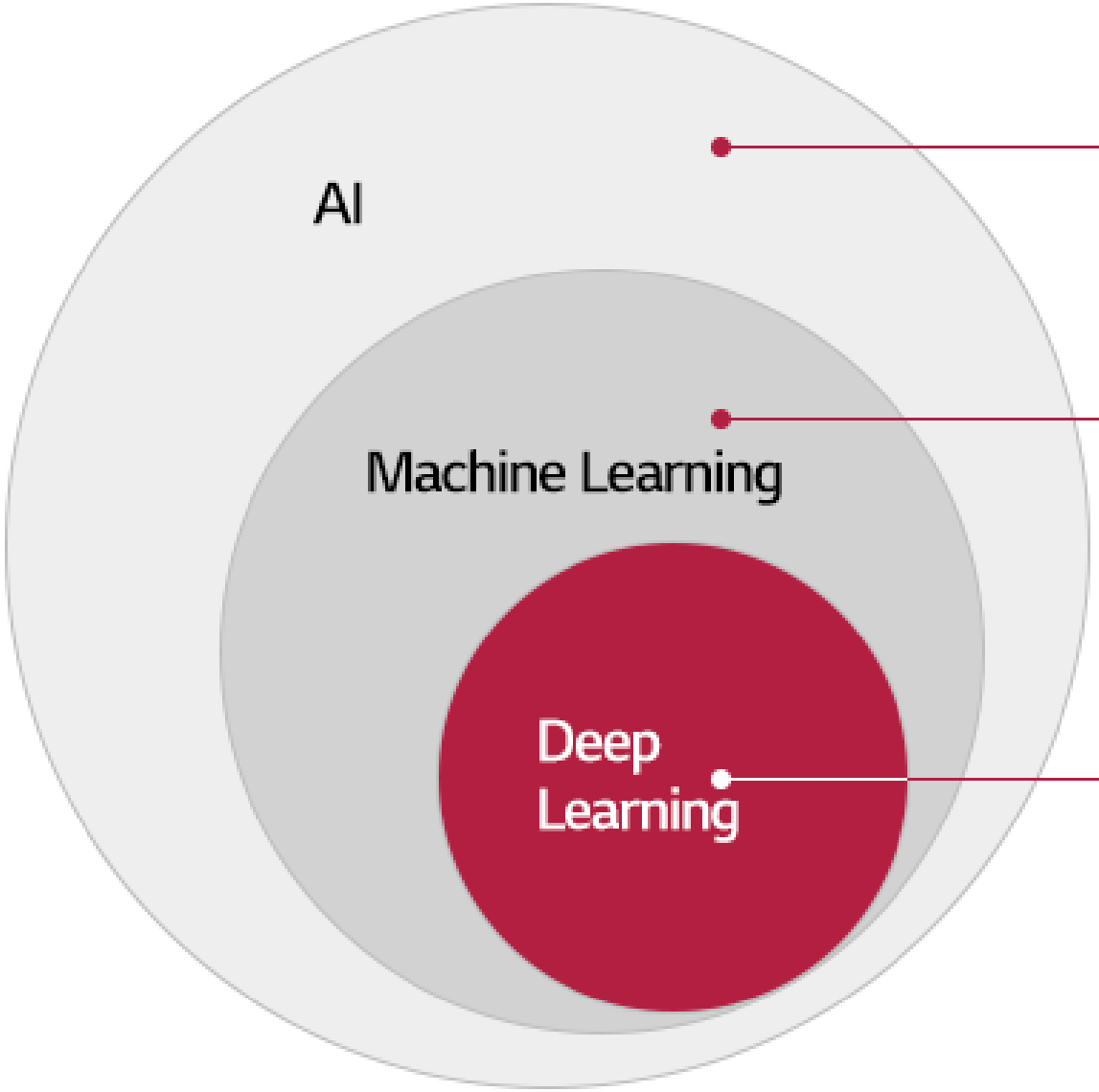
LG계약학과 제어시스템전공 2025년 동계 집중교육 특강

AI 기반 SW 설계를 위한 이론 및 실무 II

안상태 교수

2026년 2월 10일 (화)

3. 인공신경망



인공지능(Artificial Intelligence)

사람이 해야 할 일을 기계가
대신할 수 있는 모든 자동화에 해당

머신러닝(Machine Learning)

명시적으로 규칙을 프로그래밍하지 않고
데이터로부터 의사결정을 위한 패턴을
기계가 스스로 학습

딥러닝(Deep Learning)

인공신경망 기반의 모델로, 비정형
데이터로부터 특징 추출 및 판단까지
기계가 한 번에 수행

3.1 뇌의 동작을 흉내 내자 - 연결주의자의 목표

- 뇌는 뉴런neuron이라는 수없이 많은 신경세포들의 연결을 갖고 있음

- 인공 신경망은 이러한 신경세포의 동작을 흉내 내는 장치나 소프트웨어를 만들어 뇌가 수행하는 인지나 사고 능력을 갖춘 기계를 만들려는 노력

  - 이러한 방식의 연구를 통해 인공지능을 구현하려는 방식을 연결주의connectionism

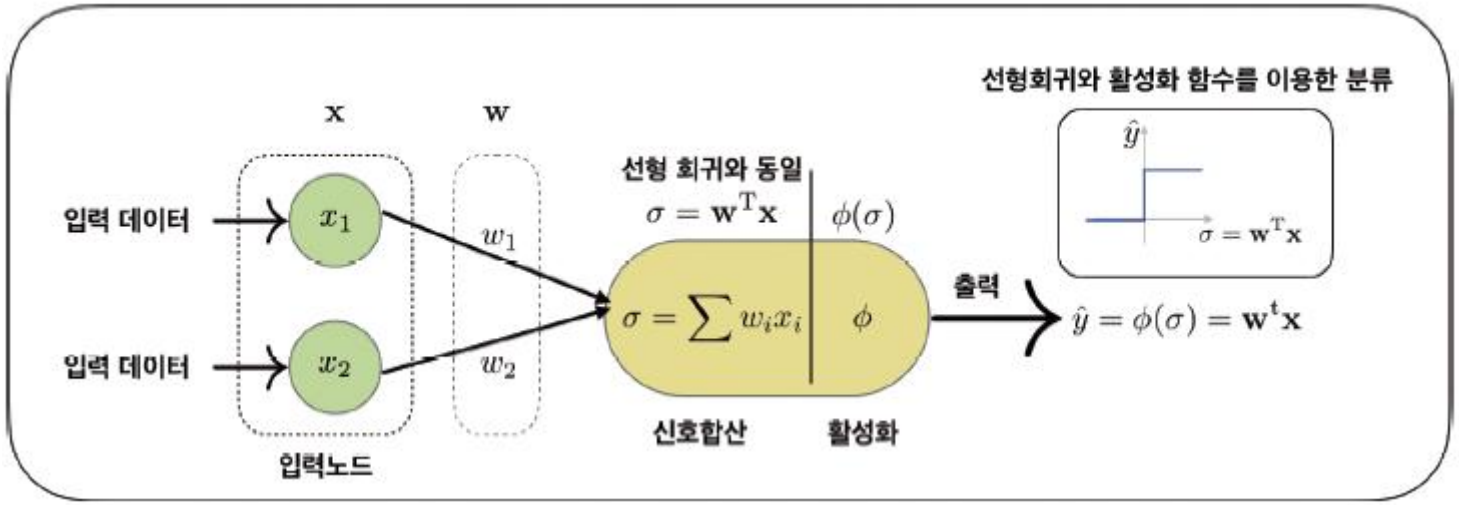
3.2 학습할 수 있는 신경 모델 - 퍼셉트론

- 학습이 가능하게 만든 것이 퍼셉트론perceptron

- 1958년에 프랭크 로젠블랫Frank Rosenblatt이 만든 퍼셉트론은 미국 해군에 의해 최초로 공개되어, 당시 인공지능의 새로운 장을 연 기술로 각광

3.2 학습할 수 있는 신경 모델 - 퍼셉트론

- 가장 중요한 변화는 그림 왼쪽의 입력을 받아들이는 신경세포 x_i들이 출력을 수행하는 노드로 연결될 때 연결강도weight w_i에 의해 조정되어 전달된다는 것
- 출력 노드는 두 가지 일을 하는데, 우선 전달되어 오는 신호를 모두 합한다.
- 합산된 신호에 따라 출력을 결정하는 활성화activation 함수가 최종 출력을 발생

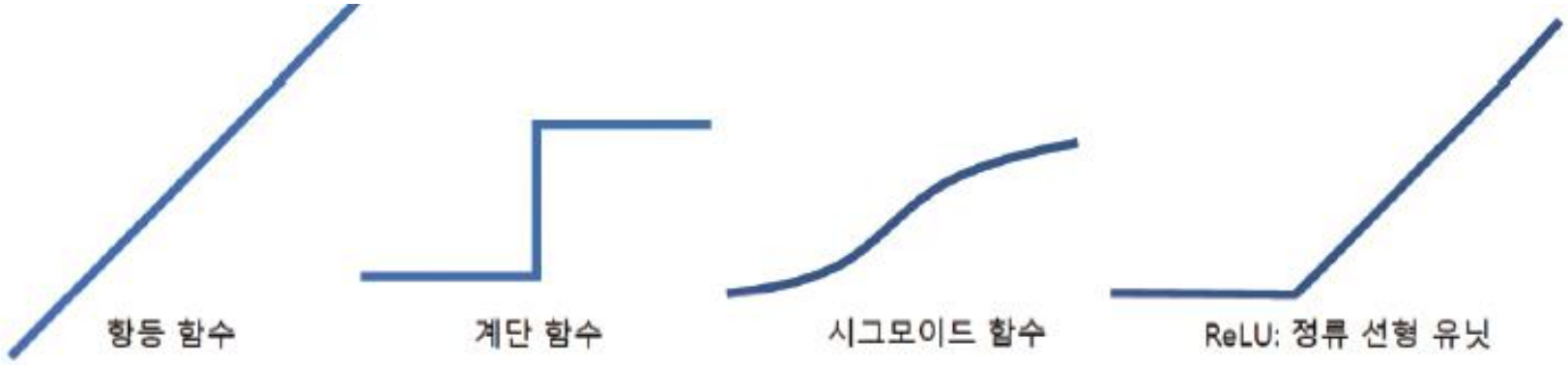


3.2 학습할 수 있는 신경 모델 - 퍼셉트론

- **학습은 경험을 통해 기능이나, 성능이 바뀌는 것**
  - 이를 위해서는 학습 과정을 통해 바뀌는 부분이 모델 내에 존재해야 함
  - 퍼셉트론은 학습을 통해 연결강도를 바꾸어 나감
  - 출력의 결과와 정답을 비교하면 오차를 알 수 있으며 이를 줄이도록 연결강도를 바꾸는 과정이 바로 학습

3.2 학습할 수 있는 신경 모델 - 퍼셉트론

- 활성화 함수는 출력 노드에 모아진 신호를 다음으로 내어 보낼때 얼마나 강하게 보낼지를 결정하는 함수
  - 계단 함수는 최초의 퍼셉트론이 채택한 활성화 함수 미분이 되는 구간에서 미분치가 언제나 0이기 때문에 경사 하강법을 통한 최적화가 불가능
  - 이러한 이유로 신경망 분야에서 새롭게 도입되어 오랫동안 사용되던 활성화 함수가 시그모이드sigmoid 함수



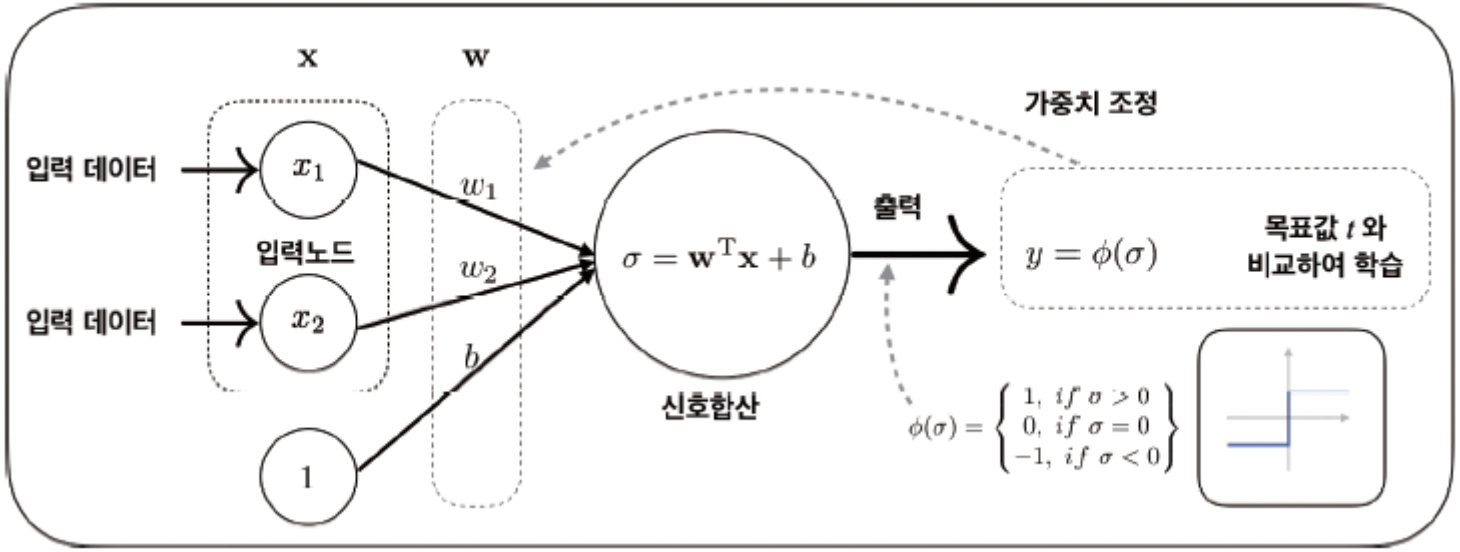
3.2 학습할 수 있는 신경 모델 - 퍼셉트론

- 신경망이 동작을 변경하는 방법은 연결강도를 조정하는 것

- 인공 신경망의 학습이라는 것은 출력을 목표치와 비교하여 오차를 계산하고 이 오차를 줄이는 방향으로 연결강도를 변경하는 일을 의미
  - 신경망 모델에서는 이 연결강도가 바로 모델의 동작을 결정하는 파라미터parameter
  - 오차를 계산하는 방법은 다양하며, 이 오차를 줄이는 방향으로 연결강도를 조정하는 일이 신경망의 최적화optimization, 즉 학습
  - 학습과정을 조절하는 하이퍼파라미터hyperparameter로는 오차에 따른 연결강도 조정 정도, 최적화 방법, 학습 반복 횟수 등이 있음

3.3 학습의 원리 - 연결강도의 변경

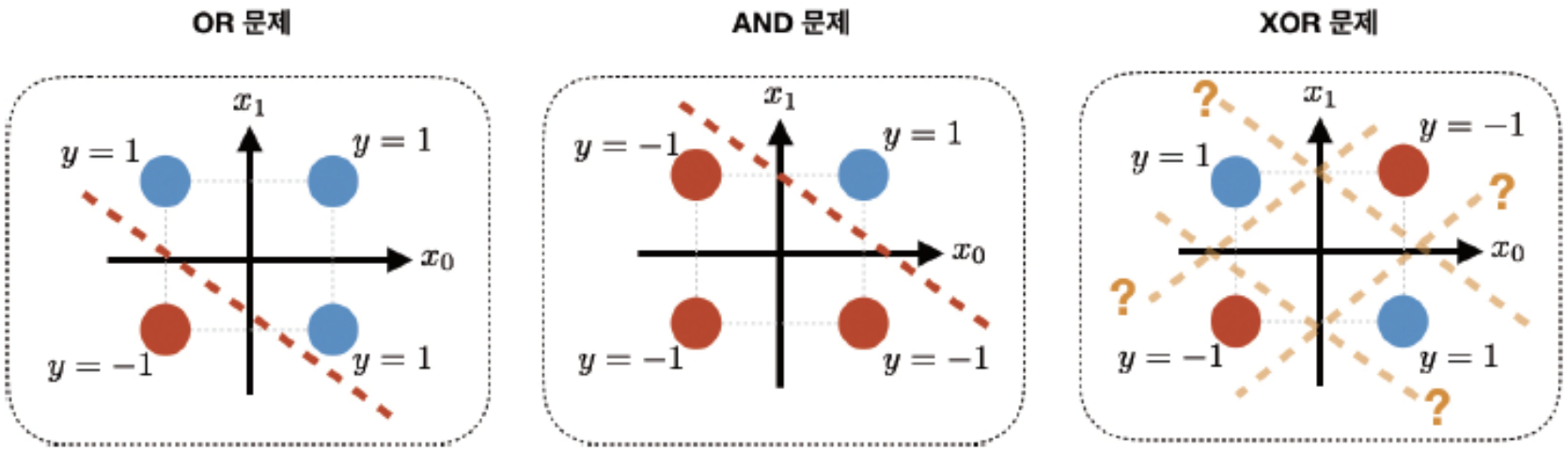
- 이항 불리언 연산binary boolean operation이 가능하도록 하는 퍼셉트론은 아래 그림처럼 만들 수 있음
- 참 또는 거짓을 갖는 입력은 x_1과 x_2에 제공
- 이 신호는 가중치가 곱해져 출력 노드로 전달
- 이때 신호의 편향bias이 이루어지도록 하는 가중치 b가 추가
- **편향은 신호값을 양의 방향이나 음의 방향으로 이동시키는 역할**



3.4 퍼셉트론, 연결주의가 누린 첫 영예와 긴 좌절

- 기호주의자들은 로젠블랫이 과학적 기준을 따르지 않고, 언론을 편파적으로 활용해 "과장된 주장overclaim"을 한다고 봄

- 마빈 민스키와 시모어 패퍼트가 쓴 "퍼셉트론"이라는 저서는 당시 엄청난 기대를 받던 신경망의 한계를 밝혀내기 위해 온 힘을 기울여 쓴 책
  - 가장 잘 알려진문제는 퍼셉트론이 XOR 연산을 구현하지 못 한다는 것

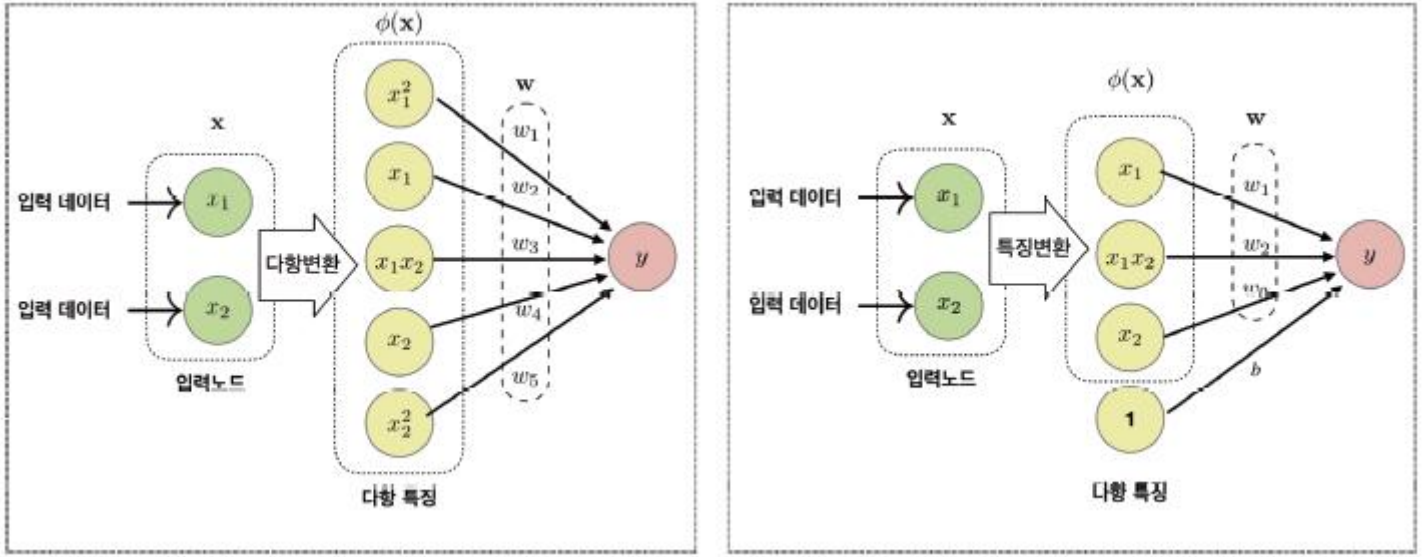


3.5 퍼셉트론이 XOR 문제를 풀 수 있게 만드는 방법

- 입력 노드가 출력으로 바로 연결되어 있는 단순한 퍼셉트론은 선형 분리만 가능하지만, 이미 맥컬록과 피츠의 신경 모델에서도 신호는 여러 층을 거쳐 전달되는 모델이 사용됨
- 퍼셉트론의 구조를 복잡하게 만들어 다양한 일을 하게 할 수 있지 않을까? 선형 함수로서 동작하는 모델을 비선형 함수로 바꾸는 방법 중에 우리가 이미 살펴본 것은 특징 벡터를 다항화하는 방법
- 단순한 퍼셉트론 모델에서는 x_1 와 x_2 의 입력을 가지므로, 이를 2차 다항화하면 다음과 같은 특징 벡터를 얻을 수 있을 것

$$\left(x_1^2, x_1, x_1 x_2, x_2^2\right)$$

- 두 개의 입력을 이렇게 다항화하고 이를 퍼셉트론으로 인지하는 모델은 아래의 왼쪽과 같은 퍼셉트론으로 설명

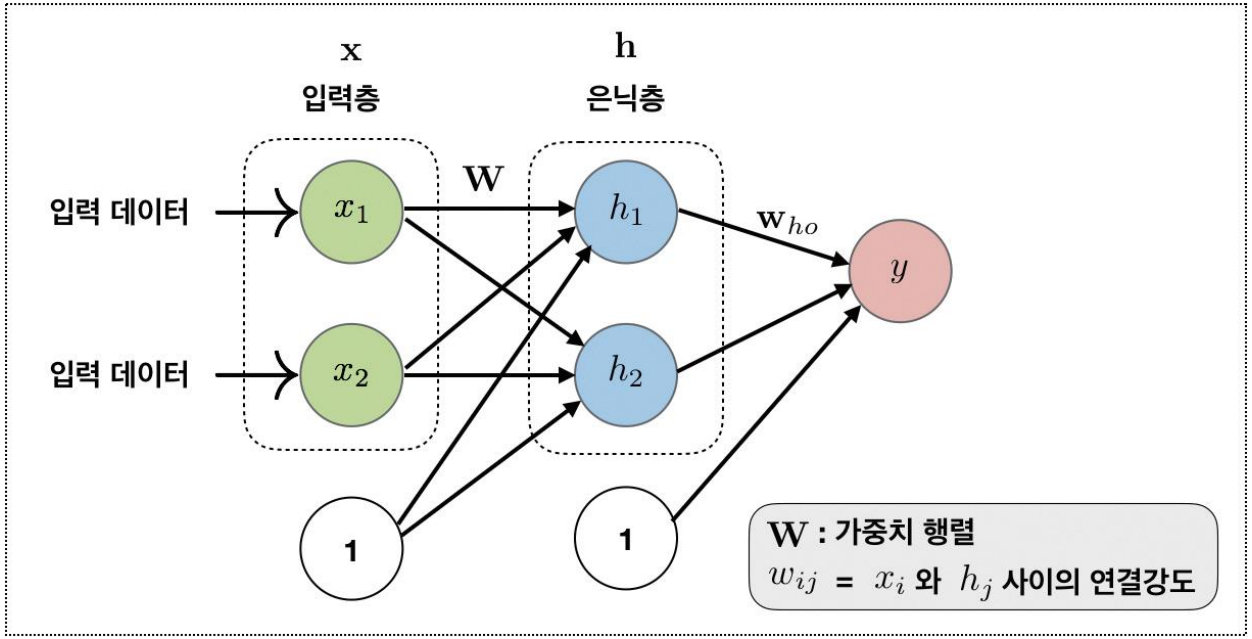


- 원래의 입력 x_1, x_2를 다항 변환하여 새로운 입력 $\phi(\mathrm{x})$로 만드는 과정은 신호를 전달하는 것이 아니라 직접적인 계산이 이루어짐
- 입력과 출력을 바로 연결하지 않고 중간에 신호를 중계하는 신경세포들을 두는 것
  - 입력과 출력 사이에 존재하는 이런 중간 노드들의 계층을 은닉층hidden layer

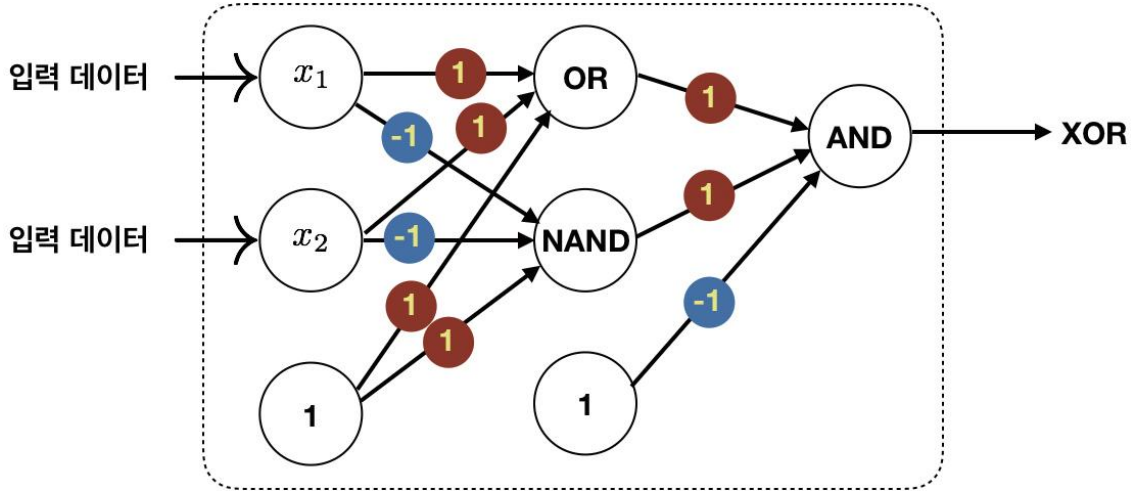
- 중간에 층을 두고 연결하면 다음과 같은 구조의 다층 퍼셉트론multi-layer perceptron을 만들 수 있을 것

- 가중치는 입력층에 있는 n개의 노드가 발생시키는 신호를 은닉층에 있는 m개의 노드로 연결하는 가중치 행렬 W에 의해 전달

- 은닉층의 노드들이 발생시키는 신호는 또 다른 연결강도 벡터 w_{ho}에 의해 출력으로 연결



- 이런 구조의 퍼셉트론은 **더 많은 수의 연결이 더 많은 단계로 연결**되어 복잡한 동작을 할 수 있음
- 적절한 연결강도를 부여하면 단순한 모델이 할 수 없었던 XOR 문제 풀이도 가능



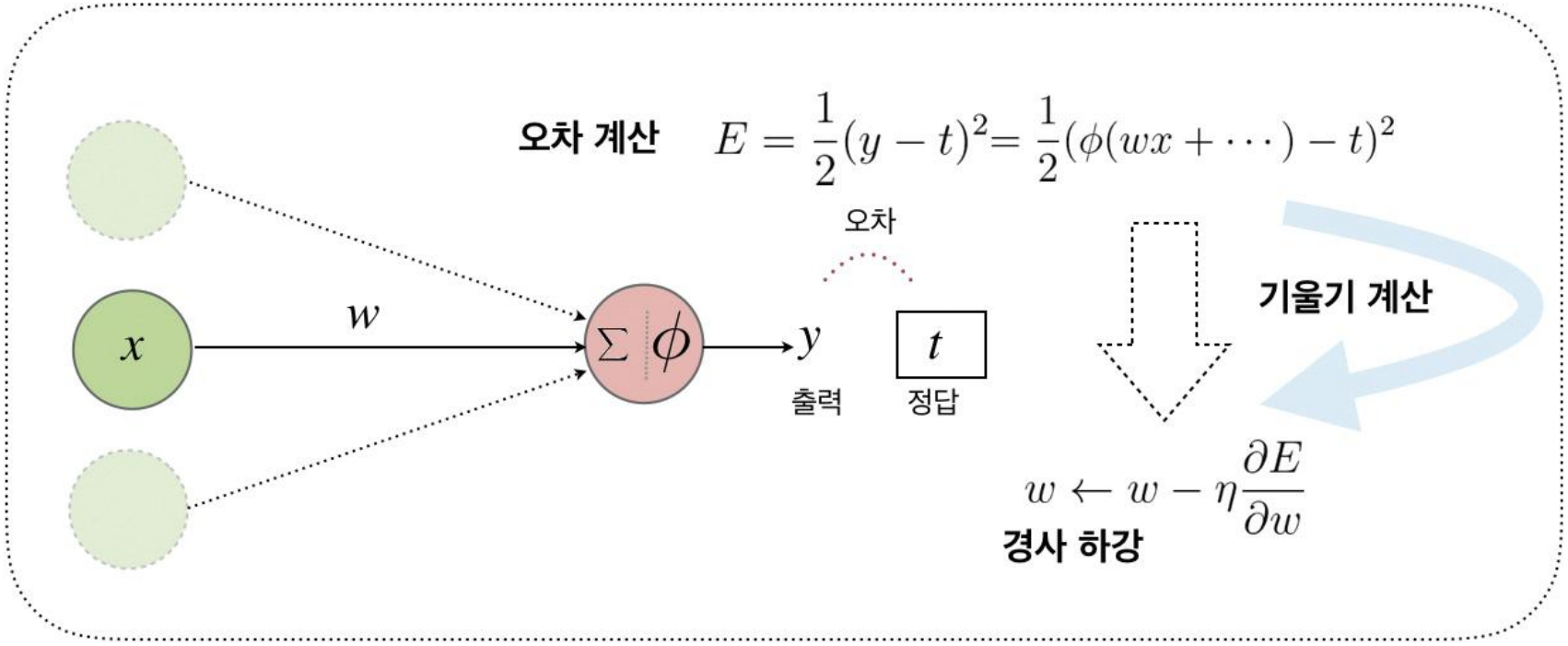
- 헵의 학습 규칙은 최종 은닉층과 출력의 연결에만 적용
- 층을 쌓아 만든 퍼셉트론이 분명 다양한 기능을 수행할 수 있지만, 오류를 줄이고 원하는 기능을 학습할 수 있도록 하는 방법이 없다면 이 모델을 훈련시킬 수 없음

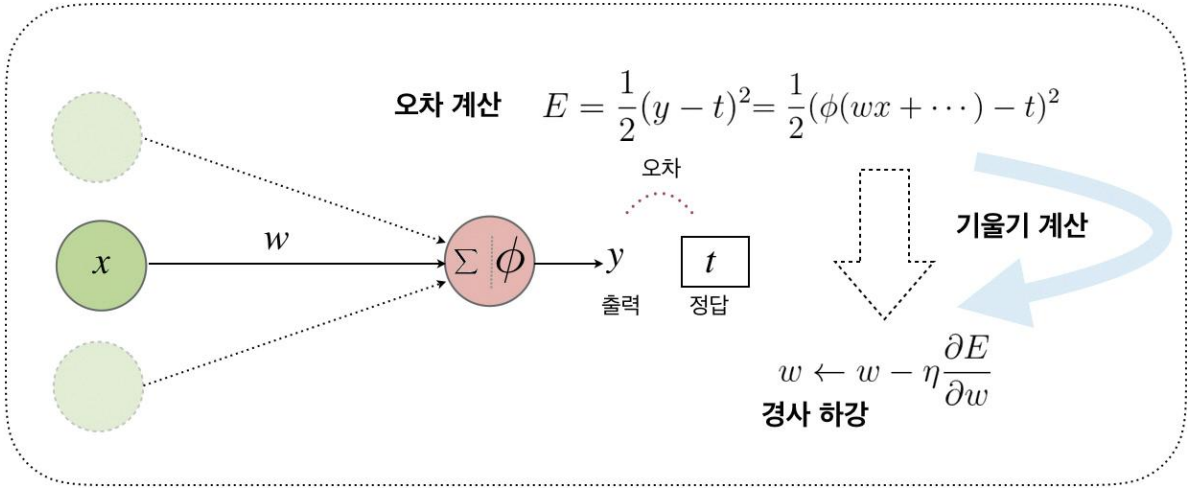
3.6 다층 퍼셉트론의 학습

- 인공 신경망에서 연결강도를 조정해 오차를 줄이는 방법

- 오차 곡면의 기울기를 연결강도에 대해 구한 뒤에 이 기울기를 따라 내려가는 경사 하강법gradient descent을 사용하여 구현할 수 있음

- x가 연결강도 w로 출력 노드에 연결되어 있다고 하자.
- 출력 노드는 x를 비롯한 여러 노드에서 신호를 수신하여 합산하는 부분과 이 값을 활성화함수에 넣어 최종 출력을 결정하는 부분으로 나뉨
- 출력과 정답(목표값)의 차이를 제곱하여 오차를 구하는 일반적인 방법을 사용 가능
- 출력 노드의 출력값 y가 $\phi(wx + \cdots)$이므로, 오차E는 $1/2\,(\phi(wx + \cdots) - t)^2$





- 오차를 연결강도 w에 대해 편미분

$$\frac{\partial E}{\partial w} = \frac{1}{2}\frac{\partial}{\partial w}(\phi(wx+\cdots)-t)^2$$

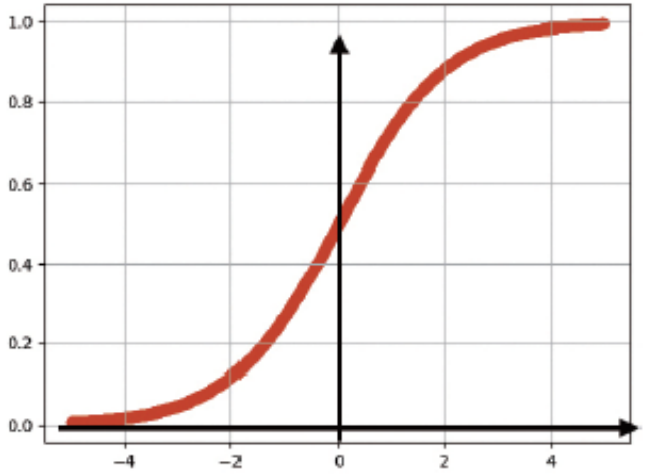
$$= (\phi(wx+\cdots)-t)\cdot \frac{\partial}{\partial w}\phi(wx+\cdots)$$

$$= (\phi(wx+\cdots)-t)\cdot \phi'(wx+\cdots)\cdot \frac{\partial}{\partial w}(wx+\cdots)$$

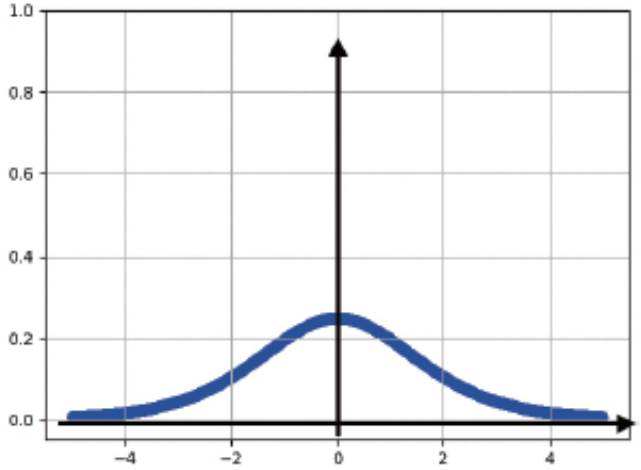
$$= (y-t)\cdot \phi'(wx+\cdots)\cdot x$$

- 기울기를 이용하여 연결강도를 수정
- 활성화 함수 $\phi()$의 미분 $\phi'()$ 만 안다면 입력 x, 출력 y, 목표값 t를 이용하여 간단히 계산할 수 있음
- 계단함수는 미분이 불가능한 지점이 있고, 미분이 되는 곳에서도 미분치가 언제나 0
  - 이 방법을 위해서는 미분이 가능한 새로운 활성화 함수가 필요
  - 신경망에서는 다음과 같은 시그모이드sigmoid 혹은 로지스틱logistic 함수를 활성화 함수로 도입

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



$$\phi'(x) = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right)$$



- 이 함수는 미분이 가능하다는 장점과 함께, 이 함수의 미분은 다음과 같이 원래의 함수를 이용하여 표현할 수 있는 장점이 있음

$$\phi'(x) = \phi(x)(1-\phi(x))$$

- 조금 전 구했던 오차의 기울기에서 활성화 함수의 미분으로 표시된 부분을 출력값 y로 표현할 수 있음

$$\begin{aligned}\frac{\partial E}{\partial w} &= (y-t)\cdot\phi'(wx+\cdots)\cdot x \\ &= (y-t)\cdot\phi(wx+\cdots)\cdot(1-\phi(wx+\cdots))\cdot x \\ &= (y-t)\cdot y\cdot(1-y)\cdot x\end{aligned}$$

- 출력값과 목표값만 알면 출력의 오차를 줄이는 연결강도 변경을 아래의 수식을 사용하여 수행할 수 있음

$$w \leftarrow w - \eta \cdot \partial E / \partial w$$

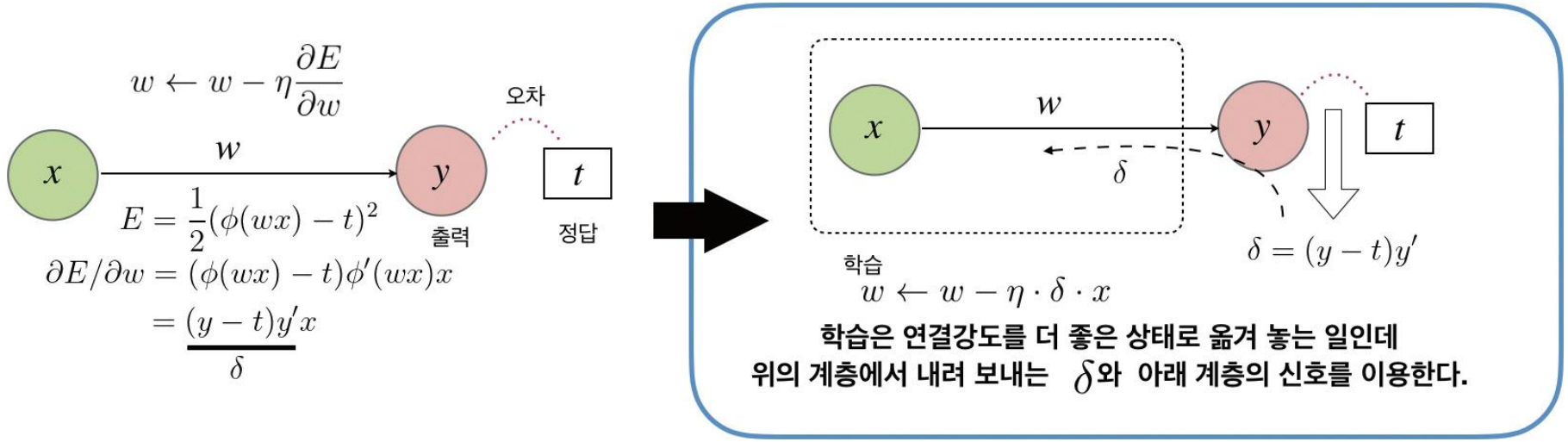
- 여기서 출력값 y와 목표값 t만 알면 연결강도 수정에 필요한 $\delta = (y - t)\phi'(wx + \cdots)$ 가 $\delta = (y - t) \cdot y \cdot (1 - y)$를 통해 쉽게 계산되며 δ를 이용하여 연결강도를 더 좋은 상태로 바꿀 수 있음

$$w \leftarrow w - \eta \cdot \delta \cdot x$$

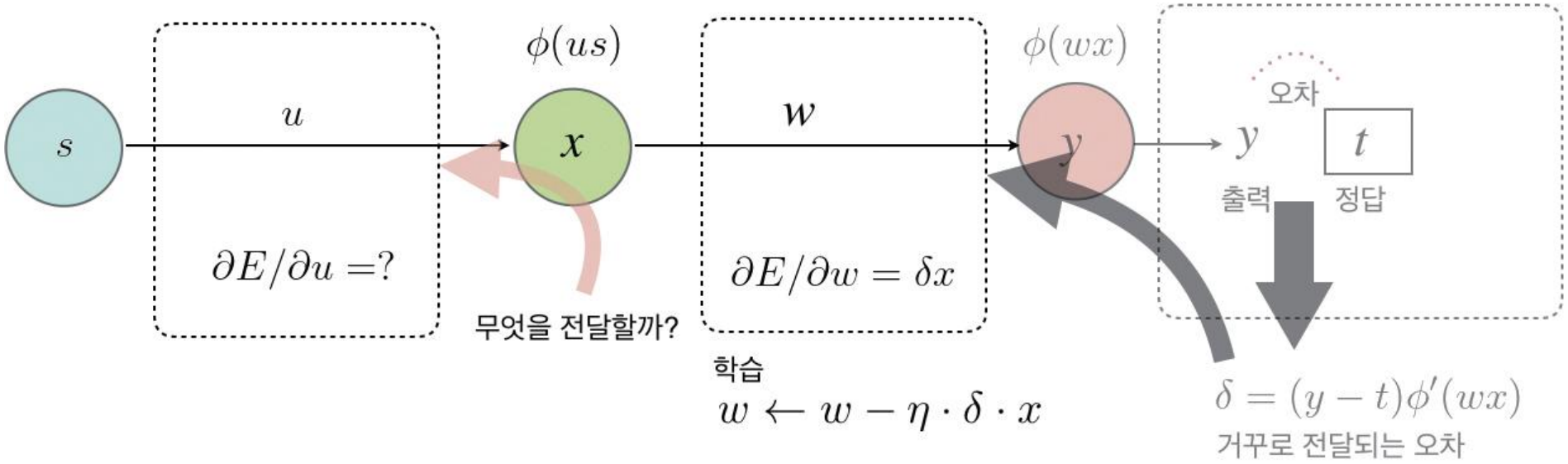
- 신경망의 각 연결강도에 적용하여 오차를 줄일 수 있다면, 복잡한 신경망도 학습이 가능

3.7 다층 퍼셉트론의 학습 – 오차 전파

- 아래 그림과 같은 신경망이 연결된 구조에서 신호가 x를 거쳐 y로 전달되는데 이 방향을 순전파forward propagation라고 함
- 앞 절에서 오차를 줄이는 방향으로 연결강도를 변경하는 방법을 아래와 같이 출력 부분에서 확인할 수 있는 목표 t와 출력 y의 차이, 그리고 출력의 미분 y'가 연결망을 거꾸로 타고 전달되어 가중치를 조정하는 것으로 봄



- s를 거쳐 x를 통과한 다음 출력노드 y로 연결되는 다층 신경망을 고려해보자.
- 연결강도 w를 갱신하는 방법을 그 이전 단계에 있는 연결강도 u 에도 전파할 수 있으며 이것이 오차 역전파error backpropagation의 기본 개념



- u를 갱신하는 학습을 위해 우리는 오차 곡면의 기울기를 u기준으로 계산

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial x} \cdot \frac{\partial x}{\partial u}$$

- $\partial E/\partial u$는 제곱 오차 $1/2\,(\phi(wx)-t)^2$를 w가 아니라 x로 미분한 것이므로 앞에서 구한 $\partial E/\partial w$식에서 w와 x의 역할을 바꿈

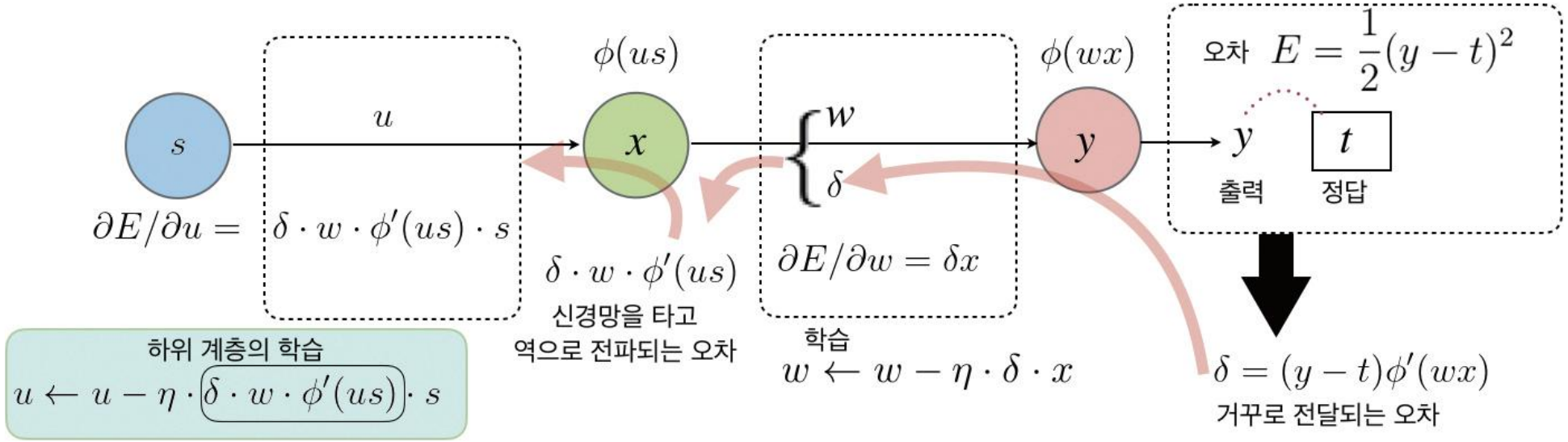
$$\partial E/\partial w = (y-t)y'x = \delta x$$
$$\partial E/\partial x = (y-t)y'w = \delta w$$

- 출력에 더 멀어진 단계의 가중치 u를 갱신하기 위한 오차의 미분을 계산하면 x값은 노드 s신호에 연결강도 u가 곱해진 뒤 활성화 함수를 통과한 결과인 $\phi(us)$이므로

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial x}\frac{\partial x}{\partial u} = \delta \cdot w \cdot \frac{\partial \phi(us)}{\partial u} = \delta \cdot w \cdot \phi'(us) \cdot s$$

- 순전파forward propagation 단계에서는 입력의 신호가 연결강도로 증폭되고 활성화 함수를 통과하여 출력 노드로 전달되는 과정

- 학습 과정은 연결강도를 수정하는 것이기 때문에, 출력단의 오차가 역방향으로 활성화 함수의 미분 함수를 거친 뒤에 연결강도에 의해 증폭되어 입력단까지 전달되는 과정으로, 이러한 신경망 학습 방법을 오차 역전파error backpropagation 알고리즘



3.8 TensorFlow

- 오차 역전파를 통한 신경망 학습의 방법을 살펴보고 직접 구현하면서 여러 층으로 이루어진 신경세포 층들 사이의 신호 전달이 동일한 방식으로 차원만 바꾸어 진행되는 행렬-벡터 연산임을 확인

- 신경망 구현은 텐서플로우TensorFlow라는 도구를 사용

TensorFlow

3.8 TensorFlow

- TensorFlow는 케라스keras라는 심층신경망 구현을 위한 API가 포함되어 있는데, 이를 이용하면 간단히 신경망을 만들고 학습 가능

> pip install tensorflow

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 텐서플로우와 케라스를 사용할 수 있도록 준비
import tensorflow as tf     # 텐서플로우는 주로 tf라는 별명을 사용한다
from tensorflow import keras
```

3.8 TensorFlow

- 모델은 신경 계층이 차례로 이어져 순차적으로 신호를 전달하는 모델
- 이런 모델은 Keras에서 제공하는 시퀀셜Sequential 클래스로 구현
- 모든 층들은 이전 단계의 층이 가지는 신경세포 노드와 빠짐없이 다 연결
  - 이러한 연결을 밀집dense 계층
  - 각 층이 가진 활성화 함수는 sigmoid

```
model = keras.models.Sequential( [
    keras.layers.Dense(6, activation= 'sigmoid'),
    keras.layers.Dense(4, activation= 'sigmoid'),
    keras.layers.Dense(1, activation= 'sigmoid'),
])
```

3.8 TensorFlow

- 전체 데이터를 다 사용하지 않고 임의로 선택된 데이터 인스턴스만을 가지고 경사 하강법을 적용하는 확률적 경사 하강법stochastic gradient descent:SGD을 사용
- 확률적 경사 하강법은 전체 데이터를 이용하여 학습하는 경사 하강법과는 다르게 임의적으로 추출한 일부 데이터를 사용해서 가중치를 조절 하기에 가중치 조절의 속도가 개선되는 효과
- 모델을 사용하기 위해서는 컴파일compile 과정
  - 컴파일 과정에서는 최적화 방법을 지정하고, 오차 측정을 어떤 기준으로 할 것인지 지정
  - 평균 제곱 오차를 손실함수로 사용한다면 'mse'를 손실 매개변수 loss의 인자로 지정

```
optimizer = keras.optimizers.SGD(learning_rate=5.0)
model.compile(optimizer=optimizer, loss='mse')
```

3.8 TensorFlow

- 모델을 훈련시킬 데이터

```
data_loc = 'https://github.com/dknife/ML/raw/main/data/'
df = pd.read_csv(data_loc+'nonlinear.csv')
X = df['x'].to_numpy()
y_label = df['y'].to_numpy()
```

- 모델의 훈련은 입력과 레이블을 fit() 메서드
- 에폭epoch을 지정하는데, Epoch은 하나의 데이터셋에 대해 몇 번 훈련을 반복할 것인지를 의미

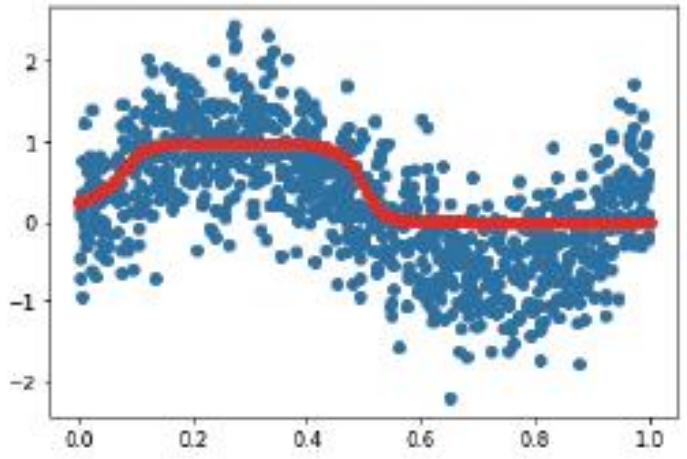
```
model.fit(X, y_label, epochs=100)
```

```
Epoch 1/100
32/32 [==============================] - 1s 1ms/step - loss: 0.6938
...
Epoch 100/100
32/32 [==============================] - 0s 1ms/step - loss: 0.3652
```

3.8 TensorFlow

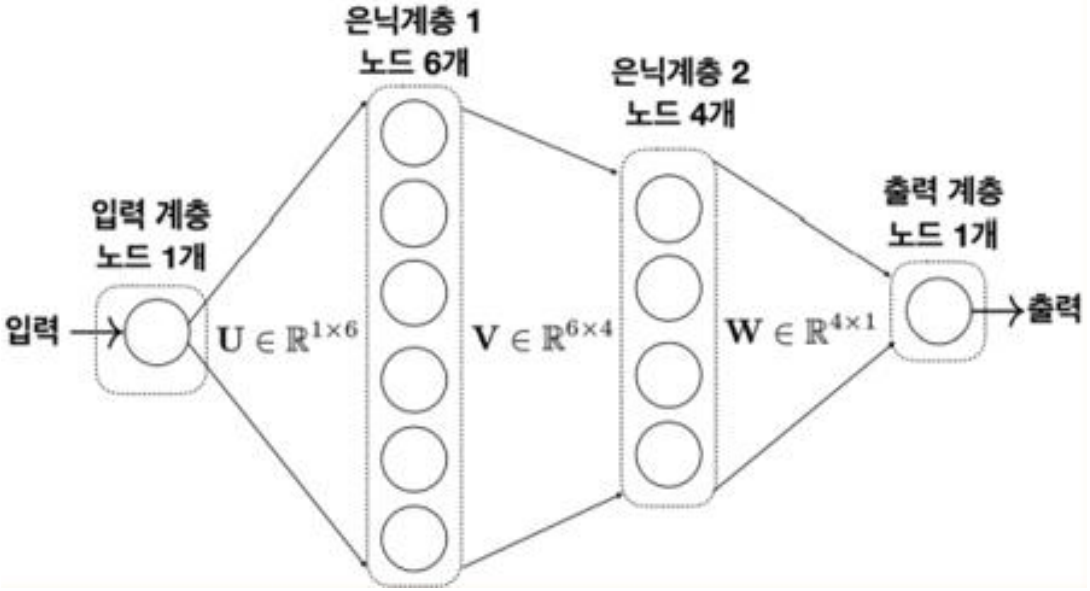
- 테스트용 입력을 모델에 넣어서 예측
- Keras 모델에서 예측을 수행하는 방법은 predict() 메서드에 입력 데이터를 넘김

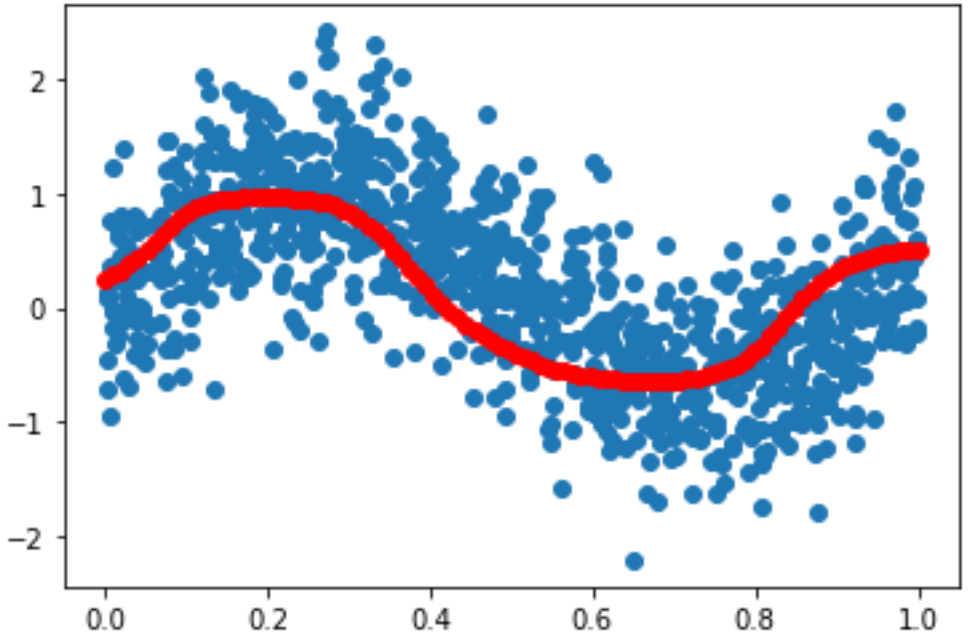
```
domain = np.linspace(0, 1, 100).reshape(-1,1) # 입력은 2차원 벡터로 변형
y_hat = model.predict(domain)
plt.scatter(df['x'], df['y'])
plt.scatter(domain, y_hat, color='r')
```



실습문제

- 아래 그림과 같은 다층 퍼셉트론을 주어진 데이터와 Keras를 이용해 구현하고 아래 그래프를 그려라.
  - “dataset2.csv” 데이터 사용
  - 곡선의 범위는 0과 1사이에 100개
  - learning rate = 0.1, epoch = 100
  - 활성화함수: tanh (hyperbolic tangent)





감사합니다