The Magic of Eigenvectors

In the previous module we learned about PCA, and how it can be applied to get patterns in the dataset and these patterns can capture more than 1 feature in the dataset.

In the user rating and holiday destination example we saw ratings for different holiday destinations can go up and down, i.e. they co-vary together or they have a high covariance. governed by some **latent factors**, for example, adventure.

Latent factors here mean hidden patterns that are inherent in the dataset but there is no feature for it and we can find this pattern using some technique, here we used PCA.

So, the goal of the PCA is to find the direction where the features co-vary the most together. This can be easily found using eigenvectors.

To find this,

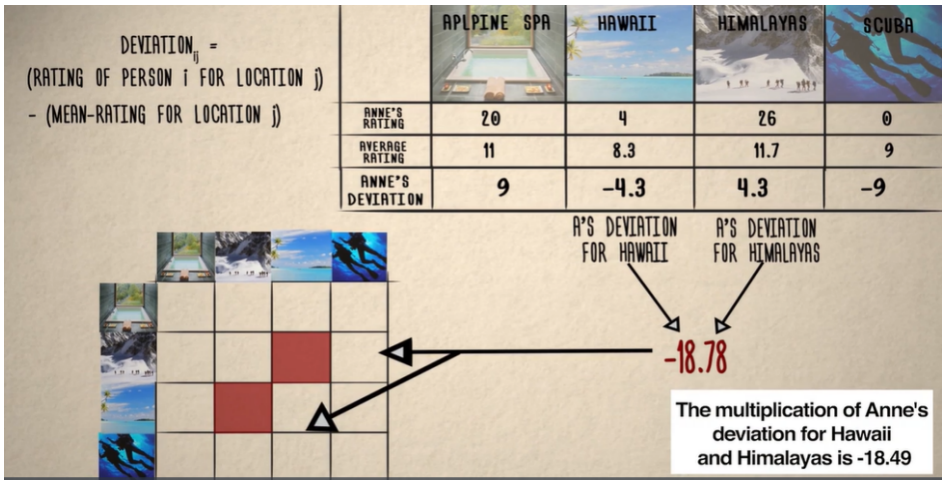
1. We first take the dataset and build the **covariance matrix** for it.

To compute the covariance matrix we first compute the mean rating for each destination, as shown in the image below.

	APLPINE SPA	HIMALAYAS	HAWAII	SCUBA
ANNE	20	4	16	0
BILL	10	2	18	10
CHRIS	13	1	19	7
JEN	1	13	7	19
JOE	18	10	10	2
MAGGIE	4	20	0	16
AVERAGE RATING	11	8.3	11.7	9

2. The next step is to find the deviation for each person rating from the mean, and the covariance between the 2 destinations, say Himalayas and Hawaii, is the

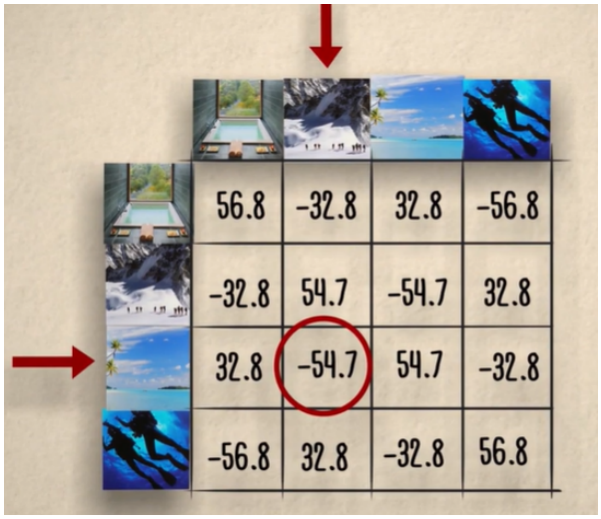
product of the deviation for both the destinations. See the image attached below.



To find the covariance between 2 destinations, we do the above settings for each person, that is find how much the person's rating deviates from the mean rating and multiply them. For example, the multiplication of Anne's deviation for the Himalayas and Hawaii is -18.49.

The covariance between, say, Himalayas and Hawaii is the sum of all the multiplications.

The covariance matrix is shown below:



Note: If we have an n feature vector, this is an n*n matrix, stating the covariance between any 2 combinations of destinations. For eg: the Himalayas are negatively correlated with Hawaii. In practicality, you don't have to do any of these calculations, **Python** did this for us. Moreover, this is similar to what we have done in the **FDS course, where we build a correlation matrix for different variables in python.**

Now, the principal components we are looking for are the eigenvectors for this covariance matrix.

What are eigenvectors?

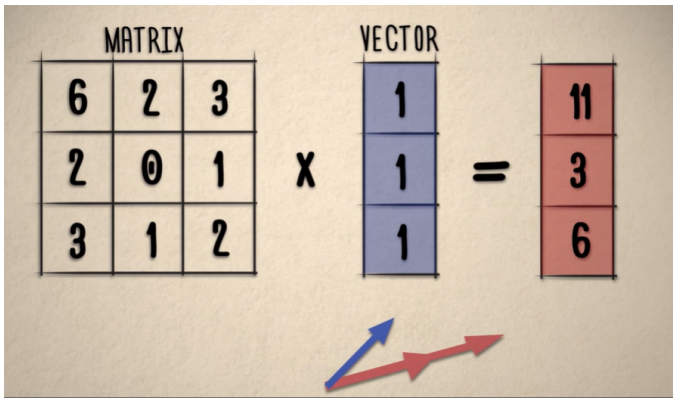
We can defined it in terms of matrix and vectors, when we multiply a DxD matrix, here 3x3, with a Dx1 vector, here 3x1, and the result is another vector,

In the below image:

11 = 6*1 + 2*1 + 3*1

3 = 2*1 + 0*1 + 1*1

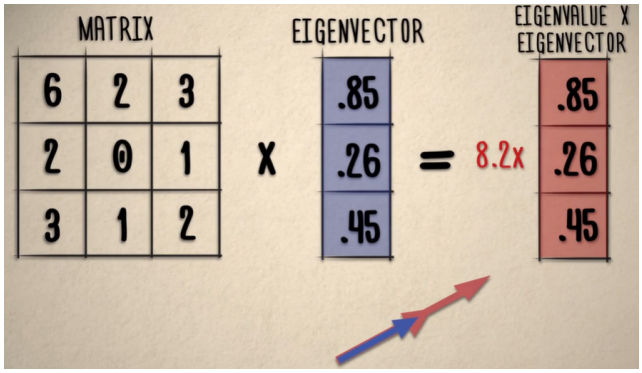
6 = 3*1 + 1*1 + 2*1



Now, when to multiply the matrix with a vector, you get another vector, the direction and the magnitude of the resultant vector (11, 3, 6) are different from the original vector (1,1,1),

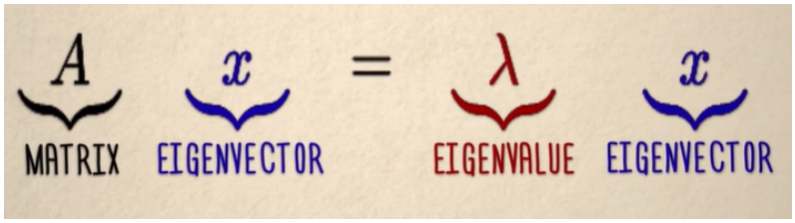
In the above example, *the* Blue vector is the original one, while the red vector is the resultant vector. The direction and magnitude of the resultant vector are different from the original.

But, i**f the original vector is an eigenvector, we can only see the magnitude change, and the direction does not change.** See the image below.



Here, the **vector (0.85, 0.26, 0.45) is an eigenvector,** and **8.2** is the corresponding **eigenvalue**, as the resulting vector is the same as the original vector, only the magnitude of the resultant vector is 8.2x.

In some sense, eigenvectors capture the **major direction that is inherent in the matrix,** and the larger the eigenvalue, the more important the vector.
The above can be written as the equation as shown below.



We don't have to worry about this equation, the computer will solve this for us in split seconds.

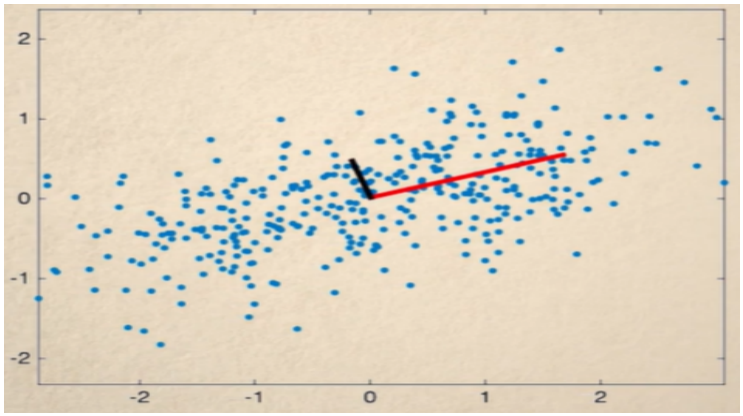
Another point to note here is that, if we have a DxD matrix, then we will have D eigenvectors and eigenvalues. The highest eigenvalue will have a corresponding eigenvector that captures the major direction in the matrix.

Now, we learned at the start of the video, that the goal of the PCA is to find the direction where the features co-vary the most together and this can be done using eigenvectors.

It can be easily understood that as in PCA we are finding the eigenvectors for the covariance matrix, and we will get the directions that capture features that describe the major covariance in the dataset.

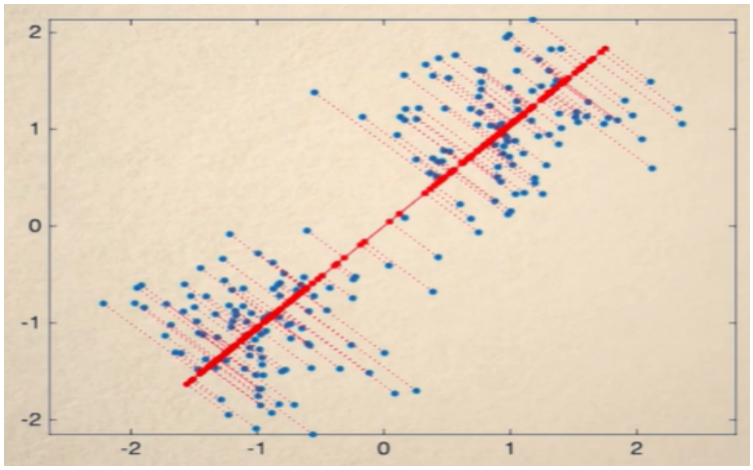
Let's understand this with an example,
The image attached below has data points in 2 dimensions and each data point has 2 features. In this data, features have a strong correlation, i.e. either they are low or both are high, and then we have some random noise. Now, as we have 2 dimensions of data, there will be 2 eigenvectors and corresponding 2 eigenvalues. **The eigenvector with the highest eigenvalue is the first principal component,** as shown in the red line. This Principal component captures the major variance in the dataset.

On the other hand, the second eigenvector is perpendicular to the first principal component and only captures a small variance of the data.



Now suppose we even encode the dataset only on **1 principal component** as shown below, by projecting the datapoint on the first principal component, and if there are some inherent clusters emerging in the dataset they still remain intact, and the dimensions are reduced (as now we have data point on 1 axis, i.e. on first principal component).

This can be easily done in high dimensional settings too.



The next question is, when we have high dimensional data, how many principal components are enough to capture most of the information in the dataset?

Theoretically, it can be done by looking at the eigenvalues, after a certain number of principal components, there is a gap in the eigenvalues.

Practically, it is very easy to do in python, using the PCA module in python.