Applied Math for Machine Learning

Prof. Kuan-Ting Lai

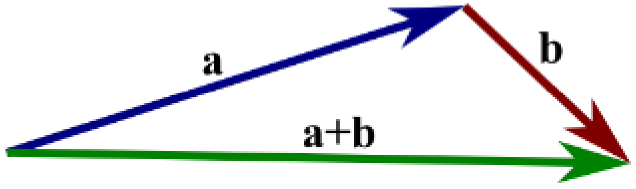
2021/3/11

Applied Math for Machine Learning

- Linear Algebra
- Probability
- Calculus
- Optimization

Linear Algebra

- Scalar
  - real numbers
- Vector (1D)
  - Has a magnitude & a direction



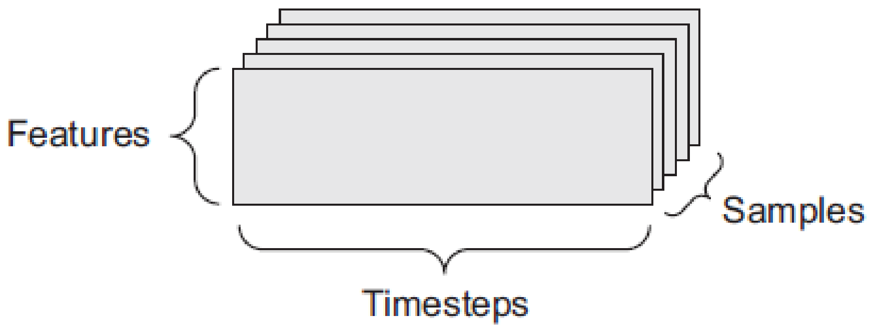
- Matrix (2D)
  - An array of numbers arranges in rows & columns

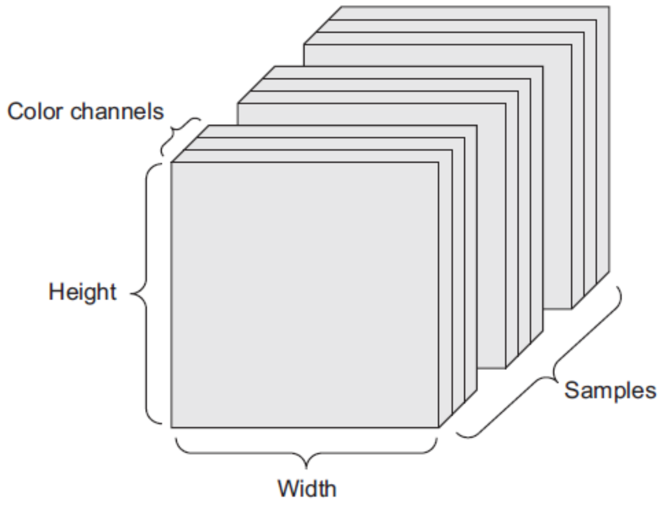
$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

- Tensor (>=3D)
  - Multi-dimensional arrays of numbers

Real-world examples of Data Tensors

- Timeseries Data – 3D (samples, timesteps, features)
- Images – 4D (samples, height, width, channels)
- Video – 5D (samples, frames, height, width, channels)





Vector Dimension vs. Tensor Dimension

- The number of data in a vector is also called “dimension”
- In deep learning , the dimension of Tensor is also called “rank”
- Matrix = 2d array = 2d tensor = rank 2 tensor

https://deeplizard.com/learn/video/AiyK0idr4uM

The Matrix

Matrix

- Define a matrix with m rows and n columns:

$$A_{m \times n} \in \mathbb{R}^{m \times n}$$

$$\begin{array}{cc} & n\ columns \xrightarrow{j} \\ a_{ij} & \\ m\ rows & \\ i \downarrow & \begin{pmatrix} a_{11} & a_{12} & a_{13} & . & . & . \\ a_{21} & a_{22} & a_{23} & . & . & . \\ a_{31} & a_{32} & a_{33} & . & . & . \\ . & . & . & . & & \\ . & . & . & & . & \\ . & . & . & & & . \end{pmatrix} \end{array}$$

Santanu Pattanayak, ”Pro Deep Learning with TensorFlow,” Apress, 2017

Matrix Operations

- Addition and Subtraction

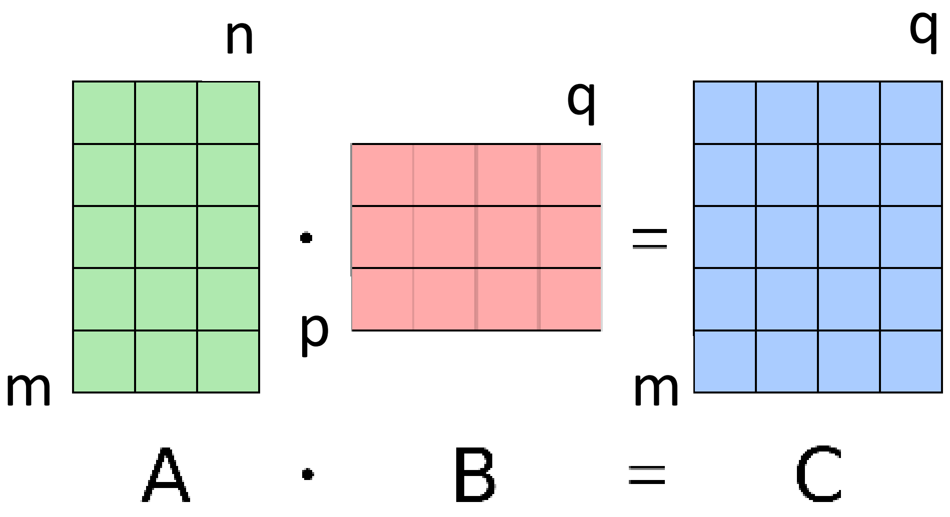
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

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

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

Matrix Multiplication

- Two matrices A and B, where $A \in \mathbb{R}^{m \times n}$ $B \in \mathbb{R}^{p \times q}$
- The columns of A must be equal to the rows of B, i.e. n == p
- A * B = C, where $C \in \mathbb{R}^{m \times q}$
- $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$



Example of Matrix Multiplication (3-1)

"Dot Product"

$1 \times 7 + 2 \times 9 + 3 \times 11$

$=$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Example of Matrix Multiplication (3-2)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ & \end{bmatrix}$$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Example of Matrix Multiplication (3-3)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

https://www.mathsisfun.com/algebra/matrix-multiplying.html

Matrix Transpose

$$A \in \mathbb{R}^{m \times n} \qquad A^{\mathrm{T}} \in \mathbb{R}^{n \times m}$$

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

$$\mathbf{A} \qquad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \Longrightarrow \mathbf{A}^{\mathrm{T}} \qquad \boxed{\begin{matrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{matrix}}$$

https://en.wikipedia.org/wiki/Transpose

Dot Product

- Dot product of two vectors become a **scalar**
- Inner product is a generalization of the dot product
- Notation: $v_1 \cdot v_2$ or $v_1{}^T v_2$

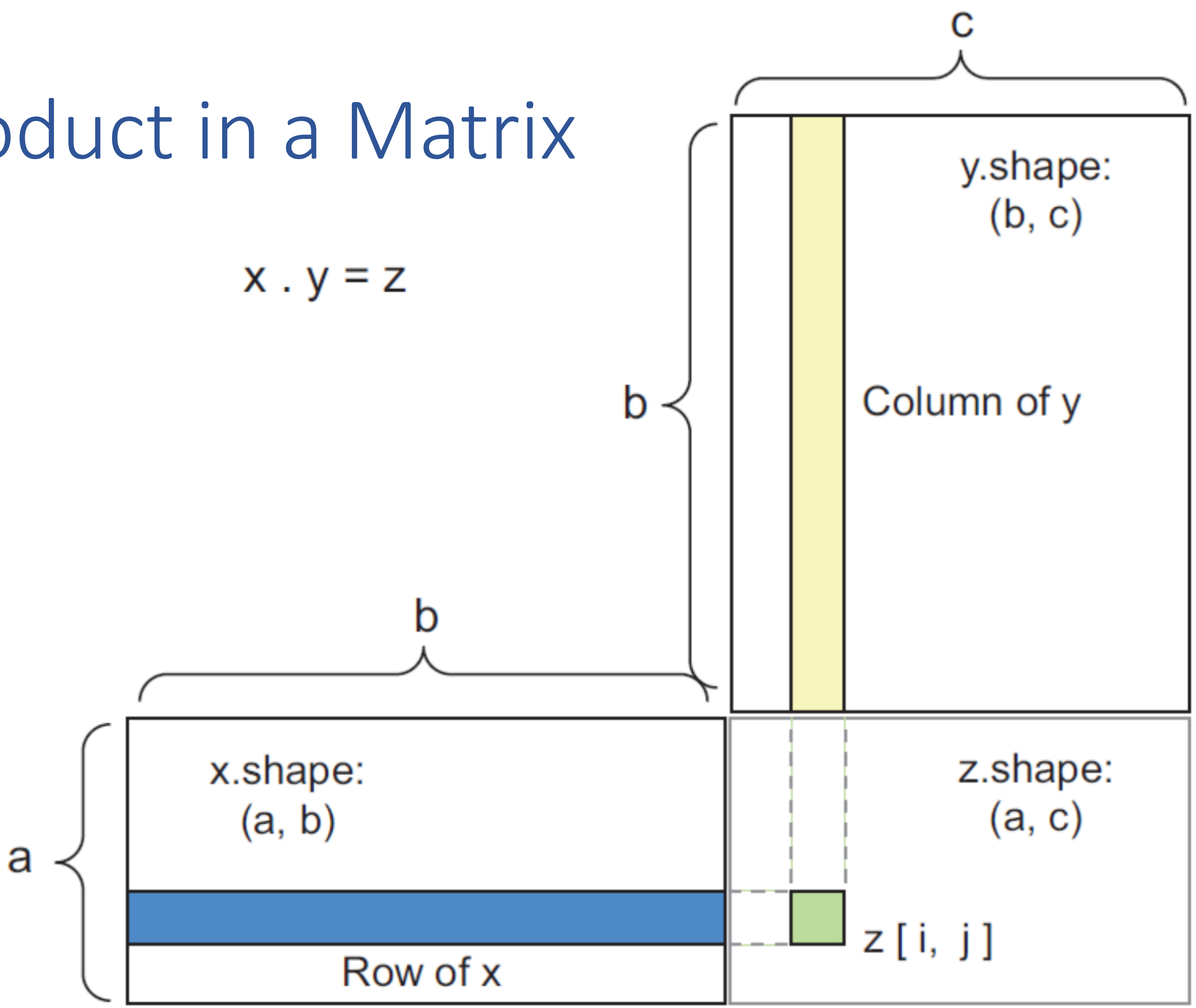
$$v_1 = \begin{bmatrix} v_{11} \\ v_{12} \\ \cdot \\ \cdot \\ \cdot \\ v_{1n} \end{bmatrix} \quad v_2 = \begin{bmatrix} v_{21} \\ v_{22} \\ \cdot \\ \cdot \\ \cdot \\ v_{2n} \end{bmatrix}$$

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

$v_1^T = [\quad\quad] \quad [\;] = v_2$

Dot Product in a Matrix

x . y = z



Outer Product

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{A} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

Or in index notation:

$$(\mathbf{u} \otimes \mathbf{v})_{ij} = u_i v_j$$

https://en.wikipedia.org/wiki/Outer_product

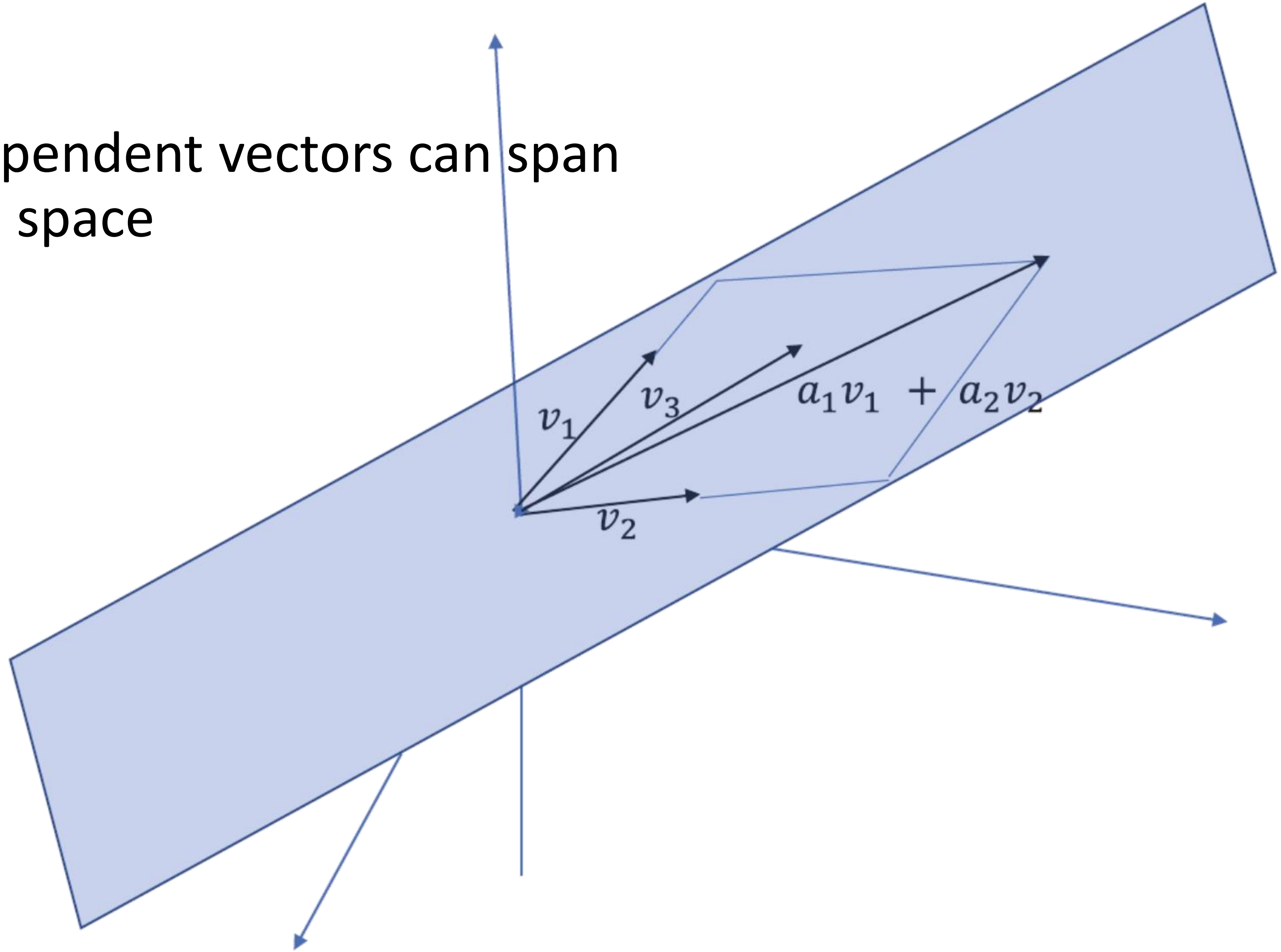
Linear Independence

- A vector is **linearly dependent** on other vectors if it can be expressed as the linear combination of other vectors $V = aV_1 + bV_2$
- A set of vectors $v_1, v_2, \cdots, v_n$ is **linearly independent** if $a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = 0$ implies all $a_i = 0, \forall i \in \{1,2,\cdots n\}$

$$[v_1 v_2 \ldots. v_n]\begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ a_n \end{bmatrix} = 0 \; where \;\; v_i \in \mathbb{R}^{m\times 1} \; \forall i \in \{1,2,\ldots,n\}, \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ a_n \end{bmatrix} \in \mathbb{R}^{n\times 1}$$

Span the Vector Space

- *n* linearly independent vectors can span *n*-dimensional space



Rank of a Matrix

- Rank is:
  - The number of linearly independent row or column vectors
  - The dimension of the vector space generated by its columns
- Row rank = Column rank
- Example:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \xrightarrow{\text{Row-echelon form}} \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Rank_(linear_algebra)

Identity Matrix I

- Any vector or matrix multiplied by I remains unchanged
- For a matrix A, $AI = IA = A$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

$$Iv = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Inverse of a Matrix

- The product of a square matrix A and its inverse matrix A^{-1} produces the identity matrix I
- $AA^{-1} = A^{-1}A = I$
- Inverse matrix is square, but not all square matrices has inverses

Pseudo Inverse

- Non-square matrix and have left-inverse or right-inverse matrix
- Example:

$$Ax = b, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n$$

  – Create a square matrix $A^T A$

$$A^T A x = A^T b$$

  – Multiplied both sides by inverse matrix $(A^T A)^{-1}$

$$x = (A^T A)^{-1} A^T b$$

  – $(A^T A)^{-1} A^T$ is the pseudo inverse function

Norm

- Norm is a measure of a vector's magnitude
- l_2 norm

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

- l_1 norm

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

- l_p norm

$$\left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

- l_∞ norm

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

Eigen Vectors

- Eigenvector is a non-zero vector that changed by only a scalar factor λ when linear transform A is applied to:

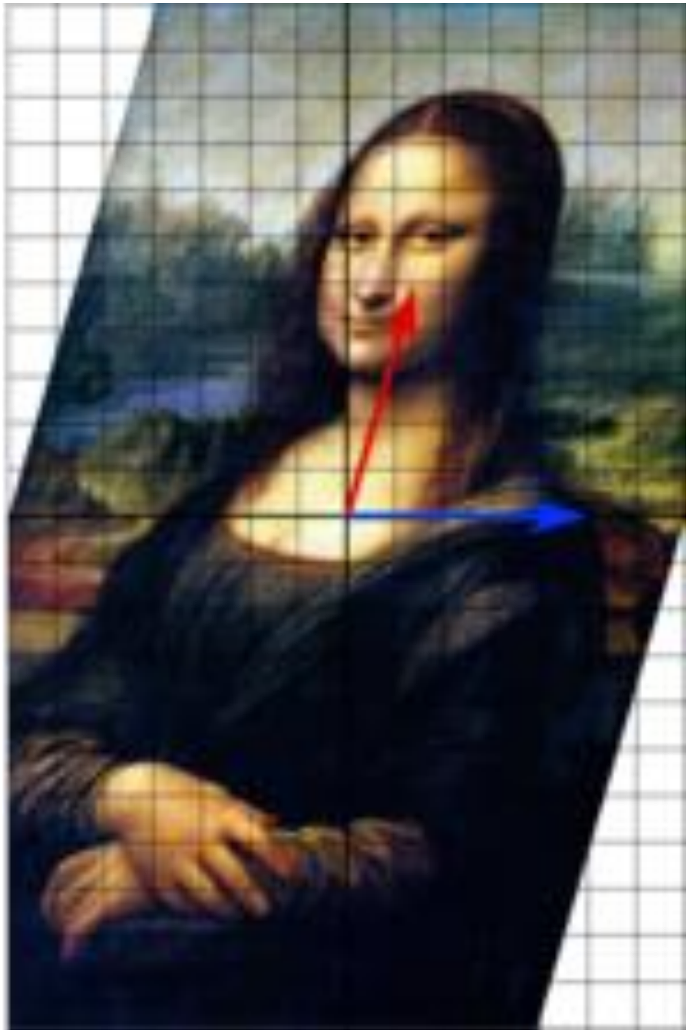
$$Ax = \lambda x, A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^{n}$$

- x are Eigenvectors and λ are Eigenvalues

- One of the most important concepts in machine learning, ex:
  - Principle Component Analysis (PCA)
  - Eigenvector centrality
  - PageRank
  - …

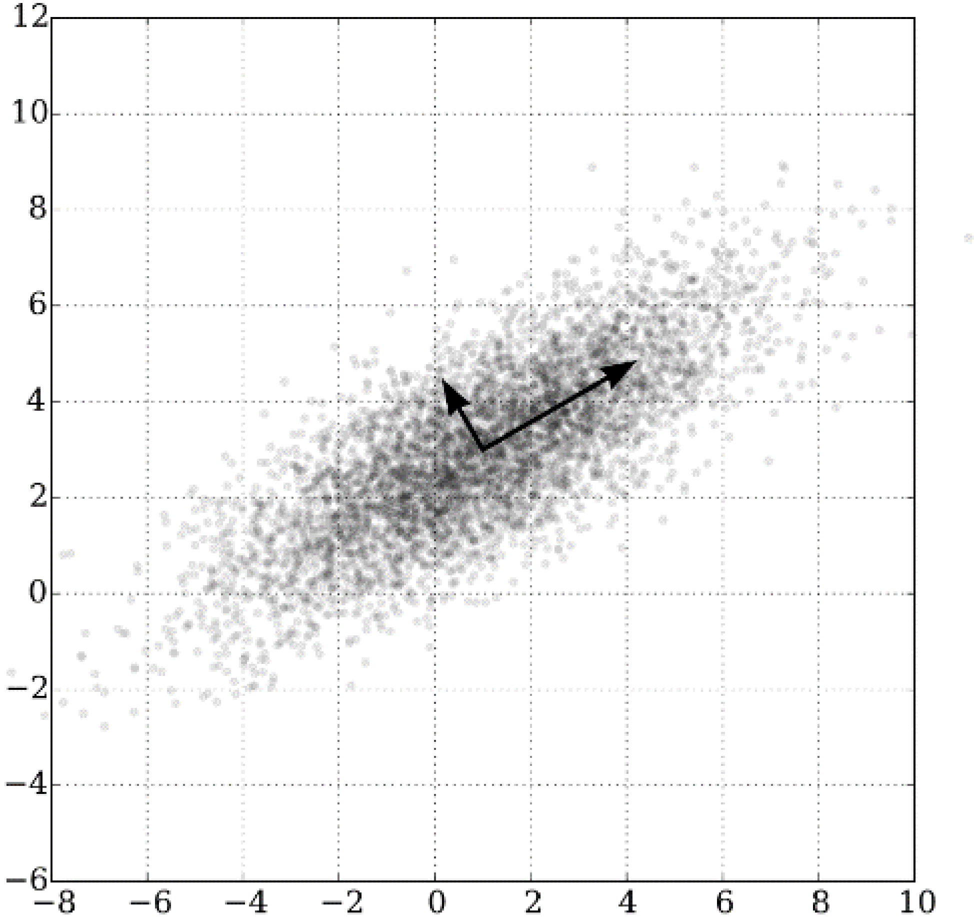
Example: Shear Mapping

- Horizontal axis is the Eigenvector



Principle Component Analysis (PCA)

- Eigenvector of Covariance Matrix



https://en.wikipedia.org/wiki/Principal_component_analysis

NumPy for Linear Algebra

- NumPy is the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

Create Tensors

Scalars (0D tensors)

```
>>> import numpy as np
>>> x = np.array(12)
>>> x
array(12)
>>> x.ndim
0
```

Vectors (1D tensors)

```
>>> x = np.array([12, 3, 6, 14])
>>> x
array([12, 3, 6, 14])
>>> x.ndim
1
```

Matrices (2D tensors)

```
>>> x = np.array([[5, 78, 2, 34, 0],
                  [6, 79, 3, 35, 1],
                  [7, 80, 4, 36, 2]])
>>> x.ndim
2
```

Create 3D Tensor

```
>>> x = np.array([[[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]],
                  [[5, 78, 2, 34, 0],
                   [6, 79, 3, 35, 1],
                   [7, 80, 4, 36, 2]]])
>>> x.ndim
3
```

Attributes of a Numpy Tensor

- Number of axes (dimensions, rank)
  - x.ndim
- Shape
  - This is a tuple of integers showing how many data the tensor has along each axis
- Data type
  - uint8, float32 or float64

Numpy Multiplication

In [ ]:
```
import numpy as np
```

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

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

In [5]:
```
y = np.array([[7, 8], [9, 10], [11, 12]])
y
```

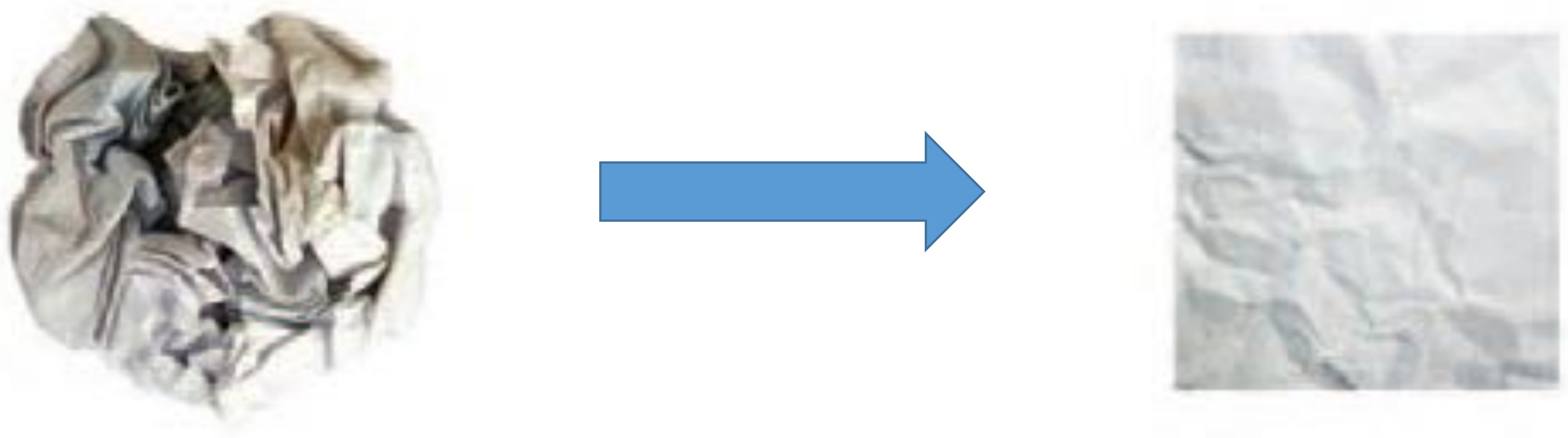
Out[5]:
```
array([[ 7,  8],
       [ 9, 10],
       [11, 12]])
```

In [10]:
```
np.matmul(x, y)
```

Out[10]:
```
array([[ 58,  64],
       [139, 154]])
```

Unfolding the Manifold

- Tensor operations are complex geometric transformation in high-dimensional space
  - Dimension reduction



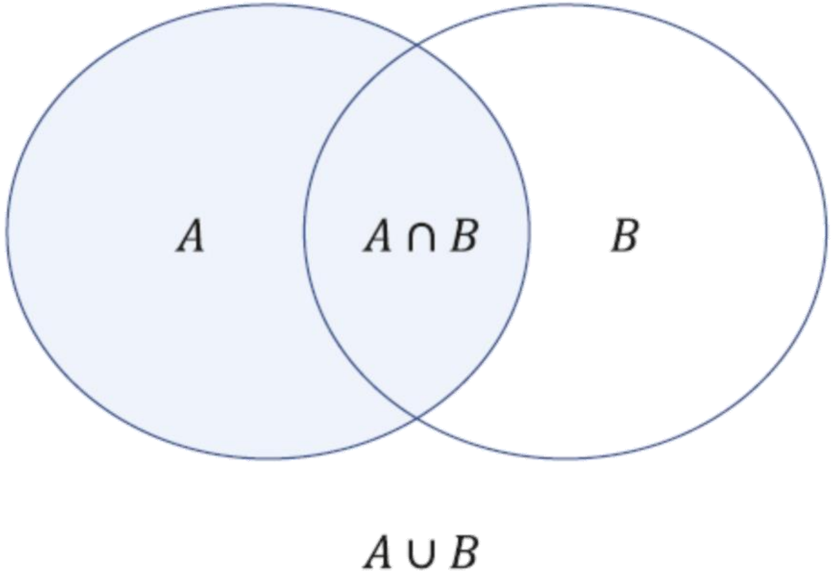
Basics of Probability

Three Axioms of Probability

- Given an Event E in a sample space S, $S = \bigcup_{i=1}^{N} E_i$
- First axiom
  - $P(E) \in \mathbb{R}, 0 \leq P(E) \leq 1$
- Second axiom
  - $P(S) = 1$
- Third axiom
  - Additivity, any countable sequence of mutually exclusive events E_i
  - $P(\bigcup_{i=1}^{n} E_i) = P(E_1) + P(E_2) + \cdots + P(E_n) = \sum_{i=1}^{n} P(E_i)$

Union, Intersection, and Conditional Probability

- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A \cap B)$ is simplified as $P(AB)$
- Conditional Probability $P(A|B)$, the probability of event A given B has occurred
  - $P(A|B) = P\left(\frac{AB}{B}\right)$
  - $P(AB) = P(A|B)P(B) = P(B|A)P(A)$



Chain Rule of Probability

- The joint probability can be expressed as chain rule

$$\frac{P(A_2\cancel{A_1})}{\cancel{P(A_1)}} \qquad \frac{P(A_1A_2A_3)}{\cancel{P(A_1A_2)}}$$

$$P(A_1A_2A_3\ldots A_n) = \cancel{P(A_1)}P(A_2/A_1)P(A_3/A_1A_2)\ldots\ldots P\left(A_n/A_1A_2..A_{(n-1)}\right)$$

Mutually Exclusive

- $P(AB) = 0$
- $P(A \cup B) = P(A) + P(B)$

Independence of Events

- Two events A and B are said to be independent if the probability of their intersection is equal to the product of their individual probabilities
  - $P(AB) = P(A)P(B)$
  - $P(A|B) = P(A)$

Bayes Rule

(Training Data)

feature class (label)

likelihood

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

prior (class)

class

features (Image)

Evidence

Proof:

Remember $P(A|B) = P\left(\frac{AB}{B}\right)$

So $P(AB) = P(A|B)P(B) = P(B|A)P(A)$

Then Bayes $P(A|B) = P(B|A)P(A)/P(B)$

Naïve Bayes Classifier

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

$$p(C_k \mid x_1, \dots, x_n)$$

$$\begin{aligned}
p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k)\, p(x_3, \dots, x_n, C_k) \\
&= \cdots \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k) \cdots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k)
\end{aligned}$$

Naïve = Assume All Features Independent

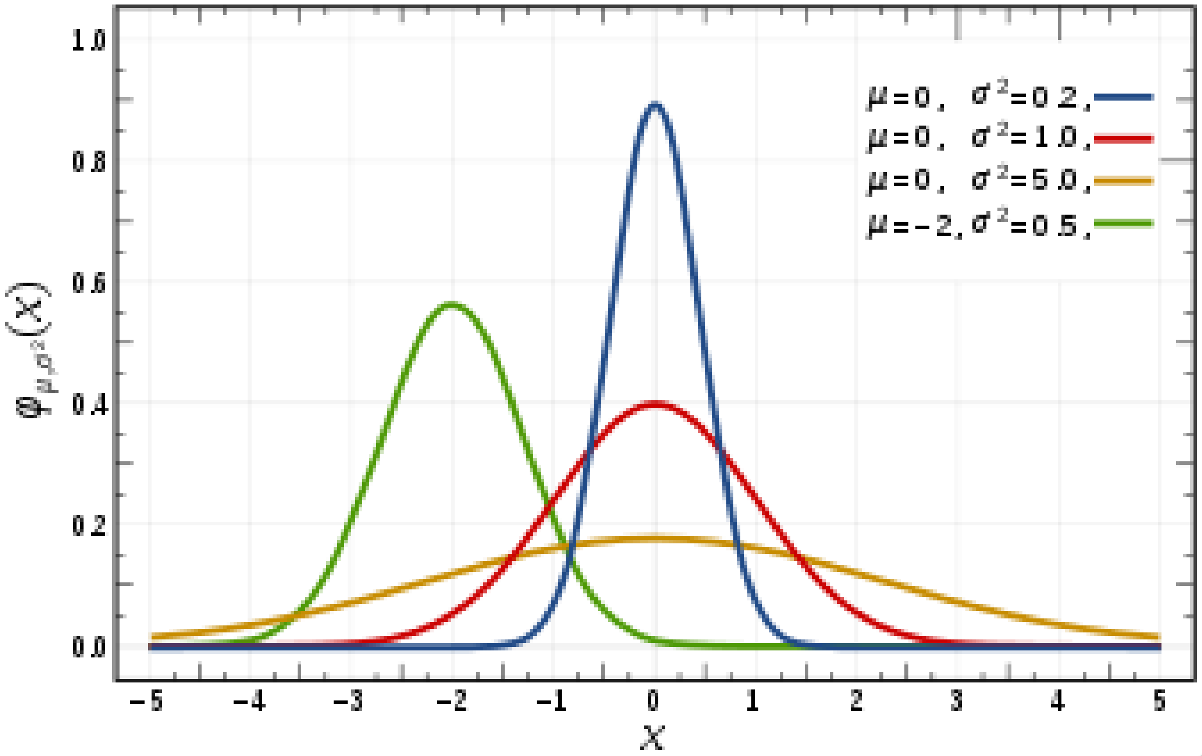
$$p(x_i \mid x_{i+1}, \cancel{\ldots}, x_n, C_k) = p(x_i \mid C_k)$$

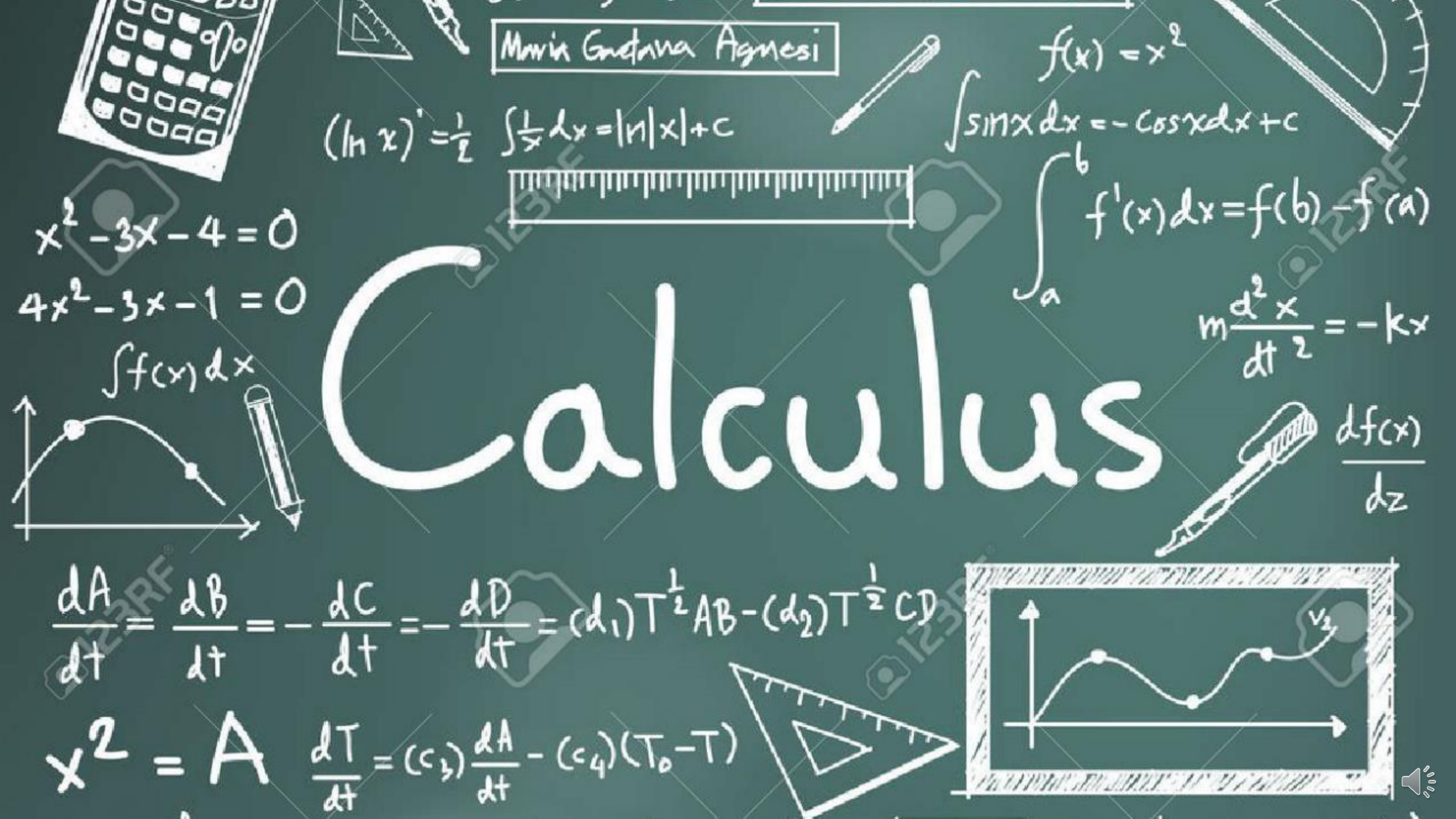
$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &= p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,, \end{aligned}$$

Normal (Gaussian) Distribution

- One of the most important distributions
- Central limit theorem
  - Averages of samples of observations of random variables independently drawn from independent distributions converge to the normal distribution

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



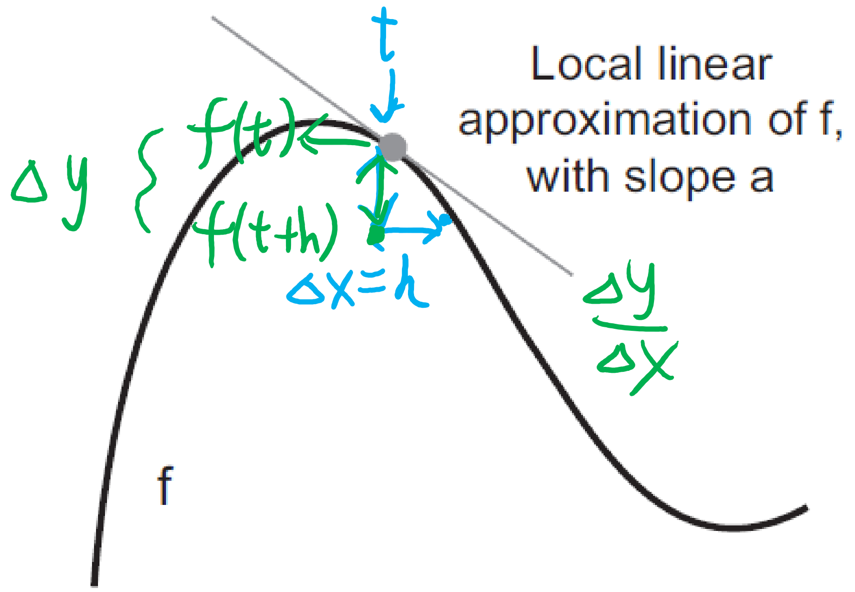

Maria Gaetana Agnesi
f(x) = x^2
(ln x)' = 1/x
∫ 1/x dx = ln|x| + c
∫ sin x dx = - cos x dx + c
∫_a^b f'(x) dx = f(b) - f(a)
x^2 - 3x - 4 = 0
4x^2 - 3x - 1 = 0
∫ f(x) dx
Calculus
m d^2x/dt^2 = -kx
df(x)/dz
dA/dt = dB/dt = - dC/dt = - dD/dt = (d1)T^(1/2) AB - (d2)T^(1/2) CD
x^2 = A
dT/dt = (c3) dA/dt - (c4)(T0 - T)
v2

Differentiation

$$\frac{df}{dt} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$

OR

$$\frac{df}{dt} = \lim_{h \to 0} \frac{f(t+h) - f(t-h)}{2h}$$



Derivatives of Basic Function $\frac{dy}{dx}$

$y = x^n \rightarrow \frac{dy}{dx} = nx^{n-1}$ $\quad \frac{d}{dx}\left(\frac{1}{x}\right) \Rightarrow \frac{-1}{x^2}$

$y = e^x \rightarrow \frac{dy}{dx} = e^x$

$y = \ln x \rightarrow y' = \frac{1}{x}$

Gradient of a Function

- Gradient is a multi-variable generalization of the derivative
- Apply partial derivatives

$$\nabla f = \left[\frac{\partial f}{\partial x_1} \; \frac{\partial f}{\partial x_2} \cdots\cdots \frac{\partial f}{\partial x_n} \right]^T$$

- Example

$$f(x, y, z) = x + y^2 + z^3$$

$$\nabla f = 1 \times 2y \times 3z^2$$

Chain Rule

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

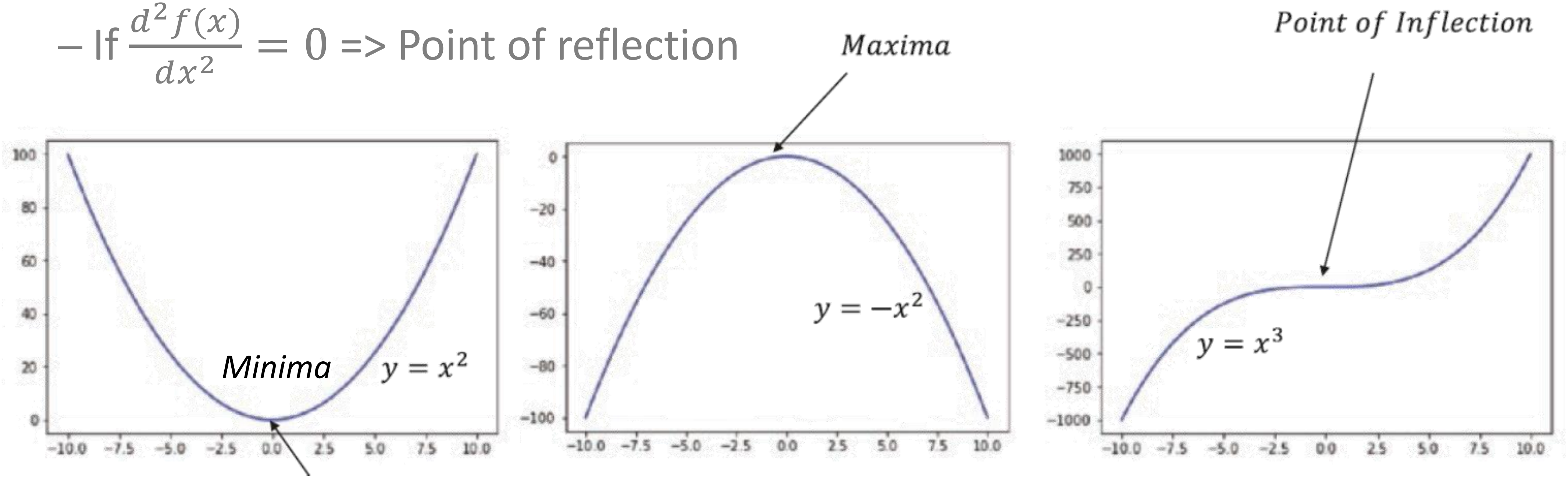
$$\frac{d^2y}{dx^2} = \frac{d^2y}{du^2}\left(\frac{du}{dx}\right)^2 + \frac{dy}{du}\frac{d^2u}{dx^2}$$

$$\frac{d^3y}{dx^3} = \frac{d^3y}{du^3}\left(\frac{du}{dx}\right)^3 + 3\,\frac{d^2y}{du^2}\frac{du}{dx}\frac{d^2u}{dx^2} + \frac{dy}{du}\frac{d^3u}{dx^3}$$

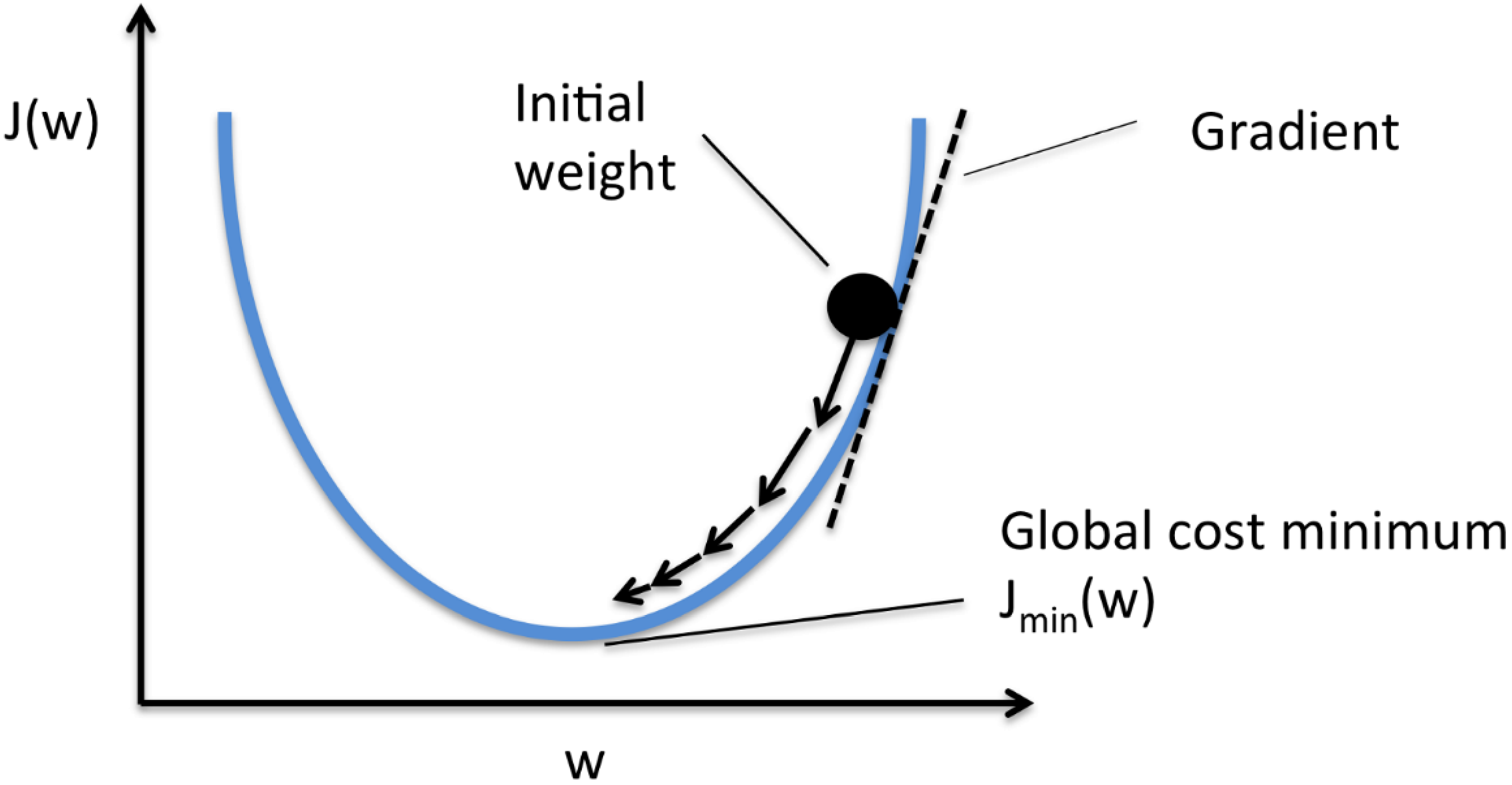
$$\frac{d^4y}{dx^4} = \frac{d^4y}{du^4}\left(\frac{du}{dx}\right)^4 + 6\,\frac{d^3y}{du^3}\left(\frac{du}{dx}\right)^2\frac{d^2u}{dx^2} + \frac{d^2y}{du^2}\left(4\,\frac{du}{dx}\frac{d^3u}{dx^3} + 3\left(\frac{d^2u}{dx^2}\right)^2\right) + \frac{dy}{du}\frac{d^4u}{dx^4}.$$

Maxima and Minima for Univariate Function

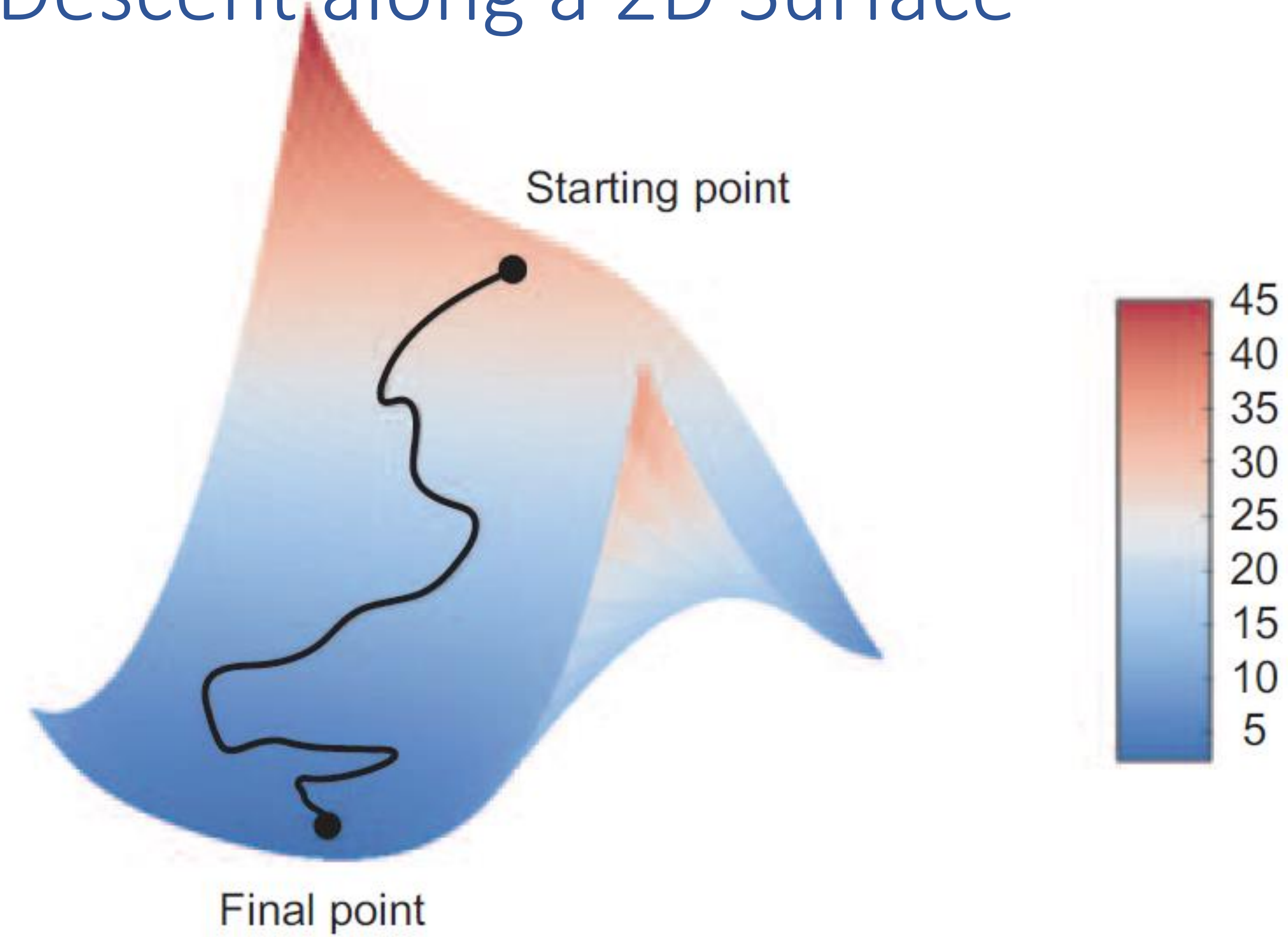
- If $\frac{df(x)}{dx} = 0$, it's a minima or a maxima point, then we study the second derivative:
  - If $\frac{d^2f(x)}{dx^2} < 0$ => Maxima
  - If $\frac{d^2f(x)}{dx^2} > 0$ => Minima
  - If $\frac{d^2f(x)}{dx^2} = 0$ => Point of reflection



Gradient Descent



Gradient Descent along a 2D Surface

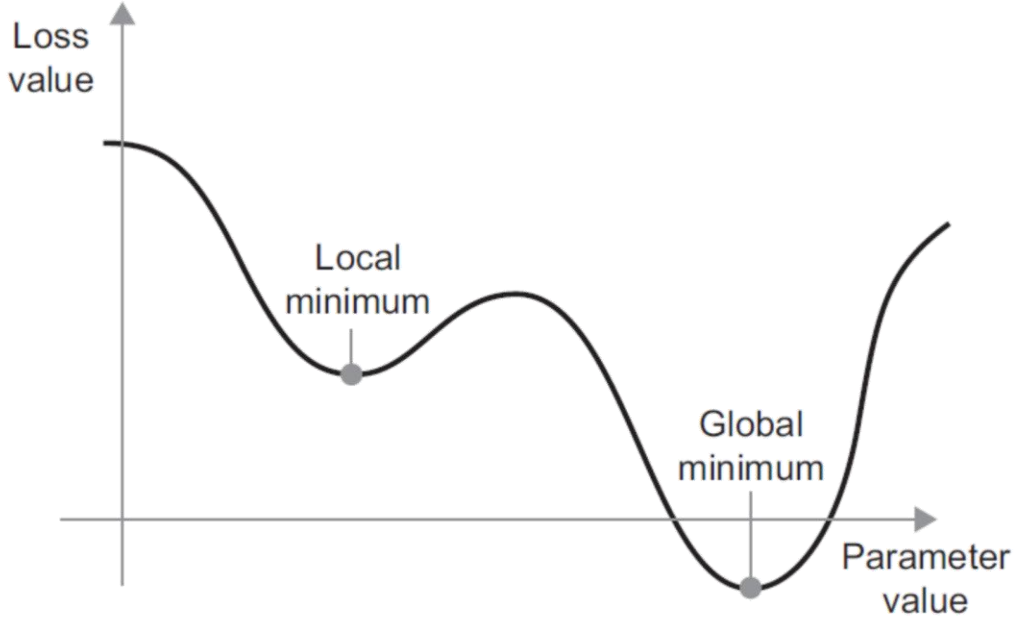


Avoid Local Minimum using Momentum

```
past_velocity = 0.
momentum = 0.1
while loss > 0.01:
    w, loss, gradient = get_current_parameters()
    velocity = past_velocity * momentum + learning_rate * gradient
    w = w + momentum * velocity - learning_rate * gradient
    past_velocity = velocity
    update_parameter(w)
```

Constant momentum factor

Optimization loop



Optimization

The *standard form* of a continuous optimization problem is[1]

min.

s.b.t.

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) \\ &\text{subject to} && g_i(x) \leq 0, \quad i = 1, \dots, m \\ &&& h_j(x) = 0, \quad j = 1, \dots, p \end{aligned}$$

where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the **objective function** to be minimized over the *n*-variable vector x,
- $g_i(x) \leq 0$ are called **inequality constraints**
- $h_j(x) = 0$ are called **equality constraints**, and
- $m \geq 0$ *and* $p \geq 0$.

https://en.wikipedia.org/wiki/Optimization_problem

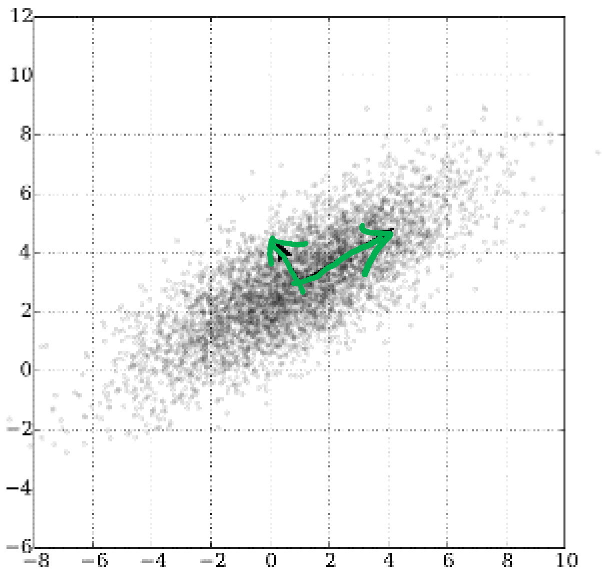
Principle Component Analysis (PCA)

$$y = W^T x$$

- Assumptions
  - *Linearity*
  - *Mean and Variance are sufficient statistics*
  - *The principal components are orthogonal*

$$\max \ \mathrm{Cov}(y, y)$$

$$\text{s.b.t.} \quad W^T W = I$$



Principle Component Analysis (PCA)

$$\max_{W} \quad \mathrm{cov}(\mathbf{Y}, \mathbf{Y})$$

$$s.b.t \quad \mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{I}$$

$W^T x$

$f = \mathrm{cov}(Y, Y) + \lambda(W^T W - I)$

$\mathrm{Cov}(X, X)$

$$\mathrm{Cov}(Y, Y) = \frac{1}{N-1}(Y - \mu_y)^T (Y - \mu_y) = \frac{1}{N-1}(W^T x - W^T \mu_x)^T (W^T x - W^T \mu_x) = W \Sigma_x W^T$$

$$\frac{df}{dw} = 0 \Rightarrow \frac{d}{dw}\left(W \Sigma_x W^T + \lambda(W^T W - I)\right)$$

$$\Rightarrow 2\Sigma_x W + 2\lambda W = 0 \Rightarrow \Sigma_x W = \lambda W$$

References

- Francois Chollet, “Deep Learning with Python,” Chapter 2 “Mathematical Building Blocks of Neural Networks”
- Santanu Pattanayak, ”Pro Deep Learning with TensorFlow,” Apress, 2017
- Machine Learning Cheat Sheet
- https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/
- https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization-How-does-it-solve-the-problem-of-overfitting-Which-regularizer-to-use-and-when
- Wikipedia