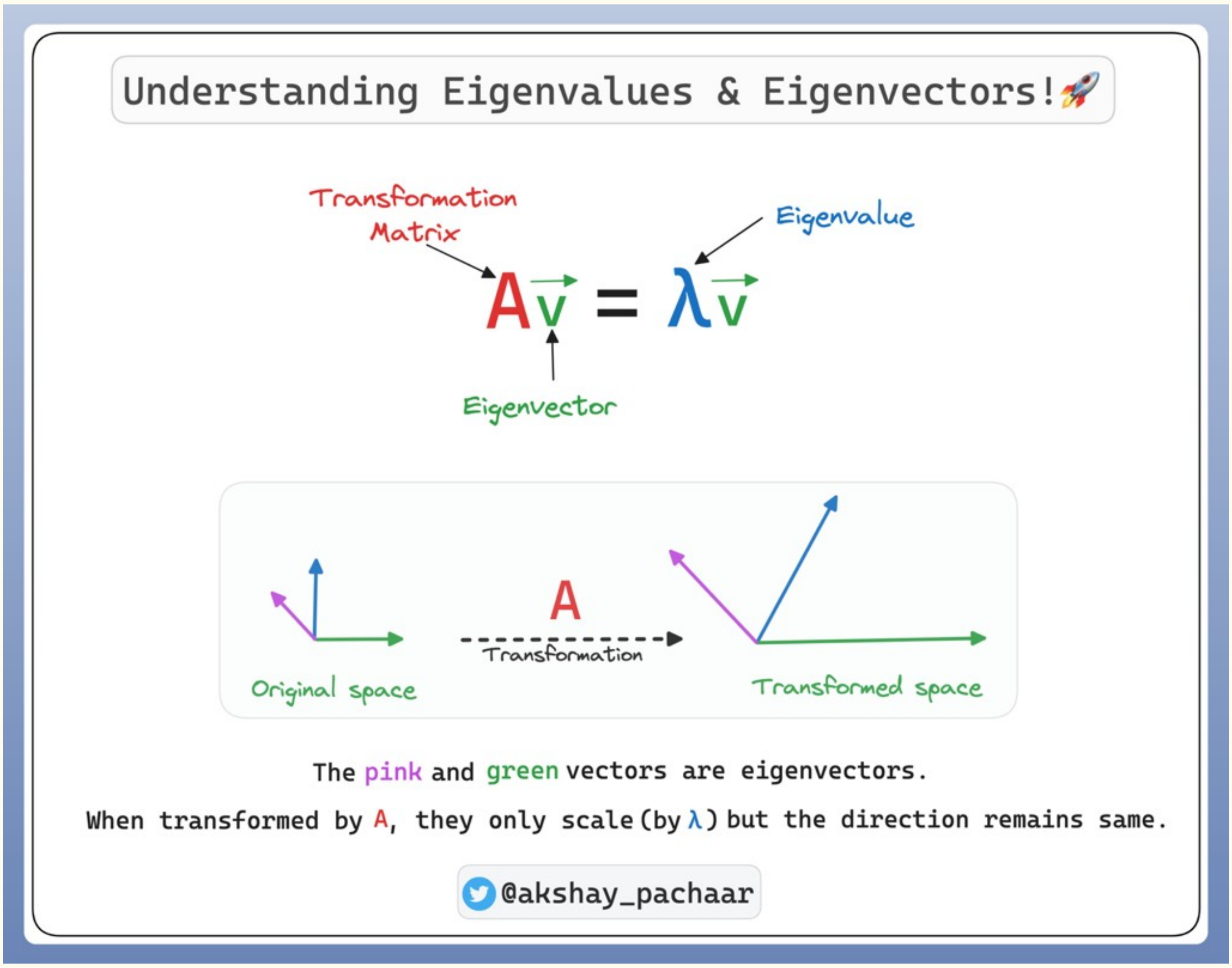
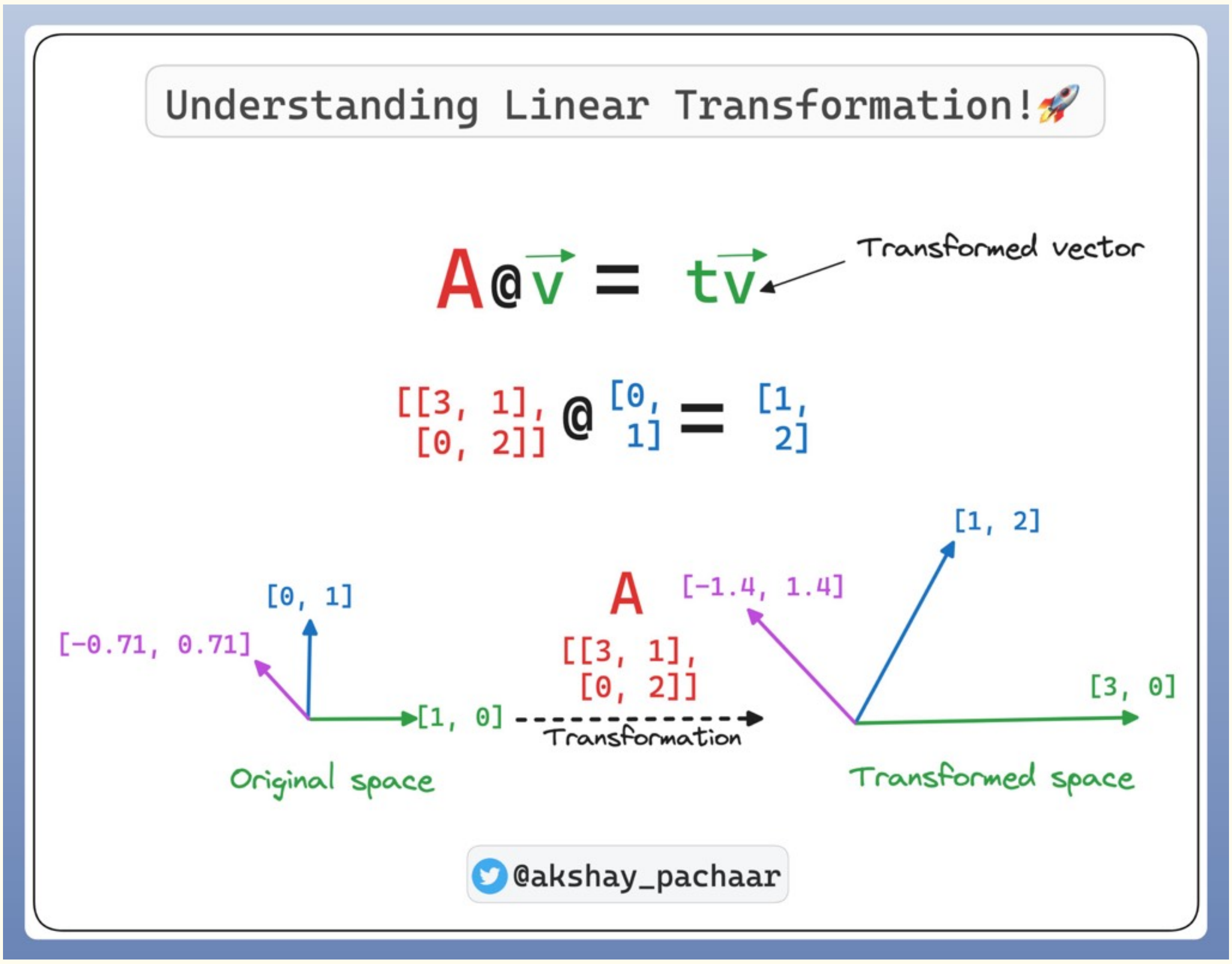

Understanding Eigenvalues & Eigenvectors!🚀
Transformation Matrix
Eigenvalue
$A\vec{v} = \lambda\vec{v}$
Eigenvector
A
Transformation
Original space
Transformed space
The pink and green vectors are eigenvectors.
When transformed by A, they only scale (by λ) but the direction remains same.
@akshay_pachaar


Understanding Linear Transformation!🚀
$A@\vec{v} = t\vec{v}$
Transformed vector
[[3, 1], [0, 2]] @ [0, 1] = [1, 2]
[0, 1]
[-0.71, 0.71]
[1, 0]
A
[[3, 1], [0, 2]]
Transformation
[-1.4, 1.4]
[1, 2]
[3, 0]
Original space
Transformed space
@akshay_pachaar

Calculating Eigenvalues & Eigenvectors!

$$A\vec{v} = \lambda\vec{v}$$

Since an Eigenvector only gets scaled by λ, we can write it's transformation like this

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

Post multiplying by Identity matrix "I" on both side & rearranging we obtain this!

To solve it for a non-zero "v" the following determinant must be zero!

$$\det(A - \lambda I) = 0$$

Solving this would give us the eigenvalues & then we can calculate the eigenvectors!

We will take an example in next tweet!

@akshay_pachaar

Let's take an example now!🚀

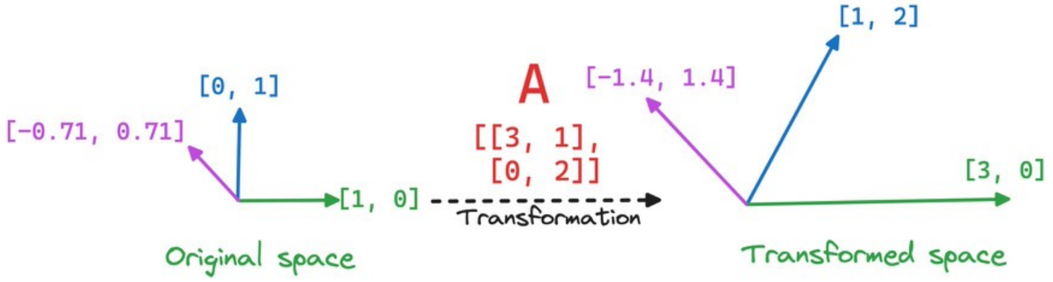
A = [[3, 1],
     [0, 2]]

$$\det(A - \lambda I) = 0 \longrightarrow \det\left(\begin{matrix}[[3-\lambda, & 1], \\ [0, & 2-\lambda]]\end{matrix}\right) = 0$$

Solving this we obtain:

Eigenvalues: 3 & 2

Eigenvectors: [1, 0] & [-0.71, 0.71]



Observe how the direction remains same for the two Eigenvectors & they get scaled by their corresponding Eigenvalues

@akshay_pachaar

PCA from scratch!! 🚀

```python
import numpy as np


class PCA:
    def __init__(self, n_components):
        self.n_components = n_components
        self.components = None
        self.mean = None

    def fit(self, X):
        # center the data
        self.mean = np.mean(X, axis=0)
        X = X - self.mean

        # compute the covariance matrix
        cov = np.cov(X, rowvar=False)

        # compute the eigenvalues and eigenvectors of the covariance matrix
        eigenvalues, eigenvectors = np.linalg.eigh(cov)

        # sort the eigenvalues and eigenvectors in decreasing order
        idx = np.argsort(eigenvalues)[::-1]
        eigenvalues = eigenvalues[idx]
        eigenvectors = eigenvectors[:, idx]

        # store the first n_components eigenvectors as the principal components
        self.components = eigenvectors[:, : self.n_components]

    def transform(self, X):
        # center the data
        X = X - self.mean

        # project the data onto the principal components
        X_transformed = np.dot(X, self.components)

        return X_transformed
```

follow:

@akshay_pachaar