6장 학습관련기술들

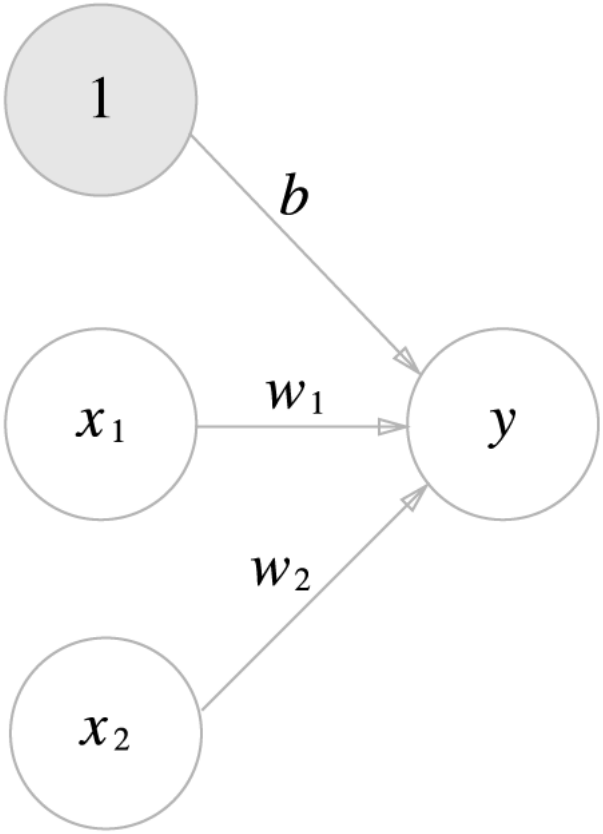
Deep learning from scratch
SLiPP study

2017년 5월 10일

지금까지

- 퍼셉트론perceptron
- 신경망neural network
- 수학적 신경망 학습numerical gradient
- 알고리듬적 신경망 학습error back propagation

단층 퍼셉트론

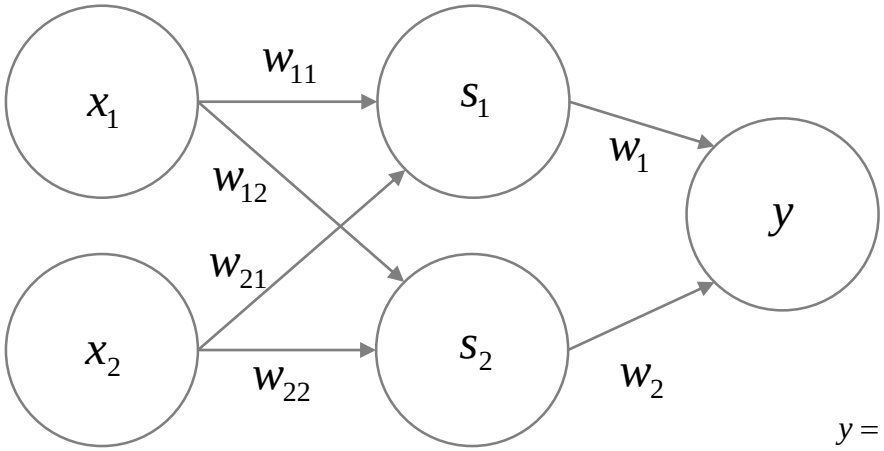


$$y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

$$y = Wx + b = [w_1 \quad w_2 \quad b] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

분류문제를 행렬연산으로 해결가능(가중치/편향값을 알고 있다면)

다층 퍼셉트론



$$y = \begin{cases} 0 & w_1(w_{11}x_1 + w_{21}x_2) + w_2(w_{12}x_1 + w_{22}x_2) \le 0 \\ 1 & w_1(w_{11}x_1 + w_{21}x_2) + w_2(w_{12}x_1 + w_{22}x_2) \le 0 \end{cases}$$

$$y = [w_1 \quad w_2]\begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, S = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow y = [w_1 \quad w_2]\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

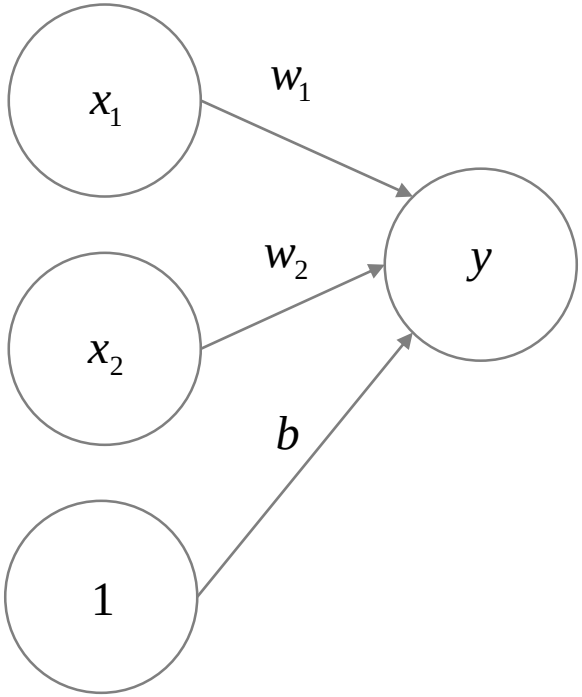
$$y = W_{s \rightarrow y} W_{x \rightarrow s} x$$

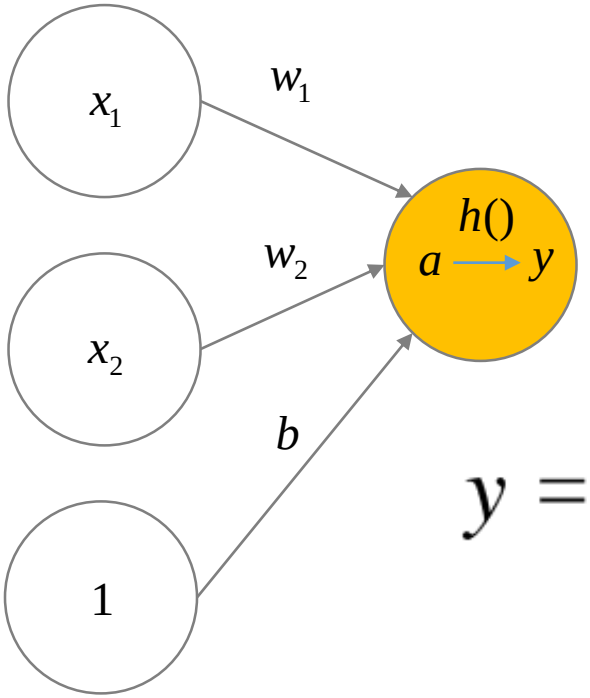
층(layer)간 연산은 단순한 행렬연산(y = Wx + b)로 계산 가능

복잡한 분류 문제도 행렬연산으로 가능(가중치/편향값을 알고 있다면)

신경망(neural network)

다층 퍼셉트론 & (활성화함수 or 소프트맥스)





$$y = h(a)$$

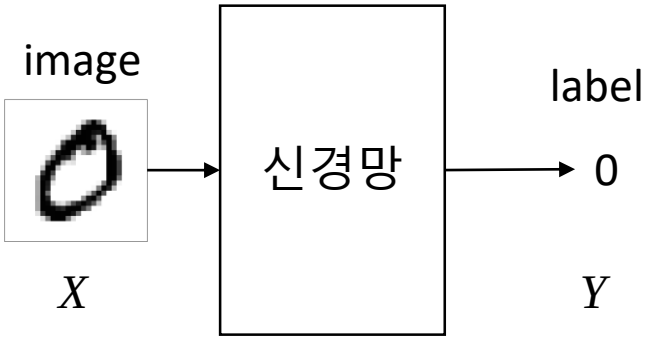
$$y = \begin{cases} 0 & b + w_1 x_1 + w_2 x_2 \le 0 \\ 1 & b + w_1 x_1 + w_2 x_2 > 0 \end{cases}$$

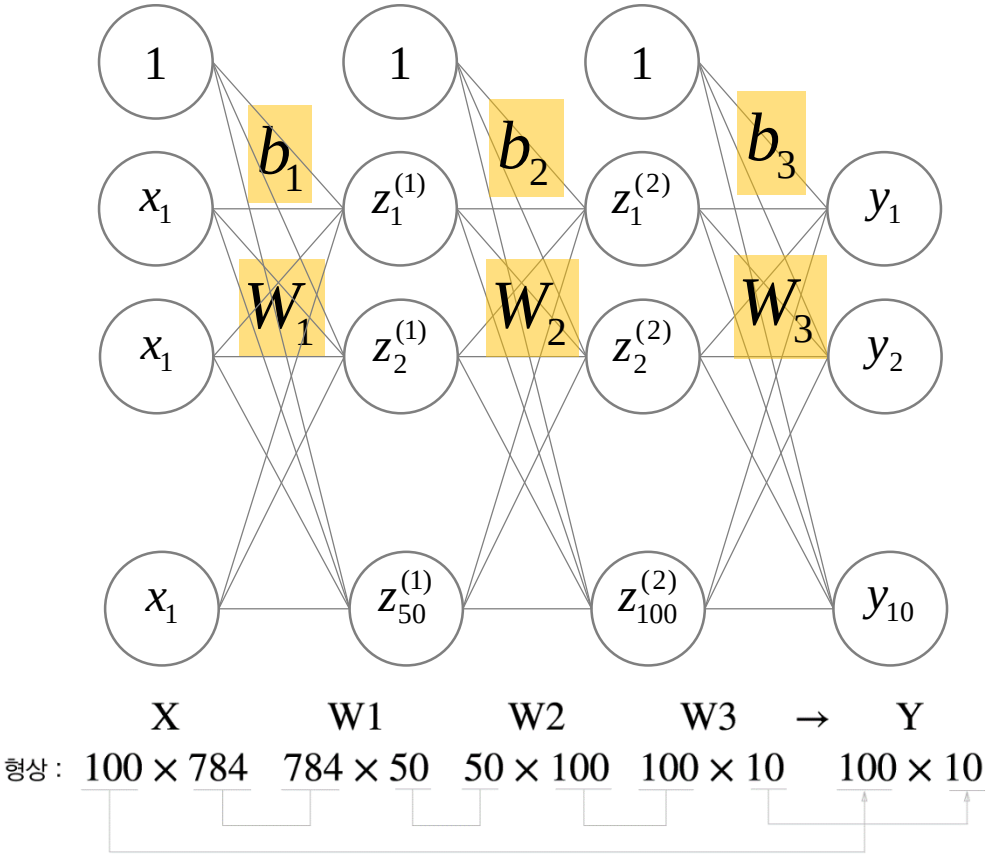
$$y = \begin{cases} 0 & h(b + w_1 x_1 + w_2 x_2) \le 0 \\ 1 & h(b + w_1 x_1 + w_2 x_2) > 0 \end{cases}$$

One layer!

Sigmoid / ReLU 모두 연속함수(미분가능)

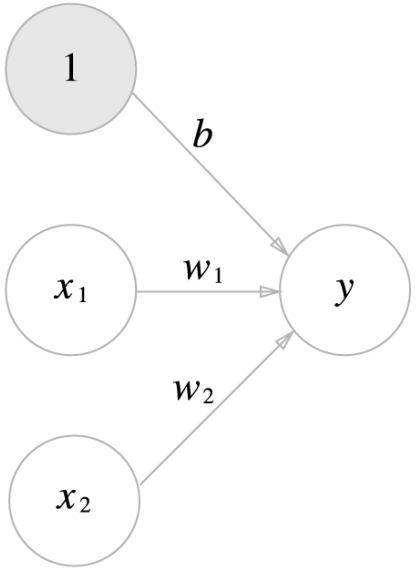
신경망(neural network) MNIST db case (숫자 이미지 입력 > 숫자 분류)



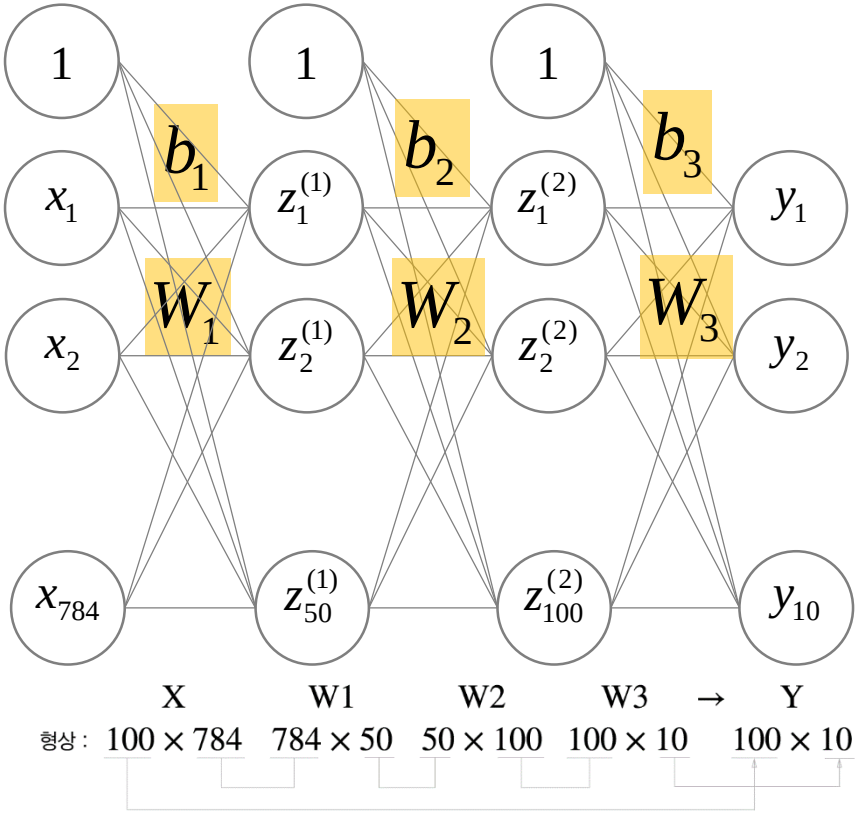


입력과 출력에 따라 신경망을 설계하면 분류기가 된다.
(여전히, 가중치(W)/편향치(b)/적합한 레이어/노드수를 안다면)

데이터 기반 학습



versus



3개

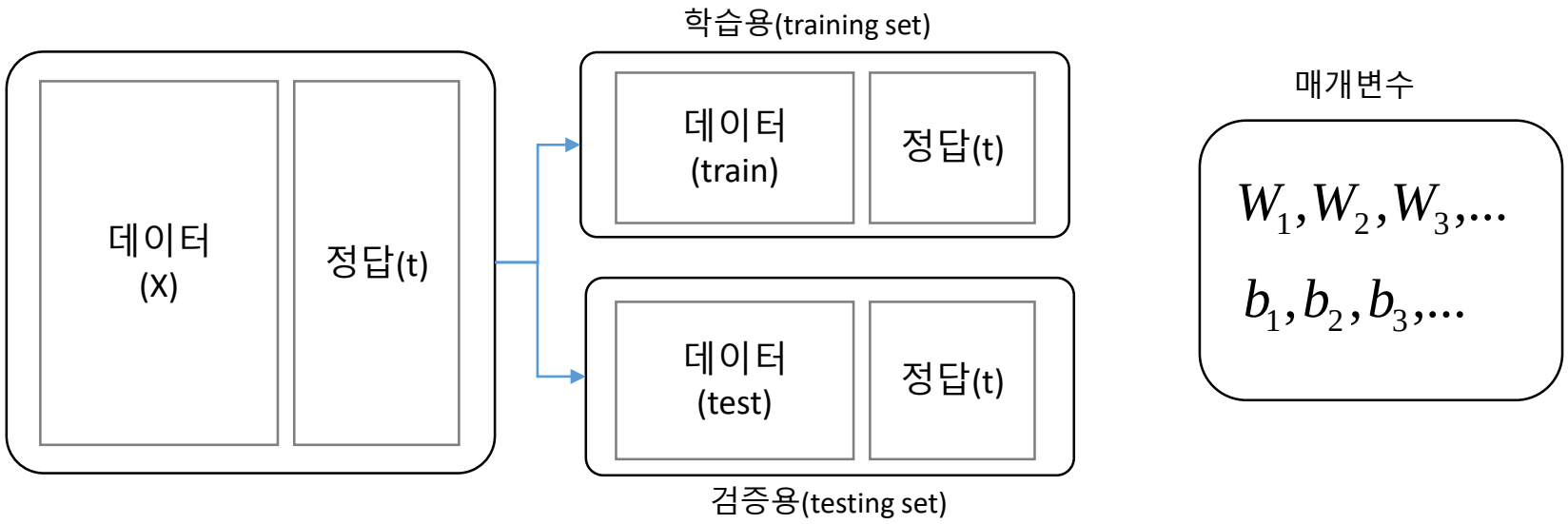
45,360개

- 데이터 기반으로 매개변수(W,b등)을 알아내는 작업 = 학습
- 기본 아이디어 = 프로그램으로 매개변수들을 움직여 최선의 파라미터 값 정하기

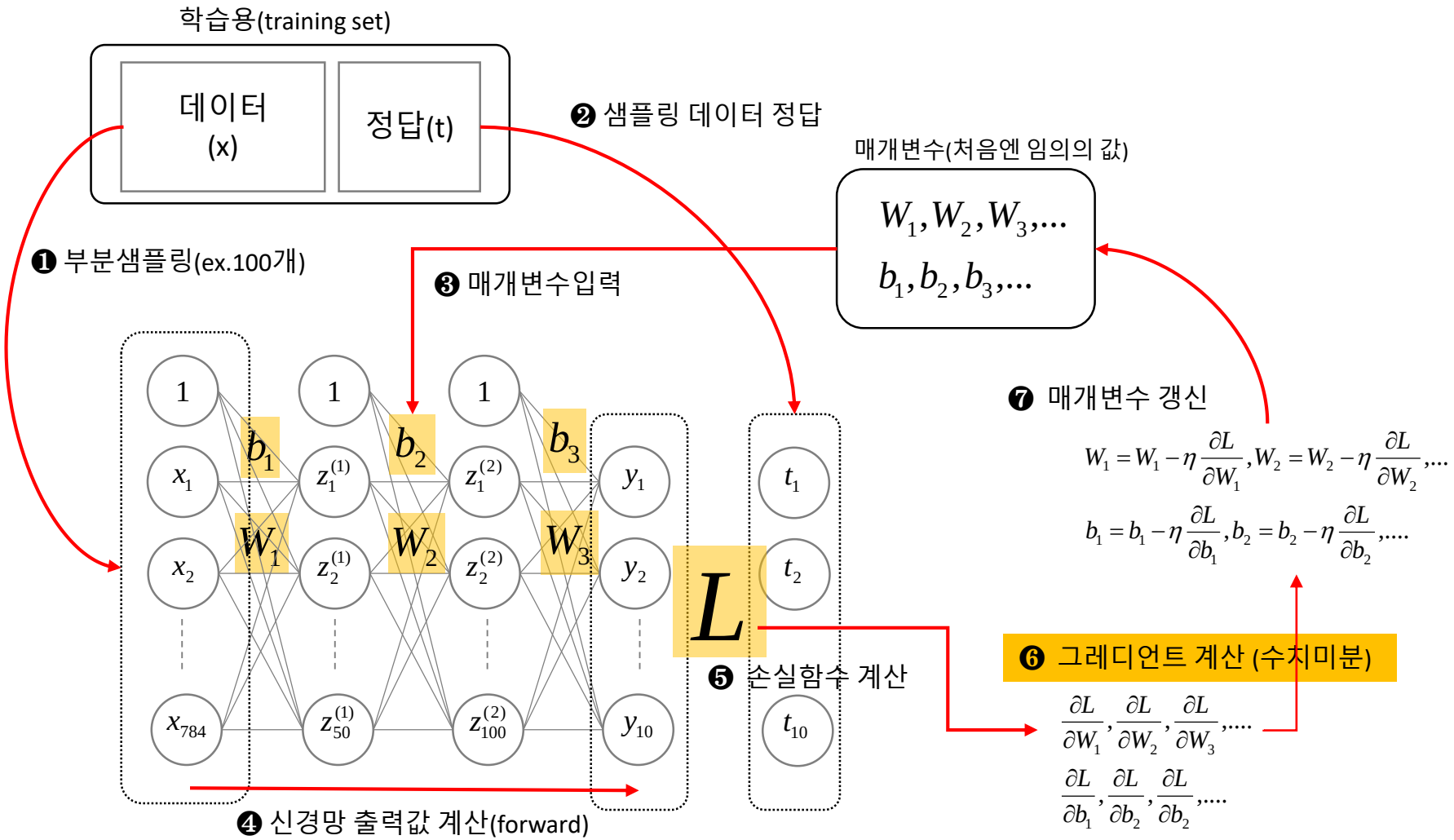
데이터 기반 학습

- 최적화 목표 = 손실함수를 정의하고 최소가 되는 방향
- 매개변수를 바꾸는 방향 = 손실함수의 매개변수 미분 방향

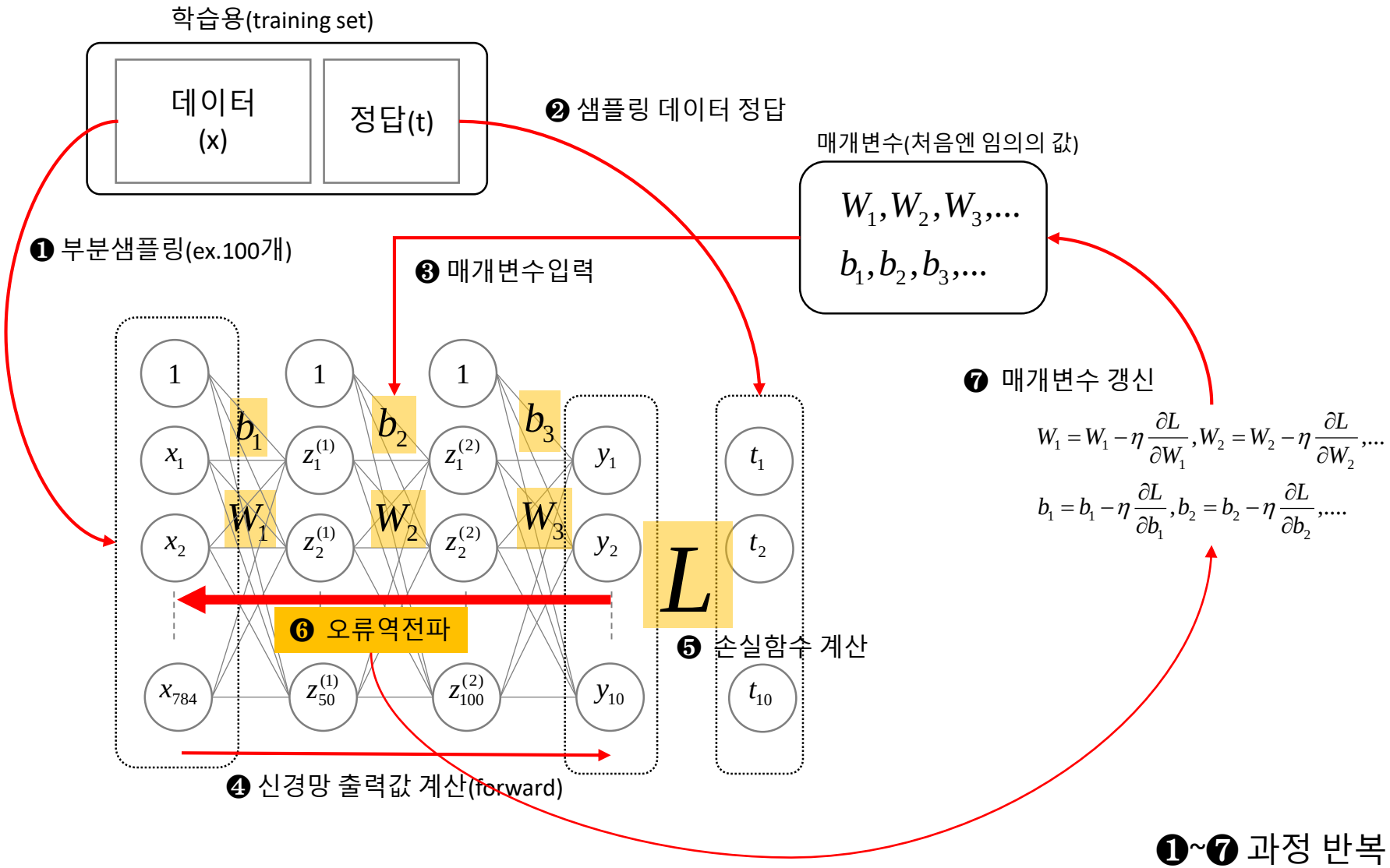
데이터 기반 학습 알고리듬



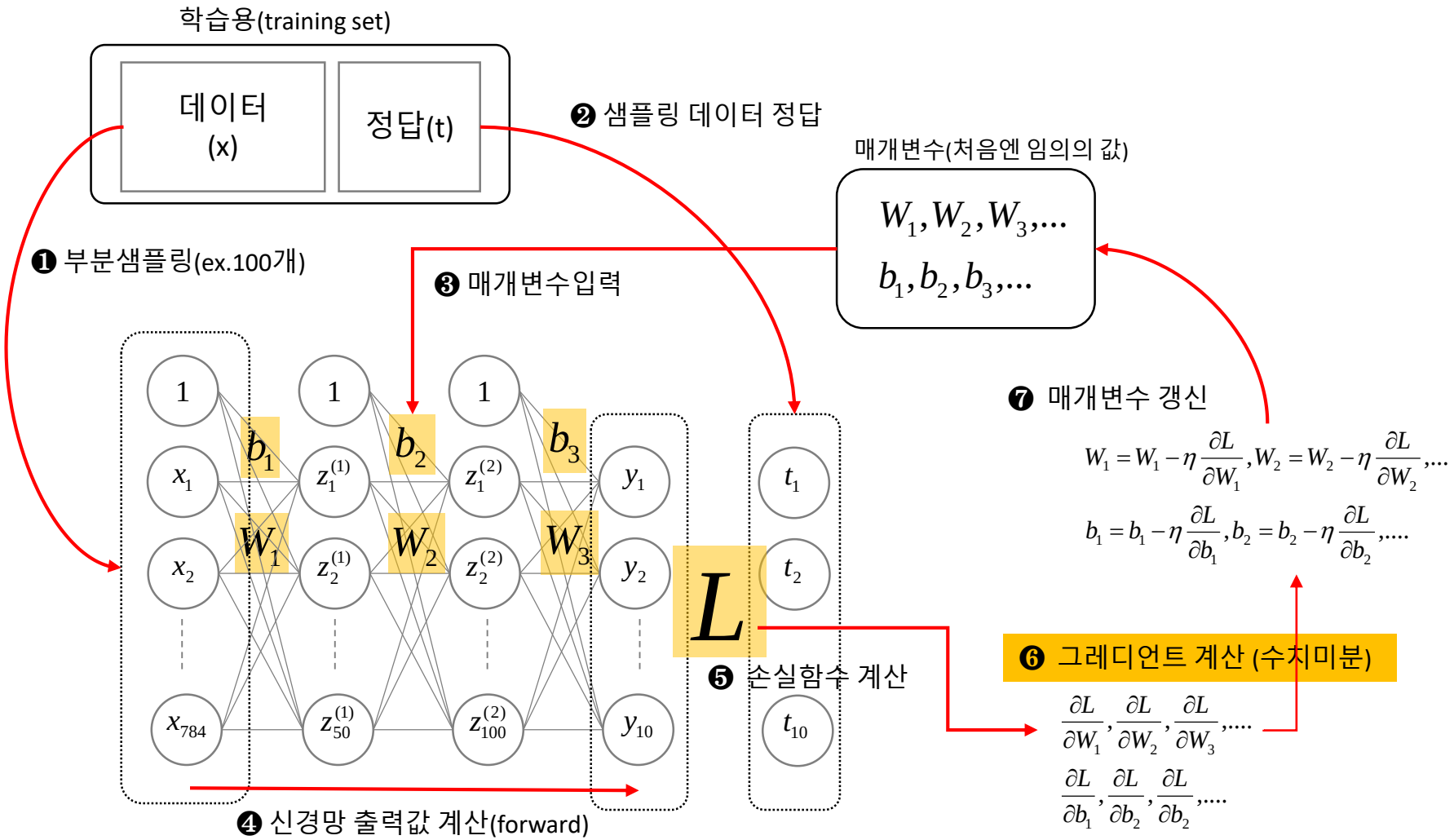
데이터 기반 학습 알고리듬 + 수치미분



데이터 기반 학습 알고리듬 + 오류역전파

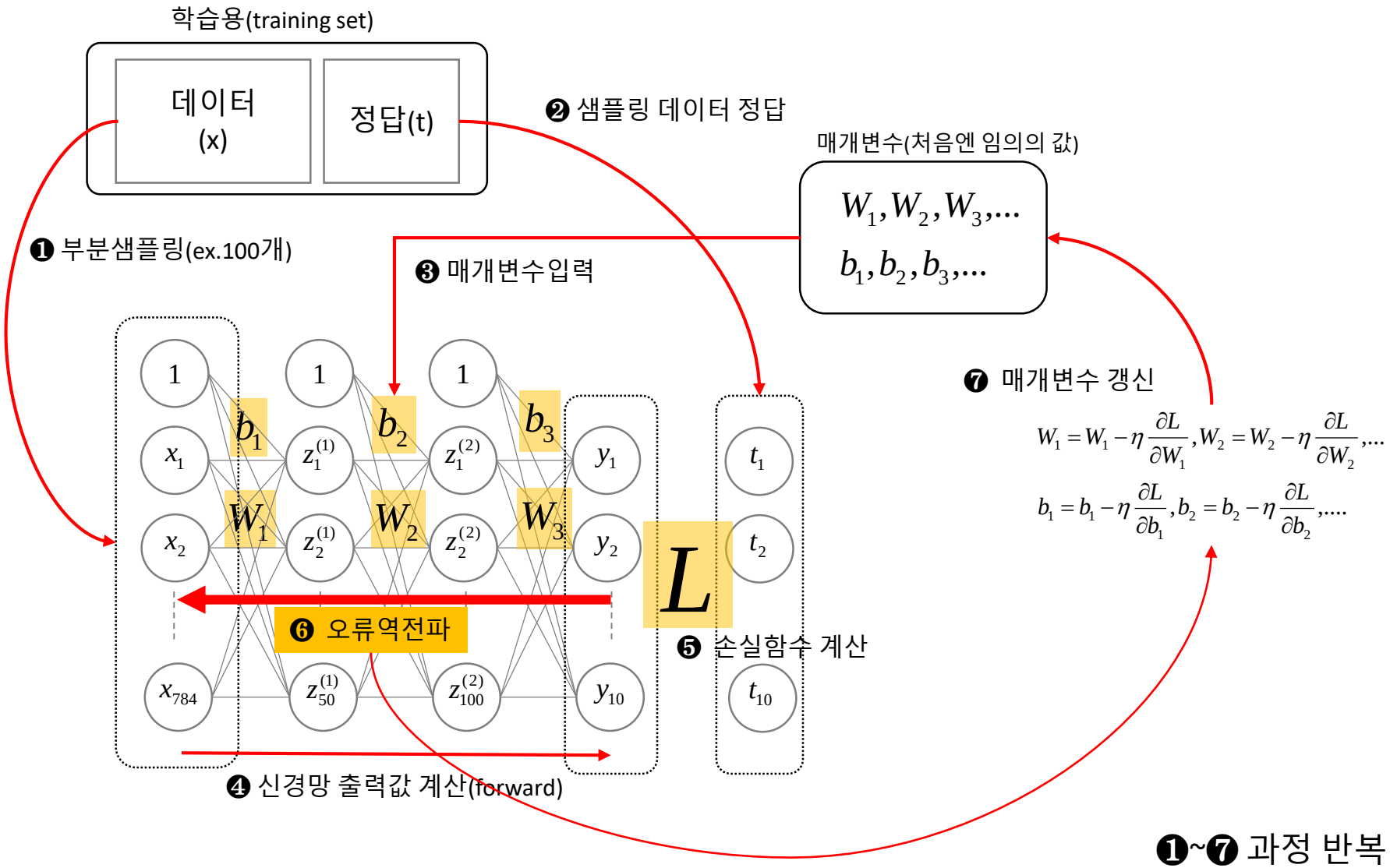


데이터 기반 학습 알고리듬



❶~❼ 과정 반복

데이터 기반 학습 알고리듬



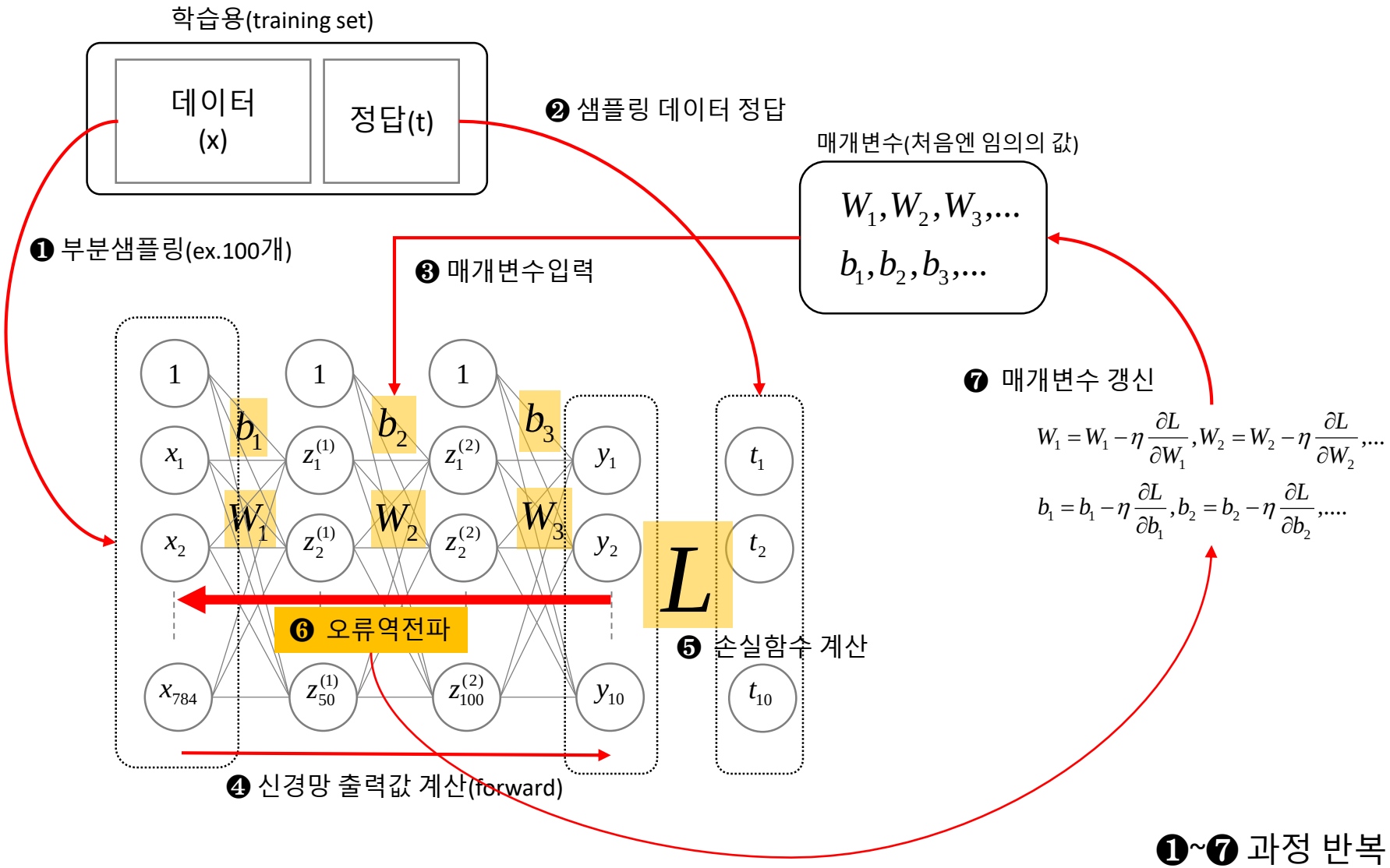
배울 것(6장)

- 매개변수 최적화 방법(optimization)
- 가중치 매개변수 초기화(initialization)
- 하이퍼 파라미터 설정 방법
- 정규화 방법(오버피팅 대응책)

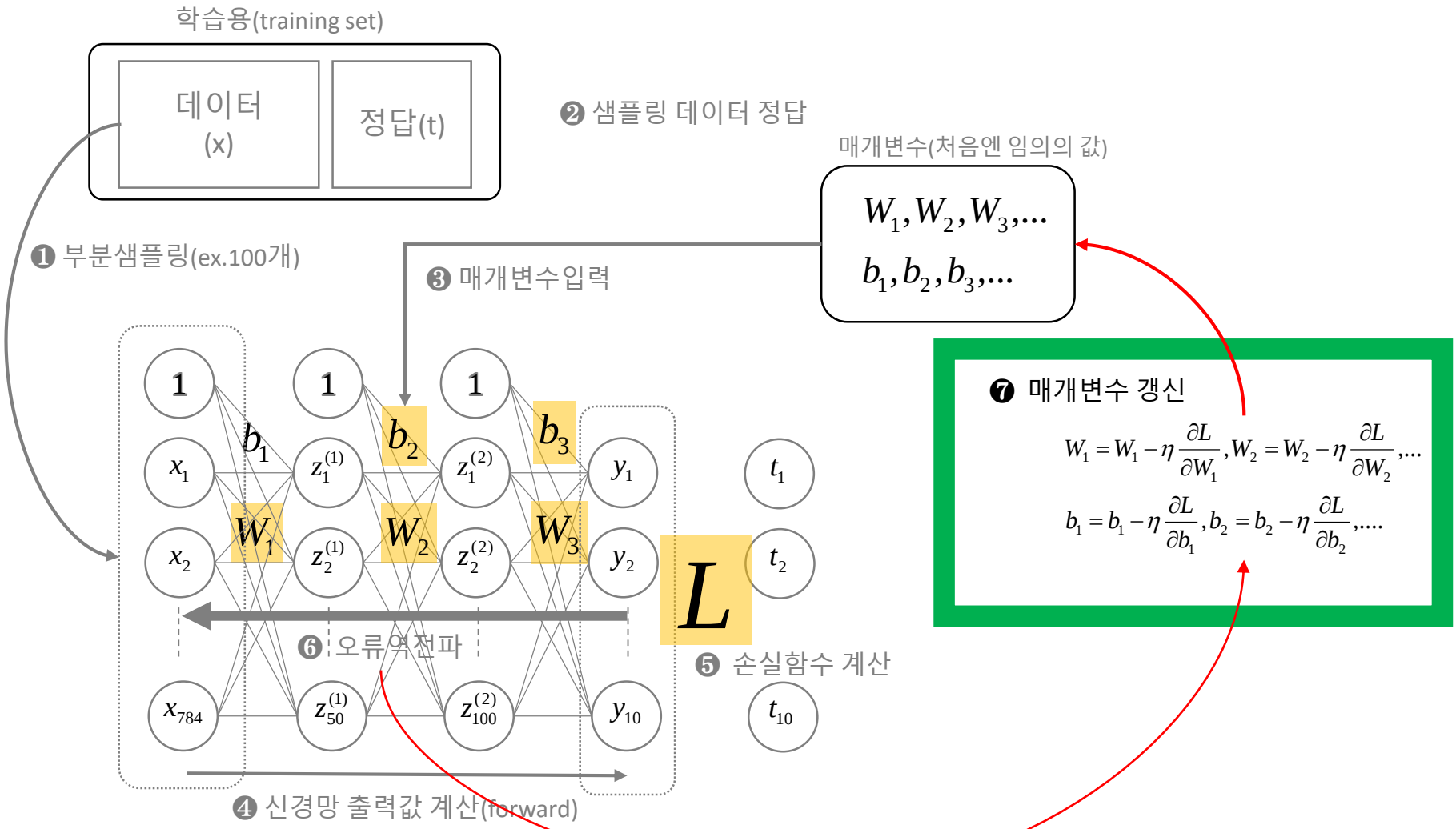
매개변수 최적화 방법

Optimizer 선택

데이터 기반 학습 알고리듬



매개변수 최적화


학습용(training set)
데이터 (x)
정답(t)
❷ 샘플링 데이터 정답
매개변수(처음엔 임의의 값)
$W_1, W_2, W_3, \ldots$
$b_1, b_2, b_3, \ldots$
❶ 부분샘플링(ex.100개)
❸ 매개변수입력
❼ 매개변수 갱신
$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1}, W_2 = W_2 - \eta \frac{\partial L}{\partial W_2}, \ldots$
$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}, b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}, \ldots$
❻ 오류역전파
❺ 손실함수 계산
❹ 신경망 출력값 계산(forward)
❶~❼ 과정 반복

매개변수 최적화

확률적 경사 하강법(Stochastic Gradient Descent, SGD)

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

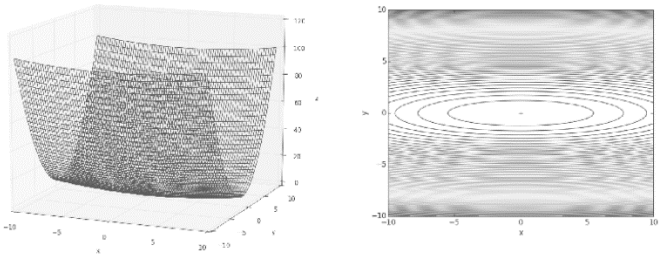
매개변수 최적화

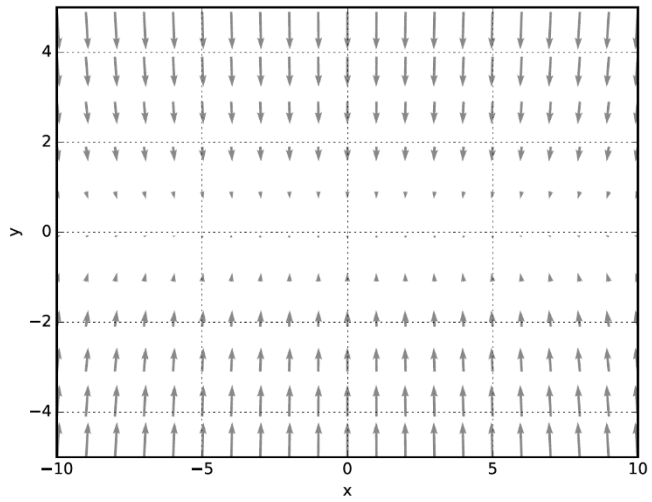
SGD의 문제점

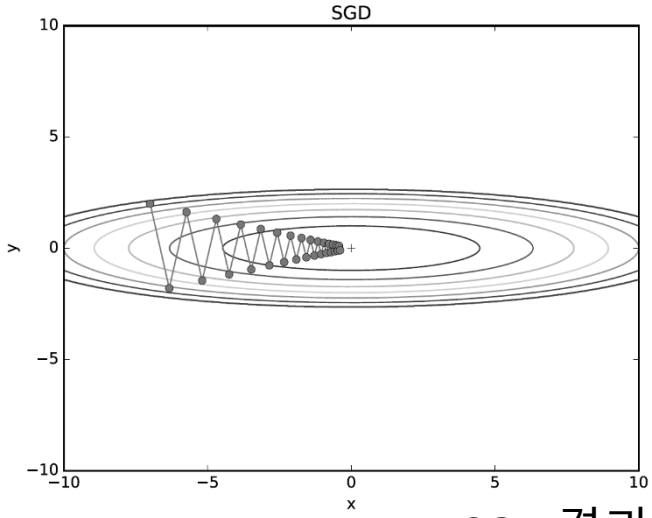
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

- 기울기들 간 차이가 크면 문제!
- 동일한 learning rate을 써서 생기는 문제!

$$f(x,y) = \frac{1}{20}x^2 + y^2$$







SGD 결과

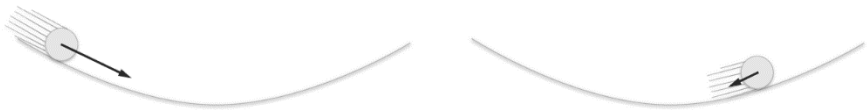
매개변수 최적화

모멘텀(momentum)

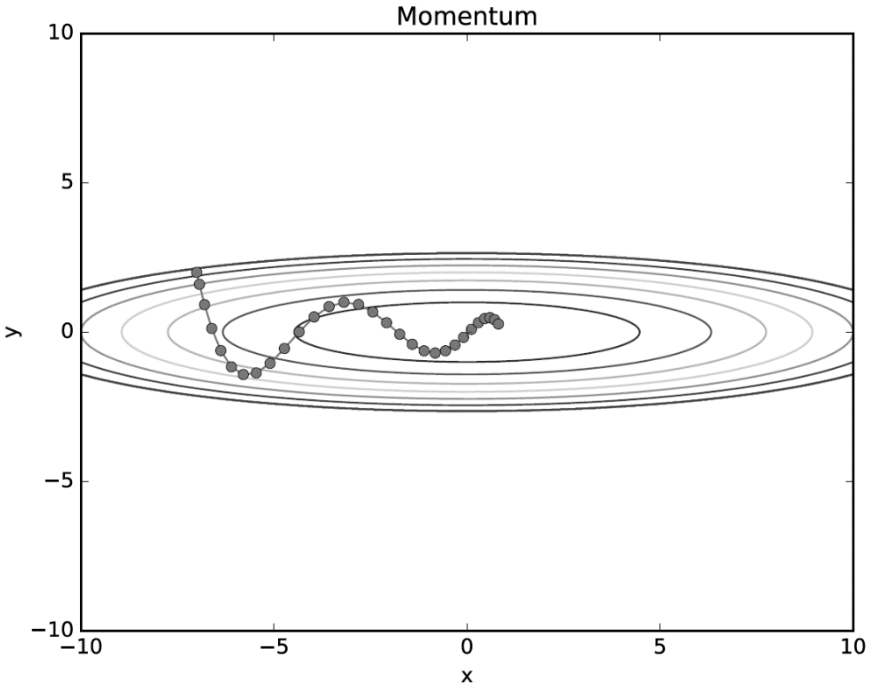
$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

- 속도(v)를 계산해 가중치에 추가 (가속도가 붙는 개념)



아이디어: 물리적 속성을 부여해보자



x 방향 기울기는 매우 작지만 가속도가 붙어 더 빨리 다가감

매개변수 최적화

AdaGrad

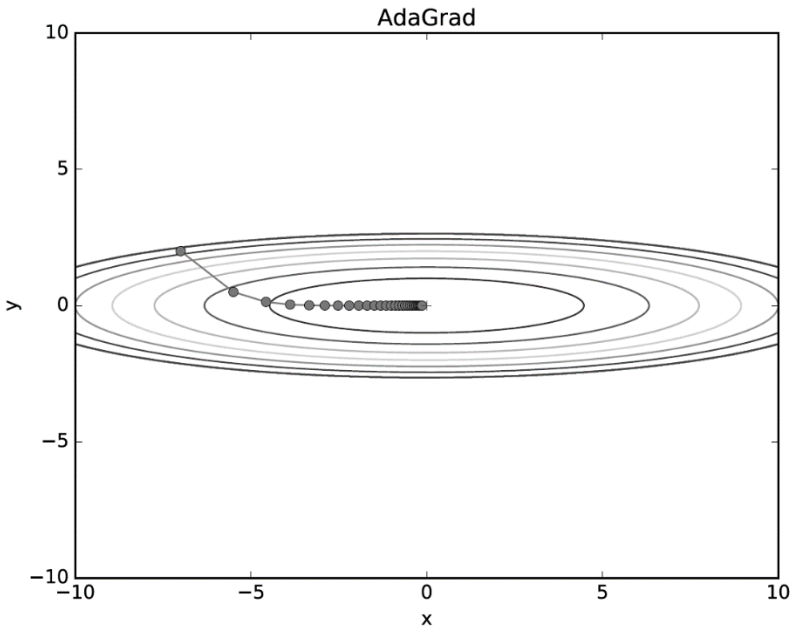
아이디어: 학습률 변수를 상황에 맞게 바꾸어 보자 (learning rate decay)

처음에는 크게 변화시켜 학습하고 점점 작게 변화시켜보자

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

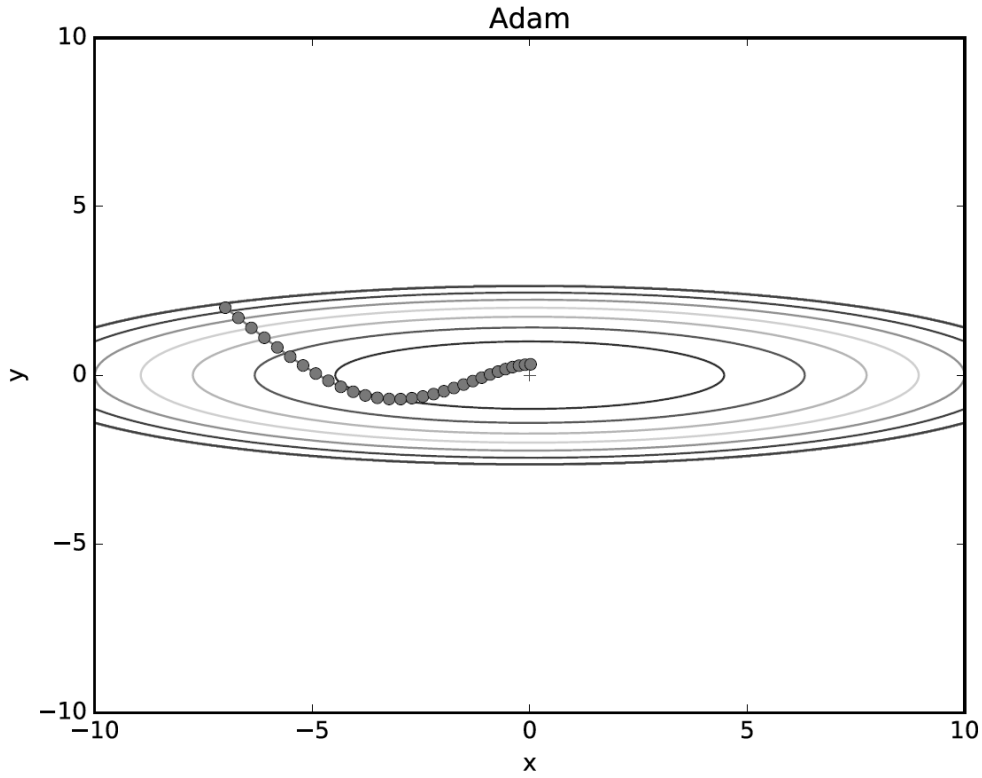
매개변수마다 h값에 의해
다른 학습률 변수가 지정된다.



매개변수 최적화

Adam

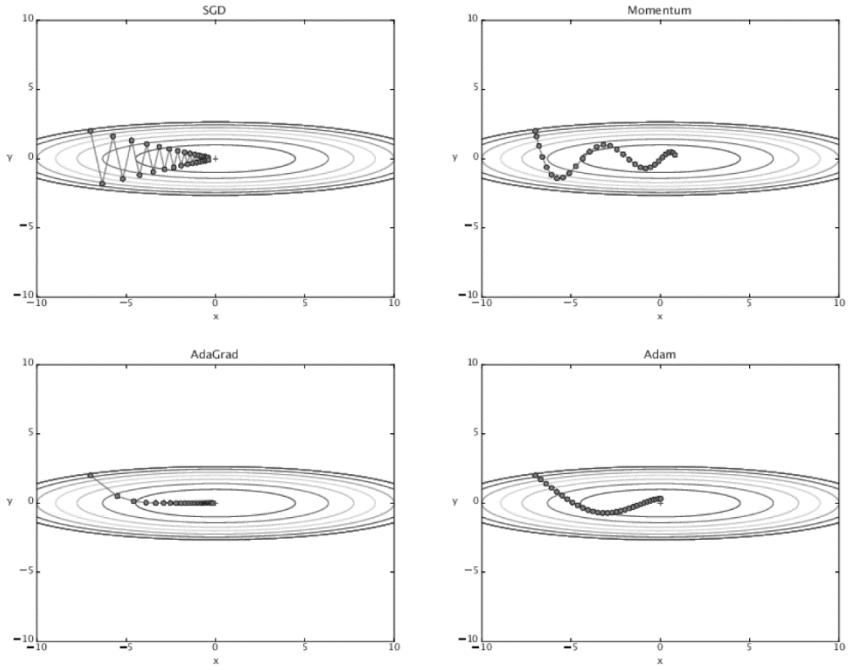
아이디어: momentum + AdaGrad

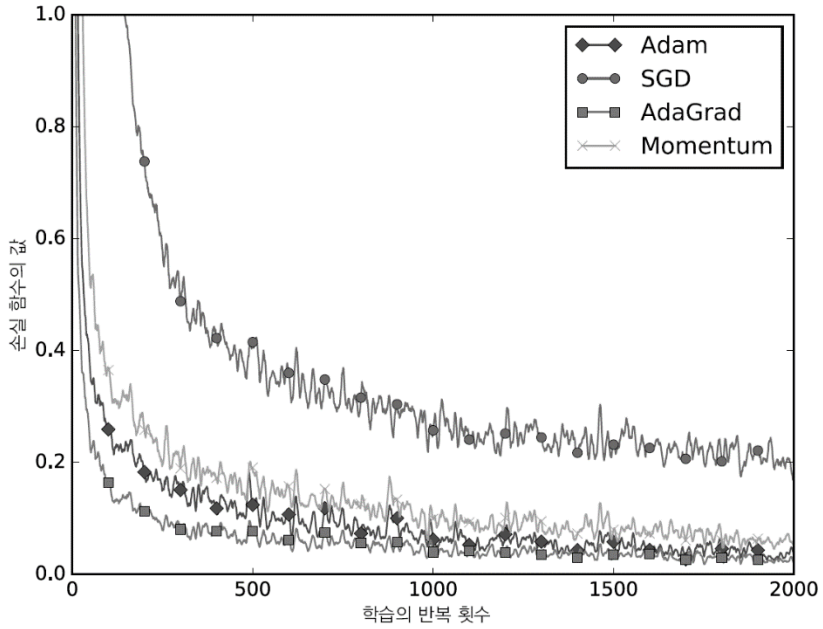


매개변수 최적화

그렇다면 무조건 Adam?

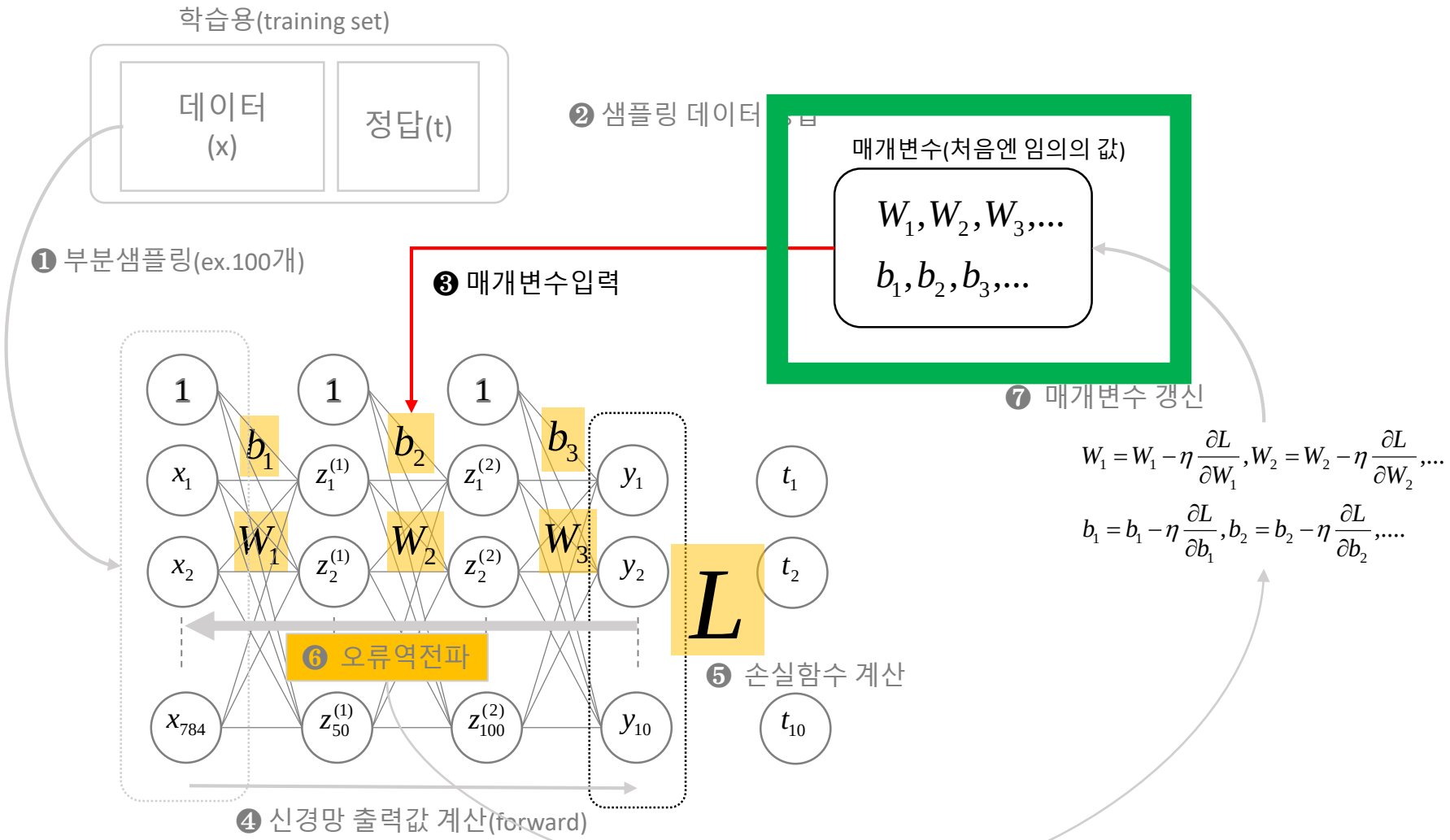
데이터 마다 다름 !!!





매개변수 초기화 방법

매개변수 초기화


학습용(training set)
데이터 (x)
정답(t)
❷ 샘플링 데이터
매개변수(처음엔 임의의 값)
$W_1, W_2, W_3, \ldots$
$b_1, b_2, b_3, \ldots$
❶ 부분샘플링(ex.100개)
❸ 매개변수입력
❼ 매개변수 갱신
$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1}, W_2 = W_2 - \eta \frac{\partial L}{\partial W_2}, \ldots$
$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}, b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}, \ldots$
❻ 오류역전파
❺ 손실함수 계산
❹ 신경망 출력값 계산(forward)
❶~❼ 과정 반복

매개변수 초기화

한 줄의 코드

정규분포로 랜덤 숫자 생성 했음!

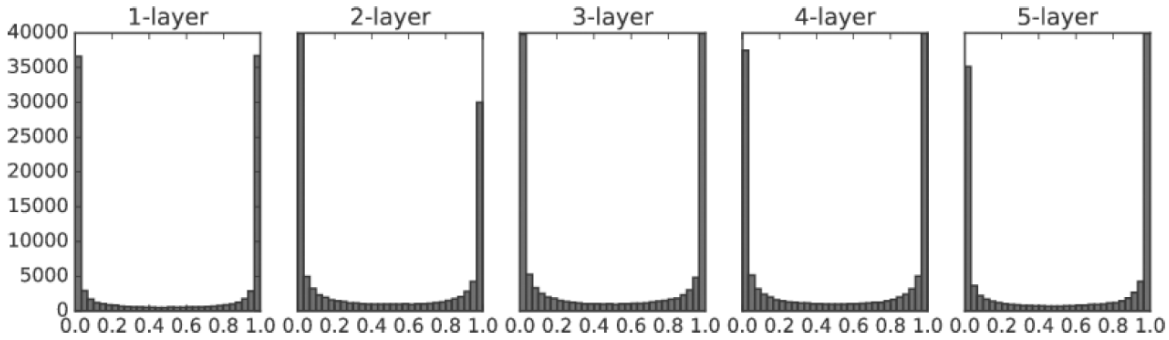
```
w = np.random.randn(…) * std + mean
```

매개변수 초기화

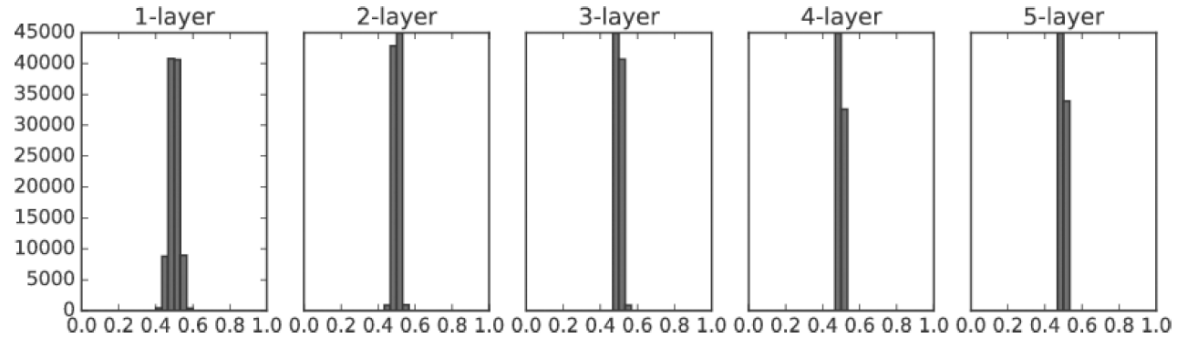
문제점

표준편차에 따라 매개변수의 분포가 달라진다. > 골고루 퍼지는 것이 목표!

```
w = np.random.randn(…) * 1
```




```
w = np.random.randn(…) * 0.01
```



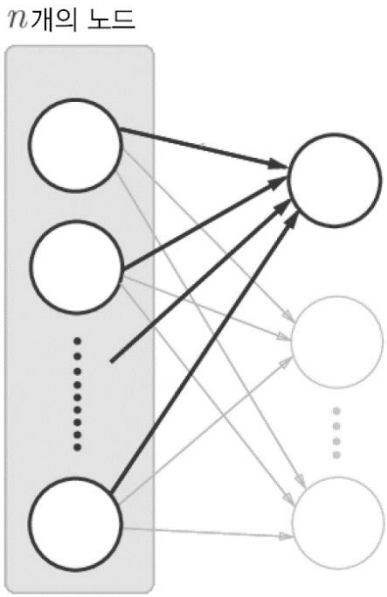
매개변수 초기화

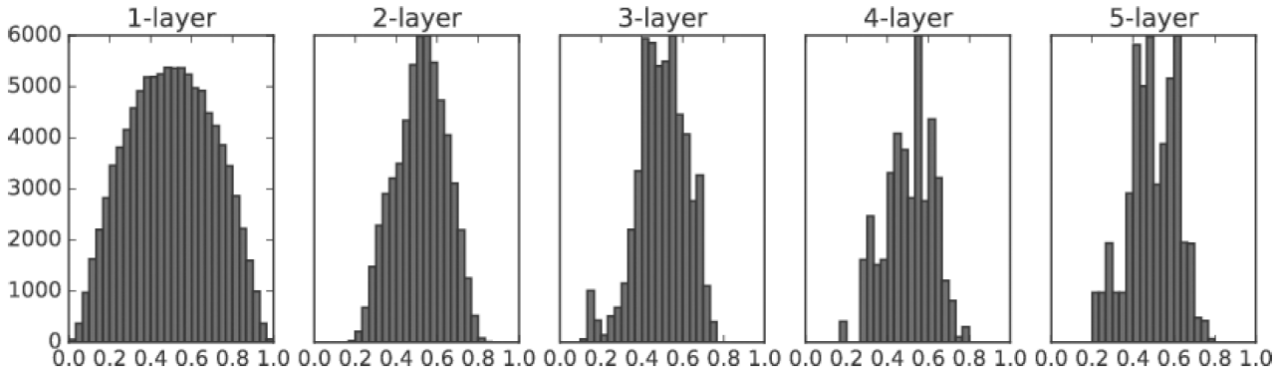
사비에르(Xavier) 초기값

$$n \rightarrow \sqrt{\frac{1}{n}}$$

노드 개수를 활용한 표준편차 사용

```
w = np.random.randn(node_num,node_num) / np.sqrt(node_num)
```

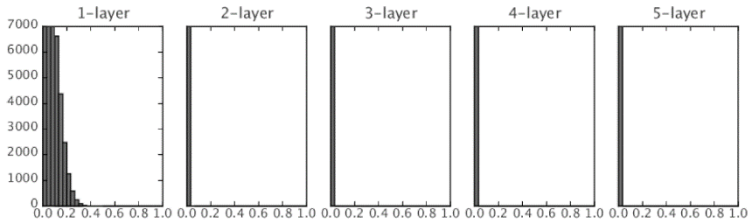




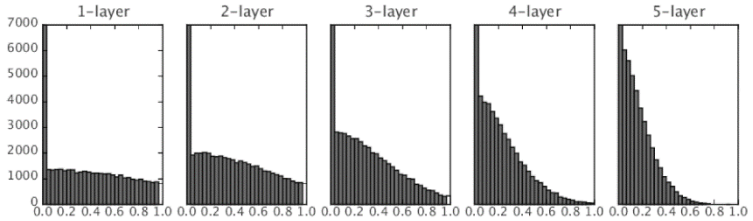
매개변수 초기화

He 초기값 (ReLU 활성화 함수용) $n \rightarrow \sqrt{\frac{2}{n}}$

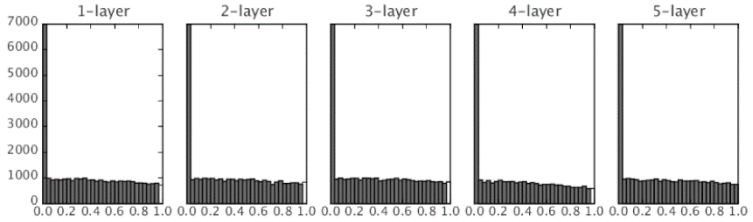
```
w = np.random.randn(node_num,node_num) * np.sqrt(2/node_num)
```



표준편차가 0.01인 정규분포를 가중치 초깃값으로 사용한 경우



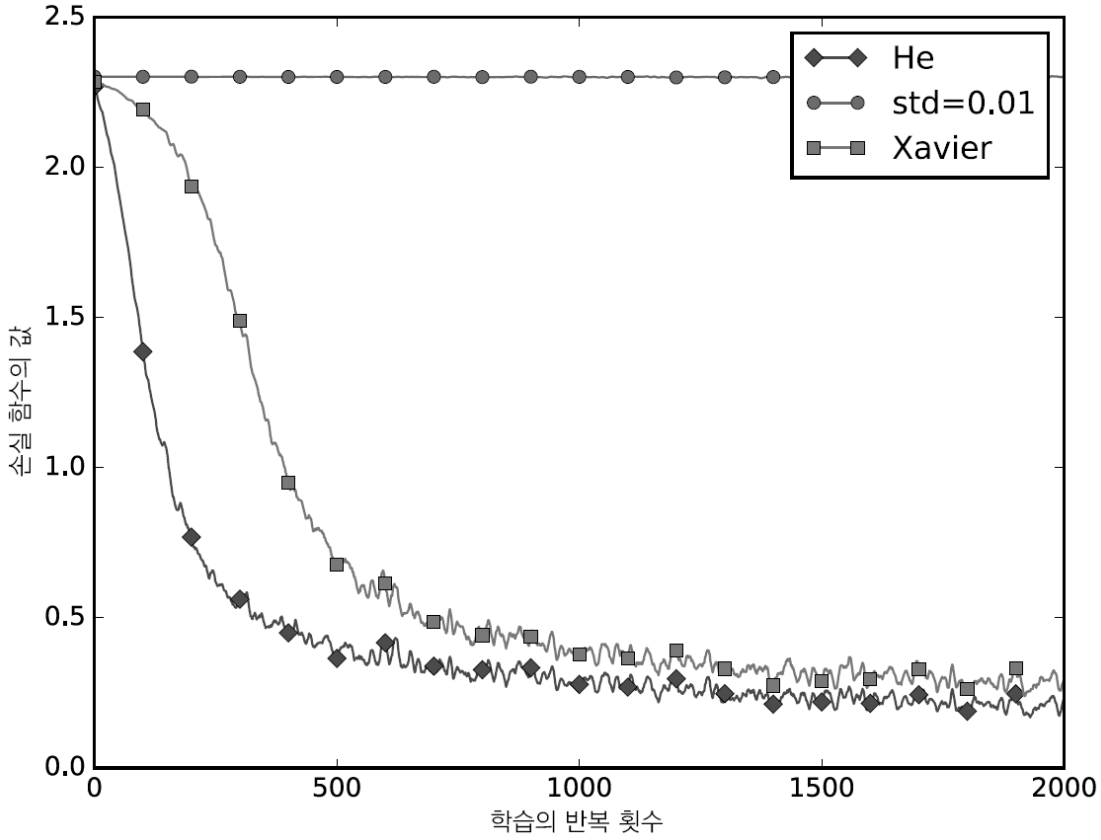
Xavier 초깃값을 사용한 경우



He 초깃값을 사용한 경우

매개변수 초기화

MNIST 케이스

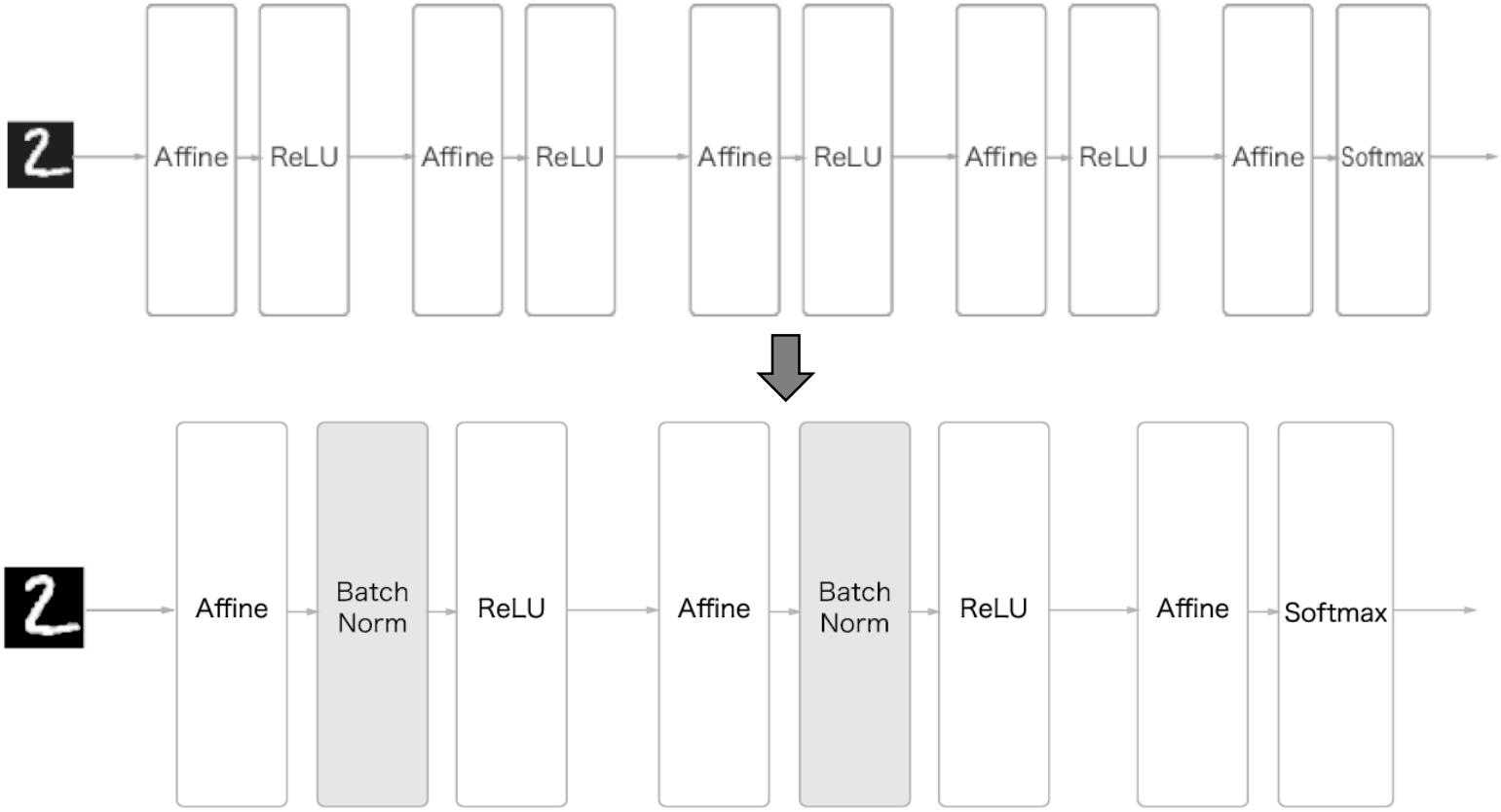


배치 정규화

적절한 분포가 좋은 결과를 유도한다면
강제로 각 레이어 마다 분포를 조절하면 어떨까?

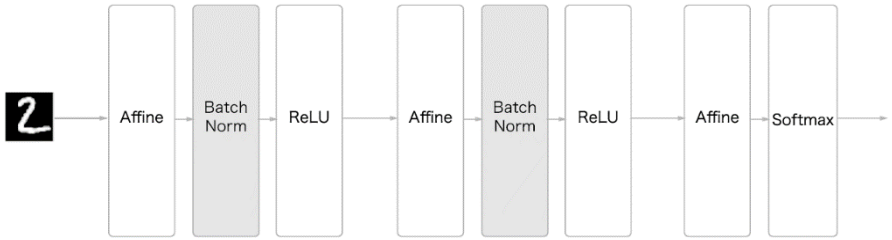
배치정규화Batch Normalization

MNIST 케이스



배치정규화 계층(Batch norm)을 아예 삽입!

배치정규화Batch Normalization



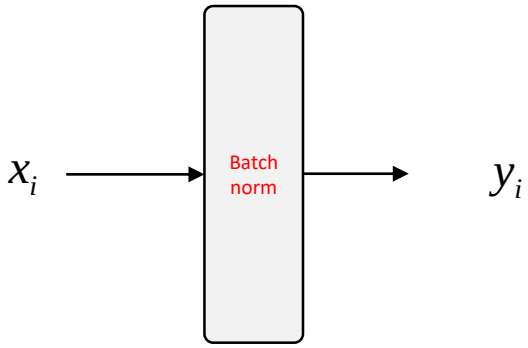
Batch norm layer

$$B = \{x_1, x_2, ..., x_m\}$$

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

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

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



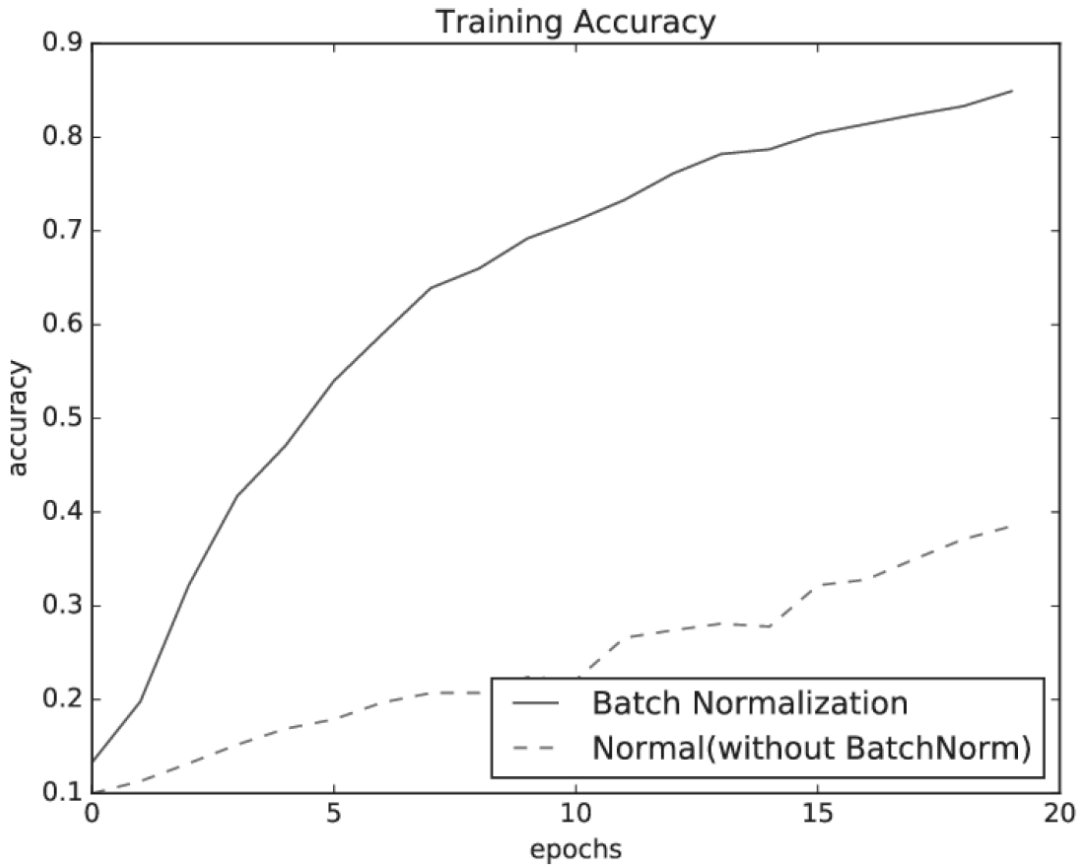
Batch norm transform

$$y_i \leftarrow \gamma \hat{x}_i + \beta$$

배치단위로 간단히 분포를 변환시키는 방법

배치정규화Batch Normalization

효과 Speed!



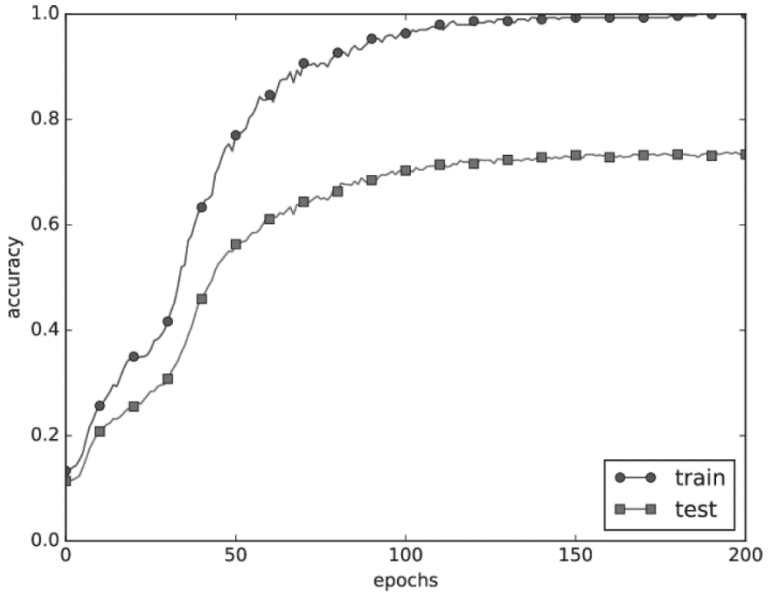
오버피팅 억제기법

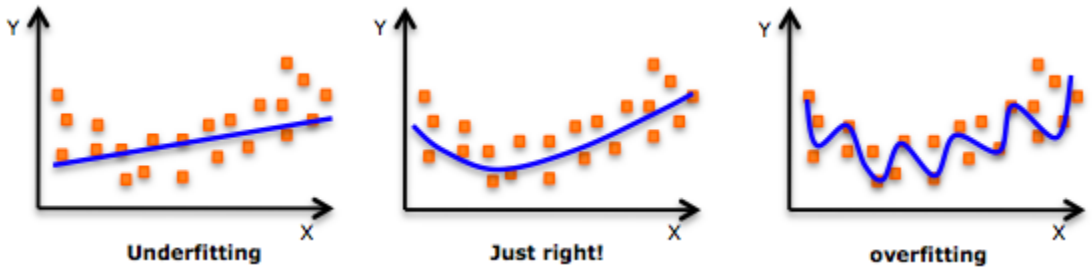
가중치 감소weight decaying

드롭아웃dropout

오버 피팅overfitting

100퍼센트? Training 데이터에 적응되는 문제!





http://sanghyukchun.github.io/75/

가중치감소weight decay

학습 과정에 가중치간 크기 차이가 커지는 현상 방지(오버피팅의 원인)

큰 가중치에 큰 패널티 값을 부여해서 가중치 차이를 줄임

손실함수에 가중치의 합(L2, L1등 여러가지 가능) 추가해서 계산

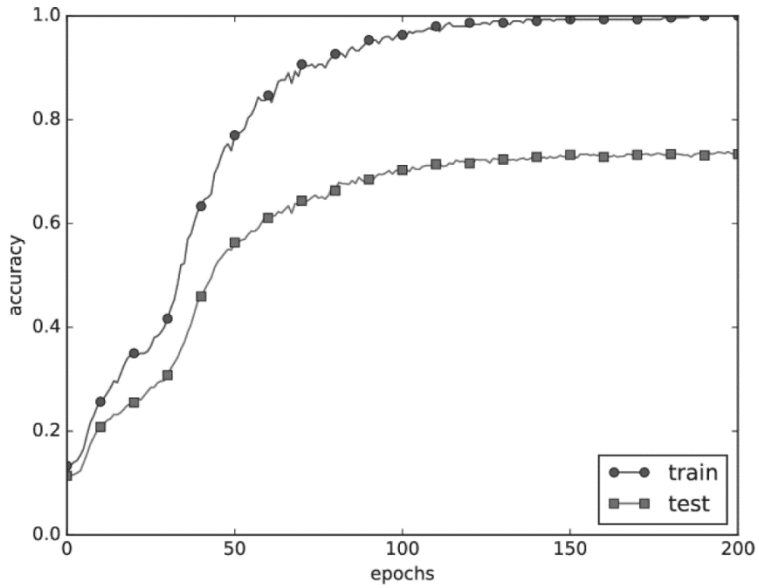
$$L = L_{data} + \frac{1}{2}\lambda\|\mathbf{W}\|^2$$

결과적으로 가중치를 갱신할 때 특정한 값(λw)을 지속적으로 뺌

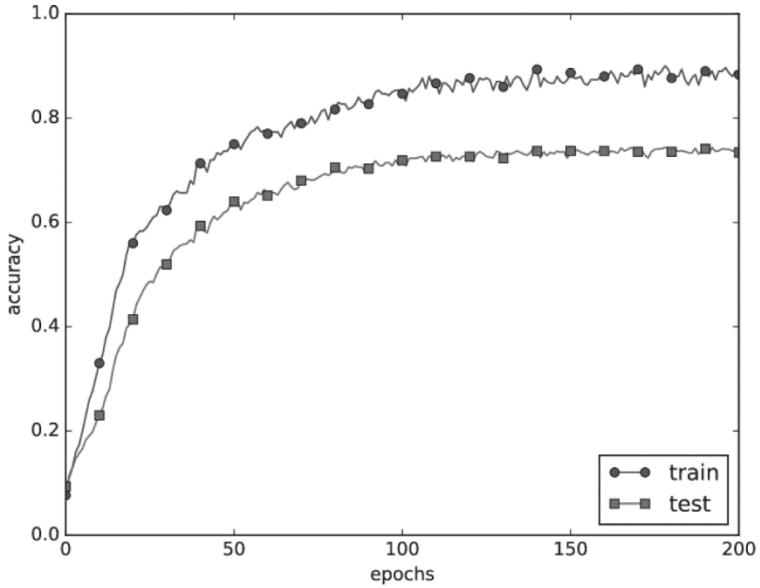
$$w_{t+1} = w_t - \alpha\Delta_w L_{data} - \lambda w$$

가중치감소weight decay

결과



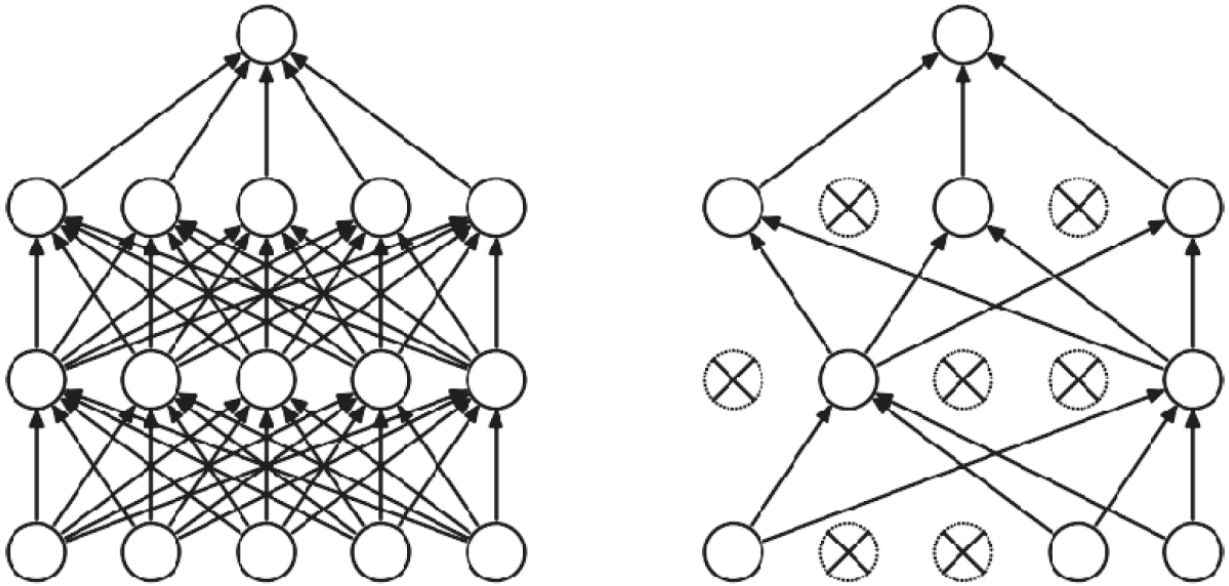
without



with

드롭아웃dropout

뉴런을 학습 과정에서 임의로 삭제하면서 학습시키는 방법

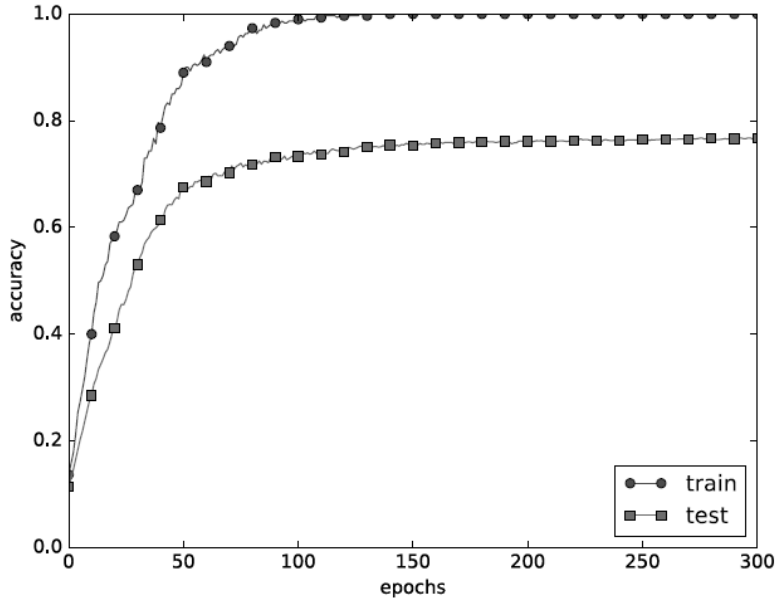


(a) 일반 신경망

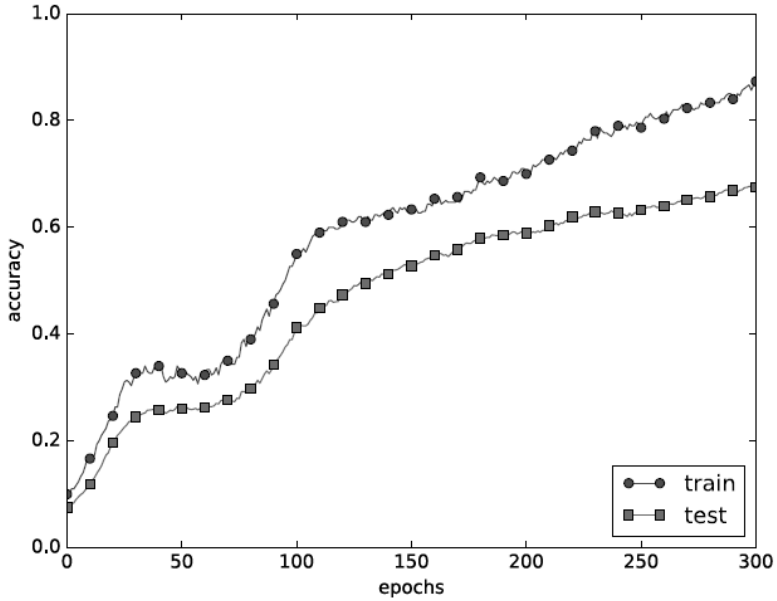
(b) 드롭아웃을 적용한 신경망

드롭아웃dropout

결과



without



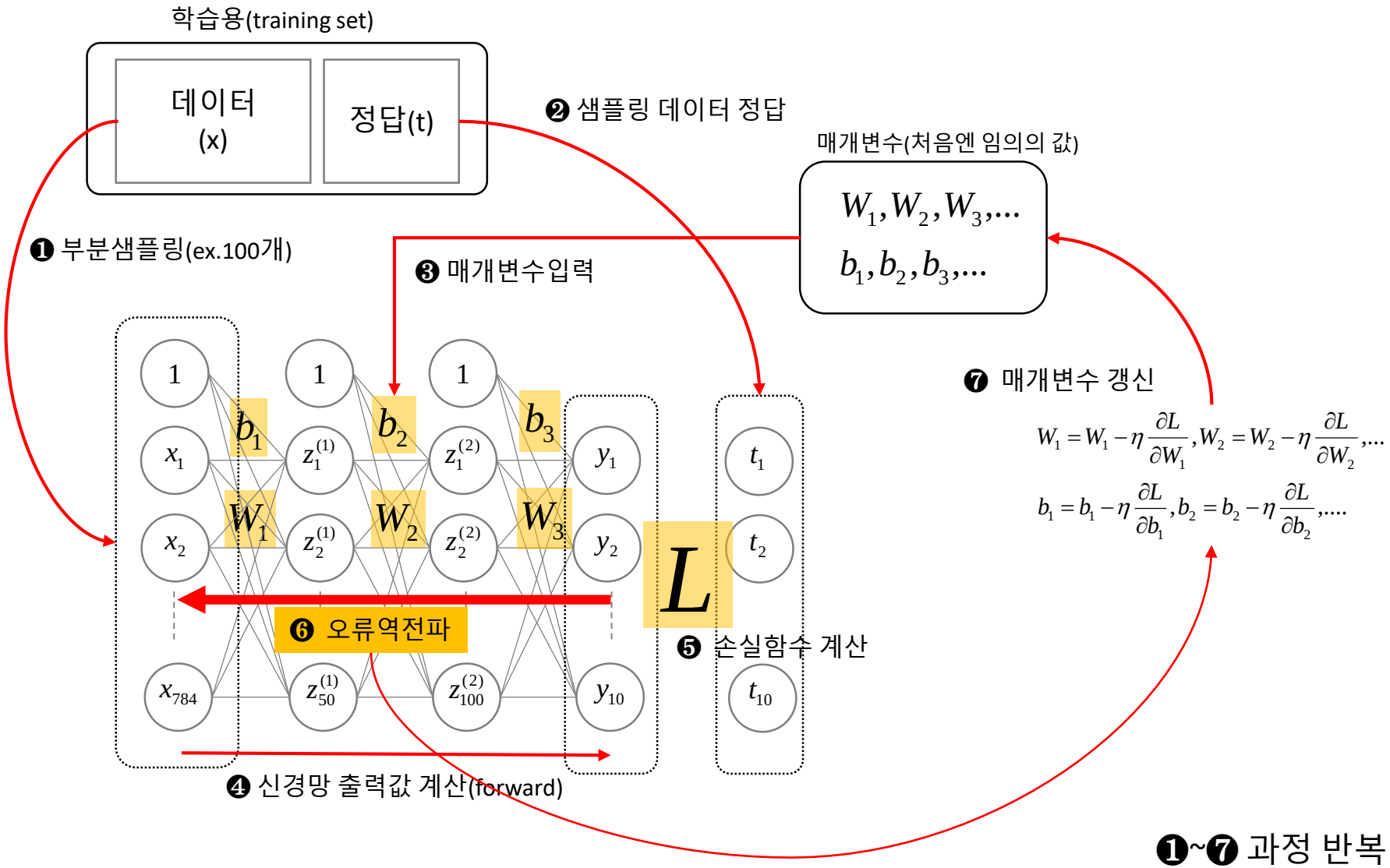
with

하이퍼 파라미터 설정방법

Parameter vs. Hyper-parameter

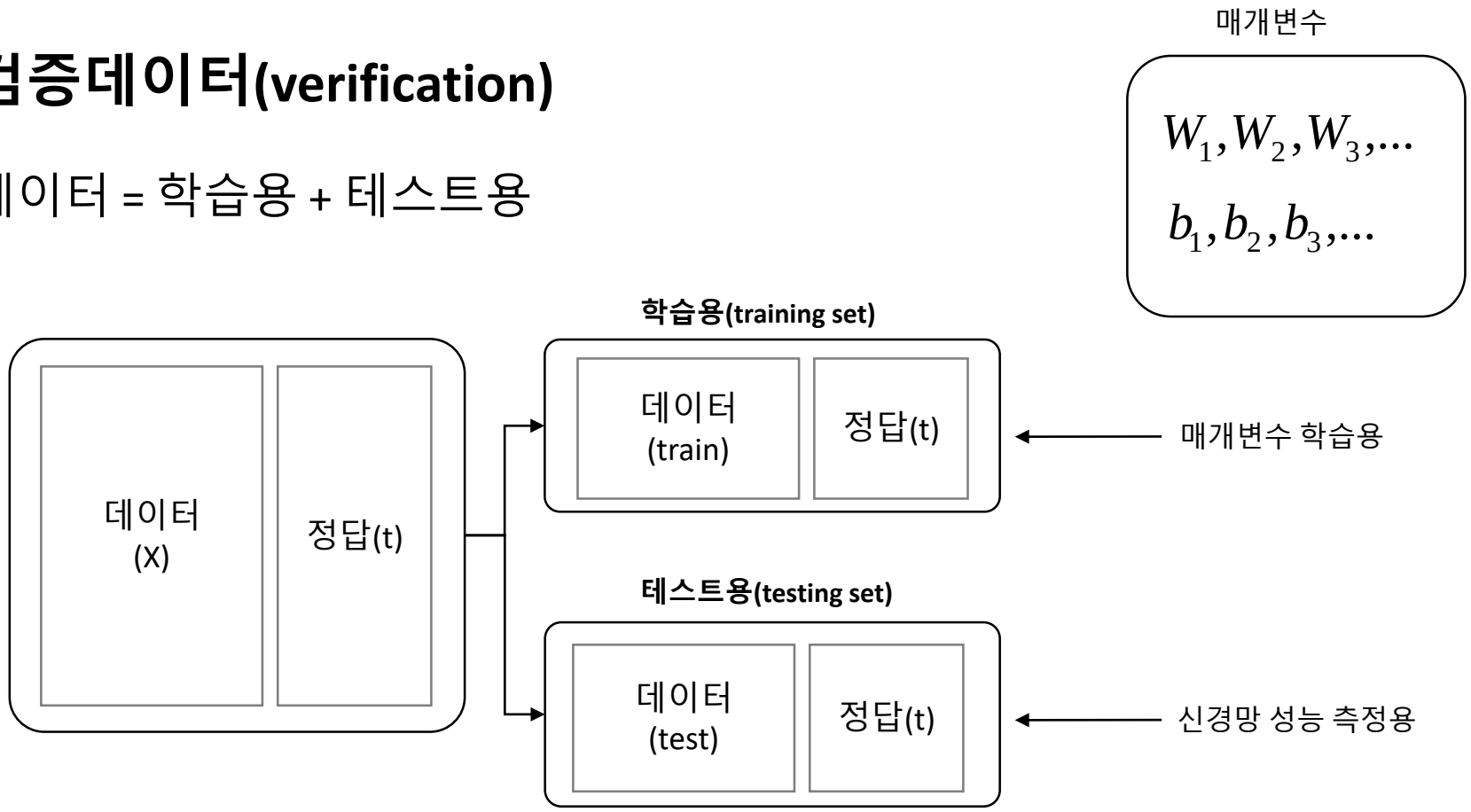
- Parameter(매개변수)
  - Ex. 가중치weight, 편향값bias
  - 학습 알고리듬에 의해서 정해지는 값
- Hyper-parameter
  - Ex. 학습률(learning rate), 신경망 노드 개수
  - 인간이 결정함

검증데이터(verification)



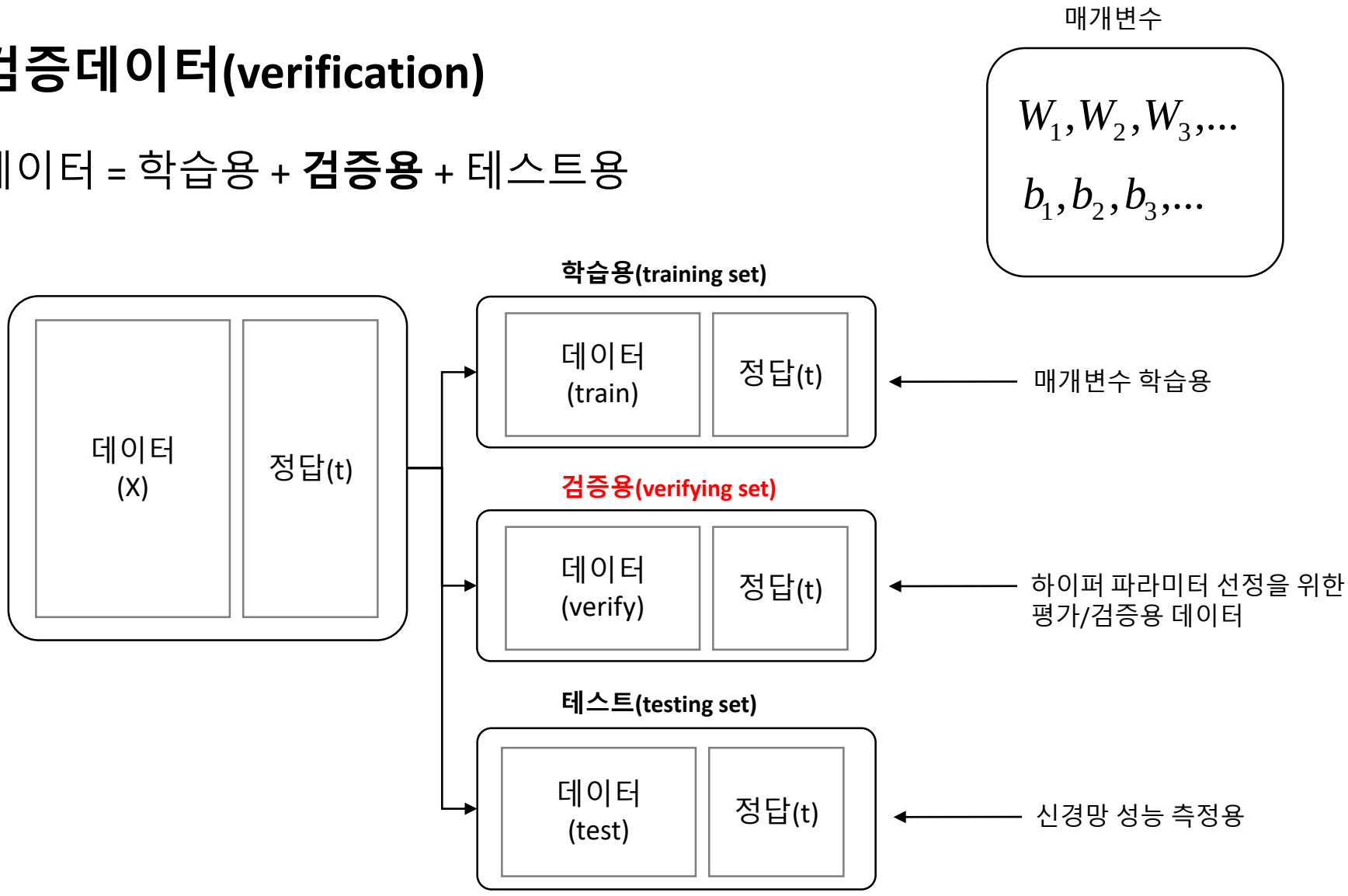
검증데이터(verification)

데이터 = 학습용 + 테스트용

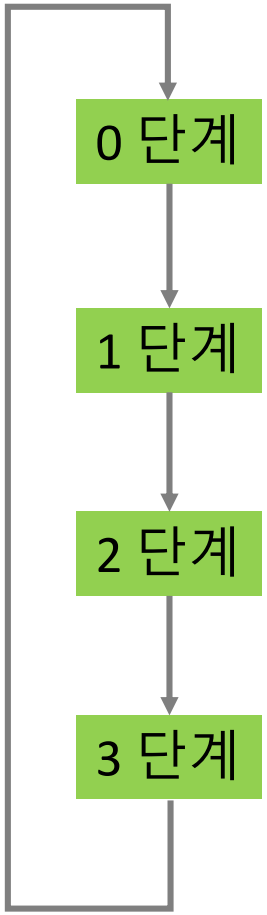


검증데이터(verification)

데이터 = 학습용 + **검증용** + 테스트용



하이퍼파라미터 최적화



하이퍼 파라미터 값의 범위 설정

설정된 범위에서 하이퍼 파라미터 값 무작위 추출

설계된 신경망을 학습하고 검증데이터로 정확도 평가

1단계/2단계를 반복하여 정확도에 따라 범위 선택

끝