FALCON

딥 러닝(Deep Learning) 기초 교육 자료

Part 2

Natural Language Processing

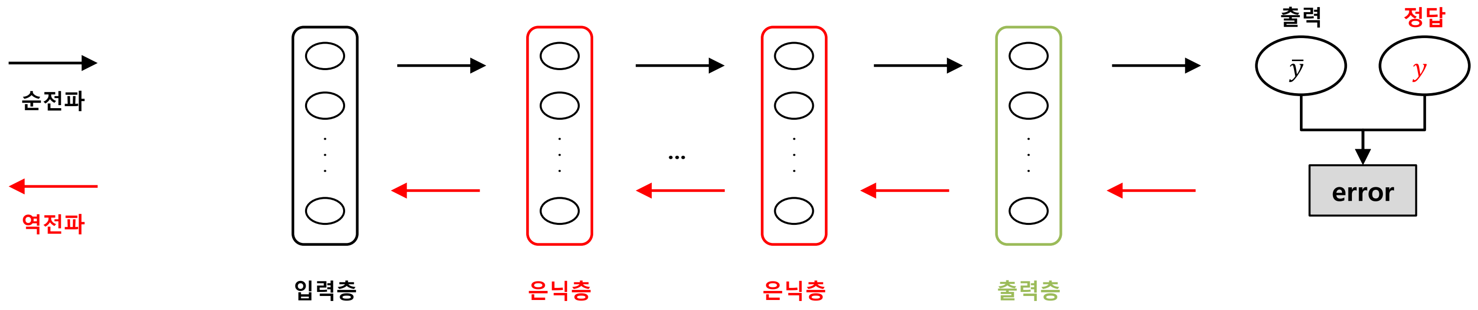
이성희

2022-04-20

목차

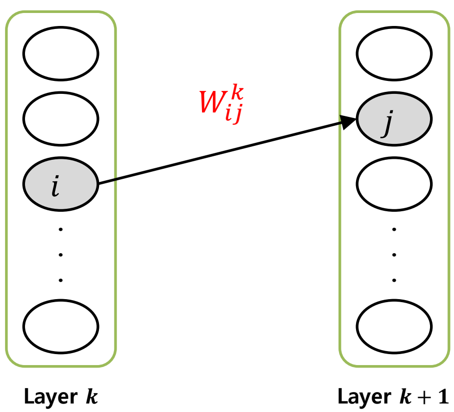
3-1. 역전파 (Backpropagation) (1/10)

- 역전파(Backpropagation)이란?
  - 먼저, 심층(인공) 신경망은 퍼셉트론으로 이루어진 층(Layer) 여러 개를 순차적으로 이어 붙인 형태이다.
    - 일반적으로 입력층에서 은닉층을 거쳐 출력층의 방향으로 연산이 수행된다.
  - 심층 신경망은 출력층에서 얻은 예측 값과 실제 정답(Label) 간의 오차(Loss)를 계산,
    이 오차가 최소가 되는 방향으로 신경망의 매개 변수들을 업데이트하고, 이 과정을 반복하는 것이 학습(Training)이다.
    - 출력층에서 계산한 오차를 이용하여, 신경망의 모든 매개 변수를 업데이트하려면 출력 값을 얻기 위한 연산 과정과는 반대로 출력층에서 은닉층을 거쳐 입력층까지 오차가 전달되어야 한다.
    - 이 때, 계산된 오차가 출력 값을 얻기 위한 과정과는 반대로 전달되면서 신경망을 학습하기 때문에, "역전파" 또는 "오차(오류) 역전파법"이라고 불린다.



3-1. 역전파 (Backpropagation) (2/10)

- 역전파(Backpropagation)를 이용한 학습 원리
  - 역전파 과정에서는 두 가지 수학적 방법을 사용한다.
    - ① Stochastic Gradient Descent (확률적 경사 하강법)
    - ② Chain Rule
    - 먼저, 경사 하강법에 의한 가중치 업데이트는 오른쪽의 수식으로 표현된다. → $W := W + \alpha\left(-\frac{\partial E(rror)}{\partial W}\right)$
    - 아래 레이어 k의 노드 i와 $k+1$의 노드 j 사이의 가중치를 W_{ij}^{k}라고 한다면, "W_{ij}^{k}가 바뀌었을 때, 에러가 얼마나 바뀌는가?"를 찾는 문제이고, 이를 W_{ji}^{k}의 변화량으로 표현한다.



$$W_{ij}^{k} := W_{ij}^{k} + \boxed{\left(-\alpha\frac{\partial E}{\partial W_{ij}^{k}}\right)}$$

$$\downarrow$$

$$\Delta W_{ij}^{k}$$

3-1. 역전파 (Backpropagation) (3/10)

- 역전파(Backpropagation)를 이용한 학습 원리
  - 즉, 역전파 과정은 "*Weight*가 바뀌면, *Error*가 얼마나 바뀌는가?"를 찾는 문제

    (※ Error가 최소가 되는 방향으로 Weight를 업데이트하는 문제)

    - W의 변화량은 W에 대한 $Error$의 편미분으로 나타낼 수 있으며, 이를 이용하여 W를 업데이트한다.

$$W_{ij}^k \coloneqq W_{ij}^k + \Delta W_{ij}^k \longrightarrow W_{ij}^k \coloneqq W_{ij}^k + \left(-\alpha \frac{\partial E}{\partial W_{ij}^k}\right)$$

    - 정답을 y, 예측 값은 $\bar{y}$, 에러는 $\frac{1}{2}(y-\bar{y})^2$, W에 대한 $\bar{y}$의 편미분을 δ(=gradient)라고 한다면, 다시 아래의 수식처럼 표현할 수 있다.

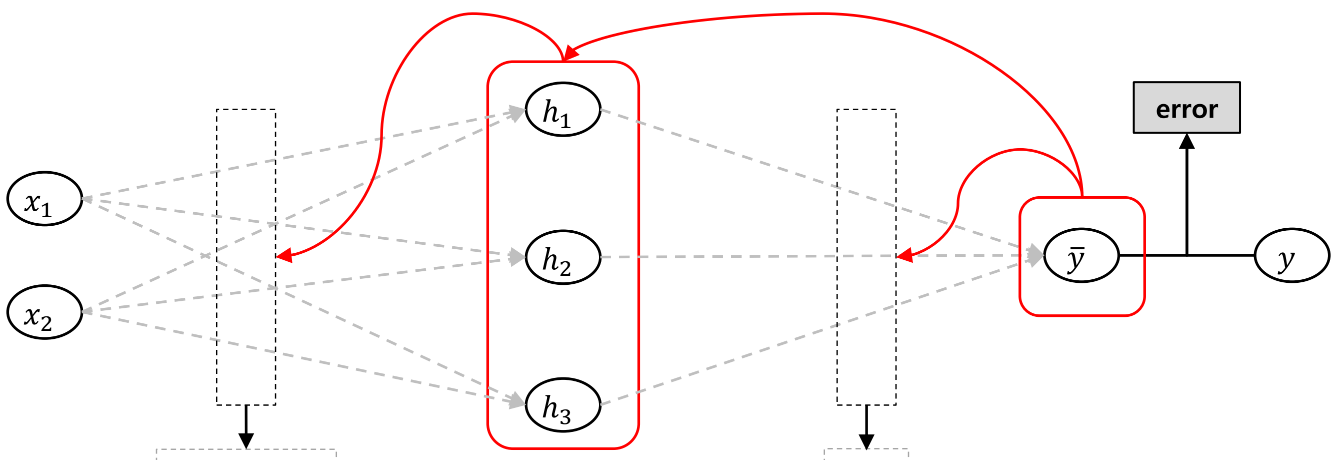
$$\Delta W_{ij}^k = -\alpha \frac{\partial E}{\partial W_{ij}^k} = -\alpha \left(\frac{\partial \frac{(y-\bar{y})^2}{2}}{\partial W_{ij}^k} \right) = -\alpha \frac{\partial}{\partial W_{ij}^k} \frac{(y-\bar{y})^2}{2} = -\alpha (y-\bar{y}) \frac{\partial}{\partial W_{ij}^k} - \bar{y}$$

$$\Delta W_{ij}^k = \alpha(y-\bar{y}) \frac{\partial \bar{y}}{\partial W_{ij}^k} = \alpha(y-\bar{y})\, \delta_{ij}^k$$

3-1. 역전파 (Backpropagation) (4/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - 신경망의 구조가 다음과 같을 때, 역전파 과정은 두 단계로 이루어진다.
    - ① $W^{(2)}$에 대한 출력 $\bar{y}$의 gradient 계산
    - ② $W^{(1)}$에 대한 (출력 $\bar{y}$의) gradient 계산

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$



$$\begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{matrix}$$

$\boldsymbol{W}^{(1)}$

$$f_1(x_1W_{11}^{(1)} + x_2W_{21}^{(1)}) = h_1$$
$$f_2(x_1W_{12}^{(1)} + x_2W_{22}^{(1)}) = h_2$$
$$f_3(x_1W_{13}^{(1)} + x_2W_{23}^{(1)}) = h_3$$

$$\begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix}$$

$\boldsymbol{W}^{(2)}$

$$(h_1W_1^{(2)} + h_2W_2^{(2)} + h_3W_3^{(2)}) = \bar{y}$$

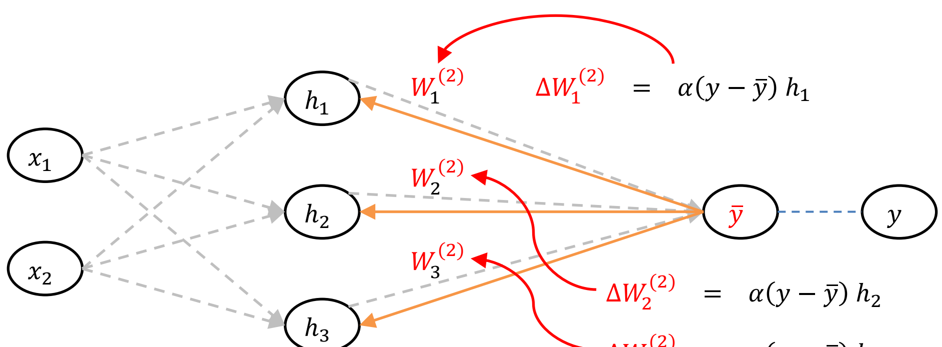
3-1. 역전파 (Backpropagation) (5/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ① $W^{(2)}$에 대한 출력 $\bar{y}$의 gradient 계산
    - hidden layer ← output

$$\frac{\partial \bar{y}}{\partial W_i^{(2)}} = \frac{\partial(\sum_i^3 h_i W_i^{(2)})}{\partial W_i^{(2)}} = \frac{\partial\left(h_1 W_1^{(2)} + h_2 W_2^{(2)} + h_3 W_3^{(2)}\right)}{\partial W_i^{(2)}} = h_i$$

    - 계산된 gradient는 결국 hidden의 출력인 h_i와 동일한 값이 되었는데, 이는 출력 $\bar{y}$가 싱글 노드이기 때문이다.

$$\Delta W_i^{(2)} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_i^{(2)}} = \alpha(y-\bar{y})\,\delta_i^{(2)} = \alpha(y-\bar{y})\,h_i$$



3-1. 역전파 (Backpropagation) (6/10)

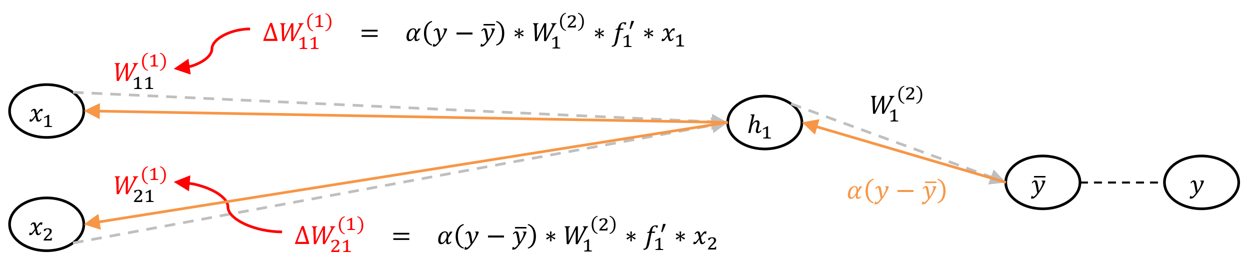
- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ② $W^{(1)}$에 대한 (출력 $\bar{y}$의) gradient 계산
    - input ← hidden layer

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$

$$\frac{\partial \bar{y}}{\partial W_{ij}^{(1)}} = \frac{\partial \bar{y}}{\partial h_j}\frac{\partial h_j}{\partial W_{ij}^{(1)}} = W_j^{(2)} * f_j' * x_i = \delta_{ij}^{(1)}$$

$$\frac{\partial \bar{y}}{\partial W_{11}^{(1)}} = \frac{\partial \bar{y}}{\partial h_1}\frac{\partial h_1}{\partial W_{11}^{(1)}} = \boxed{\frac{\partial}{\partial h_1}\left(h_1W_1^{(2)} + h_2W_2^{(2)} + h_3W_3^{(2)}\right) * \frac{\partial}{\partial W_{11}^{(1)}}\left(f_1\left(x_1W_{11}^{(1)} + x_2W_{21}^{(1)}\right)\right)} = W_1^{(2)} * f_1' * x_1$$

$$\frac{\partial \bar{y}}{\partial W_{21}^{(1)}} = \frac{\partial \bar{y}}{\partial h_1}\frac{\partial h_1}{\partial W_{21}^{(1)}} = \boxed{\frac{\partial}{\partial h_1}\left(h_1W_1^{(2)} + h_2W_2^{(2)} + h_3W_3^{(2)}\right) * \frac{\partial}{\partial W_{21}^{(1)}}\left(f_1\left(x_1W_{11}^{(1)} + x_2W_{21}^{(1)}\right)\right)} = W_1^{(2)} * f_1' * x_2$$

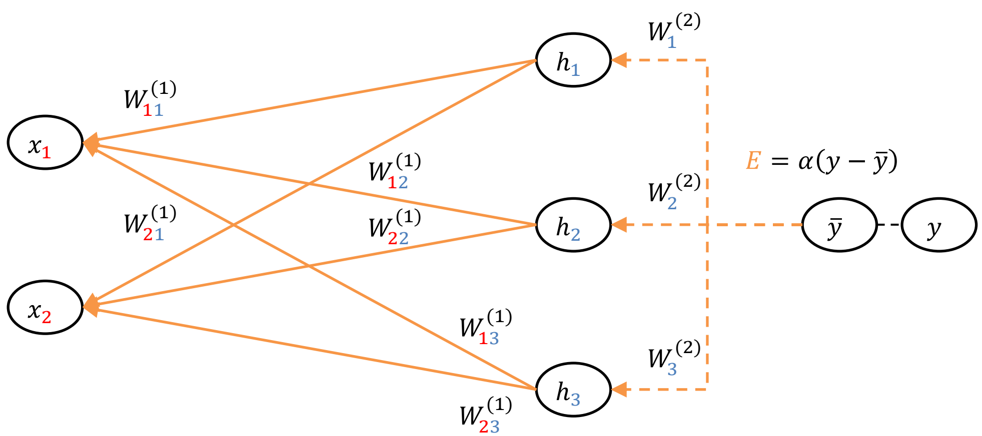


3-1. 역전파 (Backpropagation) (7/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ② $W^{(1)}$에 대한 (출력 $\bar{y}$의) gradient 계산
    - input ← hidden layer

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$

$$\frac{\partial \bar{y}}{\partial W_{ij}^{(1)}} = \frac{\partial \bar{y}}{\partial h_j}\frac{\partial h_j}{\partial W_{ij}^{(1)}} = W_j^{(2)} * f_j' * x_i = \delta_{ij}^{(1)}$$



$$\Delta W_{11}^{(1)} = E * W_1^{(2)} * f_1' * x_1$$

$$\Delta W_{21}^{(1)} = E * W_1^{(2)} * f_1' * x_2$$

$$\Delta W_{12}^{(1)} = E * W_2^{(2)} * f_2' * x_1$$

$$\Delta W_{22}^{(1)} = E * W_2^{(2)} * f_2' * x_2$$

$$\Delta W_{13}^{(1)} = E * W_3^{(2)} * f_3' * x_1$$

$$\Delta W_{23}^{(1)} = E * W_3^{(2)} * f_3' * x_2$$

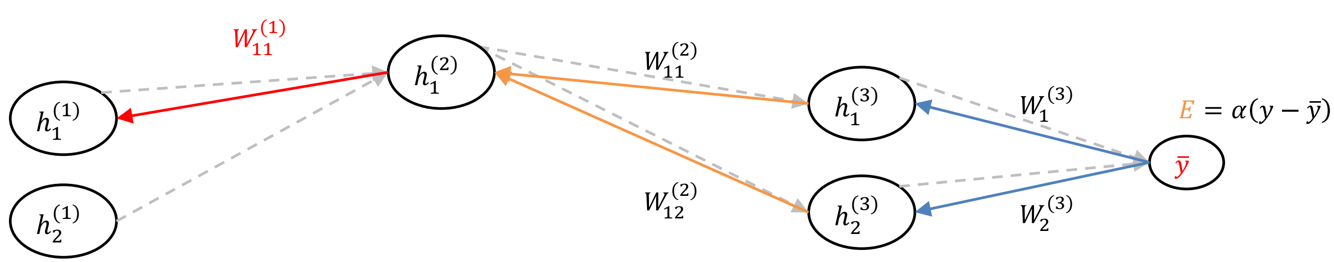
3-1. 역전파 (Backpropagation) (8/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ③ W_{ij}^{k}에 대한 (출력 $\bar{y}$의) gradient 계산
    - hidden layer ← hidden layer

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$

- 네트워크의 구조가 복잡한 신경망은 chain rule을 이용하여, W를 업데이트
  - 아래 신경망의 구조에서 $W_{11}^{(1)}$을 업데이트하기 위해선 $W_{11}^{(1)}$이 변했을 때, $\bar{y}$가 얼마나 변하는지를 계산해야 하고, $W_{11}^{(1)}$으로부터 $\bar{y}$를 얻기까지의 계산 과정은 합성 함수의 형태이므로 합성 함수의 미분 공식인 chain rule을 이용한다.

$$\frac{\partial \bar{y}}{\partial W_{11}^{(1)}} = \underbrace{\frac{\partial \bar{y}}{\partial h_{1}^{(3)}}}_{W_{1}^{(3)}}\underbrace{\frac{\partial h_{1}^{(3)}}{\partial h_{1}^{(2)}}}_{f_{1}^{(3)\prime}}\underbrace{\frac{\partial h_{1}^{(2)}}{\partial W_{11}^{(1)}}}_{f_{1}^{(2)\prime} * h_{1}^{(1)}} + \underbrace{\frac{\partial \bar{y}}{\partial h_{2}^{(3)}}}_{W_{2}^{(3)}}\underbrace{\frac{\partial h_{2}^{(3)}}{\partial h_{1}^{(2)}}}_{f_{2}^{(3)\prime}}\underbrace{\frac{\partial h_{1}^{(2)}}{\partial W_{11}^{(1)}}}_{f_{1}^{(2)\prime} * h_{1}^{(1)}}$$



3-1. 역전파 (Backpropagation) (9/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ③ W_{ij}^{k}에 대한 (출력 $\bar{y}$의) gradient 계산
    - hidden layer ← hidden layer

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$

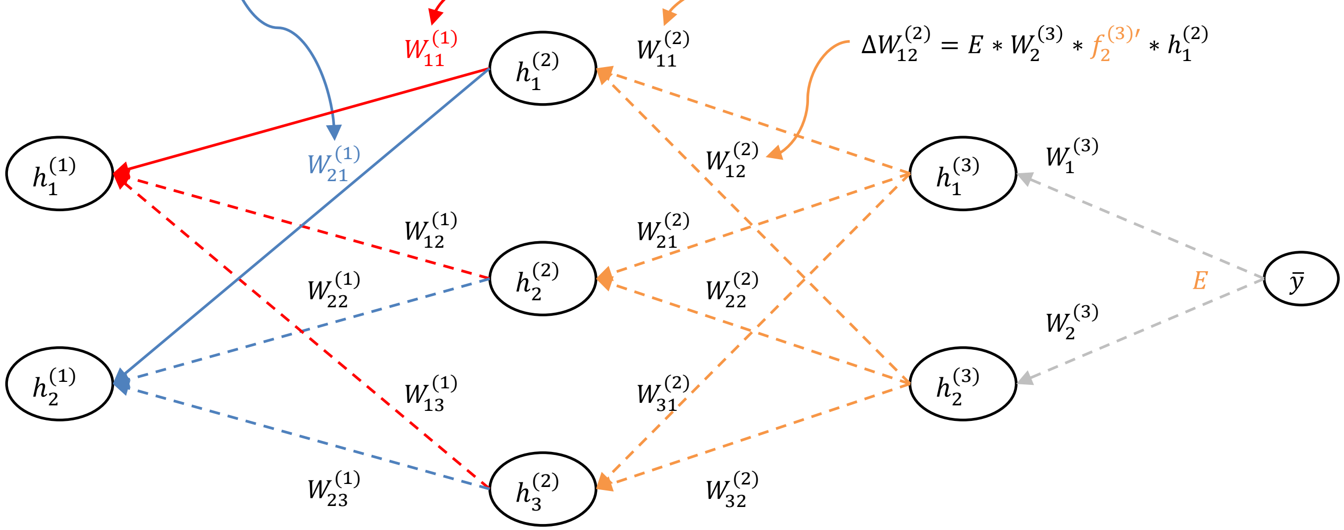
$$E = \alpha(y-\bar{y})$$

$$\Delta W_{11}^{(1)} = (E * W_{11}^{(2)} * f_1^{(3)'} * f_1^{(2)'} * h_1^{(1)}) + (E * W_{12}^{(2)} * f_2^{(3)'} * f_1^{(2)'} * h_1^{(1)})$$

$$\Delta W_{21}^{(1)} = (E * W_{11}^{(2)} * f_1^{(3)'} * f_1^{(2)'} * h_2^{(1)}) + (E * W_{12}^{(2)} * f_2^{(3)'} * f_1^{(2)'} * h_2^{(1)})$$

$$\Delta W_{11}^{(2)} = E * W_1^{(3)} * f_1^{(3)'} * h_1^{(2)}$$

$$\Delta W_{12}^{(2)} = E * W_2^{(3)} * f_2^{(3)'} * h_1^{(2)}$$



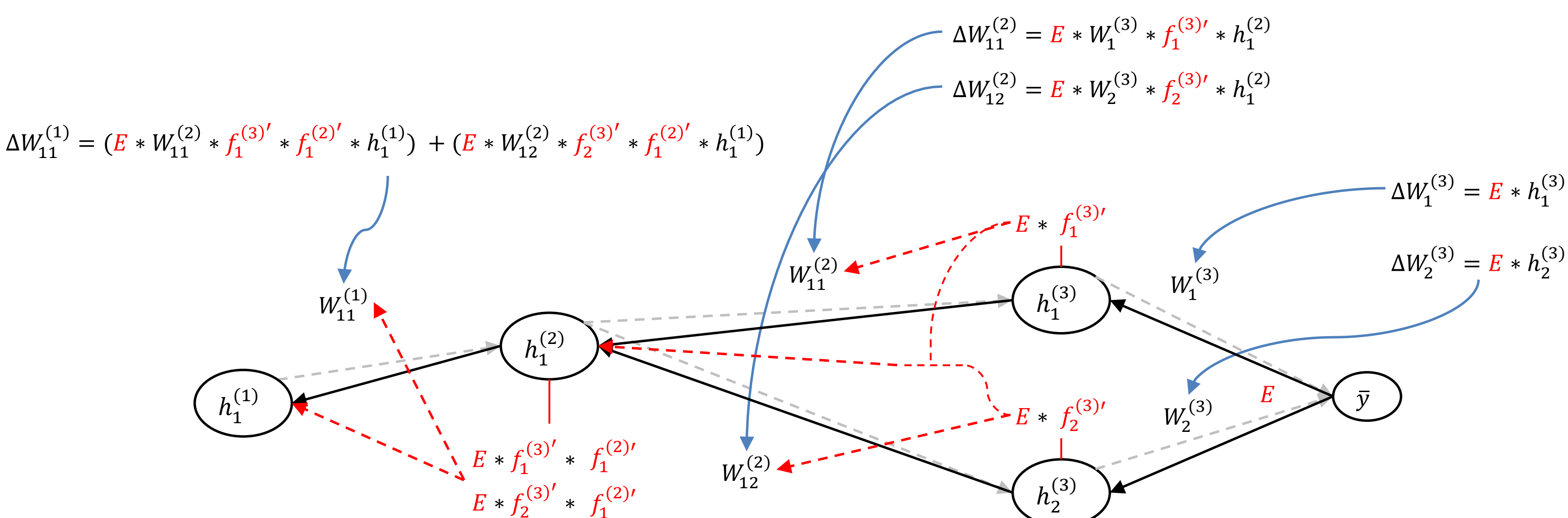
3-1. 역전파 (Backpropagation) (10/10)

- 역전파(Backpropagation)를 이용한 학습 과정 예제
  - ③ W_{ij}^{k}에 대한 (출력 $\bar{y}$의) gradient 계산
    - hidden layer ← hidden layer

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k}$$

$$E = \alpha(y-\bar{y})$$

- 수식을 보면 현재 W의 변화량을 계산할 때, 이전 노드들의 미분 값이 곱해지는 것을 볼 수 있다.
  (때문에, 역전파 과정을 국소적 미분의 곱을 이용한 연산으로도 부른다.)



3-2. 기울기 소실 문제 (Vanishing Gradient Problem) (1/2)

- 기울기 소실 문제(Vanishing Gradient Problem)이란?
  - 기울기 소실 문제는 인공 신경망을 경사 하강법으로 학습할 때, 특정 활성화 함수를 사용하면 발생한다.
    - 기울기 소실 문제를 발생시키는 대표적인 활성화 함수로 Sigmoid가 있다.
  - 신경망은 "$Weight$가 바뀌면, $Error(output)$은 얼마나 바뀌는가?"의 원리로 학습된다고 했다.
    - 이 과정에서 미분 즉, 기울기가 사용되는데 특정 활성화 함수를 사용하게 되면 기울기가 0으로 수렴하면서, $Weight$가 업데이트되지 않는다.

$$W_{ij}^{k} \coloneqq W_{ij}^{k} + \Delta W_{ij}^{k} \longrightarrow W_{ij}^{k} \coloneqq W_{ij}^{k} + \left(-\alpha \frac{\partial E}{\partial W_{ij}^{k}}\right)$$

$$\Delta W_{ij}^{k} = \alpha(y-\bar{y})\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = \alpha(y-\bar{y})\,\delta_{ij}^{k} = E * \delta_{ij}^{k}$$

$$\Delta W_{11}^{(1)} = (E * W_{11}^{(2)} * f_1^{(3)'} * f_1^{(2)'} * h_1^{(1)}) + (E * W_{12}^{(2)} * f_2^{(3)'} * f_1^{(2)'} * h_1^{(1)}) \longleftarrow \Delta W_{11}^{(2)} = E * W_1^{(3)} * f_1^{(3)'} * h_1^{(2)}$$

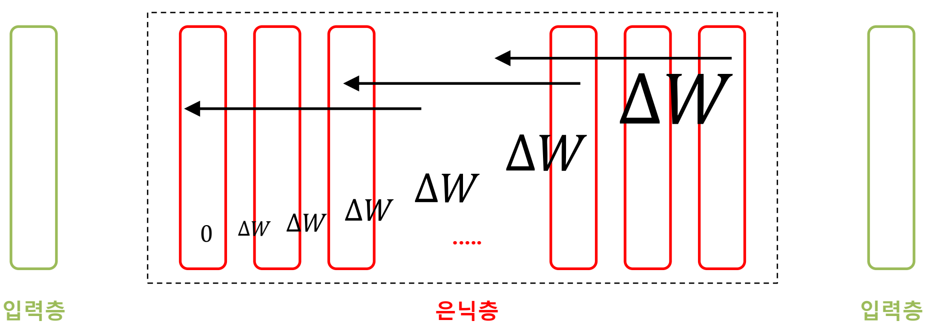
$$\begin{aligned}\Delta W_{11}^{(k)} = & \left(E * W_{11}^{(k+1)} * f_m^{(n)'} * f_m^{(n-1)'} * f_{m-1}^{(n-1)'} * \ldots * f_1^{(k+1)'} * h_1^{(k)}\right) \\ & + \left(E * W_{12}^{(k+1)} * f_{m+1}^{(n)'} * f_{m+1}^{(n-1)'} * f_m^{(n-1)'} * \ldots * f_1^{(k+1)'} * h_1^{(k)}\right) \\ & + \ldots\end{aligned}$$

3-2. 기울기 소실 문제 (Vanishing Gradient Problem) (2/2)

- 기울기 소실 문제(Vanishing Gradient Problem)이란?
  - 현재 W의 변화량을 계산할 때, 이전 노드들의 미분 값을 곱하는데 이 값은 활성화 함수를 미분하여 얻어진다.
  - 역전파가 진행될 수록 활성화 함수를 미분하여 얻어진 값이 계속해서 곱해지는데 이 값의 절대값이 1보다 작은 경우, 결국에는 W의 변화량이 0에 수렴하는 현상이 발생하고, 이를 기울기가 소실되었다고 부른다.
    - Sigmoid 함수는 미분했을 때 범위가 0 ~ 0.25이다. 이 값을 계속해서 곱한다면, 결국에는 W의 변화량이 0에 수렴하게 되어, 목표 성능에 도달하지 못한 채로 학습이 종료된다.

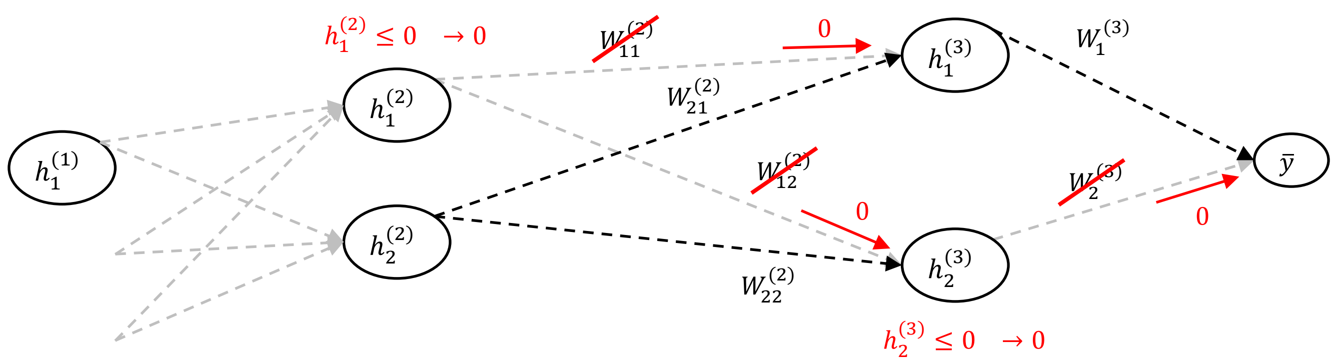
$$\boxed{\Delta W_{ij}^{k} = E\,\frac{\partial \bar{y}}{\partial W_{ij}^{k}} = E * \delta_{ij}^{k}}$$

$$\begin{aligned}\Delta W_{11}^{(k)} = & \left(E * W_{11}^{(k+1)} * \boxed{f_m^{(n)'} * f_m^{(n-1)'} * f_{m-1}^{(n-1)'} * \ldots * f_1^{(k+1)'}} * h_1^{(k)}\right) \\ & + \left(E * W_{12}^{(k+1)} * f_{m+1}^{(n)'} * f_{m+1}^{(n-1)'} * f_m^{(n-1)'} * \ldots * f_1^{(k+1)'} * h_1^{(k)}\right) \\ & + \ldots\end{aligned}$$



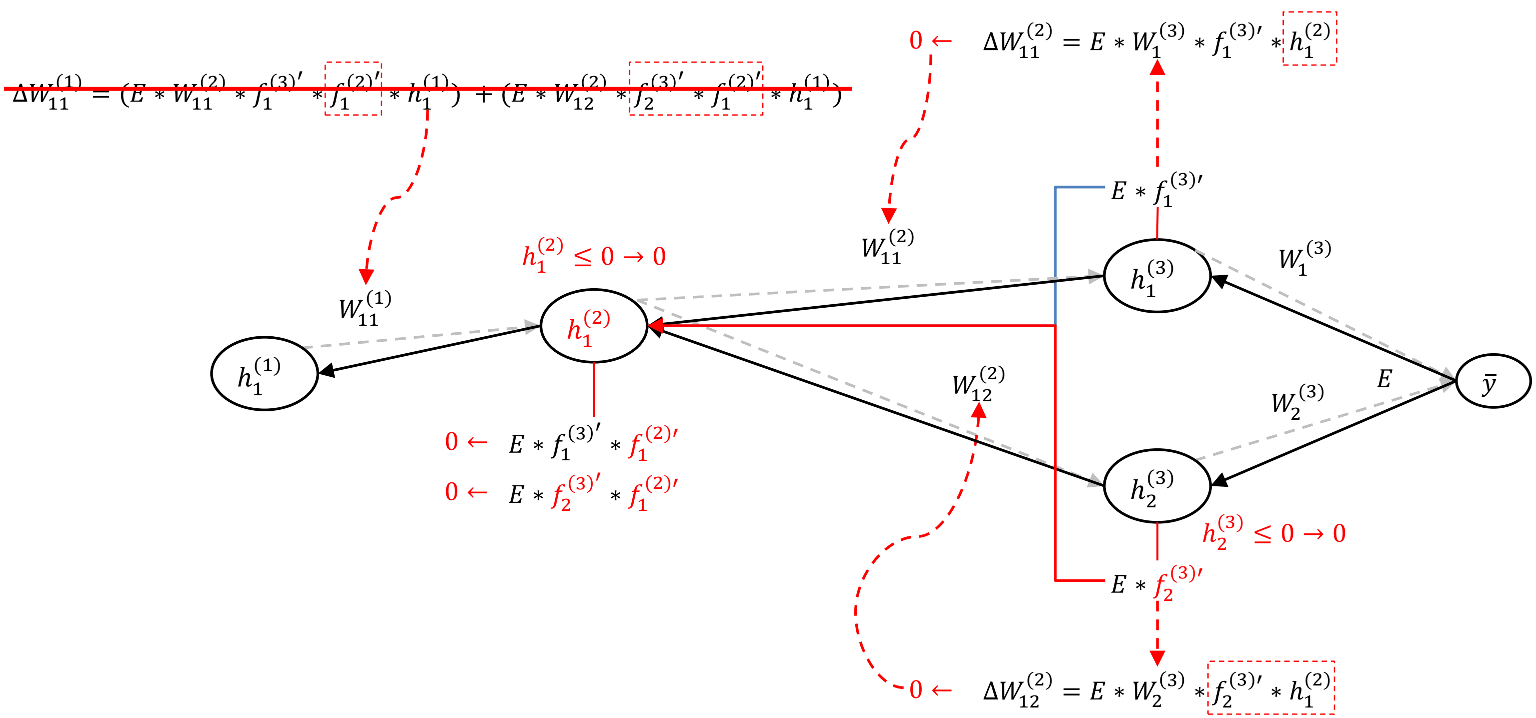
3-3. Dying ReLU (1/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - 출력이 정규화되는 함수 Sigmoid(0 ~ 1) 또는 TanH(-1 ~ 1) 등은 기울기 소실 문제가 발생할 수 있다.
    - 이를 해결하기 위해 사용되는 활성화 함수가 ReLU, 출력이 0 보다 작거나 같으면 0, 크면 입력 그대로이기 때문에 미분 값은 0 또는 1이다.
    - ReLU를 사용하면 기울기 소실 문제는 해결할 수 있지만, 순전파 과정에서 특정 노드에 음수 값이 들어온다면 출력이 0이 되고, 역전파에서도 미분한 값이 0이 되기 때문에 W의 업데이트 과정에서 문제가 발생한다.
      - 위 문제가 dying ReLU 현상이지만, 성능에 큰 영향을 미치지는 않으며, 특정 노드의 출력이 0이 됨으로써 dropout 효과를 얻을 수 있다.



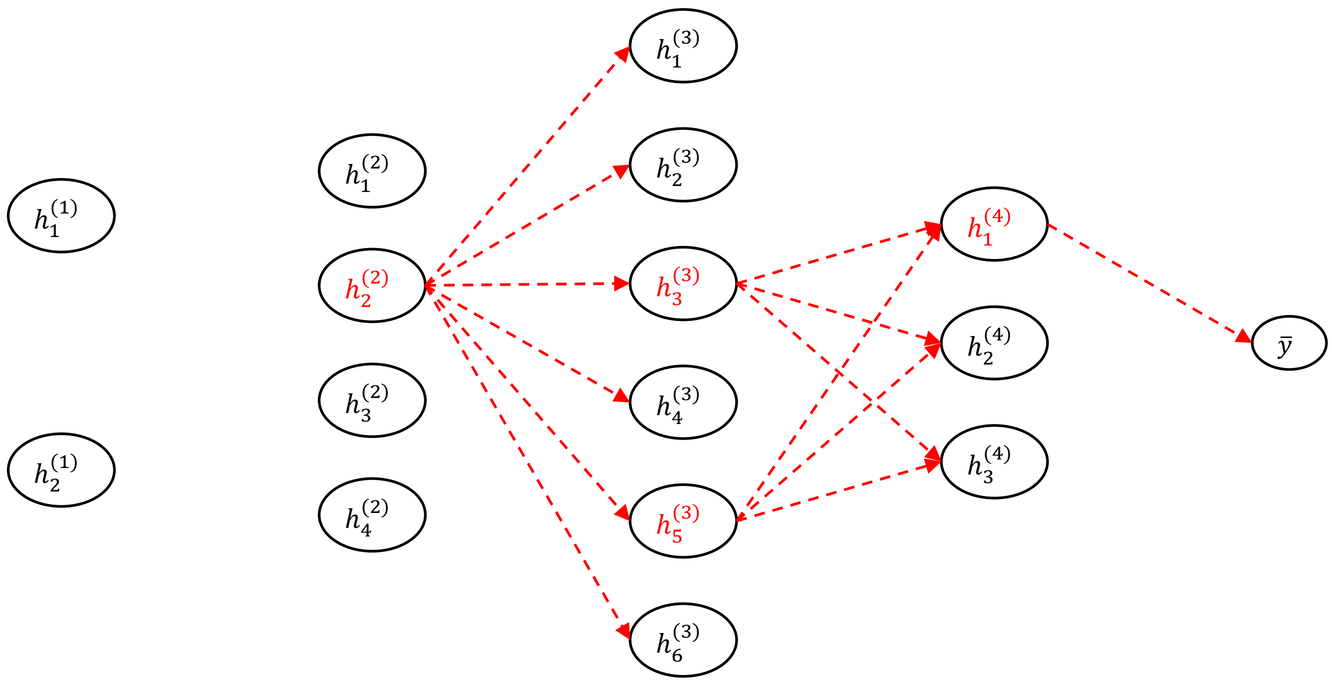
3-3. Dying ReLU (2/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - ReLU를 사용하면 기울기 소실 문제는 해결할 수 있지만, 순전파 과정에서 특정 노드에 음수 값이 들어온다면 출력이 0이 되고, 역전파에서도 미분한 값이 0이 되기 때문에 W의 업데이트 과정에서 문제가 발생한다.



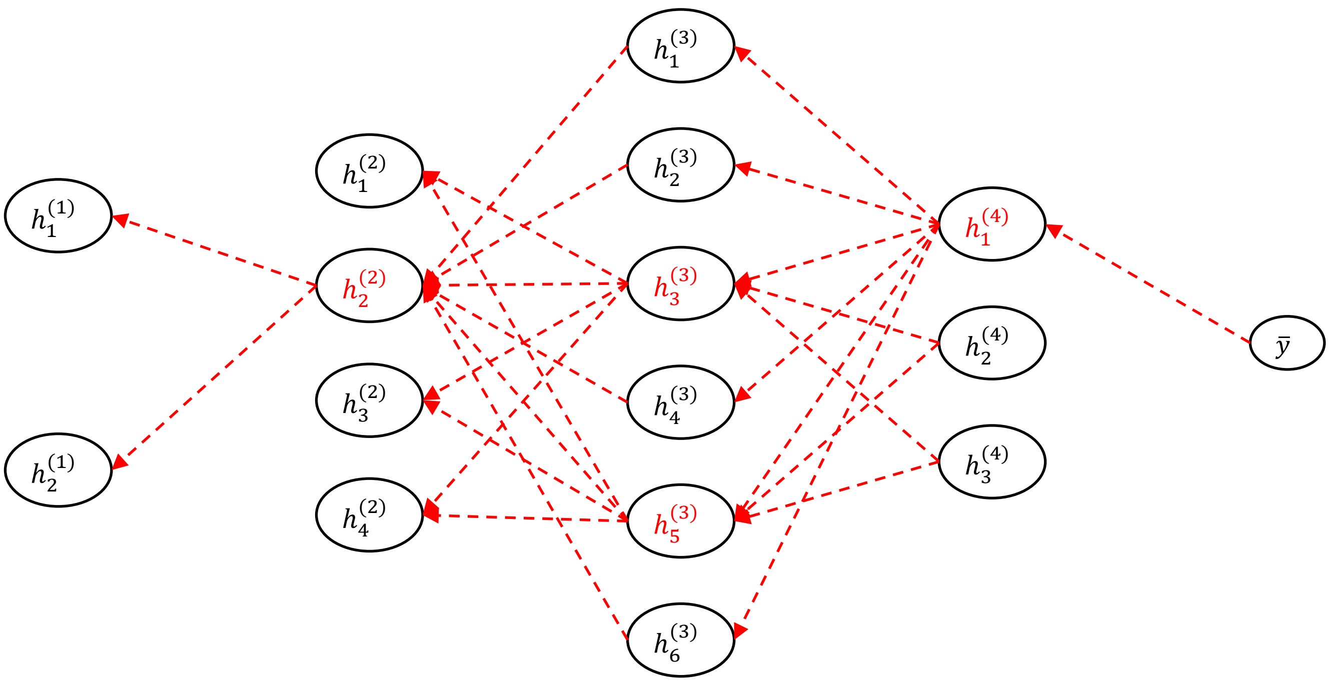
3-3. Dying ReLU (3/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - (붉은 노드는 죽어버린 상태) 붉은 선은 죽은 노드로 인하여, 0을 전달하거나 가중치가 업데이트되지 않는 경우를 뜻한다.



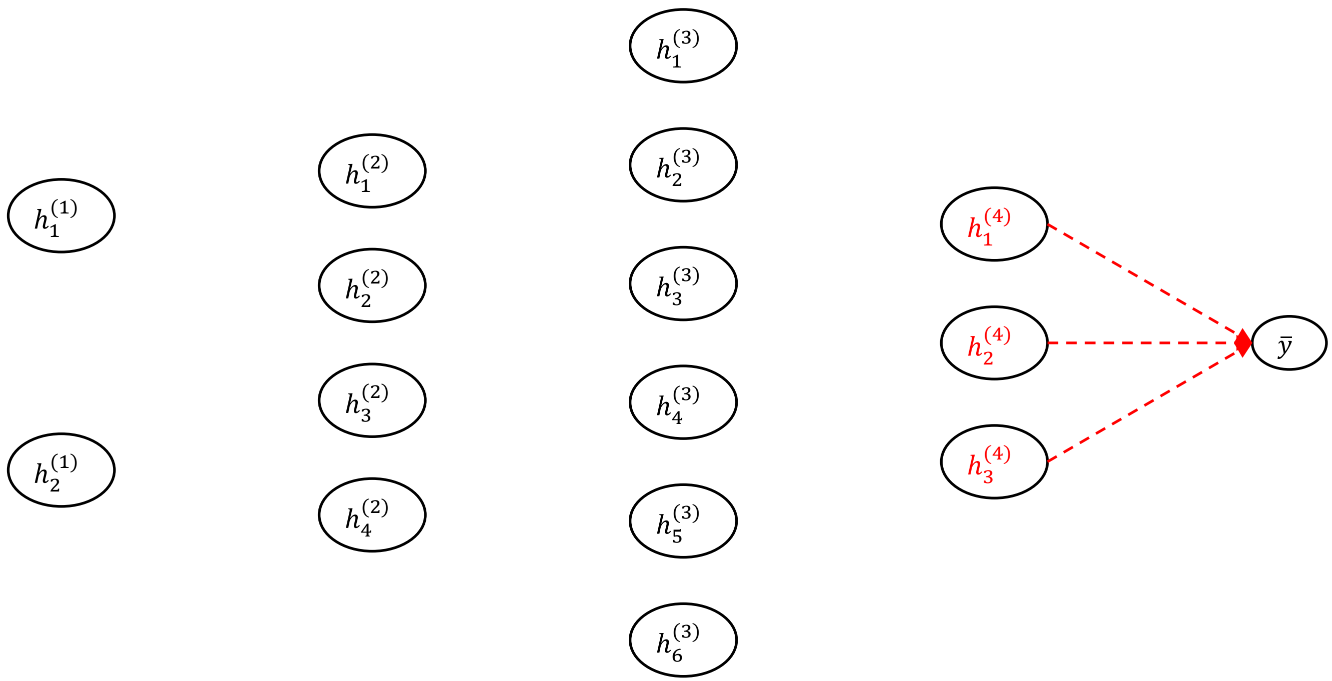
3-3. Dying ReLU (4/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - (붉은 노드는 죽어버린 상태) 붉은 선은 죽은 노드로 인하여, 0을 전달하거나 가중치가 업데이트되지 않는 경우를 뜻한다.



3-3. Dying ReLU (5/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - (붉은 노드는 죽어버린 상태) 붉은 선은 죽은 노드로 인하여, 0을 전달하거나 가중치가 업데이트되지 않는 경우를 뜻한다.



3-3. Dying ReLU (6/6)

- Dying ReLU 현상이란?
  - 활성화 함수로 ReLU를 사용했을 때, 노드(뉴런) 자체가 죽어버리는 현상
    - (붉은 노드는 죽어버린 상태) 붉은 선은 죽은 노드로 인하여, 0을 전달하거나 가중치가 업데이트되지 않는 경우를 뜻한다.