

MLPR Lab 3

Apply Principal Component Analysis (PCA) on the given image. **Do not use** inbuilt python library (`sklearn.decomposition.PCA()`) to perform PCA. Instead use NumPy functions to calculate PCA.

[How to Calculate Principal Component Analysis \(PCA\) from Scratch in Python - MachineLearningMastery.com](https://machinelearningmastery.com/how-to-calculate-principal-component-analysis-pca-from-scratch-in-python/)

Instructions:

Step 1: Import libraries

- OpenCV
- Matplotlib
- NumPy

Step 2: Load the given image and read it using OpenCV.

Step 3: Convert the image to grayscale.

Step 4: Convert the image to double for performing the mathematical operations easily.

- Use *`image.astype(np.float64)`*

Step 5: Compute the mean of each column (pixels) and subtract it from the image.

- Use *`np.mean()`* column wise and then subtract *`mean_column`* from the image to get the *`image_mean_subtracted`*.

Step 6: Compute the covariance matrix.

- Use *`np.cov()`* [numpy.cov — NumPy v1.25 Manual](https://numpy.org/doc/stable/reference/generated/numpy.cov.html)

Step 7: Get eigen values and eigen vectors.

- Use *`np.linalg.eig()`* [numpy.linalg.eig — NumPy v1.25 Manual](https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html)

Step 8: Sort eigen vectors by eigen values.

- *`np.argsort()`* [numpy.argsort — NumPy v1.25 Manual](https://numpy.org/doc/stable/reference/generated/numpy.argsort.html)

Step 9: Define the number of principal components to keep.

- **`Num_components = [10,20,30,40,50,60,91]`**, Adjust it check the variations

Step 10: for each components reconstruct the image and display output image.

`Output_images = []`

for each **`num_components`**

- Take N number of components and extract eigen vectors.
- Project the data onto the selected components.
 - **`np.dot(selected_components.T, image_mean_subtracted.T).T`**
[numpy.dot — NumPy v1.25 Manual](#)
- Reconstruct the image.
 - **`np.dot(selected_components, projected_data.T).T + mean_column`**
- Add the reconstructed image to the list.
 - Append reconstructed images to the **`Output_images[]`**

Step 11: Display the results.

- **`Plt.figure(define figure size)`**
- Provide a title to an image “Dimensionality Reduction using PCA.”
- Use for loop to define all the displaying parameters for images in **`Output_images`** and display it.

Step 12: Now using PCA function see how the dim 91 explain the 95% variance in data.

- Use from **`sklearn.decomposition import PCA`**
- **`pca = PCA(num_components = num_components)`**
- Use `pca.explained_variance_ratio_` to check the explained variance using **`num_components`**

Report:

Answer the following questions within your report:

Q1: What is the difference between PCA and Feature Selection?

Q2: Why do we standardize features before applying PCA?

Q3: What is the importance of Covariance Matrix in PCA?

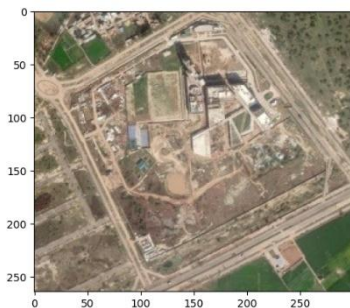
Q4: What are some limitations of PCA?

Q5: How can we figure out the importance of individual PCs in PCA?

Submission Instructions:

- Upload the main code file with the code, Grayscale image, PCA with all N components and the report in one file in PDF format.
- Your file name should be yourname_lab3.pdf. Upload it before the due time.

Output Images reference:



Grayscale Image (Satellite View of Plaksha University)



Dimensionality Reduction using PCA

