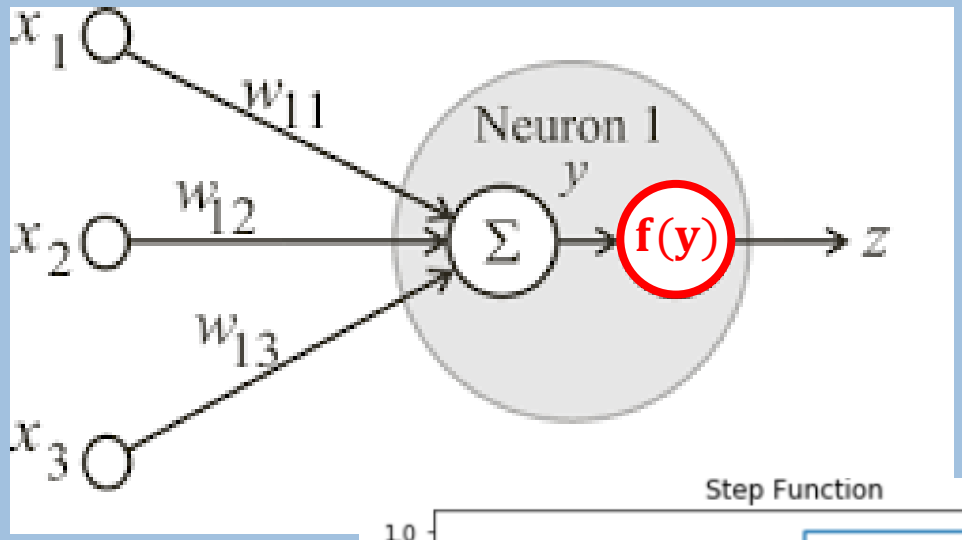
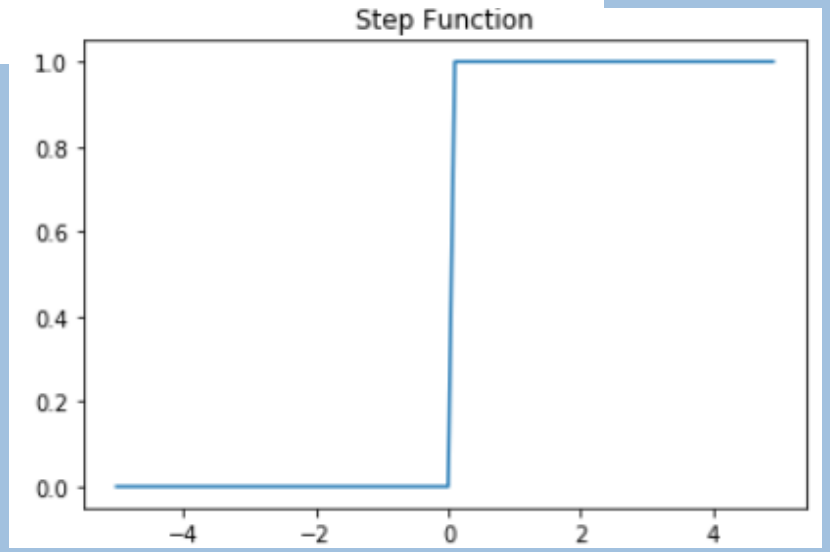
2022-1 겨울방학
딥러닝 분반

분반장: 구은아, 김혜림

활성화 함수

활성화 함수

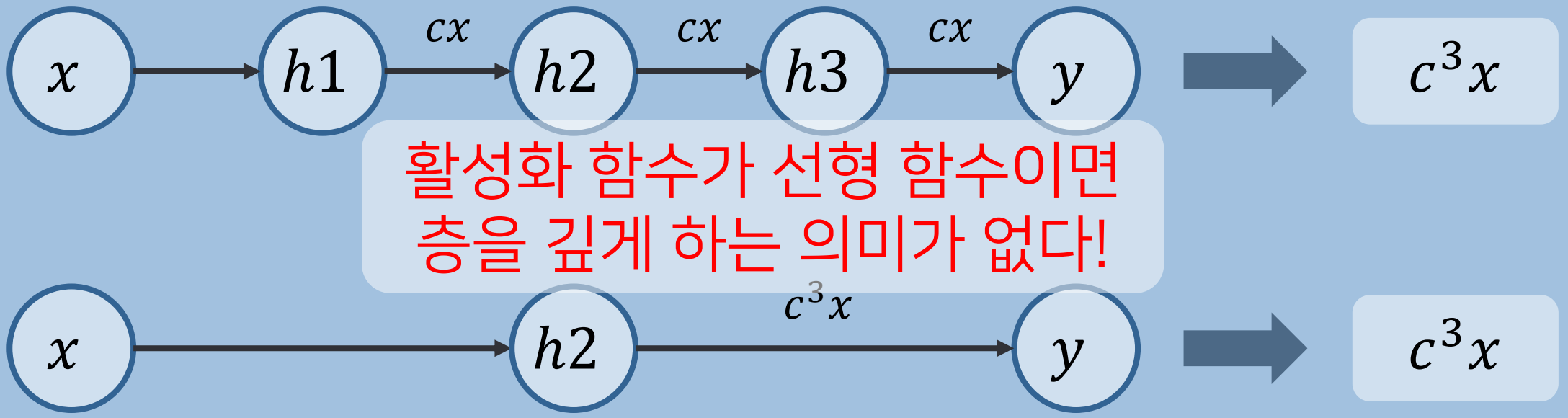




활성화 함수
activation function

인공신경망에서 입력 신호를 출력 신호로 변환하는 함수

활성화 함수가 필요한 이유

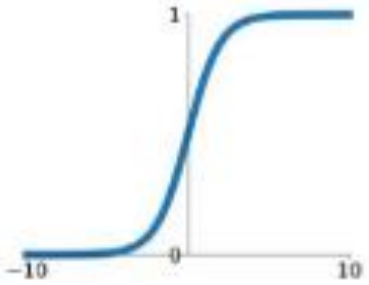


활성화 함수는 인공 신경망이 비선형성을 갖도록 하는 역할을 합니다.
인공 신경망은 비선형성을 가짐으로써 층을 여러 개 쌓을 때의 효과를 얻을 수 있습니다.

활성화 함수의 종류

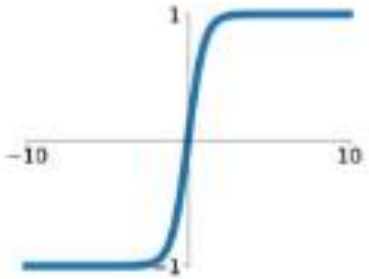
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



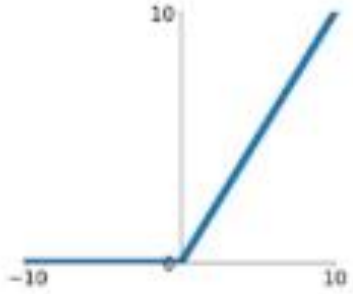
tanh

$$\tanh(x)$$



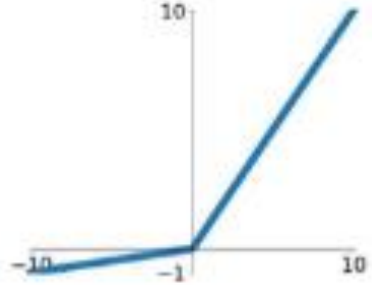
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

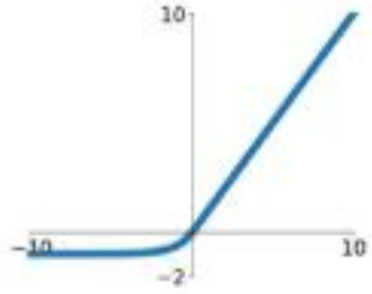


Maxout

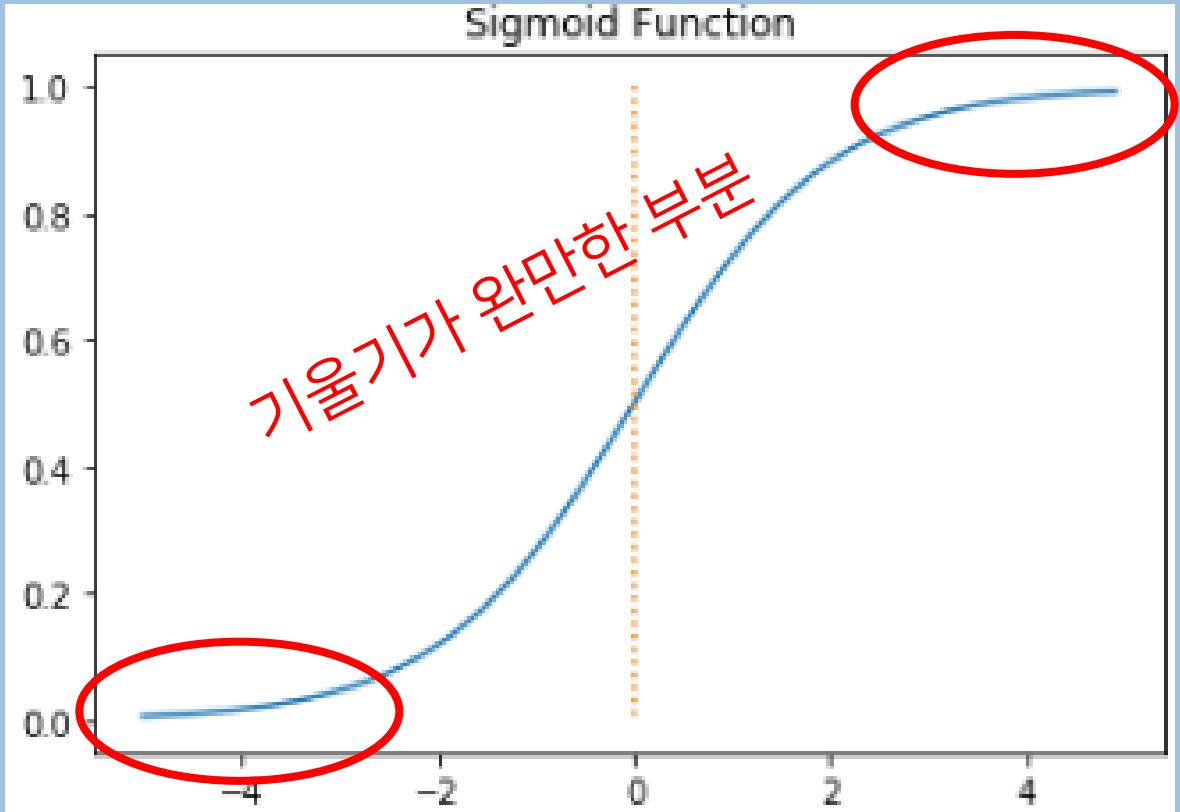
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



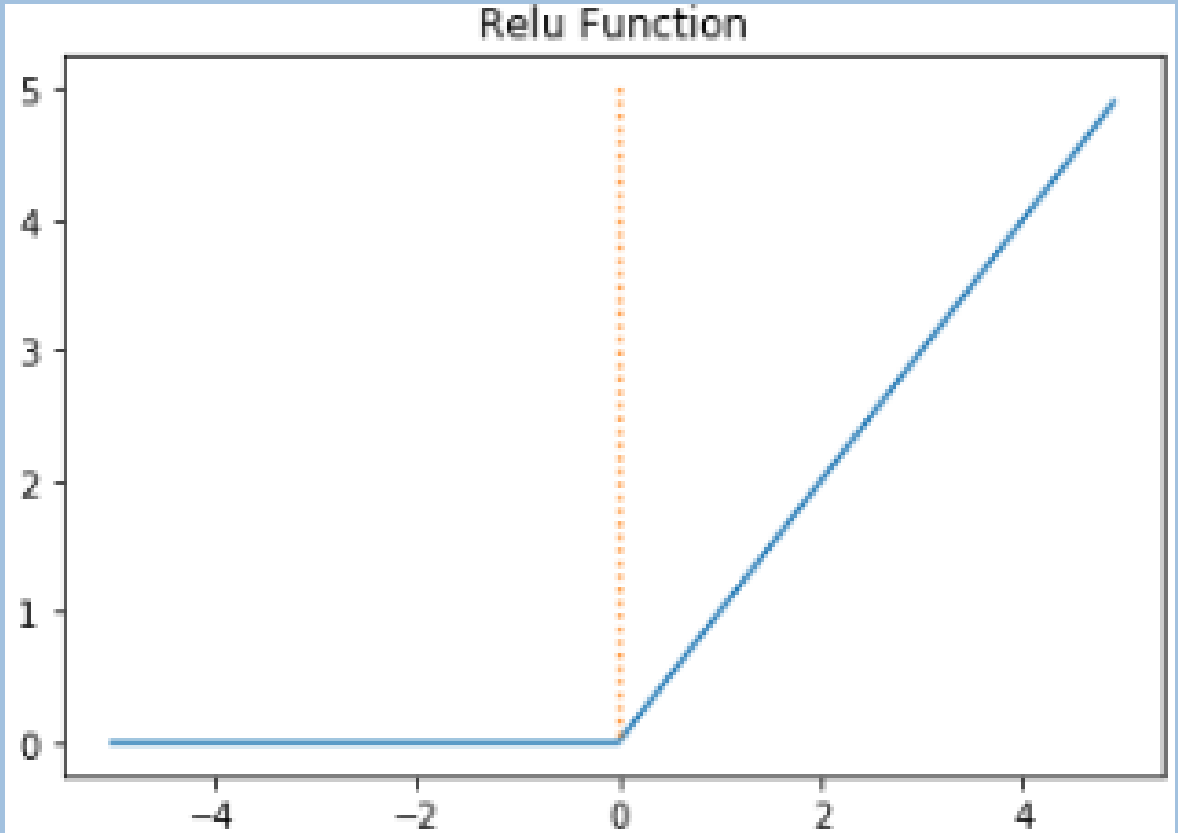
Sigmoid 함수



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- 기울기 소실 문제가 발생할 수 있다.
- 기울기 소실 문제: 역전파 과정에서 0에 가까운 작은 기울기가 계속 곱해져서 신경망 앞부분에서는 가중치가 거의 갱신되지 않는 현상. 학습이 멈춤.
- 이 문제 때문에 Sigmoid 함수를 활성화 함수로 사용하는 것은 지양되는 편이다.

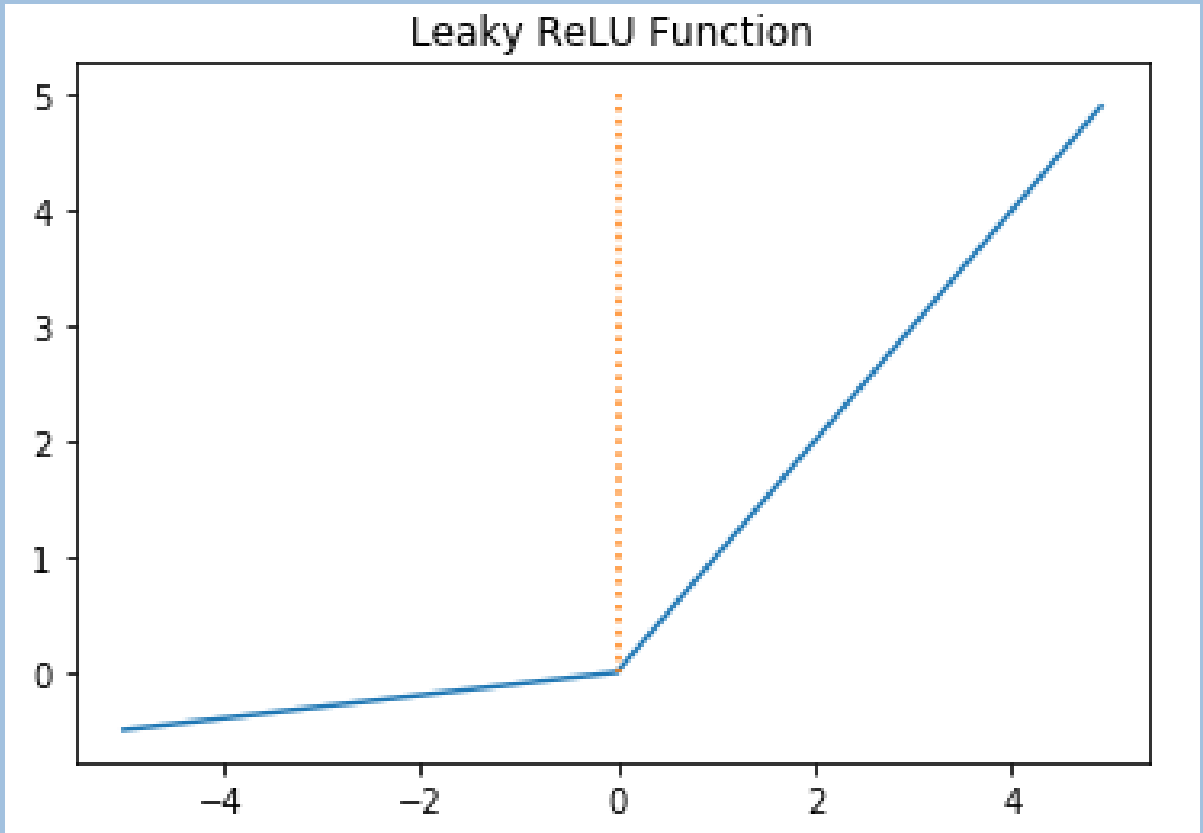
ReLU 함수



$$f(x) = \max(0, x)$$

- 입력값이 음수이면 0을, 양수이면 그대로 출력하는 함수
- Sigmoid 함수와 달리 양수 부분에서 특정 값에 수렴하지 않으므로 기울기 소실 문제가 일어나지 않음
- 활성화 함수로 굉장히 많이 쓰이는 함수

Leaky ReLU 함수



$$f(x) = \max(ax, x)$$

- 입력값이 음수이면 아주 작은 수를, 양수이면 그대로 출력하는 함수
- a는 음수일 때 기울기, 하이퍼 파라미터이며 0.01 정도의 아주 작은 값으로 설정
- 입력값이 음수일 때, 출력값이 0이 되어서 학습이 멈추는 경우를 방지

Optimizer

Optimizer

모델의 성능을
높여야 함

오차를 줄이는
방법 필요

loss function
최소화

가중치를 갱신해야 함

가중치를 갱신하는
도구 필요

optimizer

용어 정리

batch(배치)

큰 데이터를 일정한 크기(batch size)로 묶은 데이터 묶음

epoch

전체 데이터셋에 대해 순전파+역전파 과정을 통해 한 번의 학습을 하는 과정

iteration

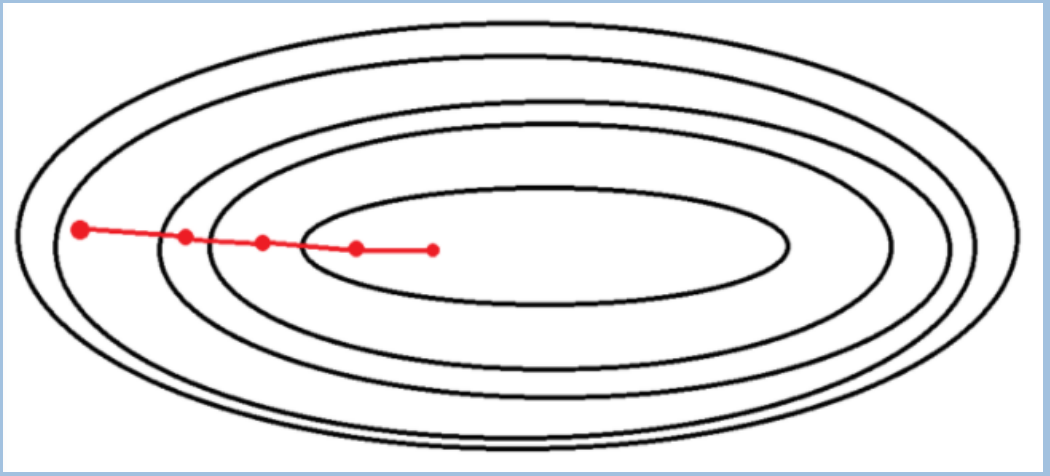
한 epoch를 끝내기 위해 (전체 데이터를 학습시키기 위해) 필요한 배치의 개수

Batch Gradient Descent

전체
데이터셋
(n개)

모든 iteration에서 사용

- 전체 데이터가 n개라면 매 iteration 마다 n개의 데이터를 모두 사용하는 방법
- 학습 시간이 오래 걸림
- 과적합 문제 발생 가능성 높음
- local minimum에 빠질 위험이 큼



Stochastic Gradient Descent

전체 데이터셋 (n개)

데이터 1개 → 한 iteration에서 사용

데이터 1개 → 한 iteration에서 사용

데이터 1개 → 한 iteration에서 사용

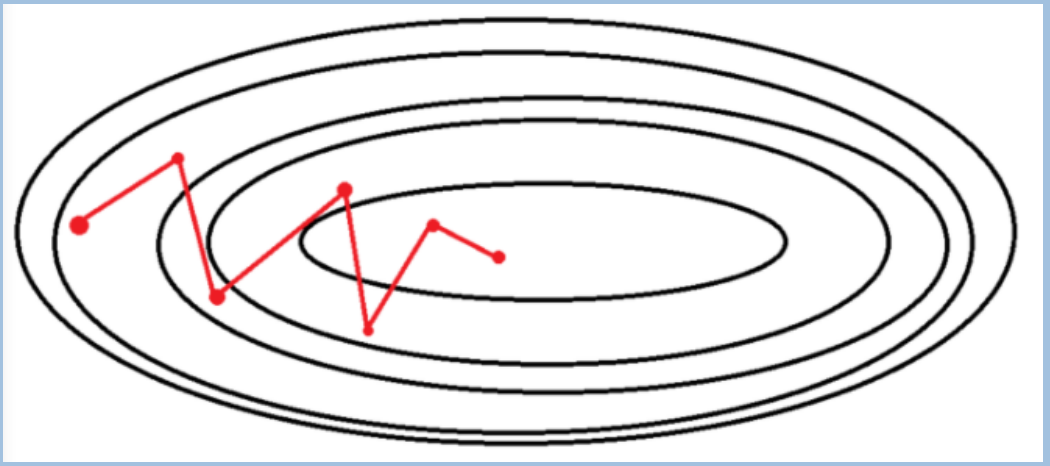
데이터 1개 → 한 iteration에서 사용

⋮ ⋮

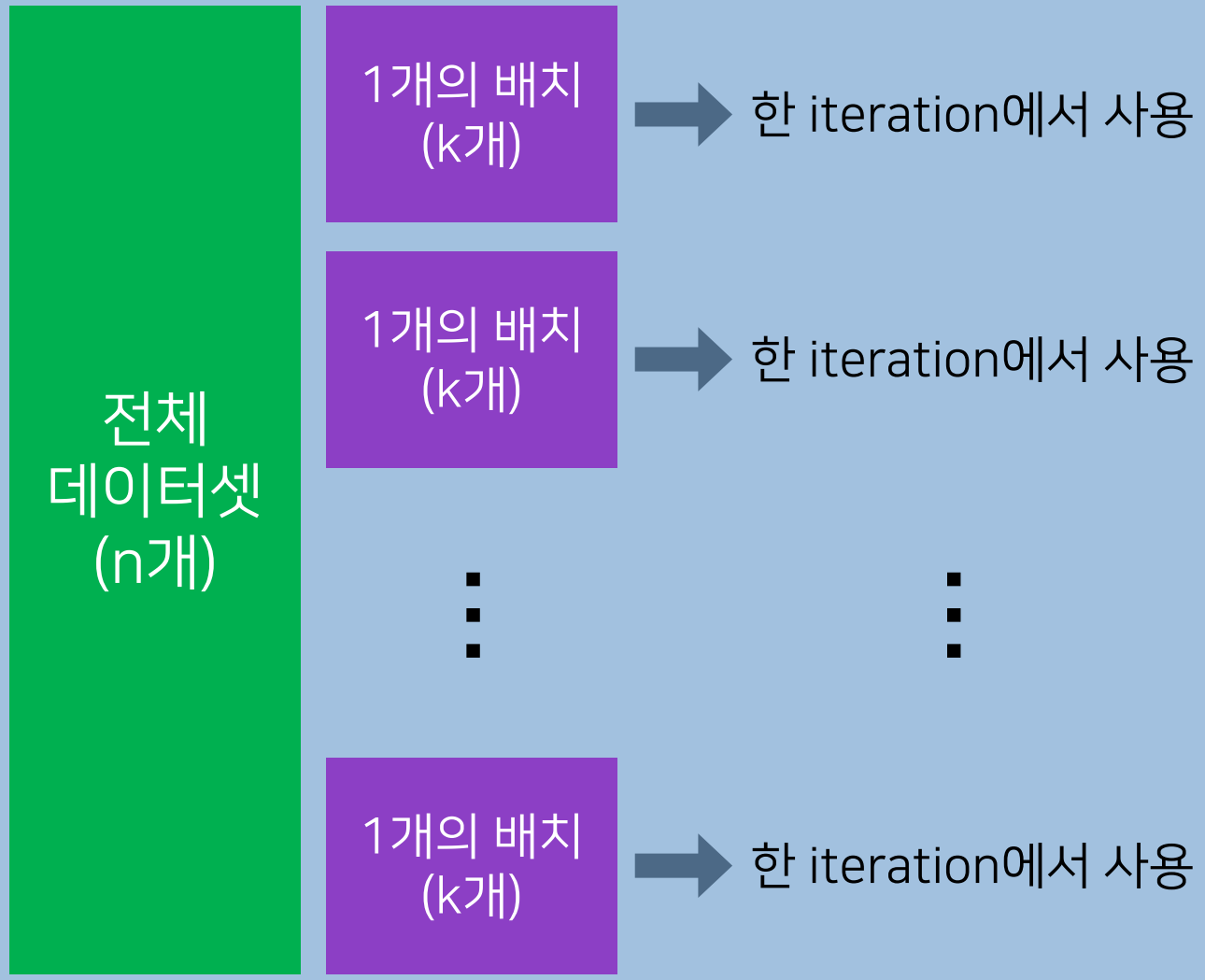
데이터 1개 → 한 iteration에서 사용

데이터 1개 → 한 iteration에서 사용

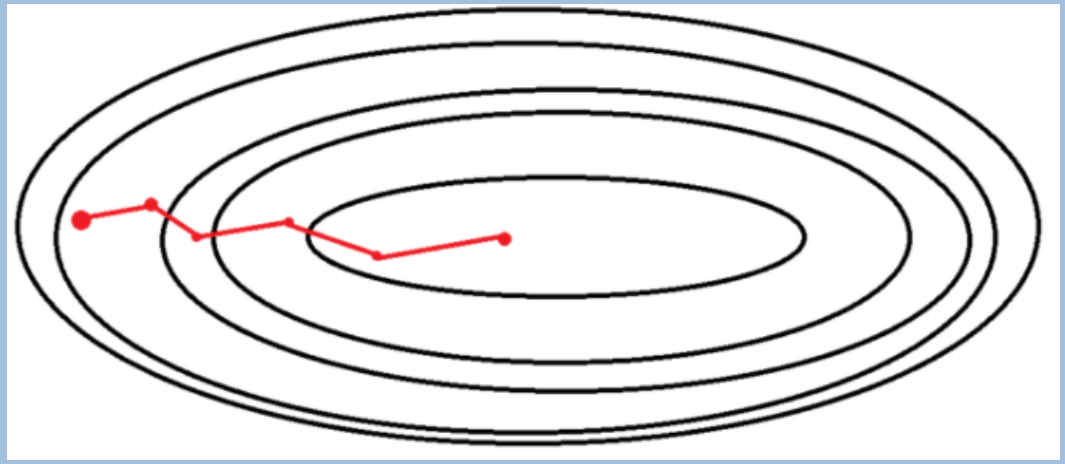
- 매 iteration마다 무작위로 1개의 데이터를 샘플링하여 학습하는 방법
- n iteration = 1 epoch
- 과적합될 위험이 적고, 학습속도가 빠름
- 최적화 갱신 경로가 비효율적



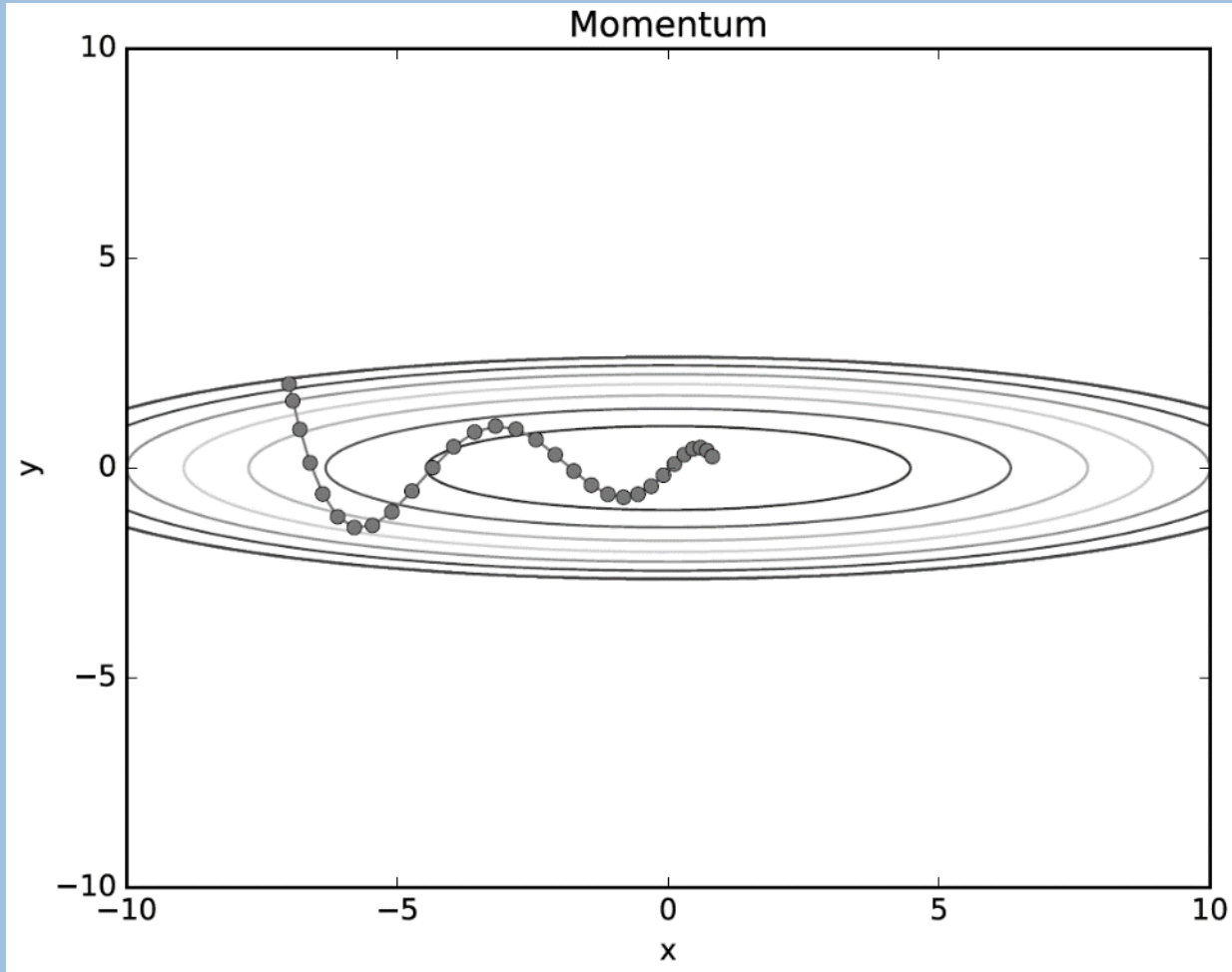
Minibatch Gradient Descent



- 전체 데이터를 몇 개의 배치로 분리하고 한 iteration에서 1개의 배치만 사용하는 방법
- n/k iteration = 1 epoch
- SGD보다 경로가 효율적이고, BGD보다 학습 속도가 빠름



Momentum

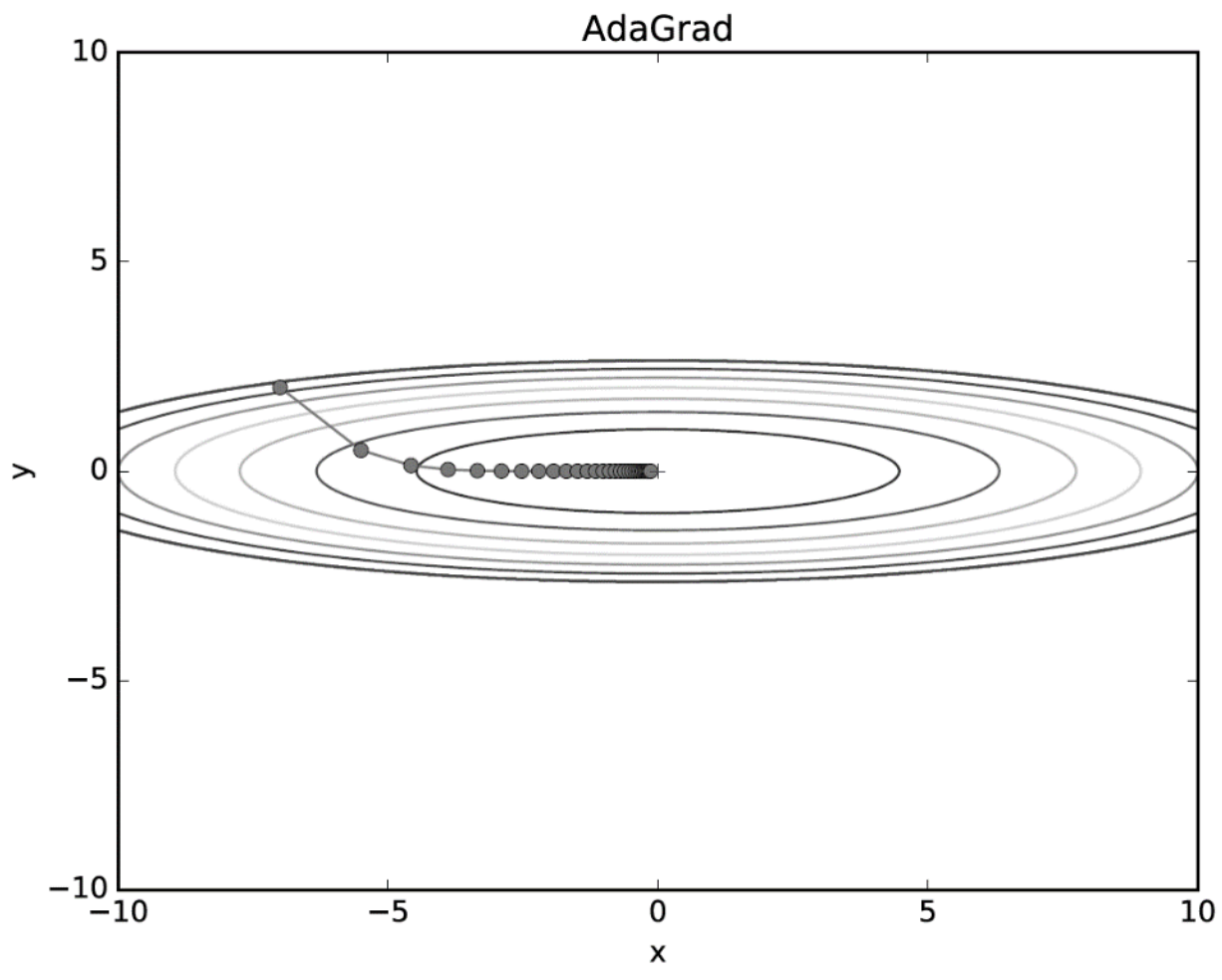


$$v \leftarrow mv - \alpha \frac{\partial L}{\partial W}$$

$$W \leftarrow W + v$$

- v : 관성, 이전 학습에서 가중치가 갱신된 방향
- m : 모멘텀 계수, 관성을 얼마나 크게 둘 것인지 결정
- 관성을 반영하여 SGD보다 더 효율적인 경로로 최적점을 찾아감

AdaGrad

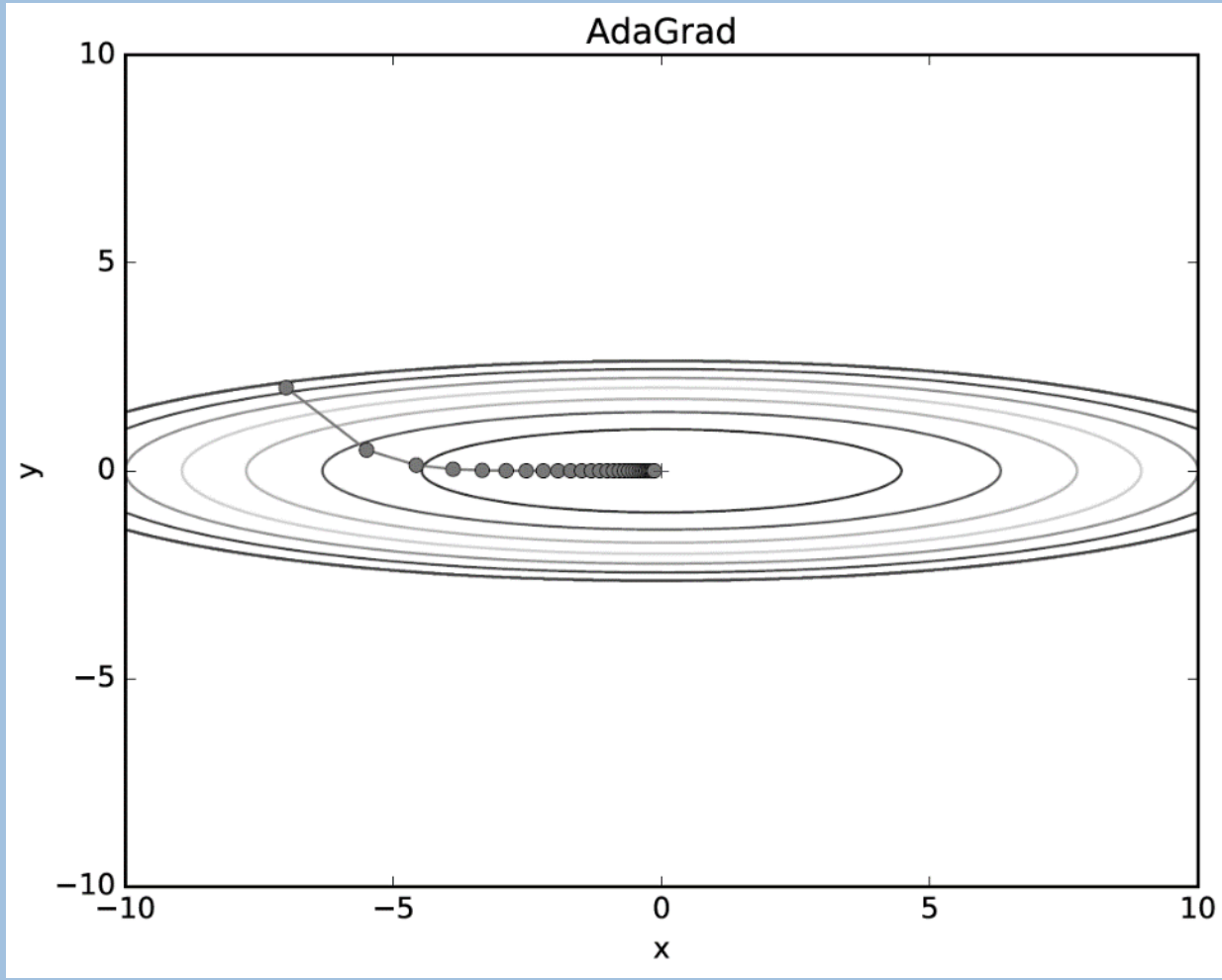


$$h \leftarrow h + \frac{\partial L}{\partial \boldsymbol{W}} \odot \frac{\partial L}{\partial \boldsymbol{W}}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \frac{1}{\sqrt{h}} \frac{\partial L}{\partial \boldsymbol{W}}$$

- h : 기존 기울기 값의 제곱의 누적합
- 갱신이 조금 된 변수는 학습률을 크게 만들어 더 크게 변화하도록
- 갱신이 많이 된 변수는 학습률을 작게 만들어 더 작게 변화하도록

RMSprop

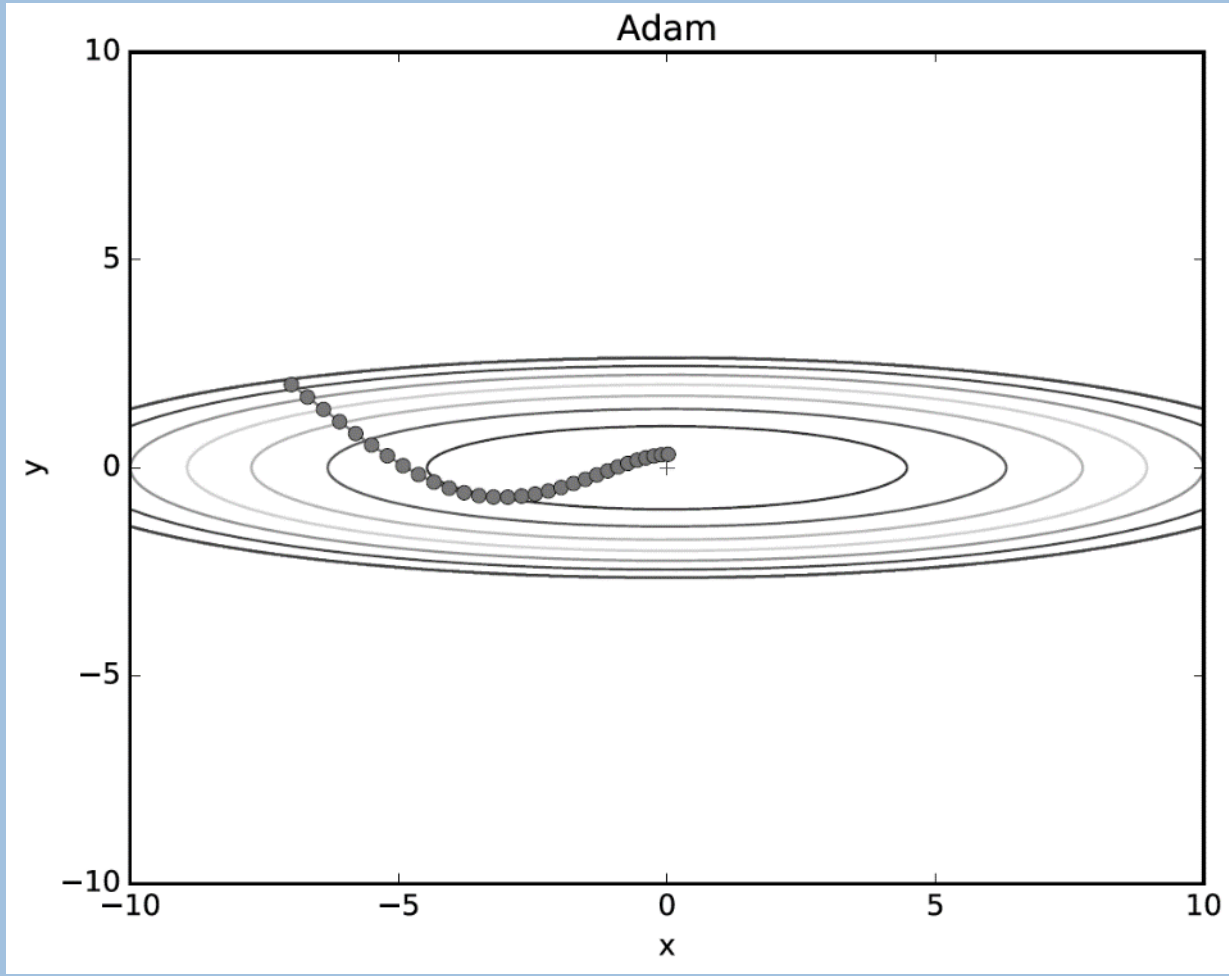


$$h_i \leftarrow \rho h_{i-1} + (1-\rho)\frac{\partial L_i}{\partial \boldsymbol{W}} \odot \frac{\partial L_i}{\partial \boldsymbol{W}}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \frac{1}{\sqrt{h_i}} \frac{\partial L_i}{\partial \boldsymbol{W}}$$

- h_i : 이전 기울기 값들의 제곱의 누적합과 현재 기울기의 제곱합의 지수 이동 평균
- 현재 변화량의 크기에 따라 현재 기울기의 영향력을 ρ로 조정하여 학습률이 극단적으로 감소하는 상황을 방지

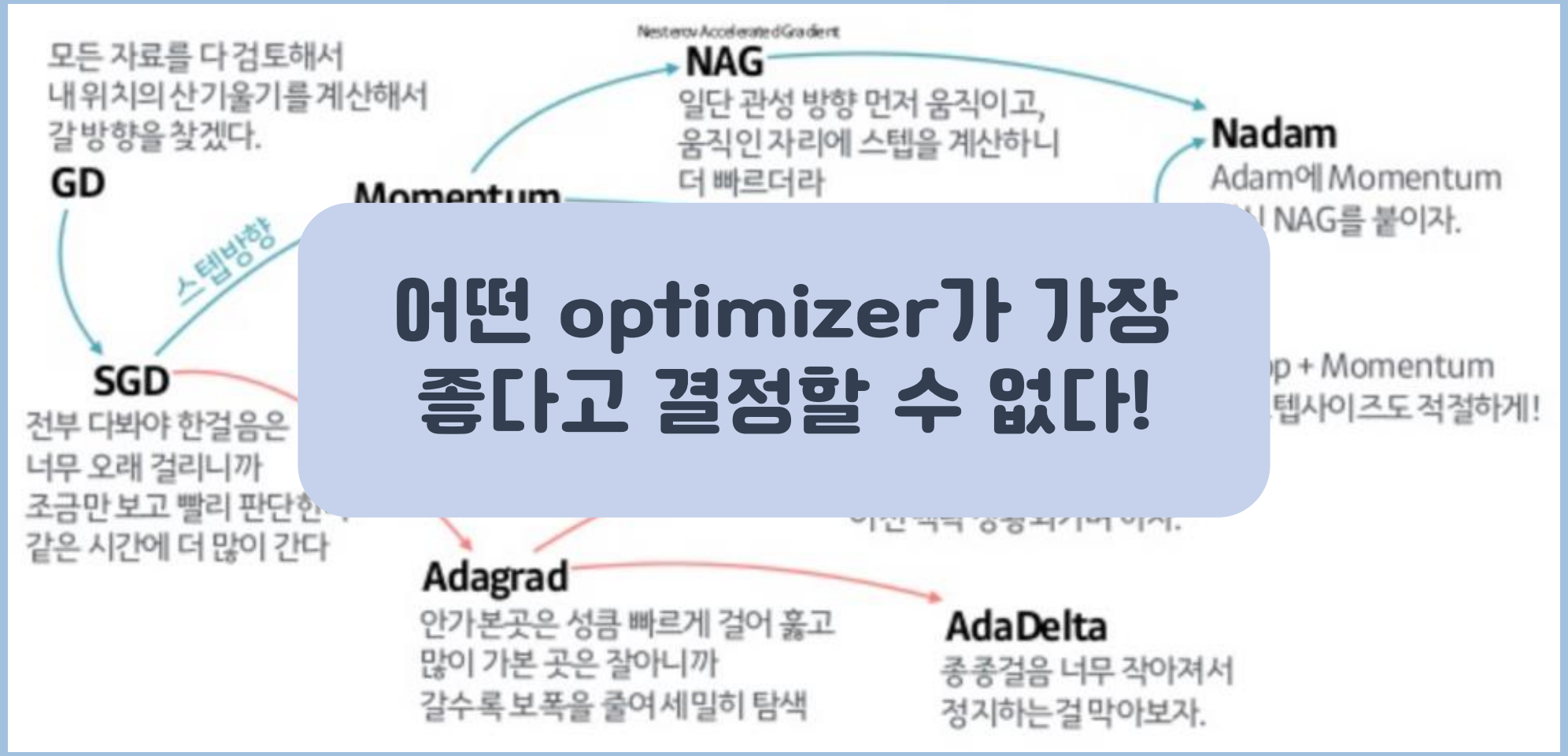
Adam



$$W \leftarrow W + m\frac{\alpha}{\sqrt{v+\epsilon}}$$

- 이전 가중치 변화의 관성 반영+학습률 조정, 2가지 기능을 모두 가진 방법

Optimizer



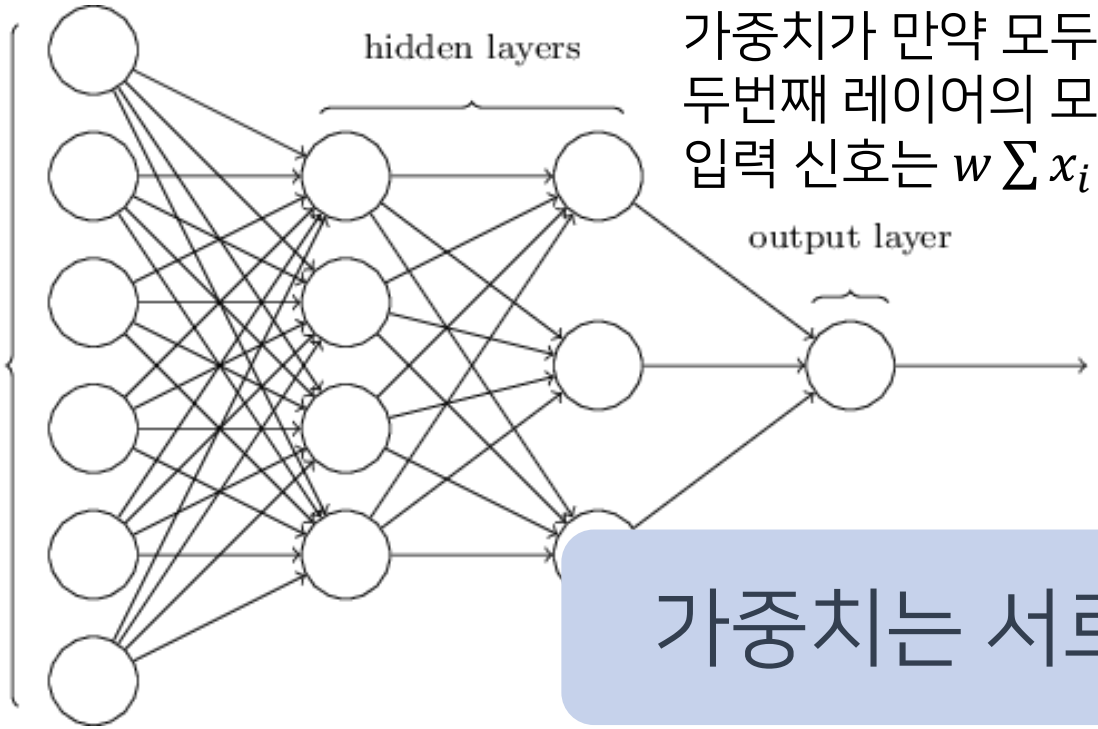
가중치 초기화

가중치가 모두 0이 된다면?

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

점점 작아지는 가중치

가중치가 아예 0이 된다면?

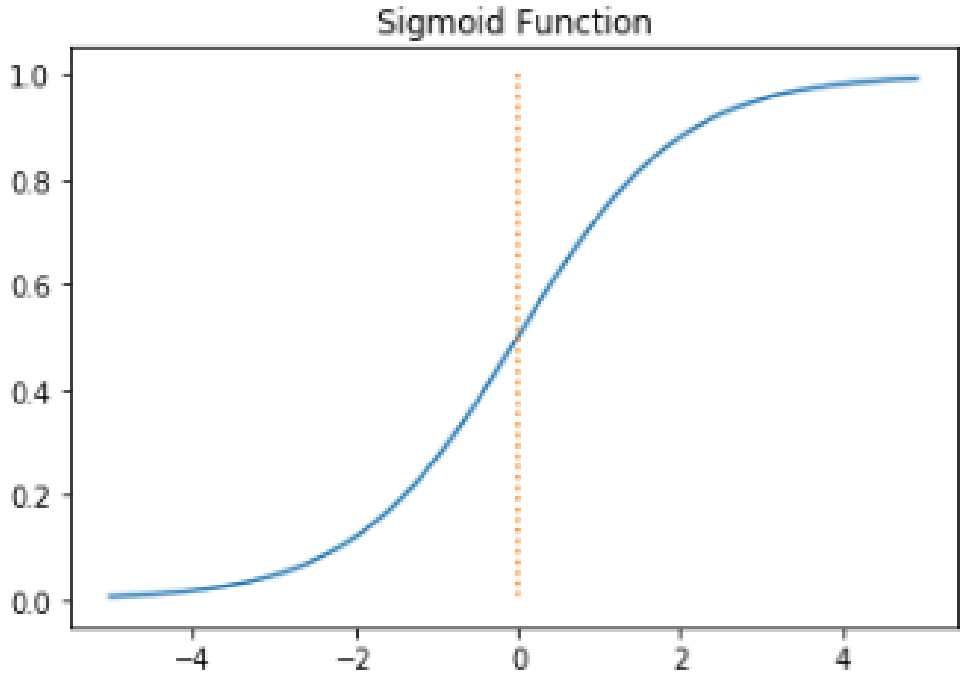


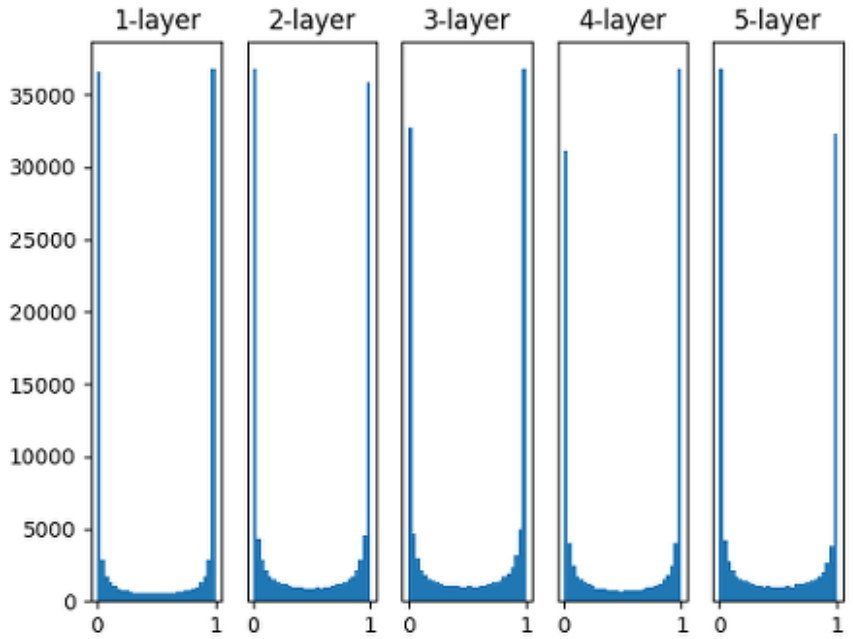
가중치가 만약 모두 w라면, 두번째 레이어의 모든 노드의 입력 신호는 $w\sum x_i$ 일 것이다.

- 가중치가 모두 같은 값이라면 레이어의 모든 노드로 전달되는 값이 똑같다.
- 또한 다음 레이어의 노드로 전달되는 값도 똑같다.
- 이는 역전파에서 가중치가 모두 동일하게 학습되는 문제를 발생시킨다.

가중치는 서로 다른 값이어야 한다

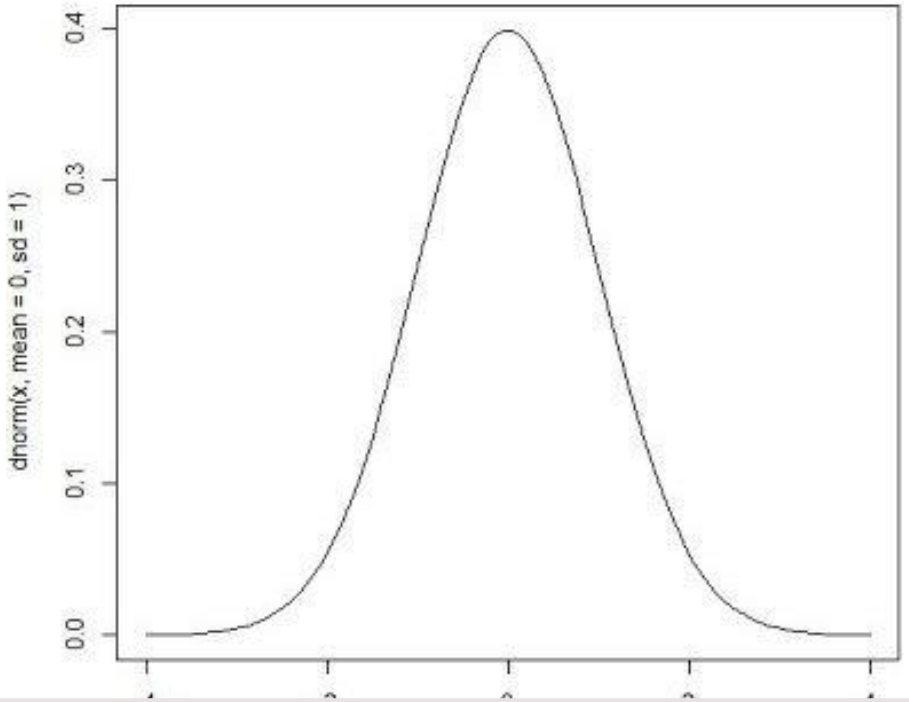
가중치 초기화

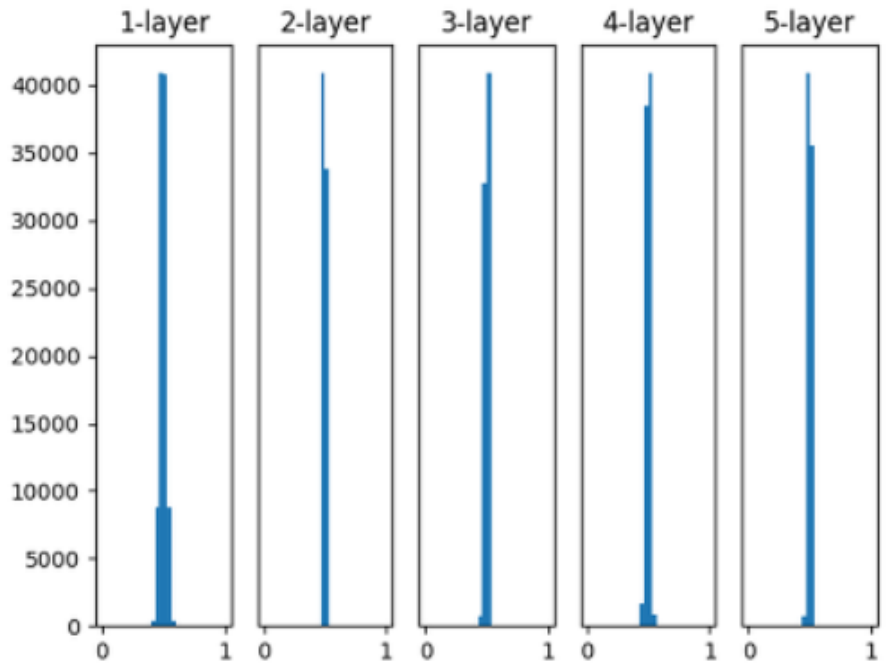




sigmoid 함수를 가중치 초기화 분포로 사용할 경우, 기울기 소실 문제가 발생한다.

가중치 초기화



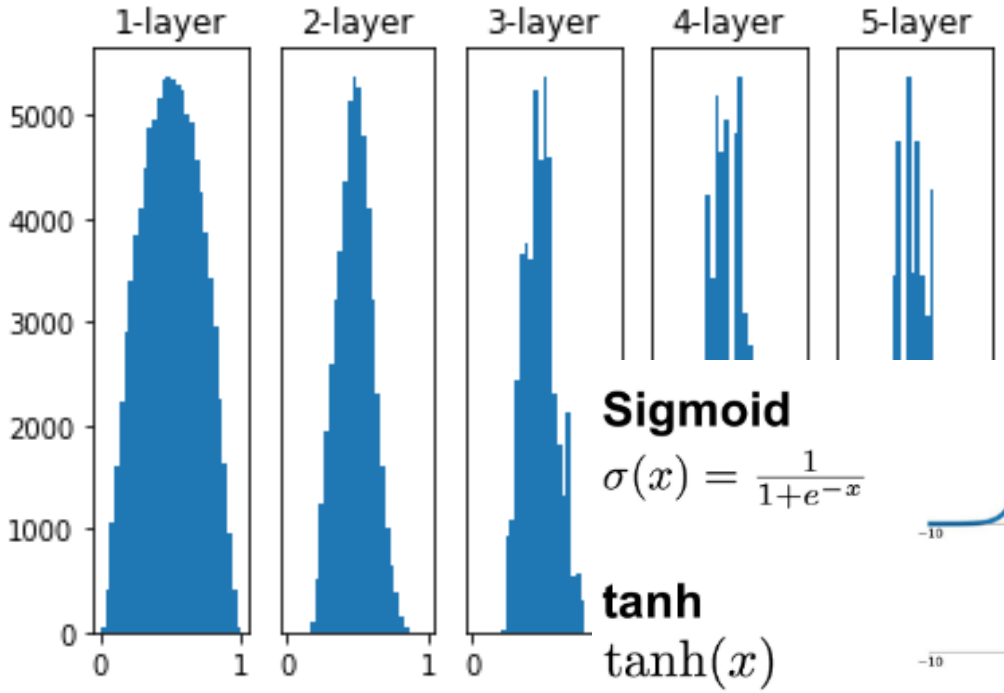


표준편차가 작은 정규분포를 사용할수록 활성화 함수의 출력값이 0과 1에 치우치지 않고 의미 있는 값을 가지게 된다. 그러나 무조건적으로 작은 값을 사용하면 위처럼 0.5 중심으로 집중되는 문제가 발생한다.

Xavier Initialization

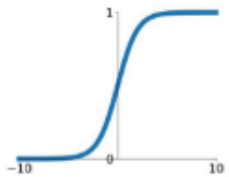
$$\sigma = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

- n_{in}: 이전 층의 뉴런의 개수
- n_{out}: 다음 층의 뉴런의 개수
- 옆의 σ가 표준편차인 정규 분포를 사용하는 것이 Xavier Initialization이다.



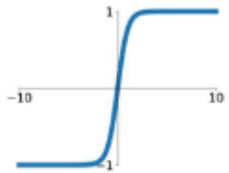
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



tanh

$\tanh(x)$

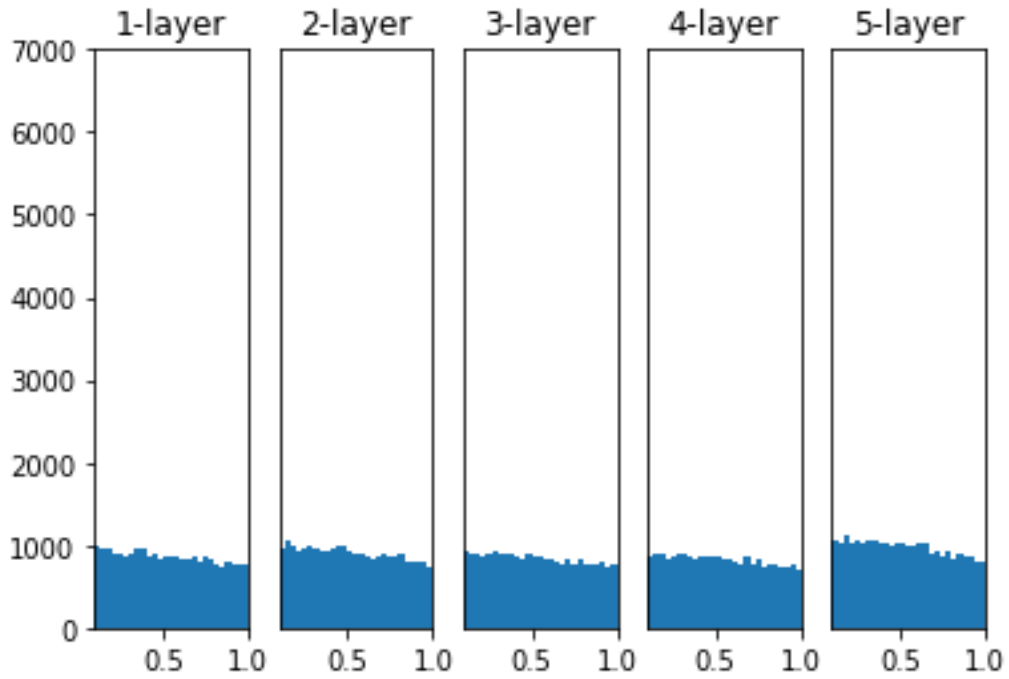


- 이웃 층의 노드 개수를 반영하여 가중치 초깃값으로 설정할 수 있는 값의 범위를 조정할 수 있다.
- 여러 기울기 분산 사이의 균형을 맞추어 특정 층의 가중치가 튀지 않도록 한다. 활성화 함수가 sigmoid, tanh일 때 유용하다.

He Initialization

$$\sigma = \sqrt{\frac{2}{n_{in}}}$$

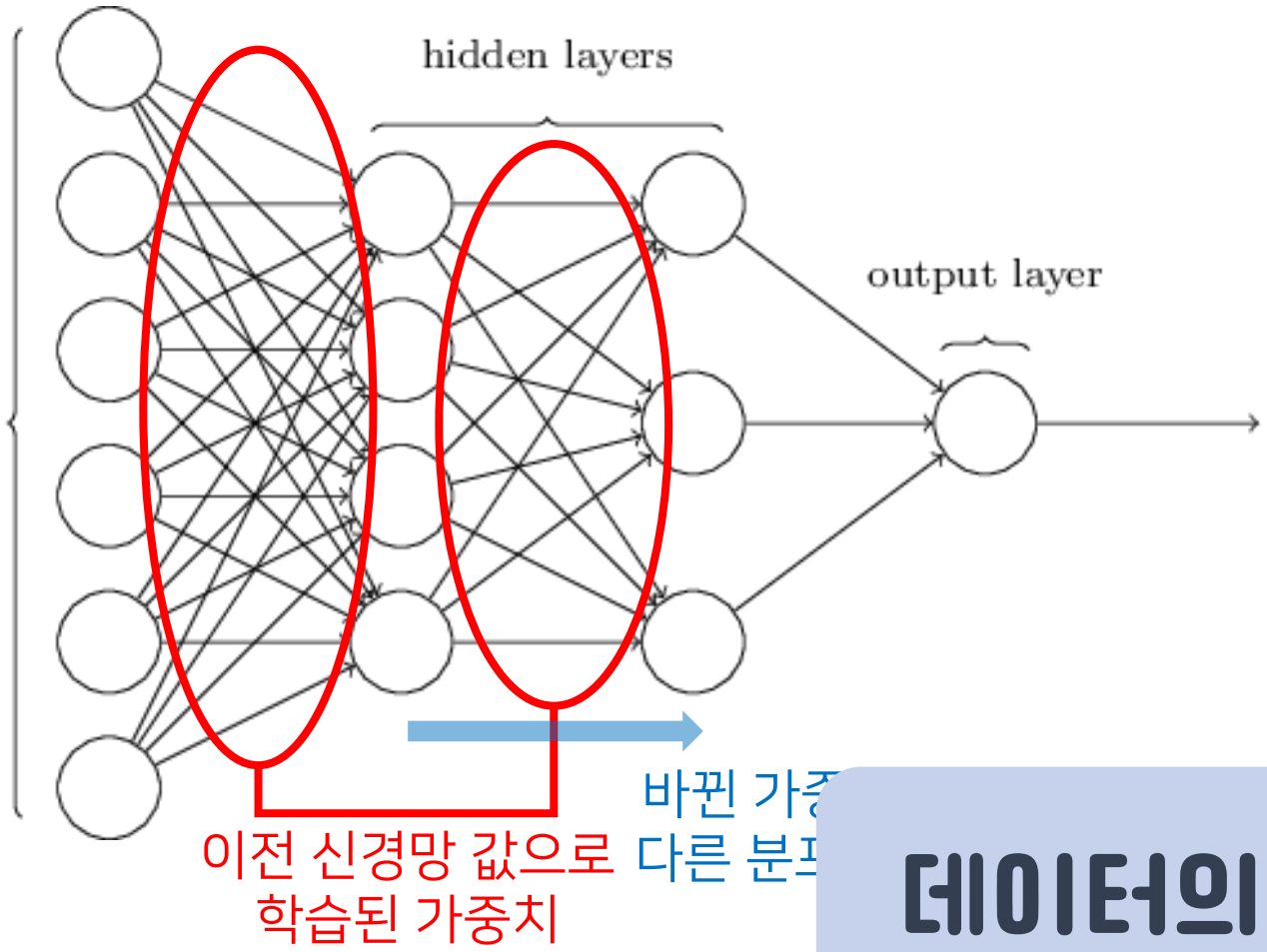
- n_{in}: 이전 층의 뉴런의 개수
- 다음 층의 뉴런 개수는 반영하지 않는다.
- 옆의 σ가 표준편차인 정규 분포를 사용하는 것이 He Initialization이다.



- Xavier와 달리 다음 층의 뉴런 개수는 반영하지 않아서 Xavier보다 더 넓은 범위에서 초깃값을 설정한다.
- 활성화 함수가 ReLU일 때 효과가 좋다.

배치 정규화

Internal covariance shift



- 가중치는 이전 신경망의 값을 기반으로 학습된다.
- 다음 순전파 과정에서는 가중치가 변경되었으므로 이전 신경망과 다른 값이 전달된다.
- 이렇게 학습했던 시점의 데이터와 현재 전달되는 데이터의 분포가 다른 것을 internal covariance shift라고 한다.

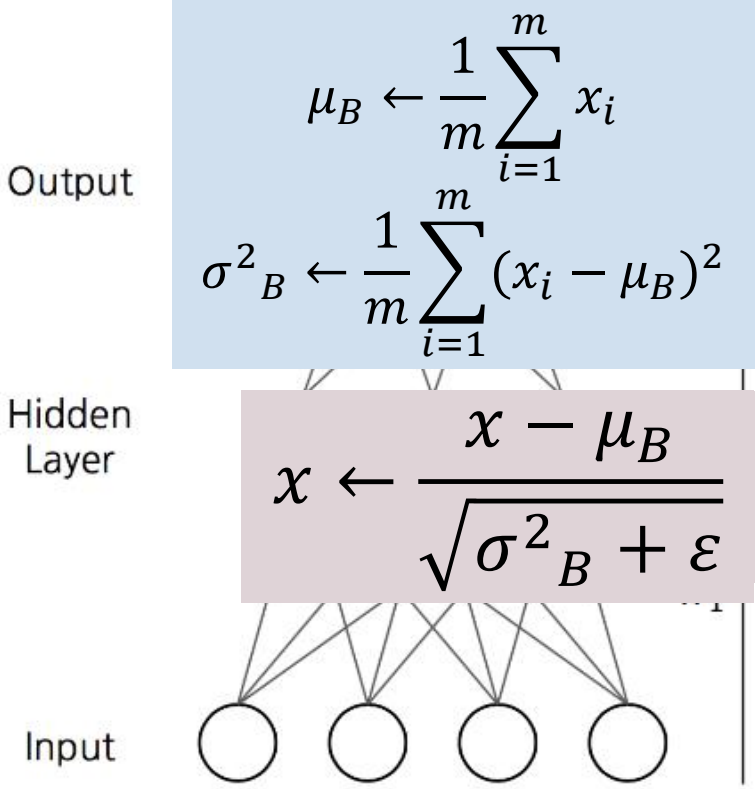
데이터의 분포는 일정한 것이 좋다!

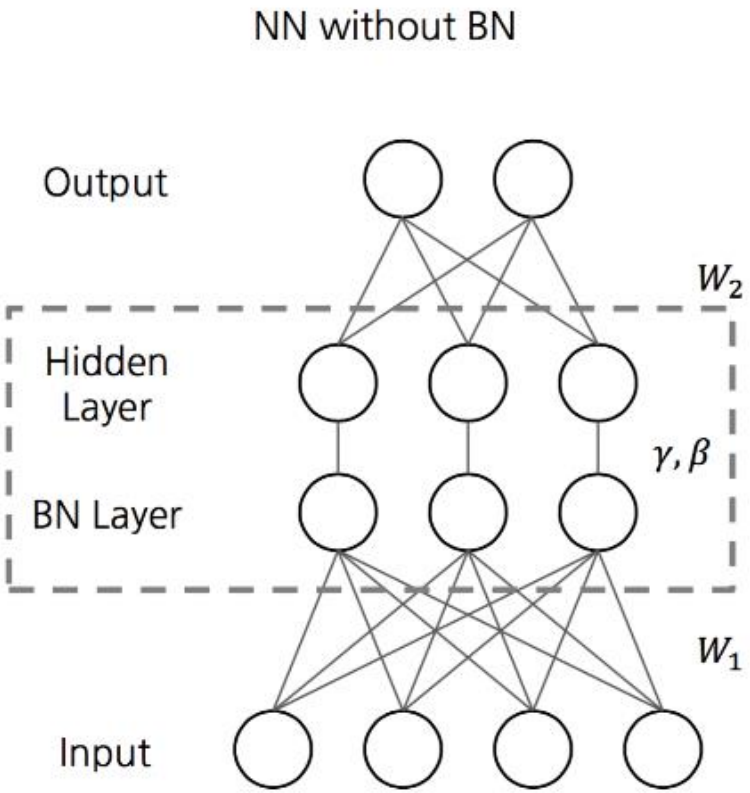
배치 정규화

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

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

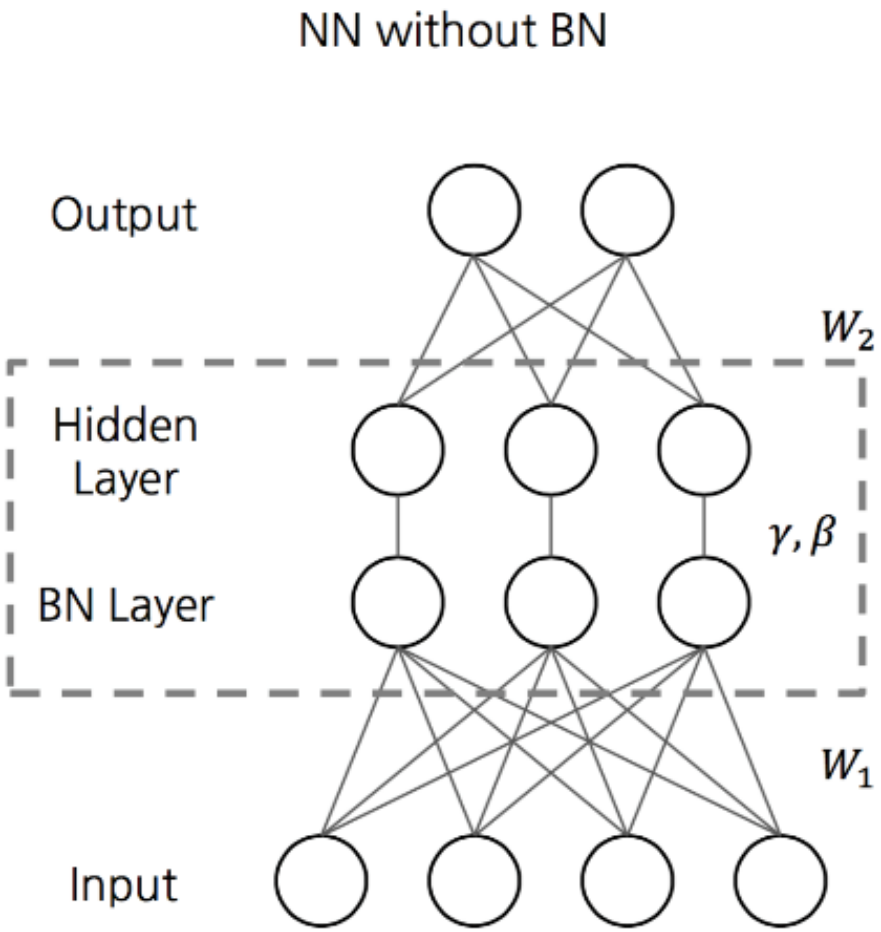
$$x \leftarrow \frac{x - \mu_B}{\sqrt{\sigma^2{}_B + \varepsilon}}$$





- 배치 단위로 값을 정규화하는 것을 배치 정규화라고 한다.
- 정규화 이후에는 확대와 이동 변환을 수행하여 음수 값이 없도록 한다.
- 배치 정규화를 거치면 활성화 값이 항상 일정한 분포를 가지게 된다.

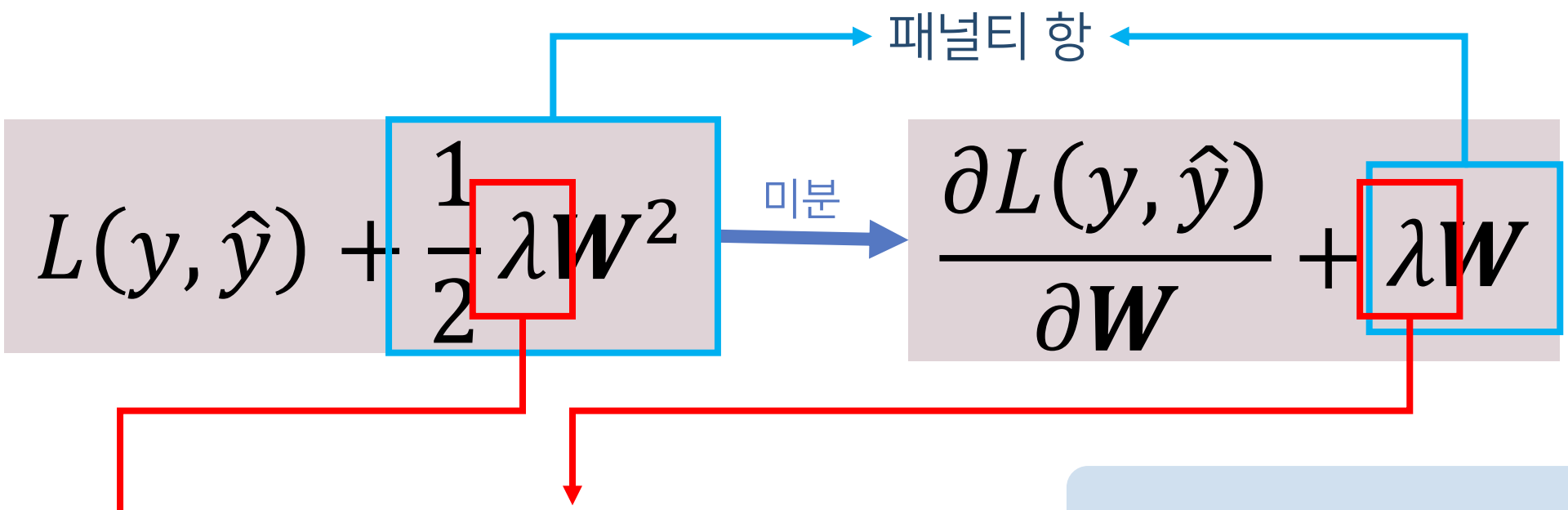
배치 정규화의 장점



- 활성화 함수 값이 0으로 수렴하더라도 정규화를 통해 기울기 소실 문제를 해결할 수 있다.
- 가중치 초깃값 문제를 어느정도 해결한다.
- 기울기의 크기나 초깃값에 덜 의존하기 때문에 기존보다 학습률을 높게 만들어도 문제가 덜 발생한다.
- 학습률을 높일 수 있으므로 학습 속도가 개선된다.
- 과적합 문제를 어느정도 해결한다.

과적합 문제

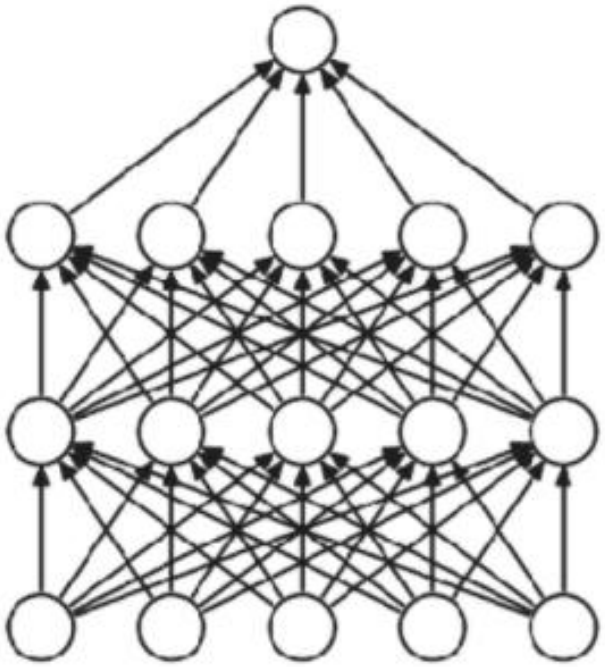
가중치 규제



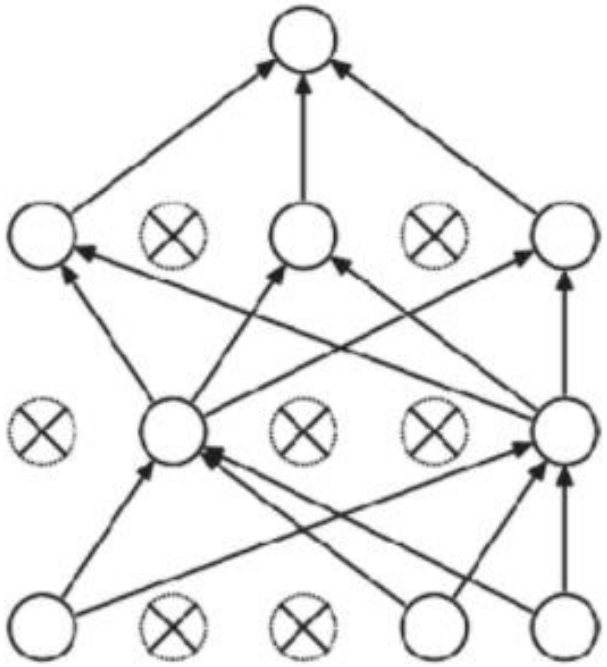
가중치 크기를 얼마나 억제할 것인지 조절하는 파라미터. 0.01, 0.001 정도의 값을 사용하나 모델마다 가변적

- 가중치의 크기를 일정 정도 제한하여 과적합을 막는 방법
- loss function에 패널티 항으로 가중치의 norm을 더함

Dropout



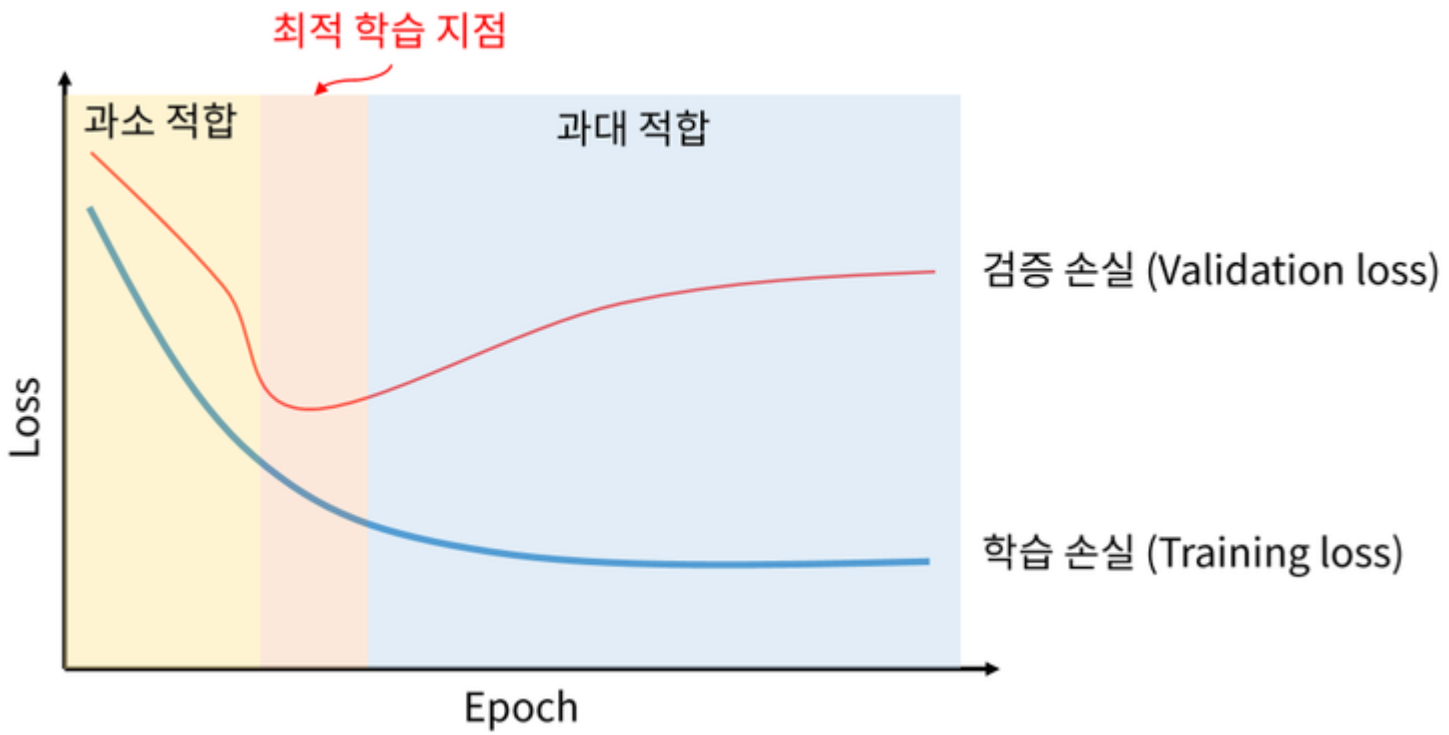
(a) 일반 신경망



(b) 드롭아웃을 적용한 신경망

- 다음 층으로 데이터를 전달할 때 랜덤으로 노드를 삭제하는 방법
- 매 층마다 다른 모델로 학습시킨 것 같은 효과를 갖기 때문에 앙상블 모델처럼 작용
- 과적합을 방지
- training에서만 적용하고 test에서는 적용하지 않음

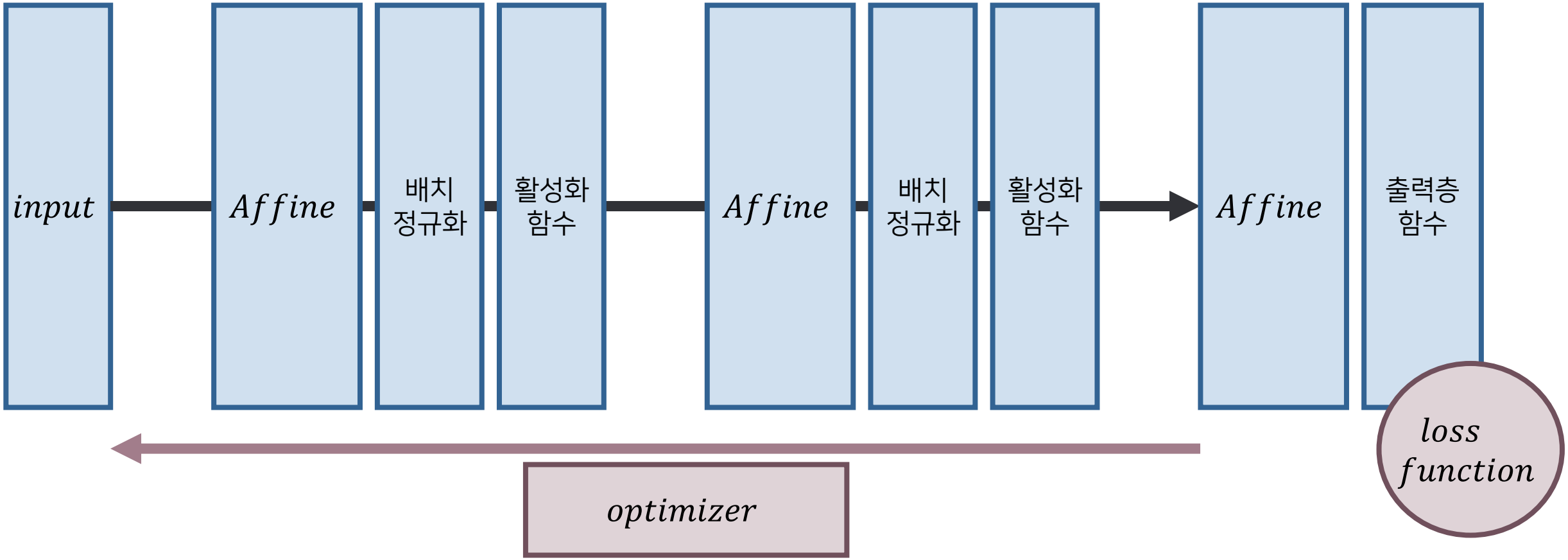
early stopping



- validation loss가 일정 반복 학습동안 더 이상 줄어들지 않을 경우, 주어진 반복 횟수를 채우기 전에 먼저 학습을 멈추는 방법
- 필요없는 반복을 하지 않기 때문에 학습 속도를 빠르게 만듦

인공신경망의 구조

인공신경망 구조



수고하셨습니다!
다음 주에 새로운 내용으로 만나요~