딥러닝 스터디 2주차

학습관련 기술들 Part1, 2

박소현

매개변수 업데이트

매개변수 업데이트

신경망을 학습하는 이유?

손실 함수의 값을 가장 작게 하는 매개변수를 찾는 것 = 최적화

어떻게 최적화된 매개변수를 찾을까?

매개변수의 기울기를 이용해 기울어진 방향으로 매개변수를 업데이트

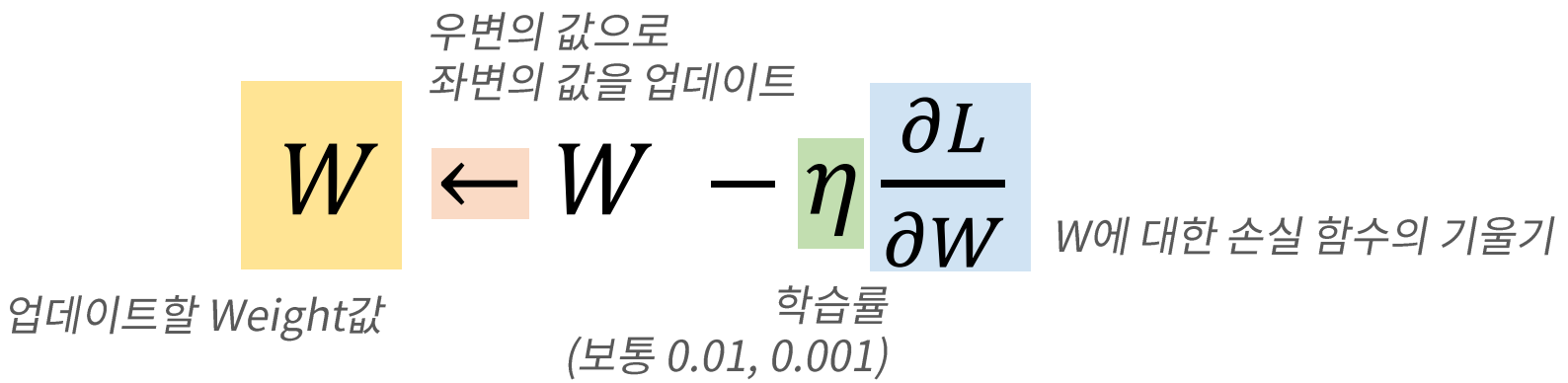
매개변수 업데이트

과정을 다시 요약하자면?

1. 매개변수의 기울기(=미분)을 구하기
2. 기울어진 방향으로 매개변수를 업데이트
3. 이를 반복하면서 최적의 매개변수 값을 찾기

방법1 : 확률적 경사하강법 - 정의

확률적 경사하강법(SGD, Stochastic Gradient Descent)는 **기울어진 방향으로 일정거리를 가는 단순한 방법**이다.



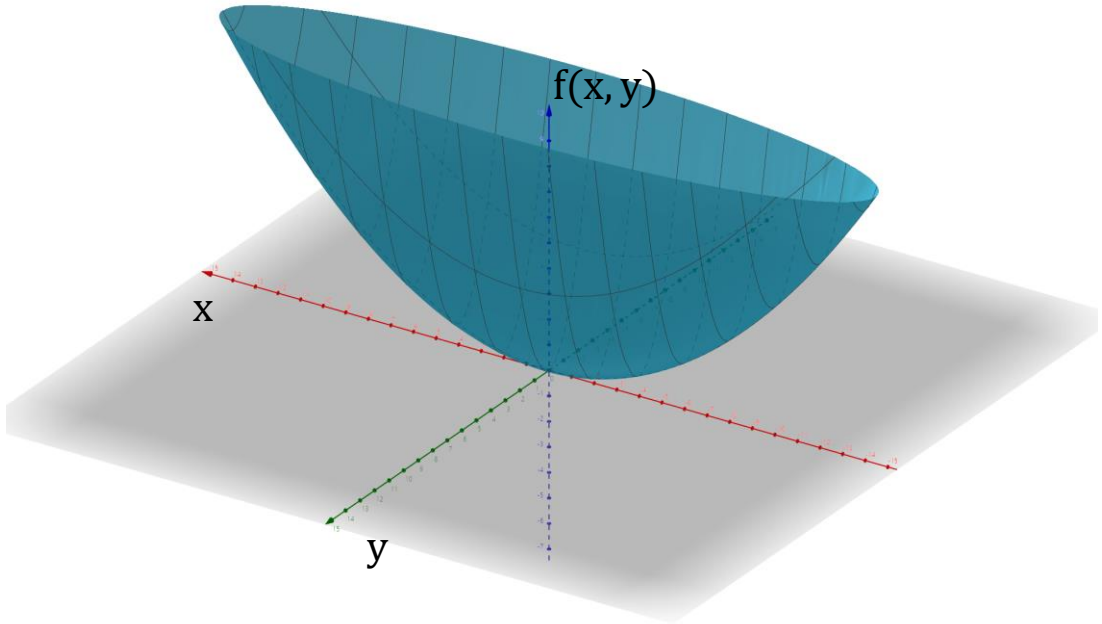
방법1 : 확률적 경사하강법 - 정의

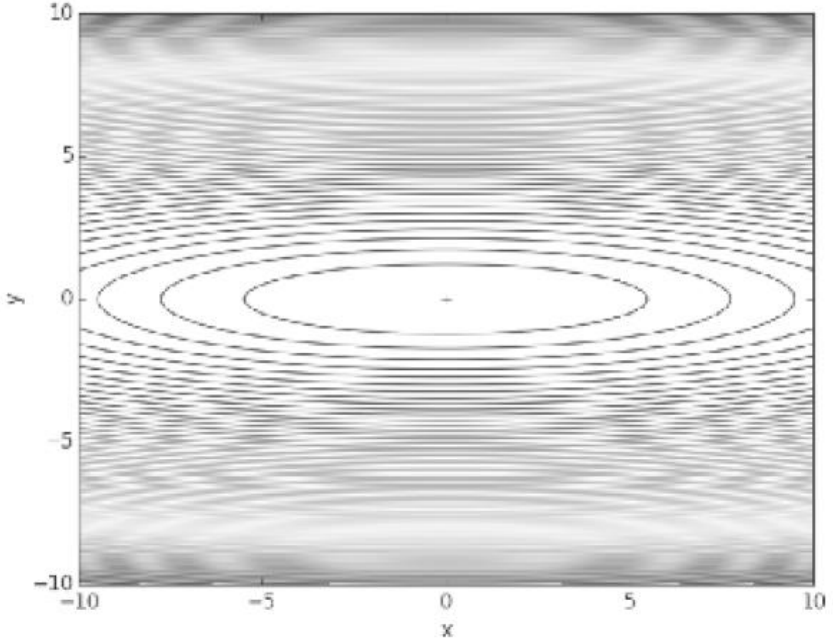
확률적 경사하강법(SGD, Stochastic Gradient Descent)는 **기울어진 방향으로 일정거리를 가는 단순한 방법**이다.

```
class SGD:
    def __init__(self, lr=0.01):
        self.lr =lr

    def update(self, params, grads):
        for key in params.key():
            params[key] -= self.lr * grads[key]
```

방법1 : 확률적 경사하강법 - 단점

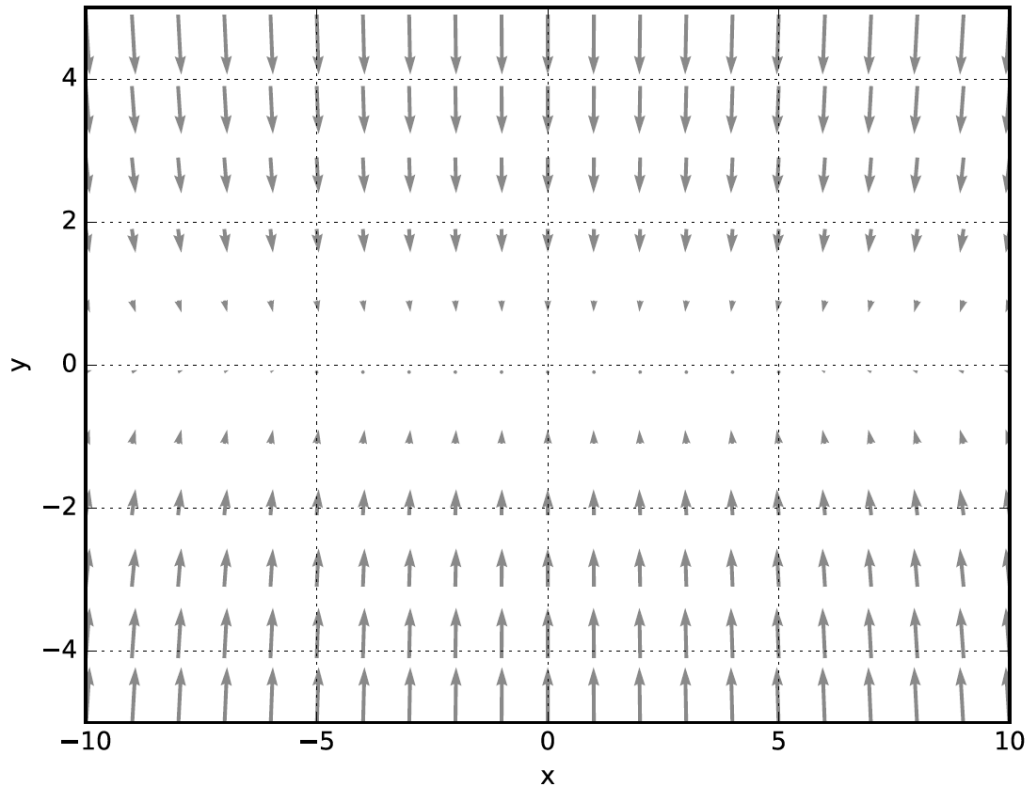
밥그릇을 X축 방향으로 늘인 듯한 함수 $f(x, y) = \frac{1}{20}x^2 + y^2$





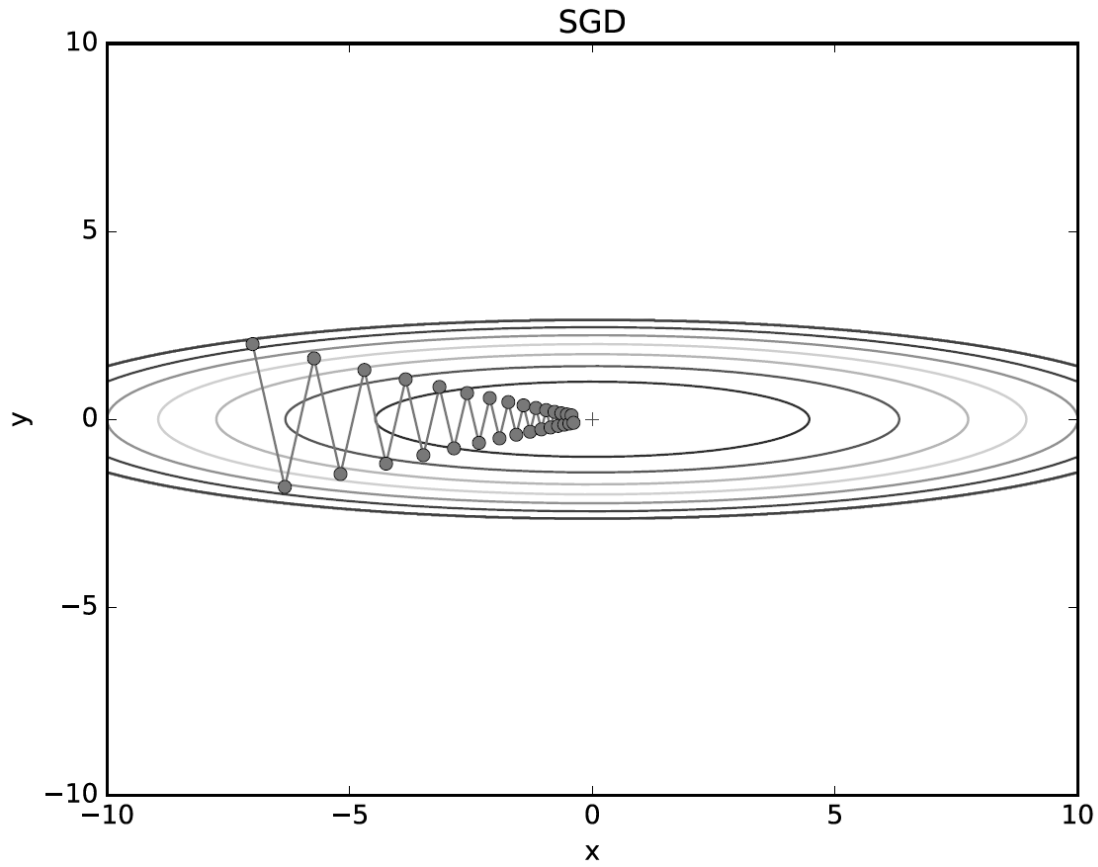
방법1 : 확률적 경사하강법 - 단점

$f(x, y) = \frac{1}{20}x^2 + y^2$의 기울기를 보면 $(0, 0)$을 가리키고 있지 않음



방법1 : 확률적 경사하강법 - 단점

이 때 SGD 방법을 적용하면 많이 요동을 치면서 (0, 0)에 다다르게 됨



방법2 : 모멘텀 - 정의

모멘텀(Momentum, 운동량)은 **기울어진 방향으로 힘을 받아 점점 하강하는 물리법칙을 이용한 방법**이다. 물리에서 **운동량 = 속도 * 질량**이다.

v는 속도로 기울기 방향으로 힘을 받아 가속됨
α는 질량 (하이퍼파라미터로 주로 0.9로 설정)

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

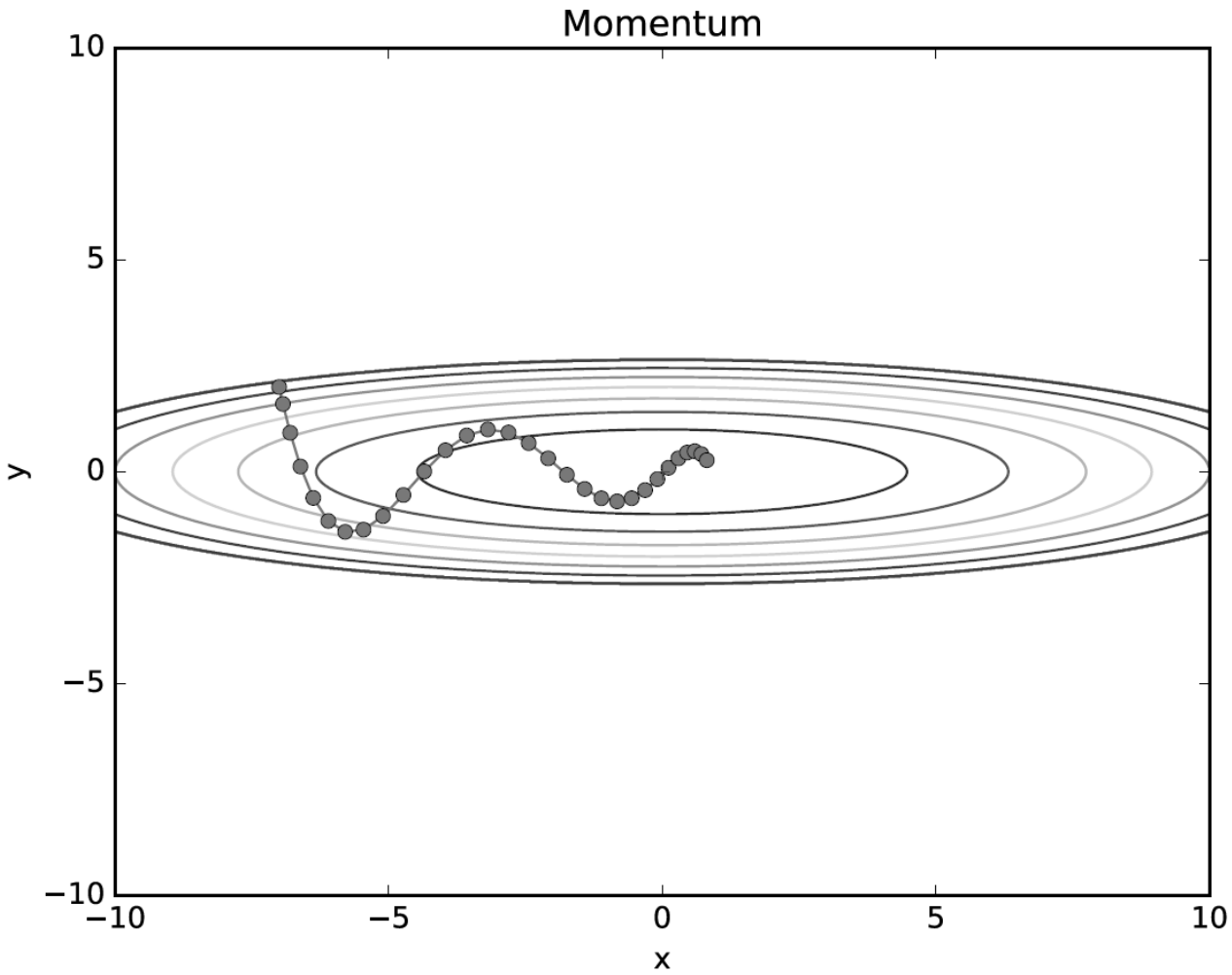
$$W \leftarrow W + v$$

방법2 : 모멘텀 - 정의

모멘텀(Momentum, 운동량)은 **기울어진 방향으로 힘을 받아 점점 하강하는 물리법칙을 이용한 방법**이다. 물리에서 **운동량 = 속도 * 질량**이다.

```
class Momentum:
    def __init__(self, lr=0.01, momentum=0.9):
        self.lr = lr                # 학습률
        self.momentum = momentum    # 운동량
        self.v = None               # 속도

    def update(self, params, grads):
        if self.v is None:
            self.v = {}
            for key, val in params.items():
                self.v[key] = np.zeros_like(val)

        for key in params.keys():
            self.v[key] = self.momentum * self.v[key] - self.lr * grads[key]
            params[key] += self.v[key]
```

방법2 : 모멘텀 - 적용



방법3 : AdaGrad - 정의

AdaGrad는 **각각의 매개변수에 맞게 학습률을 조정하는 방식**이다.

Element-wise 곱셈

기존 그리고 업데이트된 기울기의 제곱을 계속 더한 값

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

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

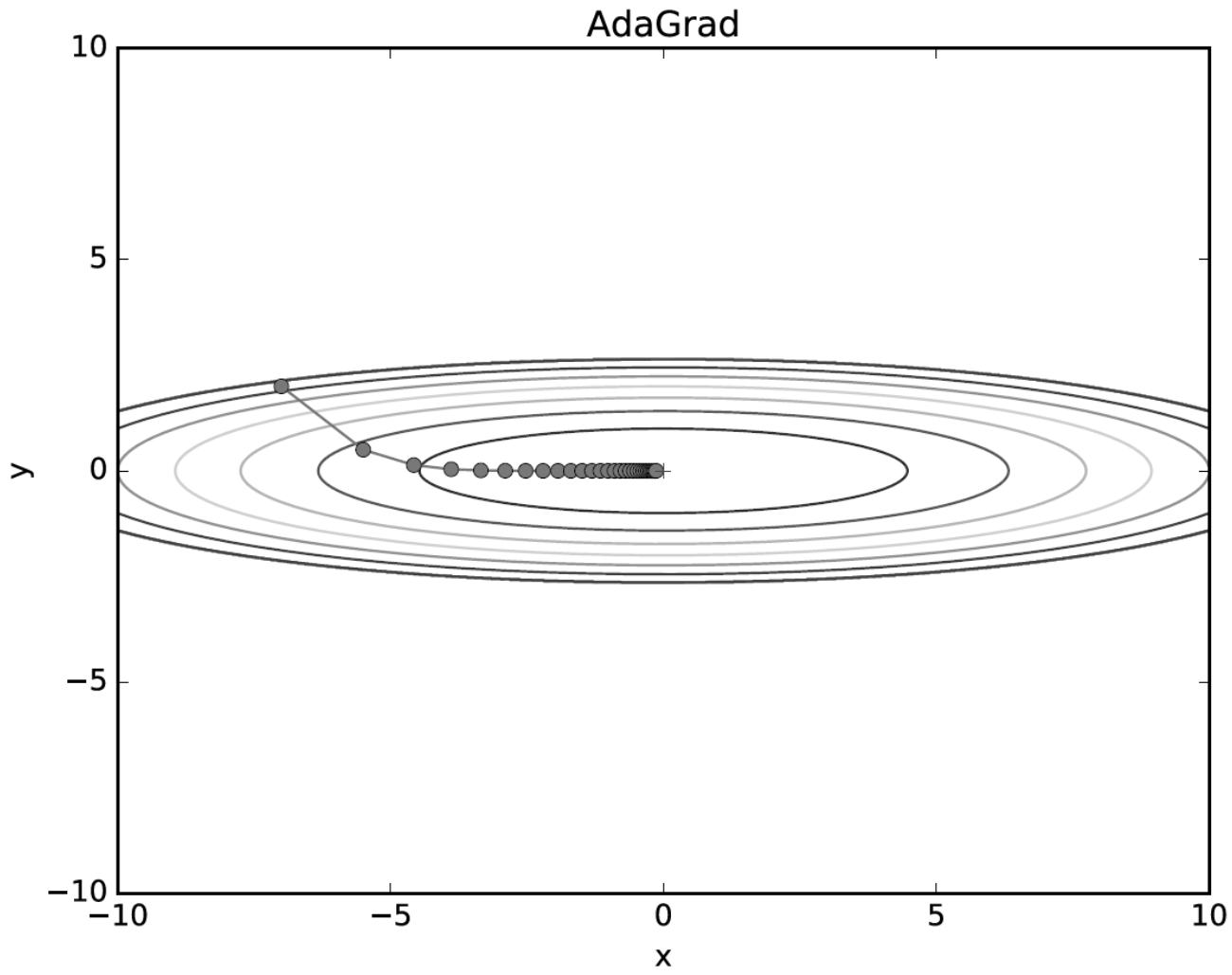
η에 $\frac{1}{\sqrt{h}}$을 곱해 학습율을 감소시킴

방법3 : AdaGrad - 정의

AdaGrad는 **각각의 매개변수에 맞게 학습률을 조정하는 방식**이다.

```python
class AdaGrad:
    def __init__(self, lr=0.01):
        self.lr = lr     # 학습률
        self.h = None    # 학습률 감소 정도

    def update(self, params, grads):
        if self.h is None:
            self.h = {}
            for key, val in params.items():
                self.h[key] = np.zeros_like(val)

        for key in params.keys():
            self.h[key] += grads[key] * grads[key]
            # 0을 나누는 것을 막기 위해 1e-7을 더해줌
            params[key] -= self.lr * grads[key] / (np.sqrt(self.h[key]) + 1e-7)
```

방법3 : AdaGrad - 적용



방법4 : Adam - 정의

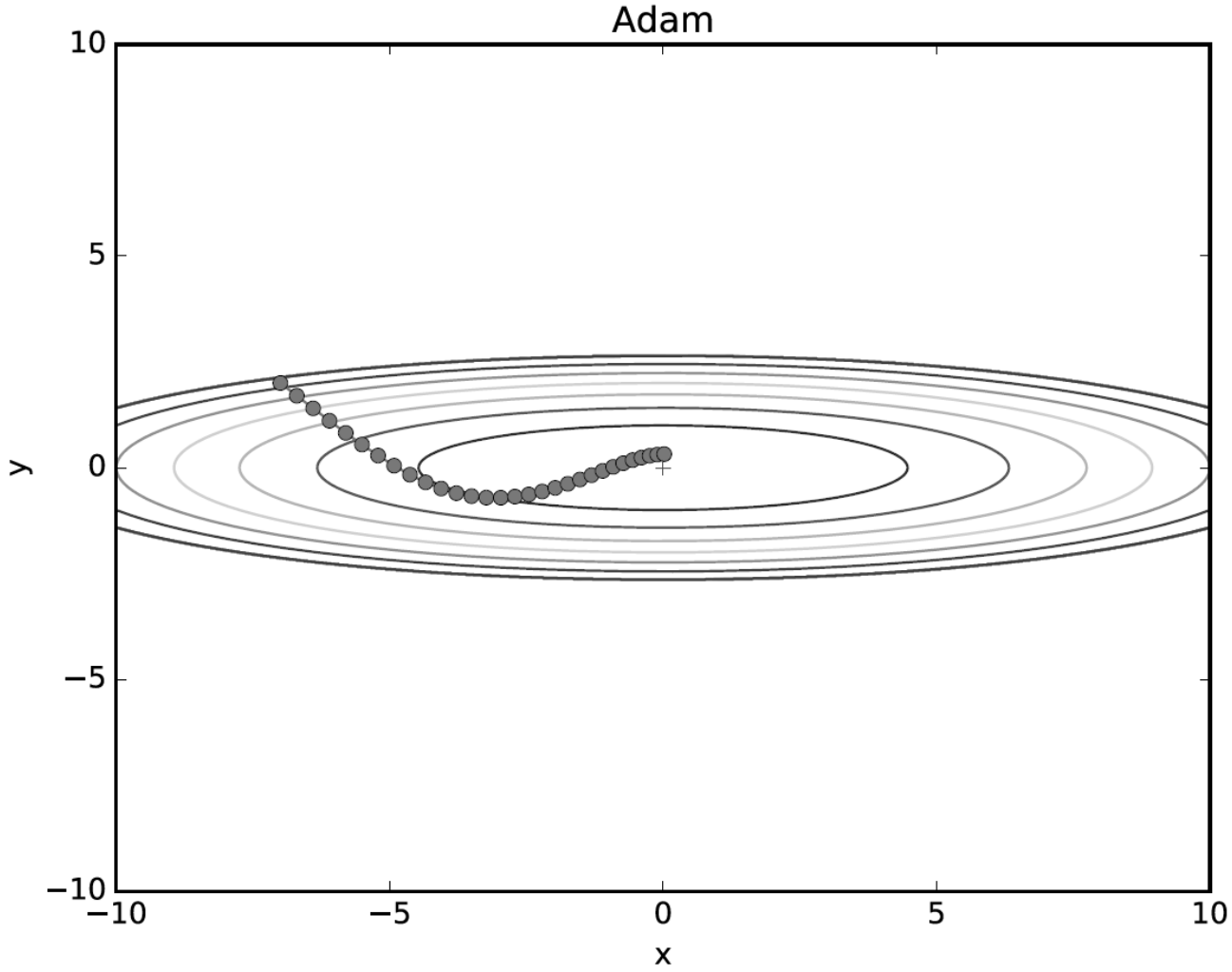
Adam은 **Momentum과 AdaGrad 방법을 합친 방법**이다.

- Momentum보다는 흔들림이 적고
- AdaGrad처럼 학습률을 각 매개변수에 맞게 조정

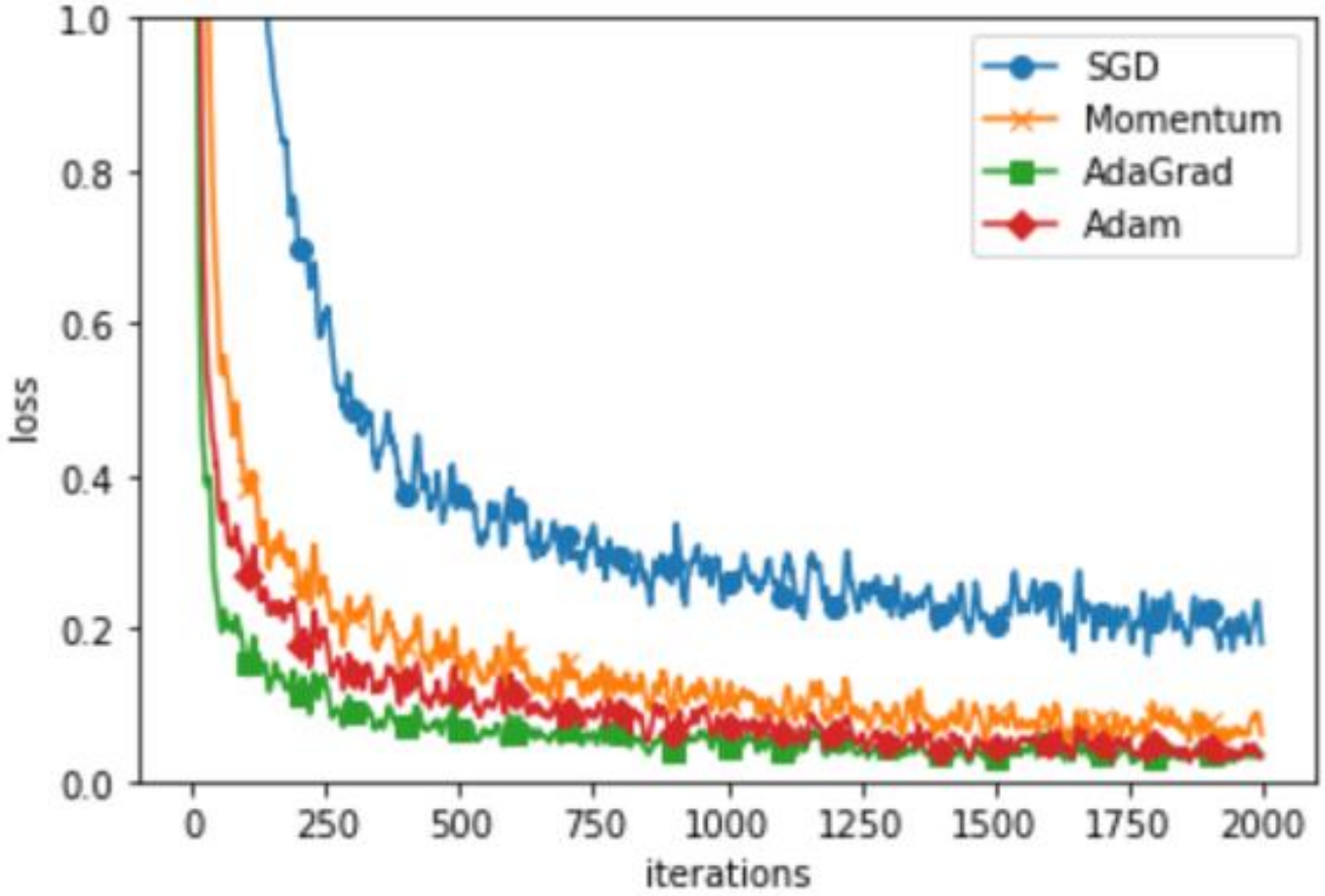
방법4 : Adam - 정의

```
class Adam:
    def __init__(self, lr=0.001, beta1=0.9, beta2=0.999):
        self.lr = lr            # 학습률
        self.beta1 = beta1      # 1차 모멘텀용 계수 (보통 0.9)
        self.beta2 = beta2      # 2차 모멘텀용 계수 (보통 0.999)
        self.iter = 0
        self.m = None           # 1차 모멘텀
        self.v = None           # 2차 모멘텀

    def update(self, params, grads):
        if self.m is None:
            self.m, self.v = {}, {}
            for key, val in params.items():
                self.m[key] = np.zeros_like(val)
                self.v[key] = np.zeros_like(val)

        self.iter += 1
        lr_t  = self.lr * np.sqrt(1.0 - self.beta2**self.iter) / (1.0 - self.beta1**
self.iter)

        for key in params.keys():
            self.m[key] += (1 - self.beta1) * (grads[key] - self.m[key])
            self.v[key] += (1 - self.beta2) * (grads[key]**2 - self.v[key])

            params[key] -= lr_t * self.m[key] / (np.sqrt(self.v[key]) + 1e-7)
```

방법4 : Adam - 적용



각 방법을 MNIST 데이터셋에 적용했을 때

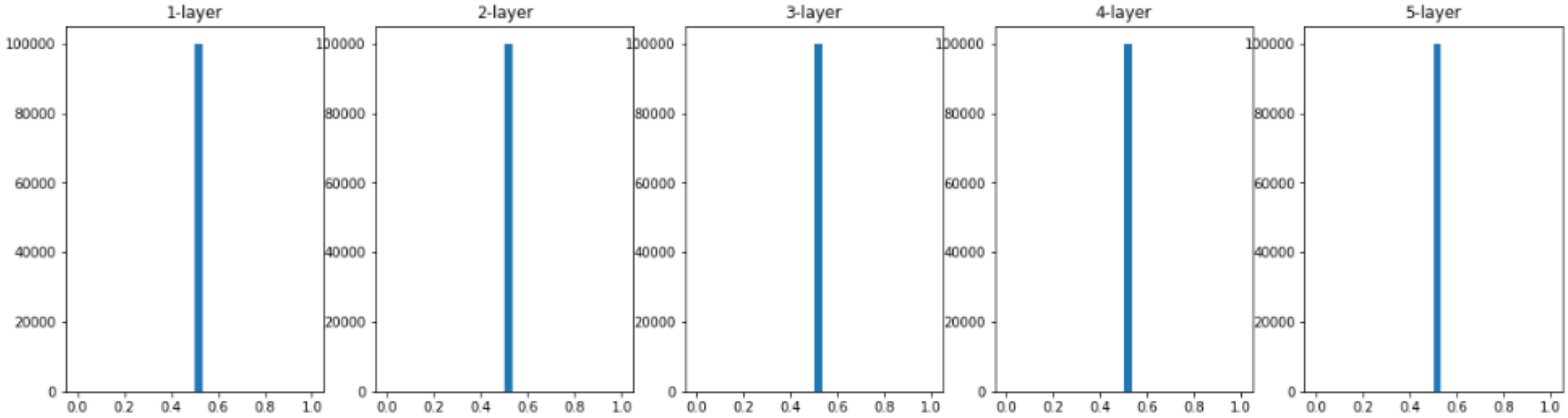


가중치의 초깃값

가중치의 초깃값

가중치 초깃값으로 0을 설정하면 어떻게 될까?

= **오차역전파법**에 의해 모든 가중치의 값이 같은 값으로 갱신



가중치의 초깃값

가중치 초깃값으로 0을 설정하면 어떻게 될까?

= **오차역전파법**에 의해 모든 가중치의 값이 같은 값으로 갱신

= 가중치가 고르게 되어버림

= 가중치를 여러 개 갖는 의미가 없어짐

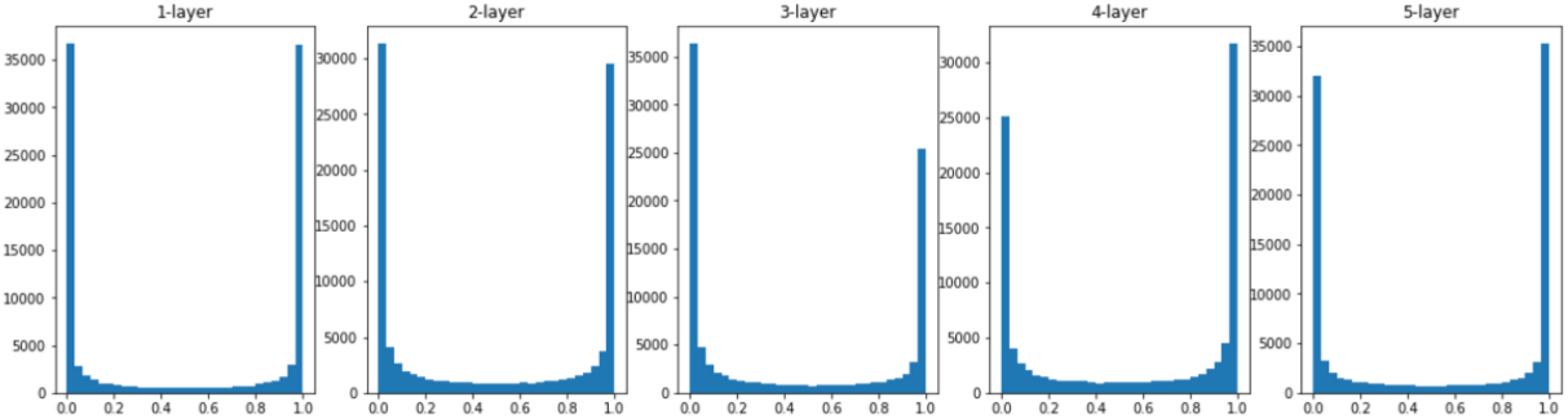
👉 **가중치가 고르게 되지 않도록 가중치의 초깃값을 임의로 설정!**

👉 **활성화값이 고르게 되도록 가중치의 초깃값을 임의로 설정!**

가중치의 초깃값

가중치 초깃값으로 표준편차가 1인 분포로 설정하면?

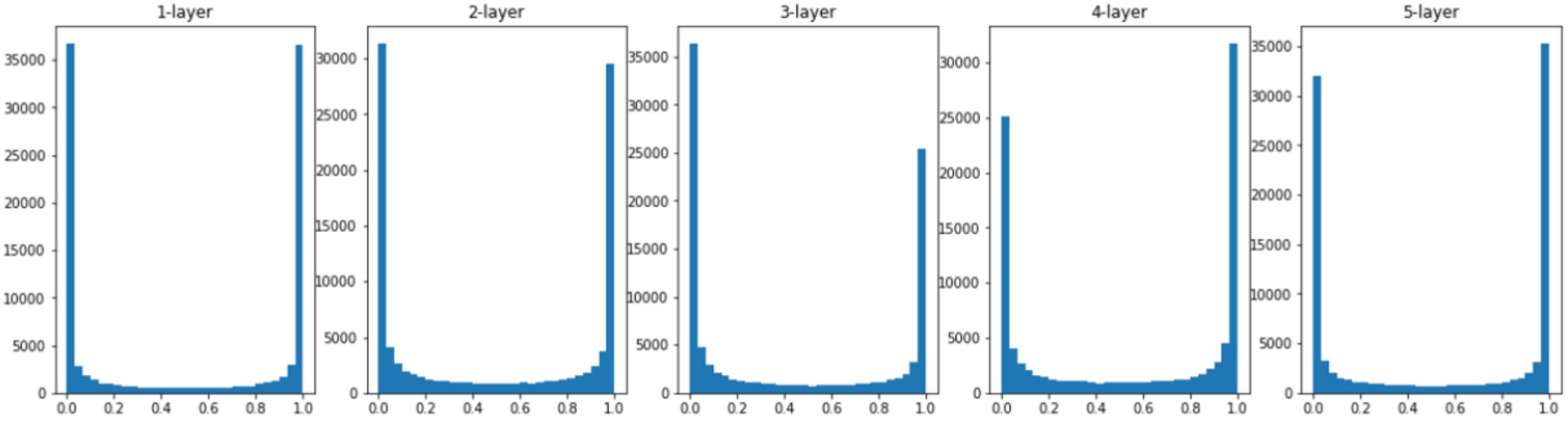
각 층의 활성화 값들이 0과 1에 치우쳐져 있음



가중치의 초깃값

가중치 초깃값으로 표준편차가 1인 분포로 설정하면?

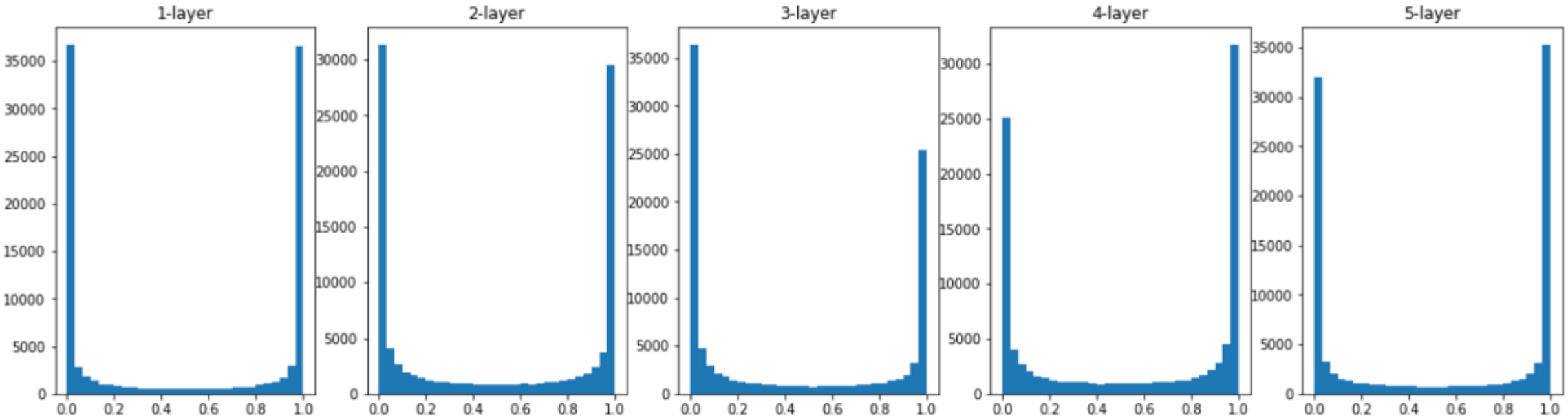
활성화 함수인 sigmoid함수는 0 또는 1에 가까워질수록 기울기가 0에 가까워짐



가중치의 초깃값

가중치 초깃값으로 표준편차가 1인 분포로 설정하면?

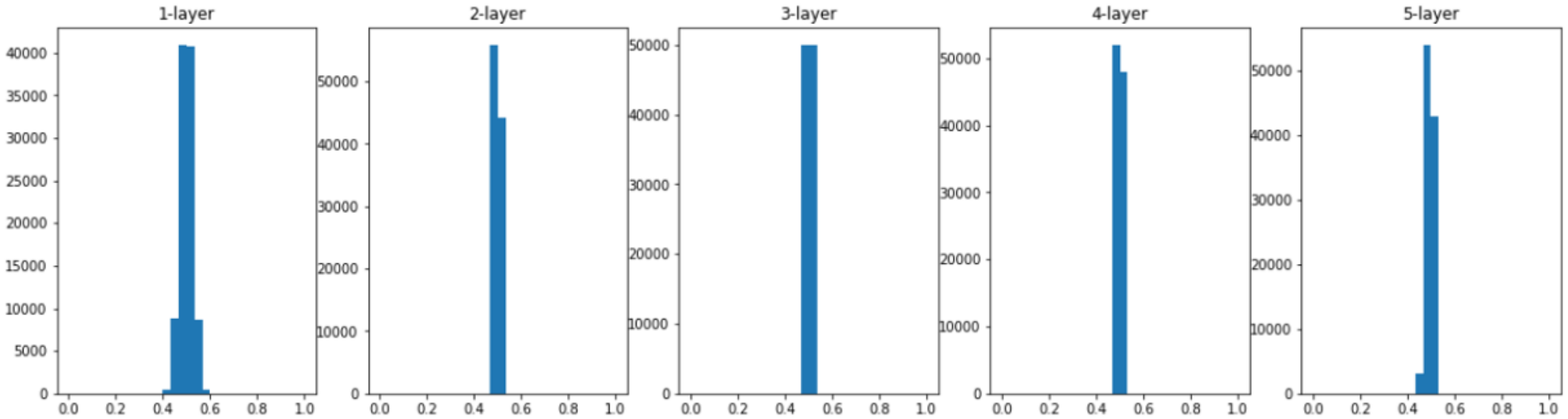
오차역전파법에 의해 기울기가 점점 작아지다 사라짐 = **기울기 소실 문제**



가중치의 초깃값

가중치 초깃값으로 표준편차가 0.01인 분포로 설정하면?

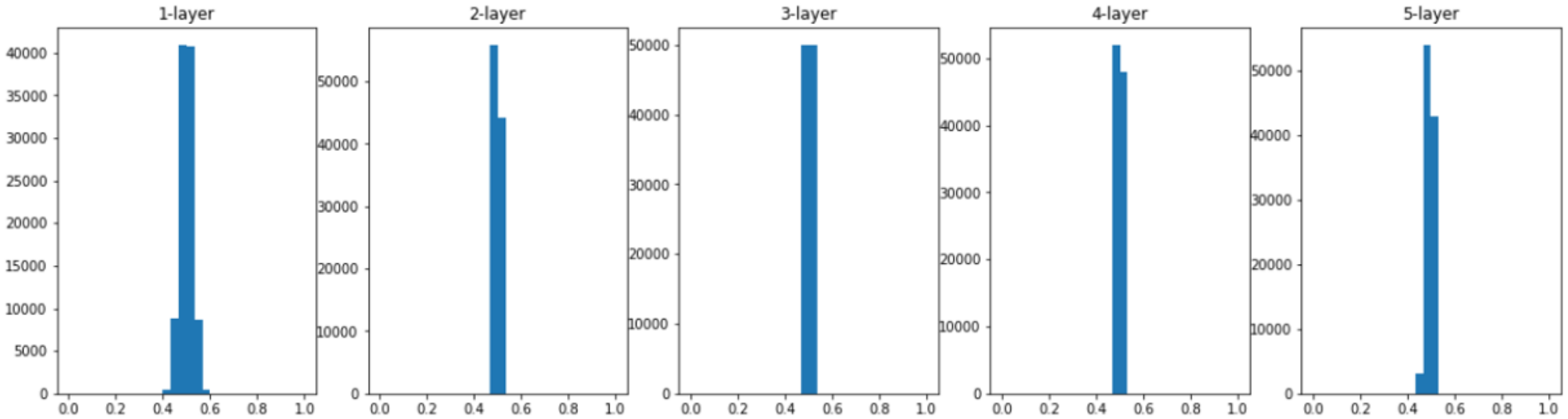
각 층의 활성화 값들이 0.5로 치우쳐져 있음



가중치의 초깃값

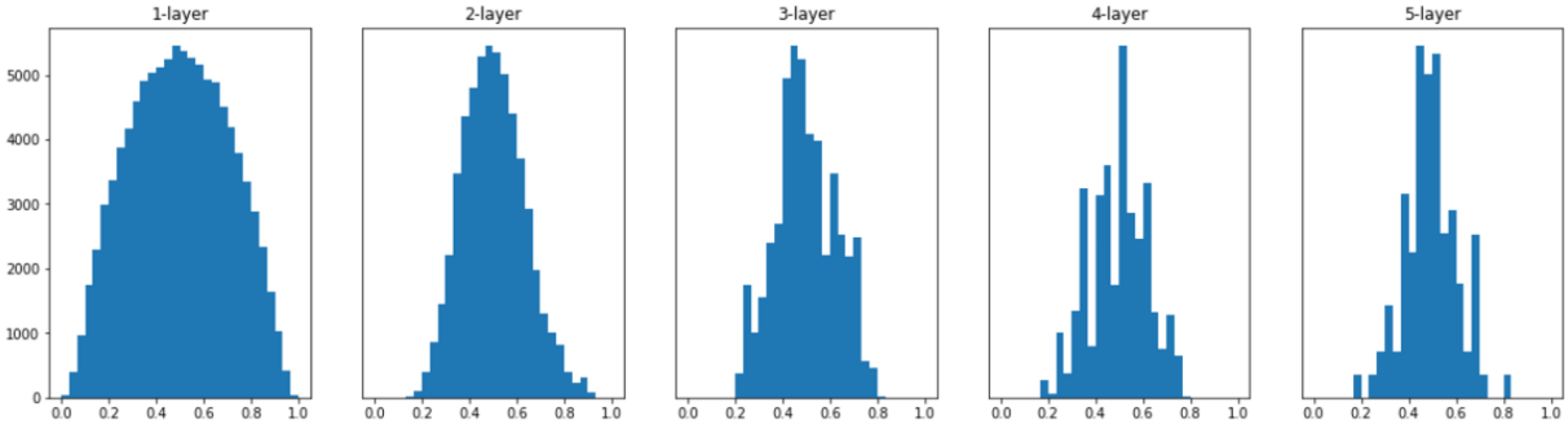
가중치 초깃값으로 표준편차가 0.01인 분포로 설정하면?

한 값으로 치우쳐져 가중치가 고르게 되는 문제가 발생



Xavier 초깃값

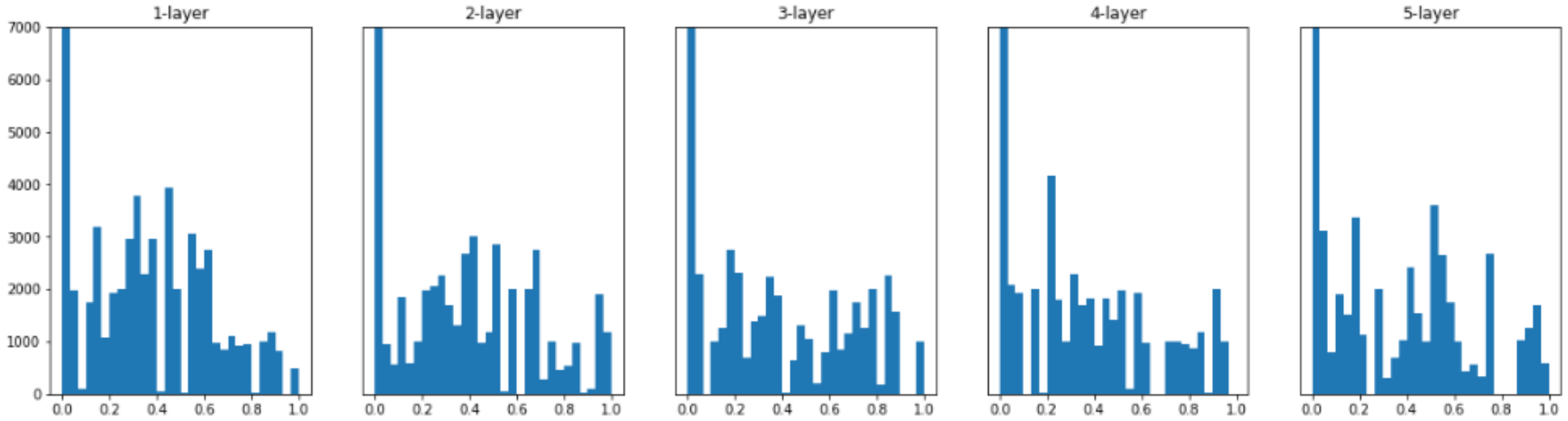
앞 계층의 노드가 n개 일 때, 표준 편차가 $\frac{1}{\sqrt{n}}$인 분포를 사용하는 방법



Xavier 초깃값 - 단점

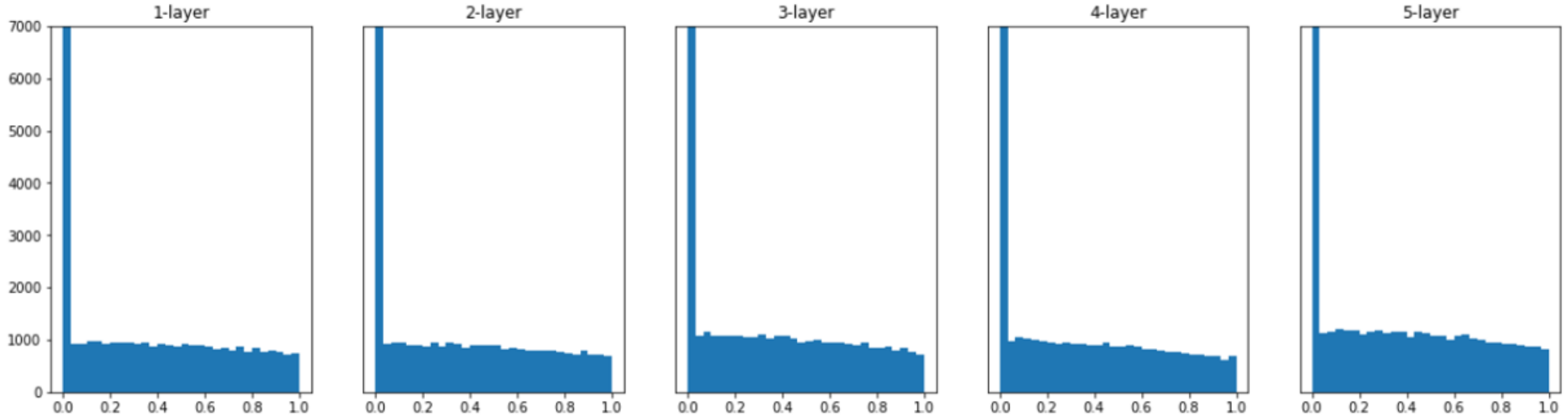
활성화 함수가 **선형 함수**인 것을 전제로 한다.

👉 이 때 활성화 함수가 비선형 함수인 ReLU를 사용하게 되면..

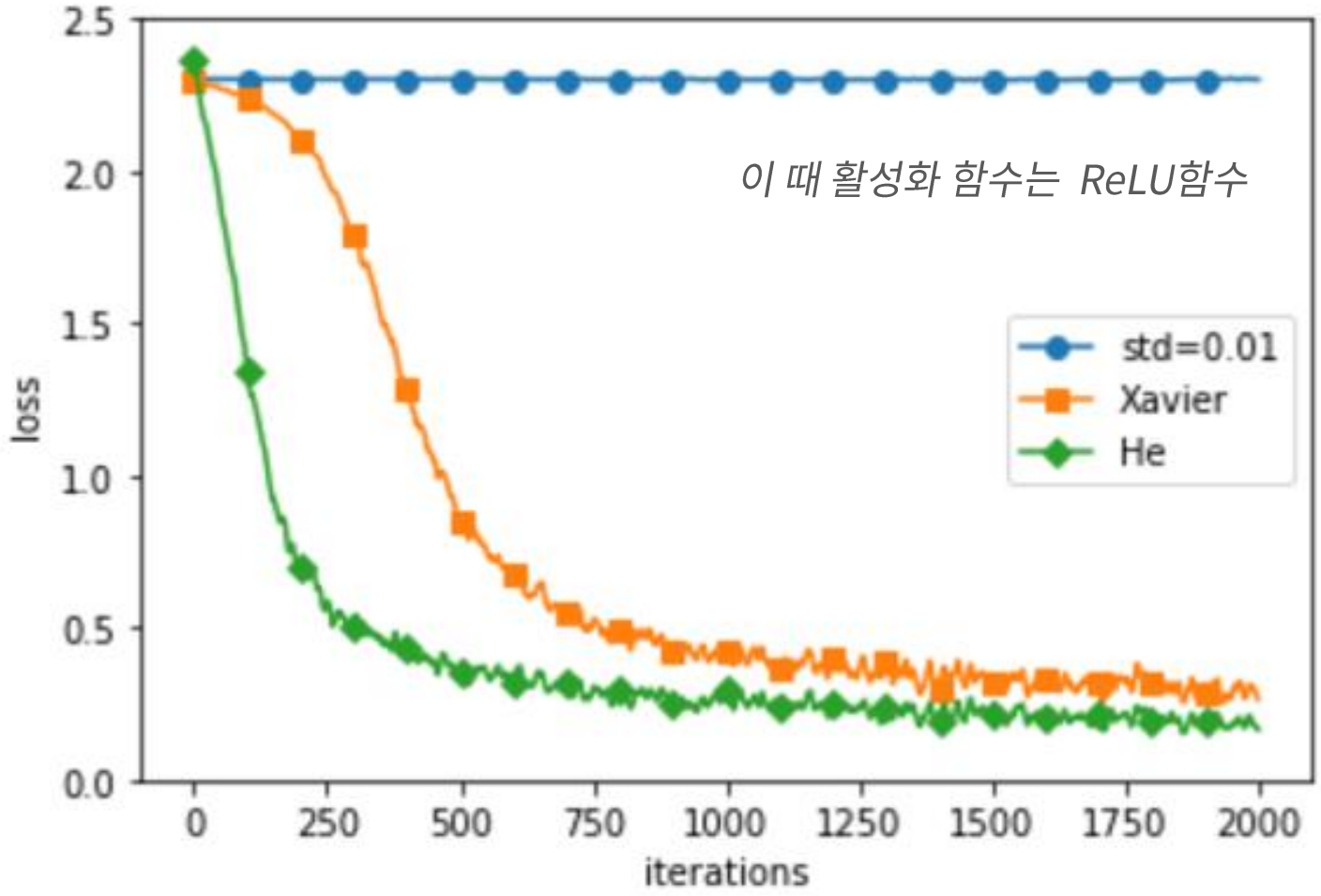


He 초깃값

앞 계층의 노드가 n개 일 때, 표준 편차가 $\sqrt{\frac{2}{n}}$인 분포를 사용하는 방법



각 방법을 MNIST 데이터셋에 적용했을 때




METROPOLIS
QnA