Understanding principal component analysis

Authors: Lin, Shiqiang, Lin, Shiqiang

Date: 2023-09-15T00:00:00+00:00

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.

Full Text

Preamble

Principal Component Analysis (PCA) is a frequently used machine learning method. In this paper, we explain the PCA operation through examples with Python program illustrations. A proof of the diagonalizability of real symmetric matrices is also included, which may help readers understand the mathematics behind PCA.

Principal Component Analysis performs linear combinations of variables to obtain a few principal components that are mutually uncorrelated. Most information from the original data is retained in these principal components. As a method of data dimensionality reduction, PCA plays an important role in data processing [**?**, **?**]. The principle of PCA is singular value decomposition (SVD), which involves several crucial concepts of linear algebra such as eigenvalues, eigenvectors, basis change, orthogonality, symmetric matrices, and matrix multiplication. 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 programs to explain PCA step by step, culminating in the mathematical proof of PCA and SVD. It is hoped that this interpretation of PCA will open the door to machine learning for numerous enthusiasts.

Results

Euler's Formula and Rotation of Points in a Two-Dimensional Plane

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

$$e^{i(\theta+\delta)} = e^{i\theta}e^{i\delta} = (\cos\theta + i\sin\theta)(\cos\delta + i\sin\delta) = \cos\theta\cos\delta + i\cos\theta\sin\delta + i\sin\theta\cos\delta + i^2\sin\theta\sin\delta = (\cos\theta\cos\delta -$$

Therefore, the coordinates of M' become $(\cos\theta\cos\delta - \sin\theta\sin\delta, \cos\theta\sin\delta + \sin\theta\cos\delta)$. For a point $N(r\cos\theta, r\sin\theta)$, where $r \in \mathbb{R}$, rotating by δ to N' changes its coordinates to $(r(\cos\theta\cos\delta - \sin\theta\sin\delta), r(\cos\theta\sin\delta + \sin\theta\cos\delta))$, which can be written as $(r\cos\theta\cos\delta - r\sin\theta\sin\delta, r\cos\theta\sin\delta + r\sin\theta\cos\delta)$. Let $r\cos\theta = a$ and $r\sin\theta = b$; then the coordinates of N' are $(a\cos\delta - b\sin\delta, a\sin\delta + b\cos\delta)$.

The rotation can be expressed using matrices. From the perspective of rotation, the left side of the rotation equation rotates point (a, b) by δ and the right side shows the result of this operation. From the perspective of basis change, the basis vectors of $[a\ b]^T$ change from $[1\ 0]^T$ and $[0\ 1]^T$ to $[\cos\delta\ \sin\delta]^T$ and $[-\sin\delta\ \cos\delta]^T$, 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 $x_i, y_i \in \mathbb{R}$ for $i = 1, 2, \cdots, N$, then the data can be visualized with a scatter diagram in the rectangular coordinate system. However, the center of these points is not necessarily at the origin $(0, 0)$. Therefore, we calculate the means of the x and y directions, which are $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, 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 calculate the variances of the original data, which are $\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N} {x'}_i^2$ and $\sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N} {y'}_i^2$.

Now consider a small dataset:

$$\text{Data} = \{(58, -88), (13, 18), (26, -29), (-26, 111), (-6, 55), (-16, 67), (103, -138), (67, -104), (20, 10), (18, 36)\}$$

which is shown with gray points in Figure 1 [Figure 1: see original paper]. The blue points indicate the centered data Data_new.

The dataset has 10 points, which are roughly distributed along a line in the scatterplot. In the direction of this line, the variance of projections is large, whereas in the perpendicular direction, the variance is small. Considering the structural

characteristics of the data, we can simplify it by reducing the two-dimensional data to one-dimensional data: find the direction with the largest variance and project the data points onto this direction. In the other (perpendicular) direction, the point values do not change much and can be discarded. The question is: how do we find the direction with the greatest variance?

Let us start with a general derivation and then examine a specific example. The sample size N is a fixed value here. Thus, we focus on S_{xx}, S_{yy}, and S_{xy} of the data:

$$S_{xx} = \sum_{i=1}^{N} {x'}_i^2, \quad S_{yy} = \sum_{i=1}^{N} {y'}_i^2, \quad S_{xy} = \sum_{i=1}^{N} x'_i y'_i$$

To find the direction of greatest variance, we rotate the centered data according to the rotation equation. 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 into the equations for S_{xx}, S_{yy}, and S_{xy}:

$$S_{xx} = \sum_{i=1}^{N} (x'_i \cos\delta - y'_i \sin\delta)^2$$

$$S_{yy} = \sum_{i=1}^{N} (x'_i \sin\delta + y'_i \cos\delta)^2$$

$$S_{xy} = \sum_{i=1}^{N} (x'_i \cos\delta - y'_i \sin\delta)(x'_i \sin\delta + y'_i \cos\delta)$$

Comparing these equations, we find that $S_{xx} + S_{yy}$ remains unchanged before and after rotation. This is easy to understand, as rotation does not change the distance between a point and the origin. It can also be shown by calculating the distance from a point to the origin directly with the distance formula.

Now consider the expression for S_{xx}. To find its extreme values, we calculate the first derivative:

$$\begin{aligned}(S_{xx})' &= \sum_{i=1}^{N} 2(x'_i \cos\delta - y'_i \sin\delta)(x'_i \cos\delta - y'_i \sin\delta)' \\ &= \sum_{i=1}^{N} 2(x'_i \cos\delta - y'_i \sin\delta)(-x'_i \sin\delta - y'_i \cos\delta) \\ &= -2\sum_{i=1}^{N} (x'_i \cos\delta - y'_i \sin\delta)(x'_i \sin\delta + y'_i \cos\delta)\end{aligned}$$

Comparison shows that $S_{xy} = 0$ when $(S_{xx})' = 0$. Similarly, calculating the first derivative of S_{yy}:

$$(S_{yy})' = \sum_{i=1}^{N} 2(x'_i \sin\delta + y'_i \cos\delta)(x'_i \sin\delta + y'_i \cos\delta)'$$

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

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

It can be shown that $S_{xy} = 0$ when $(S_{yy})' = 0$. To find the extreme values of S_{xy}, we calculate its first derivative:

$$(S_{xy})' = \sum_{i=1}^{N} [(x_i' \cos\delta - y_i' \sin\delta)'(x_i' \sin\delta + y_i' \cos\delta) + (x_i' \cos\delta - y_i' \sin\delta)(x_i' \sin\delta + y_i' \cos\delta)']$$

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

$$[(-x_i' \sin\delta - y_i' \cos\delta)(x_i' \sin\delta + y_i' \cos\delta) + (x_i' \cos\delta - y_i' \sin\delta)(x_i' \cos\delta - y_i' \sin\delta)] = 0$$

which simplifies to:

$$(x_i' \sin\delta + y_i' \cos\delta)^2 = (x_i' \cos\delta - y_i' \sin\delta)^2$$

When we compare the equations for S_{xx}, S_{yy}, and the condition above, we can easily see that $S_{yy} = S_{xx}$ at this point. Thus, when S_{xx} reaches extreme values, S_{xy} becomes zero. When S_{xy} reaches extreme values, S_{xx} equals S_{yy}.

Now let us solve $(S_{xx})' = 0$. 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 considered, where $\cos\theta \neq 0$. According to the derivative expression:

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

Dividing by $\cos^2\delta$:

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

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

$$\left(\sum_{i=1}^{N} x_i' y_i'\right) \tan^2\delta + \left(\sum_{i=1}^{N} {y'}_i^2 - \sum_{i=1}^{N} {x'}_i^2\right) \tan\delta - \sum_{i=1}^{N} x_i' y_i' = 0$$

Letting $S_{xx} = \sum {x'}_i^2$, $S_{yy} = \sum {y'}_i^2$, and $S_{xy} = \sum x_i' y_i'$, we obtain:

$$S_{xy} \tan^2\delta + (S_{yy} - S_{xx}) \tan\delta - S_{xy} = 0$$

The discriminant of this quadratic equation is $\Delta = (S_{yy} - S_{xx})^2 + 4S_{xy}^2 > 0$. The equation has two unequal roots, $\tan\delta_1$ and $\tan\delta_2$, one corresponding to the maximum S_{xx} and the other to the 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.

Summarizing these results, we find that as δ changes within $(-\pi/2, \pi/2)$: when S_{xx} reaches its maximum, S_{yy} reaches its minimum and S_{xy} becomes zero; when S_{xx} reaches its minimum, S_{yy} reaches its maximum and S_{xy} becomes zero; the δ values for S_{xx} maximum and minimum are $\pi/2$ apart.

Now consider the solution for the condition when S_{xy} reaches its extreme values. Following similar derivations, we obtain:

$$S_{xy}\tan^2\delta + 2(S_{xx} - S_{yy})\tan\delta - S_{xy} = 0$$

The discriminant is $\Delta = 4(S_{xx} - S_{yy})^2 + 4S_{xy}^2 > 0$. This equation has two unequal roots, $\tan\delta_1'$ and $\tan\delta_2'$, one corresponding to the maximum S_{xy} and the other to the 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.

Using the quadratic formula, we can obtain the roots of these equations. Here I present the Python script `PCA_{example}1.py` for performing these calculations. The script also shows the dynamic rotation process, during which changes in S_{xx}, S_{yy}, and S_{xy} can be clearly observed (Figure 2 [Figure 2: see original paper]).

Basis Change for Data Matrix

I have shown how to understand the process of finding the best direction for a dataset through rotation. Next, I will explain the process using matrix operations and basis change. The dataset is represented by a matrix, with each row representing one point. The matrix is then centered to obtain matrix A.

The centered matrix undergoes a change of basis:

$$AV = A\begin{bmatrix}\cos\delta & -\sin\delta \\ \sin\delta & \cos\delta\end{bmatrix}$$

The matrix AV remains 10×2. At this point, S_{xx}, S_{xy}, S_{yx}, and S_{yy} can be represented by the matrix $(AV)^{TAV}$. We have:

$$(AV)^{TAV} = V^{TA^{TAV}} = V^T(A^{TA})V$$

For a dataset, matrix A is fixed, and consequently A^{TA} is fixed. However, V can be changed. Let us calculate A^{TA}:

$$A^{TA} = \begin{bmatrix}14394.1 & -28652.6 \\ -28652.6 & 56895.9\end{bmatrix}$$

The matrix A^{TA} is symmetric and can be diagonalized. Using the eigenvectors and eigenvalues of A^{TA}, we obtain:

$$A^{TA} = \begin{bmatrix} -0.43618794 & -0.89985559 \\ 0.89985559 & -0.43618794 \end{bmatrix} \begin{bmatrix} 73504.40482362 & 0 \\ 0 & 505.29517638 \end{bmatrix} \begin{bmatrix} -0.43618794 & 0.89985559 \\ -0.89985559 & -0.43618794 \end{bmatrix} = Q\Lambda Q$$

In this equation, the column vectors of Q are eigenvectors of A^{TA}; the diagonal elements in the diagonal matrix Λ are the eigenvalues corresponding to the eigenvectors of A^{TA}; and Q is an orthogonal matrix satisfying $Q^{-1} = Q^T$. Then we have:

$$(AV)^{TAV} = V^{TA^{TAV}} = V^T(A^{TA})V = V^T(Q\Lambda Q^T)V = (V^{TQ})\Lambda(Q^{TV})$$

Examining the far right side of this equation, Λ is a diagonal matrix with desirable properties. Q is fixed while V can be changed. The critical step is to select V carefully [**?**] such that $Q^{TV} = I$, namely $V = Q$. Then $(Q^{TV})^T = V^{TQ} = I$, and the equation becomes:

$$(AV)^{TAV} = I\Lambda I = \Lambda$$

This result is elegant. Therefore, for the dataset that has undergone this basis change, $S_{xx} = \lambda_1 = \Lambda(1,1) = 73504.40482362$, $S_{yy} = \lambda_2 = \Lambda(2,2) = 505.29517638$, and $S_{xy} = S_{yx} = 0$. These results are identical to those obtained through the rotation method described previously (see the output of Python script `PCA_{example}1.py`).

Since rotation can be regarded as a change of basis, we compare the basis obtained via rotation with the basis obtained by carefully selecting $V = Q$. For the equation above, when $V = Q$, the basis vectors are $[-0.43618794 \; -0.89985559]^T$ and $[0.89985559 \; -0.43618794]^T$, shown with olive axes in Figure 3 [Figure 3: see original paper]. Previously, we obtained δ and the corresponding basis vectors in the rotation matrix as $[0.43618794 \; 0.89985559]^T$ and $[-0.89985559 \; 0.43618794]^T$, represented by the magenta axes in Figure 3.

Obviously, the directions of the basis vectors are opposite. The periodicity of the rotation equation is π. If we add π to δ, we obtain basis vectors identical to those of V. Alternatively, we can change the signs of basis vectors in V to match those of the rotation matrix. Here is the proof. According to the diagonalization, $A^{TA} = Q\Lambda Q^T = \lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T = \lambda_1(-q_1)(-q_1^T) + \lambda_2(-q_2)(-q_2^T)$. The rules of block matrix multiplication are used here. We will need to apply this rule once more when proving the diagonalizability of real symmetric matrices later. Thus, I have shown that sign changes of any column vectors in Q do not change A^{TA}. If signs of column vectors in Q are changed, due to $V = Q$, the basis vectors in V change accordingly. However, these operations do not change Λ.

A Slightly More Complex Example of PCA

We have learned how to find the direction of greatest variance in a two-dimensional plane. The example dataset had only two dimensions, which is the simplest case. The rotation method and the basis change method are comparable in terms of interpretability and computational complexity for such low-dimensional data. However, real-world datasets are more often high-dimensional. For these datasets, the basis change method is easier to understand and implement. Let us consider a multi-dimensional example: the Iris dataset from scikit-learn [**?**]. The sample size is 150, and each sample contains 4 features. Therefore, the original dataset can be represented as a 150×4 matrix.

The dataset is centered, and then the PCA process in scikit-learn is used to obtain the new basis V_1. We also calculate the new basis V_2 according to the basis change method (see Python script `PCA_{example}2.py`). The centered data exists in a 4-dimensional coordinate system where each point has coordinates (x, y, z, w). The sum $S_{xx} + S_{yy} + S_{zz} + S_{ww}$ remains the same before and after the basis change.

The results from scikit-learn and basis change are:

$$V_1 = \begin{bmatrix} 0.65658877 & -0.58202985 & -0.31548719 & -0.08452251 \\ 0.85667061 & -0.17337266 & 0.54583143 & -0.75365743 \\ -0.07548102 & -0.36138659 & 0.73016143 & -0.59791083 \\ -0.3197231 & -0.85667061 & -0.17337266 & -0.07623608 \end{bmatrix}$$

According to the sign change property, after changing the signs of columns 1, 3, and 4 in V_2, we obtain $V_1 = V_2'$. Comparing the singular values from scikit-learn and basis change yields identical results (see output of `PCA_{example}2.py`). The square of each singular value is the corresponding λ in the diagonal matrix Λ. In this example, $\lambda_1 = 630.0080142$, $\lambda_2 = 36.15794144$, $\lambda_3 = 11.65321551$, $\lambda_4 = 3.55142885$; $\sum_{i=1}^{4} \lambda_i = 681.3706000000001$ and $(\lambda_1 + \lambda_2)/\sum \lambda_i = 0.977$. 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 columns and discard the rest, simplifying the data while retaining most of the information from the original data.

Thus far, we have two examples in which we seek the direction of greatest variance, the direction of the second greatest variance, and so on. This process is PCA. PCA is an unsupervised data dimensionality reduction method that uses SVD to perform basis change on the data. The variance on the first basis vector of the new basis is the largest; the variance on the second basis vector is the second largest, and so on. Meanwhile, according to the eigenvalues, we can decide which columns in the transformed data to keep to achieve dimensionality reduction.

Real Symmetric Matrices are Diagonalizable

The diagonalizability of real symmetric matrices is the foundation of PCA via SVD. We need to understand this property before using PCA confidently. The proof is divided into four 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 distinct eigenvalues are orthogonal; (4) Eigenvectors with equal eigenvalues are also orthogonal.

For these four steps, (1) is guaranteed by the Fundamental Theorem of Algebra; (2) and (3) can be found in Gilbert Strang's *Introduction to Linear Algebra* [**?**]. The more challenging part is proving (4). Two proofs are provided in the book [**?**]. One uses Schur's theorem and is detailed therein. The other solves for eigenvalues and eigenvectors sequentially for a real symmetric matrix but is not elaborated. I will proceed slowly here to make the proof clear.

Before proceeding, we note that: (i) any n-dimensional nonzero vector can be used to construct an orthogonal matrix of order n; (ii) the product of two orthogonal matrices is also orthogonal. For (i), we can use Gram-Schmidt orthogonalization. (ii) is also evident: $(Q_{1Q}2)^T(Q_{1Q}2) = Q_2^T Q_1^T Q_1 Q_2 = Q_2^T Q_2 = I$.

Now let us begin. Suppose S_n is a real symmetric matrix of order n. The characteristic equation for its eigenvalues is a univariate nth-degree polynomial with n real roots, possibly including repeated roots. From these n real roots, we can find the largest root λ_1 and its corresponding eigenvector q_n. By property (i), we can construct an orthogonal matrix $Q_1 = [q_n, q_{n-1}, \cdots, q_1]$ starting from q_n.

We have:

$$Q_1^T S_n Q_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & S_{n-1} \end{bmatrix}$$

where S_{n-1} is a real symmetric matrix of order $n-1$. The right side now has λ_1 on its diagonal. We still need to place the remaining eigenvalues on the diagonal. The advantage is that S_{n-1} remains a real symmetric matrix but with order reduced by one. For S_{n-1}, we can apply the same method to find its maximum eigenvalue and corresponding eigenvector. The eigenvector of S_{n-1} is $(n-1)$-dimensional. Therefore, the matrices Q_{n-1} that are placed on both sides of S_{n-1} are orthogonal matrices of order $n-1$. To maintain dimension consistency with S_n, we embed Q_{n-1} into an nth-order matrix using block matrix multiplication rules:

$$\begin{bmatrix} 1 & 0 \\ 0 & Q_{n-1}^T \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & S_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & Q_{n-1} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & Q_{n-1}^T S_{n-1} Q_{n-1} \end{bmatrix}$$

Obviously, $Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & Q_{n-1} \end{bmatrix}$ is an orthogonal matrix. The maximum eigenvalue of S_{n-1} is λ_2, with corresponding eigenvector q_{n-1}. Continuing this process, we

obtain:

$$Q_{n-1}^T \cdots Q_2^T Q_1^T S_n Q_1 Q_2 \cdots Q_{n-1} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$$

By property (ii), the left side can be rearranged:

$$Q_{n-1}^T \cdots Q_2^T Q_1^T S_n Q_1 Q_2 \cdots Q_{n-1} = (Q_1 Q_2 \cdots Q_{n-1})^T S_n (Q_1 Q_2 \cdots Q_{n-1})$$

Let $Q = Q_1 Q_2 \cdots Q_{n-1}$; then Q is an orthogonal matrix. The equation becomes:

$$Q^T S_n Q = \Lambda$$

where Λ is the diagonal matrix of eigenvalues. Therefore:

$$S_n Q = Q\Lambda \quad \text{and} \quad S_n = Q\Lambda Q^T$$

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

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

$$(AV)^T(AV) = V^{TA^{TAV}} = V^T(A^{TA})V = V^T(Q\Lambda Q^T)V = (V^{TQ})\Lambda(Q^{TV}) = (Q^{TQ})\Lambda(Q^{TQ}) = \Lambda$$

If all diagonal elements of Λ are positive, then:

$$AV = [Av_1 \ Av_2 \ \cdots \ Av_n] = [u_1\sigma_1 \ u_2\sigma_2 \ \cdots \ u_n\sigma_n] = U\Sigma$$

where $U = [u_1 \ u_2 \ \cdots \ u_n]$ and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$. Therefore:

$$A = U\Sigma V^T$$

The column vectors on the right side of Equation 31 are orthogonal to each other. Also, the column vectors in U are orthogonal, and $\sigma_1, \sigma_2, \cdots, \sigma_n$ are called singular values. Equation 33 represents PCA, and Equation 34 represents SVD.

Discussion

Linear algebra is an important mathematical foundation of machine learning, and PCA is not only a common machine learning algorithm but also an important topic in linear algebra. Compared with other materials that introduce PCA, this paper explains PCA from easy to difficult, using examples that combine numerical and visual approaches. The writing is accessible and requires minimal mathematical background. It would be my pleasure if this helps you build a solid foundation and find machine learning enjoyable.

Methods

Two Python scripts are provided to help understand PCA. The Python version is 3.10.4, and the required packages are numpy 1.21.6 [**?**], matplotlib 3.5.2 [**?**], and sklearn 0.0 [**?**]. The appendix of reference [**?**] includes some linear algebra basics that are important for understanding PCA via 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, 5th 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.

Note: Figure translations are in progress. See original paper for figures.