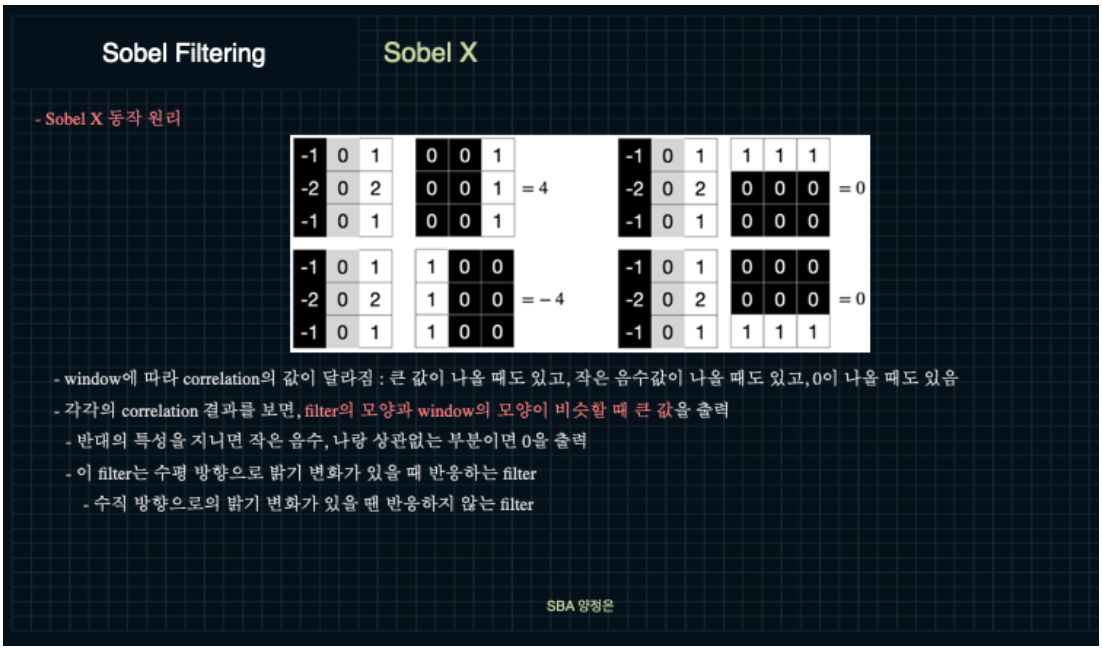
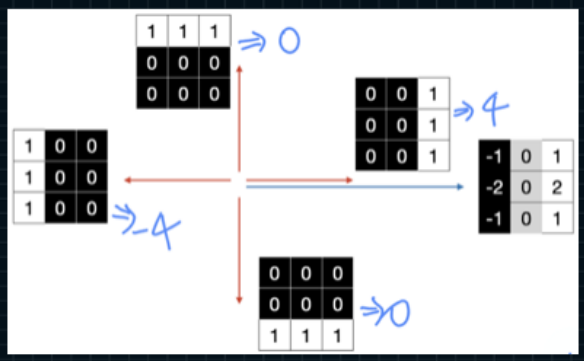
ML Day25 (Sobel Filtering)

▼ **2-D Correlation Excercise**



Sobel Filtering — Sobel X



SBA 양정은

Sobel Filtering — Filter & Window

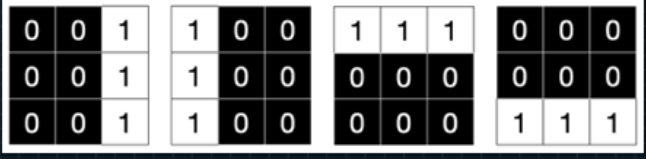
-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

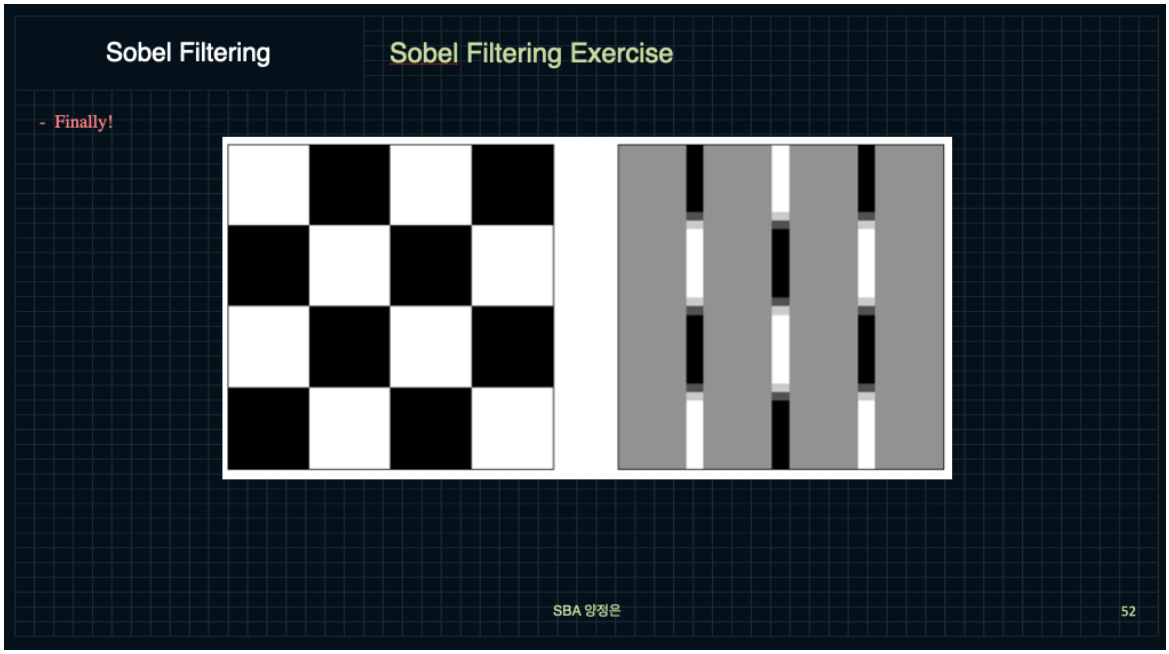
window



- 두 가지의 sobel filter가 있음
- 3x3 행렬의 모습을 가진다.
- 원소는 2, 1, 0, -1, -2의 값을 가진다.
- 좌우 대칭 또는 상하 대칭의 모습을 가진다.

- 원래는 0~255까지의 픽셀값을 가지는 것이 맞지만, 0~1이라고 가정
- window를 봅다보면, 다양한 모습이 나올텐데 그 중에서 대표적인 4가지의 모습을 단순화

SBA 양정은 39




```
import numpy as np
import matplotlib.pyplot as plt

white = 1*np.ones(shape=(10, 10))
black = 0*np.ones(shape=(10, 10))

img1 = np.hstack([white, black])
img2 = np.hstack([black, white])
img_stack = np.vstack([img1, img2])
img = np.tile(img_stack, reps=[2, 2])       # 기본 이미지: 흰,검으로 이루어진 (4, 4) 모양

fig, ax = plt.subplots(1, 3, figsize=(8, 4))

x_filter = np.array([[-1, 0, 1],
                     [-2, 0, 2],
                     [-1, 0, 1]])
y_filter = np.array([[1, 2, 1],
                     [0, 0, 0],
                     [-1, -2, -1]])
H, W = img.shape                              # img.shape: (40, 40)
F = x_filter.shape[0]                         # 임시로 생성될 window 수, F = 3
H_ = H - F + 1                                # H_ = 38
W_ = W - F + 1                                # W_ = 38

filtered_data_x = np.zeros(shape=(H_, W_))    # filtered_data_x: 0으로 채워진 (H_, W_) 형태의 행렬 생성
filtered_data_y = np.zeros(shape=(H_, W_))

# x_filter 적용
for h_idx in range(H_):                       # range(H_): 40
    for w_idx in range(W_):                   # range(W_): 40
        window = img[h_idx : h_idx + F,
                 w_idx : w_idx + F]
        z = np.sum(window * x_filter)         # for문을 돌며 변화하는 window와 x_filter의 내적을 np.sum을 통해 구하고 z변수에 대입
        filtered_data_x[h_idx, w_idx] = z     # filtered_data_x[h_idx, w_idx]위치에 위에서 구한 z값을 대입

# y_filter 적용
for h_idx1 in range(H_):
    for w_idx1 in range(W_):
        window = img[h_idx1 : h_idx1 + F,
                 w_idx1 : w_idx1 + F]
        z = np.sum(window * y_filter)
        filtered_data_y[h_idx1, w_idx1] = z

ax[0].imshow(img, cmap='gray')
ax[1].imshow(filtered_data_x, cmap='gray')
ax[2].imshow(filtered_data_y, cmap='gray')
```

```
for ax_idx in range(3):
    ax[ax_idx].tick_params(left=False, labelleft=False,
                   bottom=False, labelbottom=False)
fig.tight_layout()
plt.show()
```