Machine Learning - Assignment 1

Anu Zacharia

January 12, 2023

---

1 What is Singular Value Decomposition (SVD)?

SVD is a linear dimensionality reduction method used in feature engineering. It helps us to reduce the data to the key features that are necessary for analysing and describing the data.

1.1 Mathematical Definition

SVD is a method to factorize a real or complex matrix into three matrices as follows.

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}(V_{n\times n})^T$$

where
U contains orthonormal eigenvectors of AA^Tin its columns
V contains orthonormal eigenvectors of A^TAin its columns
Σ contains the k = min(m,n) singular values, σ_i 1,..,k on its diagonal that are the square roots of the nonzero eigenvalues of both AA^Tnd A^TA.

It is a diagonal matrix and its values are arranged in the descending order.
U and V^T are orthogonal matrices. The diagonal entries of Σ
are called the singular values of A.

The columns of U are called the left singular vectors, and those of V are called the right singular vectors. The singular values are unique, but U and V are not unique.

The number of nonzero singular values is equal to the rank of the matrix A.

1.2 Example

SVD of a rectangular matrix can be obtained as follows.

Let A = $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

1. Determine matrix V

a. Find A^TA

$$\text{A\^{}T A} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b. Find eigen values of A^TA

$$\begin{vmatrix} 1-\lambda & 1 & 0 \\ 1 & 1-\lambda & 0 \\ 0 & 0 & 1-\lambda \end{vmatrix} = 0$$

The eigen values for the matrix $\lambda 1 = 0$ $\lambda 2 = 1$ and $\lambda 3 = 2$.

c. Find eigen vectors of A^TA

The eigen vectors corresponding to the eigen value $\lambda 3 = 2$ is

$$\text{w1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

The eigen vectors corresponding to the eigen value $\lambda 3 = 1$ is

$$\text{w2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The eigen vectors corresponding to the eigen value $\lambda 3 = 0$ is

$$\text{w3} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

c. Normalizing the eigen vectors

$$\text{v1} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix} \text{ v2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ v3} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

So we get

$$\text{V} = \begin{bmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \end{bmatrix}$$

c. Find Σ which contains square root of the eigen values

$\Sigma \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

2. Determine matrix U

Method 1

a. Find AA^T

$$\text{A}A^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

b. Find eigen values of AA^T

$$\begin{vmatrix} 2-\lambda & 0 \\ 0 & 1-\lambda \end{vmatrix} = 0$$

The eigen values for the matrix $\lambda 1 = 1$ and $\lambda 2 = 2$.

c. Find eigen vectors of AA^T

The eigen vectors corresponding to the eigen value $\lambda 1 = 2$ is

w1 $= \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

The eigen vectors corresponding to the eigen value $\lambda 2 = 1$ is

w2 $= \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

So we get

U $= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Method 2

To find U , we can find u1 and u2 as follows

u1 $= 1/\sigma 1(Av1)$

$$u1 = 1/\sqrt{2} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$u2 = 1/\sigma 2(Av2)$

$$u2 = 1/1 \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

$$U = [u1,u2] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

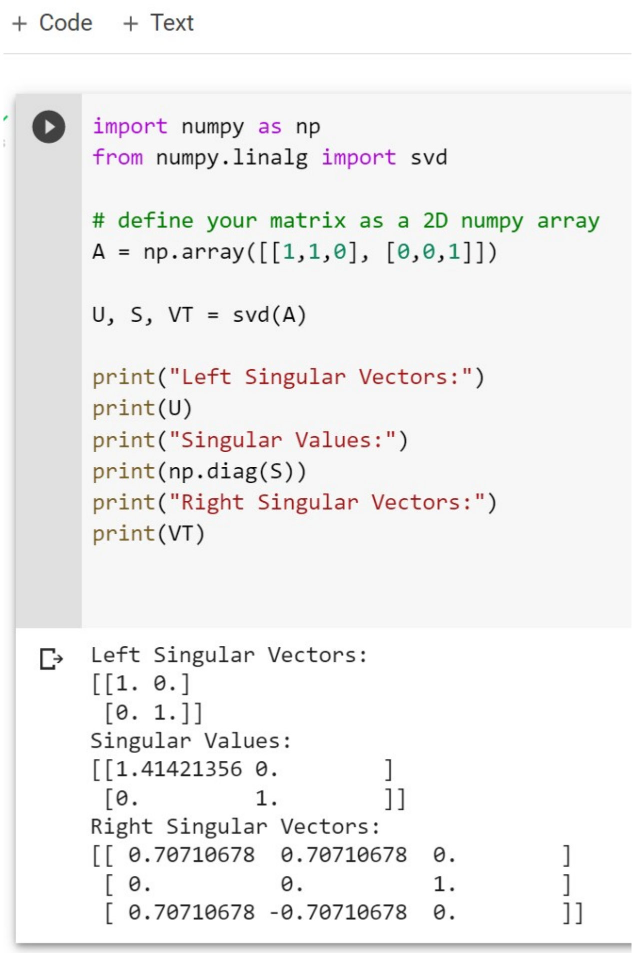
3. Find Σ which contains square root of the eigen values

$$\Sigma \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Thus we get the SVD of matrix A

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix}$$

1.3 Python implementation



```python
import numpy as np
from numpy.linalg import svd

# define your matrix as a 2D numpy array
A = np.array([[1,1,0], [0,0,1]])

U, S, VT = svd(A)

print("Left Singular Vectors:")
print(U)
print("Singular Values:")
print(np.diag(S))
print("Right Singular Vectors:")
print(VT)
```

```
Left Singular Vectors:
[[1. 0.]
 [0. 1.]]
Singular Values:
[[1.41421356 0.        ]
 [0.         1.        ]]
Right Singular Vectors:
[[ 0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]
 [ 0.70710678 -0.70710678  0.        ]]
```

1.4 How to use SVD for Dimensionality reduction

Though SVD is usually used in DSP for noise reduction , image compression it can also be successfully used for dimensionality reduction, especially when the data is sparse.

Sparse data refers to rows of data where many of the values are zero. This is often the case in some problem domains like recommender systems where a user has a rating for very few movies or songs in the database and zero ratings for all other cases. Another common example is a bag of words model of a text document, where the document has a count or frequency for some words and most words have a 0 value.

Examples of sparse data appropriate for applying SVD for dimensionality reduction:

Recommender Systems, Customer-Product purchases, User-Movie Ratings, Text Classification.

SVD can be thought of as a projection method where data with m-columns (features) is projected into a subspace with m or fewer columns, whilst retaining the essence of the original data.

To reduce dimensionality, Singular Value Decomposition (SVD) keeps lower-order bases (the ones with the largest singular values) and ignores higher-order bases (the ones with the smallest singular values). The rationale behind this strategy is that the low-order bases retain the characteristics of the data that contribute most to its variance and are likely to capture the most important aspects of the data.

Given a data set X (nxm), where n is the number of rows and m is the number of attributes, a low-rank SVD uses only k components (k min(m, n)). By eliminating the data with low sigma values , we can achieve a low rank approximation of the original data.

2 What is Linear Discrimant Analysis (LDA)?

Linear Discriminant Analysis, or LDA for short, is a predictive modeling algorithm for multi-class classification. It can also be used as a dimensionality reduction technique, providing a projection of a training dataset that best separates the examples by their assigned class. LDA is a technique for multi-class classification that can be used to automatically perform dimensionality reduction. Linear Discriminant Analysis seeks to best separate (or discriminate) the samples in the training dataset by their class value. Specifically, the model seeks to find a linear combination of input variables that achieves the maximum separation for samples between classes (class centroids or means) and the minimum separation of samples within each class. In practice, LDA for multi-class classification is typically implemented using the tools from linear algebra, and like PCA, uses matrix factorization at the core of the technique.

The goal of LDA is to maximise the difference between the average of the two classes and minimize the variance within the class.

LDA assumes that in class variance of the individual classes are same.

The aim of LDA is to find a vector which best separates the classes.

2.1 Mathematical Procedure

The steps to find the vector

1. Find in-class scatter matrix of each class S˙w

2. Find between scatter matrices among the class S˙b

3. It can be derived that the direction of vector which separates the classes by LDA is the eigen vector corresponding to the largest eigen value of the matrix S˙wˆ-1 S˙b

Example:

Consider two classes

C1 - (4,2),(2,4),(2,3),(3,6),(4,4)

C2 - (9,10),(6,8),(9,5),(8,7),(10,8)

1. **Find mean of each class**

$\mu 1 = (1/N1)\Sigma x$

$$\mu 1 = 1/5\left[\begin{pmatrix}4\\2\end{pmatrix}+\begin{pmatrix}2\\4\end{pmatrix}+\begin{pmatrix}2\\3\end{pmatrix}+\begin{pmatrix}3\\6\end{pmatrix}+\begin{pmatrix}4\\4\end{pmatrix}\right]=\begin{bmatrix}3\\3.8\end{bmatrix}$$

$$\mu 2 = 1/5\left[\begin{pmatrix}9\\10\end{pmatrix}+\begin{pmatrix}6\\8\end{pmatrix}+\begin{pmatrix}9\\5\end{pmatrix}+\begin{pmatrix}8\\7\end{pmatrix}+\begin{pmatrix}10\\8\end{pmatrix}\right]=\begin{bmatrix}8.4\\7.6\end{bmatrix}$$

2. **Find the covariance matrix of S1**

S1 = (1/N-1)$\Sigma_{x\in C1} \quad (x-\mu 1)(x-\mu 1)^T$

$$\text{S1} = \begin{bmatrix}1 & -0.25\\ -0.25 & 2.2\end{bmatrix}.$$

2. **Find the covariance matrix of S2**

S2 = (1/N-1)$\Sigma_{x \in C2} \quad (x - \mu 2)(x - \mu 2)^T$

$$\text{S2} = \begin{bmatrix} 2.3 & -0.05 \\ -0.05 & 3.3 \end{bmatrix}.$$

3. Find the within class scatter matrix S˙w

S˙w = S1+ S2

$$\text{S˙w} = \begin{bmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{bmatrix}.$$

4. Find between class scatter matrix S˙b

S˙b = $(\mu 1 - \mu 2)(\mu 1 - \mu 2)^T$

$$\text{S˙b} = \begin{bmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{bmatrix}.$$

4. Find S˙wˆ-1

$$\text{S˙wˆ-1} = \begin{bmatrix} 0.30454042 & 0.0166113 \\ 0.0166113 & 0.18272425 \end{bmatrix}.$$

5. Find S˙wˆ-1S˙b

$$\text{S˙wˆ-1S˙b} = \begin{bmatrix} 0.9.22126246 & 6.48903654 \\ 4.23388704 & 2.97940199 \end{bmatrix}.$$

Eigen values of above matrix = 12.20066445 and 0

Eigen vector corresponding to largest Eigen value is [0.90878558 0.41726342]

.

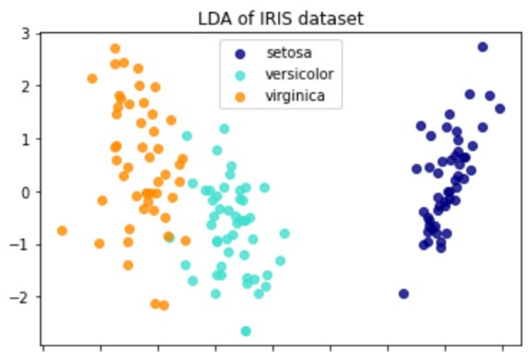
2.2 LDA using sklearn

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

lda = LinearDiscriminantAnalysis(n_components=2)
X_r2 = lda.fit(X, y).transform(X)

colors = ["navy", "turquoise", "darkorange"]
plt.figure()
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(
        X_r2[y == i, 0], X_r2[y == i, 1], alpha=0.8, color=color, label=target_name
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("LDA of IRIS dataset")
plt.show()
```


LDA of IRIS dataset
setosa
versicolor
virginica