Mathematics for Data Science

Hongchang Gao

Spring 2024

Background

- Training data

1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
2	7129300520	20141013T0(	221900	3	1	1180	5650	1	0
3	6414100192	20141209T0(	538000	3	2.25	2570	7242	2	0
4	5631500400	20150225T0(	180000	2	1	770	10000	1	0
5	2487200875	20141209T0(	604000	4	3	1960	5000	1	0
6	1954400510	20150218T0(	510000	3	2	1680	8080	1	0
7	7237550310	20140512T0(	1.23E+06	4	4.5	5420	101930	1	0
8	1321400060	20140627T0(	257500	3	2.25	1715	6819	2	0
9	2008000270	20150115T0(	291850	3	1.5	1060	9711	1	0
10	2414600126	20150415T0(	229500	3	1	1780	7470	1	0
11	3793500160	20150312T0(	323000	3	2.5	1890	6560	2	0

Background

- Linear regression

$$\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|^2$$

1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
2	7129300520	20141013T0(	221900	3	1	1180	5650	1	0
3	6414100192	20141209T0(	538000	3	2.25	2570	7242	2	0
4	5631500400	20150225T0(	180000	2	1	770	10000	1	0
5	2487200875	20141209T0(	604000	4	3	1960	5000	1	0
6	1954400510	20150218T0(	510000	3	2	1680	8080	1	0
7	7237550310	20140512T0(	1.23E+06	4	4.5	5420	101930	1	0
8	1321400060	20140627T0(	257500	3	2.25	1715	6819	2	0
9	2008000270	20150115T0(	291850	3	1.5	1060	9711	1	0
10	2414600126	20150415T0(	229500	3	1	1780	7470	1	0
11	3793500160	20150312T0(	323000	3	2.5	1890	6560	2	0

Background

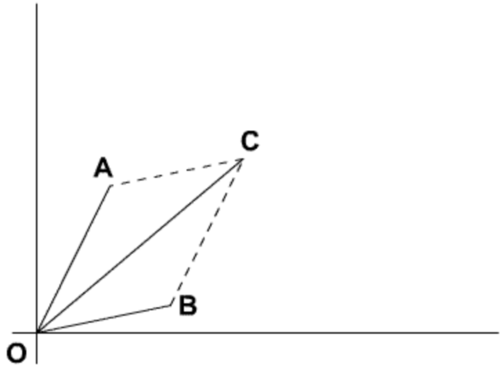
$$\mathbf{X}\mathbf{w} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

$\|\mathbf{x}\|_2$? $\|\mathbf{X}\|_F$?

Vector

- Definition

$$\mathbf{x} = (x_1, x_2, \cdots, x_d)$$



bedrooms	bathrooms	sqft_living	sqft_lot	floors
3	1	1180	5650	1
3	2.25	2570	7242	2
2	1	770	10000	1
4	3	1960	5000	1
3	2	1680	8080	1
4	4.5	5420	101930	1

1. points in a coordinate system

2. objects with magnitude and direction

3. Features (data science)

mathematics

Vector

- Column vector
  - A single column of n elements
  - In mathematics, a vector is assumed to be a column vector in default
- Row vector
  - A single row of n elements

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Column vector

$$y = (y_1, y_2, \ldots, y_n)$$

Row vector

Vector

- Addition
  - Two vectors should have the same size

$$\mathbf{x} = (x_1, x_2, \cdots, x_d)$$

$$\mathbf{y} = (y_1, y_2, \cdots, y_d)$$

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2, \cdots, x_d + y_d)$$

```python
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])
house2 = np.array([3,2,1680, 8080, 1])

print("house1: {}".format(house1))
print("house2: {}".format(house2))
print("house1+house2: {}".format(house1+house2))
```

```
house1: [   3    1 1180 5650    1]
house2: [   3    2 1680 8080    1]
house1+house2: [    6     3  2860 13730     2]
```

Vector

- Subtraction
  - Two vectors should have the same size

$$\mathbf{x} = (x_1, x_2, \cdots, x_d)$$

$$\mathbf{y} = (y_1, y_2, \cdots, y_d)$$

$$\mathbf{x} - \mathbf{y} = (x_1 - y_1, x_2 - y_2, \cdots, x_d - y_d)$$

```python
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])
house2 = np.array([3,2,1680, 8080, 1])

print("house1: {}".format(house1))
print("house2: {}".format(house2))
print("house1-house2: {}".format(house1-house2))
```

```
house1: [    3     1  1180  5650     1]
house2: [    3     2  1680  8080     1]
house1-house2: [     0     -1   -500  -2430      0]
```

Vector

- Scalar multiplication
  - The product between a scalar and a vector

$$c\mathbf{x} = (cx_1, cx_2, \cdots, cx_d)$$

```
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])

print("house1: {}".format(house1))
print("2*house1: {}".format(house1*2))
```

```
house1: [    3     1  1180  5650     1]
2*house1: [    6     2   2360 11300     2]
```

Vector

- Element-wise multiplication
  - Two vectors should have the same size

$$\mathbf{x} = (x_1, x_2, \cdots, x_d)$$

$$\mathbf{y} = (y_1, y_2, \cdots, y_d)$$

$$\mathbf{x} * \mathbf{y} = (x_1 * y_1, x_2 * y_2, \cdots, x_d * y_d)$$

```python
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])
house2 = np.array([3,2,1680, 8080, 1])

print("house1: {}".format(house1))
print("house2: {}".format(house2))
print("house1*house2: {}".format(np.multiply(house1,house2)))
```

```
house1: [   3    1 1180 5650    1]
house2: [   3    2 1680 8080    1]
house1*house2: [       9        2  1982400 45652000        1]
```

Vector

- Inner product
  - Two vectors should have the same size

$$\mathbf{x} = (x_1, x_2, \cdots, x_d)$$

$$\mathbf{y} = (y_1, y_2, \cdots, y_d)$$

$$\mathbf{x}^T\mathbf{y} = x_1 * y_1 + x_2 * y_2 + \cdots + x_d * y_d$$

```
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])
house2 = np.array([3,2,1680, 8080, 1])

print("house1: {}".format(house1))
print("house2: {}".format(house2))
print("inner product: {}".format(np.inner(house1,house2)))
```

```
house1: [    3     1  1180  5650     1]
house2: [    3     2  1680  8080     1]
inner product: 47634412
```

Vector

- Vector norm
  - Measure the magnitude of a vector

ℓ_1-norm: $\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_d|$

ℓ_2-norm (Euclidean norm): $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$

ℓ_∞-norm (Max-norm): $\|\mathbf{x}\|_\infty = \max_{1 \le i \le d} |x_i|$

Vector

- Vector norm

```
import numpy as np

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

print("l1-norm: {}".format(np.linalg.norm(x, ord=1)))
print("l2-norm: {}".format(np.linalg.norm(x, ord=2)))
print("l1-norm: {}".format(np.linalg.norm(x, ord=np.inf)))
```

```
l1-norm: 10.0
l2-norm: 5.477225575051661
l1-norm: 4.0
```

Vector

- Exercise

$$\mathbf{x} = (1, -2, 2, 0)$$

ℓ_1-norm = ?

ℓ_2-norm = ?

ℓ_∞-norm = ?

Vector

- Exercise

$$\mathbf{x} = (1, -2, 2, 0)$$

$$\mathbf{y} = (1, 3, 2, 5)$$

$$\mathbf{x}^T\mathbf{y} = ?$$

Vector

- Properties
  - $\|\mathbf{x}\| \geq 0$ for all $\mathbf{x}$,
  - $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = 0$,
  - $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$, for all $\alpha \in \mathbb{R}$,
  - $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$, the triangular equality.

Vector

- How to measure the distance between two vectors?

$$\mathbf{x} = (x_1, x_2, \cdots, x_d), \mathbf{y} = (y_1, y_2, \cdots, y_d)$$

$$\mathbf{x} - \mathbf{y} = (x_1 - y_1, x_2 - y_2, \cdots, x_d - y_d)$$

$$\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2}$$

```
import numpy as np

house1 = np.array([3,1,1180, 5650, 1])
house2 = np.array([3,2,1680, 8080, 1])

print("distance: {}".format(np.linalg.norm(house1-house2, ord=2)))
```

```
distance: 2480.90729371333
```

Vector

- Exercise

$$\mathbf{x} = (1, -2, 2, 0)$$

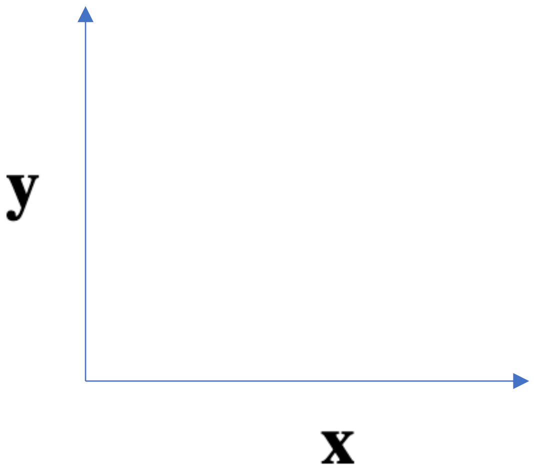
$$\mathbf{y} = (1, 3, 2, 5)$$

$$\|\mathbf{x} - \mathbf{y}\|_2 = ?$$

Vector

- Orthogonality

Two vectors $\mathbf{x}$ and $\mathbf{y}$ are orthogonal, if $\mathbf{x}^T\mathbf{y} = 0$



$$\mathbf{x} = (1, 0),\ \mathbf{y} = (0, 1)$$

$$\mathbf{x} = (-1, 1),\ \mathbf{y} = (1, 1)$$

$$\mathbf{x} = (1, 0, 0),\ \mathbf{y} = (0, 1, 0)$$

$$\mathbf{x} = (2, 2, 0),\ \mathbf{y} = (-1, 1, 10)$$

Matrix

- Definition
  - a rectangular array of numbers

column vector

$$
\text{row vector} \quad
\begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{pmatrix}
$$

matrix

bedrooms	bathrooms	sqft_living	sqft_lot	floors
3	1	1180	5650	1
3	2.25	2570	7242	2
2	1	770	10000	1
4	3	1960	5000	1
3	2	1680	8080	1
4	4.5	5420	101930	1

dataset

Matrix

- Addition
  - A and B should have the same size

$$A + B = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{pmatrix}$$

Matrix

- Addition

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
B = np.array([[11,12,13], [14,15,16]])

print("A= {}\n".format(A))
print("B= {}\n".format(B))
print("A+B= {}\n".format(A+B))
```

```
A = [[1 2 3]
 [4 5 6]]

B = [[11 12 13]
 [14 15 16]]

A+B = [[12 14 16]
 [18 20 22]]
```

Matrix

- Subtraction
  - A and B should have the same size

$$\boldsymbol{A}-\boldsymbol{B}=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}-\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}$$

$$=\begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \dots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \dots & a_{2n}-b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \dots & a_{mn}-b_{mn} \end{pmatrix}$$

Matrix

- Subtraction

```python
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
B = np.array([[11,12,13], [14,15,16]])

print("A = {}\n".format(A))
print("B = {}\n".format(B))
print("A-B = {}\n".format(A-B))
```

```
A = [[1 2 3]
 [4 5 6]]

B = [[11 12 13]
 [14 15 16]]

A-B = [[-10 -10 -10]
 [-10 -10 -10]]
```

Matrix

- Transposition

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \qquad A^T = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix}$$

$$(A^T)^T = A$$

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

Matrix

- Transposition

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
B = np.transpose(A)
C = np.transpose(B)


print("A = {}\n".format(A))
print("B = {}\n".format(B))
print("C = {}\n".format(C))
```

```
A = [[1 2 3]
 [4 5 6]]

B = [[1 4]
 [2 5]
 [3 6]]

C = [[1 2 3]
 [4 5 6]]
```

Matrix

- Scalar multiplication

$$cA = c\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \begin{pmatrix} ca_{11} & ca_{12} & \dots & ca_{1n} \\ ca_{21} & ca_{22} & \dots & ca_{2n} \\ \vdots & \vdots & & \vdots \\ ca_{m1} & ca_{m2} & \dots & ca_{mn} \end{pmatrix}$$

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
c = 2

print("A = {}\n".format(A))
print("c*A = {}\n".format(c*A))
```

```
A = [[1 2 3]
 [4 5 6]]

c*A = [[ 2  4  6]
 [ 8 10 12]]
```

oll output; double click to hide

Matrix

- Matrix-vector multiplication
  - The number of columns of A should be equal to the number of rows of x

$$\mathbf{Ax} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} x_j \\ \sum_{j=1}^{n} a_{2j} x_j \\ \vdots \\ \sum_{j=1}^{n} a_{mj} x_j \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 6 & 4 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 5 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 5 + 3 \cdot (-2) \\ 6 \cdot 5 + 4 \cdot (-2) \\ 1 \cdot 5 + 0 \cdot (-2) \end{pmatrix} = \begin{pmatrix} 4 \\ 22 \\ 5 \end{pmatrix}$$

Matrix

- Matrix-vector multiplication

```
import numpy as np

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

print("A = {}\n".format(A))
print("x = {}\n".format(x))
print("A*x = {}\n".format(np.dot(A,x)))
```

```
A = [[1 2 3]
 [4 5 6]]

x = [[2]
 [2]
 [2]]
```

output; double click to hide

```
A*x = [[12]
 [30]]
```

Matrix

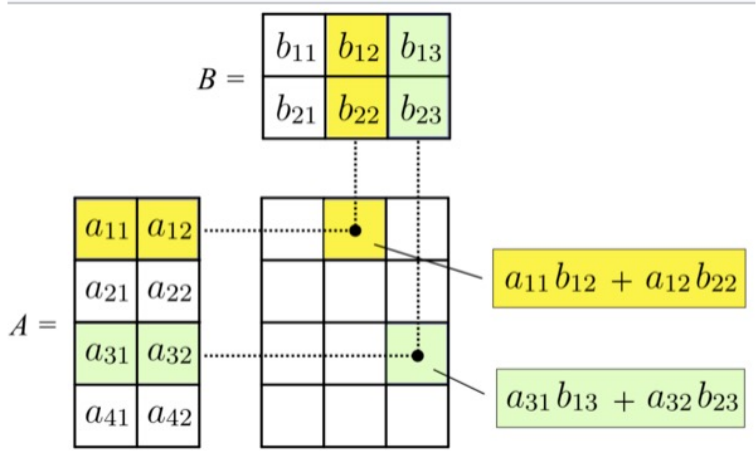
- Matrix-vector multiplication
  - The number of columns of A should be equal to the number of rows of x
  - Linear combination of columns of A

Linear combination

$$\mathbf{Ax} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \sum_{j=1}^{n} x_j \mathbf{a}_j$$

$$\mathbf{Aw} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + 0 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

Matrix



- Matrix-matrix multiplication
  - The number of columns of A should be equal to the number of rows of B

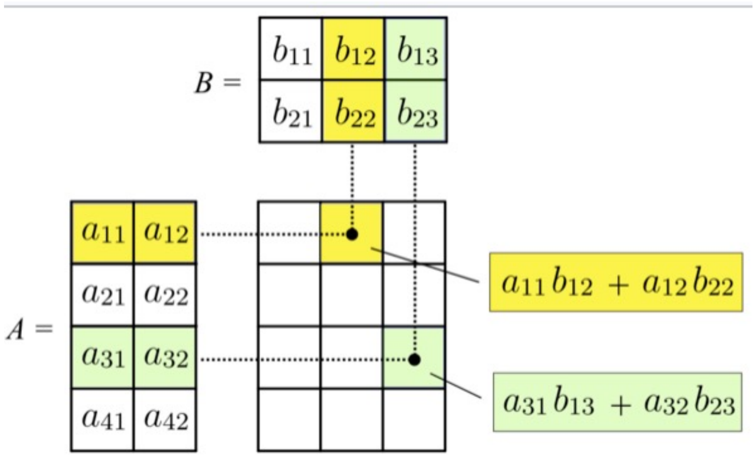
$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1d} \\ a_{21} & a_{22} & \dots & a_{2d} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{md} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{d1} & b_{d2} & \dots & b_{dn} \end{pmatrix}$$

$$C_{n\times m} = AB = \begin{pmatrix} \sum_{k=1}^{d} a_{1k}b_{k1} & \sum_{k=1}^{d} a_{1k}b_{k2} & \dots & \sum_{k=1}^{d} a_{1k}b_{km} \\ \sum_{k=1}^{d} a_{2k}b_{k1} & \sum_{k=1}^{d} a_{2k}b_{k2} & \dots & \sum_{k=1}^{d} a_{2k}b_{km} \\ \vdots & \vdots & & \vdots \\ \sum_{k=1}^{d} a_{nk}b_{k1} & \sum_{k=1}^{d} a_{nk}b_{k2} & \dots & \sum_{k=1}^{d} a_{nk}b_{km} \end{pmatrix}$$

$$AB \neq BA$$

Matrix

- Matrix-matrix multiplication



$$\begin{pmatrix} 2 & 3 \\ 6 & 4 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 5 & 1 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} 2 \cdot 5 - 3 \cdot 2 & 2 \cdot 1 + 3 \cdot 1 \\ 6 \cdot 5 - 4 \cdot 2 & 6 \cdot 1 + 4 \cdot 1 \\ 1 \cdot 5 - 0 \cdot 2 & 1 \cdot 1 + 0 \cdot 1 \end{pmatrix} = \begin{pmatrix} 4 & 6 \\ 22 & 10 \\ 5 & 1 \end{pmatrix}$$

Matrix

- Matrix-matrix multiplication

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
B = np.array([[1,1],[2,2],[3,3]])
C = np.dot(A,B)


print("A = {}\n".format(A))
print("B = {}\n".format(B))
print("C = {}\n".format(C))
```

```
A = [[1 2 3]
 [4 5 6]]

B = [[1 1]
 [2 2]
 [3 3]]

C = [[14 14]
 [32 32]]
```

Matrix

- Norm
  - Frobenius norm

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

For all scalars $\alpha \in K$ and for all matrices $A, B \in K^{m \times n}$,

- $\|\alpha A\| = |\alpha| \|A\|$ (being *absolutely homogeneous*)
- $\|A + B\| \leq \|A\| + \|B\|$ (being *sub-additive* or satisfying the *triangle inequality*)
- $\|A\| \geq 0$ (being *positive-valued*)
- $\|A\| = 0 \iff A = 0_{m,n}$ (being *definite*)

Matrix

```
import numpy as np

A = np.array([[1,2,3],[4,5,6]])
norm = np.linalg.norm(x, ord='fro')

print(" Frobenius norm: {}".format(norm))
```

```
Frobenius norm: 3.4641016151377544
```

Matrix

- Orthonormal basis

Given n vectors $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, if

$$\mathbf{x}_i^T \mathbf{x}_j = 0, i \neq j \text{ and } \|\mathbf{x}_i\|_2 = 1, \forall i$$

- Orthogonal matrix
  - A matrix $X \in \mathbf{R}^{n \times n}$ with orthonormal columns is called an orthogonal matrix

$$X^T X = I$$

Matrix

- Symmetric matrix

$$A = A^T$$

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 7 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

Gradient Descent

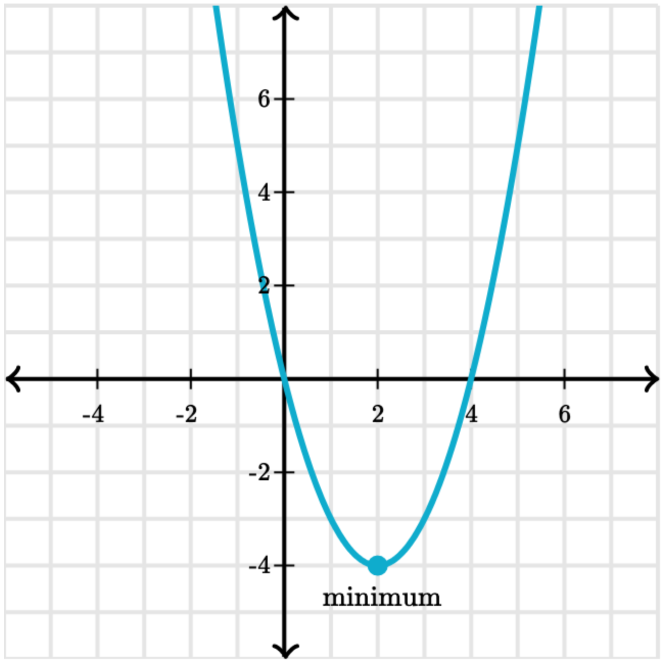
- Machine learning model is an optimization problem

$$\min_x f(x)$$

  - f(x) is the objective function, or loss function.
  - x is the model parameter.
  - Goal: find a model parameter to minimize the objective function

Gradient Descent

- Gradient Descent method:
  - Decrease the loss function along the direction of the negative gradient



Gradient Descent

- Gradient Descent method:
  - Decrease the loss function along the direction of the negative gradient

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

- x_t is the model parameter in the t-th iteration
- $\nabla f(x_t)$ is the gradient
- η is the learning rate, or step size

Gradient Descent

- Example: Linear Regression

$$\min_w \|y - Xw\|^2$$

  - Gradient:

$$-2X^T(y - Xw)$$

  - Gradient descent:

$$w_{t+1} = w_t - \eta(2X^T(Xw_t - y))$$