합성곱 연산

최석재 *lingua@naver.com*

합성곱 신경망 *Convolutional Neural Network*

- 합성곱 신경망은 입력 데이터에 대해 필터를 적용하여 합성곱된 새로운 결과를 만든다
- 시각 데이터 처리에 효과적이어서 이미지나 비디오 데이터의 패턴 인식에 자주 사용된다

- 필터 적용 즉, 합성곱(Convolution) 연산으로 데이터에서 중요한 특징을 자동으로 추출하며,
- 추출된 특징은 여러 층(Layer)으로 전달되어 분류, 탐지, 세분화 등의 작업에 사용된다
- CNN은 특정 패턴의 위치 변화에 강건(robust)하여 이미지 내 객체 인식에 유리하다

- CNN은 합성곱 신경망 외에도 활성화 함수 층, Pooling 층, 완전연결 층 등을 포함한다

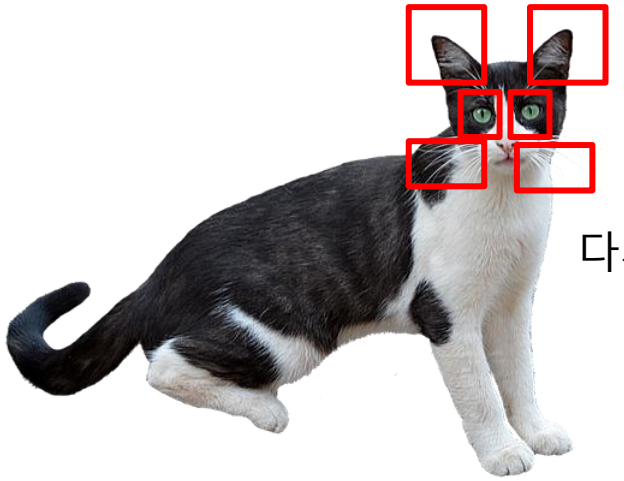




주요 특징 파악하기

- 눈이나 털의 색깔, 개별적인 점 또는 무늬 등은 제외하면서
- 고양이의 일반적인 특징을 잘 추출해야 고양이라는 객체를 잘 탐지할 수 있다

삼각형이 2개인데 서있다



그 아래에는 날카로운 점과
둥근 원이 2개씩 있다

다시 그 아래에는 가로줄이 있다

합성곱 연산

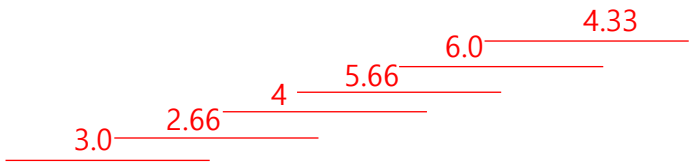
- 합성곱은 두 함수를 결합하는 연산
- 한 함수가 다른 함수 위를 움직이며 값을 곱한 후, 그 곱셈 결과들을 모두 더하는 과정

- 이미지에 합성곱 연산이 이루어지면
- 데이터의 중요한 정보를 유지하면서 노이즈를 줄일 수 있게 된다

- 합성곱 연산의 단순화된 모습으로 이동평균을 들 수 있다

단순이동평균 *Simple Moving Average*

- 이동평균이란 데이터를 따라가며 평균을 구하는 것을 말한다
- 데이터를 3개씩 묶어 산술평균을 구해보면 다음과 같다



[5, 3, 1, 4, 7, 6, 5, 2]

이동평균 : [3.0, 2.66, 4.0, 5.66, 6.0, 4.33]

(5+3+1)/3 = 3.0

단순이동평균 구현

- 단순이동평균을 코드로 구현하면 다음과 같다

- data = [5, 3, 1, 4, 7, 6, 5, 2]
- h = 1/3

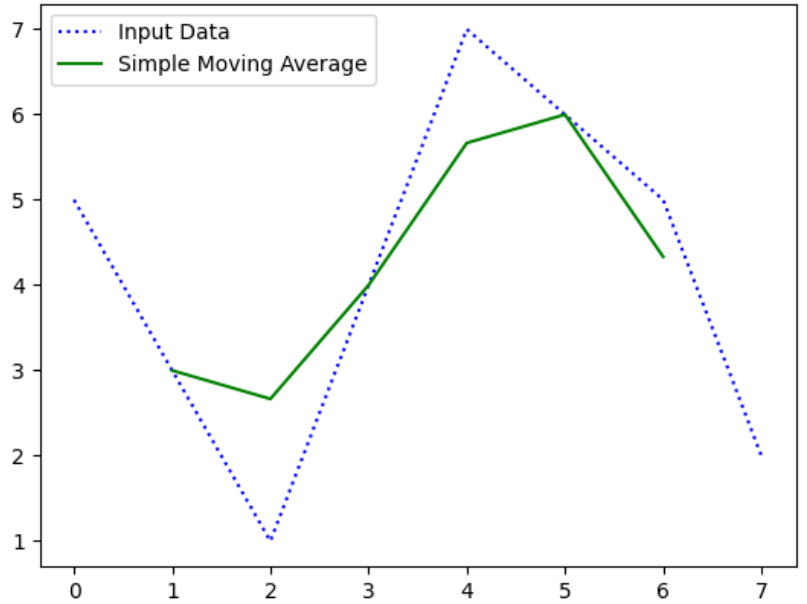
- mavg = [None] # 원본 데이터와 같은 길이를 갖게 하도록 None 입력
- for i in range(1, len(data) - 1): # 시작점과 끝점에서 [i-1]과 [i+1] 문제를 해결하기 위해 길이 제한
  mavg.append(data[i-1]*h + data[i]*h + data[i+1]*h) # [이전, 현재, 다음]의 평균
- mavg.append(None) # 원본 데이터와 같은 길이를 갖게 하도록 None 입력

원본 데이터(data)를 입력 데이터, h를 가중치로 볼 때,
각각의 입력 데이터와 가중치의 곱셈 결과를 합산하는 과정이
합성곱 연산의 기본적인 모습을 보여준다

- print(mavg)

```
[None, 3.0, 2.6666666666666665, 3.9999999999999996, 5.666666666666666, 6.0, 4.333333333333333, None]
```

단순이동평균 시각화



- 이를 시각화하면 다음과 같다
- 데이터가 갖는 패턴이 유지되는 것을 볼 수 있다

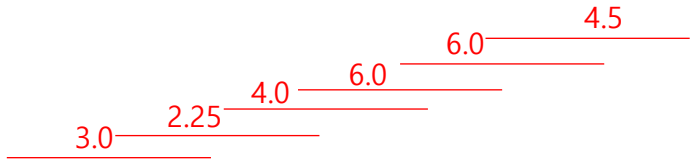
- import matplotlib.pyplot as plt

- plt.plot(data, color="blue", linestyle="dotted", label="Input Data")
- plt.plot(mavg, color="green", linestyle="solid", label="Simple Moving Average")
- plt.legend()
- plt.show()

가중이동평균 *Weighted Moving Average*

- 이미지 데이터에서는 데이터 포인트마다 동일한 가중치가 적용되지는 않는다
- 특징을 잘 포착할 수 있는 포인트에는 높은 가중치가, 그렇지 않은 곳에는 낮은 가중치가 설정된다

- 데이터 포인트마다 다른 가중치를 적용하여 평균을 구하는 것은 가중이동평균
- 현재 주목하고 있는 데이터에는 0.5의 가중치를 주고,
- 그 앞과 뒤의 데이터에는 0.25의 가중치를 주고 평균을 구해본다



[5, 3, 1, 4, 7, 6, 5, 2]

가중이동평균 : [3.0, 2.25, 4.0, 6.0, 6.0, 4.5]

5x0.25 + 3x**0.5** + 1x0.25 = 3.0

가중이동평균 구현

- 앞의 경우를 코드로 구현하면 다음과 같다

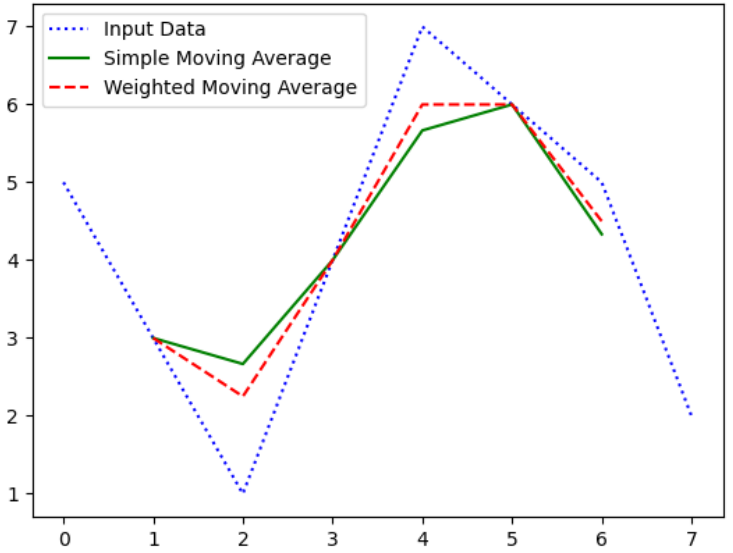
- data = [5, 3, 1, 4, 7, 6, 5, 2]
- h = [0.25, 0.5, 0.25] # 이러한 배열을 필터 커널(Filter kernel) 또는 필터, 커널, 마스크라고 부른다

- w_mavg = [None] # 원본 데이터와 같은 길이를 갖게 하도록 None 입력
- for i in range(1, len(data) - 1): # 시작점과 끝점에서 [i-1]과 [i+1] 문제를 해결하기 위해 길이 제한
  w_mavg.append(data[i-1]*h[0] + data[i]*h[1] + data[i+1]*h[2]) # 데이터 포인트마다 다른 가중치 배당
  0.25 **0.5** 0.25
- w_mavg.append(None) # 원본 데이터와 같은 길이를 갖게 하도록 None 입력

- print(w_mavg)

```
[None, 3.0, 2.25, 4.0, 6.0, 6.0, 4.5, None]
```

가중이동평균 시각화



- 역시 데이터가 갖는 패턴이 유지되는 것을 볼 수 있다

- import matplotlib.pyplot as plt

- plt.plot(data, color="blue", linestyle="dotted", label="Input Data")
- plt.plot(mavg, color="green", linestyle="solid", label="Simple Moving Average")
- plt.plot(w_mavg, color="red", linestyle="dashed", label="Weighted Moving Average")
- plt.legend()
- plt.show()

단순이동평균은 동일한 가중치를 부여하여 대상 데이터 포인트들의 변화를 비교적 평탄하게 만들지만,
가중이동평균은 현재 주목하고 있는 데이터 포인트에 더 높은 가중치를 부여하여 데이터의 변화를 더 잘 반영할 수 있다

경계 찾기에 활용

- 이미지에서 경계(Edge)에 해당하는 곳은 화소의 차이가 크게 된다
- 서로 다른 가중치를 적절히 사용하면 경계를 보다 뚜렷이 찾을 수 있다

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 116 125 171 255 255 150  93   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 169 253 253 253 253 253 253 218  30   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 169 253 253 253 213 142 176 253 253 122   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  52 250 253 210  32  12   0   6 206 253 140   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  77 251 210  25   0   0   0 122 248 253  65   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  31  18   0   0   0   0 209 253 253  65   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 117 247 253 198  10   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  76 247 253 231  63   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 128 253 253 144   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 176 246 253 159  12   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  25 234 253 233  35   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 198 253 253 141   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  78 248 253 189  12   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  19 200 253 253 141   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 134 253 253 173  12   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 248 253 253  25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 248 253 253  43  20  20  20  20   5   0   5  20  20  37 150 150 150 147  10   0]
 [  0   0   0   0   0   0   0   0 248 253 253 253 253 253 253 253 168 143 166 253 253 253 253 253 253 253 123   0]
 [  0   0   0   0   0   0   0   0 174 253 253 253 253 253 253 253 253 253 253 253 249 247 247 169 117 117  57   0]
 [  0   0   0   0   0   0   0   0   0 118 123 123 123 166 253 253 253 155 123 123  41   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

경계 찾기에 활용

- 경계를 잘 찾기 위하여 주목 데이터의 앞과 뒤에 음수값을 줄 수 있다

경계 직전에 있는 데이터를 처리할 때, 0*-1 + 0*2 + 10*-1 = -10
경계 시작점에 있는 다음 데이터는 0*-1 + 10*2 + 10*-1 = 10

- data = [0, 0, 0, 10, 10, 10, 10, 10, 10, 10, 0, 0, 0]
- h = [-1, 2, -1]

- w_mavg = [None]
- for i in range(1, len(data) - 1):
  w_mavg.append(data[i-1]*h[0] + data[i]*h[1] + data[i+1]*h[2])
- w_mavg.append(None)

- print(w_mavg)

[None, 0, -10, 10, 0, 0, 0, 0, 0, 10, -10, 0, None]

변화가 시작되거나 끝나는 경계 부분에서 크게 움직이게 된다

경계 찾기 시각화

- import matplotlib.pyplot as plt

- plt.plot(data, color="blue", label="Input Data", marker='o')
- plt.plot(w_mavg, color="red", label="Weighted Moving Average", marker='o')
- plt.legend()
- plt.show()

원본 데이터에 비하여
경계 부분에서의 변화가 더 커졌다