Interpreting Vectors Geometrically

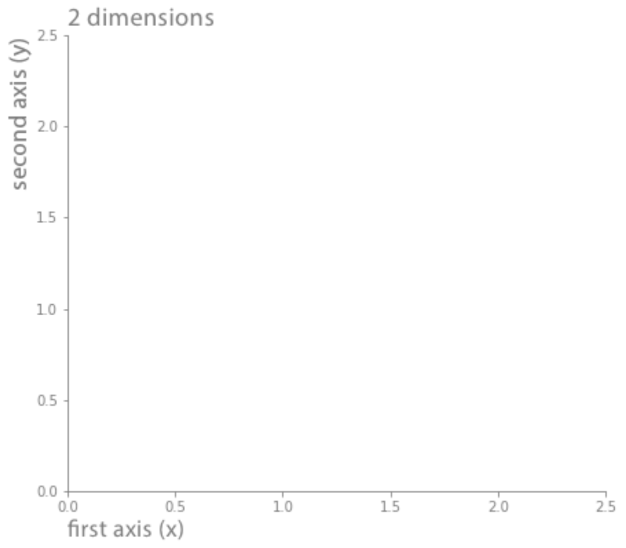
Module 0 | Chapter 2 | Notebook 2

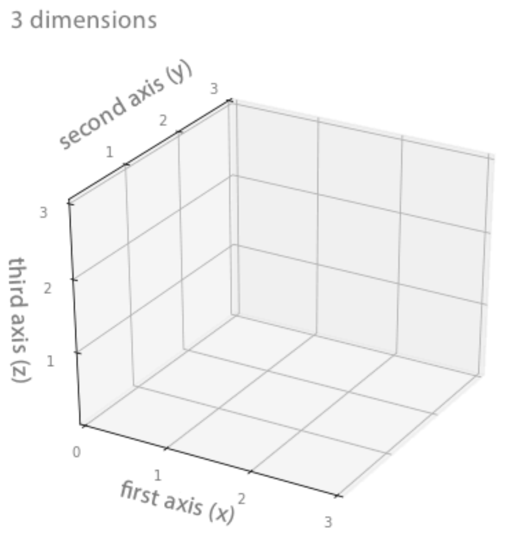
So far we have understood vectors primarily as data structures. In this notebook we will interpret them geometrically. This allows us to get a feeling for some data science methods and is fundamental for some other properties of vectors. You will then use this to calculate the similarity and write a function to make a wine recommendation. By the end of this exercise you will be able to:

- calculate the lengths of vectors,
- the angle between 2 vectors
- and the dot product

Properties of vectors

Scenario: 1001Wines is an online retailer that sells wines through its website. 1001Wines would like to recommend its customers high-quality wines that might be to their taste. They therefore want to create a recommendation tool to compare the similarity of purchased wines with other wines from the assortment. If a wine from the range is similar to what customers have already ordered, it should be recommended to them. To solve this task, we will deepen our understanding of vectors. Since the data on the wines was stored as vectors, this can help us in the next step to identify similar wines based on their characteristics.





In the last notebook we learned that vectors are a data structure where the data is lined up together in one direction. As a data type for this we have got to know the `ndarray`. However, vectors can also be interpreted geometrically. Then they describe the connection from a reference point, the coordinate origin, to a point in a plane or space. They therefore possess two properties: A length and a direction. We'll come back to that. This geometric interpretation leads us to another important concept: dimension. We know from the last lesson that in arrays, each dimension represents a direction along which data is arranged. This idea generally applies to the concept of dimensionality. A cinema screen has 2 dimensions, the height and the width. When we watch a film in 3D, we also add depth. A third axis (in the direction of the screen) on which the image information is located is created. We experience the world we live in as 3-dimensional. We perceive three axes: left/right, front/back and top/bottom. These axes are usually designated by the letters x, y and z. The coordinate systems would then look like this:

Coordinate systems in 2 and 3 dimensions

We can no longer represent more than 3 dimensions correctly, but we can still calculate them with the methods of linear algebra. A vector describes a direction to a point in space. To do this, it uses a number for each axis. Now, that might sound very abstract. Therefore we will first of all visually represent the vectors on a plane in two dimensions. This will help us later to get an idea of what we will be calculating when we determine the similarity of different wines.

First, we will work with vectors that contain only 2 numerical values. Such as the vector `[1, 2]`.

Now import `numpy` with its conventional alias and create an array called `vec` with these values.

In [32]:
```
import numpy as np
vec = np.array([1,2])
```

We will now use this vector to describe the direction towards a point on a plane. We will interpret the values of the vector so that each number of the vector indicates how many steps must be taken on an axis to reach the point. So in the case of `vec`, we have to take one step on our first axis (typically the x-axis) and 2 steps on the second axis (typically the y-axis). So that would look like this:

Vector as details of steps in x and y direction

`vec` describes the direction from the point with the coordinates `[0, 0]` to the point with the coordinates `[1, 2]`. From the last lesson you know that the sum of two vectors is formed from the sums of the values assigned to the same axis. So the steps in each direction are added up. This is like concatenating the individual vectors. Create a new vector `vec_2` with the values `[2 1]` and add it to `vec`. Store the result in `vec_sum` and print it.

In [33]:
```
vec_2 = np.array([2,1])
vec_sum = vec + vec_2
vec_sum
```

Out[33]:
```
array([3, 3])
```

The individual values were added up together according to the direction they represent. The point to which the new arrow points is the result of concatenating the two vectors.

Sum of vectors as concatenation

Lengths of vectors: The norm

Along with the direction, `vec` also describes the distance between the start point and the end point of the arrow. The distance is the same as the length of the vector. The length of a vector is calculated by squaring each element and then adding them together. At the end, you take the square root of the sum. The code for that looks like this:

In [34]:
```
vec_length = 0
for i in range(len(vec)): # loop throug every value in vec
    vec_length = vec_length + (vec[i]**2) #sum up the squares of the values
vec_length = np.sqrt(vec_length) #take the square root with np.sqrt()
print(vec_length)
```

```
2.23606797749979
```

This code describes the following formula for the length of a vector v with n entries:

\begin{equation} ||\vec{v}|| = \sqrt{v_1^2 + v_2^2 + ... + v_n^2} \end{equation}
The arrow indicates that it is a vector. The vertical lines indicate that the length of the vector is meant. The letters with the index describe the relevant elements or components of the vector. For $\vec{v}=[2,1]$, v_1 denotes the first component $v_1=2$, and $v_2=1$ denotes the second component.

This calculation is performed by the `numpy` function `norm()`. It is located in the submodule `numpy.linalg`. This provides us with many practical functions in the field of linear algebra. If you only want to import a certain function from a module or submodule, you can use the following syntax: `from module import function`. After that, you can call `function` directly without having to write the `module` specification before it. Import the `norm()` function from the submodule `numpy.linalg` and use it to calculate the length of `vec`. Do you get the same result?

In [35]:
```
from numpy.linalg import norm
norm(vec)
```

Out[35]:
```
2.23606797749979
```

Congratulations! You now know how to add vectors and determine their length. However, we will deal with the concept of norms again later.

Dot product

You have now got to know two properties of vectors. They indicate a direction and they describe the length. We can already use this to compare the vectors. For this we need the angle between them. If the angle between them is small, the vectors point in a similar direction. If the angle is 90°, the directions have actually nothing to do with each other. Then the vectors are what we call *orthogonal* to each other. This would be the case, for example, if one vector lies completely on the x axis and the other on the y axis.

To calculate the angle between two vectors, we need the **dot product**, sometimes referred to as the **scalar product**.

When forming the dot product, the entries of the two vectors are multiplied according to their order and the products are then added up. This is very similar to the length calculation. The dot product of a vector with itself is the same as the squared length of this vector. The dot product between two vectors u and v with n entries as a formula looks like this:

\begin{equation} \vec{u} \cdot \vec{v} = u_0 \cdot v_0 + u_1 \cdot v_1 + .. + u_n \cdot v_n \end{equation}
A specific example can help to better understand this formula

\begin{equation} \begin{bmatrix} {\color{red}3}\\ {\color{green}2}\\ {\color{blue}5} \end{bmatrix}\cdot \begin{bmatrix} {\color{red}4}\\ {\color{green}{1}}\\ {\color{blue}{6}} \end{bmatrix}= {\color{red} {3\cdot 4}}+{\color{green} {2\cdot 1}}+{\color{blue} {5\cdot 6}} = \color{red}{12} + \color{green}2 + \color{blue} {30}=44 \end{equation}
With the colors, we are marking which components are multiplied with one another.
You may have noticed that the symbol for the dot product is a multiplication dot. This is where it gets its name *dot product* from.

The dot product is defined as closely related to the length of the vectors and the angle they enclose. Another formula for calculating the dot product in words is:

\begin{equation} \vec{u} \cdot \vec{v} = \text{length}(\vec{u})\quad \text{times} \quad \text{length}(\vec{v}) \quad \text{times}\quad \cos(\text{angle between $\vec{u}$ and $\vec{v}$}) \end{equation}
With mathematical symbols, you can write this as follows: \begin{equation} \vec{u} \cdot \vec{v} = ||\vec{u}|| \cdot ||\vec{v}|| \cdot \cos(\sphericalangle \vec{u} \vec{v}) \end{equation}

Where $\cos \sphericalangle_{\vec{u}\vec{v}}$ is a measure of the angle. Here we can see a connection to the norm we discussed previously: If we form the dot product of a vector with itself, we get the norm of the vector squared.
If we rearrange the formula, we can use the dot product and the norm to calculate the angle measure. Use the following steps:

Calculate the dot product between `vec` and `vec_2`. Use the `np.dot()` function. Then divide the dot product by the product of the lengths of `vec` and `vec_2`. As a formula:

\begin{equation} \text{angle_vec_vec2} = \frac{\text{np.dot(vec, vec_2)}}{\text{norm(vec)} \cdot {norm(vec_2)}} \end{equation}
Store the result in `angle_vec_vec2` and print it.

In [36]:
```
angle_vec_vec2 = np.dot(vec, vec_2) / (norm(vec) * norm(vec_2))
angle_vec_vec2
```

Out[36]:
```
0.7999999999999998
```

Perform the same calculation for `vec` with itself. Store the result in `angle_vec_vec`. Compare the result with `angle_vec_vec_2`

In [37]:
```
angle_vec_vec = np.dot(vec,vec) / (norm(vec)*norm(vec))
angle_vec_vec
```

Out[37]:
```
0.9999999999999998
```

For `angle_vec_vec2` we get about 0.8 and for `angle_vec_vec` about 1.0.

Strictly speaking, the angle measure is the cosine function of the angle. The cosine is a function that converts angles into numbers between -1 and +1.

To get the angle now, you have to apply the function `np.arccos()` to `angle_vec_vec2` and `angle_vec_vec`. The arccosine is the inverse function of the cosine. We use it to calculate an angle from a number between -1 and 1. However, we get an angle that is not given in degrees as usual, but in radians, which are used more often in mathematics. To make the angle itself easier to read, we can convert it to an angle in degrees with the function `np.degrees()`. What angle do we get in each case? Print them both.

In [38]:
```
print(np.degrees(np.arccos(angle_vec_vec)))
print(np.degrees(np.arccos(angle_vec_vec2)))
```

```
1.2074182697257333e-06
36.86989764584404
```

The angle between `vec` and `vec_2` is approximately 36.9°. The following picture illustrates this once again:

angle between 2 vectors

The angle between `vec` and `vec` is very very small (it is 0, to be precise). This is because we only use one vector, `vec`. A vector does not form an angle with itself. So if the vectors are identical, we get an angle of 0° and a cosine of 1. So if the vectors are orthogonal, we get an angle of 90° and a cosine of 0. Then the vectors point in completely different directions. Thus the cosine is already suitable as a measure of similarity measure for two vectors. Now let's apply this method to our wine data set.

The dot product is sometimes called the scalar product because its result is a scalar. A scalar is a normal number. The word scalar refers to the fact that vectors can be scaled by multiplying them. So vectors can be stretched and compressed. Try it out. Is `vec*2` twice as long as `vec`?

```
In [39]: print(norm(vec)*2)
         print(norm(vec*2))
```

```
4.47213595499958
4.47213595499958
```

If you double each entry of `vec`, the length of the vector is also doubled. So the arrow points in the same direction, but is twice as long.

Congratulations: You have now learned the basic properties of vectors. You have calculated the length of vectors and the angle between them. You now know how to calculate the cosine similarity. This is often used to measure the similarity of data points. We have only represented the principles you have learned so far with vectors with two numbers. But they also apply to vectors with any number of entries. It's easiest for us to imagine it with two entries, as this is the simplest to visualize. Next, we will use our new knowledge to recommend wines to people.

Remember:

- Use the `norm()` function from `numpy.linalg` to calculate the length of vectors.
- Calculate the dot product of two vectors with `np.dot()`
- Calculate the cosine of the angle between two vectors using the ratio of the dot product and the product of the norm of the two vectors