NumPy Stack

Complementary technologies enabling each other

Mathematics in Code

Prerequisites to Machine Learning

Expectation from you…

- **Write Code.. Do not Copy Paste… Practice & Practice**
- **Refer Linear Algebra & Probability**
- **We will discuss everything to get you started..**

If you can’t implement it, then you don’t understand it…. Simple..

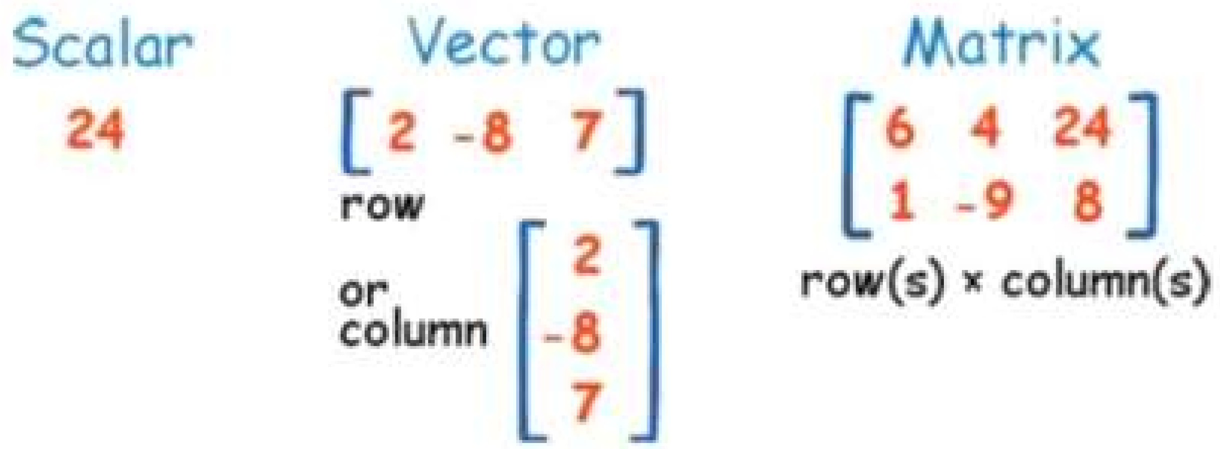
NumPy Introduction

1) Low-level: what does it do?
2) High-level: why do we need it?

- The core library
- Central object: The Numpy Array
- Be familiar with Python Lists
- One sentence summary: "Linear algebra and a bit of probability"

Vectors & Metrices

- In Linear Algebra, convention to treat vectors as 2-D
- But not in Numpy! Vectors will be 1-D (most of the time)



Dot Product / Vector Inner Product

$$a \cdot b = a^T b = \sum_{d=1}^{D} a_d b_d$$

- Multiplies element wise and sums the product
- Both input vector must have the same shape

Matrix Multiplication : C=AB

- “Generalized” dot product

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

- Number of columns in A should be equal to number of rows in B
- Inner dimensions must match. A= m x n .. B= n x p.. Then c = m x p

Element – Wise Product

- Not so common in Linear Algebra, very common in ML

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}\,b_{11} & a_{12}\,b_{12} & a_{13}\,b_{13} \\ a_{21}\,b_{21} & a_{22}\,b_{22} & a_{23}\,b_{23} \\ a_{31}\,b_{31} & a_{32}\,b_{32} & a_{33}\,b_{33} \end{bmatrix}$$

- Required both matrix to have the same shape
- Output also has the same shape

And lot more...

- Linear systems: $Ax = b$
- Inverse: A^{-1}
- Determinant: $|A|$
- Choosing random numbers (e.g. Uniform, Gaussian)

Applications

Question

The admission fee at a small fair is $1.50 for children and $4.00 for adults. On a certain day, 2200 people enter the fair, and $5050 is collected. How many children and how many adults attended?

2 equations, 2 unknowns

$$x_1 + x_2 = 2200$$
$$1.5x_1 + 4x_2 = 5050$$

Linear System in Matrix Form

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, A = \begin{pmatrix} 1 & 1 \\ 1.5 & 4 \end{pmatrix}, b = \begin{pmatrix} 2200 \\ 5050 \end{pmatrix}$$

$$Ax = b \Leftrightarrow x = A^{-1}b$$

Don’t do that literally!

- The “inverse” is slower and less accurate
- There are better algorithms to **solve** linear systems

```
x = np.linalg.solve(A, b) # yes

x = np.linalg.inv(A).dot(b) # no
```

Data Matrix

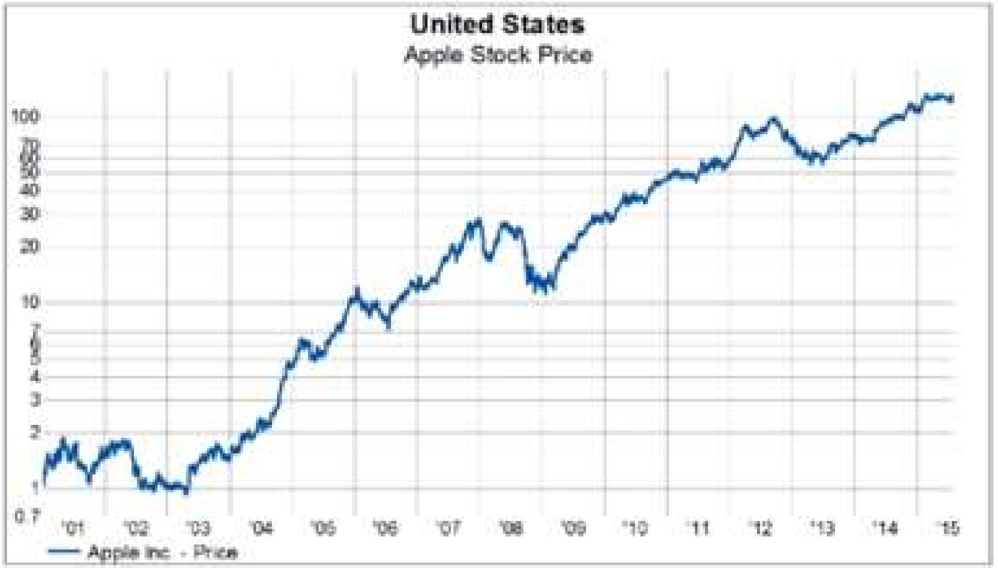
	Openness	Conscientiousness	Extroversion	Agreeableness	Neuroticism
Alice	8	5	7	7	6
Bob	3	4	2	2	2
Carol	4	2	7	5	2

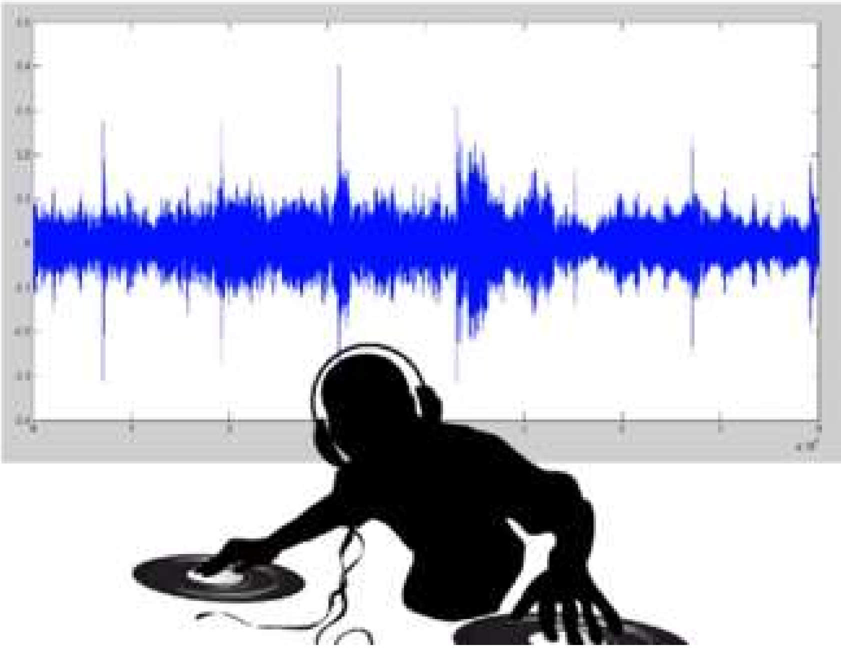
Exercise

- Earlier, we did a speed test to compare list vs. array dot product
- Do a similar speed test, but for matrix multiplication
- You will have to implement matrix multiplication for lists

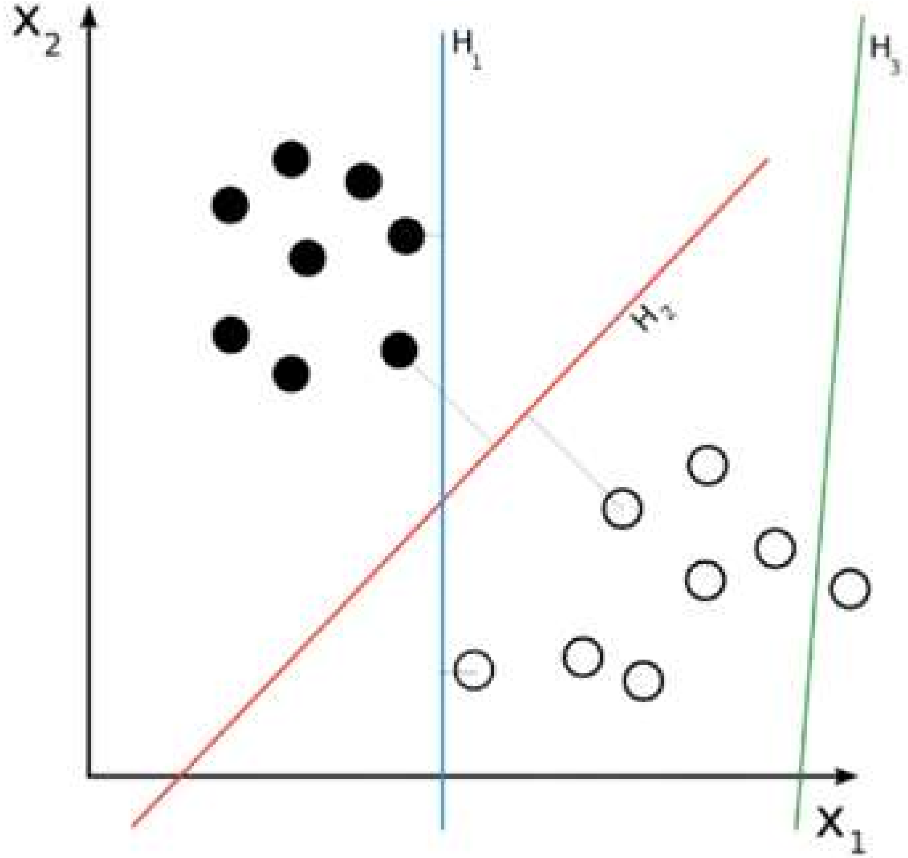
Matplotlib

Line Chart

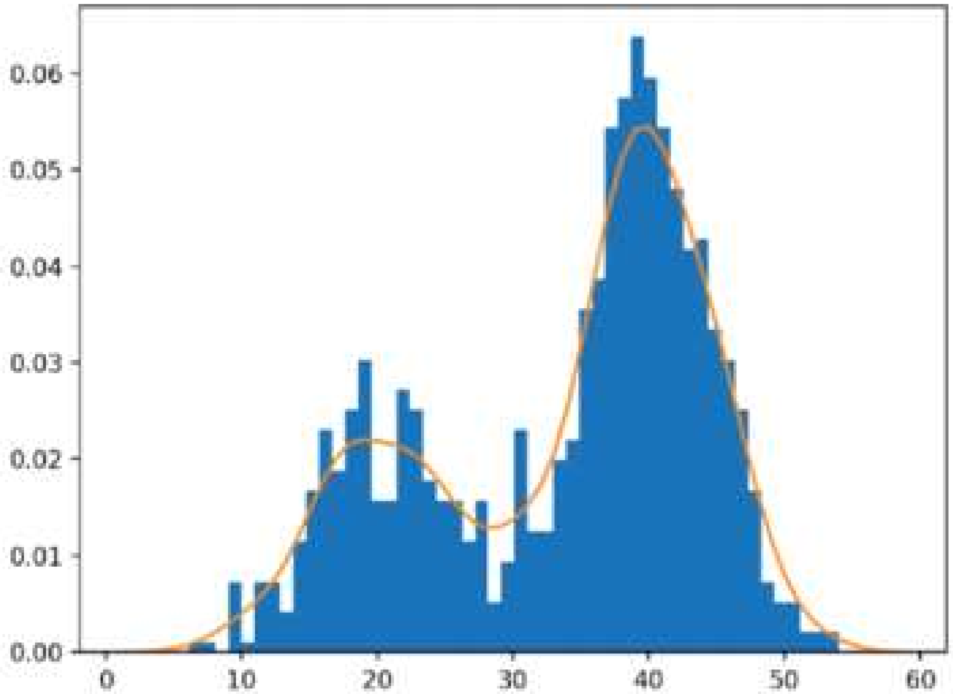

United States
Apple Stock Price
100
70
60
50
40
30
20
10
7
6
5
4
3
2
1
0.7
'01
'02
'03
'04
'05
'06
'07
'08
'09
'10
'11
'12
'13
'14
'15
Apple Inc. - Price



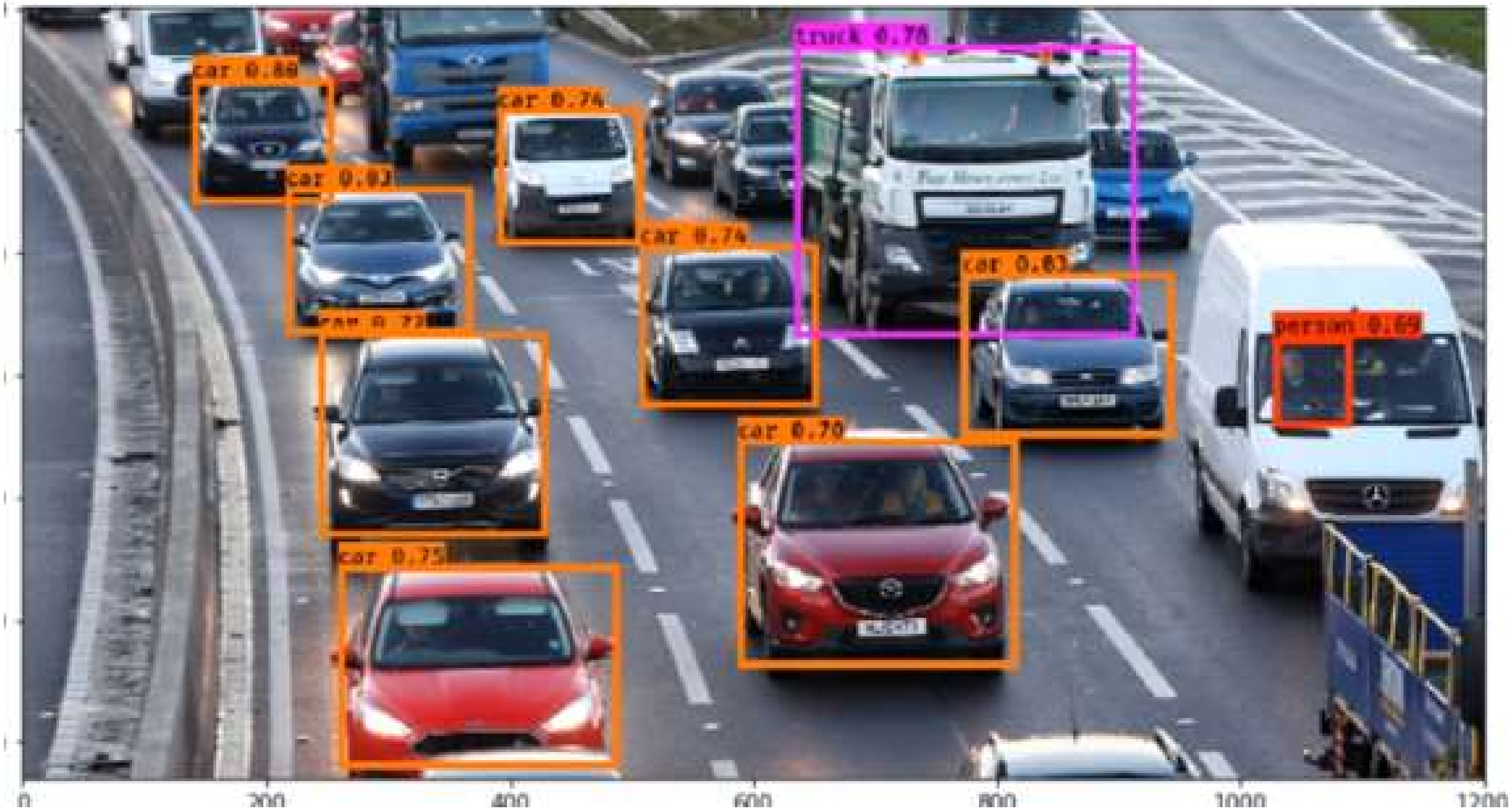
Scatter Plot



Histogram



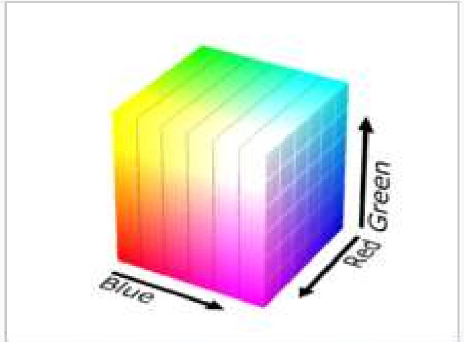
Image



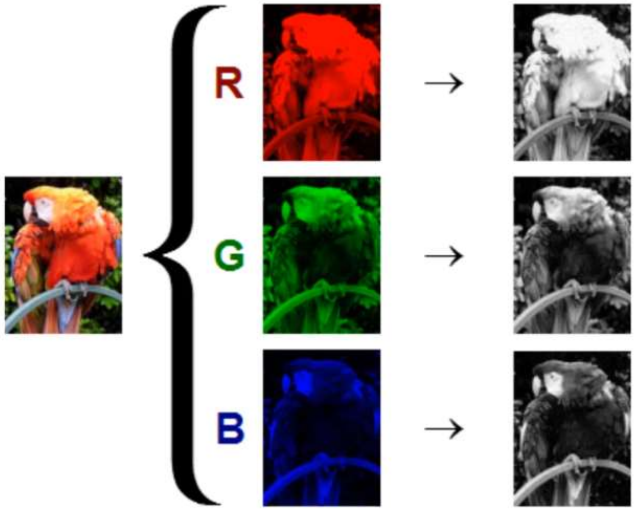
car 0.80
car 0.74
truck 0.78
car 0.83
car 0.74
car 0.83
person 0.69
car 0.70
car 0.75
0
200
400
600
800
1000
1200

RGB Color Model

Colour Model
- RGB – Additive in nature – Used in devices – TV, Computer – Primary Colour
- CMY - Subtractive in nature – Used in ink, paint, dyes etc – Secondary Colour
- HSV and HSL – Derivative of RGB Model - Computer Graphics



The RGB color model mapped to a cube. The horizontal x-axis as red values increasing to the left, y-axis as blue increasing to the lower right and the vertical z-axis as green increasing towards the top. The origin, black is the vertex hidden from view.



- Height - x
- Width - y
- Colour Channel - 3

- **Can be represented in a 3D Matrix (h x w x 3)**

- Read, write, and manipulate data
- There is *lots* we will not discuss
- Read/write CSV
- DataFrames (very familiar if you come from R / stats)
- Selecting rows and columns (unintuitive at first)
- apply() function
- Plotting

Pandas

- Adds functionality for statistics, signal processing, computer vision
- Standard normal → Multivariate normal
- PDF, CDF
- Convolution
    - (used in deep learning, computer vision, signal processing, and even statistics!)

SciPy

Exercise

- Implement edge detection
- Step 1: convolve H_x and H_y with grayscale image to obtain G_x and G_y
- Step 2: take G = sqrt(G_x^2 + G_y^2) (this is the edge-detected output)

$$H_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, H_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

$$G_x = H_x * A, G_y = H_y * A$$

$$G = \sqrt{G_x^2 + G_y^2}$$

* means convolution
A is the image