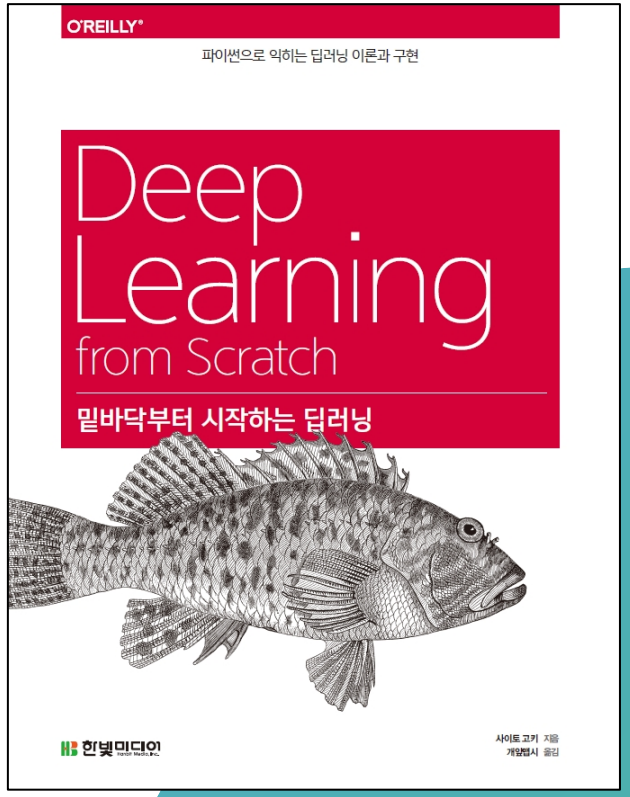
CHAPTER 1 헬로 파이썬

밑바닥부터 시작하는 딥러닝



홍익대학교 컴퓨터공학과
박 준

이 책의 학습 목표

- CHAPTER 1 파이썬에 대해 간략하게 살펴보고 사용법 익히기
- CHAPTER 2 퍼셉트론에 대해 알아보고 퍼셉트론을 써서 간단한 문제를 풀어보기
- CHAPTER 3 신경망의 개요, 입력 데이터가 무엇인지 신경망이 식별하는 처리 과정 알아보기
- CHAPTER 4 손실 함수의 값을 가급적 작게 만드는 경사법에 대해 알아보기
- CHAPTER 5 가중치 매개변수의 기울기를 효율적으로 계산하는 오차역전파법 배우기
- CHAPTER 6 신경망(딥러닝) 학습의 효율과 정확도를 높이기
- CHAPTER 7 CNN의 메커니즘을 자세히 설명하고 파이썬으로 구현하기
- CHAPTER 8 딥러닝의 특징과 과제, 가능성, 오늘날의 첨단 딥러닝에 대해 알아보기

Contents

Contents

1.5.1 넘파이 가져오기

넘파이는 외부 라이브러리.
여기서 말하는 '외부'는 표준 파이썬에는 포함되지 않는다는것.
우선 넘파이 라이브러리를 쓸 수 있도록 가져와야 import 해야 한다.

```
>>> import numpy as np
```

1.5.2 넘파이 배열 생성하기

넘파이 배열 을 만들 때는 np . array () 메서드를 이용.
np . array ()는 파이썬의 리스트를 인수로 받아 넘파이 라이브러리가 제공하는 특수한 형태의 배열(numpy . ndarray)을 반환.

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> print(x)
[1. 2. 3.]
>>> type(x)
<class 'numpy.ndarray'>
```

1.5.3 넘파이의 산술 연산

```
>>> x = np.array([1.0, 2.0, 3.0])
>>> y = np.array([2.0, 4.0, 6.0])
>>> x + y  # 원소별 덧셈
array([ 3.,  6.,  9.])
>>> x - y
array([ -1.,  -2.,  -3.])
>>> x * y  # 원소별 곱셈
array([ 2.,  8.,  18.])
>>> x / y
array([ 0.5,  0.5,  0.5])
```

여기에서 주의할 점은 배열 x 와 y 의 원소 수가 같다는 것(둘 다 원소를 3 개씩 갖는 1 차원 배열). x 와 y 의 원소 수가 같다면 산술 연산은 각 원소에 대해서 행해짐.
원소 수가 다르면 오류가 발생하니 원소 수 맞추기는 중요.
참고로, '원소별'이라는 말은 영어로element - wise 라고 합니다. 예컨대 '원소별 곱셈'은 element - wise product 라고 한다.

1.5.4 넘파이의 N차원 배열

```
>>> A = np.array([[1, 2], [3, 4]])
>>> print(A)
[[1 2]
 [3 4]]
>>> A.shape
(2, 2)
>>> A.dtype
dtype('int64')
```

방금 2×2의 A라는 행렬을 작성.
행렬의 형상*은 shape으로, 행렬에 담긴 원소의 자 료형은 dtype으로 알 수 있다

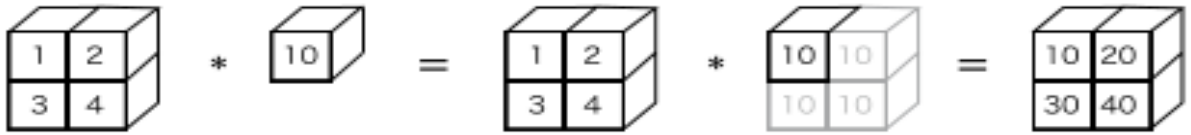
```
>>> B = np.array([[3, 0], [0, 6]])
>>> A + B
array([[ 4,  2],
       [ 3, 10]])
>>> A * B
array([[ 3,  0],
       [ 0, 24]])
```

형상이 같은 행렬끼리면 행렬의 산술 연산도 대응하는 원소별로 계산된다.

1.5.5 브로드캐스트

넘파이에서는 형상이 다른 배열끼리도 계산할 수 있다.
앞의 예에서는 2 × 2 행렬 A 에 스칼라값 10 을 곱했습니다. 이때 [그림 1 - 1]과 같이 10 이라는 스칼라값이 2 × 2 행렬로 확대된후 연산이 이뤄진다.
이 기능을 브로드캐스트 broadcast 라고 합니다.

그림 **1-1** 브로드캐스트의 예 : 스칼라값인 10이 2×2 행렬로 확대된다.



다른 예를 살펴봅시다.

```
>>> A = np.array([[1, 2], [3, 4]])
>>> B = np.array([10, 20])
>>> A * B
array([[ 10, 40],
       [ 30, 80]])
```

1.5.6 원소 접근

```
>>> X = np.array([[51, 55], [14, 19], [0, 4]])
>>> print(X)
[[51 55]
 [14 19]
 [ 0  4]]
>>> X[0]        # 0행
array([51, 55])
>>> X[0][1]     # (0, 1) 위치의 원소
55
```

for 문으로도 각 원소에 접근할 수 있습니다

```
>>> for row in X:
...     print(row)
...
[51 55]
[14 19]
[0 4]
```

1.6.1 단순한 그래프 그리기

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

# 데이터 준비
x = np.arange(0, 6, 0.1) # 0에서 6까지 0.1 간격으로 생성
y = np.sin(x)

# 그래프 그리기
plt.plot(x, y)
plt.show()
```

이 코드에서는 넘파이의 arange 메서드로 [0 , 0 . 1 , 0 . 2 , ..., 5 . 8 , 5 . 9]라는 데이터를 생성하여변수 x 에 할당.
그다음 줄에서는 x 의 각 원소에 넘파이의 sin 함수인 np . sin ()을 적용하여 변수 y 에 할당.
이제 x 와 y 를 인수로 plt . plot 메서드를 호출해 그래프를 그림. 마지막으로 plt . show ()를 호출해 그래프를 화면에 출력하고 끝납니다. 이 코드를 실행하면 [그림 1 - 3]의 이미지가 그려진다.

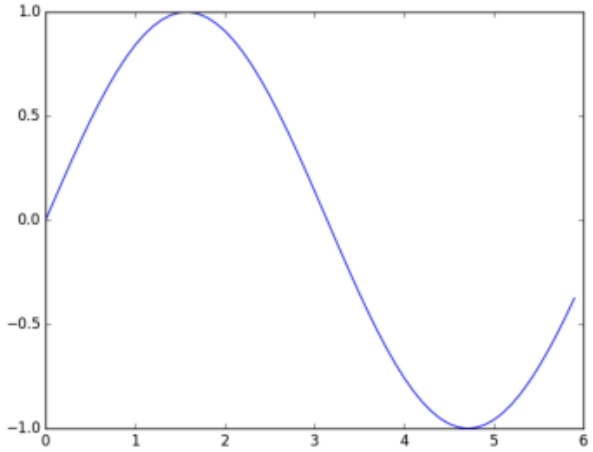


그림 **1-3** sin 함수 그래프

1.6.2 pyplot의 기능

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

# 데이터 준비
x = np.arange(0, 6, 0.1) # 0에서 6까지 0.1 간격으로 생성
y1 = np.sin(x)
y2 = np.cos(x)

# 그래프 그리기
plt.plot(x, y1, label="sin")
plt.plot(x, y2, linestyle="--", label="cos") # cos 함수는 점선으로 그리기
plt.xlabel("x") # x축 이름
plt.ylabel("y") # y축 이름
plt.title('sin & cos') # 제목
plt.legend()
plt.show()
```

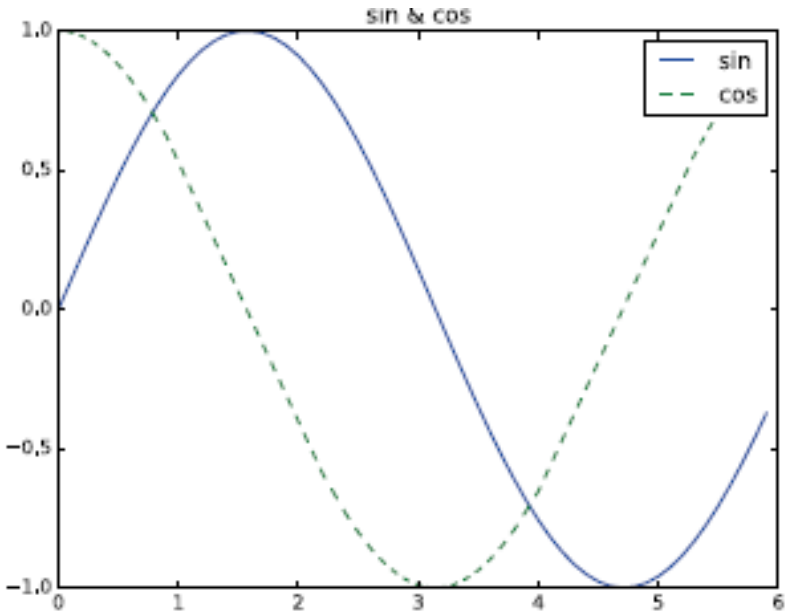


그림 **1-4** sin 함수와 cos 함수 그래프

결과는 [그림 1-4]와 같습니다. 그래프의 제목과 축 이름이 보일것이다.

1.6.3 이미지 표시하기

pyplot 에는 이미지를 표시해주는 메서드인 imshow ()도 준비되어있다.
이미지를 읽어들일 때는 matplotlib . image 모듈의 imread () 메서드를 이용한다

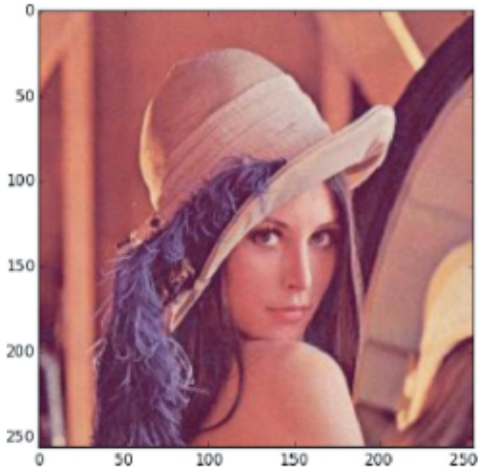
```
import matplotlib.pyplot as plt
from matplotlib.image import imread

img = imread('lena.png') # 이미지 읽어오기(적절한 경로를 설정하세요!)

plt.imshow(img)
plt.show()
```

이 코드를 실행하면 [그림 1 - 5]처럼 읽어들인 이미지가 표시됩니다.

그림 1-5 이미지 표시하기



실습

Exponential 함수와 -log 함수를 그리기
0부터 1까지 0.01 간격으로

np.exp() np.log() 사용