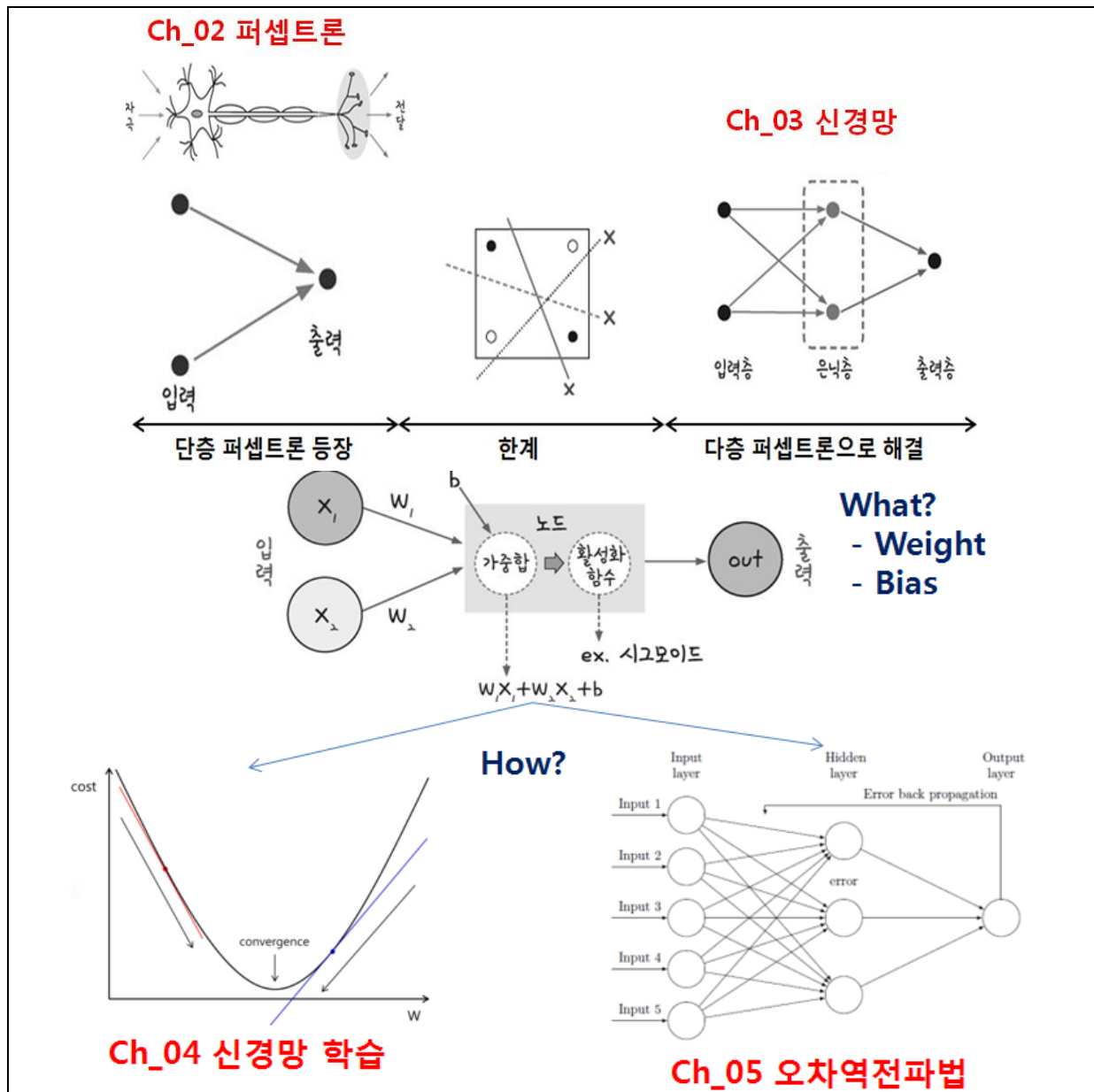


Ch_00. Summary

Ch_01	Ch_02	Ch_03	Ch_04	Ch_05
헬로 파이썬	퍼셉트론	신경망	신경망 학습	오차역전파법

[각 단원별 관계/흐름]



- Ch_02. 인간의 뉴런을 모방한 퍼셉트론을 제시, XOR 연산 구현 안 되는 한계 존재
- Ch_03. 퍼셉트론에 은닉층을 더한 심층망으로 XOR 문제 해결, 활성화 함수와 배열 활용
- Ch_04. 데이터 학습을 통해 가중치와 편향 정보를 설계, 기울기를 이용한 경사법
- Ch_05. output 에서 input 방향으로 오차를 전파해 가중치를 재업데이트 하는 방법

Ch_01. 헬로 파이썬

1. 파이썬의 개념

가. 파이썬의 정의

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
 - ※ 파이썬 소프트웨어 재단(Python Software Foundation, PSF): 2001년에 설립, 비영리 단체

나. 파이썬의 특징

공동 작업 유지보수성	- 공동 작업과 유지 보수가 매우 쉽고 편함
쉽고 빠른 습득 언어	- 쉽고 간결하며 사람의 사고 체계와 유사함
오픈 소스	- '자비로운 종신독재자(BDFL, Benevolent Dictator for Life)' 체제로 운영 ※ BDFL은 오픈소스 커뮤니티에서 사용되는 용어 소스코드를 수정하는 최종 권한을 갖거나 전체적인 개발 방향을 정해주는 사람 (주로 해당 오픈소스 기술을 처음 만든 창시자) → 파이썬 커뮤니티에서도 수많은 개발자가 소스코드 개선에 참여하지만 최종적인 수정 권한은 귀도 반 로섬이 결정
융합 접착(glue)	- 단점: 시스템 프로그래밍이나 하드웨어 제어와 같은 매우 복잡하고 반복 연산이 많은 프로그램은 파이썬과 어울리지 않음 - 극복: 다른 언어로 만든 프로그램을 파이썬 프로그램에 포함시킬 수 있음 Ex) 프로그램의 전반적인 뼈대는 파이썬으로 만들고, 빠른 실행 속도를 필요로 하는 부분은 C로 만들어서 파이썬 프로그램 안에 포함시키는 것
비동기식 코딩	- 쓰레딩 대신 단일 이벤트 루프를 사용해 소수 유닛에서 작업하는 비동기식 코드를 작성하는 데 뛰어남 - 자원 경쟁이나 교착상태를 유발하지 않고도 작성/유지보수 가능
간결	- 가장 좋은 방법 1가지만 이용하는 것을 선호(철학)

다. 파이썬 활동 영역

시스템 유틸리티 제작	- 파이썬은 운영체제(윈도우, 리눅스 등)의 시스템 명령어들을 이용할 수 있는 각종 도구를 포함
GUI 프로그래밍	- 기본 모듈인 Tkinter(티케이인터)를 이용해 만드는 GUI 프로그램 생성가능(5줄의 소스 코드만으로도 윈도우 창을 띄울 수 있음) ※ Tkinter, wxPython, PyQt, PyGTK
C/C++와의 결합 접착(glue) 언어	- 다른 언어와 결합해서 사용 1) C나 C++로 만든 프로그램을 파이썬에서 사용할 수 있음 2) 파이썬으로 만든 프로그램 역시 C나 C++에서 사용할 수 있음
웹 프로그래밍	- 브라우저를 활용해 이용하는 웹사이트 제작
수치 연산 프로그 래밍	- (부적합) 수치가 복잡하고 연산이 많다면 C같은 언어로 하는 것이 더 빠름

	<ul style="list-style-type: none"> - 파이썬 Numeric Python 수치 연산 모듈 제공 (C로 작성되었기 때문에 파이썬에서도 수치 연산을 빠르게 할 수 있음)
데이터베이스 프로그래밍	<ul style="list-style-type: none"> - 사이베이스(Sybase), 인포믹스(Infomix), 오라클(Oracle), 마이에스큐엘(MySQL), 포스트그레스큐엘(PostgreSQL) 등의 데이터베이스에 접근할 수 있게 해주는 도구들을 제공 - 피클(pickle) 모듈: 파이썬에서 사용되는 자료들을 변형 없이 그대로 파일에 저장하고 불러오는 기능 제공
데이터 분석/ 사물 인터넷	<ul style="list-style-type: none"> - 판다스(Pandas) 모듈: 쉽고 효과적인 데이터 분석 가능 - 사물 인터넷 분야에서도 파이썬은 활용도가 높음 Ex) 라즈베리파이(Raspberry Pi) 제어하는 도구로 사용

- 파이썬으로 할 수 없는 일: 시스템과 밀접한 프로그래밍 영역, 모바일 프로그래밍

2. 파이썬 프로그래밍

산술연산	- * 곱셈, ** 거듭제곱, / 나눗셈, % 나머지, // 몫	
자료형	숫자형	<ul style="list-style-type: none"> - 정수 123, -345, 0 / 실수 123.45, -1234.5, 3.4e10 8진수 0o34, 0o25 / 16진수 0x2A, 0xFF
	문자형	<ul style="list-style-type: none"> - 이스케이프 코드: \n 개행 (줄바꿈) / \t 수평 탭 / \w 문자 "w" / \w' 단일 인용부호(') / \w"이중 인용부호(") - 슬라이싱 a[0:3] / a[시작 번호:끝 번호] / 0 <= a < 3 - 포맷 코드: %s 문자열(String) / %c 문자 1개(character) / %d 정수 (Integer) / %f 부동소수(floating-point)/ %o 8진수 / %x 16진수/ %% Literal % (문자 % 자체) - 함수: count, find, index, join, upper, lower, lstrip, rstrip, strip, replace, split
	리스트	<ul style="list-style-type: none"> - 리스트명 = [요소1, 요소2, 요소3, ...] - 리스트 반복하기(*) - 함수: append, sort, reverse, index, insert, remove, pop, count, extend
	튜플	<ul style="list-style-type: none"> - 리스트는 [과]으로 둘러싸지만 튜플은 (과)으로 둘러쌈. - 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없음
	딕셔너리	<ul style="list-style-type: none"> - 리스트나 튜플처럼 순차적으로(sequential) 해당 요소값을 구하지 않고 Key를 통해 Value를 얻는 사전 같은 구조 - 함수: Key 리스트 만들기(keys) / Value 리스트 만들기(values) / Key, Value 쌍 얻기(items) / Key: Value 쌍 모두 지우기 (clear) / Key로 Value얻기(get)해당 Key가 딕셔너리 안에 있는지 조사하기(in) - Key는 고유한 값이므로 중복되는 Key 값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시된다는 점을 주의

	집합	<ul style="list-style-type: none"> - 교집합 & / intersection - 합집합 / union - 차집합 - / difference - 함수: add / update / remove
	불	- True / False
	변수	- 파이썬에서 사용하는 변수는 객체를 가리키는 것
제어문	if	if 조건문: 수행할 문장1 ... elif 조건문: ... else:
	while	while <조건문>: <수행할 문장1>
	for	for 변수 in 리스트(또는 튜플, 문자열): 수행할 문장1 ...
함수	- 명령의 묶음 def 함수명(매개변수): <수행할 문장1> ...	
클래스	Class 클래스_이름: Def __init__(self, 인수, ...): # 생성자(Constructor)	

출처: 점프 투 파이썬 <https://wikidocs.net/book/1>

3. 파이썬 주요 라이브러리

가. 주요 라이브러리

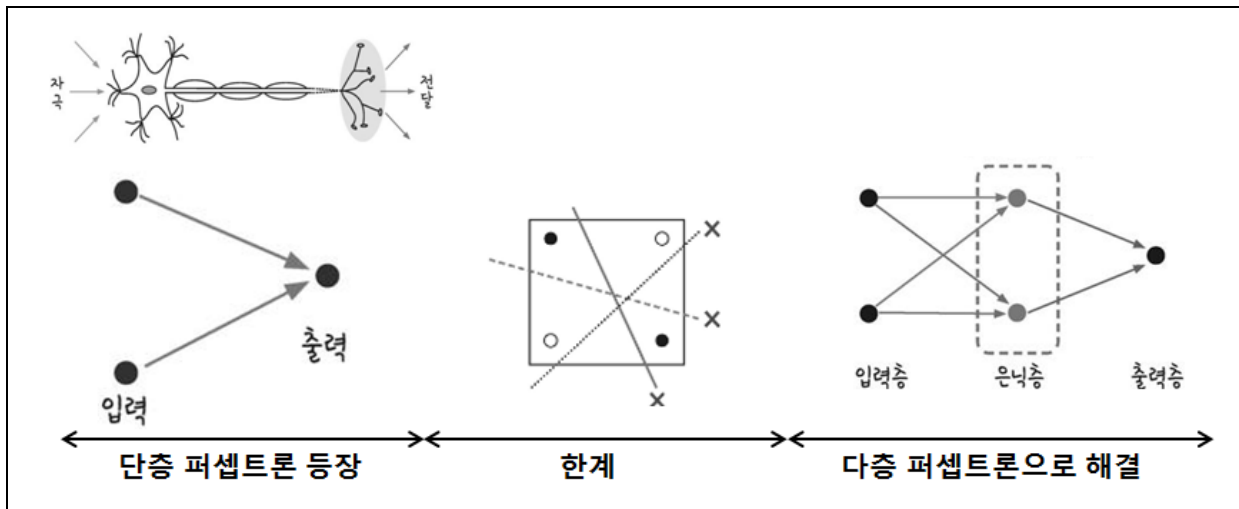
numpy	- 산술 연산, 브로드캐스트*, 배열, 행렬 연산 등
matplotlib	- 그래프 그려주는 라이브러리, 데이터 시각화 등
pyplot	- 그래프 추가 기능, 이미지 표시 등

나. 넘파이 브로드캐스트

	<ul style="list-style-type: none"> - 형상이 다른 배열간 계산 - 확장된 행렬 연산
출처: https://studymake.tistory.com/44	

Ch_02. 퍼셉트론

1. 단층 퍼셉트론의 한계와 다층 퍼셉트론의 등장

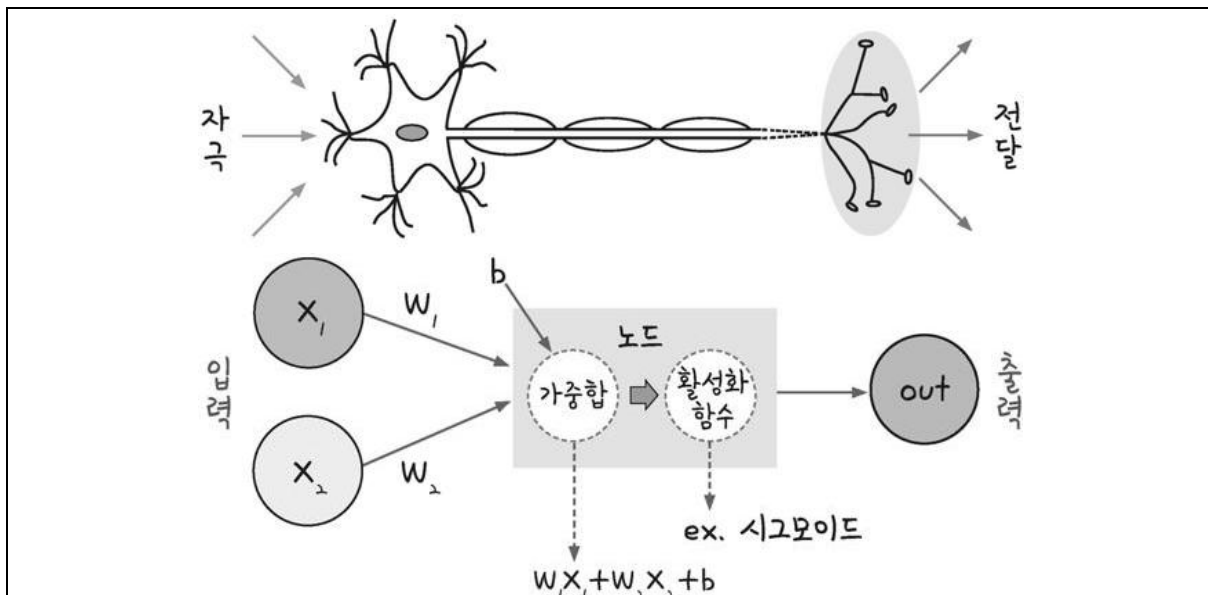


- 인간의 뉴런을 모방한 퍼셉트론 알고리즘으로 인공지능 구현이 가능할 것이라 예상했으나 XOR(Exclusive-OR) 문제조차 해결되지 않는 문제에 직면
- 이를 해결하기 위해 은닉층을 가지는 다층 퍼셉트론이 등장

2. 단층 퍼셉트론의 개념

가. 단층 퍼셉트론(perceptron)의 개념도

- 프랑크 로젠블라트가 1957년 고안한 신경망(딥러닝)의 기원이 되는 알고리즘



- 다수의 신호를 입력받아 하나의 신호(0/1)을 출력하는 알고리즘
- 자극을 받아 전위 변화를 일으킴으로써 임계값을 넘는 인간의 뉴런을 모방한 알고리즘/신경망의 단위

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

[퍼셉트론 공식]

- 가중치와 입력의 합이 임계치를 넘으면 1
- 가중치와 입력의 합이 임계치를 못 넘으면 0

나. 단층 퍼셉트론의 구성요소

구성요소		세부내용
입력	Input	- 입력 노드, (단위) 데이터
바이어스	Bias	- 편향, 임계값(threshold)를 넘는데 영향을 주는 편향 정보
가중치	Weight	- 각 입력노드가 임계값에 영향을 주는 정도(영향력)
활성화 함수	Activate Function	- 출력의 형태를 조정하여 임계치를 넘을지 결정시키는 함수 - 임계값까지 입력과 가중치의 합계를 활성화 하는 함수
출력	Output	- '흐른다/안 흐른다', '켜진다/꺼진다'와 같이 이진으로 표현될 수 있는 출력 노드

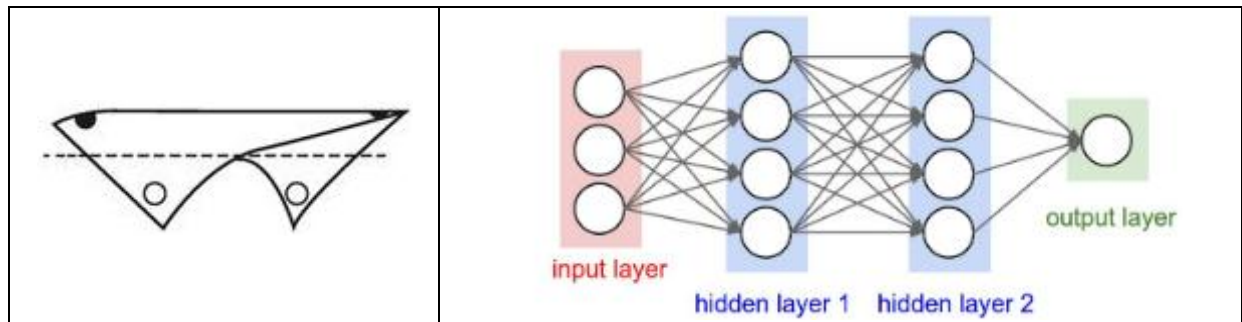
다. 진리표 구현으로 본 단층 퍼셉트론 알고리즘의 문제점

AND	OR	XOR

- 일차 방정식 형태를 가지는 Linear 한 단층 퍼셉트론으로 XOR 표현이 불가능(문제)

3. 다층 퍼셉트론의 개념

가. 다층 퍼셉트론의 개념도



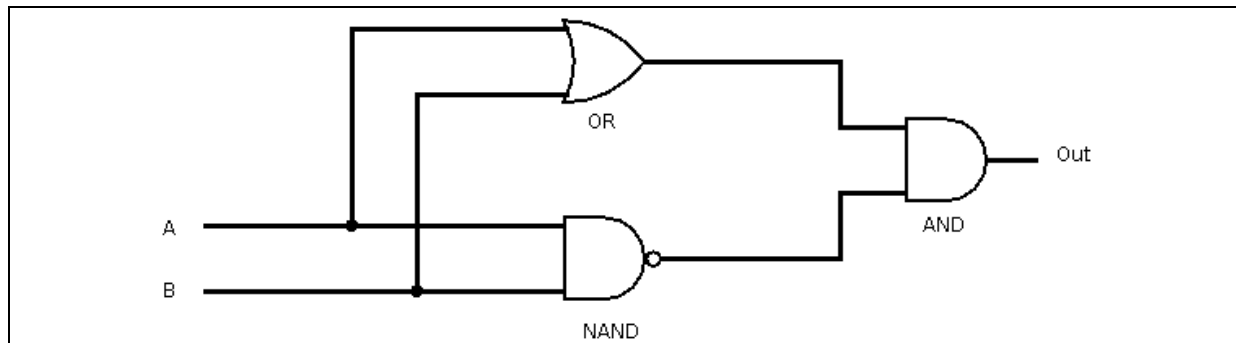
- 단층 퍼셉트론의 한계인 XOR 문제를 해결하기 위해 2차원 평면에 변화를 준 개념
- 2차원 평면에서는 Linear 하게 해결되지 않지만 3차원을 가정한다면 직선으로 XOR 문제를 해결 할 수 있으며, 은닉층을 둬서 평면(차원)을 왜곡 시킬 수 있음(해결)

나. 다층 퍼셉트론의 구성요소

구성요소		세부내용
단층 퍼셉트론의 구성요소 +		
은닉층	Hidden Layer	- Input 노드와 Output 노드 사이에 은닉한 층으로 차원을 왜곡하는 결과를 가져오는 Layer - 은닉층에서 일어나는 일은 사람이 알 수 없는 Black Box

4. 논리게이트와 파이썬 소스 코드로 표현하는 다층 퍼셉트론

가. 논리게이트로 표현하는 다층 퍼셉트론



출처: <https://sullystationtechnologies.com/npxorgate.html>

- OR, NAND, AND 를 이용하여 XOR 구현이 가능(단층 대비 Layer 증가)

X ₁	X ₂	OR
0	0	0
0	1	1
1	0	1
1	1	1

X ₁	X ₂	NAND
0	0	1
0	1	1
1	0	1
1	1	0

OR	NAND	AND
0	1	0
1	1	1
1	1	1
1	0	0

XOR
0
1
1
0

- OR, NAND 층을 추가하여 단층 퍼셉트론은 해결 못한 XOR 결과를 도출할 수 있음

나. 파이썬 소스코드로 표현하는 다층 퍼셉트론

- 4-가. 논리 게이트를 파이썬으로 작성
- AND, OR, NAND 를 먼저 정의

AND	OR	NAND
<pre>import numpy as np def AND(x1, x2): x = np.array([x1, x2]) w = np.array([0.5, 0.5]) b = -0.7 tmp = np.sum(w * x) + b if tmp <= 0: return 0 else: return 1</pre>	<pre>def OR(x1, x2): x = np.array([x1, x2]) w = np.array([0.5, 0.5]) b = -0.3 tmp = np.sum(w * x) + b if tmp <= 0: return 0 else: return 1</pre>	<pre>def NAND(x1, x2): x = np.array([x1, x2]) w = np.array([-0.5, -0.5]) b = 0.7 tmp = np.sum(w * x) + b if tmp <= 0: return 0 else: return 1</pre>
<pre>print('AND') print(AND(0, 0)) print(AND(1, 0)) print(AND(0, 1)) print(AND(1, 1))</pre>	<pre>print('OR') print(OR(0, 0)) print(OR(1, 0)) print(OR(0, 1)) print(OR(1, 1))</pre>	<pre>print('NAND') print(NAND(0, 0)) print(NAND(1, 0)) print(NAND(0, 1)) print(NAND(1, 1))</pre>
AND 0 0 0 1	OR 0 1 1 1	NAND 1 1 1 0

XOR		
<pre>def XOR(x1, x2): # layer1 s1_1 = OR(x1, x2) s1_2 = NAND(x1, x2) # layer2 s2 = AND(s1_1, s1_2) return s2</pre>	<pre>print('XOR') print(XOR(0, 0)) print(XOR(1, 0)) print(XOR(0, 1)) print(XOR(1, 1))</pre>	XOR 0 1 1 0

- 첫번째 레이어에서 OR, NAND를 수행하고 두번째 레이어에서 AND 를 수행한 결과
- XOR 출력을 얻을 수 있음

다. 계층을 추가하여 구현한 XOR

- 4-가/나 의 형태가 아니라도 XOR 구현이 가능함
- 계층을 추가하고 NAND 만을 이용하여 XOR 구현이 가능함

출처: https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:XOR_from_NAND.svg

X1	X2	NAND
0	0	1
0	1	1
1	0	1
1	1	0

X1	NAND	NAND
0	1	1
0	1	1
1	1	0
1	0	1

X2	NAND	NAND
0	1	1
1	1	0
0	1	1
0	0	1

NAND	NAND	NAND
1	1	0
1	0	1
0	1	1
1	1	0

XOR
0
1
1
0

<pre>def XOR(x1, x2): # layer1 s1 = NAND(x1, x2) # layer2 s2_1 = NAND(x1, s1) s2_2 = NAND(x2, s1) # layer3 return NAND(s2_1, s2_2)</pre>	<pre>print('XOR') print(XOR(0, 0)) print(XOR(1, 0)) print(XOR(0, 1)) print(XOR(1, 1))</pre>	XOR 0 1 1 0
--	---	-------------------------

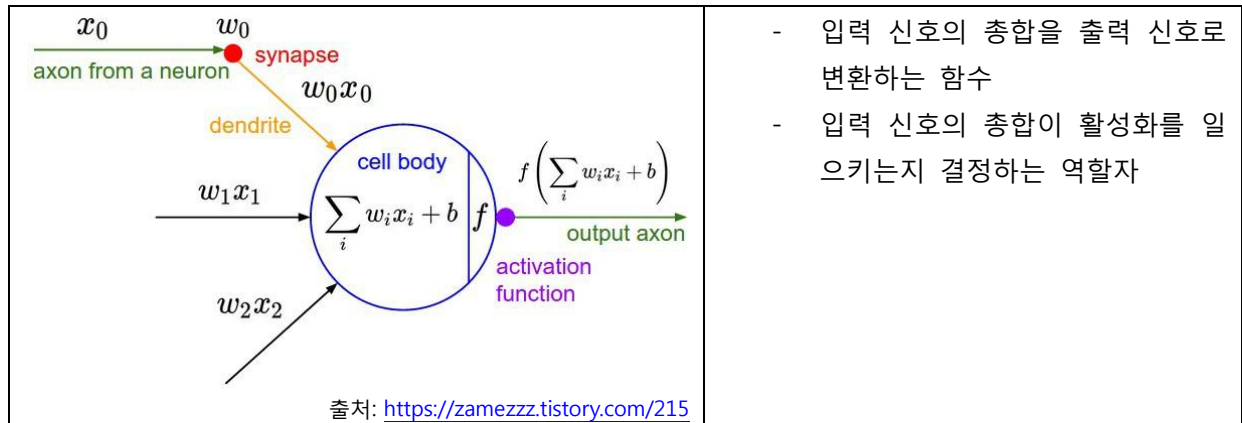
Ch_03. 신경망

1. 퍼셉트론과 신경망의 차이

가중치 설정 작업	Ch_02 퍼셉트론	Ch_03 신경망
	- 사람 수동 설정	- 데이터 자동 학습 설정

2. 활성화 함수

가. 활성화 함수의 개념



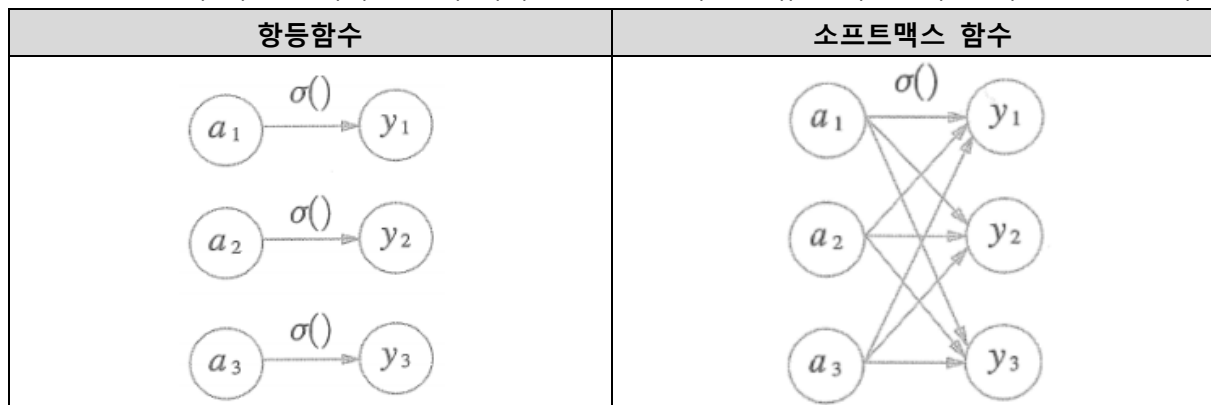
나. 활성화 함수의 유형

계단함수	시그모이드함수	ReLU 함수
$H[n] = \begin{cases} 0, & n < 0, \\ 1, & n \geq 0, \end{cases}$	$h(x) = \frac{1}{1 + \exp(-x)}$	$h(x) = \begin{cases} x(x > 0) \\ 0(x \leq 0) \end{cases}$

3. 출력층 설계하기

가. 활성화 함수

- 풀고자 하는 문제의 성질이 회귀면 항등, 2클래스 분류는 시그모이드, 다중은 소프트맥스



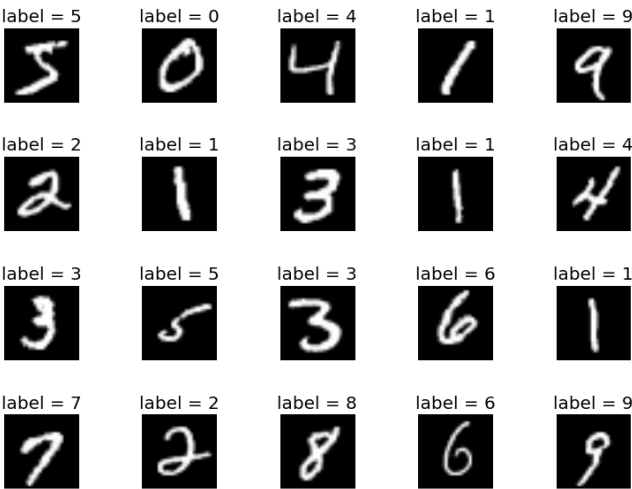
나. 소프트맥스 함수 구현 시 주의점

$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1}^n \exp(a_i)}$ $= \frac{\exp(a_k + \log C)}{\sum_{i=1}^n \exp(a_i + \log C)}$ $= \frac{\exp(a_k + C')}{\sum_{i=1}^n \exp(a_i + C')}$ <p>출처: https://eehoeskrap.tistory.com/144</p>	<ul style="list-style-type: none"> - 컴퓨터 구현 시 계산의 결함 존재 - 오버플로 - Divide by Zero - $\log C$ 방지책 - 단조 증가 되어 성질이 유지됨
--	---

- 소프트맥스 함수는 출력이 0~1.0 사이의 실수이며 출력 총합이 1인 것이 특징(확률적)
- 추론 단계에서는 소프트맥스 함수를 생략하고 학습시킬 때는 사용하는 것이 일반적

4. 손글씨 숫자 인식

가. MNIST 데이터 셋

 <p>출처: https://corochann.com/mnist-dataset-introduction-1138.html</p>	<ul style="list-style-type: none"> - 손글씨 숫자 이미지 집합 - 0~9 숫자 이미지 - 레이블 포함 - 훈련데이터 60,000 - 시험데이터 10,000 - 28 X 28 크기 0~255 픽셀 - 회색조 이미지(1채널) - load_mnist() 함수
--	---

- 원-핫 인코딩: 정답 원소만 1 나머지는 0으로 표현하는 방식

나. 데이터의 전처리

$Z = \frac{X - \mu}{\sigma}$	<ul style="list-style-type: none"> - 0.0~1.0으로 범위 변환 - 정규화 normalize = true
------------------------------	---

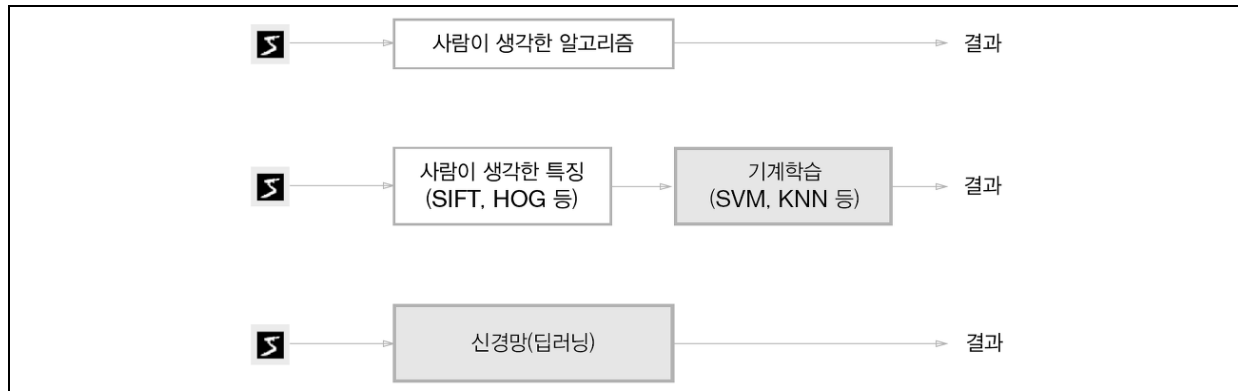
다. 배치 처리

- 컴퓨터 처리 시 이점 존재

Ch_04. 신경망 학습

1. 데이터 주도 학습

- 이미지에서 특징(feature)를 추출하고 그 특징의 패턴을 기계학습 기술로 학습하는 방법

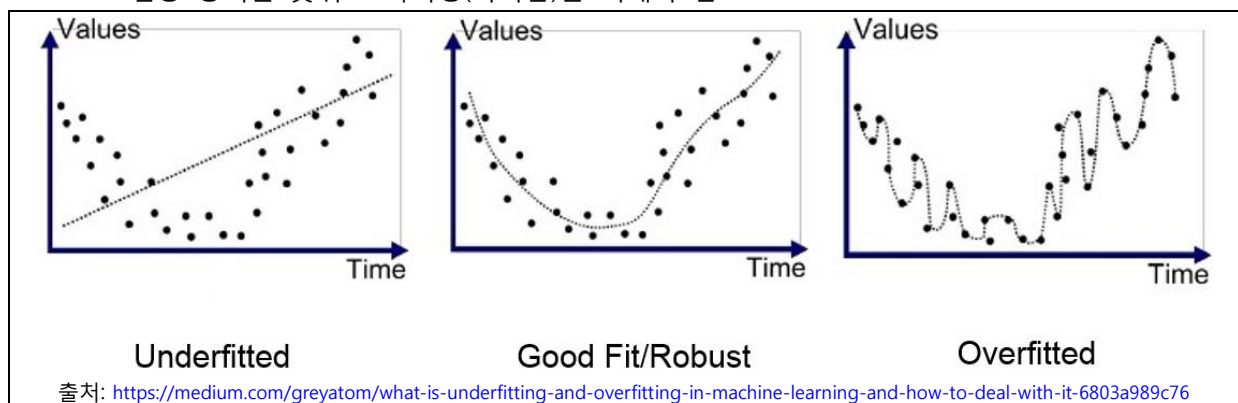


- 중요한 특징까지 기계가 학습하게 하는 종단간 기계학습이 목표

가. 데이터의 유형

훈련 데이터	시험 데이터
<ul style="list-style-type: none"> - 최적의 매개변수를 찾기 위한 데이터 	<ul style="list-style-type: none"> - 훈련한 모델을 평가하는 데이터

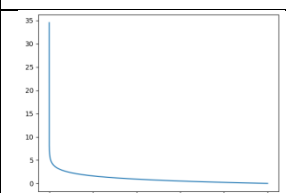
- 범용 능력을 가진 모델을 만들기 위한 방법
- 범용 능력을 갖춰 오버피팅(과적합)을 피해야 함



- 오버피팅은 조기종료, 드랍아웃 등으로 예방 가능

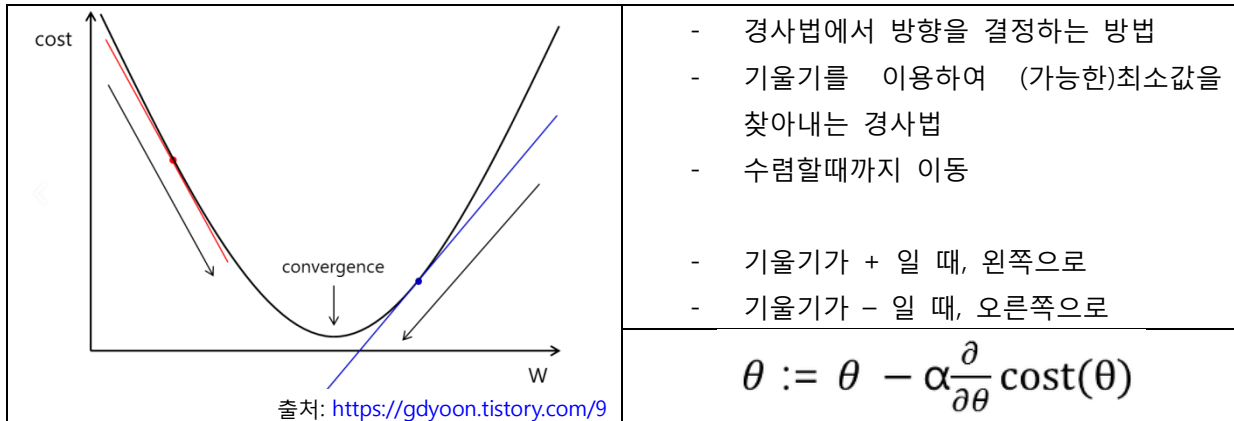
나. 손실 함수

- 매개변수를 탐색하기 위한 지표, '정확도'를 올리기 위한 지표

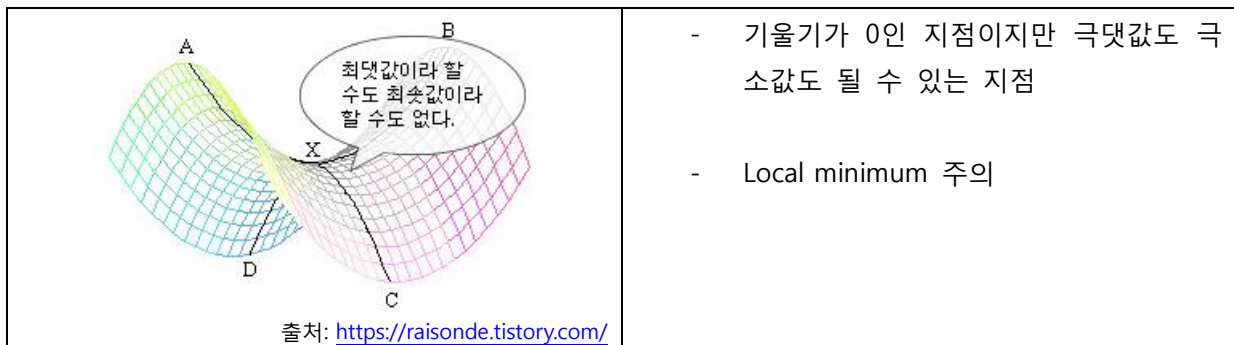
평균 제곱 오차	교차 엔트로피 오차
$E = \frac{1}{2} \sum_k (y_k - t_k)^2$	$E = - \sum_k t_k \log y_k$
<ul style="list-style-type: none"> - 정답과 예측값의 차이를 기준으로 함 <p>출처: https://kolikim.tistory.com/36</p>	 <p>t_k : 정답 레이블 y_k: 예측값</p>

2. 기울기

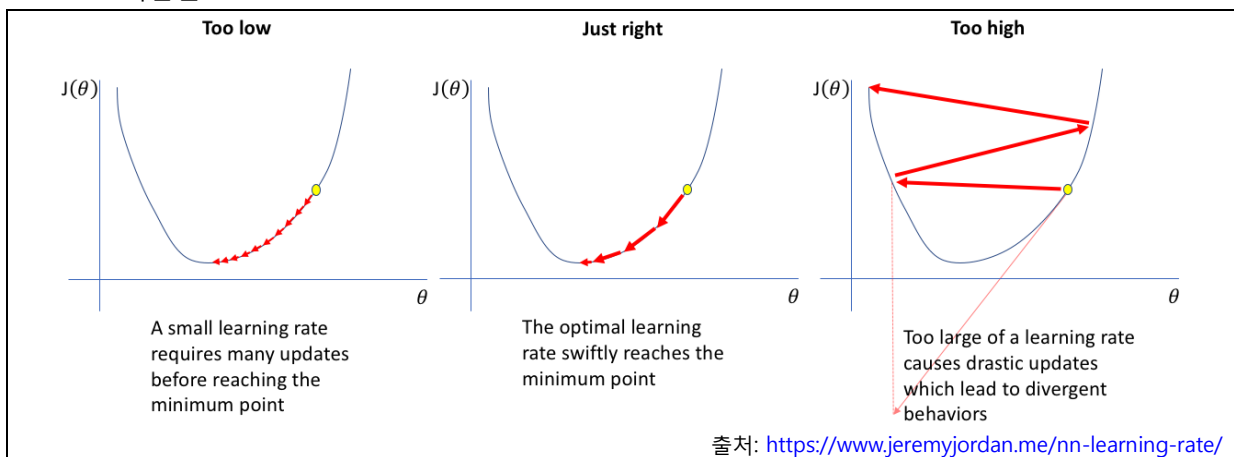
가. 경사법(경사 하강법)



a. 안장점



b. 학습률



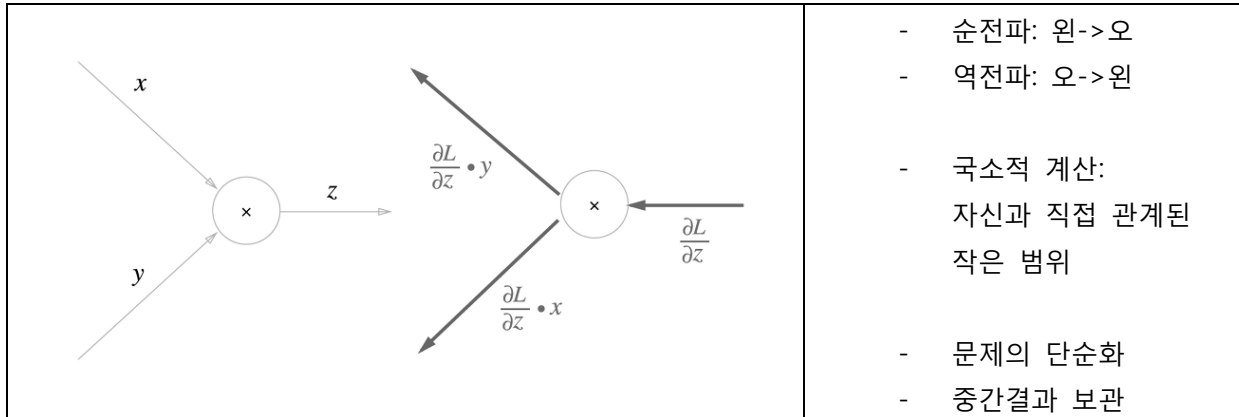
참고) 1에폭: 학습에서 훈련 데이터를 모두 소진했을 때의 횟수

Ch_05. 오차역전파법

1. 계산 그래프를 이용한 오차역전파법

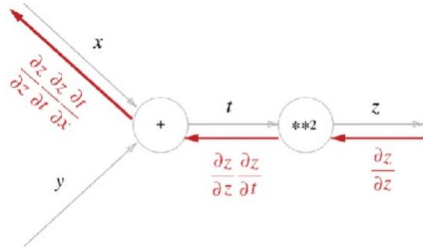
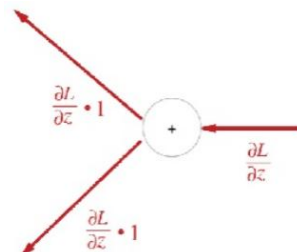
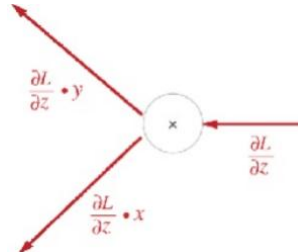
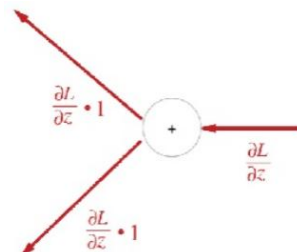
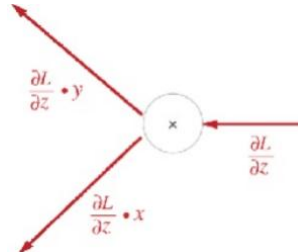
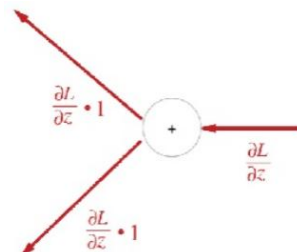
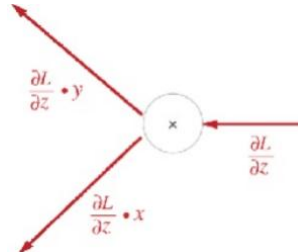
가. 계산그래프

- 노드와 엣지로 표현된 그래프

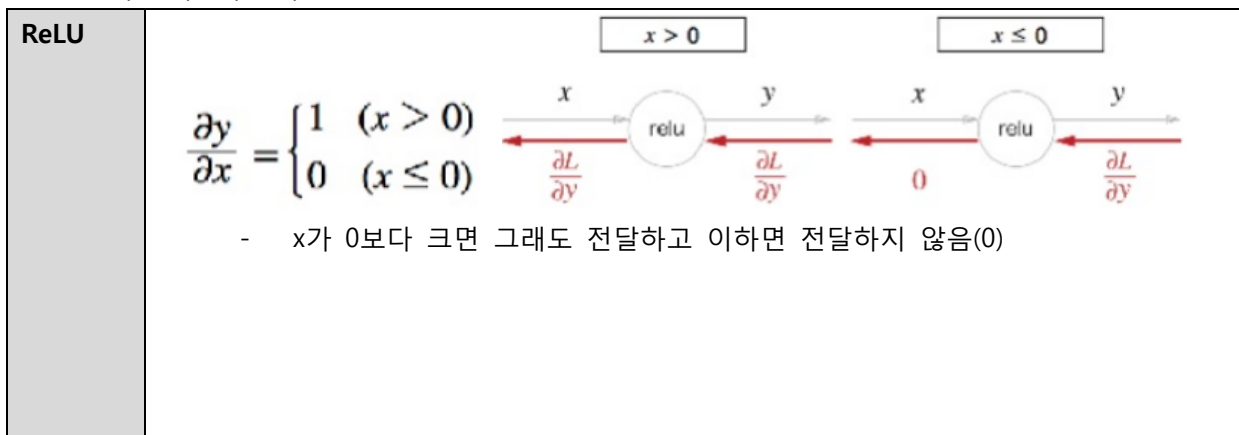


- 각 변수의 미분을 효율적으로 구할 수 있음

나. 연쇄법칙과 역전파

<div>$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x}$</div> <div>- 합성함수의 미분은 각 함수의 미분의 곱</div>	<table><tr><th>덧셈 노드 역전파</th><th>곱셈 노드 역전파</th></tr><tr><td>$\frac{\partial z}{\partial x} = 1$$\frac{\partial z}{\partial y} = 1$</td><td>$\frac{\partial z}{\partial x} = y$$\frac{\partial z}{\partial y} = x$</td></tr><tr><td></td><td></td></tr><tr><td>입력된 값 그대로 전달</td><td>편미분 역전되서 전달(x↔y)</td></tr></table>	덧셈 노드 역전파	곱셈 노드 역전파	$\frac{\partial z}{\partial x} = 1$ $\frac{\partial z}{\partial y} = 1$	$\frac{\partial z}{\partial x} = y$ $\frac{\partial z}{\partial y} = x$			입력된 값 그대로 전달	편미분 역전되서 전달(x↔y)
덧셈 노드 역전파	곱셈 노드 역전파								
$\frac{\partial z}{\partial x} = 1$ $\frac{\partial z}{\partial y} = 1$	$\frac{\partial z}{\partial x} = y$ $\frac{\partial z}{\partial y} = x$								
									
입력된 값 그대로 전달	편미분 역전되서 전달(x↔y)								

2. 활성화 함수 계층 구현



Sigmoid	<div data-bbox="331 235 1378 660"> <div> <p>순전파</p> </div> <div> <p>역전파</p> </div> </div> <div data-bbox="331 683 1378 952"> <p>- 입출력에만 집중한 간소화 버전</p> <div> <div> $\frac{\partial L}{\partial y} y^2 \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{(1 + \exp(-x))^2} \exp(-x)$ $= \frac{\partial L}{\partial y} \frac{1}{1 + \exp(-x)} \frac{\exp(-x)}{1 + \exp(-x)}$ $= \frac{\partial L}{\partial y} y(1-y)$ </div> </div> </div> <p>- 역전파는 순전파의 출력만으로 계산 가능</p>
Affine	<div data-bbox="331 1008 1378 1534"> <p>- Affine 계층: 어파인 변환을 수행하는 처리</p> <p>- 어파인 변환: 순전파 때 수행하는 행렬의 곱을 기하학에서 칭하는 용어</p> <div> <div> $\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}^T$ $\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^T \frac{\partial L}{\partial \mathbf{Y}}$ </div> <div> </div> </div> <p>- 내적을 위해 차원의 원소 수를 일치 시켜야 함</p> </div>
Softmax	<div data-bbox="331 1568 1378 1892"> <p>- Softmax-with-Loss 계층: 입력 전처리(정규화) - Softmax - 교차 엔트로피</p> <p>다음 층으로 역전파</p> <div> <div> <p>Softmax 계층</p> </div> <div> <p>Cross Entropy Error 계층</p> </div> </div> </div>