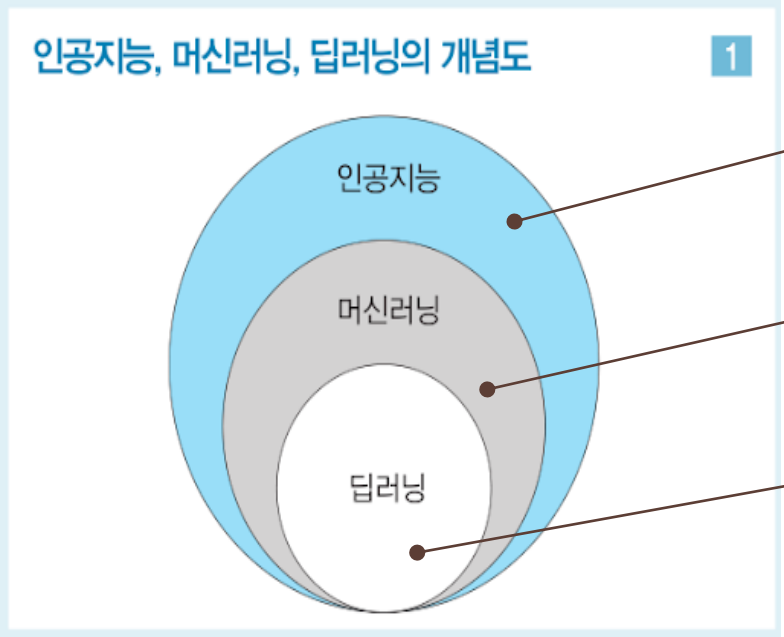
Deep Learning
week 1

문구영

인공지능과 머신러닝, 딥러닝

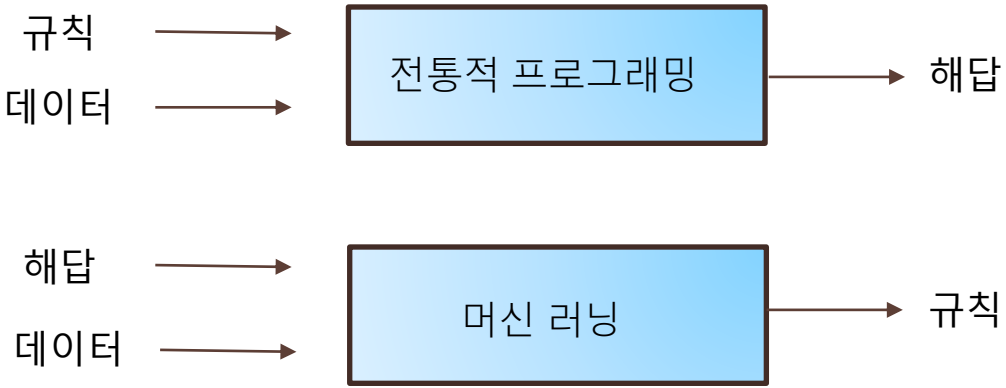


기계가 사람의 행동을 모방하게 하는 기술

코드로 명시하지 않은 동작을 데이터로부터 학습하여 실행할 수 있도록 하는 알고리즘을 개발하는 분야

머신러닝의 한 분야인 인공 신경망에 기반하여, 많은 양의 데이터를 학습해 뛰어난 성능을 이끌어내는 연구분야

머신러닝의 패러다임



- 훈련 (명시적 프로그래밍 x)
- 통계적 **구조, 규칙 학습**
- 자동화

- 특정 작업을 수행하는 법을 스스로 학습할 수 있는가?
- 어떤 것을 작동시키기 위해 '어떻게 명령한 지 알고 있는 것' 이상의 처리가 가능할까?

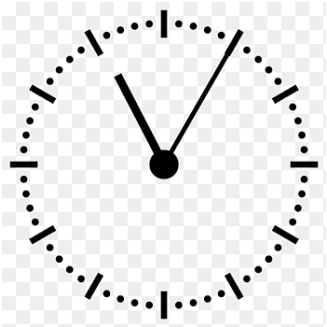
특성공학 (Feature Engineering)

- 입력 데이터 → 모델 → '의미 있는' 출력 (학습 과정)

- **'의미 있는'** 데이터로의 변환, 입력 데이터를 기반으로 기대 출력에 가깝게 만드는 **유용한 표현**을 학습

- **모델이 수월하게 작업할 수 있는 어떠한 방식**으로 데이터를 표현

특성공학 예시



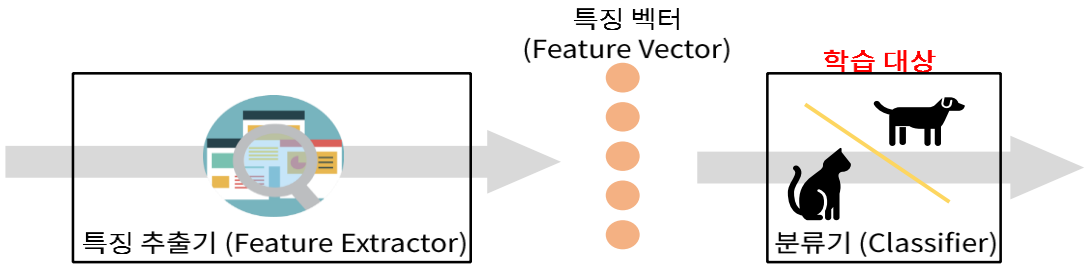
원본 데이터 : 2차원 픽셀 데이터

더 나은 표현 : {x1 : 0.3, y1 : 0.8}
{x2 : 0.3, y2 : 0.6}

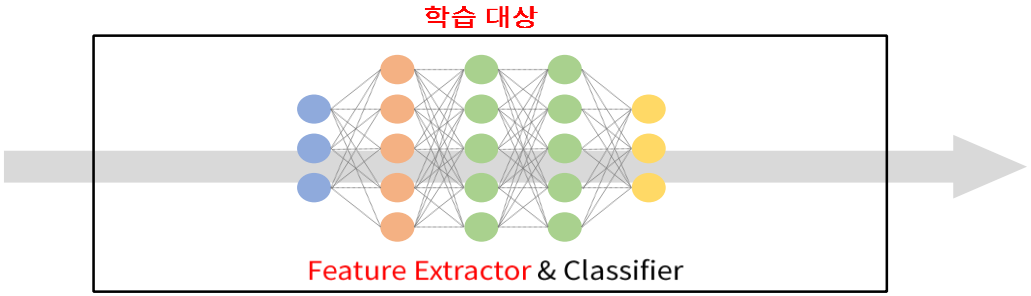
훨씬 더 좋은 특성 : theta1 : 70
theta2 : 80

- 문제를 쉽게 만드는 특성 추출이 머신 러닝의 핵심
- 직접 유용한 표현을 추출해야 함 (특성공학)
- ex) SVM : 학습을 통해 분할 경계/초평면 학습 (커널 함수 : 데이터로부터 학습 X)

딥러닝과 머신러닝



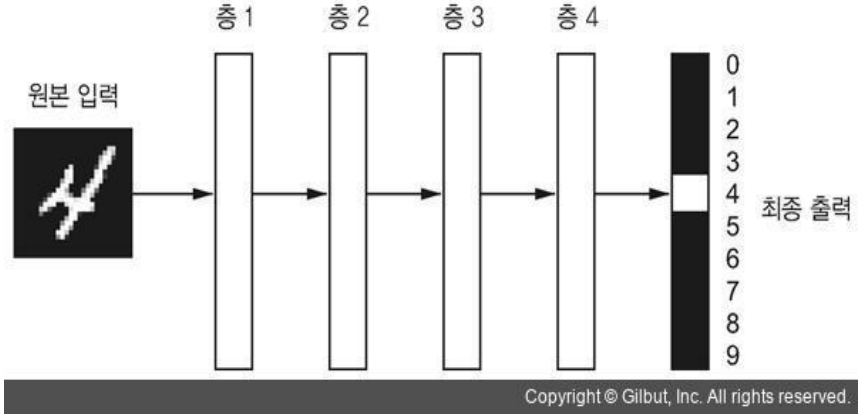
기계 학습 (Machine Learning)

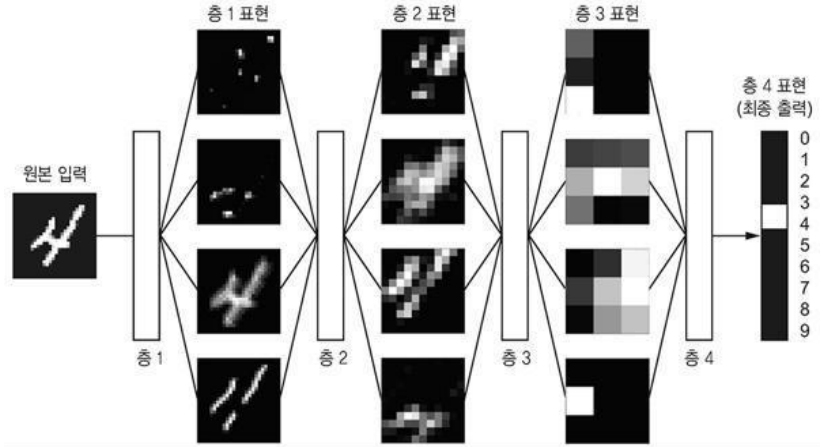


딥러닝 (Deep Learning)

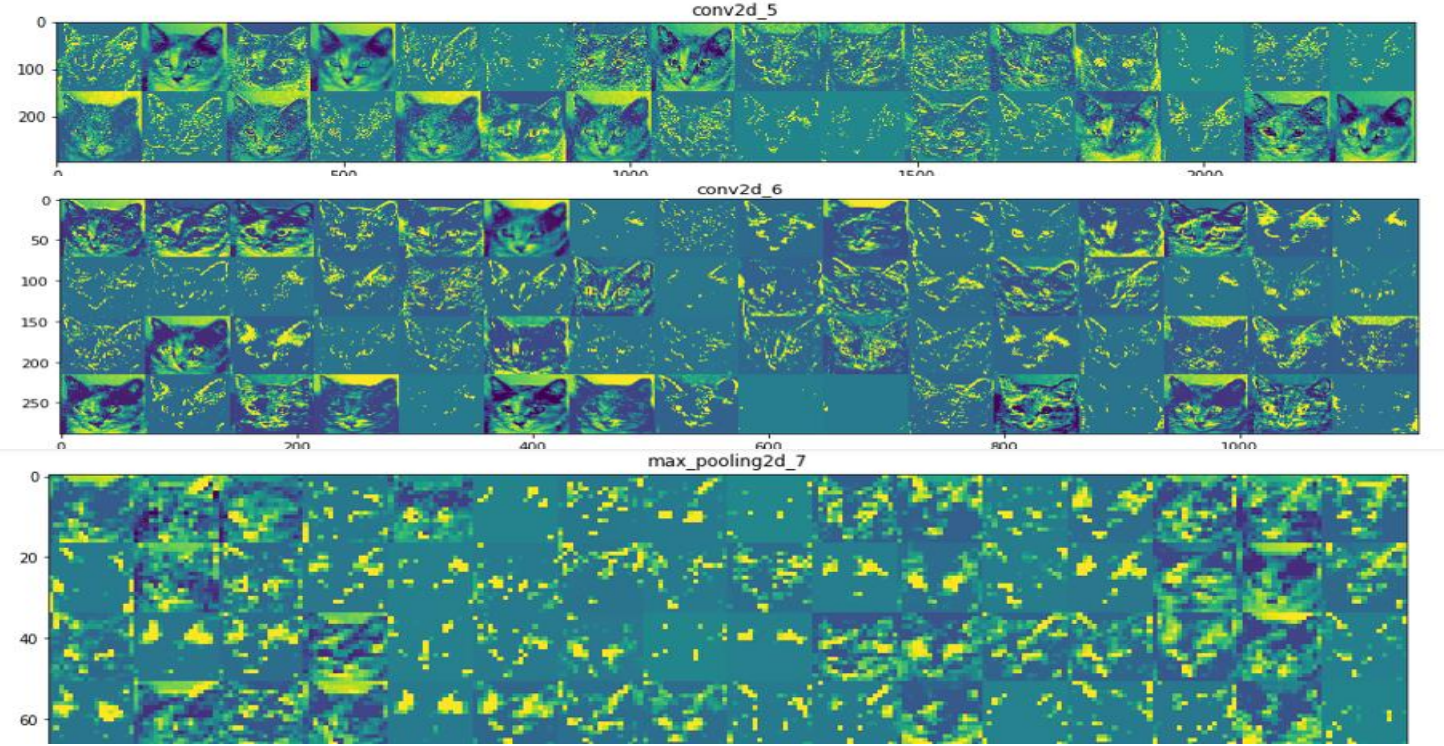
Deep?

- **연속된 층(layer)**에서 점진적으로 의미 있는 표현을 학습
- 연속된 층으로 **의미 있는 표현을 학습** (다단계 처리 방식)
- 층 기반 표현 학습, 계층적 표현 학습



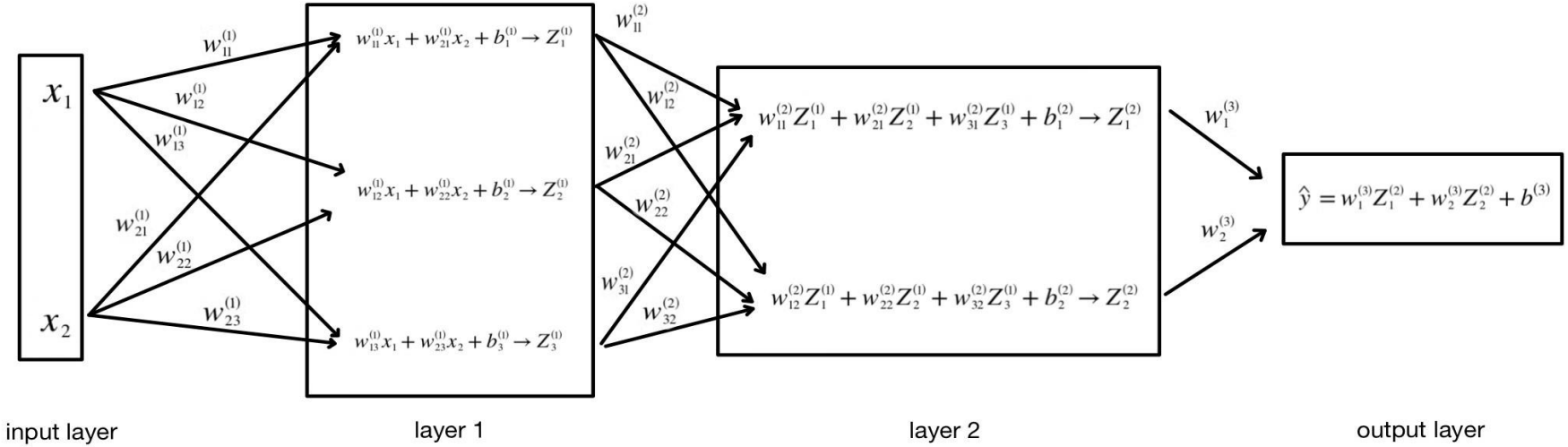


딥러닝의 시각화



- 각 층의 모든 변화는 최종 목표를 따라간다
- 층을 거치면서 **점진적으로 더 복잡한, 추상적인 표현**이 만들어짐
- **점진적인 중간 표현이 공동으로 학습** (한 층의 변화는 다른 층에 영향을 끼침)

수식적 표현



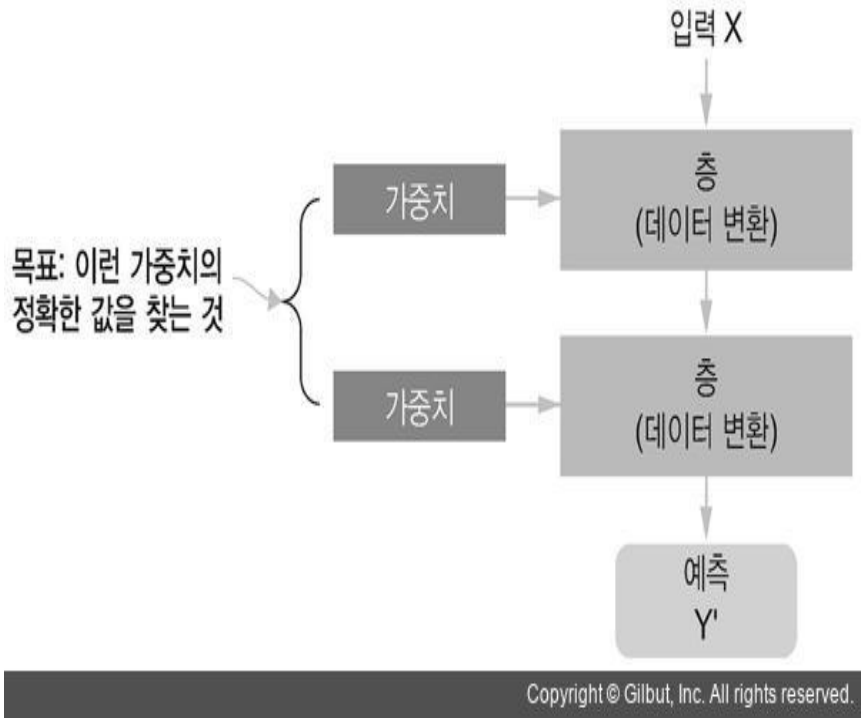
$$Z_i^{(1)} = \sigma\left(w_{1i}^{(1)}x_1 + w_{2i}^{(1)}x_2 + b_i^{(1)}\right)$$

$$i = 1, 2, 3$$

$$Z_i^{(2)} = \sigma\left(w_{1i}^{(2)}Z_1^{(1)} + w_{2i}^{(2)}Z_2^{(1)} + w_{3i}^{(2)}Z_3^{(1)} + b_i^{(2)}\right)$$

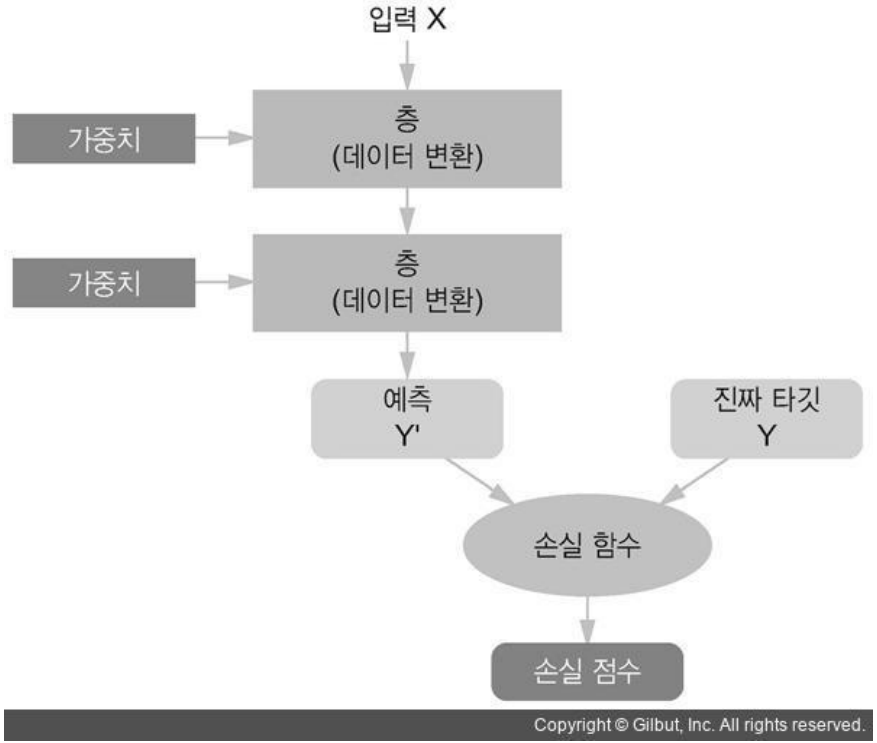
$$i = 1, 2$$

딥러닝의 작동원리 1



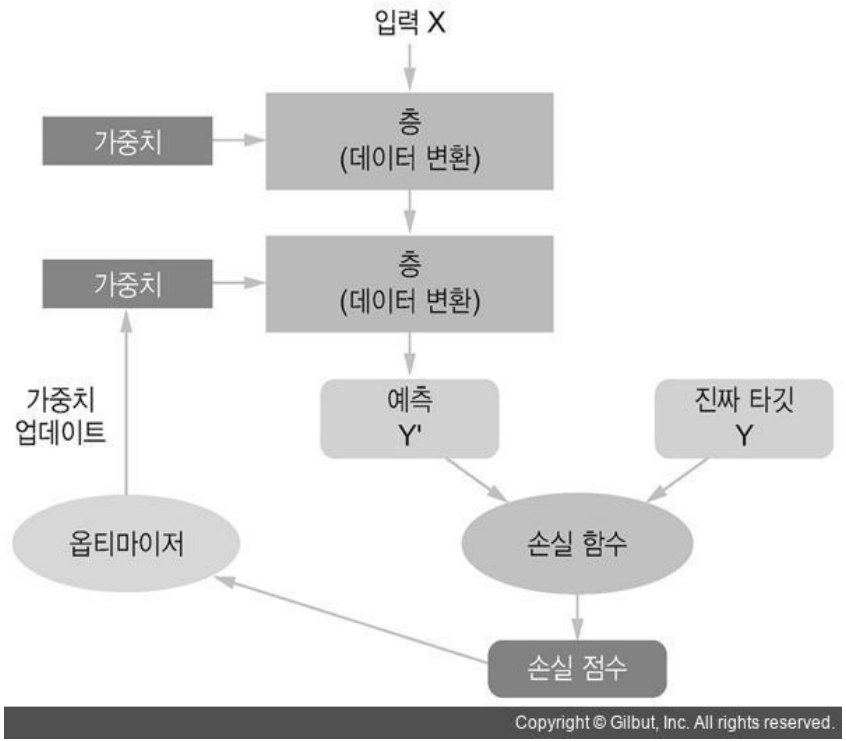
- 입력 – 타깃 (이미지 – 고양이) 매핑을 데이터 변환기(층)를 연결하여 수행
- 특정 층에서의 데이터 변환 : **층의 가중치**를 모수(parameter)로 가지는 함수로 표현
- 목표 : 입력을 정확한 타깃에 매핑 (고양이 이미지를 강아지가 아닌 고양이로 분류)
- 학습 : 주어진 입력을 정확한 타깃에 매핑할 수 있는 가중치 값 탐색
- **신경망은 가중치를 파라미터로 가진다**

딥러닝의 작동원리 2



- 하나의 파라미터가 바뀌면 다른 모든 파라미터에 영향을 끼침
- **손실 함수(loss function)**, 목적 함수(objective function), 비용 함수(cost function)
- **신경망 모델의 출력이 실제와 얼마나 다른지** 측정 (모델이 한 샘플에 대해 얼마나 잘 예측 했는지)
- **신경망의 예측과 진짜 타깃 (기대되는 값)의 차이**를 점수로 계산
- **손실 함수가 신경망의 출력 품질을 측정**

딥러닝의 작동원리 3



- 손실 점수를 피드백 신호로 사용
- 현재 샘플(입력)의 손실 점수가 감소되는 방향으로 가중치 조정
- **역전파** 알고리즘을 구현한 **옵티마이저**
- 모든 샘플을 처리하면서 가중치를 올바른 방향으로 조정, 손실 점수 감소 (훈련 반복을 통해 최소화)
- 타깃에 가능한 가장 가까운 출력을 만드는 모델
- **손실점수를 피드백 신호로 사용하여 가중치 조정**

신경망을 위한 데이터 표현

- **텐서 (Tensor)** : 머신 러닝 시스템의 기본 데이터 구조, 데이터를 담는 컨테이너
- **임의의 차원(축) 개수를 가지는 행렬**의 일반화된 모습

```
import numpy as np

x = np.array(12)

>>> x
array(12)

>>> x.ndim
0
```

스칼라

```
x = np.arrary([12, 3, 6, 14, 7])

>>> x

arrary([12, 3, 6, 14, 7])
>>> x.ndim
1
```

벡터

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

>>> x.ndim
2
```

행렬

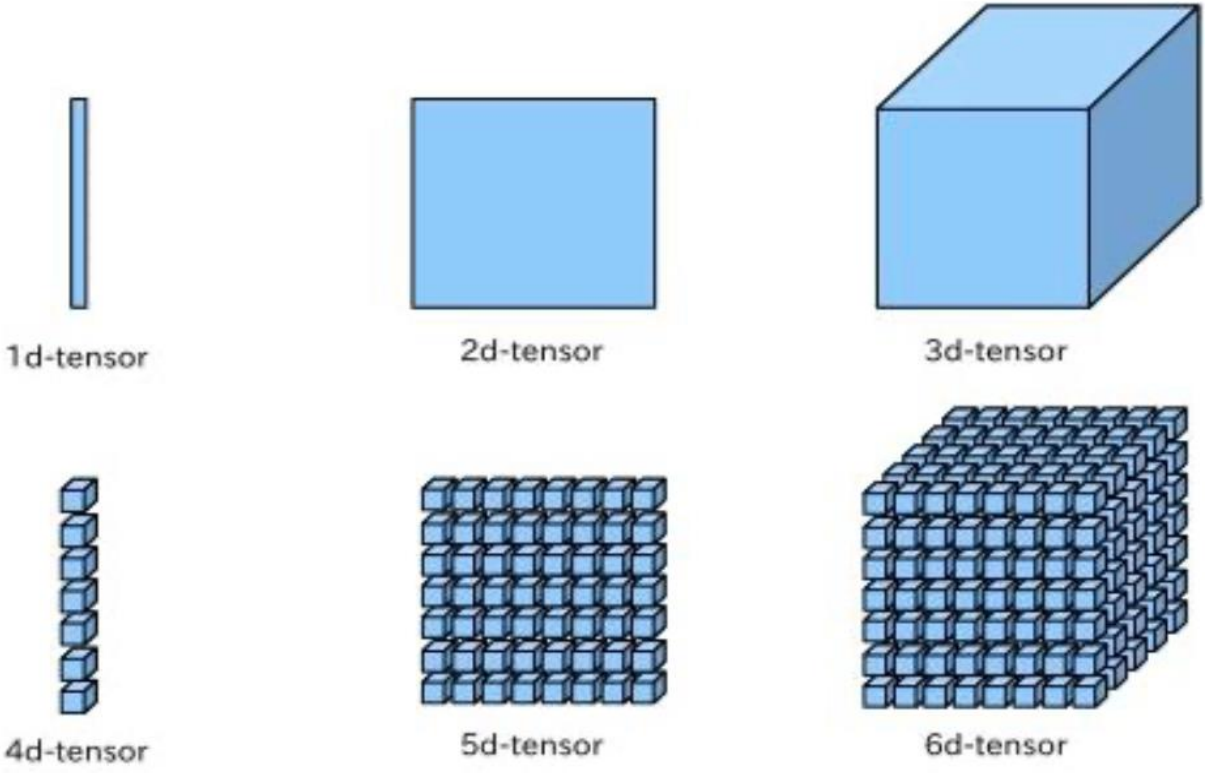
신경망을 위한 데이터 표현

```
x = np.arrary([[[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]]])
```

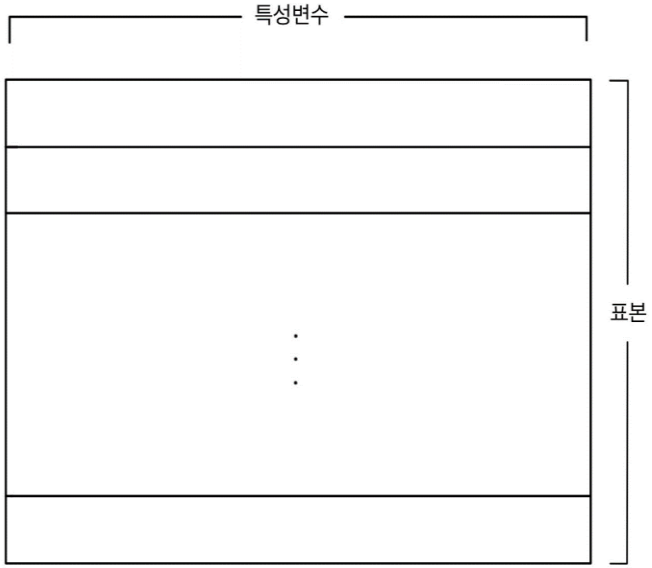
```
>>> x.ndim
3
```

- 행렬을 하나의 새로운 배열로 합침
- 3D 텐서를 하나의 배열로 합치면 4D 텐서

신경망을 위한 데이터 표현 - 그림

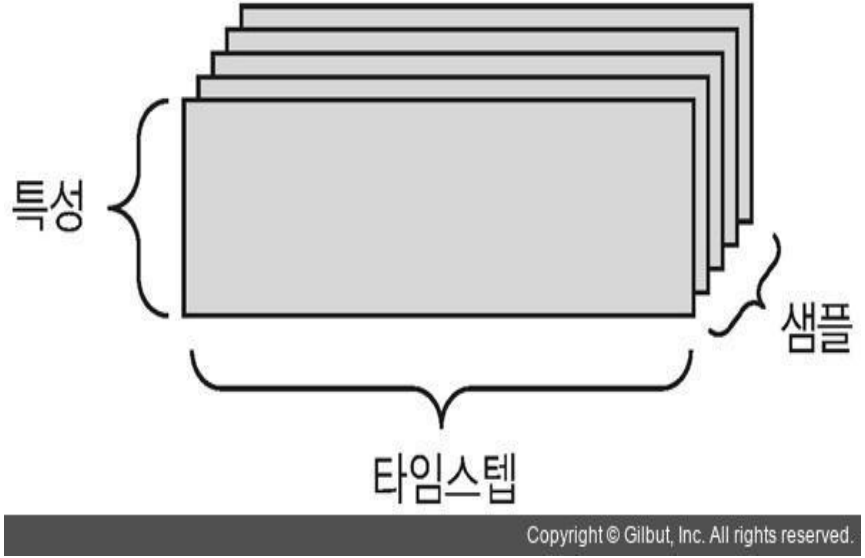


데이터의 표현



- (100, 3) : 3개의 변수를 가진 100개의 관측치
- (50, 200) : 200개의 단어로 표현된 50개의 문서
- **표본 수를 나타내는 축의 값은 생략함**
- input_shape = (None, 3)

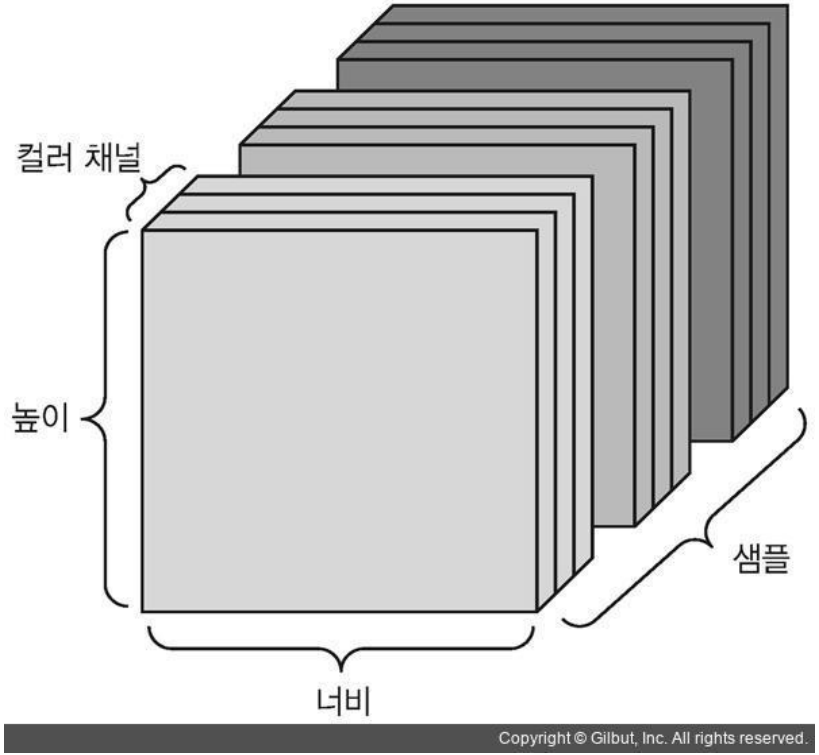
3D 텐서



- (100, 390, 3) : (390, 3)의 데이터가 100개

ex) 390분 동안의 주식가격, 최대가격, 최소가격 데이터를 100일 동안 관측

4D 텐서



- (5000, 78, 78, 3) : (78, 78, 3)의 데이터가 5000개
- (표본수, 높이, 넓이, 채널 수)
- ex) 78x78 픽셀의 컬러 이미지 5000장
  (채널 수=1 : 흑백 이미지, 칼라 : RGB 채널)
- input_shape = (None, 78, 78, 3)

딥러닝 구현하기 1

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

import numpy as np
import tensorflow as tf

X = Data_set[:, 0:8]
Y = Data_set[:, 8]
```

- X (예측변수), Y (타깃변수) 지정

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu')
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

- 모델을 설정, 실행
- 딥러닝의 구조, 층별 옵션 결정
- activation : 활성 함수

딥러닝 구현하기 2

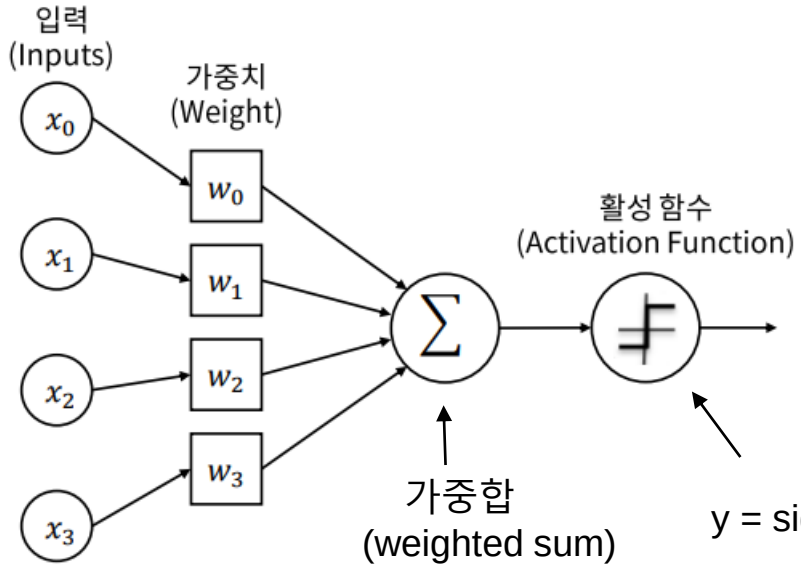
```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

- 손실 함수 지정 (이진 분류 문제)
- 최적화 (오차를 어떻게 줄여 나갈지)
- 평가지표 (최적화)

```
model.fit(X, Y, epochs=200, batch_size=10)
```

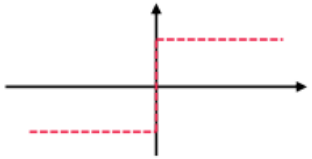
- 실제 학습 수행
- epochs : 전체 샘플 데이터를 한 바퀴 돌며 학습 (전체 샘플을 이용하여 200 바퀴 돌며 학습)
- batch_size : 학습 데이터를 10개로 분할 (가중치를 1회 업데이트 하는 과정에서 10개의 데이터를 사용)

퍼셉트론 (Perceptron)

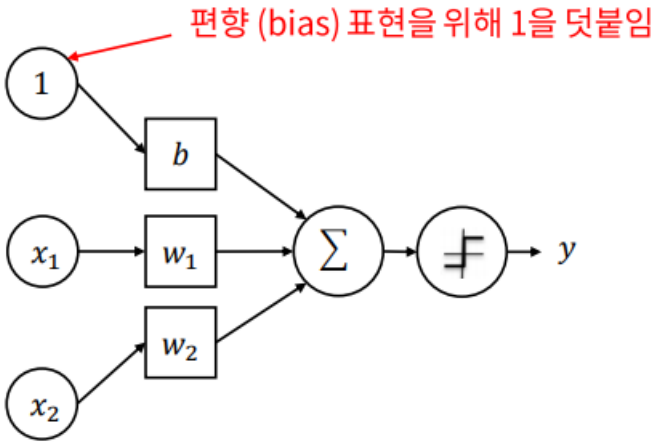


$$\sum_{i=0}^{N-1} x_i * w_i$$

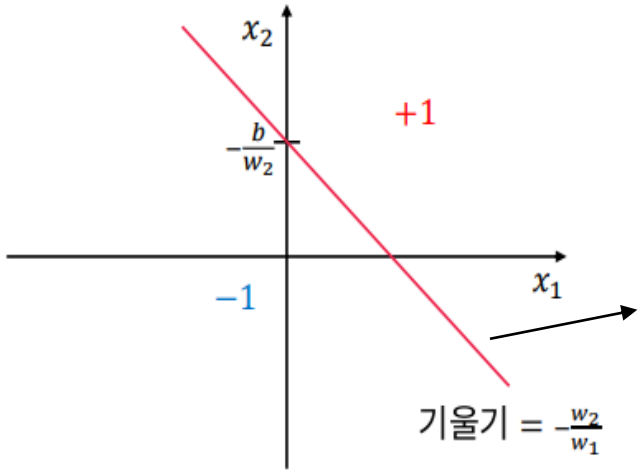
- 신경망의 가장 중요한 기본 단위
- 입력값과 활성 함수를 사용해 출력 값을 다음으로 넘기는 가장 작은 단위
- 여러 층의 퍼셉트론이 서로 연결, 조합되어 입력값에 대한 판단을 함



퍼셉트론의 동작



$$y = \begin{cases} +1, & b + w_1x_1 + w_2x_2 > 0 \\ -1, & b + w_1x_1 + w_2x_2 \leq 0 \end{cases}$$



$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

결정 경계
(이진 분류 문제 해결)

퍼셉트론의 학습

$$W_{t+1} = W_t + \theta(y - \hat{y})x$$

W_{t+1} : 업데이트 후 가중치

W_t : 업데이트 전 가중치

θ : 학습률

y : 학습 데이터 정담

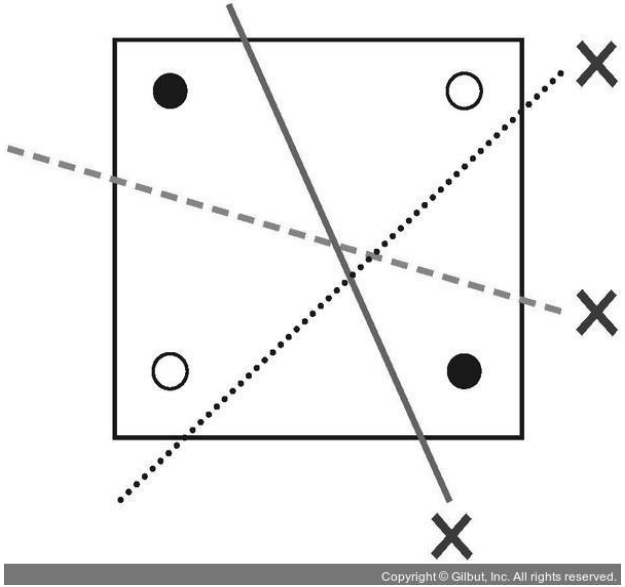
$\hat{y}$: 입력으로 추정한 출력

x : 입력 데이터

```
initialize_w(random)
for _ in range(max_iter):
    for x, y in zip(X, Y):
        h = dot_product(x, w)
        y_ = activation_func(h)
        w = w + learning_rate * (y-y_) * x
```

1. 임의로 선을 긋는다
2. 입력을 하나씩 넣어서 출력을 내본다
3. 정답과 비교해서 틀린 경우 선을 옮겨 다시 긋는다

퍼셉트론의 한계



- XOR (exclusive OR) 문제
- 선을 아무리 그어도 하나의 직선으로는 해결되지 않는다

게이트(Gate)

AND 진리표

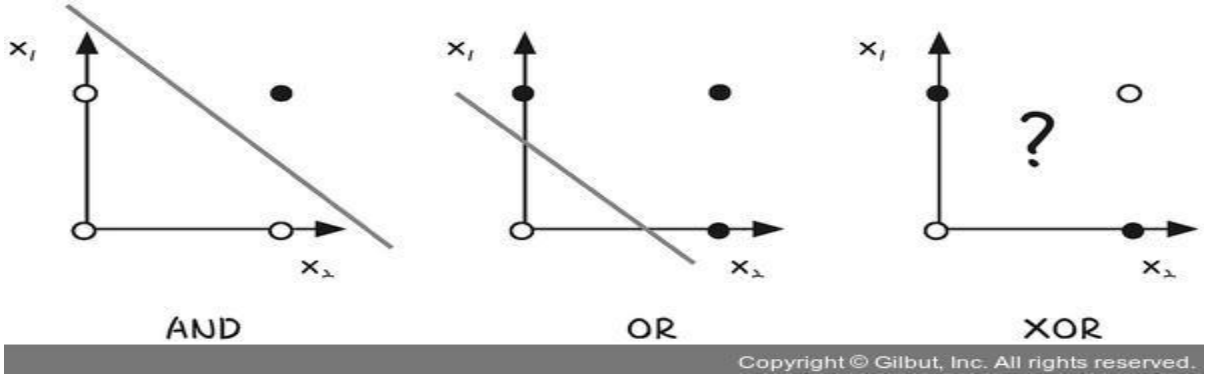
X1	X2	결과값
0	0	0
0	1	0
1	0	0
1	1	1

OR 진리표

X1	X2	결과값
0	0	0
0	1	1
1	0	1
1	1	1

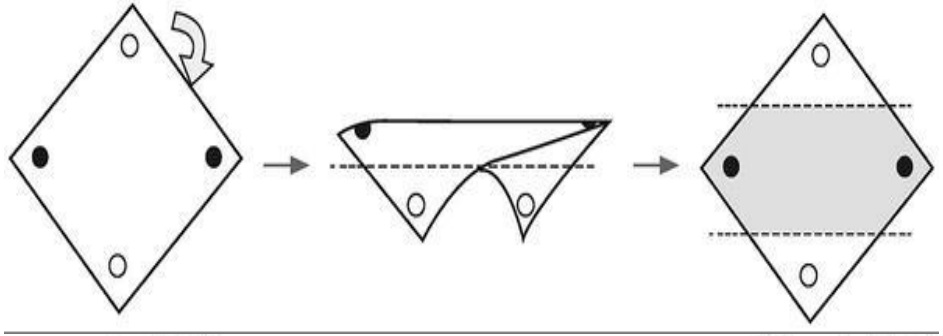
XOR 진리표

X1	X2	결과값
0	0	0
0	1	1
1	0	1
1	1	0

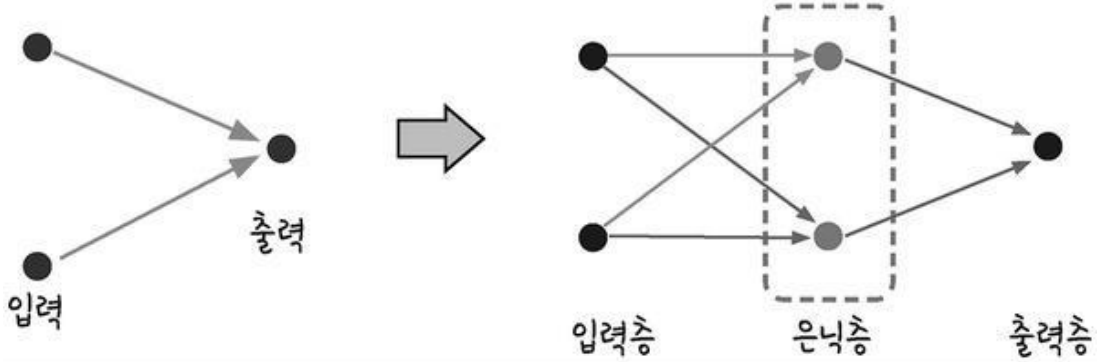


- XOR 문제에 대한 새로운 접근법 필요

다층 퍼셉트론

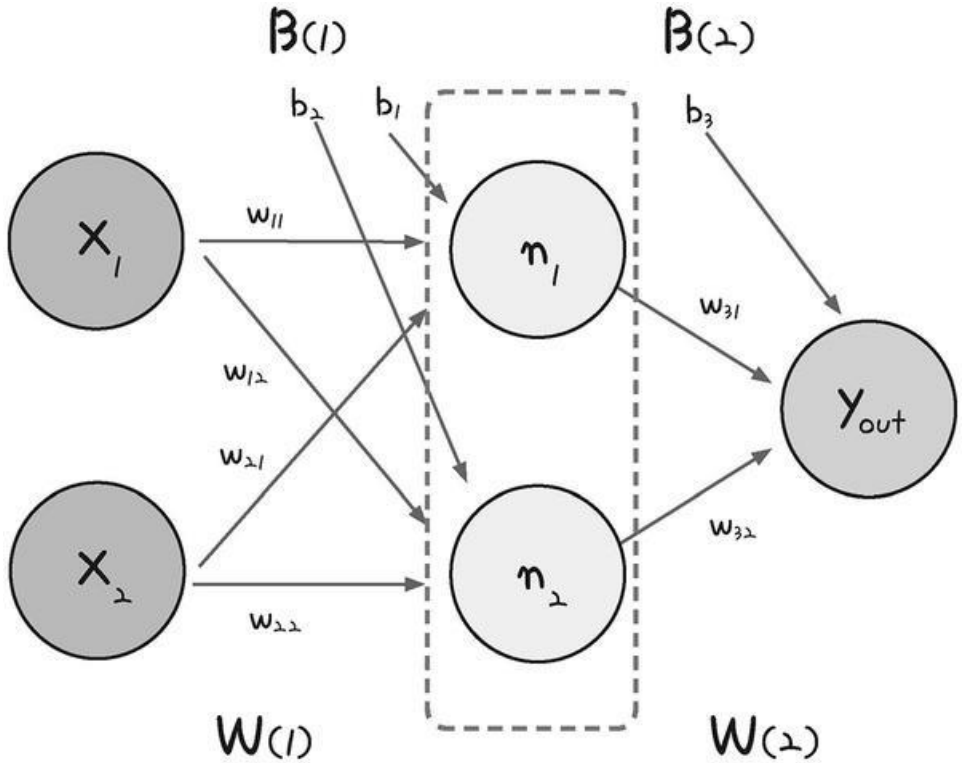


- 좌표 평면 자체에 변화를 주기



- 두개의 퍼셉트론을 한번에 계산
- 은닉층 : 좌표평면을 왜곡

다층 퍼셉트론 설계



- $n_1 = \sigma(x_1 w_{11} + x_2 w_{21} + b_1) : x * w$
- $n_2 = \sigma(x_1 w_{12} + x_2 w_{22} + b_2) : x * w$
- $y_{out} = \sigma(n_1 w_{31} + n_2 w_{31} + b_3) : \mathrm{n} * w$

- $w(1) = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$ $B(1) = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$
- $w(2) = \begin{bmatrix} w_{31} \\ w_{32} \end{bmatrix}$ $B(2) = [b_3]$

XOR 문제 해결

- $w(1) = \begin{bmatrix} -2 & 2 \\ -2 & 2 \end{bmatrix}$ $B(1) = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$
- $w(2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $B(2) = [-1]$

x_1	x_2	n_1	n_2	y_{out}	원하는 값
0	0	$\sigma(0*(-2)+0*(-2)+3) \approx 1$	$\sigma(0*2+0*2-1) \approx 0$	$\sigma(1*1+0*1-1) \approx 0$	0
0	1	$\sigma(0*(-2)+1*(-2)+3) \approx 1$	$\sigma(0*2+1*2-1) \approx 1$	$\sigma(1*1+1*1-1) \approx 1$	1
1	0	$\sigma(1*(-2)+0*(-2)+3) \approx 1$	$\sigma(1*2+0*2-1) \approx 1$	$\sigma(1*1+1*1-1) \approx 1$	1
1	1	$\sigma(1*(-2)+1*(-2)+3) \approx 0$	$\sigma(1*2+1*2-1) \approx 1$	$\sigma(0*1+1*1-1) \approx 0$	0

Q & A