**3.1 퍼셉트론에서 신경망으로**

**3.1.1 신경망의 예**

도표, 원, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명

입력층: 가장 왼쪽

출력층: 가장 오른쪽

은닉층: 중간줄, 사람 눈에 보이지 않음

**3.1.2 퍼셉트론 복습**

원, 도표, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명폰트, 텍스트, 친필, 화이트이(가) 표시된 사진

자동 생성된 설명

b: 편향을 나타내는 매개변수, 뉴런이 얼마나 쉽게 활성화되느냐를 제어

w: 각 신호의 가중치를 나타내는 매개변수, 각 신호의 영향력을 제어

원, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명 가중치 b이고, 입력이 1인 뉴런이 추가

x1, x2, 1이라는 3개의 신호가 뉴런에 입력되어, 각 신호에 가중치를 곱한 후, 다음 뉴런에 전달

* 다음 뉴런에서 이 신호의 값을 더하여, 그 합이 0을 넘으면 1을 출력, 아닐 경우 0을 출력

폰트, 타이포그래피, 친필, 서예이(가) 표시된 사진

자동 생성된 설명

폰트, 화이트, 텍스트, 친필이(가) 표시된 사진

자동 생성된 설명

**3.1.3 활성화 함수의 등장**

활성화 함수: 입력 신호의 총합을 출력 신호로 변환하는 함수, 입력 신호의 총합이 활성화를 일으키는지를 정하는 역할을 함

폰트, 타이포그래피, 친필, 서예이(가) 표시된 사진

자동 생성된 설명

폰트, 화이트, 서예, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

가중치가 곱해진 입력 신호의 총합을 계산하고, 그 합을 활성화 함수에 입력해 결과를 내는 2단계로 처리된다. 위와 같이 2개의 식으로 나눌 수 있다.

원, 도표, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

기존 뉴런의 원을 키우고, 그 안에 활성화 함수의 처리과정을 명시적으로 그려 넣었다.

즉, 가중치 신호를 조합한 결과가 a라는 노드가 되고, 활성화 함수 h()를 통과하여 y라는 노드로 변환되는 과정이 분명하게 나타나 있다.

스케치, 도표, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명

왼쪽은 일반적인 뉴런, 오른쪽은 활성화 처리 과정을 명시한 뉴런(a는 입력 신호의 총합, h()는 활성화 함수, y는 출력)

**3.2 활성화 함수**

**3.2.1 시그모이드 함수**

폰트, 화이트, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

exp(-x)는 를 뜻하며, e는 자연상수로 2.7182..의 값을 갖는 실수이다.

시그모이드 함수는 단순한 ‘함수’일 뿐이다.

h(1.0)=0.731…, h(2.0)=0.880…를 출력

신경망에서는 활성화 함수로 시그모이드 함수를 이용하여 신호를 변환하고, 그 변환된 신호를 다음 뉴런에 전달한다.

**3.2.2 계단 함수 구현하기**

def step\_function(x):

if x>0:

return 1

else:  
 return 0

인수 x는 실수만 받아들여, 넘파이 배열이 불가능

->

def step\_function(x):

y=x>0

return y.astype(np.int)

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.2.2.ipynb>

위처럼 넘파이 배열의 자료형을 반환할 때 astype()를 사용

**3.2.3 계단 함수의 그래프**

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.2.3.ipynb>

**3.2.4 시그모이드 함수 구현하기**

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.2.4.ipynb>

**3.2.5 시그모이드 함수와 계단 함수 비교**

라인, 도표, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

시그모이드 함수는 부드러운 곡선, 계단함수는 0을 기점으로 출력이 급격히 변화

출력은 0~1 사이가 공통점

**3.2.6 비선형 함수**

둘 모두 비선형 함수이다.

선형 함수는 층을 아무리 깊게 해도 ‘은닉층이 없는 네트워크’로도 똑 같은 기능을 할 수 있다.

층을 쌓는 혜택을 얻고 싶다면 활성화 함수로는 반드시 비선형 함수를 사용해야 한다.

**3.2.7 ReL U 함수**

라인, 도표, 직사각형, 그래프이(가) 표시된 사진

자동 생성된 설명

폰트, 화이트, 텍스트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

**3.3 다차원 배열의 계산**

**3.3.1 다차원 배열**

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.3.1.ipynb>

도표, 번호, 원, 디자인이(가) 표시된 사진

자동 생성된 설명

**3.3.2 행렬의 곱**

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

도표, 텍스트, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

대응하는 차원의 원소가 같아야 된다.

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.3.2.ipynb>

**3.4 3층 신경망 구현하기**

**3.4.1 표기법 설명**

도표, 텍스트, 원, 폰트이(가) 표시된 사진

자동 생성된 설명

**3.4.2 각 층의 신호 전달 구현하기**

도표, 원, 라인, 그림이(가) 표시된 사진

자동 생성된 설명

폰트, 친필, 타이포그래피, 화이트이(가) 표시된 사진

자동 생성된 설명

폰트, 텍스트, 타이포그래피, 화이트이(가) 표시된 사진

자동 생성된 설명

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.4.2.ipynb>

**3.4.3 구현 정리**

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.4.3.ipynb>

**3.5 출력층 설계하기**

**3.5.1 항등 함수와 소프트맥스 함수 구현하기**

항등 함수: 입력을 그대로 출력, 입력 신호가 그대로 출력 신호가 됨

원, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명도표, 원, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.5.1.ipynb>

**3.5.2 소프트맥스 함수 구현 시 주의점**

컴퓨터 계산할 때는 오버플로 문제가 있다

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.5.2.ipynb>

**3.5.3 소프트맥스 함수의 특징**

<https://github.com/sinyebin/DeepLearning-from-Scratch/blob/master/3.%EC%8B%A0%EA%B2%BD%EB%A7%9D/3.5.3.ipynb>

**3.5.4 출력층의 뉴런 수 정하기**

분류에서는 분류하고 싶은 클래스 수로 설정하는 것이 일반적

**3.6 손글씨 숫자 인식**

**3.6.1 MNIST 데이터셋**

**3.6 정리**

신경망에서는 활성화 함수로 시그모이드 함수와 ReLU 함수 같은 매끄럽게 변화하는 함수를 이용한다

넘파이의 다차원 배열을 잘 사용하면 신경망을 효율적으로 구현할 수 있다.

기계학습 문제는 크게 회귀와 분류로 나눌 수 있다.

출력층의 활성화 함수로는 회귀에서는 주로 항등 함수를, 분류에서는 주로 소프트맥스 함수를 이용한다.

분류에서는 출력층의 뉴런 수를 분류하려는 클래스 수와 같게 설정한다.

입력 데이터를 묶은 것을 배치라 하며, 추론 처리를 이 배치 단위로 진행하면 결과를 훨씬 빠르게 얻을 수 있다.