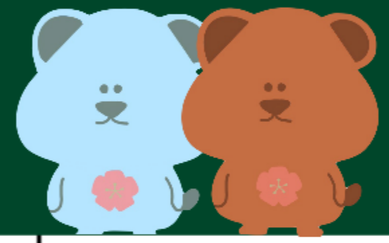
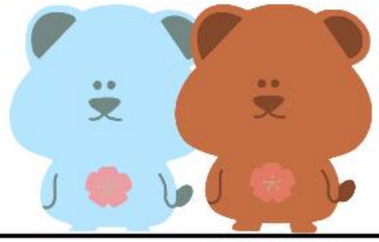


Week 3 강의 리뷰

2129006 김경민

EWHA EURON

목차



EWHA EURON

파이썬과 벡터화

EWHA EURON

#01-1 파이썬의 브로드캐스팅

📌 브로드캐스팅 예제 (1)

$$A = \begin{bmatrix} & \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \\ \text{Carb} & 56.0 & 0.0 & 4.4 & 68.0 \\ \text{Protein} & 1.2 & 104.0 & 52.0 & 8.0 \\ \text{Fat} & 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix}$$

```
# 행렬 A 생성
import numpy as np

A = np.array([[56.0, 0.0, 4.4, 68.0],
              [1.2, 104.0, 52.0, 8.0],
              [1.8, 135.0, 99.0, 0.9]])

# 음식 별 칼로리 계산 (= 열 별 합계 계산)
cal = A.sum(axis=0)
print(cal)
```

```
[ 59.  239.  155.4  76.9]
```

```
# 탄수화물, 단백질, 지방의 칼로리 비율 계산
percentage = 100 * A / cal.reshape(1,4)
print(percentage)
```

```
[[94.91525424  0.          2.83140283 88.42652796]
 [ 2.03389831 43.51464435 33.46203346 10.40312094]
 [ 3.05084746 56.48535565 63.70656371  1.17035111]]
```

① 음식별 합계 계산 :

행렬 A의 각 열별 합계를 구하여 (1, 4) 크기의 cal 변수에 저장

② 탄, 단, 지의 칼로리 비율 계산 :

A(3, 4) 행렬을 cal(1, 4) 벡터로 나누는 과정에서 브로드캐스팅이 수행됨

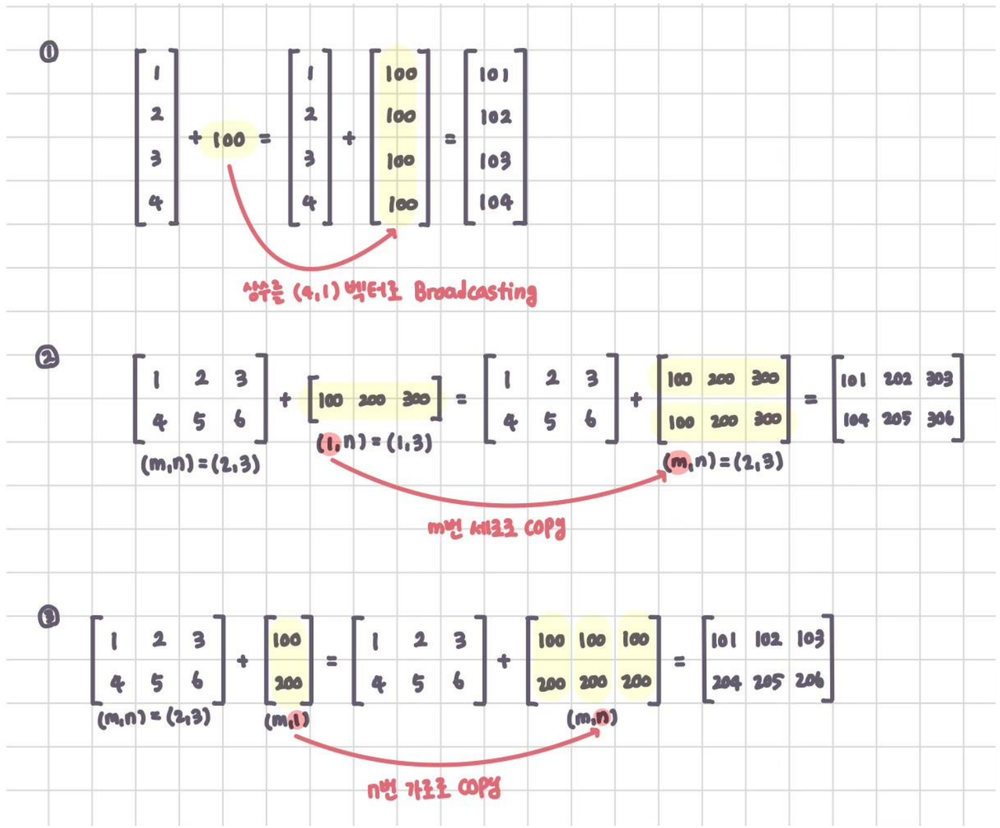
<브로드캐스팅 과정>

$$cal = \begin{bmatrix} 59. & 239. & 155.4 & 76.9 \end{bmatrix} \rightarrow cal_{broadcasting} = \begin{bmatrix} 59. & 239. & 155.4 & 76.9 \\ 59. & 239. & 155.4 & 76.9 \\ 59. & 239. & 155.4 & 76.9 \end{bmatrix}$$

EWHA EURON

#01-1 파이썬의 브로드캐스팅

📌 브로드캐스팅 예제 (2)



<Rule of Broadcasting>

Rule 1)

If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side.

Rule 2)

If the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stratched to match the other shape.

Rule 3)

If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

EWHA EURON

#01-2 파이썬과 넘파이 벡터

```
a = np.random.randn(5)  # 가우시안 분포를 따르는 난수 5개를 배열로 저장
print(a)
print(a.shape)  # (5,) : 랭크가 1인 배열, 행/열벡터가 아님
print(a.T)            # a와 같음
print(np.dot(a,a.T))  # 행렬이 아닌 상수값이 나옴
```

```
[-0.52923326 -0.07356486  1.35399627 -2.17405907 -0.37739959]
(5,)
[-0.52923326 -0.07356486  1.35399627 -2.17405907 -0.37739959]
6.98776880976423
```

```
a = np.random.randn(5,1)
print(a)
print(a.shape)  # (5,1) : 열벡터
print(a.T)      # (1,5) : 행벡터
print(np.dot(a,a.T))  # 벡터의 외적 출력
```

```
[[ 0.89393556]
 [ 1.16086202]
 [ 1.16791973]
 [-0.73096322]
 [-0.82689101]]
(5, 1)
[[ 0.89393556  1.16086202  1.16791973 -0.73096322 -0.82689101]]
[[ 0.79912078  1.03773584  1.04404497 -0.65343401 -0.73918728]
 [ 1.03773584  1.34760063  1.35579366 -0.84854744 -0.95990637]
 [ 1.04404497  1.35579366  1.3640365  -0.85370636 -0.96574233]
 [-0.65343401 -0.84854744 -0.85370636  0.53430722  0.60442691]
 [-0.73918728 -0.95990637 -0.96574233  0.60442691  0.68374875]]
```

이상한 결과와 오류가 발생하는 것을 방지하기 위해

배열(rank 1 array)보다는 행/열벡터 형태를 사용하기

- assert(a.shape == (5,1))
- a = a.reshape(5,1) : shape을 변경

EWHA EURON

#02 로지스틱 회귀 비용함수

📌 1개의 훈련 샘플에서 비용함수

$P(y=0|x) = 1-\hat{y}$

$\hat{y} = \sigma(w^T x + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$ → P(y=1|x) : x가 주어졌을 때 y가 1일 확률

$P(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)}$ → y=1일 때와 y=0일 때를 하나의 식으로 합침

$\log P(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log(1-\hat{y}) = -L(\hat{y}, y)$

→ 양변에 로그(단조증가함수)를 취함

→ 확률을 최대화하는 것 = 확률의 로그값을 최대화하는 것 = 손실함수를 최소화하는 것

EWHA EURON

#02 로지스틱 회귀 비용함수

📌 m개의 훈련 샘플에서 비용함수

$$\mathrm{P}(\text{labels in training set}) = \Pi_{i=1}^{m} P(y^{(i)}|x^{(i)}) \ (\because \text{iid})$$

$$\log \mathrm{P}(\text{labels in training set}) = \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) = -\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \rightarrow -\mathrm{L}(\hat{\mathrm{y}}^{(\mathrm{i})}, \mathrm{y}^{(\mathrm{i})})$$를 최대화

$$\mathrm{J}(\mathrm{w}, \mathrm{b}) = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{L}(\hat{\mathrm{y}}^{(\mathrm{i})}, \mathrm{y}^{(\mathrm{i})})$$ → 비용함수(음수를 제거하고 평균을 취한 형태)를 최소화

→ 비용함수를 최소화하는 것 = 로지스틱 회귀 모델의 최대 우도 추정을 하는 것

🔎 최대 우도 추정(MLE) : 훈련 세트의 타깃 확률을 최대화해주는 매개변수를 찾는 것

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left\{ -\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \right\} = \underset{\theta}{\operatorname{argmin}} \left\{ \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \right\}$$

$\hat{\theta} \rightarrow \hat{w}, \hat{b}$

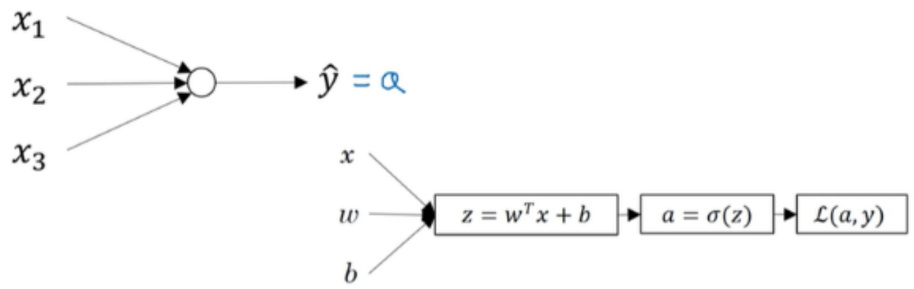
EWHA EURON

얕은 신경망 네트워크

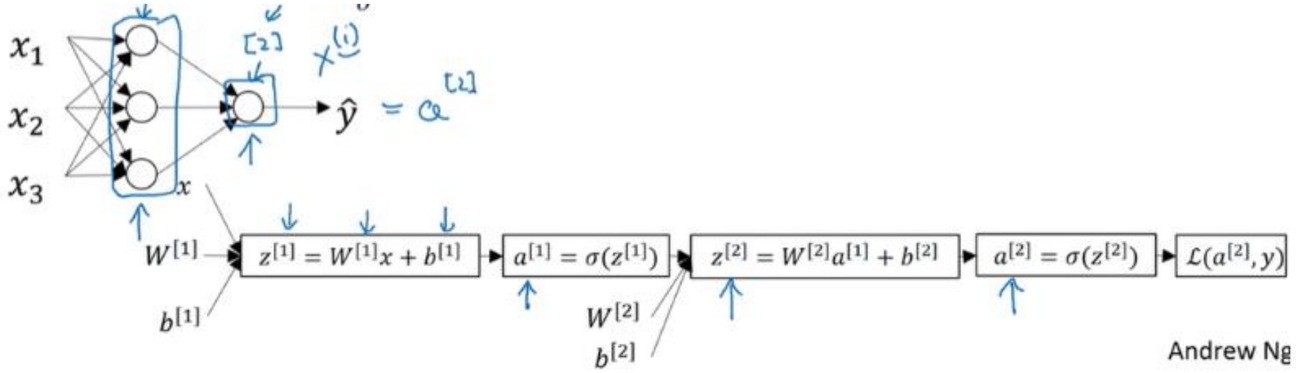
EWHA EURON

#03 신경망 네트워크의 구성

📌 로지스틱 회귀 모델



📌 신경망 모델

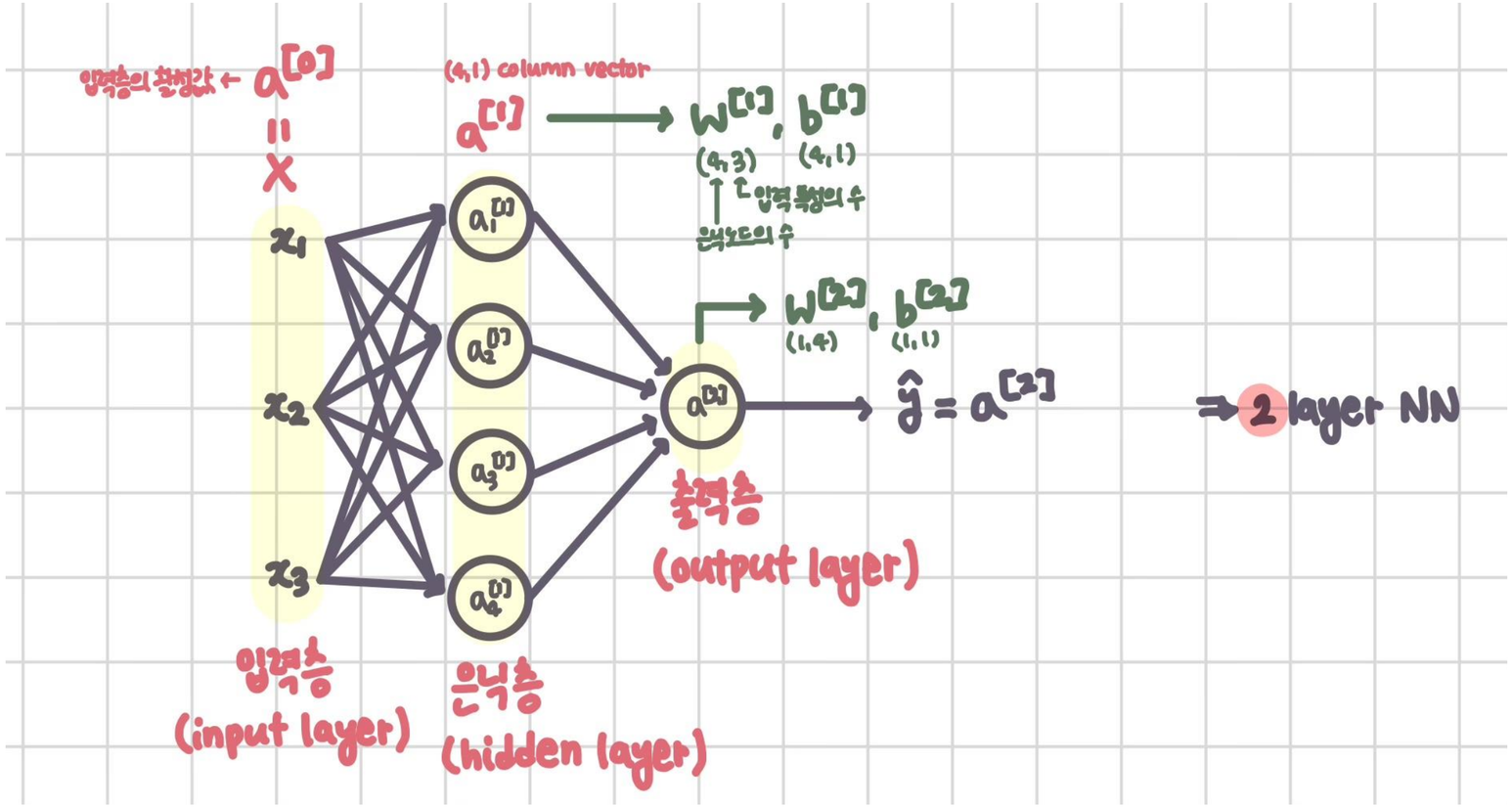


- 위첨자 [1] : layer
- Z와 a를 여러 층에서 여러 번 계산

EWHA EURON

#03 신경망 네트워크의 구성

📌 신경망 모델 구조



- a(activation) : 활성값, 신경망의 층들이 다음 층으로 전달해주는 값
- 매개변수 : w, b → 은닉층을 거치면서 업데이트 되고, 최종적으로 출력층에 전달되어 출력값 계산

EWHA EURON

#04 신경망 네트워크 출력의 계산

📌 은닉층의 계산

$$z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T}x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T}x + b_4^{[1]},\ a_4^{[1]} = \sigma(z_4^{[1]})$$

📌 벡터화

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T}x + b_1^{[1]} \\ w_2^{[1]T}x + b_2^{[1]} \\ w_3^{[1]T}x + b_3^{[1]} \\ w_4^{[1]T}x + b_4^{[1]} \end{bmatrix}$$

$$z^{[1]} = W^{[1]}X + b^{[1]}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]}) = \sigma(W^{[1]}X + b^{[1]})$$

- 노드마다 z와 a를 계산할 때 for-loop 필요 → z 열벡터와 a 열벡터 생성

#05 m개의 훈련 샘플에 대한 벡터화

📌 m개의 훈련 샘플

$$x^{(1)} \rightarrow \hat{y}^{(1)} = a^{[2](1)}, \cdots, x^{(m)} \rightarrow \hat{y}^{(m)} = a^{[2](m)}$$

for i=1 to m:

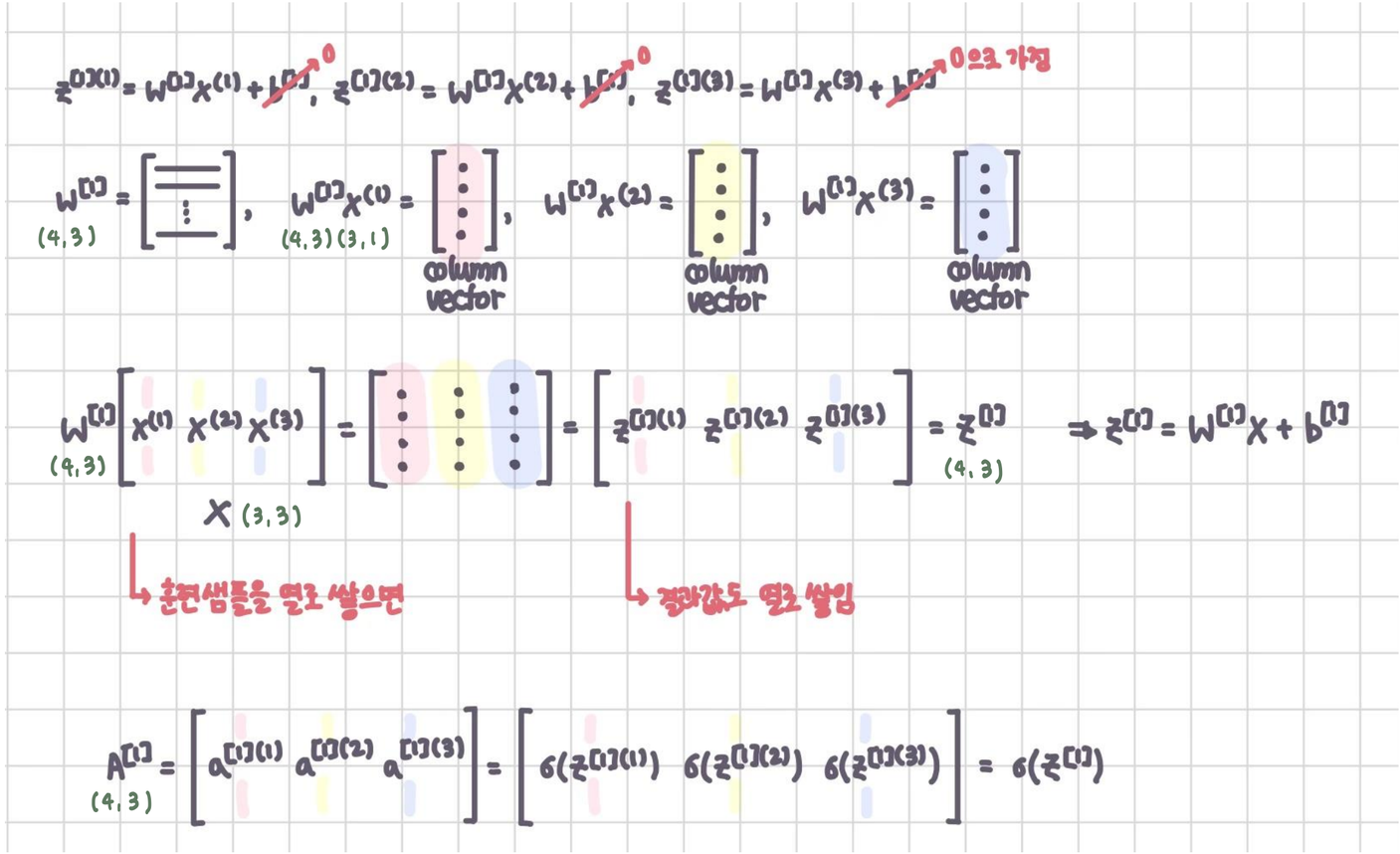
$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma(z^{[2](i)})$$

📌 벡터화



- 훈련 샘플마다 z 열벡터와 a 열벡터를 계산할 때 for-loop 필요

  → 훈련 샘플을 열로 쌓아 Z, A 행렬 생성

EWHA EURON

출석 퀴즈 리뷰

EWHA EURON

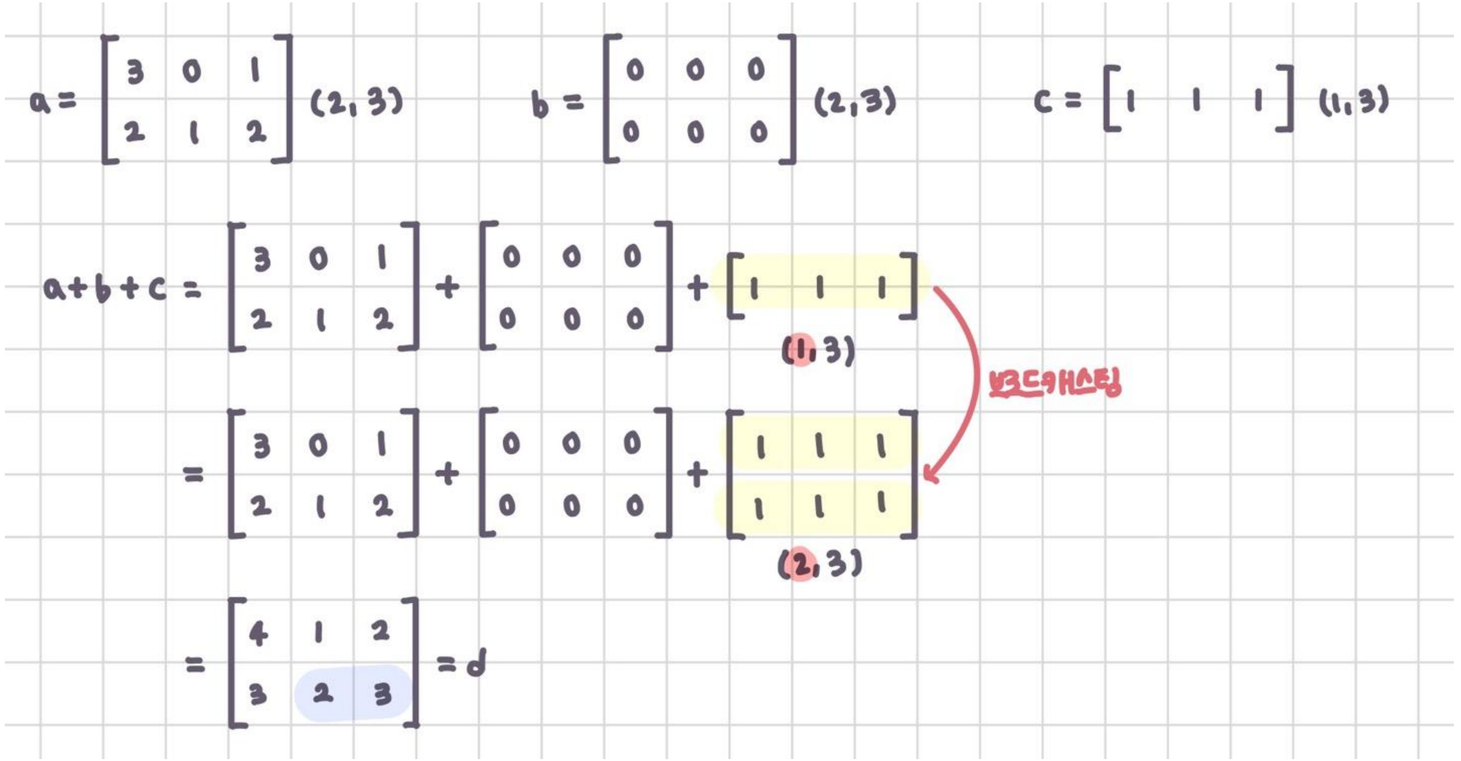
#06 출석 퀴즈 리뷰 (1)

6. 다음 코드의 결과로 옳은 것을 골라주세요. *

```
import numpy as np

a = np.array([[3,0,1],[2,1,2]])
b = np.zeros([2,3])
c = np.ones([1,3])
d = a + b + c
print(d[1,1:3])
```

- ○ [1., 2.]
- ○ [2., 1., 2.]
- ◉ [2., 3.]
- ○ [3., 2., 3.]



#06 출석 퀴즈 리뷰 (2)

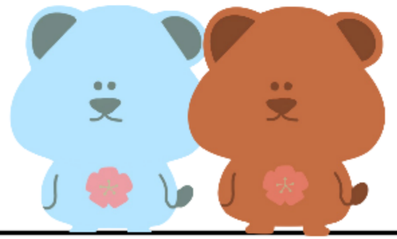
1. Numpy의 random 모듈을 사용하여 0부터 99 사이의 랜덤한 수 하나를 생성 * 1점
하는 옳은 코드를 골라주세요.

- (●) np.random.randint(100)
- ( ) np.random.rand(100)
- ( ) np.random.randn(100)
- ( ) np.random.rand()

- np.random.randint(100) : 0~99 사이의 랜덤한 정수 하나를 생성
- np.random.rand(100) : 0~1 사이의 Uniform 분포를 따르는 랜덤한 수 100개를 생성
- np.random.randn(100) : 정규분포를 따르는 랜덤한 수 100개를 생성
- np.random.rand() : 0~1 사이의 Uniform 분포를 따르는 랜덤한 수 하나를 생성

EWHA EURON

THANK YOU



EWHA EURON