SC505 Linear Algebra and Optimization
Autumn 2020
Project Title: Principle Component Analysis and its Variants
Date: December 22, 2020

Manali Bhavsar (202011022) Darshil Patel (202011034) Tarang Ranpara (202011057)

1 Introduction

For our machine learning model to perform well, we add more and more features to our data. However, after some point, our model's performance decreases with the increase in its number of features, called the "*Curse of dimensionality*". The curse of dimensionality is when we add new features to our data without adding new data samples. The dimensionality of feature space grows to become sparser, which leads our model to overfit on the dataset because it becomes easy for it to find a "*perfect*" solution. "*Overfitting*" is when our model corresponds too closely to the given dataset and fails to generalize well, which leads it to work too well on the given dataset but fail on unseen data.

To overcome this curse of dimensionality, an approach can be to perform "*Dimensionality Reduction*" on our data. Dimensionality reduction, as its name suggests, helps us reduce the number of features in our dataset. There are two popular approaches for it, "*Feature Extraction*" and "*Feature Selection*".

The process of "*Feature selection*" refers to selecting the subset of original features and eliminating redundant and irrelevant ones. On the contrary, the process of "*Feature Extraction*" projects our data in an entirely new coordinate system having fewer features that are non-redundant and informative. Dimensionality Reduction is also useful in data-visualization because higher dimensional data is not often much intuitive to understand than its 2D-3D counterpart.

"*Principal Component Analysis*" abbreviated as "PCA" is a type of unsupervised feature extraction technique/algorithm which was introduced by Pearson (1901). The core idea of PCA is to reduce the dimensionality of a dataset containing interrelated variables while retaining as much possible variance present in it. PCA transforms our dataset into a new set of variables called principal components, which are uncorrelated and ordered (according to variance, they can individually explain). It is to be noted that PCA does not retain variable names as it is transforming data into new feature-space.

This Report is mainly divided into five parts: Introduction(above), Related work, The core algorithm, Implementation, Variants of PCA and Conclusion.

2 Related work

As PCA has been in use for a long time, there are many books and book chapters dedicated to it and its variants. PCA uses many Linear algebra concepts like eigenvalue- eigenvectors, vector projection, covariance matrix, and method of SVD, which are explained by Gilbert Strang in his book [5] . To understand the overall idea of PCA and its working[2] and implementations[3] Sebastian Raschka's blog[2] can be beneficial.

PCA and its variant called "Robust PCA", are explained by I.T. Jolliffe in his book[1].This book gives understanding about PCA from basics. It contains derivation of principal components properties of principal components, graphical representation of data and usecases of PCA.In Robust PCA via Outlier Pursuit[6], authors explain how convex optimization can help us perform Robust PCA on a data that is noisy and contains outliers.

A tutorial on Principal Component Analysis by J.Shlens [4] gives idea about the mathematics behind PCA and focuses on building a solid intuition for how and why principal component analysis works.

3 Discussion of paper you have read

3.1 Derivation of Principle Components

3.1.1 Projection of Datapoints

We want to convert 2D points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ to 1D point. So, we have to project this 2D points on 1D line. Let's say these points are $p_1, p_2, \ldots, p_n$ Here, variance is the information about the distribution of the data. Variance can be find using the following equations:

- Variance along with X-axis is:

$$\sigma_x^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$

- Variance along with Y-axis is:

$$\sigma_y^2 = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{n}$$

If we project data on X-axis we loss variance of Y-axis σ_y^2 and if we project data on Y-axis we loss variance of X-axis σ_x^2.

So, while projecting the 2D points into 1D line we have to take care that we don't loose the information in other words we keep variance as more as possible.

Is there any other way that we can keep variance more than σ_x^2 or σ_y^2 ?

As shown in the diagram we can fit best line for these points such that the variance of projection of all 2D points on this line U becomes maximum.

Let's take unit vector $\vec{u}$ in the direction of the line U and points are p_1 to p_n.

Projection of $\vec{p}_i$ on $\vec{u}$ is

$$Proj_u \vec{p}_i = \frac{p_i, \vec{u}}{\|u\|} \cdot \vec{u} = \vec{q}_i$$

$$\vec{q}_i = (u^T p_i) \cdot \vec{u}$$

After projection we get all q in the direction of u so all $\vec{q}_i$s can be represented as its scalers, so $q_1, q_2, \ldots, q_n$ are represented as $u^T p_1, u^T p_2, \ldots, u^T p_n$.

Variance for these q_is can be find as:

$$\sigma_q^2 = \frac{\sum_{i=1}^{n} (q_i - \bar{q})^2}{n}$$

where $\bar{q} = \frac{\sum q_i}{n} = \frac{\sum_{i=1}^{n} u^T p_i}{n} = u^T \frac{\sum_{i=1}^{n} p_i}{n} = u^T \bar{p}$

$$
\begin{aligned}
\sigma_q^2 &= \frac{\sum_{i=1}^{n}(q_i - \bar{q})^2}{n} \\
&= \frac{\sum_{i=1}^{n}(u^T p_i - u^T \bar{p})^2}{n} \\
&= \frac{\sum_{i=1}^{n}(u^T (p_i - \bar{p}))^2}{n} \\
&= \frac{\sum_{i=1}^{n}(u^T (p_i - \bar{p}))(u^T (p_i - \bar{p}))^T}{n} \\
&= \frac{\sum_{i=1}^{n} u^T (p_i - \bar{p})(p_i - \bar{p})^T \cdot u}{n} \\
&= u^T \cdot \left[\frac{1}{n} \sum_{i=1}^{n} (p_i - \bar{p})(p_i - \bar{p})^T \right] \cdot u \\
\sigma_q^2 &= u^T S u
\end{aligned}
$$

where S is the co-variance matrix of p_is.

We want to maximize the variance of this q_is and we know that here u is the unit vector.

$$Maximize: \ u^T S u$$

$$Subject\ to\ : u^T u = 1$$

using langrange method

$$
\begin{aligned}
F(u) &= u^T S u - \lambda(u^T u - 1) \\
F'(u) &= 2Su - 2\lambda u \\
0 &= 2Su - 2\lambda u \\
Su &= \lambda u
\end{aligned}
$$

Here we can see that u is the eigen vector of the S (co-variance matrix of p_is) and λ is the eigen value related to that eigen vector.

$$
\begin{aligned}
Su &= \lambda u \\
u^T S u &= u^T \lambda u \\
\sigma_q^2 &= \lambda u^T u \\
\sigma_q^2 &= \lambda
\end{aligned}
$$

Here λ is the maximum co-variance.

In this way we can select eigen vector as u which has highest eigen value. This derivation is apply on higher dimensions also.

For example, let's take data points with m dimensions and we want to reduce that data points into n dimensions i.e. $n < m$. For that find co-variance matrix and get n eigen vector of of that matrix which have maximum eigen values.

3.1.2 Change of Basis

Let $X_{m\times n}$ is original centered dataset where each column is a single sample of dataset so we have n samples and each sample has m features.

Let Y be another $m \times n$ matrix which is related to X by linear transformation P. So, Y is new representation of the data.

$$Y = PX$$

Let take p_i are rows of P, x_i are columns of X and y_i are the columns of Y.

$$Y = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$$

$$Y = \begin{bmatrix} p_1x_1 & p_1x_2 & \dots & p_1x_n \\ p_2x_1 & p_2x_2 & \dots & p_2x_n \\ \vdots & \vdots & \ddots & \vdots \\ p_mx_1 & p_mx_2 & \dots & p_mx_n \end{bmatrix}$$

$$y_i = \begin{bmatrix} p_1x_i \\ p_2x_i \\ \vdots \\ p_mx_i \end{bmatrix}$$

Here each co-efficient of y_i is dot product of x_i with the corresponding row in P. So, y_i is the projection of x_i on the basis of $\{p_1, p_2, \dots, p_m\}$ therefor the row of P are a new set of basis vectors.

Let's take $\{p_1, p_2, \dots, p_m\}$ are orthonormal so that the P matrix will become rotation and stratched transformation and

$$P^T = P^{-1}$$
$$P \cdot P^T = I$$

These new basis $\{p_1, p_2, \dots, p_m\}$ are our principal components of X. Now, a question arise that what is a good choice of basis P?

While transforming we want to remove redundancy of data and maximize the information or signal.

The co-variance matrix of Y is

$$C_y = \frac{1}{n} YY^T$$

All the diagonal terms in C_y must be near to zero for minimizing the redundancy. Thus, C_y must be diagonal matrix or we can say that Y is decorrelated.

So, we derive our algebraic solution to PCA is, find some orthogonal matrix P in $Y = PX$ such that C_y is diagonal matrix. The rows of P are the Principal Component of X.

$$\begin{aligned} C_y &= \frac{1}{n} YY^T \\ &= \frac{1}{n} (PX)(PX)^T \\ &= \frac{1}{n} PXX^TP \\ &= P\Big(\frac{1}{n} XX^T\Big)P^T \\ C_y &= PC_xP^T \end{aligned}$$

This is the relation between C_y and C_x.

C_x is symmetric matrix so it can be diagonalized by an orthogonal matrix of its eigen vectors.

$$C_x = EDE^{-1}$$

$$C_x = EDE^T$$

where E contains the eigen vectors and D is diagonal matrix.

Here comes the trick, matrix E is orthogonal matrix and its columns are eigen vector of C_x. P is also orthogonal matrix and its rows are new basis. Let's consider these eigen vectors as new basis in other word let's make $P = E^T$.

$$\begin{aligned} C_y &= PC_xP^T \\ &= E^T(EDE^T)E \\ &= (E^TE)D(E^TE) \\ &= IDI \\ C_y &= D \end{aligned}$$

C_y becomes Diagonal matrix so, we can take $P = E^T$ in other words new basis are the eigen vectors of C_x.

If we sort the diagonal elements of C_y i.e. rearrange high to low value, we can also select corresponding eigen vector or base which has highest variance.

We can summarize the results of the PCA in matrices P and C_y,

1. The principal components of X are the eigenvectors of C_x.
2. The i^{th} diagonal value of C_y is the variance of X along P_i and also eigen value related to P_i

3.2 Algorithm of PCA

The steps for the dimensionality reduction algorithm are as given below:

1. Normalize the data.
2. Calculate the eigenvalues and eigenvectors of the covariance matrix or correlation matrix or use SVD.
3. Sort all eigenvalues and choose the k eigenvectors that are correspond to the k largest eigenvalues. Here k is the number of dimension of the new feature subspace.
4. Construct the projection matrix W from the selected k Eigen vectors.
5. Transform the original dataset X using W to obtain k dimensional new feature subspace Y.

3.3 Application of PCA

Applications of PCA are as listed below:

- In Neuroscience, PCA as a dimension reduction technique is used to detect coordinated activities of large neuronal ensembles. It has been used in determining collective variables, that is, order parameters, during phase transitions in the brain.
- PCA can be used in Quantitative finance, where we can hedge the fixed income portfolios, implement interest rate models, analyse the shape of yield curve, forecast portfolio returns, develop asset allocation algorithms and long short equity trading algorithms.
- PCA can also be used in image compression.
- PCA has been used on Medical Data to show correlation of Cholesterol with low density lipoprotein.
- PCA is applicable in the technique of the facial recognition too.
- PCA has been used on HVSR(horizontal to vertical spectral ratio) data aimed towards the seismic characterization of earthquake prone areas.
- PCA has been also used for detecting and visualizing the Computer Network Attacks.
- PCA has been utilized in Anomaly Detection.

3.4 Limitations of PCA

As we know, variance explained by $Principal\ Component_i$ in nD space is

$$PC_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

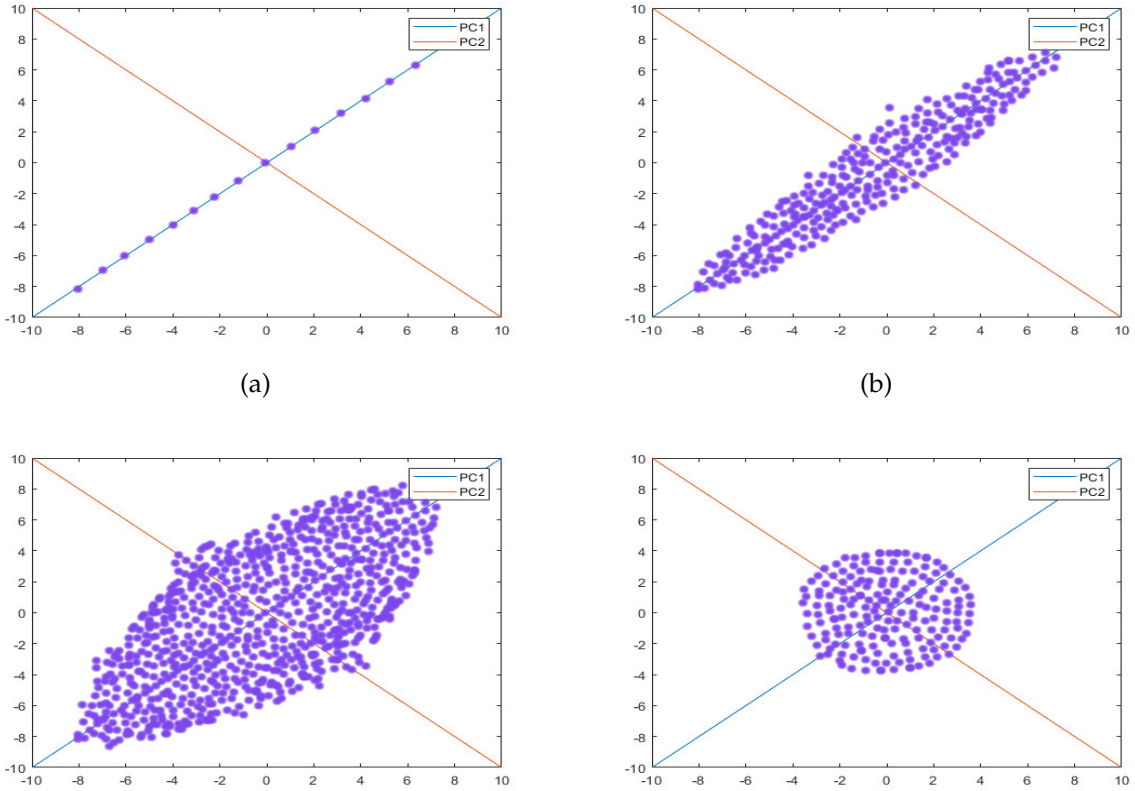
lets take example of dimensionality reduction of 2D to 1D. Let PC_1 be the first principal component with the highest variance explained, and PC_2 be the second principal component. Variance of PC_1 and PC_2 can be defined as follow:

$$\sigma_{PC_1} = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

$$\sigma_{PC_2} = \frac{\lambda_2}{\lambda_1 + \lambda_2}$$

In figure-(a), we can clearly drop PC_2 as there's no spread on PC_2 and

As we go through subsequent diagrams, the spread on PC_2 increases, and the total variance explained by PC_1 decreases. In the last diagram total variance explained by PC_1 is down to 50% which makes data loss higher if it is dropped. So this tells us that PCA works for data in which there's a significantly larger data spread on a fewer number of PCs. (It fails on cases like the last diagram)



(a) (b)

(c) (d)

Figure 1: Variations of spread on both the PCs

In the below-given case, we can clearly see that spread on PC_1 is significantly larger than PC_2 but if we choose to drop PC_2, the pattern in the data will be lost and our data will be reduced to just a set of equidistance points on PC_1.

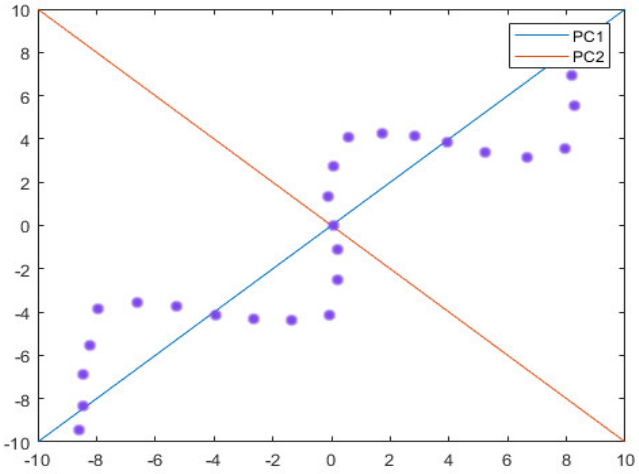


Figure 2: Data containing pattern

3.5 Varients of PCA

3.5.1 Robust PCA

Robust principal component analysis is a variant of PCA which works well with corrupted/grossly collected observations where there are good amount of outliers too (i.e. where the conventional PCA fails). Robust PCA finds its applications where data is in the form such that its matrix form is high dimensional but of low rank. In Robust PCA, we assume that our data matrix M can be decomposed into low rank matrix, and sparse error matrix.

$$M = L_0 + C_0$$

L_0 is low rank matrix whose rank is unknown. C_0 is sparse matrix whose number of non-zero columns are unknown. Our problem is like below given scenario,

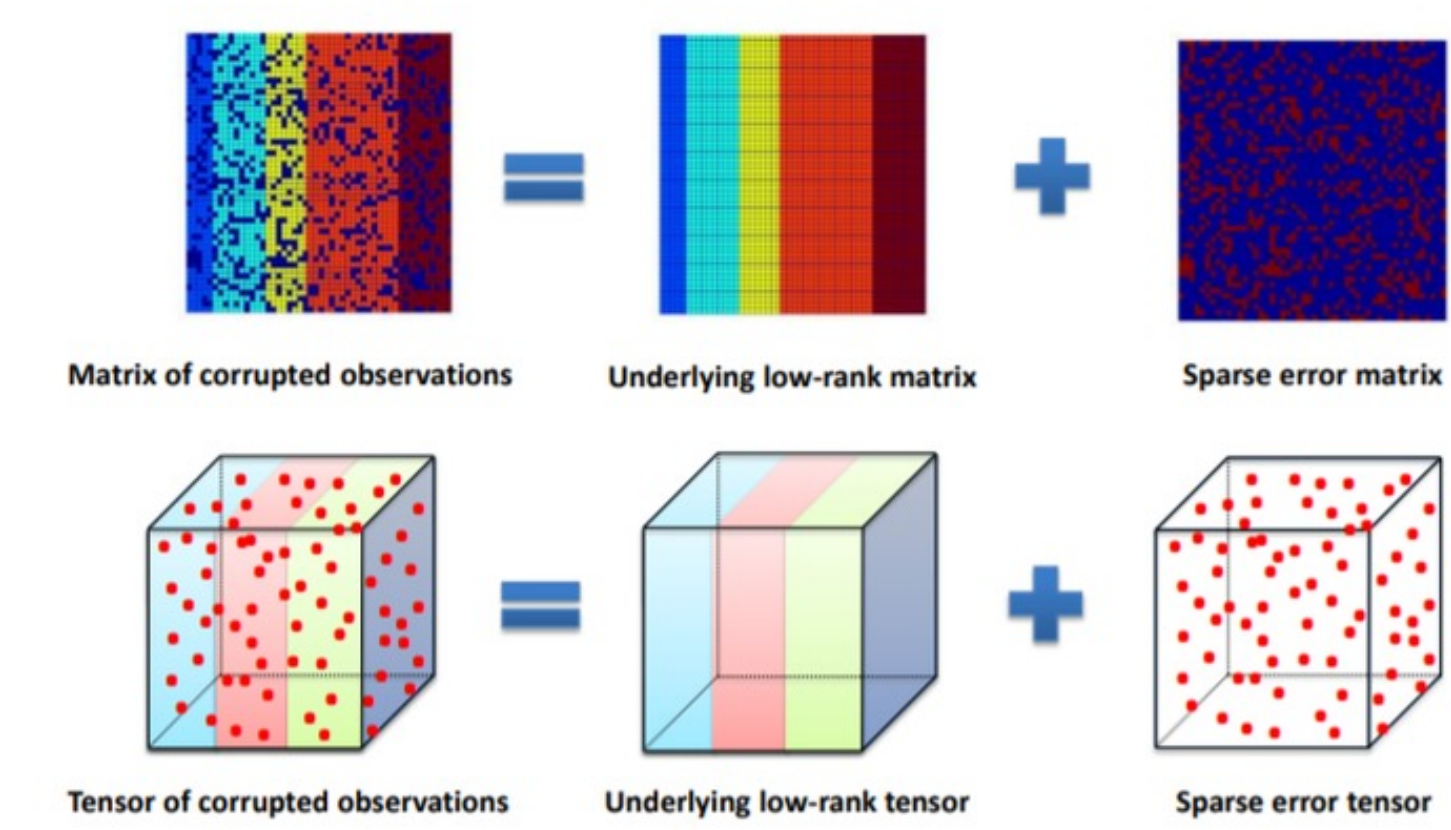


Figure 3: RPCA pictorial representation

It is known to be useful in Video surveillance where we want to detect unusual activity from sequence of frames; we take each frame as column of matrix, static background can be taken as a low rank matrix, L_0 and moving object can be taken as a sparse matrix, C_0. Our aim is to recover uncorrupted low rank, L_0 matrix from our data, M and identify our outliers, C_0.

Suppose we've been given n data points, each of p dimensions. From the given data, a fraction, gamma points are arbitrary points (which we call outliers) and fraction, (1 – gamma) points reside in r-dimensional space which are true subspace of Rp. As we know that

$$M = L_0 + C_0$$

C_0 is column sparse matrix corresponding to outliers in which $(1 - \gamma)n$ columns are zero. L_0 is low rank matrix corresponding to underlying uncorrupted data having $rank(L_0) = r$ It is to be noted that,

columns of L_0 corresponding to non-zero columns of C_0 should be all zeros. Consider L_0s SVD,

$$L_0 = U_0 \sum\nolimits_0 V_0^T$$

The columns of U_0 form an orthonormal basis for the r-dimensional subspace we wish to recover.C_0 is the matrix corresponding to the outliers; we will denote the set of non-zero columns of C_0 by I_0, with $|I_0| = \gamma n$. These non-zero columns are completely arbitrary. Our intent is to *exactly* recover the column space of L_0, and the set of outliers I_0. All we are given is the matrix M. Clearly, *exact* recovery is not always going to be possible (regardless of the algorithm used) and thus we need to impose a few weak additional assumptions.

$$M = L_0 + C_0 + N$$

and N corresponds to any additional noise. In this case we are interested in approximate identification of both the true subspace and the outliers.

Consider a case that our matrix M has only one non-zero column. this will lead L_0 to be both low rank and sparse at a time. so as to not let this happen, we impose following condition

A matrix $L \in \mathbb{R}^{p \times n}$ with SVD $L = U \sum V^T$, and $(1-\gamma)n$ of whose columns are non-zero, is said to be column-incoherent with parameter μ if

$$max_i \left\| V^T e_i \right\|^2 \leq \frac{\mu r}{(1-\gamma)n'}$$

where e_i are the coordinate unit vectors. Thus if V has a column aligned with a coordinate axis, then $\mu = (1-\gamma)n/r$. Similarly, if V is perfectly incoherent (e.g., if $r = 1$ and every non-zero entry of V has magnitude $1/\sqrt{(1-\gamma)n}$ then $\mu = 1$.

Given the data matrix M, our algorithm, called Outlier Pursuit, generates (a) a matrix U^*, with orthonormal rows, that spans the low-dimensional true subspace we want to recover, and (b) a set of column indices I^* corresponding to the outlier points. The steps for outlier pursuit algorithm is as follows:

1. Find (L^*, C^*), the optimum of the following convex optimization program
2. Minimize: $\|L\|_* + \lambda \|C\|_{1,2}$
   Subject to: $M = L + C$
3. Compute SVD $L^* = U_1 \sum_1 V_1^T$ and output $U^* = U_1$.
4. Output the set of non-zero columns of C^*, i.e., $I^* = \{j : c_{ij}^* \neq 0 for some i\}$

While in the noiseless case there are simple algorithms with similar performance, the benefit of the algorithm, and of the analysis, is extension to more realistic and interesting situations where in addition to gross corruption of some samples, there is additional noise. Adapting the Outlier Pursuit algorithm, we have the following variant for the noisy case.

Noisy Outlier Pursuit:
Minimize: $\|L\|_* + \lambda \|C\|_{1,2}$
Subject to: $\left\| M - (L + C) \right\|_F \leq \epsilon$

Outlier Pursuit (and its noisy variant) is a convex surrogate for the following natural (but combinatorial and intractable) first approach to the recovery problem:

$$\begin{array}{c} \text{Minimize: } rank(L) + \lambda \|C\|_{0,c} \\ \text{Subject to: } M = L + C \end{array}$$

where $\|.\|_{0,c}$ stands for the number of non-zero columns of a matrix.

3.5.2 Non Linear PCA

Traditional PCA is optimal for dimensionality reduction only if data distribution falls along orthogonal axes. We Can define nonlinear generalisation, Nonlinear principal component analysis (NLPCA), which can robustly characterise nonlinear low-dimensional structure in datasets. NLPCA is commonly seen as a nonlinear generalization of standard principal component analysis (PCA). It generalizes the principal components from straight lines to curves. Thus, the subspace in the original data space which is described by all nonlinear components is also curved.

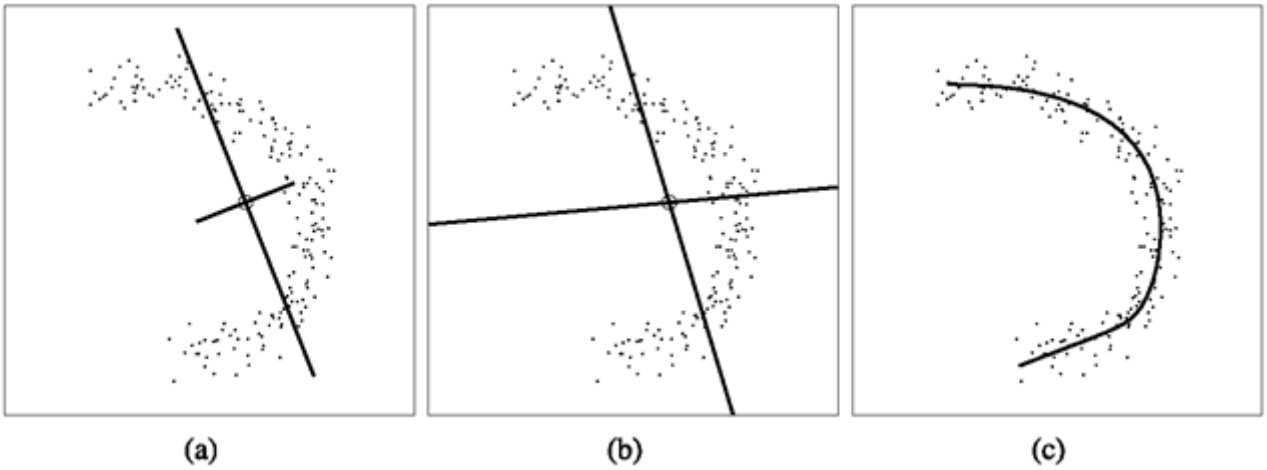


Figure 4: NLPCA pictorial representation

As shown in figure the best fit of data is a curve(c) which is nonlinear estimation of the data. So, here NLPCA is more accurate than PCA.

The main goal of non NLPCA is to find functions with $M < P$

$$s_f : R^P \rightarrow R^M, f : R^M \rightarrow R^P$$

$$\text{such that } X(t) = (f \circ s_f)(X(t)) + \epsilon(t)$$

where

- $E\{\left\|\epsilon^2\right\|\}$ is minimised.
- $f(\lambda)$ approximation manifold.
- $\lambda(t) = s_f(X(t))$ manifold parameterisation (time series).

Nonlinear PCA can be achieved by using a neural network. Implementation of NLPCA is difficult and lacking in underlying theory.

4 Implementation/Experiments

We use Algorithm as mentioned in section 3.2 to implement PCA Dimensionality Reduction in python.

In implementation we most popular Iris dataset. The Iris dataset contains measurement of 150 iris flowers from three different species Iris-Setoso, Iris-Versicolor and Iris-virginica. The four features are sepal length(cm), sepal width(cm), petal length(cm) and petal width(cm).

Using PCA we reduce dimension of data and convert 4 features into 2 new features PC_1 and PC_2.

Data was not in standard form so to standardize data we use sklearn's StandardScaler module. Calculation of covariance matrix, correlation matrix, eigenvalues and eigenvectors of these matrices, SVD,

Sorting of pairs of eigenvalues and eigenvectors, construction of the projection matrix and creation of new featured data is implemented using numpy.

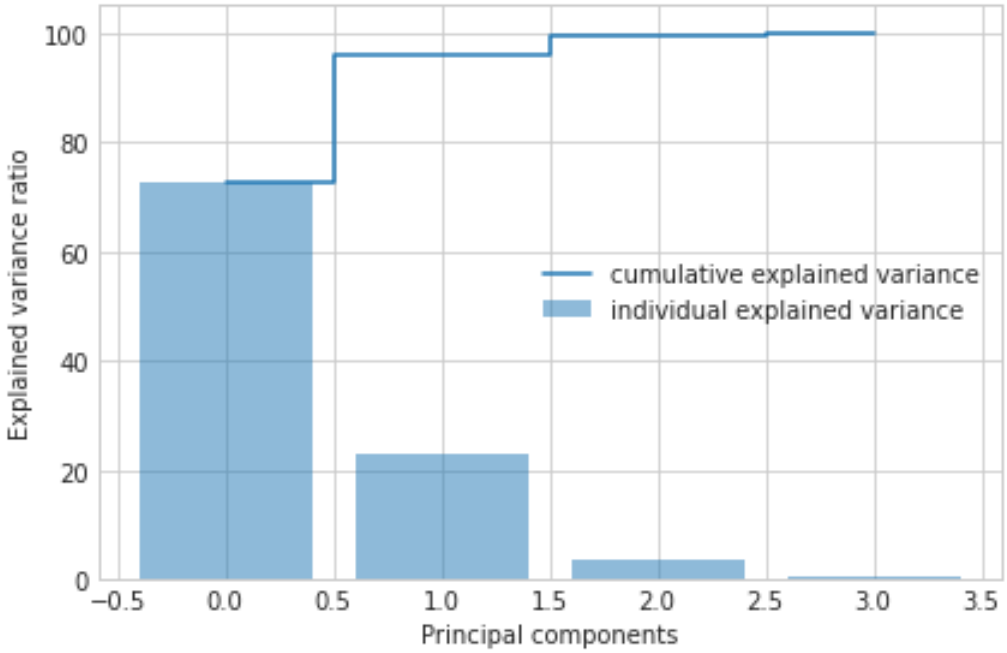


Figure 5: Variance ratio of PCs

As shown in above figure Individual explained variance of the Pcs are 72.77%, 23.03%, 3.68% and 0.52%. As we can see the 95.8% of the total variance is covered by the first two pcs. So our final data has 95.8% accurate information.

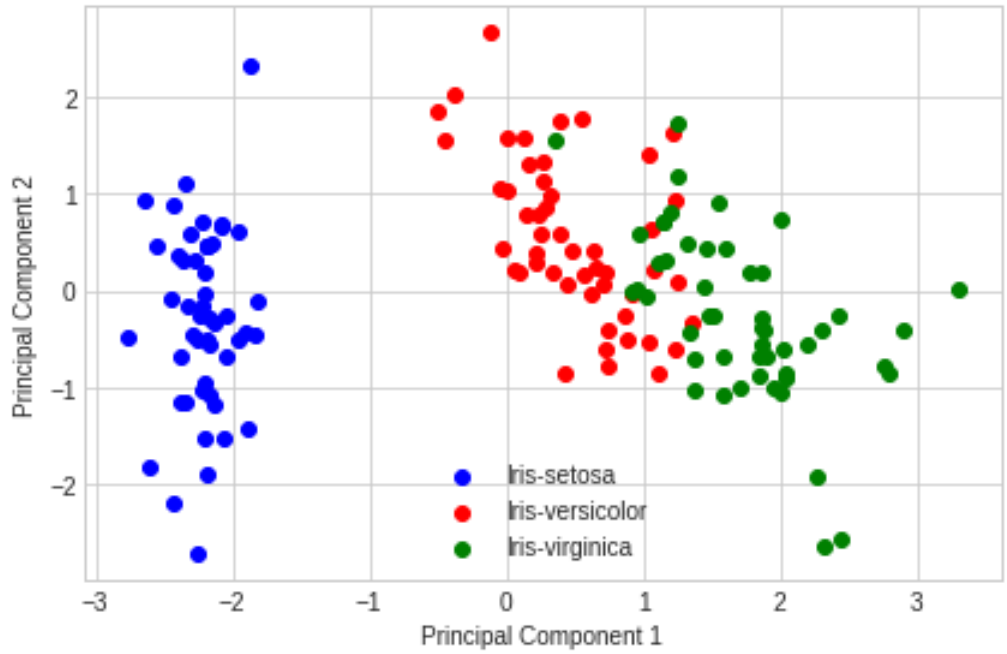


Figure 6: Projection of data on PC_1 and PC_2

These points are projected data where X-axis is PC_1 and Y-axis is PC_2. Here we can differentiate the different cluster of each class.

5 Discussion & Conclusion

Principal component analysis is probably the oldest and best known of the techniques of multivariate analysis. Inspite of emergence of new dimensionality reduction algorithms like t-SNE and neural AutoEncoders, PCA is still heavily used among machine learning community due to its intuitive procedure and inherent simplicity. In some cases if we can not find best fit subspace using PCA we can use RPCA or NLPCA to fit data in proper subspace.

References

[1] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[2] Sebastian Raschka. Implementing a principal component analysis (pca). *SebastianRaschka, https://sebastianraschka. com/Articles/2014_pca_step_by_step. html*, 2014.

[3] Sebastian Raschka. Implementing a principal component analysis (pca) in python step by step, 2014.

[4] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[5] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, fourth edition, 2009.

[6] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. *IEEE transactions on information theory*, 58(5):3047–3064, 2012.