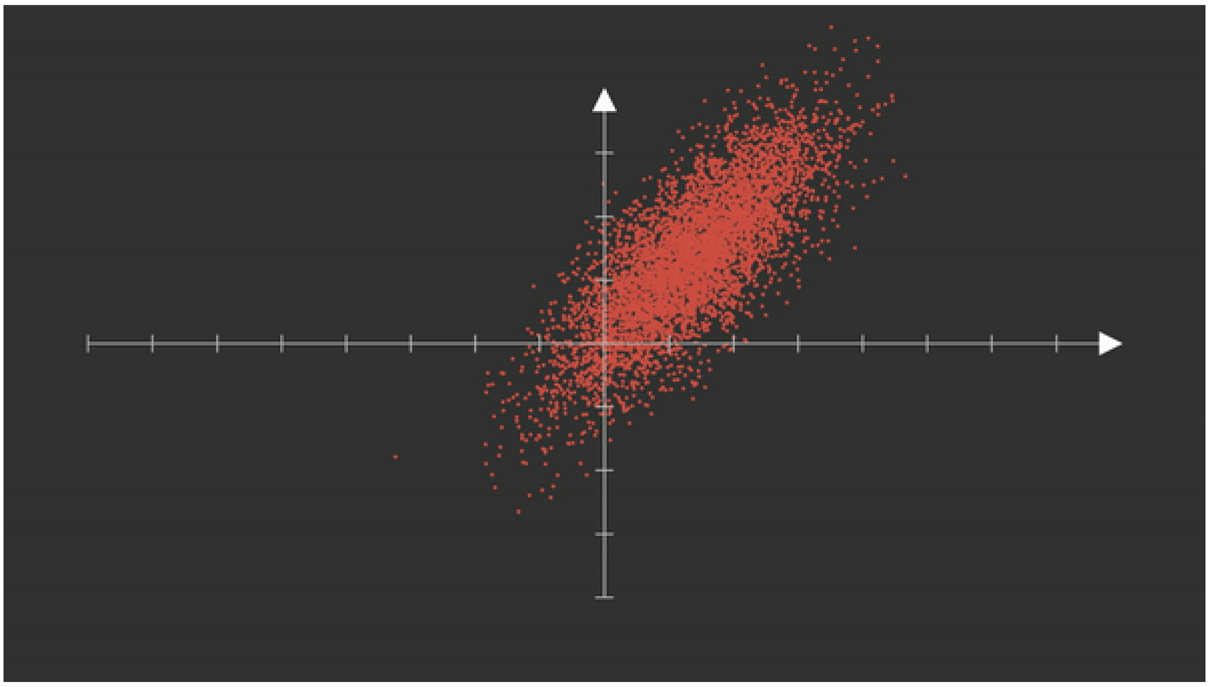
Published in Towards Data Science

Vincent Mueller Follow

Sep 18, 2021 · 9 min read · ✦ · Listen

Eigenvalues and eigenvectors in PCA

What do they tell us about our data?


(GIF by author)

Intro

I have learned about eigenvalues and eigenvectors in University in a linear algebra course. It was very dry and mathematical, so I did not get, what it is all about. But I want to present this topic to you in a more intuitive way and I will use many animations to illustrate it.

First, we will look at how applying a matrix to a vector **rotates** and **scales** a vector. This will show us what **eigenvalues** and **eigenvectors** are. Then we will learn about **principal components** and that they are the eigenvectors of the **covariance matrix**. This knowledge will help us understand our final topic, **principal component analysis**.

Matrix multiplication

To understand eigenvalues and eigenvectors, we have to first take a look at matrix multiplication.

Let's consider the following matrix.

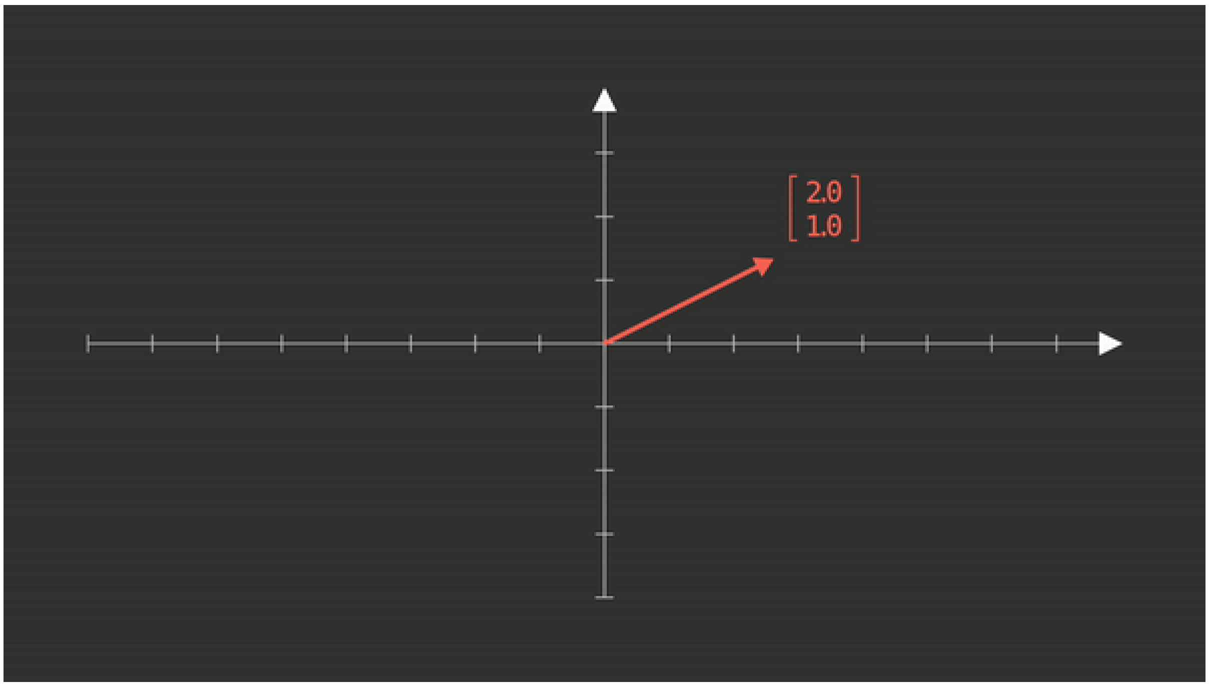
$$\begin{bmatrix} 0 & 1.4 \\ 1.4 & 0 \end{bmatrix}$$

(Image by author)

When we take the dot product of a matrix and a vector, the resulting vector is a **rotated** and **scaled** version of the original one.

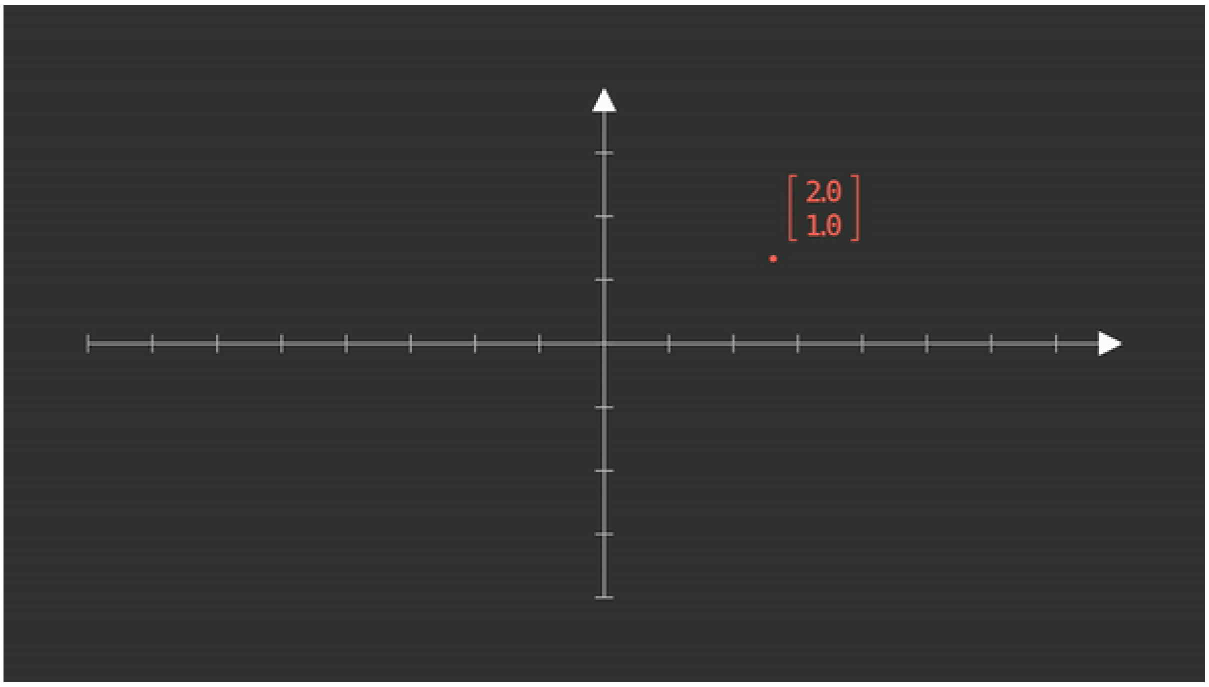
$$\begin{bmatrix} 0 & 1.4 \\ 1.4 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 2.8 \end{bmatrix}$$

(Image by author)



(GIF by author)

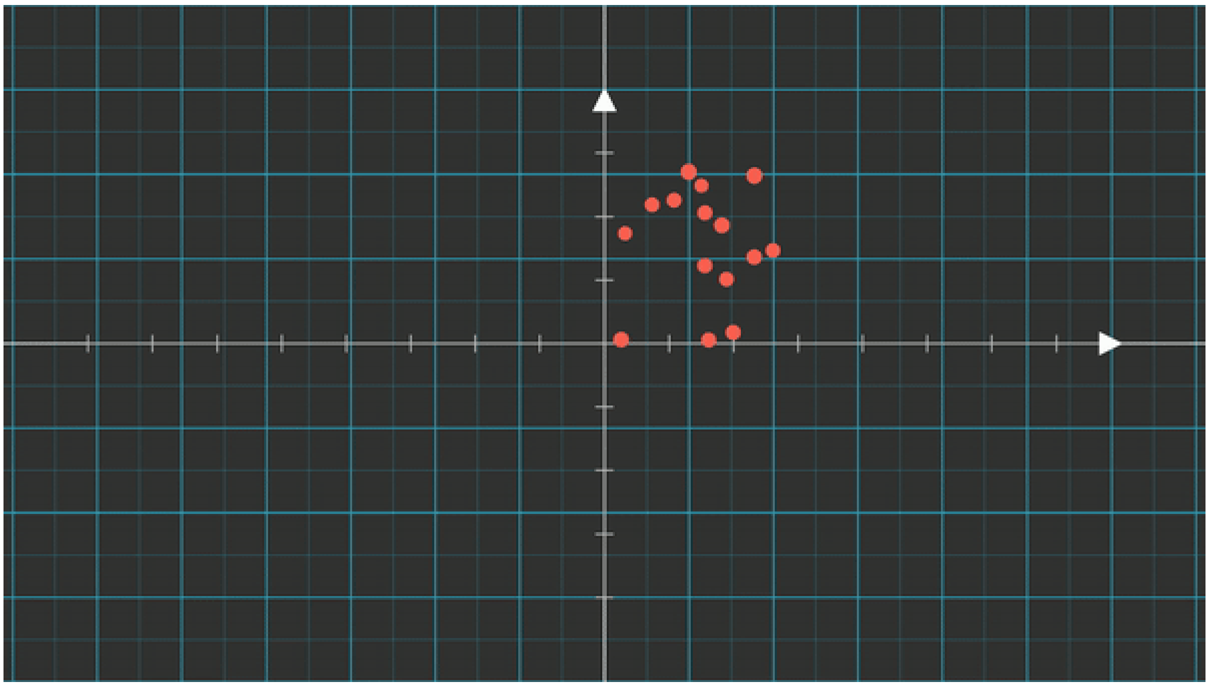
In data science, we mostly talk of data points, not vectors. But they are the same essentially and can be used interchangeably. And data points can also be transformed by matrix multiplication in the same way as vectors.



(GIF by author)

But even if matrix multiplication rotates and scales, it is a **linear** transformation.

Why is matrix multiplication a **linear** transformation? Consider a bunch of data points (denoted in red). Imagine a grid on which these points are located. When we apply the matrix to our data points and move the grid along with the data points, we see that the lines of the grid remain straight. If the lines would curve, then the transformation would be non-linear.


(GIF by author)

Eigenvectors

We consider the same matrix as above.

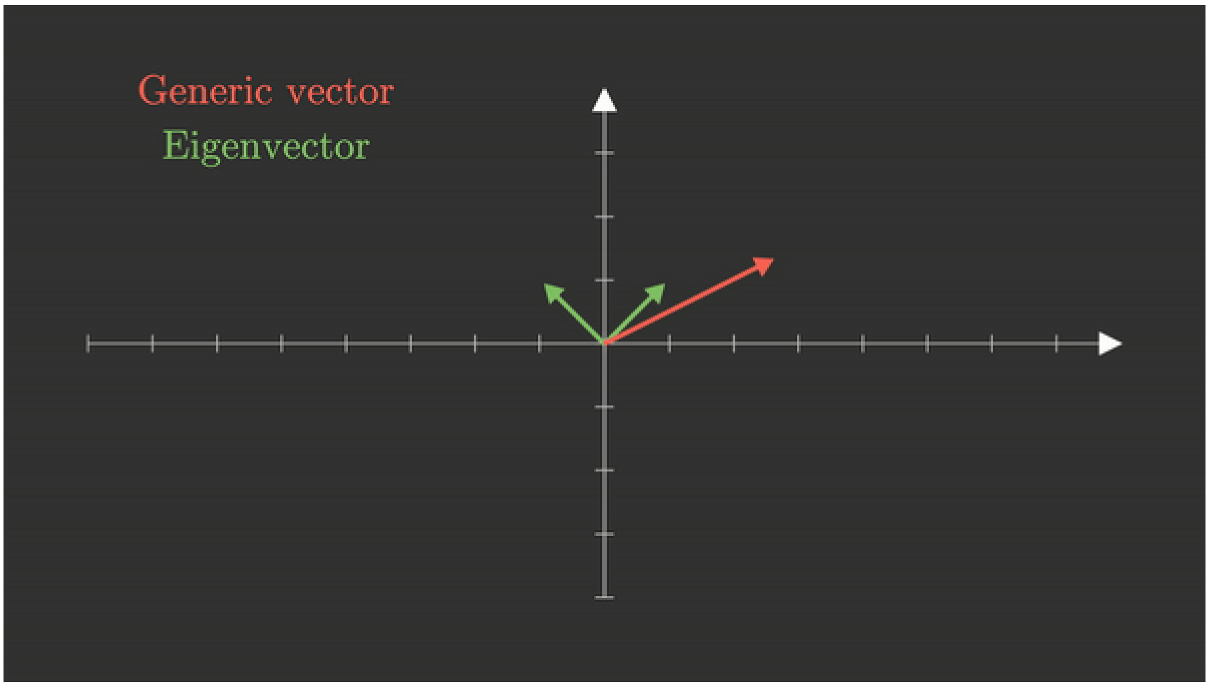
$$\begin{bmatrix} 0 & 1.4 \\ 1.4 & 0 \end{bmatrix}$$

(Image by author)

When applying this matrix to different vectors, they behave differently. Some of them may get **rotated and scaled**. Some of them **only rotated**, some of them **only scaled** and some of them may **not change at all.**

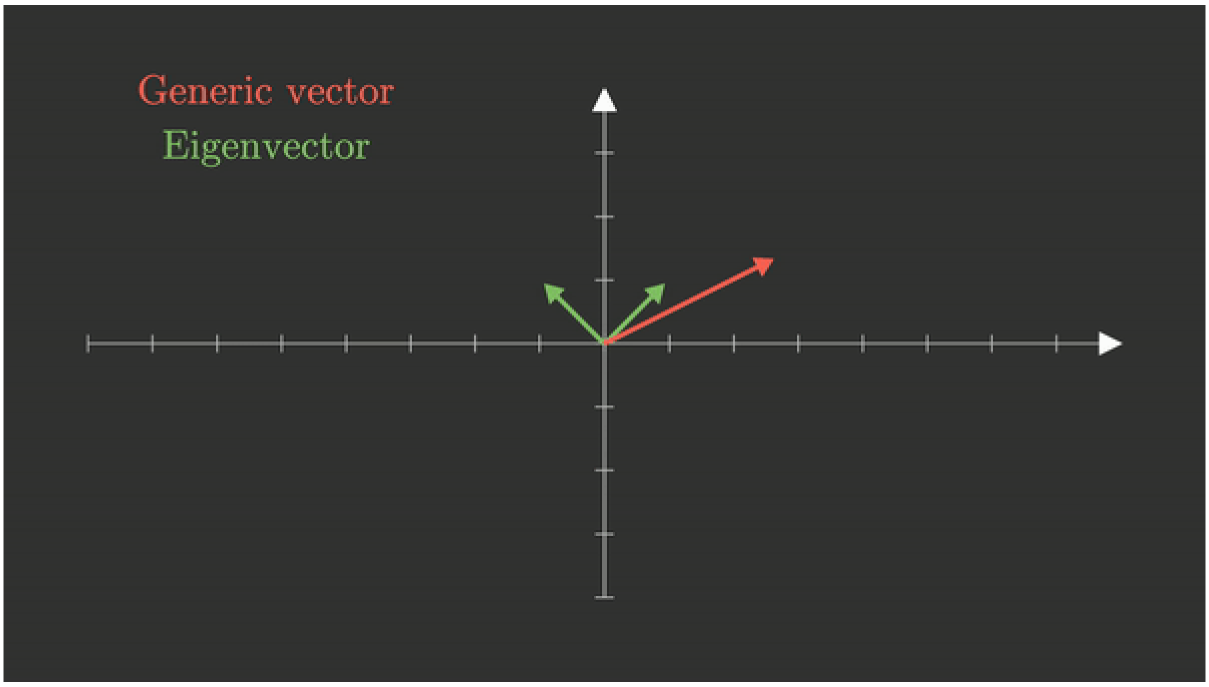
Eigenvectors are the vectors, which

- **only get scaled.**
- or do **not change at all.**



(GIF by author)

You can see, that the eigenvectors stay on the same line and other vectors(generic vectors) get rotated by some degree.



(GIF by author)

A 2x2 matrix has always two eigenvectors, but there are not always orthogonal to each other.

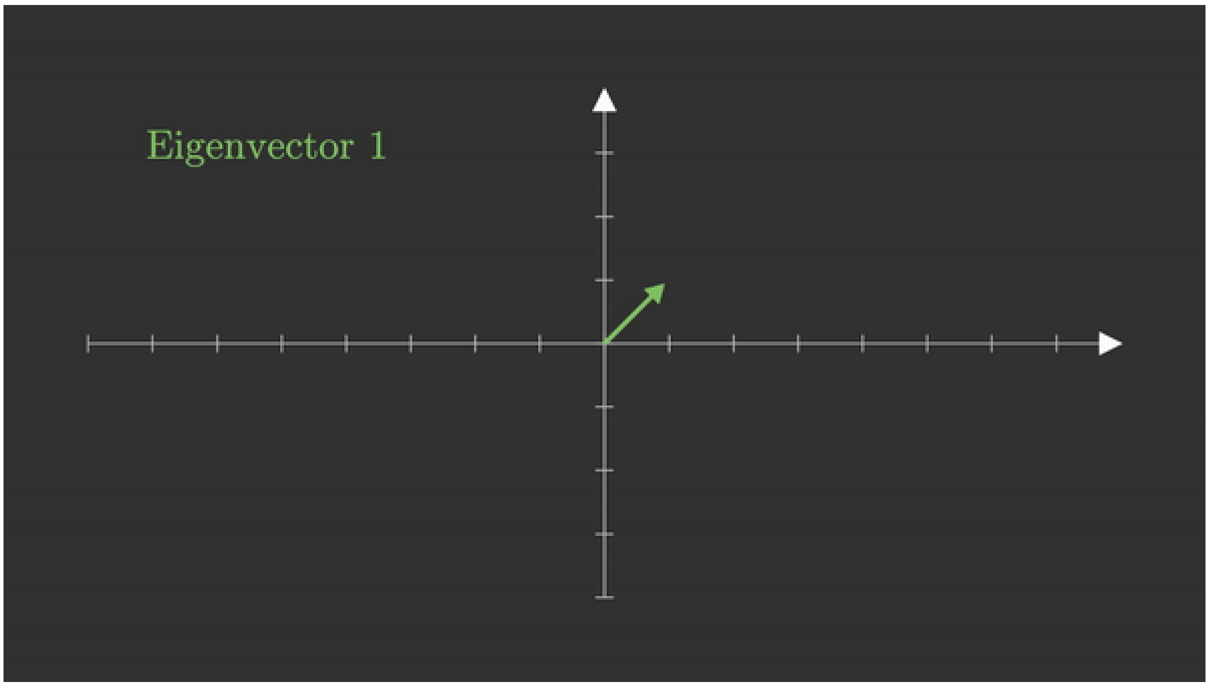
Eigenvalues

Each Eigenvector has a corresponding eigenvalue. It is the factor by which the eigenvector gets scaled, when it gets transformed by the matrix. We consider the same matrix and therefore the same two eigenvectors as mentioned above.

$$\begin{bmatrix} 0 & 1.4 \\ 1.4 & 0 \end{bmatrix}$$

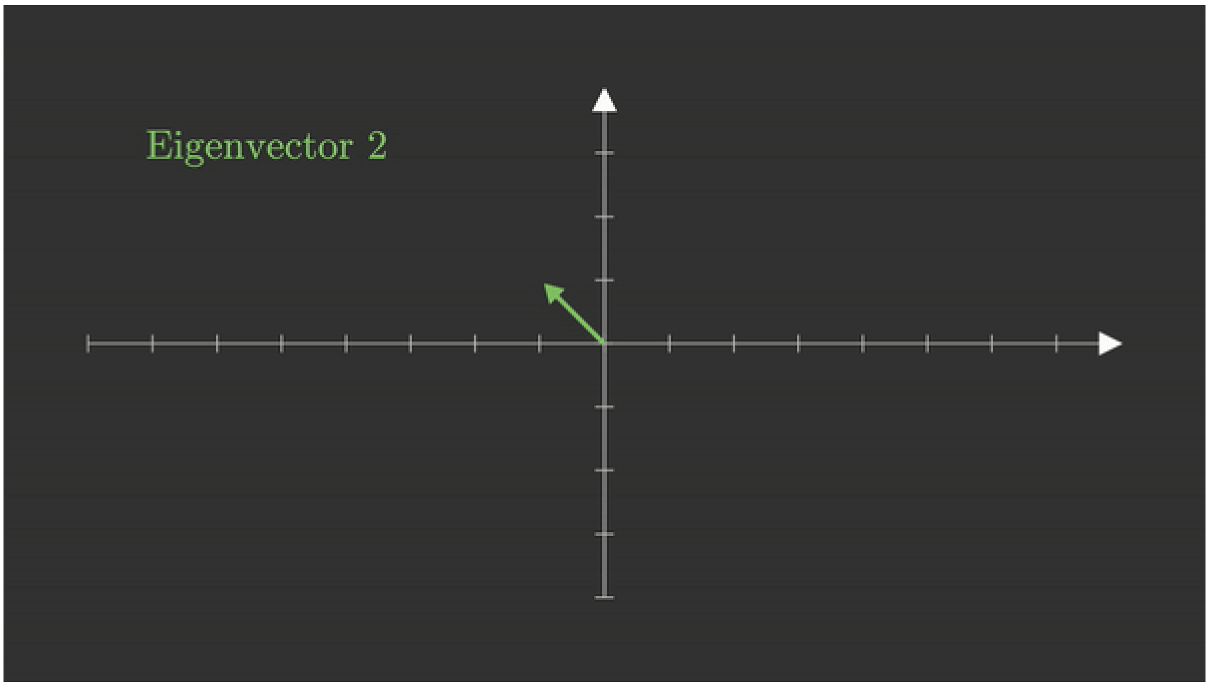
(Image by author)

One of the two eigenvectors of this matrix (I call it Eigenvector 1, but this is arbitrary) is scaled by a factor of 1.4.



(GIF by author)

Eigenvector 2 get’s also scaled by a factor of 1.4 but it’s direction get’s inverted. Therefore, eigenvalue 2 is -1.4.



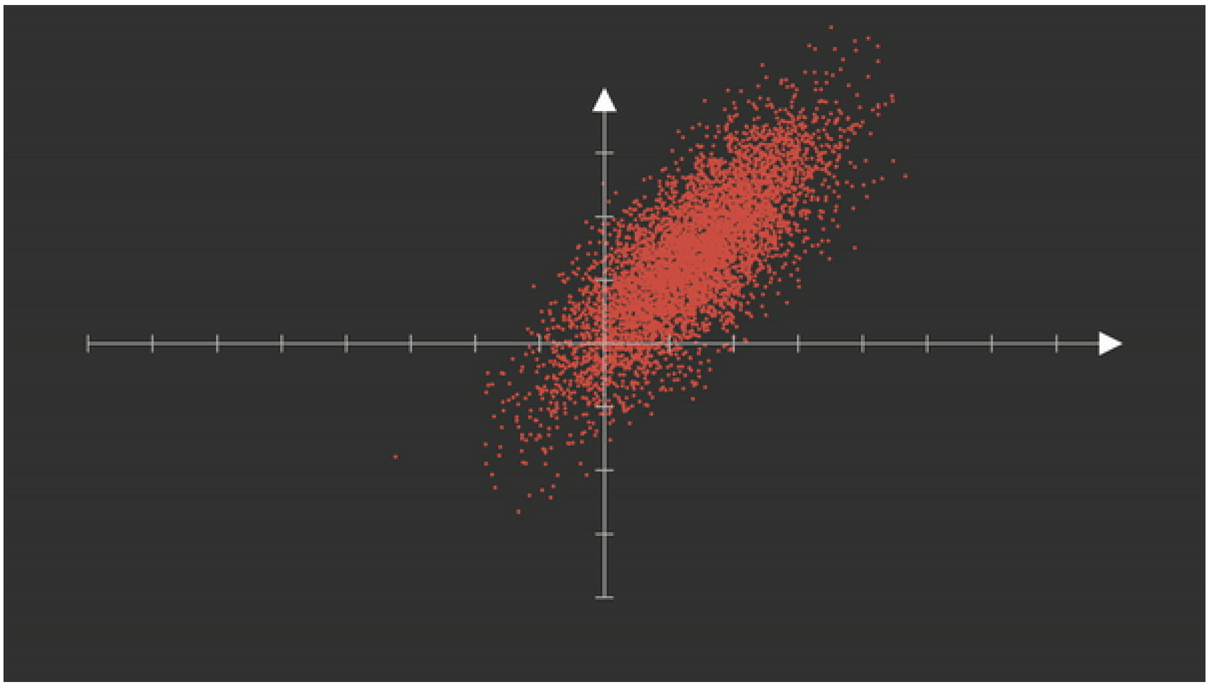
(GIF by author)

Applications of eigenvectors and eigenvalues values in Data Science

Principal components

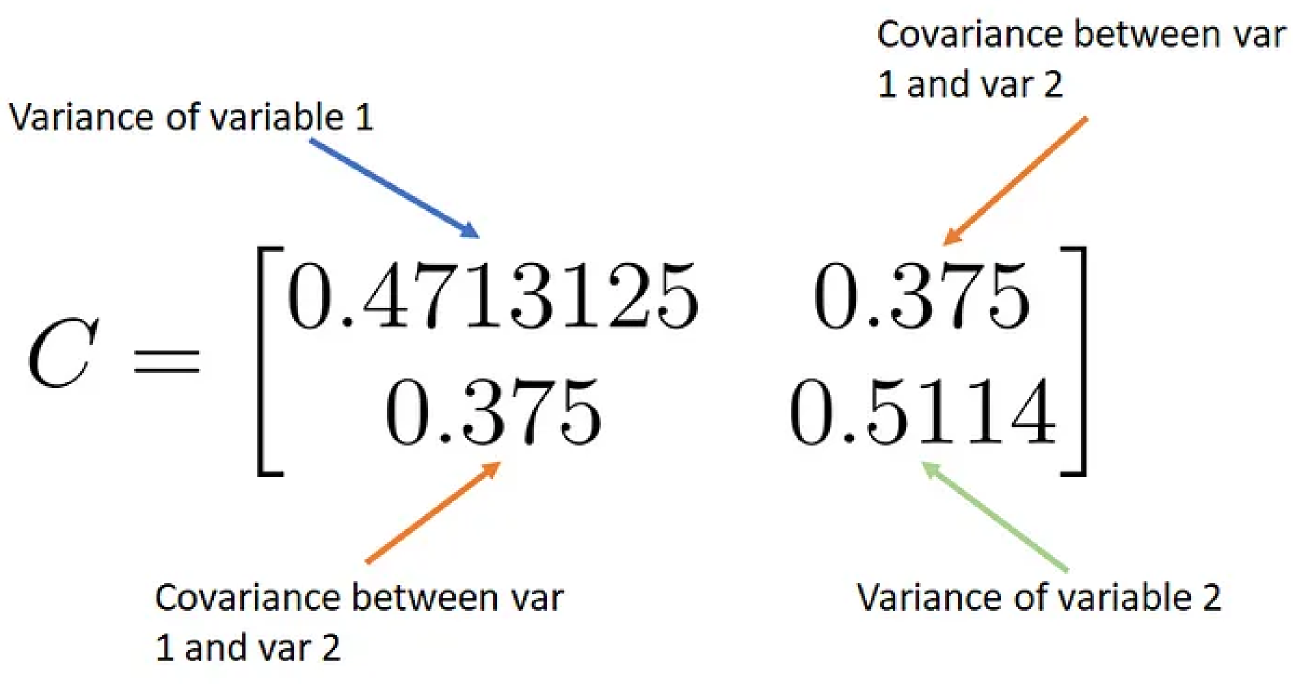
Using eigenvalues and eigenvectors, we can find the main axes of our data. The first main axis (also called "first principal component") is the axis in which the data varies the most. The second main axis (also called "second principal component") is the axis with the second largest variation and so on.

Let's consider a 2-dimensional dataset.



(GIF by author)

To find the principal components, we first calculate the Variance-Covariance matrix C.



$$C = \begin{bmatrix} 0.4713125 & 0.375 \\ 0.375 & 0.5114 \end{bmatrix}$$

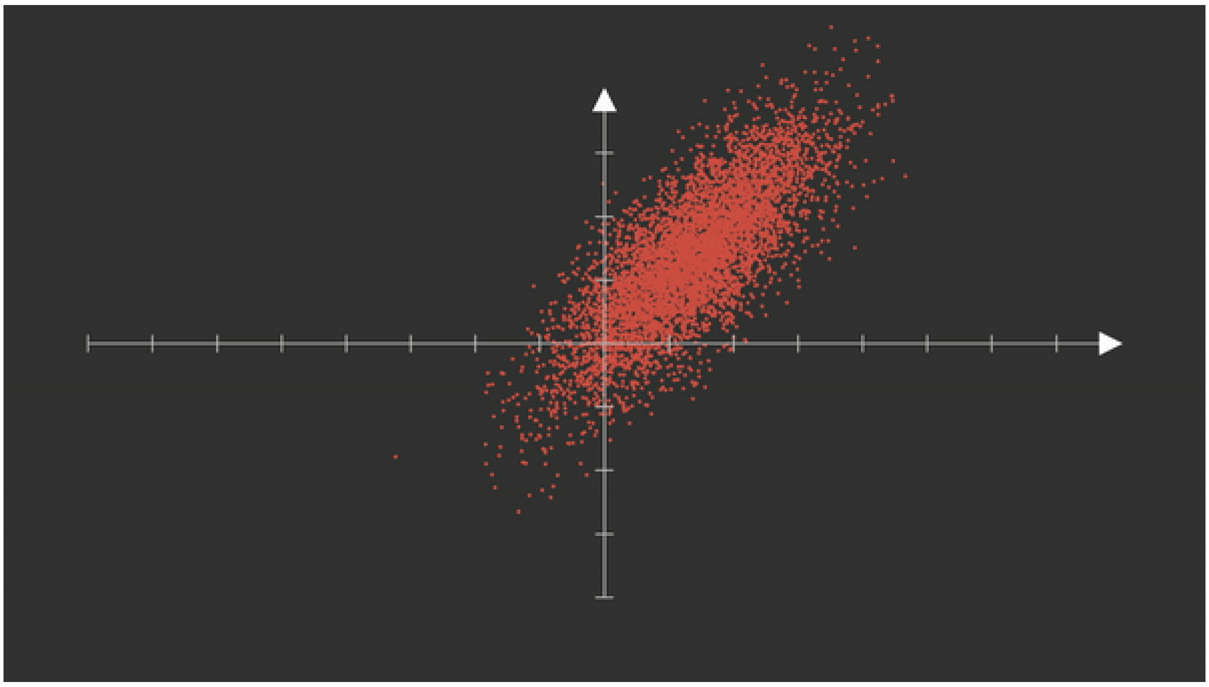
Variance of variable 1
Covariance between var 1 and var 2
Covariance between var 1 and var 2
Variance of variable 2

(Image by author)

We can use numpy to calculate them. Note that our data (X) must be ordered like a pandas data frame. Each column represents a different variable/feature.

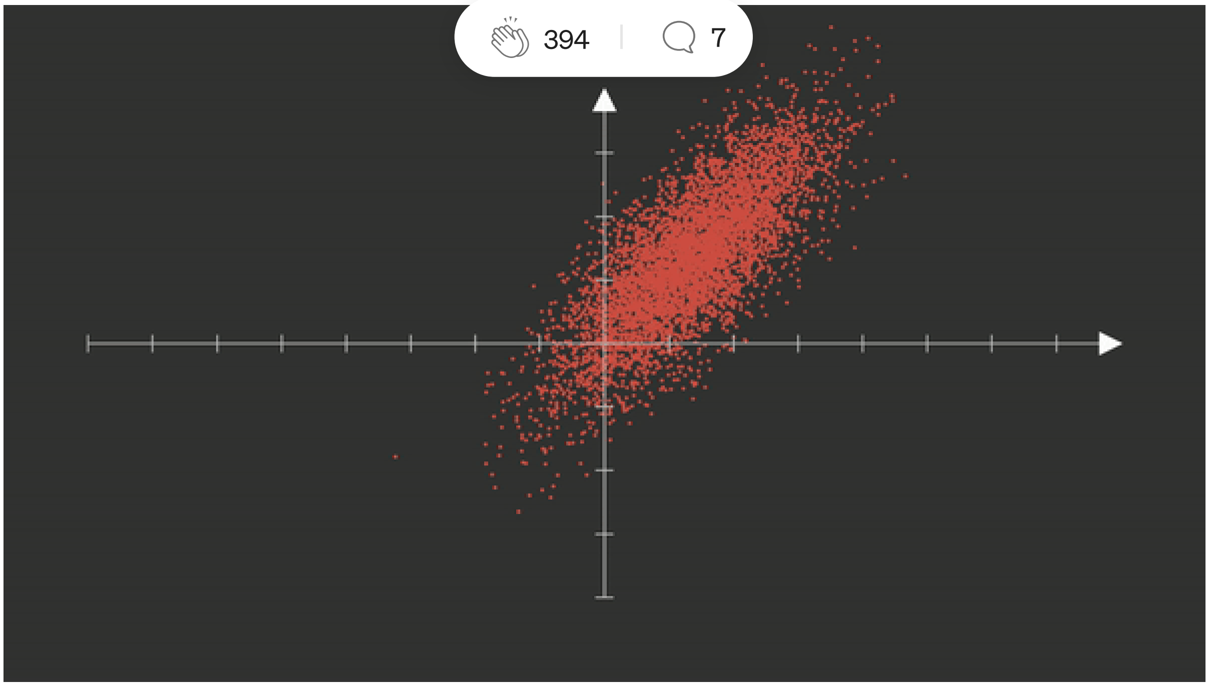
```
import numpy as np
eigenvalues,eigenvectors = np.linalg.eig(C)
```

The eigenvectors show us the direction of our main axes (principal components) of our data. The greater the eigenvalue, the greater the variation along this axis. So the eigenvector with the largest eigenvalue corresponds to the axis with the most variance.


(GIF by author)

We should remember, that matrices represent a linear transformation. When we multiply the Covariance matrix with our data, we can see that the center of the data does not change. And the data gets stretched in the direction of the eigenvector with the bigger variance/eigenvalue and squeezed along the axis of the eigenvector with the smaller variance.

Data points lying directly on the eigenvectors do not get rotated.


(GIF by author)

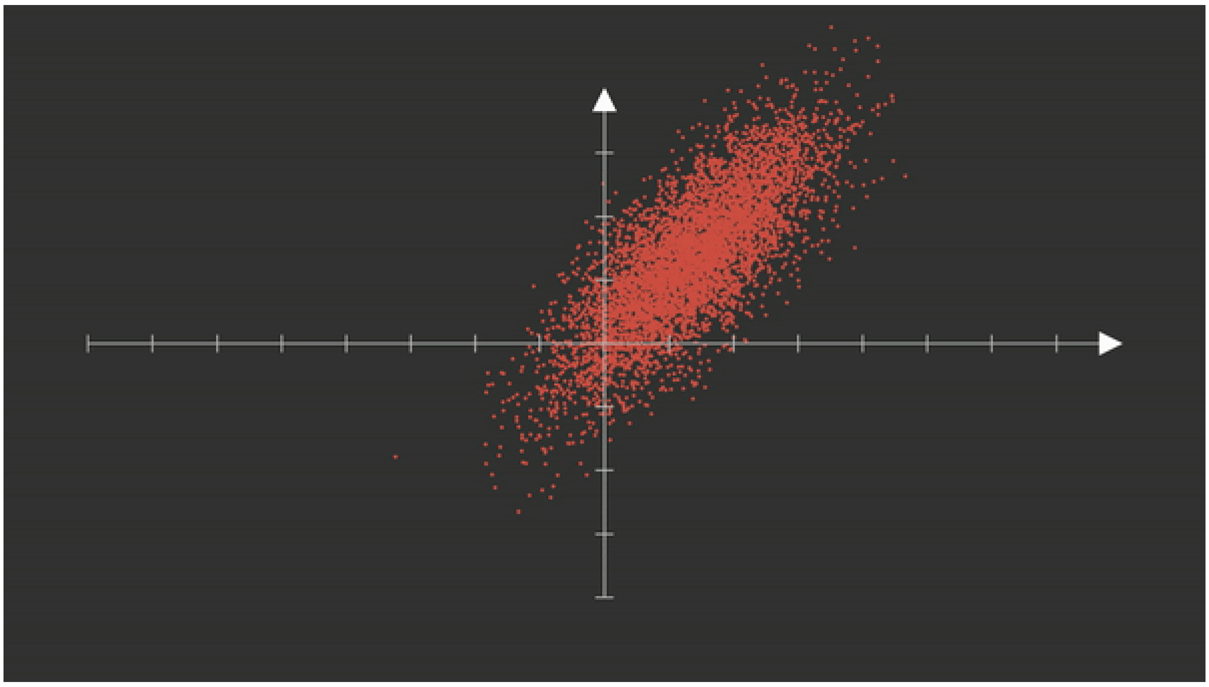
Principal component analysis (PCA)

Principal component analysis uses the power of eigenvectors and eigenvalues to reduce the number of features in our data, while keeping most of the variance (and therefore most of the information). In PCA we specify the number of components we want to keep beforehand.

The PCA algorithm consists of the following steps.

1. Standardizing data by subtracting the mean and dividing by the standard deviation
2. Calculate the Covariance matrix.
3. Calculate eigenvalues and eigenvectors
4. Merge the eigenvectors into a matrix and apply it to the data. This rotates and scales the data. The principal components are now aligned with the axes of our features.
5. Keep the new features which account for the most variation and discard the rest.

Let's look at what PCA does on a 2-dimensional dataset. In this example, we do not reduce the number of features. Reducing the number of features makes sense for high dimensional data because then it reduces the number of features.

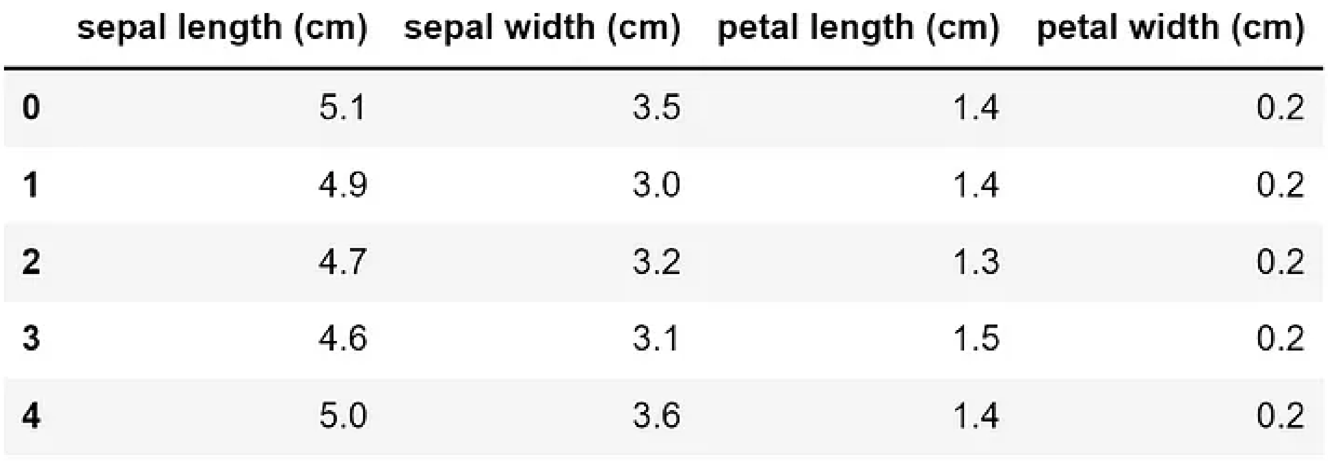

(GIF by author)

Let’s load the iris dataset. It contains measurements of three different species of iris flowers. Those species are iris-virginica, iris-versicolor and iris-setosa.

(Code by author)

Let’s take a quick glimpse at the dataset.

```
print(iris_df.head())
```

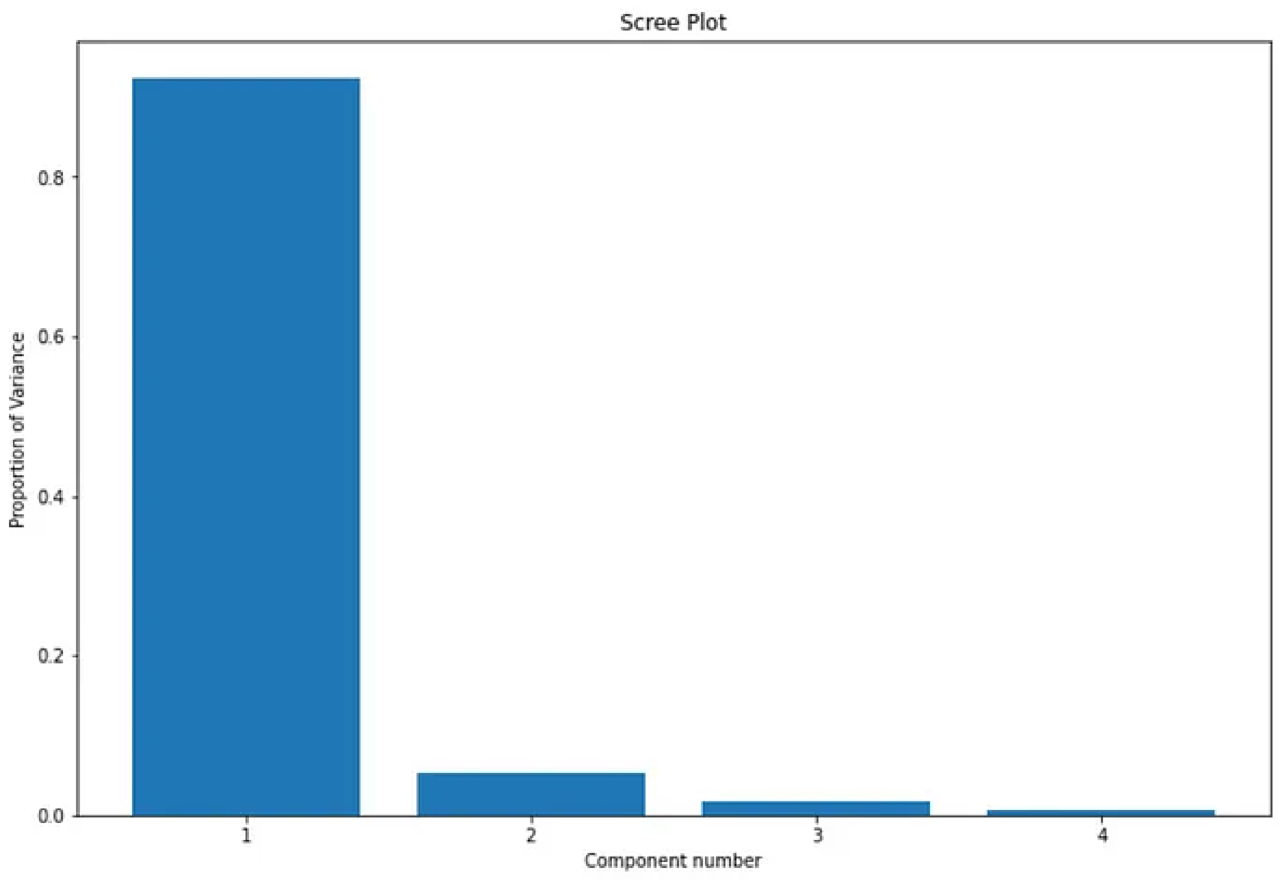


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

(GIF by author)

We can create a so called "scree plot" to look at which variables account for the most variability in the data. For this, we perform a first PCA.

(Code by author)

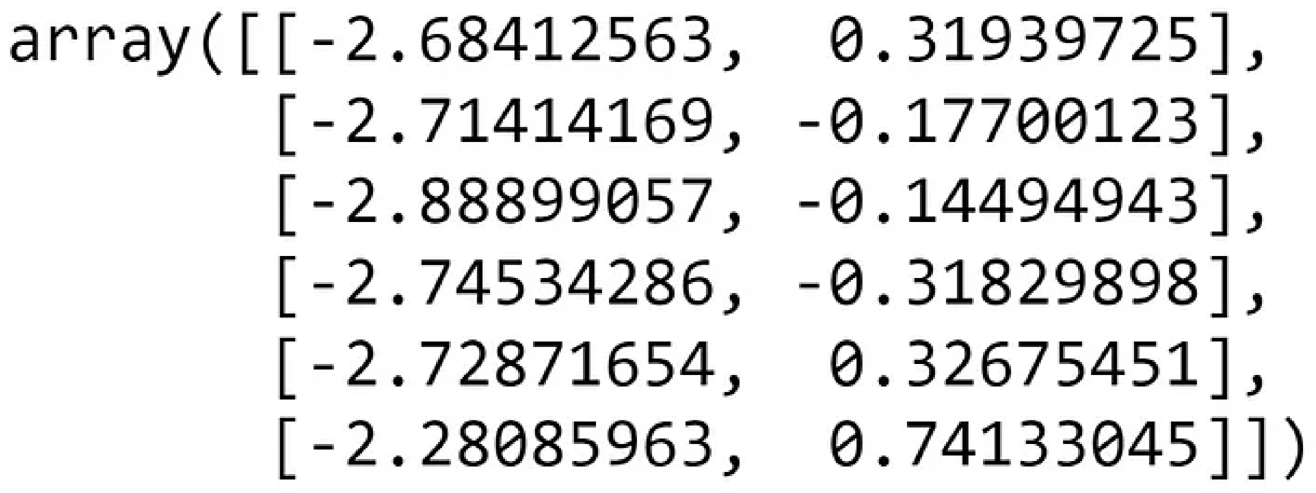


(Image by author)

As we can see, the first two components account for most of the variability in the data. Therefore I have decided to keep only the first two components and discard the Rest. When having determined the number of components to keep, we can run a second PCA in which we reduce the number of features.

We take a look at our data, which is an array now.

```
print(iris_transformed[:5,:])
```



```
array([[-2.68412563,  0.31939725],
       [-2.71414169, -0.17700123],
       [-2.88899057, -0.14494943],
       [-2.74534286, -0.31829898],
       [-2.72871654,  0.32675451],
       [-2.28085963,  0.74133045]])
```

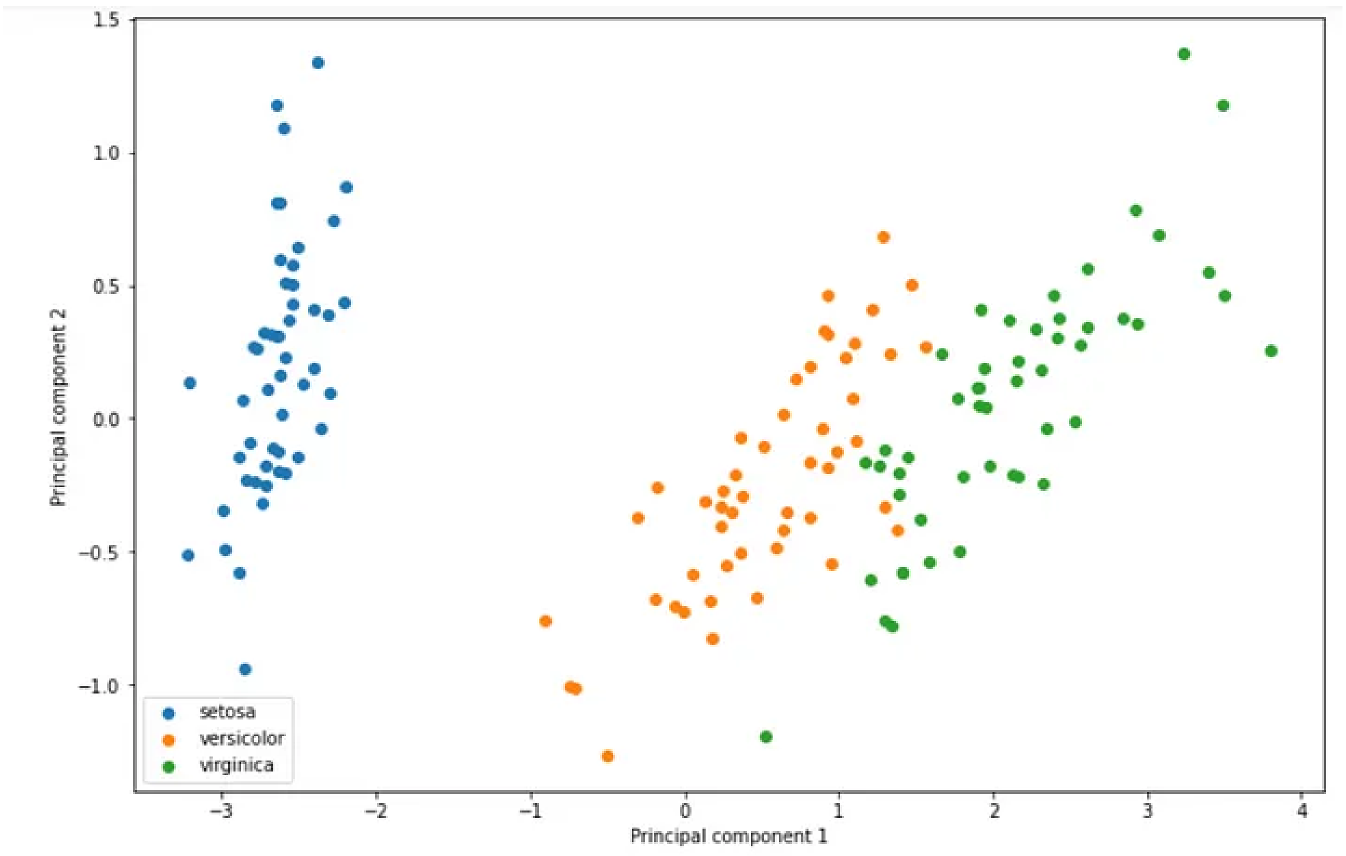
(Image by author)

We can see, that we have only two columns left. These columns/variables are a linear combination of our original data and do not correspond to a feature of the original dataset (like sepal width, sepal length, ...).

Let's visualize our data.

(Code by author)

We see our new combined features on the x and y axes. The plant species is indicated by the color of a data point.



(Image by author)

We can see, that much of the information in the data has been preserved and we could now train an ML model, that classifies the data points according to the three species.

Conclusion

I will now summarize the most important concepts.

Matrix multiplication

When we multiply a matrix with a vector, the vector get's transformed linearly. This linear transformation is a mixture of rotating and scaling the vector. The vectors, which get only scaled and not rotated are called eigenvectors. The factor by which they get scaled is the corresponding eigenvalue.

Principal components

Principal components are the axes in which our data shows the most variation. The first principal component explains the biggest part of the observed variation and the second principal component the second largest part and so on. The Principal components are the eigenvectors of the covariance matrix. The first principal component corresponds to the eigenvector with the largest eigenvalue.

PCA

Principal component analysis is a technique to reduce the number of features in our dataset. It consists of the following processing steps.

1. Standardizing data by subtracting the mean and dividing by the standard deviation
2. Calculate the covariance matrix.
3. Calculate eigenvalues and eigenvectors
4. Merge the eigenvectors into a matrix and apply it to the data. This rotates and scales the data. The principal components are now aligned with the axes of our features.
5. Keep as many new features as we specified and discard the rest. We keep the features which can explain the most variation in the data.

Sources

https://datascienceplus.com/understanding-the-covariance-matrix/

https://en.wikipedia.org/wiki/Iris_flower_data_set

https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html

Datasets

Iris

The Iris dataset and license can be found under:

https://www.openml.org/d/61

It is licensed under creative commons which means you can copy, modify, distribute and perform the work, even for commercial purposes, all without asking for permission.

Related articles

Einstein index notation

Einstein summation, index notation and numpys np.einsum

towardsdatascience.com

Backpropagation in Neural Networks

Neural Networks from scratch including math and python code

towardsdatascience.com

Matrix calculus for data scientists

Take the red pill and learn about matrix calculus!

towardsdatascience.com

Other articles by author

How you can use GPT-J

GPT-J explained and 3 easy ways how you can access it

towardsdatascience.com

Deep Q learning is no rocket science

Deep Q and double Q learning explained and coded in pytorch

towardsdatascience.com

Deep transfer learning

The art of reusing models trained by others

towardsdatascience.com

Want to connect?

Linkedin
https://www.linkedin.com/in/vincent-m%C3%BCller-6b3542214/
Facebook
https://www.facebook.com/profile.php?id=100072095823739
Twitter
https://twitter.com/Vincent02770108
Medium
https://medium.com/@Vincent.Mueller
You can become a Medium member and support me at the same time
https://medium.com/@Vincent.Mueller/membership

Math appendix

Calculating eigenvalues and eigenvectors

If you like mathematics and want to dive deeper, I have summarized some of the math used in this blog post.

We can easily calculate the eigenvectors and eigenvalues in python.

```
import numpy as np
eigenvalues,eigenvectors = np.linalg.eig(M)
```

If we want to calculate them by hand, it gets a little bit more complicated.

As we have seen, when we multiply the matrix **M** with an eigenvector (denoted by v), it is the same as scaling its eigenvalue λ.

$$\boldsymbol{A} \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v}$$

(Image by author)

We now rearrange the equation.

$$(\boldsymbol{A} - \lambda \cdot I) \cdot \boldsymbol{v} = \vec{0}$$

(Image by author)

Where **I** is the identity matrix, which has ones in the diagonal and zeros elsewhere. It has the same shape as **A**.

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \dots & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

(Image by author)

$$\vec{0} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

(Image by author)

And the only way for this equation to be true.

$$(\boldsymbol{A} - \lambda \cdot I) \cdot \boldsymbol{v} = \vec{0}$$

(Image by author)

This is only true when the determinant of the matrix (**A** -λ·**I**) becomes 0.

> *The determinant of a matrix is the factor by which the matrix scales the area in case of a 2x2 matrix and the volume in case of a 3x3 matrix. If the determinant is zero, then the matrix (**A** -λ·**I**) squeezes points to the origin (origin is the zero point). This is the only way for a non-zero vector to become a zero-vector.*

So we search for all eigenvalues λ, which make the determinant 0.

After we found the eigenvalues, we can solve this equation:

$$(\boldsymbol{A} - \lambda \cdot I) \cdot \boldsymbol{v} = \vec{0}$$

(Image by author)

And we find the eigenvectors.

Covariance matrix

The variance-covariance matrix can be estimated from data using the following formula:

$$\boldsymbol{C} = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T (\boldsymbol{X}_i - \bar{\boldsymbol{X}})$$

(Image by author)

Machine Learning Artificial Intelligence Mathematics Data Science

Data Visualization

edge research to original features you don't want to miss. Take a look.