

#Index

2021년 2월 8일 월요일 오전 9:28

Index

단원	번호	상세 목차	이론설명	수업문제
1장 딥러닝 이란?	1	딥러닝으로 할 수 있는 것은?	바로가기	
	2	넘파이(Numpy) 배열 생성하기		
	3	넘파이(Numpy) 산술 연산		
	4	넘파이(Numpy) N차원 배열		
	5	넘파이(Numpy) 브로드캐스트		
	6	넘파이(Numpy) 원소접근		
	7	Matplotlib 로 그래프 그리기		

번호	상세목차	이론설명	수업문제
2장 퍼셉트론	1	퍼셉트론이란 ?	바로가기
	2	단순한 논리회로	
	3	퍼셉트론 구현하기	
	4	퍼셉트론의 한계	
	5	다층 퍼셉트론 구현하기	

단원	번호	상세목차	이론설명	수업문제
3장 신경망	1	퍼셉트론에서 신경망으로		
	2	활성화 함수	바로가기	
	3	다차원 배열의 계산		
	4	3층 신경망 구현하기	바로가기	바로가기
	5	출력층 설계하기		
	6	손글씨 숫자 인식 구현하기	바로가기	
	7	미니배치로 구현하기	바로가기	

단원	번호	상세목차	이론설명	수업문제
4장 신경망 학습	1	손실함수		
	2	수치미분	바로가기	
	3	경사하강법	바로가기	
	4	SimpleNet 구현하기 (단층)	바로가기	바로가기 바로가기
	5	수치미분으로 2층 신경망 구현하기	바로가기	바로가기 바로가기

단원	번호	상세목차	이론설명	수업문제
5장 오차역전파법	1	계산 그래프	바로가기 바로가기	
	2	연쇄법칙	바로가기	
	3	역전파		
	4	단순한 계층 구현하기		바로가기
	5	활성화 함수 계층 구현하기(렐루)	바로가기	
	6	활성화 함수 계층 구현하기(시그모이드)	바로가기	
	6	Affine 계층 구현하기	바로가기 바로가기	
	7	Softmax 계층 구현하기	바로가기	바로가기
	8	Orderdictionary	바로가기	
	9	오차 역전파법 구현하기	바로가기	
	기타	자코비안 행렬	바로가기	

단원	번호	상세목차	이론설명	수업문제
6장 학습 관련 기술들	1	매개변수 갱신	바로가기	바로가기
	2	가중치 초기값	바로가기	바로가기 바로가기
	3	배치정규화	바로가기	
	4	드롭아웃	바로가기	바로가기 바로가기
	5	keras 가상환경 만들기	바로가기	
	6	keras 의 early stopping 기능 구현하기	바로가기	바로가기

단원	번호	상세목차	이론설명	수업문제
7장 합성곱 신경망	1	완전 연결 계층의 문제점	바로가기	
	2	합성곱 연산	바로가기	바로가기
	3	패딩	바로가기	
	4	스트라이드	바로가기	
	5	3차원 합성곱 연산 첫번째	바로가기1 바로가기2	바로가기1 바로가기2
	6	3차원 합성곱 연산 두번째	바로가기	바로가기
	7	블럭으로 생각하기	바로가기	
	8	배치처리	바로가기	
	9	풀링계층	바로가기	
	10	im2col 함수	바로가기 바로가기	바로가기
	11	CNN 구현하기	바로가기	
	12	CNN 전체코드	바로가기	
	13	최종 스크립트의 큰 그림	바로가기	
	14	고전 CNN LeNet 구현하기	바로가기	

단원	번호	상세목차	이론설명	수업문제
8장 텐서플로우	1	텐서 플로우란?	바로가기	
	2	텐서 플로우 환경구성 (Tensorflow 1.x)	바로가기	
	3	텐서 플로우 기본 문법		
	4	넘파이와 텐서 플로우 비교		
	5	Mnist 로 단층 신경망 구현하기		
	6	텐서 플로우로 구현하는 비용함수		
	7	경사감소법을 텐서 플로우로 구현	바로가기	
	8	텐서 플로우로 다층 신경망 구현하기		
	9	텐서 플로우로 CNN 신경망 구현하기		
	10	딥러닝의 역사	바로가기	
	11	신경망에 데이터 로드 함수들 (cifar)	바로가기	
	12	신경망에 데이터 로드 함수들 (이파리)	바로가기	

	13	사진을 32x32로 resize 하는 함수	바로가기	
	14	GPU 에 CUDA / cuDNN 설치 하는 방법	바로가기	확인방법
	15	Tensorflow 1.x 버전 + keras 2.x 설치	바로가기	GPU 확인
	16	Tensorflow 1.x 버전 + keras 2.x 에서 이파리 데이터 분류 신경망 구현하기	바로가기	1.x 와 2.x 비교하기
	17	Tensorflow 1.x 버전 + keras 2.x 에서 이파리 데이터 모델 불러오는 코드	바로가기	
	18	Tensorflow 2.x 환경구성	바로가기	
	19	Tensorflow 2.x 에서 퍼셉트론 구현	바로가기	
	20	Tensorflow 2.x 에서 mnist 신경망	바로가기	
	21	keras 첫번째 층에 input_shape	바로가기	
	22	Tensorflow 2.x 에서 CNN 안쓴 신경망	바로가기	드롭아웃
	23	Tensorflow 2.x 에서 CNN 사용 신경망	바로가기	이파리
	24	Tensorflow 2.x 에서의 이미지 증식	바로가기	
	25	Tensorflow 2.x 에서의 전이학습	바로가기	
	26	이미지넷 우승 CNN - VGG 구현하기	바로가기	바로가기
	27	이미지넷 우승 CNN - ResNet 구현하기	바로가기	
	28	이미지 웹스크롤링 코드 총정리	바로가기	
	29	신경망 활용 홈페이지 만들기	바로가기	바로가기 이파리
	30	Obejct Detection	바로가기	바로가기 바로가기
	31	Gan 으로 이미지 생성	바로가기	바로가기1 바로가기2
	32	미술작품 스타일로 일반 사진 변환	바로가기	환경구성 구현

#EndLine=====

#0304

2021년 2월 8일 월요일 오전 9:28

■ Deep Learning

딥러닝으로 할 수 있는것들

<https://cafe.daum.net/oracleoracle/Sedp/5>

DL 종류 / 책

 <p>파이썬으로 익히는 딥러닝 이론과 구현</p> <p>Deep Learning from Scratch</p> <p>밑바닥부터 시작하는 딥러닝</p>	 <p>파이썬으로 직접 구현하며 배우는 손목 신경망과 자연어 처리</p> <p>Deep Learning from Scratch 2</p> <p>밑바닥부터 시작하는 딥러닝 2</p>	 <p>파이썬으로 직접 구현하며 배우는 딥러닝 프레임워크</p> <p>Deep Learning from Scratch 3</p> <p>밑바닥부터 시작하는 딥러닝 3</p>
CNN : 인공지능의 눈	RNN : 인공지능의 입과 귀 자연어 처리	딥러닝 프레임워크
학원진도는 1권으로만 진행		

ML 과 DL 의 차이점은?

머신러닝으로 기계학습을 하려면 정형화된 데이터가 있어야 한다.

표 형태의 정형화된 데이터를 만들기 위해 판다스를 이용했는데, 이런 정형화된 데이터를 만드는 것은 많은 노력과 시간이 필요하다. 그래서 딥러닝은(CNN) image를 신경망에 보여주기만 하면 스스로 학습하여 해당 이미지를 신경망이 구분할 수 있는 상태가 된다. (해당 이미지의 levels 를 구분할 수 있는 상태까지)

딥러닝 기술을 이용해서 현업에서 하고 있는 일들

파이썬으로 신경망을 짤 수 있는 사람이 C 언어 계열로도 변환 혹은 구현이 가능한 개발자 ---> 국내 100명 내외(고급인력)

현업 및 개발 예시

1. 인천공항에 컨테이너 검색대에 물건을 올리면 CNN으로 판별하여 위반되는 물품이 있는지 검사하는 신경망 구현
2. 의료계 : X-ray 사진과 의료영상 사진의 질병여부를 컴퓨터가 판별
3. 물품(의류)의 불량품을 판별하는 신경망 개발 + 인터페이스
4. 주가 예측 신경망 개발
5. 외국어 번역하는 신경망
6. 인공지능 변호사 (법률책 ---> 신경망 ---> 판결)

7. 음악 작곡 신경망(Gan) ---> 아마존 키보드(일반음악을 jazz 풍으로 변경 등)

□넘파이(numpy) 배열 생성하기

numpy 는 신경망 개발의 핵심 요소로 사용된다.

numpy

파이썬 언어에서 기본적으로 지원하지 않는 배열(array), 행렬(matrix)의 계산을 쉽게 구현해주는 라이브러리

머신러닝에서 많이 사용하는 선형대수학에 관련된 수식들을 파이썬에서 쉽게 프로그래밍할 수 있는 기본 라이브러리로 활용

문제1. 아래의 행렬을 numpy 로 생성하시오!

```
1 2
3 4
```

```
import numpy as np
x = np.array([[1,2],
              [3,4]])
print(x)
```

문제2. 위 a 행렬의 각 요소에 숫자 5를 더하시오! (broadcasting)

```
import numpy as np
x = np.array([[1,2],
              [3,4]])

x+5
↓
array([[6, 7],
       [8, 9]])
```

문제3. 아래의 두 행렬의 합을 구하시오!

```
1 5 3 4
6 7 2 1
```

```
a = np.array([[1,5],[6,7]])
b = np.array([[3,4],[2,1]])
a+b
↓
array([[4, 9],
       [8, 8]])
```

numpy 의 유용한 기능 중 하나인 브로드캐스트(broadcast)

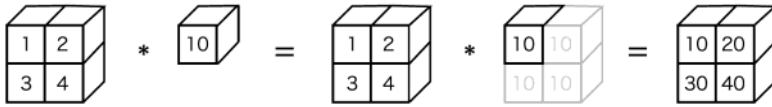
numpy 에서는 형상이 다른 배열끼리도 계산이 가능하다.

문제 2의 경우처럼 숫자 5가 2 by 2 행렬로 확대된 후 연산이 이루어지는 것을 예시로 볼 수 있다.

브로드캐스팅은 사칙연산이 모두 가능하다.

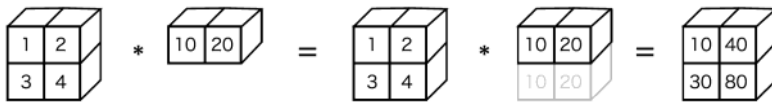
형식은 scalar(상수) * array(배열) 의 형식으로 이루어진다.

문제4. 아래의 브로드 캐스트 기능을 numpy 로 구현하시오!



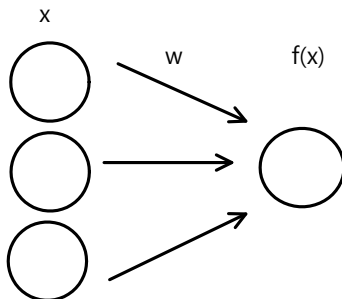
```
import numpy as np
a = np.array([[1,2],[3,4]])
a*10
↓
array([[10, 20],
       [30, 40]])
```

문제5. 그림 1-2 도 브로드캐스팅을 이용해서 구현하시오!



```
import numpy as np
a = np.array([[1,2],[3,4]])
b = np.array([[10,20]])
print(a.shape) #(2, 2)
print(b.shape) #(1, 2) #row 가 확대되어 연산이 된 것을 확인가능
a*b
↓
array([[10, 40],
       [30, 80]])
```

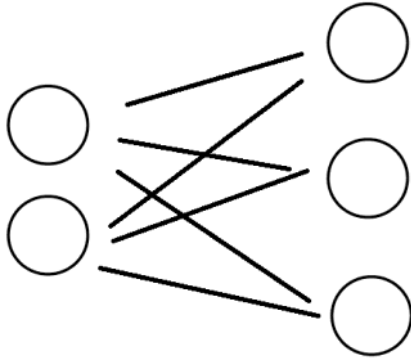
문제6. 아래의 신경망을 numpy 행렬로 구현하시오!



$$x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 = f(x) = a$$

```
import numpy as np
x = np.array( [[2,4,8]] )
w = np.array( [[4,3,2]] )
a = np.sum(x*w)
a #36
```

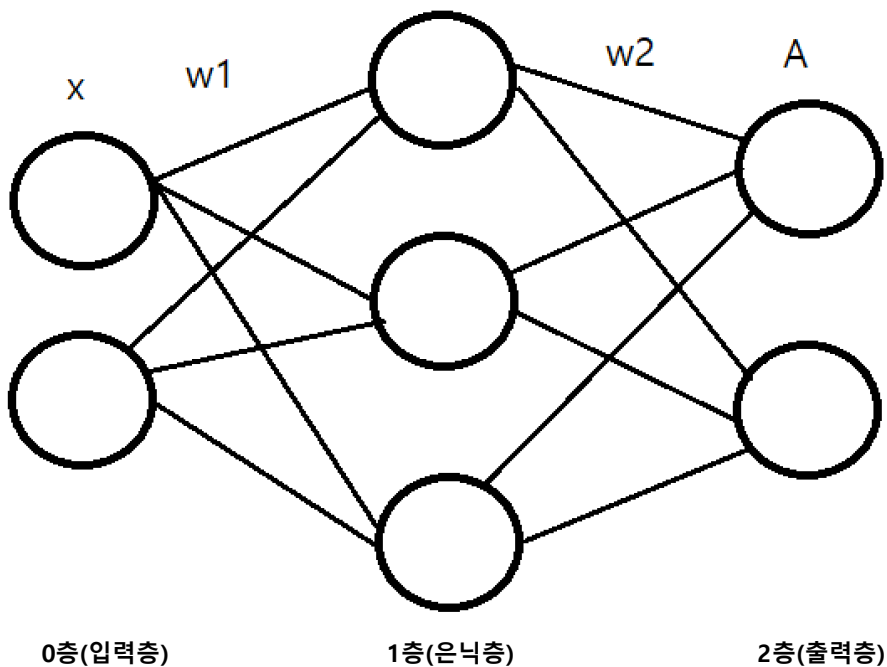
문제6. 아래의 신경망을 numpy 행렬로 구현하시오!



(1, 2) np.dot [[1,3,5], [2,4,6]] = [k1, k2, k3]

```
import numpy as np
x = np.array( [1,2] )
w = np.array( [[1,3,5],[2,4,6]] )
print(x.shape) #(2,)
print(w.shape) #(2, 3)
a = np.dot(x,w) #np.dot() 으로 내적
print(a.shape) #(3,)
a
↓
array([ 5, 11, 17])
```

문제8. 아래의 신경망을 numpy 로 구현하시오!



```
import numpy as np
x = np.array( [1,2] ) #x #(2,) #1차원 ndarray
w1 = np.array( [[1,3,5],[2,4,6]] ) #weight1 #(2,3)
w2 = np.array( [[3,4],[5,6],[7,8]] ) #weight1 #(3,2)

a1 = np.dot(x,w1) #은닉층 노드값
```

```
a2 = np.dot(a1,w2)
```

```
np.ndim(a2) #1 #np.ndim() 은 괄호 안 배열의 차원을 출력  
a2 #array([189, 222]) #result #2, #1차원 ndarray를 출력
```

□ numpy N차원 배열(p.38)

numpy 는 1차원 배열 뿐 아니라 다차원 배열로도 작성 가능

```
import numpy as np  
x = np.array([[1,2],[3,4]])  
print(x.shape) #(2,2)  
print(np.ndim(x)) #2 #2차원 ndarray 값 확인  
type(x) #numpy.ndarray
```

문제9. 아래와 같이 5 by 5 행렬을 생성하시오!

```
1 2 3 4 5  
2 4 3 2 4  
3 1 4 2 1  
2 7 3 5 4  
1 5 6 3 1
```

```
import numpy as np  
x = [1,2,3,4,5,2,4,3,2,4,3,1,4,2,1,2,7,3,5,4,1,5,6,3,1]  
x2 = np.array(x) #리스트를 행렬로 변환  
x3 = x2.(5,5) #행렬의  
print(x3)  
x3.shape
```

넘파이 특징

넘파이 배열은 N차원 배열을 작성할 수 있다.

1차원, 2차원, 3차원 배열과 같이 원하는 차수의 배열을 자유롭게 생성 가능하다.

수학에서는 1차원 행렬은 벡터(vector), 2차원 배열은 행렬(matrix)이며, 벡터와 행렬을 일반화한 것을 텐서(tensor)라고 한다.

tensorflow

tensor : 다차원

flow : 흐름

즉, tensorflow 는 '행렬이 계산되면서 흘러간다' 란 의미를 갖는다.

구글에서 만든 텐서플로우는 다차원 배열의 연산을 빠르게 할 수 있도록 구현되어 있다.

tensorflow 장점

1. 코드가 간결하다
2. 신경망 구현에 필요한 모든 함수들이 내장
3. 속도가 매우 빠름
4. GPU 연산이 구현되어 있음

☆문제10. 아래 행렬의 내적을 구현하시오!

x	function	w	=	f(x)=y
2x3	np.dot	3x2	=	2x2

```
import numpy as np
```

```
x = np.array( [[3,4,2],[4,1,3]])
w = np.array( [[1,5],[2,3],[4,1]])
a = np.dot(x,w)
print(a.shape) #(2,2)
a
↓
array([[19, 29],
       [18, 26]])
```

□numpy 원소 접근

numpy 배열안에 요소들에 대한 접근은 numpy 를 이용하지 않았을 때보다 훨씬 간단하게 구현가능

문제11. 아래의 리스트를 만들고 해당 리스트에서 15 이상인 숫자들만 출력하시오!

```
x = [51,55,14,19,0,4]
for i in x:
    if i >= 15:
        print(i)
```

문제12. 아래의 행렬식을 만들고 아래의 행렬의 요소에서 15 이상인 것만 출력하시오! (numpy 를 이용하지 마시오)

```
x = [[51,55],[14,19],[0,4]]

a = []
for i in x:
    for j in i:
        if j >= 15:
            a.append(j)
print(a)
```

문제13. 위 예제를 numpy 를 이용해서 구현하시오!

```
import numpy as np

x = [51,55,14,19,0,4]
x2 = np.array(x)
x3 = x2.reshape(3,-1) #(3,2)행렬로 재생성(reshape)
print(x3[x3>=15]) #[51 55 19]
#numpy 를 이용하면 코드가 간결해지고 복잡한 계산도 쉽게 수행 가능하다
```

문제14. 아래의 행렬을 만들고 아래의 행렬에서 3 이상인 것만 출력하시오!

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2]])
```

```
import numpy as np
iris = [5.1, 3.5, 1.4, 0.2, 4.9, 3, 1.4, 0.2, 4.7, 3.2, 1.3, 0.2, 4.6, 3.1, 1.5, 0.2]
iris2 = np.array(iris)
iris3 = iris2.reshape(4,-1)

print(iris3.shape) #(4,4)
print(iris3[iris3>=3])
```

□matplotlib 시각화

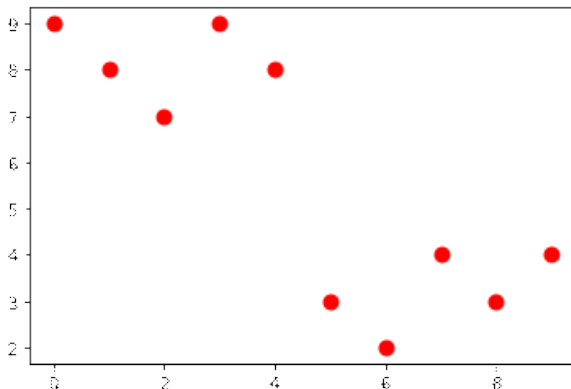
딥러닝 실험에서는 그래프 그리기와 데이터 시각화가 중요하다.

matplotlib 은 그래프를 그리기 위한 가장 대표적인 파이썬 라이브러리 중 하나로, 간단한 코드로 그래프 출력이 가능하다.

신경망에서 주로 사용하는 그래프는 라인그래프이다. 정확도가 점점 증가하는지, error 가 감소하는지 등을 직관적으로 확인 가능.

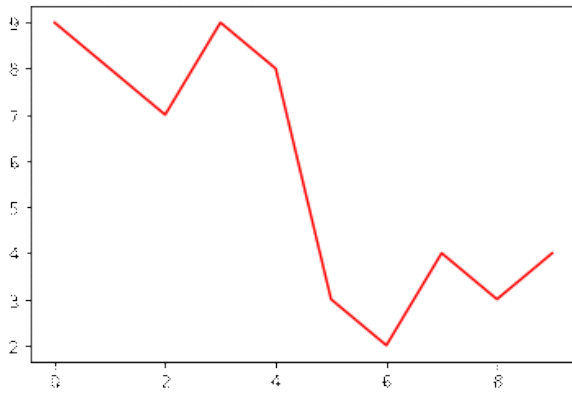
예제1. 파이썬으로 산포도 그래프 그리기

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array( [ 0,1,2,3,4,5,6,7,8,9])
y = np.array( [ 9,8,7,9,8,3,2,4,3,4])
plt.scatter(x, y, color='red', s=80) #s는 점 사이즈 조절 옵션
plt.show()
```



문제15. 위 그래프의 라인 그래프로 그리시오!

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array( [ 0,1,2,3,4,5,6,7,8,9])
y = np.array( [ 9,8,7,9,8,3,2,4,3,4])
plt.plot(x, y, color='red') #plt.plot() 으로 라인그래프 출력
plt.show()
```



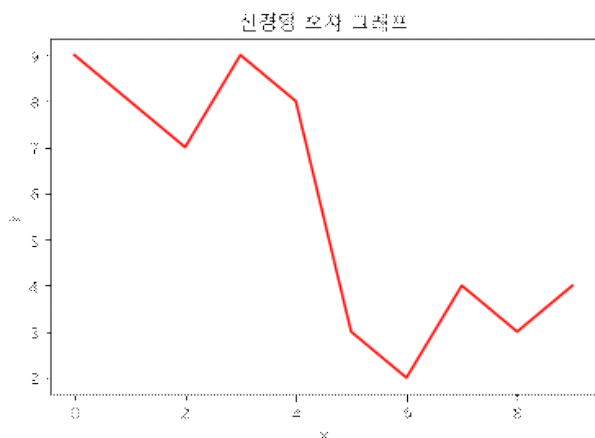
문제16. 문제15번의 제목을 붙이시오!

한글 폰트 지원 코드

파일경로에 폰트raw 파일이 있어야 한다

```
from matplotlib import font_manager, rc
font_path = "C:\\Users\\W82103\\OneDrive\\Fonts\\malgun.ttf" #폰트파일의 위치 실습파일안에있음
font_name = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font_name)
```

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array( [ 0,1,2,3,4,5,6,7,8,9])
y = np.array( [ 9,8,7,9,8,3,2,4,3,4])
plt.plot(x, y, color='red')
plt.xlabel("x")
plt.ylabel("y")
plt.title("신경망 오차 그래프") #한글폰트지원 코드 필요
plt.show()
```



1장 내용정리

1. 딥러닝은 무엇인가? (면접질문 빈도 많음)

Q1. 딥러닝이 무엇인가요?

A1. 여러 비선형 변환 기법의 조합을 통해 높은 수준의 추상화를 시도하는 기계학습 알고리즘의 집합

Q2. 추상화는 무엇인가요?

A.2 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 기법

딥러닝이 부각된 이유

- 1) 기존 인공신경망 모델의 문제점인 기울기 소실이 해결되었다. ---> google : 'gradient vanishing' 검색
- 2) 강력한 GPU 연산을 활용하여 하드웨어 연산속도를 높였다 ---> google colab 을 이용해서 무료로 사용가능
- 3) 빅데이터의 등장과 SNS 의 활용이 증가하여(ex. kakaotalk) 학습에 필요한 데이터 확보가 가능해짐

2. 신경망 구현시 numpy 는 왜 사용하는 것인가?

행렬연산을 빠르게 할 수 있는 기능이 numpy 에 내장되어 있다(C언어 기반).

신경망에 데이터를 입력하면 행렬 연산이 일어나고 맨 마지막에 확률이 출력된다.

---> 입력사진(얼룩말) --- node ---- node2 --- 출력값(얼룩말(70%), 재규어(30%)) --- 이 중 높은 확률을 결과로 출력
위와 같은 과정은 다 연산으로 이루어지는데 이 때 다차원 행렬 계산이 필요하며 numpy 를 사용하여 연산을 수행한다.

3. numpy 의 주요 기능 중 broadcast 는 무엇인가?

형상이 다른 배열끼리 계산을 스마트하게 할 수 있게 해주는 numpy 의 기능

4. 딥러닝 실험을 위해 시각화가 필요한 이유는 무엇인가?

신경망이 제대로 학습이 되고 있는지 확인하려면 학습과정을 시각화 해보아야 한다.

■2장 : 퍼셉트론

퍼셉트론이란?

인간의 뇌세포 하나를 컴퓨터로 흉내낸 것

1957년에 프랑크 로젠블라트가 퍼셉트론을 고안해 낸다. 사람의 뇌의 동작을 전기 스위치 on/off 로 흉내낼 수 있는 이론을 증명

퍼셉트론은 고등학교 생물 시간에 배운 3가지 용어로 개념을 설명할 수도 있다.

1. 자극(stimulus)
2. 반응(response)
3. 역치(threshod)

"특정 자극이 있다면 그 자극이 어느 역치 이상이어야 세포가 반응한다."

ex) 짜게 먹는 사람은 평소에 먹는 만큼의 염분이 포함되지 않은 음식에 대해 싱겁다고 생각하는데,
이는 역치 이하의 자극에는 반응하지 않는것을 의미한다.

단순한 논리회로(p.49)

1. AND gate
2. OR gate
3. NAND gate
4. XOR gate

문제17. 아래의 AND gate 를 위한 데이터 행렬을 numpy array 로 구현하시오!

x1	x2	y
----	----	---

0	0	0
1	0	0
0	1	0
1	1	1

```
import numpy as np
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)

y = np.array( [0,0,0,1] )
y2 = y.reshape(4,-1)

print(x2.shape) #(4,2)
print(y2.shape) #(4,1)
```

문제18. AND gate 퍼셉트론 함수를 생성하시오!

```
def AND(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1 *w1 + x2 *w2
    if tmp <= theta:
        return 0
    else:
        return 1

print(AND(0,0)) #0
print(AND(0,1)) #0
print(AND(1,0)) #0
print(AND(1,1)) #1
# print 를 하여 y행렬과 동일하게 출력되는 것을 확인
# AND function 내부에서 정의된 가중치들은 x2 행렬데이터에 대해 AND gate 로 동작한다.
```

문제19. OR 게이트 퍼셉트론 함수를 생성하시오!

```
def OR(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.3 #theta 값을 AND 게이트의 값에서 수정
    tmp = x1 *w1 + x2 *w2
    if tmp <= theta:
        return 0
    else:
        return 1

print(OR(0,0)) #0
print(OR(0,1)) #1
print(OR(1,0)) #1
print(OR(1,1)) #1
```

x2 행렬(=x행렬) 의 인자값을 하나씩 다 빼는 for 코드

```
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        print(x2[i][j])
```

문제20. AND gate 행렬에서 X 행렬에 1행 1열에 데이터를 가져오시오!

```
import numpy as np
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)
y = np.array( [0,0,0,1] )
y2 = y.reshape(4,-1)

print(x2[0][0]) #0
x2
```

문제21. 위에서 만든 퍼셉트론 함수 AND 가지고 입력값 X에 데이터를 입력받아 아래와 같이 결과를 출력하시오!

```
import numpy as np
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)
y = np.array( [0,0,0,1] )
y2 = y.reshape(4,-1)

def AND(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1 *w1 + x2 *w2
    if tmp <= theta:
        return 0
    else:
        return 1

bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2]
k2 = bin[1::2]

for i, j in zip(k1,k2):
    print(AND(i,j))
```

문제22. 아래의 표와 같은 NAND gate 를 만들고 출력하시오!

```
def NAND(x1, x2):
    w1, w2, theta = -0.5, -0.5, -0.7
    tmp = x1 *w1 + x2 *w2
    if tmp <= theta:
        return 0
    else:
        return 1

import numpy as np
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)
y = np.array( [0,0,0,1] )
```

```
y2 = y.reshape(4,-1)
```

```
bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2]
k2 = bin[1::2]

for i, j in zip(k1,k2):
    print(NAND(i,j))
```

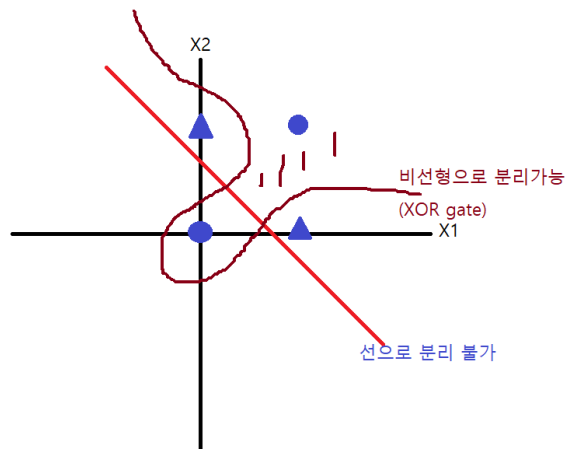
●XOR 게이트 p.54

XOR gate : exculsive OR gate : 배타적 OR 게이트

- 자기 자신 외에는 거부하는 gate
- 1957년 로젠블라트가 퍼셉트론을 제시했을 때에는 AND, OR, NAND 게이트만 있었다.
- 그러부터 2년 뒤인 1959년 민스키가 퍼셉트론의 XOR 게이트는 분류를 못하는 문제점을 지적하였다.
- AND, OR, NAND 는 단층 신경망으로 구현이 되는데 XOR 게이트는 단층 구현이 불가능하다(선형 불가)
- 위와 같은 문제점이 지적된 뒤 20년 후 다층 퍼셉트론으로 비선형 영역을 분류하였다.
- XOR 게이트의 그림 예시

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

x1과 x2가 둘 다 같은값이면 0, 다른값이면 1로 결과를 출력
자기 자신 외에는 거부하는 게이트로 이해



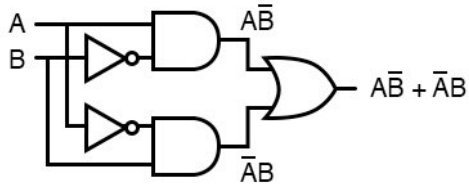
p.59 그림2-12 표

x1	x2	s1	s2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

입력층	입력층	은닉층	은닉층	출력층
0층	0층	1층	1층	2층
-	-	NAND(x1,x2)	OR(x1,x2)	AND(s1,s2)



... is equivalent to ...



$$A \oplus B = A\bar{B} + \bar{A}B$$

문제23. 그림2-11 에 나오는 그림으로 입력값 데이터를 받아서 XOR 게이트 결과가 출력되게 하시오! **

Full code(함수 처음부터 다 포함)

```
def AND(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1 * w1 + x2 * w2
    if tmp <= theta:
        return 0
    else:
        return 1

def OR(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.3
    tmp = x1 * w1 + x2 * w2
    if tmp <= theta:
        return 0
    else:
        return 1

def NAND(x1, x2):
    w1, w2, theta = -0.5, -0.5, -0.7
    tmp = x1 * w1 + x2 * w2
    if tmp <= theta:
        return 0
    else:
        return 1

def XOR(x1, x2):
    s1 = NAND(x1,x2)
    s2 = OR(x1,x2)
    y = AND(s1,s2)
    return y
```

```

## X행렬(x2)과 y행렬(label=y2) 생성
import numpy as np
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)
y = np.array( [0,1,1,0] )
y2 = y.reshape(4,-1)

# loop 를 사용하여 X행렬의 인자값들을 저장
bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2] #홀수번째 인자
k2 = bin[1::2] #짝수번째 인자

# XOR 게이트에 각 인자를 대입하여 결과값 출력
for i, j in zip(k1,k2):
    print(XOR(i,j) )

# label column과 비교
print('\n', y)

```

●가중치와 편향 구현하기 p.52

신경망에서의 파라미터 : 가중치(w), 편향(b)

신경망에서의 하이퍼 파라미터 : learning rate, decay(가중치 감소), 층수, 뉴런의 개수 등

가중치(weight) : 입력신호가 결과에 주는 영향력(중요도)를 조절하는 매개변수

편향(bias) : 뉴런이 얼마나 쉽게 활성화 하느냐를 조절하는 매개변수 (활성화=1)

ex) OR 게이트처럼 입력신호가 x1과 x2 값을 받는 경우에 편향(x0)이 없으면 target 을 분류하는 직선은 무조건 원점을 통과해야만 하기 때문에 제대로 분류할 수 없다.

문제24. 편향을 넣어서 AND 게이트 함수를 생성하시오!

```

import numpy as np

def AND(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5, 0.5]) #실제로는 이 w와 b를 컴퓨터(기계)가 학습하여 알아내야 한다.
    b = -0.7
    tmp = np.sum(x*w)+b
    if tmp <= 0:
        return 0
    else:
        return 1

```

```

## X행렬 생성
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)

# loop 를 사용하여 X행렬의 인자값들을 저장
bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2] #홀수번째 인자
k2 = bin[1::2] #짝수번째 인자

# AND 게이트에 각 인자를 대입하여 결과값 출력
for i, j in zip(k1,k2):
    print(AND(i,j) )

# label column과 비교
# print('Wn', y)

```

문제25. 편향을 넣어서 OR 게이트 함수를 생성하시오!

##편향(bias) 추가 OR gate

```

import numpy as np

def OR(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5, 0.5]) #실제로는 이 w와 b를 컴퓨터(기계)가 학습하여 알아내야 한다.
    b = -0.4
    tmp = np.sum(x*w)+b
    if tmp <= 0:
        return 0
    else:
        return 1

## X행렬 생성
x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)

# loop 를 사용하여 X행렬의 인자값들을 저장
bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2] #홀수번째 인자
k2 = bin[1::2] #짝수번째 인자

# OR 게이트에 각 인자를 대입하여 결과값 출력

```

```

for i, j in zip(k1,k2):
    print(OR(i,j) )

# label column과 비교
# print('Wn', y)

```

문제26. NAND 게이트 함수를 만들고 XOR 게이트 함수를 만들어서 아래의 결과가 출력되게 하시오!

```
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(x*w)+b
    if tmp <= 0:
        return 0
    else:
        return 1

```

```

def OR(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5, 0.5])
    b = -0.4
    tmp = np.sum(x*w)+b
    if tmp <= 0:
        return 0
    else:
        return 1

```

```

def NAND(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(x*w)+b
    if tmp <= 0:
        return 1
    else:
        return 0

```

```

def XOR(x1, x2):
    s1 = NAND(x1,x2)
    s2 = OR(x1,x2)
    y = AND(s1,s2)
    return y

```

```
## X행렬(x2)과 y행렬(label=y2) 생성
```



```

x = np.array( [0,0,0,1,1,0,1,1] )
x2 = x.reshape(4, -1)
y = np.array( [0,1,1,0] )
y2 = y.reshape(4,-1)

# loop 를 사용하여 X행렬의 인자값들을 저장
bin = []
for i in range(x2.shape[0]):
    for j in range(x2.shape[1]):
        bin.append(x2[i][j])
k1 = bin[::2] #홀수번째 인자
k2 = bin[1::2] #짝수번째 인자

# XOR 게이트에 각 인자를 대입하여 결과값 출력
for i, j in zip(k1,k2):
    print(XOR(i,j) )

# label column과 비교
print('\n', y)

0
1
1
0

[0 1 1 0]

```

```
#EndLine=====
```

#0305

2021년 2월 8일 월요일 오전 9:28

■ Review

- 1장의 내용은 신경망 내부에서 행렬 연산을 편하게 하기 위해 사용되는 모듈인 numpy모듈을 배웠다.
- 2장의 퍼셉트론 구현하기 위해서 함수 4개를 생성 (AND, OR, NAND, XOR)

1. 단층 신경망

입력층 -----> 출력층

0층 1층

구현 가능한 게이트 : AND, OR, NAND

2. 다층 신경망

입력층 -----> 은닉층-----> 출력층

구현 가능한 게이트 : XOR

●기계학습을 통해 가중치(w)를 알아내기

예제1. 아래의 두 행렬을 생성하시오!

```
import numpy as np
```

$$x = [-1, 0, 0, -1, 1, 0, -1, 0, 1, -1, 1, 1]$$
$$w = [0.3, 0.4, 0.1]$$

```
x2 = np.array(x)
```

```
w2 = np.array(w)
```

```
x3 = x2.reshape(4,-1)
```

```
w3 = w2.reshape(3)
```

```
print(x3.shape)  #(4,3)
```

```
print(w3.shape)  #(3,
```

```
print(x3.ndim) #2
```

```
print(x3.ndim) #2
```

#x3

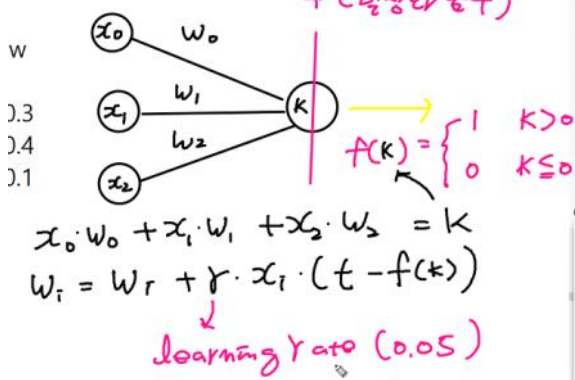
```
array([[ -1,  0,  0],
       [ -1,  1,  0],
       [ -1,  0,  1],
       [ -1,  1,  1]])
```

#w3

```
array([0.3, 0.4, 0.1])
```

예제2. 손으로 퍼셉트론을 구현

니다.



문제27. 어제 문제24번의 AND 게이트 함수를 가져와서 위에서 구한 가중치와 bias 를 대입해서 target(정답)을 잘 예측하는지 테스트 하시오!

```
import numpy as np
```

```
x = np.array([-1,0,0,-1,1,0,-1,0,1,-1,1,1])
```

```
w = np.array([0.35,0.35,0.1])
```

```
x = x.reshape(4,-1) #x0(intercept), x1,x2 : X
```

```
w = w.reshape(3) #w0(bias), w1,w2(weight)
```

```
t = np.array([0,0,0,1]) #Target = label
```

```
print(x)
```

```
print(w)
```

```
print(t)
```

#출력결과(캡처그림)

```
[[-1  0  0]
 [-1  1  0]
 [-1  0  1]
 [-1  1  1]]
```

```
[0.35 0.35 0.1 ]
```

```
[0 0 0 1]
```

↓↓↓↓↓

```
def AND(x1, x2):
```

```
    x = np.array([x1,x2])
```

```
    w = np.array([0.35, 0.1])
```

```
    b = -0.35 #b=w0*x0
```

```
    tmp = np.sum(x*w)+b
```

```
    if tmp <= 0:
```

```
        return 0
```

```
    else:
```

```
        return 1
```

X행렬(x2)과 y행렬(label=y2) 생성

```
x = np.array( [0,0,0,1,1,0,1,1] )
```

```
x2 = x.reshape(4, -1)
```

```
y = np.array( [0,0,0,1] )
```

```
y2 = y.reshape(4,-1)
```

```
# loop 를 사용하여 x행렬의 인자값들을 저장
```

```
bin = []
```

```
for i in range(x2.shape[0]):
```

```
    for j in range(x2.shape[1]):
```

```
        bin.append(x2[i][j])
```

```
k1 = bin[::2] #홀수번째 인자
```

```
k2 = bin[1::2] #짝수번째 인자
```

```
# XOR 게이트에 각 인자를 대입하여 결과값 출력
```

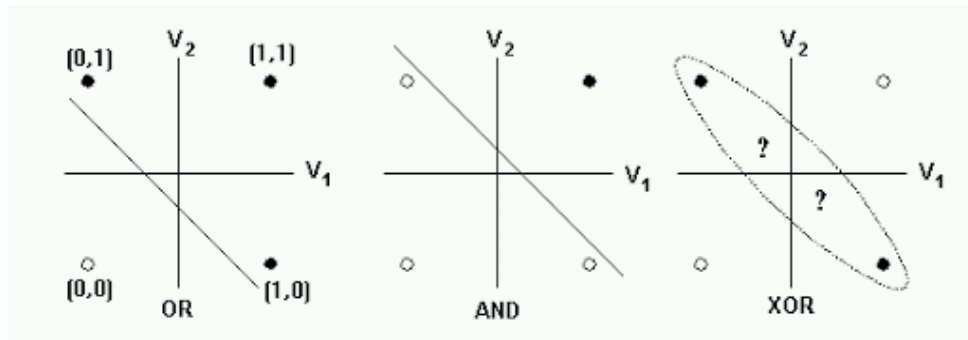
```
for i, j in zip(k1,k2):
```

```
    print(AND(i,j) )
```

```
# label column과 비교
```

```
print('\n', y)
```

and 게이트 데이터는 어떻게 분류되는 것인가?



[code]

```
import numpy as np
```

```
x = np.array([-1,0,0,-1,1,0,-1,0,1,-1,1,1])
```

```
w = np.array([0.3,0.4,0.1])
```

```
x = x.reshape(4,-1) #x0(intercept), x1,x2 : X
```

```
w = w.reshape(3) #w0(bias), w1,w2(weight)
```

```
t = np.array([0,0,0,1]) #Target = label
```

```
print(x)
```

```
print("")
```

```
print(w)
```

```
np.dot(x,w) #x0*w0 + X1*W1 + X2*W2
```

```
#result
```

```
[[ -1  0  0]
 [ -1  1  0]
 [ -1  0  1]
 [ -1  1  1]]
```

```
[0.3 0.4 0.1]
```

```
array([-0.3,  0.1, -0.2,  0.2])
```

예제1~3 입력값과 가중치의 곱의 합을 구현하는 predict 함수를 만드시오!

선생님 답

```
def predict(x, w):  
    a = np.sum( x*w.T)  
    return a  
  
for i in range( len(x) ):  
    print ( predict(x[i], w) )
```

#a = pred 이라고 생각

#a : 뉴런(활성함수에 들어가기 전)

#target : t : 실제값(y=label)

(현재 a 값은 $x_0*w_0 + x_1*w_1 + x_2*w_2$ 의 값이므로 정확하게는 예측값이라고 할 수 없음)

import numpy as np

x = np.array([-1,0,0,-1,1,0,-1,0,1,-1,1,1])

w = np.array([0.3,0.4,0.1])

x = x.reshape(4,-1) #x0(intercept), x1,x2 : X

w = w.reshape(3) #w0(bias), w1,w2(weight)

t = np.array([0,0,0,1]) #Target = label

a = np.dot(x,w) # $x_0*w_0 + X_1*W_1 + X_2*W_2$

print(x, end='\n\n')

print(w, end='\n\n')

print('a : ',a)

print('target : ',t)

예제4. 입력되는 값이 0보다 작거나 같으면 0을 출력하고 else: return 1 인 step_function 함수를 생성하시오!

step_function 은 위 예제의 a 뉴런값을 인자로 받는 활성함수로 이해할 수 있다. (ex. step, sigmoid, Relu ..etc)

```
def step_function(a):  
    if a > 0:  
        return 1  
    else:  
        return 0  
  
print( step_function(2) ) #1  
print( step_function(0) ) #0  
print( step_function(-0.2) ) #0
```

예제5. 위에서 출력한 a 값을 step_function 에 넣고 값을 출력하시오!

... 개인 코드 작성

☆문제28. 위의 코드를 수정해서 마지막에 갱신된 w 값만 출력되게 하시오!

```

## Import module
import numpy as np

## Functions
def step_function(a):
    if a > 0:
        return 1
    else:
        return 0

def predict(x, w):
    a = np.dot(x, w)

    pred = []
    for i in range(a.shape[0]):
        pred.append(step_function(a[i]))

    pred_a = np.array(pred)
    return(pred_a)

def renew_w(x, w, t):
    lr = 0.05 #learning rate
    weight = w

    for i in range(x.shape[0]):
        cost = t[i] - predict(x,w)[i]
        if cost != 0:
            weight = weight.T + np.array(lr*cost*x[i])
            weight = weight.T

    return(weight)

##Dataset
x = np.array([-1,0,0,-1,1,0,-1,0,1,-1,1,1])
w = np.array([0.3,0.4,0.1])

x = x.reshape(4,-1) #x0(intercept), x1,x2 : X
w = w.reshape(3,-1) #w0(bias), w1,w2(weight)
t = np.array([0,0,0,1]) #Target = label

### 이부분에서 cut 후 아래코드를 실행하여 가중치 갱신과정 확인
##Renew weight (loop x1)
print(w)
print(w.shape, end='\n\n')

```

```
w = renew_w(x,w,t)
print(w)
print(w.shape) #(1,3)
```

문제31. 문제29,30 포함) 위 코드를 개선하시오!

```
## Import module
import numpy as np

## Functions
def step_function(a):
    if a > 0:
        return 1
    else:
        return 0

def predict(x, w):
    a = np.dot(x, w)

    pred = []
    for i in range(a.shape[0]):
        pred.append(step_function(a[i]))

    pred_a = np.array(pred)
    return(pred_a)

def renew_w(x, w, t):
    lr = 0.05 #learning rate
    weight = w
    cnt2 = 0

    while True:
        cnt = 0
        cnt2 += 1
        for i in range(x.shape[0]):
            cost = t[i] - predict(x,w)[i]
            cnt += cost

        if cost != 0:
            weight = weight.T + np.array(lr*cost*x[i])
            weight = weight.T
            w = weight
        else:
            continue

    print(f"index : {cnt2}", weight.T)
```

```

        if cnt == 0:
            break

    return(weight)

##Dataset
x = np.array([-1,0,0,-1,1,0,-1,0,1,-1,1,1])
w = np.array([0.7,0.4,0.1])

x = x.reshape(4,-1) #x0(intercept), x1,x2 : X
w = w.reshape(3,-1) #w0(bias), w1,w2(weight)
t = np.array([0,0,0,1]) #Target = label

###
%%
##Renew weight (loop x1)
print('raw w :',w.T) #보기 편하도록 전치행렬로 출력
print('shape :',w.shape, end='\n\n')

w = renew_w(x,w,t)
print('\n\n')

print('renew w :',w.T) #보기 편하도록 전치행렬로 출력
print('shape :',w.shape) #(1,3)

raw w : [[0.7 0.4 0.1]]
shape : (3, 1)

index : 1 [[0.65 0.45 0.15]]
index : 2 [[0.6 0.5 0.2]]
index : 3 [[0.6 0.5 0.2]]

renew w : [[0.6 0.5 0.2]]
shape : (3, 1)

```

2장 요약

1. 퍼셉트론 : 인간의 뇌에 있는 뉴런 세포를 컴퓨터로 흉내낸 것
2. 퍼셉트론의 종류
 - 1) 단층 퍼셉트론 : and, or, nand gate
 - 2) 다층 퍼셉트론 : xor gate
3. 인공신경망에 최종적으로 산출해야할 값 : parameters(bias, weight)

■ 3장. 신경망

저자가 만들어온 가중치를 세팅해서 필기체 데이터를 인식하는 3층신경망 생성
(저자가 학습을 해서 만들어온 가중치를 사용)
날코딩을 하는것이 주 내용

신경망안에 들어가는 함수 소개

1. 활성화 함수 (ex. step, sigmoid, Relu)
2. 출력층 함수 (ex. 항등함수(회귀), softmax(분류))
3. 오차함수 (function)

날코딩으로 신경망의 기초를 다진 뒤 학습내용

tensorflow, pytorch 를 사용해서 신경망 구현

□활성함수

계단함수

"숫자 0과 1을 리턴하는 함수"

입력값 $x \leq 0$ ---> return 0

입력값 $x > 0$ ---> return 1

예제1

```
import numpy as np
```

```
def step_function(x):
```

```
    if x > 0:
```

```
        return 1
```

```
    else:
```

```
        return 0
```

```
print(step_function(3.0)) #1
```

```
print(step_function(0.0)) #0
```

```
print(step_function(-2.0)) #0
```

```
# 일반 실수형 데이터는 잘 적용된다
```

그런데 신경망에는 ndarray 형태의 데이터가 필요하기 때문에 ndarray를 생성하여 함수에 적용하면 err 발생한다.

```
x_data = np.array([-1,0,1])
```

```
print(step_function(x_data))
```

```
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

기존에 정의한 step_function() 은 수정이 필요하다.

예제2. numpy 배열을 사용할 수 있는 함수로 수정

```
def step_function(x):
```

```
    y = x>0
```

```
    print(y)
```

```
x_data = np.array([-1,0,1])
```

```
step_function(x_data)  #[False False  True]
```

↓ ↓ ↓ 위 코드의 컨셉을 이용해서 수정

```
import numpy as np
```

```
def step_function(x):
    y = x>0
    return y.astype(np.int)
```

```
x_data = np.array([-1,0,1])
print(step_function(x_data)) #[0,0,1] #numpy.ndarray #(3, )
```

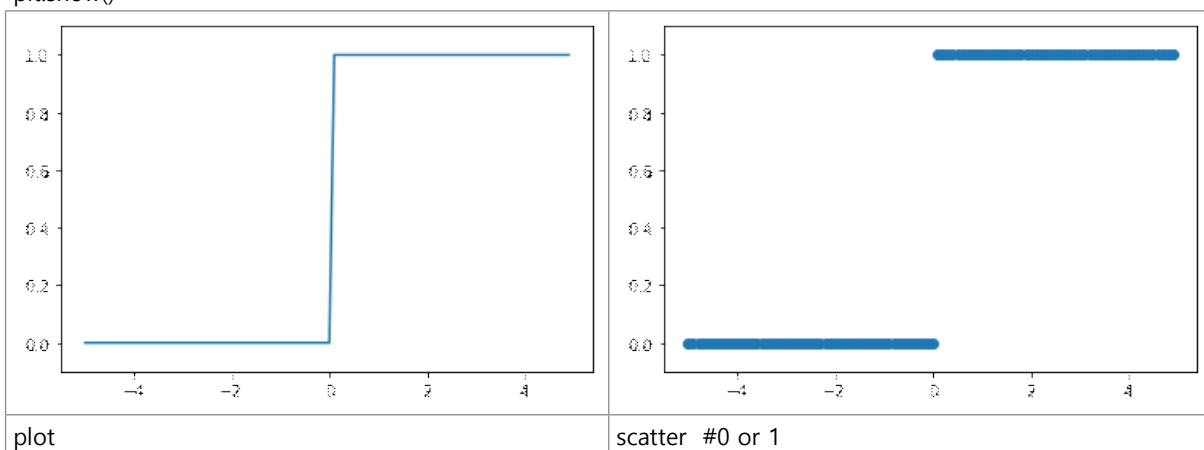
#신경망에서 흘러가는 모든 데이터는 numpy array 의 형태의 다차원 데이터 ---> 함수도 그에 맞게 정의해야 함

문제34. 위에서 만든 step_function 을 이용해서 p.71 그림3-6 을 구현하시오!

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def step_function(x):
    y = x>0
    return y.astype(np.int)
```

```
x = np.arange(-5.0, 5.0, 0.1)
y = step_function(x)
plt.plot(x,y)
plt.ylim(-0.1, 1.1)
plt.show()
```



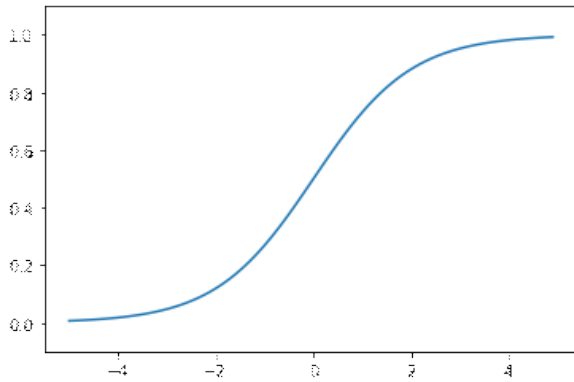
Sigmoid function

문제35. 시그모이드 함수를 파이썬으로 구현하시오! p.72

```
def sigmoid(x):
    return 1 / (1+np.exp(-x))
x = np.array([-1.0, 1.0, 2.0])
sigmoid(x) #array([0.26894142, 0.73105858, 0.88079708])
```

문제36. 시그모이드 함수 그래프를 그리시오!

```
x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)
plt.plot(x,y)
plt.ylim(-0.1, 1.1)
plt.show()
```



시그모이드 함수의 유래

<https://cafe.daum.net/oracleoracle/Sedp/195>

오즈함수
→
로짓함수
→
시그모이드 함수

성공
실패 = $\frac{p}{1-p}$

$\log(\text{오즈함수}) = \log\left(\frac{p}{1-p}\right)$

$\ln(\text{오즈함수}) = \ln\left(\frac{p}{1-p}\right)$

$\ln\left(\frac{p}{1-p}\right) = w \cdot x + b$

$\ln\left(\frac{p}{1-p}\right) = w \cdot x + b$

$e^{\ln\left(\frac{p}{1-p}\right)} = e^{w \cdot x + b}$

$\left(\frac{p}{1-p}\right)^{\log e^e} = e^{w \cdot x + b}$

$\frac{p}{1-p} = e^{w \cdot x + b}$

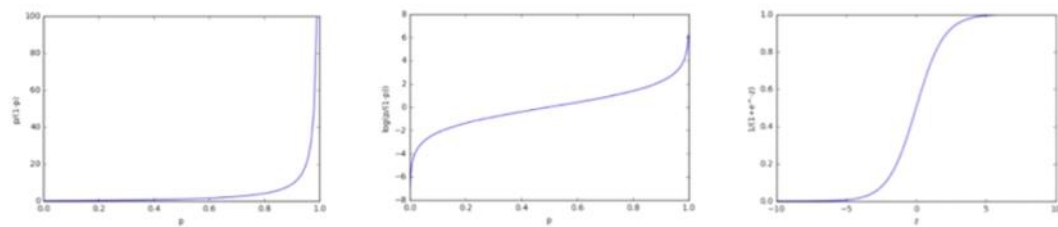
$\frac{1-p}{p} = e^{-(w \cdot x + b)}$

$\frac{1}{p} - 1 = e^{-(w \cdot x + b)}$

$\frac{1}{p} = 1 + e^{-(w \cdot x + b)}$

$P = \frac{1}{1 + e^{-(w \cdot x + b)}} \quad \# w \cdot x + b = k$

$f(k) = \frac{1}{1 + e^{-k}}$



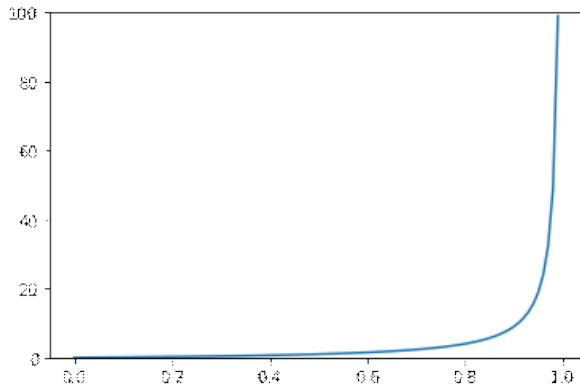
오즈비율함수	로짓함수	로짓함수
실패할 확률대비 성공확률	오즈비율 함수에 로그사용	로짓함수를 신경망에서 p(확률)값을 계산하기 편하도록 지수형태로 바꾼 함수
$y = p / (1-p)$	$y = \log(p / (1-p))$	$y = 1 / (1+\exp(-x))$
		신경망에서 $x = f(x)$

desmos 의 sigmoid set 참고

37. 오즈비율 함수 그래프를 그리시오!

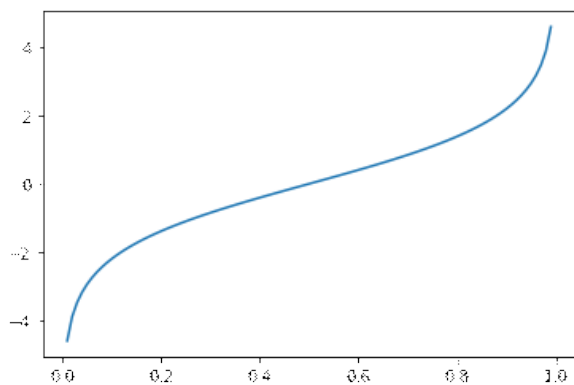
```
def oz(x):
    return x / (1-x)
x = np.arange(0.0, 1.0, 0.01)
y = oz(x)
```

```
plt.plot(x,y)
plt.ylim(0, 100)
plt.show()
```



38. 로짓함수도 그래프로 시각화하시오!

```
def logit(x):
    return np.log( x / (1-x) )
x = np.arange(0.01, 1, 0.01)
y = logit(x)
plt.plot(x,y)
plt.show()
```



■3월 7일 밤 12까지 제출을 편하게 할 수 있는 90% 정도 완성된코드 : kaggle

개선방법

1. 알고리즘 변경 : Xgboost, Adaboost
2. 결측치 치환 ---> SQL ex) 이름의 호칭의 평균값으로 결측치를 채움**
3. 운임의 이상치를 제거
4. 나이의 결측치를 치환하는 방법
 - 1) mean
 - 2) mod
 - 3) regression ** : 나이를 종속변수, 나머지를 X로 두고 age 값을 추정 후 대입(결측치에)

#EndLine=====

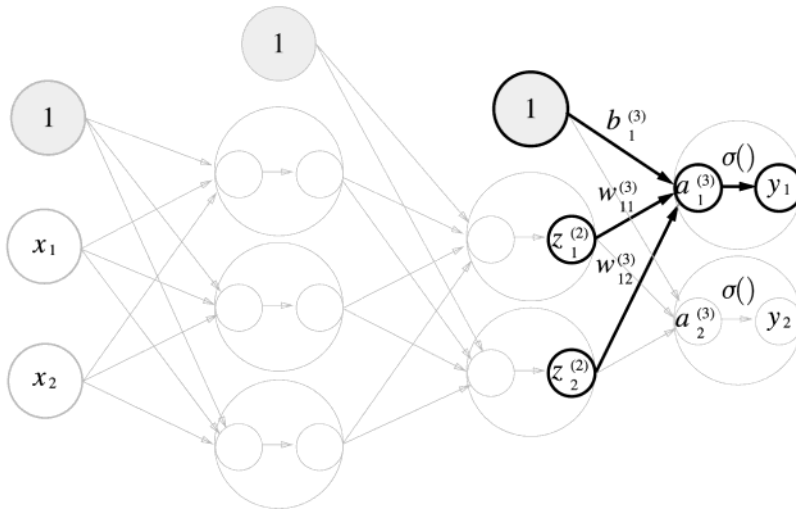
#0308

2021년 2월 8일 월요일 오전 9:28

■Review

- 1장. numpy 사용법 ---> 신경망의 학습 시 사용되는 행렬 연산에 최적화된 모듈
- 2장. 퍼셉트론 ---> 파이썬으로 퍼셉트론을 구현. 가중치가 어떻게 만들어지는지 이해
- 3장. 3층 신경망 구성(필기체 데이터 6만장 학습시킨 가중치를 저자가 기입, 그 가중치를 세팅하여 필기체 인식)

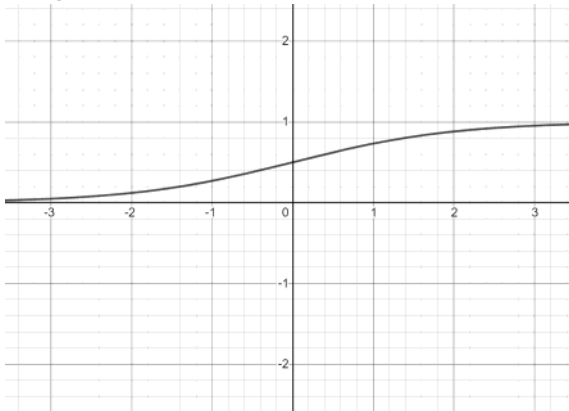
[그림 3-20]



활성화함수

활성화함수 : 신경망에 들어오는 데이터를 신경망에서 신호로 바꾸어 계속 흘러보낼지 말지 결정하는 함수

1. **sigmoid** : 'S' 자 모양이라는 의미를 갖고 있음



#1 / 1+exp(-x) #desmos

[code]

```
import numpy as np
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

시그모이드 함수의 단점

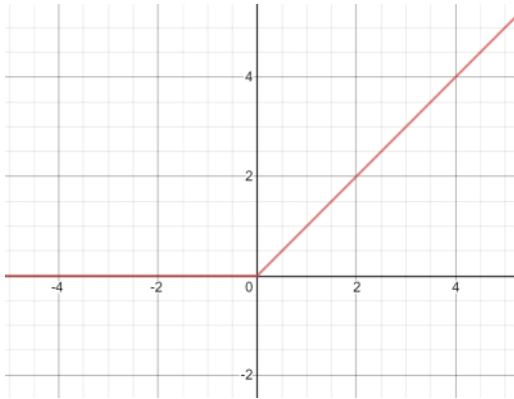
gradient vanishing : x값이 0에서 멀어질수록 기울기(미분값)가 점점 작아진다.

신경망이 깊어질수록 이 기울기 소실 문제는 더 확대되는 경향이 있다.

2. ReLU (Rectified Linear Unit)

rectified(정류된)는 전기회로 용어로, +/- 가 반복되는 교류에서 - 흐름을 차단하는 회로(반파정류회로)

ReLU 함수는 0 이하에서는 값을 흘려보내지 않고 양수에서만 값을 그대로 흘려보내는 함수이다.



예제1. Relu 함수를 생성하시오!

```
import numpy as np
def relu(x):
    return np.maximum(0, x) #0과 x중 큰 값을 출력
x = [3,1.4,0,-0.7,-2]
for i in x:
    print(relu(i))
3
1.4
0
0.0
0

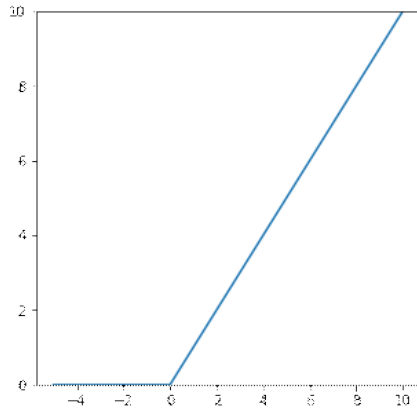
3
1.4
0
0.0
0
```

문제39. relu 함수를 그래프로 시각화하시오!

```
import numpy as np
import matplotlib.pyplot as plt

def relu(x):
    return np.maximum(0, x) #0과 x중 큰 값을 출력

x = np.arange(-5.0, 10.1, 0.1)
y = relu(x)
plt.figure(figsize=(5,5))
plt.plot(x,y)
plt.ylim(0, 10)
plt.show()
```

● 다차원 배열 계산 p.77

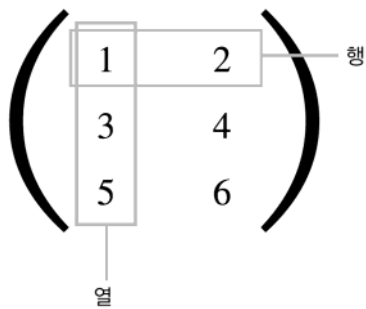
다차원 배열도 그 기본은 '숫자의 집합' 이다.

숫자를 한 줄로 늘어선 것이나 정사각형으로 늘어 놓은것이나 3차원으로 늘어놓은 것 모두 차원 배열이라고 한다.

예제1. 1차원 배열 생성

```
import numpy as np
a = np.array( [1,2,3,4] )
print(np.ndim(a) ) #1 #1차원 의미
```

문제40. 아래의 3 by 2 행렬을 만들고 차원을 확인하시오!



```
a = [1,2,3,4,5,6]
a2 = np.array(a)
a3 = a2.reshape(3,2)
print(np.ndim(a3) ) #2
↓:아래의 코드로 간략하게 작성가능
a = [1,2,3,4,5,6] #a=list(range(1,7))
a2 = np.array(a).reshape(3,2)
a2
Out[22]: array([[1, 2],
               [3, 4],
               [5, 6]])
```

문제41. 3차원 배열을 만들어 보시오!

```
import numpy as np
b = np.array([ [1,2],[3,4]], [[5,6],[7,8]] )
print(b)
print(np.ndim(b))
print(b.shape)
```

```
[[[1 2]
   [3 4]]

  [[5 6]
   [7 8]]]
3
(2, 2, 2)
```

문제42. 아래의 2차원 배열에서 숫자 3을 출력하시오!

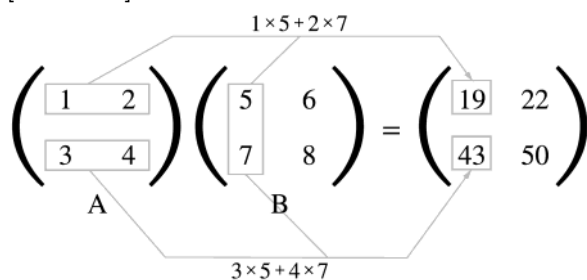
```
import numpy as np
a = np.array([1,2,3,4,5,6])
a2 = a.reshape(3,2)
a2
Out[31]: array([[1, 2],
                [3, 4],
                [5, 6]])
a2[1,0] #3
```

문제43. 아래의 3차원 배열에서 숫자 5을 출력하시오!

```
import numpy as np
b = np.array([ [1,2],[3,4], [5,6],[7,8] ])
b[1,0,0]
```

● 행렬의 내적(행렬곱) p.79

[그림 3-11]



위 그림의 내적을 구현(3가지 방법)

```
1. np.dot : np.array
import numpy as np
x = np.array( [[1,2], [3,4]] )
y = np.array( [[5,6], [7,8]] )
print(np.dot(x,y))
```

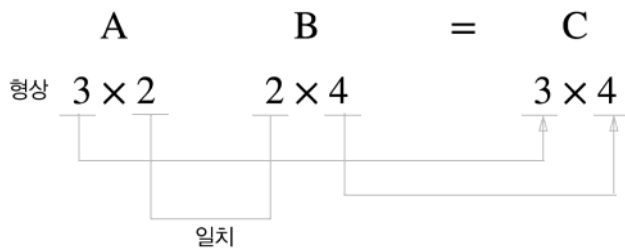
```
2. x*y : np.matrix
x = np.matrix( [[1,2], [3,4]] )
y = np.matrix( [[5,6], [7,8]] )
print(x*y)
```

```
3. x@y
x = np.array( [[1,2], [3,4]] )
y = np.array( [[5,6], [7,8]] )
print(x@y)
```

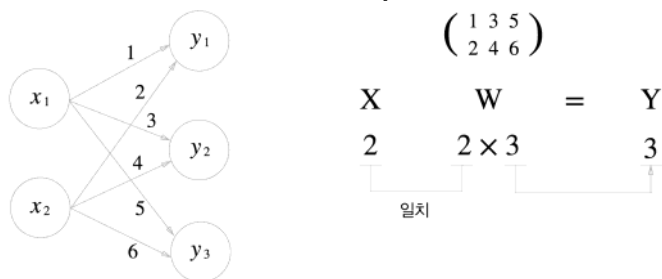
array 와 matrix 의 기능 차이

array 는 다차원으로 나타낼 수 있는데 matrix 는 2차원만 가능
 신경망은 다차원을 다루기 때문에 보통 array 를 이용하여 행렬을 생성한다

행렬의 내적 할때는 내적의 맞닿은 면의 차원을 일치시켜야 한다

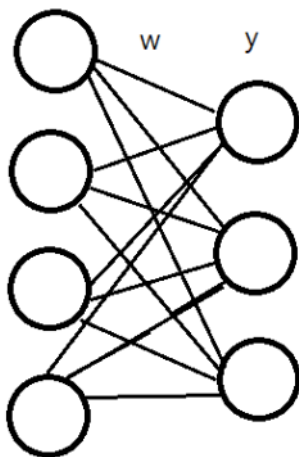


내적이 필요한 이유 : 신경망 내적 p.82



문제44. 아래 신경망의 출력 y 값을 구하시오!

4,5,7,2: X



```
import numpy as np
x = np.array( [4,5,7,2] )
w = np.array( [8,21,1,4,5,9,6,34,4,12,2,5] )
```

```
x2 = x.reshape(4)
w2 = w.reshape(4,-1)
```

```
print(x2.shape) #(4,)
print(w2.shape) #(4,3)
```

```
y = np.dot(x2,w2)
```

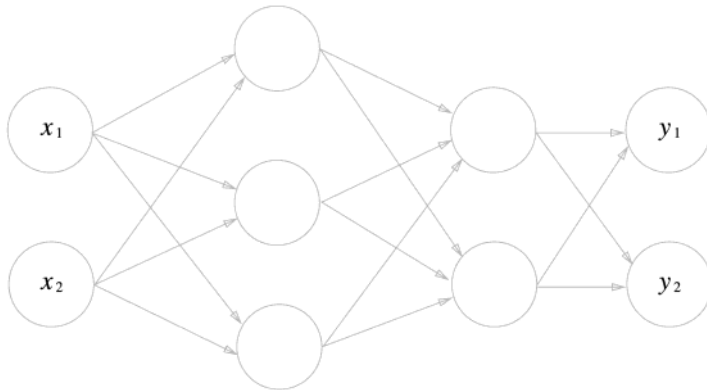
y

```
(4,)
(4, 3)
```

```
Out[61]: array([118, 351, 87])
```

● 3층 신경망 구현하기 p.83

[그림 3-15]



예제1. 입력층 ---> 은닉 1층

```
import numpy as np
x = np.array( [1,2] )
w1 = np.array( [[1,3,5], [2,4,6]] )
y = np.dot(x, w1)
print(y) #[5,11,17]
```

1.2 #sigmoid #활성함수 통과시키기

```
import numpy as np
def sigmoid(x):
    return 1/(1+np.exp(-x) )

y_hat = sigmoid(y)
print(y_hat) #[0.99330715 0.9999833 0.99999996]
```

예제2. 입력층 ---> 은닉1층 ---> 은닉2층

```
import numpy as np

def sigmoid(x):
    return 1/(1+np.exp(-x) )

x = np.array( [1,2] ) #(2,)
w1 = np.array( [[1,3,5], [2,4,6]] ) #(2,3)
w2 = np.array( [[3,4],[5,6],[7,8]] ) #(3,2)

a = np.dot(x, w1)
a_sig = sigmoid(a)

a2 = np.dot(a_sig, w2)
a2_sig= sigmoid(a2)
a2_sig #array([0.99999969, 0.99999998]) #(1,2)
```

☆문제45. 위의 신경망을 출력층까지 구현하시오! (3층(출력층) 까지 구현!)

```
import numpy as np

def sigmoid(x):
    return 1/(1+np.exp(-x) )

x = np.array( [1,2] ) #(2,)
w1 = np.array( [[1,3,5], [2,4,6]] ) #(2,3)
```

```
w2 = np.array( [[3,4],[5,6],[7,8]] ) #(3,2)
w3 = np.array( [[4,5],[6,7]] ) #(2,2)
```

```
a = np.dot(x, w1)
a_sig = sigmoid(a) #(3, )
```

```
a2 = np.dot(a_sig, w2)
a2_sig= sigmoid(a2) #(2, )
```

```
a3 = np.dot(a2_sig, w3)
k = sigmoid(a3) #(2, )
```

```
k #array([0.9999546 , 0.99999386]) #k값은 아직 출력함수 미적용 상태
```

● 출력층 함수 p.90

"그동안 흘러왔던 확률들의 숫자를 취합해서 결론을 내주어야 하는 함수"

신경망으로 구현시..

1. 회귀 : 항등함수 ---> 입력값을 받아 그대로 출력

ex) concrete.csv : 분류를 하지 않고 회귀를 통해 수치를 예측(콘크리트의 강도)

2. 분류 : softmax function : p.91 식[3.10]

입력값을 받아서 확률벡터로 출력하는 함수

ex) 정상 폐사진 vs 폐결절 사진

softmax function

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

위 식을 파이썬으로 그대로 만들면 err 발생

지수함수는 쉽게 아주 큰 값을 출력하는데, 컴퓨터가 큰 값을 출력하게 되면 overflow 가 출력되면서 err 발생

overflow : 매우 큰 값을 pc 에서 출력할 때 나타나는 불안정한 결과값

$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1}^n \exp(a_i)} \\ &= \frac{\exp(a_k + \log C)}{\sum_{i=1}^n \exp(a_i + \log C)} \\ &= \frac{\exp(a_k + C')}{\sum_{i=1}^n \exp(a_i + C')} \end{aligned}$$

softmax 함수는 확률출력함수이다

softmax 함수의 모든 인자값을 다 더하면 1이 발생한다.

softmax func 자연상수의 지수함수는 아주 큰 값을 출력한다

exp(10) 은 2만이 넘고, exp(100) 은 40자리 이상, exp(1000) 은 inf(무한대) 로 출력되어 계산이 힘들다

ex)

```
import numpy as np
print( np.exp(10) )
print( np.exp(100) )
print( np.exp(500) )
print( np.exp(1000) )
```

```
[output]
22026.465794806718
2.6881171418161356e+43
1.4035922178528375e+217
inf
```

ex2)

```
import numpy as np
a = np.array([1010, 1000, 990])
print(np.exp(a)) #[inf inf inf]
```

ex3) solution

```
import numpy as np
a = np.array([1010, 1000, 990])
C = np.max(a) # 리스트 내부의 차를 반환
minus = a-C # minus 변수 지정

print((a-C)) #[ 0 -10 -20]
print(np.exp(a)) #[inf inf inf]
print(np.exp(minus)) #[1.00000000e+00 4.53999298e-05 2.06115362e-09]
```

위 코드를 활용하여 softmax function 만들기

[1-분자 구현]

```
import numpy as np
a = np.array([1010, 1000, 990])
```

```
def softmax(a):
    C = np.max(a)
    minus = a-C
    np_exp = np.exp(minus)
    return np_exp
```

[softmax function]

```
import numpy as np
```

```
def softmax(a):
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
a = np.array([1010, 1000, 990])
print(softmax(a)) #[9.99954600e-01 4.53978686e-05 2.06106005e-09]
```

```
print(sum(softmax(a))) #1 #probability
```

문제47. 아래의 raw 요소 3개 중 어떤것이 가장 큰 값인지 인덱스 번호로 출력하시오!

```
raw = [9.99954600e-01 4.53978686e-05 2.06106005e-09]
```

```
print(np.argmax(softmax(a))) #0 #numpy 의 리스트 요소 중 최대값의 index 를 출력
```

문제48. 방금 만든 softmax fuction 을 점심시간에 만든 3층 신경망 출력함수로 적용하여 3층 신경망 구현을 완료하시오!

```
## Import modules
```

```
import numpy as np
```

```
## Function
```

```
def sigmoid(x):
```

```
    return 1/(1+np.exp(-x) )
```

```
def softmax(a):
```

```
    C = np.max(a)
```

```
    minus = a-C
```

```
    exp_a = np.exp(minus)
```

```
    sum_exp_a = np.sum(exp_a)
```

```
    y = exp_a / sum_exp_a
```

```
    return y
```

```
## 3rd Nerual_net
```

```
# Dataset
```

```
x = np.array( [1,2] ) #(2,)
```

```
w1 = np.array( [[1,3,5], [2,4,6]] ) #(2,3)
```

```
w2 = np.array( [[3,4],[5,6],[7,8]] ) #(3,2)
```

```
w3 = np.array( [[4,5],[6,7]] ) #(2,2)
```

```
# stack
```

```
a = np.dot(x, w1)
```

```
a_sig = sigmoid(a) #(3, )
```

```
a2 = np.dot(a_sig, w2)
```

```
a2_sig= sigmoid(a2) #(2, )
```

```
a3 = np.dot(a2_sig, w3)
```

```
k = sigmoid(a3) #(2, ) #array([0.9999546 , 0.99999386])
```

```
y_hat = softmax(k)
```

```
y_hat #array([0.49999019, 0.50000981])
```

문제49. 위 3층 신경망 코드에서 w1, w2, w3 가중치를 하나로 모아서 심플한 코드로 작성하시오!

파이썬의 자료형 : 문자형, 숫자형, 리스트형, 딕셔너리형, 튜플

딕셔너리 형 구조로 생성 : 딕셔너리 : 키와 값으로 구성되어 있음

```

import numpy as np
network = {}
network['W1'] = np.array([[1,3,5],[2,4,6]])
network['W2'] = np.array([[3,4],[5,6],[7,8]])
network['W3'] = np.array([[4,5],[6,7]])

print( network['W1'] )
[[1 3 5]
 [2 4 6]]

```

```

network
Out[86]: {'W1': array([[1, 3, 5],
                      [2, 4, 6]]),
          'W2': array([[3, 4],
                      [5, 6],
                      [7, 8]]),
          'W3': array([[4, 5],
                      [6, 7]])}

```

[code]

```

## Import modules
import numpy as np

```

```

## Function
def sigmoid(x):
    return 1/(1+np.exp(-x) )

```

```

def softmax(a):
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y

```

```

## 3rd Nerual_net

```

```

#weight
def init_network():
    network = {}
    network['W1'] = np.array([[1,3,5],[2,4,6]])
    network['W2'] = np.array([[3,4],[5,6],[7,8]])
    network['W3'] = np.array([[4,5],[6,7]])
    return network

```

```

network = init_network()
W1,W2,W3 = network['W1'], network['W2'], network['W3']
x = np.array( [1,2] ) #(2,)

```

```

# stack
a = np.dot(x, W1)
a_sig = sigmoid(a) #(3, )

```



```

a2 = np.dot(a_sig, W2)
a2_sig= sigmoid(a2) #(2, )

a3 = np.dot(a2_sig, W3)
k = sigmoid(a3) #(2, ) #array([0.9999546 , 0.99999386])

y_hat = softmax(k)
y_hat #array([0.49999019, 0.50000981])

```

문제00. 위 **function** 들을 **jupyter notebook** 워킹디렉토리에 **common.py** 로 저장하고 함수를 호출해서 코드를 재작성하시오!

```

import numpy as np
from common import init_network, sigmoid, softmax

network = init_network()

```

● 필기체 데이터를 신경망에 로드 p.96

필요파일

1. 필기체 데이터(dataset.zip)
2. 저자가 이미 만들어 놓은 신경망 코드

mnist 데이터를 숫자 0~9까지의 숫자 이미지로 구성되어 있고 훈련데이터는 6만개, 테스트데이터는 1만개로 준비

[그림 3-24]



mnist 이미지 데이터를 28x28 크기의 회색조 이미지(1채널)이며, 각 픽셀은 0~255 값을 취한다.

● mnist 데이터를 파이썬으로 로드하기

1. 저자가 제공하고 있는 dataset 폴더를 주피터의 워킹디렉토리로 복사 #0308 폴더
2. 주피터 노트북에서 아래 code 입력

[code]

```

import sys, os
sys.path.append(os.pardir)
from dataset.mnist import load_mnist

```

```

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=False)
# normalize : 이미지의 픽셀값을 0.0 ~ 1.0 사이로 정규화할지여부를 결정 #단위를 일치시켜줌
# flatten : 입력이미지를 1차원 배열로 생성할지를 정한다. (원래는 2차원데이터)

```

```

print(x_train.shape) #(60000, 784)

```

문제51. 테스트 데이터는 몇장이 있는지 확인하시오!

```
print(x_test.shape) #(10000, 784)
```

문제52. 훈련데이터의 첫 번째 필기체의 숫자가 무엇인지 정답을 출력해서 알아내시오!

```
print(t_train[0]) #5
```

문제53. 훈련 데이터의 첫 번째 필기체 데이터를 2차원으로 시각화하시오!

```
anconda prompt> pip install Pillow
```

```
Python>
```

```
from PIL import Image
```

```
import numpy as np
```

```
def img_show(img):
```

```
    pil_img = Image.fromarray(np.uint8(img))
```

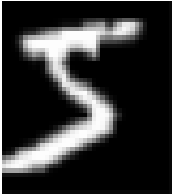
```
    pil_img.show()
```

```
img = x_train[0] #데이터 정의
```

```
print(img.shape) #(784,)
```

```
img = img.reshape(28,28) #재배열
```

```
img_show(img)
```



문제54. mnist 데이터를 flatten 시키지 말고 출력하시오!

```
import sys, os
```

```
sys.path.append(os.pardir)
```

```
from dataset.mnist import load_mnist
```

```
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False, normalize=False)
```

```
print(x_train.shape) #(60000, 1, 28, 28)
```

```
# 60000 : 전체 장 수 : rows
```

```
# 1 : 색조
```

```
# 28 : 가로
```

```
# 28 : 세로
```

문제55. irin 사진을 파이썬에서 시각화하시오!

```
from PIL import Image
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def img_show(img):
```

```

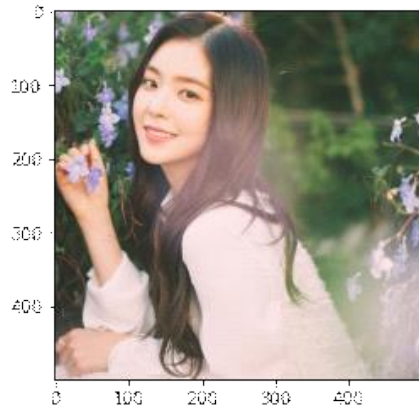
pil_img = Image.fromarray(np.uint8(img))
pil_img.show()

```

```

img = Image.open("C:\\Users\\WW82103\\OneDrive\\Jupyter Notebook\\0308\\아이린.jpg")
img_pixel = np.array(img) #ndarray 로 이미지 변환
plt.imshow(img_pixel) #이미지 시각화
print(img_pixel.shape) #(500, 500, 3) #500:가로, 500:세로, 3:색조 (RGB)
# 색조 부분에서 R:0, G:1, B:2 에 해당한다

```



문제56. 아이린 사진에서 red 부분의 행렬을 취하고 red 부분만 이미지로 시각화하시오!

```

from PIL import Image

```

```

import numpy as np

```

```

import matplotlib.pyplot as plt

```

```

def img_show(img):

```

```

    pil_img = Image.fromarray(np.uint8(img))

```

```

    pil_img.show()

```

```

img = Image.open("C:\\Users\\WW82103\\OneDrive\\Jupyter Notebook\\0308\\아이린.jpg")

```

```

img_pixel = np.array(img) #ndarray 로 이미지 변환

```

```

print(img_pixel[:,0]) #red 부분 행렬만 출력

```

```

img_pixel[:,1]=0 #green 부분을 전부 0으로 변경

```

```

img_pixel[:,2]=0 #blue 부분을 전부 0으로 변경

```

```

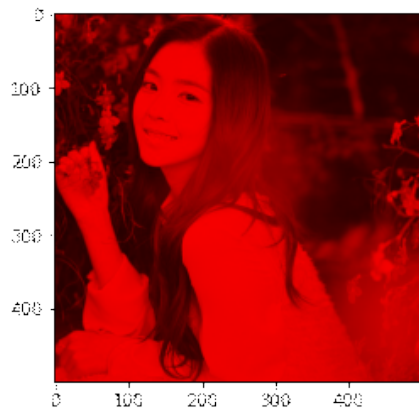
plt.imshow(img_pixel) #이미지 시각화

```

```

[[ 79 113 147 ... 50 50 50]
 [101 137 169 ... 51 51 50]
 [116 152 181 ... 49 49 49]
 ...
 [219 219 222 ... 163 173 184]
 [222 222 224 ... 163 174 184]
 [222 223 225 ... 165 176 184]]

```



문제57. 아이린 사진에서 green 부분만 시각화하시오!

```
from PIL import Image
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def img_show(img):
```

```
    pil_img = Image.fromarray(np.uint8(img))
```

```
    pil_img.show()
```

```
img = Image.open("C:\\Users\\WW82103\\OneDrive\\Jupyter Notebook\\0308\\아이린.jpg")
```

```
img_pixel = np.array(img) #ndarray 로 이미지 변환
```

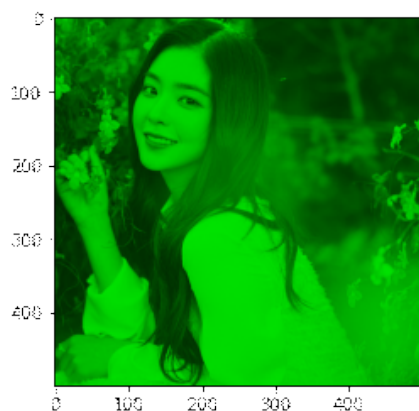
```
print(img_pixel[:,1]) #green 부분 행렬만 출력
```

```
img_pixel[:,0]=0 #red 부분을 전부 0으로 변경
```

```
img_pixel[:,2]=0 #blue 부분을 전부 0으로 변경
```

```
plt.imshow(img_pixel) #이미지 시각화
```

```
[[ 96 118 137 ... 72 72 72]
 [118 142 159 ... 70 70 69]
 [132 156 171 ... 68 68 68]
 ...
 [199 199 202 ... 148 157 168]
 [202 202 204 ... 148 158 168]
 [202 203 205 ... 150 160 168]]
```



문제58. 인터넷에서 원하는 사진을 받아서 파이썬으로 시각화하는데 red, green, blue 중 하나로만 시각화하시오!

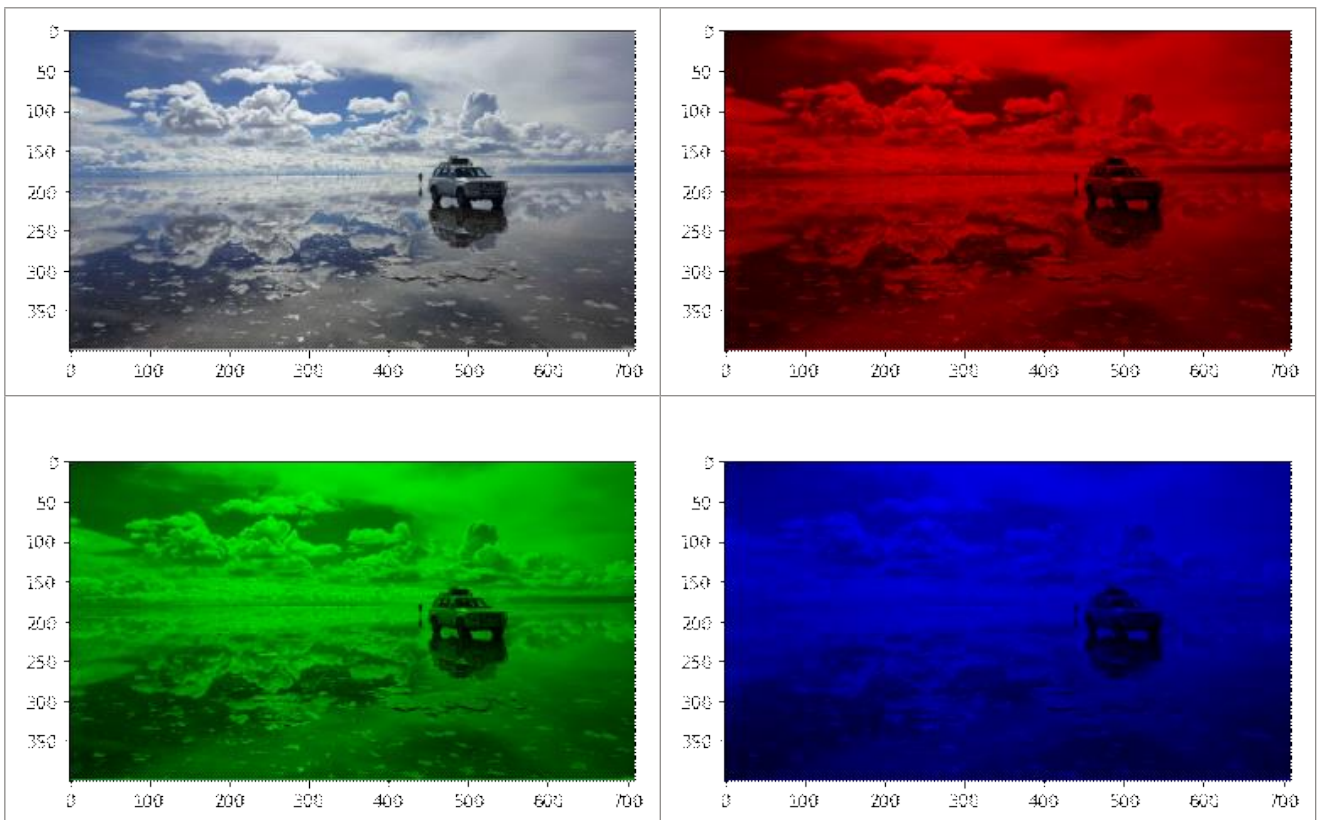
```
from PIL import Image
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def img_show(img):  
    pil_img = Image.fromarray(np.uint8(img))  
    pil_img.show()
```

```
img = Image.open("C:\\Users\\WW82103\\OneDrive\\Jupyter Notebook\\0308\\아이린.jpg")  
img_pixel = np.array(img) #ndarray 로 이미지 변환  
print(img_pixel[:,1]) #green 부분 행렬만 출력  
img_pixel[:,0]=0 #red 부분을 전부 0으로 변경  
img_pixel[:,2]=0 #blue 부분을 전부 0으로 변경  
plt.imshow(img_pixel) #이미지 시각화
```



#EndLine=====

##0309

2021년 3월 1일 월요일 오후 4:52

문제59. 아이린 사진을 흑백으로 시각화하시오!

import numpy as np #신경망에 사진을 입력할 때 숫자행렬로 입력하기 때문에 필요

import matplotlib.pyplot as plt #사진을 파이썬에서 시각화하기 위해 필요

import matplotlib.image as mpimg #사진을 불러와서 숫자로 변화해주는 모듈

def rgb2gray(rgb): #흑백으로 색을 변경하기 위한 함수

 return np.dot(rgb[...,:3], [0.299, 0.587, 0.114]) #

rgb[행, 열, 색조] ---> [:, :, :3] == [..., :3]

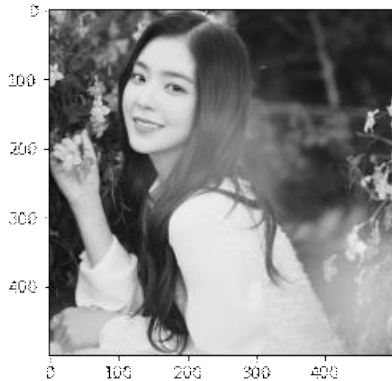
j= 'C:\Users\Wicr\OneDrive\Jupyter Notebook\#0308\아이린.jpg'

img = mpimg.imread(j)

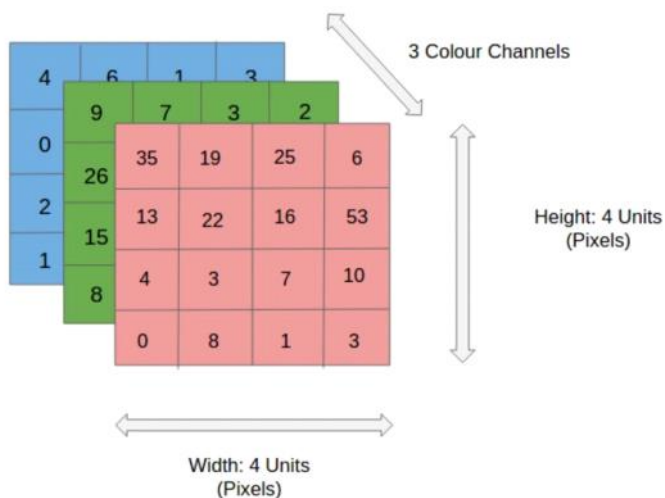
gray = rgb2gray(img)

plt.imshow(gray, cmap = plt.get_cmap('gray'))

plt.show()



컬러사진의 구조



컬러 : RGB

흑백 : Gray

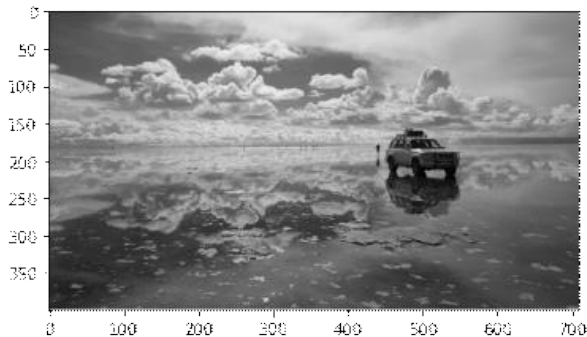
문제60. 어제 마지막 문제로 풀었던 사진을 흑백으로 변경하고 1차원으로 flatten 시키시오!

import numpy as np #신경망에 사진을 입력할 때 숫자행렬로 입력하기 때문에 필요

```
import matplotlib.pyplot as plt #사진을 파이썬에서 시각화하기 위해 필요
import matplotlib.image as mpimg #사진을 불러와서 숫자로 변화해주는 모듈
```

```
def rgb2gray(rgb): #흑백으로 색을 변경하기 위한 함수
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114]) #
# rgb[ 행, 열, 색조] --->[:, :, :3] == [..., :3]
```

```
j= 'C:\Users\wicr\OneDrive\Jupyter Notebook\0308\우유사막.jpg'
img = mpimg.imread(j)
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()
```



[code]

common.py file을 계속 update ---> 날짜별 파일로 진행상황을 알 수 있도록 복사해서 수정
3층 신경망 구현시 **common.py** 내부의 **init_network()**, **softmax()**, **sigmoid()** 함수를 활용

[1] common.py

```
def sigmoid(x):
    import numpy as np
    return 1/(1+np.exp(-x))
```

```
def softmax(a):
    import numpy as np
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
def init_network():
    import numpy as np
    network = {}

    network['W1'] = np.array([[1,3,5],[2,4,6]])
    network['W2'] = np.array([[3,4],[5,6],[7,8]])
    network['W3'] = np.array([[4,5],[6,7]])
    network['b1'] = np.array([0.1,0.2,0.3])
```



```
network['b2'] = np.array([0.1,0.2])
network['b3'] = np.array([0.1,0.2])
```

```
return network
```

[2] NNet

```
## Import modules
import numpy as np
import sys, os
from common import init_network, sigmoid, softmax
sys.path.append(os.pardir)

network = init_network()
W1,W2,W3 = network['W1'], network['W2'], network['W3']
b1,b2,b3 = network['b1'], network['b2'], network['b3']

## stack
#Floor 0
x = np.array( [1,2] ) #(2,)

#Floor 1
a1 = np.dot(x, W1)+b1
a1_sig = sigmoid(a1) #(3, )

#Floor 2
a2 = np.dot(a1_sig, W2)+b2
a2_sig= sigmoid(a2) #(2, )

#Floor 3
a3 = np.dot(a2_sig, W3)+b3
y_hat = softmax(a3) #(2, )

y_hat #array([0.49999099, 0.50000901])
```

● 저자가 만들어온 가중치 피클파일을 이해하기 위한 사전 작업

힘들게 신경망을 학습시켜서 만들어 놓은 가중치를 파일로 생성하는 방법
"pickle" 이용

예제. pickle 파일 생성 예제

```
import pickle
a = ['a','b','c'] #학습시킨 가중치 값이라고 가정
with open("c:\Users\Wicecr\OneDrive\JupyterNotebook\0309\wa.pkl", "wb") as f:
    pickle.dump(a, f)
#wb 옵션은 binary(이진) 파일로 생성하는 옵션

#a.pkl 파일이 생성되었다
```

이름	수정한 날짜	유형	크기
.ipynb_checkpoints	2021-03-09 오전 8:55	파일 폴더	
__pycache__	2021-03-09 오전 10:36	파일 폴더	
dataset	2021-03-09 오전 10:20	파일 폴더	
a.pkl	2021-03-09 오전 10:51	PKL 파일	1KB
common.py	2021-03-09 오전 10:35	PY 파일	1KB
Untitled.ipynb	2021-03-09 오전 10:51	IPYNB 파일	7KB

예제2. pickle 파일을 파이썬으로 로드하는 예제

```
with open("c:\Users\Wicr\OneDrive\JupyterNotebook\#0309\#a.pkl", "rb") as f:
    data=pickle.load(f)
print(data)
```

저자가 미리 만들어온 가중치와 바이어스가 들어있는 pickle 파일을 파이썬으로 불러와서 bias, w 의 가중치 확인

```
import pickle
def init_network():
    with open("c:\Users\Wicr\OneDrive\JupyterNotebook\#0309\sample_weight.pkl", "rb") as f:
        network=pickle.load(f)
    return network
```

```
network = init_network()
```

```
print(network.keys()) #dict_keys(['b2', 'W1', 'b1', 'W2', 'W3', 'b3'])
print(network['W1'].shape) #(784, 50)
print(network['W2'].shape) #(50, 100)
print(network['W3'].shape) #(100, 10)
print(network['b1'].shape) #(50,)
print(network['b2'].shape) #(100,)
print(network['b3'].shape) #(10,)
```

입력층 -----> 은닉1층 -----> 은닉 2층 -----> 출력층 3층
 (60000, 784) \odot (784, 50) --> (60000, 50) \odot (50, 100) --> (60000, 100) \odot (100, 10)
 앞뒤 앞뒤 앞뒤

mnist 데이터의 N = 60000

신경망에 60000개를 한번에 넣으면 성능이 매우 뛰어난 pc도 버거워하기 때문에 100개만 넣어서 실습

활성화함수는 sigmoid, 출력함수로는 확률벡터를 반환하는 softmax 를 사용

저자가 구해놓은 pickle file 을 로드하는 함수를 init_network 에 적용하여 common.py를 수정하고, 코드도 수정하시오!

[common.py]

```
import numpy as np
```

```
import pickle
```

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

```
def softmax(a):
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y

def init_network():
    with open("c:\\Users\\wicecr\\OneDrive\\Jupyter Notebook\\0309\\sample_weight.pkl", "rb") as f:
        network=pickle.load(f)
    return network
```

[code]

```
import numpy as np
from common import init_network, sigmoid, softmax

network = init_network()

print(network.keys()) #dict_keys(['b2', 'W1', 'b1', 'W2', 'W3', 'b3'])
print(network['W1'].shape) #(784, 50)
print(network['W2'].shape) #(50, 100)
print(network['W3'].shape) #(100, 10)
print(network['b1'].shape) #(50,)
print(network['b2'].shape) #(100,)
print(network['b3'].shape) #(10,)
```

필기체 데이터 불러오기

```
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
import sys, os
sys.path.append(os.pardir)

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
print(x_train.shape) #(60000, 784)
# normalize : 이미지의 픽셀값을 0.0 ~ 1.0 사이로 정규화할지여부를 결정 #단위를 일치시켜줌
# flatten : 입력이미지를 1차원 배열로 생성할지를 정한다. (원래는 2차원데이터)
# one_hot_label : 정답을 숫자로 표현할지 one_hot_encoding 할지 결정하는 파라미터 (0,1 binary)

print(t_train[0:10])
#one_hot_label=False #[5 0 4 1 9 2 1 3 1 4]
#one_hot_label=True
# [[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
# [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
# [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
# [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
# [0. 0. 0. 0. 0. 0. 0. 0. 1.]
# [0. 0. 1. 0. 0. 0. 0. 0. 0.]
# [0. 1. 0. 0. 0. 0. 0. 0. 0.]
# [0. 0. 0. 1. 0. 0. 0. 0. 0.]
# [0. 1. 0. 0. 0. 0. 0. 0. 0.]
# [0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

●● 필기체 데이터 10개를 3층 신경망에 흘려보내서 예측값을 출력하기

```
# Import modules
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
import sys, os

#Path
sys.path.append(os.pardir)

#Ddivide db
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
print(x_train.shape) #(60000, 784)

#Setting weight, bias
network = init_network()
W1,W2,W3 = network['W1'], network['W2'], network['W3']
b1,b2,b3 = network['b1'], network['b2'], network['b3']

##Stack
#Floor 0
x = x_train[0:10] #Batch=10

#Floor 1
a1 = np.dot(x, W1)+b1
a1_sig = sigmoid(a1)

#Floor 2
a2 = np.dot(a1_sig, W2)+b2
a2_sig= sigmoid(a2)

#Floor 3
a3 = np.dot(a2_sig, W3)+b3
y_hat = softmax(a3)

y_hat #y_hat.shape : (10,10) #10개의 img(row)에 대해 10개의 확률벡터값을 반환
np.argmax(y_hat, axis=1) #y_hat.shape : (10,10) ---> axis=1(column) 들 중 max 값을 반환
```

문제61. 위에서 예측한 10장 중 실제로 몇개를 맞추었는지 확인하시오!

```
np.argmax(y_hat, axis=1) #pred
print(t_train[0:10]) #Label
```

```
#[5 0 4 1 9 2 1 3 1 4]
#[5 0 4 1 9 2 1 3 1 4]
```

☆문제62. 훈련데이터 총 100장을 흘려보내고 그 중 몇개를 맞추었는지 확인하시오! -----

[Create 3rd floors]

```
# Import modules
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
import sys, os

#Path
sys.path.append(os.pardir)

#Ddivide db
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
print(x_train.shape) #(60000, 784)

#Setting weight, bias
network = init_network()
W1,W2,W3 = network['W1'], network['W2'], network['W3']
b1,b2,b3 = network['b1'], network['b2'], network['b3']

##Stack
#Floor 0
x = x_train[0:100] #Batch=10

#Floor 1
a1 = np.dot(x, W1)+b1
a1_sig = sigmoid(a1)

#Floor 2
a2 = np.dot(a1_sig, W2)+b2
a2_sig= sigmoid(a2)

#Floor 3
a3 = np.dot(a2_sig, W3)+b3
y_hat = softmax(a3)

y_hat #y_hat.shape : (10,10) #10개의 img(row)에 대해 10개의 확률벡터값을 반환
np.argmax(y_hat, axis=1) #y_hat.shape : (10,10) ---> axis=1(column) 들 중 max 값을 반환

[get acc]
pred = np.argmax(y_hat, axis=1)
label = t_train[0:100]

temp = pred==label
acc = temp.astype(int).sum() / len(label)
acc #0.96
```

[common.py]

```
import numpy as np
import pickle
```

```
def sigmoid(x):
    return 1/(1+np.exp(-x) )
```

```
def softmax(a):
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
def init_network():
    with open("c:\\Users\\wicr\\OneDrive\\Jupyter Notebook\\0309\\sample_weight.pkl", "rb") as f:
        network=pickle.load(f)
    return network
```

3장 p.100 코드 ---> 3장의 내용 완성 코드

"저자가 만들어온 가중치와 바이어스를 이용해서 신경망을 만들고 필기체 데이터를 잘 예측하는지 확인하는 코드"

1. 파이썬 기초
2. 함수 생성
3. 함수로 클래스 생성

저자와 같이 위에서 만든 코드를 함수로 생성해 보자

[code]

```
# Import modules
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
import sys, os

#Path
sys.path.append(os.pardir)

#1. Ddivide db
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
    return x_test, t_test

#2. w, b를 불러와서 3층 신경망에 흘려보내는 함수
def predict(network, x):
    #network = init_network()
```

```
W1,W2,W3 = network['W1'], network['W2'], network['W3']
b1,b2,b3 = network['b1'], network['b2'], network['b3']
```

```
#Floor 1
```

```
a1 = np.dot(x, W1)+b1
a1_sig = sigmoid(a1)
```

```
#Floor 2
```

```
a2 = np.dot(a1_sig, W2)+b2
a2_sig= sigmoid(a2)
```

```
#Floor 3
```

```
a3 = np.dot(a2_sig, W3)+b3
y_hat = softmax(a3)
return y_hat
```

```
y_hat #y_hat.shape : (10,10) #10개의 img(row)에 대해 10개의 확률벡터값을 반환
np.argmax(y_hat, axis=1) #y_hat.shape : (10,10) ---> axis=1(column) 들 중 max 값을 반환
```

#3. get_data, predict 함수를 가져와서 실행하는 코드

```
x, t = get_data() #x_test, t_test 를 호출
network = init_network() #W, bias 호출
```

```
accuracy_cnt = 0
for i in range(len(x)):
    y = predict(network, x[i])
    p = np.argmax(y)
    if p == t[i]:
        accuracy_cnt += 1
print("accuracy :", accuracy_cnt / len(x) )
accuracy : 0.9352
```

문제63. 10000장 전체를 다 돌리지 말고 1000장만 돌리시오!

```
x, t = get_data() #x_test, t_test 를 호출
network = init_network() #W, bias 호출
```

```
accuracy_cnt = 0
for i in range(1000):
    y = predict(network, x[i])
    p = np.argmax(y)
    if p == t[i]:
        accuracy_cnt += 1
print("accuracy :", accuracy_cnt / 1000 )

print(x.shape)
```

문제64. 훈련데이터 6만장을 다 불러와서 신경망에 넣고 전체 6만개 중 몇개를 맞추는지 정확도를 확인하시오!

```
return x_test, t_test <--- 이부분만 수정
accuracy : 0.9357666666666666
```

문제 65. 위 예제들의 #1, #2 를 class 로 구현하시오!

```
class Three_nn():
    # Import modules
    import numpy as np
    from common import sigmoid, softmax
    from dataset.mnist import load_mnist
    import sys, os

    #Path
    sys.path.append(os.pardir)

    def init_network(self):
        import pickle
        with open("c:\\Users\\wicr\\OneDrive\\JupyterNotebook\\0309\\sample_weight.pkl", "rb") as f:
            network=pickle.load(f)
        return network

    #1. Load db and setting
    def get_data(self):
        (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
        return x_train, t_train

    #2. w, b를 불러와서 3층 신경망에 흘려보내는 함수
    def predict(self, network, x):
        W1,W2,W3 = network['W1'], network['W2'], network['W3']
        b1,b2,b3 = network['b1'], network['b2'], network['b3']

        #Floor 1
        a1 = np.dot(x, W1)+b1
        a1_sig = sigmoid(a1)

        #Floor 2
        a2 = np.dot(a1_sig, W2)+b2
        a2_sig= sigmoid(a2)

        #Floor 3
        a3 = np.dot(a2_sig, W3)+b3
        y_hat = softmax(a3)
        return y_hat

# Class 생성 및 method 사용
n1 = Three_nn()
x, t = n1.get_data() #x_test, t_test 를 호출
network = n1.init_network() #W, bias 호출
```



```

accuracy_cnt = 0
for i in range(len(x)):
    y = predict(network, x[i])
    p = np.argmax(y)
    if p == t[i]:
        accuracy_cnt += 1
print("accuracy :", accuracy_cnt / len(x))

```

문제66. 훈련 데이터의 첫 번째 데이터를 3층 신경망에 넣고 예측값을 출력하시오!

```

# Class 생성 및 method 사용
n1 = Three_nn()
x, t = n1.get_data() #x_test, t_test 를 호출
network = n1.init_network() #W, bias 호출

res = n1.predict(network, x[0])
print(np.argmax(res)) #5
print(t[0]) #5

```

문제67. 아이린 사진을 필기체를 인식하는 3층 신경망에 넣으면 무엇이 출력될까?

```

아이린 사진 flatten --> irin.shape : (1,250000)
저자가 만든 가중치 W ---> w1.shape : (784, 50)
# 내적이 불가능하다

```

[참고코드]

```

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
img = Image.open("C:\\Users\\wiccr\\OneDrive\\JupyterNotebook\\0308\\아이린.jpg")
img_pixel = np.array(img) #ndarray 로 이미지 변환
plt.imshow(img_pixel) #이미지 시각화
print(img_pixel.shape) #(500, 500, 3) #500:가로, 500:세로, 3:색조 (RGB)

```

```

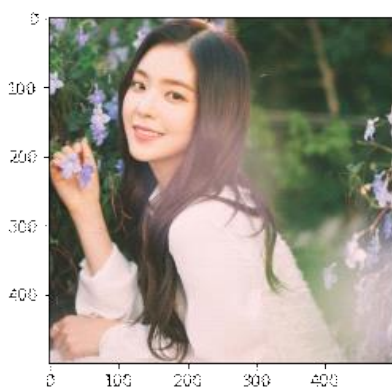
img_pixel = img_pixel.flatten() #3차원 ndarray를 1차원으로 변환
print(img_pixel.shape) #(750000,) #1차원 ndarray로 변환

```

```

(500, 500, 3)
(750000,)

```



배치처리 p.102

훈련데이터가 6만장 ---> 한번에 nnet 으로 학습하면 pc 가 momory 사용량을 초과하여 작업수행이 불가능하다.

ex) 사람이 책을 읽을 때도 한번에 책 한권을 똑딱하고 볼 수 없고 1page 씩 보는 것과 유사

필기체 6만장을 학습할 때 1개씩 학습한다면 perdict 함수를 6만번 실행해야 한다.

하지만 6만장 학습시 100개 묶음으로 학습한다면 predict 함수를 600번만 실행하면 된다.

이와같은 처리를 '배치처리'라고 하며 배치처리로 학습수행시 computational cost 가 감소한다.

normal equation : 수학적 적근(gradient descent) ---> pc로 접근이 불가

문제68. 지금까지의 코딩은 훈련데이터 6만장을 한번에 predict 에 넣고 예측하는 것 --> 100개씩 배치처리로 구현하시오!

```
class Three_nn():
    # Import modules
    import numpy as np
    from common import sigmoid, softmax
    from dataset.mnist import load_mnist
    import sys, os

    #Path
    sys.path.append(os.pardir)

    def init_network(self):
        import pickle
        with open("c:\\Users\\Wicecr\\OneDrive\\JupyterNotebook\\0309\\sample_weight.pkl", "rb") as f:
            network=pickle.load(f)
        return network

    #1. Load db and setting
    def get_data(self):
        (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
        return x_train, t_train

    #2. w, b를 불러와서 3층 신경망에 흘려보내는 함수
    def predict(self, network, x):
        W1,W2,W3 = network['W1'], network['W2'], network['W3']
        b1,b2,b3 = network['b1'], network['b2'], network['b3']

        #Floor 1
        a1 = np.dot(x, W1)+b1
        a1_sig = sigmoid(a1)

        #Floor 2
        a2 = np.dot(a1_sig, W2)+b2
        a2_sig= sigmoid(a2)

        #Floor 3
```

```

a3 = np.dot(a2_sig, W3)+b3
y_hat = softmax(a3)
return y_hat

# Class 생성 및 method 사용
n1 = Three_nn()
x, t = n1.get_data() #x_test, t_test 를 호출
network = n1.init_network() #W, bias 호출
a=[]
batch_size=100 #set batch
accuracy_cnt =0

for i in range(0, len(x), batch_size): #i=0, 100, 200, ... , 60000
    y = predict(network, x[i:i+batch_size]) #x[0:100], x[100:200], x[200:300], ...x[59900:60000] #start=0
    y_hat = np.argmax(y, axis=1) #100개의 예측값 반환(argmax로 최대확률값 반환)
    a.append( sum( (y_hat==t[i:i+batch_size])/100 ) ) #Accuracy ---> append in a list

print(len(a)) #600
print(np.mean(a)) #0.93576666666666673
a

```

●● 3장 정리

1. 신경망에 들어가는 함수 : sigmoid, ReLU, softmax ---> python 으로 구현
2. 저자가 만들어온 가중치 피클파일(.plk)로 3층 신경망 구성
3. 배치처리로 데이터를 신경망에 흘려보내는 이유와 그 구현

■ 4장. 신경망 학습

"이 장에서는 신경망의 bias 와 Weight를 직접 gradient descent 등의 방법을 사용하여 학습시킨다"

신경망 학습을 위해 알아야 하는 4장내용

1. 오차함수 : loss function : 신경망 가중치 갱신의 기준
2. 미니배치 : mini batch : 배치단위로 신경망을 효율적으로 학습(less computational cost)
3. 수치미분 : derivative : 오차함수의 기울기만큼 가중치 갱신 도구

□ 오차함수 p.111

- 예상값과 실제값과의 오차를 신경망에 역전파 시켜주기 위해서 필요한 함수
 - 신경망의 잘못을 깨닫게 해주는 함수
1. 평균제곱 오차함수 : MSE(mean squared error) : 회귀분석시 사용
 2. 교차엔트로피 오차함수 : CEE(cross entropy error) : 분류문제를 풀 때 사용

MSE : 평균제곱 오차 함수

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

[code]

```
import numpy as np

def MSE(y, t):
    return 0.5 * np.sum( (y-t)**2 )

y = np.array([0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]) #pred=3
t = np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0]) #label=3

print(MSE(y,t)) #0.097500000000000003
```

69) 숫자 7로 예측한 결과와 정답 숫자 2와의 오차를 구하시오!

```
import numpy as np
def MSE(y, t):
    return 0.5 * np.sum( (y-t)**2 )

y = np.array([0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]) #pred=7
t = np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0]) #label=3
print(MSE(y,t)) #0.5975
# 오차가 상승한 것을 볼 수 있다
# 오차 0.5975로 상승했지만, 분류문제를 해결하기 위해서는 더 큰 오차를 반환해야 한다.
# 이 때 필요한 함수가 교차엔트로피 함수이다(MSE 대신 사용)
```

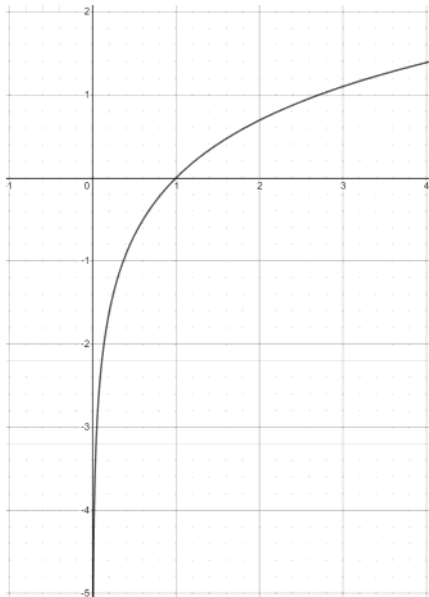
교차 엔트로피 함수 : CEE(cross entropy error)

$$E = - \sum_k t_k \log y_k$$

```
def CEE(y, t):
    delta = 1e-7
    return -np.sum(t * np.log(y + delta) )    #np.log(0) ---> - infinite

y = np.array([0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]) #pred=7
t = np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0]) #label=3
print(CEE(y,t)) #2.302584092994546 #MSE 에 비해서 더 높은 오차를 반환
print(np.log(0)) #-inf

# CEE function 에서 delta 값을 y 에 + 해주는 이유
# ln0 = - 무한대 이기 때문
```



#y = lnX

□ 미니배치 학습 p.115

minibatch : sampling

3장에서는 저자가 .plk 파일에 저장한 weight, bias 값으로 신경망을 학습하였다.

4장에서는 minibatch, cost function, derivate(수치미분)을 활용해서 신경망을 학습하는 코드를 작성한다. (weight, bias 도출)

앞서 설명한 batch 개념과 유사하다

(100, 784) ⊙ (784, 50) --> (100, 50) ⊙ (50, 100) ---> (100, 100) ⊙ (100,10)

--> (100, 10)

앞의 100 이 minibatch 단위인데, sampling 할 때 복원추출, 비복원추출 여부는 크게 중요하지 않으며 합승 N수가 커지면 학습이 된다.

epoch

훈련단위로써, trainset 의 N=60000 일 때 minibatch=100 이면, 1epoch = N/minibatch = 600 이 된다.

즉 1epoch = trainset의 N을 다 사용하는 minibatch의 반복 횟수라고 생각할 수 있다

문제70. 숫자 0~60000 숫자 중 무작위로 10개를 출력하시오!

```
import numpy as np
```

```
x = np.random.choice(np.arange(0, 60000), 10)
```

```
x #array([46782, 53727, 56931, 25422, 57480, 30552, 59837, 2749, 36610, 49793]) #수행할때마다 달라진다
```

문제71. 숫자 0~60000 sampling 100

```
import numpy as np
```

```
x = np.random.choice(np.arange(0, 60000), 100)
```

```
x
```

72. 3장에서 마지막으로 작성한 필기체 숫자 예측하는 전체 코드를 로드하여 100개를 랜덤추출하여 예측하도록 코드를 수정하시오!

[code] -----#0309최종-----

```
class Three_nn():
```

```
    def init_network(self):
```

```
        #Path
```

```

import sys, os
sys.path.append(os.pardir)

import pickle
with open("c:\\Users\\wicecr\\OneDrive\\Jupyter Notebook\\0309\\sample_weight.pkl", "rb") as f:
    network=pickle.load(f)
return network

```

#1. Load db and setting

```

def get_data(self):
    from dataset.mnist import load_mnist
    (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
    return x_train, t_train

```

#2. w, b를 불러와서 3층 신경망에 흘려보내는 함수

```

def predict(self, network, x):
    from common import sigmoid, softmax
    import numpy as np

    W1,W2,W3 = network['W1'], network['W2'], network['W3']
    b1,b2,b3 = network['b1'], network['b2'], network['b3']

    #Floor 1
    a1 = np.dot(x, W1)+b1
    a1_sig = sigmoid(a1)

    #Floor 2
    a2 = np.dot(a1_sig, W2)+b2
    a2_sig= sigmoid(a2)

    #Floor 3
    a3 = np.dot(a2_sig, W3)+b3
    y_hat = softmax(a3)
    return y_hat

```

Class 생성 및 method 사용

```

import numpy as np
n1 = Three_nn()
x, t = n1.get_data() #x_test, t_test 를 호출
network = n1.init_network() #W, bias 호출
a=[]
batch_size=100 #set batch
accuracy_cnt =0

for i in range(0, len(x), batch_size): #i=0, 100, 200, ... , 60000
    batch_mask=np.random.choice(len(x), batch_size) #random sampling #batch_mask
    y = n1.predict(network, x[batch_mask] )
    y_hat = np.argmax(y, axis=1) #100개의 예측값 반환(argmax로 최대확률값 반환)
    a.append( sum( (y_hat==t[batch_mask])/100 ) ) #Accuracy ---> append in a list

```

```
print(len(a)) #600
print(np.mean(a)) #random sampling 이 포함되어 있음 #0.93 정도로 수렴
```

[common.py]

```
def sigmoid(x):
    import numpy as np
    return 1/(1+np.exp(-x) )

def softmax(a):
    import numpy as np
    C = np.max(a)
    minus = a-C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

□ 수치미분 p.121

가중치를 갱신하기 위해 미분이 필요하다.

가중치 = 가중치-기울기

[code]

```
def numerical_diff(f,x):
    h = 1e-50
    return( (f(x+h)-f(x)) /h)
```

```
def f(x):
    return 2* x**2 + 2
```

```
numerical_diff(f,4) #0
```

진정한 미분은 컴퓨터로 구현할 수가 없다.

컴퓨터로 구현하면 분모가 0 이 되어서 계산이 안된다.

$$\text{도함수 공식 : } \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{(x+h) - (x-h)} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2*h}$$

위의 식을 파이썬으로 구현하면 ?

```
def numerical_diff(f,x):
    h = 1e-4
    return (f(x+h)-f(x-h)) / (2*h)
```

```
def f(x):
```

```
return 2*x**2 + 2
```

```
numerical_diff(f, 4) #16
```

문제73. (오늘의 마지막 문제) 위에서 만든 미분함수를 이용해서 아래의 함수를 미분해서 기울기를 구하는데 $x = 6$ 일때의 미분계수를 구하시오 !

```
def numerical_diff( f, x):
```

```
    h = 0.0001 # 극한값을 구현
```

```
    return ( f(x+h) - f(x-h) ) / (2*h)
```

$$f(x) = 3x^4 + 2x^3 + 6x^2 + 7$$

5시 신호 보냈습니다.

```
def numerical_diff(f,x):
```

```
    h = 1e-4
```

```
    return (f(x+h)-f(x-h)) / (2*h)
```

```
def f(x):
```

```
    return 3*x**4 + 2*x**3 + 6*x**2 + 7
```

```
numerical_diff(f, 6) #2880
```

```
#####endLine=====
```


##0310

2021년 3월 1일 월요일 오후 4:52

■ Review

- 1장. numpy 사용법
- 2장. 퍼셉트론
- 3장. 3층 신경망 구현(저자가 미리 만들어온 가중치를 세팅해서 신경망 구성)
- 4장. 2층 신경망 구현(w, bias 직접 수정)

methods

- 1. 오차함수 : 신경망 가중치 갱신을 위한 함수(loss)
- 2. 미니배치 : 신경망 데이터 학습시 묶음 단위(속도향상, 계산가능위한 방법)
- 3. 수치미분 : 인공신경망 구현의 최종목적은 신경망의 파라미터 (weight, bias)의 최적값을 찾는 것

examples

- 필기체 데이터 분류 : weight, bias
- 폐결절과 정상폐 분류 : weight, bias

추가로 알고 있어야 하는 필수기술

- 1. 웹스크롤링 기술
- 2. 이미지를 신경망에 넣고 학습시키는 기술

인공지능 종류

인공지능의 눈(cnn)

- 1. 웹스크롤링 기술 필요
- 2. 이미지를 신경망에 넣고 학습시키는 기술

인공지능의 입과 귀(rnn)

- 1. 웹스크롤링 기술 필요
- 2. 음악, 텍스트(판결문, 신문기사)를 신경망에 넣고 학습시키는 기술

직군

- 데이터 분석가
- 딥러닝 개발자(연구원)
- 인공지능 개발자

■ 4장. 신경망 학습(이어서)

수치미분 ---> 코드로 구현

구글에서 아래 식 검색

$$z = x^2 + y^2$$

□ 편미분

변수가 2개 이상인 함수를 미분할 때 미분 대상 변수 외에 나머지 변수를 상수처럼 고정시켜서 미분하는 것

$w_1 = w_1$ - 기울기

$w_2 = w_2$ - 기울기

문제74. 아래의 수학식을 편미분하시오!

$$y = 2x^2 + 4z^3 + 3$$

1. $4x$

2. $12z^2$

문제75. 아래의 수학식을 오차함수로 생성하시오!

[식 4.6]

$$f(x_0, x_1) = x_0^2 + x_1^2$$

```
import numpy as np
def loss_func(x):
    return x[0]**2 + x[1]**2
x = np.array( [3.0, 4.0] )
print(loss_func(x) ) #25.0
```

문제76. 위 loss_func() 함수를 $x_0=3$, $x_1=4$ 에서 x_0 에 대해 편미분했을때의 기울기는?

1. $2x_0$ ----> 6

2. $2x_1$ ----> 8

[code]

```
import numpy as np
```

```
def numerical_diff(f,x):
    h = 1e-4
    return (f(x+h)-f(x-h)) / (2*h)
```

```
def function_tmp1(x0):
    return x0**2 + 4**2 #x1 변수를 4로 상수취급
```

```
x = np.array( [3.0, 4.0] )
```

```
print(numerical_diff( function_tmp1, 3) ) #6.00000000000378 #6으로 딱 떨어지지 않는것은 중앙차분오차가 존재하기 때문이다.
```

문제76. 위 loss_func() 함수를 $x_0=3$, $x_1=4$ 에서 x_1 에 대해 편미분했을때의 기울기는?

[code]

```
import numpy as np
```

```
def numerical_diff(f,x):
    h = 1e-4
    return (f(x+h)-f(x-h)) / (2*h)
```

```
def function_tmp1(x1):
    return x1**2 + 3**2 #x0 변수를 3으로 상수취급

x = np.array( [3.0, 4.0] )

print(numerical_diff( function_tmp1, 4) ) #7.999999999999119
```

● 편미분 numerical_gradient 함수 생성 p.127

```
import numpy as np
```

```
def numerical_gradient(f, x):
    h = 0.0001
    grad = np.zeros_like(x) #x와 형상이 같은 zero 배열을 생성 [0, 0]

    for i in range(x.size): #x.size=2 #array의 메소드인 size는 인자의 개수를 반환
        tmp_val = x[i]
        x[i] = tmp_val + h
        fxh1 = f(x)
        x[i] = tmp_val - h
        fxh2 = f(x)

        grad[i] = (fxh1 - fxh2) / (2*h) #x[i]에 대한 편미분
        x[i] = tmp_val #값 복원
    return grad
```

```
def loss_func(x):
    return x[0]**2 + x[1]**2
```

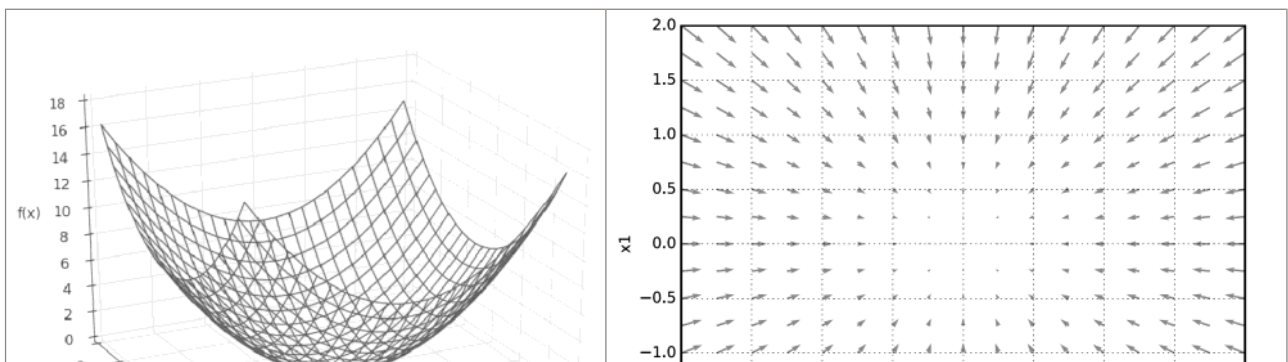
```
x = np.array( [3.0, 4.0] )
print(numerical_gradient(loss_func, x)) #[6. 8.]
```

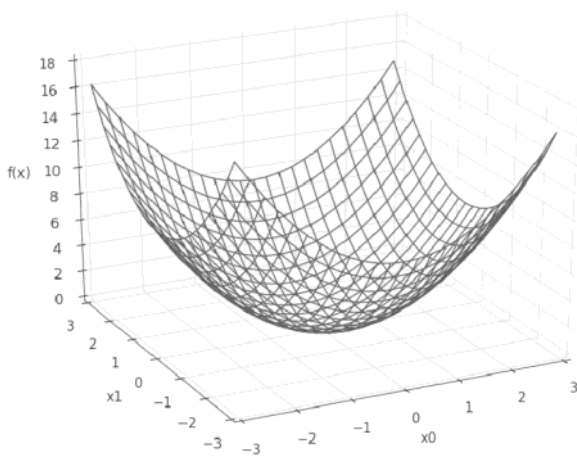
문제78. 위 함수를 디버깅하는데, i=1 일때 하시오!

...

● 경사하강법(p.129)

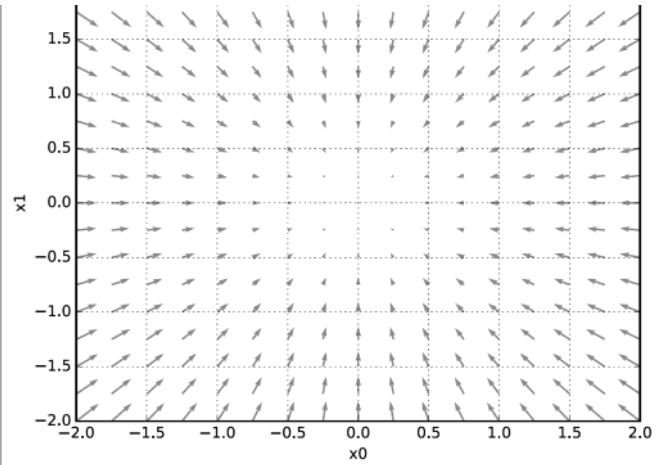
방금 만든 numerical_gradient 함수는 산에서 내려오기 위해서 내가 서있는 곳에서 어느쪽으로 가야 산 아래로 내려갈 수 있는지 내가 서 있는 곳의 기울기를 구하는 함수.





[그림 4-8]

3차원 그래프



[그림 4-9]

w1, w2 2개의 가중치의 편미분 값의 벡터를 2차원공간에 표현
(w1 편미분값, 기울기) 형식이므로 위와 같이 표현됨

갱신

[가중치 -= 기울기]

위 식을 loop 문을 반복해서 수행해서 기울기가 0이 되면 가중치가 변경되지 않는다. 그 지점까지 수행하여 최적값 산출

갱신 형태

$$W = W - a * \text{grad}$$

변수설명

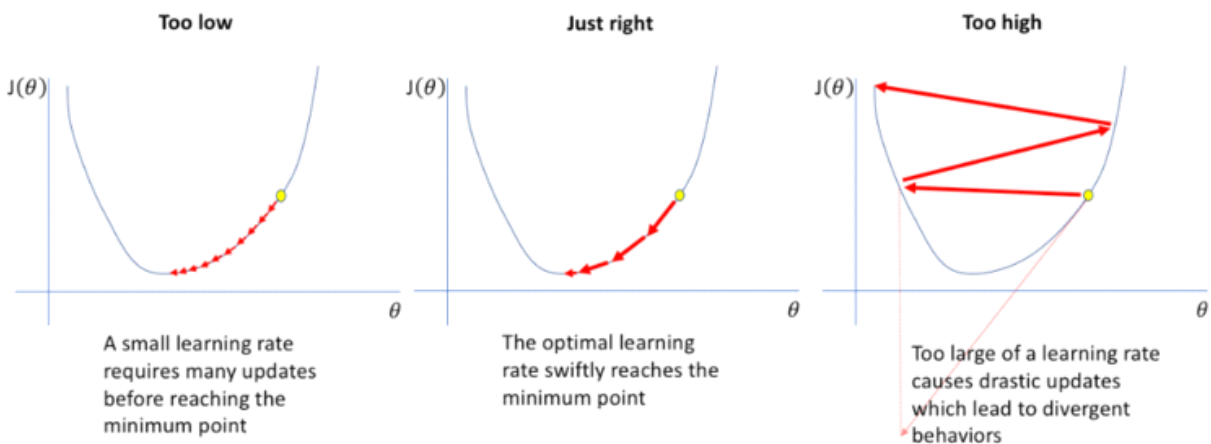
W : 가중치

a : learning rate(학습률) : 한 번의 학습으로 매개변수 갱신폭을 결정하는 hyper parameter

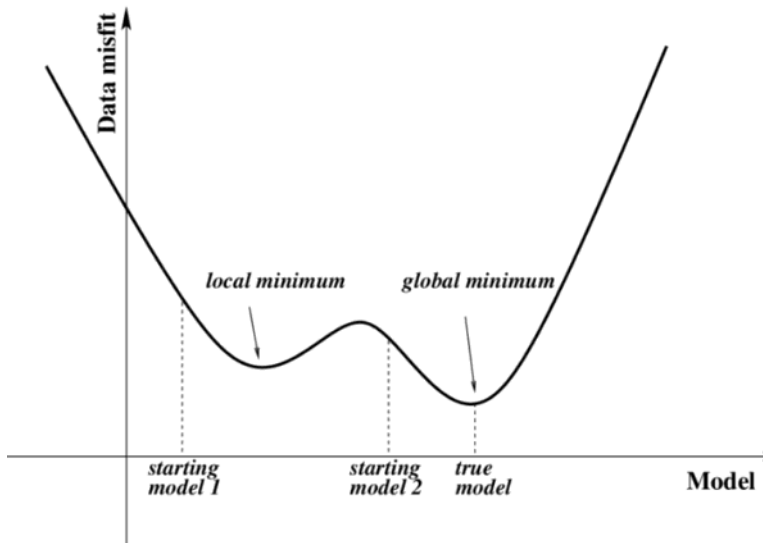
grad : 기울기(벡터)

학습률은 개발자가 0.01, 0.001 등 개발자가 임의로 생성해주는데, 이 값이 너무 크거나 작으면 global minima 를 찾을 수 없다.

learning rate 의 값에 따른 loss function 의 도함수 탐색 모양 example



local minimum, global minimum(=optimizer)



문제79. 방금 생성한 `numerical_gradient` 함수를 이용해서 경사하강을 하는 `gradient_descent` 함수를 생성하시오! p.131

```
def gradient_descent(f, init_x, lr=0.01, step_num=100):
    x = init_x
    for i in range(step_num):
        grad = numerical_gradient(f, x)
        x -= lr * grad
    return x
```

문제80. p.132 와 같이 함수 $f(x_0, x_1) = x_0^2 + x_1^2$ 함수를 오차함수로 두고 처음 지점을 [3.0, 4.0] 으로 지정해서 경사하강을 하여 최소지점이 최소값인 [0,0] 지점으로 경사하강되게 하시오!

```
def loss_func(x):
    return x[0]**2 + x[1]**2
```

```
def gradient_descent(f, init_x, lr=0.01, step_num=100):
    x = init_x
    for i in range(step_num):
        grad = numerical_gradient(f, x)
        x -= lr * grad
    return x
```

```
x = np.array( [3.0, 4.0] )
```

```
gradient_descent(loss_func, x, lr=0.1, step_num=100).round() #array([0., 0.])
```

문제81. `learning rate=0.1` 이 아닌 매우 작은 값을 주어서 최소지점에 도달하지 못하는지 확인하시오! (step=100)

```
import numpy as np
```

```
def numerical_gradient(f, x):
    h = 0.0001
    grad = np.zeros_like(x) #x와 형상이 같은 zero 배열을 생성 [0, 0]
```

```
    for i in range(x.size): #x.size=2 #array의 메소드인 size는 인자의 개수를 반환
        tmp_val = x[i]
```

```

x[i] = tmp_val + h
fxh1 = f(x)
x[i] = tmp_val - h
fxh2 = f(x)

grad[i] = (fxh1 - fxh2) / (2*h) #x[i]에 대한 편미분
x[i] = tmp_val #값 복원

```

```
return grad
```

```
def loss_func(x):
    return x[0]**2 + x[1]**2

```

```
def gradient_descent(f, init_x, lr=0.01, step_num=100):
    x = init_x
    for i in range(step_num):
        grad = numerical_gradient(f, x)
        x -= lr * grad
    return x

```

```

x = np.array( [3.0, 4.0] )
print(gradient_descent(loss_func, x, lr=0.00001, step_num=100)) #[2.99400594 3.99200791]

```

설현 사진을 mnist 신경망에 넣으면 숫자가 무엇으로 나오는지?

file : 설현.py

#실습할 때 참고

#0310 #워킹 디렉토리 필요 파일 & 폴더 목록

.ipynb_checkpoints	2021-03-10 오후 1:50	파일 폴더	
__pycache__	2021-03-10 오후 2:32	파일 폴더	
data10	2021-03-10 오후 2:29	파일 폴더	
dataset	2021-03-10 오후 2:31	파일 폴더	
resize	2021-03-10 오후 2:30	파일 폴더	
common.py	2021-03-09 오후 4:41	Python File	1KB
sample_weight.pkl	2017-02-26 오후 2:20	PKL 파일	178KB
Untitled.ipynb	2021-03-10 오후 2:00	IPYNB 파일	8KB
설현.ipynb	2021-03-10 오후 2:51	IPYNB 파일	114KB

문제82. 다른 설현 사진을 신경망에 넣으면 어떤 숫자가 나오는가?

file : 설현.py

1~10 사이의 숫자 반환된다

□ 편미분을 하는 gradient_descent function 으로 2층 신경망 구현

저자가 집적 만들어온 소스 코드 내 common 이라는 폴더가 있다. 이 폴더를 jupyter working directory 에 위치시킨다.

기존에 있던 common.py 는 common7.py 로 이름을 변경하시오!

common 폴더(패키지) 안에는 funtions.py 도 있고 gradient.py 도 존재한다.

funtions.py ---> 신경망에 필요한 함수

gradient.py ---> 편미분하는 함수 numerical_gradient 함수

●● 2층 신경망 만들기

[code]

```
import sys, os
```

```
sys.path.append(os.pardir) #부모 디렉토리의 파일에 접근가능하도록 설정
```

```
import numpy as np
```

```
from common.functions import softmax, cross_entropy_error
```

```
from common.gradient import numerical_gradient
```

```
##test
```

```
z = np.array([0.1, 0.9])
```

```
print(softmax(z)) #[0.31002552 0.68997448]
```

예제1. 가중치 행렬을 2x3 으로 랜덤으로 숫자를 생성하기

```
import numpy as np
```

```
print(np.random.randn(2,3))
```

```
[[ 0.3183936 -0.276435  1.24411778]
 [ 0.8529612 -0.6737312  0.95753502]]
```

예제2. 아래의 입력 데이터를 1x2 행렬로 만들고 위에서 만든 가중치 행렬과 내적하시오!

```
x = np.array( [0.6, 0.9] ) #example:설현사진
```

```
W = np.random.randn(2,3)
```

```
print(np.dot(x, W)) #[ 0.71873875 -0.16287323  0.521407 ]
```

예제3. 설현 사진(x)을 입력해서 가중치 행렬과 내적해서 나온 결과 행렬을 softmax 함수에 넣어서 확률을 출력하시오!

```
import numpy as np
```

```
from common.functions import softmax, cross_entropy_error
```

```
from common.gradient import numerical_gradient
```

```
x = np.array( [0.6, 0.9] ) #example:설현사진
```

```
W = np.random.randn(2,3)
```

```
a = np.dot(x, W)
```

```
y_hat = softmax(a)
```

```
print(y_hat.sum())
```

```
y_hat #array([0.50072376, 0.13621669, 0.36305955])
```

예제4. 위에서 출력된 확률 벡터를 정답과 함께 오차함수에 넣어서 오차를 구하시오!

```
import numpy as np
```

```
from common.functions import softmax, cross_entropy_error
```

```
from common.gradient import numerical_gradient
```



```

np.random.seed(1) #seed
x = np.array( [0.6, 0.9] ) #example:설현사진
W = np.random.randn(2,3)
t = np.array([0,1,0]) #label

```

```

a = np.dot(x, W)
y_hat = softmax(a)

```

```

loss = cross_entropy_error(y_hat, t)
loss #0.5476576361865481

```

예제5. 비용함수를 생성해서 비용함수를 미분하시오!

```

f = lambda w : net.loss(x, t) #비용함수 생성 #w값이 입력값으로 들어간다
dW = numerical_gradient(f, net.W)
print(dW) #기울기 출력

```

●● 예제6. 2층 신경망 전체 코드 구현

```

[code]
import sys, os
sys.path.append(os.pardir) #부모 디렉토리의 파일에 접근가능하도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient

class simpleNet:
    def __init__(self): #설계도(클래스), 객체 생성시 자동실행:init
        self.W = np.random.randn(2,3) #랜덤으로 가중치 행렬 생성

    def predict(self, x): #예측함수
        return np.dot(x, self.W)

    def loss(self, x, t):
        z = self.predict(x) #설현 사진을 넣어서 z 값을 출력
        y = softmax(z) #z값을 받아서 확률벡터를 생성
        loss = cross_entropy_error(y, t) #확률벡터와 정답(label)로 오차 정의
        return loss

x = np.array([0.6,0.9])
t = np.array([0,0,1])

net = simpleNet() #클래스(2층 신경망 설계도) #클래스 생성시 객체 생성(-->net으로 변수지정)

f = lambda w : net.loss(x,t) #비용함수 생성

dW = numerical_gradient(f, net.W) #기울기
print(dW)

```

가중치가 2x3 이므로 기울기도 2x3 으로 나와야 가중치에서 기울기를 뺄 수 있다.

딥러닝 책 내용

- 1장. numpy 사용법 : 신경망에서 예측하는 모든 수학식이 행렬계산 ---> numpy 모듈에 대한 이해가 중요
- 2장. 퍼셉트론
- 3장. 3층 신경망 구현
- 4장. 2층 신경망 구현

입력값 ----> 2층 신경망 ----> 기울기

↓ ↓ ↓

<Simplenet class>

(__init__ func : 가중치 행렬을 생성

predict func : 입력값을 받아 가중치 행렬과 내적인 결과 출력

loss func :)

문제83. 아래의 입력데이터와 target(정답)을 simplenet 에 입력하고 오차를 출력하시오!

x = np.array([0.8, 0.2]) ----> 입력데이터

t = np.array([0,0,1]) ----> 정답 1개

#단순적용

문제84. simpleNet() 클래스에 있는 가중치 행렬 W를 출력하시오!

```
import sys, os
sys.path.append(os.pardir) #부모 디렉토리의 파일에 접근가능하도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient
```

```
class simpleNet:
    def __init__(self): #설계도(클래스), 객체 생성시 자동실행:init
        self.W = np.random.randn(2,3) #랜덤으로 가중치 행렬 생성

    def predict(self, x): #예측함수
        return np.dot(x, self.W)

    def loss(self, x, t):
        z = self.predict(x) #설현 사진을 넣어서 z 값을 출력
        y = softmax(z) #z값을 받아서 확률벡터를 생성
        loss = cross_entropy_error(y, t) #확률벡터와 정답(label)로 오차 정의
        return loss
```

```
x = np.array([0.6,0.9])
```

```
t = np.array([0,0,1])
```

```
net = simpleNet()
```

```
print(net.W)
```

문제85. simpleNet() 클래스에 있는 predict 함수에 아래의 입력 데이터를 넣고 결과를 출력하시오!

```
import sys, os
sys.path.append(os.pardir) #부모 디렉토리의 파일에 접근가능하도록 설정
import numpy as np
```

```

from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient

class simpleNet:
    def __init__(self): #설계도(클래스), 객체 생성시 자동실행:init
        self.W = np.random.randn(2,3) #랜덤으로 가중치 행렬 생성

    def predict(self, x): #예측함수
        return np.dot(x, self.W)

    def loss(self, x, t):
        z = self.predict(x) #설현 사진을 넣어서 z 값을 출력
        y = softmax(z) #z값을 받아서 확률벡터를 생성
        loss = cross_entropy_error(y, t) #확률벡터와 정답(label)로 오차 정의
        return loss

x = np.array([0.9, 0.6])
t = np.array([0,0,1])

net = simpleNet()
res = net.predict(x)
print(res)

```

필기체 데이터를 학습시키는 2층 신경망 전체 full code p.137

file : ch04/two_layer_net.py

[code]

```

# coding: utf-8
import sys, os
sys.path.append(os.pardir) # 부모 디렉터리의 파일을 가져올 수 있도록 설정
from common.functions import *
from common.gradient import numerical_gradient

class TwoLayerNet:

    def __init__(self, input_size, hidden_size, output_size, weight_init_std=0.01): #weight_init_std=0.01 ---> 가중치초기값 설정 파라미터
        # 가중치 초기화
        self.params = {} #weight, bias 를 딕셔너리로 구현하기 위한 bin dictionary 생성
        self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden_size)
        self.params['b1'] = np.zeros(hidden_size)
        self.params['W2'] = weight_init_std * np.random.randn(hidden_size, output_size)
        self.params['b2'] = np.zeros(output_size)

    def predict(self, x):
        W1, W2 = self.params['W1'], self.params['W2']
        b1, b2 = self.params['b1'], self.params['b2']

        a1 = np.dot(x, W1) + b1

```

```

z1 = sigmoid(a1) #활성함수 통과
a2 = np.dot(z1, W2) + b2
y = softmax(a2) #출력함수 통과

return y

# x : 입력 데이터, t : 정답 레이블
def loss(self, x, t):
    y = self.predict(x)

    return cross_entropy_error(y, t)

def accuracy(self, x, t):
    y = self.predict(x)
    y = np.argmax(y, axis=1) #출력함수 softmax으로 각 y의 값들을 확률로 변환된 것들 중 가장 높은값 반환 #y_hat
    t = np.argmax(t, axis=1)

    accuracy = np.sum(y == t) / float(x.shape[0])
    return accuracy

# x : 입력 데이터, t : 정답 레이블
def numerical_gradient(self, x, t):
    loss_W = lambda W: self.loss(x, t)

    grads = {}
    grads['W1'] = numerical_gradient(loss_W, self.params['W1'])
    grads['b1'] = numerical_gradient(loss_W, self.params['b1'])
    grads['W2'] = numerical_gradient(loss_W, self.params['W2'])
    grads['b2'] = numerical_gradient(loss_W, self.params['b2'])

    return grads

def gradient(self, x, t):
    W1, W2 = self.params['W1'], self.params['W2']
    b1, b2 = self.params['b1'], self.params['b2']
    grads = {}

    batch_num = x.shape[0]

    # forward
    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    y = softmax(a2)

    # backward
    dy = (y - t) / batch_num
    grads['W2'] = np.dot(z1.T, dy)
    grads['b2'] = np.sum(dy, axis=0)

    da1 = np.dot(dy, W2.T)

```

```

dz1 = sigmoid_grad(a1) * da1
grads['W1'] = np.dot(x.T, dz1)
grads['b1'] = np.sum(dz1, axis=0)

return grads

```

문제86. 오늘 만든 **simpleNet** 클래스를 객체화시켜서 실행하는데, 이 신경망에 설현사진을 입력할 수 있도록 가중치 행렬의 **shape** 을 수정하고 설현사진을 입력해서 확률벡터를 출력하시오! (**resize = 28x28**)

[code]

```

## Load img & resize(28*28)
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import os

path = "C:\\Users\\82103\\OneDrive\\Jupyter Notebook\\data10"
file_list = os.listdir(path)
for k in file_list:
    img = cv2.imread(path + '\\' + k)
    width, height = img.shape[:2]
    resize_img = cv2.resize(img, (28, 28), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('C:\\Users\\82103\\OneDrive\\Jupyter Notebook\\resize' + k, resize_img)

plt.imshow(resize_img)
plt.show()
print(resize_img.shape)  #(28, 28, 3)

## 흑백사진으로 변환 (28, 28)
j= 'C:\\Users\\82103\\OneDrive\\Jupyter Notebook\\resize\\a.jpg'
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

img = mpimg.imread(j)
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()

## flatten
x = gray.flatten()
print(gray.shape)  #(28, 28)
print(x.shape)  #(784, )

```

```

## Simplenet
import sys, os
sys.path.append(os.pardir) #부모 디렉토리의 파일에 접근가능하도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient

np.random.seed(1) #seed
class simpleNet:
    def __init__(self): #설계도(클래스), 객체 생성시 자동실행:init
        self.W = np.random.randn(784,5) #랜덤으로 가중치 행렬 생성

    def predict(self, x): #예측함수
        return np.dot(x, self.W)

    def loss(self, x):
        z = self.predict(x) #설현 사진을 넣어서 z 값을 출력
        y = softmax(z) #z값을 받아서 확률벡터를 생성
        return z,y

## class 생성 및 결과 출력
net = simpleNet()
res1, res2 = net.loss(x)

print(res1) #[-3390.63259455 2859.0658698 1940.27618646 6911.67836448 -1202.64899989]
print(res2) #[0. 0. 0. 1. 0.] #확률값

###Softmax 함수는 인자값의 자리수가 커지면 0 또는 1만 반환하는 형태로 전환되는 것 같습니다
## Testing ---> softmax func
res3 = np.array([-1,2,3,-4,5])
print(softmax(res3)) #[2.08697568e-03 4.19180271e-02 1.13945011e-01 1.03904401e-04 8.41946081e-01]

res4 = res3*100
print(softmax(res4)) #[2.65039655e-261 5.14820022e-131 1.38389653e-087 0.00000000e+000 1.00000000e+000]

res5 = res3*10000
print(softmax(res5)) #[0. 0. 0. 0. 1.]

#####endLine=====

```

##0311

2021년 3월 1일 월요일 오후 4:52

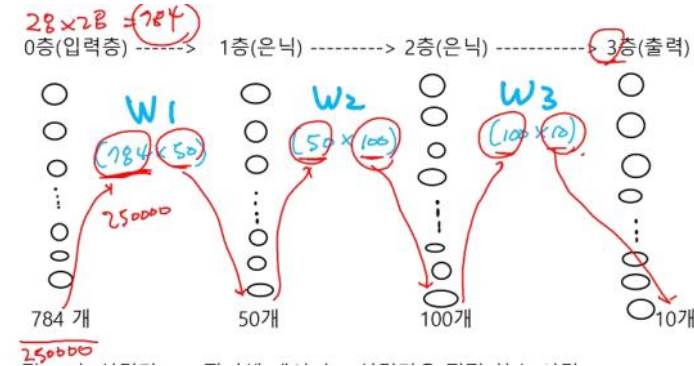
■ Review

딥러닝이 무엇인가요? 답변을 세련되고 정중하게 대답할 수 있도록 생각하기

1장. numpy를 왜 사용해야 하는지?

2장. 퍼셉트론

3장. 3층 신경망 --> 저자가 만들어 온 가중치를 셋팅해서 필기체를 분류하는 신경망



4장. 2층 신경망 --> 필기체 데이터로 신경망을 직접 학습 시킴 (p 143)

--> 설현과 수지사진을 분류하는 신경망

5장. 오차 역전파

6장. 언더피팅과 오버피팅을 해결하는 방법들

7장. CNN

8장. 최신 기술들 소개 --> tensorflow, pytorch로 신경망 구현

□ (필기체 데이터를 인식하게끔 학습시키는) 2층 신경망 구현하기 p137-144

1. 4장 전체코드를 먼저 수행합니다.

2. debugging : file : 4장 전체코드.ipynb

문제87. TwoLayerNet class의 메소드 중 하나인 numerical_gradient 함수를 실행해서 기울기를 출력하시오! (입력데이터 : 100개)

```
network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
network.numerical_gradient(x_train[:100], t_train[:100])
```

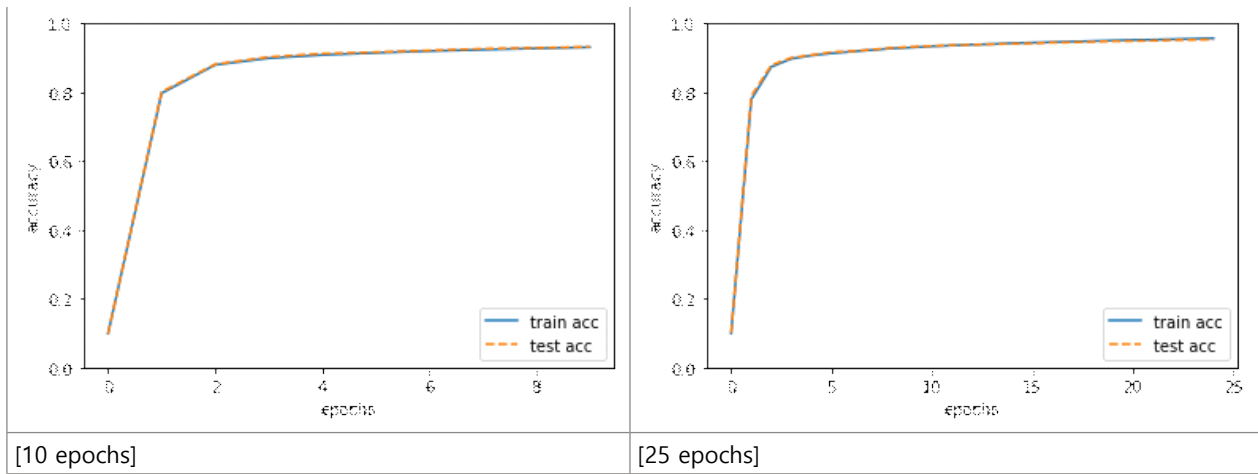
수치미분

TwoLayerNet 의 numerical_gradient 는 수치미분을 활용하는데 계산 cost 가 매우 높다.

5장에서는 오차역전파를 사용해서 가중치 기울기를 찾고 가중치와 bias를 갱신한다.

코드 중 gradient 메소드는 5장에서 사용하는 오차역전파 함수이다.

문제88. 10에폭 ---> 20에폭으로 코드 수정 후 그래프 첨부



● 가중치 파일로 생성하기

"pickle" 을 이용해서 .pkl 파일을 생성

예제1. pickle 파일을 생성하는 예제

```
import pickle
params = [3.1, 4.3, 5.3]
with open( "c:\\Users\\wicr\\OneDrive\\Jupyter Notebook\\mnist_weight7.pkl", "wb") as f:
    pickle.dump(params, f)
```

문제89. 4장에서 만든 2층 신경망의 가중치와 바이어스를 pickle file로 내리시오!

```
with open( "c:\\Users\\wicr\\OneDrive\\Jupyter Notebook\\mnist_weight.pkl", "wb") as f:
    pickle.dump(network.params, f)
# network.params : 가중치 딕셔너리
```

문제90. 3장에서 사용한 3층 신경망 코드를 2층 신경망으로 변경하고

위의 pickle 파일을 세팅해서 필기체를 분류할 수 있는 신경망을 만드시오!

```
import pickle
with open( "c:\\Users\\wicr\\OneDrive\\Jupyter Notebook\\mnist_weight.pkl", "wb") as f:
    pickle.dump(network.params, f)
# network.params : 가중치 딕셔너리

import numpy as np
from common.functions import *
from dataset.mnist import load_mnist
import pickle

def init_network():
    with open("c:\\Users\\wicr\\OneDrive\\Jupyter Notebook\\mnist_weight.pkl", "rb") as f:
        network = pickle.load(f)
    return network

# 1. 데이터를 불러옵니다.
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True, one_hot_label=False)
```



```
# 2. 가중치와 바이어스 값을 불러옵니다. (저자가 미리 학습 시킨 가중치와 바이어스)
network = init_network()
w1, w2 = network['W1'], network['W2']
b1, b2 = network['b1'], network['b2']

# 3. 신경망을 구성합니다.
# 0층
x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.

# 1층
y = np.dot(x, w1) + b1
y_hat = sigmoid(y)

# 2층
z = np.dot(y_hat, w2) + b2
z_hat = softmax(z)
a = np.argmax(z_hat, axis=1) # axis=1 이 축 # 예측값
b = t_train[0:100] # 실제 정답

print('총 ', len(a), '중에서 ', sum(a==b), '개 맞추었습니다') # 총 100 중에서 97 개 맞추었습니다
```

● 4장 요약

1. 신경망을 학습되게 하려면 필요한 것 : 오차함수(항등함수, 교차엔트로피, MSE 등), 수치미분, 미니배치
 2. 미분함수를 파이썬으로 구현
 3. 편미분함수를 파이썬으로 구현
 4. 2층 신경망을 구현(학습할 수 있는 신경망)
 5. 신경망의 가중치를 pickle file 로 받고, 신경망 학습때 로드 및 적용
- # 수치미분으로 신경망을 학습하면 너무 느려서 학습을 할 수 없다.
이를 해결하기 위한 방법으로 오차 역전파를 이용한다.

■ 6장. 오차역전파 p.147

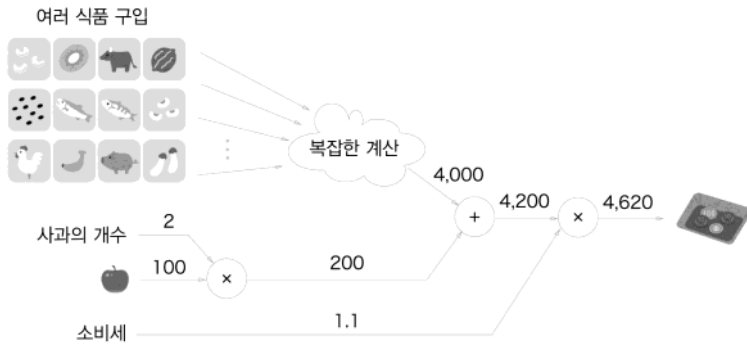
"수치미분의 계산 코스트가 너무 커서 오차 역전파를 이용해서 기울기 최소값을 찾는다"

계산그래프의 장점

국소적 계산을 할 수 있다.

국소적 계산 : 전체에서 어떤 일이 벌어지던 상관없이 자신과 관련된 정보만으로 원하는 결과를 얻어낼 수 있는 계산 방식

[그림 5-4]



4000원이라는 숫자가 어떻게 계산되었느냐와는 상관없이 사과가 어떻게 200원이 되었는지만 고려하는 것이 국소적 계산이다.

계산그래프로 문제해결 이유? p.151

계산 전체 과정의 복잡도와 상관없이 각 노드에서 단순한 계산에 집중하여 문제를 단순화시킬 수 있다.

큰 문제를 해결하는 방법은 결국 작은 문제들이 여러개로 묶여있기 때문에 작은 단위부터 해결해 나가는 것이 최선(다이나믹 프로그래밍)

과정	입력값	→	활성화함수	→	출력층함수	→	오차함수	
종류			sigmoid		softmax		교차엔트로피	(분류)
			ReLU		항등함수		평균제곱오차(MSE)	(회귀)

최종적으로 산출하고자 하는 것 : 오차가 가장 적은 가중치

가중치(w_1)에 변화가 생겼을 때 오차는 얼마나 달라지는지?

↓

사과값이 '아주 조금' 올랐을 때 '지불금액'이 얼마나 증가하는지 알고싶다면?

사과값이 아주 조금 올랐을 때 지불금액이 얼마나 증가하는지 알고 싶다면 아래와 같이 편미분을 하면 된다.

∂ 지불금액 / ∂ 사과값 ----> 미분값

[그림5-4] 계산 그래프의 역전파를 이용하면 위 계산을 할 수 있다.

[그림5-1]

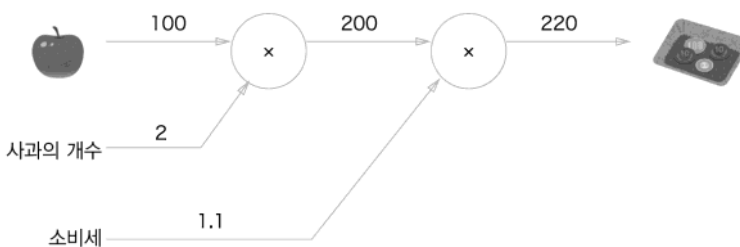


예제1. 현빈군은 슈퍼에서 1개에 100원인 사과를 2개를 샀다. 이때의 지불금액을 구하시오! (tax=10%)

위 예제에 대한 답이 그림5-1 이다.

이것을 계산그래프로 나타내면 그림 5-2 와 같다

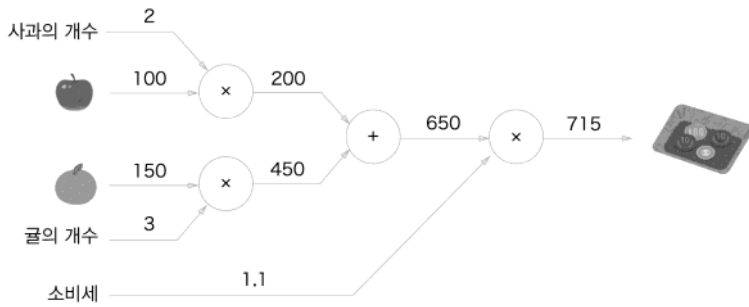
[그림 5-2]



예제2. 현빈군은 슈퍼에서 사과를 2개, 귤을 3개 샀다. 사과는 1개에 100원, 귤은 1개에 150원이다.

소비세가 10% 일 때, 지불금액을 구하시오!

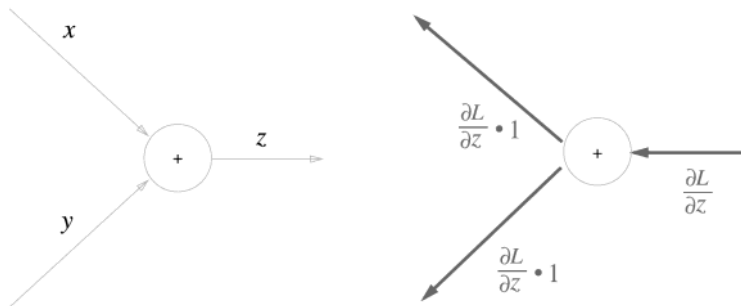
[그림 5-3]



총 지불금액은 가장 우측에 위치한 수인 715원이 된다.

계산그래프로 표현된 모습

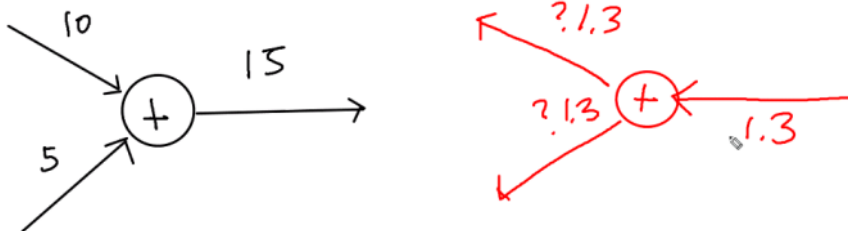
[그림 5-9] 덧셈 계산 그래프



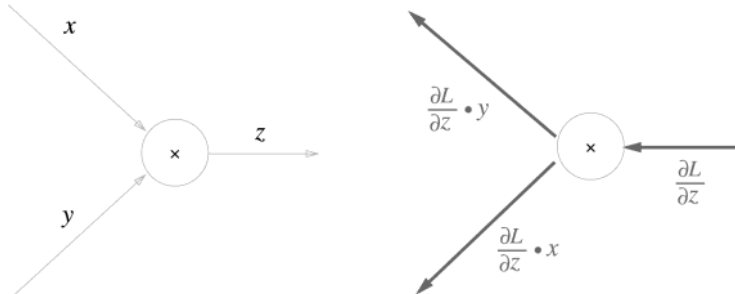
좌(순전파), 우(역전파)

덧셈의 역전파는 상위에서 흘러온 값*1 이므로 그대로 흘러보낸다

문제91. 아래의 덧셈노드 그래프의 순전파 값과 역전파 값을 적으시오!



[그림 5-12] 곱셈 계산 그래프



좌(순전파), 우(역전파)

곱셈의 역전파는 '상위에서 흘러온 값*반대편 값' 으로 계산된다.

식 예시

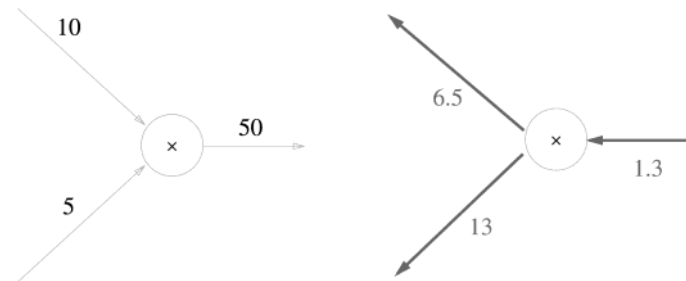
$$p = f(x, y) = x \cdot y$$

$$\partial p / \partial y = x$$

$$\partial p / \partial x = y$$

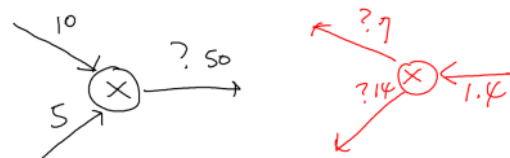
위 식에서 순전파에서 곱했던 서로 반대편 값을 곱한것이 역전파 과정임을 알 수 있다.

[그림 5-13] 곱셈의 역전파 예시



좌(순전파), 우(역전파)

문제92. 아래의 곱셈 노드의 순전파값과 역전파 값을 각각 적으시오!



문제93. 곱셈계층을 파이썬으로 구현하시오! p.161

```
class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y):
        self.x = x
        self.y = y
        out = x*y
        return out

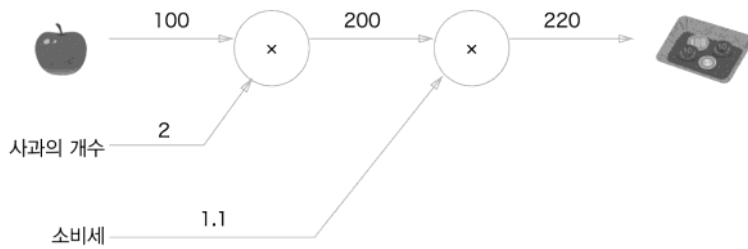
    def backward(self, dout):
        dx = dout *self.y #x와 y를 바꾸기
        dy = dout *self.x
        return dx, dy
```

문제94. 위에서 만든 곱셈 클래스를 객체화시켜서 아래의 사과가격을 구하시오!

```
apple = 100
apple_num = 2

mul_apple_layer = MulLayer()
apple_price = mul_apple_layer.forward(apple, apple_num)
print(apple_price) #200
```

문제95. 곱셈계층 클래스를 객체화 시켜서 아래의 그림 5-2를 계산하시오!



apple = 100

apple_num = 2

tax = 1.1

```
mul_apple_layer = MulLayer()
```

```
mul_tax_layer = MulLayer()
```

```
apple_price = mul_apple_layer.forward(apple, apple_num)
```

```
price = mul_tax_layer.forward(apple_price, tax)
```

```
print(price) #220.00000000000003 #220
```

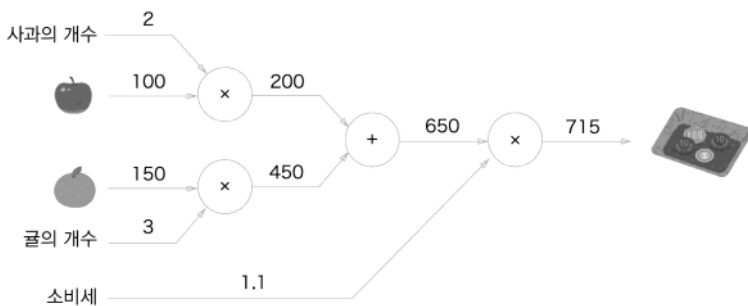
문제96. 덧셈 계층 클래스를 파이썬으로 구현하시오! p.163

```
class AddLayer:
    def __init__(self):
        pass

    def forward(self, x, y):
        out = x+y
        return out

    def backward(self, dout):
        dx = dout*1
        dy = dout*1
        return dx, dy
```

문제97. 위에서 만든 곱셈 클래스와 덧셈 클래스를 이용해서 그림 5-3의 계산 그래프를 구현하고 최종값을 계산하시오!



apple = 100

apple_num = 2

orange = 150

orange_num = 3

tax = 1.1

```
mul_apple_layer = MulLayer()
```

```
mul_orange_layer = MulLayer()
```

```
add_apple_orange_layer = AddLayer()
```

```
mul_tax_layer = MulLayer()
```

```
# Forward
apple_price = mul_apple_layer.forward(apple, apple_num)
orange_price = mul_orange_layer.forward(orange, orange_num)
all_price = add_apple_orange_layer.forward(apple_price, orange_price)
price = mul_tax_layer.forward(all_price, tax)
print(price) #715.0000000000001 #715

#Backward
dprice = 1
dall_price, dtax = mul_tax_layer.backward(dprice)
dapple_price, dorange_price = add_apple_orange_layer.backward(dall_price)
dorange, dorange_num = mul_orange_layer.backward(dorange_price)
dapple, dapple_num = mul_apple_layer.backward(dapple_price)
print(dapple_num, dorange_num, dapple, dorange, dtax, sep='\n')
110.00000000000001
165.0
2.2
3.3000000000000003
650
```

□ Relu 함수를 위한 계산 그래프

[그림 5-18]



Relu class 생성시 중요문법 2가지

1. copy module
2. `x[x <= 0]`

copy module 사용법**

[예제1]

```
a = [1,2,3]
b = a #b는 a와 서로 같은 곳을 바라본다 #b는 a와 같은 memory 주소를 가진다 #b는 별도의 객체가 아니다
print(a) #[1,2,3]
print(b) #[1,2,3]
```

```
a[1] = 6
print(a) #[1,6,3]
print(b) #[1,6,3]
```

[예제2]

```
a = [1,2,3]
b = a.copy() #b와 a는 서로 같은 곳을 바라보지 않는다 #a와 독립적인 memory 주소를 가진다 #b는 별도의 객체가 된다
print(a) #[1,2,3]
print(b) #[1,2,3]
```

```
a[1] = 6
print(a) #[1,6,3]
print(b) #[1,2,3] #a 값이 바뀌었지만 그것에 영향을 받기 전에 copy() 로 지정되었기 때문에 [1,2,3] 의값을 반환
```

[예제3] copy 모듈을 임포트하고 copy(a) 형식으로 지정해줄 수도 있다.

```
from copy import copy
a = [1,2,3]
b = copy(a)
print(a) #[1,2,3]
print(b) #[1,2,3]
```

```
a[1] = 6
print(a) #[1,6,3]
print(b) #[1,2,3]
```

x[x<=0] 의 의미**

'Relu' funtion 은 순전파 때 보냈던 신호를 기억하고 있어야 하기 때문에 위 개념이 필요하다.

[예제1]

```
import numpy as np
x = np.array( [[1.0, -0.5], [-2.0, 3.0]] )
print(x, end='\n\n')
mask = (x<=0)
print(mask)
[[ 1. -0.5]
 [-2.  3. ]]

[[False  True]
 [ True False]]

# true 자리에서(음수) 신호가 안 가는 것을 mask 로 기억
# 이를 위 copy 함수와 조합하여 relu 함수를 만들 수 있다.
```

[예제1-2]

```
out = x.copy()
out[mask] = 0
print(out)
[[1.  0.]
 [0.  3.]]
```

☆문제98. p.166 의 Relu 클래스를 생성하시오!

```
class Relu:
    def __init__(self):
        self.mask = None

    def forward(self, x):
        self.mask = (x <= 0)
        out = x.copy()
        out[self.mask] = 0
        return out

    def backward(self, dout):
        dout[self.mask] = 0
```

```
dx = dout
return dx
```

```
import numpy as np
x = np.array([1.0, -0.5, -2.0, 3.0])
x = x.reshape(2, -1)
relu = Relu()
print(relu.forward(x), end='\n\n')
```

```
dout = np.array([[2.0, 3.0], [-3.0, -4.0]])
print(relu.backward(dout))
[[1.  0.]
 [0.  3.]]

[[ 2.  0.]
 [ 0. -4.]]
```

```
#####endLine=====
```


##0315

2021년 3월 1일 월요일 오후 4:52

■ 최종 포트폴리오에 관해

주제 : 자유롭게 하되 너무 어렵지 않게

조구성 : 지금 조 | 새로운 조 생성(최소 2명 이상)

UI : R shine >>> web browser

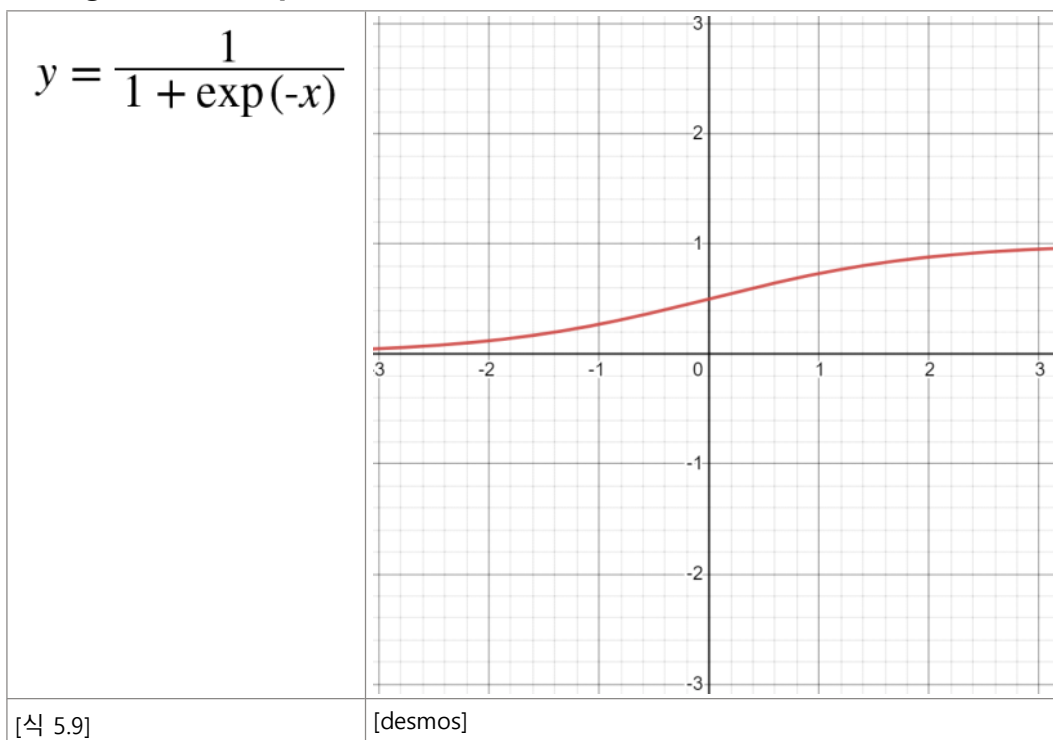
■ Review

1. numpy 사용법
2. 퍼셉트론
3. 3층 신경망
4. 2층 신경망
5. 오차 역전파를 이용한 2층 신경망
6. 오버피팅과 언더피팅을 막는 방법들
7. CNN
8. 딥러닝의 역사(tensorflow, 파이토치)

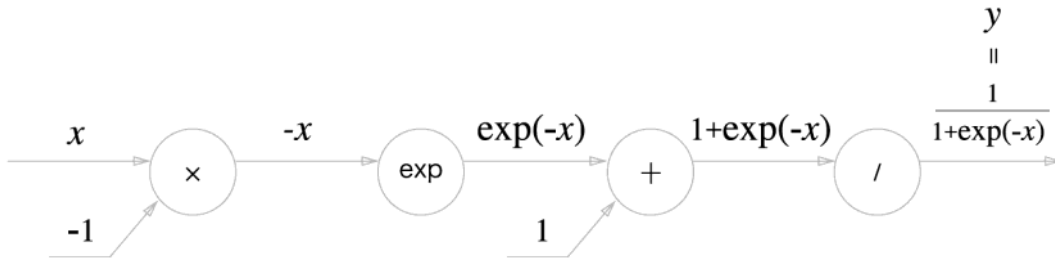
■ 포트폴리오 (DNN)

1. 직접 스크롤링한 데이터를 분류하는 신경망 활용 홈페이지
(파이썬 스크롤링 + 신경망 구현)
2. Object Detection (영상)
(영상에 적용한 결과를 YouTube 등에 업로드)
3. RNN (시계열)
(장기 개인 프로젝트)

□ Sigmoid 계층(p. 167)



[그림 5-19] 순전파



예제1. sigmoid func의 순전파를 파이썬 코드로 구현하시오! (역전파도 포함) p.170 [code] 참고

[code]

```
class sigmoid:
    def __init__(self):
        self.out = None

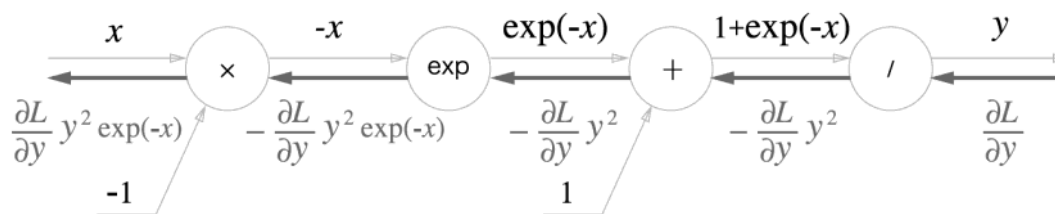
    def forward(self, x):
        out = 1 / (1+np.exp(-x))
        self.out = out
        return out

    def backward(self, dout):
        dx = dout * (1.0 - self.out) * self.out
        return dx
```

문제99. sigmoid class를 객체화시켜서 아래의 데이터를 **forward** 함수로 순전파 작업을 수행하시오!

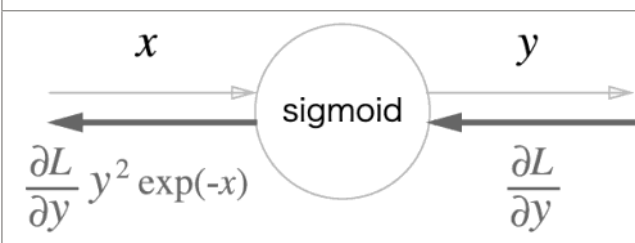
```
import numpy as np
x = np.array([24, 32, 4]) #input data
sig = sigmoid()
sig.forward(x) #array([1.      , 1.      , 0.98201379])
```

[그림 5-20] 역전파

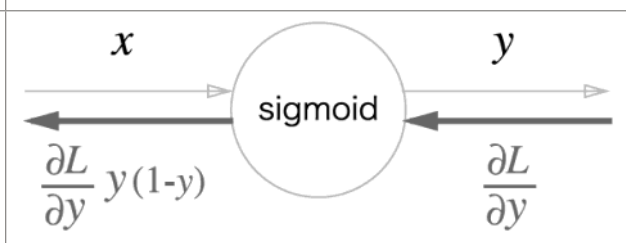


$f(x) = 1/x$ 의 미분을 구해보자
f prime = $-1/x^2$

[그림 5-21]



[그림 5-21]



코드는 위에서 구현한 sigmoid 클래스 참고
마지막 $y(1-y)$ 를 기억

문제100. 위에서 만든 **sigmoid class**를 객체화시켜서 순전파와 역전파를 각각 구현하시오!

```
import numpy as np

# Forward
x = np.array( [[1.0, -0.5], [-2.0, 3.0]] )
sig = sigmoid()
print(sig.forward(x))
[[0.73105858 0.37754067]
 [0.11920292 0.95257413]]

# Backward
dout = np.array( [[2.0, 3.0], [-3.0, -4.0]] )
print(sig.backward(dout))
[[ 0.39322387  0.70501114]
 [-0.31498076 -0.18070664]]

# 순전파와 역전파의 노드는 같은 형상이어야 한다
# 순전파일때는 입력값이 흘러가고 시그모이드는 0~1, 소프트맥스를 통과하면 확률로 출력
# 역전파일때는 오차가 흘러온다. 오차가 흘러오면서 미분대신에 만든 도함수와 오차가 같이 계산
# 최종적으로 가중치에 오차 반영 (가중치 갱신)
```

■ TensorFlow 2.0

□ 환경설정

[데이터의 증가와 기계학습 | \[쉬움주의\] 케라스 가상 환경 만들기 - Daum 카페](#) #최근 사용하는 버전이 2.0 이기 때문에 2.0으로 수업진행
3번은 시간이 오래걸리므로 시간 여유가 있을 때 설치
8번은 안될때 solution

가상환경을 설정해주는 이유

1. 일반사진을 고해상도로 변환(환경구성1)
2. 옥주현 ai 를 만들고 싶다(환경구성2)

두 가지 환경을 별도로 분리해서 사용하는것이 바람직하며, 이를 가능하게 해 주는 것이 가상환경 설정이다.

□ 실습 : sigmoid

TensorFlow 와 Python 날코딩의 차이

Python	TensorFlow
sigmoid 함수를 직접 생성	google 의 개발자가 sigmoid func 개발

[tensorflow code]

```
import tensorflow as tf
x = tf.constant([24.0, 32.0, 5.0]) #3개의 상수값을 생성(float type)
tf.math.sigmoid(x)
# <tf.Tensor: shape=(3,), dtype=float32, numpy=array([1.      , 1.      , 0.9933071], dtype=float32)>
```

문제101. 아래의 2x2 행렬을 텐서플로의 시그모이드에 흘려보내고 결과를 보시오!

```
dout = np.array( [[2.0, 3.0], [-3.0, -4.0]] )
tf.math.sigmoid(dout)
<tf.Tensor: shape=(2, 2), dtype=float64, numpy=
array([[0.88079708, 0.95257413],
       [0.04742587, 0.01798621]])>
```

□ 5장의 오차 역전파를 구현할 파이썬 코드 구현

1. relu
2. sigmoid
3. Affine
4. softmax
5. Loss func

Affine 계층 p. 170

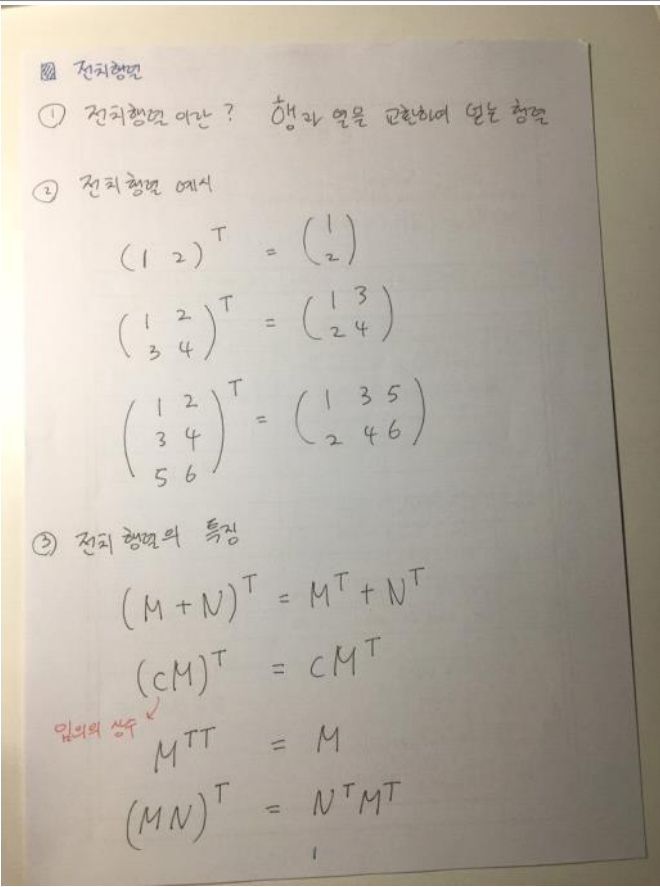
"신경망의 순전파시 수행하는 행렬의 내적을 기하학에서는 어파인 변환이라고 한다"

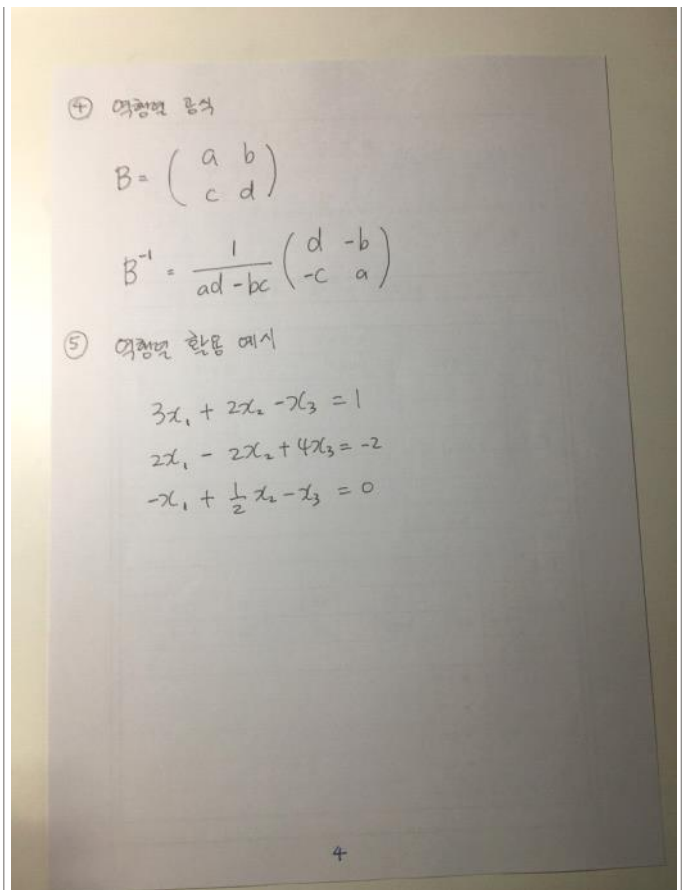
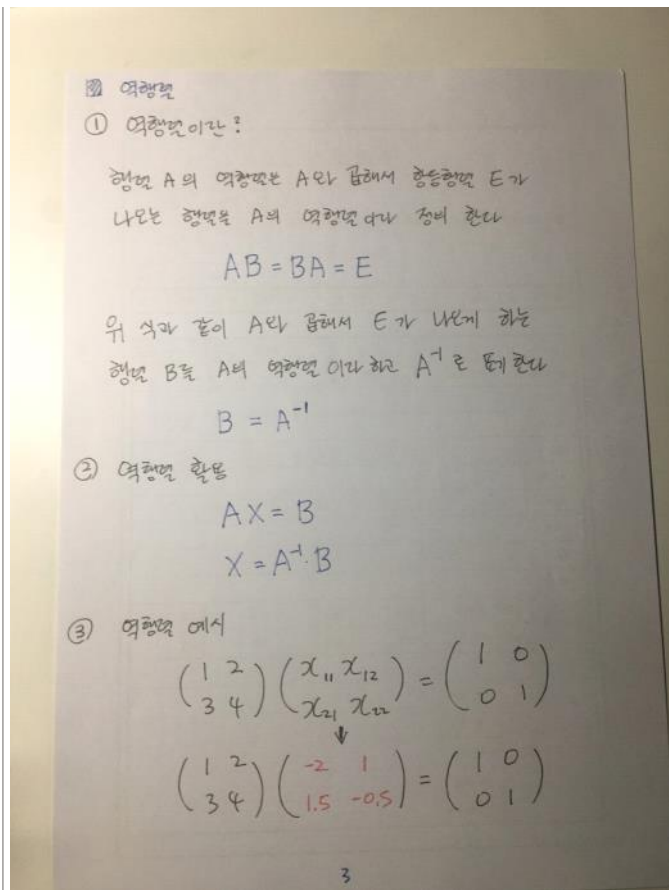
신경망에서 입력값과 가중치의 내적의 합에 바이어스를 더하는 층을 Affine(완전연결) 계층이라고 하여 구현한다

지금까지의 계산 그래프는 노드 사이에 '스칼라 값' 이 흘렀다

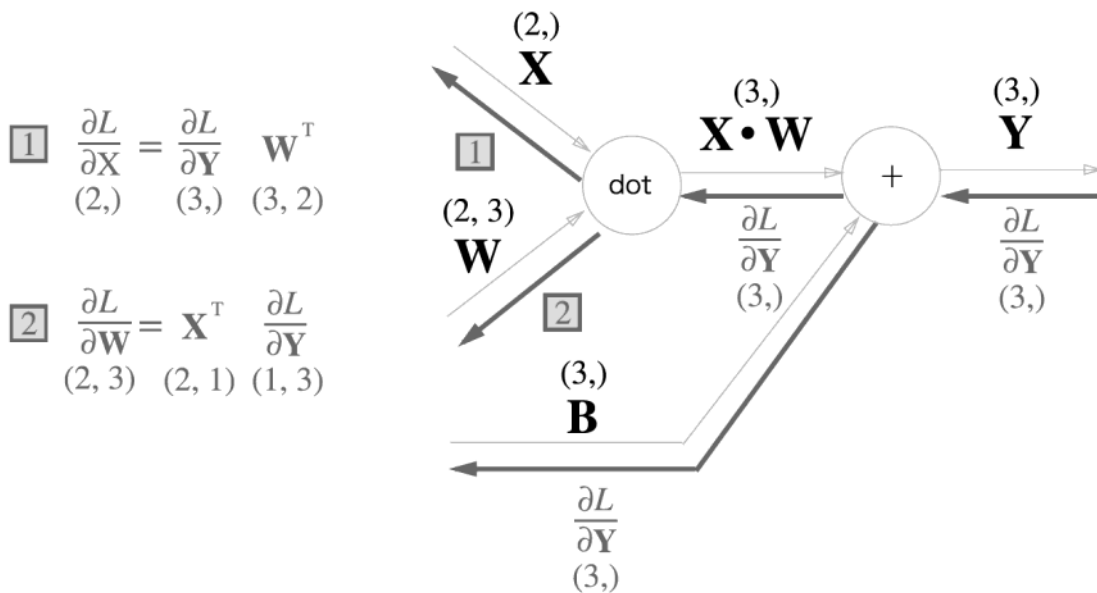
이번에는 '행렬' 이 흐르고 있기 때문에 Affine 계층 구현이 필요하다

Affine 계층을 이해하기 위한 행렬의 종류

<p>전치행렬</p>  <p>전치행렬</p> <p>① 전치행렬이란? 행과 열을 교환하여 얻는 행렬</p> <p>② 전치행렬 예시</p> $(1 \ 2)^T = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$ <p>③ 전치행렬의 특징</p> $(M + N)^T = M^T + N^T$ $(cM)^T = cM^T$ <p>임의의 상수</p> $M^{TT} = M$ $(MN)^T = N^T M^T$	
<p>역행렬 1</p>	<p>역행렬 2</p>



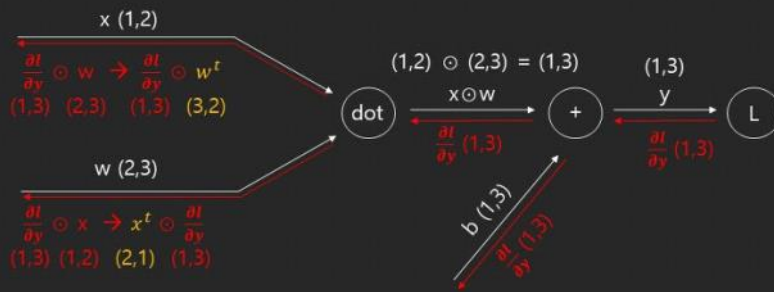
[그림 5-25]



위 그림의 x 데이터를 2차원으로 하여 좀 더 직관적으로 이해할 수 있는 자료
[데이터의 증가와 기계학습 | \[쉬움주의\] Affine 계층 계산그래프 - Daum 카페](#)

“역전파 일때의 행렬의 shape은 순전파 일때의 행렬의 shape와 동일 ”

$$y = x \odot w + b$$



문제102. p.175 에 나오는 Affine 계층 클래스를 생성하시오!

[code]

```
class Affine:
    def __init__(self, W, b):
        self.W = W
        self.b = b
        self.x = None
        self.dW = None
        self.db = None

    def forward(self, x):
        self.x = x
        out = np.dot(x, self.W) + self.b
        return out

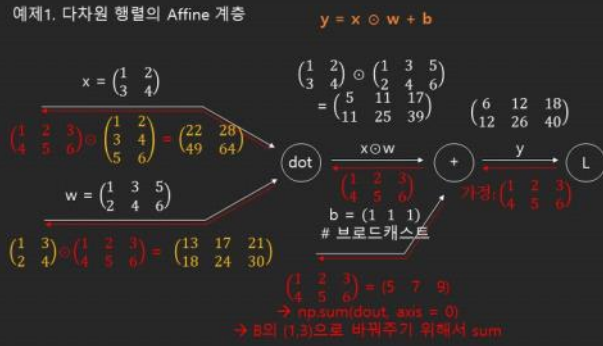
    def backward(self, dout):
        dx = np.dot(dout, self.W.T)
        self.dW = np.dot(self.x.T, dout)
        self.db = np.sum(dout, axis=0)
        return dx

#Testing
import numpy as np
x = np.array([[1,2]])
W = np.array([[1,3,5], [2,4,6]])
b = np.array([1,1,1])

affine1 = Affine(W,b)
affine1.forward(x) #array([[ 6, 12, 18]])
```

문제103. 아래에 나오는 batch 데이터를 신경망에 입력했을때의 순전파를 구하시오!

예제1. 다차원 행렬의 Affine 계층



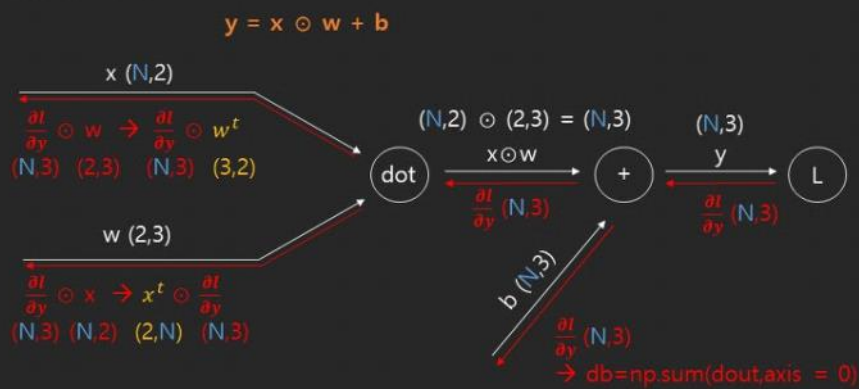
```
import numpy as np
x = np.array([[1,2], [3,4]])
W = np.array([[1,3,5], [2,4,6]])
b = np.array([1,1,1])
```

```
affine1 = Affine(W,b)
affine1.forward(x) #array([[ 6, 12, 18]])
Out[35]: array([[ 6, 12, 18],
               [12, 26, 40]])
```

Affine 계층 역전파

역전파를 계산그래프를 그려서 도함수를 확인하여 도함수로 backward 함수 구현

■ Batch용 Affine 계층



Jacobian Matrix

$$\frac{dL}{dX} = \frac{dL}{dY} \times \frac{dY}{dX} \stackrel{?}{=} \frac{dL}{dY} \times W^T \quad \text{즉, } \frac{dY}{dX} = W^T \text{ 인가?}$$

$$Y = X \cdot W + b$$

$$= [x_0, x_1] \begin{bmatrix} w_0 & w_1 & w_2 \\ w_3 & w_4 & w_5 \end{bmatrix} + b$$

$$= \begin{bmatrix} w_0x_0 + w_1x_1 + w_2x_2 + b & w_3x_0 + w_4x_1 + w_5x_2 + b \\ y_0 & y_1 & y_2 \end{bmatrix}$$

즉, $[y_0, y_1, y_2]$ 인 1×3 행렬을 $[x_0, x_1]$ 1×2 행렬로 따지면

Jacobian matrix 에 의하여

$$\frac{dY}{dX} = \begin{bmatrix} \frac{\partial y_0}{\partial x_0} & \frac{\partial y_0}{\partial x_1} \\ \frac{\partial y_1}{\partial x_0} & \frac{\partial y_1}{\partial x_1} \\ \frac{\partial y_2}{\partial x_0} & \frac{\partial y_2}{\partial x_1} \end{bmatrix} = \begin{bmatrix} \frac{\partial (w_0x_0 + w_1x_1 + b)}{\partial x_0} & \frac{\partial (w_0x_0 + w_1x_1 + b)}{\partial x_1} \\ \frac{\partial (w_3x_0 + w_4x_1 + b)}{\partial x_0} & \frac{\partial (w_3x_0 + w_4x_1 + b)}{\partial x_1} \\ \frac{\partial (w_2x_0 + w_5x_1 + b)}{\partial x_0} & \frac{\partial (w_2x_0 + w_5x_1 + b)}{\partial x_1} \end{bmatrix}$$

$$= \begin{bmatrix} w_0 & w_1 \\ w_3 & w_4 \\ w_2 & w_5 \end{bmatrix} = W^T$$

Jacobian Matrix - 다변수 함수의 열과 미분

영역 $D \subset \mathbb{R}^n$ 에서 정의된 다변수 함수 $f: D \rightarrow \mathbb{R}^m$ 가

각각의 스칼라 함수 $f_1, \dots, f_m: D \rightarrow \mathbb{R}$ 에 대하여

$$f(x_1, x_2, \dots, x_n) = \begin{bmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_m(x_1, \dots, x_n) \end{bmatrix} \text{ 와 같이 정의된다면}$$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \text{ 을 } f \text{ 의 야코비 행렬이라 한다.}$$

example) 극좌표계 $x = r \cos \theta, y = r \sin \theta \quad / \quad x(r, \theta), y(r, \theta)$

$$\frac{\partial(x, y)}{\partial(r, \theta)} = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos \theta & -r \sin \theta \\ \sin \theta & r \cos \theta \end{vmatrix} = r(\cos^2 \theta + \sin^2 \theta) = r$$

$$\iint_A dxdy = \int_0^{2\pi} \int_0^r r \, dr \, d\theta = \frac{1}{2}(r^2 - 0) \times (2\pi - 0) = \pi r^2$$



example) $F(x, y) = (2x + y, xy) \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$

$$J_F(x, y) = \begin{pmatrix} 2 & 1 \\ y & x \end{pmatrix} = \begin{pmatrix} \frac{\partial x_1}{\partial x_0} & \frac{\partial x_1}{\partial x_1} \\ \frac{\partial x_2}{\partial x_0} & \frac{\partial x_2}{\partial x_1} \end{pmatrix}$$

추가정보 url : <https://angeloyeo.github.io/2020/07/24/Jacobian.html>

class Affine:

def __init__(self, W, b):

self.W = W

self.b = b

self.x = None

self.dW = None

self.db = None

def forward(self, x):

self.x = x

out = np.dot(x, self.W) + self.b

return out

def backward(self, dout):

dx = np.dot(dout, self.W.T)

self.dW = np.dot(self.x.T, dout)

self.db = np.sum(dout, axis=0)

return dx

Affine

Affine 계층의 역할은 신경망에서 층을 하나 구성하는 것 (완전연결 Layer)

p.182 의 Affine 계층 생성하는 코드 # python 날코딩

self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])

□ 텐서플로우 2.x 버전으로 퍼셉트론 구현하기 **

퍼셉트론 : 인공 신경 세포 하나를 구현

```
pwd #tensorflow working directory
#'C:\WINDOWS\system32\cmd.exe'
```

퍼셉트론에서 쓰이는 입력데이터와 타겟 4가지

1. and 게이트
2. or 게이트
3. not and 게이트 단층으로 구현
4. xor 게이트 다층으로 구현

and 게이트 구현

```
import tensorflow as tf
tf.random.set_seed(777) # 시드를 설정한다.
import numpy as np
from tensorflow.keras.models import Sequential # 신경망 모델 구성
from tensorflow.keras.layers import Dense # 완전 연결계층 # Affine Layer
from tensorflow.keras.optimizers import SGD # 경사감소법 # Stochastic Gradient Descent # Sampling
from tensorflow.keras.losses import mse # 오차함수

# 데이터 준비
x = np.array([ [0, 0], [1, 0], [0, 1], [1, 1] ]) #shape : (4,2) #Features
y = np.array([ [0], [0], [0], [1] ]) #shape : (4,1) #Label

#모델 구성하기 #Sequential() 객체 생성
model = Sequential()

#단층 퍼셉트론 구현하기
model.add( Dense( 1, input_shape =( 2, ), activation ='linear' ) ) #Dense( depth, x : shape, activation = 'linear' | 'polynomial'? )
# weight 는 Dense method 에 포함되어져 있음
# 아래 문제에서는 위 weight 값을 추출할 수 있는 코드가 있으니 확인

# 모델 준비하기
model.compile( optimizer= SGD(), #Adam, Adagrad, momentum 등 다양한 경사하강법 method 옵션 존재
               loss= mse, #Mse : Mean Squared Error : 회귀 #CrossEntropyLoss : 분류
               metrics = ['acc'] ) # list 형태로 평가지표를 전달한다. #Train_Evaluation_Condition

# 학습 시키기
model.fit(x, y, epochs = 500) #500 epochs #x : feature #y : Label
```

문제104. AND 게이트 퍼셉트론을 구현한 신경망에서 만들어낸 가중치를 출력하시오 !

```
print (model.get_weights())
```

```
[array([[0.45816228], [0.3393745 ]], dtype=float32), array([-0.12992516], dtype=float32)]
```

문제105. 위의 and 게이트 퍼셉트론 신경망이 예측한 결과를 출력하시오!

```
model.evaluate(x , y ) # 모델을 평가한다.
result = model.predict(x) # 입력값 넣어서 예측값을 출력
print( result )
print("")
print( result.round() )
1/1 [=====] - 0s 18ms/step - loss: 0.0697 - acc: 1.0000
[[-0.12992516]
 [ 0.32823712]
 [ 0.20944935]
 [ 0.6676116 ]]

[[-0.]
 [ 0.]
 [ 0.]
 [ 1.]]
```

문제106. or 게이트를 구현하시오 !

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import mse, categorical_crossentropy, binary_crossentropy
tf.random.set_seed(777)

# 데이터 준비하기
x = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
y = np.array([[0], [1], [1], [1]])

# 모델 구성하기
model = Sequential()
model.add(Dense(1, input_shape = (2, ), activation = 'linear')) # 단층 퍼셉트론을 구성합니다

# 모델 준비하기
model.compile(optimizer = SGD(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다

# 학습시키기
model.fit(x, y, epochs = 500)

# 평가 및 결과출력
model.evaluate(x , y ) # 모델을 평가한다.
result = model.predict(x) # 입력값 넣어서 예측값을 출력
print( result ) #회귀값 #mse Loss function 사용해서 회귀로 결과값이 출력된 모습
print("")
print( result.round() )
```

```
[[0.3271933 ]
 [0.8215078 ]
 [0.70271987]
 [1.1970344 ]]
```

```
[[0.]
 [1.]
 [1.]
 [1.]]
```

문제107. Not and 게이트를 구현하시오 !

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import mse, categorical_crossentropy, binary_crossentropy
tf.random.set_seed(777)

# 데이터 준비하기
x = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
y = np.array([[1], [1], [1], [0]])

# 모델 구성하기
model = Sequential()
model.add(Dense(1, input_shape = (2, ), activation = 'linear')) # 단층 퍼셉트론을 구성합니다

# 모델 준비하기
model.compile(optimizer = SGD(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다

# 학습시키기
model.fit(x, y, epochs = 500)

# 평가 및 결과출력
model.evaluate(x , y ) # 모델을 평가한다.
result = model.predict(x) # 입력값 넣어서 예측값을 출력
print( result ) #회귀값 #mse Loss function 사용해서 회귀로 결과값이 출력된 모습
print("")
print( result.round() )
[[1.0968213 ]
 [0.78535557]
 [0.66656774]
 [0.35510206]]

[[1.]
 [1.]
 [1.]
 [0.]]
```

문제108. Xor 게이트를 구현하시오 ! hint : 다층으로 구현 다층으로 아래의 데이터를 넣으면 0 1 0 1 이렇게 출력한다.

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
```

```

from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.losses import mse
tf.random.set_seed(777)

# 데이터 준비하기
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
label = np.array([[0], [1], [1], [0]]) #XOR gate

# 모델 구성하기
model = Sequential()
model.add(Dense(32, input_shape = (2, ), activation = 'relu')) # 1층 구현
model.add(Dense( 1, activation='sigmoid')) # 2층 출력층 구현
#항상 모델의 첫번째 층은 데이터의 형태(위의 코드에서 input_shape인자) 를 전달해줘야한다는 점을 기억
#python 날코딩과 형태가 조금 다름 #가중치 shape을 맞춰준다

# 모델 준비
model.compile(optimizer = RMSprop(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다
# 모델을 model.add 로 구성했으면 컴파일(compile) 함수를 호출해서 학습과정을 설정
# optimizer, loss, metrics 3가지 옵션을 지정해서 compile 해 준다

# 학습시키기
model.fit(x, y, epochs = 500)

# 평가 및 결과출력
model.evaluate(x , y ) # 모델을 평가한다.
result = model.predict(x) # 입력값 넣어서 예측값을 출력
print( result ) #회귀값 #mse Loss function 사용해서 회귀로 결과값이 출력된 모습
print("")
print( result.round() )
[[0.42804596]
 [0.9674044 ]
 [0.9899212 ]
 [0.99968493]]

[[0.]
 [1.]
 [1.]
 [1.]]

### optipn 설명
## compile option : gradient method ---> optimizer -----
#1) 평균제곱오차 회귀문제
model.compile(optimizer = RMSprop(), loss = mse, metrics = ['acc'])

#2) 이항분류 문제 ( 이파리나 개고양이와 같은 2가지 분류)
model.compile(optimizer = RMSprop(), loss = binary_crossentropy, metrics = ['acc'])

#3) 다항 분류 문제
model.compile(optimizer = RMSprop(), loss = categorical_crossentropy, metrics = ['acc'])

#옵티마이저의 종류 : SGD(), RMSProp(), Adam(), NAadm() 등이 있다.

```

```
# 손실 함수(오차함수)의 종류: mse ( mean squared_error), binary_crossentropy, categorical_crossentropy 등이 있다.
# metrics = ['acc'] : 학습과정을 모니터링 하기 위해서 설정 : acc 와 같이 문자열로 지정해서 전달
# -----

# -----
model.fit(data, label, epochs = 100)
model.fit(data, label, epochs = 100, validation_data=(val_data, val_label) )
# validation_data는 검정 데이터인데 모델의 성능을 모니터링 하기 위해서 사용한다.
# 검정 데이터는 훈련 데이터를 일부를 가지고 만든 데이터로 훈련이 잘되는지 성능을 보기위한 평가지표로 사용한다.

model.evaluate(data, label)
# evaluate() 함수를 사용하면 손실과 평가 지표에 대한 정보를 확인가능

결과의 예: [ 0.210610177, 1.0 ]
# -----
```

●●● 5장의 mnist 신경망 (필기체 분류 신경망)을 Tensor 2.0 으로 구현 (3rd layer)

Index **

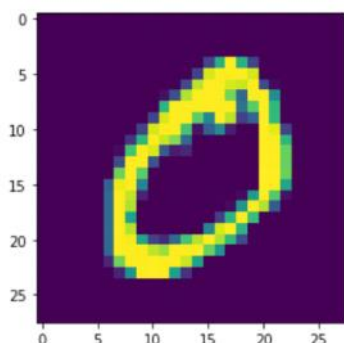
1. Load data
2. Validation : Train_data ---> trainset : testset
3. Pre-processing : Normalization
4. Label ---> one-hot encoding
5. Create object(model)
6. Compile object (object=model)
7. Fitting object (object=model)
8. Verify trained object

##1. Load mnist data

```
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz')
print(x_train.shape, y_train.shape) #(60000, 28, 28) (60000, )
print(y_train) #Label
```

1st rows data ---> Visualization

```
import matplotlib.pyplot as plt
img = x_train[1, :]
plt.figure()
plt.imshow(img)
Out[16]: <matplotlib.image.AxesImage at 0x23cd791c0a0>
```



##2. Validation : Traindata ---> trainset : testset = 7:3

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_state=777) #val means validation
print(x_train.shape) #(42000, 28, 28)
print(x_val.shape) #(18000, 28, 28)
print(x_test.shape) #(10000, 28, 28)
```

##3. Pre-processing : Normalization

```
# 정규화 이유 : 신경망은 입력 데이터의 scale 에 매우 민감
# 특히 이미지의 경우 숫자 하나의 픽셀이 0~255 사이 범위이므로 scaling 필요
# x_train, x_val, x_test -----> 정규화 필요 데이터
# 정규화를 진행하면 0~1 사이로 스케일링
```

```
num_x_train = x_train.shape[0]
num_x_val = x_val.shape[0]
num_x_test = x_test.shape[0]

x_train = x_train.reshape(num_x_train, 28*28) /255
x_val = x_val.reshape(num_x_val, 28*28) /255
x_test= x_test.reshape(num_x_test, 28*28) /255

print(x_train.shape) #(42000, 784)
print(x_val.shape) #(18000, 784)
print(x_test.shape) #(10000, 784)
```

##4. Label ---> one-hot encoding

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

##5. Create object(model)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(100, input_shape = (784, ), activation = 'relu')) #1st floor
model.add(Dense(50, input_shape = (784, ), activation = 'relu')) #2nd
model.add(Dense(10, activation='softmax') ) #3rd
```

##6. Compile object (object=model)

```
from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )
```

```
# learning rate default : 0.001
```

##7. Fitting object (object=model)

```
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )
```

##8. Verify trained object

```
results = model.predict(x_test)
print(results)
```

##9. 확률벡터 10000개에 numpy argmax 를 이용해서 최대값의 인덱스 출력

```
import numpy as np
np.argmax(results, axis=1)
```

☆ 위 출력된 예측 숫자들 10000개와 테스트 데이터 숫자 10000개와 비교해서 정확도를 출력하시오!

```
pred = np.argmax(results, axis=1)
label = np.argmax(y_test, axis=1)
a = pred==label
```

```
a.astype(int).sum() / len(a) #0.9702
```

```
#####endLine=====
```

##0316

2021년 3월 1일 월요일 오후 4:52

■ Review

수치미분이 너무 느리기 때문에 오차역전파를 사용해서 기울기를 구하고, 가중치를 갱신한다.

5장에서는 오차 역전파를 이해하기 위한 FlowChart 역할을 하는 계산그래프를 이용해서 가이드를 제시해 준다.

1. 계산그래프를 통해서 덧셈 그래프와 곱셈 그래프를 이해
2. Relu 함수 클래스 생성
3. sigmoid 함수 계산 그래프, 클래스 생성(순전파, 역전파)
4. Affine 계층(행렬의 내적) 계산 그래프, 클래스 생성
5. 텐서플로우 2.0 으로 3층 신경망 구현

##1. Load mnist data

```
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz')
print(x_train.shape, y_train.shape) #(60000, 28, 28) (60000, )
print(y_train) #Label
```

```
# 1st rows data ---> Visualization
import matplotlib.pyplot as plt
img = x_train[1, :]
plt.figure()
plt.imshow(img)
```

##2. Validation : Traindata ---> trainset : testset = 7:3

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_state=777) #val means validation
print(x_train.shape) #(42000, 28, 28)
print(x_val.shape) #(18000, 28, 28)
print(x_test.shape) #(10000, 28, 28)
```

##3. Pre-processing : Normalization

```
# 정규화 이유 : 신경망은 입력 데이터의 scale 에 매우 민감
# 특히 이미지의 경우 숫자 하나의 픽셀이 0~255 사이 범위이므로 scaling 필요
# x_train, x_val, x_test -----> 정규화 필요 데이터
# 정규화를 진행하면 0~1 사이로 스케일링
```

```
num_x_train = x_train.shape[0]
num_x_val = x_val.shape[0]
num_x_test = x_test.shape[0]
```

```
x_train = x_train.reshape(num_x_train, 28*28) /255
x_val = x_val.reshape(num_x_val, 28*28) /255
x_test= x_test.reshape(num_x_test, 28*28) /255
```



```
print(x_train.shape) #(42000, 784)
print(x_val.shape) #(18000, 784)
print(x_test.shape) #(10000, 784)
```

##4. Label ---> one-hot encoding

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

##5. Create object(model)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(100, input_shape = (784, ), activation = 'relu')) #1st floor
model.add(Dense(50, activation = 'relu')) #2nd
model.add(Dense(10, activation='softmax') ) #3rd
```

##6. Compile object (object=model)

```
from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )

# learning rate default : 0.001
```

##7. Fitting object (object=model)

```
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )
```

##8. Verify trained object

```
results = model.predict(x_test)
print(results)
```

##9. Results

```
import numpy as np
pred = np.argmax(results, axis=1)
label = np.argmax(y_test, axis=1)
a = pred==label
a.astype(int).sum() / len(a) #0.9702
```

문제109. 지금 만든 3층 신경망의 활성화함수를 relu로 했을 때의 정확도는 0.9702 이다.

sigmoid 함수를 활성화함수로 사용했을때의 정확도는 어떻게 되는가?

```
##5. Create object(model)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(100, input_shape = (784, ), activation = 'sigmoid')) #1st floor
model.add(Dense(50, input_shape = (784, ), activation = 'sigmoid')) #2nd
model.add(Dense(10, activation='softmax') ) #3rd

##6. Compile object (object=model)
from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )

# learning rate default : 0.001

##7. Fitting object (object=model)
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )

##8. Verify trained object
results = model.predict(x_test)
print(results)

##9. Results
import numpy as np
pred = np.argmax(results, axis=1)
label = np.argmax(y_test, axis=1)
a = pred==label
a.astype(int).sum() / len(a) #0.9732
```

문제110. 위 결과를 시각화하시오!

```
# 시각화
import matplotlib.pyplot as plt

his_dict = history.history
loss = his_dict['loss']
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.

epochs = range(1, len(loss) + 1)
```

```
fig = plt.figure(figsize = (10, 5))
```

```
# 훈련 및 검증 손실 그리기
```

```
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()
```

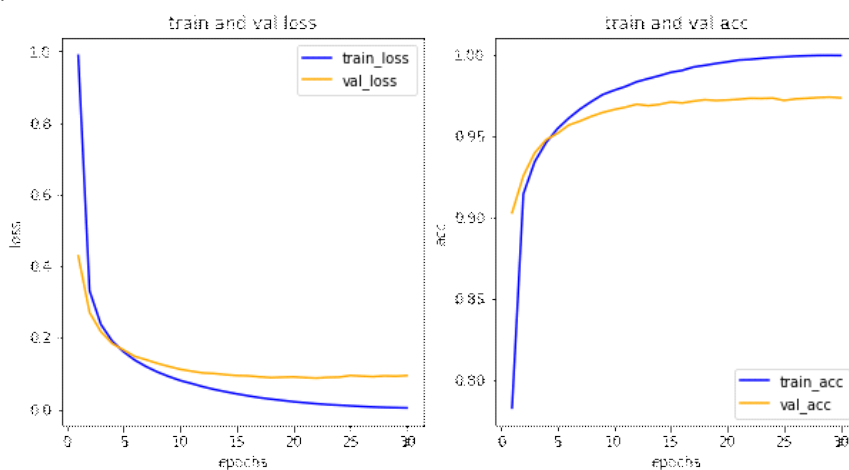
```
acc = his_dict['acc']
```

```
val_acc = his_dict['val_acc']
```

```
# 훈련 및 검증 정확도 그리기
```

```
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_acc')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_acc')
ax2.set_title('train and val acc')
ax2.set_xlabel('epochs')
ax2.set_ylabel('acc')
ax2.legend()
```

```
plt.show()
```



문제111. 위 코드를 이용해서 4층으로 신경망을 구현하시오!

```
##5. Create object(model)
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
```

```
model = Sequential()
```

```
model.add(Dense(100, input_shape = (784, ), activation = 'sigmoid')) #1st floor
```

```
model.add(Dense(50, activation = 'sigmoid')) #2nd
```

```
model.add(Dense(50, activation = 'sigmoid')) #3nd
```

```
model.add(Dense(10, activation='softmax')) #3rd
```

```

##6. Compile object (object=model)
from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )

# learning rate default : 0.001

##7. Fitting object (object=model)
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )

##8. Verify trained object
results = model.predict(x_test)
print(results)

##9. Results
import numpy as np
pred = np.argmax(results, axis=1)
label = np.argmax(y_test, axis=1)
a = pred==label
a.astype(int).sum() / len(a) #0.9673
# 층수를 3층에서 4층으로 변경했는데 ACC는 오히려 떨어졌다. (깊은 층이 더 좋은 예측도 제공 x)

```

python 날코딩 코드

<https://cafe.daum.net/oracleoracle/Sedp/235>

수치미분과 오차역전파의 차이

<https://cafe.daum.net/oracleoracle/Sedp/202>

■ 6장. 학습 관련 기술들

인공신경망의 언더피팅과 오버피팅을 막는 방법

1. underfitting 을 방지하는 방법

- 고급 경사하강법의 종류
- 가중치 초기값 설정
- 배치 정규화

2. overfitting 방지

- 드롭아웃

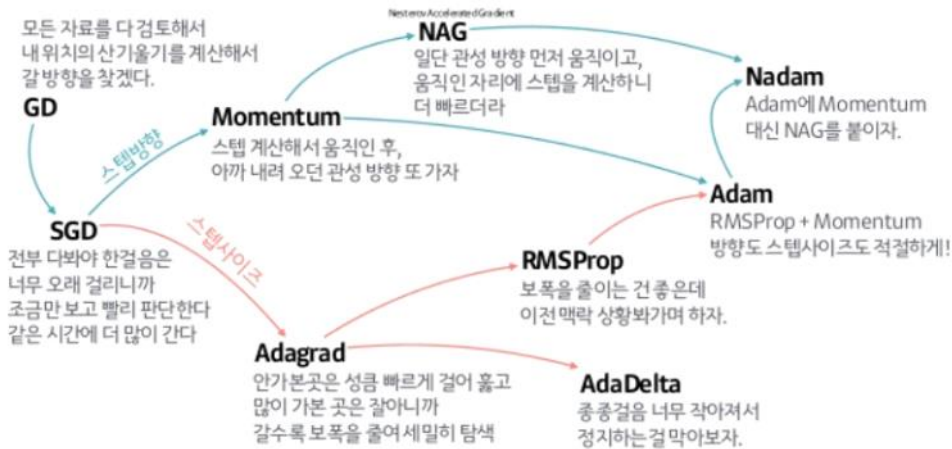
- 엘리스탑 (keras 의 기능, 책에서는 나오지 않으나 현업에서 많이 사용)
- L2 정규화

□ 경사하강법 종류

<https://cafe.daum.net/oracleoracle/Sedp/236>

고급 경사하강법의 정의

산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보



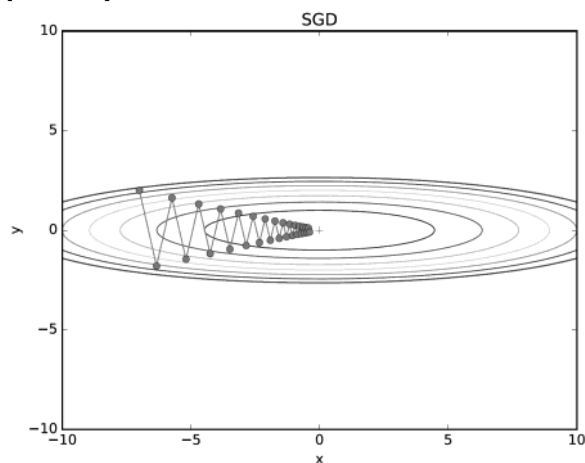
출처: 하용호, 자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다

1. GD(Gradient Descent) : 학습 데이터를 다 입력하고 한 걸음 이동하는 경사하강법

학습데이터를 신경망에 다 입력해서 한 걸음 이동하기 때문에 계산코스트가 높다

2. SGD(Stochastic Gradient Descent) : 확률적 경사하강법

[그림 6-3]



GD 의 단점 개선

확률적 경사하강법으로 복원추출 미니배치를 이용해서 경사가 기울어진 방향으로 학습

단점 : Local minimum 에 잘 빠진다

문제112. 5장 전체코드(텐서플로우 2.0) 의 경사하강법을 SGD 로 구현해서 수행, 테스트 데이터 정확도와 시각화 업로드

##1. Load mnist data

```
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz')
print(x_train.shape, y_train.shape) #(60000, 28, 28) (60000, )
```

```
print(y_train) #Label
```

```
# 1st rows data ---> Visualization
```

```
import matplotlib.pyplot as plt
```

```
img = x_train[1, :]
```

```
plt.figure()
```

```
plt.imshow(img)
```

```
##2. Validation : Traindata ---> trainset : testset = 7:3
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_state=777) #val means validation
```

```
print(x_train.shape) #(42000, 28, 28)
```

```
print(x_val.shape) #(18000, 28, 28)
```

```
print(x_test.shape) #(10000, 28, 28)
```

```
##3. Pre-processing : Normalization
```

```
# 정규화 이유 : 신경망은 입력 데이터의 scale 에 매우 민감
```

```
# 특히 이미지의 경우 숫자 하나의 픽셀이 0~255 사이 범위이므로 scaling 필요
```

```
# x_train, x_val, x_test -----> 정규화 필요 데이터
```

```
# 정규화를 진행하면 0~1 사이로 스케일링
```

```
num_x_train = x_train.shape[0]
```

```
num_x_val = x_val.shape[0]
```

```
num_x_test = x_test.shape[0]
```

```
x_train = x_train.reshape(num_x_train, 28*28) /255
```

```
x_val = x_val.reshape(num_x_val, 28*28) /255
```

```
x_test= x_test.reshape(num_x_test, 28*28) /255
```

```
print(x_train.shape) #(42000, 784)
```

```
print(x_val.shape) #(18000, 784)
```

```
print(x_test.shape) #(10000, 784)
```

```
##4. Label ---> one-hot encoding
```

```
from tensorflow.keras.utils import to_categorical
```

```
y_train = to_categorical(y_train)
```

```
y_val = to_categorical(y_val)
```

```
y_test = to_categorical(y_test)
```

```
##5. Create object(model)
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
```

```
model = Sequential()
```

```
model.add(Dense(100, input_shape = (784, ), activation = 'relu')) #1st floor
```

```
model.add(Dense(50, activation = 'relu')) #2nd
```

```
model.add(Dense(10, activation='softmax')) #3rd
```

```
##6. Compile object (object=model)
```

```
from tensorflow.keras.losses import categorical_crossentropy
```

```
model.compile(optimizer = 'SGD',  
              loss = 'categorical_crossentropy',  
              metrics = ['acc'] )
```

```
# learning rate default : 0.001
```

```
##7. Fitting object (object=model)
```

```
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )
```

```
##8. Verify trained object
```

```
results = model.predict(x_test)
```

```
print(results)
```

```
##9. Results
```

```
import numpy as np
```

```
pred = np.argmax(results, axis=1)
```

```
label = np.argmax(y_test, axis=1)
```

```
a = pred==label
```

```
a.astype(int).sum() / len(a) #0.9564
```

```
# 시각화
```

```
import matplotlib.pyplot as plt
```

```
his_dict = history.history
```

```
loss = his_dict['loss']
```

```
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.
```

```
epochs = range(1, len(loss) + 1)
```

```
fig = plt.figure(figsize = (10, 5))
```

```
# 훈련 및 검증 손실 그리기
```

```
ax1 = fig.add_subplot(1, 2, 1)
```

```
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
```

```
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
```

```
ax1.set_title('train and val loss')
```

```
ax1.set_xlabel('epochs')
```

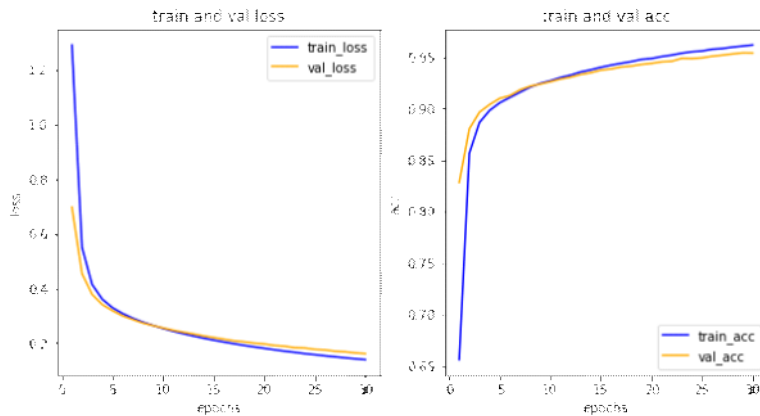
```
ax1.set_ylabel('loss')
```

```
ax1.legend()
```

```
acc = his_dict['acc']
val_acc = his_dict['val_acc']
```

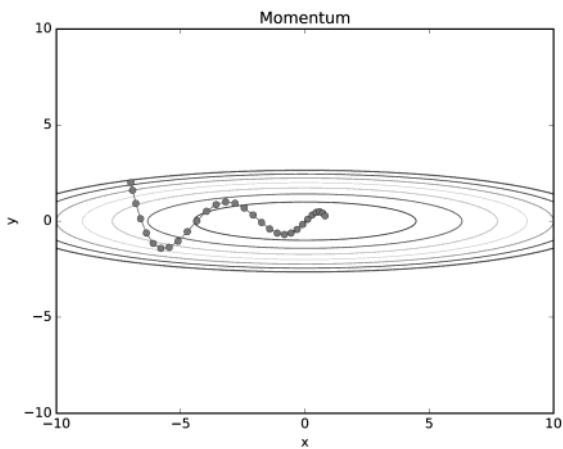
훈련 및 검증 정확도 그리기

```
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_acc')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_acc')
ax2.set_title('train and val acc')
ax2.set_xlabel('epochs')
ax2.set_ylabel('acc')
ax2.legend()
```



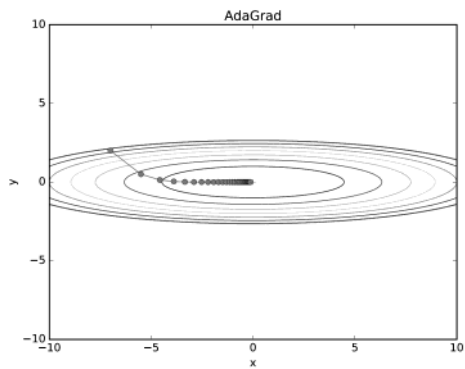
3. Momentum

관성을 이용해서 local minima 에서 빠져나가는 경사하강법



모멘텀은 tensorflow2.0 에는 따로 없지만 Adam과 다른 경사하강법 종류라는 것에 의의가 있음

4. Adagrad p.197



learning rate 을 자동조절하는 경사하강법

초기 학습때는 학습률이 크고, 진행될수록 적응적으로 학습률을 조정한다(학습률 값 감소)

갱신 강도가 약해져서 global optima 에 가기 전 갱신량이 0이 되는 문제가 있다

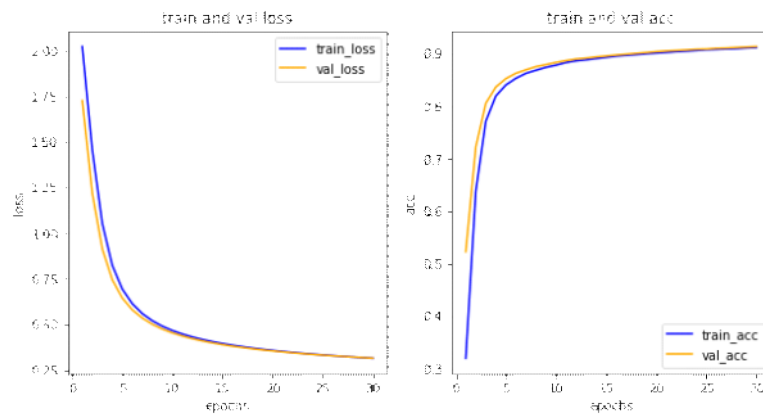
위 문제를 개선한 방법으로 RMSProp 를 사용

RMSProp는 과거의 모든 기울기를 균일하게 더하는 것이 아닌 과거의 기울기는 서서히 잊고, 새로운 기울기 정보를 크게 반영함

이러한 방법을 지수이동평균이라고 하며, 과거 기울기의 반영 규모를 기하급수적으로 감소시킨다

문제113. Adagrad 로 3층 신경망의 경사하강법을 학습하고 정확도를 확인

Acc : 0.9192



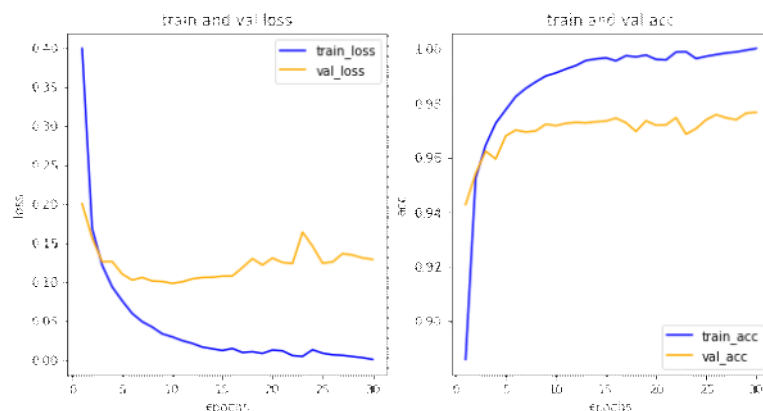
4. Adam

momentum 의 장점 + Adagrad 의 장점을 살린 경사하강법

(관성을 이용) (learning rate 자동조절)

문제114. 3층신경망의 경사하강법을 Adam으로 하고 학습시키시오!

Acc : 0.976



오버피팅이 조금 발생한 모습

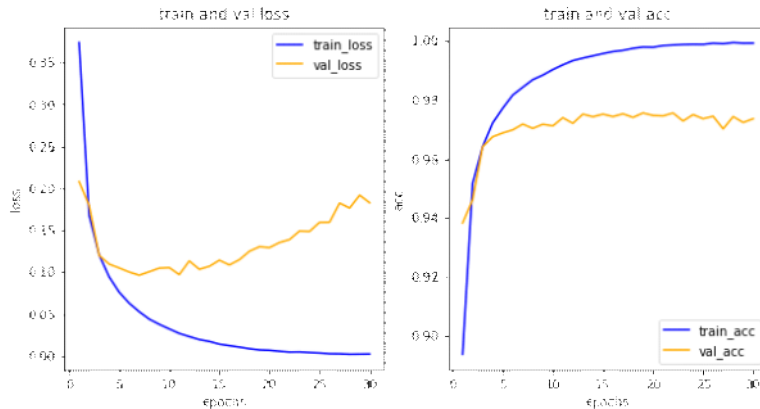
5. RMSProp

Adagrad 의 단점을 개선

RMSProp는 과거의 모든 기울기를 균일하게 더하는 것이 아닌 과거의 기울기는 서서히 잊고, 새로운 기울기 정보를 크게 반영함
이러한 방법을 지수이동평균이라고 하며, 과거 기울기의 반영 규모를 기하급수적으로 감소시킨다

115. RMSProp 으로 경사하강법을 변경해서 정확도를 살펴보세요!

Acc : 0.9742



정확도는 높지만 오버피팅은 여전히 발생

□ 가중치 초기값 설정 p.202

랜덤으로 생성되는 가중치 W 의 초기값을 어떻게 선정하느냐에 따라 학습이 잘 될수도 있고, 안 될수도 있다.
학습이 잘 되려면 가중치 초기값들의 분포가 정규분포 형태를 이루어야 한다.

`np.random.randn` 는 가우시안 정규분포(평균이 0, 표준편차가 1)를 따르는 난수를 생성한다.

가중치 초기화

<https://cafe.daum.net/oracleoracle/Sedp/238>

1. 생성되는 가중치의 표준편차가 너무 큰 경우

시험문제가 너무 어려우면 아주 잘하는 학생들과 아주 못하는 학생들로 점수가 나뉜다.

ex)

```
1 *np.random.randn(784, 50)
```

2. 생성되는 가중치의 표준편차가 너무 작은 경우

시험문제가 너무 쉬우면 학생들의 점수가 평균에 가까워진다.

ex)

```
0.01 * np.random.randn(784, 50)
```

다른 표준편차를 지정하는 방법

1. Xavier 초기값 선정 : sigmoid 와 함께 사용

2. He 초기값 선정 : Relu 와 함께 사용

```
model = Sequential()
```

```
initializer = tf.keras.initializers.GlorotNormal()
```

```
model.add( Dense(100, activation='relu', kernel_initializer=initializer, input_shape=(784, )))
```

```
model.add( Dense(50, activation='relu', kernel_initializer=initializer))
```

```
model.add( Dense(10, activation='softmax', kernel_initializer=initializer))
```

□ 배치 정규화

가중치 초기화 값을 적절히 설정하면 각 층의 활성화 값이 분포가 적당히 퍼지는 효과를 보인다.

이때 신경망이 깊어지고 학습이 반복되면 각 층의 활성화 값이 분포가 정규성을 잃어버리는 현상이 발생한다.

그래서 이 정규성을 계속해서 유지시키도록 강제화 하는 방법이 '배치정규화' 이다.

면접질문

Q1. 딥러닝이 무엇인가요?

Q2. 배치정규화가 무엇인가요?

텐서플로우 2.0으로 구현 (코드수정한 부분 추가) ***

##1. Load mnist data

```
import tensorflow as tf
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz')
print(x_train.shape, y_train.shape) #(60000, 28, 28) (60000, )
print(y_train) #Label
```

1st rows data ---> Visualization

```
import matplotlib.pyplot as plt
img = x_train[1, :]
plt.figure()
plt.imshow(img)
```

##2. Validation : Traindata ---> trainset : testset = 7:3

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_state=777) #val means validation
print(x_train.shape) #(42000, 28, 28)
print(x_val.shape) #(18000, 28, 28)
print(x_test.shape) #(10000, 28, 28)
```

##3. Pre-processing : Normalization

정규화 이유 : 신경망은 입력 데이터의 scale 에 매우 민감

특히 이미지의 경우 숫자 하나의 픽셀이 0~255 사이 범위이므로 scaling 필요

x_train, x_val, x_test -----> 정규화 필요 데이터

정규화를 진행하면 0~1 사이로 스케일링

```
num_x_train = x_train.shape[0]
num_x_val = x_val.shape[0]
num_x_test = x_test.shape[0]
```

```
x_train = x_train.reshape(num_x_train, 28*28) /255
```

```
x_val = x_val.reshape(num_x_val, 28*28) /255
```

```
x_test= x_test.reshape(num_x_test, 28*28) /255
```

```
print(x_train.shape) #(42000, 784)
print(x_val.shape) #(18000, 784)
print(x_test.shape) #(10000, 784)
```

```
##4. Label ---> one-hot encoding
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

```
##5. Create object(model)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization
```

```
model = Sequential()
initializer = tf.keras.initializers.GlorotNormal()
model.add(Dense(100, activation = 'relu', kernel_initializer = initializer, input_shape = (784, )))
model.add(BatchNormalization())
model.add(Dense(50, activation = 'relu', kernel_initializer = initializer))
model.add(BatchNormalization())
model.add(Dense(10, activation = 'softmax', kernel_initializer = initializer))
```

```
##6. Compile object (object=model)
from tensorflow.keras.losses import categorical_crossentropy
```

```
model.compile(optimizer = 'RMSProp',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )
```

```
# learning rate default : 0.001
```

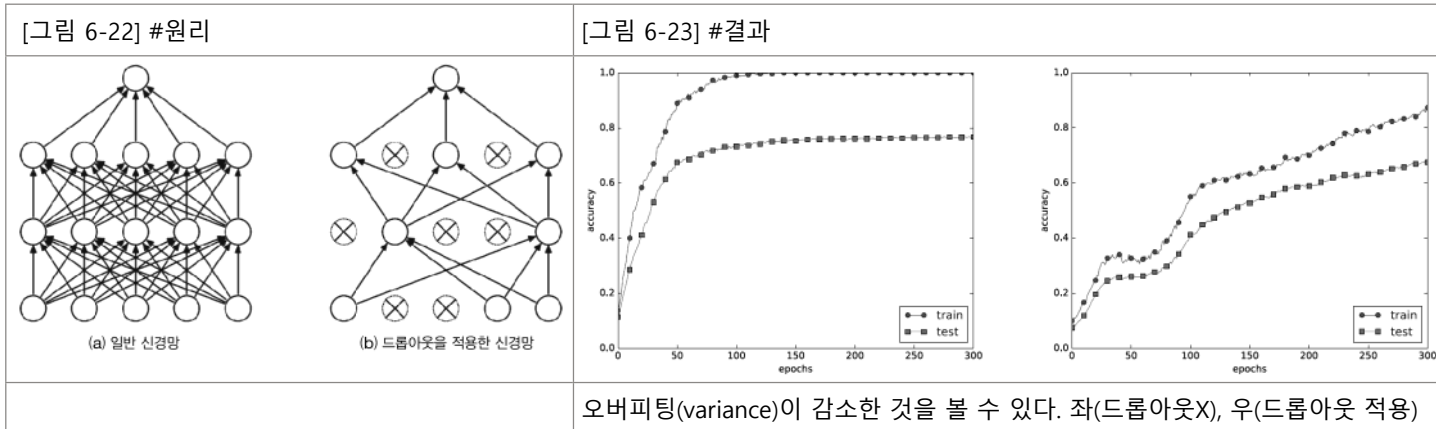
```
##7. Fitting object (object=model)
history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )
```

```
##8. Verify trained object
results = model.predict(x_test)
print(results)
```

```
##9. Results
import numpy as np
pred = np.argmax(results, axis=1)
```

```
label = np.argmax(y_test, axis=1)
a = pred==label
a.astype(int).sum() / len(a) #0.9754 #Acc
```

●6.4.3 드롭아웃 p.219



드롭아웃은 ML(기계학습)에서 앙상블 학습이 여러 모델의 평균을 내는 것과 같은 효과를 낸다

Regularization 방법 : L1 Regularization, L2 Regularization, **Dropout**, Early stopping

Early stopping 은 ML에서도 자주 사용하는 방식으로, 일정 range 를 지나도록 정해진 기준(Cost)이 개선되지 않으면 학습을 종료

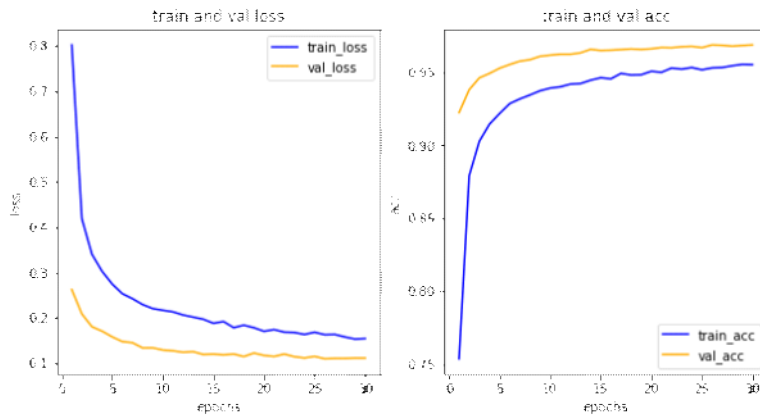
TensorFlow 2.0 적용 code

```
##5. Create object(model)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout

model = Sequential()
initializer = tf.keras.initializers.GlorotNormal()
model.add(Dense(100, activation = 'relu', kernel_initializer = initializer, input_shape = (784, )))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(50, activation = 'relu', kernel_initializer = initializer))
model.add(BatchNormalization())
model.add(Dropout(0.5))

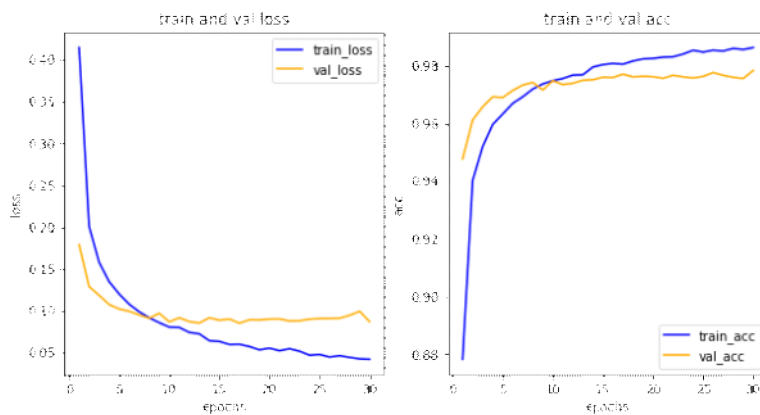
model.add(Dense(10, activation = 'softmax', kernel_initializer = initializer))
~
Acc : 0.9706
```



이전에 비해서 오버피팅은 줄어들고, 언더피팅이 조금 일어난 모습이다.

이번에는 0.5 가 아닌 0.2 로 20% dropout 을 진행

Acc : 0.9782



● 엘리스탑(early stop)

DNN API 종류

1위 : tensorflow ---> keras 인수

2위 : keras

3위 : pytorch

현업에서는 tensorflow 와 pytorch 를 이용

엘리스탑 기능은 원래 keras 에 있었고, 현재는 tensorflow 에서 이용할 수 있음

엘리스탑 : 훈련데이터의 정확도와 validation 정확도가 같아지는 부분에서 멈추는 기능

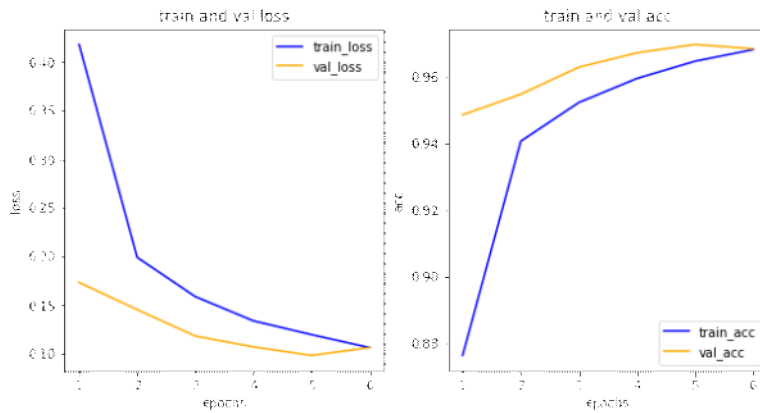
[code]

```
from tensorflow.python.keras.callbacks import EarlyStopping
```

```
early_stopping = EarlyStopping()
```

```
history = model.fit(x_train, y_train,
                    epochs = 100,
                    batch_size = 100,
                    validation_data = (x_val, y_val),
                    callbacks=[early_stopping])
```

Acc : 0.9696



● L2 정규화 p.217

가중치 감소(Weight Decay)

학습과정에서 큰 가중치에 대해서 그에 상응하는 큰 페널티를 부여하여 오버피팅을 억제하는 방법

입력층 -----> 은닉층 -----> 출력층

- -----> 귀가 있어요 -----> 아주 큰 가중치가 출력
- -----> 꼬리가 있어요 -----> 가중치 출력
- -----> 집계발을 가지고 있어요 -----> 가중치 출력
- -----> 장난스럽게 보여요 -----> 가중치 출력

귀가없는 고양이 사진이 들어오는 경우 이 사진은 고양이가 아니라고 판단을 한다

----> 이는 오버피팅을 유발한다

위와같은 상황을 해결하기 위해 큰 가중치에 대해서 큰 페널티를 부여해서 가중치 매개변수가 trainset 에 오버피팅되지 않도록 조절

tensorflow 2.0 [code]

##5. Create object(model)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, BatchNormalization, Dropout

model = Sequential()

initializer = tf.keras.initializers.GlorotNormal()

model.add(Dense(100, input_shape = (784,),

activation = 'relu',

kernel_initializer = initializer,

kernel_regularizer='l2'))

model.add(BatchNormalization())

model.add(Dropout(0.2))

model.add(Dense(50, activation = 'relu',

kernel_initializer = initializer,

kernel_regularizer='l2'))

model.add(BatchNormalization())

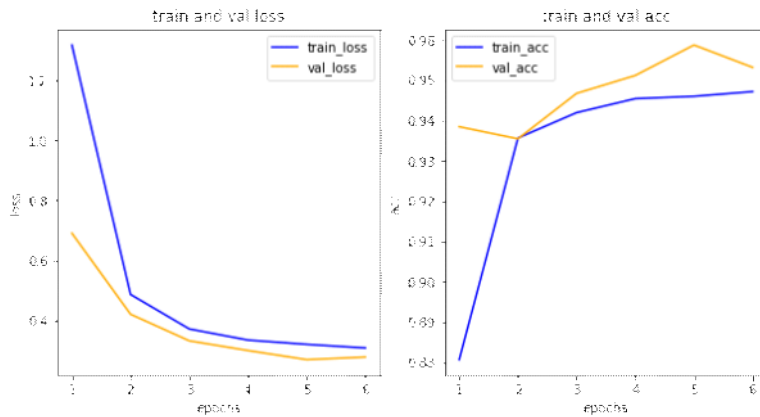
model.add(Dropout(0.2))

model.add(Dense(10, activation = 'softmax',

kernel_initializer = initializer,

kernel_regularizer='l2'))

Acc : 0.9542



6장 요약

1. 언더피팅 억제

- 1) 고급경사감소법
- 2) 가중치 초기값 설정
- 3) 배치 정규화

2. 오버피팅 억제

- 1) 드롭아웃
- 2) 얼리스탑
- 3) l2 정규화

■ 포트폴리오 준비 1

mnist 말고 다른 사진을 신경망에 넣을 줄 알아야 한다

1. 스크롤링한 사진이 있으면 그 사진을 사용하고 없으면 선생님이 보내준 leaf 사진 사용
2. 일단 정제된 데이터는 Leaf 데이터로 실습해보기

이파리 사진의 구조

Train : 19000장 (정상 1~9500 + 질병 9501~19000)

Test : 1000장 (정상 1~500 + 질병 501~1000)

필요한 코드 :

1. 신경망 코드 (이미 만들었고 계속해서 발전 예정) ---> googling 으로 찾을 수 있음
2. 신경망에 사진을 로드하는 코드 ---> googling 해도 안나옴

[code]

예제1

```
import os
tset_image = 'C:\Users\wwicecr\OneDrive\Images\Leaf\train'

def image_load(path):
    file_list = os.listdir(path)
    return file_list

print (image_load(test_image) )
```

예제2. 위 결과에서 .jpg 는 빼고 숫자만 출력되게 하시오!


```

import os
import re
test_image = 'C:\\Users\\wicecr\\OneDrive\\images\\Leaf\\test'

def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null)로 바꿔주는 전처리 코드
        file_name.append(a)
    return file_name

print (image_load(test_image) )

```

☆문제116. 위 결과가 정렬되서 출력되게 하시오!

```

import os
import re
test_image = 'C:\\Users\\wicecr\\OneDrive\\images\\Leaf\\test'

def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null)로 바꿔주는 전처리 코드
        file_name.append(a)
    file_name.sort()
    return file_name

print (image_load(test_image) )

```

●●●mnist 추출 및 시각화 코드 (최종정리)

```

##1. Load mnist data
import tensorflow as tf
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz')
print(x_train.shape, y_train.shape) #(60000, 28, 28) (60000, )
print(y_train) #Label

# 1st rows data ---> Visualization
import matplotlib.pyplot as plt
img = x_train[1, :]
plt.figure()
plt.imshow(img)

```

```

##2. Validation : Traindata ---> trainset : testset = 7:3

```

```

from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3, random_state=777) #val means validation
print(x_train.shape) #(42000, 28, 28)
print(x_val.shape) #(18000, 28, 28)
print(x_test.shape) #(10000, 28, 28)

```

##3. Pre-processing : Normalization

```

# 정규화 이유 : 신경망은 입력 데이터의 scale 에 매우 민감
# 특히 이미지의 경우 숫자 하나의 픽셀이 0~255 사이 범위이므로 scaling 필요
# x_train, x_val, x_test ----> 정규화 필요 데이터
# 정규화를 진행하면 0~1 사이로 스케일링

```

```

num_x_train = x_train.shape[0]
num_x_val = x_val.shape[0]
num_x_test = x_test.shape[0]

```

```

x_train = x_train.reshape(num_x_train, 28*28) /255
x_val = x_val.reshape(num_x_val, 28*28) /255
x_test = x_test.reshape(num_x_test, 28*28) /255

```

```

print(x_train.shape) #(42000, 784)
print(x_val.shape) #(18000, 784)
print(x_test.shape) #(10000, 784)

```

##4. Label ---> one-hot encoding

```

from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)

```

##5. Create object(model) & Add layer**

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout

```

```

initializer = tf.keras.initializers.GlorotNormal() #'He' initializer

```

```

model = Sequential()

```

#1st #hidden layer

```

model.add(Dense(100, input_shape = (784, ),
              activation = 'relu',
              kernel_initializer = initializer,
              kernel_regularizer='l2'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

```

```

#2nd #hidden layer
model.add(Dense(50, activation = 'relu',
                kernel_initializer = initializer,
                kernel_regularizer='l2'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

#3rd #output layer
model.add(Dense(10, activation = 'softmax',
                kernel_initializer = initializer,
                kernel_regularizer='l2'))

##6. Compile object (object=model)
from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'RMSProp',
              loss = 'categorical_crossentropy',
              metrics = ['acc'] )
# learning rate default : 0.001

##7. Fitting object (object=model)
from tensorflow.python.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping()

history = model.fit(x_train, y_train,
                   epochs = 30,
                   batch_size = 100,
                   validation_data = (x_val, y_val),
                   callbacks=[early_stopping])

#history = model.fit( x_train, y_train, epochs=30, batch_size=100, validation_data=(x_val, y_val) )

##8. Verify trained object
results = model.predict(x_test)
print(results)

##9. Results
import numpy as np
pred = np.argmax(results, axis=1) #index of max value (within axis=1)
label = np.argmax(y_test, axis=1)
a = pred==label
a.astype(int).sum() / len(a) #0.9754

```

시각화

```
import matplotlib.pyplot as plt

his_dict = history.history
loss = his_dict['loss']
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.

epochs = range(1, len(loss) + 1)
fig = plt.figure(figsize = (10, 5))

# 훈련 및 검증 손실 그리기
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()

acc = his_dict['acc']
val_acc = his_dict['val_acc']

# 훈련 및 검증 정확도 그리기
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_acc')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_acc')
ax2.set_title('train and val acc')
ax2.set_xlabel('epochs')
ax2.set_ylabel('acc')
ax2.legend()

#####endLine=====
```

##0317

2021년 3월 1일 월요일 오후 4:52

■ Review

1. numpy 사용법
2. 퍼셉트론
3. 3층 신경망 생성(저자의 .pkl 파일 사용)
4. 2층 신경망 생성(수치미분 사용)
5. 3층 신경망 생성(오차역전파 사용)
6. 언더피팅과 오버피팅 억제
7. CNN
8. 딥러닝의 역사

예제4. (이파리 데이터 전처리) 위에서 출력된 결과를 아래와 같이 .jpg 가 붙어서 출력되게 하시오!

```
import os
import re
test_image = 'C:\Users\wicecr\OneDrive\Images\Leaf\test'

def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null)로 바꿔주는 전처리 코드
        file_name.append(a)
    file_name.sort()

    file_res = []
    for j in file_name:
        file_res.append('%d.jpg' %j)

    return file_res

print (image_load(test_image) )
```

예제5. 이미지 이름 앞에 절대경로가 아래처럼 붙게 하시오!

```
import os
import re
test_image = 'C:\Users\wicecr\OneDrive\Images\Leaf\test'

def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null)로 바꿔주는 전처리 코드
        file_name.append(a)
    file_name.sort()
```

```

file_res = []
for j in file_name:
    file_res.append('%s\%d.jpg' %(path,j) )

return file_res

print (image_load(test_image) )

```

예제6. opencv 를 이용해서 이미지를 numpy array 형태의 숫자로 변환하시오!

anaconda prompt> pip install opencv-python

```

import os
import re
import cv2
import numpy as np
test_image = 'C:\Users\Wwicecr\OneDrive\images\Leaf\test'

def image_load(path):
    file_list = os.listdir(path)
    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null) 로 바꿔주는 전처리 코드
        file_name.append(a)
    file_name.sort()

    file_res = []
    for j in file_name:
        file_res.append('%s\%d.jpg' %(path,j) )

    image = []
    for k in file_res:
        img = cv2.imread(k)
        image.append(img)

    return np.array(image)

x = image_load(test_image)
x.shape # (1000, 256, 256, 3)

```

설명

신경망에 이미지를 인식시키기 위해서는 2개의 함수가 필요

1. image_load : 예제 6번 코드 : 이미지를 숫자로 변환해주는 함수
2. label_load : 정답 라벨을 one hot encoding 하는 함수 (뒤에 등장)

예제7. 훈련 데이터의 라벨을 train_label.csv, 테스트 데이터의 라벨을 test_label.csv 로 생성하시오!

(1~9500 : 1(정상), 9501~19000 : 0(질병)) // (1~500 : 1(정상), 501~1000 : 0(질병))

```
path = 'C:\\Users\\wicecr\\OneDrive\\Images\\Leaf\\train_label.csv'
```

```
file = open( path, 'w')
for i in range(1, 9501):
    file.write(str(1) + '\\n')
```

```
for i in range(1, 9501):
    file.write(str(1) + '\\n')
for i in range(1, 9501):
    file.write(str(0) + '\\n')
```

```
file.close
```

예제8. 테스트 데이터 라벨도 생성

```
path = 'C:\\Users\\wicecr\\OneDrive\\Images\\Leaf\\test_label.csv'
```

```
file = open( path, 'w')
```

```
for i in range(500):
    file.write(str(1) + '\\n')
for i in range(500):
    file.write(str(0) + '\\n')
```

```
file.close()
```

예제9. train_label.csv 의 내용을 불러오는 함수를 생성하시오!

```
train_label = 'C:\\Users\\wicecr\\OneDrive\\Images\\Leaf\\test_label.csv'
```

```
import csv
```

```
def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist=[]
    for i in labeldata:
        labellist.append(i)
    return labellist
```

```
print(label_load(train_label) )
```

```
# 결과는 문자로 출력된다
```

예제10. 예제9번의 결과는 문자로 출력되었다. 이를 숫자로 출력되게 하시오!

```
import csv
import numpy as np
import os
import re
import cv2
```

```
train_label = 'C:\\Users\\wicecr\\OneDrive\\Images\\Leaf\\train_label.csv'
```

```
def label_load(path):
    file = open(path)
```

```

labeldata = csv.reader(file)
labellist=[]

for i in labeldata:
    labellist.append(i)

label = np.array(labellist)
label = label.astype(int)
return label

print(label_load(train_label) )

```

예제 11. np.eye 연습

```

import numpy as np
print( np.eye(10)[4] )
# [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

```

예제 12. label_load 함수의 결과가 one hot encoding 된 결과로 출력하시오!

```

import csv
import numpy as np
import os
import re
import cv2

train_label = 'C:\Users\Wicr\OneDrive\Images\Leaf\train_label.csv'

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist=[]

    for i in labeldata:
        labellist.append(i)

    label = np.array(labellist)
    label = label.astype(int)
    label = np.eye(2)[label]

    return label

print(label_load(train_label) )
print(label_load(train_label).shape ) #(19000, 1, 2) #3차원

```

예제 13. 위의 결과는 3차원이다. 신경망에서 label 로 사용하려면 2차원 데이터여야 한다. label 의 차원을 2차원으로 축소시켜 출력하시오!

```

import csv
import numpy as np
import os

```



```

import re
import cv2

train_label = 'C:\\Users\\wicecr\\OneDrive\\images\\Leaf\\train_label.csv'

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)

    labellist=[]
    for i in labeldata:
        labellist.append(i)

    label = np.array(labellist)
    label = label.astype(int)
    label = np.eye(2)[label]
    label = label.reshape(-1,2)

    return label

print(label_load(train_label) )
print(label_load(train_label).shape ) #(19000, 2)

```

예제14. 위에서 만든 함수 2개를 loader_leaf.py 로 저장하시오!

```

import os
import re
import cv2
import csv
import numpy as np

def image_load(path):
    file_list = os.listdir(path)

    file_name = []
    for i in file_list:
        a = int(re.sub('[^0-9]', '', i)) #^ : not #i가 숫자가 아니면 "(null) 로 바꿔주는 전처리 코드
        file_name.append(a)
    file_name.sort()

    file_res = []
    for j in file_name:
        file_res.append('%s\\%d.jpg' %(path,j) )

    image = []
    for k in file_res:
        img = cv2.imread(k)
        image.append(img)

```

```

return np.array(image)

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)

    labellist=[]
    for i in labeldata:
        labellist.append(i)

    label = np.array(labellist)
    label = label.astype(int)
    label = np.eye(2)[label]
    label = label.reshape(-1,2)

    return label

```

예제15. 이파리 사진을 256x256 에서 32x32 로 resize 하시오! (채널 유지)

예제16. 어제 만들었던 mnist 3층 신경망에 이파리 데이터를 로드하고 학습시키시오!

```

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

path = "C:\\Users\\wicr\\OneDrive\\images\\Leaf\\test"
file_list = os.listdir(path)

for k in file_list:
    img = cv2.imread(path + '\\' + k)
    width, height = img.shape[:2]
    resize_img = cv2.resize(img, (32, 32), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite("C:\\Users\\wicr\\OneDrive\\images\\Leaf\\test_resize" + k, resize_img)

plt.imshow(resize_img)
plt.show()
print(resize_img.shape) #(28, 28, 3)

```

■ 7장. CNN(Convolution Neural Network)

합성곱 신경망

자세한 내용(필독) : <https://cafe.daum.net/oracleoracle/Sedp/199>

합성곱 연산을 컴퓨터로 구현

원본 이미지 한장을 filter 100개로 합성곱 연산을 하면 feature map 이 100개가 생성된다.

합성곱의 역할 : 이미지의 특징을 잡아낸다.

<https://cafe.daum.net/oracleoracle/Sedp/279>

문제117. 합성곱 연산을 위한 rawdata = x 와 filter 을 생성하시오!

```
import numpy as np
x = np.array([1,2,3,0,0,1,2,3,3,0,1,2,2,3,0,1])
x = x.reshape(4,-1)
print(x, end='\n\n')
```

```
filter = np.array([2,0,1,0,1,2,1,0,2])
filter = filter.reshape(3,-1)
print(filter)
```

문제118. 입력이미지 x 에서 아래의 3x3 영역만 가져오시오!

```
x[0:3, 0:3]
```

문제119. 위에서 가져온 3x3 영역과 filter 와 합성곱 연산을 수행하시오!

```
x = x[0:3, 0:3]
x*filter
```

문제120. 위의 곱셈을 한 결과의 원소들을 다 더한 값은 무엇인가?

```
print(np.sum(x[0:3, 0:3] * filter) )
```

문제121. 원본 이미지의 다른 영역(padding=1)과 filter을 곱한 원소들의 합은 무엇인가?

위 과정 반복

문제122. 다른 영역값은?

위 과정 반복

문제123. 위 계산을 하드코딩하지 않고 이중 루프문으로 수행하시오! (함수로도 연습)

#[0. data]

```
import numpy as np
x = np.array([1,2,3,0,0,1,2,3,3,0,1,2,2,3,0,1])
x = x.reshape(4,-1)
print(x, end='\n\n')
```

```
filter = np.array([2,0,1,0,1,2,1,0,2])
filter = filter.reshape(3,-1)
print(filter)
```

#[1. loop]

```
a = []
for i in range(2):
    for j in range(2):
        a.append(np.sum( x[i:3+i, j:3+j] *filter) )
a = np.array(a)
a = a.reshape(2, -1)
a
```

#[2. def 활용, padding은 미적용, 예제 사이즈와 동일한 stride와 rawdata size, filter size에서만 적용가능]

```
def convolution(stride=1, rawdata=None, filter=None):
    k1 = filter.shape[0]
    k2 = filter.shape[1]

    F = []
    F.append(np.sum(rawdata[:k1, :k2]*filter))
    F.append(np.sum(rawdata[:k1, stride:k2+stride]*filter))
    F.append(np.sum(rawdata[stride:k1+stride, :k2]*filter))
    F.append(np.sum(rawdata[stride:k1+stride, stride:k2+stride]*filter))

    F = np.array(F)
    F = F.reshape(2,-2)
    return F
```

```
convolution(stride=1, rawdata=x, filter=filter)
# array([[15, 16],
#        [ 6, 15]])
```

문제124. 위 숫자 4개를 a리스트에 append 하시오!

문제125. 위 a 리스트의 shape를 2x2 로 변경(ndarray로 변경 후 수행)

위 문제들에 대한 해답은 문제 123 번의 답에 포함

문제126. 위 결과에서 편향 3을 더하시오!

#[0. data]

```
import numpy as np
x = np.array([1,2,3,0,0,1,2,3,3,0,1,2,2,3,0,1])
x = x.reshape(4,-1)
print(x, end='\n\n')
```

```
filter = np.array([2,0,1,0,1,2,1,0,2])
filter = filter.reshape(3,-1)
print(filter)
```

#[1. loop]

```
a = []
for i in range(2):
    for j in range(2):
```

```

        a.append(np.sum( x[i:3+i, j:3+j] *filter) +3)
a = np.array(a)
a = a.reshape(2, -1)
a

```

#[2. def 활용, padding은 미적용, 예제 사이즈와 동일한 stride와 rawdata size, filter size에서만 적용가능]

```

def convolution(stride=1, rawdata=None, filter=None, bias=None):
    k1 = filter.shape[0]
    k2 = filter.shape[1]

    F = []
    F.append(np.sum(rawdata[:k1, :k2]*filter))
    F.append(np.sum(rawdata[:k1, stride:k2+stride]*filter))
    F.append(np.sum(rawdata[stride:k1+stride, :k2]*filter))
    F.append(np.sum(rawdata[stride:k1+stride, stride:k2+stride]*filter))

    F = np.array(F)+3
    F = F.reshape(2,-2)
    return F

```

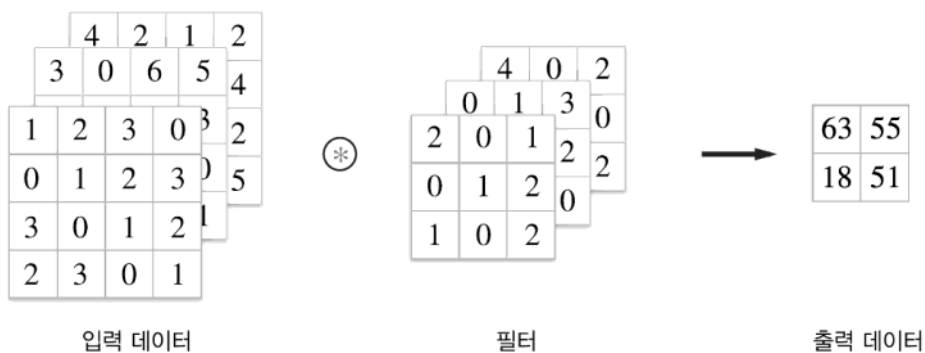
```

convolution(stride=1, rawdata=x, filter=filter, bias=3)
# array([[18, 19],
#        [ 9, 18]])

```

위 문제들에서 수행한 합성곱은 2차원 합성곱이고, 우리가 주로 수행할 합성곱은 3차원 합성곱이다.

●● 3차원 합성곱



```
import numpy as np
```

```

x=np.array([[[1,2,3,0], # --> red 행렬
             [0,1,2,3],
             [3,0,1,2],
             [2,3,0,1]],
            [[2,3,4,1], # --> green 행렬
             [1,2,3,4],
             [4,1,2,3],
             [3,4,1,2]],
            [[3,4,5,2], # --> blue 행렬

```

```

        [2,3,4,5],
        [5,2,3,4],
        [4,5,2,3]]])
print (x.ndim) #3
print (x.shape) #(3,4,4) (차원,행,열)

```

```

f=np.array([[[[2,0,1],
              [0,1,2],
              [1,0,2]],
            [[3,1,2],
              [1,2,3],
              [2,1,3]],
            [[4,2,3],
              [2,3,4],
              [3,2,4]]]])
print ( f.ndim ) #3
print ( f.shape ) #(3,3,3)

```

문제127. 3차원 합성곱을 하기 전 먼저 x 이미지에서 red 행렬만 출력하시오!

```

print('red', x[0], sep='\n', end='\n\n')
print('green', x[1], sep='\n', end='\n\n')
print('blue', x[2], sep='\n', end='\n\n')
red
[[1 2 3 0]
 [0 1 2 3]
 [3 0 1 2]
 [2 3 0 1]]

green
[[2 3 4 1]
 [1 2 3 4]
 [4 1 2 3]
 [3 4 1 2]]

blue
[[3 4 5 2]
 [2 3 4 5]
 [5 2 3 4]
 [4 5 2 3]]

```

문제128. 3차원 합성곱을 수행하시오!

```

a = []
for k in range(3):
    for i in range(2):
        for j in range(2):
            a.append(np.sum( x[k, i:3+i, j:3+j] *f[k,:,:] ) )
a = np.array(a)
a = a.reshape(3,-1)
a2 = a.sum(axis=0)
a3 = a2.reshape(2,-1)

print(a, end='\n\n')
print(a2, end='\n\n')
print(a3)

```

```
[[15 16  6 15]
 [46 48 36 46]
 [95 98 84 95]]

[156 162 126 156]

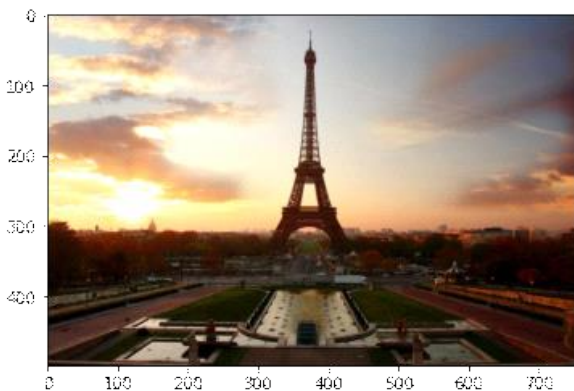
[[156 162]
 [126 156]]
```

a3 의 결과가 합성곱의 결과이며, feature map(특징맵) 이라고 정의되곤 한다.
 # 3차원 raw image 와 3차원 filter 을 이용하여 2차원의 feature map 을 생성하였다.
 # 합성곱을 수행할때마다 filter 로 raw image 의 특징을 추출한다고 생각할 수 있다.

문제129. 유럽.png 를 아래의 3차원 필터로 합성곱 연산을 수행하시오! (우선 유럽.png 를 숫자로 변환)

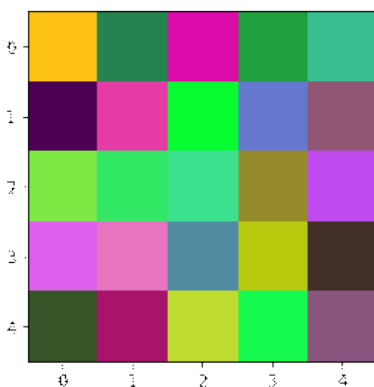
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mping

img = Image.open("C:\\Users\\WW82103\\OneDrive\\images\\practice\\유럽.png")
img_pixel = np.array(img) #ndarray 로 이미지 변환
print(img_pixel.shape) #(499, 756, 3)
plt.imshow(img_pixel) #이미지 시각화
```



문제130. 유럽 이미지와 합성곱을 수행할 필터를 아래와 같이 랜덤으로 생성하시오!

```
import numpy as np
filter3 = np.random.rand(5,5,3)
print(filter3.shape) #(5, 5, 3)
plt.imshow(filter3)
```



위 필터는 랜덤으로 생성된다

문제131. 문제128번에서 사용했던 3차원 합성곱 코드를 이용하여 유럽.png 에 3차원 필터와 합성곱하여 feature map 을 생성하시오!

문제132. 유럽사진의 피쳐맵을 시각화하시오!

Load image & transfer img ---> ndarray

```
from PIL import Image
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

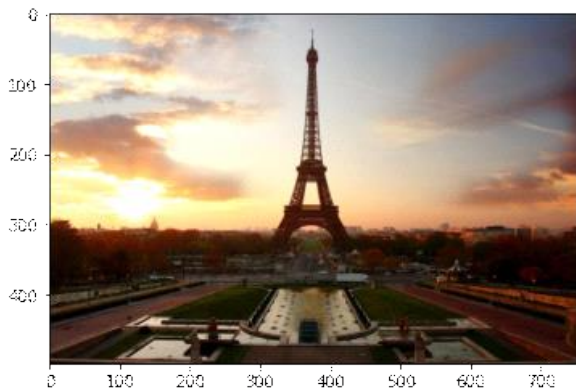
```
import matplotlib.image as mping
```

```
img = Image.open("C:\\Users\\82103\\OneDrive\\images\\practice\\Europe\\Europe.png")
```

```
img_pixel = np.array(img) #ndarray 로 이미지 변환
```

```
print(img_pixel.shape) #(499, 756, 3)
```

```
plt.imshow(img_pixel) #이미지 시각화
```



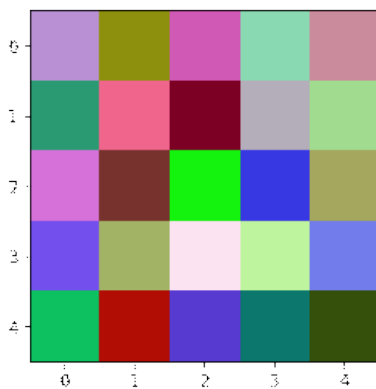
Filter

```
import numpy as np
```

```
filter3 = np.random.rand(5,5,3)
```

```
filter3.shape #(5, 5, 3)
```

```
plt.imshow(filter3)
```



shape 확인

```
print(img_pixel.shape) #(499, 756, 3)
```

```
print(filter3.shape) #(5, 5, 3)
```

Convolution (1 picture)

```
a = []
```

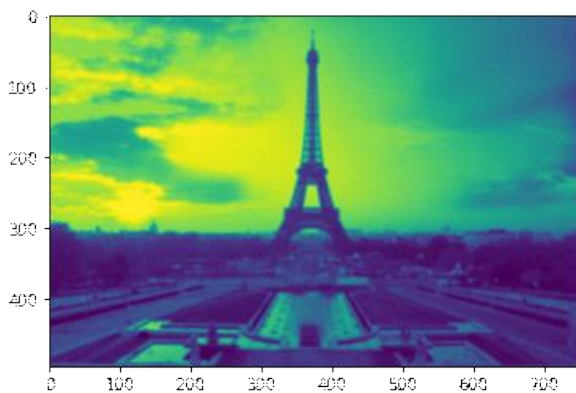
```
row = img_pixel.shape[0]-filter3.shape[0]+1
```



```
col = img_pixel.shape[1]-filter3.shape[1]+1
filter_length = filter3.shape[0]
```

```
for i in range(row):
    for j in range(col):
        b=0
        for k in range(3):
            b += np.sum(img_pixel [i: 5 + i, j:5+j, k] *filter3[:, :, k] )
        a.append(b)
```

```
c = np.array(a).reshape(row,col)
print(c.shape) #(495, 752)
plt.imshow(c)
```



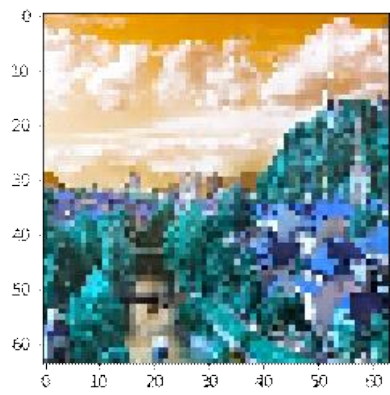
resize

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os
import cv2
import numpy as np
```

```
path = "C:\\Users\\82103\\OneDrive\\Images\\practice\\Europe"
file = os.listdir(path)
```

```
for k in file:
    img = cv2.imread(path + '\\'+ k)
    resize_img1 = cv2.resize(img, (64, 64), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite("C:\\Users\\82103\\OneDrive\\Images\\practice\\resize\\" + k, resize_img1)
```

```
plt.imshow(resize_img1)
plt.show()
print(resize_img1.shape) #(32, 32, 3)
```



#####endLine=====

2021년 3월 1일 월요일 오후 4:52

새 섹션 1 페이지 147

```
[[ 0  0  0  0]
 [ 0 18 19  0]
 [ 0  9 18  0]
 [ 0  0  0  0]]
```

문제133. 아래의 행렬을 제로패딩1 한 후의 결과를 확인하시오!

```
2 0 1
6 7 3
4 5 2
```

```
import numpy as np
x = np.array( [[2,0,1], [6,7,3], [4,5,2] ])
x_pad = np.pad( x, pad_width=1, mode='constant', constant_values=0)
print(x_pad)
[[0 0 0 0 0]
 [0 2 0 1 0]
 [0 6 7 3 0]
 [0 4 5 2 0]
 [0 0 0 0 0]]
```

패딩(padding) 사용의 이유

원본 이미지를 계속 같은 형상으로 유지시키기 위함

convolution layer 을 지날때마다 rawdata 형상이 계속 작아지기 때문에 padding 을 사용해서 형상을 유지

원본이미지와 같은 형상(행렬)으로 피쳐맵을 출력하기 위해서는?

파이썬 날코딩 : padding = ? (직접 계산을 해야 함)

텐서플로우 : padding = same

계산공식 p.234

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

입력크기(H, W)

필터크기(FH, FW)

출력크기(OH, OW)

패딩 : P

스트라이드 : S

$$p = \frac{(OH-1) * S - H + FH}{2} = \frac{(4-1) * 1 - 4 + 3}{2} = 1$$

스트라이드(stride)

<https://cafe.daum.net/oracleoracle/Sedp/281>

[tensorflow 2.0]

입력이미지와 출력이미지의 사이즈를 같게 설정

↑

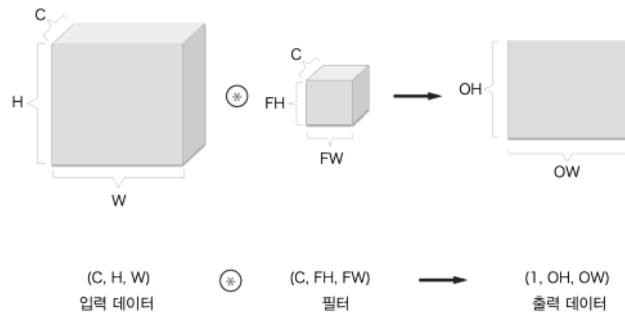
model.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))

\downarrow \downarrow \downarrow
 뉴런의개수 필터사이즈 입력층의 뉴런의 개수

[python] p.251
 conv_params = {'filter_num':32, 'filter_size':3, 'pad':1, 'stride':1 }
 # stride 와 pad 는 직접 계산해서 값을 지정
 # 파이썬 날코딩이 더 불편함

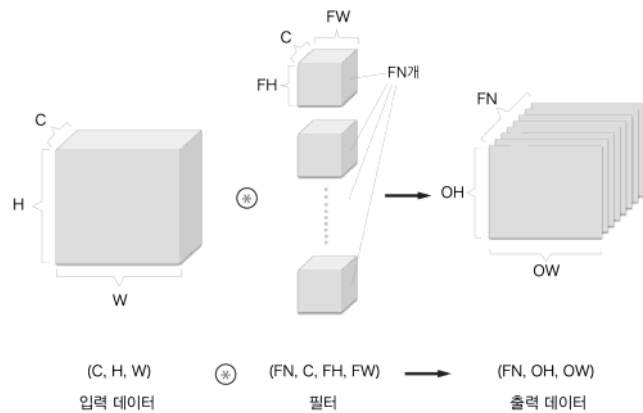
convolution 층에서 하는 작업은 원본이미지를 가지고 여러개의 비슷한 이미지를 만들어내는 작업
 비슷한 이미지들은 feature map 이라고 하며, 개수는 filter_num 과 일치
 필터의 개수 = 뉴런의 개수

[그림 7-10]



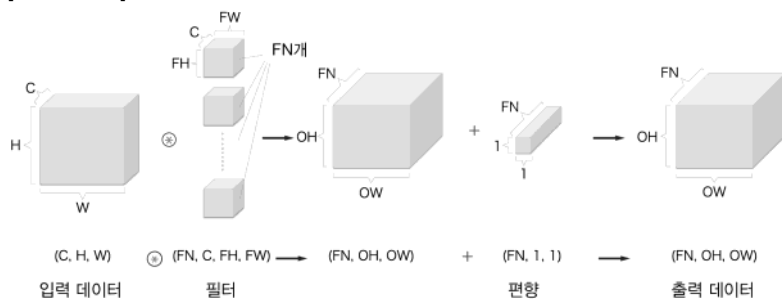
1장의 rawdata, 1개의 필터, 1개의 feature map
 # filter : chanel, FH, HW 로 구성
 # 낱장 1개가 feature map

[그림 7-11]



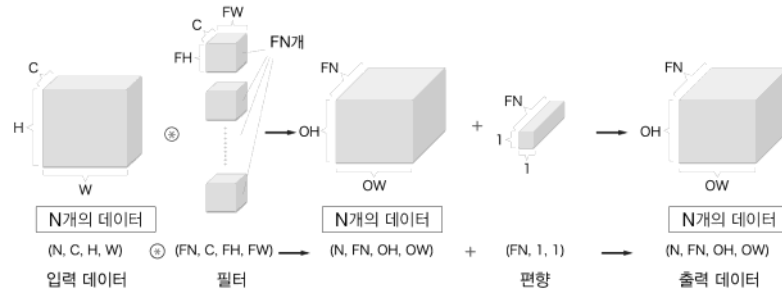
1장의 rawdata, 다수의(FN) 필터, 다수의(FN) feature map
 # filter : FN, channel, FH, HW
 # feature map : FN, OH, OW
 # 낱장 1개가 feature map

[그림 7-12]



add bias
 # bias : broadcasting
 # feature map 이 FN 개가 모인 블록 관찰 가능

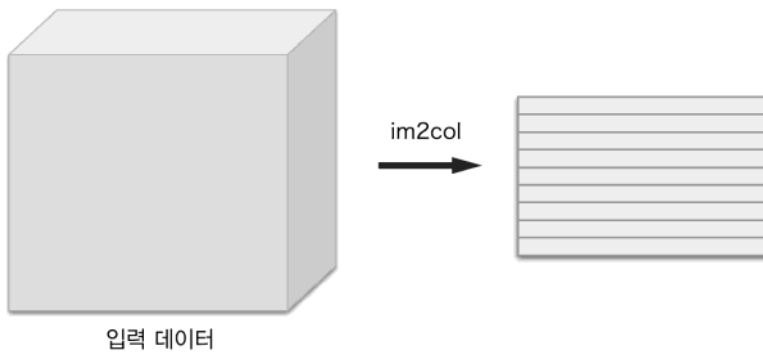
[그림 7-13] p.239



Batch size : N
 # 위 과정들과 동일하며 N의 배치를 추가해서 4차원 데이터 입력 ~ 4차원 데이터 출력
 # 속도향상을 위해 batch 활용

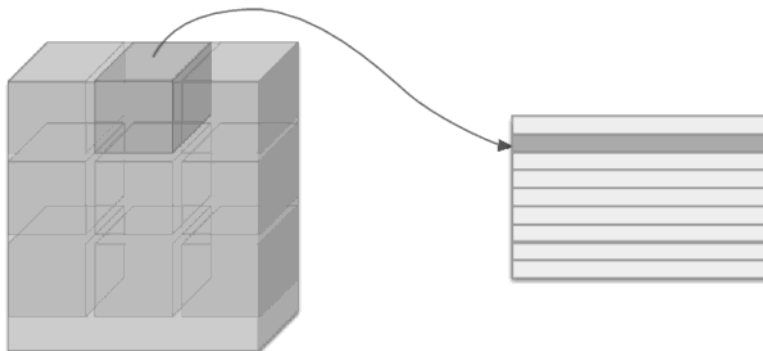
□ im2col p.243

[7-17]

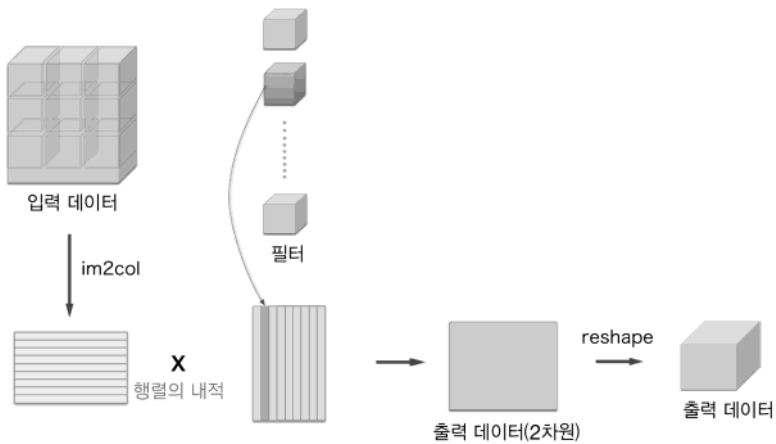


3차원 데이터를(4차원도 가능) 2차원으로 변환
 # 추후 reshape 을 이용해서 원본 형태로 변경

[7-18]



[7-19]



im2col 함수의 과정

im2col 과 반대인 col2im 도 있다

예제1. 사진 한장을 4차원 행렬로 만드시오!

```
import numpy as np
x1 = np.random.rand(1,3,7,7)
print(x1)
print(x1.shape) #(1, 3, 7, 7)
print(x1.ndim) #4
```

예제2. 사진 10장을 4차원으로 생성하시오!

```
import numpy as np
x1 = np.random.rand(10,3,7,7)
print(x1)
print(x1.shape) #(10, 3, 7, 7)
print(x1.ndim) #4
```

예제3. 예제2번의 10장의 사진을 im2 col 함수에 넣어서 2차원 행렬로 변환하시오!

4차원 -----> 2차원
(10, 3, 7, 7) (90, 75)

```
def im2col(input_data, filter_h, filter_w, stride=1, pad=0):
    """다수의 이미지를 입력받아 2차원 배열로 변환한다(평탄화).
```

Parameters

input_data : 4차원 배열 형태의 입력 데이터(이미지 수, 채널 수, 높이, 너비)
 filter_h : 필터의 높이
 filter_w : 필터의 너비
 stride : 스트라이드
 pad : 패딩

Returns

col : 2차원 배열
 """

```

N, C, H, W = input_data.shape
out_h = (H + 2 * pad - filter_h) // stride + 1
out_w = (W + 2 * pad - filter_w) // stride + 1

img = np.pad(input_data, [(0, 0), (0, 0), (pad, pad), (pad, pad)], 'constant')
col = np.zeros((N, C, filter_h, filter_w, out_h, out_w))

for y in range(filter_h):
    y_max = y + stride * out_h
    for x in range(filter_w):
        x_max = x + stride * out_w
        col[:, :, y, x, :] = img[:, :, y:y_max:stride, x:x_max:stride]

col = col.transpose(0, 4, 5, 1, 2, 3).reshape(N * out_h * out_w, -1)
return col

```

결과보기

```

import numpy as np
x1 = np.random.rand(10,3,7,7)
col = im2col(x1, 5, 5, stride=1, pad=0)
print(col.shape) #(90, 75)

```

im2col function

신경망에서 합성곱을 진행하는데 입력되는 4차원 데이터를 2차원 데이터로 차원 축소해서 2차원 필터와 내적 후 4차원으로 reshape

[tensorflow 2.0 code]
model.add(Conv2D(32, (3,3)))
3x3 필터 32개를 raw img(100, 3, 32, 32) 와 합성곱 연산하여 32x100 = 3200 개의 feature map 생성
이때 im2col 함수는 4차원 raw img 를 2차원으로 변경하는 역할
im2col function 은 Conv2D 내에 내장되어 있다.

<https://cafe.daum.net/oracleoracle/Sedp/350>

pdf file link

<https://cafe.daum.net/oracleoracle/SqRM/264>

□ 4차원 필터를 2차원으로 변경

예제1. 아래의 5x5 행렬의 RGB 필터 10개를 3차원으로 변경하시오!

```

(10, 3, 5, 5) -----> (10, 3, 25)
import numpy as np
filter = np.random.rand(10, 3, 5, 5)
filter2 = filter.reshape(10, 3, -1)
filter2.shape #(10, 3, 25)

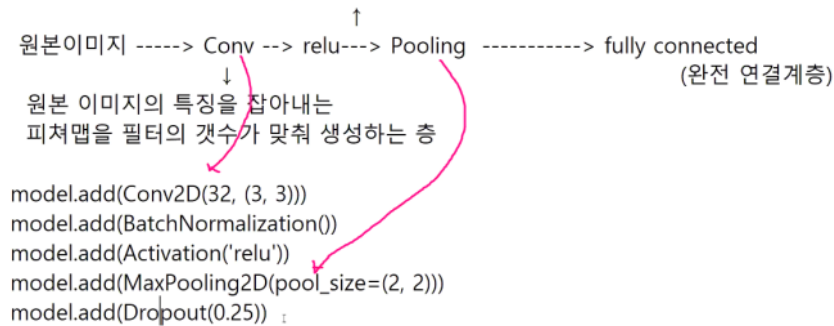
```

예제2. 아래의 4차원 행렬을 2차원 행렬로 변경하시오!


```
(10, 3, 5, 5) -----> (10, 75)
import numpy as np
filter = np.random.rand(10, 3, 5, 5)
filter2 = filter.reshape(10, -1)
filter2.shape # (10, 75)
```

□ 풀링(pooling) layer

피쳐맵 이미지의 각 부분에서 대표값들을 뽑아 사이즈가 작은 이미지를 만드는 역할(이미지가 선명해진다)



풀링층의 역할

convolution 층이 이미지의 특징을 잡아내는 역할을 한다면 pooling 층은 feature map 이미지를 선명하게 만드는 역할을 수행

풀링의 종류

1. max pooling : 이미지를 선명하게 만드는 효과
2. average pooling : 이미지를 부드럽게 만드는 효과(거친 이미지를 부드럽게 -)
3. stochastic poolig : convolution window 에서 임의확률로 값을 선정. 효과는?

예제1. 아래의 행렬을 생성하시오!

```
21 8 8 12
12 19 9 7
8 10 4 3
18 12 9 10
import numpy as np
x = np.array([21, 8, 8, 12, 12, 19, 9, 7, 8, 10, 4, 3, 18, 12, 9, 10])
x = x.reshape(4, -1)
x
```

예제2. 아래의 행렬에서 max pooling 을 시도해서 아래의 결과를 출력하시오!

```
import numpy as np
x = np.array([21, 8, 8, 12, 12, 19, 9, 7, 8, 10, 4, 3, 18, 12, 9, 10])
x = x.reshape(1, 1, 4, -1)
print(x.shape) # (1, 1, 4, 4)
x
Out [28]: array([[[[21, 8, 8, 12],
[12, 19, 9, 7],
[ 8, 10, 4, 3],
[18, 12, 9, 10]]]])
```

```
a = im2col(x, 2, 2, stride=2, pad=0)
print(a)
```

```
[[21.  8. 12. 19.]  
 [ 8. 12.  9.  7.]  
 [ 8. 10. 18. 12.]  
 [ 4.  3.  9. 10.]]
```

```
b = np.max(a, axis=1)
```

```
b = b.reshape(2,-1)
```

```
b
```

```
Out[31]: array([[21., 12.],  
               [18., 10.]])
```

```
#####endLine=====
```

##0322

2021년 3월 1일 월요일 오후 4:52

■ Colab - 이파리 데이터 분류

1. 구글 코랩 가입
2. 새 노트를 열고 GPU를 사용할 수 있도록 설정
3. 카페 게시판 274 번의 두 개의 파일을 download
4. 이파리분류.ipynb 노트를 코랩에서 오픈

```
es4# mkdir train_resize3
es4# mkdir test_resize3
es4#
es4# mv *.jpg ./test_resize3/
```


#####endLine=====

##0323_re

2021년 3월 1일 월요일 오후 4:52

■ 밑바닥 딥러닝 복습

- 1장. numpy 사용법
- 2장. 퍼셉트론
- 3장. 3층 신경망 구현(학습 x)
- 4장. 2층 신경망 구현(학습o) - 수치미분을 이용
- 5장. 2층 신경망 구현(학습o) - 오차역전파를 이용
- 6장. 언더피팅과 오버피팅을 방지하는 방법
- 7장. CNN 층
- 8장. 딥러닝 역사를 텐서플로우 2.0을 이용해서 구현해보기

* 딥러닝을 발전 시킨 유명한 신경망들

- 1. CNN을 사용한 신경망 구현 (fashion mnist)
- 2. VGG 신경망

* 신경망을 사용할 때 불편해서 구현하게 된 기타 기능들

- 1. 이미지 증식 시키기
- 2. 케라스의 콜백 기능

* 신경망 활용 홈페이지 생성 (기본 포트폴리오)

■ CNN을 사용한 신경망 구현(수지와 설현사진 분류)

Fashion mnist 전체코드를 가져와서 수정한다.

- 1. 제규어와 얼룩말 데이터 2000장(한클래스당 1000장)을 받고 코랩에 올리기

- 2. gpu를 사용하는지 확인한다.

!nvidia-smi -L

- 3. drive를 마운트 시킨다.

```
from google.colab import drive
drive.mount('/content/drive')
```

- 4. 테스트데이터는 /content/gdrive/MyDrive/samples6/train_resize2
훈련데이터는 /content/gdrive/MyDrive/samples6/train_resize2

터미널 여는법:

```
!pip install kora
from kora import console
console.start() # and click lin
```

5. 얼룩말과 제규어 사진을 가지고 홈페이지 구현을 위한 신경망 모델 생성하는 코드를 수행한다.

아래의 코드를 코랩에서 엽니다.

1. 제규어와 지브라.ipynb

■ 텐서 플로우2.x의 전의 학습 기능

전의학습은 사전에 학습된 네트워크의 가중치 또는 이미지넷 대회에서 우승한 유명한 신경망 모델을 가져와서 사용하는 기능을 말한다.

1. 모델을 변형하지 않고 그대로 사용하는 방법

예 : `from keras.applications import VGG16`

```
vgg16 = VGG16(weights = 'imagenet', input_shape = (32, 32, 3), include_top = False)
```

설명 : `weights` : imagenet 데이터를 학습시킨 가중치의 사용여부를 결정한다.

기본값은 None (1000개의 이미지 사진들이 있고 그 사진들을 학습시킨 가중치)

`input_shape` = 입력데이터의 사이즈를 기술한다.

`include_top` = 완전연결계층의 모델의 분류기를 내가 직접 기술할지 말지를 결정한다.

(False로 되어있으면 내가 직접 완전연결계층을 짜겠다)

■ 텐서 플로우 2.0의 장점

1. 실행모드로 실행할 수 있다. (파이썬처럼 실행이 가능하다)
2. 이미지넷 대회에서 우승한 유명한 신경망 모델 설계와 가중치 파일을 내가 분류하고자하는 신경망 코드에 쉽게 가져올 수 있다. (전이 학습)
3. 이미지를 증식 시키고 증식 시킨 이미지들을 쉽게 신경망에 입력할 수 있다.
4. EarlyStopping 기능을 구현할 수 있다.

■ EarlyStopping 기능

EarlyStopping 콜백을 직역하면 '이른 멈춤'이다. 즉 모델 학습시 지정된 기간동안 모니터링하는 평가지표에서 성능향상이 일어나지 않은 경우 학습을 중단한다. 주로 많이 사용하는 콜백인자는 다음과 같다.

예제:

```
EarlyStopping( monitor='val_loss', patience=2, verbose=0, mode='auto')
```

`monitor` : 모니터링 할 평가지표를 설정한다. (오차: `val_loss`, 정확도 : `val_acc`)

`verbose` : 콜백의 수행과정 노출여부를 결정한다.

0 : 아무런 표시를 안하겠다.

1 : 프로그래스 바(progress bar)를 출력

2 : 에폭마다 수행과정을 출력

`patience` : 지정한 수만큼의 기간에서 평가지표의 향상이 일어나지 않을 경우 학습을 중단하겠다.

`patience = 5` 를 썼다면 5번은 참아주겠다는 뜻이다.

구현 :

```
callbacks = [ EarlyStopping( monitor='val_loss', patience=2, verbose=0, mode='auto') ]
```

```
model.fit( x_train, y_train,
          batch_size = 32,
          validation_data=( x_val, y_val ),
          epochs = 10,
          callbacks = callbacks )
```

```
model.summary()
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
filepath = 're_w2.h5'
```

```
callbacks = [ ModelCheckpoint( filepath = filepath , monitor = 'val_loss' , verbose=1, save_best_only=True ) ]
```

훈련 데이터는 정확도 0.99, 검증 데이터와 테스트 데이터의 정확도 : 0.91

■ R shiny를 이용해서 신경망을 쉽게 이용하는 방법

카페 글 285번 참조

<https://cafe.daum.net/oracleoracle/SgRM/285>

검증 데이터셋을 만듭니다.

```
from sklearn.model_selection import train_test_split
np.random.seed(111)
# 훈련/테스트 데이터를 0.7/0.3의 비율로 분리합니다.
```

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
                                                  test_size = 0.2, random_state = 777)
```

```
setwd("D:\W\Wys267")
```

```
packages <- c('imager', 'shiny', 'jpeg', 'png', 'reticulate', 'devtools')
```

```
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packagesvx ())))
}
```

```
if (length(setdiff("keras", rownames(installed.packages()))) > 0) {
  devtools::install_github("rstudio/keras")
}
```



```

require(imager)
require(shiny)
require(jpeg)
require(png)
library(reticulate)
library(keras)

#setwd(tempfile())
#setwd("/Users/aiden/Desktop/data/cifar10_densenet")

load("envir.RData")
model <- load_model_hdf5("re_w3.h5")

synsets <- readLines("synset.txt")

server <- shinyServer(function(input, output) {
  ntext <- eventReactive(input$goButton, {
    print(input$url)
    if (input$url == "http://") {
      NULL
    } else {
      tmp_file <- tempfile()
      download.file(input$url, destfile = tmp_file, mode = 'wb')
      tmp_file
    }
  })

  output$originImage = renderImage({
    list(src = if (input$tabs == "Upload Image") {
      if (is.null(input$file1)) {
        if (input$goButton == 0 || is.null(ntext())) {
          '904_google.jpg'
        } else {
          ntext()
        }
      } else {
        input$file1$datapath
      }
    } else {
      if (input$goButton == 0 || is.null(ntext())) {
        if (is.null(input$file1)) {
          '904_google.jpg'
        } else {
          input$file1$datapath
        }
      } else {
        ntext()
      }
    },
    title = "Original Image")
  }, deleteFile = FALSE)

```

```

output$res <- renderText({
  src = if (input$tabs == "Upload Image") {
    if (is.null(input$file1)) {
      if (input$goButton == 0 || is.null(n$text())) {
        '904_google.jpg'
      } else {
        n$text()
      }
    } else {
      input$file1$datapath
    }
  } else {
    if (input$goButton == 0 || is.null(n$text())) {
      if (is.null(input$file1)) {
        '904_google.jpg'
      } else {
        input$file1$datapath
      }
    } else {
      n$text()
    }
  }
})

img <- load.image(src)
plot(img)
img <- image_load(src, target_size = c(32,32))
img
x <- image_to_array(img)
# ensure we have a 4d tensor with single element in the batch dimension,
x <- array_reshape(x, c(1, dim(x)))

# normalize
x[,,,1] <- x[,,,1] / 255
x[,,,2] <- x[,,,2] / 255
x[,,,3] <- x[,,,3] / 255

# predict
preds <- model %>% predict(x)

# output result as string
max.idx <- order(preds[1,], decreasing = TRUE)[1]
result <- synsets[max.idx]
res_str <- ""
tmp <- strsplit(result[1, " "][[1]]
res_str <- paste0(res_str, tmp[2])
res_str
})
})

```

```

require(imager)
require(shiny)
require(jpeg)
require(png)

ui <- shinyUI(
  fluidPage(
    includeCSS("bootstrap.css"),

    pageWithSidebar(
      headerPanel(title = '수지설현 using DenseNet',
        windowTitle = 'Image Classification(수지설현) using DenseNet'),

      fluidRow(
        column(1,
          column(9,
            tabsetPanel(
              id = "tabs",
              tabPanel("Upload Image",
                fileInput('file1', 'Upload a PNG / JPEG File:')),
              tabPanel(
                "Use the URL",
                textInput("url", "Image URL:", "http://"),
                actionButton("goButton", "Go!")
              )
            ),
            h3(titlePanel("DESCRIPTION - 수지설현 분류")),
            h3(titlePanel("수지와 설현"))

          ),
          column(2)
        ),

        mainPanel(
          h3("Image"),
          tags$hr(),
          imageOutput("originImage", height = "auto"),
          tags$hr(),
          h3("What is this?"),
          tags$hr(),
          verbatimTextOutput("res")
        )
      )
    )
  )

shinyApp(ui = ui, server = server)

```


#####endLine=====

##0324

2021년 3월 1일 월요일 오후 4:52

R notepad

<https://cafe.daum.net/oracleoracle/SeFi/39>

#####endLine=====