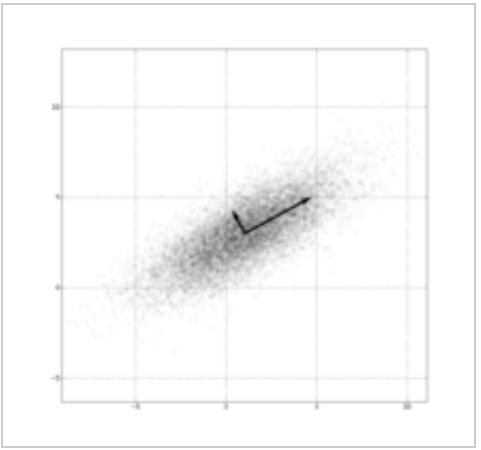
Principal component analysis

From Wikipedia, the free encyclopedia

Principal component analysis (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.



PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.878, 0.478) direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.

Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of $\mathbf{X}$ (Golub and Van Loan, 1983), eigenvalue decomposition (EVD) of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of[1]), Eckart–Young theorem (Harman, 1960), or Schmidt–Mirsky theorem in psychometrics, empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition (Sirovich, 1987), empirical component analysis (Lorenz, 1956), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

PCA was invented in 1901 by Karl Pearson,[2] as an analogue of the principal axes theorem in mechanics; it was later independently developed (and named) by Harold Hotelling in the 1930s.[3] The method is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute.[4] The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).[5]

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its (in some sense; see below) most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

PCA is also related to canonical correlation analysis (CCA). CCA defines coordinate systems that optimally describe the cross-covariance between two datasets while PCA defines a new orthogonal coordinate system that optimally describes variance in a single dataset.[6][7]

Contents

Intuition

PCA can be thought of as fitting an *n*-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipse is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipse, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then, we must orthogonalize the set of eigenvectors, and normalize each to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

It is important to note that this procedure is sensitive to the scaling of the data, and that there is no consensus as to how to best scale the data to obtain optimal results.

Details

PCA is mathematically defined[1] as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a data matrix, **X**, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the *n* rows represents a different repetition of the experiment, and each of the *p* columns gives a particular kind of datum (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of *p*-dimensional vectors of weights or *loadings* $\mathbf{w}_{(k)} = (w_1, \dots, w_p)_{(k)}$ that map each row vector $\mathbf{x}_{(i)}$ of **X** to a new vector of principal component *scores* $\mathbf{t}_{(i)} = (t_1, \dots, t_p)_{(i)}$, given by

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$$

in such a way that the individual variables of **t** considered over the data set successively inherit the maximum possible variance from **x**, with each loading vector **w** constrained to be a unit vector.

First component

The first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \arg\max_{\|\mathbf{w}\|=1} \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \|\mathbf{X}\mathbf{w}\|^2 \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \right\}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \arg\max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

The quantity to be maximised can be recognised as a Rayleigh quotient. A standard result for a symmetric matrix such as $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is that the quotient's maximum possible value is the largest eigenvalue of the matrix, which occurs when ***w*** is the corresponding eigenvector.

With $\mathbf{w}_{(1)}$ found, the first component of a data vector $\mathbf{x}_{(i)}$ can then be given as a score $t_{1(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}\}\ \mathbf{w}_{(1)}$.

Further components

The *k*th component can be found by subtracting the first $k - 1$ principal components from **X**:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X} \mathbf{w}_{(s)} \mathbf{w}_{(s)}^{\mathrm{T}}$$

and then finding the loading vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \|\hat{\mathbf{X}}_{k-1} \mathbf{w}\|^2 \right\} = \arg\max \left\{ \frac{\mathbf{w}^T \hat{\mathbf{X}}_{k-1}^T \hat{\mathbf{X}}_{k-1} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

It turns out that this gives the remaining eigenvectors of $\mathbf{X}^T\mathbf{X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues.

The *k*th principal component of a data vector $\mathbf{x}_{(i)}$ can therefore be given as a score $t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$ in the transformed co-ordinates, or as the corresponding vector in the space of the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}\} \mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$ is the *k*th eigenvector of $\mathbf{X}^T\mathbf{X}$.

The full principal components decomposition of $\mathbf{X}$ can therefore be given as

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

where $\mathbf{W}$ is a *p*-by-*p* matrix whose columns are the eigenvectors of $\mathbf{X}^T\mathbf{X}$

Covariances

$\mathbf{X}^T\mathbf{X}$ itself can be recognised as proportional to the empirical sample covariance matrix of the dataset $\mathbf{X}$.

The sample covariance *Q* between two of the different principal components over the dataset is given by:

$$\begin{aligned} Q(\mathrm{PC}_{(j)}, \mathrm{PC}_{(k)}) &\propto (\mathbf{X}\mathbf{w}_{(j)}) \cdot (\mathbf{X}\mathbf{w}_{(k)}) \\ &= \mathbf{w}_{(j)}^T \mathbf{X}^T \mathbf{X} \mathbf{w}_{(k)} \\ &= \mathbf{w}_{(j)}^T \lambda_{(k)} \mathbf{w}_{(k)} \\ &= \lambda_{(k)} \mathbf{w}_{(j)}^T \mathbf{w}_{(k)} \end{aligned}$$

where the eigenvalue property of $\mathbf{w}_{(k)}$ has been used to move from line 2 to line 3. However eigenvectors $\mathbf{w}_{(j)}$ and $\mathbf{w}_{(k)}$ corresponding to eigenvalues of a symmetric matrix are orthogonal (if the eigenvalues are different), or can be orthogonalised (if the vectors happen to share an equal repeated value). The product in the final line is therefore zero; there is no sample covariance between different principal components over the dataset.

Another way to characterise the principal components transformation is therefore as the transformation to coordinates which diagonalise the empirical sample covariance matrix.

In matrix form, the empirical covariance matrix for the original variables can be written

$$\mathbf{Q} \propto \mathbf{X}^T\mathbf{X} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T$$

The empirical covariance matrix between the principal components becomes

$$\mathbf{W}^T\mathbf{Q}\mathbf{W} \propto \mathbf{W}^T\mathbf{W}\,\mathbf{\Lambda}\,\mathbf{W}^T\mathbf{W} = \mathbf{\Lambda}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues $\lambda_{(k)}$ of $\mathbf{X}^T\mathbf{X}$

($\lambda_{(k)}$ being equal to the sum of the squares over the dataset associated with each component *k*: $\lambda_{(k)} = \Sigma_i\, t_k^2{}_{(i)} = \Sigma_i\, (\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)})^2$)

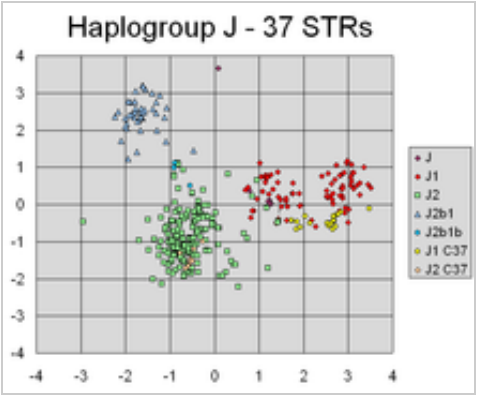
Dimensionality reduction

The faithful transformation $\mathbf{T} = \mathbf{X}\,\mathbf{W}$ maps a data vector $\mathbf{x}_{(i)}$ from an original space of p variables to a new space of p variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L loading vectors, gives the truncated transformation

$$\mathbf{T}_L = \mathbf{X}\mathbf{W}_L$$

where the matrix $\mathbf{T}_L$ now has n rows but only L columns. In other words, PCA learns a linear transformation $t = W^T x, x \in R^p, t \in R^L$, where the columns of $p \times L$ matrix W form an orthogonal basis for the L features (the components of representation t) that are decorrelated.[8] By construction, of all the transformed data matrices with only L columns, this score matrix maximises the variance in the original data that has been preserved, while minimising the total squared reconstruction error $\|\mathbf{T}\mathbf{W}^T - \mathbf{T}_L\mathbf{W}_L^T\|_2^2$ or $\|\mathbf{X} - \mathbf{X}_L\|_2^2$.

Such dimensionality reduction can be a very useful step for visualising and processing high-dimensional datasets, while still retaining as much of the variance in the dataset as possible. For example, selecting $L = 2$ and keeping only the first two principal components finds the two-dimensional plane through the high-dimensional dataset in which the data is most spread out, so if the data contains clusters these too may be most spread out, and therefore most visible to be plotted out in a two-dimensional diagram; whereas if two directions through the data (or two of the original variables) are chosen at random, the clusters may be much less spread apart from each other, and may in fact be much more likely to substantially overlay each other, making them indistinguishable.



A principal components analysis scatterplot of Y-STR haplotypes calculated from repeat-count values for 37 Y-chromosomal STR markers from 354 individuals.
PCA has successfully found linear combinations of the different markers, that separate out different clusters corresponding to different lines of individuals' Y-chromosomal genetic descent.

Similarly, in regression analysis, the larger the number of explanatory variables allowed, the greater is the chance of overfitting the model, producing conclusions that fail to generalise to other datasets. One approach, especially when there are strong correlations between different possible explanatory variables, is to reduce them to a few principal components and then run the regression against them, a method called principal component regression.

Dimensionality reduction may also be appropriate when the variables in a dataset are noisy. If each column of the dataset contains independent identically distributed Gaussian noise, then the columns of $\mathbf{T}$ will also contain similarly identically distributed Gaussian noise (such a distribution is invariant under the effects of the matrix $\mathbf{W}$, which can be thought of as a high-dimensional rotation of the co-ordinate axes). However, with more of the total variance concentrated in the first few principal components compared to the same noise variance, the proportionate effect of the noise is less—the first few components achieve a higher signal-to-noise ratio. PCA thus can have the effect of concentrating much of the signal into the first few principal components, which can usefully be captured by dimensionality reduction; while the later principal components may be dominated by noise, and so disposed of without great loss.

Singular value decomposition

The principal components transformation can also be associated with another matrix factorisation, the singular value decomposition (SVD) of **X**,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T$$

Here **Σ** is a *n*-by-*p* rectangular diagonal matrix of positive numbers $\sigma_{(k)}$, called the singular values of **X**; **U** is an *n*-by-*n* matrix, the columns of which are orthogonal unit vectors of length *n* called the left singular vectors of **X**; and **W** is a *p*-by-*p* whose columns are orthogonal unit vectors of length *p* and called the right singular vectors of **X**.

In terms of this factorisation, the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ can be written

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= \mathbf{W}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{W}^T \\ &= \mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^T\end{aligned}$$

Comparison with the eigenvector factorisation of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ establishes that the right singular vectors **W** of **X** are equivalent to the eigenvectors of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, while the singular values $\sigma_{(k)}$ of **X** are equal to the square roots of the eigenvalues $\lambda_{(k)}$ of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.

Using the singular value decomposition the score matrix **T** can be written

$$\begin{aligned}\mathbf{T} &= \mathbf{X}\mathbf{W} \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} \\ &= \mathbf{U}\mathbf{\Sigma}\end{aligned}$$

so each column of **T** is given by one of the left singular vectors of **X** multiplied by the corresponding singular value.

Efficient algorithms exist to calculate the SVD of **X** without having to form the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, so computing the SVD is now the standard way to calculate a principal components analysis from a data matrix, unless only a handful of components are required.

As with the eigen-decomposition, a truncated *n*-by-*L* score matrix $\mathbf{T}_{\mathrm{L}}$ can be obtained by considering only the first L largest singular values and their singular vectors:

$$\mathbf{T}_L = \mathbf{U}_L\mathbf{\Sigma}_L = \mathbf{X}\mathbf{W}_L$$

The truncation of a matrix **M** or **T** using a truncated singular value decomposition in this way produces a truncated matrix that is the nearest possible matrix of rank *L* to the original matrix, in the sense of the difference between the two having the smallest possible Frobenius norm, a result known as the Eckart–Young theorem [1936].

Further considerations

Given a set of points in Euclidean space, the first principal component corresponds to a line that passes through the multidimensional mean and minimizes the sum of squares of the distances of the points from the line. The second principal component corresponds to the same concept after all correlation with the first principal component has been subtracted from the points. The singular values (in $\mathbf{\Sigma}$) are the square roots of the eigenvalues of the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Each eigenvalue is proportional to the portion of the "variance" (more correctly of the sum of the squared distances of the points from their multidimensional mean) that is correlated with each eigenvector. The sum of all the eigenvalues is equal to the sum of the squared distances of the points from their multidimensional mean. PCA essentially rotates the set of points around their mean in order to align with the principal components. This moves as much of the variance as possible (using an orthogonal transformation) into the first few dimensions. The values in the remaining dimensions, therefore, tend to be small and may be dropped with minimal loss of information (see below). PCA is often used in this manner for dimensionality reduction. PCA has the distinction of being the optimal orthogonal transformation for keeping the subspace that has largest "variance" (as defined above). This advantage, however, comes at the price of greater computational requirements if compared, for example and when applicable, to the discrete cosine transform, and in particular to the DCT-II which is simply known as the "DCT". Nonlinear dimensionality reduction techniques tend to be more computationally demanding than PCA.

PCA is sensitive to the scaling of the variables. If we have just two variables and they have the same sample variance and are positively correlated, then the PCA will entail a rotation by 45° and the "loadings" for the two variables with respect to the principal component will be equal. But if we multiply all values of the first variable by 100, then the first principal component will be almost the same as that variable, with a small contribution from the other variable, whereas the second component will be almost aligned with the second original variable. This means that whenever the different variables have different units (like temperature and mass), PCA is a somewhat arbitrary method of analysis. (Different results would be obtained if one used Fahrenheit rather than Celsius for example.) Note that Pearson's original paper was entitled "On Lines and Planes of Closest Fit to Systems of Points in Space" – "in space" implies physical Euclidean space where such concerns do not arise. One way of making the PCA less arbitrary is to use variables scaled so as to have unit variance, by standardizing the data and hence use the autocorrelation matrix instead of the autocovariance matrix as a basis for PCA. However, this compresses (or expands) the fluctuations in all dimensions of the signal space to unit variance.

Mean subtraction (a.k.a. "mean centering") is necessary for performing PCA to ensure that the first principal component describes the direction of maximum variance. If mean subtraction is not performed, the first principal component might instead correspond more or less to the mean of the data. A mean of zero is needed for finding a basis that minimizes the mean square error of the approximation of the data.[9]

PCA is equivalent to empirical orthogonal functions (EOF), a name which is used in meteorology.

An autoencoder neural network with a linear hidden layer is similar to PCA. Upon convergence, the weight vectors of the K neurons in the hidden layer will form a basis for the space spanned by the first K principal components. Unlike PCA, this technique will not necessarily produce orthogonal vectors.

PCA is a popular primary technique in pattern recognition. It is not, however, optimized for class separability.[10] An alternative is the linear discriminant analysis, which does take this into account.

Another application of PCA is reducing the number of parameters in the process of generating computational models of oil reservoirs.[11]

Table of symbols and abbreviations

Symbol	Meaning	Dimensions	Indices
$\mathbf{X} = \{X[i,j]\}$	data matrix, consisting of the set of all data vectors, one vector per row	$n \times p$	$i = 1 \dots n$ $j = 1 \dots p$
n	the number of row vectors in the data set	1×1	*scalar*
p	the number of elements in each row vector (dimension)	1×1	*scalar*
L	the number of dimensions in the dimensionally reduced subspace, $1 \leq L \leq p$	1×1	*scalar*
$\mathbf{u} = \{u[j]\}$	vector of empirical means, one mean for each column *j* of the data matrix	$p \times 1$	$j = 1 \dots p$
$\mathbf{s} = \{s[j]\}$	vector of empirical standard deviations, one standard deviation for each column *j* of the data matrix	$p \times 1$	$j = 1 \dots p$
$\mathbf{h} = \{h[i]\}$	vector of all 1's	$1 \times n$	$i = 1 \dots n$
$\mathbf{B} = \{B[i,j]\}$	deviations from the mean of each column *j* of the data matrix	$n \times p$	$i = 1 \dots n$ $j = 1 \dots p$
$\mathbf{Z} = \{Z[m,n]\}$	z-scores, computed using the mean and standard deviation for each row *m* of the data matrix	$n \times p$	$i = 1 \dots n$ $j = 1 \dots p$
$\mathbf{C} = \{C[k,l]\}$	covariance matrix	$p \times p$	$k = 1 \dots p$ $l = 1 \dots p$
$\mathbf{R} = \{R[k,l]\}$	correlation matrix	$p \times p$	$k = 1 \dots p$ $l = 1 \dots p$
$\mathbf{V} = \{V[j,k]\}$	matrix consisting of the set of all eigenvectors of **C**, one eigenvector per column	$p \times p$	$j = 1 \dots p$ $k = 1 \dots p$
$\mathbf{D} = \{D[k,l]\}$	diagonal matrix consisting of the set of all eigenvalues of **C** along its principal diagonal, and 0 for all other elements	$p \times p$	$k = 1 \dots p$ $l = 1 \dots p$
$\mathbf{W} = \{W[j,k]\}$	matrix of basis vectors, one vector per column, where each basis vector is one of the eigenvectors of **C**, and where the vectors in **W** are a sub-set of those in **V**	$p \times L$	$j = 1 \dots p$ $k = 1 \dots L$
$\mathbf{T} = \{T[i,k]\}$	matrix consisting of *n* row vectors, where each vector is the projection of the corresponding data vector from matrix **X** onto the basis vectors contained in the columns of matrix **W**.	$n \times L$	$i = 1 \dots n$ $k = 1 \dots L$

Properties and limitations of PCA

Properties[12]

***Property 1*:** For any integer *q, 1 ≤ q ≤ p,* consider the orthogonal linear transformation

$$y = \mathbf{B}'x$$

where y is a *q-element* vector and $\mathbf{B}'$ is a *(q × p)* matrix, and let $\mathbf{\Sigma}_y = \mathbf{B}'\mathbf{\Sigma}\mathbf{B}$ be the variance-covariance matrix for y. Then the trace of $\mathbf{\Sigma}_y$, denoted $tr(\mathbf{\Sigma}_y)$, is maximized by taking $\mathbf{B} = \mathbf{A}_q$, where $\mathbf{A}_q$ consists of the first q columns of $\mathbf{A}$ ($\mathbf{B}'$is the transposition of $\mathbf{B}$).

***Property 2*:** Consider again the orthonormal transformation

$$y = \mathbf{B}'x$$

with $x, \mathbf{B}, \mathbf{A}$ and $\mathbf{\Sigma}_y$ defined as before. Then $tr(\mathbf{\Sigma}_y)$is minimized by taking $\mathbf{B}=\mathbf{A}_q^*$, where $\mathbf{A}_q^*$ consists of the last q columns of $\mathbf{A}$.

The statistical implication of this property is that the last few PCs are not simply unstructured left-overs after removing the important PCs. Because these last PCs have variances as small as possible they are useful in their own right. They can help to detect unsuspected near-constant linear relationships between the elements of x, and they may also be useful in regression, in selecting a subset of variables from x, and in outlier detection.

***Property 3*:** (*the Spectral Decomposition of* $\mathbf{\Sigma}$)

$$\mathbf{\Sigma} = \lambda_1\alpha_1\alpha_1' + \lambda_2\alpha_2\alpha_2' + ... + \lambda_p\alpha_p\alpha_p'$$

Before we look at its usage, we first look at diagonal elements,

$$var(x_j) = \sum_{\mathbf{k=1}}^{\mathbf{P}} \lambda_k\alpha_{kj}^2$$

Then, perhaps the main statistical implication of the result is that not only can we decompose the combined variances of all the elements of x into decreasing contributions due to each PC, but we can also decompose the whole covariance matrix into contributions $\lambda_k\alpha_k\alpha_k'$ from each PC. Although not strictly decreasing, the elements of $\lambda_k\alpha_k\alpha_k'$ will tend to become smaller as k increases, as $\lambda_k\alpha_k\alpha_k'$ decreases for increasing k, whereas the elements of α_k tend to stay 'about the same size'because of the normalization constraints: $\alpha_k'\alpha_k = 1, k = 1, 2, ..., p$

Limitations

As noted above, the results of PCA depend on the scaling of the variables. A scale-invariant form of PCA has been developed.[13]

The applicability of PCA is limited by certain assumptions[14] made in its derivation.

PCA and information theory

The claim that the PCA used for dimensionality reduction preserves most of the information of the data is misleading. Indeed, without any assumption on the signal model, PCA cannot help to reduce the amount of information lost during dimensionality reduction, where information was measured using Shannon entropy.[15]

Under the assumption that

$$\mathbf{x} = \mathbf{s} + \mathbf{n}$$

i.e., that the data vector $\mathbf{x}$ is the sum of the desired information-bearing signal $\mathbf{s}$ and a noise signal $\mathbf{n}$ one can show that PCA can be optimal for dimensionality reduction also from an information-theoretic point-of-view.

In particular, Linsker showed that if $\mathbf{s}$ is Gaussian and $\mathbf{n}$ is Gaussian noise with a covariance matrix proportional to the identity matrix, the PCA maximizes the mutual information $I(\mathbf{y}; \mathbf{s})$ between the desired information $\mathbf{s}$ and the dimensionality-reduced output $\mathbf{y} = \mathbf{W}_L^T \mathbf{x}$.[16]

If the noise is still Gaussian and has a covariance matrix proportional to the identity matrix (i.e., the components of the vector $\mathbf{n}$ are iid), but the information-bearing signal $\mathbf{s}$ is non-Gaussian (which is a common scenario), PCA at least minimizes an upper bound on the *information loss*, which is defined as[17][18]

$$I(\mathbf{x}; \mathbf{s}) - I(\mathbf{y}; \mathbf{s}).$$

The optimality of PCA is also preserved if the noise $\mathbf{n}$ is iid and at least more Gaussian (in terms of the Kullback–Leibler divergence) than the information-bearing signal $\mathbf{s}$.[19] In general, even if the above signal model holds, PCA loses its information-theoretic optimality as soon as the noise $\mathbf{n}$ becomes dependent.

Computing PCA using the covariance method

The following is a detailed description of PCA using the covariance method (see also here (http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf)) as opposed to the correlation method.[20] But note that it is better to use the singular value decomposition (using standard software).

The goal is to transform a given data set **X** of dimension *p* to an alternative data set **Y** of smaller dimension *L*. Equivalently, we are seeking to find the matrix **Y**, where **Y** is the Karhunen–Loève transform (KLT) of matrix **X**:

$$\mathbf{Y} = \mathbb{KLT}\{\mathbf{X}\}$$

Organize the data set

Suppose you have data comprising a set of observations of *p* variables, and you want to reduce the data so that each observation can be described with only *L* variables, $L < p$. Suppose further, that the data are arranged as a set of *n* data vectors $\mathbf{x}_1 \ldots \mathbf{x}_n$ with each $\mathbf{x}_i$ representing a single grouped observation of the *p* variables.

- Write $\mathbf{x}_1 \ldots \mathbf{x}_n$ as row vectors, each of which has *p* columns.
- Place the row vectors into a single matrix **X** of dimensions $n \times p$.

Calculate the empirical mean

- Find the empirical mean along each dimension $j = 1, ..., p$.

- Place the calculated mean values into an empirical mean vector **u** of dimensions $p \times 1$.

$$u[j] = \frac{1}{n}\sum_{i=1}^{n} X[i,j]$$

Calculate the deviations from the mean

Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data.[21] Hence we proceed by centering the data as follows:

- Subtract the empirical mean vector **u** from each row of the data matrix **X**.
- Store mean-subtracted data in the $n \times p$ matrix **B**.

$$\mathbf{B} = \mathbf{X} - \mathbf{h}\mathbf{u}^T$$

where **h** is an $n \times 1$ column vector of all 1s:

$$h[i] = 1 \qquad \text{for } i = 1, \ldots, n$$

Find the covariance matrix

- Find the $p \times p$ empirical covariance matrix **C** from the outer product of matrix **B** with itself:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^* \cdot \mathbf{B}$$

where

$*$ is the conjugate transpose operator. Note that if B consists entirely of real numbers, which is the case in many applications, the "conjugate transpose" is the same as the regular transpose.

- Please note that outer products apply to vectors. For tensor cases we should apply tensor products, but the covariance matrix in PCA is a sum of outer products between its sample vectors; indeed, it could be represented as B*.B. See the covariance matrix sections on the discussion page for more information.
- The reasoning behind using N-1 instead of N to calculate the covariance is Bessel's correction

Find the eigenvectors and eigenvalues of the covariance matrix

- Compute the matrix **V** of eigenvectors which diagonalizes the covariance matrix **C**:

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$$

where **D** is the diagonal matrix of eigenvalues of **C**. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues. These algorithms are

readily available as sub-components of most matrix algebra systems, such as R, MATLAB,[22][23] Mathematica,[24] SciPy, IDL (Interactive Data Language), or GNU Octave as well as OpenCV.

- Matrix **D** will take the form of an $p \times p$ diagonal matrix, where

$$D[k,l] = \lambda_k \qquad \text{for } k = l$$

is the *j*th eigenvalue of the covariance matrix **C**, and

$$D[k,l] = 0 \qquad \text{for } k \neq l.$$

- Matrix **V**, also of dimension $p \times p$, contains p column vectors, each of length p, which represent the p eigenvectors of the covariance matrix **C**.
- The eigenvalues and eigenvectors are ordered and paired. The *j*th eigenvalue corresponds to the *j*th eigenvector.

Rearrange the eigenvectors and eigenvalues

- Sort the columns of the eigenvector matrix **V** and eigenvalue matrix **D** in order of *decreasing* eigenvalue.
- Make sure to maintain the correct pairings between the columns in each matrix.

Compute the cumulative energy content for each eigenvector

- The eigenvalues represent the distribution of the source data's energy among each of the eigenvectors, where the eigenvectors form a basis for the data. The cumulative energy content g for the *j*th eigenvector is the sum of the energy content across all of the eigenvalues from 1 through j:

$$g[j] = \sum_{k=1}^{j} D[k,k] \qquad \text{for} \qquad j = 1, \ldots, p$$

Select a subset of the eigenvectors as basis vectors

- Save the first L columns of **V** as the $p \times L$ matrix **W**:

$$W[k,l] = V[k,l] \qquad \text{for} \qquad k = 1, \ldots, p \qquad l = 1, \ldots, L$$

where

$$1 \leq L \leq p.$$

- Use the vector **g** as a guide in choosing an appropriate value for L. The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis. For

example, you may want to choose L so that the cumulative energy g is above a certain threshold, like 90 percent. In this case, choose the smallest value of L such that

$$\frac{g[L]}{g[p]} \geq 0.9$$

Convert the source data to z-scores (optional)

- Create an $p \times 1$ empirical standard deviation vector **s** from the square root of each element along the main diagonal of the diagonalized covariance matrix **C**. (Note, that scaling operations do not commute with the KLT thus we must scale by the variances of the already-decorrelated vector, which is the diagonal of **C**) :

$$\mathbf{s} = \{s[j]\} = \{\sqrt{C[j,j]}\} \qquad \text{for } j = 1, \ldots, p$$

- Calculate the $n \times p$ z-score matrix:

$$\mathbf{Z} = \frac{\mathbf{B}}{\mathbf{h} \cdot \mathbf{s}^T} \text{ (divide element-by-element)}$$

- Note: While this step is useful for various applications as it normalizes the data set with respect to its variance, it is not integral part of PCA/KLT

Project the z-scores of the data onto the new basis

- The projected vectors are the columns of the matrix

$$\mathbf{T} = \mathbf{Z} \cdot \mathbf{W} = \mathbb{KLT}\{\mathbf{X}\}.$$

- The rows of matrix **T** represent the Karhunen–Loeve transforms (KLT) of the data vectors in the rows of matrix **X**.

Derivation of PCA using the covariance method

Let **X** be a d-dimensional random vector expressed as column vector. Without loss of generality, assume **X** has zero mean.

We want to find $(*)$ a $d \times d$ orthonormal transformation matrix **P** so that **PX** has a diagonal covariant matrix (*i.e.* **PX** is a random vector with all its distinct components pairwise uncorrelated).

A quick computation assuming P were unitary yields:

$$\begin{aligned} \operatorname{var}(PX) &= \mathbb{E}[PX\ (PX)^\dagger] \\ &= \mathbb{E}[PX\ X^\dagger P^\dagger] \\ &= P\ \mathbb{E}[X X^\dagger] P^\dagger \\ &= P\ \operatorname{var}(X) P^{-1} \end{aligned}$$

Hence $(*)$ holds if and only if $\operatorname{var}(X)$ were diagonalisable by P.

This is very constructive, as var(**X**) is guaranteed to be a non-negative definite matrix and thus is guaranteed to be diagonalisable by some unitary matrix.

Iterative computation

In practical implementations especially with high dimensional data (large *p*), the covariance method is rarely used because it is not efficient. One way to compute the first principal component efficiently[25] is shown in the following pseudo-code, for a data matrix **X** with zero mean, without ever computing its covariance matrix.

$\mathbf{r} =$ a random vector of length p
do c times:
    $\mathbf{s} = 0$ (a vector of length p)
    for each row $\mathbf{x} \in \mathbf{X}$
        $\mathbf{s} = \mathbf{s} + (\mathbf{x} \cdot \mathbf{r})\mathbf{x}$
    $\mathbf{r} = \frac{\mathbf{s}}{|\mathbf{s}|}$
return $\mathbf{r}$

This algorithm is simply an efficient way of calculating $\mathbf{X^T X}\ \mathbf{r}$, normalizing, and placing the result back in **r** (power iteration). It avoids the np^2 operations of calculating the covariance matrix. **r** will typically get close to the first principal component of **X** within a small number of iterations, *c*. (The magnitude of **s** will be larger after each iteration. Convergence can be detected when it increases by an amount too small for the precision of the machine.)

Subsequent principal components can be computed by subtracting component **r** from **X** (see Gram–Schmidt) and then repeating this algorithm to find the next principal component. However this simple approach is not numerically stable if more than a small number of principal components are required, because imprecisions in the calculations will additively affect the estimates of subsequent principal components. More advanced methods build on this basic idea, as with the closely related Lanczos algorithm.

One way to compute the eigenvalue that corresponds with each principal component is to measure the difference in mean-squared-distance between the rows and the centroid, before and after subtracting out the principal component. The eigenvalue that corresponds with the component that was removed is equal to this difference.

The NIPALS method

For very-high-dimensional datasets, such as those generated in the *omics sciences (e.g., genomics, metabolomics) it is usually only necessary to compute the first few PCs. The non-linear iterative partial least squares (NIPALS) algorithm calculates $\mathbf{t_1}$ and $\mathbf{w_1}^T$ from **X**. The outer product, $\mathbf{t_1 w_1}^T$ can then be subtracted from **X** leaving the residual matrix $\mathbf{E_1}$. This can be then used to calculate subsequent PCs.[26] This results in a dramatic reduction in computational time since calculation of the covariance matrix is avoided.

However, for large data matrices, or matrices that have a high degree of column collinearity, NIPALS suffers from loss of orthogonality due to machine precision limitations accumulated in each iteration step.[27] A Gram–Schmidt (GS) re-orthogonalization algorithm is applied to both the scores and the loadings at each iteration step to eliminate this loss of orthogonality.[28]

Online/sequential estimation

In an "online" or "streaming" situation with data arriving piece by piece rather than being stored in a single batch, it is useful to make an estimate of the PCA projection that can be updated sequentially. This can be done efficiently, but requires different algorithms.[29]

PCA and qualitative variables

In PCA, it is common that we want to introduce qualitative variables as supplementary elements. For example, many quantitative variables have been measured on plants. For these plants, some qualitative variables are available as, for example, the species to which the plant belongs. These data were subjected to PCA for quantitative variables. When analyzing the results, it is natural to connect the principal components to the qualitative variable *species*. For this, the following results are produced.

- Identification, on the factorial planes, of the different species e.g. using different colors.
- Representation, on the factorial planes, of the centers of gravity of plants belonging to the same species.
- For each center of gravity and each axis, p-value to judge the significance of the difference between the center of gravity and origin.

These results are what is called *introducing a qualitative variable as supplementary element*. This procedure is detailed in and Husson, Lê & Pagès 2009 and Pagès 2013. Few software offer this option in an "automatic" way. This is the case of SPAD (http://www.coheris.com/produits/analytics/logiciel-data-mining/) that historically, following the work of Ludovic Lebart, was the first to propose this option, and the R package FactoMineR (http://factominer.free.fr/).

Applications

Neuroscience

A variant of principal components analysis is used in neuroscience to identify the specific properties of a stimulus that increase a neuron's probability of generating an action potential.[30] This technique is known as spike-triggered covariance analysis. In a typical application an experimenter presents a white noise process as a stimulus (usually either as a sensory input to a test subject, or as a current injected directly into the neuron) and records a train of action potentials, or spikes, produced by the neuron as a result. Presumably, certain features of the stimulus make the neuron more likely to spike. In order to extract these features, the experimenter calculates the covariance matrix of the *spike-triggered ensemble*, the set of all stimuli (defined and discretized over a finite time window, typically on the order of 100 ms) that immediately preceded a spike. The eigenvectors of the difference between the spike-triggered covariance matrix and the covariance matrix of the *prior stimulus ensemble* (the set of all stimuli, defined over the same length time window) then indicate the directions in the space of stimuli along which the variance of the spike-triggered ensemble differed the most from that of the prior stimulus ensemble. Specifically, the eigenvectors with the largest positive eigenvalues correspond to the

directions along which the variance of the spike-triggered ensemble showed the largest positive change compared to the variance of the prior. Since these were the directions in which varying the stimulus led to a spike, they are often good approximations of the sought after relevant stimulus features.

In neuroscience, PCA is also used to discern the identity of a neuron from the shape of its action potential. Spike sorting is an important procedure because extracellular recording techniques often pick up signals from more than one neuron. In spike sorting, one first uses PCA to reduce the dimensionality of the space of action potential waveforms, and then performs clustering analysis to associate specific action potentials with individual neurons.

Relation between PCA and *K*-means clustering

It was asserted in [31][32] that the relaxed solution of *k*-means clustering, specified by the cluster indicators, is given by the PCA (principal component analysis) principal components, and the PCA subspace spanned by the principal directions is identical to the cluster centroid subspace. However, that PCA is a useful relaxation of k-means clustering was not a new result (see, for example,[33]), and it is straightforward to uncover counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.[34]

Relation between PCA and factor analysis[35]

Principal component analysis creates variables that are linear combinations of the original variables. The new variables have the property that the variables are all orthogonal. The principal components can be used to find clusters in a set of data. PCA is a variance-focused approach seeking to reproduce the total variable variance, in which components reflect both common and unique variance of the variable. PCA is generally preferred for purposes of data reduction (i.e., translating variable space into optimal factor space) but not when detect the latent construct or factors.

Factor analysis is similar to principal component analysis, in that factor analysis also involves linear combinations of variables. Different from PCA, factor analysis is a correlation-focused approach seeking to reproduce the inter-correlations among variables, in which the factors “represent the common variance of variables, excluding unique variance[36]" . Factor analysis is generally used when the research purpose is detecting data structure (i.e., latent constructs or factors) or causal modeling.

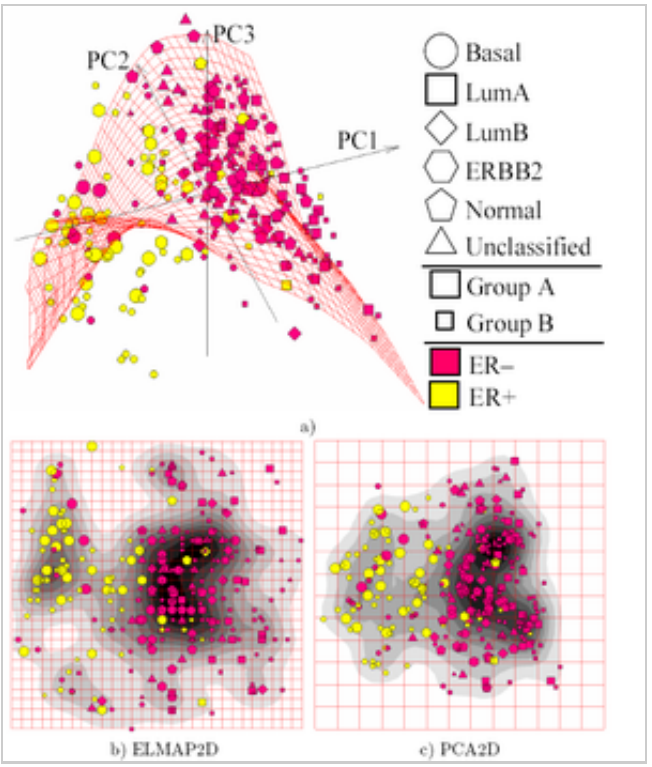
Correspondence analysis

Correspondence analysis (CA) was developed by Jean-Paul Benzécri[37] and is conceptually similar to PCA, but scales the data (which should be non-negative) so that rows and columns are treated equivalently. It is traditionally applied to contingency tables. CA decomposes the chi-squared statistic associated to this table into orthogonal factors.[38] Because CA is a descriptive technique, it can be applied to tables for which the chi-squared statistic is appropriate or not. Several variants of CA are available including detrended correspondence analysis and canonical correspondence analysis. One special extension is multiple correspondence analysis, which may be seen as the counterpart of principal component analysis for categorical data.[39]

Generalizations

Nonlinear generalizations

Most of the modern methods for nonlinear dimensionality reduction find their theoretical and algorithmic roots in PCA or K-means. Pearson's original idea was to take a straight line (or plane) which will be "the best fit" to a set of data points. **Principal curves and manifolds**[43] give the natural geometric framework for PCA generalization and extend the geometric interpretation of PCA by explicitly constructing an embedded manifold for data approximation, and by encoding using standard geometric projection onto the manifold, as it is illustrated by Fig. See also the elastic map algorithm and principal geodesic analysis. Another popular generalization is kernel PCA, which corresponds to PCA performed in a reproducing kernel Hilbert space associated with a positive definite kernel.



Linear PCA versus nonlinear Principal Manifolds[40] for visualization of breast cancer microarray data: a) Configuration of nodes and 2D Principal Surface in the 3D PCA linear manifold. The dataset is curved and cannot be mapped adequately on a 2D principal plane; b) The distribution in the internal 2D non-linear principal surface coordinates (ELMap2D) together with an estimation of the density of points; c) The same as b), but for the linear 2D PCA manifold (PCA2D). The "basal" breast cancer subtype is visualized more adequately with ELMap2D and some features of the distribution become better resolved in comparison to PCA2D. Principal manifolds are produced by the elastic maps algorithm. Data are available for public competition.[41] Software is available for free non-commercial use.[42]

Multilinear generalizations

In multilinear subspace learning,[44] PCA is generalized to multilinear PCA (MPCA) that extracts features directly from tensor representations. MPCA is solved by performing PCA in each mode of the tensor iteratively. MPCA has been applied to face recognition, gait recognition, etc. MPCA is further extended to uncorrelated MPCA, non-negative MPCA and robust MPCA.

Higher order

N-way principal component analysis may be performed with models such as Tucker decomposition, PARAFAC, multiple factor analysis, co-inertia analysis, STATIS, and DISTATIS.

Robustness – weighted PCA

While PCA finds the mathematically optimal method (as in minimizing the squared error), it is sensitive to outliers in the data that produce large errors PCA tries to avoid. It therefore is common practice to remove outliers before computing PCA. However, in some contexts, outliers can be difficult to identify. For example in data mining algorithms like correlation clustering, the assignment of points to clusters and outliers is not known beforehand. A recently proposed generalization of PCA[45] based on a **weighted PCA** increases robustness by assigning different weights to data objects based on their estimated relevancy.

Robust PCA via Decomposition in Low Rank and Sparse Matrices

Robust principal component analysis (RPCA) is a modification of the widely used statistical procedure Principal component analysis (PCA) which works well with respect to grossly corrupted observations.

Sparse PCA

A particular disadvantage of PCA is that the principal components are usually linear combinations of all input variables. Sparse PCA overcomes this disadvantage by finding linear combinations that contain just a few input variables.

Software/source code

- An Open Source Code (https://code.msdn.microsoft.com/Machine-Learnin-in-2f63c0ac/) and Tutorial in MATLAB and C++.
- FactoMineR (http://factominer.free.fr/) - Probably the more complete library of functions for exploratory data analysis.
- Mathematica - Implements principal component analysis with the PrincipalComponents command[46] using both covariance and correlation methods.
- NAG Library - Principal components analysis is implemented via the `g03aa` routine (available in both the Fortran[47] and the C[48] versions of the Library).
- SIMCA - Commercial software package available to perform PCA analysis.[49]
- MATLAB Statistics Toolbox - The functions `princomp` and `pca` (R2012b) give the principal components, while the function `pcares` gives the residuals and reconstructed matrix for a low-rank PCA approximation. An example MATLAB implementation of PCA is available.[50]
- Oracle Database 12c - Implemented via `DBMS_DATA_MINING.SVDS_SCORING_MODE` by specifying setting value `SVDS_SCORING_PCA` [51]
- GNU Octave - Free software computational environment mostly compatible with MATLAB, the function `princomp`[52] gives the principal component.
- R - Free statistical package, the functions `princomp`[53] and `prcomp`[54] can be used for principal component analysis; `prcomp` uses singular value decomposition which generally gives better numerical accuracy. Some packages that implement PCA in R, include, but are not limited to: `ade4`, `vegan`, `ExPosition`, and `FactoMineR`[55]
- SAS, PROC FACTOR - Offers principal components analysis.
- MLPACK - Provides an implementation of principal component analysis in C++.
- XLMiner - The principal components tab can be used for principal component analysis.
- Stata - The pca command provides principal components analysis.
- Cornell Spectrum Imager - Open-source toolset built on ImageJ, enables PCA analysis for 3D datacubes.[56]
- imDEV - Free Excel addon to calculate principal components using R package[57][58]
- ViSta: The Visual Statistics System - Free software that provides principal components analysis, simple and multiple correspondence analysis.[59]

- Spectramap - Software to create a biplot using principal components analysis, correspondence analysis or spectral map analysis.[60]
- FinMath - .NET numerical library containing an implementation of PCA.[61]
- Unscrambler X - Multivariate analysis software enabling Principal Component Analysis (PCA) with PCA Projection.{[62]}
- OpenCV[63]
- NMath - Proprietary numerical library containing PCA for the .NET Framework.
- IDL - The principal components can be calculated using the function `pcomp`.[64]
- Weka - Computes principal components.[65]
- Qlucore - Commercial software for analyzing multivariate data with instant response using PCA[66]
- Orange (software) - Supports PCA through its Linear Projection widget.
- EIGENSOFT - Provides a version of PCA adapted for population genetics analysis.[67]
- Partek Genomics Suite - Statistical software able to perform PCA.[68]
- libpca C++ library (https://sourceforge.net/projects/libpca/) - Offers PCA and corresponding transformations.
- Origin - Contains PCA in its Pro version.
- Scikit-learn - Python library for machine learning which contains PCA, Probabilistic PCA, Kernel PCA, Sparse PCA and other techniques in the decomposition module.[69]
- Knime[70]- A java based nodal arrenging software for Analysis, in this the nodes called PCA, PCA compute, PCA Apply, PCA inverse make it easily.
- Julia - Supports PCA with the `pca` function in the MultivariateStats package [71]

See also

- Correspondence analysis (for contingency tables)
- Multiple correspondence analysis (for qualitative variables)
- Factor analysis of mixed data (for quantitative **and** qualitative variables)
- Canonical correlation
- CUR matrix approximation (can replace of low-rank SVD approximation)
- Detrended correspondence analysis
- Dynamic mode decomposition
- Eigenface
- Exploratory factor analysis (Wikiversity)
- Factorial code
- Functional principal component analysis
- Geometric data analysis
- Independent component analysis

- Kernel PCA
- Low-rank approximation
- Matrix decomposition
- Non-negative matrix factorization
- Nonlinear dimensionality reduction
- Oja's rule
- Point distribution model (PCA applied to morphometry and computer vision)
- Principal component analysis (Wikibooks)
- Principal component regression
- Singular spectrum analysis
- Singular value decomposition
- Sparse PCA
- Transform coding
- Weighted least squares

Notes

1. ^ ***a*** ***b*** Jolliffe I.T. Principal Component Analysis (http://www.springer.com/west/home/new+%26+forthcoming+titles+%28default%29?SGWID=4-40356-22-2285433-0), Series: Springer Series in Statistics (http://www.springer.com/west/home/statistics/statistical+theory+and+methods?SGWID=4-10129-69-173621571-0), 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4
2. ^ Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (http://stat.smmu.edu.cn/history/pearson1901.pdf) (PDF). *Philosophical Magazine* **2** (11): 559–572. doi:10.1080/14786440109462720 (https://dx.doi.org/10.1080%2F14786440109462720).
3. ^ Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24**, 417-441, and 498-520.
   Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, **27**, 321-77
4. ^ Abdi. H., & Williams, L.J. (2010). "Principal component analysis.". *Wiley Interdisciplinary Reviews: Computational Statistics,* **2**: 433–459. doi:10.1002/wics.101 (https://dx.doi.org/10.1002%2Fwics.101).
5. ^ Shaw P.J.A. (2003) *Multivariate statistics for the Environmental Sciences*, Hodder-Arnold. ISBN 0-340-80763-6.
6. ^ Barnett, T. P., and R. Preisendorfer. (1987). "Origins and levels of monthly and seasonal forecast skill for United States surface air temperatures determined by canonical correlation analysis.". *Monthly Weather Review 115*.
7. ^ Hsu, Daniel, Sham M. Kakade, and Tong Zhang (2008). "A spectral algorithm for learning hidden markov models.". *arXiv preprint arXiv:0811.4413*.
8. ^ Bengio, Y. et al (2013). "Representation Learning: A Review and New Perspectives" (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6472238) (PDF). *Pattern Analysis and Machine Intelligence* **35** (8). doi:10.1109/TPAMI.2013.50 (https://dx.doi.org/10.1109%2FTPAMI.2013.50).
9. ^ A. A. Miranda, Y. A. Le Borgne, and G. Bontempi. New Routes from Minimal Approximation Error to Principal Components (http://www.ulb.ac.be/di/map/yleborgn/pub/NPL_PCA_07.pdf), Volume 27, Number

3 / June, 2008, Neural Processing Letters, Springer
10. ^ Fukunaga, Keinosuke (1990). *Introduction to Statistical Pattern Recognition* (http://books.google.com/books?visbn=0-12-269851-7). Elsevier. ISBN 0-12-269851-7.
11. ^ Gharib Shirangi, M., History matching production data and uncertainty assessment with a truncated SVD parameterization algorithm, Journal of Petroleum Science and Engineering, http://www.sciencedirect.com/science/article/pii/S0920410513003227
12. ^ Jolliffe, I. T. (2002). *Principal Component Analysis,* second edition Springer-Verlag. ISBN 978-0-387-95442-4.
13. ^ Leznik, M; Tofallis, C. 2005 [uhra.herts.ac.uk/bitstream/handle/2299/715/S56.pdf Estimating Invariant Principal Components Using Diagonal Regression.]
14. ^ Jonathon Shlens, A Tutorial on Principal Component Analysis. (http://arxiv.org/abs/1404.1100)
15. ^ Geiger, Bernhard; Kubin, Gernot (Sep 2012). "Relative Information Loss in the PCA" (http://arxiv.org/abs/1204.0429). *Proc. IEEE Information Theory Workshop*: 562–566.
16. ^ Linsker, Ralph (March 1988). "Self-organization in a perceptual network". *IEEE Computer* **21** (3): 105–117. doi:10.1109/2.36 (https://dx.doi.org/10.1109%2F2.36).
17. ^ Deco & Obradovic (1996). *An Information-Theoretic Approach to Neural Computing*. New York, NY: Springer.
18. ^ Plumbley, Mark (1991). "Information theory and unsupervised neural networks".Tech Note
19. ^ Geiger, Bernhard; Kubin, Gernot (January 2013). "Signal Enhancement as Minimization of Relevant Information Loss" (http://arxiv.org/abs/1205.6935). *Proc. ITG Conf. on Systems, Communication and Coding*.
20. ^ "Engineering Statistics Handbook Section 6.5.5.2" (http://www.itl.nist.gov/div898/handbook/pmc/section5/pmc552.htm). Retrieved 19 January 2015.
21. ^ A.A. Miranda, Y.-A. Le Borgne, and G. Bontempi. New Routes from Minimal Approximation Error to Principal Components (http://www.ulb.ac.be/di/map/yleborgn/pub/NPL_PCA_07.pdf), Volume 27, Number 3 / June, 2008, Neural Processing Letters, Springer
22. ^ eig function (http://www.mathworks.com/access/helpdesk/help/techdoc/ref/eig.html#998306) Matlab documentation
23. ^ MATLAB PCA-based Face recognition software (http://www.mathworks.com/matlabcentral/fileexchange/24634)
24. ^ Eigenvalues function (http://reference.wolfram.com/mathematica/ref/Eigenvalues.html) Mathematica documentation
25. ^ Roweis, Sam. "EM Algorithms for PCA and SPCA." Advances in Neural Information Processing Systems. Ed. Michael I. Jordan, Michael J. Kearns, and Sara A. Solla The MIT Press, 1998.
26. ^ Geladi, Paul; Kowalski, Bruce (1986). "Partial Least Squares Regression:A Tutorial". *Analytica Chimica Acta* **185**: 1–17. doi:10.1016/0003-2670(86)80028-9 (https://dx.doi.org/10.1016%2F0003-2670%2886%2980028-9).
27. ^ Kramer,R., (1998) Chemometric Techniques for Quantitative Analysis (CRC Press, New York).
28. ^ M. Andrecut. Parallel GPU Implementation of Iterative PCA Algorithms. Journal of Computational Biology, 16(11), Nov. 2009.
29. ^ Warmuth, M. K.; Kuzmin, D. (2008). "Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension". *Journal of Machine Learning Research* **9**: 2287–2320.
30. ^ Brenner, N., Bialek, W., & de Ruyter van Steveninck, R.R. (2000).
31. ^ H. Zha, C. Ding, M. Gu, X. He and H.D. Simon (Dec 2001). "Spectral Relaxation for K-means Clustering"

(http://ranger.uta.edu/~chqding/papers/Zha-Kmeans.pdf). *Neural Information Processing Systems vol.14 (NIPS 2001)* (Vancouver, Canada): 1057–1064.

32. ^ Chris Ding and Xiaofeng He (July 2004). "K-means Clustering via Principal Component Analysis" (http://ranger.uta.edu/~chqding/papers/KmeansPCA1.pdf). *Proc. of Int'l Conf. Machine Learning (ICML 2004)*: 225–232.
33. ^ Drineas, P.; A. Frieze; R. Kannan; S. Vempala; V. Vinay (2004). "Clustering large graphs via the singular value decomposition" (http://www.cc.gatech.edu/~vempala/papers/dfkvv.pdf). *Machine learning* **56**: 9–33. doi:10.1023/b:mach.0000033113.59016.96 (https://dx.doi.org/10.1023%2Fb%3Amach.0000033113.59016.96). Retrieved 2012-08-02.
34. ^ Cohen, M.; S. Elder; C. Musco; C. Musco; M. Persu (2014). "Dimensionality reduction for k-means clustering and low rank approximation (Appendix B)" (http://arxiv.org/abs/1410.6801). *ArXiv*. Retrieved 2014-11-29.
35. ^ http://www.linkedin.com/groups/What-is-difference-between-factor-107833.S.162765950
36. ^ Timothy A. Brown. Confirmatory Factor Analysis for Applied Research Methodology in the social sciences. Guilford Press, 2006
37. ^ Benzécri, J.-P. (1973). *L'Analyse des Données. Volume II. L'Analyse des Correspondances*. Paris, France: Dunod.
38. ^ Greenacre, Michael (1983). *Theory and Applications of Correspondence Analysis*. London: Academic Press. ISBN 0-12-299050-1.
39. ^ Le Roux, Brigitte and Henry Rouanet (2004). *Geometric Data Analysis, From Correspondence Analysis to Structured Data Analysis*. Dordrecht: Kluwer.
40. ^ A. N. Gorban, A. Y. Zinovyev, Principal Graphs and Manifolds (http://arxiv.org/abs/0809.0490), In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques, Olivas E.S. et al Eds. Information Science Reference, IGI Global: Hershey, PA, USA, 2009. 28-59.
41. ^ Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J. et al.: Gene expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer" *Lancet* 365, 671-679 (2005); Data online (http://www.ihes.fr/~zinovyev/princmanif2006/)
42. ^ A. Zinovyev, ViDaExpert (http://bioinfo-out.curie.fr/projects/vidaexpert/) – Multidimensional Data Visualization Tool (free for non-commercial use). Institut Curie, Paris.
43. ^ A.N. Gorban, B. Kegl, D.C. Wunsch, A. Zinovyev (Eds.), Principal Manifolds for Data Visualisation and Dimension Reduction, (http://pca.narod.ru/contentsgkwz.htm) LNCSE 58, Springer, Berlin – Heidelberg – New York, 2007. ISBN 978-3-540-73749-0
44. ^ Lu, Haiping; Plataniotis, K.N.; Venetsanopoulos, A.N. (2011). "A Survey of Multilinear Subspace Learning for Tensor Data" (http://www.dsp.utoronto.ca/~haiping/Publication/SurveyMSL_PR2011.pdf). *Pattern Recognition* **44** (7): 1540–1551. doi:10.1016/j.patcog.2011.01.004 (https://dx.doi.org/10.1016%2Fj.patcog.2011.01.004).
45. ^ Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2008). "A General Framework for Increasing the Robustness of PCA-Based Correlation Clustering Algorithms". *Scientific and Statistical Database Management*. Lecture Notes in Computer Science **5069**: 418. doi:10.1007/978-3-540-69497-7_27 (https://dx.doi.org/10.1007%2F978-3-540-69497-7_27). ISBN 978-3-540-69476-2.
46. ^ PrincipalComponents (http://reference.wolfram.com/mathematica/ref/PrincipalComponents.html) Mathematica Documentation
47. ^ The Numerical Algorithms Group. "NAG Library Routine Document: nagf_mv_prin_comp (g03aaf)"

(http://www.nag.co.uk/numeric/fl/nagdoc_fl23/pdf/G03/g03aaf.pdf). *NAG Library Manual, Mark 23*. Retrieved 2012-02-16.

48. ^ The Numerical Algorithms Group. "NAG Library Routine Document: nag_mv_prin_comp (g03aac)" (http://www.nag.co.uk/numeric/CL/nagdoc_cl09/pdf/G03/g03aac.pdf). *NAG Library Manual, Mark 9*. Retrieved 2012-02-16.
49. ^ `PcaPress` (http://www.umetrics.com/products/simca) http://www.umetrics.com/products/simca
50. ^ `PcaPress` (http://www.utdallas.edu/~herve/abdi-PCA4Wiley.zip) www.utdallas.edu
51. ^ Oracle documentation [1] (http://docs.oracle.com/database/121/ARPLS/d_datmin.htm#ARPLS73749) http://docs.oracle.com
52. ^ `princomp` (http://octave.sourceforge.net/statistics/function/princomp.html) octave.sourceforge.net
53. ^ `princomp` (http://stat.ethz.ch/R-manual/R-patched/library/stats/html/princomp.html)
54. ^ `prcomp` (http://stat.ethz.ch/R-manual/R-patched/library/stats/html/prcomp.html)
55. ^ Multivariate (http://cran.r-project.org/web/views/Multivariate.html) cran.r-project.org
56. ^ Cornell Spectrum Imager https://code.google.com/p/cornell-spectrum-imager/wiki/Home (https://code.google.com/p/cornell-spectrum-imager/wiki/Home)
57. ^ imDEV (https://sourceforge.net/apps/mediawiki/imdev/index.php?title=Main_Page) sourceforge.net
58. ^ pcaMethods (http://www.bioconductor.org/packages/1.9/bioc/html/pcaMethods.html) www.bioconductor.org
59. ^ "ViSta: The Visual Statistics System" (http://www.mdp.edu.ar/psicologia/vista/vista.htm) www.mdp.edu.ar
60. ^ "Spectramap" (http://www.coloritto.com) www.coloritto.com
61. ^ FinMath (https://rtmath.net/products/finmath/) rtmath.net
62. ^ http://www.camo.com
63. ^ Computer Vision Library (http://sourceforge.net/projects/opencvlibrary/) sourceforge.net
64. ^ PCOMP (IDL Reference) | Exelis VIS Docs Center (http://www.exelisvis.com/docs/PCOMP.html) IDL online documentation
65. ^ javadoc (http://weka.sourceforge.net/doc.dev/weka/attributeSelection/PrincipalComponents.html) weka.sourceforge.net
66. ^ Software for analyzing multivariate data with instant response using PCA (http://www.qlucore.com) www.qlucore.com
67. ^ EIGENSOFT (http://genepath.med.harvard.edu/~reich/Software.htm) genepath.med.harvard.edu
68. ^ Partek Genomics Suite (http://www.partek.com/partekgs) www.partek.com
69. ^ [2] (http://scikit-learn.org/stable/modules/classes.html) http://scikit-learn.org
70. ^ [3] (https://www.knime.org)
71. ^ MultivariateStats.jl (https://github.com/JuliaStats/MultivariateStats.jl) www.github.com

References

- Jackson, J.E. (1991). *A User's Guide to Principal Components* (Wiley).
- Jolliffe, I. T. (1986). *Principal Component Analysis* (http://www.springer.com/west/home/new+%26+forthcoming+titles+%28default%29?SGWID=4-40356-22-2285433-0). Springer-Verlag. p. 487. doi:10.1007/b98835 (https://dx.doi.org/10.1007%2Fb98835). ISBN 978-0-387-95442-4.
- Jolliffe, I.T. (2002). *Principal Component Analysis,* second edition (Springer).

- Husson François, Lê Sébastien & Pagès Jérôme (2009). *Exploratory Multivariate Analysis by Example Using R*. Chapman & Hall/CRC The R Series, London. 224p. |isbn=978-2-7535-0938-2
- Pagès Jérôme (2014). *Multiple Factor Analysis by Example Using R*. Chapman & Hall/CRC The R Series London 272 p

External links

Wikimedia Commons has media related to ***Principal component analysis***.

- University of Copenhagen video by Rasmus Bro (https://www.youtube.com/watch?v=UUxIXU_Ob6E) on YouTube
- Stanford University video by Andrew Ng (https://www.youtube.com/watch?v=ey2PE5xi9-A#t=2385) on YouTube
- A Tutorial on Principal Component Analysis (http://arxiv.org/abs/1404.1100)
- A layman's introduction to principal component analysis (https://www.youtube.com/watch?v=BfTMmoDFXyE) on YouTube (a video of less than 100 seconds.)
- Source Code of Principal Component Analysis and Eigenfaces (https://code.msdn.microsoft.com/Machine-Learnin-in-2f63c0ac)

Categories: Multivariate statistics | Matrix decompositions | Data analysis | Dimension reduction

---