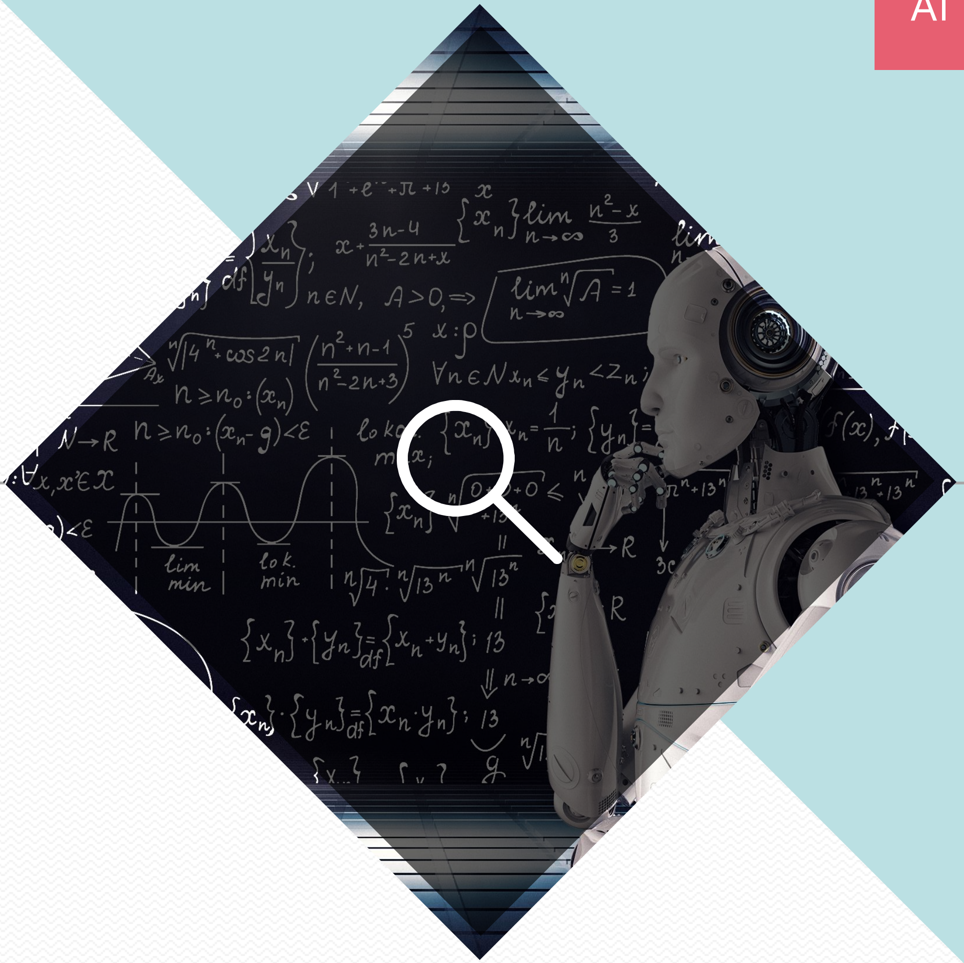
AI TRAINING SERIES



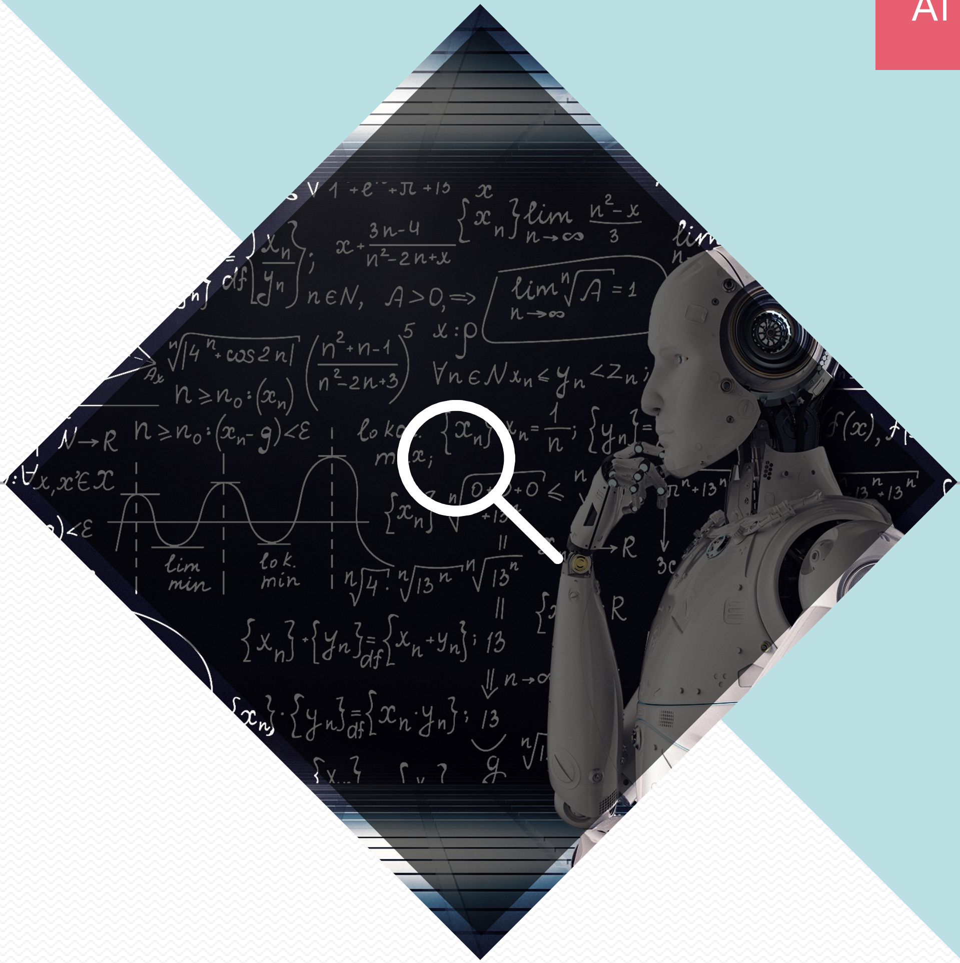
Vector/Matrix/ Entropy

Vector & matrix

CONTENTS

Vector & Matrix

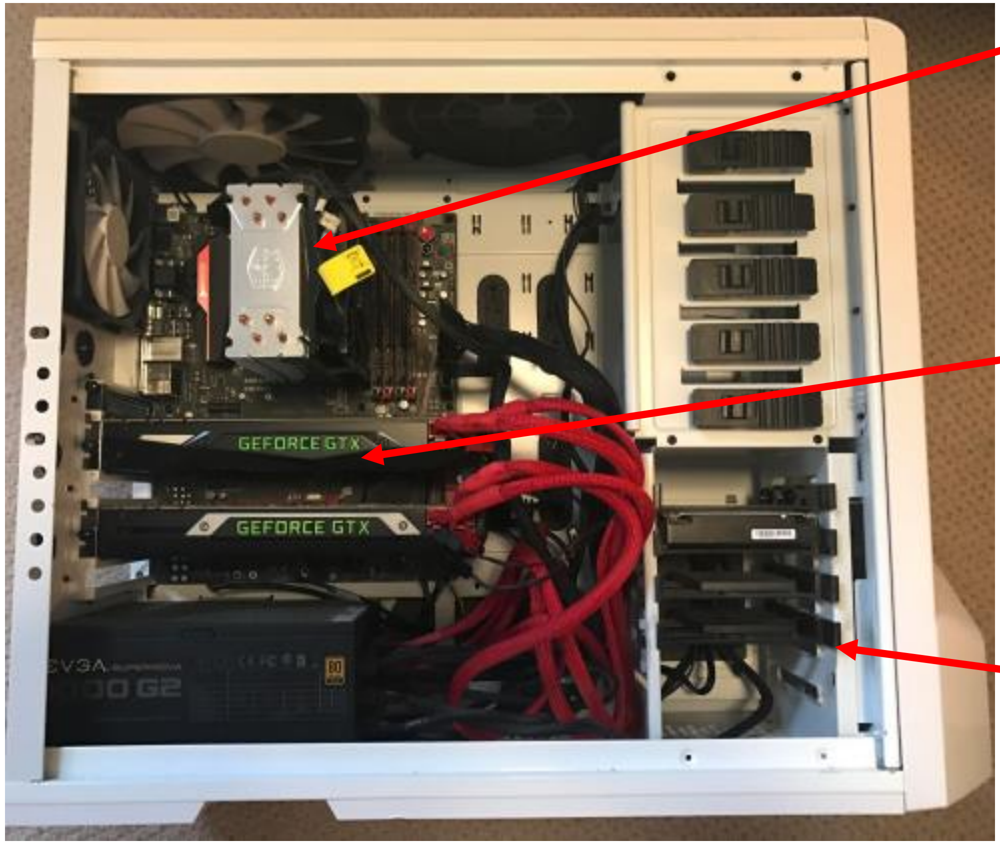
AI TRAINING SERIES



Tensor Intro

Why is Tensor? How is it Defined?

My computer

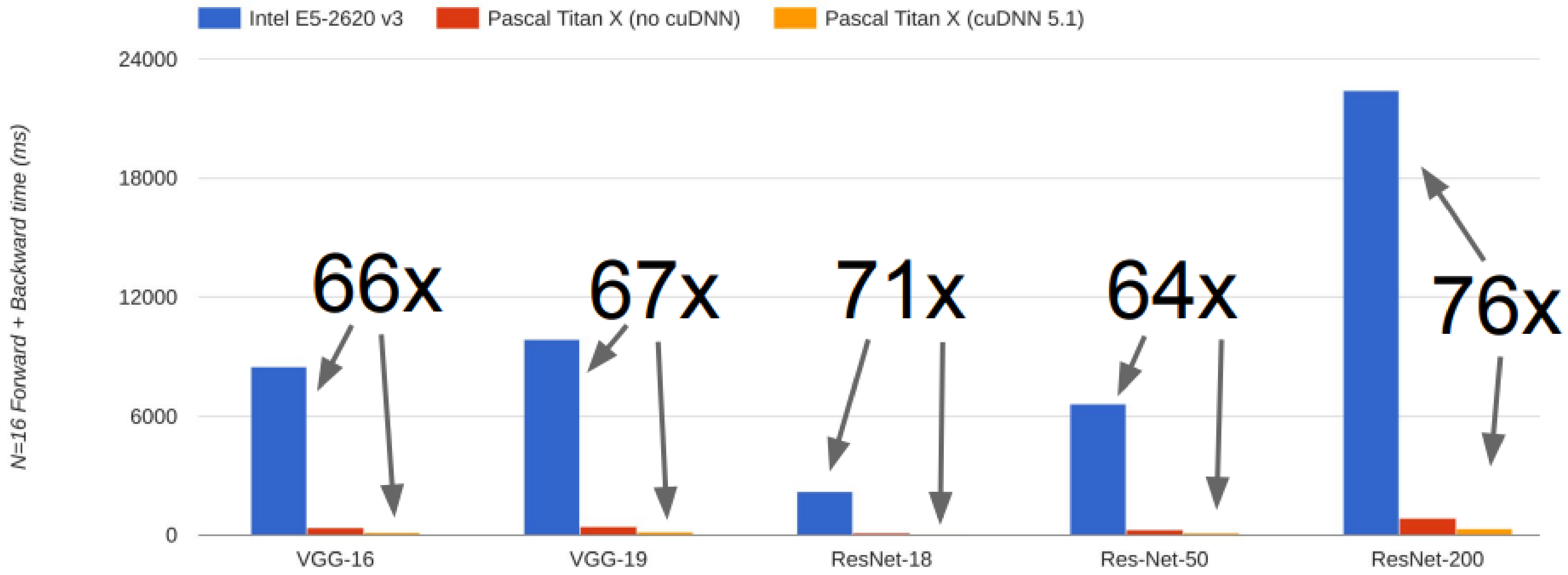


Central processing unit

Graphics processing unit

Memory

CPU vs GPU in practice



Data from https://github.com/jcjohnson/cnn-benchmarks

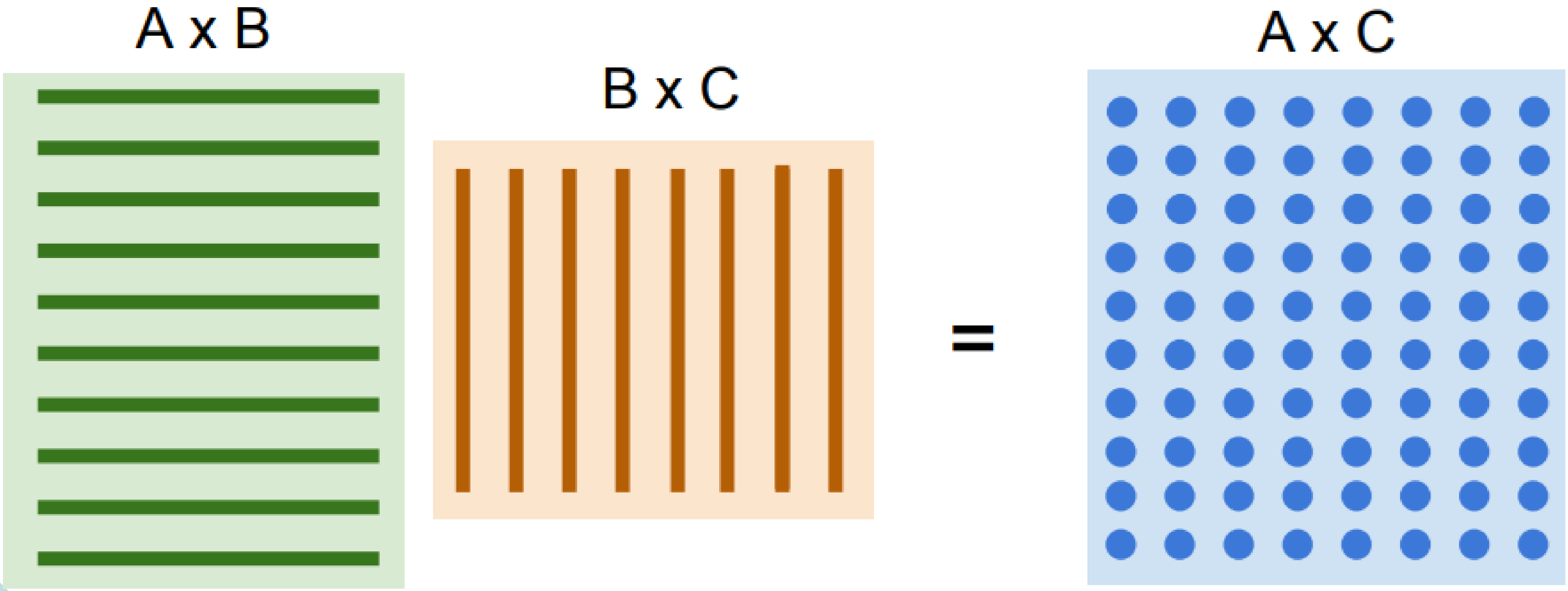
CPU vs GPU in practice

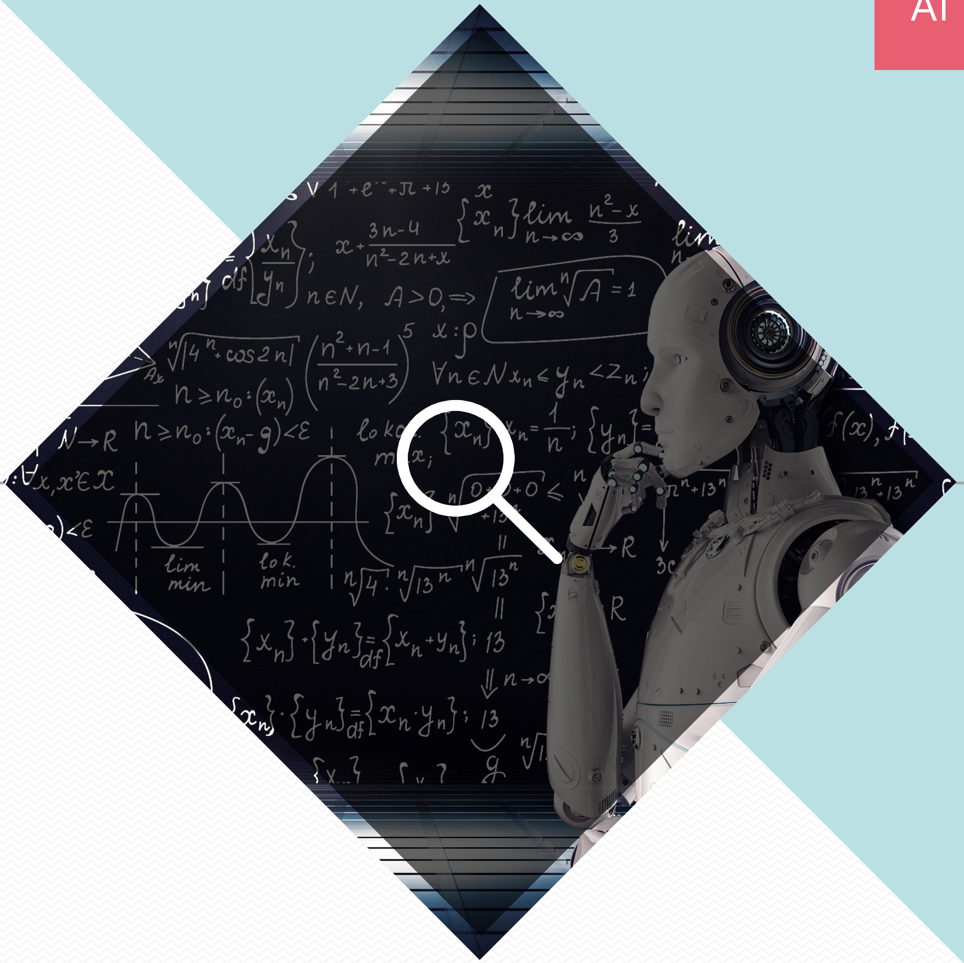
	# Cores	Clock Speed	Memory	Price
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.4 GHz	Shared with system	$339
CPU (Intel Core i7-6950X)	10 (20 threads with hyperthreading)	3.5 GHz	Shared with system	$1723
GPU (NVIDIA Titan Xp)	3840	1.6 GHz	12 GB GDDR5X	$1200
GPU (NVIDIA GTX 1070)	1920	1.68 GHz	8 GB GDDR5	$399

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

Matrix Multiplication





Vector

What is Vector? How is it Defined?

Vector vs Scalar

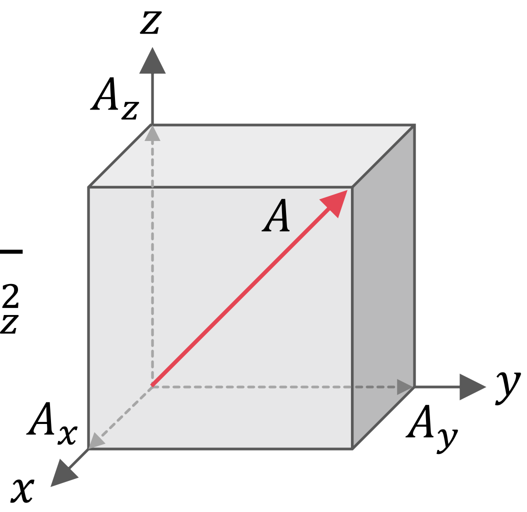
벡터 (vector)	물리량을 표현하는 방법으로 크기와 방향을 모두 갖는 양

벡터(vector)

- 크기와 방향을 모두 갖는 양

예 - 힘
- 속도
- 가속도

$$|A| = A = \sqrt{A_x^2 + A_y^2 + A_z^2}$$



스칼라(scalar)

- 하나의 실수만으로 표시되는, 방향을 갖지 않는 양
- 크기 또는 수치로만 표현될 수 있는 물리량

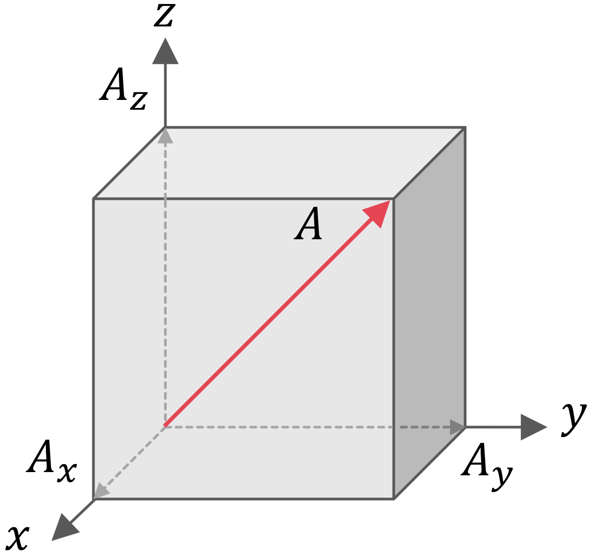
예 5[kg], 30[cm], 20[m^3], 18[℃], 5[V]

- 두 벡터의 스칼라 곱도 스칼라양

Vector의 표현

벡터는 좌표점을 이용하거나 기본 벡터를 이용해 표현 가능

- 스칼라(고딕체)와 구별하여 굵은 글자나 윗부분에 화살표를 붙인 글자로 표시
- 종점 좌표에서 시작점 좌표를 빼는 방식으로 표시



01 좌표점을 이용하여 표시

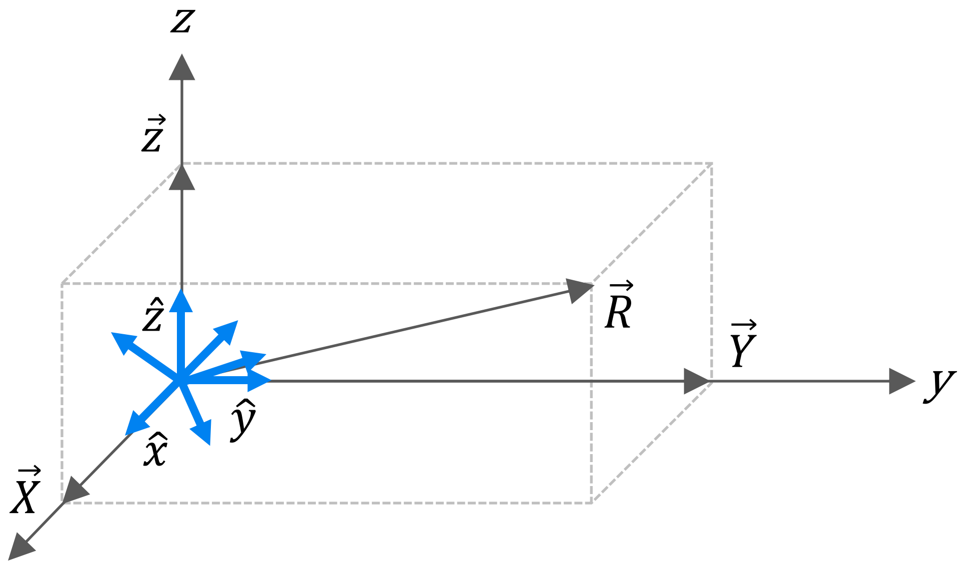
$$\vec{A} = [2, 2, 4] = [2-0, 2-0, 4-0]$$

02 기본 벡터 이용

$$\vec{A} = 2\hat{x} + 2\hat{y} + 4\hat{z}$$

Vector의 크기

단위 벡터 어떤 방향을 가지고 크기가 1인 벡터



벡터를 그 벡터의 크기로 나눈 값

$$\hat{a} = \frac{\vec{A}}{|\vec{A}|}$$

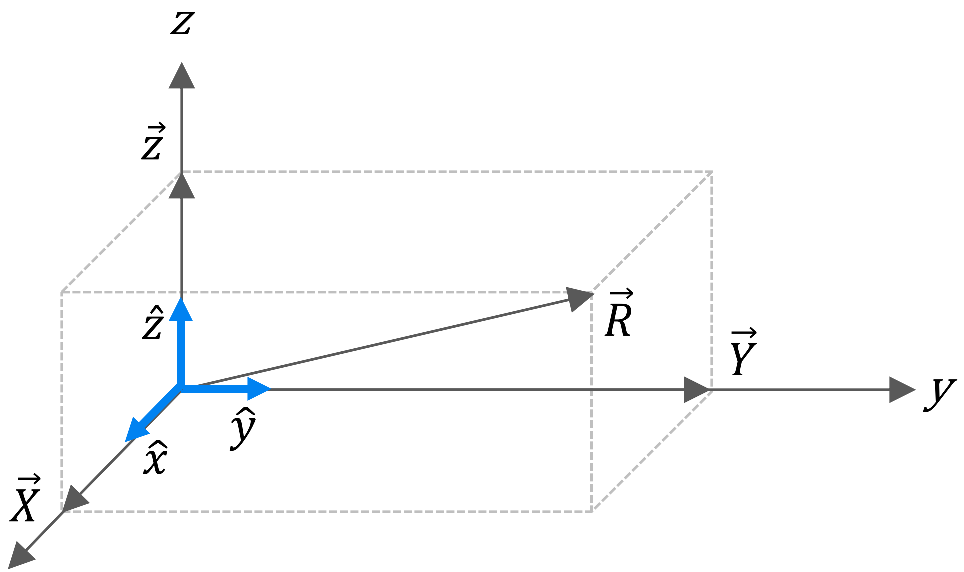
$$|A| = A = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

Vector의 크기

기본벡터

기본 벡터 | 각 x, y, z축 위에 존재하는 단위 벡터

- 통상적으로 소문자로 표시하며, $\hat{x}, \hat{y}, \hat{z}$ 로 표시



기본 벡터도 단위 벡터에 포함됨

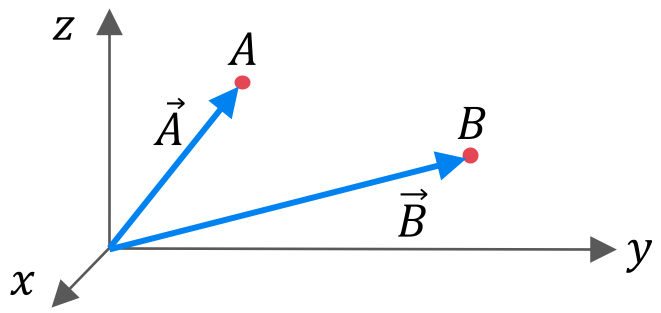
$$\hat{x} = (1, 0, 0)$$
$$\hat{y} = (0, 1, 0)$$
$$\hat{z} = (0, 0, 1)$$

Vector의 연산

벡터의 덧셈과 뺄셈

덧셈/뺄셈은 두 좌표의 각 축의 값을 더하거나 빼서 계산

[벡터의 합성]



$$\vec{A}+\vec{B}=A_x\hat{x}+A_y\hat{y}+A_z\hat{z}+B_x\hat{x}+B_y\hat{y}+B_z\hat{z}=(A_x+B_x)\hat{x}+(A_y+B_y)\hat{y}+(A_z+B_z)\hat{z}$$

$$\vec{A}-\vec{B}=(A_x\hat{x}+A_y\hat{y}+A_z\hat{z})-(B_x\hat{x}+B_y\hat{y}+B_z\hat{z})=(A_x-B_x)\hat{x}+(A_y-B_y)\hat{y}+(A_z-B_z)\hat{z}$$

$$\vec{A}+\vec{B}=[A_x,A_y,A_z]+[B_x,B_y,B_z]=[A_x+B_x,A_y+B_y,A_z+B_z]$$

$$\vec{A}-\vec{B}=[A_x,A_y,A_z]-[B_x,B_y,B_z]=[A_x-B_x,A_y-B_y,A_z-B_z]$$

Vector의 곱셈

- 곱셈(내적) : 점곱(dot product)

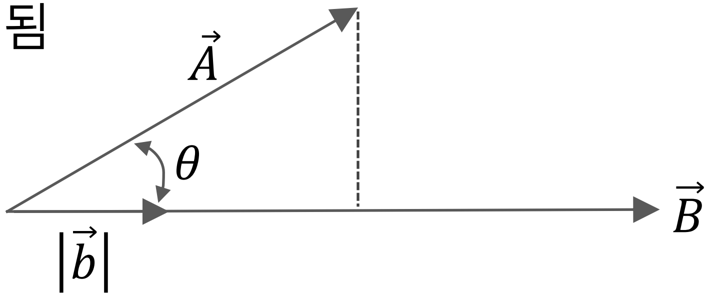
$$A \cdot B = |A||B|cos\theta = A_x B_x + A_y B_y + A_z B_z$$

- 각 기본 $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$ 간의 각도는 90도이므로 내적은 0이 됨

$$A \cdot B = (A_x \hat{x} + A_y \hat{y} + A_z \hat{z}) \cdot (B_x \hat{x} + B_y \hat{y} + B_z \hat{z}) = A_x B_x + A_y B_y + A_z B_z$$

스칼라 곱(scalar product) : 점곱의 다른 말

$\vec{b}$ 가 단위 벡터일 경우는 $\vec{A}$ 의 $\vec{b}$ 방향의 성분 벡터가 됨



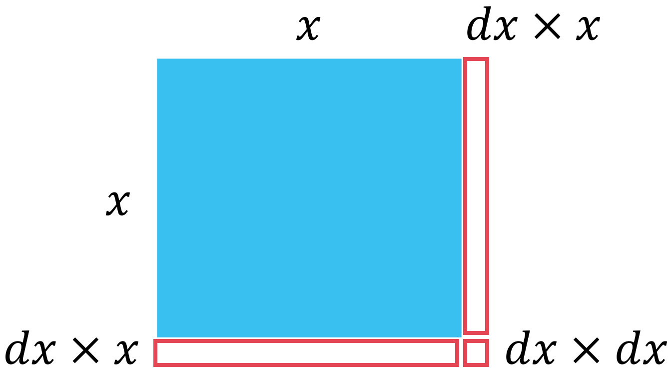
$$cos\theta = \frac{A_x B_x + A_y B_y + A_z B_z}{|A||B|}$$

6) Derivatives

- x의 dx만큼 변화 → f(x)의 변화량 관찰

$$\frac{\partial f(x_0, y_0)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x}$$

$$\boldsymbol{y = x^2}$$



$$\begin{aligned} dy &= (x + dx)^2 - y \\ &= \cancel{x^2} + 2xdx + (dx)^2 \\ &\quad \cancel{-x^2} \\ &= \boxed{2xdx} + (dx)^2 \end{aligned}$$

$$\boldsymbol{y = x^3}$$

$$\begin{aligned} dy &= (x + dx)^3 - y \\ &= \cancel{x^3} + 3x^2dx + 3x(dx)^2 + (dx)^3 \\ &\quad \cancel{-x^3} \\ &= \boxed{3x^2dx} + 3x(dx)^2 + (dx)^3 \end{aligned}$$

1 벡터(vector)

6) **Derivatives** : 1 variable → 2 variable

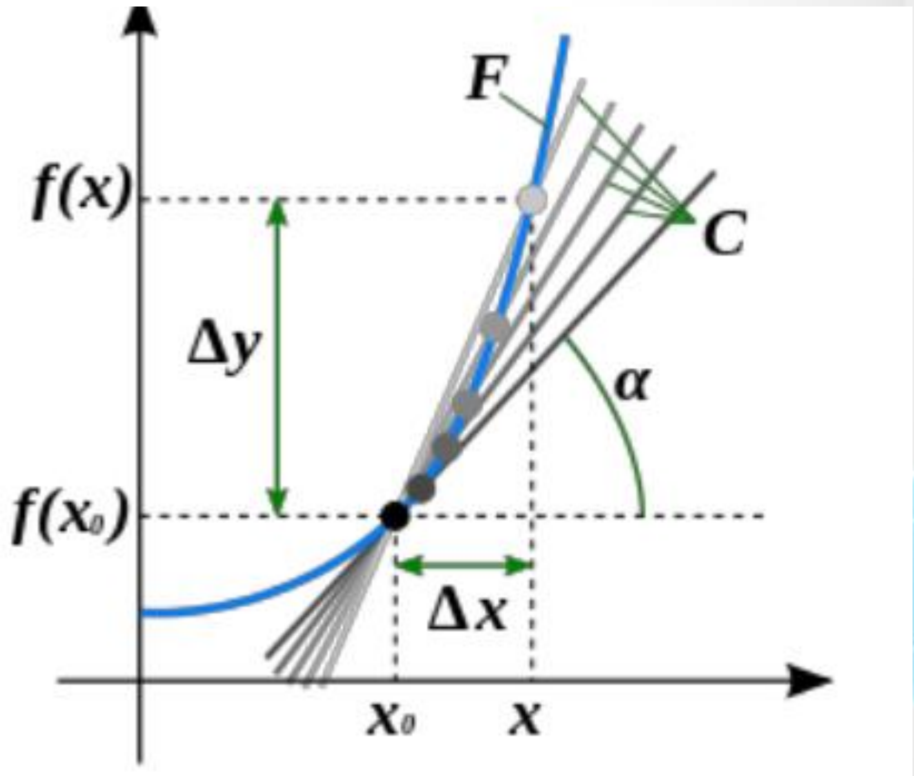
- First order

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

"편미분으로 바로 차원을 늘릴 수 있을까?"

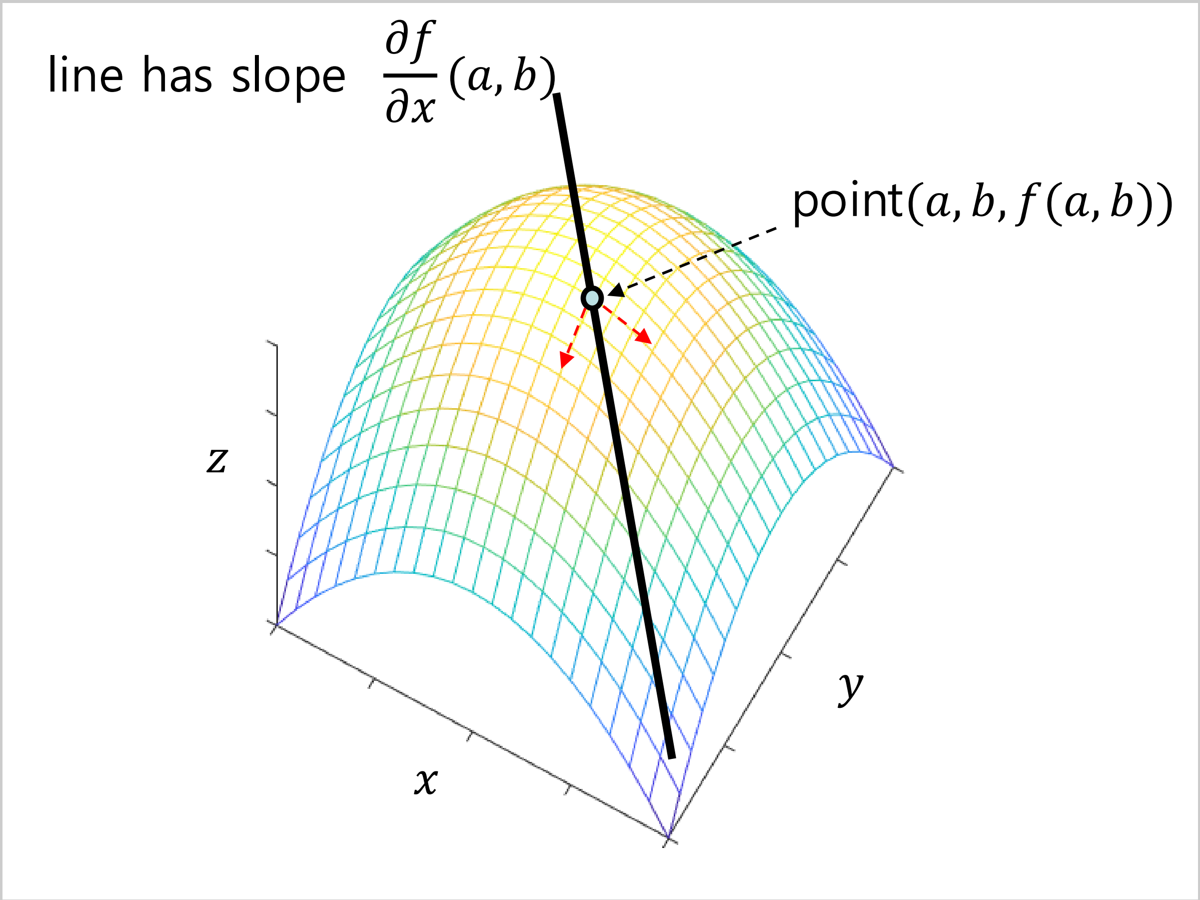
$$\frac{\partial f(x_0, y_0)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x}$$

$$\frac{\partial f(x_0, y_0)}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x_0, y_0 + \Delta y) - f(x_0, y_0)}{\Delta y}$$



1 벡터(vector)

6) **Derivatives** : multivariable of partial derivatives



$$\nabla A(x,y) = \frac{\partial A}{\partial x}\hat{x} + \frac{\partial A}{\partial y}\hat{y}$$

"각 좌표축이
독립적이므로 편미분
사용가능"

$$\frac{\partial f(x_0,y_0)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x, y_0) - f(x_0,y_0)}{\Delta x}$$

$$\frac{\partial f(x_0,y_0)}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x_0, y_0+\Delta y) - f(x_0,y_0)}{\Delta y}$$

1 벡터(vector)

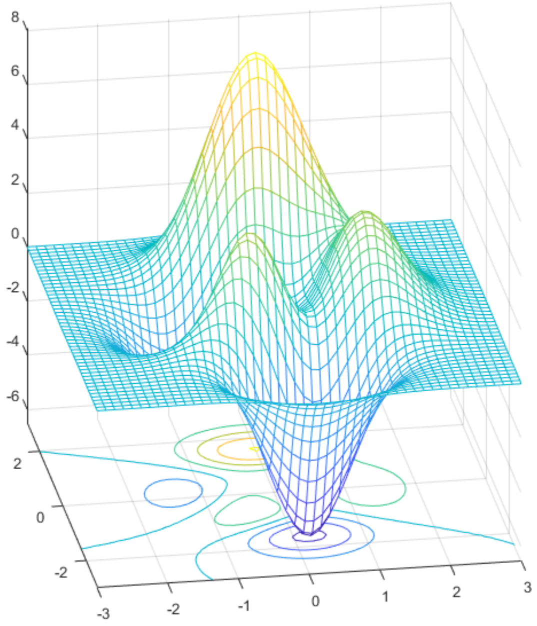
7) Gradient(∇) : Derivatives of vector

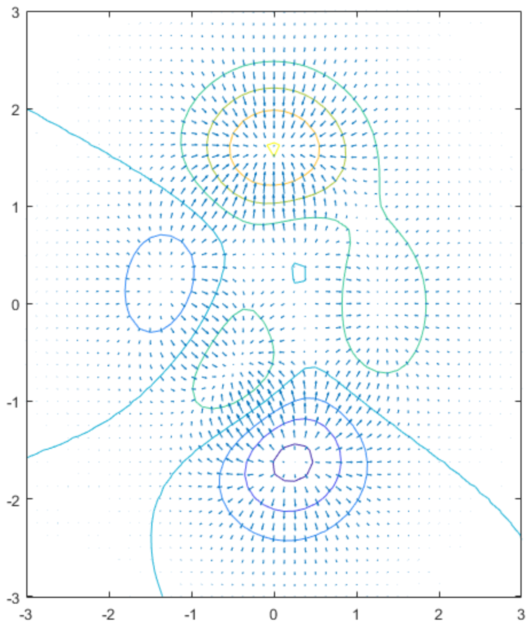
- The gradient is a vector of partial derivatives representing the rate and direction of the steepest ascent of a function.
  - gradient는 편미분값의 벡터 표현임

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

  - n차원에서는

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$



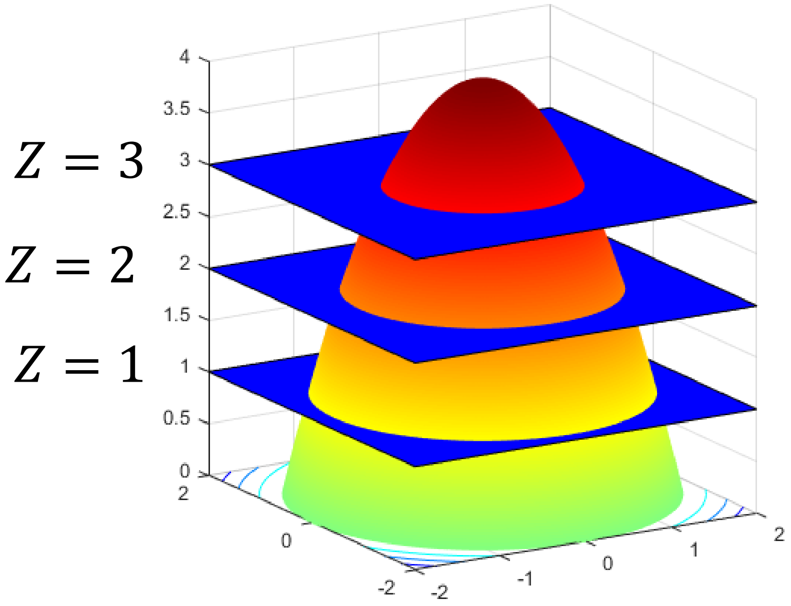


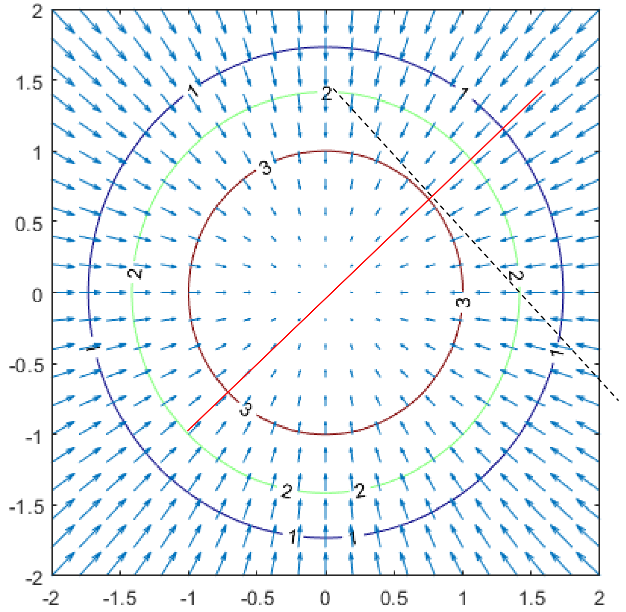
1 벡터(vector)

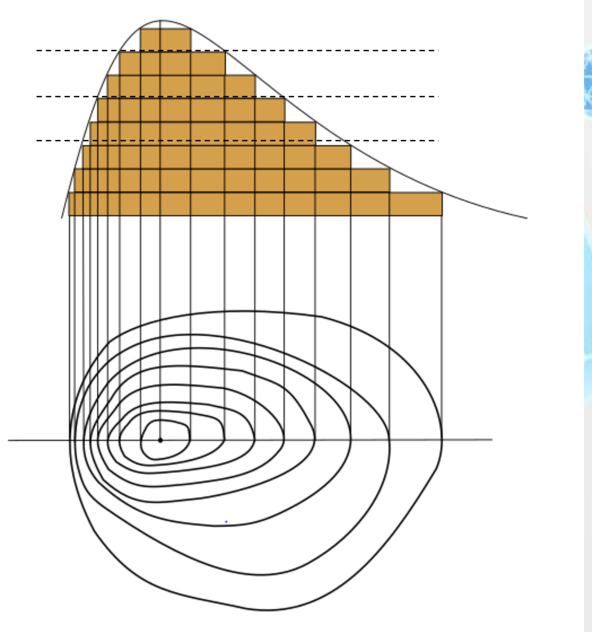
7) Gradient(∇) : Derivatives of vector

$$f(x,y) = 4 - x^2 - y^2$$

- $\boldsymbol{X = 1, y = 1}$에서 가장 가파른 방향은 어느 방향일까?







1 벡터(vector)

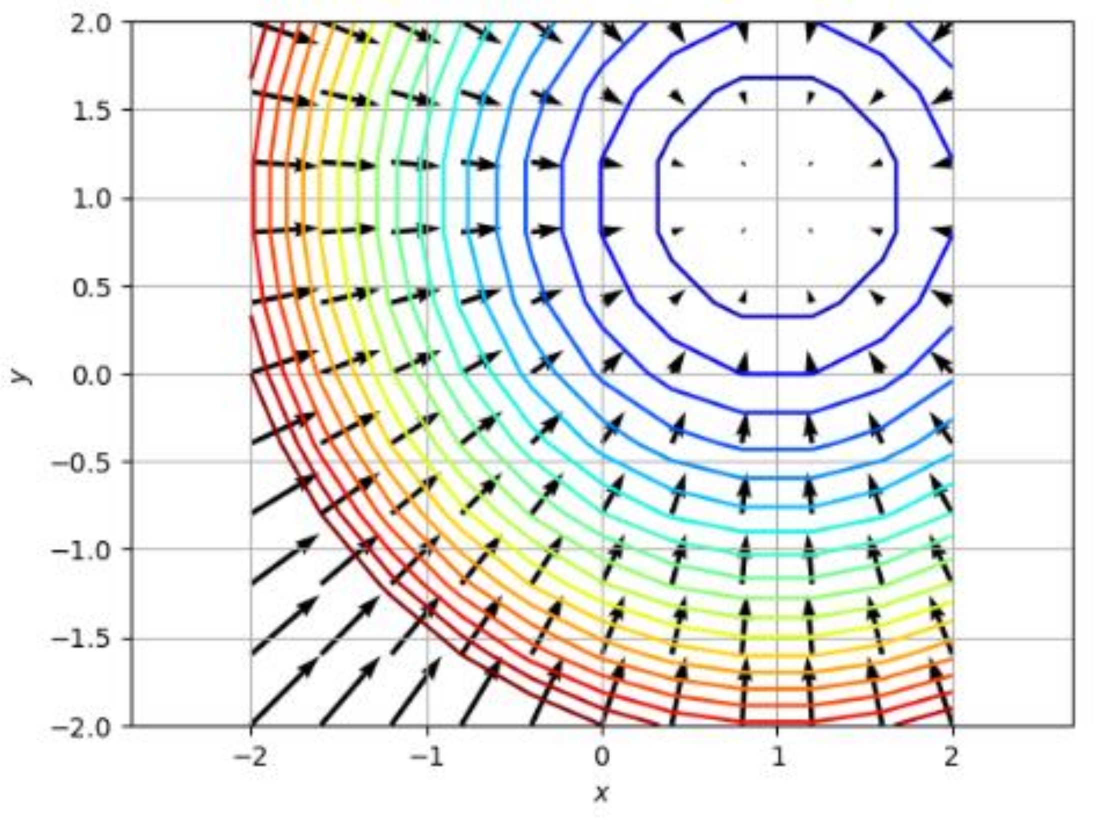
7) Gradient(∇) : Derivatives of vector

```
import numpy as np
import matplotlib.pylab as plt
x = np.linspace(-2,2, 11)
y = np.linspace(-2,2, 11)

print(x)
print(y)

x,y = np.meshgrid(x,y)
print(x)
print(y)

f = lambda x,y : (x-1)**2 + (y-1)**2
z = f(x,y)
print(z)
```

```
grad_f_x = lambda x, y: 2 * (x-1)
grad_f_y = lambda x, y: 2 * (y-1)
```

```
dz_dx = grad_f_x(x,y)
dz_dy = grad_f_y(x,y)
```

```
ax = plt.axes()
ax.contour(x, y, z, levels=np.linspace(0, 10, 20), cmap=plt.cm.jet)
ax.quiver(x, y, -dz_dx, -dz_dy)
ax.grid()
ax.axis('equal')
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
plt.show()
```



1 벡터(vector)

8) Chain rule

$$x \longrightarrow f(x) = \left(x^2+1\right)^2$$

$$x \rightarrow x^2 \rightarrow x^2+1 \rightarrow \left(x^2+1\right)^2$$

$$\frac{d\left(\left(x^2+1\right)^2\right)}{dx} = \frac{d\left(\left(x^2+1\right)^2\right)}{d\left(x^2+1\right)}\frac{d\left(x^2+1\right)}{dx^2}\frac{dx^2}{dx}$$

$$f = \left(x^2+1\right)^2,\ g = x^2+1,\ h = x^2$$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dh}\frac{dh}{dx}$$

$$\frac{d\left(\left(x^2+1\right)^2\right)}{dx} = \frac{df}{dg}\frac{dg}{dh}\frac{dh}{dx} = 2\left(x^2+1\right)\cdot 1\cdot 2x = 4x\left(x^2+1\right)$$

1 벡터(vector)

8) Chain rule

$$\frac{\partial f\left(g\left(h(x)\right)\right)}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial x} = f'\left(g\left(h(x)\right)\right)g'\left(h(x)\right)h'(x) = (f \circ g \circ h)'$$

$$f(x) = \left(x^2+1\right)^2$$

$$f(g) = g^2 \qquad g(h) = h+1 \qquad h(x) = x^2$$

$$f'(g) = 2g \qquad g'(h) = 1 \qquad h'(x) = 2x$$

<u>Direct method</u>

$$f(x) = \left(x^2+1\right)^2 = x^4 + 2x^2 + 1$$

$$f'(x) = 4x^3 + 4x = 4x\left(x^2+1\right)$$

<u>Chain rule</u>

$$f'(x) = 2\left(x^2+1\right)\cdot 1 \cdot 2x$$
$$= 4x\left(x^2+1\right)$$

1 벡터(vector)

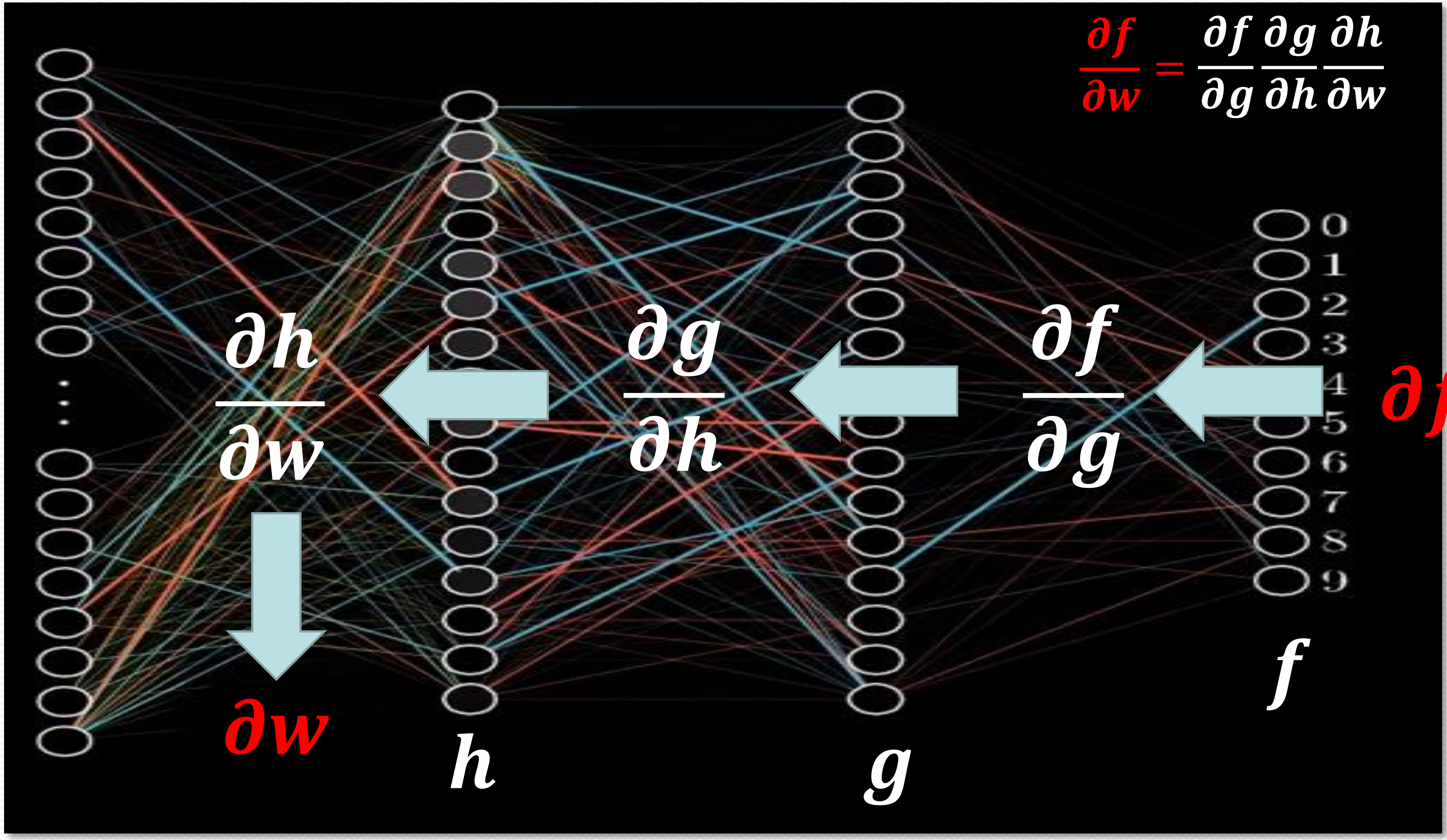
8) Chain rule

$$(f \circ g \circ h)' = \frac{\partial f(g(h))}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial x}$$

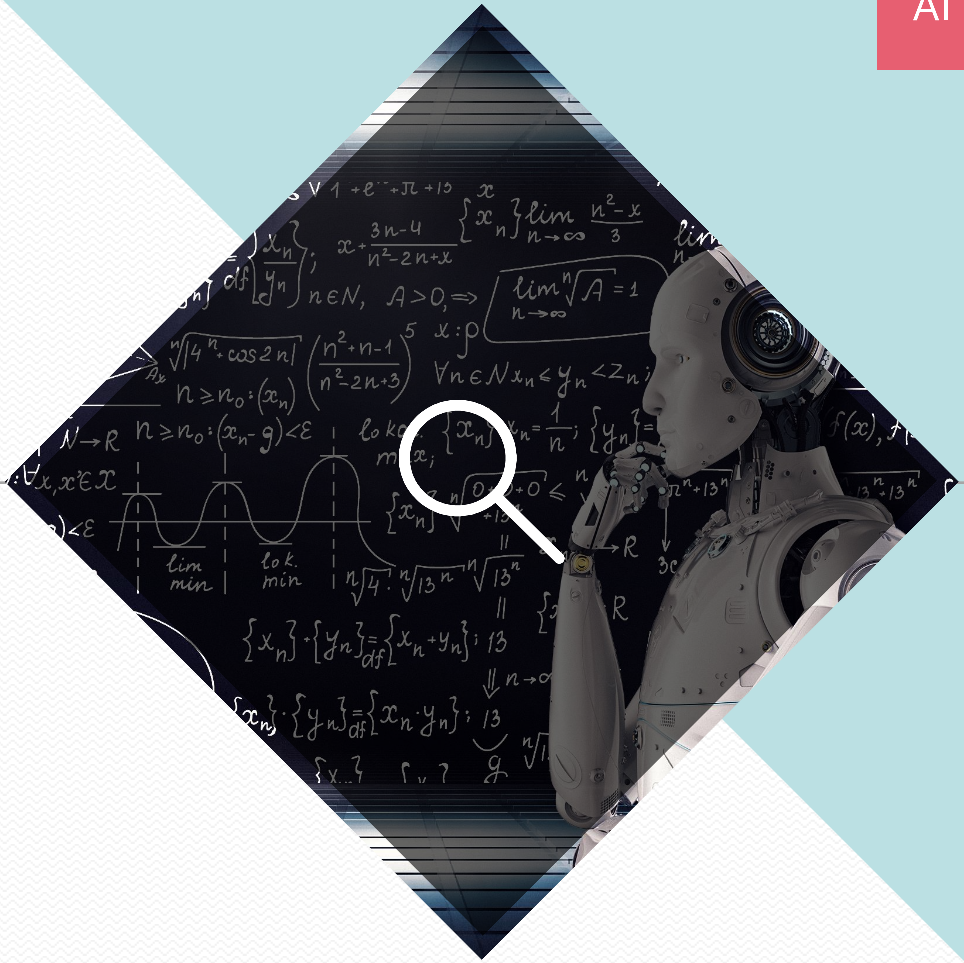
- 체인룰은 변수가 여러 개일 때, 어떤 변수에 대한 다른 변수의 변화율을 알아내기 위해 쓰임
- Back propagation에서 주로 사용됨.

Error Back Propagation

CALCULATE THE ∂w from ∂f



AI TRAINING SERIES



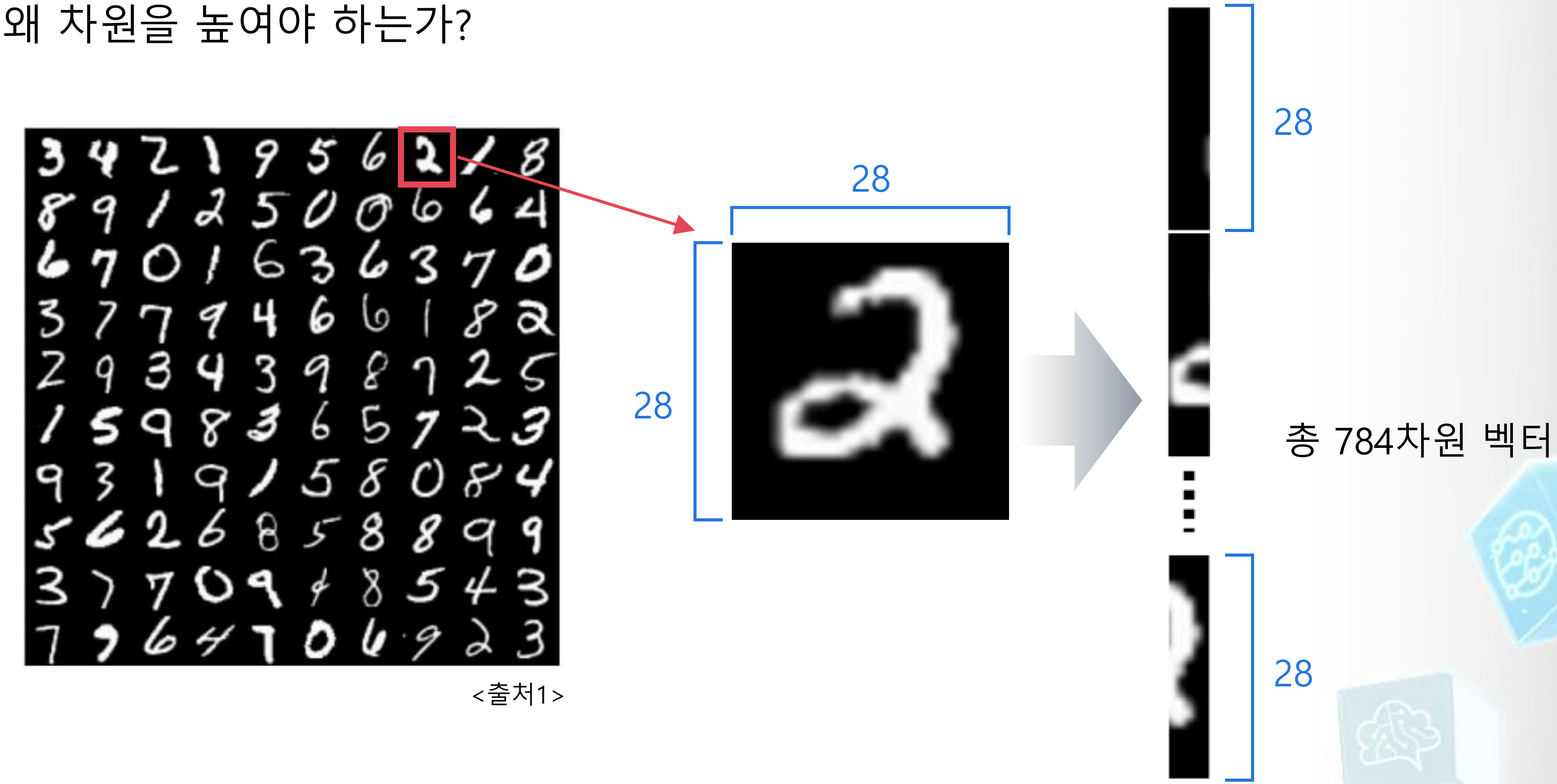
Matrix

차원의 확장

2 matrix

1) 차원의 확장

- 왜 차원을 높여야 하는가?



<출처1>

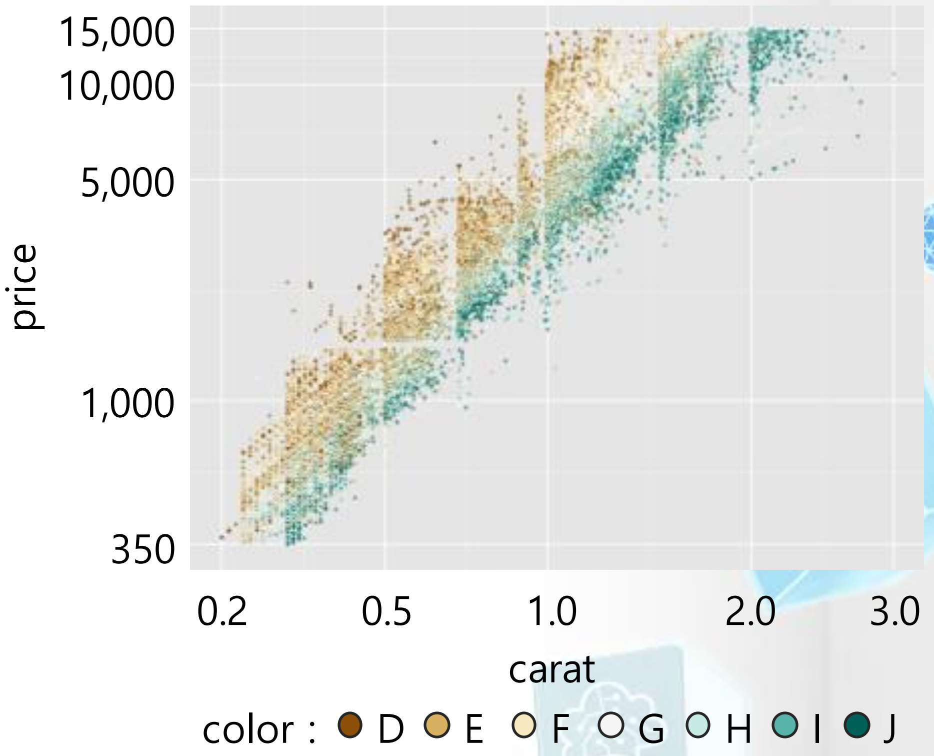
1) 차원의 확장

- 왜 차원을 높여야 하는가?
  - 다이아몬드의 가격 예측

1 캐럿
2 컷
3 색깔
4 투명도
5 깊이
6 x방향 길이
7 y방향 길이
8 z방향 길이

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49

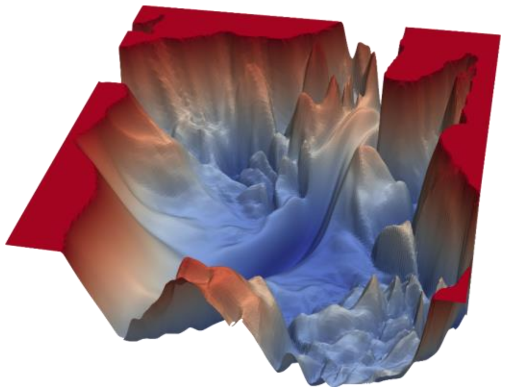
[price(log 10) by cube-root of carat and color]



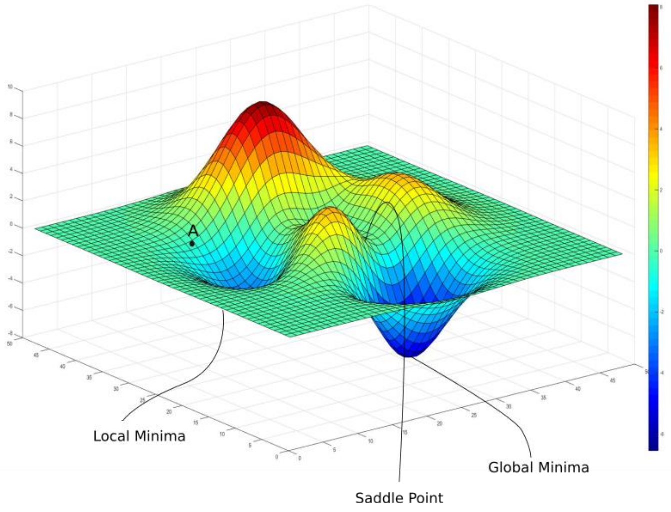
<출처2>

1) 차원의 확장

- Local minima



loss contour of a VGG-56 DNN's loss function



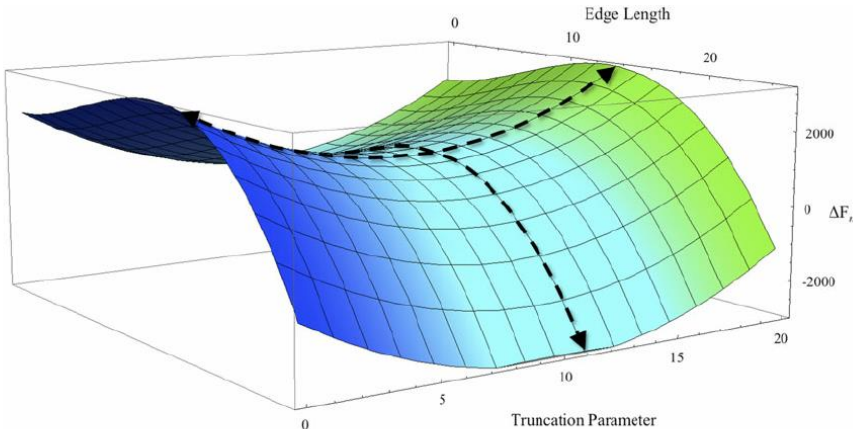
- Gradient of local and global minima is zero
- Improper initialization point may cause convergence to a local minima – you're doomed!

2 matrix

1) 차원의 확장

- Multi dimension

Saddle Points



- A minima in one direction, a maxima in another direction
- Occurs where two maxima meet

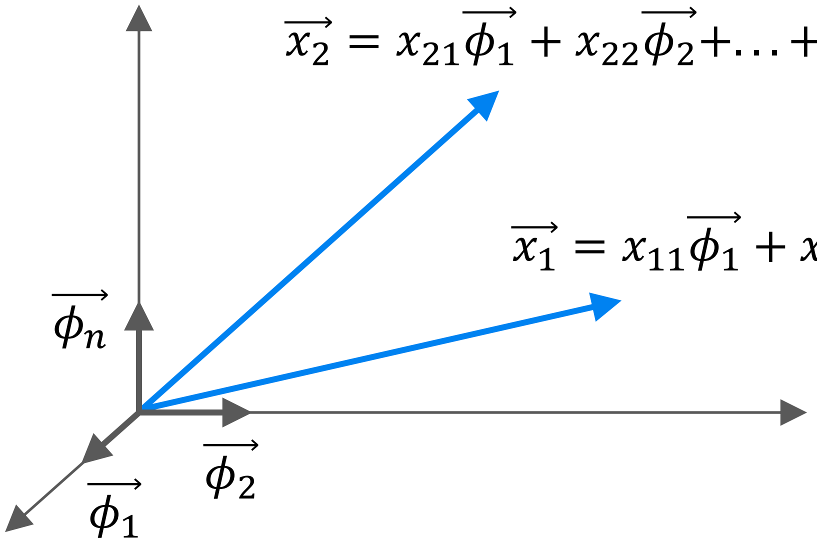
1) 차원의 확장

- 다차원 벡터의 내적
  - 두 벡터 $\overrightarrow{x_1}, \overrightarrow{x_2}$ 의 내적은 두 벡터 간의 projection 에너지의 적분으로 정의됨
  - 두 벡터가 신호이고, 같은 신호일 경우 신호의 에너지가 됨

$$\langle \overrightarrow{x_1}, \overrightarrow{x_2} \rangle = x_{11}x_{21} + x_{12}x_{22} + \ldots + x_{1n}x_{2n}$$

$$\overrightarrow{x_2} = x_{21}\overrightarrow{\phi_1} + x_{22}\overrightarrow{\phi_2} + \ldots + x_{2n}\overrightarrow{\phi_n}$$

$$\overrightarrow{x_1} = x_{11}\overrightarrow{\phi_1} + x_{12}\overrightarrow{\phi_2} + \ldots + x_{1n}\overrightarrow{\phi_n}$$



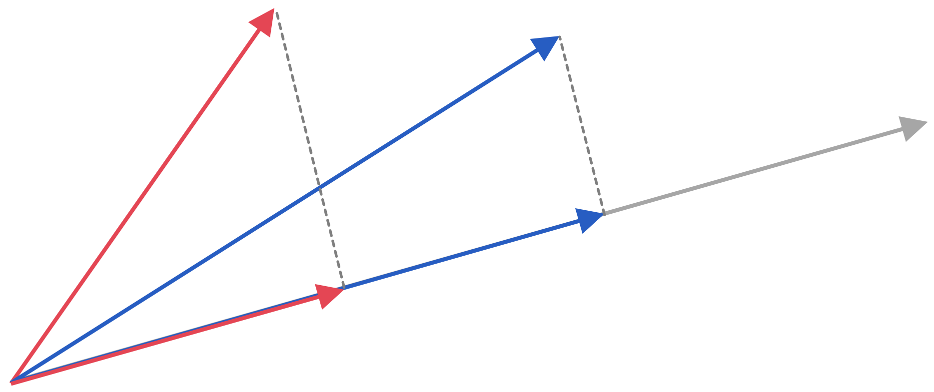
2 matrix

1) 차원의 확장

- 방향성분 추출
  - $\overrightarrow{x_n}$ 에 관한 기저함수 $\overrightarrow{\emptyset_n}$의 가중치 혹은 계수는 $\overrightarrow{x_n}$ 와 $\overrightarrow{\emptyset_n}$의 내적으로 추출됨
  - $\overrightarrow{x_n}$ 와 $\overrightarrow{\emptyset_n}$ 의 내적은 $\overrightarrow{x_n}$ 의 $\overrightarrow{\emptyset_n}$ 방향의 정사형 성분으로 $\overrightarrow{\emptyset_n}$ 방향의 가중치로 해석됨

$$\vec{x} = x_1\overrightarrow{\phi_1} + x_2\overrightarrow{\phi_2} + \ldots + x_n\overrightarrow{\phi_n}$$

$$\overrightarrow{x_n} = \langle \overrightarrow{x_n}, \overrightarrow{\phi_m} \rangle$$



1) 차원의 확장

- 직교성과 기본 벡터(basis function)
  - 직교성(orthogonal & orthonormal)
    - 신호 $x(t)$에 관한 식의 기저함수를 $\overrightarrow{\phi_n}$라고 할 때, 다른 기저함수 간의 내적이 0일 경우 orthogonal이라고 함
    - 동일 기저함수 간의 내적이 1일 경우, orthonormal이라고 함

$$\vec{x} = x_1\overrightarrow{\phi_1} + x_2\overrightarrow{\phi_2} + \ldots + x_n\overrightarrow{\phi_n}$$

$$\langle\overrightarrow{\phi_n}, \overrightarrow{\phi_m}\rangle = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases} \Rightarrow \text{Set } \{\overrightarrow{\phi_n}\} \text{ are Orthonormal}$$

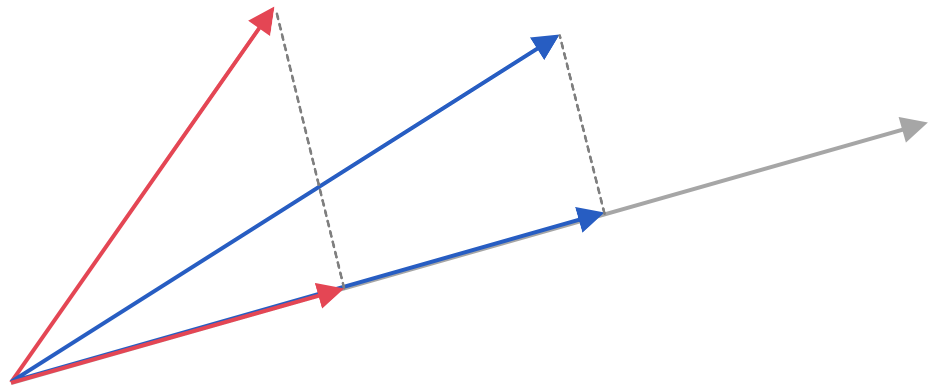
2 matrix

1) 차원의 확장

- sin/cos함수 기본 벡터
  - 지수 기저함수(파동의 주파수 기본으로 모델링)

$$\vec{x} = x_1\overrightarrow{\phi_1} + x_2\overrightarrow{\phi_2} + \ldots + x_n\overrightarrow{\phi_n}$$

$$\overrightarrow{x_n} = \langle \overrightarrow{x_n}, \overrightarrow{\phi_m} \rangle$$



$$\phi_m(t) = e^{-i2\pi f_0 t}$$

$$e^{-i2\pi f_0 t} = \cos 2\pi f_0 t + \mathrm{i} \sin 2\pi f_0 t \qquad f_0 = {}^{1}/_{T_0}$$

2 matrix

1) 차원의 확장

- 다차원 벡터의 내적
  - 벡터 간의 연산이므로 차원이 같아야 함
    - 내적의 결과는 스칼라!!

$$\langle x, y \rangle = x^T y$$

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, y = \begin{bmatrix} 5 \\ -2 \\ 1 \end{bmatrix} \qquad x^T y = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ -2 \\ 1 \end{bmatrix} = 4$$

np.dot(A,x)

1) 차원의 확장

- matrix 확장
  - matrix로 확장하면 여러 벡터의 내적을 한꺼번에 진행 가능

$$\langle x, y\rangle = x^T y$$

$$x^T y = \begin{bmatrix}1 & 2 & 3\end{bmatrix}\begin{bmatrix}5\\-2\\1\end{bmatrix} = 4$$

$$\boxed{\begin{bmatrix}1 & 2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{bmatrix}}\begin{bmatrix}5\\-2\\1\end{bmatrix} = \begin{bmatrix}4\\4\\4\end{bmatrix}$$

matrix

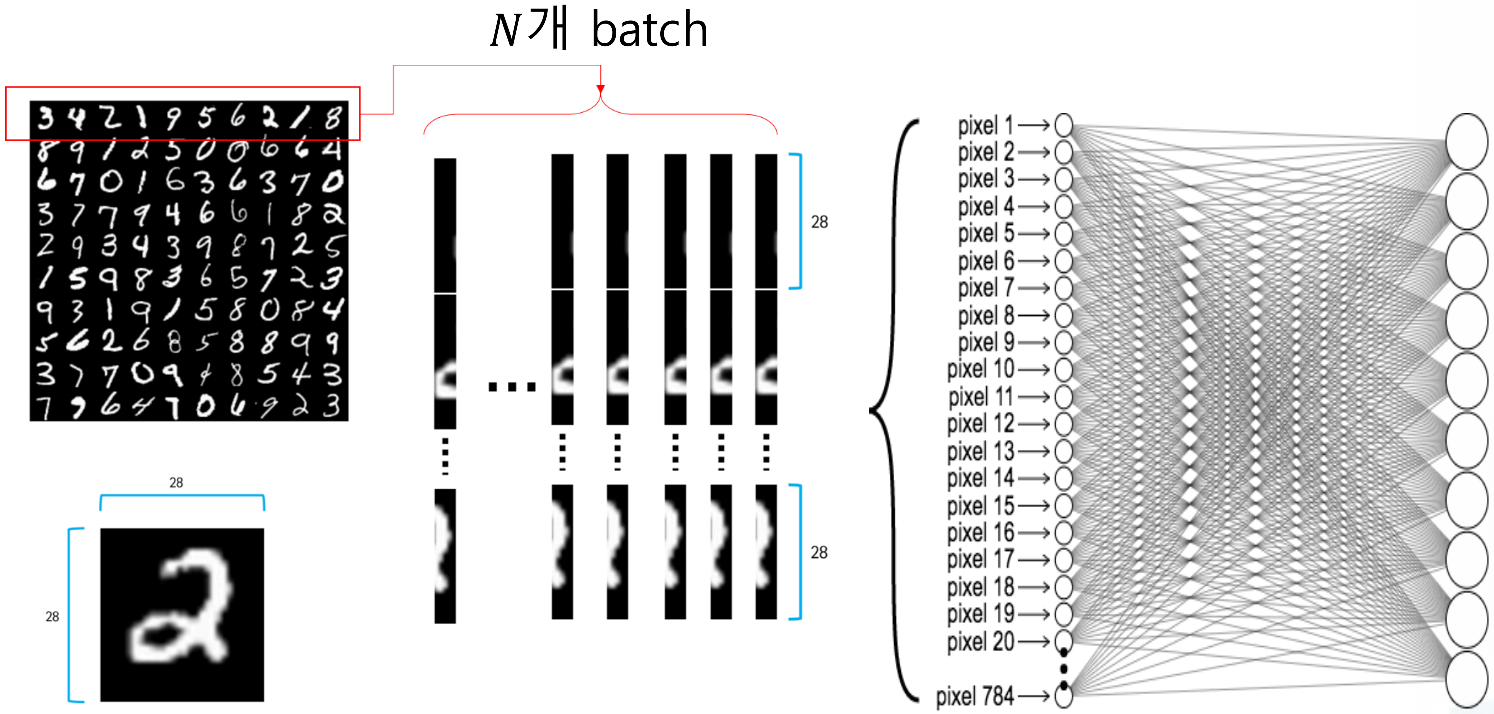
$$\underbrace{\begin{bmatrix}\mathbf{1} & \mathbf{2} & \mathbf{3}\\\mathbf{1} & \mathbf{2} & \mathbf{3}\\\mathbf{1} & \mathbf{2} & \mathbf{3}\end{bmatrix}}_{\text{matrix}}\underbrace{\begin{bmatrix}\mathbf{5} & \mathbf{5} & \mathbf{5}\\\mathbf{-2} & \mathbf{-2} & \mathbf{-2}\\\mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}}_{\text{matrix}} = \underbrace{\begin{bmatrix}\mathbf{4} & \mathbf{4} & \mathbf{4}\\\mathbf{4} & \mathbf{4} & \mathbf{4}\\\mathbf{4} & \mathbf{4} & \mathbf{4}\end{bmatrix}}_{\text{matrix}}$$

np.dot(A,x)

2 matrix

1) 차원의 확장

- matrix 확장
  - matrix로 확장하면 여러 벡터의 내적을 한꺼번에 진행 가능



$$W\begin{bmatrix} x_1 \\ \vdots \\ x_{784} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{10} \end{bmatrix} + \dots + W\begin{bmatrix} x_N \\ \vdots \\ x_{784} \end{bmatrix} = \begin{bmatrix} y_N \\ \vdots \\ y_{10} \end{bmatrix} \Rightarrow W\begin{bmatrix} x_1 & \cdots & x_N \\ \vdots & \cdots & \vdots \\ x_{784} & \cdots & x_{N,784} \end{bmatrix} = \begin{bmatrix} y_1 & \cdots & y_N \\ \vdots & \cdots & \vdots \\ y_{10} & \cdots & y_{N,10} \end{bmatrix}$$

2 matrix

2) matrix 곱셈

Numpy lib array의 A.dot(B)를 이용해 A와 B의 행렬곱(내적)을 구할 수 있음

행! 렬(열)!

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \boxed{a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 2 & 0 \\ 9 & 5 & 0 & 0 \\ 4 & 0 & 2 & 4 \\ 6 & 1 & 8 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 4 \cdot 2 + 2 \cdot 3 + 0 \cdot 4 \\ 9 \cdot 1 + 5 \cdot 2 + 0 \cdot 3 + 0 \cdot 4 \\ 4 \cdot 1 + 0 \cdot 2 + 2 \cdot 3 + 4 \cdot 4 \\ 6 \cdot 1 + 1 \cdot 2 + 8 \cdot 3 + 3 \cdot 4 \end{bmatrix} = \begin{bmatrix} 15 \\ 19 \\ 26 \\ 44 \end{bmatrix}$$

2 matrix

3) matrix 선형 방정식

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$Ax = b$$

b=np.matmul(A,x)

b=A@x

4) 역행렬

역행렬을 이용하여 선형 방정식의 해를 찾을 수 있음(np.linalg.inv(A))

- 원래행렬 A에 역행렬 A^{-1}를 곱하면 단위행렬(I)을 얻을 수 있음
- 역행렬은 np.linalg.inv(A)를 이용하여 구할 수 있음
- 판별식(determinant)값이 0이면 역행렬은 존재하지 않음
- 고차원 행렬의 판별식을 구하는 방법은 np.linalg.det를 사용함
- 단위행렬은 판별식이 1이고, 어떤 행렬을 곱해도 자기 자신이 나오는 행렬임

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$AA^{-1} = A^{-1}A = I$$

(np.linalg.inv(A))

2 matrix

4) 역행렬

- 가우스-조단 소거법

$$\begin{matrix} x+3y=5 \\ 3x+5y=11 \end{matrix} \rightarrow \begin{matrix} 3x+9y=15 \\ 3x+5y=11 \end{matrix} \rightarrow \begin{matrix} 3x+9y=15 \\ 4y=4 \end{matrix} \rightarrow \begin{matrix} \frac{1}{3}x+y=\frac{5}{3} \\ y=1 \end{matrix} \rightarrow \begin{matrix} \frac{1}{3}x=\frac{2}{3} \\ y=1 \end{matrix} \rightarrow \begin{matrix} x=2 \\ y=1 \end{matrix}$$

$$\left[\begin{array}{cc|c} 1 & 3 & 5 \\ 3 & 5 & 11 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 3 & 9 & 15 \\ 3 & 5 & 11 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 3 & 9 & 15 \\ 0 & 4 & 4 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} \frac{1}{3} & 1 & \frac{5}{3} \\ 0 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 0 & 2 \\ 0 & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{cc|cc} a & b & 1 & 0 \\ c & d & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|cc} a & b & 1 & 0 \\ 0 & d-\frac{bc}{a} & -\frac{c}{a} & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|cc} 1 & \frac{b}{a} & 1 & 0 \\ 0 & 1 & -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{array}\right] \rightarrow \left[\begin{array}{cc|cc} 1 & 0 & \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ 0 & 1 & -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{array}\right]$$

4) 역행렬

역행렬을 이용하여 선형 방정식의 해를 찾을 수 있음(np.linalg.inv(A))

$$A^{-1} = \frac{1}{Det(A)} \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}$$

$$\boldsymbol{C_{ij} = (-1)^{i+j} Det(M_{ij})}$$

$$M_{ij} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix}$$

i th row 제외

i th column 제외

2 matrix

5) 전치행렬

- matrix A의 transpose

$$\begin{bmatrix} 4 & 5 & 2 & 1 \\ 2 & 3 & 8 & 0 \\ 1 & 0 & 7 & 2 \end{bmatrix}^T = \begin{bmatrix} 4 & 2 & 1 \\ 5 & 3 & 0 \\ 2 & 8 & 7 \\ 1 & 0 & 2 \end{bmatrix}$$

- symmetric matrix A

$$A^T = A$$

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 0 \\ 1 & 0 & 7 \end{bmatrix}^T = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 0 \\ 1 & 0 & 7 \end{bmatrix}$$

$$(A_{ij})^T = A_{ji}$$

A.T

1) 차원의 확장

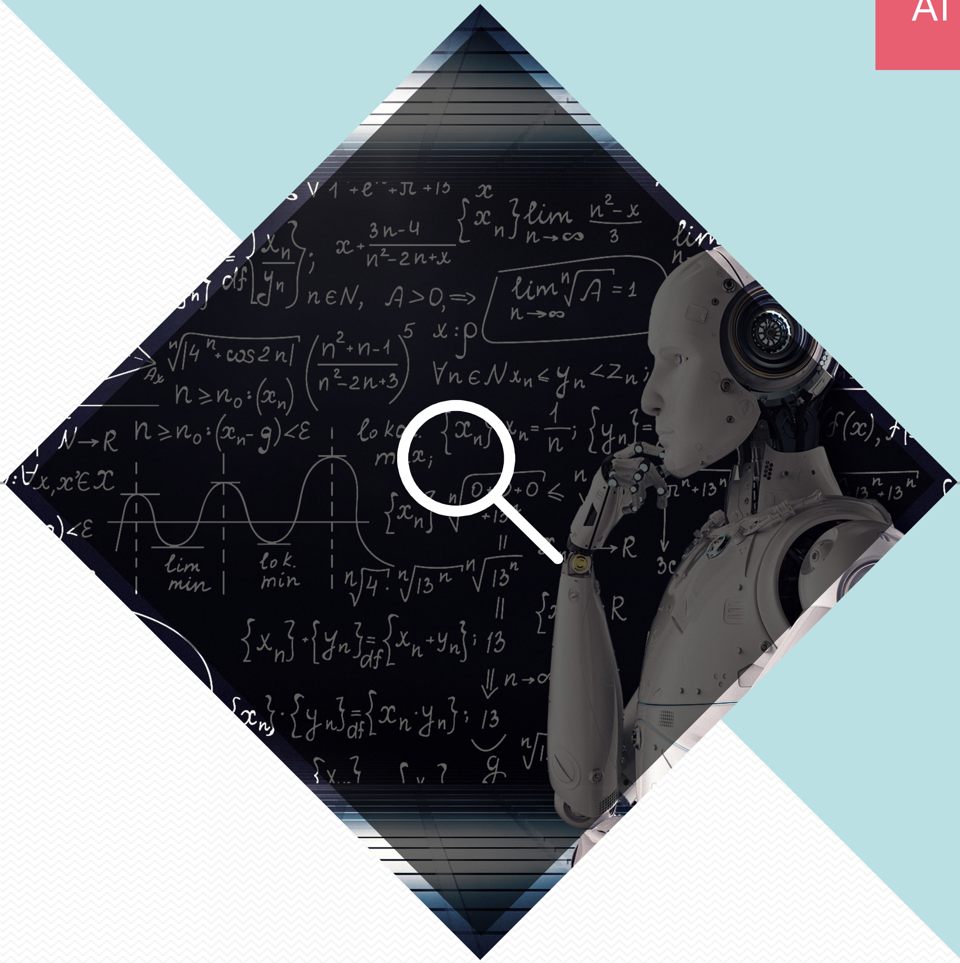
- norm

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}} \qquad \text{for } p \in \mathbb{R}, \mathrm{p} \geq 1.$$

np.linalg.norm(A,ord=1)

L1 놈	L2 놈	무한 놈
벡터와 원소의 절대값의 합	유클리디안 거리로 계산된 백터의 길이	벡터의 원소 중 절대값이 가장 큰 값

AI TRAINING SERIES

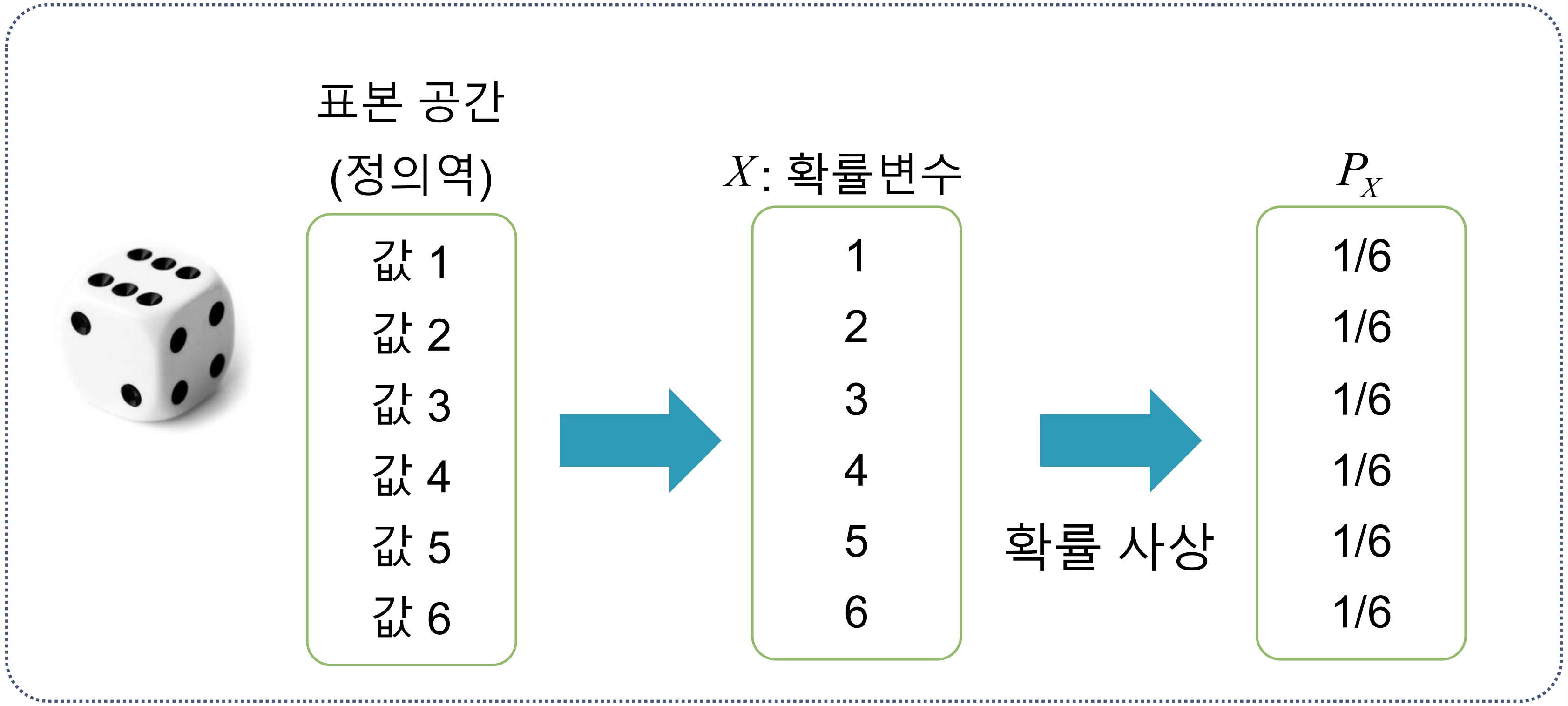


Entropy

What is Vector? How is it Defined?

2 Entropy

1) 확률



- 실험 결과들에 대해 어떤 규칙을 정하여 값을 부여하는데, 이러한 규칙을 확률변수라 함
- 가능한 모든 상황에서 어떤 이벤트가 발생할 수 있는 빈도수가 확률

2 Entropy

2) 평균과 분산

평균

$$m_X = E\left[X^n\right] = \sum_{i=-\infty}^{\infty} i^n P\left(X = i\right)$$

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

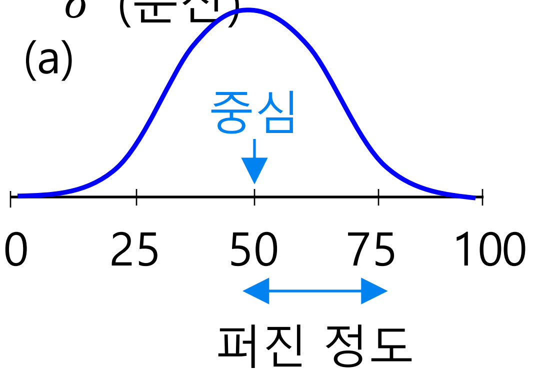
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

분산

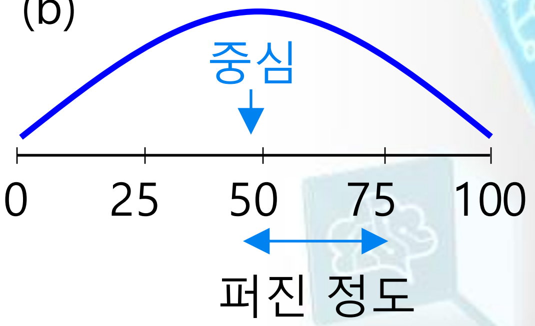
$$\begin{aligned}\sigma^2 &= Var\left[X\right] = E\left[\left(X - m\right)^2\right] \\ &= E\left[X^2 - 2mX + m^2\right] \\ &= E\left[X^2\right] - E\left[2mX\right] + E\left[m^2\right] \\ &= E\left[X^2\right] - m^2\end{aligned}$$

- 퍼진정도 : σ(표준편차), σ^2(분산)
- 중심 : m(평균)

(a)



(b)



2 Entropy

2) 평균과 분산

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.mean(speed)

print(x)
```

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

```
import numpy

speed = [99,86,87,88,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

```
from scipy import stats

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = stats.mode(speed)

print(x)
```

2 Entropy

2) 평균과 분산

```python
import numpy

speed = [32,111,138,28,59,77,97]

x = numpy.std(speed)

print(x)
```

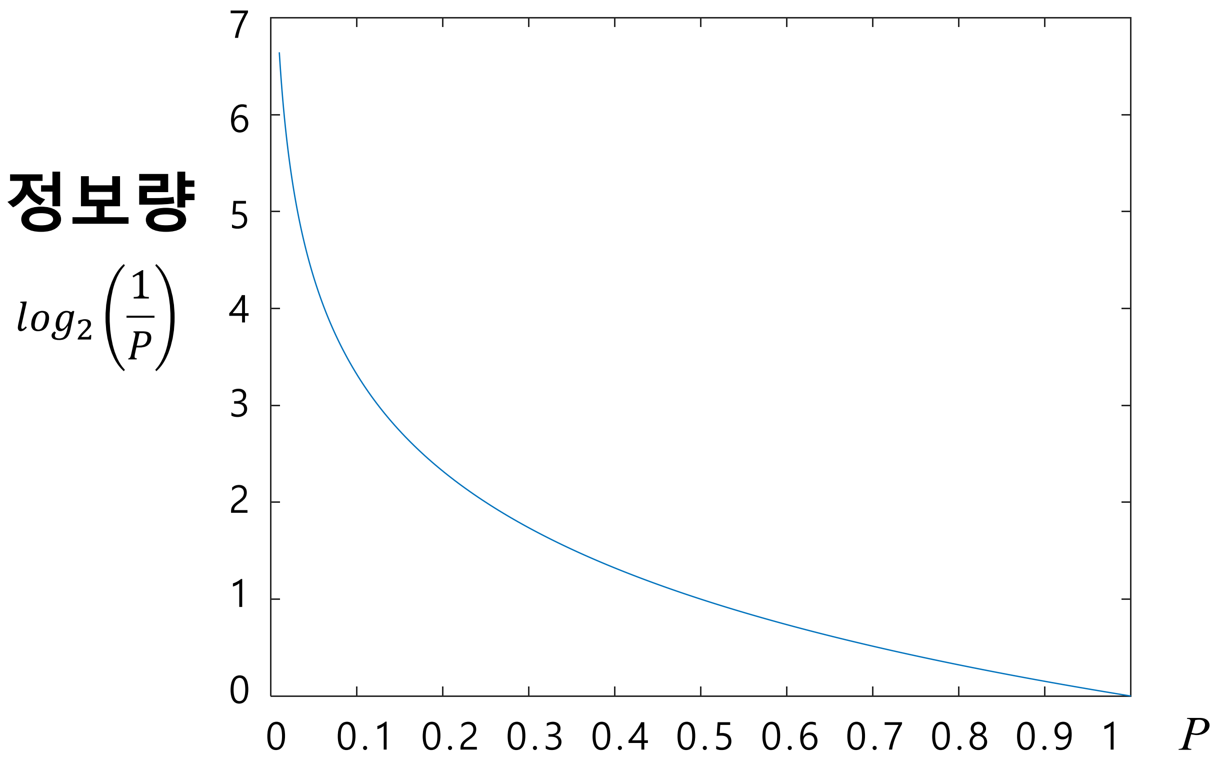
```python
import numpy

speed = [32,111,138,28,59,77,97]

x = numpy.var(speed)

print(x)
```

2 Entropy

3) 소스 심볼의 정보량

- 소스 심볼에 대한 정보량 I 는 심볼의 발생 확률(의외성) P로부터 계산됨

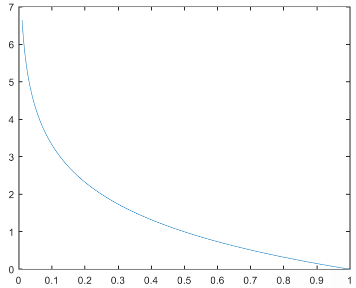
$$p = \frac{1}{2^I} \quad \Rightarrow \quad I = log_2\left(\frac{1}{P}\right) = -log_2 P \text{ [비트]}$$



2 Entropy

3) 소스 심볼의 정보량

- 소스 심볼의 정보량($log_2\left(\frac{1}{P}\right)$)
  - $X_1, X_2, X_3, \dots, X_M$으로 나타내어지는 M개의 심볼이 있는 경우
  - 심볼이 일어날 수 있는 확률 : $P_1, P_2, P_3, \dots, P_M$
  - 심볼 X_1이 M개의 총 정보 속에서 차지하는 비율 : $-P_1 log_2 P_1$
  - **심볼의 평균 정보량**
    - 엔트로피(entropy, 무질서도)로 표현
    - 샤논(Shannon)이 수학적으로 정의



$$E(X) = \sum_{m=0}^{M-1} p_m \log_2(1/p_m) = -\sum_{m=0}^{M-1} p_m \log_2 p_m \text{ [비트/심볼]}$$

- Entropy : $-\sum_{m=0}^{M-1} p_m \log_2 p_m$
- Cross-Entropy: $-\sum_{m=0}^{M-1} p_m \log_2 q_m$
- KL-divergence: $-\sum_{m=0}^{M-1} p_m \log_2 p_m + \sum_{m=0}^{M-1} p_m \log_2 q_m$

- 전혀 정보를 가지지 못하는 $M = 1$인 경우 심볼의 확률은 1이고, $X = 0$

- $\boxed{C(H(x), y) = -y * log(H(X)) - (1-y) * log(1 - H(X))}$

3) 소스 심볼의 정보량 : Logistic regression 의 cost 와 각 분류모델의 기대값

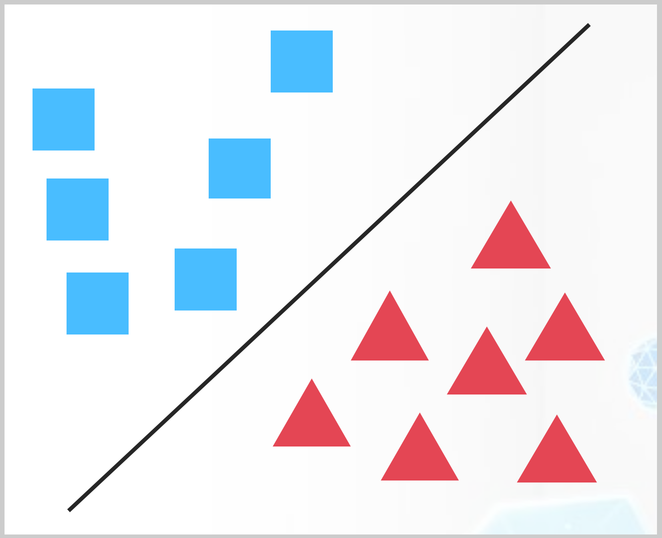
- Cost function with Condition

$$C(H(x), y) = \begin{cases} -\log(H(X)) & if\ y = 1 \\ -\log(1 - H(x)) & if\ y = 0 \end{cases}$$

single function expression

$$C(H(x), y) = -y * log(H(X)) - (1 - y) * log(1 - H(X))$$

Binary crossentropy func.



2 Entropy

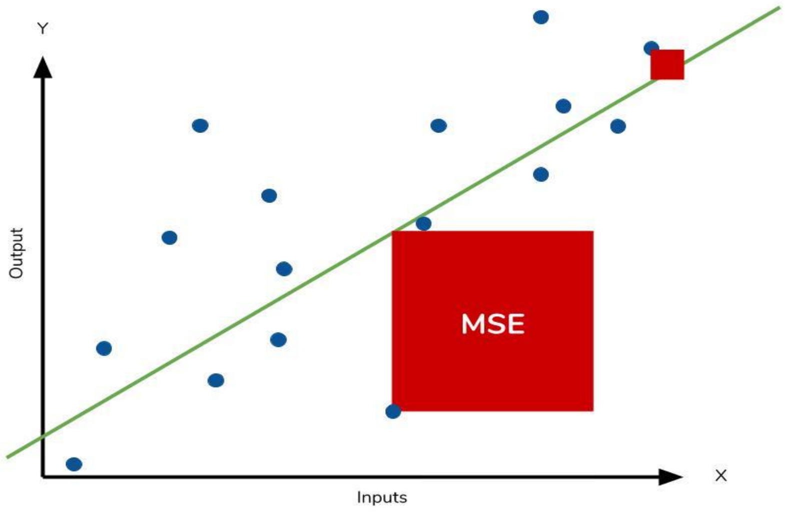
4) Cost function

01 MSE(Mean Squared Error)

- 평균 제곱 오차는 선형 회귀의 손실 함수
- 연속형 변수를 예측할 때 사용

```
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
```

```
model.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredError(), metrics=['mse'])
```



https://www.tensorflow.org/api_docs/python/tf/keras/losses

2 Entropy

4) Cost function

02 이진 크로스 엔트로피(binary cross-entropy)

- 이항 교차 엔트로피라고도 부르는 손실 함수
- 출력층에서 시그모이드 함수를 사용하는 이진 분류(binary classification)의 경우 binary_crossentropy를 사용함

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
```

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer='adam', metrics=['acc'])
```

https://www.tensorflow.org/api_docs/python/tf/keras/losses

03 카테고리칼 크로스 엔트로피(categorical cross-entropy)

- 범주형 교차 엔트로피라고도 부르는 손실 함수
- 출력층에서 소프트맥스 함수를 사용하는 다중 클래스 분류(multi-class classification)일 경우 categorical_crossentropy를 사용
- 소프트맥스 회귀에서 사용했던 손실 함수
- 정수값을 가진 레이블에 대해서 다중 클래스 분류를 수행하고 싶다면 'sparse_categorical_crossentropy'를 사용

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

```
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(), optimizer='adam', metrics=['acc'])
```

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(), optimizer='adam', metrics=['acc'])
```

https://www.tensorflow.org/api_docs/python/tf/keras/losses

THANK YOU