# Linear Algebra Basics

- array from nested sequences, all must with type dimension data
- array(shape, dtype=None, buffer=None) and zeros and ones are also all equivalent as long as the higher dimensions like 3D dimension or 3D8D
- ndim, shape, size, dtype
- ndarray.astype(dtype) this allows, when one the represented using those dtypes, once one's data is represented there is no longer any data loss and represented by machine learning algorithms

# Tensor

- Tensor is a data structure, a container to save data locally numbers
- 0D Tensor is a scalar, 1D Tensor is a vector, 2D Tensor is a matrix, 3D Tensor is a cube, 4D Tensor is a hypercube

```
import numpy as np
x = np.array(1)
x
```

```
array(1)
```

```
x.ndim
```

```
0
```

# Vector

- 1d tensor/ vector (1,1,1,1)
- 2d tensor/ matrix (1,1,1,1)
- 3d tensor/ cube (1,1,1,1)
- 4d tensor/ 4d cube/ hypercube (1,1,1,1)
- 5d tensor/ 5d cube/ hypercube (1,1,1,1,1)

```
x = np.array([1,2,3,4])
x
```

```
array([1, 2, 3, 4, 5])
```

```
x.ndim
```

```
1
```

# Matrices

- 2d tensor/ matrix (1,1,1,1) array([[1,2,3],[4,5,6]])

```
x = np.array([[1,2,3],[4,5,6]])
x
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

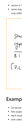
```
x.ndim
```

```
2
```

# nd Tensors

- with n tensors/hierarchies to create nd tensors

# rank axis shape




- rank, axis 3D 3D tensor
- shape (3,2,3) 3 axis, axis 0 has size dim size 3 (3,2,3)
- axis 1 has dim size 2 (3,2,3)

# Example 1d tensors

- vector is 1d, 5d tensor dimension is 1, 5d data cube, 5 axis, 5 nd 5d dimension vector
- 1d tensor/ vector (1,1,1,1) 5d tensor/ cube (1,1,1,1,1) 5d tensor/ cube/ hypercube (1,1,1,1,1) 5d tensor/ cube/ hypercube (1,1,1,1,1)
- array(shape,dtype=None,buffer=None,order=None) array(shape,dtype=None,buffer=None,order=None)
- array(shape,dtype=None,buffer=None,order=None) array(shape,dtype=None,buffer=None,order=None)

# Example 2d tensors

- 2d tensor/ matrix (1,1,1,1)
- 2d tensor/ matrix (1,1,1,1) 2d tensor/ matrix (1,1,1,1)
- array(shape,dtype=None,buffer=None,order=None) array(shape,dtype=None,buffer=None,order=None)
- array(shape,dtype=None,buffer=None,order=None) array(shape,dtype=None,buffer=None,order=None)

# Example 3d tensors

- 3d tensor/ cube (1,1,1,1)
- array(shape,dtype=None,buffer=None,order=None) array(shape,dtype=None,buffer=None,order=None)

# Example of 4d tensor

- 4d tensor/ hypercube (1,1,1,1)
- image is a collection of pixel each pixel is a 3d tensor
- 4d tensor/ image/ cube (1,1,1,1) array([[[[1,2,3]

- batches of images is 4d tensor

# Example of 5d Tensor

- video
- 5d tensor/ cube (1,1,1,1)
- batches of images are frequently in high power/ more than one permeance of vision which is 5 images in a set. 1Days which gives an impression
- Image 1d 3d tensor 3d collection of images is 5d tensor

# summary

- Scalar, Scalars are single numbers, 0d, 1d, 2d, 3d and 4 data science is represent through the feed values a have with a single number
- Series (1D tensors) Series are lists of numbers that go in a row or column. They represent things that have both size and direction. Series are also used to store things like distances of a physical 3d/ cube, weight, and weights in the neural network.
- Matrices (2D tensors) Matrices are like grids of numbers with rows and columns. They're used to organize data in 3d collections with axis 0 and axis 1. Matrices number like a grid. Matrices are also the organizing data, doing calculations for linear regression or neural network.
- Tensor (nD tensors) Tensors are like more complex versions of series and matrices that can have any number of dimensions. They're used to represent more complicated data structures like images and multiple layers in neural nets. Tensors are the building blocks of machine learning, because they're used to store all data, from inputs to outputs to weights.

# Vectors

- series are 1d tensors
- vector is a tensor/ cube can 1d tensors, if they have 4 inputs then it will be series, if it 1d tensor then it will be 1d series
- vectors are a set/ cube/ hypercube of data sets are corresponding to transform alphabets into numbers.

**Row Vector & Column Vector**

# vector distance from origin

# euclidean distance

```
import numpy as np
A = np.array([1,2,3,4,5])
B = np.array([6,7,8,9,10])
euclidean_dist(A,B)
euclidean_dist(B,A)
euclidean_dist(A,A)
euclidean_dist(B,B)
```

```
5.0
5.0
0.0
0.0
```

# Scalar operations

# addition/subtraction (shifting)

# mean centring

- When centering in linear algebra means subtracting the average value of each variable from all the data points. This helps to center the data around zero, making it easier to understand and analyze it. Improves in algorithm performance, convergence & stability.
  - 1 Calculate the average value of each variable in the data.
  - 2 Subtract these average values from every data point.
- This process shifts the data cloud around to center the origin, simplifying analysis & making it easier to work with.




# dot product (returns a scalar)




- dot product can also be used as matrix multiplication




- **note**
- commutative (a1 * b1) = (b1 * a1)

```
x = np.array([1,2,3])
y = np.array([4,5,6])
x.shape
y.shape
np.dot(x,y)
```

```
(3,)
(3,)
32
```

```
x = np.array([[1,2,3],[4,5,6]])
y = np.array([[7,8],[9,10],[11,12]])
x.shape
y.shape
np.dot(x,y)
```

```
(2,3)
(3,2)
array([[ 58, 64],
       [139, 154]])
```

# usecase

- used in compute similarity in 2 vectors
- used in compute matrix multiplication
- perform matrix multiplication

# angle b/w 2 vectors

- continuation b/ w vectors from above




# cosine similarity

- Cosine similarity is a measure used to determine how similar two vectors are in a multi dimensional space. It calculates the cosine of the angle between two vectors, providing a value between -1 and 1. A value closer to 1 indicates that the vectors are




- Identical: While a value closer to 1 suggests dissimilarity, a value closer to -1 indicates opposite vectors.
- So cosine is a similarity measure not a metric for find how similar and 2 vectors in multi dimensional space




- Cosine similarity range from -1 to 1
- cos(0) = 1 (vectors point in the same direction), 0 (vectors are orthogonal), -1 (vectors point in opposite directions)
- dot product of 2 vectors vectors are opposite (θ = 180)
- dot product of 2 vectors vectors are (θ = 90)
- For cosine distance from 1 to 0 the angle is acute and for cosine between -1 and 0 the angle is obtuse
- Recommender systems often use cosine similarity to measure similarity between items or users based on their feature vectors.

```
import numpy as np
x = np.array([1,2,3])
y = np.array([4,5,6])
x.shape
y.shape
cosine_similarity(x, y) # [0.97463185]  1 is (being similar)  -1 is (being opposite)
cosine_similarity(x, x)
cosine_similarity(x, -x) # [-1.]  1 is (being similar)  -1 is (being opposite)
cosine_similarity(x, y)
```

```
(3,)
(3,)
array([0.97463185])
array([1.])
array([-1.])
array([0.97463185])
```

# Equation of hyperplane

- In higher dimensional spaces, beyond the familiar 3D and 2D, we generalize the concept of lines and planes to "hyperplanes"
- In 2D, a line is represented by two points
- In 3D, a plane is represented by three points (or a point and a normal vector)
- In higher dimensional spaces, a hyperplane is represented by n-1 points in n-dimensional space
- A hyperplane separates groups of points with two or more dimensions. Also only its normal vector is relevant "hyperplane"

- Equation of line (in 2d space)

$$ax + by = c$$

- In n-2 dim space it has the single element $x_1$

- In n-dim hyperplane separating two groups of points, we can write

$$w_1x_1 + w_2x_2 + \dots + w_nx_n = b$$

- Or in vector form

$$w^Tx = b$$

- Equation of plane (in 3d space)

$$ax + by + cz = d$$

- Equation of hyperplane (in n-dim space)

$$w_1x_1 + w_2x_2 + \dots + w_nx_n = b$$

- Or in vector form

$$w^Tx = b$$

- Equation of line - 2 points

$$(x_2 - x_1)(y - y_1) = (y_2 - y_1)(x - x_1)$$

- Equation of plane - 3 points

$$(x - x_1) \cdot ((y_2 - y_1) \times (z_3 - z_1)) = (y - y_1) \cdot ((x_2 - x_1) \times (z_3 - z_1)) + (z - z_1) \cdot ((x_2 - x_1) \times (y_3 - y_1))$$

- Equation of hyperplane - n points

$$\det \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ x_{11} & x_{12} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,1} & x_{n-1,2} & \dots & x_{n-1,n} \end{bmatrix} = 0$$

- Equation of line - 1 point + slope

$$y - y_1 = m(x - x_1)$$

- Equation of plane - 1 point + normal vector

$$(x - x_1) \cdot n_x + (y - y_1) \cdot n_y + (z - z_1) \cdot n_z = 0$$

- Equation of hyperplane - 1 point + normal vector

$$(x - x_1) \cdot n_x + (y - y_1) \cdot n_y + (z - z_1) \cdot n_z + \dots = 0$$

- Equation of line - 2 points - 2 dimensions

$$\det \begin{bmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} = 0$$

- Equation of plane - 3 points - 3 dimensions

$$\det \begin{bmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{bmatrix} = 0$$

- Equation of hyperplane - n points - n dimensions

$$\det \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ x_{11} & x_{12} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,1} & x_{n-1,2} & \dots & x_{n-1,n} \end{bmatrix} = 0$$

## Metrics

- A matrix is a rectangular array of numbers arranged in rows & columns
- each cell is identified by

$$A_{ij}$$

- where i is row, j is column
- i refers to row, j refers to column

### Types of matrices

- row matrix or row vector

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad 1 \times 3$$

- column matrix or column vector

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad 3 \times 1$$

- square matrix (rows = columns)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad 3 \times 3$$

- diagonal of square matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{diagonal} = 1, 5, 9$$

- identity matrix (square matrix where all diagonal items are 1)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad I_3$$

- diagonal matrix (a square matrix where non diagonal items are 0)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

- scalar matrix (a diagonal matrix whose all diagonal items are equal)

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

- identity matrix (a square matrix where all items on diagonal are 1)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad I_3$$

- zero matrix (where all items are 0)

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

### Metrics equality (when 2 metrics are equal)

- order should be same means same dims (metrics a & b) (the number of rows & columns are the same)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

- all items should match at corresponding position (a's metrics = b's)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

### scalar operations on metrics

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\rightarrow 4A = \begin{bmatrix} 4 & 8 & 12 \\ 16 & 20 & 24 \end{bmatrix}$$

- addition
- subtraction
- multiplication

### metrics operations

- addition
- subtraction
- multiplication
- division

**In order to perform operations on metrics, EQUAL sized metrics should be same**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

**metrics addition:**

- metrics addition means we perform 2 scalar operations. 1 first we perform multiplication with "1" in A because "1" is an active cell and then we perform addition with 5 & 6

**commutative law** $A + B = B + A$

**Associative law** $(A + B) + C = (B + C) + A$

### Additiive identity & Additive inverse

- if you have a metrix A and you want to find what you can add to it to get A back, you're essentially looking for the additive identity, which is the zero matrix

$$A + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{additive identity}} = A \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$A + \underbrace{\begin{bmatrix} -1 & -2 & -3 \\ -4 & -5 & -6 \end{bmatrix}}_{\text{additive inverse}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- if you have a metrix A and you want to find what you can add to it to get its negative, which is negative A, or -A

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -3 \\ -4 & -5 & -6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}_{\text{additive identity}} + \underbrace{\begin{bmatrix} -1 & -2 & -3 \\ -4 & -5 & -6 \end{bmatrix}}_{\text{additive inverse}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = X$$

$$2 \times 3 \quad 3 \times 2$$

$$2 \times 3 \quad 3 \times 2$$

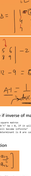
### Metric Multiplication

- **Rule**
- The amount of columns of metrix a = the amount of rows of metrix b
- Then metrix shape = 1st metrix row x 2nd metrix column

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} \quad 2 \times 2$$

$$2 \times 3 \quad 3 \times 2$$

$$2 \times 3 \quad 3 \times 2 \rightarrow 2 \times 2$$




- **step by step**
- Multiply each row in the first metrix element by element in a single column in the second metrix (row 1 x column 1)
- Add up the results (multiplication) and concatenate the product under neighbor
- Descendents (DES) => (NEC)
- Ancestors (ANC) => (NEC) => (DES)

## Multiplicative Identity

## Transpose of a Metrix

- when you transpose a metrix, essentially the X-axis to diagonal. Imagine you have a grid of numbers. If you transpose it, you're flipping the grid so that the rows now become the columns and the columns become the rows. It's like looking at the grid from a different angle. For example, if you have a 2x3 grid, it becomes a 3x2 grid. The numbers don't change, just their position.

## Symmetric Metrix

- A metrix which remains unchanged when transposed

$$\text{Symmetric Metrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

$$A = A^T$$

## Skew Symmetric Metrix

- A metrix which equals negative transposed, just negative metrix

$$A^T = -A$$

## Important Rules

$$(AB)^T = B^TA^T$$

$$(A + B)^T = A^T + B^T$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

## How to calculate Inverse of a ++++ (a-1)

- whenever we need to divide 2 metrix, since division is not possible, we do multiplication "metrix with "inverse of another metrix"

$$\frac{a}{b} = a \times \frac{1}{b} \quad \frac{1}{b} = \text{multiplicative inverse}$$

$$\frac{a}{b} = a \times b^{-1} \quad X = \frac{B}{A} \quad X = B \times \frac{1}{A}$$

- for that we need to calculate a-1

- To find a-1, you're essentially look at metrix and modifying, only square metrics can be inverted. not all square metrics can be inverted (no)

$$A^{-1} = \frac{1}{\det(A)} \cdot \text{Adj}(A)$$

- Find a determinant of a metrix
- Find adjoint (a matrix of cofactors)
- Divide 1 by the determinant
- Multiply the adjoint by the result from step 3 to get the inverse matrix

## Determinant

- In linear algebra, the determinant is a special number that can calculate for square matrices. It tells us a few important things:
- Metric invertibility: If the determinant of a matrix is not zero, then the matrix is invertible, meaning we can find its inverse. This is useful for solving systems of linear equations.
- Scaling factor of transformations: Geometrically, the absolute value of the determinant represents how much the transformation represented by the matrix scales areas or volumes. For example, if the determinant is 2, the transformation will double the area (in 2D) or volume (in 3D).
- Volume Folding factor: When we apply linear transformations to geometric shapes, the determinant tells us how much the volume of the shape will be scaled. A negative value means the orientation of the shape is also flipped (like a mirror image).
- In 2D, for example, the determinant is a number that tells us how much the area changes. If it's positive, the shape stays the same size or gets bigger. If it's negative, it also flips the shape.

## Determinant calculation

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \det(A) = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$$

$$\det(A) = \Delta = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \Delta = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \quad \begin{matrix} -3x - 2x \\ 1 \end{matrix}$$

$$\Delta = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} \quad \begin{matrix} -3x - 2x \\ 1 \end{matrix} \rightarrow \begin{matrix} -3x - 2x \\ 1 \end{matrix} \rightarrow \text{order}$$

$$\boxed{3x3} \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad 3 \times 3$$

$$\det(A) = \Delta = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} \begin{matrix} 1 & 2 \\ 4 & 5 \\ 7 & 8 \end{matrix}$$

$$\Delta = 3 \times \begin{vmatrix} 1 & 2 \\ 4 & 5 \end{vmatrix} - 2 \times \begin{vmatrix} 1 & 2 \\ 7 & 8 \end{vmatrix} + 3 \times \begin{vmatrix} 1 & 2 \\ 7 & 8 \end{vmatrix}$$

$$= -3 + 6 - 3 = 0 \quad \text{order} \rightarrow \text{order}$$

$$\text{in } A \Rightarrow \frac{1}{\det(A)} \quad \text{Invert } X$$

## How to decide if inverse of metrix is possible?

1. You should be square matrix
2. det(A) "determinant" of A is not equal to zero
3. det(A) "determinant" of A is not equal to zero, if it is not zero according to inverse formula "A-1 = 1/det(A) * adj(A)"
4. det(A) = zero determinant "A" is 0 which singular metrix, we can only find inverse of non-singular metrix

## Minor Calculation

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- each element in metrix will have its own minor
- 1st minor is 1st row, 1st column & 1st row, 1st column

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- If you want to find minor of 1 then delete its row & its column to find determinant of it
- In the order, 1st is column & determinant of row is 1st determinant of row

$$M_{11} = 0 \quad M_{12} = 0$$

$$M_{13} = 0 \quad M_{21} = 0$$

- the rule for 1st row, 1st column, delete the row & column to find determinant of it

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \quad M_{11} = \begin{vmatrix} 5 & 6 \\ 8 & 9 \end{vmatrix}$$

## Cofactor

- minor metrix items were denoted by a minor was denoted by M, cofactors denoted by C
- delta sign has a cofactor just like M

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \quad \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

- $M_{ij} \rightarrow (-1)^{i+j} M_{ij}$

## Adjoint

- Adjoint also called "Adjugate"
- First choose of every item in metrix, take cofactor of it & transpose of that metrix = Adjoint

## Lets now calculate Inverse of A

$$A^{-1} = \frac{1}{\det(A)} \cdot \text{Adj}(A)$$

$$\text{order} \rightarrow \text{order}$$

$$A \times A^{-1} = I$$

$$AX = B \leftarrow$$

$$X = A^{-1}B$$

$$A \times A^{-1} = A^{-1}B$$

$$I \times X = A^{-1}B$$

$$X = A^{-1}B$$

- [x] [x] [x] [x] [x]
- [x] [x]
- [x] [x]

```
[1, 3, 4]
[6, 4, 1]
[2, 1, 2]
```

```
[5, 16, 11]
```

```
A = np.array([[1,3,4],[6,4,1],[2,1,2]])
```

```
print("A: \n",
      A,
      "\n")
```

```
A: 
 [[1 3 4]
 [6 4 1]
 [2 1 2]] 
```

```
B = np.array([5,16,11])
```

```
print("B: \n",
      B,
      "\n")
```

```
B: 
 [ 5 16 11] 
```

```
X = np.linalg.solve(A, B)
```

```
print("X: \n",
      X,
      "\n")
```

```
X: 
 [-1.5 2.5 3.5] 
```

```
X = np.linalg.inv(A) @ B
```

```
print("X: \n",
      X,
      "\n")
```

```
X: 
 [-1.5 2.5 3.5] 
```

```
np.dot(A,X), np.dot(A, X)*B, (A*X)-B
```

```
(array([5., 5., 5.]),
 array([0., 0., 0.]),
 array([0., 0., 0.]))
```