Principal Component Analysis and its Application to Nominal Gross Domestic Product

Suchitra Hiregowda\textsuperscript{1,4}, GH Pujar\textsuperscript{2,4*} and VS Betageri\textsuperscript{3,4}

\textsuperscript{1}Department of Computer Science & Engineering, GM Institute of Technology, Davangere, Karnataka, India.
\textsuperscript{2}Research Centre, Department of Physics, GM Institute of Technology, Davangere, Karnataka, India.
\textsuperscript{3}Research Centre, Department of Chemistry, GM Institute of Technology, Davangere, Karnataka, India.
\textsuperscript{4}Visvesvaraya Technological University, Belagavi, Karnataka, India
\textsuperscript{1}suchitra_18cs065@gmit.ac.in, \textsuperscript{2}puttpuja@gmail.com, \textsuperscript{3}virupaxb@gmail.com

Abstract. Gross Domestic Product refers to financial or market value of all goods and services which are produced within the countries in a specific time duration. As expansive measure of universal domestic production, it functions as a comprehensive record of the country’s economic health. Nominal Gross Domestic Product is the computation of raw data and is more utilitarian comparing national economies on the international market. Some countries have very differing Gross Domestic Product per capita between its regions. A countries region tend to intersect overtime, the discrepancy between the poorer and richer regions is kept over decades in other cases. Here, we considered progress in the nominal Gross Domestic Product per capita across 19 regions in Spain to provide the analysis of progress. This could answer few questions with respect to Nominal Gross Domestic Product viz., Have the regions converged? Which is the spread between regions? The central theme of this article is to answer whether we can make a cluster analysis of the regions after applying principal component analysis?

Keywords. Cluster Analysis, GDP, Nominal GDP, PCA, Spanish regions.

1. Introduction

GDP calculation gives an overall record of the state of the economy to that of a satellite in the space which helps in surveying the weather across an entire continent. This record also provides the policy makers and the central banks to judge if the economy is contracting or expanding and also if the needs required to be boosted or restrained. It helps to understand if there are any threats like recession or rampant inflation loom on the horizon.

Generally, the dataset can be used to calculate GDP which has been compiled by the INE (Spanish institute of statistics) and published on Kaggle - Nominal GDP per capita of Spam. This dataset contains information on the Nominal GDP per unit of population covering the 19 Spanish zones (autonomous
communities and cities) from 2000 to 2016. To answer the questions that have been highlighted in the Abstract and to figure out whether there has been a convergence, we need to calculate whether the nominal GDP has been steady by looking at the spread between the maximum and the minimum values of the nominal GDP per unit of population covering the Spanish zones. But our true goal is to perform a cluster analysis pertaining the 19 nominal GDP per capita across the 19 regions/zones of Spain. So as to perform cluster analysis we first need to select the appropriate features to visualize the different cluster classes and their effects on the Nominal GDP. Since our data set consists of 18 columns (here dimensions), we will have to perform PCA for feature extraction before the visualization [1-6].

PCA is a powerful statistical technique in Linear Algebra that will reduce the proportions of the data and assist us to understand, organize the data with a minor proportion as compared to the original data. The chief constituents are primarily the vectors that are linearly unrelated and have variation within the data. From the fundamental constituents, the top p picked out which has the most difference. In PCA our goal is to separate the data by illustrating a line or a plane between data by puzzling out the dimension which has the highest variance [7-15].

2. Background of PCA Module

PCA uses the Rotation tools to analyze the multiband datasets. There is a high correlation in the Data bands due to their analogous spectral domains. It could remove superfluous spectral information using multiband datasets. Remote sensing application uses this PCA to create a smaller dataset from multiple bands by retaining original spectral information as much as possible [14, 15].

2.1. Linear Algebra Background Theorem

**Theorem 1.** If the matrix A is symmetric (meaning \( A^T = A \)), then A is orthogonally diagonalizable and has only real eigenvalues. In other words, there exist real numbers \( \lambda_1, \lambda_2, ..., \lambda_n \) (the eigenvalues) and orthogonal, non-zero real vectors \( v_1, v_2, v_3, ..., v_n \) (the eigenvectors) such that for each \( i = 1, 2, ..., n \)

\[
A v_i = \lambda_i v_i
\]

(1)

2.2. Implementation

We have implemented PCA on the basis the minimization of the reconstruction error. The Frobenius standard of dissimilarity b/w original & reconstructed matrix is called as the reconstruction error.

Particularly, among all the feasible linear projections from dimensions’ \( n \) to \( m \), taking the first \( k \)-components of PCA transformation of \( X \) reduces the reconstruction error.

2.3. Proof

The input to the problem is \( m \) data points of the form \( x_i = (x_1^i, x_2^i, ..., x_n^i) \), \( i = (1, 2, .. m) \). The aim of the PCA algorithm is to transform this \( n \)-dimensional feature vector to a \( k \)-dimensional space with an orthonormal basis \( u_1, u_2, ..., u_k \). The new set of transformed co-ordinates in the \( k \)-dimensional space IS represented as \( z_1, z_2, ..., z_k \) where \( z_i = (x - \bar{x}) \). The mean of \( x \) over all the data points is \( u_i \) and \( \bar{x} \). From the new set of co-ordinate axes, we attempt to reconstruct the data \( X \). The reconstructed data example \( \hat{x} \) is given as

\[
\hat{x} = \bar{x} + \sum_{j=1}^{n} z_j^i u_i
\]

(2)

The objective of PCA IS to minimize the error created by this reconstruction, given by

\[
\text{error}_m = \sum_{j=1}^{n} (x^l - x^j)^2
\]

(3)

Substituting the above values in the error equation, we get the following result.

\[
\text{error}_k = \frac{1}{m} \sum_{j=1}^{n} (x^l - \bar{x})(x^l - \bar{x})^T
\]

(4)

where, \( \sum = \frac{1}{m} \sum_{j=1}^{n} (x^l - \bar{x})(x^l - \bar{x})^T \) (5)
3. Algorithm to Perform PCA

To implement PCA, it is required to get Singular Value Decomposition of matrix $A$ that results the eigenvalues & eigenvectors of $A^T A$ immediately.

**Part 1: Function to return the orthogonal vector**

```python
def householder_reflection(a, e):
    assert a.ndim == 1
    assert np.allclose(1, np.sum(e**2))
    u = a - np.sign(a[0]) * np.linalg.norm(a) * e
    v = u / np.linalg.norm(u)
    H = np.eye(len(a)) - 2 * np.outer(v, v)
    return H
```

The Householder reflection is an alternate method of orthogonal transformation that transforms a vector $x$ into a unit vector $y$ parallel to $x$. The Householder reflection matrix $H$ with a normal vector $v$ takes the form:

$$H = 1 - 2vv^T$$ (6)

A matrix $H$ must be constructed such that $Hx = \propto e_1$ for a constant $\propto$ and $e_1 = [1 0 0]^T$. Since the matrix $H$ is orthogonal, $|Hx| = |x|$ and $|\propto e_1| = |\propto x|$ and therefore $\propto = \pm |x|$

The symbol is so chosen that it has the contrasting symbol of $x_1$, hence the vector $u$ is

$$u = \begin{bmatrix} x_1 + \text{sign}(x_1)|x_1| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$ (7)

With the unit vector $v$ interpreted as $\frac{y}{|v|}$, the corresponding Householder reflection matrix is

$$H = 1 - 2vv^T = 1 - 2u u^T$$ (8)

**Part 2: Function to return the QR Decomposition**

```python
def qr_decomposition(A):
    n, m = A.shape
    assert n >= m
    Q = np.eye(n)
    R = A.copy()
    for i in range(m - int(n==m)):
        r = R[i:, i]
        if np.allclose(r[1:], 0):
            continue
        # e is the i-th basis vector of the minor matrix.
        e = np.zeros(n-i)
        e[0] = 1
        H = np.eye(n)
        H[i:, i:] = householder_reflection(r, e)
        Q = Q @ H.T
        R = H @ R
    return Q, R
```

The Householder reflection technique of QR decomposition is done by finding appropriate $H$ matrices and multiplying them from the left by the actual matrix $A$ to construct the upper triangular matrix $R$. As we have seen earlier, unlike the GramSchmidt policy, the Householder reflection approach doesn’t accurately form the matrix $Q$. Although, the matrix $Q$ can be found by taking the dot product of each consecutively formed a Householder matrix.

$$Q = H1H2...Hm2Hm - 1$$ (9)
Part 3: Function to return Eigen decomposition of matrix A

def eigen_decomposition(A, max_iter=100):
    A_k = A
    Q_k = np.eye(A.shape[1])
    for k in range(max_iter):
        Q, R = qr_decomposition(A_k)
        Q_k = Q_k @ Q
        A_k = R @ Q
    eigenvalues = np.diag(A_k)
    eigenvectors = Q_k
    return eigenvalues, eigenvectors

This algorithm is Iterative: In every step, we have to calculate $A_{k+1}$ by considering the QR decomposition of $A_k$, reversing the order of Q & R, and multiplication of matrices with each other. The off-diagonals become meagre and at the end, we return the respective eigenvalues and eigenvectors.

Part 4: Implementation of PCA Algorithm

class PCA:
    def __init__(self, n_components=None):
        self.n_components = n_components
    def fit(self, X):
        n, m = X.shape
        # subtract off the mean to center the data.
        self.mu = X.mean(axis=0)
        X = X - self.mu
        # Eigen Decomposition of the covariance matrix
        C = X.T @ X / (n-1)
        self.eigenvalues, self.eigenvectors = eigen_decomposition(C)
        # truncate the number of components
        if self.n_components is not None:
            self.eigenvalues = self.eigenvalues[0:self.n_components]
            self.eigenvectors = self.eigenvectors[:, 0:self.n_components]
            descending_order = np.flip(np.argsort(self.eigenvalues))
            self.eigenvalues = self.eigenvalues[descending_order]
            self.eigenvectors = self.eigenvectors[:, descending_order]
        return self

    def transform(self, X):
        X = X - self.mu
        return X @ self.eigenvectors

    def proportion_variance_explained(self):
        return self.eigenvalues / np.sum(self.eigenvalues)

The following steps are followed before we implement PCA In the idealized form in the fit() function.
1. Making sure that the data is centered
2. Optionally we "whiten" the data such that each attribute has unit variance.
3. Keeping Eigenvalues in declining order
This is done by subtracting off the mean to center the data and figuring out the Eigen covariance matrix. Since we are performing dimensionality reduction we also truncate the number of components and finally sort the eigenvalues in the descending order to return them.

4. Results and Discussions
First performing the PCA for feature extraction is needed to reduce the dimensionality for cluster analysis. This part is implemented in the PCA module. Using the PCA module, the dataset was transformed to generate the first 3 components of PCA. Using this transformed data, the cluster analysis was performed with the K-means clustering.

Algorithm:
```python
cpa = PCA(n_components=3)
cpa.fit(x)
x_prime = cpa.transform(x)
x_prime = pd.DataFrame(x_prime)
```

**Front-end implementation of PCA**

K indicates that algorithm is a iterative one which separates dataset into K pre-defined definite diverging clusters in which every data is owned by only one cluster. Here, k value is 3. It makes the intracluster data points which are as analogous as feasible and parallelly considering the groups also as different or as far as possible. It allocates data points to a group so that total sum of (distance)$^2$ between data points and groups centroid (AM of all data points belonging to group) is at minima. The minimum variation is observed within clusters (see Figure 1).

```python
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit(x_prime)
```

**K-Means implementation**

![Figure 1. Similarity proportions between the regions of Spain.](image)
Visualizations:
Our dataset contains the compilation of the nominal GDP per unit of population for each zone/autonomous entity of Spain and the national average (in € / from 2000 to 2016). We first analyze the territorialized evolution of the nominal GDP per unit of population in Spain, Identify the economic circles, the spread between regions, and examine if there has been a convergence (in GDP per capita) between regions from 2000 to 2016 (Figure 2).

So we create a BOXPLOT to visualize the distribution of the nominal GDP per capita between the Spanish regions (time period: 2000-2016)

![Figure 2. Boxplots, Spanish regions distribution of the nominal GDP per capita.](image)

And we created a cloud PLOT to visualize the trend and spread of the regions as shown in Figure 3.

![Figure 3. Nominal GDP per capita trend and differences across Spanish regions.](image)
Both plots emanate a relatively high spread in the nominal GDP per unit of population between the sectors. To generate a visualization/examination of the divergence between regions we can concatenate the series the percentage difference of both the maximum and the minimum values by creating a new data frame. We can infer from the plots that the series is regular. The plot that describes the MIN and MAX nominal GDP per Capita across Spanish regions is shown in Figure 4.

![Figure 4. MIN and MAX values of the nominal GDP per unit of population in the Spanish sectors.](image)

5. Conclusions
During the last 16 years, the spread between the maximum and the minimum values in the nominal GDP per capita over all the Spanish regions has remained very steady. The differences in 2016 are almost the same as those of the year 2000. We can conclude that in spite of the efforts by the Spanish government to redistribute the country’s wealth across its regions; there has not been a regional convergence in phrase of nominal GDP per unit of population. From the Cluster Analysis and the Plots, we can distinguish 3 economic circles (3 classes from the clusters)

1. From 2000 to 2008: a period of continuous growth of the nominal GDP per unit of population in all the regions.
2. beginning of 2008 to 2013: global recession started a strong downturn in all regions.
3. From 2013 to 2016: the start of a partial recovery of the nominal GDP per capita In all regions

PCA refers to a statistical approach which uses the orthogonal transformation to transfigure a set of observations of possibly corresponding variables into a set of values of linearly non-corresponding variables. Implementing neural networks with a layer of PCA would help us to optimize the model. There will be fewer chances of over fitting and the visualizations would be Improved. The main advantage of PCA is to optimize the data and that would be helpful in any algorithm as it eradicates correlated features.
References

[1] Jolliffe IT and Cadima J. 2016 Principal component analysis: a review and recent developments Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374 20150202.

[2] Li Y, Wang N and Carroll RJ 2013 Selecting the number of principal components in functional data Journal of the American Statistical Association 108 1284-94

[3] Sara Noori Mohammad Ali 2016 Risk Factor of Heart Attack in Sulaimany City IOSR Journal of Nursing and Health Science 5 10-13

[4] Vinod HD 2011 Hands-on Matrix Algebra Using R: Active and Motivated Learning with Applications World scientific publishing Co. Pte. Ltd, Singapore

[5] Lolliffe IT 2002 Principal Component analysis, Second Edition Springer Verlag, New York, USA, (Chapter 3 - Properties of Sample Principal Components 29 - 59

[6] Vander Plas J 2016 Python Data Science Handbook First Edition O'Reilly, Sebastopol, USA, (Chapter - Machine Learning - In Depth: Principal Component Analysis 433 - 445

[7] Papi M and Caracciolo G 2018 Principal component analysis of personalized biomolecular corona data for early disease detection Nano Today 21 14-7.

[8] Polo E, Collado M, Pelaz B and del Pino P 2017 Advances toward more efficient targeted delivery of nanoparticles in vivo: understanding interactions between nanoparticles and cells ACS nano 11 2397-402.

[9] Boselli L, Polo E, Castagnola V and Dawson KA 2017 Regimes of biomolecular ultrasmall nanoparticle interactions Angewandte Chemie 129 4279-82

[10] Caracciolo G, Farokhzad OC and Mahmoudi M 2017 Biological identity of nanoparticles in vivo: clinical implications of the protein corona Trends in biotechnology 35 257-64

[11] Westmeier D, Chen C, Stauber RH and Docter D 2015 The bio-corona and its impact on nanomaterial toxicity European Journal of Nanomedicine 7 153-68

[12] Mahmoudi M, Bertrand N, Zope H and Farokhzad OC 2016 Emerging understanding of the protein corona at the nano-bio interfaces Nano Today 11 817-32

[13] Mirshafiee V, Kim R, Park S, Mahmoudi M and Kraft ML 2016 Impact of protein pre-coating on the protein corona composition and nanoparticle cellular uptake Biomaterials 75 295-304

[14] Palchetti S, DiGiacomo L, Pozzi D, Peruzzi G, Micarelli E, Mahmoudi M and Caracciolo G 2016 Nanoparticles-cell association predicted by protein corona fingerprints Nanoscale 8 12755-63

[15] Schöttler S, Becker G, Winzen S, Steinbach T, Mohr K, Landfeiter K, Mailänder V, Wurm FR, 2016 Protein adsorption is required for stealth effect of poly(ethylene glycol)- and poly(phosphoester)-coated nanocarriers Nat. Nanotechnol. 11 372-377.