PRINCIPAL COMPONENT ANALYSIS FOR MACHINE LEARNING

Polina Shulpina,
MTUCI, Moscow, Russia
polli-lionet@yandex.ru

V. A. Dokuchaev,
MTUCI, Moscow, Russia
v.a.dokuchaev@mtuci.ru

DOI: 10.36724/2664-066X-2022-8-6-18-24

ABSTRACT

Training a Supervised Machine Learning model involves several stages. In the first stage, the data is passed via model, creating predictions (forecasts). The next stage is to compare these forecasts with factual values (ground truth). The final stage is optimizing the model by minimizing a certain cost function. The model improves that way. Occasionally, an input sample contains many columns. Using each column in a model leads to problems, the curse of dimensionality. At that rate, it is necessary to be selective about functions. We will embrace Principal Component Analysis (PCA), that is one of the main ways to reduce the dimensionality of data, losing the least amount of information.

KEYWORDS: *principal component analysis, PCA, machine learning, deep learning, feature scaling, feature extraction, data preprocessing*

Information about authors:
Polina Shulpina, *Network Information Technologies and Services, MTUCI, Moscow, Russia*
V.A. Dokuchaev, *DSc, Prof., Network Information Technologies and Services, MTUCI, Moscow, Russia*

Introduction

The digital transformation currently underway affects all aspects of human life: economic, political, social, etc. [1-3]. In the conditions of robotization, automation and informatization of technological processes and business, the correctness of decisions made and the reduction of the time required for this is becoming increasingly important. Therefore, the creation and implementation of decision support systems using the AI concept becomes relevant. At the same time, the problem of Big Data arises – the necessity to handle a big quantity of various data in order to make the right decision, taking into account new potential risks [4-7]. One way to meet the challenge is to use Principal Component Analysis.

The principal component analysis is a linear algebraic dimensionality reduction method. First, it is necessary to consider why the specifics of PCA can be beneficial for machine learning (ML) projects. Thereby, the first thing to pay attention to is the "curse of dimensionality" [8] that is particularly powerful with older ML algorithms (Naive Bayes, K-Nearest Neighbour). Let’s discuss the distinction between Feature Extraction (FE) and Feature Selection (FS) on to PCA and what we can do to struggle against dimensionality reduction [9].

When training a Supervised Machine Learning model, we follow a three-stage iterative process, which is depicted in Figure 1.

The first stage in model training is feeding samples to the model. The model has just been initialized. Predictions are generated for every sample and they probably are incoherent.

The next stage is to compare forecasts to ground truth. The simile manufactures loss value or an error that shows how inadequate the model satisfies.

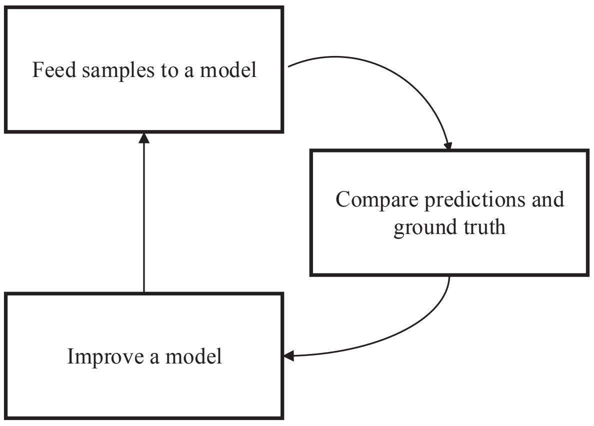

Fig. 1. Three-stage iterative process for model training

The third stage is improving the model. Optimization occurs differently. Gradients are computed using back propagation and then optimizers change the model's internal components. In addition, it is possible to change weights via minimization just a single function.

After optimizing the model, the predictions get a little better. Iterating is needed to be kept until we are contented with the outcome. After this, we finish the learning process.

I. Underfitting and overfitting a model

At the initial steps of the process of learning, a model is probably not suitable to grab samples in a dataset (Fig. 2). The decision is elementary: we should continue learning until we reach the correct conformity for the dataset (Fig. 4). We can't keep training forever. To the contrary, we will face the problem of overfitting. Overfitting is the phenomenon when the quality of the model on the training dataset significantly exceeds the quality of the model on the test set.

As a result, a model acclimatized to a concrete dataset, visible in Figure 3. Both an underfitted and overfitted model cannot generalize well to the test data. Therefore, we should find a balance between these two aspects.

The curse of dimensionality is one of the biggest problems in Machine Learning, which states that the higher the dimensionality, the more sparse the data. In other words, as the number of features grows, the amount of data we need to generalize grows exponentially.

It is worth distinguishing between feature extraction and feature selection. Feature selection is the selection of a sub-set of features from the available ones without transforming the object. Feature extraction is the transformation of an object, i.e. transformation of the selected features into a lower-dimensional space. Let’s discuss them [10].

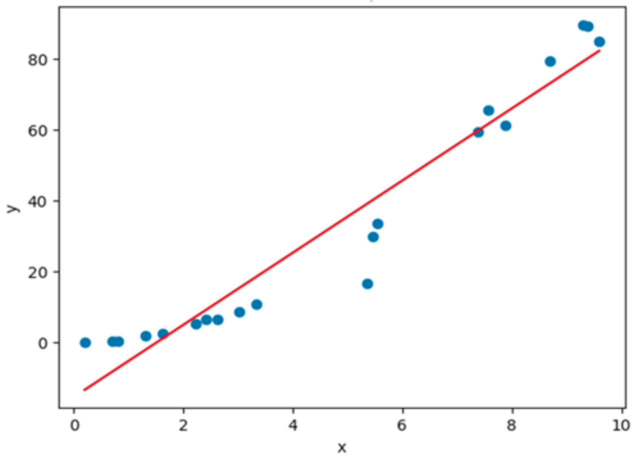

Fig. 2. Model that does not catch a pattern in the dataset

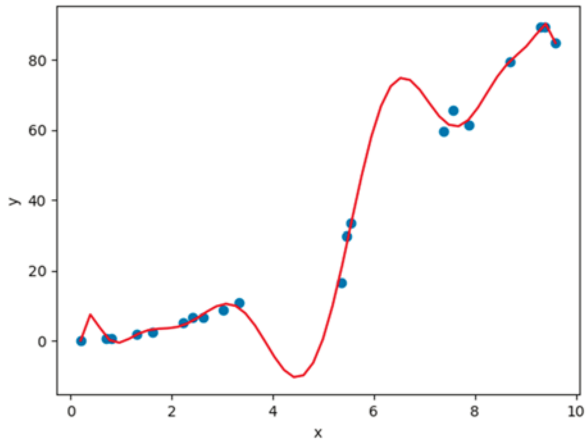

Fig. 3. Model that fits the dataset too well

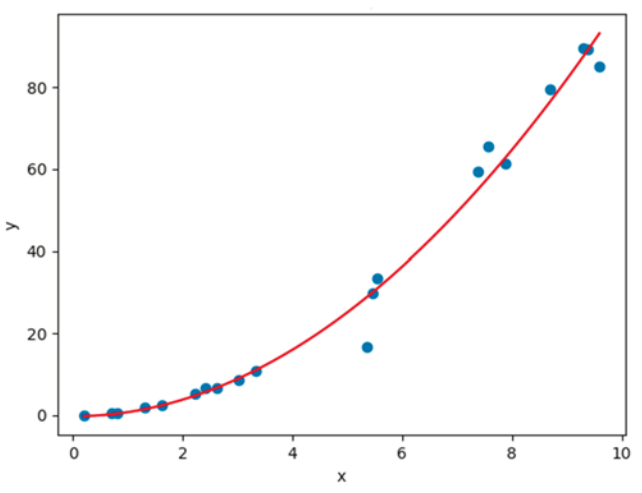

Fig. 4. Model that fits well to describe the dataset

The problem of too many features is known as the Curse of Dimensionality. When the number of features (data dimensionality) grows, it becomes increasingly difficult for the model to learn the correspondence between features and goals. In the case of Dimensionality Reduction, there are two approaches: Feature Selection and Feature Extraction.

- Feature Selection. Each object is characterized by a set of features. An attribute of an object (image, text, sound) can be any quantitative characteristic; the main thing is that this characteristic is unique for this type of objects. The more features, the longer the algorithm works. Feature Selection helps to solve the following problems: simplify the model in order to better understand its operation, reduce training time, avoid the "curse of dimensionality", and reduce overfitting.
- Although feature selection is formally a special case of feature extraction, feature selection uses its own unique algorithms and methods. Feature selection techniques can be divided into three methods: wrapper methods, filter methods, and embedded methods. Feature Selection is an integral portion of the training process, so you need to do Feature Selection independently for the training and test sets. If this is not done and Feature Selection is carried out for the entire database, then the data will inadvertently be distorted, which will lead to overfitting.
- Feature Extraction. Feature Extraction is associated with a decline in the dimension of the feature space. Feature Extraction is a set of methods that map input functions to new output functions. Feature Extraction accelerates the convergence of machine learning algorithms, making them applicable in practice.
- It is reduction the dimension of the feature space by applying several mapping functions. It retrieves the most informatory features based on the selected metrics. In contrast to Feature Selection, Feature Extraction modifies the master features. The main part of Feature Extraction is the mapping function.

II. Introduction to Principal Component Analysis

The principal component method is a way to decrease the dimensionality of the information while losing the least quantity of data. In other words, the sense of this method is that every principal component is connected with a specific ratio of the overall variance (dispersion) of the master dataset (a burden). The dispersion, that is a gauge of the volatility of the data, can represent the rate of information content of the data [9].

The task of reducing the dimensionality of the dataset is to describe the data points using a number smaller than the dimensionality of the space.

The task of PCA: to create a novel feature space of lower dimensionality whose variance between the axes are reallocated to increase the variance for every axe.

There are the following steps of PCA:

1. The overall dispersion of the master feature space is rated. It cannot be made by merely folding the dispersion for every variable, since they are, in most cases, not autonomous. Hence, it is necessary to summarize the reciprocal dispersions of the variables, which are defined of the covariance matrix.
2. The eigenvalues and eigenvectors of the covariance matrix are calculated.
3. Dimensionality reduction is performed. Diagonal principles of the covariance matrix indicate the dispersion under the master system of axes and its eigenvalues under the novel coordinate system. By separating the dispersion connected with every principal component via the amount of the dispersions of each component, we get the ratio of dispersion connected with every component. Then, a great deal of principal components are ejected, the fraction of the remaining components is 80-90%.

There are two methods to decompose the dataset into two vectors. The first method is decomposing the covariance matrix into eigenvectors. The second method is decomposing the covariance matrix into singular values [11].

Consider we create a dataset based on two overlapping patches that are parts of only one dataset. The spread of the dataset is shown in Figure 5.

The dataset mostly spreads out in two directions. These are the directions from the upper right-hand corner to the lower left-hand corner and from the bottom right middle to the top left middle.

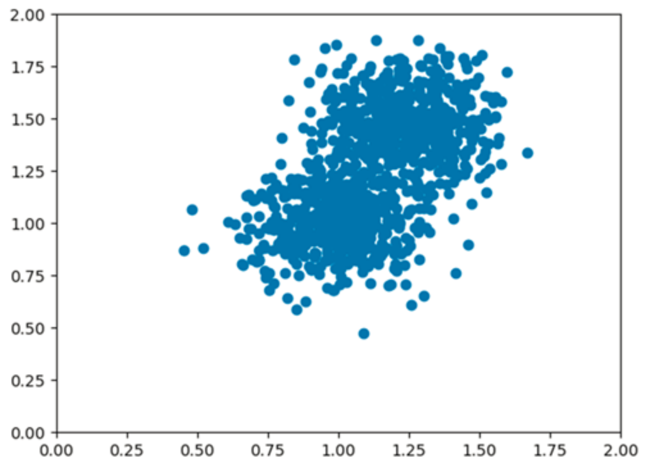

Fig. 5. Spread of the dataset

Now it is possible to depict trends as a pair of two vectors (principal directions of the data). The number of principle directions is equal to the number of measurements. We have two dimensions. The scatter of the dataset as a pair of two vectors is shown in Figure 6.

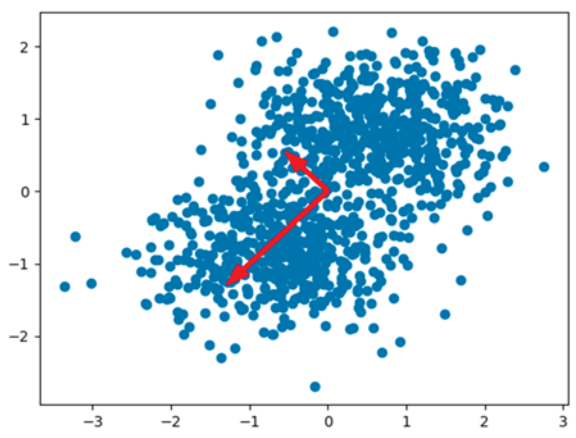

Fig. 6. Scatter expression of the dataset

These vectors are called eigenvectors. Their length is represented by the so-called eigenvalue. The data along new feature axes are explained by eigenvalues, so we can determine two trends and amount of the variance in the dataset [12].

Since the axes and vectors are orthogonal with one another, we can transform the dataset by making the directions of the axes equivalent to the directions of the eigenvectors.

Vector sorting and dimension reduction takes place before the dataset is projected onto the principal components. The eigenvectors show the direction of our projection. The corresponding eigenvalues, in turn, show the important principal direction in interpretation the dispersion of the dataset, so that we can discard insignificant directions.

One of our objectives is to retain the main directions that contribute a lot to the spread in the dataset. We can achieve this by reducing the dimensionality. Then let's proceed to sort the eigenvectors and eigenvalues in descending order of importance.

When performing sorting, make sure that the sorted list with eigenvectors picked out likewise as our sorted list with eigenvalues. The highest eigenvalues should be at the top of the list, as they tell us the largest scatter in the dataset.

Figure 6 shows the scatter expression of the dataset. The eigenvalue of the down- targeted eigenvector transcends the eigenvalue of the up- targeted vector. For our example, the complete deposit of the eigenvectors to the variance is as follows:

$$[0.76318124; 0.23681876]$$

In the dataset, 76.3% of the changes are explained by the first eigenpair, while the second explains just 23.7%. Together these eigenpairs describe 100% of the scatter. Applying PCA to reduce dimension gives a chance to go ahead only with vectors with the highest contribution [11].

In case it is necessary to reduce the dimensionality to unity, it would be necessary to go further and take for the data projection an eigenvector with a contribution of 0.763. This means a loss of 0.237 data about the scatter. However we would gain a smaller amount of measurements in return.

In order to project our data onto the principal components, we will change the axes by equating them to eigenvectors.

In Figure 7, we have projected the data onto a single eigenvector. After projecting, just the x-axis has values, so the feature space has shrunk to one measurement. Thus, we reduced the dimension by using PCA [11].

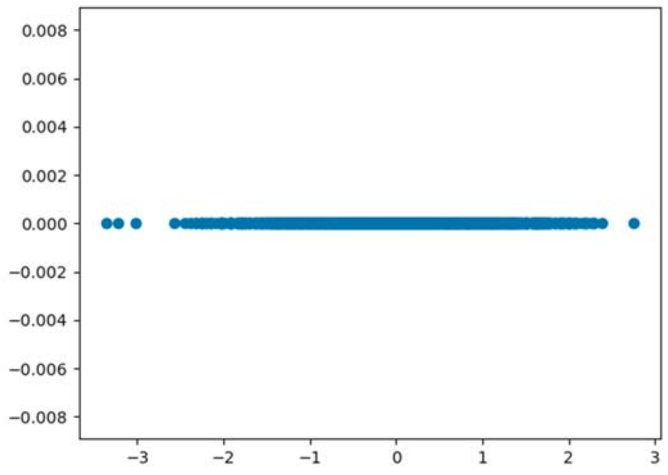

Fig. 7. Projecting a dataset onto an eigenvector

III. EIGENVECTOR DECOMPOSITION OR SINGULAR VALUE DECOMPOSITION

Two methods are used to scatter data using eigenpairs: Eigenvector Decomposition ("EIG") and Singular Value Decomposition ("SVD") [13]. Matrix decomposition is a beneficial implement for diminishing a matrix to its composite portions with an eye to streamline a number of more complicated procedures. The decomposition, which decomposes the matrix into eigenvectors and eigenvalues, is the most popular form of matrix decompositions. A singular decomposition is a decomposition of a real matrix with the aim to reduce it to a canonic form. Singular value decomposition is an easy way for operating with matrices. It displays the geometrical pattern of a matrix and allows depicting the affordable data.

A. *Eigenvector Decomposition*

If the vector v is an eigenvector of a square matrix *A*, it must be expressed in the following form: $Av = \lambda v$. At this time, λ is called the eigenvalue corresponding to the eigenvector v, and the group of eigenvectors of the matrix is a group of orthogonal vectors. The eigenvalue decomposition consists in decomposing the matrix into the following form: $A = Q\Sigma Q^{-1}$, where *Q* is a matrix made up of the eigenvectors of this matrix A, Σ is a diagonal matrix, and each element on the diagonal is an eigenvalue.

The decomposed Σ -matrix is a diagonal matrix. The eigenvalues are ordered from larger to smaller. The eigenvectors corresponding to these eigenvalues describe the direction of change of the matrix – from major to minor changes.

When a matrix is multidimensional, then that matrix is a linear transformation in a multidimensional space. This linear change cannot be represented by images, but it is possible that this transformation also has multiple transformation directions. The first *N* eigenvectors obtained by value decomposition correspond to the most important *N* directions of change of this matrix. We can approximate this matrix (transformation) using the first *N* changing directions.

So, eigenvalue expansion allows you to get eigenvalues and eigenvectors. The eigenvalue indicates how important the feature is, and the eigenvector indicates what the feature is. Each eigenvector can be understood as a linear subspace. You can do a lot with these linear subspaces. However, eigenvalue decomposition also has many limitations: for example, the transformed matrix must be a square matrix.

Let's work with the popular Iris dataset, part of the free scikit-learn machine learning library. This set is often used for classification and clustering tasks. There are four sample features in this dataset: tepal length (0), tepal width (1), petal length (2) and petal width (3).

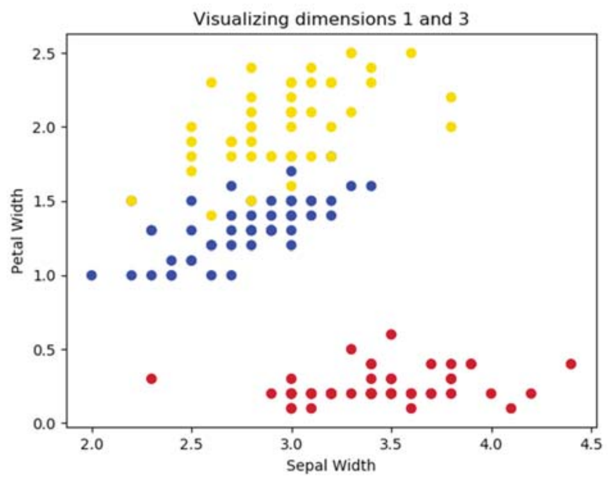

Fig. 8. Visualization of measurements 1 and 3

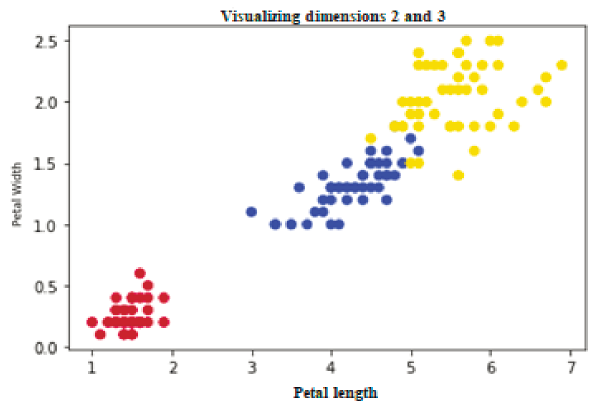

Fig. 9. Visualization of measurements 2 and 3

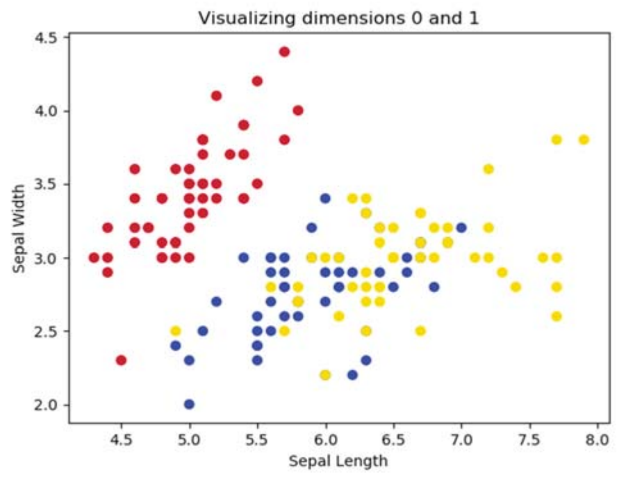

Fig. 10. Visualization of measurements 0 and 1

The figures demonstrate that two iris flowers cannot be linearly separated, however it is possible to separate this group from another iris flower. We have only 150 samples, but our feature space is four-dimensional. This is a case where feature extraction can be useful for training our model [10].

We equate the standard deviation (σ) to one, and the average value of the sequence of numerical data to zero ($\mu = 0.0$), by doing $x = \frac{x-\mu}{\sigma}$ for each dimension. As a result, we have changed the way values are displayed in the model space [14].

The following stage is to estimate the covariance matrix for the dataset. The covariance matrix in probability theory is a matrix made up of the pairwise covariances of the elements of one or two random vectors [8].

A variable. X is a mathematical representation of one measuremen measurement t of the dataset. Supposing that X presents petal width, numbers that present the petal width for one flower, can be also explained by the variable X.

Variable mean. Since the width of the lobe is dim (dimensionality of space points) = 3 in the visualization above, with an average value of 1.1993, it's possible to catch sight of how the numbers upward fit.

Variance (dispersion). Shows the «distribution» of data nearly the variable: $(x-\mu)^2$, where μ is expected value.

Covariance. Describes the direction of the linear dependence between variables:

$Cov(x, y) = (x-\mu_x)(y-\mu_y)$, where μ is expected value.

Covariance matrix for n variables. A covariance matrix for two dimensions X и Y looks as follows:

$$\begin{bmatrix} Cov(X,X) & Cov(X,Y) \\ Cov(Y,X) & Cov(Y,Y) \end{bmatrix}$$

Covariance matrix properties:

$Cov(X,X) = Var(X), Cov(X,Y) = Cov(Y,X)$.

So, our covariance matrix can be written as follows:

$$\begin{bmatrix} Var(X) & Cov(X,Y) \\ Cov(Y,X) & Var(Y) \end{bmatrix}.$$

In the matrix above, we see that the size of the covariance matrix is $n \times n$. It is essentially a symmetric matrix, i.e. a quadrature matrix that is equal to its transposition. The terms that construct a covariance matrix are called variations of a given variable that forms the diagonal of the matrix, or covariances of the two variables that fill the rest of the space. The j variable covariance with the k variable is equal to the covariance of the k variable with the j variable, i.e. «sjk»= «skj» [9].

With EIG-PCA we have the opportunity to expand the covariance matrix into eigenvectors and eigenvalues: $C = VLVT$, where V is the matrix of eigenvectors, is a diagonal matrix with eigenvalues and VT is the transpose of V.

Eigenvector matrix:

$$\begin{bmatrix} 0.52103086 & -0.37921152 & -0.71988993 & 0.25784482 \\ -0.27132907 & -0.92251432 & 0.24581197 & -0.12216523 \\ 0.57953987 & -0.02547068 & 0.14583347 & -0.80138466 \\ 0.56483707 & -0.06721014 & 0.63250894 & 0.52571316 \end{bmatrix}$$

Eigenvalues: 2.91912926, 0.91184362, 0.144265, 0.02476212.

In explaining variance, each principal dimension contributes, as follows: $[0.72978232; 0.2279609; 0.03606625; 0.00619053]$.

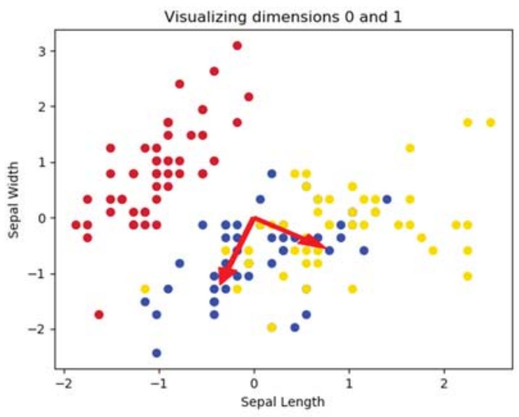

Figure 11 – Visualization of measurements 0 and 1

There are 73% is the contribution of the first principal dimension, 23% is the contribution of the second principal dimension. By changing the dimensionality to two, we get the dispersion description equal to 96% (73% + 23%).

Eigenpairs are sorted by eigenvalues: eigenvalues and corresponding eigenvectors must be sorted in descending order.

Thus, we have sorted eigenpairs, as we can see from the eigenvalues: 2.919129264835876, 0.9118436180017795, 0.14426499504958146, 0.024762122112763244.

The projection matrix projects a vector of observed values onto a vector of fitted values. It describes the effect of each observed value on each fitted value. Graph of the projected data is shown in Figure 12.

In this way, we have decreased the dimension to two without a big loss of data in the dataset.

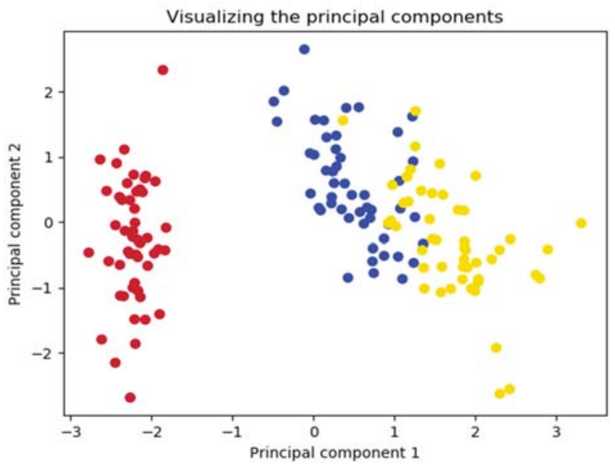

Fig. 12. Visualization of the principal components

B. Performing SVD on a data matrix

A convenient tool for decomposing a covariance matrix into eigenvectors and eigenvalues is the singular expansion. A singular decomposition is a representation of a matrix as a product of three matrices of a special form. Let M be a matrix of size $m \times n$, then it is represented by SVD as follows: $M = ULV^*$, where L is a matrix of size $m \times n$ with singular numbers on the main diagonal and the rest of its elements are zero, $V(n \times n)$ and $U(m \times m)$ are right and left singular matrices, V^* is a conjugate-transformed matrix to V (in case of real matrices equivalent to transpose). Omitting mathematical calculations, we note that the matrices U and V^T consist of eigenvectors of matrices $M * M^T$ and $M^T * M$, respectively.

If we zero some singular number in L, its corresponding eigenvector will be excluded from the product ULV^T. Hence, in order to apply the principal component method using SVD, the matrix $X = ULV^T$ must be decomposed. Depending on the problem, the components to get rid of are determined (usually the directions with the smallest variance). The singular numbers in L corresponding to these directions. These directions should be zeroed and a new value $ULV^T = X'$, where X' is the new matrix, which contains the data projected onto several principal components.

Let's look translating SVD outputs to usable vectors and values. The eigenvectors of the covariance matrix are as follows:

$$\begin{bmatrix} 0.52103086 & -0.37921152 & -0.71988993 & 0.25784482 \\ -0.27132907 & -0.92251432 & 0.24581197 & -0.12216523 \\ 0.57953987 & -0.02547068 & 0.14583347 & -0.80138466 \\ 0.56483707 & -0.06721014 & 0.63250894 & 0.52571316 \end{bmatrix}$$

We can collate them to the output of vh:

$$\begin{bmatrix} 0.52106591 & -0.26934744 & 0.5804131 & 0.56485654 \\ -0.37741762 & -0.92329566 & -0.02449161 & -0.06694199 \\ 0.71956635 & -0.24438178 & -0.14212637 & -0.63427274 \\ 0.26128628 & -0.12350962 & -0.80144925 & 0.52359713 \end{bmatrix}$$

As a result, with the exception of the sign, the columns of vh are equivalent to rows of the EIG-based eigenvectors.

Here we can also simply choose n components. As in the PCA-EIG script, we take $n = 2$ and consequently decrease the dimensionality from 4 to 2.

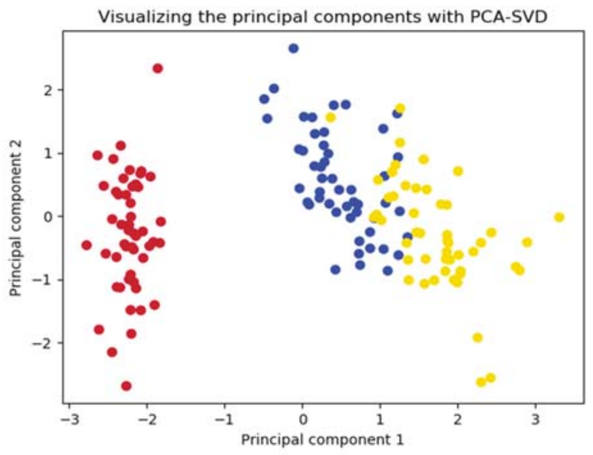

Fig. 13. Visualization of principal components using PCA-SVD

Now we can easily build a projection matrix like the one we did with PCA-EIG, project our data onto the main components, and plot the projection graph.

The final result is shown in Figure 13. It is identical to the result we obtained with the PCA-EIG approach.

Conclusion

To summarize, we reiterate that the use of PCA gives the opportunity to reconstruct the feature space with a lower dimensionality with a minimum of data loss. In this article, we have calculated the principal components and projected the dataset onto these components.

Eigenvector Decomposition (EIG) and Singular Value Decomposition (SVD) are two methods for obtaining eigenvectors, which were also discussed in our article. To find principal directions in a dataset, you need to compute eigenvalues, eigenvectors and then sort them to find principal directions in the dataset.

We have considered two approaches to perform the principal component method: PCA-EIG and PCA-SVD. PCA-EIG has been shown to work well with symmetric and quadratic matrices. However, it should not be forgotten that this method tends be numerically volatile. Because of this, PCA-SVD is often used in modern machine learning libraries. Singular value decomposition may be used on a matrix of standardized data to obtain eigenvectors. We ended up with the same final result as using PCA-EIG.

References

[1] V. A. Dokuchaev, “Digital Transformation: New Drivers and New Risks,” *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, 2020, pp. 1-7, doi: 10.1109/EMCTECH49634.2020.9261544.

[2] V.A. Dokuchaev, A.A. Kalfa. V.V. Maklachkova (2020). Architecture of Data Centers. Moscow: Hot Line-Telecom. 240 p. ISBN 978-5-9912-0849-9.

[3] V. A. Dokuchaev, V. V. Maklachkova, V. Yu. Statev, “Data subject as augmented reality,” *SYNCHROINFO JOURNAL*, vol.6, no.1, 2020, pp.11-15, doi: 10.36724/2664-066X-2020-6-1-11-15.

[4] S.V. Pavlov, V.A. Dokuchaev, V.V. Maklachkova, and S.S. Mytenkov, “Features of supporting decision making in modern enterprise infocommunication systems,” *T-Comm*, vol. 13, no. 3, pp. 71-74, 2019, doi: 10.24411/2072-8735-2018-10252.

[5] V.Yu. Statev, V.A. Dokuchaev, V.V Maklachkova, “Information security in the big data space”. *T-Comm*, vol. 16, no.4, 2022, pp. 21-28. (in Russian).

[6] V. A. Dokuchaev, E. V. Gorban and V. V. Maklachkova, "The System of Indicators for Risk Assessment in High-Loaded Infocommunication Systems," *2019 Systems of Signals Generating and Processing in the Field of on Board Communications*, Moscow, Russia, 2019, pp. 1-4, doi: 10.1109/SOSG.2019.8706726.

[7] S. V. Pavlov, V. A. Dokuchaev, V. V. Maklachkova, S. S. Mytenkov, “Features of supporting decision making in modern enterprise infocommunication systems”, *T-Comm*, vol. 13, no.3, 2019, pp. 71-74.

[8] X. Zhu, H. Dong, P. S. Rossi, M. Landro, «Feature Selection based on Principal Component Analysis for Underwater Source Localization by Deep Learning», Department of Electronic Systems, Norwegian University of Science and Technology, 2020, pp. 1-15, doi: 10.3390/rs13081486.

[9] Relationship between SVD and PCA. How to use SVD to perform PCA, StackExchange, Available: https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca.

[10] V.M. Efimov, K.V. Efimov, V.Y. Kovaleva, «Principal component method and its generalizations for sequences of any type», *Vavilovsky Journal of Genetics and Selection*, 2019, pp. 1032-1036, doi: 10.18699/VJ19.584.

[11] S. Raschka, «Principal Component Analysis», Available: https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html

[12] J. V. Lambers, «PCA Mathematical Fundamentals, Available: https://www.math.usm.edu/lambers/cos702/cos702_files/docs/PCA.pdf.

[13] N. Cristianini, J. Shawe-Taylor, «An Introduction to Support Vector Machines and other kernel-based learning methods», *Cambridge University Press*, 2000, pp. 687-689, doi: 10.1017/CBO9780511801389.

[14] The Complete Guide to Principal Component Analysis - PCA in Machine Learning, machinelearningmastery, Available: https://en.wikipedia.org/wiki/Covariance_matrix.

[15] Understanding the output of SVD when used for PCA, StackExchange, Available: https://stats.stackexchange.com/questions/96482/understanding-the-output-of-svd-when-used-for-pca.

[16] Why does Andrew Ng prefer to use SVD and not EIG of covariance matrix to do PCA, StackExchange, Available: https://stats.stackexchange.com/questions/314046/why-does-andrew-ng-prefer-to-use-svd-and-not-eig-of-covariance-matrix-to-do-pca.