7장 합성곱 신경망(CNN)

합성곱 신경망(Convolutional Neural Network)

→ 이미지 인식, 음성 인식 등 많이 사용됨. 특히 이미지 인식 분야에서 기초가 됨

전체 구조

지금까지 본 신경망 : 인접하는 계층의 모든 뉴런과 결합

⇒ 완전연결(fully connected)

⇒ 완전히 연결된 계층 = Affine 계층

Affine 계층

- 신경망의 순전파 때 수행하는 행렬의 곱 = 어파인 변환 ⇒ 어파인 변환을 수행하는 처리를 Affine 계층으로 구현
- 행렬의 곱 계산 → np.dot 이용

  ⇒ 차원의 원소 수를 일치시키는 게 핵심

완전연결 신경망

→ affine 계층 뒤에 활성화함수를 갖는 ReLU(or sigmoid) 계층이 이어짐

CNN 구조

→ 합성곱 계층(convolutional layer)과 풀링 계층(pooling layer)가 추가됨

Conv - ReLU - (Pooling) 흐름으로 연결

출력에 가까운 층에서는 Affine - ReLU 구성 사용 가능

마지막 출력 계층에선 Affine - Softmax 그대로 사용

합성곱 계층

완전 연결 계층의 문제점

완전 연결 계층

이미지는 통상 3차원 데이터(세로, 가로, 채널(색상))지만 완전연결 계층에 입력시 3차원 데이터를 1차원 데이터로 평탄화

⇒ 데이터의 형상이 무시됨

이미지는 3차원 형상, 공간적 정보가 담겨있음

⇒ 3차원 속에서 의미를 갖는 본질적 패턴이 숨어있음

→ 그러나 완전연결 계층은 형상을 무시하고 모든 입력 데이터를 동등한 뉴런(같은 차원의 뉴런)으로 취급해 형상에 담긴 정보를 살릴 수 없음

합성곱 계층

합성곱 계층은 형상을 유지!

→ 이미지처럼 형상을 가진 데이터를 제대로 이해할 가능성이 있는 것!

합성곱 계층의 입출력 데이터 = 특징 맵 (feature map)

- 입력 데이터 = 입력 특징 맵 input feature map
- 출력 데이터 = 출력 특징 맵 output feature map

  이 책에선 '입출력 데이터'와 '특징 맵'을 같은 의미로 사용

합성곱 연산

입력 데이터에 필터(=커널)을 적용해서 계산

필터의 윈도우를 일정 간격으로 이동해가며 입력데이터에 적용

→ 대응하는 원소끼리 곱한 후 그 총합을 구해서(=FMA) 출력의 해당 장소에 저장

CNN에서는 필터의 매개변수가 그동안의 '가중치'에 해당함

CNN에도 편향이 존재

- 편향은 필터를 적용한 후에 데이터에 더해짐
- 편향은 항상 하나(1x1) → 하나의 값을 필터를 적용한 모든 원소에 더함

패딩 Padding

합성곱 연산 전에 입력 데이터 주변을 특정 값으로 채워줌

⇒ 주로 출력 크기를 조정할 목적으로 사용

> 합성곱 연산을 거칠 때마다 크기가 작아지면 어느 시점에선 출력 크기가 1 ⇒ 더이상 합성곱 연산 적용 불가
>
> ⇒ 막기 위해 패딩을 사용
>
> ⇒ 입력 데이터의 공간적 크기를 고정한 채 다음 계층에 전달 가능

스트라이드 Stride

보폭을 의미

필터를 적용하는 윈도우가 몇 칸씩 이동할건지 정함

출력 크기 계산

→ 이때 OH, OW는 원소의 개수를 의미하는것이므로 정수로 나눠떨어지는 값이어야함

> ⇒ 딥러닝 프레임워크 중에 값이 딱 나눠떨어지지 않을 때 처리해주는 경우도 있음

3차원 데이터의 합성곱 연산

2차원에서 길이 방향(=채널 방향)으로 특징 맵이 늘어남

→ 채널 쪽으로 특징 맵이 여러개라면 입력 데이터와 필터의 합성곱 연산을 채널마다 수행하고, 그 결과를 더해서 하나의 출력을 얻음

과정 예시

주의할 점 : 입력 데이터의 채널 수와 필터의 채널 수가 같아야함!

필터 자체의 크기는 원하는 값으로 설정 가능. 단, 모든 채널의 필터가 같은 크기여야 함

블록으로 생각하기

3차원 합성곱 연산에서 데이터, 필터를 직육면체 블록이라고 생각하자.

합성곱 연산의 출력으로 다수의 채널을 내보내려면?

필터(=가중치)를 여러개 사용하면 된다.

→ 필터 FN개를 사용하면, 출력 맵도 FN개가 생성됨

배치 처리

각 계층을 흐르는 데이터 차원을 하나 늘려, 4차원 데이터로 저장

(데이터 수, 채널 수, 높이, 너비)

⇒ 신경망에 4차원 데이터가 하나 흐를 때마다 데이터 N개에 대한 합성곱 연산이 이뤄짐

⇒ N회 분의 처리를 한 번에 수행한다

풀링 계층

세로, 가로 방향의 공간을 줄이는 연산

→ 이미지 분야에서는 주로 최대 풀링 사용

예) 2x2 최대 풀링 : 스트라이드 2로 2x2 영역에서 가장 큰 원소 하나를 꺼냄

→ 풀링의 윈도우 크기와 스트라이드 크기는 같은 값으로 설정하는 것이 일반적

특징

1. 학습해야 할 매개변수 없음
2. 채널 수 변하지 않음

   → 채널마다 독립적으로 계산하기 때문
3. 입력의 변화에 영향을 적게 받음 = 강건

   → 입력 데이터의 차이를 풀링이 흡수해 사라지게 함

im2col (image to column)

입력 데이터를 필터링(가중치 계산)하기 좋게 전개하는 함수

→ 이렇게 펼쳐서 계산한 후에 2차원 출력 데이터를 4차원으로 변형(reshape)

```
x1=np.random.rand(1,3,7,7)
col1=im2col(x1,5,5,stride=1,pad=0) #(데이터 수, 채널 수 , 높이, ㄴ
print(col1.shape) #(9,75)

x2=np.random.rand(10,3,7,7)
col2=im2col(x2,5,5,stride=1,pad=0) #데이터 10개
print(col2.shape) #(90,75)
```

→ 배치 크기에 따라 데이터 저장됨이 다르다.

합성곱 계층의 역전파에서는 im2col을 역으로 처리해야함!

→ col2im 함수 사용

⇒ col2im을 사용한다는 점을 제외하면 합성곱 계층의 역전파는 Affine 계층과 똑같음!

합성곱 계층 구현

```
class Convolution:
    def __inti__(self, w, b, stride=1, pad=0):
        self.w=w
        self.b=b
        self.stride=stride
        self.pad

    def forward(self, x):
        fn, c, fh, fw=self.w.shape
        n,c,h,w=x.shape
        out_h=int(1+(h+2*self.pad-fh)/self.stride)
        out_w=int(1+(w+2*self.pad-fw)/self.stride)

        col=im2col(x, fh, fw, self.stride, self.pad)
        col_w=self.w.reshape(fn, -1).T
        out=np.dot(col,col_w)+self.b

        out=out.reshpae(n, out_h, out_w, -1).transpose(0,3,
1,2)

        return out
```

풀링 계층 구현

```
class Pooling:
    def __init__(self, pool_h, pool_w, stride=1, pad=0):
        self.pool_h=pool_h
        self.pool_w=pool_w
        self.stride=stride
        self.pad=pad

    def forward(self, x):
        n, c, h, w=x.shape
        out_h=int(1+(h-self.pool_h)/self.stride)
        out_w=int(1+(w-self.pool_w)/self.stride)
```

```
        # 1. 전개
        col=im2col(x, self.pool_h, self.pool_w, self.strid
e, self.pad)
        col=col.reshape(-1, self.pool_h*self.pool_w)

        # 2. 최댓값
        out=np.max(col, axis=1)

        # 3. 성형
        out=out.reshape(n, out_h, out_w, c).transpose(0, 3,
1, 2)

        return out
```

1. 입력 데이터 전개
2. 행별 최댓값 구하기
3. 적절한 모양으로 성형

CNN 시각화하기

CNN이 보고 있는 것은 무엇일까?

합성곱 계층의 필터는 에지, 블롭 등 원시적 정보를 추출할 수 있음 → 이 정보가 뒷단 계층에 전달됨

층 깊이에 따른 추출 정보 변화

계층이 깊어질수록 추출되는 정보 (=강하게 반응하는 뉴런)은 더 **추상화** 된다.

→ 층이 깊어지면서 더 복잡하고 추상화된 정보가 추출됨

⇒ 층이 깊어지면서 뉴런이 반응하는 대상이 단순한 모양에서 '고급' 정보로 변화함

⇒ 사물의 '의미'를 이해하도록 변화함

대표적 CNN

LeNet

합성곱 계층과 풀링 계층을 반복하고 마지막으로 완전연결 계층을 거치면서 결과를 출력

LeNet vs 현재의 CNN

- LeNex는 시그모이드 함수, 현재는 ReLU 사용
- LeNex는 서브샘플링을 하여 중간데이터 크기를 줄이지만 현재는 최대 풀링이 주류

‘첫 CNN’

AlexNet (2012)

합성곱 계층과 풀링 계층을 거듭하며 마지막으로 완전연결 계층을 거쳐서 결과를 출력

- 활성화함수로 ReLU 사용
- LRN(Local Response Normalization)이라는 국소적 정규화하는 계층을 이용
- 드롭아웃 사용