Sharif University of Technology
Electrical Engineering Department

Convex Optimization
CHW 1

Amir Hossein Yari
99102507

April 14, 2023

Contents

1. Eigenvalues and Eigenvectors

1. Explanation of the algorithm

We know:

- A matrix A can be decomposed like: $A = QR$, where R is an upper triangular matrix, and Q is an orthonormal matrix.
- Because Q is orthonormal: $Q^T = Q^{-1}$
- We define two matrices A and B as being similar if there exists a non-singular matrix P such that: $B = P^{-1}AP$
- Two similar matrices A and B have the same eigenvalues.
- Eigenvalues of the upper triangular matrix are the diagonal entries of the matrix.

Now, there's a type of factorization called Schur Factorization that says that A can be written as: $A = QUQ^H = QUQ^{-1}$, where Q is a unitary matrix and U is an upper triangular matrix. Additionally, every square matrix A has a Schur Factorization.
So A and U similar $\Rightarrow$ They have the same eigenvalues.
$A_k = Q_k R_k \Rightarrow Q_k^{-1} A_k = Q_k^{-1} Q_k R_k = R_k \Rightarrow Q_k^{-1} A_k Q_k = R_k Q_k$
k is the iteration index.
By repeating the steps the matrix A converges to an upper triangular matrix.
Also the eigenvectors is Q of the QR decomposition of $A\times$ the eigenvectors of the previous step.

2. Python code of algorithm

```
# add required packages
import numpy as np

# function that returns eigenvalues
def QR_eigenvalues(A, max_iterations=1000, tolerance=1e-10):
n = A.shape[0]
B = A.copy()
Q, R = np.linalg.qr(A)
eigenvector = np.eye(n)
for i in range(max_iterations):
```

```
A = np.dot(R, Q)
Q, R = np.linalg.qr(A)
eigenvalues = np.diag(A)
Q1, R1 = np.linalg.qr(B @ eigenvector)
eigenvector = Q1
if np.allclose(np.triu(A, 1), np.zeros((n, n)), atol=tolerance):
break
#sorting eigenvalues and eigenvector
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvector = eigenvector[:,idx]
return eigenvalues, eigenvector


# Example usage
n = 5
A = np.random.rand(n, n)
A = (A + A.T)/2
MyEigenvalues, MyEigvector = QR_eigenvalues(A)
print("eigenvalues(my function) :")
print(MyEigenvalues)
print("eigenvector(my function) :")
print(MyEigvector)
# We compare our results with the official numpy algorithm
eigval, eigvec = np.linalg.eig(A)
#sorting eigenvalues and eigenvector
idx = eigval.argsort()[::-1]
eigval = eigval[idx]
eigvec = eigvec[:,idx]
print("eigenvalues(python function) :")
print(eigval)
print("eigenvector(python function) :")
print(eigvec)
```

3. Algorithm limitations

One limitation of the algorithm that finds eigenvalues by QR decomposition is that it may not converge for certain matrices. In particular, for matrices that are close to being singular or have multiple eigenvalues with small separations, the algorithm may fail to converge or produce inaccurate results. Additionally, the algorithm can be computationally expensive for large matrices due to the need for repeated QR decomposition.
Another limitation is that matrix must be real and symmetric in this algorithm.

2. SVD and Image Processing

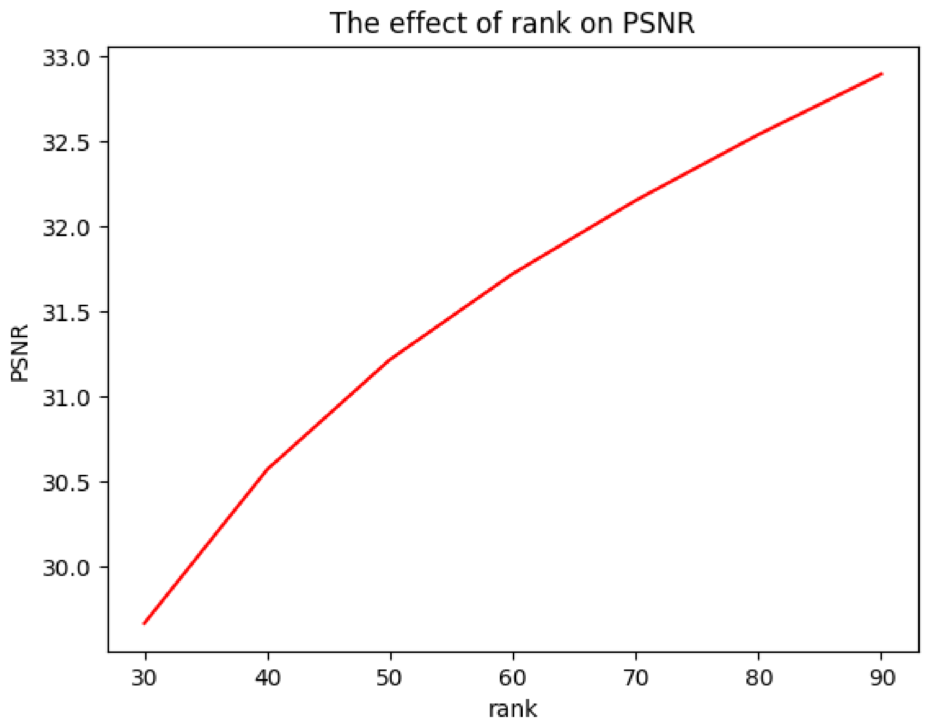
1. Image compression

PSNR stands for Peak Signal-to-Noise Ratio. It is a metric used to evaluate the quality of a reconstructed or compressed image or video. It measures the ratio of the maximum possible power of a signal (the original image) to the power of the noise that affects the fidelity of the reconstructed image. In other words, it measures how much the reconstructed image or video differs from the original signal in terms of noise (error) introduced during compression or transmission. The higher the PSNR value, the better the reconstructed image or video quality.

$$PSNR = 20log(\frac{L-1}{\sqrt{MSE}})$$

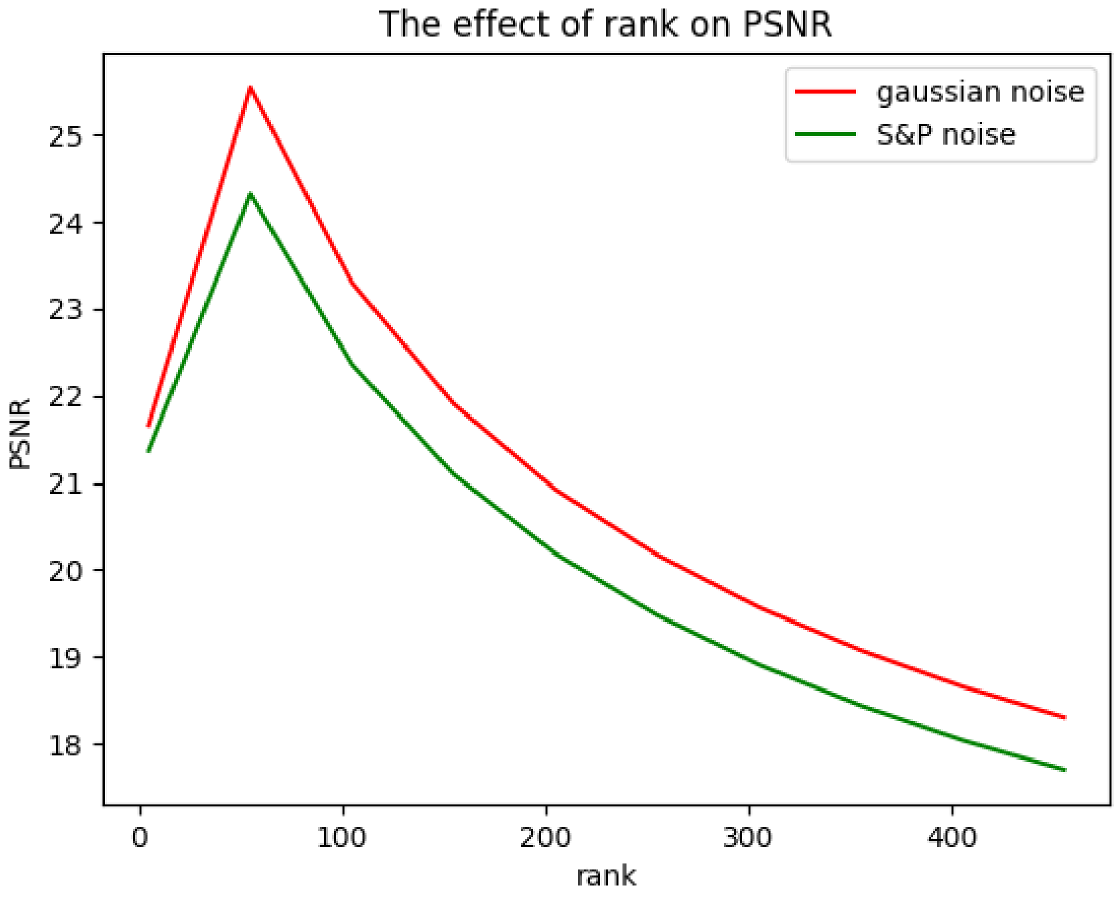
$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(Orginal(i,j) - reconstruct(i,j))^2$$

As can be seen from the results, by increase rank, the components of the compressed image become closer to the original image and PSNR increases.



2. Delete noise

As you can see in the figure below, for both the noise by increasing the rank, PSNR increases and then decreases. The drawn outputs are also a confirmation of this statement.
According to the figure, PSNR of Gaussian noise is more than salt and pepper noise, and this method is better for eliminating Gaussian noise than salt and pepper noise.



3. Dimensional reduction of data with PCA method

The PCA method involves the following steps:

1. Standardize the data: For PCA to work correctly, the data must be standardized. This means that each variable should have zero mean.
2. Compute the covariance matrix: The covariance matrix is used to determine the relationships between the different variables in the dataset.

3. Compute the eigenvectors and eigenvalues: The eigenvectors and eigenvalues of the covariance matrix are calculated by SVD(covariance matrix is PSD so eigenvalues and eigenvectors have relation with U , Σ , V). The eigenvectors represent the principal components of the data, while the eigenvalues represent the amount of variance explained by each principal component.
4. Determine the number of principal components: The number of principal components to retain is determined. This can be done using a scree plot or by selecting a threshold value for the eigenvalues.
5. Project the data into the new feature space: Finally, the original data is projected onto the new feature space defined by the retained principal components.

1. What does the covariance matrix of the data set have to do with $\tilde{\mathbf{X}}$ and more generally with the output parameters of the aforementioned analysis?

covariance matrix of the data set and $\tilde{\mathbf{X}}$ are equal because $Cov(X - a) = Cov(X)$. Output parameters of SVD of covariance matrix is eigenvalues and eigenvector of this matrix and $Covariance\ Matrix = U\Sigma U^T$

2. According to the given definitions, say why the claim is true and In a more general case, what do the three matrices U, Σ, V mean in this problem?

U is a base for principle component in dimensional reduction of data with PCA method because they represent the directions with the highest amount of variation in the data.
The first principal component corresponds to the direction of maximum variance in the data. The subsequent principal components are orthogonal to each other and follow the direction of maximum variance in the remaining data.
The eigenvectors of the covariance matrix are used to calculate the principal components because they represent the directions of maximum covariance. As a result, eigenvectors become the basis for representing the principal components.
columns of U is eigenvectors of covariance matrix.
diag of Σ is eigenvalues of covariance matrix.
columns of V is eigenvectors of covariance matrix.(covariance matrix(A) is PSD and the columns of U and V are eigenvectors of AA^T and A^TA, so $U = V$)

3. Having three matrices U, Σ, V, propose a method to convert the dimension of the problem into a specific dimension such as l so that $l < n$.

According to the above algorithm First, we select the vector of eigenvalues corresponding to the l largest eigenvalue. After that the original data is projected onto the new feature space defined by the retained principal components through dot of standard data and ordered eigenvectors.

4. Introduce and explain another algorithm for this task based on linear algebra methods.

Independent component analysis (ICA) is a popular algorithm used for the dimensional reduction of data. It is a statistical technique that aims to separate multivariate signals into independent, non-Gaussian components, based on the assumption that the signals are generated by linear mixtures of independent sources. ICA is useful for reducing the dimensionality of high-dimensional data, where the number of observations is much larger than the number of features. By extracting independent source components from this data, ICA can help to identify the underlying structure of the data and reduce the data into a more manageable form. The basic steps involved in performing ICA for data dimensional reduction are as follows:

1. Preprocessing the data: The first step is to preprocess the data, which involves cleaning, normalization, and standardization. This step helps to ensure that the data is ready for analysis.

2. Choosing the number of independent components: The next step is to choose the number of independent components for the analysis. This can be done using various techniques such as the Kaiser criterion, the scree plot method, or cross-validation.

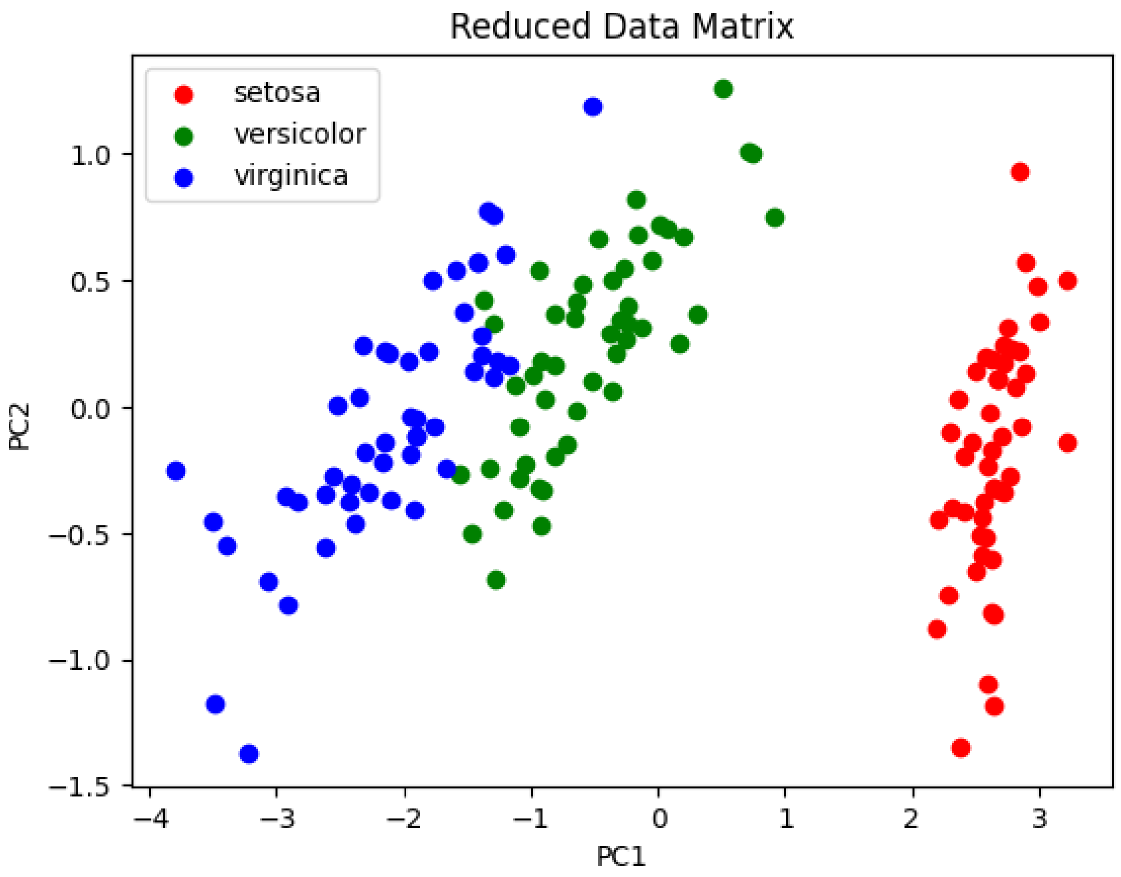
3. Applying the ICA algorithm: Once the number of independent components has been determined, the ICA algorithm is applied to the data. The algorithm works by separating out the independent components from the data by minimizing the mutual information between them.

4. Visualizing the results: After applying the ICA algorithm, the independent components are obtained. These components can be visualized using various techniques such as scatter plots or heat maps.

5. Reducing the dimensionality: Finally, the dimensionality of the data is reduced by selecting only the relevant independent components that capture the most important information. This step helps to reduce noise and improve the performance of subsequent analyses.

5. Using the given explanation, implement the PCA algorithm code using the numpy library. Then draw the graph of this data set in the obtained space so that the color of each point is the type of flower. Then say if this graph corresponds to the result you expected?

Yes; As you can see, all three types of flowers can be distinguished by two components.


Reduced Data Matrix
setosa
versicolor
virginica
PC2
PC1