Understanding principal component analysis

Shiqiang Lin *1

1 *College of Life Science, Fujian Agriculture and Forestry University*

Abstract The principal component analysis (PCA) is a frequently used machine learning method. In this paper, the PCA operation is explained by examples with Python program illustration. A proof of the diagonalizability of real symmetric matrix is also included, which may help to understand the mathematics behind PCA.

The principal component analysis (PCA) performs linear combinations of the variables to obtain the few principal components that are mutually uncorrelated. Most information of the original data is kept in the principal components. As a method of data dimensionality reduction, the PCA plays an important role in data processing [1, 2]. The principle of PCA is singular value decomposition (SVD), which involves quite a few crucial concepts of linear algebra such as eigenvalue, eigenvecotor, basis change, orthogonality, symmetric matrix, and matrix multiplication, and so on. The linear algebra is essential to the theory and practice of big data, machine learning, algorithms, and programming. In this paper I use examples with Python program to explain PCA, step by step, till the mathematical proof of PCA and SVD. It is hoped that the interpretation of PCA will open the door of machine learning for the numerous machine learning enthusiasts.

Results

Euler's formula and rotation of points in a two-dimensional plane

The Euler's formula states that $e^{i\theta} = cos\theta + isin\theta$, where e is the natural logarithmic base, i is the imaginary unit, and θ is a real number. Now suppose a point M with coordinate $(cos\theta, sin\theta)$ within the unit circle in a two-dimensional plane. By rotating the M along the unit circle to M' with coordinate $(cos(\theta+\delta), sin(\theta+\delta))$, we have

$$\begin{aligned} e^{i(\theta+\delta)} &= e^{i\theta}e^{i\delta} \\ &= (cos\theta + isin\theta)(cos\delta + isin\delta) \\ &= cos\theta cos\delta + icos\theta sin\delta + isin\theta cos\delta + i^2 sin\theta sin\delta \\ &= (cos\theta cos\delta - sin\theta sin\delta) + i(cos\theta sin\delta + sin\theta cos\delta) \end{aligned}$$

Therefore, the coordinate of M' becomes $(cos\theta cos\delta - sin\theta sin\delta, cos\theta sin\delta + sin\theta cos\delta)$. For a point $N(rcos\theta, rsin\theta)$, $r \in \mathbb{R}$, rotating by δ to N' will change the coordinate to $(r(cos\theta cos\delta - sin\theta sin\delta),$ $r(cos\theta sin\delta + sin\theta cos\delta))$, which can be written as $(rcos\theta cos\delta - rsin\theta sin\delta, rcos\theta sin\delta + rsin\theta cos\delta)$. Let $rcos\theta = a, rsin\theta = b$, the coordinate of N' is $(acos\delta - bsin\delta, asin\delta + bcos\delta)$. The rotation can be shown with matrices.

$$\begin{bmatrix} cos\delta & -sin\delta \\ sin\delta & cos\delta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} acos\delta - bsin\delta \\ asin\delta + bcos\delta \end{bmatrix} \tag{1}$$

$$= a\begin{bmatrix} cos\delta \\ sin\delta \end{bmatrix} + b\begin{bmatrix} -sin\delta \\ cos\delta \end{bmatrix} \tag{2}$$

*Correspondence: linshiqiang@fafu.edu.cn or

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} cos\delta & sin\delta \\ -sin\delta & cos\delta \end{bmatrix} = \begin{bmatrix} acos\delta - bsin\delta & asin\delta + bcos\delta \end{bmatrix} \tag{3}$$

$$= a \begin{bmatrix} cos\delta & sin\delta \end{bmatrix} + b \begin{bmatrix} -sin\delta & cos\delta \end{bmatrix} \tag{4}$$

In view of rotation, for Equation 1, the left side rotates point (a, b) by δ and the right side shows the result of operation. From the perspective of basis change, the basis vectors of $[a \quad b]^{\mathrm{T}}$ changes from $[1 \quad 0]^{\mathrm{T}}$ and $[0 \quad 1]^{\mathrm{T}}$ to $[cos\delta \quad sin\delta]^{\mathrm{T}}$ and $[-sin\delta \quad cos\delta]^{\mathrm{T}}$, respectively, shown in Equation 2. The situation of Equation 3 is similar, in which the left side represents rotation and the right side shows what is obtained. Equation 4 shows the basis change from $[1 \quad 0]$ and $[0 \quad 1]$ to $[cos\delta \quad sin\delta]$ and $[-sin\delta \quad cos\delta]$, respectively.

Variance, covariance and rotation

If we have a dataset $Data = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$, where $x_i, y_i \in \mathbb{R}$, $i = 1, 2, \cdots, N$, then the data can be drawn with a scatter diagram in the rectangular coordinate system. However, the center of these points is not necessarily the origin $(0, 0)$. Therefore, we get the means of x direction and y direction, which are $\bar{x} = \sum_1^N x_i$, $\bar{y} = \sum_1^N y_i$, respectively, and then subtract them from the original data to obtain the centered data $Data\_new = \{(x_1 - \bar{x}, y_1 - \bar{y}), (x_2 - \bar{x}, y_2 - \bar{y}), \cdots, (x_N - \bar{x}, y_N - \bar{y})\}$, designated as $Data\_new = \{(x'_1, y'_1), (x'_2, y'_2), \cdots, (x'_N, y'_N)\}$. The centered data has the same internal structure as the original data. Now we can use the centered data to get the variances of original data, which are $\sigma_x^2 = \sum_1^N x_i'^2/(N-1)$, $\sigma_y^2 = \sum_1^N y_i'^2/(N-1)$.

Now consider a small dataset $Data = \{(58, -88), (13, 18), (26, -29), (-26, 111), (-6, 55), (-16, 67), (103, -138), (67, -104), (20, 10), (18, 36)\}$, which is shown with gray points in Figure 1. The blue points indicates the centered data $Data\_new$.

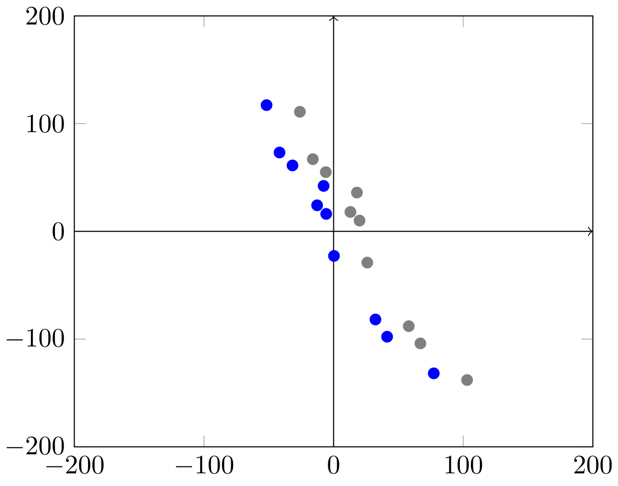

Figure 1: Scatterplot of the datasets

$Data$ has 10 points, which is roughly distributed along a line in the scatterplot. In the direction of this line, the variance of projections is large, whereas that of projections is small in the perpendicular direction. Considering the structural characteristics of the data, we can simplify it, that is, simplify the two-dimensional data into one-dimensional data by finding the direction with the largest variance, and projecting the data points to this direction. In the other (perpendicular) direction, the point values do not change much, and they can be discarded. The question is, how to find the direction with the greatest variance?

Let's start with a general derivation and then look at the specific example. The sample size N is a fixed value here. Thus, we pay our attention to the S_{xx}, S_{yy}, and S_{xy} of the $Data$.

$$S_{xx} = x_1'^2 + x_2'^2 + \cdots + x_N'^2 = \sum_1^N x_i'^2 \tag{5}$$

$$S_{yy} = y_1'^2 + y_2'^2 + \cdots + y_N'^2 = \sum_1^N y_i'^2 \tag{6}$$

$$S_{xy} = x_1'y_1' + x_2'y_2' + \cdots + x_N'y_N' = \sum_1^N x_i'y_i' \tag{7}$$

To find the direction of the greatest variance, we rotate the centered data according to Equation 3. The data after rotation becomes $\{(x_1'cos\delta - y_1'sin\delta, x_1'sin\delta + y_1'cos\delta), (x_2'cos\delta - y_2'sin\delta, x_2'sin\delta + y_2'cos\delta), \cdots, (x_N'cos\delta - y_N'sin\delta, x_N'sin\delta + y_N'cos\delta)\}$, which can be substituted to Equations 5–7.

$$S_{xx} = \sum_1^N (x_i'cos\delta - y_i'sin\delta)^2 \tag{8}$$

$$S_{yy} = \sum_1^N (x_i'sin\delta + y_i'cos\delta)^2 \tag{9}$$

$$S_{xy} = \sum_1^N (x_i'cos\delta - y_i'sin\delta)(x_i'sin\delta + y_i'cos\delta) \tag{10}$$

Comparing Equations 5–6 and Equations 8–9, we can find that the $S_{xx}+S_{yy}$ keeps unchanged before and after rotation. This is easy to understand, as rotation does not change the distance between point and origin. It can also be shown by calculating the distance from point to origin directly with the distance formula. Now consider Equation 8. To get the extreme values, we need to calculate the first derivative.

$$\begin{aligned}(S_{xx})' &= \sum_1^N 2(x_i'cos\delta - y_i'sin\delta)(x_i'cos\delta - y_i'sin\delta)' \\ &= \sum_1^N 2(x_i'cos\delta - y_i'sin\delta)(-x_i'sin\delta - y_i'cos\delta) \\ &= -2\sum_1^N (x_i'cos\delta - y_i'sin\delta)(x_i'sin\delta + y_i'cos\delta)\end{aligned} \tag{11}$$

Comparison of Equation 11 and Equation 10 shows that $S_{xy} = 0$, when $(S_{xx})' = 0$. Let's calculate the first derivative of Equation 9.

$$\begin{aligned}(S_{yy})' &= \sum_1^N 2(x_i'sin\delta + y_i'cos\delta)(x_i'sin\delta + y_i'cos\delta)' \\ &= 2\sum_1^N (x_i'cos\delta - y_i'sin\delta)(x_i'sin\delta + y_i'cos\delta)\end{aligned} \tag{12}$$

It can be shown by comparing Equation 12 and Equation 10 that $S_{xy} = 0$ when $(S_{yy})' = 0$. To get the extreme values, we need to calculate the first derivative of Equation 10.

$$(S_{xy})' = \sum_1^N [(-x_i'sin\delta - y_i'cos\delta)(x_i'sin\delta + y_i'cos\delta) + (x_i'cos\delta - y_i'sin\delta)(x_i'cos\delta - y_i'sin\delta)] \tag{13}$$

When $(S_{xy})' = 0$,

$$\sum_1^N (x_i'sin\delta + y_i'cos\delta)^2 = \sum_1^N (x_i'cos\delta - y_i'sin\delta)^2 \tag{14}$$

When we compare Equations 8, 9, and 14, we can easily see that $S_{yy} = S_{xx}$ at this time. Thus, when S_{xx} reaches extreme values, S_{xy} gets to zero. When S_{xy} reaches extreme values, S_{xx} equates S_{yy}.

Now let's deal with the solutions of $(S_{xx})' = 0$. Due to the circumferential periodicity of δ, we need only consider $(-\pi, \pi)$. As the data has been centered, only $(-\pi/2, \pi/2)$ needs to be included, in which $cos\theta \neq 0$. According to Equation 11,

$$-2\sum_{1}^{N}(x_i'cos\delta - y_i'sin\delta)(x_i'sin\delta + y_i'cos\delta) = 0$$

$$\sum_{1}^{N}(x_i' - y_i'tan\delta)(x_i'tan\delta + y_i') = 0$$

$$\sum_{1}^{N}(x_i'^2tan\delta + x_i'y_i' - x_i'y_i'tan^2\delta - y_i'^2tan\delta) = 0$$

$$\sum_{1}^{N}x_i'y_i'tan^2\delta - \sum_{1}^{N}(x_i'^2 - y_i'^2)tan\delta - \sum_{1}^{N}x_i'y_i' = 0$$

$$tan^2\delta - \sum_{1}^{N}\frac{x_i'^2 - y_i'^2}{x_i'y_i'}tan\delta - 1 = 0 \tag{15}$$

The discriminant of Equation 15

$$\Delta = (-\sum_{1}^{N}\frac{x_i'^2 - y_i'^2}{x_i'y_i'})^2 + 4 > 0$$

Equation 15 has two unequal roots, $tan\delta_1$ and $tan\delta_2$, one of which corresponds to maximum S_{xx} and the other corresponds to minimum S_{xx}. According to Vieta's Theorem, $tan\delta_1 tan\delta_2 = -1$. Within $(-\pi/2, \pi/2)$, δ_1 and δ_2 are $\pi/2$ apart.

Briefing Equations 8-15, we find that as δ changes within $(-\pi/2, \pi/2)$, when S_{xx} reaches maximum, S_{yy} gets to minimum and S_{xy} gets to zero; when S_{xx} reaches minimum, S_{yy} gets to maximum and S_{xy} gets to zero; the δs are $\pi/2$ apart for S_{xx} maximum and minimum.

Now we consider the solution of Equation 14.

$$\sum_{1}^{N}(x_i'tan\delta + y_i')^2 - \sum_{1}^{N}(x_i' - y_i'tan\delta)^2 = 0$$

$$\sum_{1}^{N}(x_i'^2 - y_i'^2)tan^2\delta + 4\sum_{1}^{N}x_i'y_i'tan\delta - \sum_{1}^{N}(x_i'^2 - y_i'^2) = 0$$

$$tan^2\delta + 4\sum_{1}^{N}\frac{x_i'y_i'}{x_i'^2 - y_i'^2}tan\delta - 1 = 0 \tag{16}$$

The discriminant of Equation 16

$$\Delta = (4\sum_{1}^{N}\frac{x_i'y_i'}{x_i'^2 - y_i'^2})^2 + 4 > 0$$

Equation 16 has two unequal roots, $tan\delta_1'$ and $tan\delta_2'$, one of which corresponds to maximum S_{xy} and the other corresponds to minimum S_{xy}. According to Vieta's Theorem, $tan\delta_1' tan\delta_2' = -1$. Within $(-\pi/2, \pi/2)$, δ_1' and δ_2' are $\pi/2$ apart.

With the quadratic formula, we can get the roots of Equations 15 and 16. Here I present the Python script PCA_example_1.py for doing this job. Also, the script shows the dynamic process of rotating during which changes of S_{xx}, S_{yy}, and S_{xy} can be clearly seen (Figure 2).

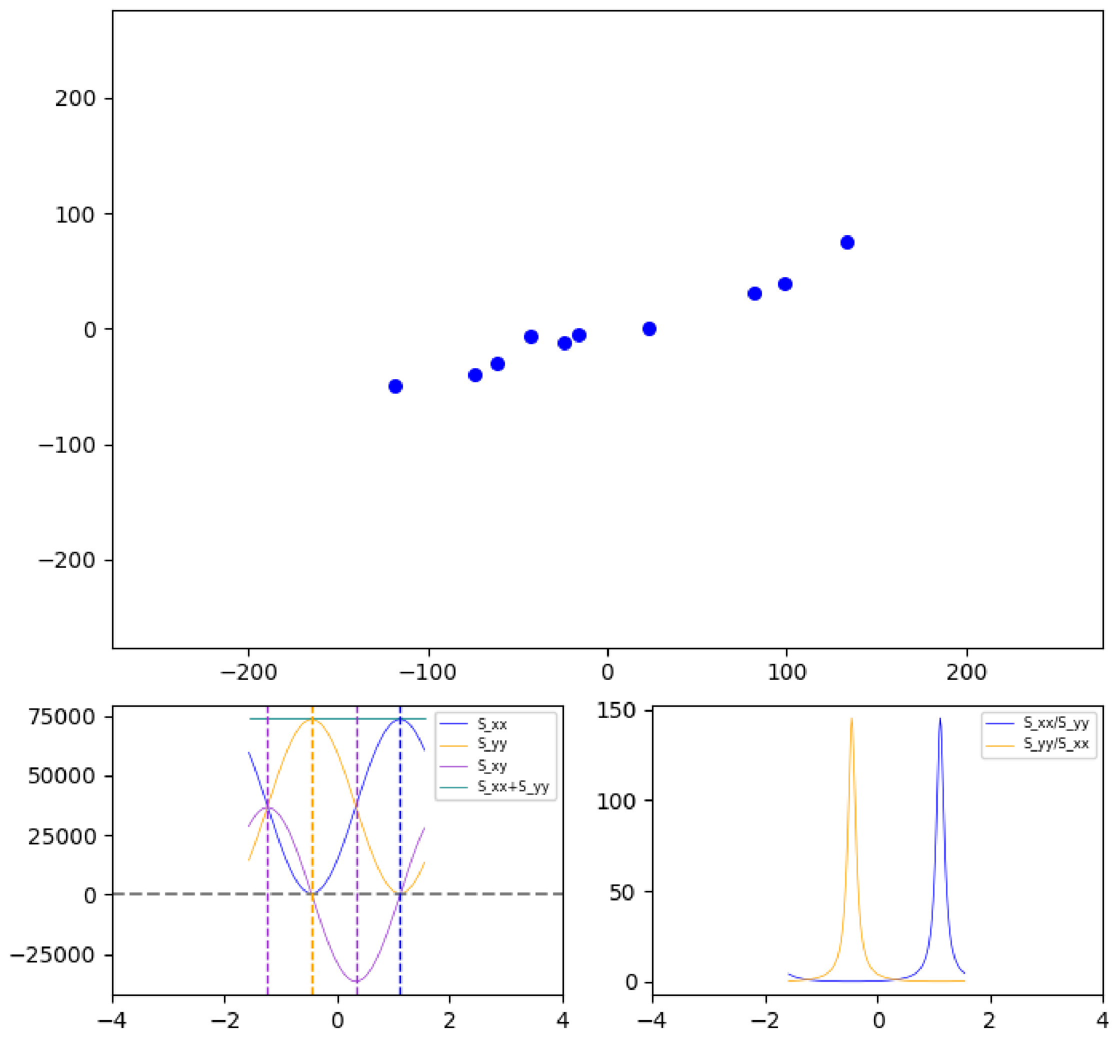

Figure 2: Changes of S_{xx}, S_{yy}, and S_{xy}during rotation

Basis change for data matrix

I have shown how to understand the process of finding the best direction of a dataset by rotation. Next, I would like to explain the process with matrix and change of basis. The dataset *Data* is represented by a matrix, with each line for one point. The matrix is then centered to matrix A.

$$Data = \begin{bmatrix} 58 & -88 \\ 13 & 18 \\ 26 & -29 \\ -26 & 111 \\ -6 & 55 \\ -16 & 67 \\ 103 & -138 \\ 67 & -104 \\ 20 & 10 \\ 18 & 36 \end{bmatrix} \xrightarrow{-column\ mean} \begin{bmatrix} 32.3 & -81.8 \\ -12.7 & 24.2 \\ 0.3 & -22.8 \\ -51.7 & 117.2 \\ -31.7 & 61.2 \\ -41.7 & 73.2 \\ 77.3 & -131.8 \\ 41.3 & -97.8 \\ -5.7 & 16.2 \\ -7.7 & 42.2 \end{bmatrix} = A$$

Then conduct change of basis,

$$\begin{bmatrix} 32.3 & -81.8 \\ -12.7 & 24.2 \\ 0.3 & -22.8 \\ -51.7 & 117.2 \\ -31.7 & 61.2 \\ -41.7 & 73.2 \\ 77.3 & -131.8 \\ 41.3 & -97.8 \\ -5.7 & 16.2 \\ -7.7 & 42.2 \end{bmatrix} \begin{bmatrix} cos\delta & sin\delta \\ -sin\delta & cos\delta \end{bmatrix} = AV$$

The matrix AV is still 10×2. At this time, the S_{xx}, S_{xy}, S_{yx}, and S_{yy} can be represented by matrix $(AV)^{\mathrm{T}}AV$. We have $(AV)^{\mathrm{T}}AV = V^{\mathrm{T}}A^{\mathrm{T}}AV = V^{\mathrm{T}}(A^{\mathrm{T}}A)V$. For a dataset, the A is fixed. Consequently, $A^{\mathrm{T}}A$ is fixed. However, V is changeable. Let's caculate $A^{\mathrm{T}}A$.

$$A^{\mathrm{T}}A = \begin{bmatrix} 14394.1 & -28652.6 \\ -28652.6 & 59615.6 \end{bmatrix} \tag{17}$$

The $A^{\mathrm{T}}A$ is a symmetric matrix, which can be diagonalized. With the eigenvectors and eigenvalues of $A^{\mathrm{T}}A$, we get

$$\begin{aligned} A^{\mathrm{T}}A &= \begin{bmatrix} 14394.1 & -28652.6 \\ -28652.6 & 59615.6 \end{bmatrix} \\ &= \begin{bmatrix} -0.43618794 & -0.89985559 \\ 0.89985559 & -0.43618794 \end{bmatrix} \begin{bmatrix} 73504.40482362 & 0 \\ 0 & 505.29517638 \end{bmatrix} \begin{bmatrix} -0.43618794 & 0.89985559 \\ -0.89985559 & -0.43618794 \end{bmatrix} \\ &= Q\Lambda Q^{\mathrm{T}} \end{aligned} \tag{18}$$

In Equation 18, the column vectors of Q are eigenvectors of $A^{\mathrm{T}}A$; the diagonal elements in the diagonal matrix Λ are the eigenvalues corresponding to the eigenvectors of $A^{\mathrm{T}}A$; Q is an orthogonal matrix satisfying $Q^{-1} = Q^{\mathrm{T}}$. Then we have

$$(AV)^{\mathrm{T}}AV = V^{\mathrm{T}}A^{\mathrm{T}}AV = V^{\mathrm{T}}(A^{\mathrm{T}}A)V = V^{\mathrm{T}}(Q\Lambda Q^{\mathrm{T}})V = (V^{\mathrm{T}}Q)\Lambda(Q^{\mathrm{T}}V) \tag{19}$$

Now look carefully at the far right of Equation 19. Λ is a diagonal matrix, which behaves well. Q is fixed while V can be changed. Now is the most critical time. If we select V carefully [3], which satisfies $Q^{\mathrm{T}}V = I$, namely, $V = Q$, then $(Q^{\mathrm{T}}V)^{\mathrm{T}} = V^{\mathrm{T}}Q = I$. Then Equation 19 becomes

$$(AV)^{\mathrm{T}}AV = I\Lambda I = \Lambda \tag{20}$$

It looks neat. Therefore, for the dataset that has undergone such a basis change, $S_{xx} = \lambda_1 = \Lambda(1,1) = 73504.40482362$, $S_{yy} = \lambda_2 = \Lambda(2,2) = 505.29517638$, $S_{xy} = S_{yx} = 0$. Such results are the same as with those of the rotation method that we have done previously (see the output of Python script PCA_example_1.py).

Since rotation can be regarded as change of basis, we compare the basis via rotation with the basis via careful selection of $V = Q$. For Equation 19, when $V = Q$, the basis is V and the basis vectors are $[-0.43618794 \quad -0.8998555]$ and $[0.89985559 \quad -0.43618794]$, which are shown with oliver axes in Figure 3. Previously, we get the δ and corresponding basis vectors in the rotation matrix are $[0.43618794 \quad 0.89985559]$ and $[-0.89985559 \quad 0.43618794]$, which is represented by the magenta axes in Figure 3.

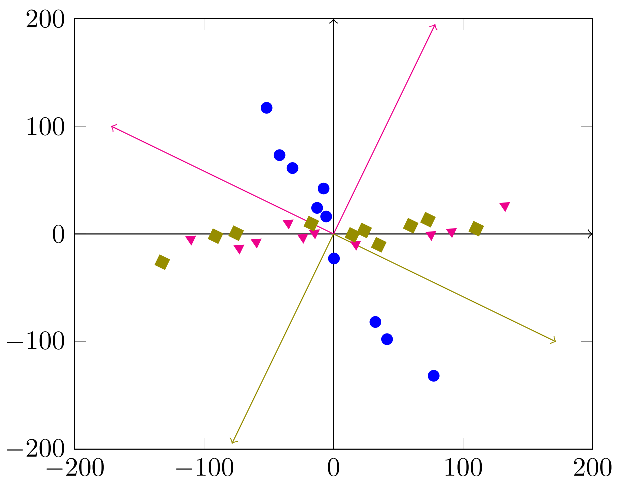

Figure 3: rotation and basis change of dataset

Obviously, the directions of the basis vectors are opposite. The periodicity of Equation 15 is π. If we add a π to the δ, then we get the basis vectors the same as those of V. Or, we can change the signs of basis vectors in V to get the same basis vectors as those of rotation matrix. Here is the proof. According to Equation 18,

$$\begin{aligned}
A^{\mathrm{T}}A &= Q\Lambda Q^{\mathrm{T}} \\
&= \begin{bmatrix} \boldsymbol{q_1} & \boldsymbol{q_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{q_1^{\mathrm{T}}} \\ \boldsymbol{q_2^{\mathrm{T}}} \end{bmatrix} \\
&= \lambda_1 \boldsymbol{q_1} \boldsymbol{q_1^{\mathrm{T}}} + \lambda_2 \boldsymbol{q_2} \boldsymbol{q_2^{\mathrm{T}}} \\
&= \lambda_1(-\boldsymbol{q_1})(-\boldsymbol{q_1^{\mathrm{T}}}) + \lambda_2(-\boldsymbol{q_2})(-\boldsymbol{q_2^{\mathrm{T}}}) \\
&= \begin{bmatrix} -\boldsymbol{q_1} & -\boldsymbol{q_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} -\boldsymbol{q_1^{\mathrm{T}}} \\ -\boldsymbol{q_2^{\mathrm{T}}} \end{bmatrix}
\end{aligned} \tag{21}$$

It should be noted that the rules of block multiplication for matrices is used here. We shall need to apply the rule once more when prove the diagonalizability of real symmetric matrices later. Thus, I have shown that sign changes of any column vectors in Q, do not change $A^{\mathrm{T}}A$. If signs of column vectors in Q are changed, due to $V = Q$, the basis vectors in V are changed accordingly. However, these operations do not change Λ.

A slightly more complex example of PCA

We have known how to find the direction of greatest variance in a two-dimensional plane. The dimension of example dataset is only two, which are the simplest. The method of rotation and the method of basis change are pretty much the same, in terms of interpretability and computational complexity. However, the datasets we meet with in reality are more often than not high-dimensional. For these datasets, it is easier for us to understand and operate by method of basis change. Let's consider a multi-dimensional example, which is the Iris dataset from scikit-learn [4]. The sample size is 50, and each individual contains 4 features. Therefore the original dataset can be represented with a 150×4 matrix.

The dataset is centered, and then the PCA process in the scikit-learn is used to get the new basis $V1$. We also calculate the new basis $V2$ according to the method of basis change (see Python script PCA_example_2.py). The centered data is in the 4-dimensional coordinate system where each point has a coordinate (x, y, z, w). The sum of S_{xx}, S_{yy}, S_{zz}, and S_{ww} is the same before and after basis change.

Let's see the results of scikitlearn and basis change.

$$V1 = \begin{bmatrix} 0.36138659 & 0.65658877 & -0.58202985 & -0.31548719 \\ -0.08452251 & 0.73016143 & 0.59791083 & 0.3197231 \\ 0.85667061 & -0.17337266 & 0.07623608 & 0.47983899 \\ 0.3582892 & -0.07548102 & 0.54583143 & -0.75365743 \end{bmatrix}$$

$$V2 = Q = \begin{bmatrix} -0.36138659 & 0.65658877 & 0.58202985 & 0.31548719 \\ 0.08452251 & 0.73016143 & -0.59791083 & -0.3197231 \\ -0.85667061 & -0.17337266 & -0.07623608 & -0.47983899 \\ -0.3582892 & -0.07548102 & -0.54583143 & 0.75365743 \end{bmatrix}$$

According Equation 21, the signs of column vectors in Q can be changed. After change the signs of columns 1, 3, and 4 in $V2$, $V1 = V2'$. We go on to compare the singular values from scikit-learn and basis change, and the results are the same (see output of PCA_example_2.py). The square of singular value is the λ in diagonal matrix Λ. In this example, $\lambda_1 = 630.0080142$, $\lambda_2 = 36.15794144$, $\lambda_3 = 11.65321551$, $\lambda_4 = 3.55142885$; $\sum_1^4 \lambda_i = 681.3706000000001$ and $\lambda_1 + \lambda_2$ account for 0.977. This means that most of the variance is contributed by λ_1 and λ_2, which correspond to the 1st and 2nd columns in the data after basis change. Therefore, we can keep the first two and discard the rest columns in the data after basis change, which simplifies the data while retains most of the information in original data.

Till now we have two examples, in which we look for the direction of the greatest variance, the direction of the second greatest variance, and so on. This process is PCA. PCA is an unsupervised data dimensionality reduction method, using SVD to perform basis change for the data. The variance on the first basis vector of the new basis is the largest; the variance on the second basis vector of the new basis is the second largest, and so on. Meanwhile, according to the eigenvalues, we can decide which columns in the data after basis change are to be kept so as to achieve data dimensionality reduction.

Real symmetric matrices are diagonalizable

The diagonalizability of real symmetric matrices is the foundation of PCA by SVD. We need to understand the diagonalizability before we are able to use PCA confidently. The proof is divided into these steps. (1) A real symmetric matrix of order n has n eigenvalues (including repeated eigenvalues); (2) All eigenvalues of a real symmetric matrix are real numbers; (3) Eigenvectors with unequal eigenvalues are orthogonal; (4) Eigenvectors with equal eigenvalues are also orthogonal.

For the above four steps, (1) is guaranteed by the Fundamental Theorem of Algebra; (2) and (3) can refer to Gilbert Strang's *Introduction to Linear Algebra* [5]. The more troublesome thing is the proof of (4). Two proofs are provided in the book [5]. One uses Schur's theorem and the proof given has been detailed. The other is to solve the eigenvalues and eigenvectors one by one for a real symmetric matrix, which has not been elaborated on therein. I will go slowly here and make the proof understood once for all.

Before going, we have (i) Any n-dimensional nonzero vector can start from it to construct an orthogonal matrix of order n; (ii) The product of two orthogonal matrices is also an orthogonal matrix. For (i), we can do easily with Gram-Schmdit. (ii) is also evident.

$$(Q_1Q_2)^{\mathrm{T}}(Q_1Q_2) = Q_2^{\mathrm{T}}Q_1^{\mathrm{T}}Q_1Q_2 = Q_2^{\mathrm{T}}IQ_2 = Q_2^{\mathrm{T}}Q_2 = I$$

Now let's start. Suppose S_n is a real symmetric matrix of order n. The solution equation for its eigenvalues is a univariate nth degree equation with n real roots, possibly including repeated roots. From these n real roots, we can find the maximum root, λ_1, and get the corresponding eigenvecotor $\boldsymbol{q_1^n}$. By (i) previously, we can start from $\boldsymbol{q_1^n}$ to construct an orthogonal matrix of order n, $[\boldsymbol{q_1^n} \;\; \boldsymbol{q_2^n} \cdots \boldsymbol{q_n^n}]$, designated as Q_1. The $\boldsymbol{q_1^n}$ is orthogonal to $\boldsymbol{q_2^n}, \boldsymbol{q_3^n}, \cdots, \boldsymbol{q_n^n}$, and $(Q_1^{\mathrm{T}}S_nQ_1)^{\mathrm{T}} = Q_1^{\mathrm{T}}S_n^{\mathrm{T}}Q_1^{\mathrm{T}\mathrm{T}} = Q_1^{\mathrm{T}}S_nQ_1$.

We have

$$
\begin{aligned}
Q_1^{\mathrm{T}} S_n Q_1 &= \begin{bmatrix} (\boldsymbol{q_1^n})^{\mathrm{T}} \\ (\boldsymbol{q_2^n})^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{q_n^n})^{\mathrm{T}} \end{bmatrix} S_n \begin{bmatrix} \boldsymbol{q_1^n} & \boldsymbol{q_2^n} & \cdots & \boldsymbol{q_n^n} \end{bmatrix} \\
&= \begin{bmatrix} (\boldsymbol{q_1^n})^{\mathrm{T}} \\ (\boldsymbol{q_2^n})^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{q_n^n})^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} S_n\boldsymbol{q_1^n} & S_n\boldsymbol{q_2^n} & \cdots & S_n\boldsymbol{q_n^n} \end{bmatrix} \\
&= \begin{bmatrix} (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_1^n} & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_1^n} & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_1^n} & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \end{bmatrix} \\
&= \begin{bmatrix} (\boldsymbol{q_1^n})^{\mathrm{T}} \lambda_1 \boldsymbol{q_1^n} & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ (\boldsymbol{q_2^n})^{\mathrm{T}} \lambda_1 \boldsymbol{q_1^n} & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{q_n^n})^{\mathrm{T}} \lambda_1 \boldsymbol{q_1^n} & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \end{bmatrix} \\
&= \begin{bmatrix} \lambda_1 & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_1^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ 0 & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \end{bmatrix} \\
&= \left[\begin{array}{c:ccc} \lambda_1 & 0 & \cdots & 0 \\ \hdashline 0 & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_2^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_2^n} & \cdots & (\boldsymbol{q_n^n})^{\mathrm{T}} S_n \boldsymbol{q_n^n} \end{array}\right] \\
&= \begin{bmatrix} \lambda_1 & \boldsymbol{0} \\ \boldsymbol{0} & S_{n-1} \end{bmatrix} \qquad (22)
\end{aligned}
$$

In Equation 22, S_{n-1} is a real symmetric matrix. Now the right side of Equation 22 has λ_1 in its diagonal. We still have other eigenvalues that need to be placed on the diagonal. The advantage now is that S_{n-1} remains a real symmetric matrix, but with one less order than the original, only order n-1. For S_{n-1}, the same method can be applied to find its maximum eigenvalue and corresponding eigenvector. In this case the eigenvector of S_{n-1} is n-1 dimensional. Therefore, the $(Q_1^{n-1})^{\mathrm{T}}$ and Q_1^{n-1}, which are placed on both sides of S_{n-1}, are orthogonal matrices of order n-1. To keep the dimension consistent with S_n, we need to put Q_1^{n-1} into the nth order matrix. Here we use rules of block multiplication for matrices,

$$
\left[\begin{array}{c:c} \boldsymbol{I} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{A} \end{array}\right] \left[\begin{array}{c:c} \boldsymbol{I} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{B} \end{array}\right] = \left[\begin{array}{c:c} \boldsymbol{I} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{AB} \end{array}\right] \qquad (23)
$$

Equation 23 shows that the multiplication can be regarded as A acts on B. Therefore, a new round of operation upon Equation 22 is

$$
Q_2^{\mathrm{T}} Q_1^{\mathrm{T}} S_n Q_1 Q_2 = \left[\begin{array}{c:c:c} \lambda_1 & 0 & \boldsymbol{0} \\ \hdashline 0 & \lambda_2 & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{0} & S_{n-2} \end{array}\right] \qquad (24)
$$

where

$$
Q_2 = \left[\begin{array}{c:cccc} 1 & 0 & 0 & \cdots & 0 \\ \hdashline \boldsymbol{0} & \boldsymbol{q_1^{n-1}} & \boldsymbol{q_2^{n-1}} & \cdots & \boldsymbol{q_{n-1}^{n-1}} \end{array}\right]
$$

Obviously, Q_2 is an orthogonal matrix. The maximum eigenvalue of S_{n-1} is λ_2, and the corresponding eigenvector is $\boldsymbol{q_1^{n-1}}$. Continue as described above, we have

$$\begin{aligned} Q_{n-1}^{\mathrm{T}} \cdots Q_2^{\mathrm{T}} Q_1^{\mathrm{T}} S_n Q_1 Q_2 \cdots Q_{n-1} &= \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & S_1 \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \end{aligned} \tag{25}$$

Due to (i) previously, the left side of Equation 25 can be rearranged.

$$\begin{aligned} Q_{n-1}^{\mathrm{T}} \cdots Q_2^{\mathrm{T}} Q_1^{\mathrm{T}} S_n Q_1 Q_2 \cdots Q_{n-1} &= (Q_{n-1}^{\mathrm{T}} \cdots Q_2^{\mathrm{T}} Q_1^{\mathrm{T}}) S_n (Q_1 Q_2 \cdots Q_{n-1}) \\ &= (Q_1 Q_2 \cdots Q_{n-1})^{\mathrm{T}} S_n (Q_1 Q_2 \cdots Q_{n-1}) \end{aligned} \tag{26}$$

For the right side of Equation 26, let $Q_1 Q_2 \cdots Q_{n-1} = Q$, then Q is an orthogonal matrix. The Equation 26 can be written as

$$Q_{n-1}^{\mathrm{T}} \cdots Q_2^{\mathrm{T}} Q_1^{\mathrm{T}} S_n Q_1 Q_2 \cdots Q_{n-1} = Q^{\mathrm{T}} S_n Q \tag{27}$$

Substitute into Equation 25, we have

$$Q^{\mathrm{T}} S_n Q = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

$$S_n Q = Q\Lambda \tag{28}$$

$$S_n = Q\Lambda Q^{\mathrm{T}} \tag{29}$$

Equation 28 proves the column vectors of Q are eigenvectors of S_n. The number of eigenvectors is n and eigenvectors are orthogonal to each other. Equation 29 proves real matrices are diagonalizable. ■

Now we can better understand PCA by SVD. The standardized data with mean zero and variance one is stored in a matrix A. The shape of A is $m \times n$, where m is sample size and n is data dimension. Then $A^{\mathrm{T}} A$ is a real symmetric matrix. By Equation 29, $A^{\mathrm{T}} A = Q\Lambda Q^{\mathrm{T}}$. Let $V = Q$, and conduct basis change to A with V. Then we have

$$\begin{aligned} (AV)^{\mathrm{T}}(AV) &= V^{\mathrm{T}} A^{\mathrm{T}} A V \\ &= V^{\mathrm{T}} (A^{\mathrm{T}} A) V \\ &= V^{\mathrm{T}} (Q\Lambda Q^{\mathrm{T}}) V \\ &= (V^{\mathrm{T}} Q) \Lambda (Q^{\mathrm{T}} V) \\ &= (Q^{\mathrm{T}} Q) \Lambda (Q^{\mathrm{T}} Q) \\ &= \Lambda \end{aligned} \tag{30}$$

If all elements on diagonal of Λ in Equation 30 are positive, then

$$
\begin{aligned}
AV &= [A\boldsymbol{v_1} \quad A\boldsymbol{v_2} \cdots X\boldsymbol{v_n}] & (31)\\
&= [\frac{A\boldsymbol{v_1}}{\sqrt{\lambda_1}} \quad \frac{A\boldsymbol{v_2}}{\sqrt{\lambda_2}} \cdots \frac{A\boldsymbol{v_n}}{\sqrt{\lambda_n}}] \begin{bmatrix} \sqrt{\lambda_1} & & & \\ & \sqrt{\lambda_2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_n} \end{bmatrix} \\
&= [\boldsymbol{u_1} \quad \boldsymbol{u_2} \cdots \boldsymbol{u_n}] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} & (32)\\
&= U\Sigma & (33)\\
A &= U\Sigma V^{\mathrm{T}} & (34)
\end{aligned}
$$

Obviously, the column vectors in the right side of Equation 31 are orthogonal to each other. Also, the column vectors in U in Equation 33 are orthogonal to each other and σ_1, σ_2, $\cdots$, σ_n are called sigular values. Equation 33 is PCA and Equation 34 is SVD. ■

Discussion

Linear algebra is an important mathematical foundation of machine learning, and PCA is not only one of the common algorithms of machine learning, but also an important content of linear algebra. Compared with other materials that introduce PCA, this paper explains PCA from easy to difficult, by examples, and with numbers and shapes combined. The writing is easy to understand and the requirements for mathematical background are few. It would be my pleasure if you could lay a solid foundation and feel fun for machine learning.

Methods

Two Python scripts are provided to help understand PCA. The Pytohn version is 3.10.4 and the packages needed are numpy 1.21.6 [6], matplotlib 3.5.2 [7], and sklearn 0.0 [4]. The appendix of reference [3] includes some of the basics of linear algebra that are important for understanding PCA by SVD.

Code availability

The Python scripts are available at https://github.com/shiqiang-lin/understanding_pca.

References

[1] Jake Lever, Martin Krzywinski, and Naomi Altman. Principal component analysis. *Nature Methods*, 14(7):641–642, Jul 2017.

[2] Michael Greenacre, Patrick J. F. Groenen, Trevor Hastie, Alfonso Iodice D'Enza, Angelos Markos, and Elena Tuzhilina. Principal component analysis. *Nature Reviews Methods Primers*, 2(1):100, Dec 2022.

[3] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Gilbert Strang. *Introduction to Linear Algebra*. CUP, 5 edition, 2021.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren

Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.