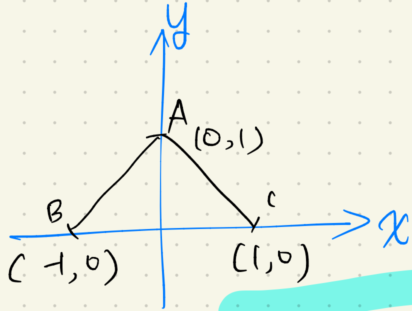
3 basic linear transformations

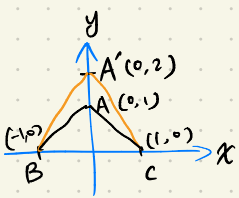
- Scaling
- Translation
- Rotation



일반적인
사칙연산

$$\underbrace{\begin{bmatrix}0\\1\end{bmatrix}} \times \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}0\\2\end{bmatrix}$$

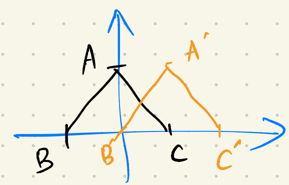
$\vec{A}$ x scaling factor = scaled $\vec{A}$



$$\underbrace{\begin{bmatrix}1\\0\end{bmatrix}}_{\vec{C}} + \underbrace{\begin{bmatrix}1\\0\end{bmatrix}}_{\text{translation factor}} = \begin{bmatrix}2\\0\end{bmatrix}$$; translated $\vec{C}$

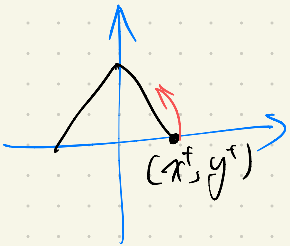
$$\underbrace{\begin{bmatrix}0\\1\end{bmatrix}}_{\vec{A}} + \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$$; translated $\vec{A}$

$$\underbrace{\begin{bmatrix}-1\\0\end{bmatrix}}_{\vec{B}} + \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$; translated $\vec{B}$



Rotation - Trigonometry

삼각법 활용



given a point (x^f, y^f)

회전공식

$$\begin{cases} x' = x^f \cos(A) + y^f(-\sin(A)) \\ y' = x^f \sin(A) + y^f \cos(A) \end{cases}$$

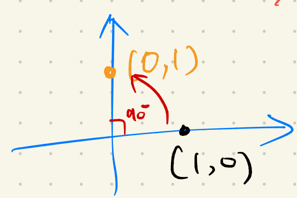
new point

(1, 0) 점을
anti-clockwise 방향으로
90° 회전

(Sol)

$$\begin{cases} x' = 1^f \cos(90^\circ) + \cancel{0^f(-\sin(90^\circ))}^{\,0} \\ y' = 1^f \sin(90^\circ) + \cancel{0^\circ \cos(90^\circ)}^{\,0} \end{cases}$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



앞의 Rotation 연산을 Table로 정리해보자.

	x^+	y^+
$x'_{new}=$	$\cos(A)$	$-\sin(A)$
$y'_{new}=$	$\sin(A)$	$\cos(A)$

	1	0
$x'_{new}=$	$\cos(90^\circ)=0$	$-\sin(90^\circ)=-1$
$y'_{new}=$	$\sin(90^\circ)=1$	$\cos(90^\circ)=0$

	1	0
$x'_{new}=$	$\cos(90^\circ)=0$	$+-\sin(90^\circ)=-1$
$y'_{new}=$	$\sin(90^\circ)=1$	$+\cos(90^\circ)=0$

$$x' = 0 + 0 = 0$$

$$y' = 1 + 0 = 1$$

앞의 회전공식 연산을 Table로 정리해서 연산

→ Vector-matrix multiplication

좀더 체계적으로 정리

$$\begin{array}{c|c|c|l} & x & y & \\ \hline x_{new} & a & + \; b & = xa + yb \\ \hline y_{new} & c & + \; d & = xc + yd \end{array}$$

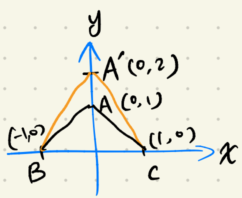
$$\begin{vmatrix} x \\ y \end{vmatrix} \times \begin{vmatrix} a & b \\ c & d \end{vmatrix} = \begin{vmatrix} xa + yb \\ xc + yd \end{vmatrix}$$

같은 방식으로 Scaling & Translation 또한 정리해보자.

새로운 연산 Format 만들기

$$\begin{bmatrix}0\\1\end{bmatrix} \times \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}0\\2\end{bmatrix}$$

$\vec{A}$ x scaling factor = scaled $\vec{A}$



	0	1
x_{new}	1	+ 0
y_{new}	0	+ 2

$= 0 + 0$

$= 0 + 2$

$$\begin{vmatrix}0\\1\end{vmatrix} \times \begin{vmatrix}1 & 0\\0 & 2\end{vmatrix} = \begin{vmatrix}0\times1 + 0\times1\\0\times0 + 1\times2\end{vmatrix} = \begin{vmatrix}0\\2\end{vmatrix}$$

Scaling matrix

연산 결과가 동일함.

(Q) 결과가 같은 것은 알겠다. 하지만, 왜 이렇게 복잡하게 연산하지?

(A) 동일한 변환을 모든 벡터 vertex에 적용할 때 반복연산 해야하는데, for-loop 없이 빠르게 연산가능하게 만듦.

	$x_1, x_2, \dots, x_n$	$y_1, y_2, \dots, y_n$
x_{new}	1	0
y_{new}	0	2

for (x, y) in $[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$:

scaled_point = $\begin{bmatrix} x \\ y \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

배치 벡터에 대한
변환 연산을
배치 끝내도록 해줌.

$$\underbrace{\begin{vmatrix} 1 & 0 \\ 0 & 2 \end{vmatrix}}_{\text{scaling\_matrix}} \times \underbrace{\begin{vmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{vmatrix}}_{\text{batch\_vertices}} = \begin{vmatrix} x_1 \times 1 + y_1 \times 0 & x_2 \times 1 + y_2 \times 0 & \cdots \\ x_1 \times 0 + y_1 \times 2 & x_2 \times 0 + y_2 \times 2 & \cdots \end{vmatrix}$$

scaled_vector① scaled_vector②

Translation을 위한 matrix 연산은?.

* Affine transformation

앞서 Scaling & Rotation에 대한 matrix-vector multiplication을 정리함.

(Q) Translation도 가능?.

(A) Yes but 좀더 복잡함.

multiplication의 어디에나 존재한다.

Translation은 덧셈(additive)이고 이런 연산 규칙으로는 표현이 불가능하다.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \times \begin{vmatrix} x \\ y \end{vmatrix} = \begin{vmatrix} xa + yb \\ xc + yd \end{vmatrix} \Rightarrow$$

어떻게 Translation을 matrix 연산으로 표현할까?

$$Vector_{new} = Matrix \times Vector_{old} + Translation$$

표현하고 싶은 형태

Scaling & Rotation 표현 가능

해당 연산을 matrix multiplication에 넣는 건 불가능.
Because, 곱셈 없이 단순 덧셈이기에.

$$\begin{vmatrix} x_{new} \\ y_{new} \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} e \\ f \end{vmatrix}$$

연산 Trick

같은 연산 결과 :)

$$\begin{vmatrix} x_{new} \\ y_{new} \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} xa + yb + e \\ xc + yd + f \\ 0 + 0 + 1 \end{vmatrix}$$

matrix multiplication
덧셈 넣기
성공

(정리)

$$\begin{vmatrix} x_{new} \\ y_{new} \end{vmatrix} = \underbrace{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}_{\text{Rotation \& Scaled transformation}} \times \begin{vmatrix} x \\ y \end{vmatrix} + \underbrace{\begin{vmatrix} e \\ f \end{vmatrix}}_{\text{Translation}}$$

; Affine transformation

$$\Downarrow \text{ 연산 trick}$$

$$\begin{vmatrix} x_{new} \\ y_{new} \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{vmatrix} \times \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Affine transformation을 Matrix multiplication으로 구현함.

결국

Fully-Connected layer의

$$\sigma \begin{vmatrix} y \\ 1 \end{vmatrix} = \begin{vmatrix} W & b \\ 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ 1 \end{vmatrix}$$

$$y = \underbrace{W}_{\text{weight}} x + \underbrace{b}_{\text{bias}}$$

또한

$$\boxed{\begin{vmatrix} y_1 \\ y_2 \\ 1 \end{vmatrix} = \begin{vmatrix} w_{11} & w_{12} & b_1 \\ w_{21} & w_{22} & b_2 \\ 0 & 0 & 1 \end{vmatrix} \times \begin{vmatrix} x_1 \\ x_2 \\ 1 \end{vmatrix}}$$

같은

Matrix-vector multiplication 형태로 표현이 가능하기 때문에 NYU 강의에서 통쳐서 Rotation 이라고 말한 듯.

$\underbrace{\text{Rotation}}_{\text{linear}} \rightarrow \underbrace{\text{Squashing}}_{\text{non-linear}} \rightarrow \text{Rotation} \rightarrow \text{Squashing} \rightarrow \cdots$