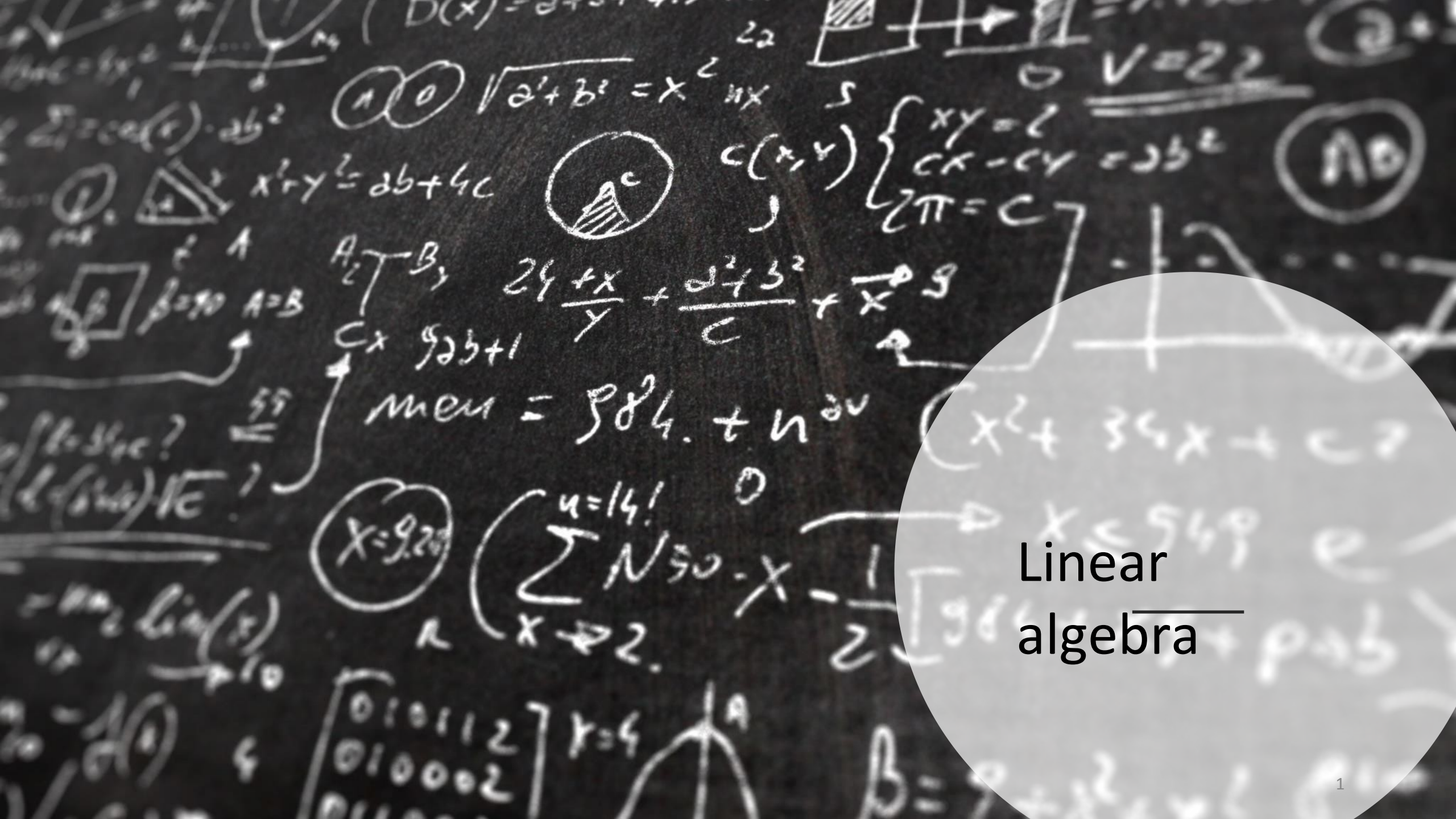


Linear algebra

Linear Algebra for data science

Session 1


IDEA
IDEAS
IDEAS
IDEAS
IDEAS
IDEAS
IDEAS
IDEAS
IDEAS


Brainstorming

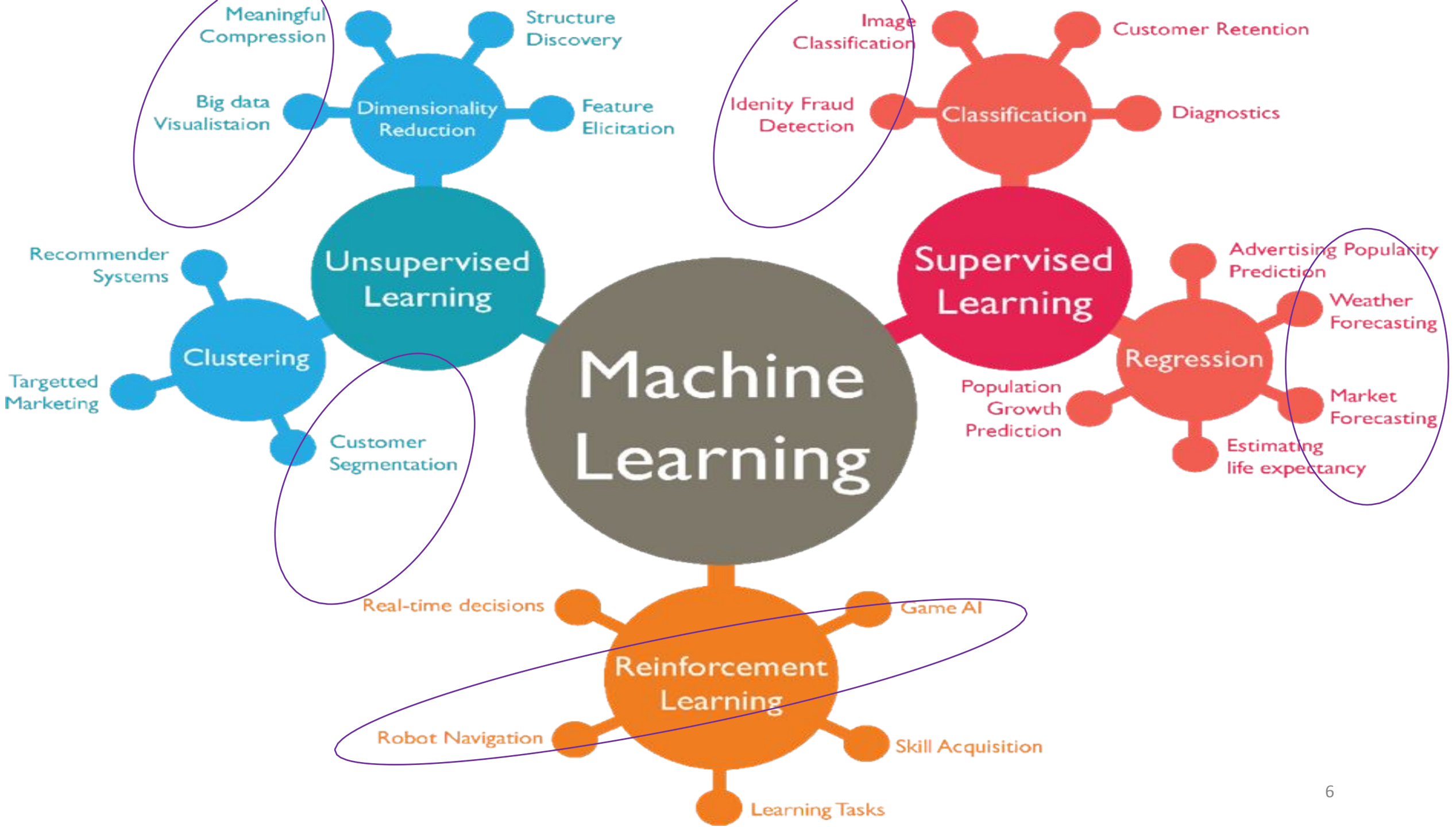
Pattern recognition

- Machine Learning.
- Statistical analysis

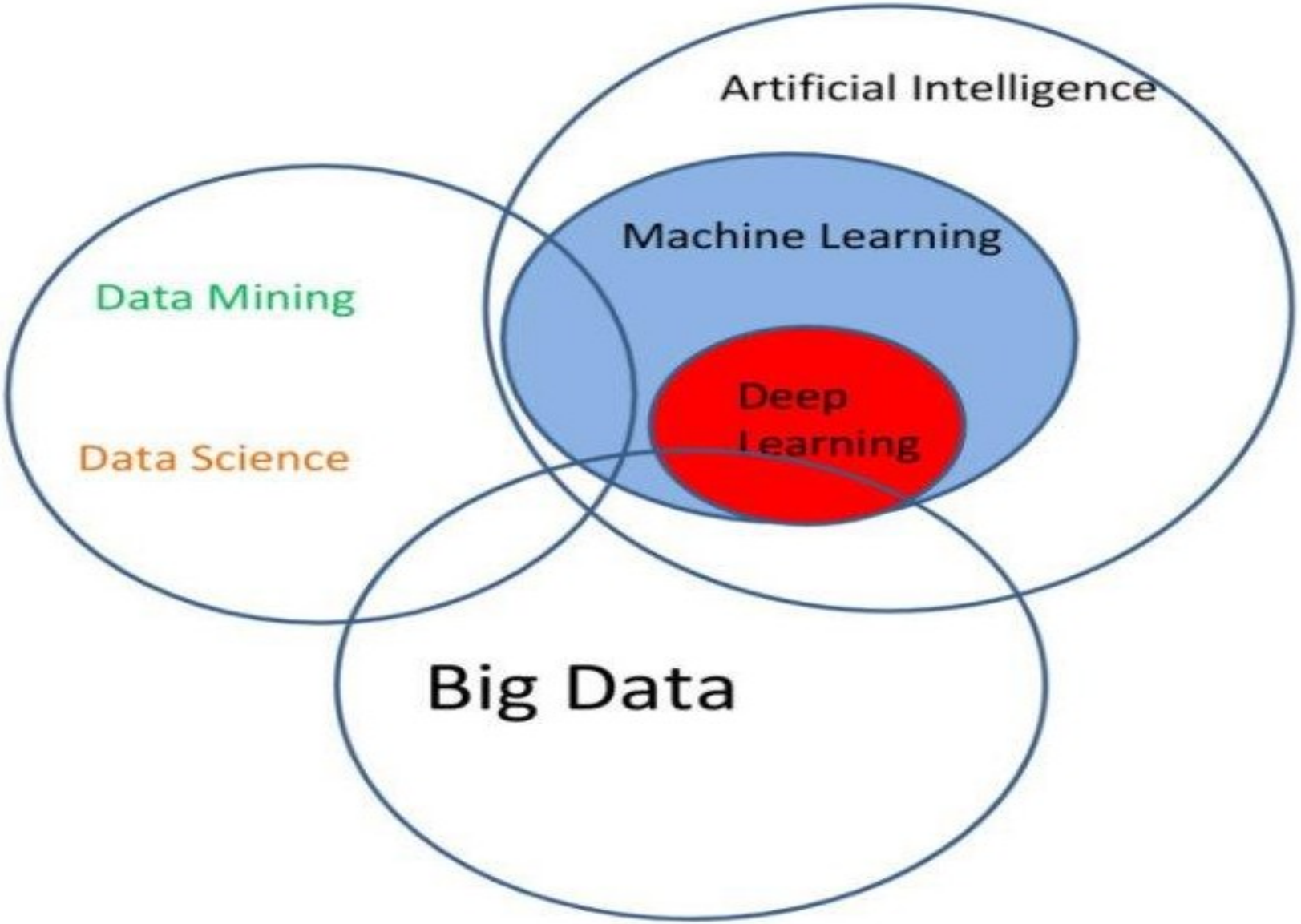
  .
- Computer vision.

Machine learning definition

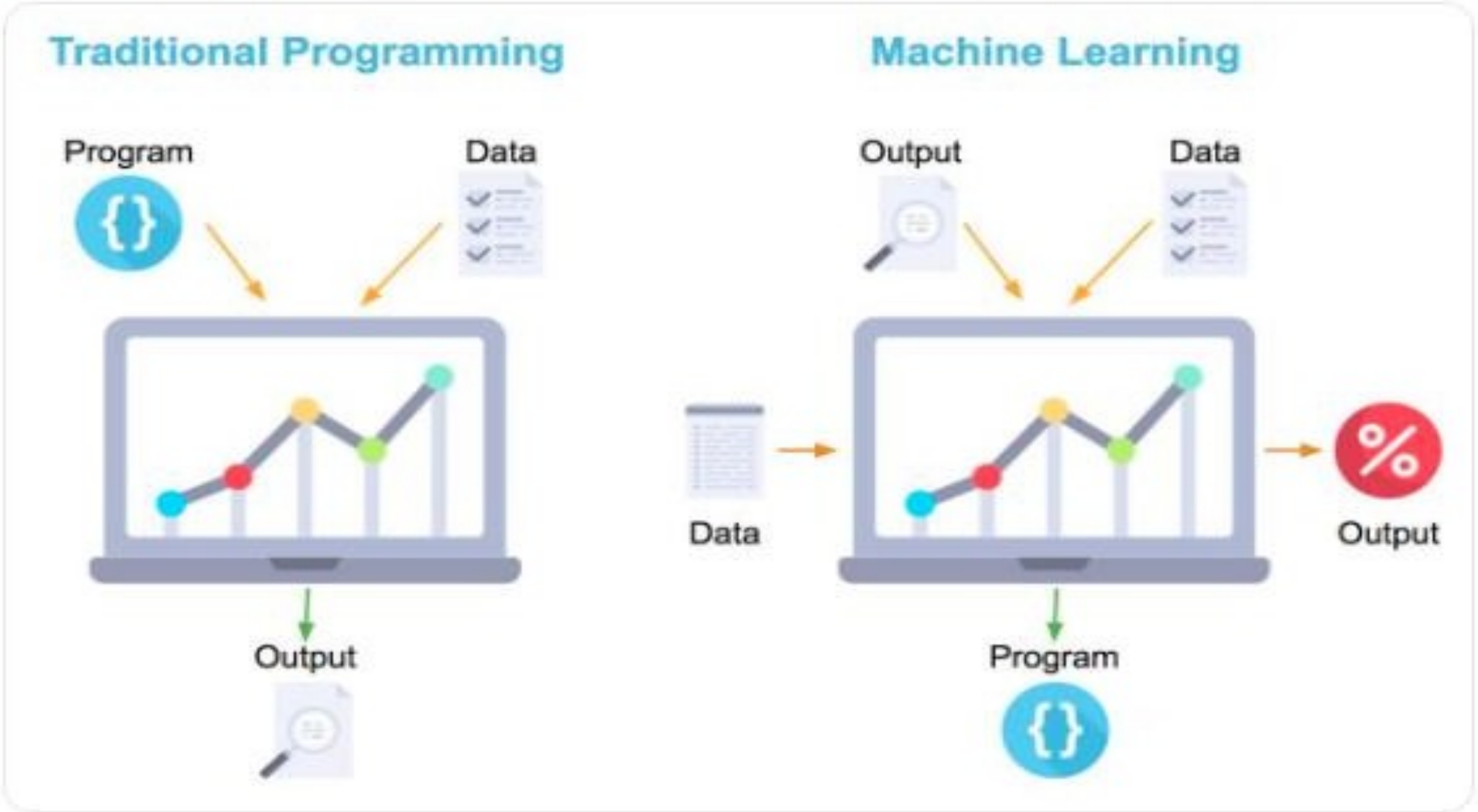
In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".


Machine Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning
Dimensionality Reduction
Clustering
Classification
Regression
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Recommender Systems
Targetted Marketing
Customer Segmentation
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Machine learning related fields



Traditional programming or machine learning



Data + rules □ result

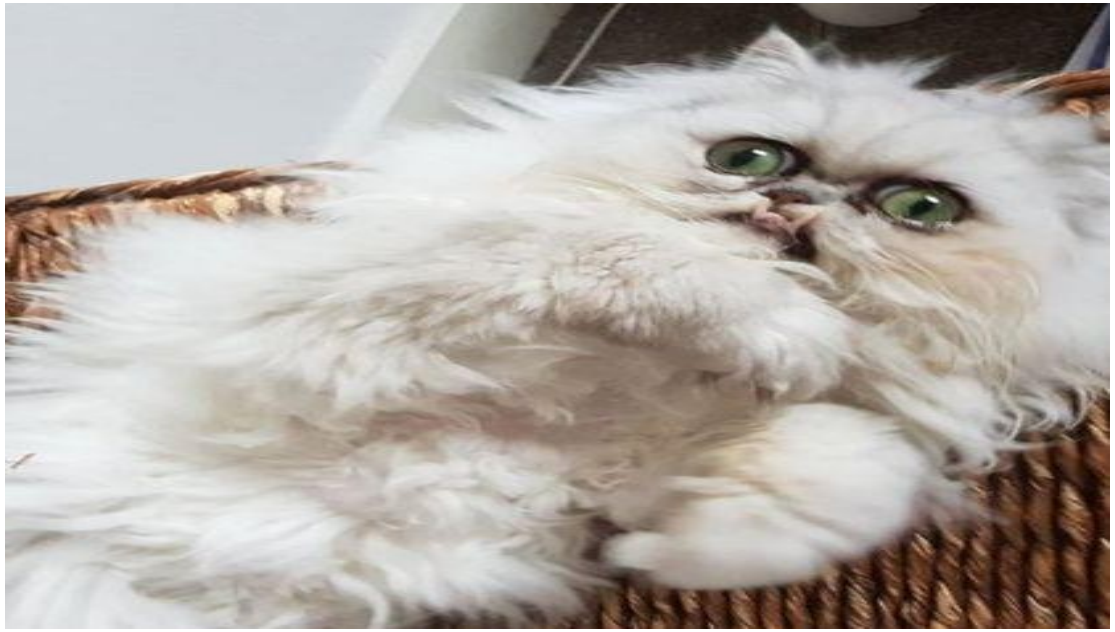
area sqare

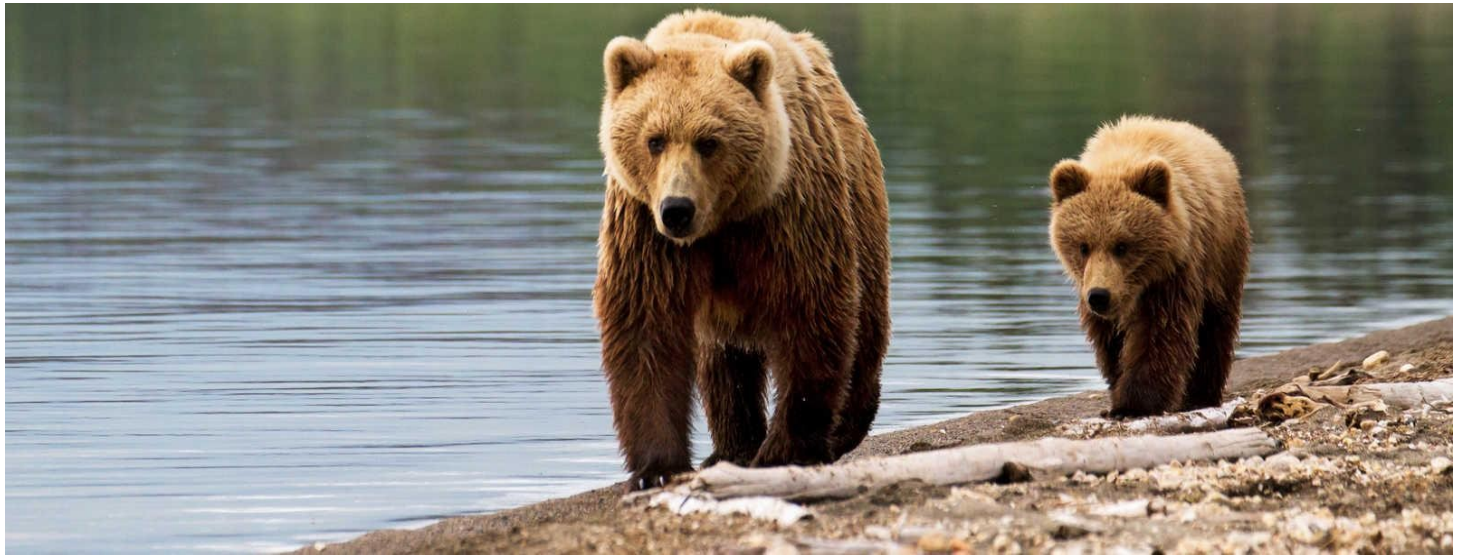
ML

data + results □ rules

area ,num of rooms , location + prices □ rule

Traditional programming or machine learning

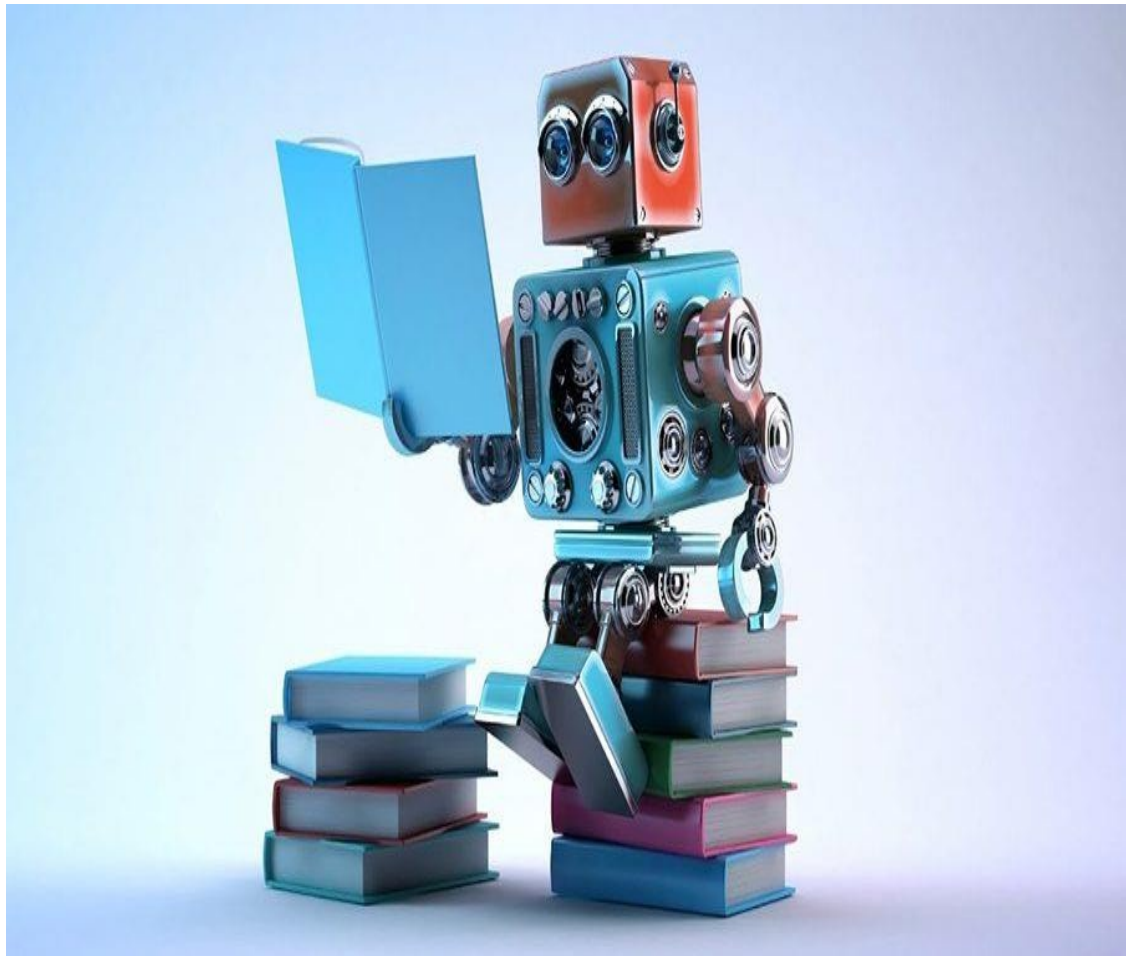





© Reuters/V. Yarinskiy


B·A·B
THE WIZARD OF OZ
80

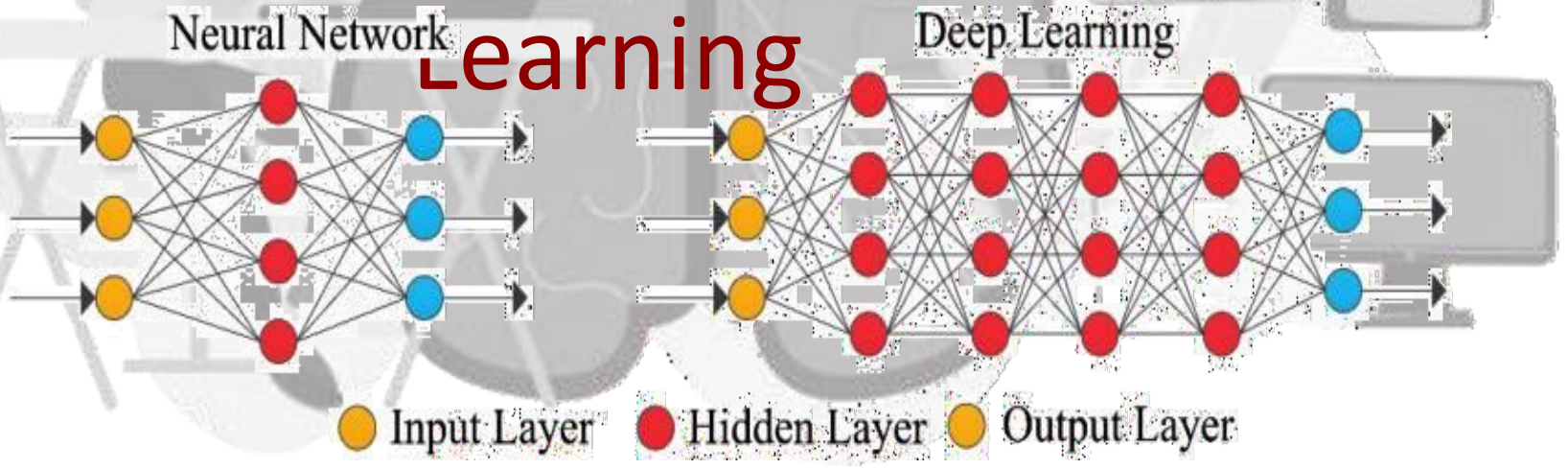
Learning scope



Machine Learning (ML)

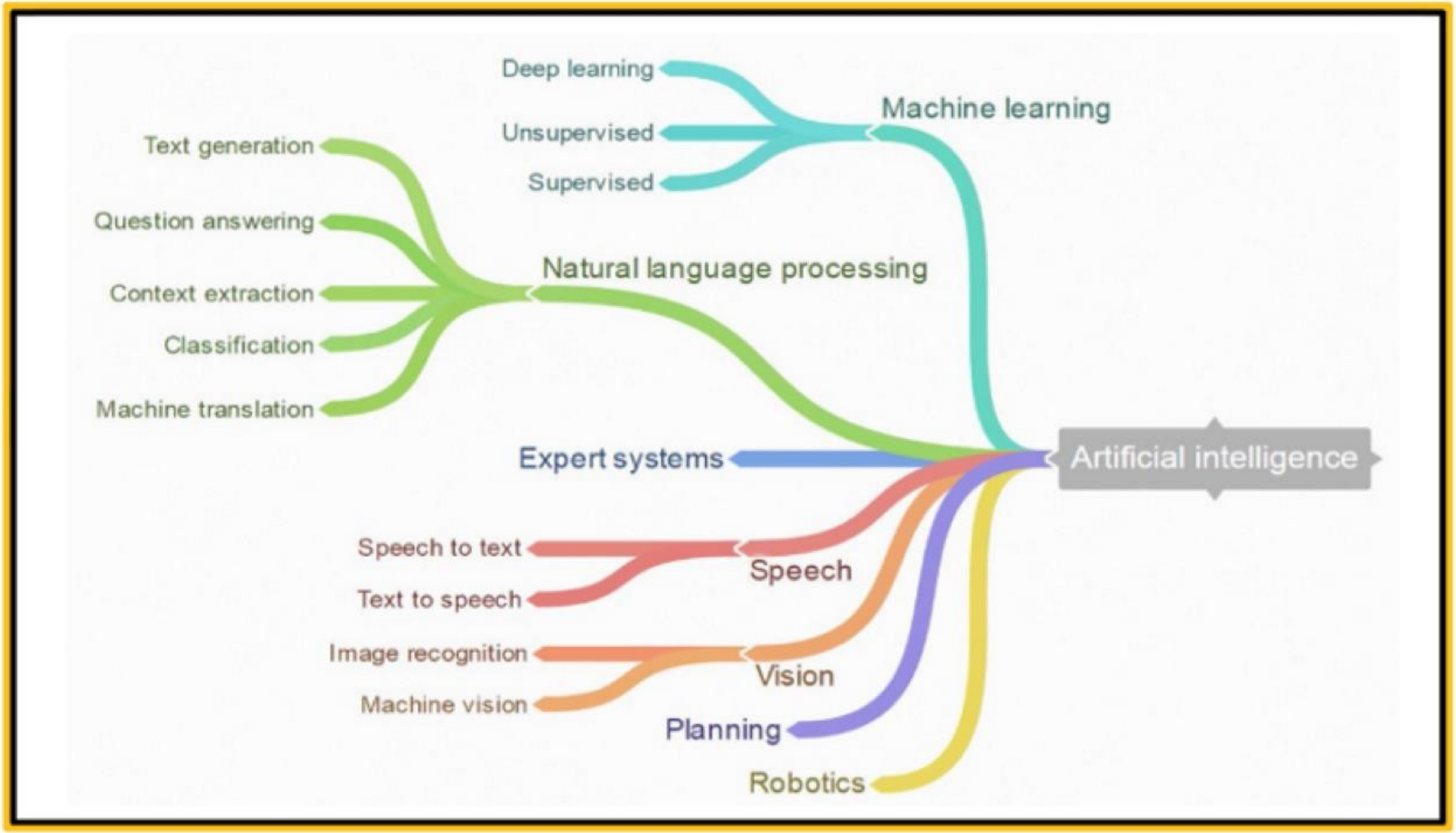
- The study of computer algorithms that have the ability to automatically learn and improve from experience without being explicitly programming.
- AI is heavily dependent on ML.

Deep Learning

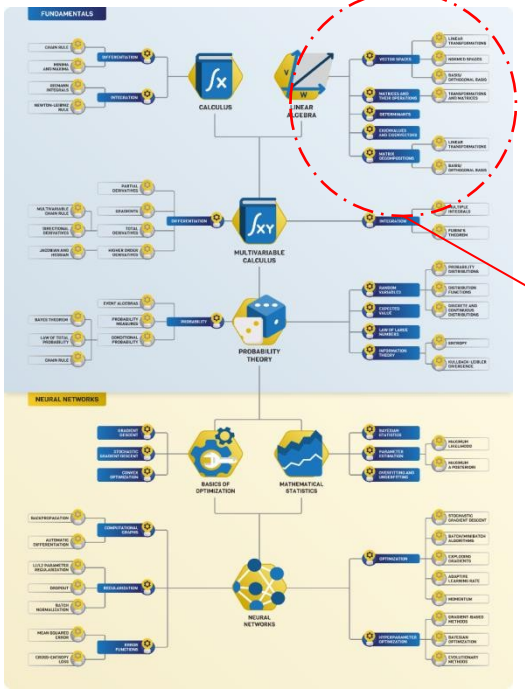


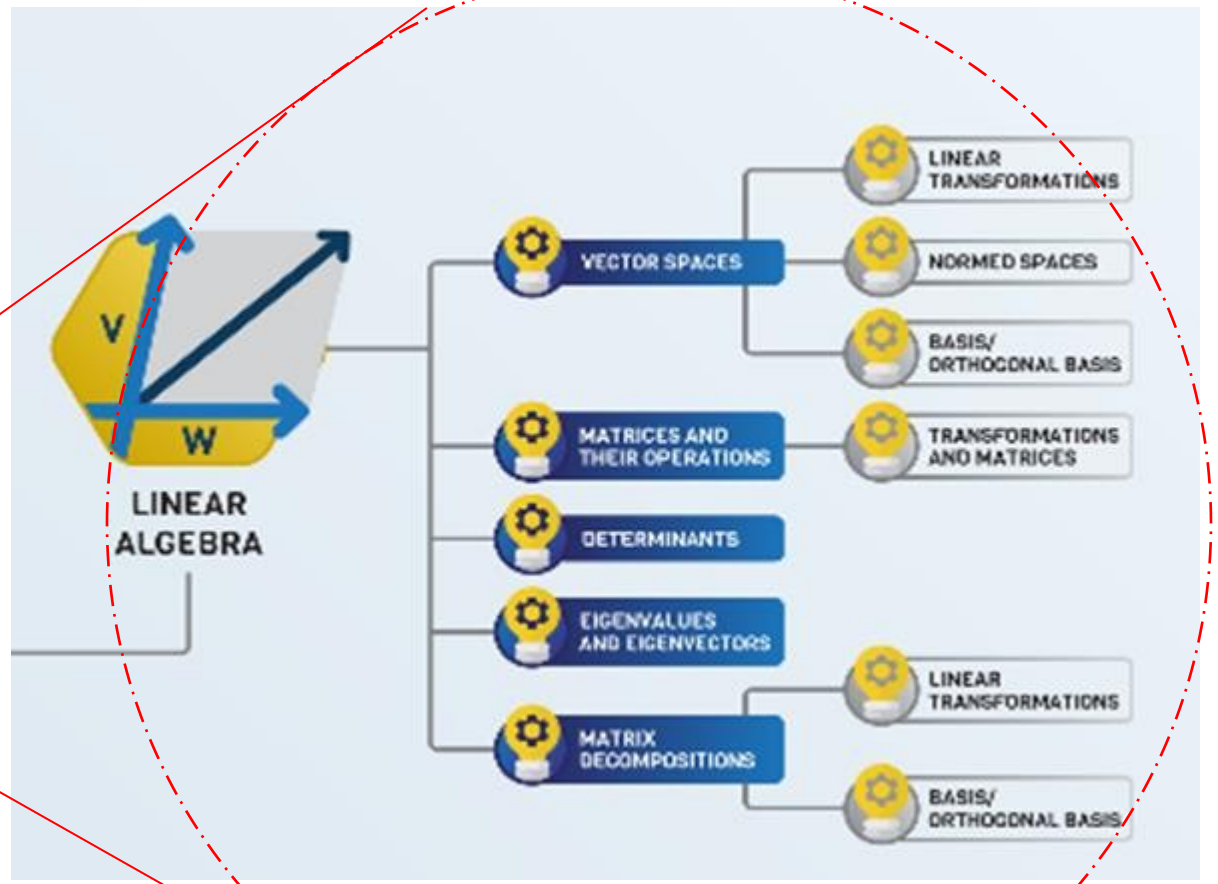
[Source: DOI:

Machine learning related fields



Road Map from Linear Algebra to ML





Road Map from Linear Algebra to ML

Learning Outcomes

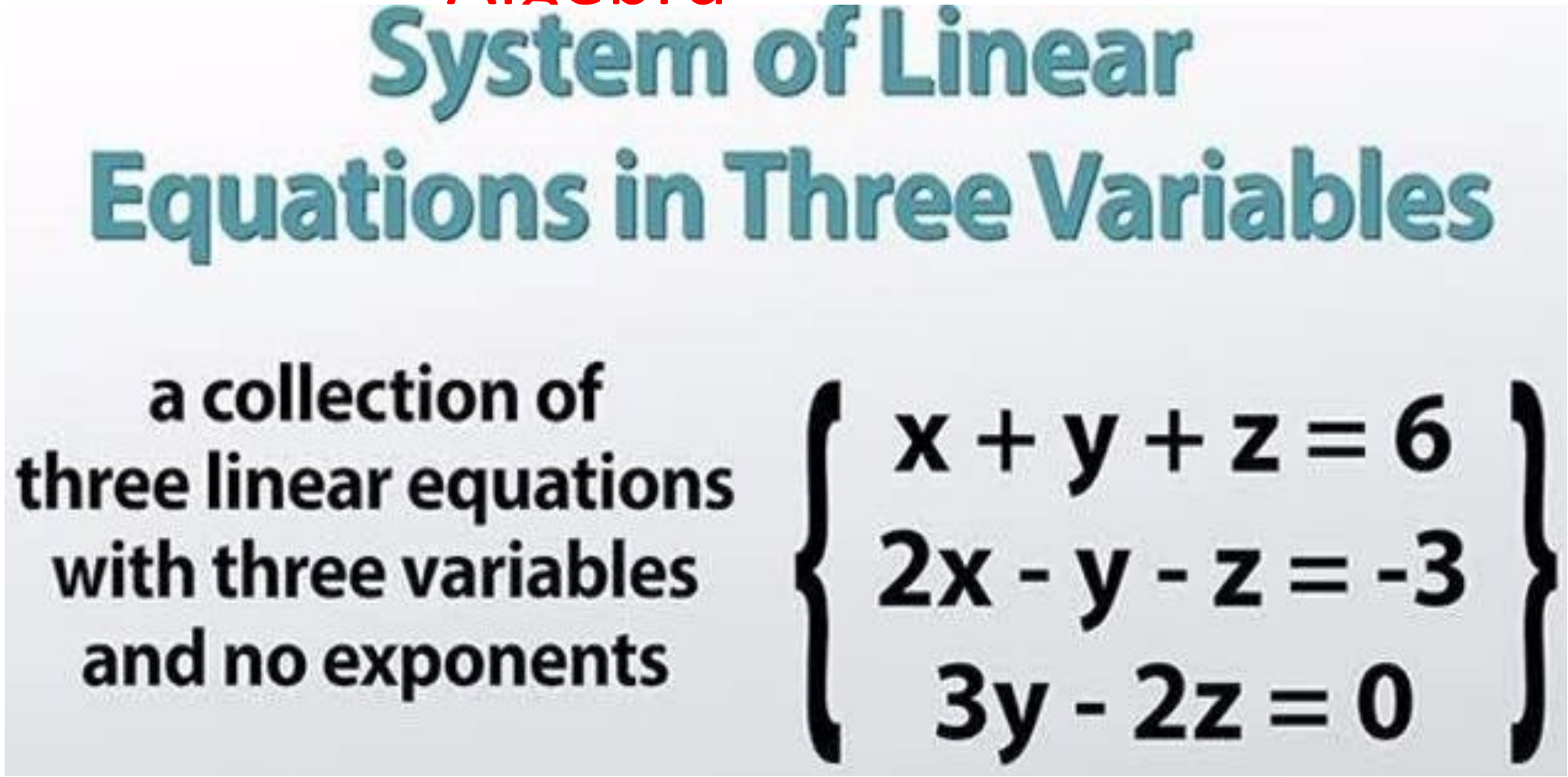
- Connection between linear algebra and data science
- Linear system of equation and linear transformations: intuition of vectors and matrices
- Matrix/matrix, vector/vector sum and matrix/vector, scalar/vector, scalar/matrix products
- Linear combination, linear independence
- Vector space
- Basis and dimension
- Determinant: definition and properties
- Inversion of matrices
- Eigenvectors and eigenvalues

Road Map from Linear Algebra to ML

Learning Outcomes

- Connection between linear algebra and data science
- Linear system of equation and linear transformations: intuition of vectors and matrices
- Matrix/matrix, vector/vector sum and matrix/vector, scalar/vector, scalar/matrix products
- Linear combination, linear independence
- Vector space
- Basis and dimension
- Determinant: definition and properties
- Inversion of matrices
- Eigenvectors and eigenvalues

Why Linear Algebra

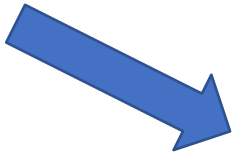


Applications for solving linear systems of equation

- Planning
- Decision Engineering
- Financial forecasting
- Technological trend forecasting
- Natural Language processing

System of Linear Equations

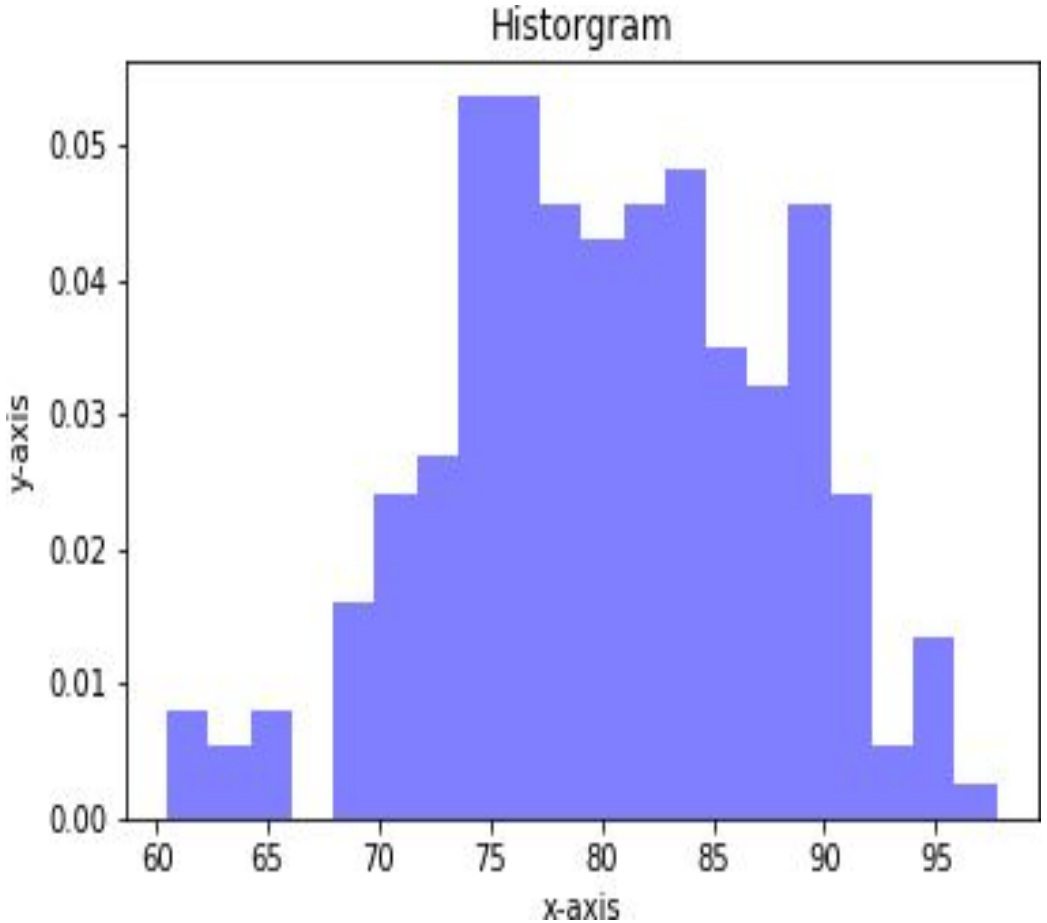
$$
\begin{aligned}
x + y + z &= 6 \\
2y + 5z &= -4 \\
2x + 5y - z &= 27
\end{aligned}
$$

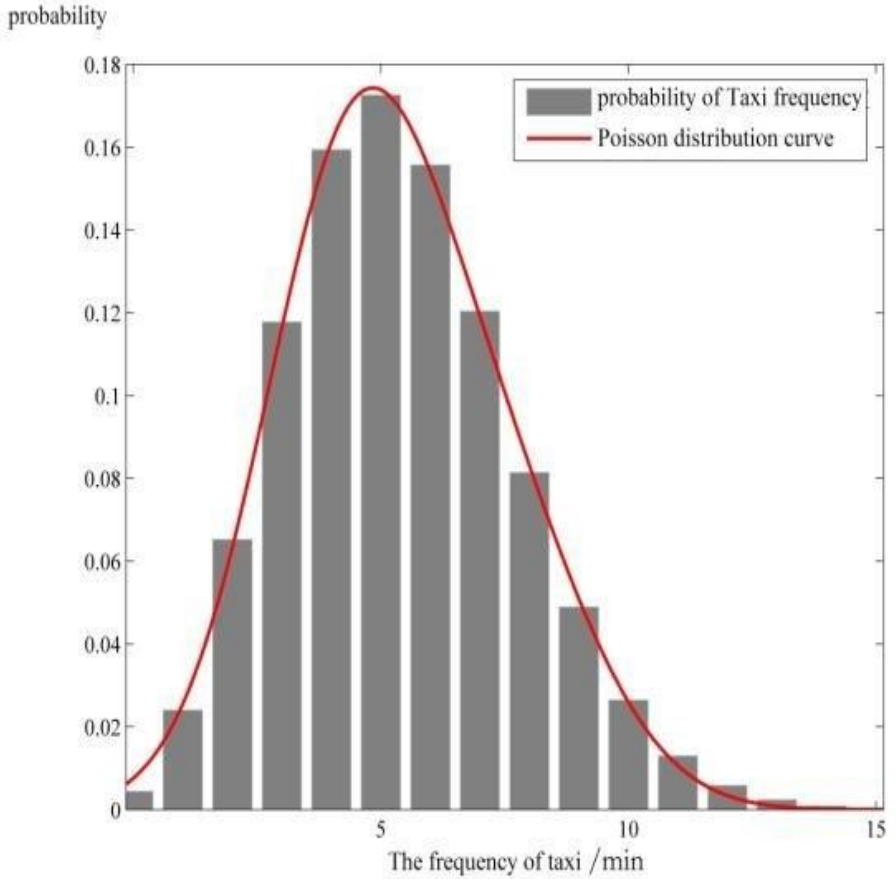


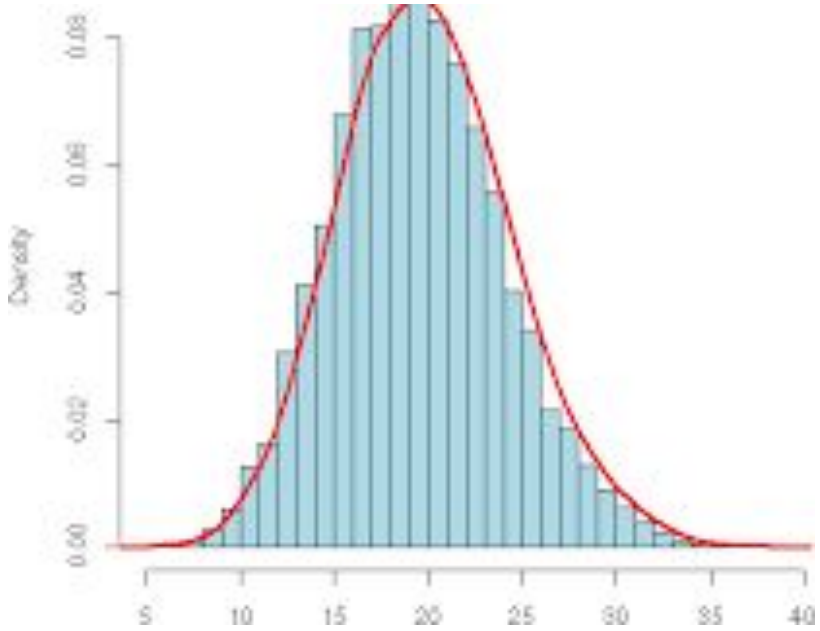
$$
\begin{aligned}
1 \quad 1 \quad 1 &= 6 \\
0 \quad 2 \quad 5 &= -4 \\
2 \quad 5 \quad -1 &= 27
\end{aligned}
$$

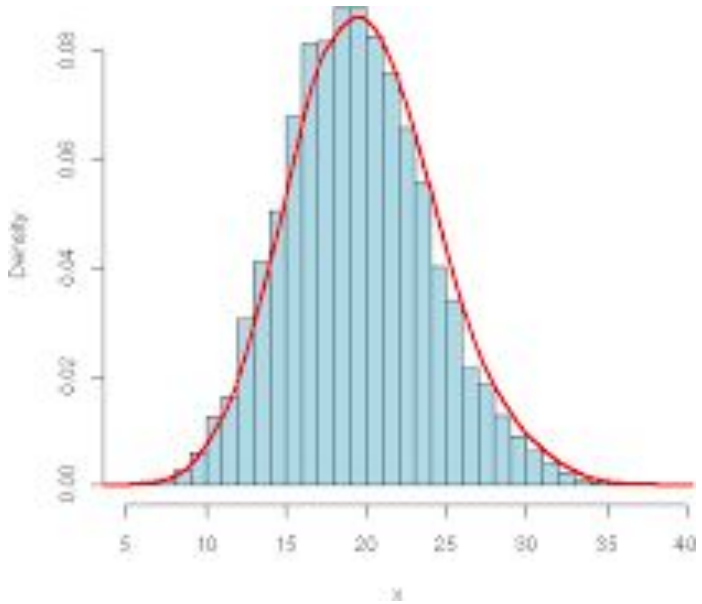
$$
\begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 5 \\ 2 & 5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ -4 \\ 27 \end{bmatrix}
$$

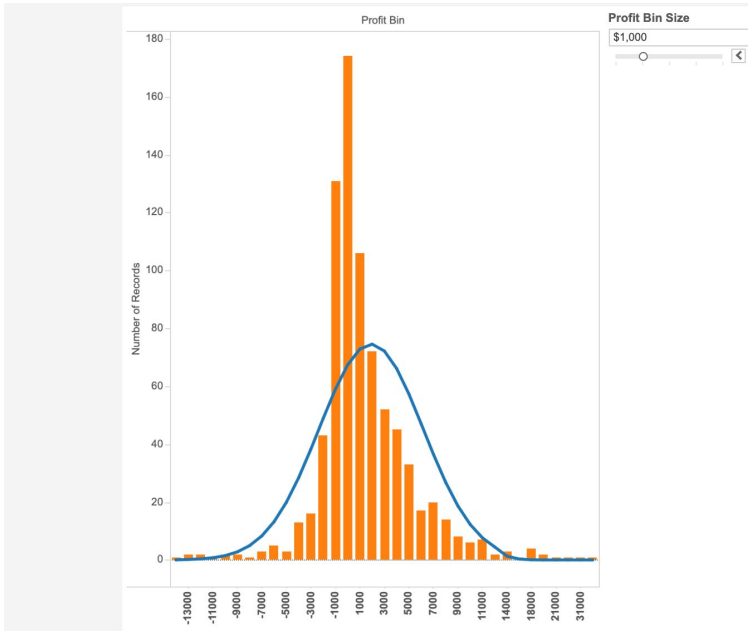
Why Linear Algebra

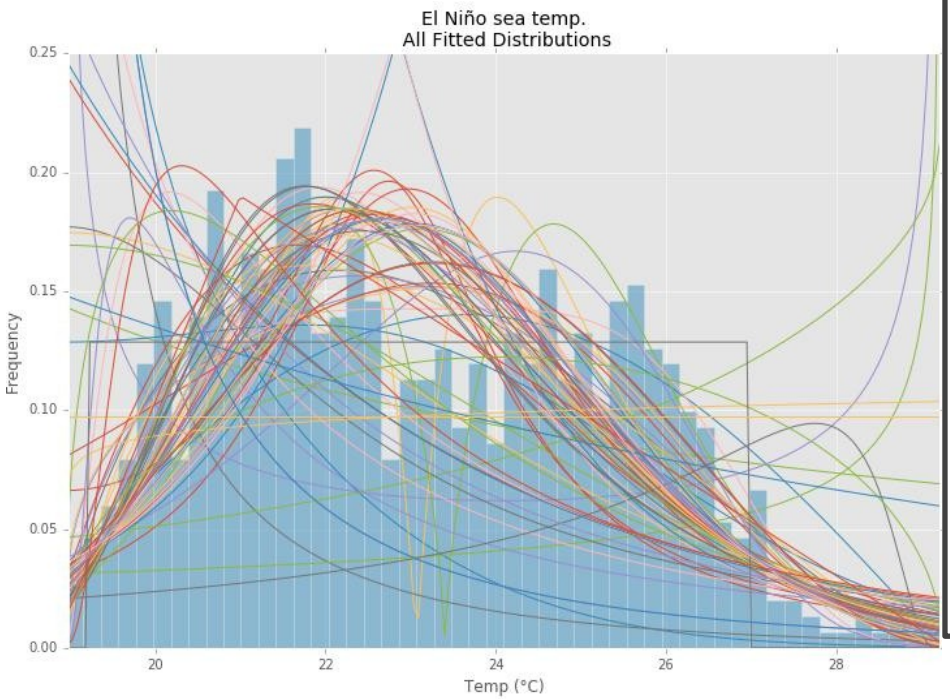








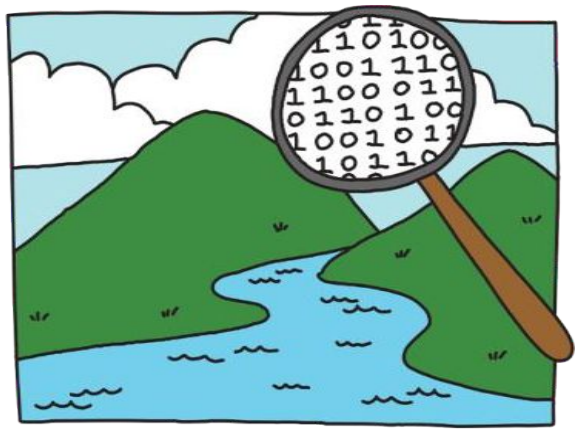




Solving optimization problem to fit input data

Why Linear Algebra

This is how a computer sees an image

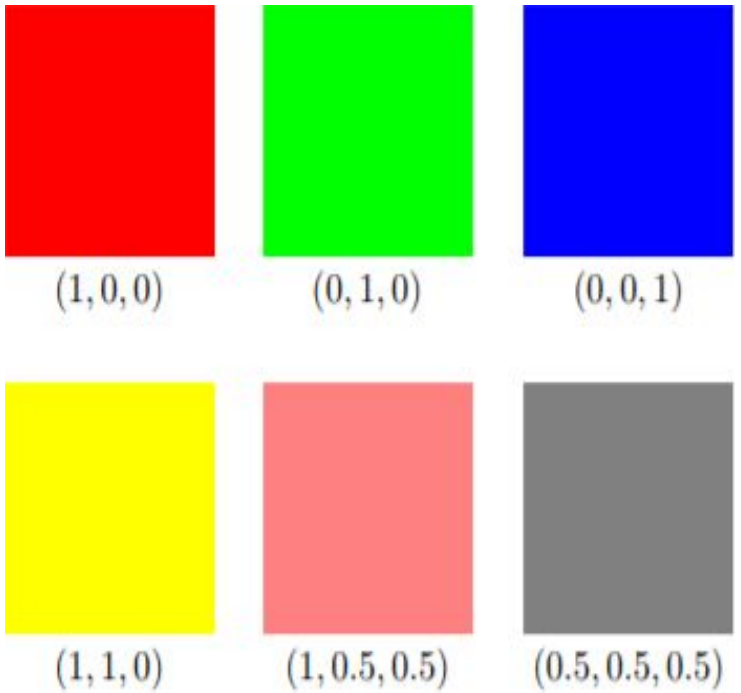


```
[[  9   1  29  70 114  76   0   8   4   5   5   0 111 162   9   8  62  62]
 [  3   0  33  61 102 106  34   0   0   0   0  49 182 150   1  12  65  62]
 [  1   0  40  54 123  90  72  77  52  51  49 121 205  98   0  15  67  59]
 [  3   1  41  57  74  54  96 181 220 170  90 149 208  56   0  16  69  59]
 [  6   1  32  36  47  81  85  90 176 206 140 171 186  22   3  15  72  63]
 [  4   1  31  39  66  71  71  97 147 214 203 190 198  22   6  17  73  65]
 [  2   3  15  30  52  57  68 123 161 197 207 200 179   8   8  18  73  66]
 [  2   2  17  37  34  40  78 103 148 187 205 225 165   1   8  19  76  68]
 [  2   3  20  44  37  34  35  26  78 156 214 145 200  38   2  21  78  69]
 [  2   2  20  34  21  43  70  21  43 139 205  93 211  70   0  23  78  72]
 [  3   4  16  24  14  21 102 175 120 130 226 212 236  75   0  25  78  72]
 [  6   5  13  21  28  28  97 216 184  90 196 255 255  84   4  24  79  74]
 [  6   5  15  25  30  39  63 105 140  66 113 252 251  74   4  28  79  75]
 [  5   5  16  32  38  57  69  85  93 120 128 251 255 154  19  26  80  76]
 [  6   5  20  42  55  62  66  76  86 104 148 242 254 241  83  26  80  77]
 [  2   3  20  38  55  64  69  80  78 109 195 247 252 255 172  40  78  77]
 [ 10   8  23  34  44  64  88 104 119 173 234 247 253 254 227  66  74  74]
 [ 32   6  24  37  45  63  85 114 154 196 226 245 251 252 250 112  66  71]]
```

Why Linear Algebra

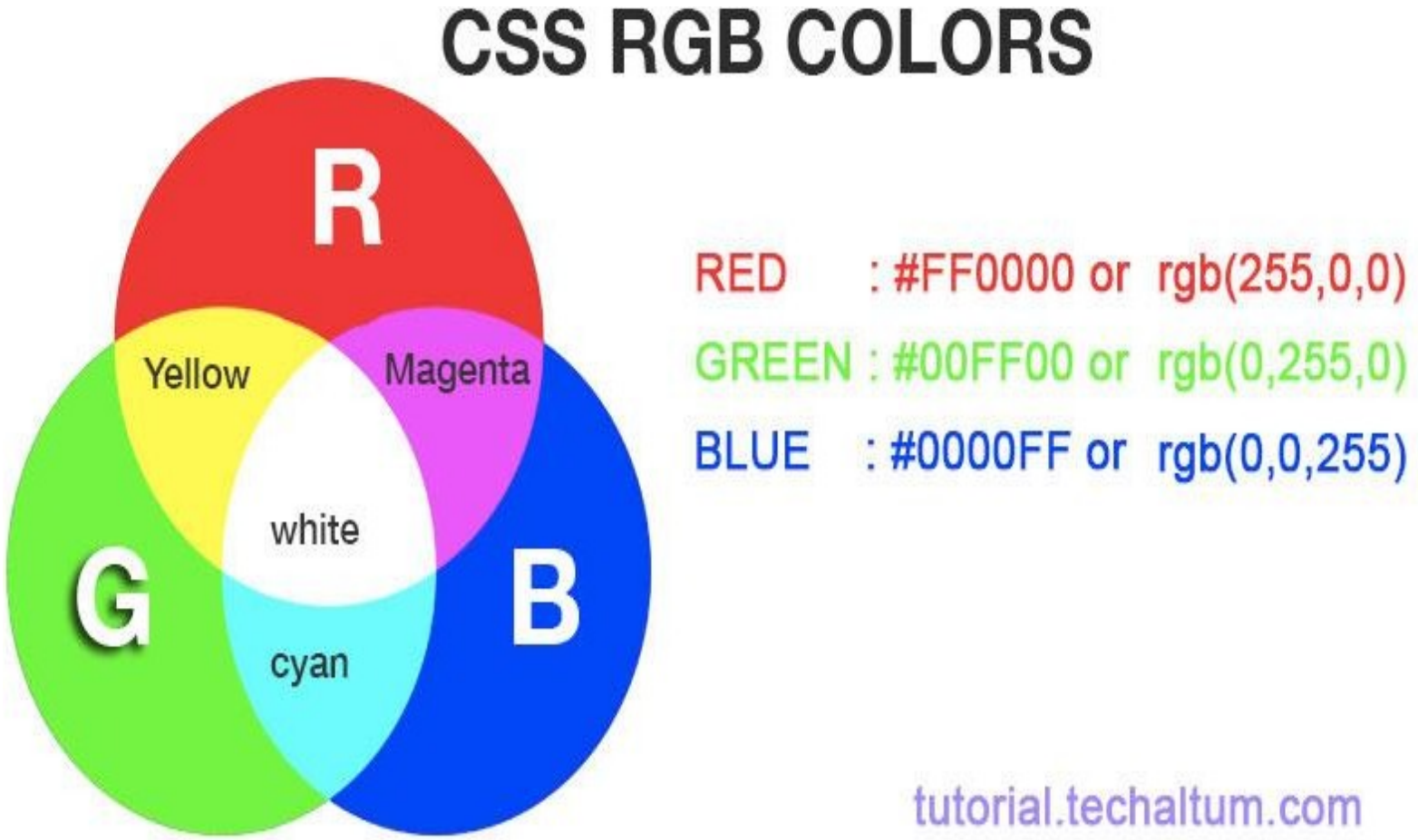
1- Vectors and matrices are used for Data representation … for example: in representing colors RGB system (Red,Green,Blue)

" Computer Graphics"



Computer graphics & Image processing

vectors and matrices are used for Data representation … for example: in representing colors RGB system (Red,Green,Blue) “ Computer Graphics”



Computer graphics & Image processing

Simple Internet HTML Color Table

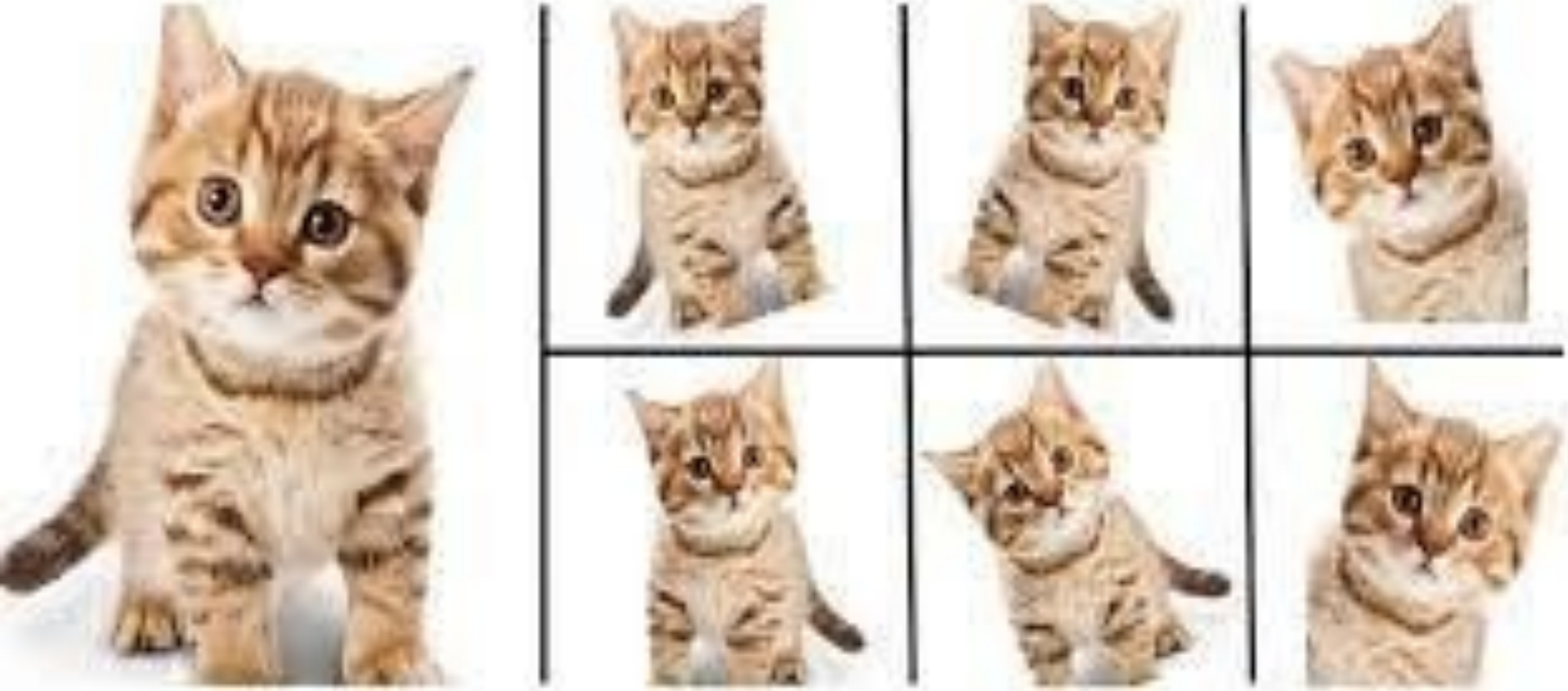
with HTML Codes - 81 color set.

	Col. 1	Col. 2	Col. 3	Col. 4	Col. 5	Col. 6	Col. 7	Col. 8	Col. 9
Row 1	FFFFFF	000000	333333	666666	999999	CCCCCC	CCCC99	9999CC	666699
Row 2	660000	663300	996633	003300	003333	003399	000066	330066	660066
Row 3	990000	993300	CC9900	006600	336666	0033FF	000099	660099	990066
Row 4	CC0000	CC3300	FFCC00	009900	006666	0066FF	0000CC	663399	CC0099
Row 5	FF0000	FF3300	FFFF00	00CC00	009999	0099FF	0000FF	9900CC	FF0099
Row 6	CC3333	FF6600	FFFF33	00FF00	00CCCC	00CCFF	3366FF	9933FF	FF00FF
Row 7	FF6666	FF6633	FFFF66	66FF66	66CCCC	00FFFF	3399FF	9966FF	FF66FF
Row 8	FF9999	FF9966	FFFF99	99FF99	66FFCC	99FFFF	66CCFF	9999FF	FF99FF
Row 9	FFCCCC	FFCC99	FFFFCC	CCFFCC	99FFCC	CCFFFF	99CCFF	CCCCFF	FFCCFF

Excel color table with RGB values

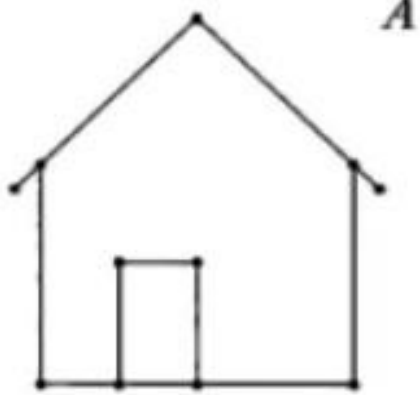
Made by Teoalida - www.teoalida.com/database/colors

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	252	248	244	240	236	232	228	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	228	232	236	240	244	248	252	255	255	255	255	255	255	255	255	255
224	228	232	236	240	244	248	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	252	248	244	240	236	232	228	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224
224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	228	232	236	240	244	248	252	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	252	248	244	240	236	232	228	224
255	255	255	255	255	255	255	255	255	248	240	232	224	216	208	200	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	200	208	216	224	232	240	248	255	255	255	255	255	255	255	255	255
192	200	208	216	224	232	240	248	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	248	240	232	224	216	208	200	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192
192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	200	208	216	224	232	240	248	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	248	240	232	224	216	208	200	192
255	255	255	255	255	255	255	255	255	244	232	220	208	196	184	172	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	172	184	196	208	220	232	244	255	255	255	255	255	255	255	255	255
160	172	184	196	208	220	232	244	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	244	232	220	208	196	184	172	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	172	184	196	208	220	232	244	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	244	232	220	208	196	184	172	160
255	255	255	255	255	255	255	255	255	240	224	208	192	176	160	144	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	144	160	176	192	208	224	240	255	255	255	255	255	255	255	255	255
128	144	160	176	192	208	224	240	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	240	224	208	192	176	160	144	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	144	160	176	192	208	224	240	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	240	224	208	192	176	160	144	128
255	255	255	255	255	255	255	255	255	236	216	196	176	156	136	116	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	116	136	156	176	196	216	236	255	255	255	255	255	255	255	255	255
96	116	136	156	176	196	216	236	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	236	216	196	176	156	136	116	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96
96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	116	136	156	176	196	216	236	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	236	216	196	176	156	136	116	96
255	255	255	255	255	255	255	255	255	232	208	184	160	136	112	88	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	88	112	136	160	184	208	232	255	255	64	64	64	64	64	64	255
64	88	112	136	160	184	208	232	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	232	208	184	160	136	112	88	64	64	64	64	64	64	64	64	64	64	232	232	232	232	232	232	64
64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	88	112	136	160	184	208	232	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	232	208	184	160	136	112	88	64
255	255	255	255	255	255	255	255	255	228	200	172	144	116	88	60	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	60	88	116	144	172	200	228	255	255	255	255	255	255	255	255	255
32	60	88	116	144	172	200	228	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	228	200	172	144	116	88	60	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	60	88	116	144	172	200	228	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	228	200	172	144	116	88	60	32
255	255	255	255	255	255	255	255	255	224	192	160	128	96	64	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	64	96	128	160	192	224	255	255	255	255	255	255	255	255	255
0	32	64	96	128	160	192	224	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	224	192	160	128	96	64	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	64	96	128	160	192	224	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	224	192	160	128	96	64	32	0
224	224	224	224	224	224	224	224	224	196	168	140	112	84	56	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	56	84	112	140	168	196	224	224	224	224	224	224	224	224	224
0	28	56	84	112	140	168	196	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	196	168	140	112	84	56	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	56	84	112	140	168	196	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	224	196	168	140	112	84	56	28	0
192	192	192	192	192	192	192	192	192	168	144	120	96	72	48	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	48	72	96	120	144	168	192	192	192	192	192	192	192	192	192
0	24	48	72	96	120	144	168	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	168	144	120	96	72	48	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	48	72	96	120	144	168	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	168	144	120	96	72	48	24	0
160	160	160	160	160	160	160	160	160	140	120	100	80	60	40	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	40	60	80	100	120	140	160	160	160	160	160	160	160	160	160
0	20	40	60	80	100	120	140	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	140	120	100	80	60	40	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	40	60	80	100	120	140	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	160	140	120	100	80	60	40	20	0
128	128	128	128	128	128	128	128	128	112	96	80	64	48	32	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	32	48	64	80	96	112	128	128	128	128	128	128	128	128	128
0	16	32	48	64	80	96	112	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	112	96	80	64	48	32	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	32	48	64	80	96	112	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	112	96	80	64	48	32	16	0
96	96	96	96	96	96	96	96	96	84	72	60	48	36	24	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	24	36	48	60	72	84	96	96	96	96	96	96	96	96	96
0	12	24	36	48	60	72	84	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	84	72	60	48	36	24	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	24	36	48	60	72	84	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	84	72	60	48	36	24	12	0
64	64	64	64	64	64	64	64	64	56	48	40	32	24	16	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	24	32	40	48	56	64	64	64	64	64	64	64	64	64
0	8	16	24	32	40	48	56	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	56	48	40	32	24	16	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	24	32	40	48	56	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	56	48	40	32	24	16	8	0
32	32	32	32	32	32	32	32	32	28	24	20	16	12	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	8	12	16	20	24	28	32	32	32	32	32	32	32	32	32
0	4	8	12	16	20	24	28	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	28	24	20	16	12	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	8	12	16	20	24	28	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	28	24	20	16	12	8	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

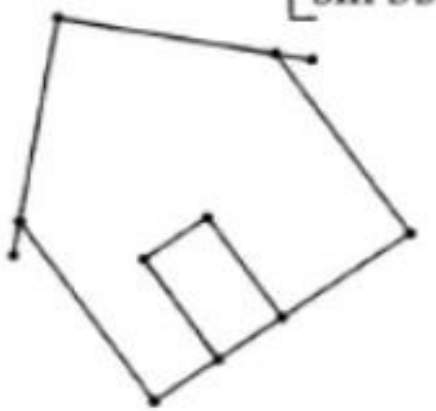


Enlarge your Dataset

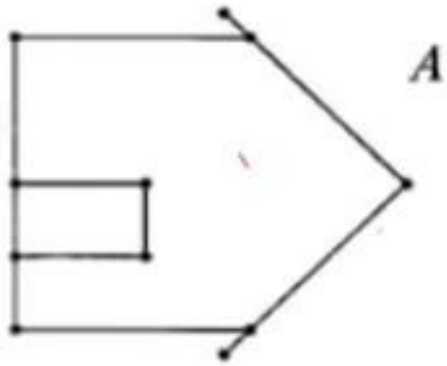
Compute
r
graphics



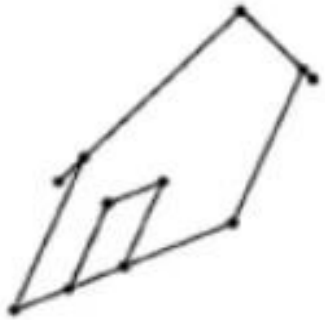
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} \cos 35° & -\sin 35° \\ \sin 35° & \cos 35° \end{bmatrix}$$



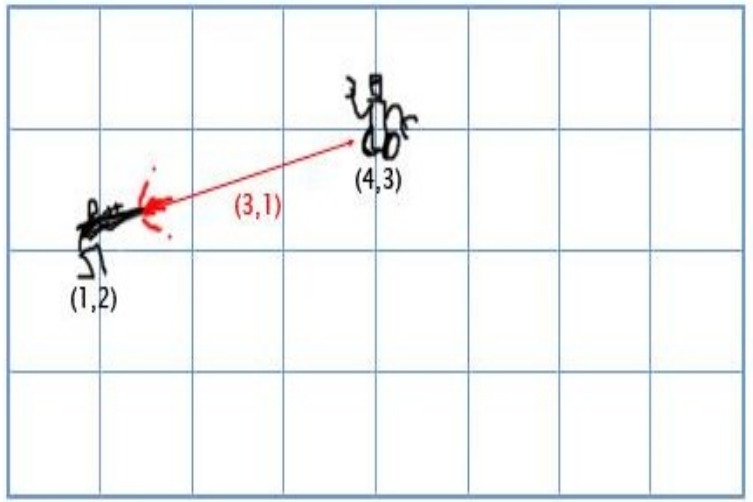
$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

Figure 7.2: Linear transformations of a house drawn by **plot2d**$(A * H)$.

Why Linear Algebra



may be in representing the
position of the player and we can
use it’s
operations to determine the direction
of
that bullet.

Let’s us know more about that

….

So… Linear Algebra is definitely every where

Profiling

- How much do you know about the fundamentals of linear algebra?

Rate yourself from 1 (min knowledge) to 10 OR

Take this quick quiz (Kahoot link)

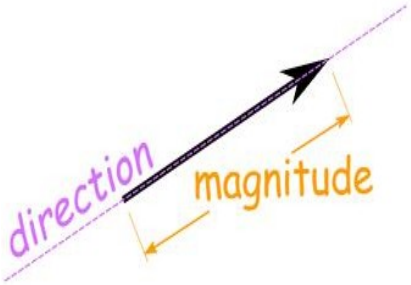
Contents

- **Basics of linear algebra**
  - Vectors, vector space, norms
  - dot product
  - Matrices
  - Operations on matrices

What is a Scalar and What is a vector?

Scala
r
24

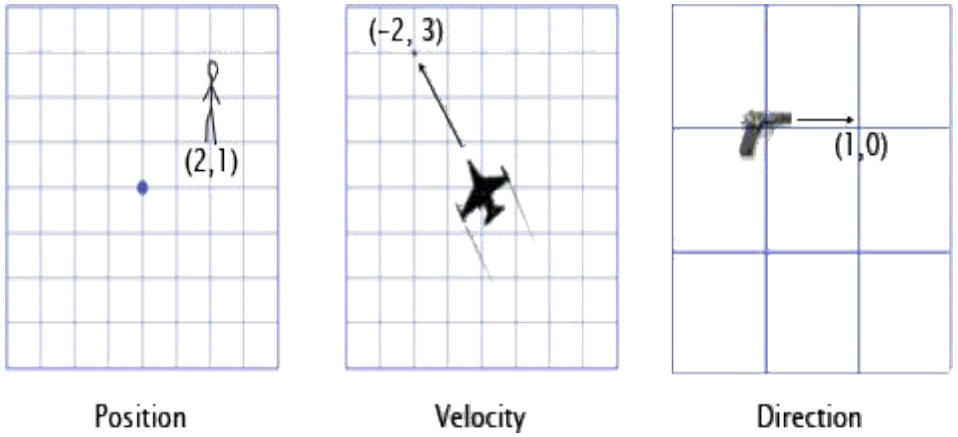
Vector
[1,2,3,4
]



Scalar:

- one-dimensional vector is a scalar.

  - is a quantity that has only magnitude and no direction. Unlike the vector that has direction and magnitude.

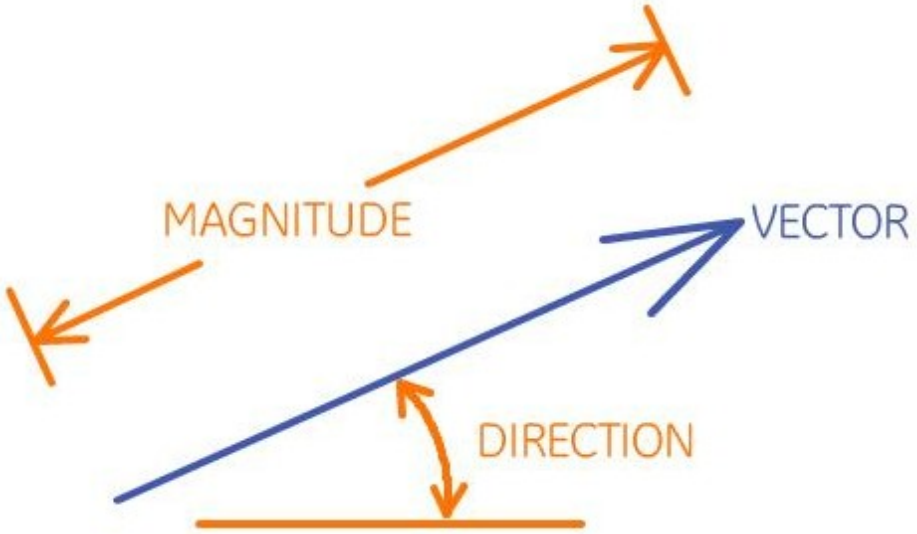


Vector: An array of numbers, either continuous or discrete, but most machine-learning /data science problems deal with fixed-length vectors.

Having only magnitude

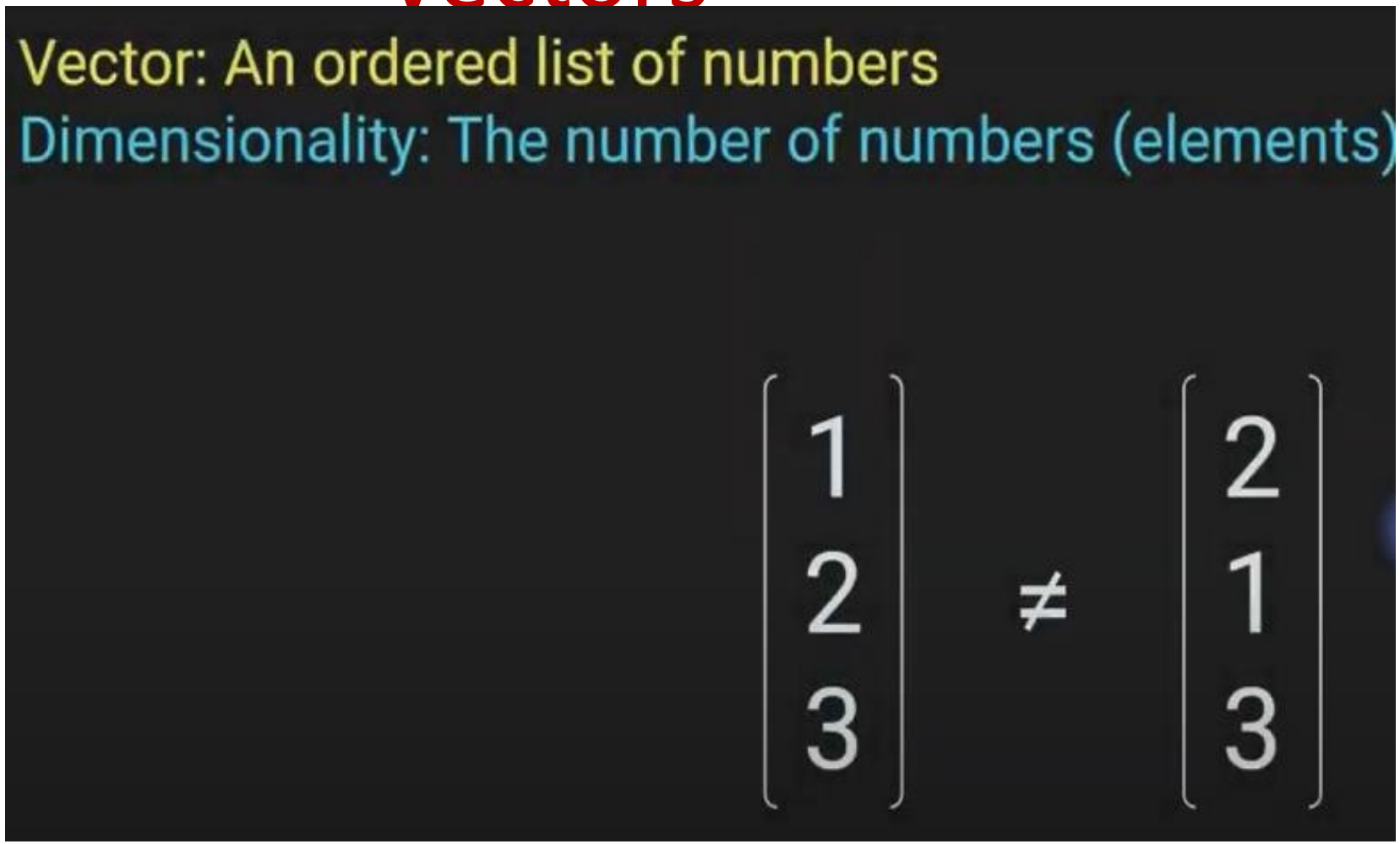
Having both magnitude and direction

direction AND magnitude

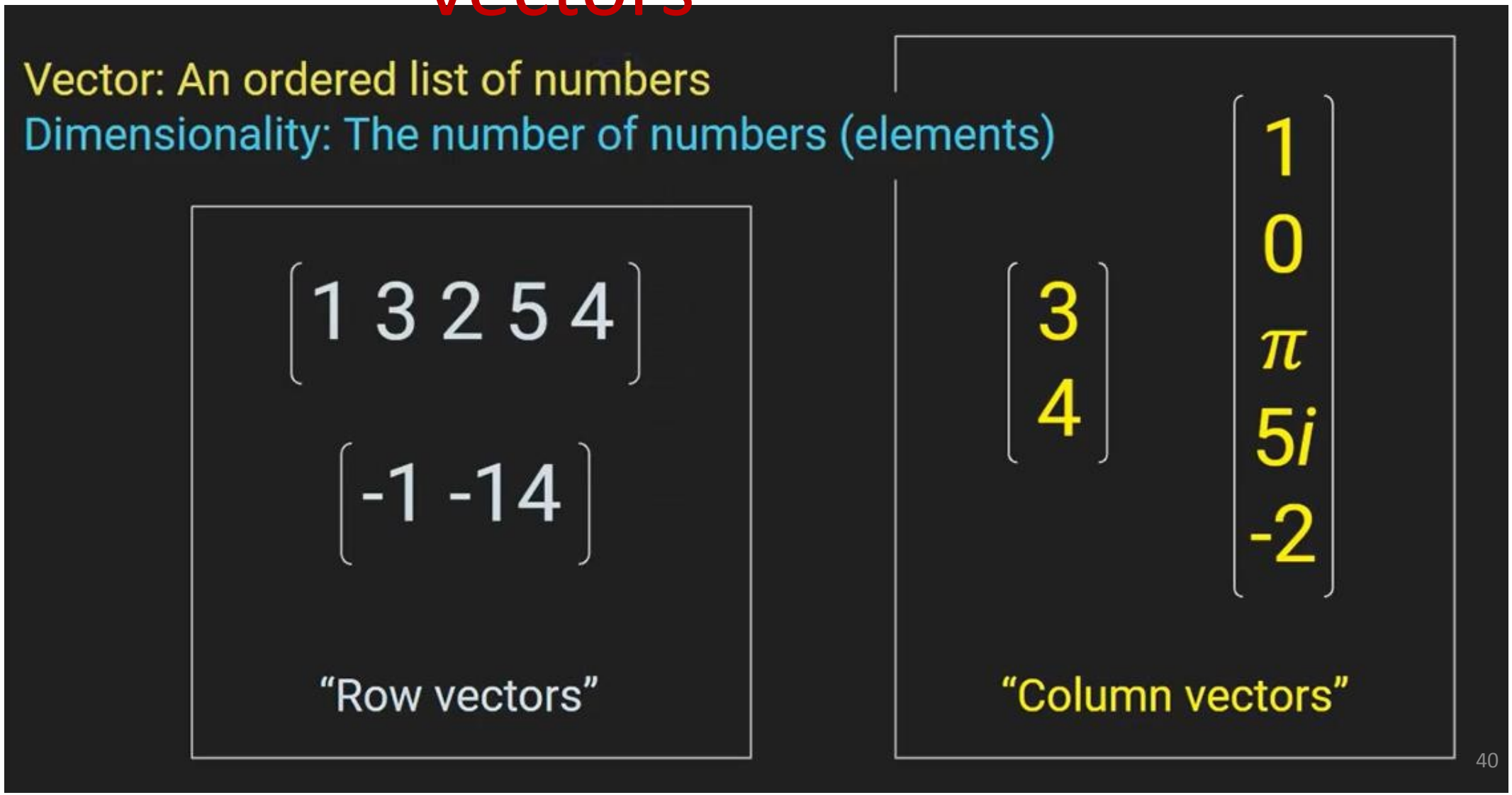


What is a Scalar and What is a vector?

Algebraic vectors



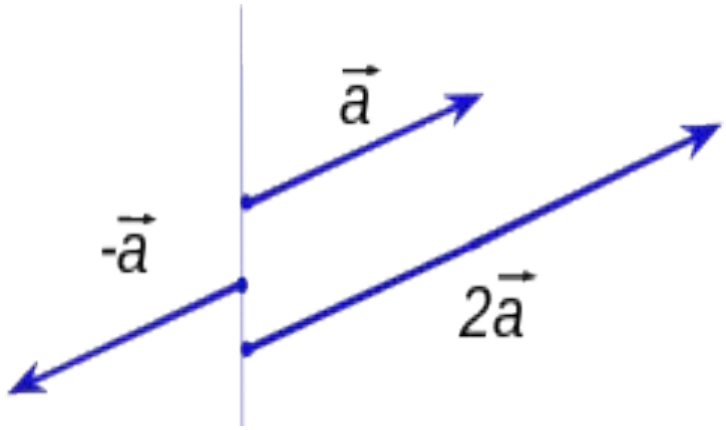
Algebraic vectors



Geometric vectors

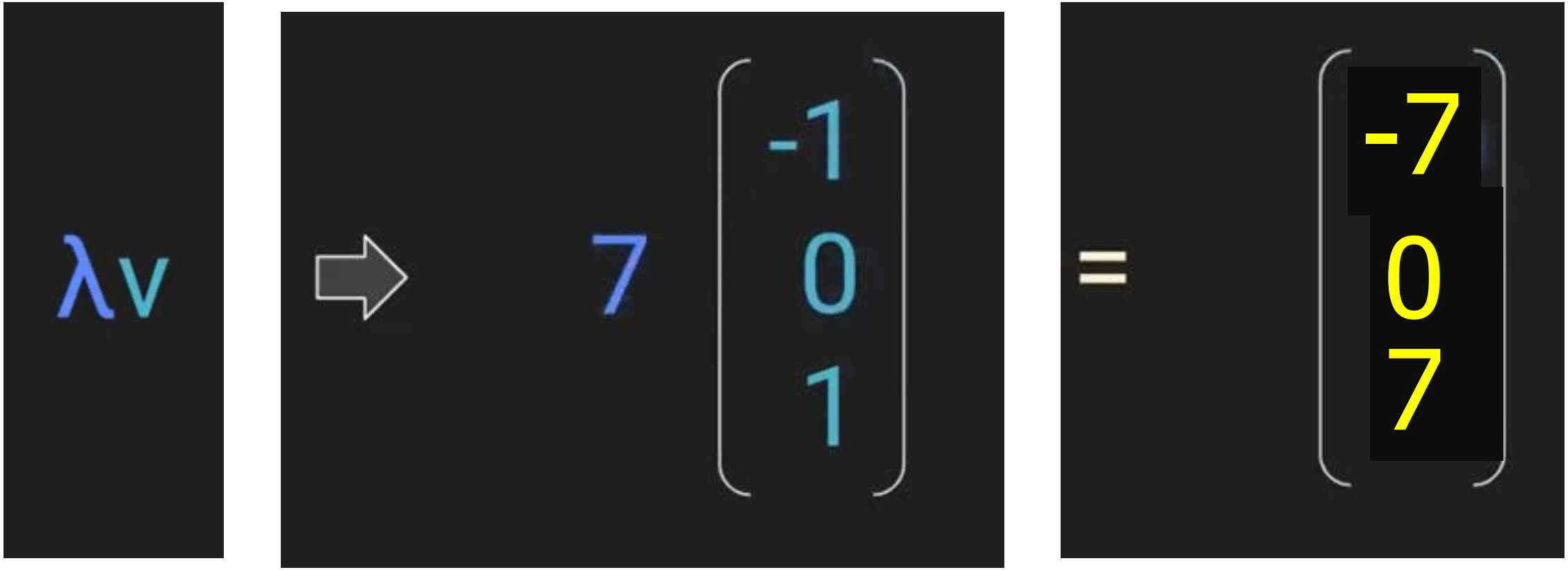
Vector Scaling

We can multiply vectors by scalars that we mentioned to change its scale



Interesting point: They named it like that acts on the space by "scaling" a vector (stretching or contracting or reflecting it).

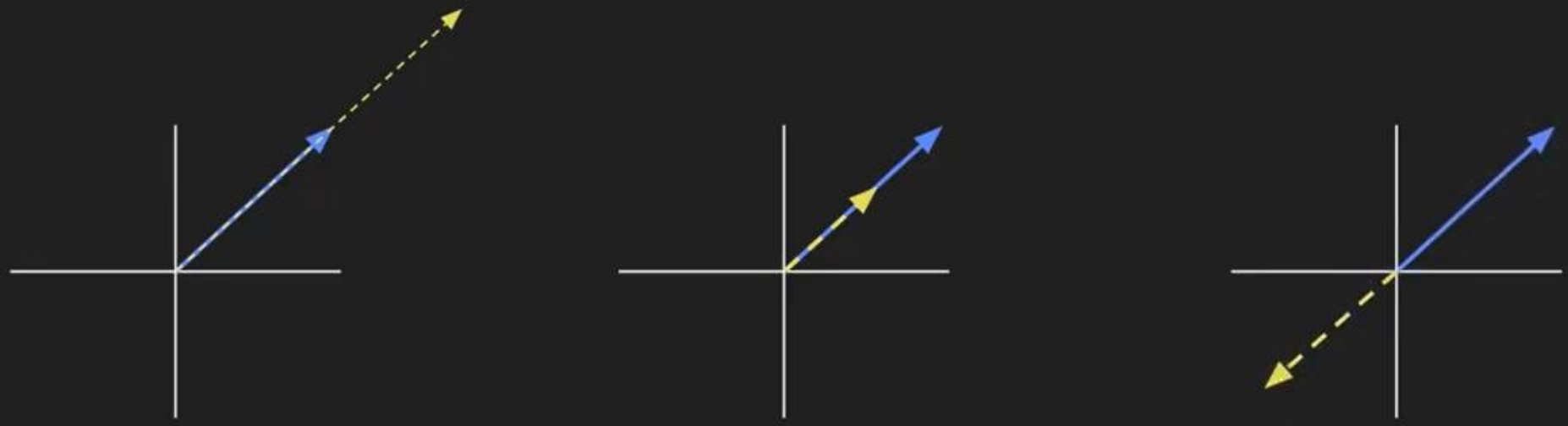
Vector scalar



Vector scalar multiplication:

Geometry

$\mathbf{w} = \lambda \mathbf{v}$



$\lambda > 1$

$\lambda \in (0,1)$

$\lambda < 0$

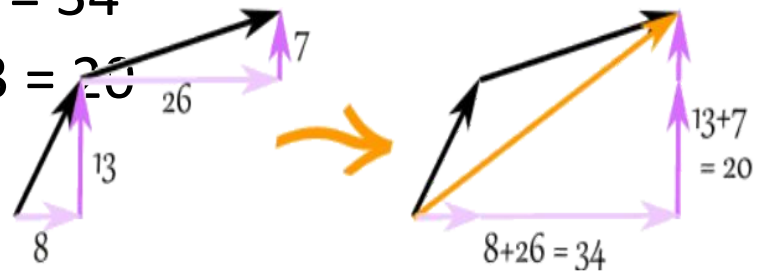
Vector Addition

Here let us add two vectors as in the following image:

The Addition can be done in two ways : Algebraically & graphically

1- Algebraically (by using the components):-

- We have vector (8,13) and vector (26,7)
- Which we can express in terms of x and y
- Adding the **x components** 8+ 26 = 34
- Adding the **y components** 7 + 13 = 20
- So, Our **result** is **vector** (34,20)



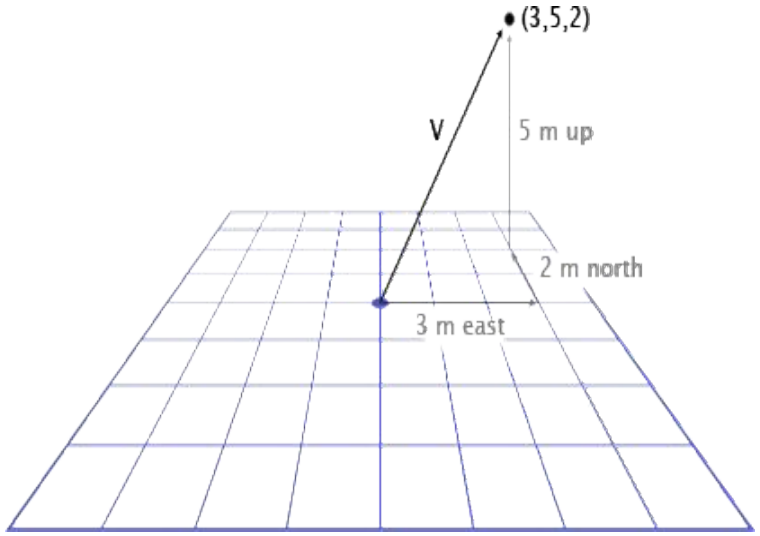
The vector (8, 13) and the vector (26, 7) add up to the vector (34, 20)

2- And Graphically as Shown in the

Vector Addition in 3-dimensional space

Here let us get the displacement “shortest path between two points” of point origin (0,0,0) and point (3,5,2)

- In order to reach this point we will have to go through three main vectors according to our coordinate system:
  first: (3,0,0) : (x,y,z)
  second: (3,0,2)
  third(v): (3,5,2)



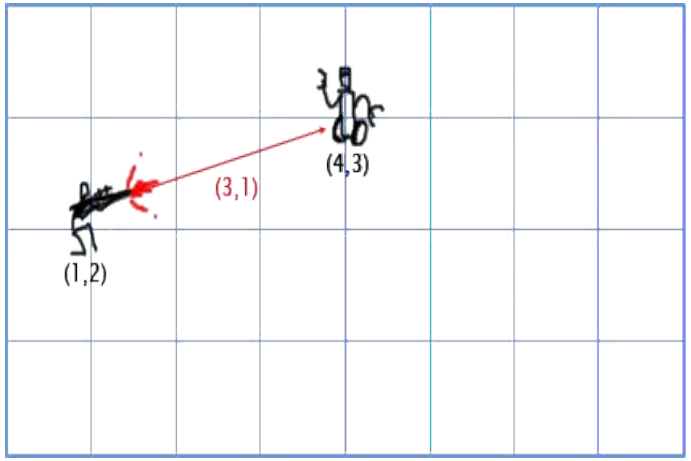
- Or by another way we added
  (3,0,0) + (0,5,0) + (0,0,2) = (3,5,2)

Vector subtraction

Do you remember This Guy ?!

- This guy stands in position which is represented by a vector (1,2) according to our co-ordinate system here.
- His enemy is standing in position (4,3) according to our co-ordinate system
- To know the direction of his bullet direction.
  we will subtract enemyPos – guyPos to get the bullet direction which is

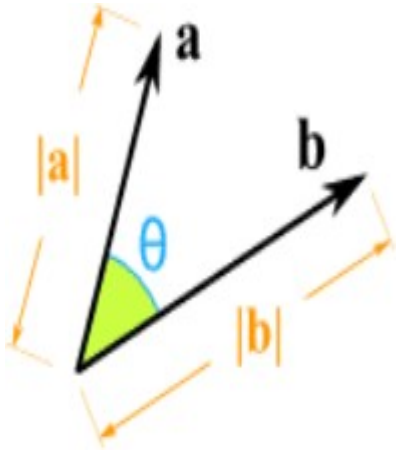
  (4,3) – (1,2) = (3,1)



Vector Multiplication

There are two ways to do multiplications:

- Dot product: The result is Scalar
- cross product: The result is Vector



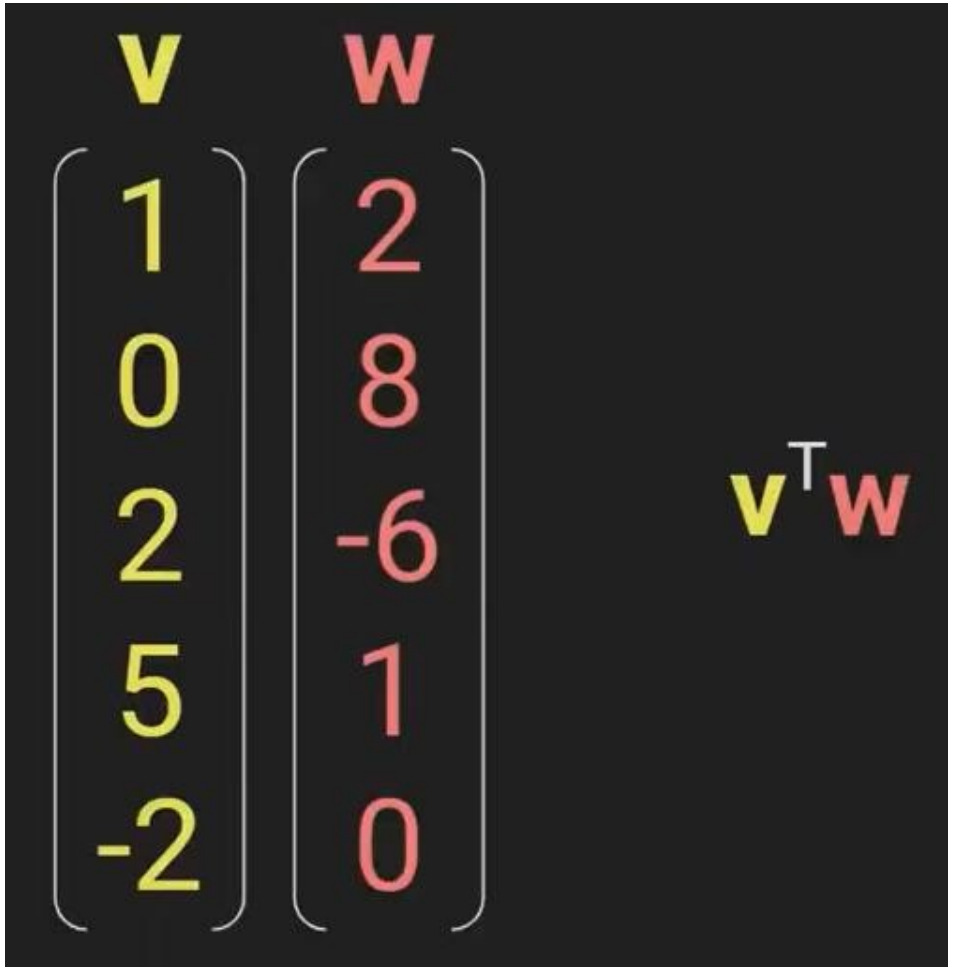
Notations and definition of the dot product

$$\alpha = \mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^{\mathrm{T}} \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$$

Notations and definition of the dot product

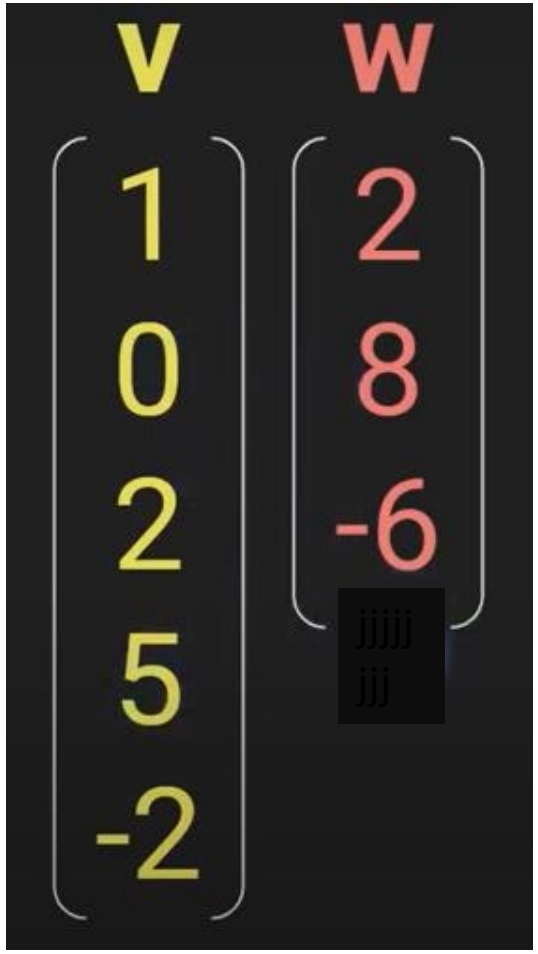
$$\alpha = \mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^{\mathrm{T}} \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$$

Dot product example



$$v^T w = 1*2 + 0*8 + 2*(-6) + 5*1 + (-2)*0$$
$$= 2 - 12 + 5$$
$$= -5$$

Dot product example



$$v^T w = 1*2 + 0*8 + 2*(-6) + 5*? + (-2)*?$$

Vectors : dot/Inner product

Any vector of dimension n can be represented as a matrix $v \in R^{n\times 1}$. Let us denote two n dimensional vectors $v_1 \in R^{n\times 1}$ and $v_2 \in R^{n\times 1}$

$$v_1 = \begin{bmatrix} v_{11} \\ v_{12} \\ . \\ . \\ . \\ v_{1n} \end{bmatrix} \quad v_2 = \begin{bmatrix} v_{21} \\ v_{22} \\ . \\ . \\ . \\ v_{2n} \end{bmatrix}$$

The dot product of two vectors is the sum of the product of corresponding components—i.e., components along the same dimension—and can be expressed as

$$v_1.v_2 = v_1^T v_2 = v_2^T v_1 = v_{11}v_{21} + v_{12}v_{22} + .. + v_{1n}v_{2n} = \sum_{k=1}^{n} v_{1k}v_{2k}$$

Vectors : dot/Inner product

Let us see an example:

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad v_2 = \begin{bmatrix} 3 \\ 5 \\ -1 \end{bmatrix}$$

$$v_1 . v_2 = v_1^T v_2 = 1 \times 3 + 2 \times 5 - 3 \times 1 = 10$$

Vectors : dot/Inner product

Properties of the Dot Product

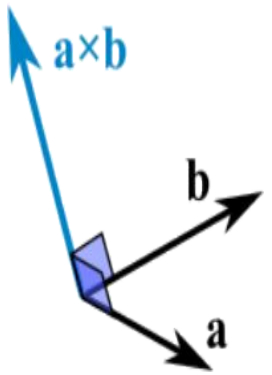
Commutative Property: $\vec{p} \cdot \vec{q} = \vec{q} \cdot \vec{p}$,

Distributive Property: $\vec{p} \cdot (\vec{q} + \vec{r}) = \vec{p} \cdot \vec{q} + \vec{p} \cdot \vec{r}$,

Magnitudes Property: $\vec{p} \cdot \vec{p} = |\vec{p}|^2$,

Associative Property with a scalar K: $(k\vec{p}) \cdot \vec{q} = \vec{p} \cdot (k\vec{q}) = k(\vec{p} \cdot \vec{q})$

Vectors: Cross product



The Cross Product **a** × **b** of two vectors is **another vector** that is at right angles to both.

Let us calculate it:

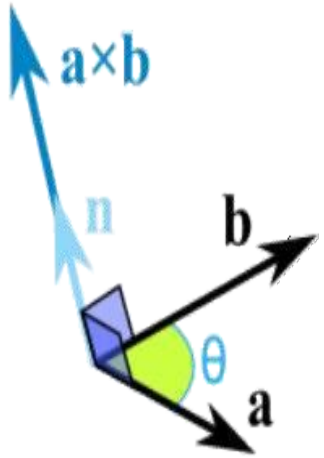
$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}|\ |\mathbf{b}|\ \sin(\theta)\ \mathbf{n}$$

$|\mathbf{a}|$ is the magnitude (length) of vector **a**

$|\mathbf{b}|$ is the magnitude (length) of vector **b**

θ is the angle between **a** and **b**

n is the unit vector at right angles to both **a** and **b**



* Unit vector : a vector with magnitude = 1.

Vectors: Norm of a vector

The norm of a vector is a measure of its magnitude. There are several kinds of such norms. The most familiar is the Euclidean norm, defined next. It is also known as the l^2 norm.

For a vector $x \in R^{n\times 1}$ the l^2 norm is as follows:

$$\|x\|_2 = \left(|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2\right)^{1/2} = (x.x)^{1/2} = \left(x^T x\right)^{1/2}$$

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|$$

Similarly, the l^1 norm is the sum of the absolute values of the vector components.

$$\lim_{p\to\infty} \|x\|_p = \lim_{p\to\infty}\left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

$$= max\left(x_1, x_2, \ldots., x_n\right)$$

I

- In general, the l^p norm of a vector can be defined as follows when $1 < p < \infty$:

prediction = a + b *x [3,2,9,7]
Y actual [6,2,7,7]
E = (4-6) + (2-2) +(9-7) +(7-7) = 0
MSE = 4 + 0+ 4 +0 = 8
MAE = 2 + 0+2+0 = 4

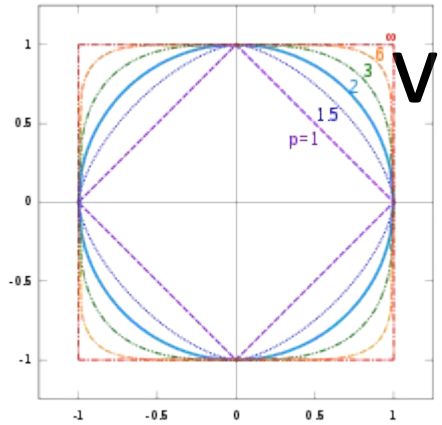
Vectors: Norm of a

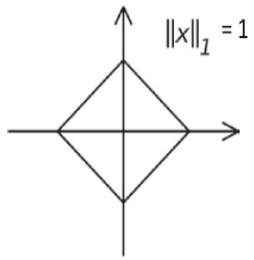
$$\|\mathbf{x}\|_1 = \sum_{i=1}^{N} |x_i| = |x_1| + |x_2| + \ldots + |x_N|$$

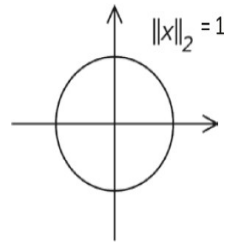
$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{N} |x_i|^2 \right)^{1/2} = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}$$

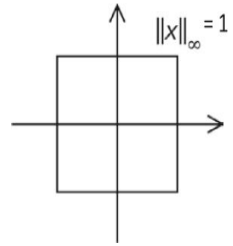
$$L^2 : \|v\|_2 = \left\| \begin{bmatrix} -1 \\ -2 \\ 3 \\ 4 \end{bmatrix} \right\|_2 = \sqrt{\sum_i |\, v_i \,|^2} = \sqrt{|\,-1\,|^2 + |\,-2\,|^2 + |\,3\,|^2 + |\,4\,|^2} = \sqrt{30}$$

Vectors: Norm of a vector







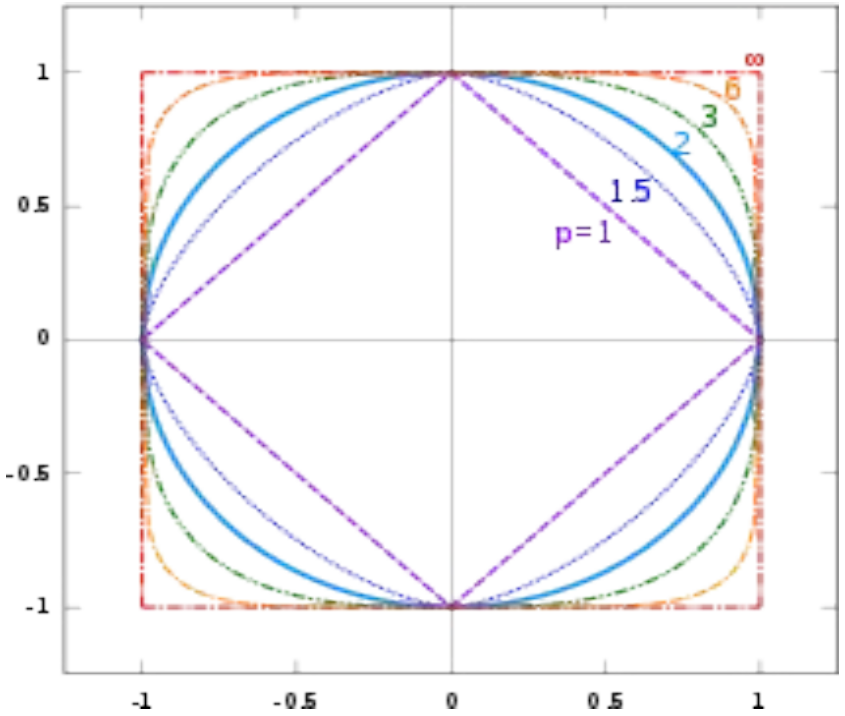


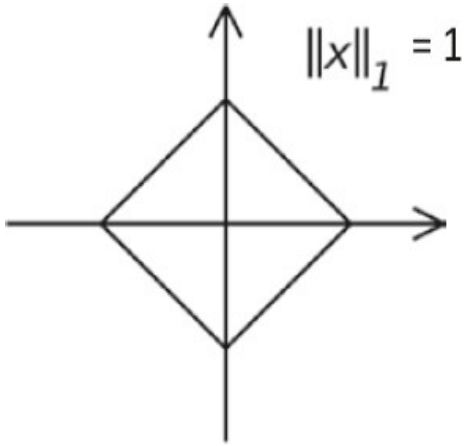
Generally, for machine learning we use both l_2 and l_1 norms for several purposes. For instance, the least square cost function that we use in linear regression is the l_2 norm of the error vector; i.e., the difference between the actual target-value vector and the predicted target-value vector.

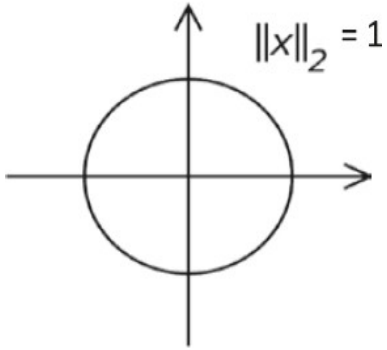
Similarly, very often we would have to use regularization for our model, with the result that the model doesn't fit the training data very well and fails to generalize to new data.

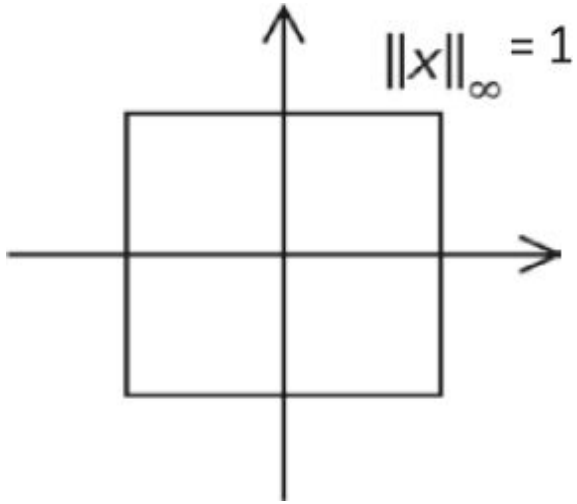
To achieve regularization, we generally add the square of either the l_2 norm or the l_1 norm
of the parameter vector for the model as a penalty in the cost function for the model. When the l_2 norm of the parameter vector is used for regularization, it is generally known as Ridge Regularization, whereas when the l_1 norm is used instead it is known as Lasso Regularization.

Vectors: Norm of a vector









Vectors

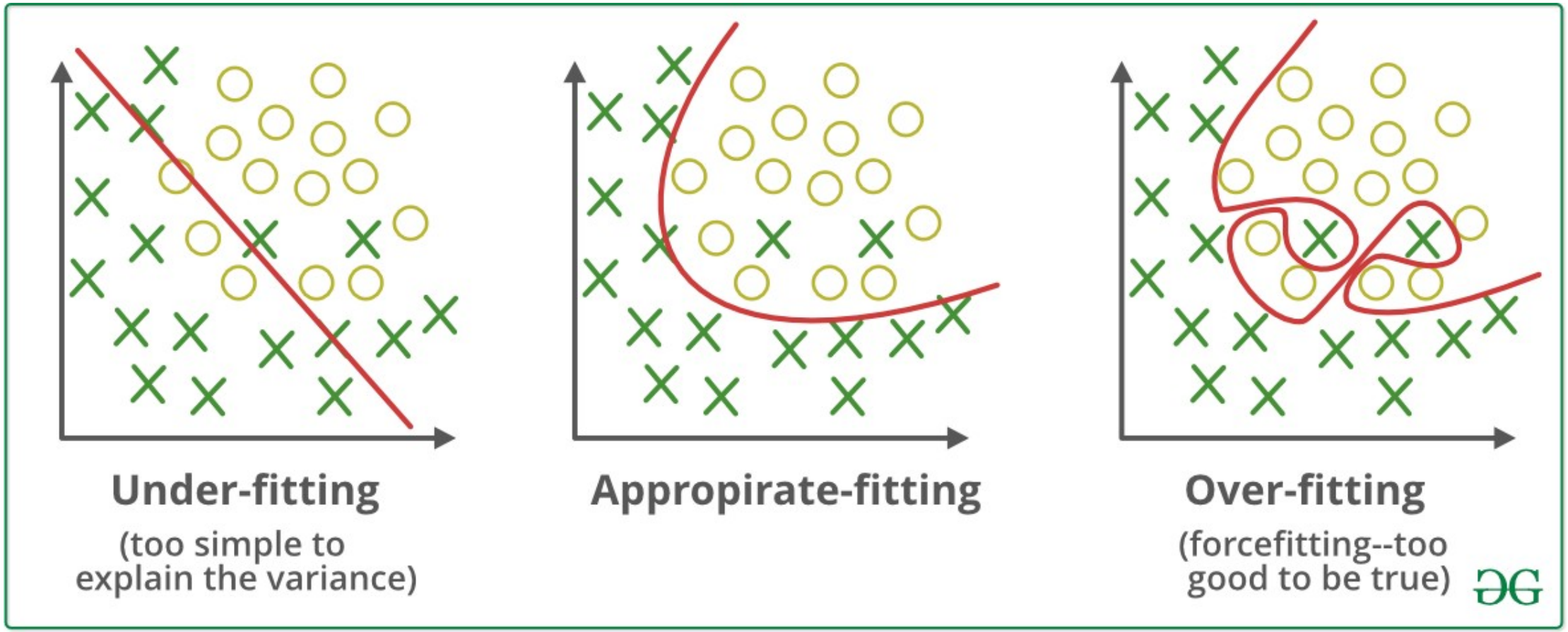
Norm of a vector

- Generally, for machine learning we use both *l*2 and *l*1 norms for several purposes. For instance,

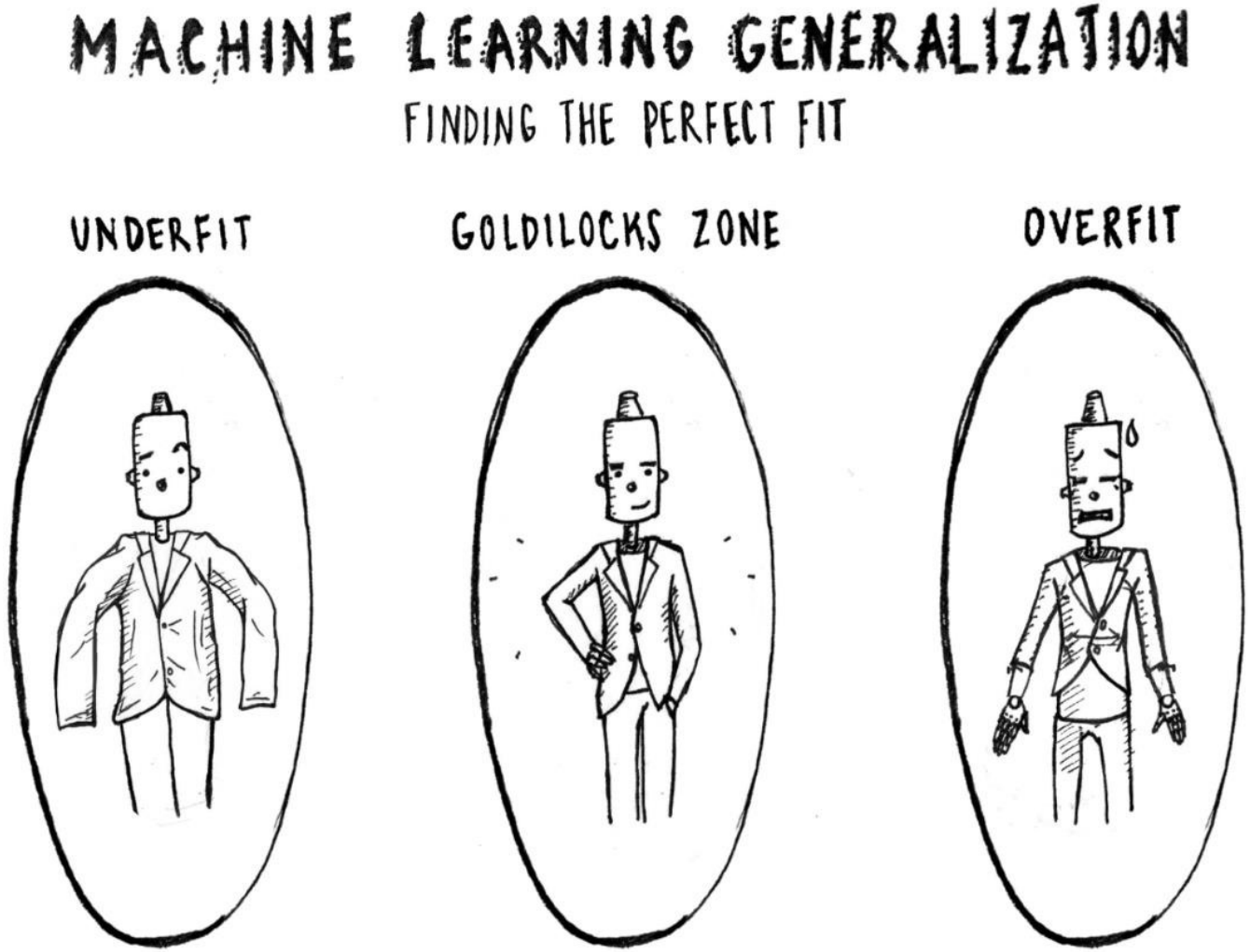
1. the **least square cost function** that we use in linear regression is **the *l*2 norm** of the error vector; i.e., the difference between the actual target-value vector and the predicted target-value vector.

2. Similarly, very often we would have to use **regularization** for our model, with the result that the model doesn't fit the training data very well and fails to generalize to new data. To achieve regularization, we generally add the square of either the *l*2 norm or the *l*1 norm of the parameter vector for the model as a penalty in the cost function for the model. When the *l*2 norm of the parameter vector is used for regularization, it is generally known as Ridge Regularization, whereas when the *l*1 norm is used instead it is known as Lasso Regularization.

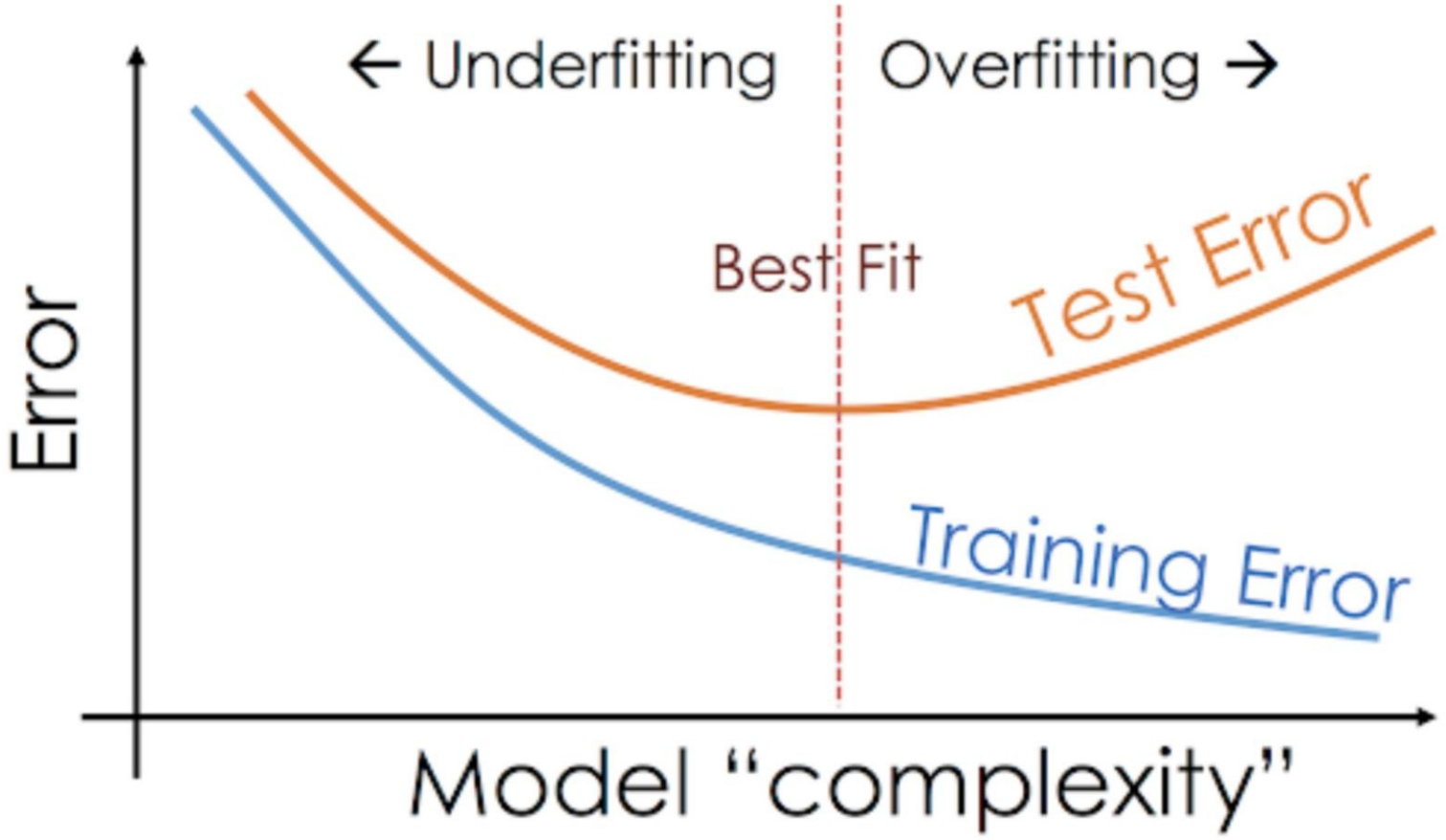
Overfitting, underfitting, best fitting

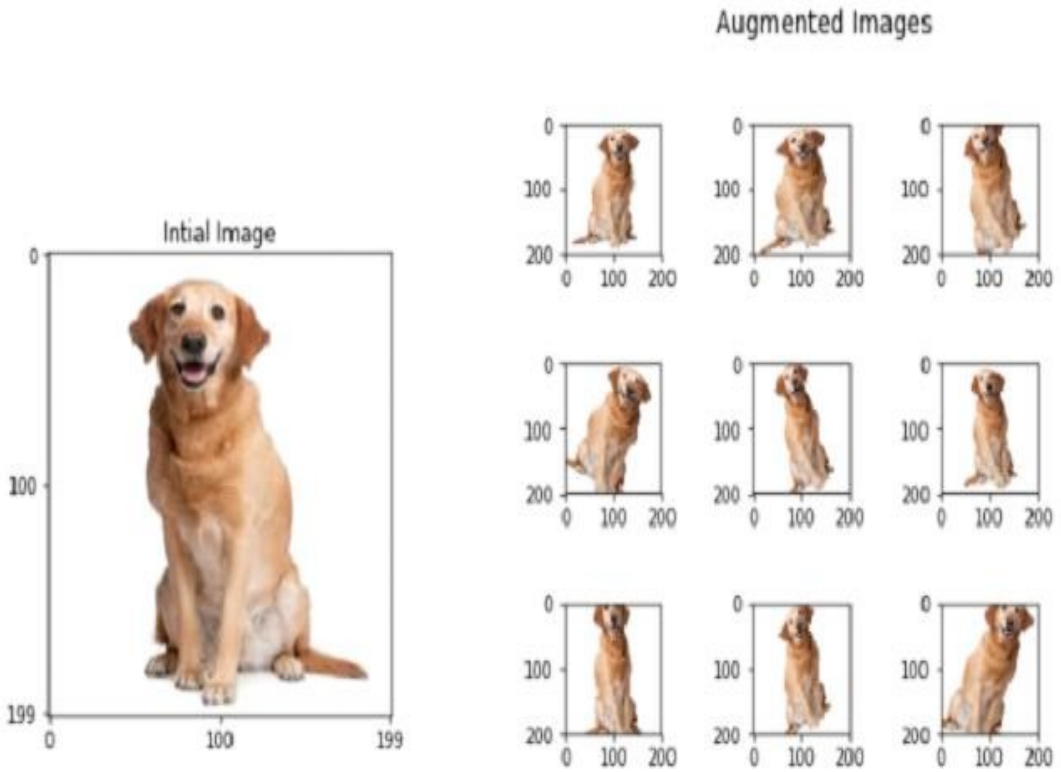


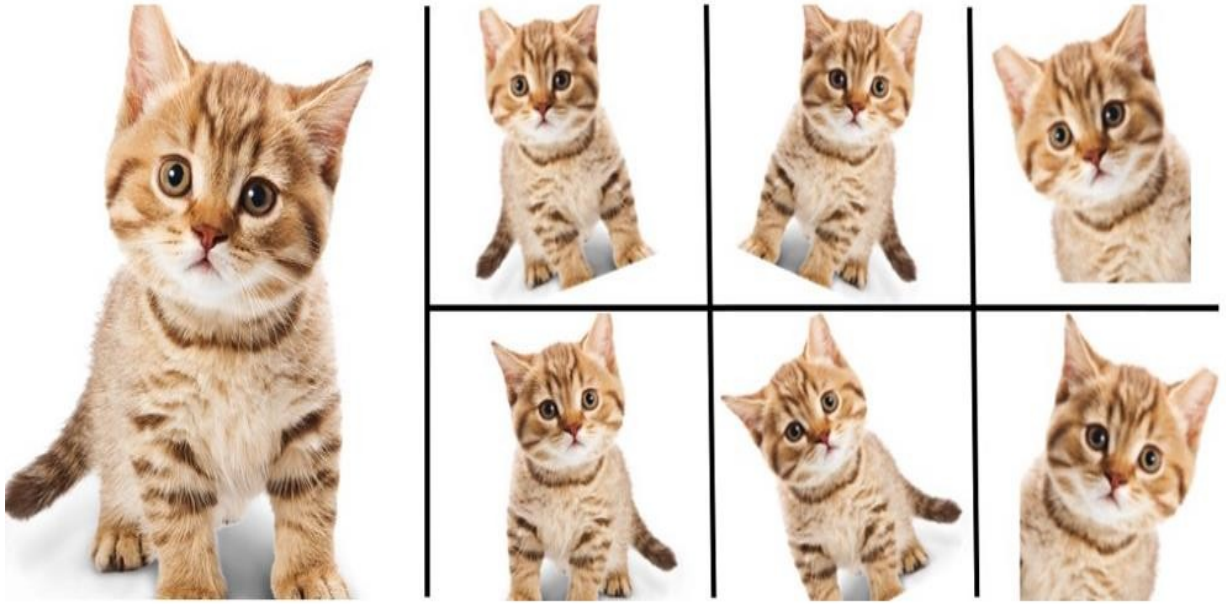
Overfitting, underfitting , best fitting



Overfitting, underfitting, best fitting



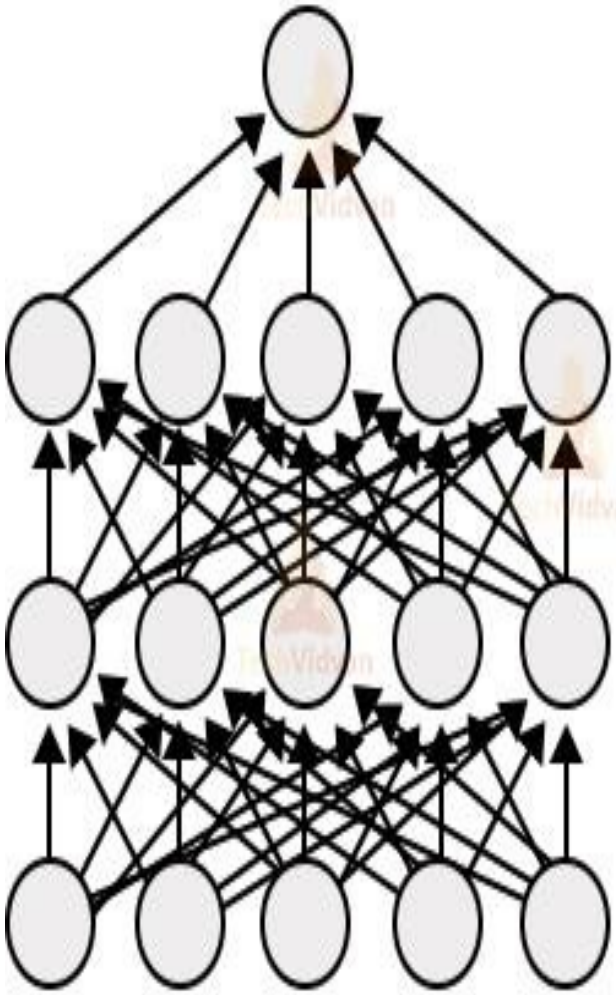




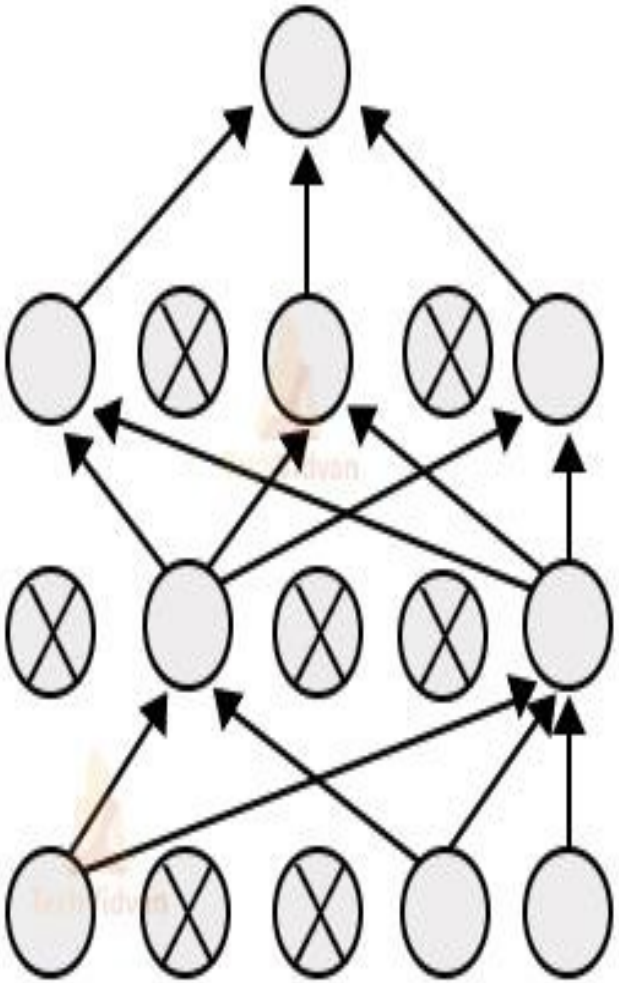
Enlarge your Dataset

To solve overfitting

- 1- Data augmentation



(a) Standard Neural Net



(b) After applying dropout

To solve overfittin g

- **1- Drop out layers**

To solve overfitting

L1 and L2 regularization

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

Matrices

Types of a matrices: **Identity Matrix**

An **Identity Matrix** has **1**s on the main diagonal and **0**s everywhere else:

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

- It is square (same number of rows as columns)
- It can be large or small (2×2, 100×100, ... whatever)
- Its symbol is the capital letter **I**

Matrices

Types of a matrices: **Diagonal Matrix**

$$I = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A diagonal matrix has zero anywhere not on the main diagonal:

Types of a matrices: **Scalar Matrix**

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

A scalar matrix has all main diagonal entries the same, with zero everywhere else:

Types of a matrices: **Triangular Matrix**

Lower triangular is when all entries above the main diagonal are zero:

$$I = \begin{bmatrix} 5 & 0 & 0 \\ 2 & 1 & 0 \\ 7 & 6 & -3 \end{bmatrix}$$

Upper triangular is when all entries below the main diagonal are zero:

$$I = \begin{bmatrix} 2 & -2 & 7 \\ 0 & 4 & 11 \\ 0 & 0 & 5 \end{bmatrix}$$

Matrices

Types of a matrices: **Symmetric**

In a Symmetric matrix matching entries either side of the main diagonal are **equal**, like this:

$$I = \begin{bmatrix} 3 & 2 & 11 & 5 \\ 2 & 4 & -1 & 6 \\ 11 & -1 & 0 & 7 \\ 5 & 6 & 7 & 9 \end{bmatrix}$$

It must be square, and is equal to its own transpose $A = A^T$

Operations on matrices

- Most deep-learning computational activities are done through basic matrix operations, such as multiplication, addition, subtraction, transposition, and so forth.
- A matrix A of m rows and n columns can be considered a matrix that contains n number of column vectors of dimension m stacked side-by-side. We represent the matrix as $A_{m \times n} \in R^{m \times n}$

Matrices

Addition of Two Matrices

The addition of two matrices *A* and *B* implies their element-wise addition. We can only add two matrices, provided their dimensions match. If *C* is the sum of matrices *A* and *B*, then

$$c_{ij} = a_{ij} + b_{ij} \quad \forall\, i \in \{1,2,..m\}, \forall\, j \in \{1,2,..n\}$$

$$where\, a_{ij} \in A, b_{ij} \in B, c_{ij} \in C$$

Exampl
e

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \text{ then } A + B = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Matrices

Subtraction of Two Matrices

The subtraction of two matrices A and B implies their element-wise subtraction. We can only subtract two matrices provided their dimensions match.

If C is the matrix representing A - B, then

$$c_{ij} = a_{ij} - b_{ij} \quad \forall\, i \in \{1,2,..m\}, \forall\, j \in \{1,2,..n\}$$

$$where\, a_{ij} \in A, b_{ij} \in B, c_{ij} \in C$$

Example

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \text{ then } A - B = \begin{bmatrix} 1-5 & 2-6 \\ 3-7 & 4-8 \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

Matrices

Product of Two Matrices

For two matrices $A \in R^{m\times n}$ and $B \in R^{p\times q}$ to be multipliable, *n* should be equal to *p*. The resulting matrix is $A \in R^{m\times n}$. The elements of *C* can be expressed as $C \in R^{m\times q}$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \forall i \in \{1,2,..m\}, \forall j \in \{1,2,..q\}$$

Example

The matrix multiplication of the two matrices $\mathrm{A}, B \in R^{2\times 2}$ can be computed as seen here:

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

$$c_{11} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = 1\times 5 + 2\times 7 = 19 \quad c_{12} = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ 8 \end{bmatrix} = 1\times 6 + 2\times 8 = 22$$

$$c_{21} = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = 3\times 5 + 4\times 7 = 43 \quad c_{22} = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 6 \\ 8 \end{bmatrix} = 3\times 6 + 4\times 8 = 50$$

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Matrices

Transpose of a Matrix

The transpose of a matrix $A \in R^{m \times n}$ is generally represented by A^T $\in R^{n \times m}$ and is obtained by transposing the column vectors as row vectors.

$$a'_{ji} = a_{ij} \quad \forall i \in \{1,2,..m\}, \forall j \in \{1,2,..n\}$$

$$where \ a'_{ji} \in A^T \ and \ a_{ij} \in A$$

Exampl
e

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ then } A^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Matrices

Example:

The transpose of the product of two matrices *A* and *B* is the product of the transposes of matrices *A* and *B* in the reverse order; i.e., $(AB)^T = B^T A^T$

For example, if we take two matrices $A = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$ and $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$,

$$(AB) = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 95 & 132 \\ 301 & 400 \end{bmatrix} \text{ and hence } (AB)^T = \begin{bmatrix} 95 & 301 \\ 132 & 400 \end{bmatrix}$$

$$\text{Now, } A^T = \begin{bmatrix} 19 & 43 \\ 22 & 50 \end{bmatrix} \text{ and } B^T = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix} \qquad B^T A^T = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}\begin{bmatrix} 19 & 43 \\ 22 & 50 \end{bmatrix} = \begin{bmatrix} 95 & 301 \\ 132 & 400 \end{bmatrix}$$

Hence, the equality $(AB)^T = B^T A^T$ holds.

Introduction to numpy

Quick review on numpy library and the most important array commands for vector and matrix operations

To be added later

Introduction to NumPy

NumPy Library

- The fundamental package for scientific computing with Python
- NumPy v1.20.0 (https://numpy.org/)
- NumPy advantages
  - ✔ Powerful N-dimensional arrays
  - ✔ Numerical computing tools (NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more).
  - ✔ Interoperable (NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries).
  - ✔ Open source
  - ✔ Easy to use
  - ✔ Performant (The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.)

NumPy Library

```
conda install numpy
```

```
pip install numpy
```

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called *axes*.

NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further