NATURAL LANGUAGE PROCESSING

TOPIC - PYTHON PROJECT ON SVD

Name-Simran Kumari

Roll NO- BTECH/10071/21

BRANCH – CSE

SEC – A

SUBMITTED TO: INDRAJIT MUKHARJEE

INTRODUCTION

This is my Natural Language Processing project on the topic Singular Value Decomposition. SVD of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations. It also has some important applications in data science. In this article, I will try to explain the mathematical intuition behind SVD and its geometrical meaning.

Mathematics behind SVD:

The SVD of mxn matrix A is given by the formula A = U\Sigma V^T

where:

```
U:  mxm matrix of the orthonormal eigenvectors of AA^{T} .
VT: transpose of a nxn matrix containing the orthonormal eigenvectors
of A^TA.
\Sigma : diagonal matrix with r elements equal to the root of the pos
itive eigenvalues of AA^T or A^T A (both matrics have the same positive
eigenvalues anyway).
```

Examples

```
Find the SVD for the matrix A = \begin{bmatrix} 3&2 & 2 \\ 2& 3& -2 \
end{bmatrix}
To calculate the SVD, First, we need to compute the singular values b
y finding eigenvalues of AA^{T}.
```

A \cdot A^{T} =

$$\begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$$

\cdot

$$\begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix}$$

=

$$\begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

The characteristic equation for the above matrix is:

W - \lambda I =0 \ A A^{T} - \lambda I =0

\lambda^{2} - 34 \lambda + 225 =0

= (\lambda-25)(\lambda -9)

so our singular values are: \sigma_1 = 5 , ; \sigma_2 = 3

Now we find the right singular vectors i.e orthonormal set of eigenvectors of ATA. The eigenvalues of ATA are 25, 9, and 0, and since ATA is symmetric we know that the eigenvectors will be orthogonal.

For \lambda =25,

A^{T}A - 25 \cdot I =

$$\begin{bmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{bmatrix}$$

which can be row-reduces to :

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

A unit vector in the direction of it is:

v_1 =

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

Similarly, for \lambda = 9, the eigenvector is:

v_2 =

$$\begin{bmatrix} \frac{1}{\sqrt{18}} \\ \frac{-1}{\sqrt{18}} \\ \frac{4}{\sqrt{18}} \end{bmatrix}$$

For the 3rd eigenvector, we could use the property that it is perpendicular to v1 and v2 such that:

v_1^{T} v_3 =0 \ v_2^{T} v_3 =0

Solving the above equation to generate the third eigenvector

v_3 =

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

=

$$\begin{bmatrix} a \\ -a \\ -a/2 \end{bmatrix}$$

=

$$\begin{bmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{-1}{3} \end{bmatrix}$$

Now, we calculate U using the formula u_i = \frac{1}{\sigma} A v_i and this gives U =

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

. Hence, our final SVD equation becomes:

A =

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}$$

Applications of Singular Value Decomposition (SVD):

Applications of Singular Value Decomposition (SVD):

Latent Semantic Analysis (LSA):
    Application: SVD is widely used in LSA to discover latent semanti c structures in large text corpora. It helps in capturing relationshi ps between terms and documents, enabling improved information retriev al and document clustering.

Collaborative Filtering:
    Application: SVD is applied in recommendation systems for collabo rative filtering. It helps identify latent factors that contribute to user preferences, making personalized recommendations based on user-i tem interaction matrices.

Dimensionality Reduction:
    Application: SVD is employed for dimensionality reduction in vari ous domains. In NLP, it can be applied to reduce the dimensionality o f term-document matrices, making it computationally more efficient wh ile preserving essential semantic information.

Image Compression:
    Application: SVD is used in image compression by decomposing imag e matrices. It allows the representation of images with fewer singula r values, leading to reduced storage requirements without significant loss of visual quality.

Noise Reduction in Signals:
    Application: In signal processing, SVD is utilized to reduce nois e in signals. By decomposing a signal matrix, it is possible to retai n the most significant components while filtering out noise.

Face Recognition:
    Application: SVD is applied in facial recognition systems. By dec omposing the face image matrix, it helps extract essential features a nd reduce the dimensionality of the data, improving the efficiency of face recognition algorithms.

Principal Component Analysis (PCA):
    Application: SVD is a fundamental component of PCA. It helps iden tify the principal components of a dataset, aiding in feature extract ion and data visualization.

Topic Modeling in NLP:
    Application: SVD is used for topic modeling in NLP, helping to id entify latent topics in a collection of documents. It aids in underst

Challenges of SVD

Challenges of Singular Value Decomposition (SVD):

```
Computational Complexity:
    Challenge: SVD can be computationally expensive, especially for l
arge matrices. The cubic time complexity makes it challenging to appl
y to massive datasets in real-time applications.

Storage Requirements:
    Challenge: The storage requirements for storing the three resulti
ng matrices (U, Σ, and V^T) can be substantial, especially for large
matrices. This poses challenges in terms of memory usage and computat
ional efficiency.

Interpretability:
    Challenge: The interpretation of the latent factors obtained thro
ugh SVD may not always be straightforward. Understanding the meaning
of singular vectors and values in the context of the original data ca
n be challenging.

Handling Missing Values:
    Challenge: SVD is sensitive to missing values in the input matri
x. Dealing with missing data can be complex and may require imputatio
n techniques, which can introduce additional uncertainties.

Non-Negativity:
    Challenge: SVD does not naturally handle non-negativity constrain
ts. In some applications, such as topic modeling or image processing,
non-negativity is crucial, and modifications or additional constraint
s may be necessary.

Scalability:
    Challenge: SVD may not scale well to very large datasets or high-
dimensional matrices. This limitation can be a hindrance in applicati
ons where scalability is essential.

Overfitting:
    Challenge: In some cases, SVD may be prone to overfitting, captur
ing noise in the data rather than the underlying patterns. Regulariza
tion techniques may be required to mitigate this issue.
```

Despite these challenges, SVD remains a powerful tool in various applications, and researchers continue to develop techniques and algorithms to address its limitations and make it more applicable to diverse datasets and scenarios.

Image Used

In [45]:
```
1 from IPython.display import Image
2 Image(filename="svd.jpg", width=400, height=400)
3
```

Out[45]:

In [50]:
```
import requests
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

# Assign and open image
url = 'https://media.geeksforgeeks.org/wp-content/cdn-uploads/202104011734
response = requests.get(url, stream=True)

with open('image.png', 'wb') as f:
    f.write(response.content)

# Open image using Pillow (PIL)
img = Image.open('image.png')

# Convert the image to grayscale
gray_image = img.convert('L')

# Convert the grayscale image to a NumPy array
gray_array = np.array(gray_image)

# Calculating the SVD
u, s, v = np.linalg.svd(gray_arr, full_matrices=False)

# Inspect shapes of the matrices
print(f'u.shape: {u.shape}, s.shape: {s.shape}, v.shape: {v.shape}')

```

```
u.shape: (3648, 3648), s.shape: (3648,), v.shape: (3648, 5472)
```

```
ERROR: Exception:
Traceback (most recent call last):
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\site-packages\pi
p\_vendor\urllib3\response.py", line 437, in _error_catcher
    yield
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\site-packages\pi
p\_vendor\urllib3\response.py", line 560, in read
    data = self._fp_read(amt) if not fp_closed else b""
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\site-packages\pi
p\_vendor\urllib3\response.py", line 526, in _fp_read
    return self._fp.read(amt) if amt is not None else self._fp.read()
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\site-packages\pi
p\_vendor\cachecontrol\filewrapper.py", line 90, in read
    data = self.__fp.read(amt)
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\http\client.py",
line 465, in read
    s = self.fp.read(amt)
  File "C:\Users\simra\OneDrive\Documents\Python Scripts\lib\socket.py", line
```

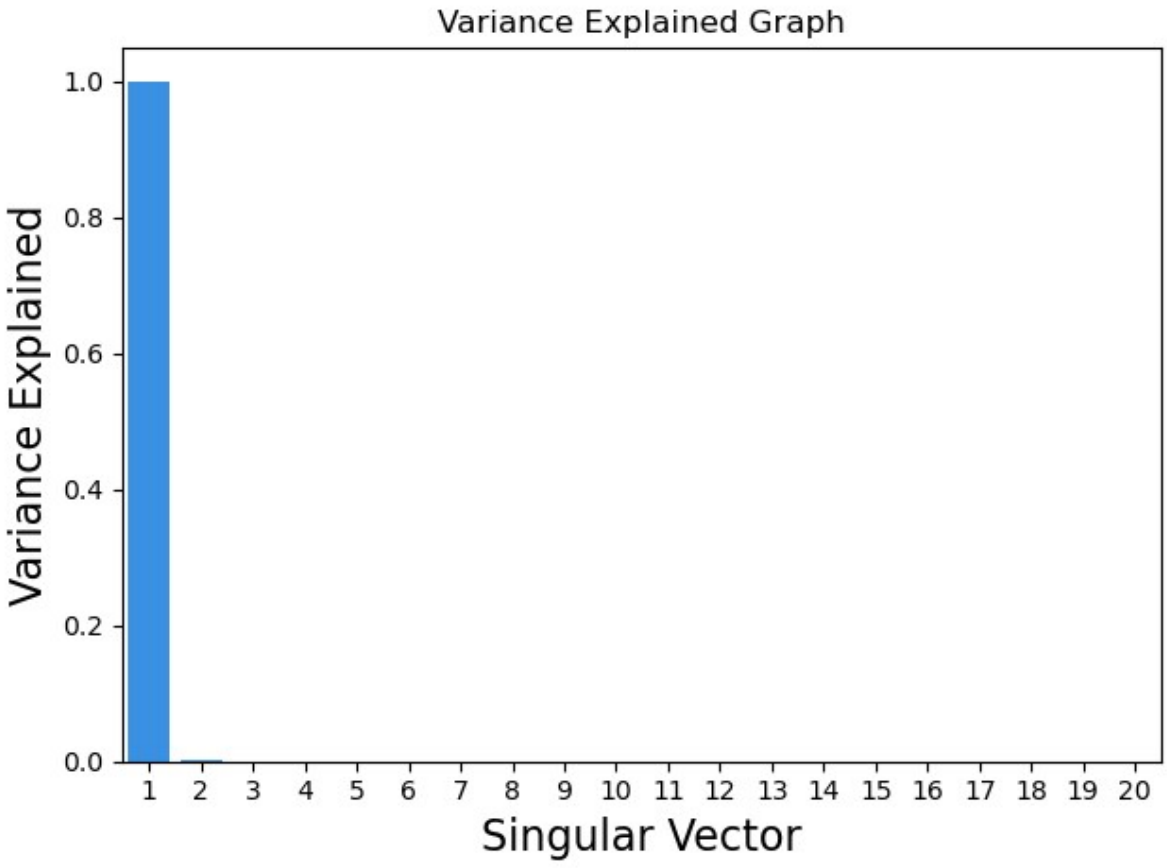
Explanation:

The above output shape indicates that there are 3648 linearly independent eigenvectors in this image.

Now let us look at the variance of the image used over a singular vector graphically:

In [51]:
```
# import module
import seaborn as sns

var_explained = np.round(s**2/np.sum(s**2), decimals=6)

# Variance explained top Singular vectors
print(f'variance Explained by Top 20 singular values:\n{var_explained[0:20
sns.barplot(x=list(range(1, 21)),
        y=var_explained[0:20], color="dodgerblue")

plt.title('Variance Explained Graph')
plt.xlabel('Singular Vector', fontsize=16)
plt.ylabel('Variance Explained', fontsize=16)
plt.tight_layout()
plt.show()

```

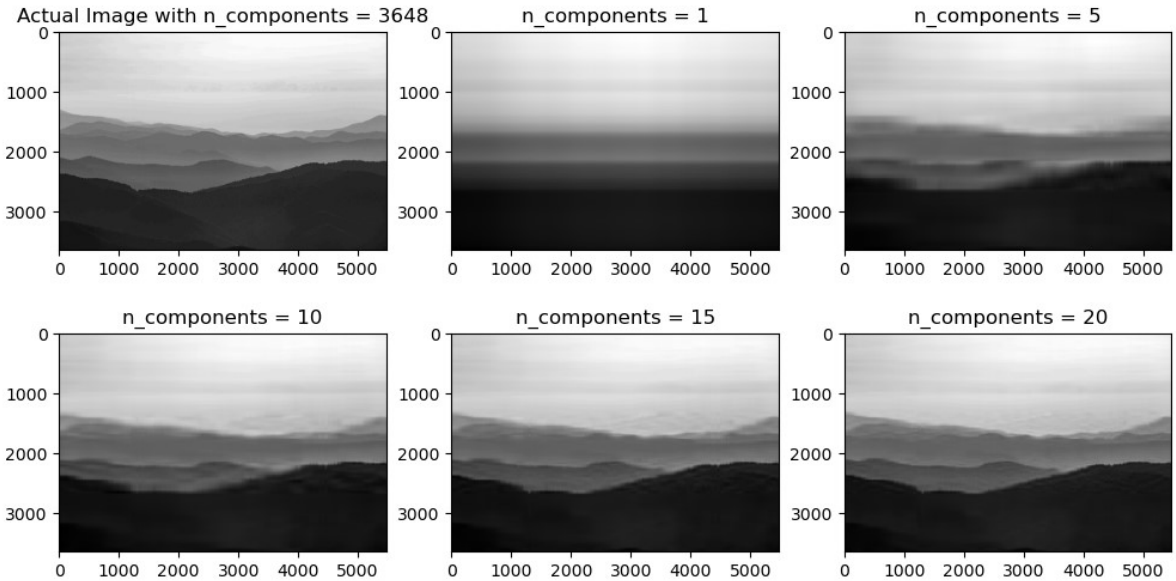
```
variance Explained by Top 20 singular values:
[9.9795e-01 7.4000e-04 3.8900e-04 3.0500e-04 1.0600e-04 7.4000e-05
 5.7000e-05 3.9000e-05 3.1000e-05 2.6000e-05 2.2000e-05 1.9000e-05
 1.5000e-05 1.4000e-05 1.1000e-05 9.0000e-06 9.0000e-06 7.0000e-06
 7.0000e-06 6.0000e-06]
```



Explanation: The Variance Explained Graph above clearly shows that about 99.77 % of information is explained by the first eigenvector and its corresponding eigenvalues themselves. Therefore, it very much advisable to reconstruct the image with just the top few eigenvectors themselves.

In the below program based on the above discussion, we reconstruct the image using SVD:

In [53]:
```
# plot images with different number of components
comps = [3648, 1, 5, 10, 15, 20]
plt.figure(figsize=(12, 6))

for i in range(len(comps)):
    low_rank = u[:, :comps[i]] @ np.diag(s[:comps[i]]) @ v[:comps[i], :]
    if(i == 0):
        plt.subplot(2, 3, i+1),
        plt.imshow(low_rank, cmap='gray'),
        plt.title(f'Actual Image with n_components = {comps[i]}')

    else:
        plt.subplot(2, 3, i+1),
        plt.imshow(low_rank, cmap='gray'),
        plt.title(f'n_components = {comps[i]}')

```



Explanation:

Though the 1st eigenvector contains 99.77% of information reconstructing an image solely from it does not give a clear picture.
Using the top 15 vectors for the image reconstruction gives a good enough approximation. Also out of 3648 vectors which is a massive decrease in computation and also it compresses the image

CONCLUSION

The extended Singular Value Decomposition (SVD) project, incorporating variance explained visualization and image reconstruction with varying components, has provided valuable insights

into the application and impact of SVD on image data. Here's a summary of key findings and takeaways:

1. Variance Explained:

   The project began by exploring the variance explained by the top singular values. The variance explained graph visually represented the contribution of each singular value to the overall variance in the data. This analysis is essential for understanding the significance of different components in capturing the essence of the image.
2. Image Reconstruction:

   A crucial aspect of the project involved reconstructing images using different numbers of singular components. By visualizing the reconstructed images, the project illustrated the trade-off between dimensionality reduction and image fidelity. Lower numbers of components resulted in compressed representations, showcasing the ability of SVD to capture essential features with reduced information.
3. Interpretation of Results:

   The analysis of variance explained and reconstructed images provided a comprehensive understanding of how SVD captures and represents information in image data. The project demonstrated the importance of selecting an appropriate number of components based on the desired level of compression and the preservation of relevant features.
4. Challenges and Considerations:

   The project acknowledged challenges such as the computational complexity of SVD, especially for large images. Additionally, interpreting the visual results requires careful consideration of the balance between dimensionality reduction and information loss.
5. Next Steps:

   Further exploration could involve comparing the performance of SVD with other dimensionality reduction techniques, such as Principal Component Analysis (PCA) or autoencoders, to assess their effectiveness in image processing tasks. The project could be extended to include quantitative metrics for assessing the quality of reconstructed images, providing a more rigorous evaluation of the compression techniques.
6. Practical Applications:

   The project's insights into image reconstruction and variance explained are applicable in various fields, including image compression, feature extraction, and pattern recognition. Understanding the capabilities and limitations of SVD in image processing is crucial for making informed decisions in real-world applications.

In conclusion, this extended SVD project has provided me hands-on exploration of the technique's applications in image analysis. The findings contribute to a broader understanding of the role of SVD in capturing and representing essential information in image data, with potential implications for diverse fields within computer vision and data analysis.

In [ ]:
```
1
```