The linear algebra of Principal Component Analysis

(with some Python)

Kamila Zdybał

Université libre de Bruxelles, kamila.zdybal@ulb.ac.be
camillejr.github.io/science-docs, kamila.zdybal@gmail.com

Preface

These are notes on **Principal Component Analysis** (PCA), a dimensionality reduction technique in which the data set is reduced to maintain only the directions of the largest variance.

The deep and intuitive understanding of PCA requires the deep and intuitive understanding of basic linear algebra. I highly recommend a very pleasant to watch course by 3Blue1Brown YouTube channel, which, to my belief, will be everything you need to master about linear algebra to gain a great intuition behind PCA.

This document is still in preparation. Please feel free to contact me with any suggestions, corrections or comments.

Keywords

principal component analysis, data reduction, dimensionality reduction, linear algebra, MATLAB®, Python

Contents

1 Motivation for data reduction

There are several questions which stimulated the usage of data reduction and data decomposition techniques:

1. Can we send less data but preserve maximum information contained by the data (data compression)?
2. Can we predict what the outcome of another observation will be?
3. Can we build a model in a purely data-driven way - based on data that we have collected (e.g. from experiments or numerics)?
4. Can we make sense of a large, multi-dimensional data set (which cannot be plotted graphically)?
5. Can we extract low-dimensional features that underlay the data?

2 Data sets for PCA

For the rest of this document, we assume that the dataset for performing PCA is structured as follows: each column represents one variable Var_j and the rows correspond to variable observations Obs_i (e.g. at different positions in space, or in time). The raw data matrix $\boldsymbol{X}$ is size $(n \times Q)$.

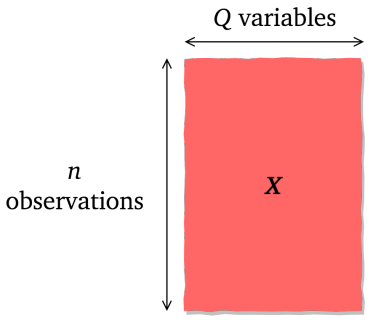

Figure 1: Data matrix for PCA.

This is also the data format that is needed for the MATLAB® command `pca`. From the documentation [1]:

> `pca(X)` Rows of X correspond to observations and columns correspond to variables.

3 Data pre-processing

In data science we are typically given a raw data set which is not centered and not scaled.

Data centering allows to look at it as variations from some center. Typically we might center each variable by subtracting the mean of this variable's observations (realizations). This would substitute the original data set with:

$$\boldsymbol{X'} = \boldsymbol{X} - \text{mean}(\boldsymbol{X}) \tag{1}$$

Such centering will shift the "cloud" of data points (which in general is multi-dimensional) to the origin. However, other values for centering are possible.
Scaling, on the other hand, allow us to cancel the effect of unit that a variable might have and treat all variables with equal importance.

$$\boldsymbol{X''} = \frac{\boldsymbol{X'}}{\boldsymbol{D}} \tag{2}$$

You might for instance think about a set of variables from a single experiment, representing temperature in the units of $[K]$ and range from $300K$-$1500K$ and associated pressures in the units of $[atm]$ which range from $1atm$-$1.1\ atm$. If we did not scale the data, the largest "spread" or the variance would be found in temperature, since on purely numerical grounds, the range 300-1500 is much larger than the range 1-1.1.
For simplicity, for the rest of this document we assume that $\boldsymbol{X}$ represents pre-processed data.

4 Covariance matrix

The starting point of performing PCA is to compute a *covariance matrix* from the raw data set. The covariance matrix is given by:

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} \tag{3}$$

The covariance matrix $\boldsymbol{S}$ is therefore size $(Q \times Q)$.
We will start by exploring the meaning of $\boldsymbol{X}^T\boldsymbol{X}$[1]. Let's look at the graphical representation of this matrix multiplication in Fig. 2.

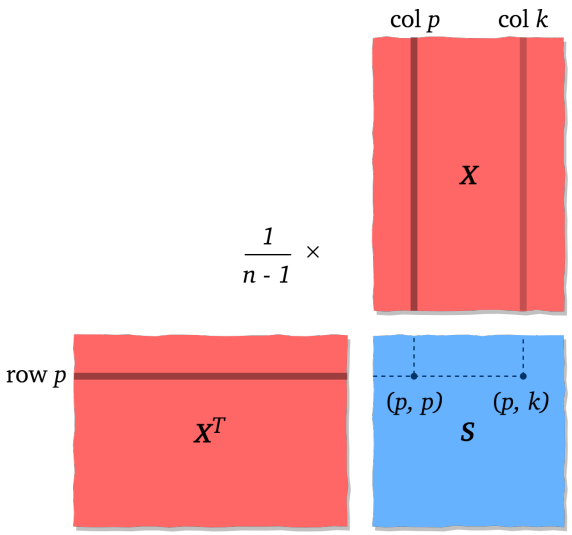

Figure 2: Covariance matrix $\boldsymbol{S}$ graphical interpretation.

[1]It is worth noticing here, that a matrix constructed as: $\boldsymbol{S} = \boldsymbol{A}^T\boldsymbol{A}$ is square and symmetric and has got at all eigenvalues at least non-negative. It is therefore at least a *positive semidefinite* matrix. If it also happens that this matrix has got only positive eigenvalues we call it a *positive definite* matrix.

Any given column, say p or k, of the data matrix $\boldsymbol{X}$ represents all measurements of a single variable Var_p (or Var_k) and can be viewed as a single n-dimensional vector. The same can be said about any row of a matrix $\boldsymbol{X}^T$.

Notice that an element at position (p,k) inside the matrix $\boldsymbol{S}$ has the interpretation of a dot product between a vector formed by the p-th row of a matrix $\boldsymbol{X}^T$ and a k-th column of a matrix $\boldsymbol{X}$. There is also the $\frac{1}{n-1}$ factor out front to which we will come back later.

$$s(p,k) = \frac{1}{n-1}dot(\boldsymbol{X}^T(p,:), \boldsymbol{X}(:,k)) \tag{4}$$

In a special case, where we multiply the row p with the column p, we get a dot product of a vector with itself.

$$s(p,p) = \frac{1}{n-1}dot(\boldsymbol{X}^T(p,:), \boldsymbol{X}(:,p)) \tag{5}$$

In general, the dot product between two vectors x and y represents how much vector x lays in the direction of vector y (and vice versa) - and it is zero when two vectors are perpendicular to each other. This intuition can be carried to our covariance matrix $\boldsymbol{S}$. If any off-diagonal element is non-zero, say element at position (p,k) this means that some information about variable Var_p is carried by a variable Var_k (and vice versa).

In other words, every element in the covariance matrix is:

$$s(i,j) = \frac{1}{n-1}\sum_{k=1}^{n}\text{Var}_i(k)\text{Var}_j(k) \tag{6}$$

5 PCA workflow

The next step in PCA is to perform the eigendecomposition of the covariance matrix:

$$\text{eig}(\boldsymbol{S}) = [\boldsymbol{A}, \boldsymbol{\Lambda}] \tag{7}$$

The matrix $\boldsymbol{A}$ is a matrix of eigenvectors and it is size $(Q \times Q)$. Each eigenvector is called a *principal component* (PC). The principal components are orthogonal to each other, and therefore the following important property holds: $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$ (proof[2]).

The diagonal matrix $\boldsymbol{\Lambda}$ of size $(Q \times Q)$ is a matrix of corresponding eigenvalues.

Given the eigendecomposition of matrix $\boldsymbol{S}$, we may state that: $\boldsymbol{S} = \boldsymbol{A}\boldsymbol{\Lambda}\boldsymbol{A}^T$.

The principal components form a new basis in which we can represent our data set. We perform a transformation of the original data matrix $\boldsymbol{X}$ from the original space to the new space represented by the PCs. This transformation is achieved by the following multiplication:

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{A} \tag{8}$$

[2]Proof: for orthogonal columns of $\boldsymbol{A}$ we have $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ Multiplying both sides by $\boldsymbol{A}^{-1}$ we get $(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{A}^{-1} = \boldsymbol{I}\boldsymbol{A}^{-1}$. Since matrix multiplication is associative, we may also perform: $\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^{-1}) = \boldsymbol{A}^{-1}$. From definition of an inverse matrix, $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$, hence: $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$.

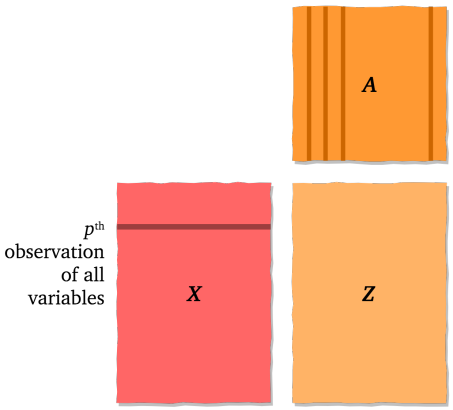

Figure 3: Data transformation to a new basis.

The new matrix $\boldsymbol{Z}$ is still our dataset $\boldsymbol{X}$ but represented in the basis associated with the matrix $\boldsymbol{A}$. It is also called the *PC-scores* matrix, since one may think of every element in this matrix as the "score" that the corresponding element in $\boldsymbol{X}$ gets when represented in the new coordinate system after transformation.

In the matrix multiplication from eq.(8), every variable vector inside $\boldsymbol{X}$ gets transformed by the transformation matrix $\boldsymbol{A}$ and attains new scores in the basis associated with $\boldsymbol{A}$. The new representation of the old variable is now kept in the matrix $\boldsymbol{Z}$.

We now approach the dimensionality reduction but first let's obtain the original data set back, given the PC-scores and the transformation matrix:

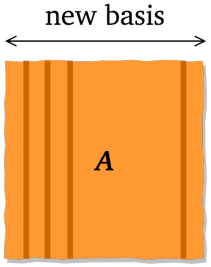

Figure 4: Eigenvalues of the covariance matrix form a new basis.

$$\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{A}^T \tag{9}$$

The above equation is our route back to obtain the original data set in which the PC-scores are projected on the basis associated with a transposed eigenvectors matrix $\boldsymbol{A}$ (recall that $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$).

Suppose that we would like to find the approximation of the data matrix $\boldsymbol{X}$ with only q principal components (we project the PC-scores onto only q out of Q principal components).

We shrink the transformation matrix $\boldsymbol{A}$ to be of size $(Q \times q)$ (we only keep q principal components). To match the matrix sizes we also need to shrink in size the PC-scores matrix which originally is size $(n \times Q)$ - the same size as the data matrix $\boldsymbol{X}$. We will denote these truncated matrices $\boldsymbol{A_q}$ and $\boldsymbol{Z_q}$ respectively.

Projecting $\boldsymbol{Z_q}$ onto the basis $\boldsymbol{A_q^T}$ will result in an an approximation of the original data set:

$$\boldsymbol{X_q} = \boldsymbol{Z_q}\boldsymbol{A_q^T} \tag{10}$$

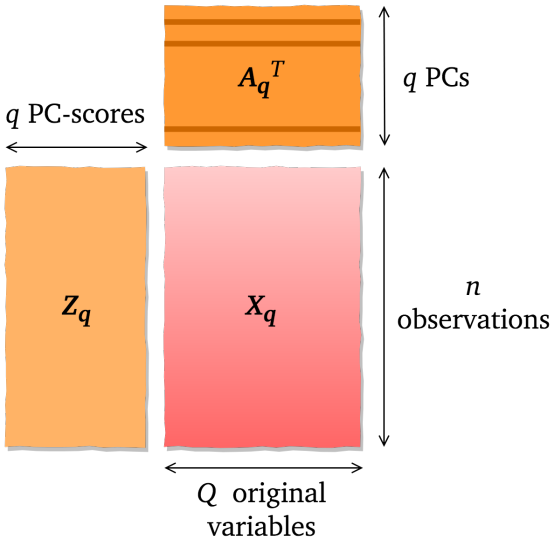

Figure 5: Data approximation with q-PCs.

6 Why does it need to be eigenvectors?

In this section we come back to equation (7) and answer the question: why are principal components the eigenvectors of a covariance matrix? If you are new to PCA, this indeed might seem like a not-related-to-anything thing to do. So why is it eigenvectors? And why does it need to be eigenvectors of this special matrix called covariance matrix?
To begin the understanding, let's look back at Figure 14. What we want to achieve with PCA is to diagonalize the new covariance matrix. And we already have produced one diagonal matrix even before computing PC-scores - it was the matrix of eigenvalues $\boldsymbol{\Lambda}$.

7 PCA Python example

We will now go on to visualizing on an artificial 2D data set every step of PCA. We will use the `PCA` function from a Python library `sklearn.decompositions`. The full code can be accessed in the GitHub repository. Here, we will only recall elements of that code to explain what is happening.
We create the `Dataset` (the equivalent of $\boldsymbol{X}$) as follows:

```
import numpy as np
Np = 100
x = np.linspace(3, 6, Np)
y = 0.8*x + 1*np.random.rand(Np)
Dataset = np.column_stack((x, y))
```

Let's assume that the first column of this data set are realizations of the first variable x and the second column are realizations of the second variable y.
The two variables x and y span two dimensional space but the data set exhibits a low-dimensional structure which is easily visible to the eye just by looking at the Figure 6. We already see that the data seems to be spread along some linear function. Perhaps changing the basis to a basis associated with this linear function will be a more effective representation by our data set? Note here also, that for multidimensional data sets "seeing" such data structures is no longer possible (as

it still is in 2-D or 3D). We need to rely on the dimensionality reduction technique that we chose to find this low-dimensional structure for us.

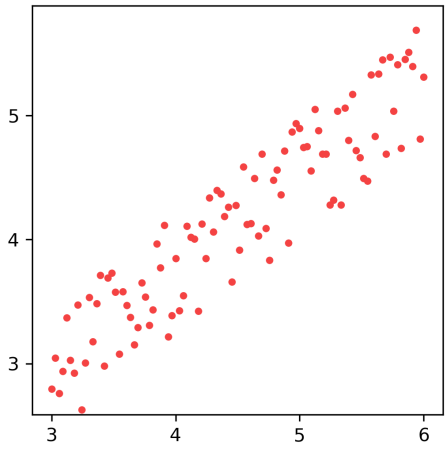

Figure 6: Raw data set.

We center the dataset, which simply moves the center of the cloud of points to the origin of the coordinate system. If necessary, data set would also be scaled to allow for even comparison of the two variables.

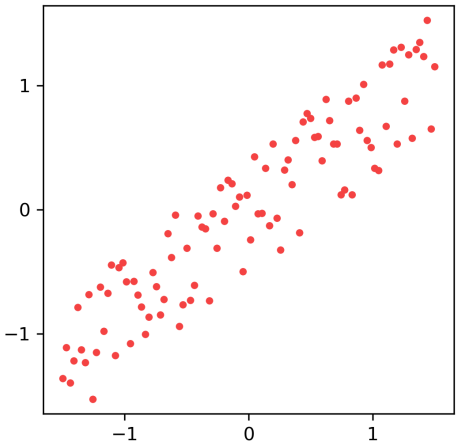

Figure 7: Data set centered.

Next, PCA is performed on the dataset and the eigenvectors (the Principal Components) `PCs` are found, with the corresponding eigenvalues `eigvals`.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(Dataset)
PCs = pca.components_
eigvals = pca.explained_variance_ratio_
```

In the above code, we create an object `pca` of class `PCA`. We train the model with our `Dataset` using the `fit` function.

The eigenvectors are plotted on the dataset in Figure 8. Their lengths are proportional to their corresponding eigenvalue. Notice that PCA was able to find the direction of the largest variance in the data marked by the direction of the first, longest Principal Component. The second PC is simply perpendicular to the first one.

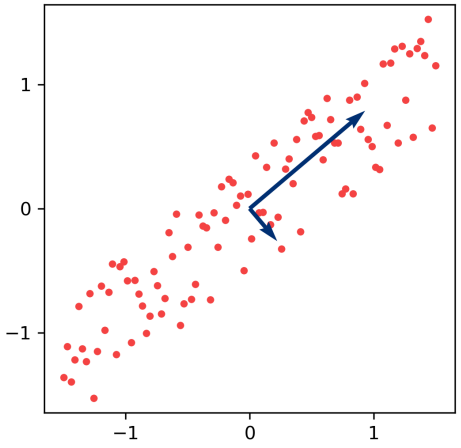

Figure 8: Data set with principal components.

We may now find the PC-scores matrix which represents the ”scores” each point from the data set attains when the new coordinate system was associated with the directions of the Principal Components.

```
scores = pca.transform(Dataset)
```

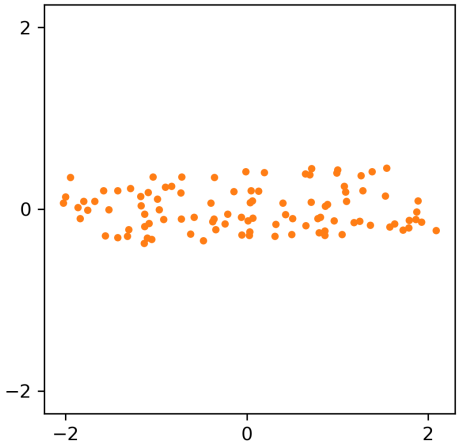

Figure 9: PC-scores.

Next, the PC-scores can be projected on the first PC, to reduce the dimensionality - from two dimensions to one. The data representation from Figure 10 can be viewed as the ”scores” each data point would attain it represented on 1-dimensional structure associated with the first Principal Component.

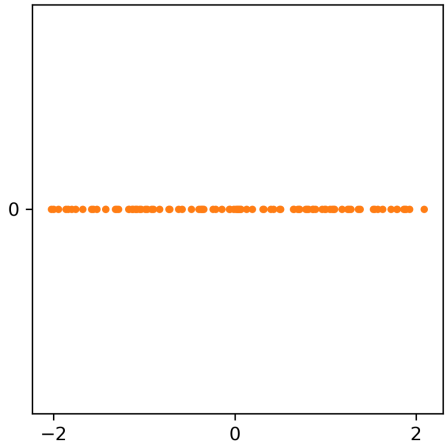

Figure 10: Data projection on low dimension.

The data set can be reconstructed. This represents going back from the 1-dimensional space to the original dimensions. The mean of the data set is added back to undo the data centering.

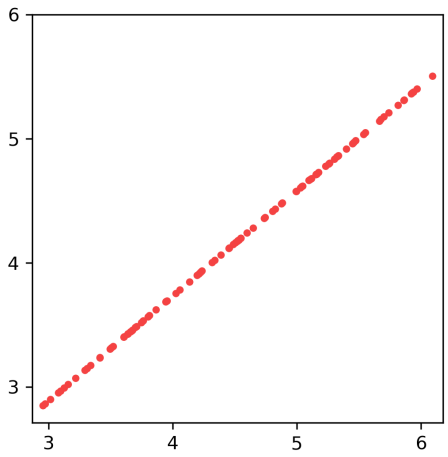

Figure 11: Data approximation with $q = 1$.

8 Another look, or how it all links together

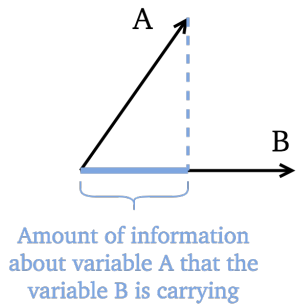

Figure 12: Variable basis might not be optimal.

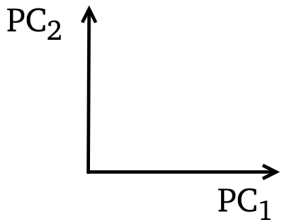

Figure 13: PC-basis is orthogonal.

9 Interesting questions

9.1 Is the covariance matrix of PC-scores a diagonal matrix?

The goal of PCA in terms of a covariance matrix

Principal Component Analysis aims to find a new, transformed data set $\boldsymbol{Z}$ such that if we computed a new covariance matrix:

$$\boldsymbol{S}_Z = \boldsymbol{Z}^{\boldsymbol{T}}\boldsymbol{Z} \tag{11}$$

the variances (the elements on the diagonal) are maximized and the covariances (the off-diagonal elements) are zero. This means that no more information about any column of $\boldsymbol{Z}$ is carried by any other column of $\boldsymbol{Z}$.

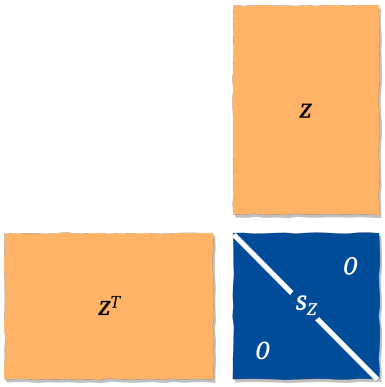

Figure 14: New transformed data set $\boldsymbol{Z}$ and its covariance matrix $\boldsymbol{S}_Z$.

A Properties of interesting matrices

A.1 Symmetric matrices

The properties of symmetric matrices:

1. Their eigenvectors are orthogonal.
2. Their eigenvalues are real.

B APP2

References

[1] `https://nl.mathworks.com/help/stats/pca.html`

[2] Ian T. Jolliffe, *Principal Component Analysis*, Second Edition, 1986

[3] Gilbert Strang, *Introduction to Linear Algebra*, Fifth Edition, 2016

[4] Jonathon Shlens, *A Tutorial on Principal Component Analysis*, 2016, `https://arxiv.org/abs/1404.1100`

[5] `http://people.sju.edu/ pklingsb/dot.cov.pdf`

[6] J. Edward Jackson, *A User's Guide To Principal Components*, 1991

[7] Lindsay I. Smith, *A tutorial on Principal Component Analysis*, 2002