTERPRETATION IN USER-MOVIE CONTEXT:
============================================================
MATRIX LAYOUT (Users × Movies):
• Rows = Users (Alice, Bob, Carol, Dave, Eve)
• Columns = Movies (The Matrix, Titanic, Star Wars, The Godfather)
• Values = Ratings (1-5 scale)

U MATRIX (Users × Latent Features):
• Each row represents a user's preferences across latent features
• Latent features might represent genres like 'Action', 'Romance', etc.
• U[i,j] = how much user i likes latent feature j

Σ MATRIX (Feature Importance):
• Diagonal values show importance of each latent feature
• Larger values = more important patterns in the data
• Helps identify dominant preferences in the dataset

V^T MATRIX (Features × Movies):
• Each column represents a movie's characteristics
• Each row represents a latent feature (e.g., genre)
• V^T[i,j] = how much movie j belongs to latent feature i

PRACTICAL APPLICATIONS:
• Recommend movies: Find similar users (rows in U)
• Find similar movies: Find similar columns in V^T
• Reduce storage: Use only top k features
• Handle missing ratings: SVD can help fill gaps

============================================================
KEY PROPERTIES OF SVD:
============================================================
1. MATRIX REQUIREMENTS:
   • Can be applied to ANY matrix (square or rectangular)
   • Must have ALL values filled in (no missing entries)
   • In our example: 5×4 matrix (5 users, 4 movies)

2. DECOMPOSITION COMPONENTS:
   • U: Left singular vectors (users → latent features)
   • Σ: Singular values (feature importance, always  0)
   • V^T: Right singular vectors (latent features → movies)

3. MATHEMATICAL PROPERTIES:
   • U columns are orthonormal (U^T × U = I)
   • V^T rows are orthonormal (V^T × V = I)
   • Singular values are in descending order
```

4. APPLICATIONS:
    - Dimensionality reduction (keep top k components)
    - Collaborative filtering for recommendations
    - Data compression and noise reduction
    - Matrix completion (filling missing values)
    - Principal Component Analysis (PCA)

[ ]:
```

```