AI 프로그래밍

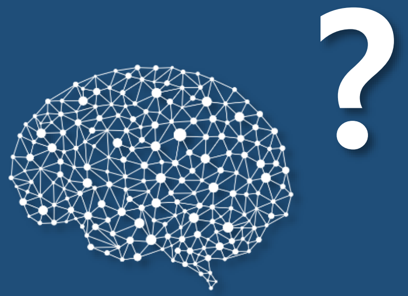
ANN를 위한 선형대수학 및 미분

01

ANN의 구조

ANN의 구조

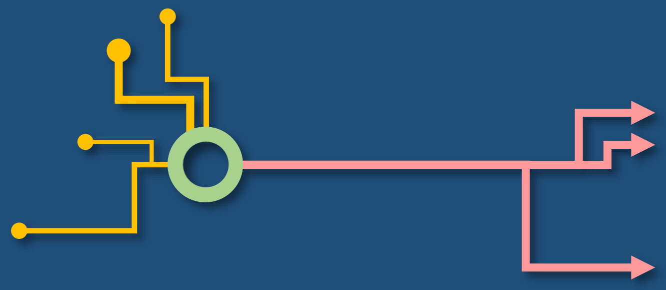
- ANN
  - 현대 AI는 ANN (인공 신경망, Artificial Neural Network)을 활용
    - ANN이란 동물의 뇌를 소프트웨어로 구현한 모델
    - 인간이 인지하지 못했던 feature (특징)를 식별하고 활용하기 위해 자연을 모방



    - 신경망을 이루는 많은 수의 뉴런을 어떻게 구현할 것인가?

ANN의 구조

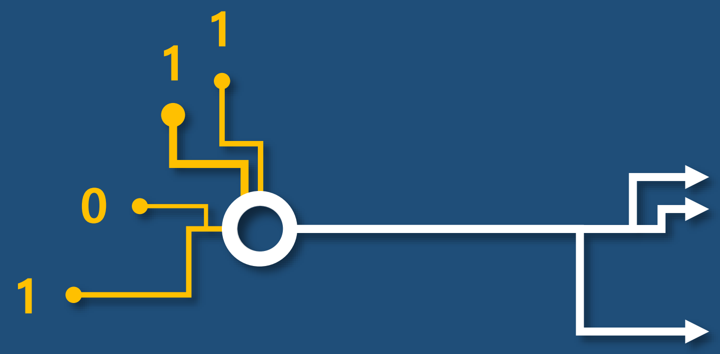
- 인공 뉴런의 구조



- 가지 돌기 : 외부 입력이나 다른 뉴런으로부터 신호를 전달 받음
- 세포체 : 전달 받은 신호를 취합, 역치(threshold)를 초과할 경우 출력 신호 생성
- 축색 돌기 : 세포체가 생성한 신호를 다른 뉴런이나 외부 장치로 전달

ANN의 구조

- 인공 뉴런의 작동 방식



1. 외부에서 입력 신호가 들어옴
2. 각 입력의 중요도에 따라 weight (가중치)를 곱하여 bias (편차)와 합산
3. 합산한 신호의 양의 역치(threshold)보다 큰지 확인
4. 출력 신호를 생산하여 다른 뉴런에게 전달

ANN의 구조

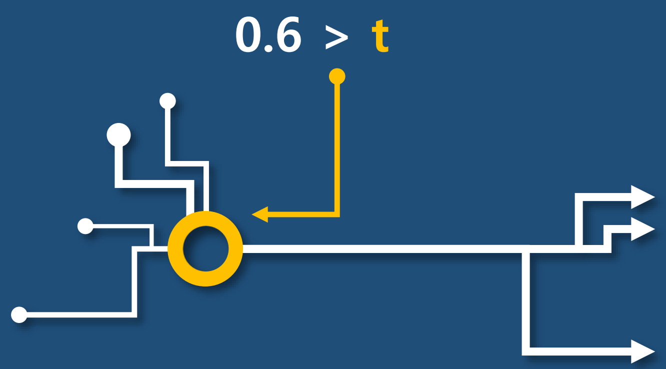
- 인공 뉴런의 작동 방식

0.2×1 + 0.1×0 + 0.5×1 + 0.2×1 + -0.3 = 0.6

1. 외부에서 입력 신호가 들어옴
2. 각 입력의 중요도에 따라 weight (가중치)를 곱하여 bias (편차)와 합산
3. 합산한 신호의 양의 역치(threshold)보다 큰지 확인
4. 출력 신호를 생산하여 다른 뉴런에게 전달

ANN의 구조

- 인공 뉴런의 작동 방식



1. 외부에서 입력 신호가 들어옴
2. 각 입력의 중요도에 따라 weight (가중치)를 곱하여 bias (편차)와 합산
3. 합산한 신호의 양의 역치(threshold)보다 큰지 확인
4. 출력 신호를 생산하여 다른 뉴런에게 전달

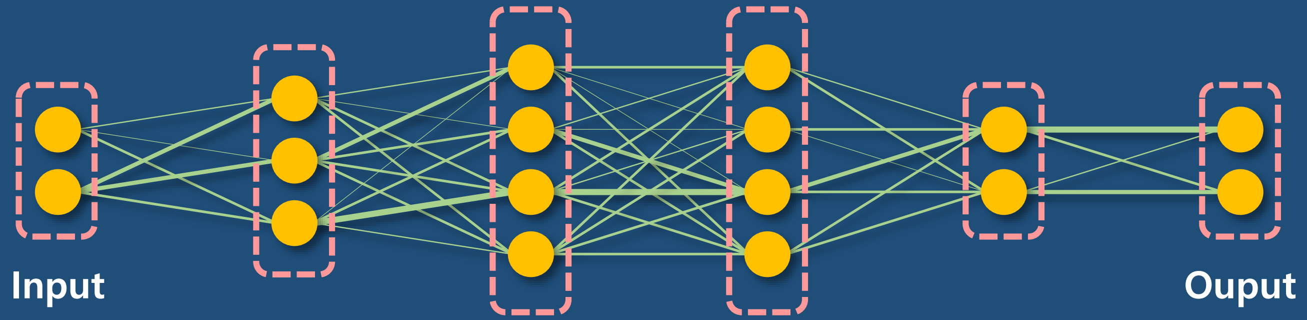
ANN의 구조

- 인공 뉴런의 작동 방식

1. 외부에서 입력 신호가 들어옴
2. 각 입력의 중요도에 따라 weight (가중치)를 곱하여 bias (편차)와 합산
3. 합산한 신호의 양의 역치(threshold)보다 큰지 확인
4. 출력 신호를 생산하여 다른 뉴런에게 전달

ANN의 구조

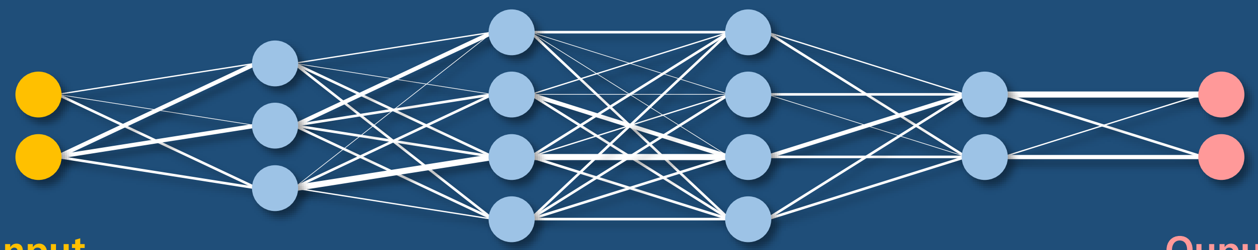
- ANN의 구조



- Node (정점) : 각 뉴런의 세포체, 신호 값을 저장함
- Edge (간선) : 각 뉴런 사이를 잇는 돌기, 각자 다른 weight (가중치)를 가질 수 있음
- Layer (층) : 같은 레벨의 node들, 고유의 bias (편차)를 가짐

ANN의 구조

- ANN의 구조



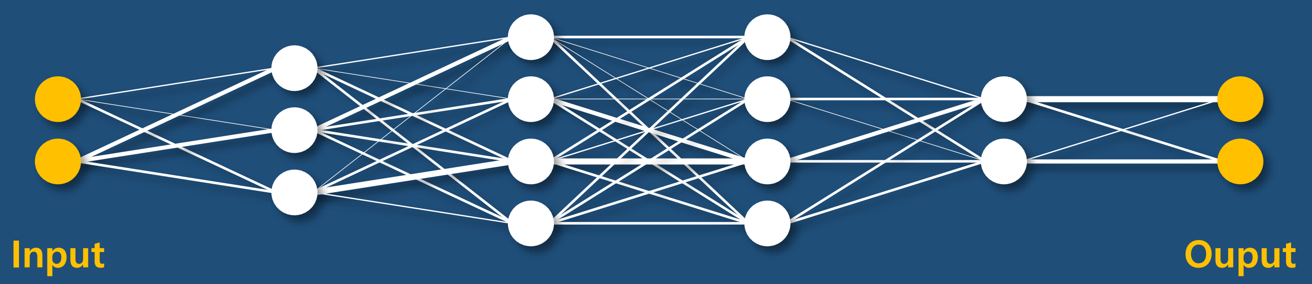
- Input layer (입력층) : ANN의 첫 layer, 외부 input을 받아 저장함
- Hidden layer (은닉층) : Input layer와 output layer 사이에 숨어있는 중간 layer
- Output layer (출력층) : ANN의 마지막 layer, 목적에 따라 다양한 종류의 output 생성

ANN의 구조

- ANN의 구조
  - Output layer의 종류
    - Regression layer
      - Layer의 각 node가 범위가 무한대인 실수의 값을 그대로 출력
      - 주로 연속된 값(continuous value)을 예측해야 하는 ANN에 활용
    - Binary classifier layer
      - Layer의 각 node가 확률을 출력, 0.5를 기준으로 0 또는 1로 해석
      - 주로 0 또는 1로 이루어진 데이터를 생성하는 ANN에 활용
    - Softmax layer
      - Layer의 노드 값 합이 1, 최대 값을 가진 node만 1, 나머지는 0으로 해석
      - 주로 이산 선택(discrete choice)을 해야 하는 ANN에 활용

ANN의 구조

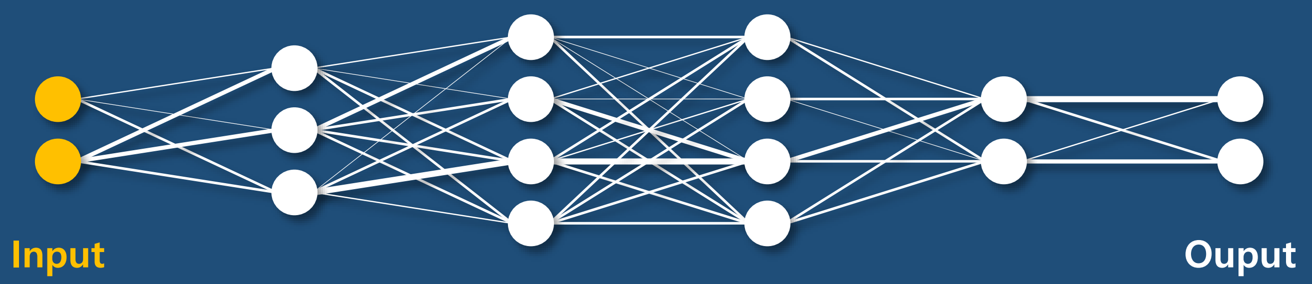
- ANN의 inference



- Input이 주어지면 각 layer를 차례대로 거치며 뉴런의 연산이 일어남
- 최종 layer의 연산을 마치면 ANN의 output이 생성됨
- 주어진 input으로부터 output을 구하는 과정을 inference (추론)이라 함
- Input에서부터 output으로 전파되므로 feed-forward라고도 함

ANN의 구조

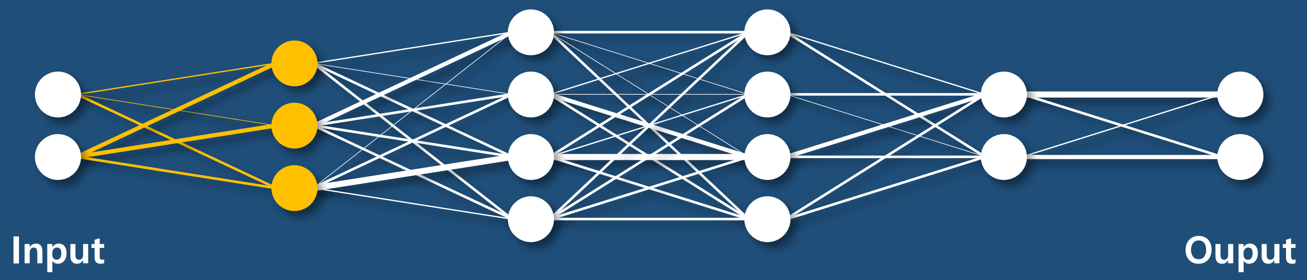
- ANN의 inference



1. Input layer의 각 node에 input 값 저장
2. 각 뉴런의 연산 결과를 다음 layer로 전달
3. Output layer까지 반복, 최종 output 생성

ANN의 구조

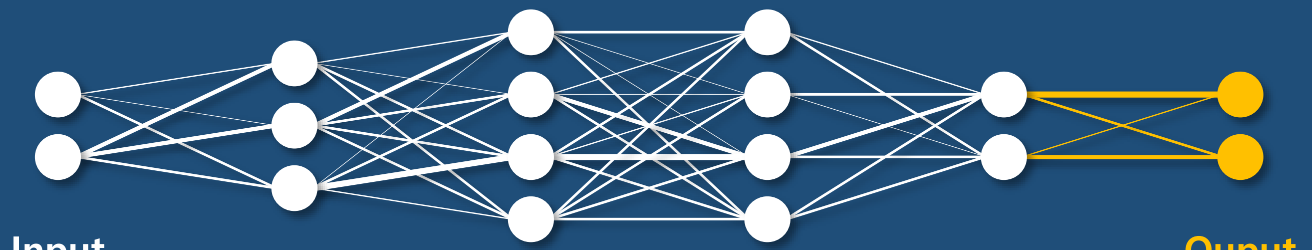
- ANN의 inference



1. Input layer의 각 node에 input 값 저장
2. 각 뉴런의 연산 결과를 다음 layer로 전달
3. Output layer까지 반복, 최종 output 생성

ANN의 구조

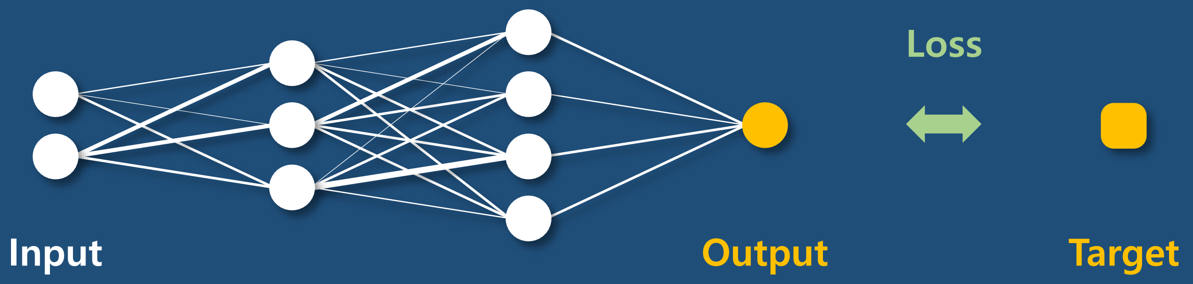
- ANN의 inference



1. Input layer의 각 node에 input 값 저장
2. 각 뉴런의 연산 결과를 다음 layer로 전달
3. Output layer까지 반복, 최종 output 생성

ANN의 구조

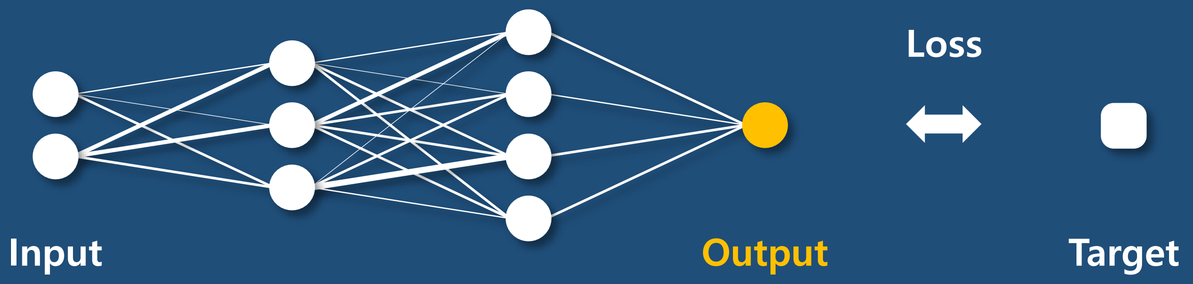
- ANN의 training



- ANN은 feed-forward를 통해 inference 수행, output을 생성함
- 주어진 input으로부터 올바른 output을 구하려면 올바른 weight와 bias가 필요함
- Output과 target을 비교해 올바른 weigh와 bias를 구하는 training (학습)과정이 필요함
- Output에서부터 input으로 전파되므로 back-propagation이라고도 함

ANN의 구조

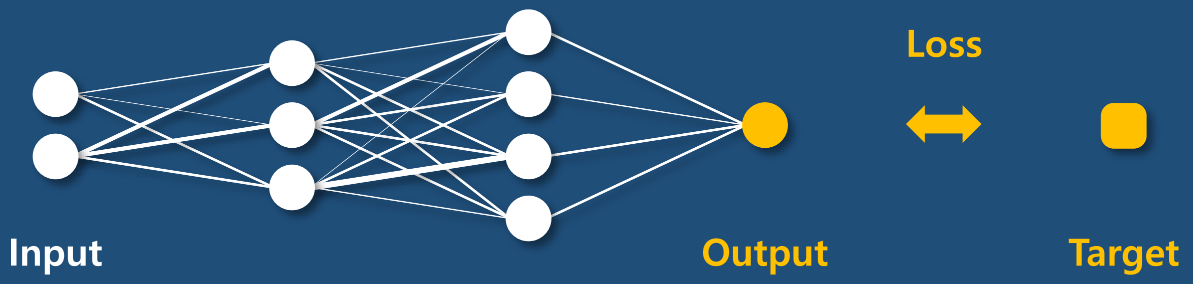
- ANN의 training



1. Feed-forward를 통해 inference 수행, 주어진 input에 대한 output 생성
2. Output과 target을 비교해 error의 정도를 loss로 표현
3. Output layer부터 loss를 줄이기 위한 weight와 bias의 변화량을 구함
4. Input layer까지 반복
5. ANN 전체의 weight와 bias를 수정

ANN의 구조

- ANN의 training



1. Feed-forward를 통해 inference 수행, 주어진 input에 대한 output 생성
2. Output과 target을 비교해 error의 정도를 loss로 표현
3. Output layer부터 loss를 줄이기 위한 weight와 bias의 변화량을 구함
4. Input layer까지 반복
5. ANN 전체의 weight와 bias를 수정

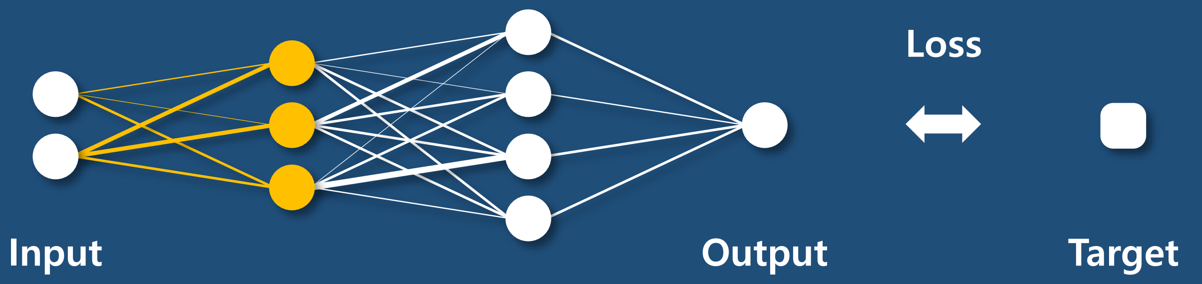
ANN의 구조

- ANN의 training

1. Feed-forward를 통해 inference 수행, 주어진 input에 대한 output 생성
2. Output과 target을 비교해 error의 정도를 loss로 표현
3. Output layer부터 loss를 줄이기 위한 weight와 bias의 변화량을 구함
4. Input layer까지 반복
5. ANN 전체의 weight와 bias를 수정

ANN의 구조

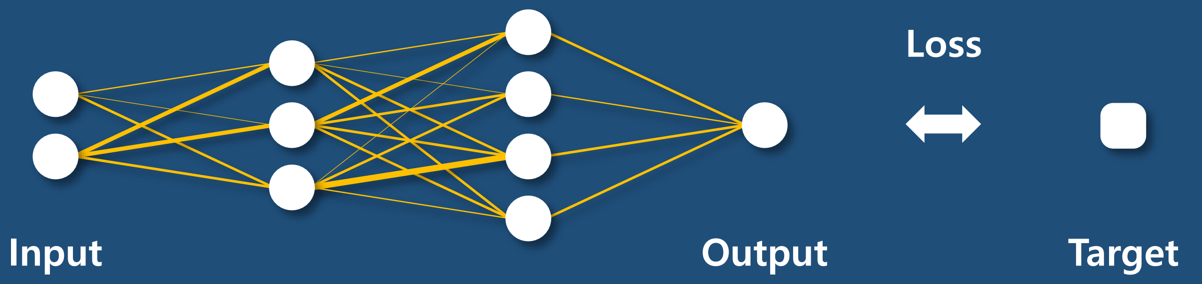
- ANN의 training



1. Feed-forward를 통해 inference 수행, 주어진 input에 대한 output 생성
2. Output과 target을 비교해 error의 정도를 loss로 표현
3. Output layer부터 loss를 줄이기 위한 weight와 bias의 변화량을 구함
4. Input layer까지 반복
5. ANN 전체의 weight와 bias를 수정

ANN의 구조

- ANN의 training



1. Feed-forward를 통해 inference 수행, 주어진 input에 대한 output 생성
2. Output과 target을 비교해 error의 정도를 loss로 표현
3. Output layer부터 loss를 줄이기 위한 weight와 bias의 변화량을 구함
4. Input layer까지 반복
5. ANN 전체의 weight와 bias를 수정

02

ANN을 위한 선형대수학

ANN을 위한 선형대수학

- 선형대수학(linear algebra)
  - 벡터 공간의 개체들의 선형 표현 및 이동을 다루는 수학
  - 선형성(linearity)이란 어떠한 개체를 아래와 같이 1차 함수로 표현할 수 있는 성질을 뜻함

$$y = f(\vec{w}, \vec{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

  - 벡터 공간의 개체들은 차원에 따라 아래와 같이 분류함
    - 스칼라(scalar) : 양을 나타내는 하나의 수, 소문자 얇은 글꼴로 표시($x \in \mathbb{R}^1$)
    - 벡터(vector) : 방향을 나타내는 1차원 수열, 소문자 굵은 글꼴로 표시($\vec{\boldsymbol{x}} \in \mathbb{R}^n$)
    - 행렬(matrix) : 2차원 수열, 대문자 굵은 글꼴로 표시($\boldsymbol{X} \in \mathbb{R}^{m \times n}$)
    - 텐서(tensor) : 3차원 이상의 수열, 대문자 굵은 글꼴로 표시($\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$)

ANN을 위한 선형대수학

- 선형대수학(linear algebra)
  - 벡터(vector)의 차원
    - 기본적으로 벡터는 **세로**(column) 방향으로 표현함
      - $\vec{v} \in \mathbb{R}^n$
      - $\vec{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$
      - $\vec{v}^T = (v_1 \ \cdots \ v_n)$
      - $(\vec{v}^T)^T = \vec{v}$

ANN을 위한 선형대수학

- 선형대수학(linear algebra)
  - 벡터(vector)의 점곱(dot product)
    - 스칼라곱(scalar product)라고도 함
    - 두 벡터의 차원이 같을 때만 가능
    - 앞의 벡터는 가로(row) 방향으로 전치(transpose)되어 있어야 함
      - $\vec{a} \in \mathbb{R}^n$, $\vec{b} \in \mathbb{R}^n$
      - $\vec{a}^T \cdot \vec{b} = (a_1 \quad \cdots \quad a_n)\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$

$$= a_1 b_1 + \cdots + a_n b_n$$

$$= s \in \mathbb{R}^1$$

ANN을 위한 선형대수학

- 선형대수학(linear algebra)
  - 행렬(matrix)의 차원
    - m×n의 행렬이 있을 때, m개의 행과 n개의 열로 이루어짐
      - $\boldsymbol{M} \in \mathbb{R}^{m \times n}$
      - $\boldsymbol{M} = \begin{pmatrix} m_{1,1} & \cdots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{m,1} & \cdots & m_{m,n} \end{pmatrix}$
      - $\boldsymbol{M}^T \in \mathbb{R}^{n \times m}$
      - $\boldsymbol{M}^T = \begin{pmatrix} m_{1,1} & \cdots & m_{m,1} \\ \vdots & \ddots & \vdots \\ m_{1,n} & \cdots & m_{m,n} \end{pmatrix}$

ANN을 위한 선형대수학

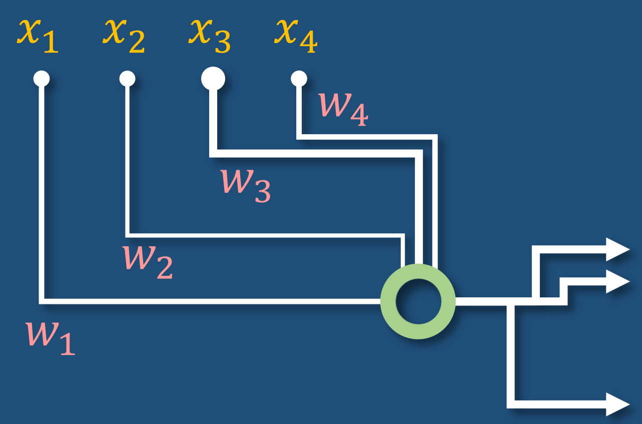
- 선형대수학(linear algebra)
  - 행렬(matrix)의 점곱(dot product)
    - 앞 행렬의 열 차원과 뒤 행렬의 행 차원이 같을 때만 가능
      - $\boldsymbol{A} \in \mathbb{R}^{i \times j}$, $\boldsymbol{B} \in \mathbb{R}^{j \times k}$
      - $\boldsymbol{A} \cdot \boldsymbol{B} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,j} \\ \vdots & \ddots & \vdots \\ a_{i,1} & \cdots & a_{i,j} \end{pmatrix} \begin{pmatrix} b_{1,1} & \cdots & b_{1,k} \\ \vdots & \ddots & \vdots \\ b_{j,1} & \cdots & b_{j,k} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} + \cdots + a_{1,j}b_{j,1} & \cdots & a_{1,1}b_{1,k} + \cdots + a_{1,j}b_{j,k} \\ \vdots & \ddots & \vdots \\ a_{i,1}b_{1,1} + \cdots + a_{i,j}b_{j,1} & \cdots & a_{i,1}b_{1,k} + \cdots + a_{i,j}b_{j,k} \end{pmatrix}$
      - $\boldsymbol{A} \cdot \boldsymbol{B} \in \mathbb{R}^{i \times k}$

ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - 뉴런 작동 방식의 선형대수학적 표현
    - Input : $\vec{x} = (x_1, x_2, x_3, x_4)^T$
    - Weight : $W = (w_1, w_2, w_3, w_4)$
    - Bias : $b \in \mathbb{R}^1$
    - Output : $y \in \mathbb{R}^1$
    - $y = W \cdot \vec{x} + b$

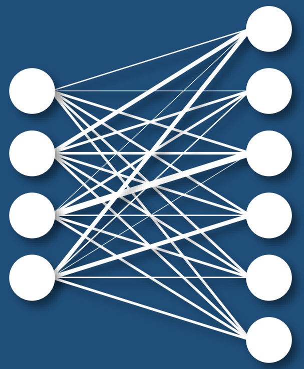
$$= (w_1, w_2, w_3, w_4)\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + b$$

$$= w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$



ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN feed-forward의 선형대수학적 표현
    - Input : $\vec{x} \in \mathbb{R}^4$
    - Weight : $\boldsymbol{W} \in \mathbb{R}^{6\times 4}$
    - Bias : $b \in \mathbb{R}^1$
    - Output : $\vec{y} \in \mathbb{R}^6$



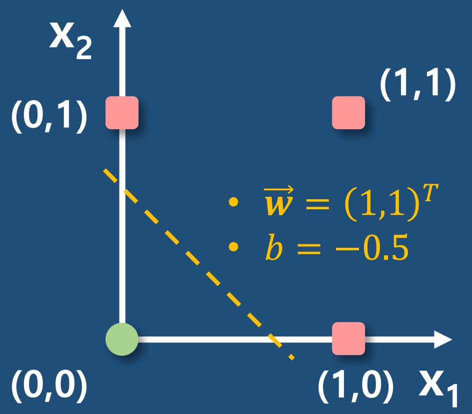
- $\vec{y} = \boldsymbol{W} \cdot \vec{x} + b$

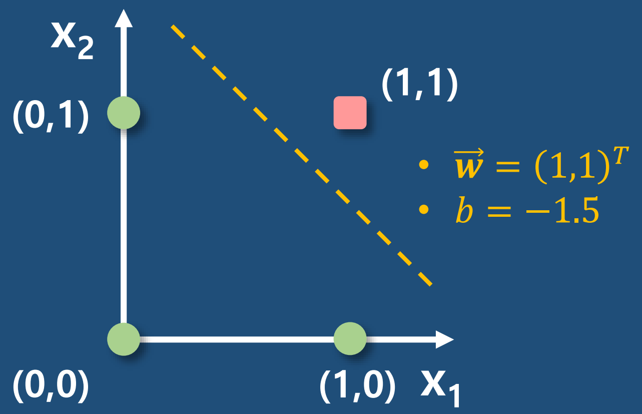
$$= \begin{pmatrix} w_{1,1} & \cdots & w_{1,4} \\ \vdots & \ddots & \vdots \\ w_{6,1} & \cdots & w_{6,4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + b$$

$$= \begin{pmatrix} w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 + w_{1,4}x_4 + b \\ \vdots \\ w_{6,1}x_1 + w_{6,2}x_2 + w_{6,3}x_3 + w_{6,4}x_4 + b \end{pmatrix}$$

ANN을 위한 선형대수학

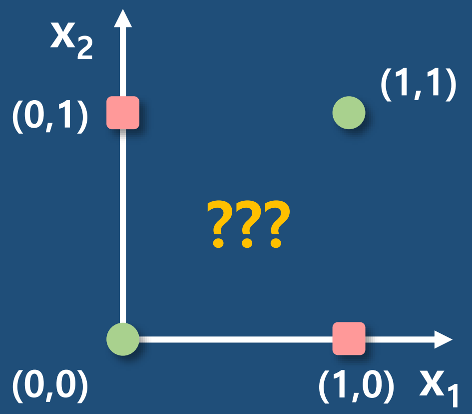
- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - 아래 데이터를 선형대수학을 이용해 색에 따라 분류하시오





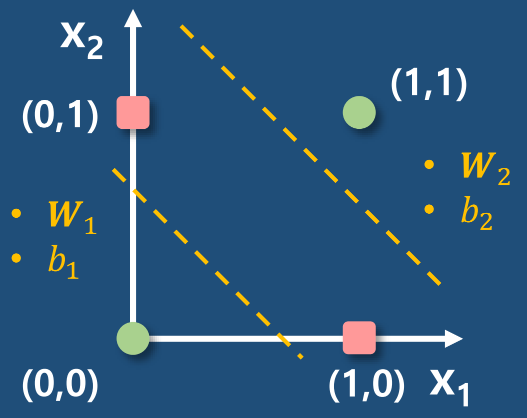
ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - 아래 데이터를 선형대수학을 이용해 색에 따라 분류하시오



ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - 아래 데이터를 선형대수학을 이용해 색에 따라 분류하시오



    - 2개의 결정 경계(decision boundary)을 모두 활용하여 구분 가능
      - 2개의 결정 경계를 어떻게 선형대수학으로 표현하는가?

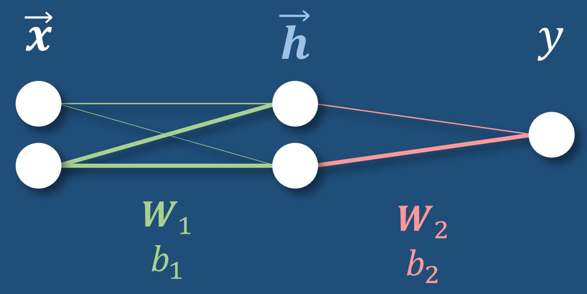
ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - $\vec{h} = \boldsymbol{W}_1 \cdot \vec{\boldsymbol{x}} + b_1$
    - $y = \boldsymbol{W}_2 \cdot \vec{h} + b_2$

⬇

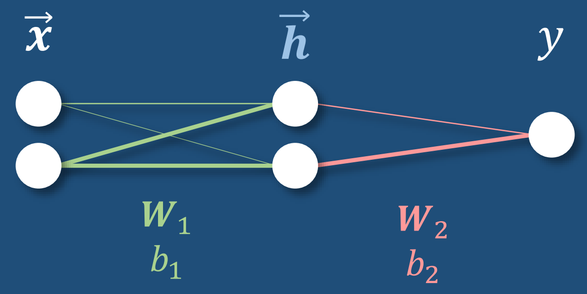
- $\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} w_{1,1,1} & w_{1,1,2} \\ w_{1,2,1} & w_{1,2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b_1$
- $y = \begin{pmatrix} w_{2,1} & w_{2,2} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} + b_2$

  $= ???$



ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학



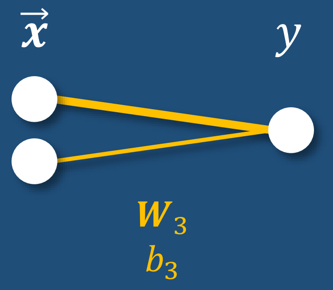
- $$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} w_{1,1,1} & w_{1,1,2} \\ w_{1,2,1} & w_{1,2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b_1$$
$$= \left(w_{1,1,1}x_1 + w_{1,1,2}x_2 + b_1,\ w_{1,2,1}x_1 + w_{1,2,2}x_2 + b_1\right)$$

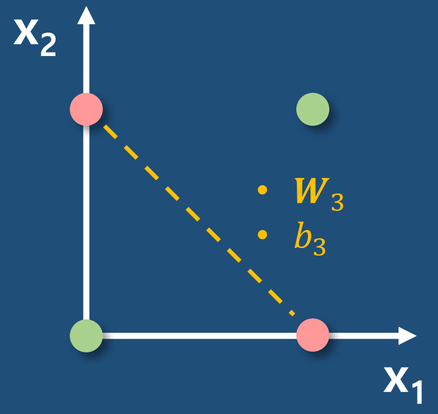
- $$y = \begin{pmatrix} w_{2,1} & w_{2,2} \end{pmatrix} \begin{pmatrix} x_1 w_{1,1,1} + x_2 w_{1,2,1} + b_1 \\ x_1 w_{1,1,2} + x_2 w_{1,2,2} + b_1 \end{pmatrix} + b_2$$
$$= w_{2,1}\left(x_1 w_{1,1,1} + x_2 w_{1,2,1} + b_1\right) + w_{2,2}(x_1 w_{1,1,2} + x_2 w_{1,2,2} + b_1) + b_2$$
$$= \left(w_{2,1}w_{1,1,1} + w_{2,2}w_{1,1,2}\right)x_1 + \left(w_{2,1}w_{1,2,1} + w_{2,2}w_{1,2,2}\right)x_2 + w_{2,1}b_1 + w_{2,2}b_1 + b_2$$
$$= \begin{pmatrix} w_{2,1}w_{1,1,1} + w_{2,2}w_{1,1,2} & w_{2,1}w_{1,2,1} + w_{2,2}w_{1,2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + w_{2,1}b_1 + w_{2,2}b_1 + b_2$$
$$= \begin{pmatrix} w_{3,1} & w_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b_3$$

ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - $\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} w_{1,1,1} & w_{1,1,2} \\ w_{1,2,1} & w_{1,2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b_1$
    - $y = \begin{pmatrix} w_{2,1} & w_{2,2} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} + b_2$

      $= \begin{pmatrix} w_{3,1} & w_{3,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b_3$
  - 선형 연산을 연달아 수행하면 하나의 선형 연산이 되어버림
  - 선형 연산만으로는 layer를 쌓을 수 없음

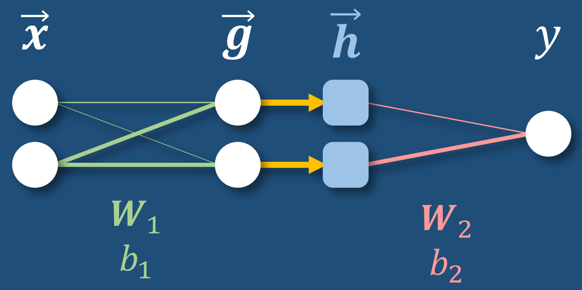




ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - Layer를 쌓기 위해선 중간 layer의 결과를 비선형(Non-linear) 함수를 통해 변형해야 함
    - Layer를 쌓음으로써 생기는 중간 layer를 Hidden layer (은닉층)라 함

- $\begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} w1_{1,1} & w1_{1,2} \\ w1_{2,1} & w1_{2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot + b_1$
- $\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} f(g_1) \\ f(g_2) \end{pmatrix}$

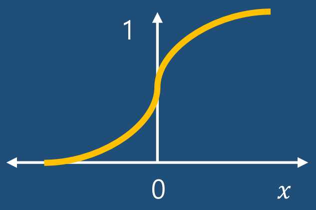


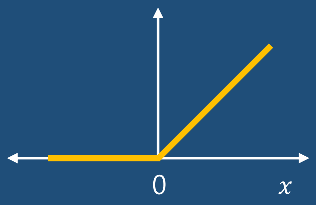
ANN을 위한 선형대수학

- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - ANN에서 주로 사용하는 비선형(Non-linear) 함수

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



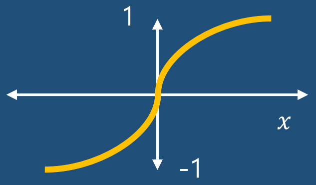


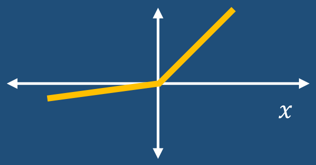
ReLU
(Rectified Linear Unit)

$$max(0, x)$$

Hyperbolic tangent

$$tanh(x)$$





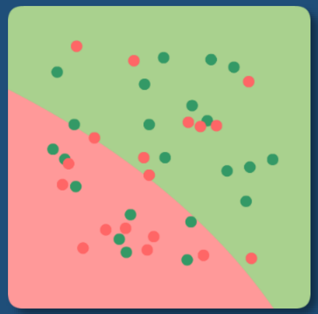
Leaky ReLU

$$max(\gamma x, x)$$
$$\gamma < 1$$

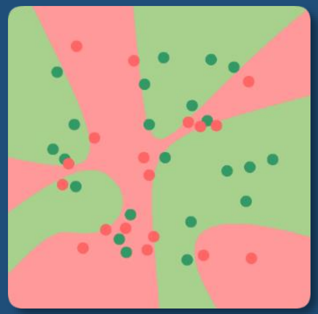
    - $f(0)$일 때 결정 경계(decision boundary)를 형성
      - 결정 경계를 역치(threshold)로 활용하므로 activation function (활성화 함수)라 부름

ANN을 위한 선형대수학

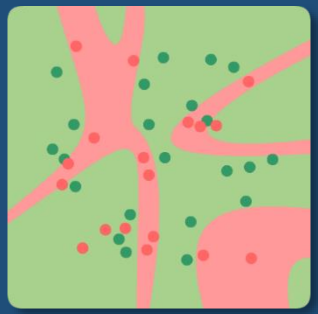
- ANN을 위한 선형대수학
  - ANN layer와 선형대수학
    - Hidden layer의 개수에 따른 ANN의 분류 성능
      https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html



0 Hidden Layer



1 Hidden Layer



4 Hidden Layers

    - Hidden layer가 많은 ANN을 학습시키는 것을 deep-learning이라 함

03

ANN을 위한 미분

ANN을 위한 미분

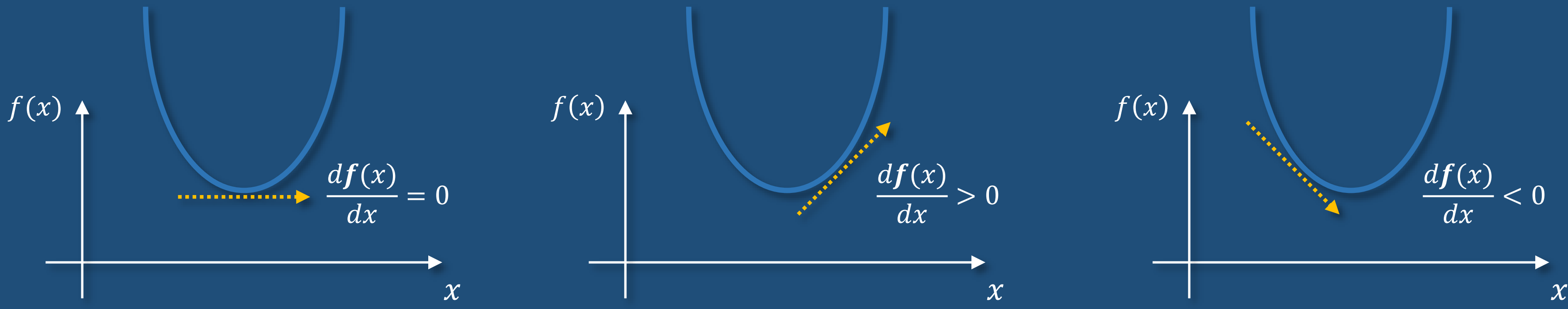
- 미분(differentiation)
  - 함수의 정의역(x)이 아주 미세하게 변할 때 치역($f(x)$)의 변화량을 구하는 도함수

$$f(x)' = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

  - 함수의 입력이 변할 때 출력이 어떻게 변할 것인지 예측, 즉 기울기를 측정하는데 유용
    - 함수의 치역이 언제 최대 또는 최소가 되는지 추정할 수 있음
    - 함수의 치역을 증가 또는 감소시키고자 할 때 정의역을 어떻게 수정해야 할지 결정할 수 있음

ANN을 위한 미분

- 미분(differentiation)



- 기울기(미분계수)가 0이 되는 지점의 치역이 최대 또는 최소값
- 기울기가 양수인 경우, 정의역이 증가하면 치역도 증가
- 기울기가 음수이면, 정의역이 증가하면 치역은 감소

ANN을 위한 미분

- 미분(differentiation)
  - 편미분(partial derivative)
    - 함수의 정의역 중 하나를 제외한 나머지를 상수로 간주
    - 고차원 함수에서 특정 차원에 대한 변화량을 구할 때 유용

$$\frac{\delta f(x_1, x_2, \cdots, x_i, \cdots, x_n)}{\delta x_i}$$

- 예시
  - $f(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$
  - $\frac{\delta f(x_1, x_2, x_3)}{\delta x_2} = \frac{\delta (w_1 x_1 + w_2 x_2 + w_3 x_3 + b)}{\delta x_2}$

$$= \frac{d w_2 x_2}{d x_2}$$

ANN을 위한 미분

- 미분(differentiation)
  - 연쇄 법칙(chain rule)
    - 합성 함수를 차례대로 미분하는 방법
    - 복잡한 합성 함수를 큰 단위로 먼저 미분해 쉽게 기울기를 구할 수 있음

    - 예시
      - $y = g(x)$
      - $z = f(y) = f\big(g(x)\big)$
      - $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$
        $= \frac{df(y)}{dy}\frac{dg(x)}{dx}$

ANN을 위한 미분

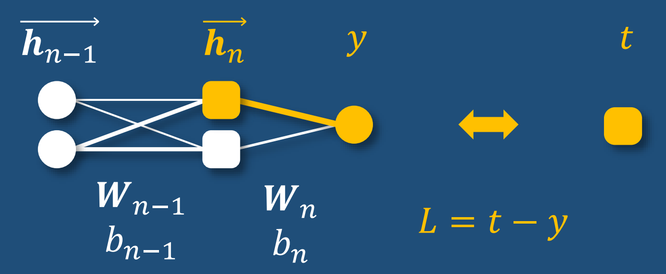
- 미분(differentiation)
  - 선형대수학과 미분
    - 스칼라를 벡터로 미분
      - $y \in \mathbb{R}^1$
      - $\vec{x} \in \mathbb{R}^n$
      - $\frac{\delta y}{\delta \vec{x}} = \begin{pmatrix} \frac{\delta y}{\delta x_1} & \cdots & \frac{\delta y}{\delta x_n} \end{pmatrix} \in \mathbb{R}^{1 \times n}$
    - 스칼라를 행렬로 미분
      - $y \in \mathbb{R}^1$
      - $\boldsymbol{X} \in \mathbb{R}^{m \times n}$
      - $\frac{\delta y}{\delta \boldsymbol{X}} = \begin{pmatrix} \frac{\delta y}{\delta x_{1,1}} & \cdots & \frac{\delta y}{\delta x_{m,1}} \\ \vdots & \ddots & \vdots \\ \frac{\delta y}{\delta x_{1,n}} & \cdots & \frac{\delta y}{\delta x_{n,m}} \end{pmatrix} \in \mathbb{R}^{n \times m}$

ANN을 위한 미분

- 미분(differentiation)
  - 선형대수학과 미분
    - 벡터를 스칼라로 미분
      - $\vec{y} \in \mathbb{R}^m$
      - $x \in \mathbb{R}^1$
      - $\frac{\delta \vec{y}}{\delta x} = \begin{pmatrix} \frac{\delta y_1}{\delta x} \\ \vdots \\ \frac{\delta y_m}{\delta x} \end{pmatrix} \in \mathbb{R}^{m \times 1}$
    - 벡터를 벡터로 미분
      - $\vec{y} \in \mathbb{R}^m$
      - $\vec{x} \in \mathbb{R}^n$
      - $\frac{\delta \vec{y}}{\delta \vec{x}} = \begin{pmatrix} \frac{\delta y_1}{\delta x_1} & \cdots & \frac{\delta y_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta y_m}{\delta x_1} & \cdots & \frac{\delta y_m}{\delta x_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$

ANN을 위한 미분

- ANN을 위한 미분
  - 개별 weight 변화에 따른 loss의 변화
    - $L = t - y$
    - $y = (w_{n,1} \quad w_{n,2}) \begin{pmatrix} h_{n,1} \\ h_{n,2} \end{pmatrix} + b_n$
      $= w_{n,1}h_{n,1} + w_{n,2}h_{n,2} + b_n$
    - $\frac{\delta L}{\delta w_{n,1}} = \frac{\delta L}{\delta y}\frac{\delta y}{\delta w_{n,1}}$
      $= \frac{\delta(t-y)}{\delta y}\frac{\delta(w_{n,1}h_{n,1}+w_{n,2}h_{n,2}+b_n)}{\delta w_{n,1}}$
      $= -1 \cdot h_{n,1}$

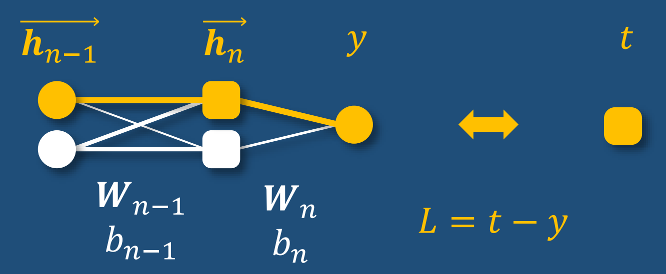


ANN을 위한 미분

- ANN을 위한 미분
  - 개별 weight 변화에 따른 loss의 변화
    - $L \quad = t - y$
    - $y \quad = \begin{pmatrix} w_{n,1} & w_{n,2} \end{pmatrix} \begin{pmatrix} h_{n,1} \\ h_{n,2} \end{pmatrix} + b_n$

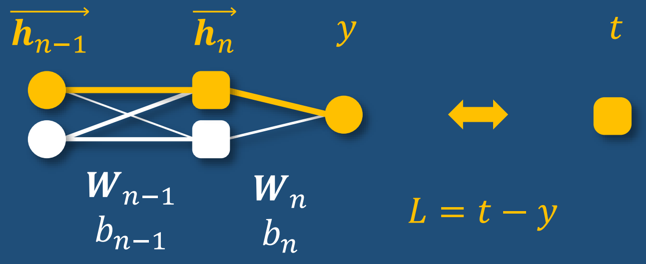
      $= w_{n,1} h_{n,1} + w_{n,2} h_{n,2} + b_n$
    - $\overrightarrow{h_n} \quad = \left(\tanh(g_{n,1}), \tanh(g_{n,2}) \right)$
    - $\overrightarrow{g_n} \quad = \begin{pmatrix} w_{n-1,1,1} & w_{n-1,1,2} \\ w_{n-1,2,1} & w_{n-1,2,2} \end{pmatrix} \begin{pmatrix} h_{n-1,1} \\ h_{n-1,2} \end{pmatrix} + b_{n-1}$

      $= \begin{pmatrix} w_{n-1,1,1} h_{n-1,1} + w_{n-1,1,2} h_{n-1,2} + b_{n-1} \\ w_{n-1,2,1} h_{n-1,1} + w_{n-1,2,2} h_{n-1,2} + b_{n-1} \end{pmatrix}$
  - $w_{n-1,1,1}$의 변화에 따른 Loss의 변화는?



ANN을 위한 미분

- ANN을 위한 미분
  - 개별 **weight** 변화에 따른 **loss**의 변화



$$\frac{\delta L}{\delta w_{n-1,1,1}} = \frac{\delta L}{\delta y}\frac{\delta y}{\delta h_{n,1}}\frac{\delta h_{n,1}}{\delta w_{n-1,1,1}}$$

$$= \frac{\delta(t-y)}{\delta y}\frac{\delta(h_{n,1}w_{n,1}+h_{n,2}w_{n,2}+b_n)}{\delta h_{n,1}}\frac{\delta h_{n,1}}{\delta w_{n-1,1,1}}$$

$$= \frac{\delta(t-y)}{\delta y}\frac{\delta(h_{n,1}w_{n,1}+h_{n,2}w_{n,2}+b_n)}{\delta h_{n,1}}\frac{\delta\tanh(g_{n,1})}{\delta g_{n,1}}\frac{\delta(g_{n,1})}{\delta w_{n-1,1,1}}$$

$$= \frac{\delta(t-y)}{\delta y}\frac{\delta(h_{n,1}w_{n,1}+h_{n,2}w_{n,2}+b_n)}{\delta h_{n,1}}\frac{\delta\tanh(g_{n,1})}{\delta g_{n,1}}\frac{\delta(w_{n-1,1,1}h_{n-1,1}+w_{n-1,1,2}h_{n-1,2}+b_{n-1})}{\delta w_{n-1,1,1}}$$

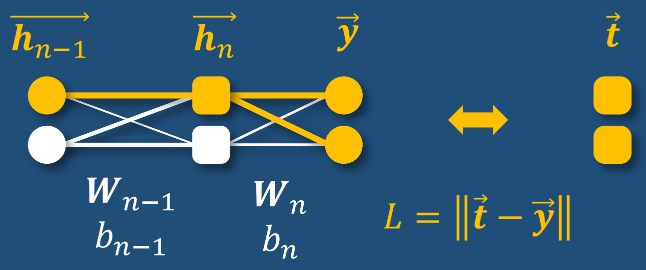
$$= -1 \cdot w_{n,1} \cdot (1 - \tanh^2(w_{n-1,1,1}h_{n-1,1} + w_{n-1,1,2}h_{n-1,2} + b_{n-1})) \cdot h_{n-1,1}$$

ANN을 위한 미분

- ANN을 위한 미분
  - 개별 weight 변화에 따른 loss의 변화
    - $L = \|\vec{t} - \vec{y}\| = \sqrt{(t_1 - y_1)^2 + (t_2 - y_2)^2}$
    - $\vec{y} = \begin{pmatrix} w_{n,1,1} & w_{n,1,2} \\ w_{n,2,1} & w_{n,2,2} \end{pmatrix} \begin{pmatrix} h_{n,1} \\ h_{n,2} \end{pmatrix} + b_n$

$$= \begin{pmatrix} w_{n,1,1} h_{n,1} + w_{n,1,2} h_{n,2} + b_n \\ w_{n,2,1} h_{n,1} + w_{n,2,2} h_{n,2} + b_n \end{pmatrix}$$

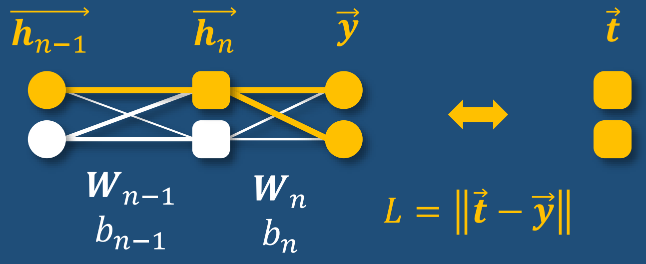
$$= \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$



- $w_{n-1,1,1}$의 변화에 따른 loss의 변화는?
  - $w_{n-1,1,1}$의 변화로 인해 $h_{n,1}$이 변하면 y_1과 y_2가 모두 영향을 받음

ANN을 위한 미분

- ANN을 위한 미분
  - 개별 weight 변화에 따른 loss의 변화



$$\frac{\delta L}{\delta w_{n-1,1,1}} = \frac{\delta L}{\delta \vec{y}} \frac{\delta \vec{y}}{\delta w_{n-1,1,1}}$$

$$= \begin{pmatrix} \frac{\delta L}{\delta y_1} & \frac{\delta L}{\delta y_2} \end{pmatrix} \begin{pmatrix} \frac{\delta y_1}{\delta w_{n-1,1,1}} \\ \frac{\delta y_2}{\delta w_{n-1,1,1}} \end{pmatrix}$$

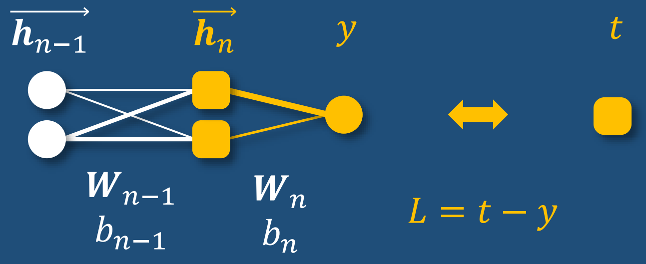
$$= \frac{\delta L}{\delta y_1} \frac{\delta y_1}{\delta w_{n-1,1,1}} + \frac{\delta L}{\delta y_2} \frac{\delta y_2}{\delta w_{n-1,1,1}}$$

$$= \frac{\delta L}{\delta y_1} \frac{\delta y_1}{\delta h_{n,1}} \frac{\delta h_{n,1}}{\delta w_{n-1,1,1}} + \frac{\delta L}{\delta y_2} \frac{\delta y_2}{\delta h_{n,1}} \frac{\delta h_{n,1}}{\delta w_{n-1,1,1}}$$

$$= \sum_i \left(\frac{\delta L}{\delta y_i} \frac{\delta y_i}{\delta h_{n,1}} \frac{\delta h_{n,1}}{\delta w_{n-1,1,1}} \right)$$

ANN을 위한 미분

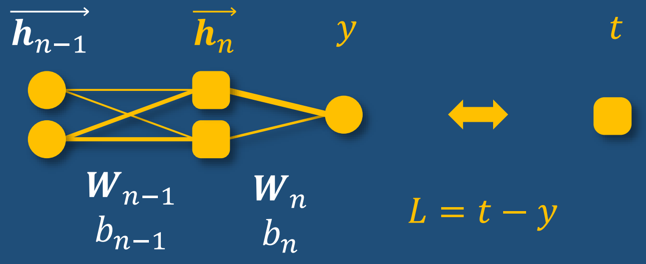
- ANN을 위한 미분
  - Layer별 bias 변화에 따른 loss의 변화



- $L = t - y$
- $y = (w_{n,1} \quad w_{n,2}) \begin{pmatrix} h_{n,1} \\ h_{n,2} \end{pmatrix} + b_n$
  $= w_{n,1}h_{n,1} + w_{n,2}h_{n,2} + b_n$
- $\dfrac{\delta L}{\delta b_n} = \dfrac{\delta L}{\delta y}\dfrac{\delta y}{\delta b_n}$
  $= \dfrac{\delta(t-y)}{\delta y}\dfrac{\delta(h_{n,1}w_{n,1}+h_{n,2}w_{n,2}+b_n)}{\delta b_n}$
  $= -1 \cdot 1$

ANN을 위한 미분

- ANN을 위한 미분
  - Layer별 bias 변화에 따른 loss의 변화
    - $L \quad = t - y$
    - $y \quad = \begin{pmatrix} w_{n,1} & w_{n,2} \end{pmatrix} \begin{pmatrix} h_{n,1} \\ h_{n,2} \end{pmatrix} + b_n$
      $= w_{n,1} h_{n,1} + w_{n,2} h_{n,2} + b_n$
    - $\overrightarrow{h_n} \quad = \left(\tanh(g_{n,1}), \tanh(g_{n,2})\right)$
    - $\overrightarrow{g_n} \quad = \begin{pmatrix} w_{n-1,1,1} & w_{n-1,1,2} \\ w_{n-1,2,1} & w_{n-1,2,2} \end{pmatrix} \begin{pmatrix} h_{n-1,1} \\ h_{n-1,2} \end{pmatrix} + b_{n-1}$
      $= \begin{pmatrix} w_{n-1,1,1} h_{n-1,1} + w_{n-1,1,2} h_{n-1,2} + b_{n-1} \\ w_{n-1,2,1} h_{n-1,1} + w_{n-1,2,2} h_{n-1,2} + b_{n-1} \end{pmatrix}$



ANN을 위한 미분

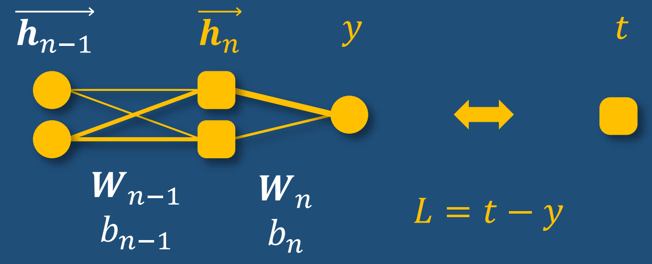
- ANN을 위한 미분
  - Layer별 bias 변화에 따른 loss의 변화

$$\frac{\delta L}{\delta b_{n-1}} = \frac{\delta L}{\delta y}\frac{\delta y}{\delta \overrightarrow{h_n}}\frac{\delta \overrightarrow{h_n}}{\delta b_{n-1}}$$

$$= \frac{\delta L}{\delta y}\begin{pmatrix}\frac{\delta y}{\delta h_{n,1}} & \frac{\delta y}{\delta h_{n,2}}\end{pmatrix}\begin{pmatrix}\frac{\delta h_{n,1}}{\delta b_{n-1}} \\ \frac{\delta h_{n,2}}{\delta b_{n-1}}\end{pmatrix}$$

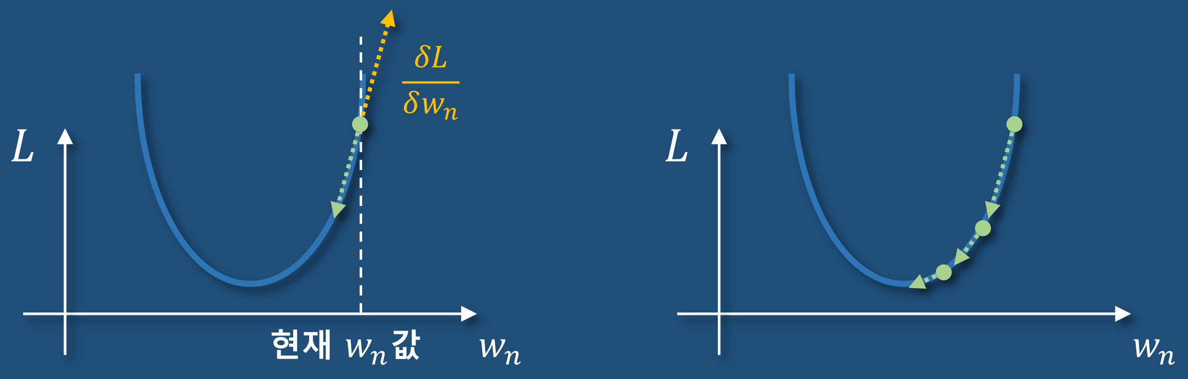
$$= \frac{\delta L}{\delta y}\left(\frac{\delta y}{\delta h_{n,1}}\frac{\delta h_{n,1}}{\delta b_{n-1}} + \frac{\delta y}{\delta h_{n,2}}\frac{\delta h_{n,2}}{\delta b_{n-1}}\right)$$

$$= \sum_i \frac{\delta L}{\delta y}\left(\frac{\delta L}{\delta h_{n,i}}\frac{\delta h_{n,i}}{\delta b_{n-1}}\right)$$



ANN을 위한 미분

- 미분을 통한 weight와 bias 수정



- 미분을 통해 각 weight 및 bias에 대한 loss 함수의 gradient(기울기)를 계산
- Loss를 줄이기 위해 각 weight와 bias를 gradient 반대 방향으로 조금씩 반복적으로 수정하여 loss 함수의 local minima(극소점)를 찾음

ANN을 위한 미분

- 미분을 통한 weight와 bias 수정
  - Loss 함수의 gradient(기울기)가 0이 되는 지점을 바로 찾지 않는 이유
    - 현대 ANN은 빅데이터를 활용함
    - Loss 함수는 주어진 데이터의 input과 target에 의해 결정됨
    - 엄청난 양의 데이터를 모두 고려하여 closed-form의 수식을 세우는 것보다 각 데이터에 따라 weight와 bias를 조금씩 수정하는 것이 더 쉬움

  - 이러한 이유와 컴퓨터 성능의 발달로 인해 weight와 bias를 조금씩 반복적으로 수정하는 방법을 선호
  - 이러한 방식의 ANN training을 gradient descent라 함

Fin

감사합니다