



데이터문제해결및실습1

5주차-1

Chapter_10_다시살펴보는딥러닝주요개념

세종대학교

인공지능데이터사이언스학과

박동현 교수



★★★ 반드시 내 것으로 ★★★

#MUSTHAVE

탄탄한 기본기 + 전략적 사고로 문제해결 역량을 레벨업하자

머신러닝 · 딥러닝 문제해결 전략



- 본 강의는 골든래빗 출판사의 머신러닝/딥러닝 문제해결전략이 제공하는 강의 교안에 기반함.

Chapter

10

다시살펴보는 딥러닝주요개념



□ 학습 목표

□ 학습 순서

3부를 진행하는 데 필요한 주요 딥러닝 개념들을 요약·정리합니다. 궁금한 개념이 있다면 가볍게 살펴본 후 바로 다음 경진대회에 도전하기 바란다. 경진대회 문제를 풀다가 언뜻 떠오르지 않는 개념이 있을 때 참고한다.

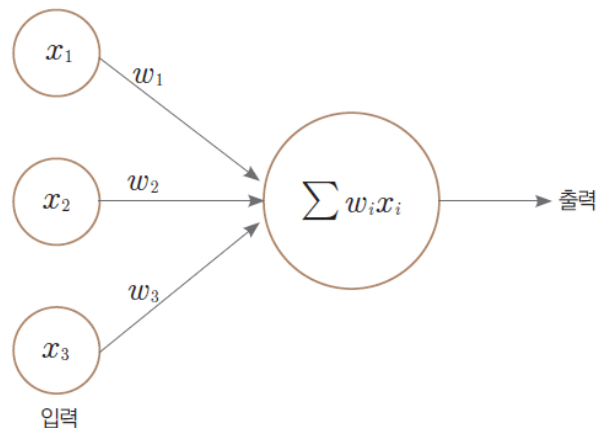
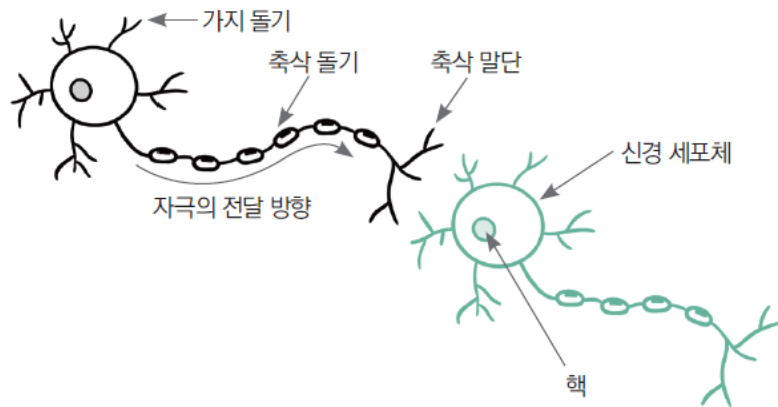


10.1 인공 신경망

인공 신경망(artificial neural network)은 인간의 생물학적 신경망과 유사한 구조를 갖는 인공 시스템이다. 인공 신경망은 뉴런 구조에서 영감을 얻어 고안한 것으로 딥러닝 구조의 핵심이다.

10.1.1 퍼셉트론

- 퍼셉트론(perceptron)은 뉴런의 원리를 본떠 만든 인공 구조다. 다음과 같이 뉴런과 유사한 방식으로 동작한다.



10.1 인공 신경망

10.1.1 퍼셉트론

- 퍼셉트론의 출력값을 구하는 절차는 다음과 같다.
 1. 입력값과 가중치를 곱한다.
 2. 곱한 값들의 총합을 구한다.
 3. 총합이 0을 넘으면 1, 넘지 않으면 0을 출력한다.
- 3번 단계에서 0 초과 여부에 따라 출력값을 결정하는 역할을 활성화 함수가 한다.
- **활성화 함수** : 입력값을 최종적으로 어떤 값으로 변환해 출력할지를 결정하는 함수
- 퍼셉트론 출력값을 수식으로 나타내면 다음과 같다.

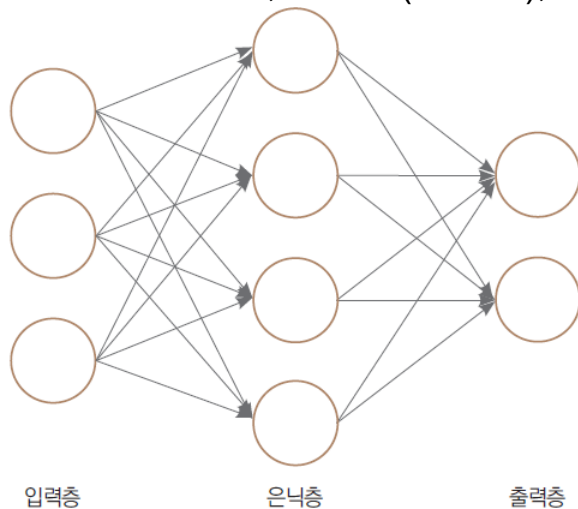
$$\text{출력값} = \begin{cases} 0, & w_1x_1 + w_2x_2 + w_3x_3 \leq 0 \\ 1, & w_1x_1 + w_2x_2 + w_3x_3 > 0 \end{cases}$$

- 입력값이 같더라도 가중치를 조절하면 출력값이 바뀐다.
- **훈련(training) 혹은 학습**: 원하는 출력값을 내보내도록 가중치를 조정해가는 작업
- 단순 퍼셉트론은 선형 분류 문제밖에 풀지 못한다는 한계가 있다.
- 비선형 분류 문제를 풀기 위해 다층 퍼셉트론(multi-layer perceptron)을 만들어 해결한다.

10.1 인공 신경망

10.1.2 신경망

- 신경망은 입력층, 은닉층(중간층), 출력층으로 구성된다.



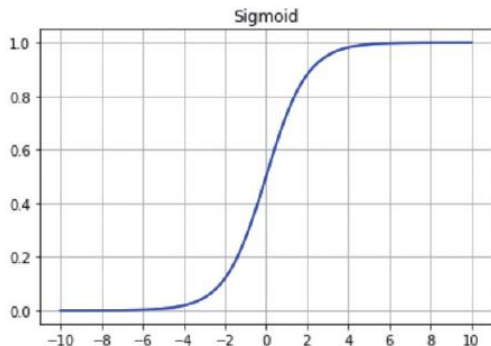
10.1 인공 신경망

10.1.3 활성화 함수

- **활성화 함수** : 입력값을 최종적으로 어떤 값으로 변환해 출력할지를 결정하는 함수. 입력값과 가중치를 곱한 값들은 활성화 함수를 거쳐 출력값이 된다.

시그모이드 함수(sigmoid function)

: S자 곡선을 그리는 수학 함수이다. 그래프로 그리면 다음과 같다.

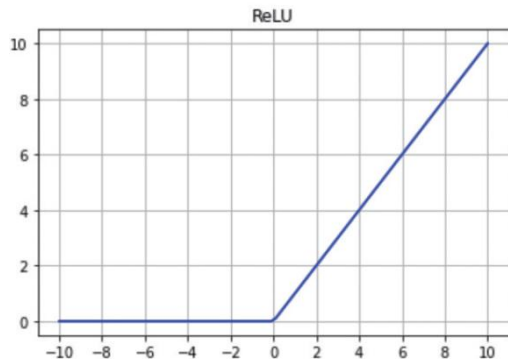


10.1 인공 신경망

10.1.3 활성화 함수

ReLU 함수(rectified linear unit function)

: 입력값이 0보다 크면 입력값을 그대로 출력하고, 입력값이 0 이하이면 0을 출력한다. 활성화 함수로 시그모이드를 사용할 때보다 대체로 성능이 좋아서 ReLU 함수가 더 많이 활용된다.

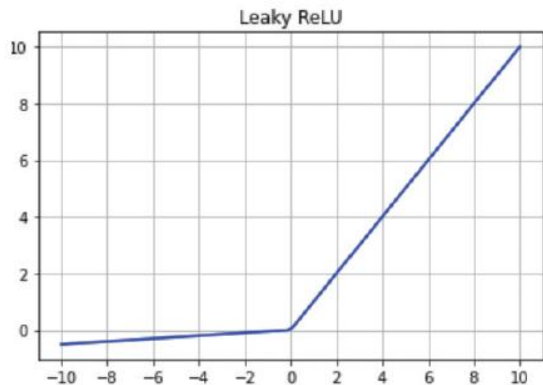


10.1 인공 신경망

10.1.3 활성화 함수

Leaky ReLU 함수

: ReLU를 약간 변형한 함수로, 역시 자주 쓰인다. ReLU와 달리, 입력값이 0 이하일 때 약간의 음숫값을 살려둔다.

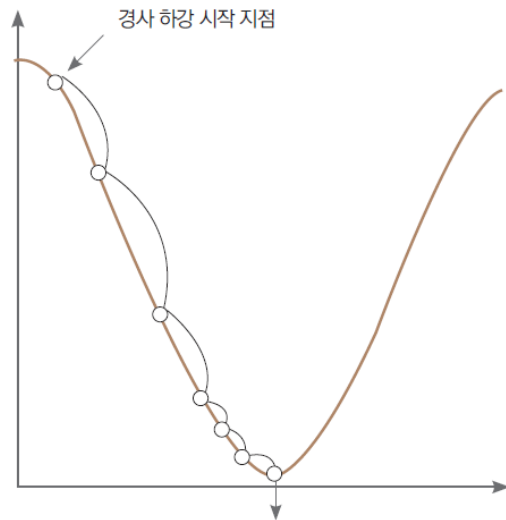


$$\begin{cases} x & x > 0 \\ 0.01x & x \leq 0 \end{cases}$$

10.1 인공 신경망

10.1.4 경사 하강법

- 손실 함수(loss function) : 모델 성능이 얼마나 나쁜지를 측정하는 함수. 손실 함수의 값이 작을수록 좋은 모델이다.
- 대표적인 손실 함수로 **평균 제곱 오차(mean squared error)**와 **교차 엔트로피(cross entropy)**가 있다.
- 경사하강법(**gradient descent**)의 일반적인 절차
 - 현재 위치에서 기울기(경사)를 구한다.
 - 기울기 아래 방향으로 일정 거리를 이동한다.
 - 손실 함수가 최소가 될 때까지, 즉 현재 위치의 기울기가 0이 될 때까지 1~2단계를 반복한다.



10.1 인공 신경망

10.1.4 경사 하강법

- 학습률(learning rate) : 기울기 방향으로 얼마만큼 이동할지 결정하는 값

$$W = W - \eta \frac{\partial L}{\partial W}$$

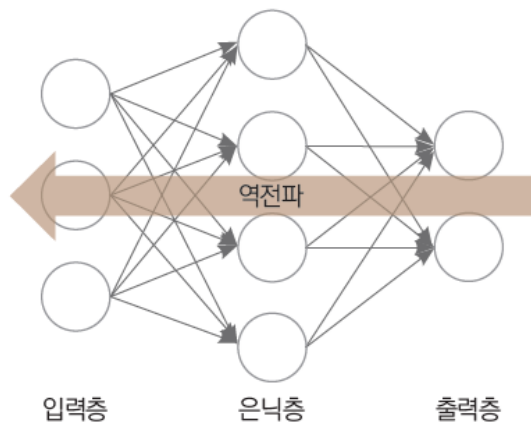
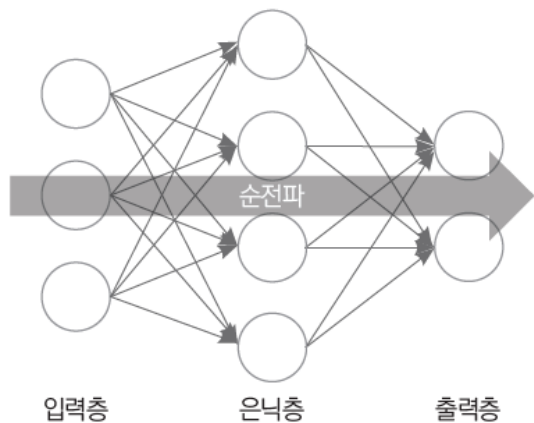
손실 함수 미분값(기울기) |
|
갱신할 가중치 | 학습률

- 확률적 경사 하강법(stochastic gradient descent, SGD) : 전체 학습 데이터에서 개별 데이터를 무작위로 뽑아 경사 하강법을 수행하는 알고리즘
- 미니배치 경사 하강법(mini batch gradient descent) : 데이터를 하나씩 훈련하기보다는 여러 묶음으로 묶어 처리하는 경사 하강법

10.1 인공 신경망

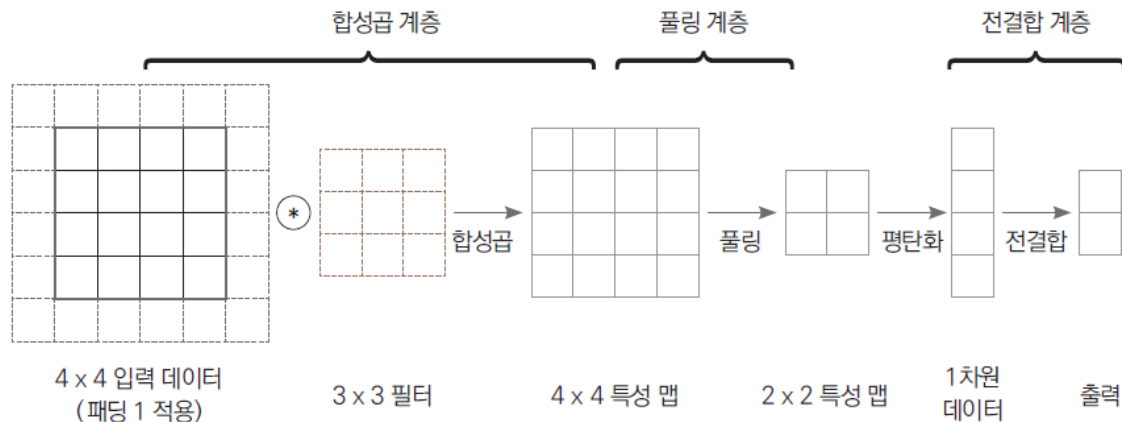
10.1.5 순전파와 역전파

- 순전파(forward propagation) : 신경망에서 입력값이 입력층과 은닉층을 거쳐 출력층에 도달하기까지의 계산 과정
- 역전파(back propagation) : 순전파의 반대 개념



10.2 합성곱 신경망(CNN)

합성곱 신경망(convolutional neural network, CNN)은 컴퓨터 비전 분야에서 주로 쓰이는 신경망이다. 합성곱 신경망은 다음과 같이 여러 구조가 모여 구성된다.



10.2 합성곱 신경망(CNN)

10.2.1 합성곱 계층

- 합성곱 계층(convolutional layer) : 합성곱으로 이루어진 신경망 계층
- 합성곱(convolution) : 2차원 데이터의 일정 영역 내 값들을 하나의 값으로 압축한 연산
- 필터(filter) : 입력 데이터에서 특정한 특성을 필터링하는 역할
- 크기가 4 x 4인 데이터와 3 x 3인 필터를 활용해 합성곱 연산하는 예

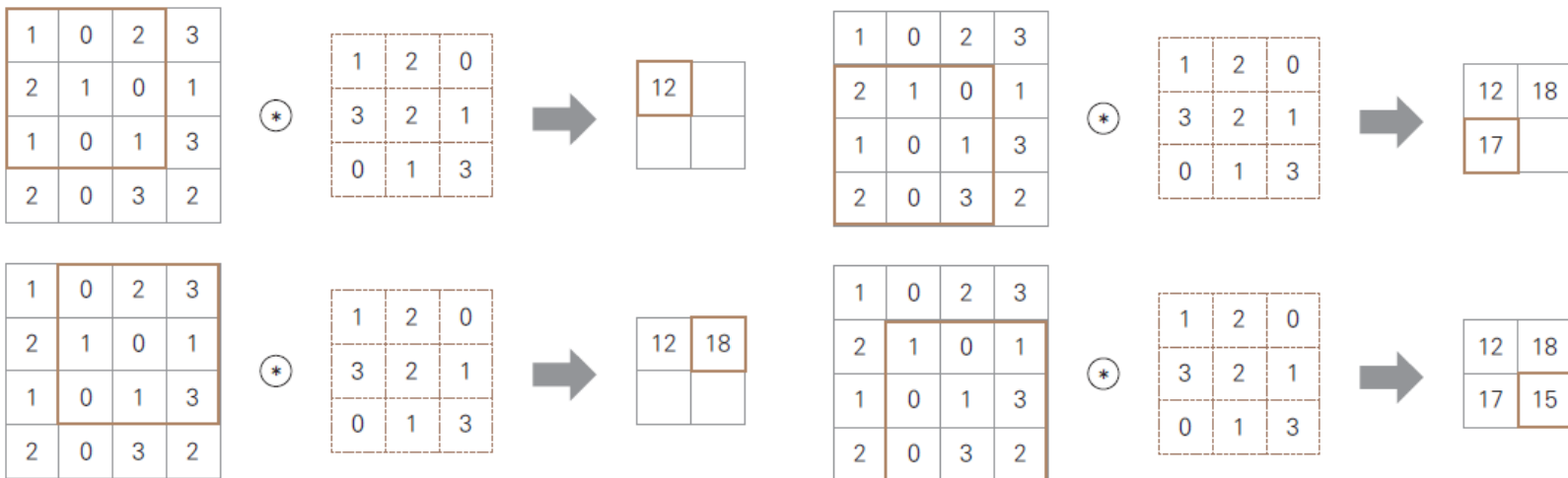


- 특성 맵 혹은 피쳐 맵(feature map) : 합성곱 연산으로 얻은 결과

10.2 합성곱 신경망(CNN)

10.2.1 합성곱 계층

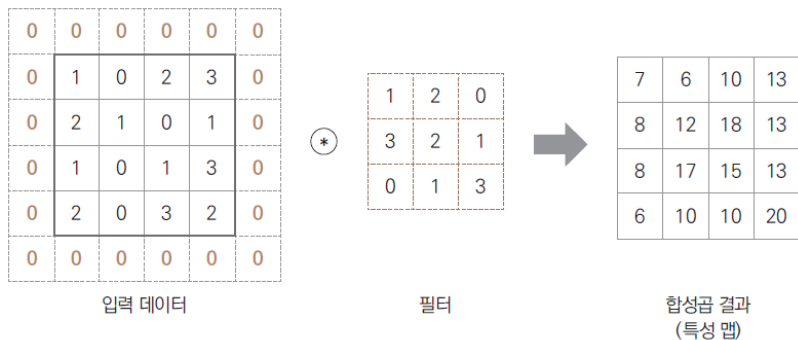
- 합성곱 연산을 수행하는 절차



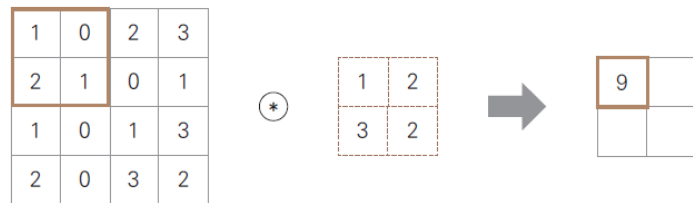
10.2 합성곱 신경망(CNN)

10.2.2 패딩과 스트라이드

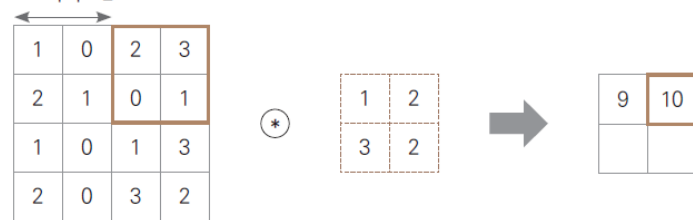
- **패딩(padding)**: 입력 데이터 주변을 특정 값으로 채우는 것. 보통 입력 데이터 주변을 0으로 채운다.
- **스트라이드(stride)**: 합성곱 연산을 수행할 때, 필터가 한 번에 이동하는 간격



▼ 스트라이드가 2일 때의 합성곱 연산



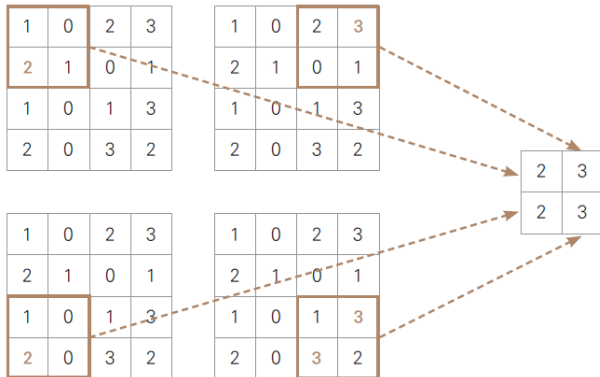
스트라이드 2



10.2 합성곱 신경망(CNN)

10.2.3 풀링

- 풀링(pooling): 특성 맵 크기를 줄여 이미지의 요약 정보를 추출하는 기능
- 위치 불변성(location invariance): 풀링이 특정 영역의 요약 정보(대푯값)를 가져오므로 위치가 변해도 같은 물체로 판단하는 것
- 최대 풀링(max pooling): 풀링 영역에서 가장 큰 값을 취하는 방법
- 평균 풀링(average pooling): 풀링 영역의 평균을 구하는 방법
- 풀링 크기를 2x2로 설정하여 최대 풀링을 하는 예시



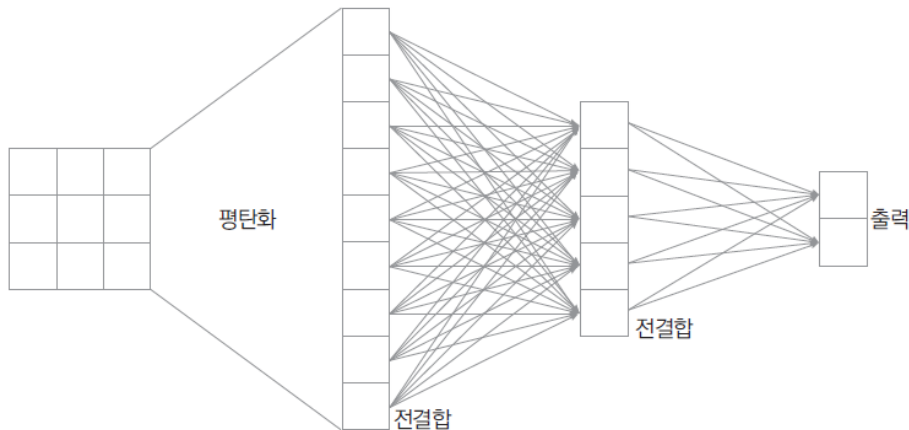
$$N_{out} = \left\lfloor \frac{N_{in} + 2P - K}{S} \right\rfloor + 1$$

$$N_{out} = \left\lfloor \frac{N_{in} + 2 \times 0 - K}{K} \right\rfloor + 1 = \left\lfloor \frac{N_{in}}{K} \right\rfloor$$

10.2 합성곱 신경망(CNN)

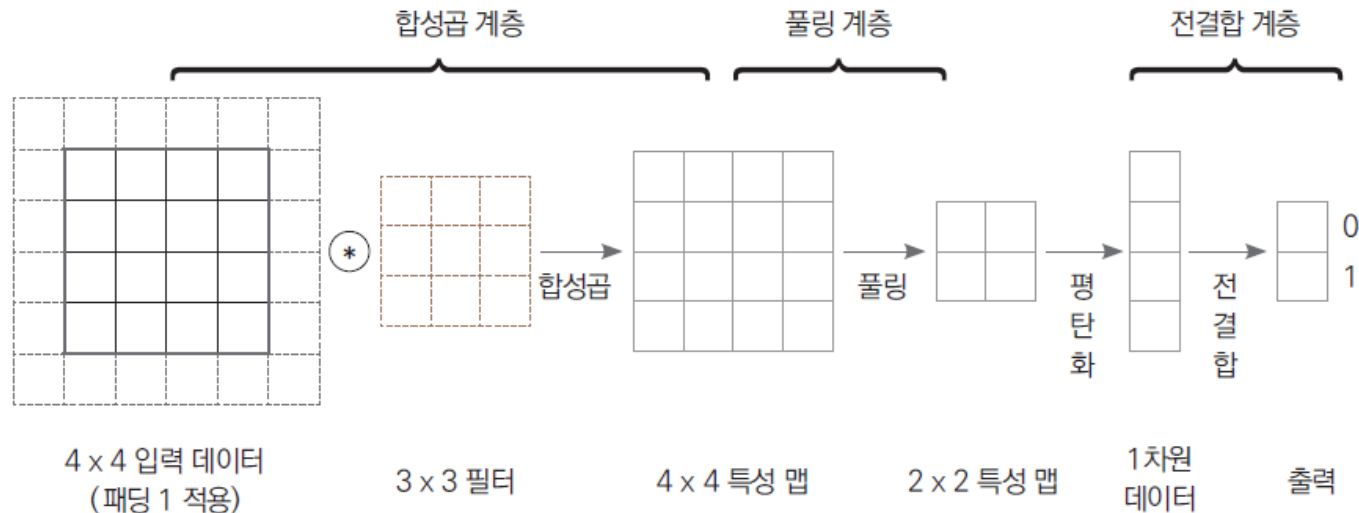
10.2.4 전결합

- **전결합**(fully-connected) : 이전 계층의 모든 노드 각각이 다음 계층의 노드 전부와 연결된 결합
- **전결합 계층**(fully-connected layer) 혹은 **밀집 계층**(dense layer) : 전결합으로 구성된 계층
- **평탄화** : 다차원 데이터를 1차원 데이터로 바꾸는 작업



10.2 합성곱 신경망(CNN)

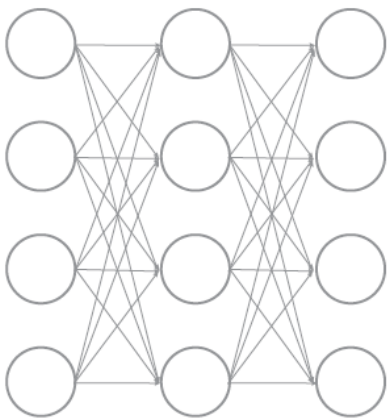
10.2.5 합성곱 신경망 전체 구조



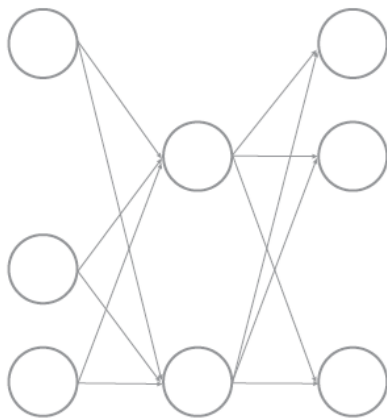
10.3 성능 향상을 위한 딥러닝 알고리즘

10.3.1 드롭아웃

- **드롭아웃(dropout)** : 과대적합을 방지하기 위하여 신경망 훈련 과정에서 무작위로 일부 뉴런을 제외하는 기법



기본 신경망 구조



드롭아웃을 적용한 신경망 구조

10.3 성능 향상을 위한 딥러닝 알고리즘

10.3.2 배치 정규화

- **배치 정규화(batch normalization)**: 과대적합 방지와 훈련 속도 향상을 위한 기법으로, 내부 공변량 변화 현상을 해결한다. 배치 정규화는 이름에서 알 수 있듯이 ‘배치’ 단위 ‘정규화’한다. 이때 배치는 미니배치이며, 정규화는 데이터가 정규분포(평균 0, 분산 1)를 따르도록 분포를 변환하는 작업이다. 추가로 이렇게 정리한 데이터를 확대/축소하고 이동 변환까지 수행한다.
 1. 입력 데이터 미니 배치를 평균이 0, 분산이 1이 되게 정규화한다.
 2. 정규화한 데이터의 스케일을 조정하고 이동시킨다.

1. 정규화

미니배치를 평균이 0, 분산이 1이 되게 정규화해야 한다.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} x_i$$

$$\sigma_B^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} (x_i - \mu_B)^2$$

2. 스케일 조정 및 이동

정규화한 데이터의 스케일을 조정하고 이동시켜야 한다.

$$z_i = \gamma \hat{x}_i + \beta$$

10.3 성능 향상을 위한 딥러닝 알고리즘

10.3.3 옵티마이저

- 옵티마이저(optimizer) : 신경망의 최적 가중치를 찾아주는 알고리즘

모멘텀(Momentum)

물리학의 관성 개념을 추가한 옵티마이저

Adagrad

최적 파라미터에 도달할수록 학습률을 낮추도록 한 옵티마이저

RMSProp

Adagrad의 단점을 보완한 방법. Adagrad는 훈련 시작 단계부터 기울기를 누적해 학습률을 낮추지만, RMSProp은 최근 기울기만 고려해 학습률을 낮춘다. 훈련을 오래 지속해도 학습률이 0에 수렴하지 않는다.

Adam

모멘텀과 RMSProp의 장점을 결합한 방법. 딥러닝 모델을 설계할 때 가장 많이 사용하는 옵티마이저다. RMSProp처럼 적응적 학습률을 적용한다.

10.3 성능 향상을 위한 딥러닝 알고리즘

10.3.4 전이 학습

- **전이학습**(transfer learning) : 한 영역에서 사전 훈련된 모델(pretrained model)에 약간의 추가 학습을 더해 유사한 다른 영역에서도 활용하는 기법

▼ 전이 학습 시 성능 비교(예시)

