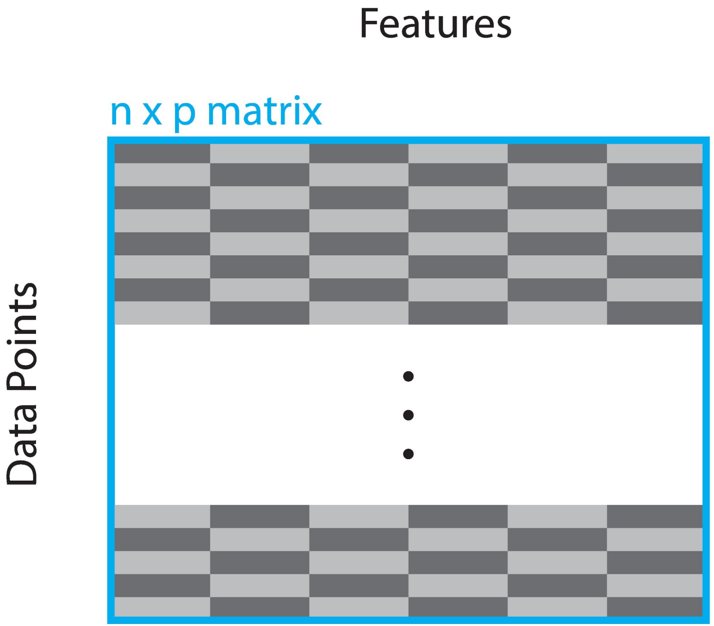
# Linear Algebra

# Basic data structure

Feature matrix:

**X** =




CAMBRIDGE SPARK

# Why does Linear Algebra matter?

- So much in Data Science boils down to **matrix manipulation**

- Definitions:
    - Vector *or* 1D Array - An ordered collection of *N* items
    - Matrix *or* 2D Array - An indexed collection of *N* x *D*
    - Data point - A single data item, may be a vector, typically a **row** of a matrix
    - Feature - Something that was being measured from each data point, typically a **column** of a matrix

# Example of a data point

| | Age | Workclass | fnlwgt | Education | Education-Num | Martial Status | Occupation | Relationship | Race | Sex | Capital Gain | Capital Loss | Hours per week | Country | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | 40 | United-States | <=50K |
| **1** | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 13 | United-States | <=50K |
| **2** | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-States | <=50K |
| **3** | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | 40 | United-States | <=50K |
| **4** | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cuba | <=50K |

CAMBRIDGE SPARK

# Example of a feature

| | Age | Workclass | fnlwgt | Education | Education-Num | Martial Status | Occupation | Relationship | Race | Sex | Capital Gain | Capital Loss | Hours per week | Country | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | 40 | United-States | <=50K |
| **1** | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | 13 | United-States | <=50K |
| **2** | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-States | <=50K |
| **3** | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | 40 | United-States | <=50K |
| **4** | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | 40 | Cuba | <=50K |

CAMBRIDGE SPARK

# Mathematical vs. compsci indexing and matrices

- Typically you'll see 1-based indexing (python is 0-based)
- $\boldsymbol{X}_{23}$ is equivalent to `X[1, 2]`
- Unfortunately context and convenience can sometimes change 'standards' e.g. whether a vector is considered a column or row

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1D} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \dots & x_{ND} \end{bmatrix}$$

$$\boldsymbol{X}_{23} = x_{23}$$

$$\boldsymbol{x}_3 = (x_{31}, x_{32}, \dots, x_{3D})^T$$

# Mathematical notation

- Identity matrix $\mathbb{I}$
- Identity vector $\mathbb{1}$
- Matrix transpose
- Sum notation
- Product notation
- Dot product
- Rules of multiplication:
    - Adjacent dimension sizes must match
    - e.g. if **X** - *N* x *D*, then to do **X Y**, Y must be…
    - **Y** must be *D* x *P*, the resulting matrix size…
    - ...will be *N* x *P*
    - Typically denoted with nothing - adjacent symbols

$$\mathbb{I} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\mathbf{1} = (1, 1, \dots, 1)^T$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad \text{i.e. } \boldsymbol{X}^T_{ij} = \boldsymbol{X}_{ji}$$

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

$$\prod_{i=1}^{N} x_i = x_1 \times x_2 \times \ldots \times x_N$$

$$\boldsymbol{x} \cdot \boldsymbol{w} = \sum_{i=1}^{N} x_i w_i = \boldsymbol{x}^T \boldsymbol{w}$$

CAMBRIDGE SPARK

# Matrix multiplication with numpy

```
$  import numpy as np
$  ones = np.ones((4, 1))  # 4 x 1
$  eye = np.eye(3)  # 3 x 3
$  mat = np.arange(12).reshape(3, 4)
$  ones.T @ ones
$  eye @ mat
$  mat @ ones
$  mat @ np.arange(4).reshape(4, 1)
$  np.arange(3) @ mat
$  mat @ ones * eye  # Ordering? Broadcasting?!
```

CAMBRIDGE SPARK

# Relation to sum notation

```
$  np.sum(np.arange(10))
$  np.arange(10) @ np.ones(10)
$  x = np.array([2, 3, 0, 0, 1, 10])
$  w = np.array([1, 1, 2, 0.5, 10, 0.5])
$  x @ w
$  X = np.random.randn(4, 6)
$  y = np.zeros(4)
   for ii in range(4):
       for jj in range(6):
           y[ii] += X[ii, jj] * w[jj]
$  X @ w
```

$$\boldsymbol{x} \cdot \boldsymbol{w} = \sum_{d=1}^{D} x_d w_d$$

if: $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$

$$\Rightarrow y_n = \boldsymbol{x}_n \cdot \boldsymbol{w} = \sum_{d=1}^{D} x_{nd} w_d$$

CAMBRIDGE SPARK

# Summing errors/measuring similarity

## Norm functions

$$\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{y}_{\text{pred}}$$

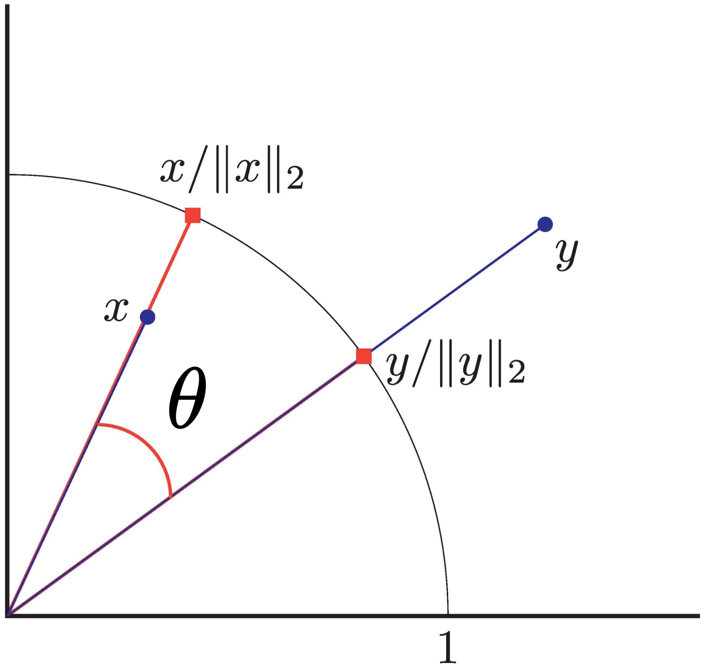
$$\ell_1 \text{ norm: } \|\boldsymbol{r}\|_1 = \sum_{i=1}^{N} |r_i|$$

$$\ell_2 \text{ norm: } \|\boldsymbol{r}\|_2 = \sum_{i=1}^{N} r_i^2$$

```
$  pred1 = np.array([-1, 1])  # predictions for a 2D datapoint
$  pred2 = np.array([-2, 2])
$  pred3 = np.array([.5, .5])
$  y = np.array([0, 0])  # true values
$  predictions = [pred1, pred2, pred3]  # Best pred? By how much?
$  [np.sum(y - pred) for pred in predictions]  # why bad?
$  [np.sum(np.abs(y - pred)) for pred in predictions]  # L1
$  [np.sqrt(np.sum((y - pred)**2)) for pred in predictions]  # L2
```

CAMBRIDGE SPARK

# Visualising the inner product




Cosine Similarity!

$$\cos(\theta) = \frac{x \cdot y}{\|x\|_2 \|y\|_2}$$

How does cos(θ) change with θ?

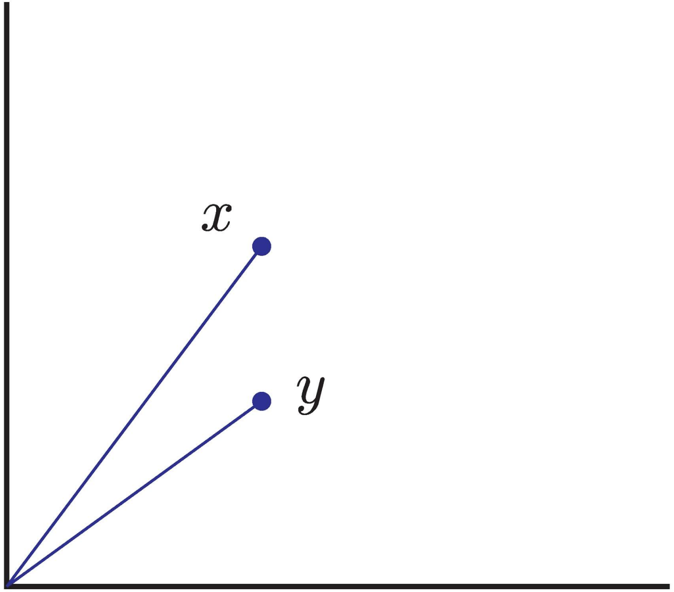
What is the range of cos(θ)?

How can we use this to measure similarity?

CAMBRIDGE SPARK

# Visualising the inner product

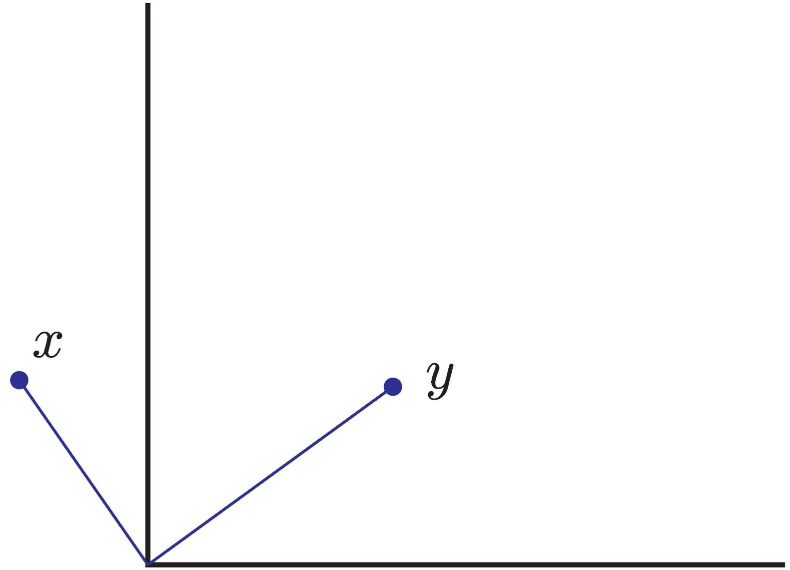
## θ near 0




$$\langle x, y \rangle \approx 1 \times \|x\|_2 \|y\|_2$$

# Visualising the inner product

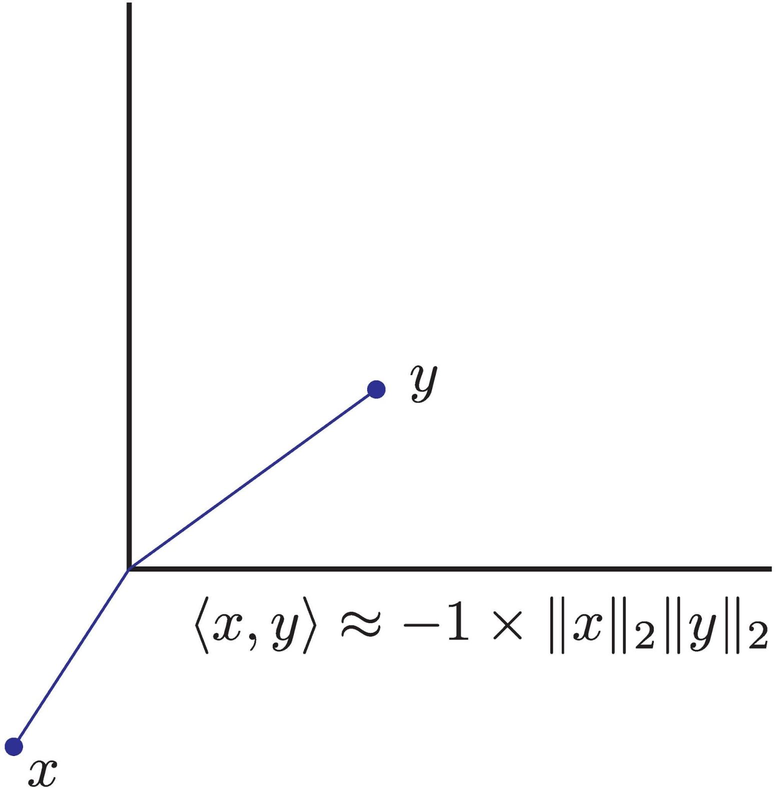
## θ = π/2 (perpendicular)




$$\langle x, y \rangle \approx 0$$

# Visualising the inner product

## θ near π




$$\langle x, y \rangle \approx -1 \times \|x\|_2 \|y\|_2$$

# Summing errors/measuring similarity

## Inner products & cosine similarity

```
$  import matplotlib.pyplot as plt
$  theta = np.linspace(0, 1, 100) * np.pi
$  plt.plot(theta, np.cos(theta))
$  y = np.array([1, 0])
$  y_pred = np.array([0, 1])
$  cosim = lambda x, y: x@y / np.sqrt(x@x*y@y)
$  cosim(y, y_pred)
$  cosim(y, np.array([1, 1]))
$  cosim(y, np.array([0, 0]))  # uh oh...
```

$$
\begin{aligned}
\boldsymbol{r} &= \boldsymbol{y} - \boldsymbol{y}_{\text{pred}} \\
< \boldsymbol{r}, \boldsymbol{r} > &= \boldsymbol{r} \cdot \boldsymbol{r} \\
&= \boldsymbol{r}^T \boldsymbol{r} \\
&= \sum_{n=1}^{N} r_n^2 \\
&= (\|r\|_2)^2
\end{aligned}
$$

$$
\cos(\theta) = \frac{\boldsymbol{y} \cdot \boldsymbol{y}_{\text{pred}}}{\|\boldsymbol{y}\|_2 \|\boldsymbol{y}_{\text{pred}}\|_2}
$$

CAMBRIDGE SPARK

Hands-on session

# 01-linear_algebra-skeleton.ipynb

# 30 mins

# Recommended further reading

If you found that a bit hard going, an excellent introduction is Prof. Gilbert Strang's course on linear algebra:

https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/

Another great well partitioned introduction at Khan Academy:

https://www.khanacademy.org/math/linear-algebra