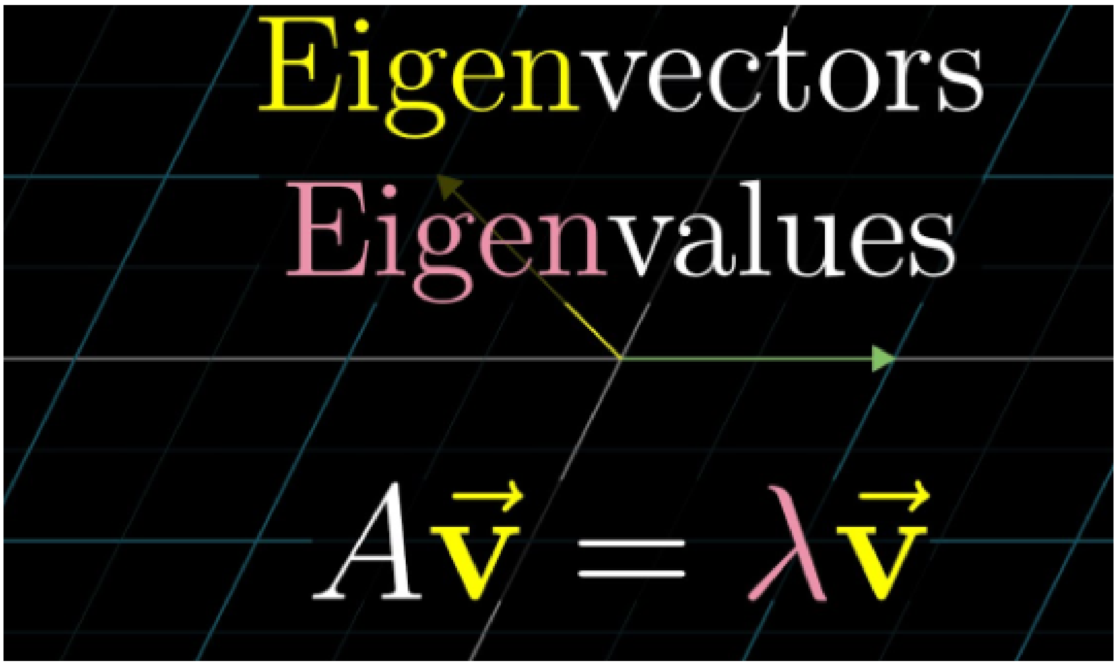
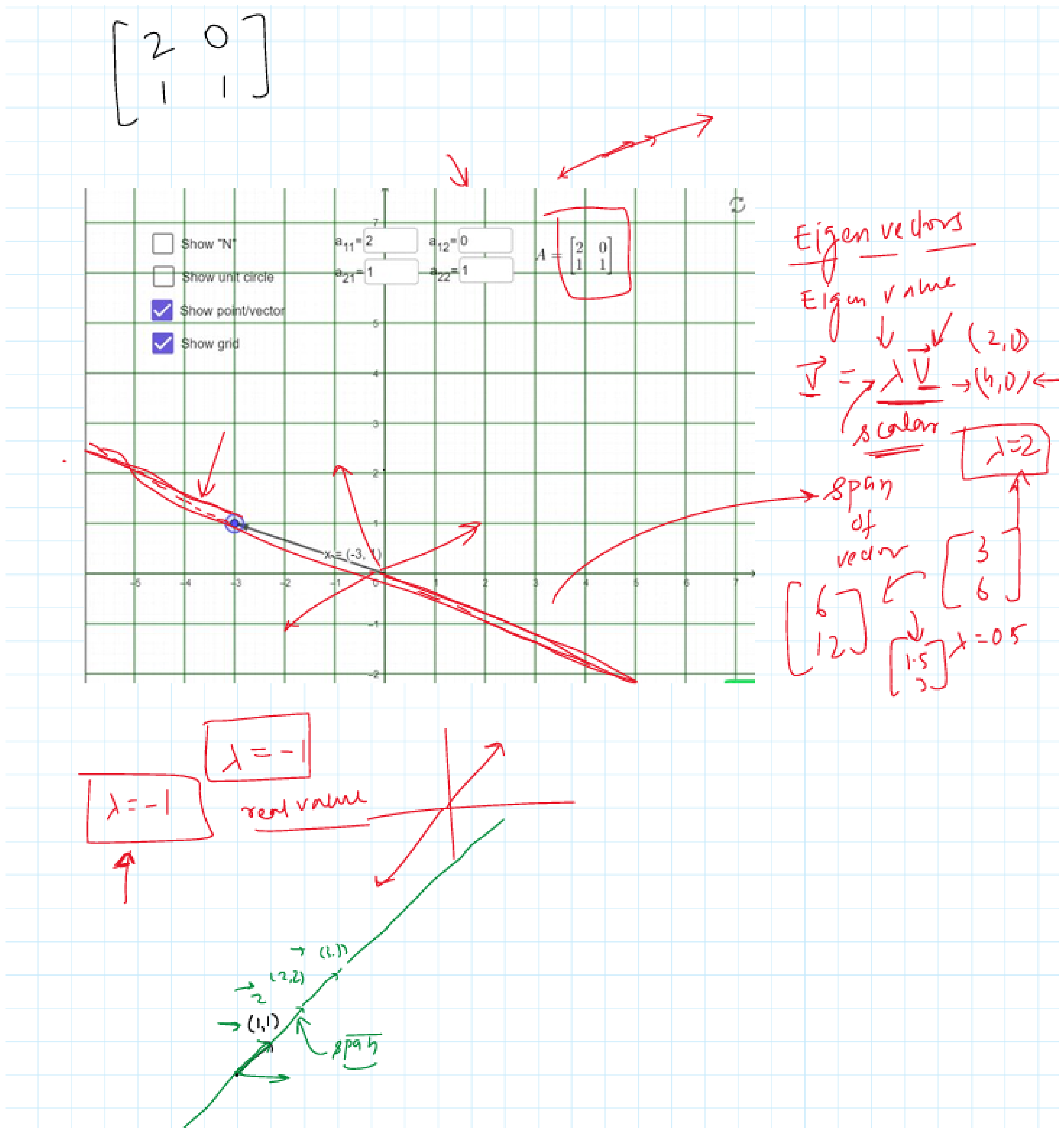
What is Eigenvectors & Eigenvalues?

Eigenvectors are special vectors that don't change their direction when multiplied by a matrix. Instead, they only get stretched or shrunk by a scalar value called the eigenvalue.



Imagine you have a matrix A and a vector v. If Av is a scaled version of v, then v is an eigenvector of A, and the scaling factor is the eigenvalue.

- Eigenvalues represent how much an eigenvector gets stretched or shrunk when multiplied by a matrix. Each eigenvalue corresponds to one or more eigenvectors.
- Eigenvectors associated with different eigenvalues point in different directions and are independent of each other.
- Eigenvectors and eigenvalues have various practical applications, such as reducing data dimensionality, analyzing systems with random transitions, and image processing techniques like face recognition and compression.
- In essence, eigenvectors and eigenvalues provide us with important information about the behavior and properties of matrices and transformations.



What is span ?

The span of a set of vectors is the collection of all possible vectors that can be formed by stretching or shrinking those vectors and adding them together. It represents all the directions and magnitudes that can be achieved by combining the given vectors.

For example, if we have two vectors in two-dimensional space, the span would be the entire plane. Any point in the plane can be reached by scaling and adding the given vectors appropriately.

The span helps us understand the extent to which a set of vectors covers a space. If the span includes all possible vectors in a space, then the set of vectors spans the entire space. If the span is smaller, it means that the set of vectors does not cover the entire space.

The concept of span is important in various areas of mathematics, such as solving equations,

When the transformation is a rotation?

Given that as we said when the transformation is applied the eigenvector is not moved away from its own span, this vector is effectively the axis of rotation of the transformation itself. So, If you are in a 3D space, it's probably easier to think of a rotation as the combination of axes of rotation and the angle by which it rotates, rather than in terms of the matrix used to apply the transformation.

https://www.youtube.com/watch?v=e50Bj7jn9IQ (https://www.youtube.com/watch?v=e50Bj7jn9IQ)

When we need to compute the power of a matrix?

Computing the power of a non-diagonal matrix is a fairly painful process. However, if that matrix has enough eigenvectors to span the whole space you can change the coordinate system of the transformation so that the eigenvectors are the basis. In this new coordinate system, the transformation is expressed through a diagonal matrix having the eigenvalues on its diagonal. It's now much easier to compute whatever power of the diagonal matrix and then move the final result back to the original coordinate system once done.

https://www.youtube.com/watch?v=PFDu9oVAE-g&t=782s (https://www.youtube.com/watch?v=PFDu9oVAE-g&t=782s)

How to calculate Eigen Vectors and Eigen Values?

To calculate eigenvalues and eigenvectors, you need to follow these steps:

1. Start with a square matrix A for which you want to find the eigenvalues and eigenvectors.
2. Formulate the characteristic equation by subtracting the identity matrix I from A and calculating its determinant. The characteristic equation is det(A - λI) = 0, where λ is the eigenvalue.
3. Solve the characteristic equation to find the eigenvalues. The solutions to the equation are the eigenvalues of the matrix A.
4. For each eigenvalue, substitute it back into the equation (A - λI)v = 0 and solve for the corresponding eigenvector v. The resulting vector v is the eigenvector associated with that eigenvalue.

Let's illustrate this process with an example using a 2x2 matrix:

1. Start with a matrix A: A = | a b |

```
| c   d |
```

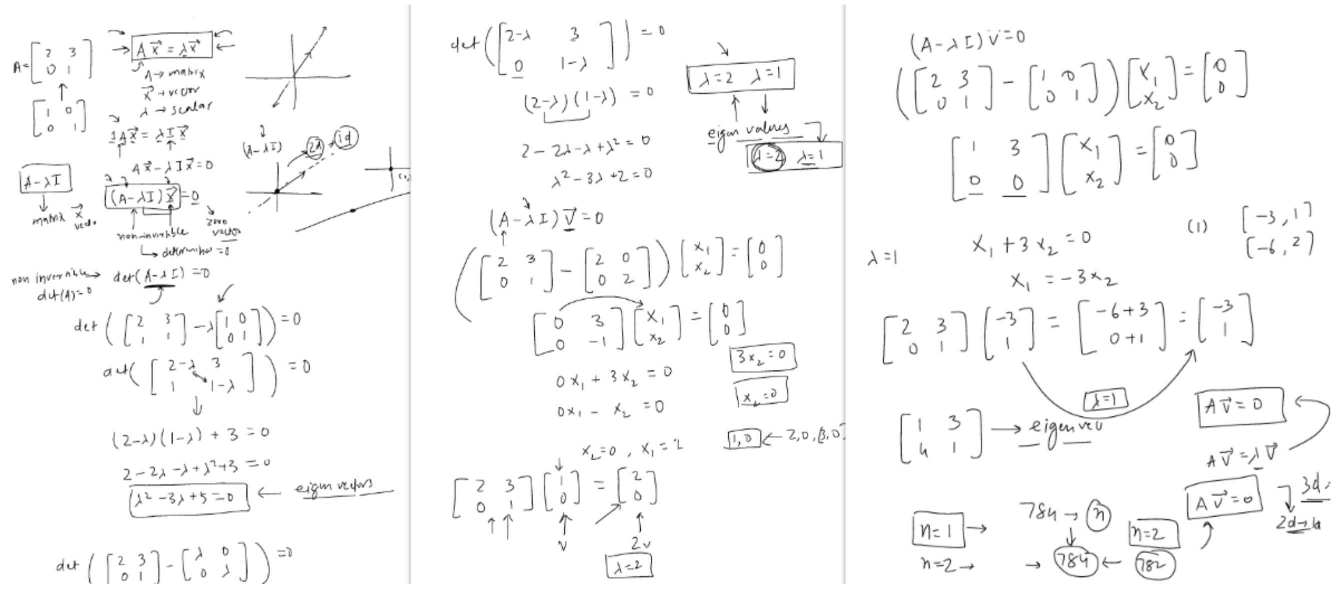
2. Subtract the identity matrix I: A - λI = | a-λ b |

```
| c     d-λ  |
```

3. Calculate the determinant of (A - λI) and set it equal to zero: det(A - λI) = (a-λ)(d-λ) - bc = 0

4. Solve the characteristic equation for λ to find the eigenvalues.

5. For each eigenvalue, substitute it back into the equation (A - λI)v = 0 and solve for the corresponding eigenvector v. This involves solving a system of linear equations.



$A = \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}$ $\boxed{A\vec{x} = \lambda\vec{x}}$

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

A → matrix

$\vec{x}$ → vector

λ → scalar

$A\vec{x} = \lambda I\vec{x}$

$A\vec{x} - \lambda I\vec{x} = 0$

$\boxed{A - \lambda I}$ matrix $\vec{x}$ vector

$\boxed{(A - \lambda I)\vec{x}} = \underline{0}$ zero vector

non-invertible → determinant = 0

non invertible → $\det(A) = 0$ $\det(A - \lambda I) = 0$

$(A - \lambda I)$ 2λ $1d$

$\det\left(\begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$

$\det\left(\begin{bmatrix} 2-\lambda & 3 \\ 1 & 1-\lambda \end{bmatrix}\right) = 0$

$(2-\lambda)(1-\lambda) + 3 = 0$

$2 - 2\lambda - \lambda + \lambda^2 + 3 = 0$

$\boxed{\lambda^2 - 3\lambda + 5 = 0}$ ← eigen vectors

$\det\left(\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = 0$

$\det\left(\begin{bmatrix} 2-\lambda & 3 \\ 0 & 1-\lambda \end{bmatrix}\right) = 0$

$(2-\lambda)(1-\lambda) = 0$

$2 - 2\lambda - \lambda + \lambda^2 = 0$

$\lambda^2 - 3\lambda + 2 = 0$

$\boxed{\lambda = 2 \quad \lambda = 1}$ eigen values $\boxed{\lambda = 2 \quad \lambda = 1}$

$(A - \lambda I)\vec{V} = 0$

$\left(\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\right)\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 3 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$0x_1 + 3x_2 = 0$ $\boxed{3x_2 = 0}$

$0x_1 - x_2 = 0$ $\boxed{x_2 = 0}$

$x_2 = 0, \; x_1 = 1$ $\boxed{1, 0}$ ← 2,0, [3,0]

$\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ v 2v $\boxed{\lambda = 2}$

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

$\left(\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\lambda = 1$ $x_1 + 3x_2 = 0$ (1) $[-3, 1]$ $[-6, 2]$

$x_1 = -3x_2$

$\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} -3 \\ 1 \end{bmatrix} = \begin{bmatrix} -6+3 \\ 0+1 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$ $\boxed{\lambda = 1}$

$\begin{bmatrix} 1 & 3 \\ 4 & 1 \end{bmatrix}$ → eigenvec

$\boxed{A\vec{v} = 0}$ $A\vec{v} = \lambda\vec{v}$ $\boxed{A\vec{v} = 0}$ 3d 2d → 1d

$\boxed{n = 1}$ → 784 → n $\boxed{n = 2}$

n=2 → 784 ← 782

Properties

- **Sum of Eigenvalues**: The sum of all the eigenvalues of a matrix is equal to its trace (the sum of the diagonal elements of the matrix). This holds true regardless of whether the matrix is square or not.

- **Product of Eigenvalues**: The product of all the eigenvalues of a matrix is equal to its determinant. This also holds for square matrices.

- **Eigenvectors corresponding to different eigenvalues are orthogonal**: If a matrix A is symmetric (i.e., A = A^T), the eigenvectors corresponding to distinct eigenvalues are orthogonal to each other

- **Eigenvalue of a Identity Matrix** : For an identity matrix, the eigenvalues are all 1, regardless of the dimension of the matrix.

- **Eigenvalue of a Scalar Multiple** : If B is a matrix obtained by multiplying a scalar c to a matrix A (i.e., B = cA), then the eigenvalues of B are just the eigenvalues of A each multiplied by c

- **Eigenvalues of a Diagonal Matrix**: For a diagonal matrix, the eigenvalues are the diagonal elements themselves.

- **Eigenvalues of a Transposed Matrix**: The eigenvalues of a matrix and its transpose are the same

In [1]:

```python
# code

import numpy as np

# Define the matrix
matrix = np.array([[1, 2],
                   [3, 4]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(matrix)

# Print the eigenvalues
print("Eigenvalues:")
for eigenvalue in eigenvalues:
    print(eigenvalue)

# Print the eigenvectors
print("Eigenvectors:")
for i in range(len(eigenvectors)):
    print("Eigenvector", i+1, ":", eigenvectors[:, i])
```

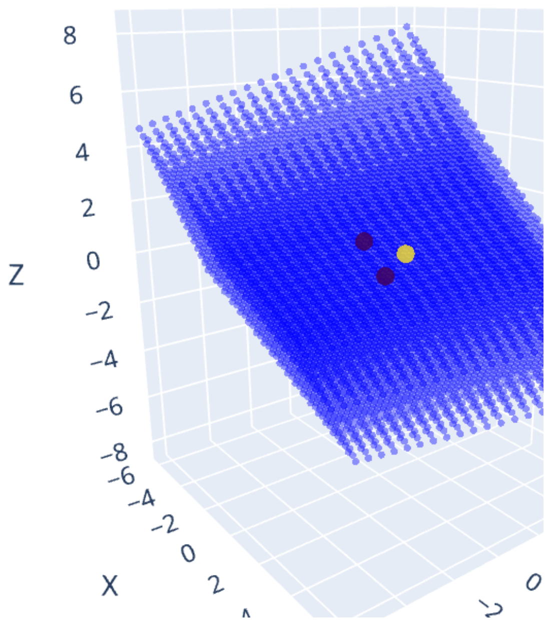
```
Eigenvalues:
-0.3722813232690143
5.372281323269014
Eigenvectors:
Eigenvector 1 : [-0.82456484  0.56576746]
Eigenvector 2 : [-0.41597356 -0.90937671]
```

In [ ]:

```
```

In [5]:

```
import numpy as np
import plotly.graph_objects as go

# Define the matrix
matrix = np.array([[2, -1, 0],
                   [0, 3, 1],
                   [-1, 0, 2]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(matrix)

# Extract real and imaginary components of eigenvectors
x = np.real(eigenvectors[:, 0])
y = np.real(eigenvectors[:, 1])
z = np.real(eigenvectors[:, 2])

# Create a meshgrid of coefficients
coef_range = np.linspace(-5, 5, 20)
coef_x, coef_y, coef_z = np.meshgrid(coef_range, coef_range, coef_range, indexing='ij

# Compute the linear combinations
linear_combinations = (
    coef_x * x[0] + coef_y * x[1] + coef_z * x[2],
    coef_x * y[0] + coef_y * y[1] + coef_z * y[2],
    coef_x * z[0] + coef_y * z[1] + coef_z * z[2]
)

# Get the span vectors
span_x, span_y, span_z = linear_combinations

# Create a 3D scatter plot for the span vectors
fig = go.Figure(data=go.Scatter3d(
    x=span_x.ravel(), y=span_y.ravel(), z=span_z.ravel(),
    mode='markers',
    marker=dict(
        size=2,
        color='blue',
        opacity=0.4
    )
))

# Add the eigenvectors to the plot
fig.add_trace(go.Scatter3d(
    x=x, y=y, z=z,
    mode='markers',
    marker=dict(
        size=5,
        color=eigenvalues.real,
        colorscale='Viridis',
        opacity=0.8
    )
))

# Set axis titles and layout
fig.update_layout(
    scene=dict(
        xaxis_title='X',
        yaxis_title='Y',
        zaxis_title='Z'
    )
)

# Show the 3D plot
fig.show()
```

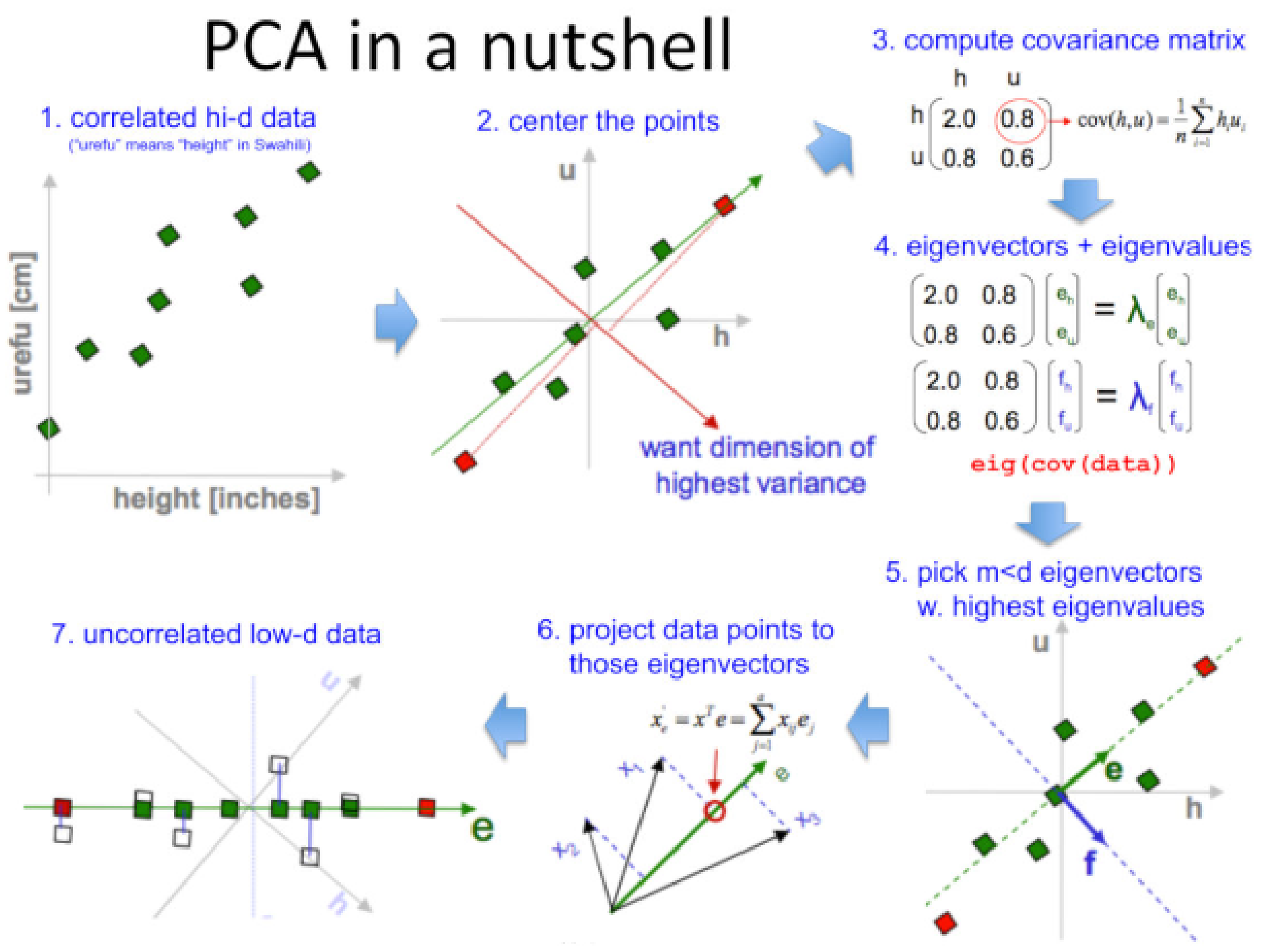


Explantion

- First, the necessary libraries are imported. numpy is imported as np for numerical operations, and plotly.graph_objects is imported as go for creating the 3D plot.
- Next, the matrix for which eigenvectors and eigenvalues are to be calculated is defined. The matrix is a 3x3 array.
- The eigenvalues and eigenvectors of the matrix are calculated using the np.linalg.eig() function from the NumPy library. This function returns two arrays: eigenvalues and eigenvectors.
- The real components of the eigenvectors are extracted using the np.real() function from NumPy. The real components are stored in the variables x, y, and z respectively.

- A meshgrid of coefficients is created using np.meshgrid(). The coef_range array is generated using np.linspace() to create a range of values from -5 to 5 with 20 points. The meshgrid is created using coef_range for each axis (coef_x, coef_y, coef_z), with the indexing set to 'ij'.
- The linear combinations of the eigenvectors are computed by multiplying the coefficients with the corresponding components of the eigenvectors. This is done for each axis (x, y, z) and stored in the linear_combinations tuple.
- The span vectors (span_x, span_y, span_z) are extracted from the linear_combinations tuple.
- A 3D scatter plot is created using go.Figure() and go.Scatter3d(). The span vectors are used to create the scatter plot for the span vectors, with blue markers of size 2 and opacity 0.4.
- The eigenvectors are added to the plot as a separate scatter plot using go.Scatter3d(). The eigenvectors are represented by markers with size 5 and their color is determined by the real values of the eigenvalues. The colorscale is set to 'Viridis' and opacity is set to 0.8.
- The axis titles and layout of the plot are set using fig.update_layout().
- Finally, the 3D plot is displayed using fig.show().

why using Eigenvectors and Eigenvalues in PCA



PCA in a nutshell
1. correlated hi-d data
("urefu" means "height" in Swahili)
urefu [cm]
height [inches]
2. center the points
u
h
want dimension of highest variance
3. compute covariance matrix
h u
h 2.0 0.8
u 0.8 0.6
$cov(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$
4. eigenvectors + eigenvalues
$\begin{pmatrix}2.0 & 0.8\\0.8 & 0.6\end{pmatrix}\begin{pmatrix}e_h\\e_u\end{pmatrix} = \lambda_e\begin{pmatrix}e_h\\e_u\end{pmatrix}$
$\begin{pmatrix}2.0 & 0.8\\0.8 & 0.6\end{pmatrix}\begin{pmatrix}f_h\\f_u\end{pmatrix} = \lambda_f\begin{pmatrix}f_h\\f_u\end{pmatrix}$
eig(cov(data))
5. pick m<d eigenvectors w. highest eigenvalues
u
h
e
f
6. project data points to those eigenvectors
$x_e' = x^T e = \sum_{j=1}^{d} x_j e_j$
7. uncorrelated low-d data
e

Eigenvectors and eigenvalues are used in PCA because they can be used to **find the directions of maximum variance in the data**. The eigenvectors of a covariance matrix correspond to the directions of maximum variance in the data, and the eigenvalues of the covariance matrix correspond to the amount of variance in each direction.

PCA works by projecting the data onto the eigenvectors of the covariance matrix. This projection results in a new set of features that are uncorrelated and ordered by their importance. The most important features are the ones that correspond to the eigenvectors with the largest eigenvalues.

Using eigenvalues and eigenvectors in PCA has a number of advantages. First, it allows us to find the directions of maximum variance in the data, which is important for understanding the structure of the data. Second, it allows us to reduce the dimensionality of the data without losing too much information. Third, it is a relatively simple and efficient algorithm.

Here are some of the specific reasons why eigenvalues and eigenvectors are used in PCA:

- Eigenvectors point in the direction of maximum variance in the data. This means that they can be used to find the directions in which the data is most spread out.
- Eigenvalues measure the amount of variance in each direction. This means that they can be used to determine how important each direction is.
- The eigenvectors of a covariance matrix are orthogonal to each other. This means that they are independent of each other, which is important for reducing the dimensionality of the data.

Eigenvectors and eigenvalues are used in PCA (Principal Component Analysis) to determine the principal components of the data and perform dimensionality reduction. Here's how they are utilized in PCA:

1. **Covariance Matrix**: PCA starts by computing the covariance matrix of the data. The covariance matrix provides information about the relationships and variances among the different features or dimensions of the data.
2. **Eigenvectors**: The eigenvectors represent the directions or axes along which the data has the highest variance. They define the principal components of the data. Each eigenvector corresponds to a principal component.
3. **Eigenvalues**: The eigenvalues associated with the eigenvectors represent the amount of variance explained by each principal component. Higher eigenvalues indicate more significant variance along the corresponding eigenvectors.
4. **Variance Explained**: The eigenvalues can be used to calculate the proportion of variance explained by each principal component. The ratio of an eigenvalue to the sum of all eigenvalues provides a measure of how much information or variance is captured by a particular principal component.
5. **Dimensionality Reduction**: Based on the eigenvalues and eigenvectors, PCA allows for dimensionality reduction. The eigenvectors are sorted in descending order based on their corresponding eigenvalues. The eigenvectors with the highest eigenvalues, i.e., the principal components with the most significant variance, are selected. These principal components form a new basis for the data.
6. **Projection**: The original data is projected onto the selected principal components or eigenvectors to obtain the lower-dimensional representation of the data. This projection reduces the dimensionality while preserving the maximum amount of information or variance possible.

In detailed:

https://datahacker.rs/eigenvectors-and-eigenvalues/ (https://datahacker.rs/eigenvectors-and-eigenvalues/)

In [ ]: