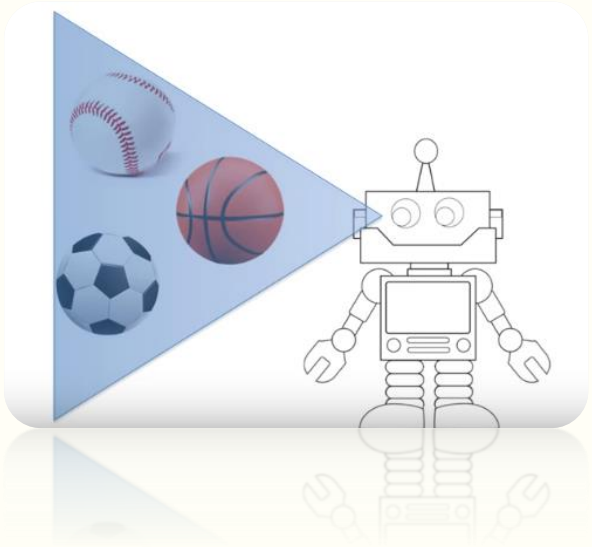
[파이토치 첫걸음]

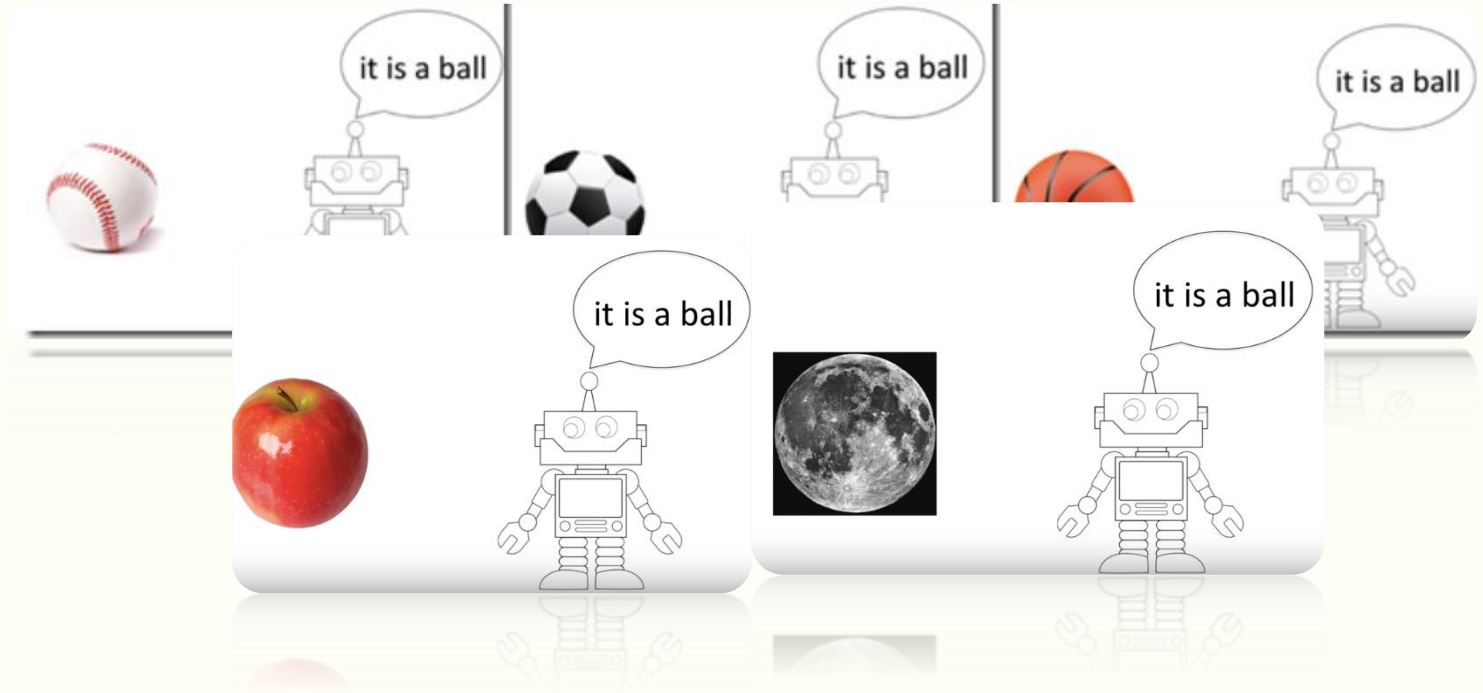
7장 : 학습 시 생길 수 있는 문제점과 해결방안

1. 오버피팅과 언더피팅

1. **동그랗게 생긴 것**은 공이야!

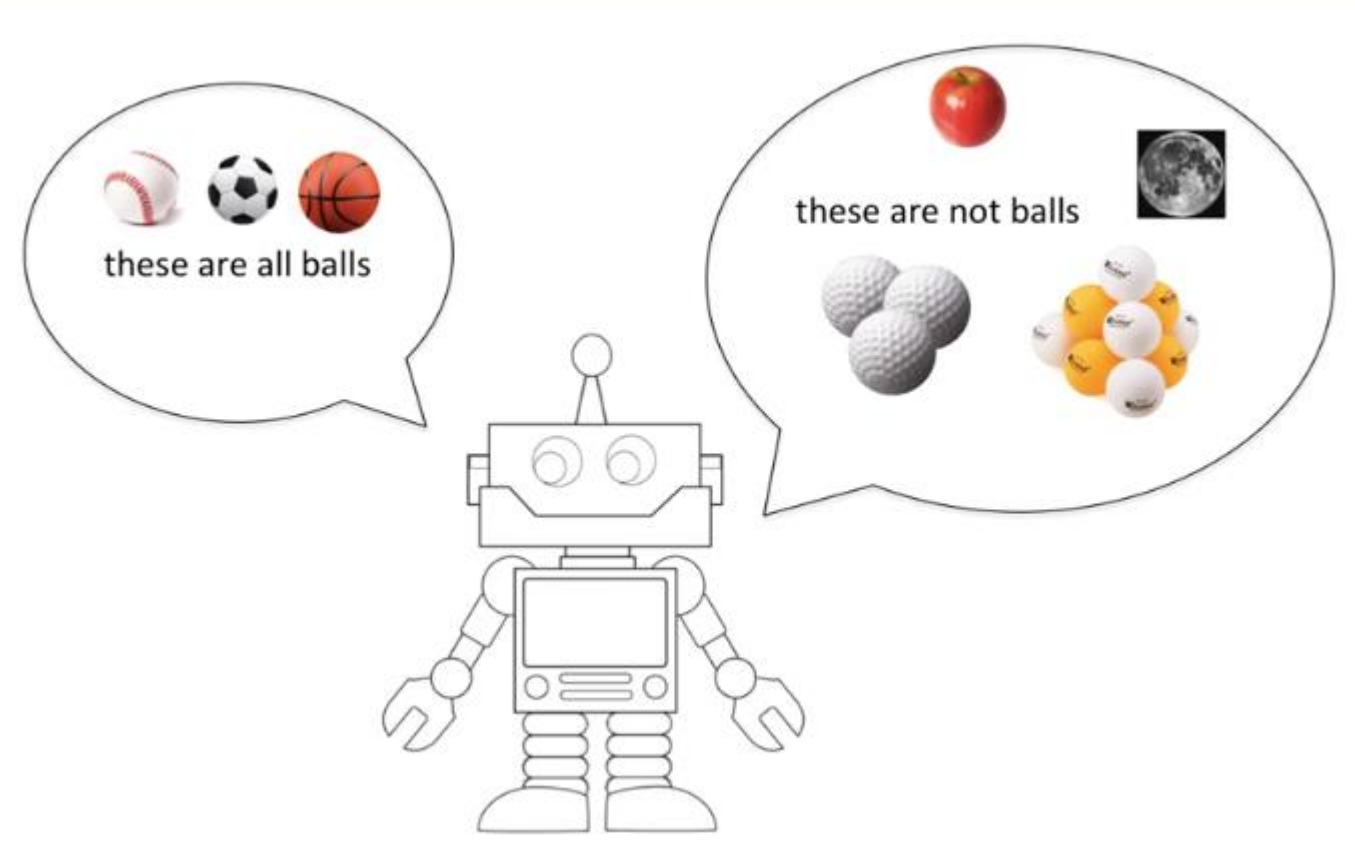


2. **동그래서 공**이라고 판단



많은 공통된 특성이 있음에도 불구하고 **한가지 특성으로만 학습**시킬 경우 **too BIAS** 하게된
모델이라고 하고, testing 단계에서 **새로운 데이터를 너무 잘 예측**해버리는 모델을 **언더피팅**이라고 함

1. 오버피팅과 언더피팅



실밥을 가지는 것, 지름이 7cm
보다 큰 것 등을 더 학습시킴

⬇

골프공, 탁구공까지 배제해버려

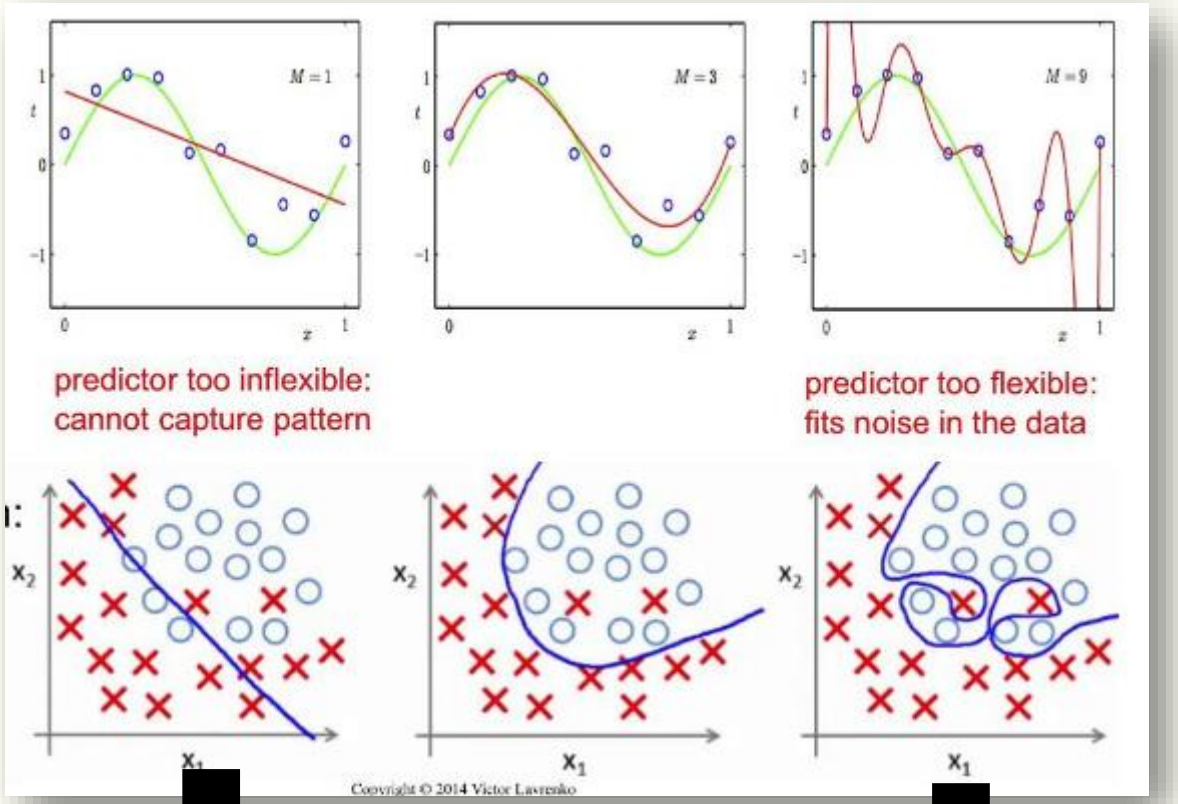
⬇

봤던 데이터를 잘 맞추는데 새로운
데이터는 잘 예측하지 못한다
= 높은 variance

⬇

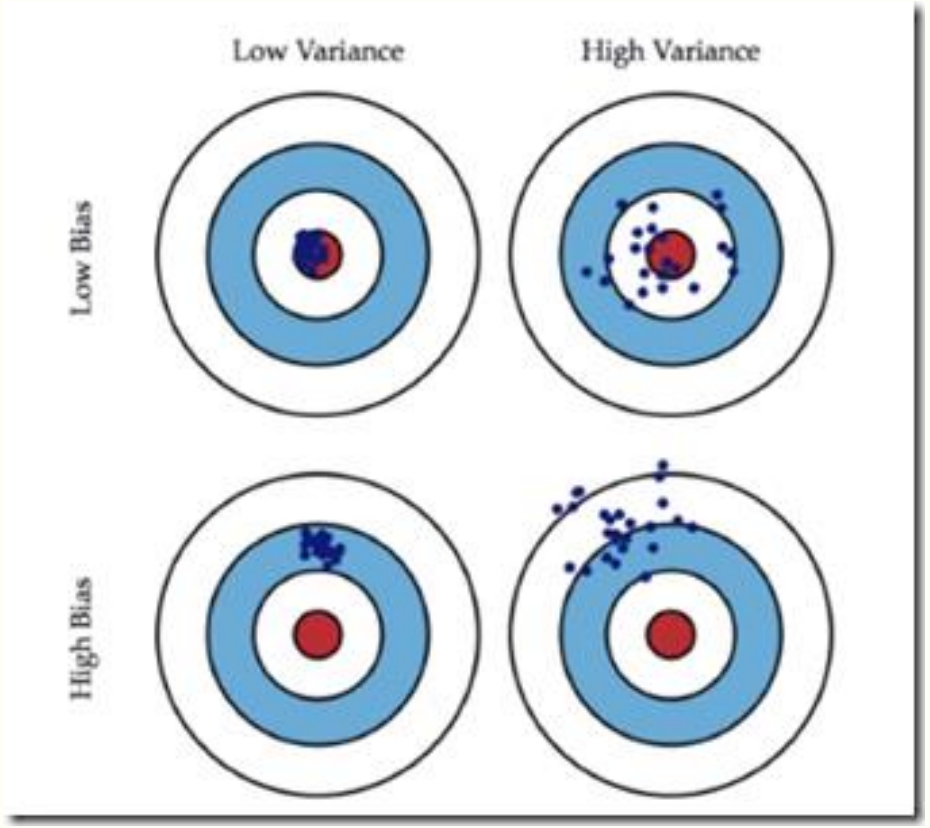
Testing 단계에서 새로운 데이터를
예측하지 못하는 오버피팅 모델

1. 오버피팅과 언더피팅



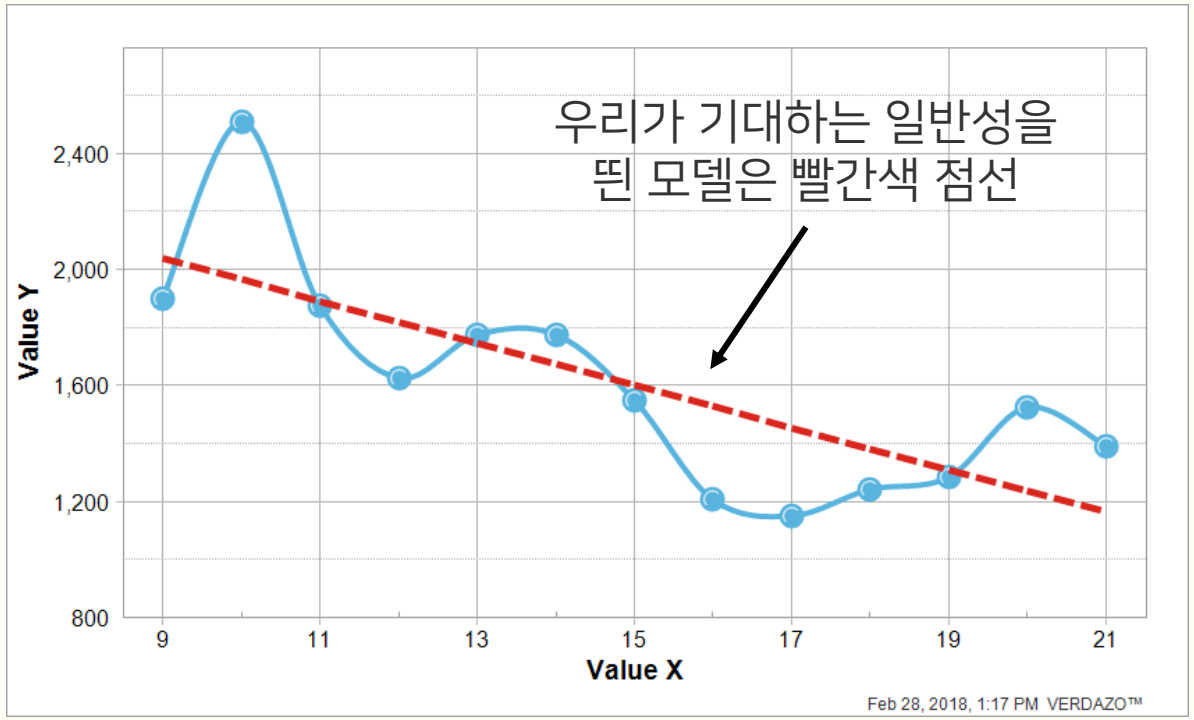
언더피팅

오버피팅



2. 정형화

1. 정형화 하는 이유 ? = **오버피팅을 막기 위해**



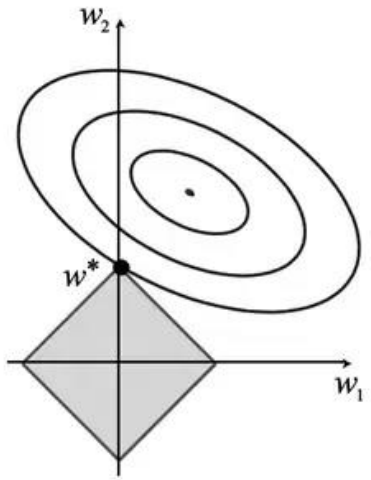
과적합하다는 것은 모델로 그래프를 그렸을 때,
너무 구불구불 하다는 것이고,
그래서 weight를 너무 큰 값을 주지 않도록 !

얼마나 중요해 ? (상수)

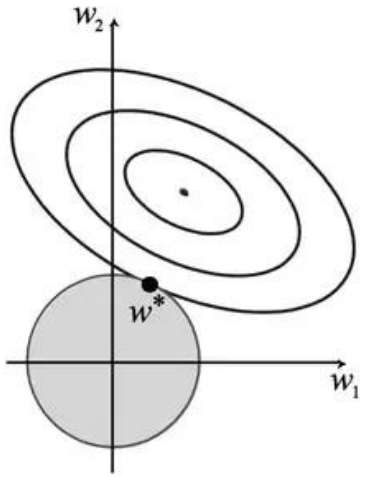
2. L1, L2 정형화 식 = 평균제곱오차 + 정형화식

$$J(\theta) = MSE(\theta) + \lambda \sum_{i=1}^{n} |\theta_i|$$

$$J(\theta) = MSE(\theta) + \lambda \sum_{i=1}^{n} \theta_i^2$$



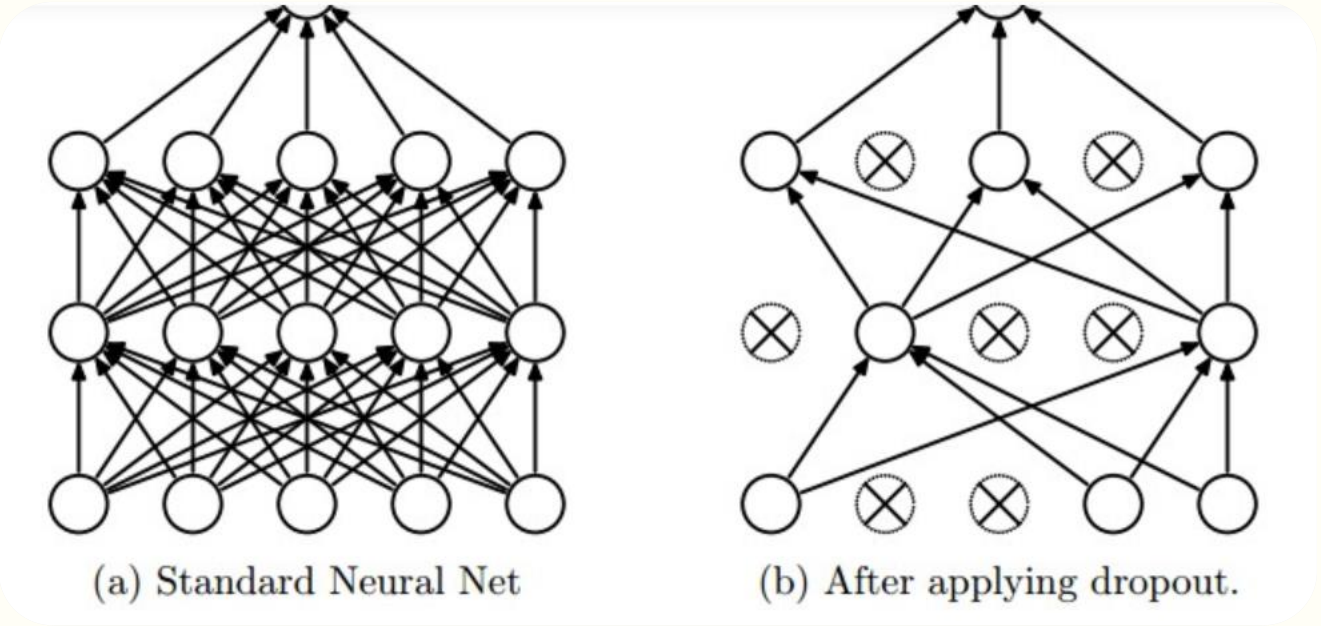
L1



L2

3. 드롭다운

정형화를 적용하는 또 다른 방법



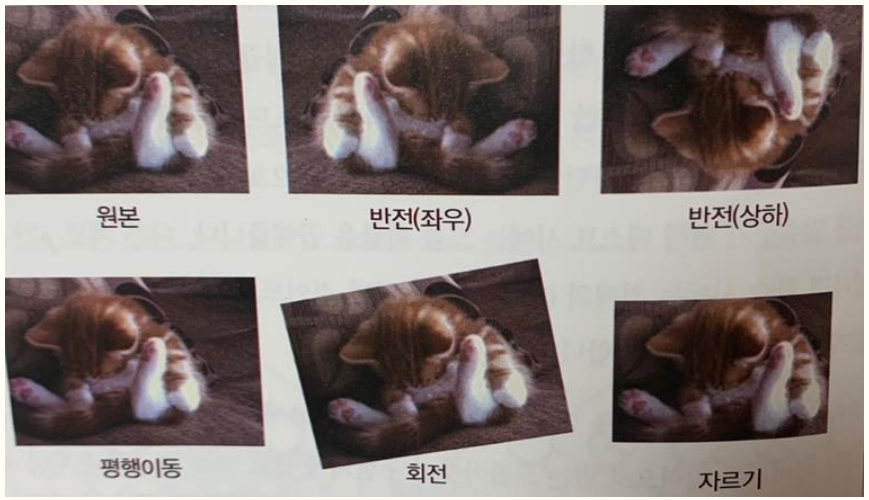
✓ P (0~1) 의 확률로 꺼버려

✓ Parameter 개수가 적게 -> 간단하게 예측 가능하다는 장점

4. 데이터 증강

말그대로 데이터를 늘리는 방법으로 이미지를 돌리거나 뒤집거나

사람의 눈으로는 별 차이가 없지만 컴퓨터가 보기에는 전혀 다른 수치

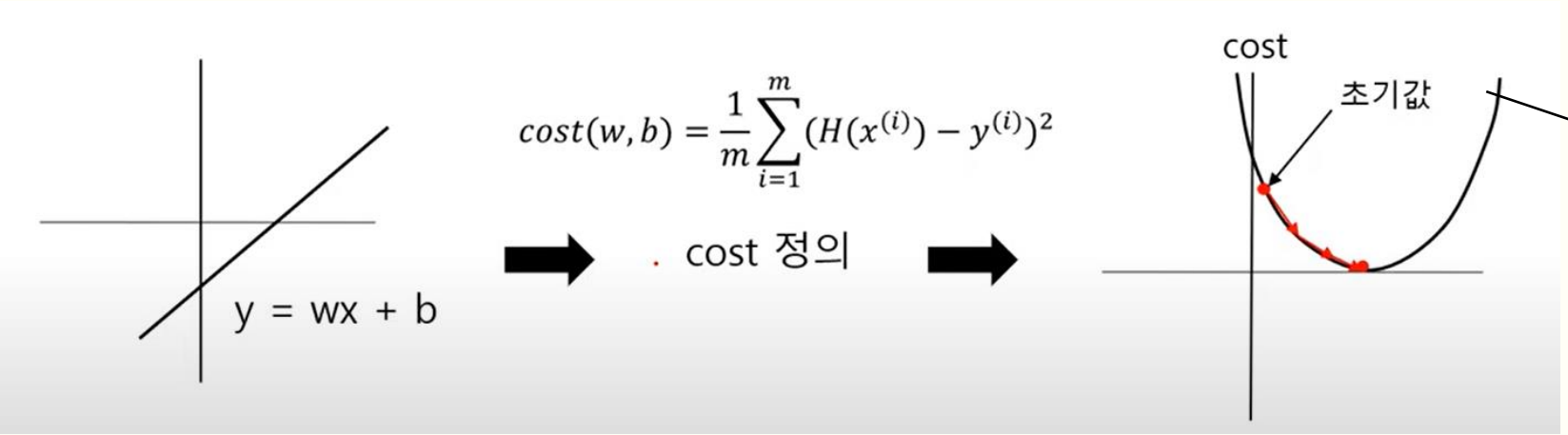


ToTensor, ToPILImage, Normalize, Resize, Scale,
CenterCrop Pad, Lambda, RandomCrop,
RandomHorizontalFlip, RandomVerticalFlip 등

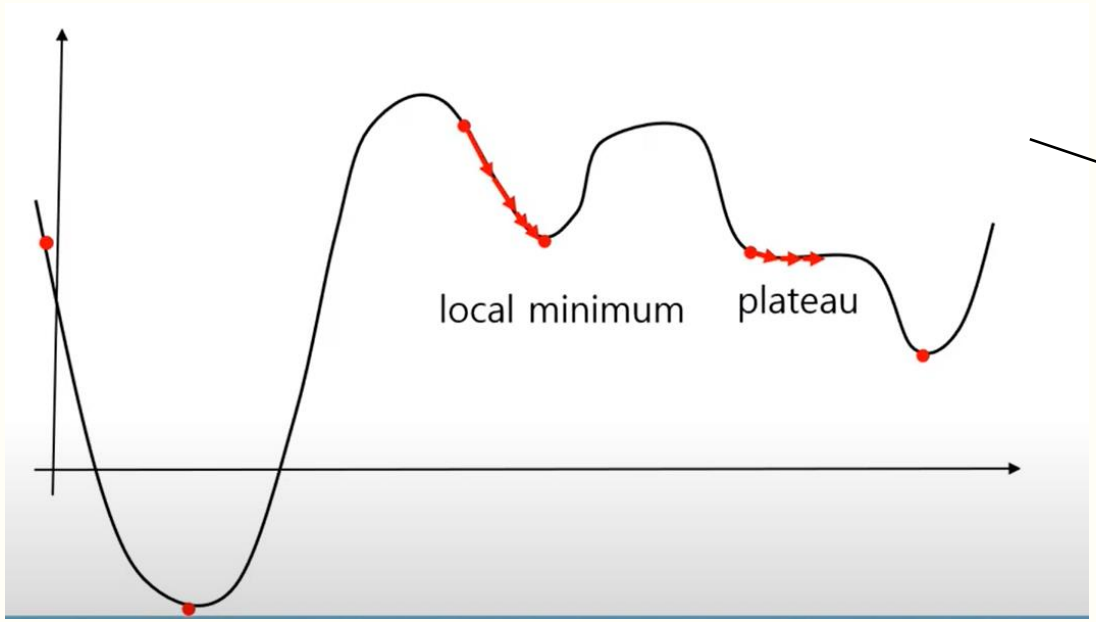
```
mnist_train = dset.MNIST("./", train=True,
                         transform = transforms.Compose([
                             transforms.Resize(34),
# 원래 28x28인 이미지를 34x34로 늘립니다.
                             transforms.CenterCrop(28),
# 중앙 28x28를 뽑아냅니다.
                             transforms.RandomHorizontalFlip(),
# 랜덤하게 좌우반전 합니다.
                             transforms.Lambda(lambda x: x.rotate(90)),
# 람다함수를 이용해 90도 회전해줍니다.
                             transforms.ToTensor(),
# 이미지를 텐서로 변형합니다.
                         ]),
                         target_transform=None,
                         download=True)
```

5. 초기화

하는 이유 ? 손실 함수 공간을 최적화가 쉬운 형태로 바꾸는 방법



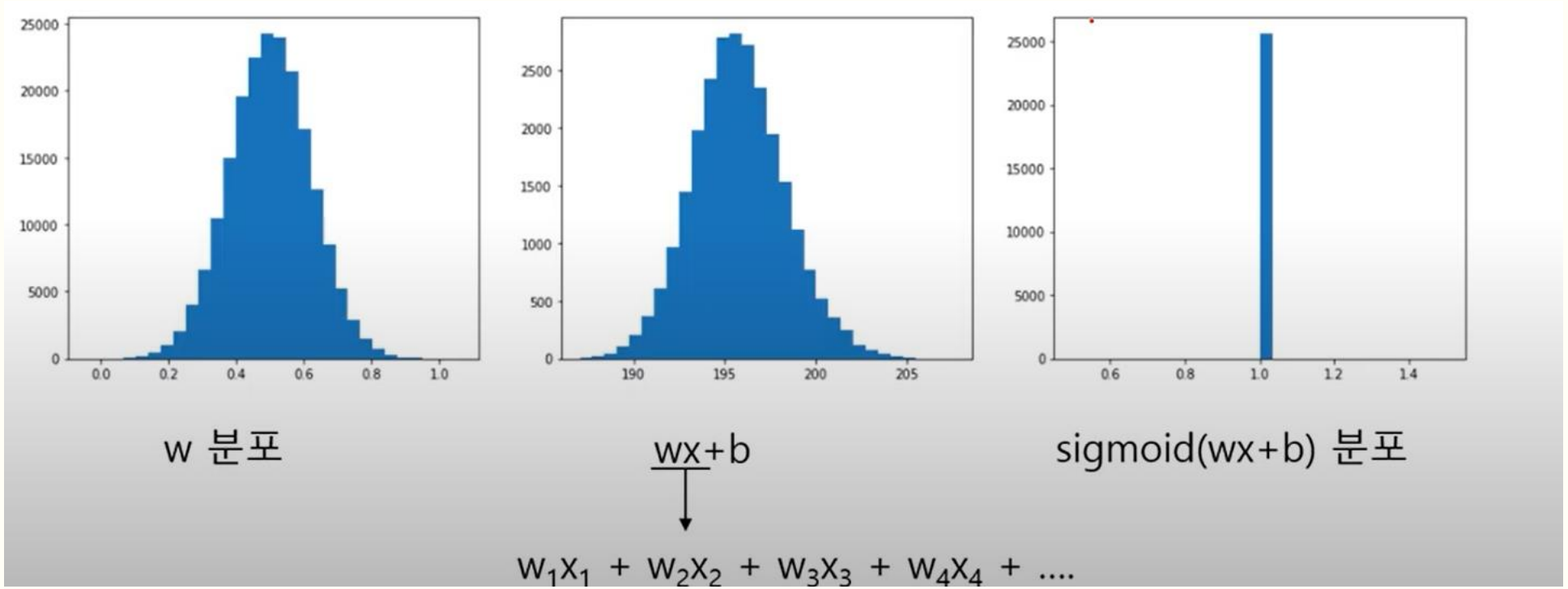
선형함수의 경우, 쉽게
최적값을 찾을 수 있어



출발지점을 어디에서 하느냐에
따라서 최적값이 달라져 ,
잘못 계산

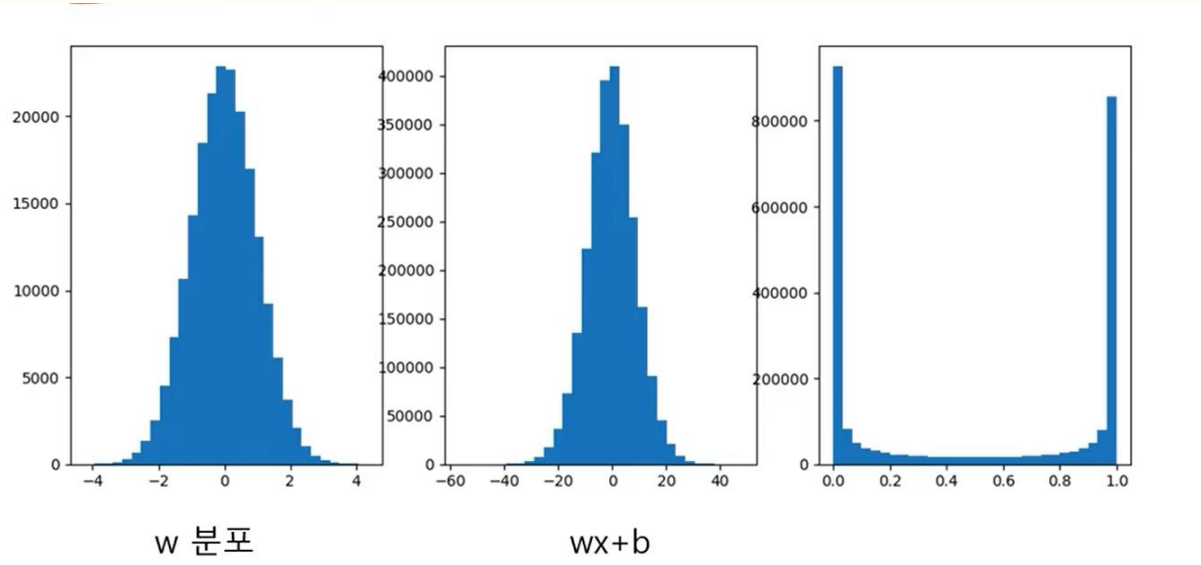
5. 초기화

많은 학습 알고리즘에서 특별한 초기화 방법이 존재하지 않는 경우, 정규분포로부터 랜덤하게 초기화

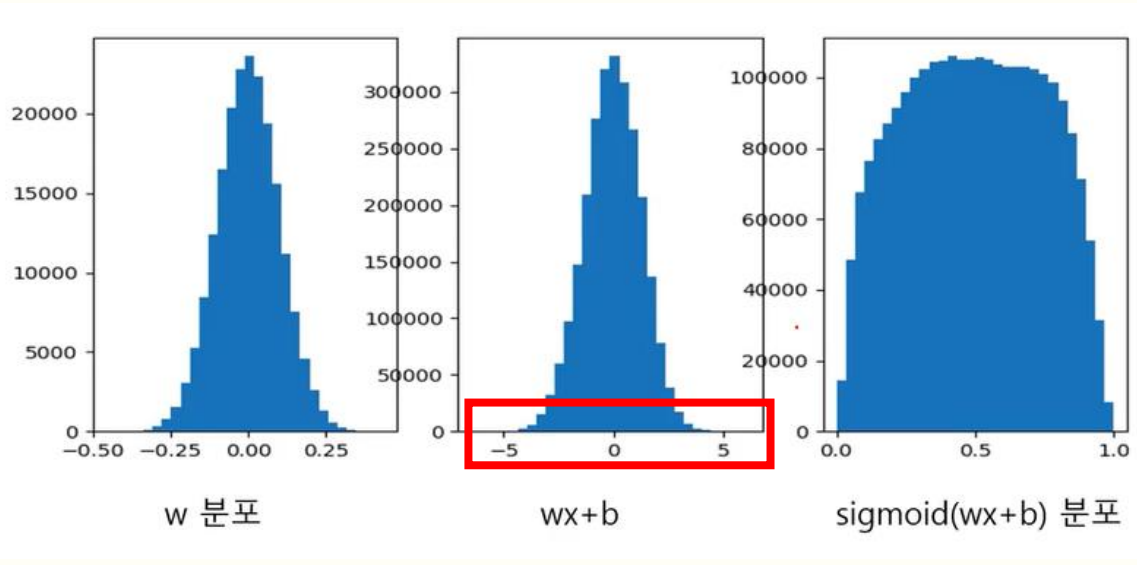


5. 초기화

-4에서 4로 초기화



표준편차를 이용한 최소화 N(0, 0.1)



그러면 표준 편차를 어떻게 설정할 것인가?

5. 초기화

데이터의 특성을 반영하여 표준편차 초기화를 시킴

1. Xavier Gloot

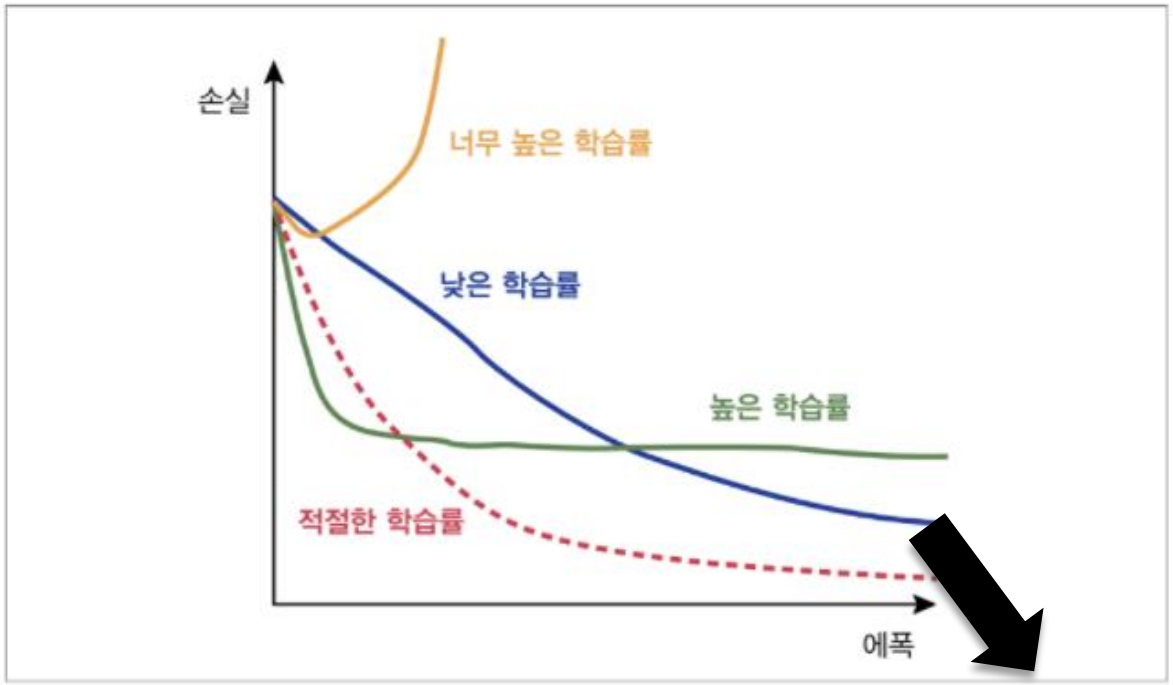
이전 은닉층의 노드의 개수가 n 개이고 현재 은닉층의 노드가 m 개일 때,

$\frac{2}{\sqrt{n+m}}$ 을 표준편차로 하는 정규분포로 가중치를 초기화

2. Kaiming He 초기화

$$N\left(0, \text{var} = \frac{2}{\left(1+a^2\right)\times n_{in}}\right)$$

6. 학습률



학습률과 손실 그래프

학습률이 너무 높다면 업데이트 방향이 맞아도
너무 크게 업데이트되고, 너무 낮다면 지엽적인 공간에서
극솟값에만 도달하므로 전체 손실 공간에서
극솟값에 도달할 수 없어

비교적 높은 학습률에서 시작해
학습률을 점차 낮추는 전략 사용

```
torch.optim.lr_scheduler.StepLR
torch.optim.lr_scheduler.ExponentialLR
torch.optim.lr_scheduler.MultiStepLR
```

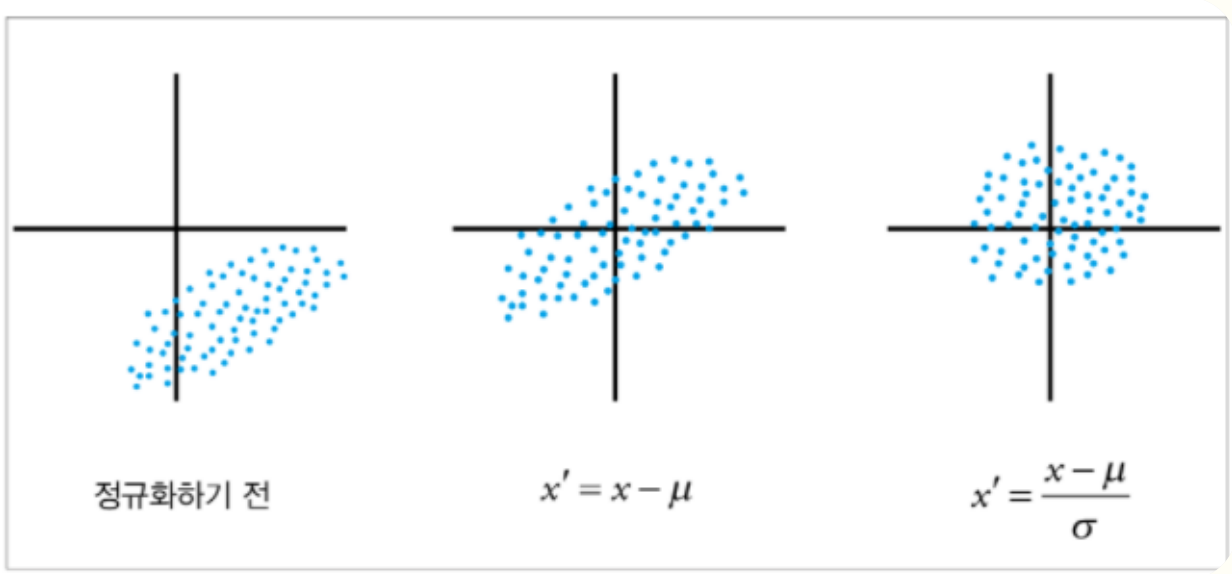
Stepsize(에폭 수)

에폭마다 학습률에 gamma
를 곱해서 감소시킴

원하는 지점에
학습률을 감소시킴

7. 정규화

학습 시 데이터 간의 분포가 다르면 각 분포에 맞춰 변수가 업데이트될 테니
그 데이터를 그대로 쓰면 학습이 제대로 안 될 것



데이터에서 평균을 빼고, 표준편차를
나누어주는 과정

```
x = ( x - x.min() ) / ( x.max()-x.min() )
```

최소극대화정규화의 방법으로
0에서 1로 압축, 늘리는 방법이 있음

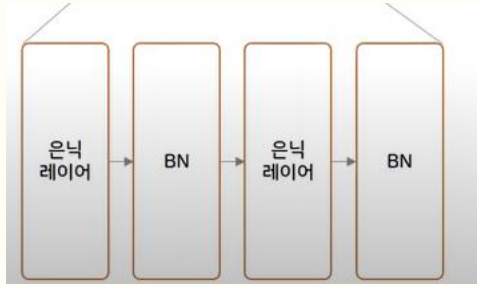
8. 배치정규화

정규화의 필요성 ?

신경망에 데이터를 입력으로 넣을 때는 스케일러를 활용해 모든 **데이터를 공통범위로 배치**
매우 다른 크기의 데이터는 다른 크기의 활성화를 생성하는 경향이 있어 훈련을 불안정하게 함

StandardScaler : 평균과 표준편차 활용
MinMaxScaler : 최대/ 최소값이 각각 1과 0이 되도록 함

입력 전에 하는거 말고, **은닉층에서 정규화**를 하는 것이 배치 정규화

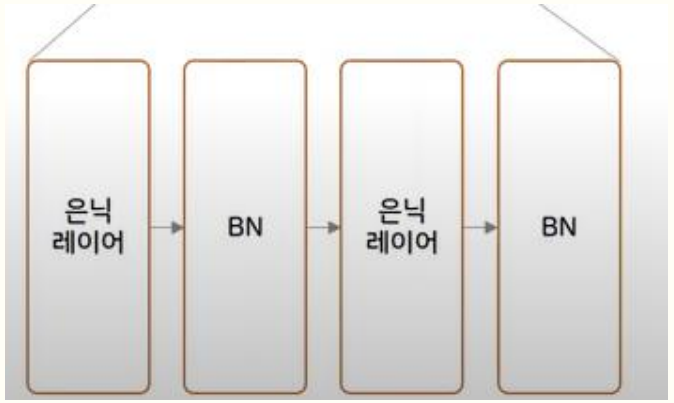


학습 속도 향상
오버피팅 억제
가중치 초기화 민감도 해소

8. 배치정규화

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer = nn.Sequential(
            nn.Conv2d(1,16,3,padding=1),  # 28
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.Conv2d(16,32,3,padding=1), # 28
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2,2), # 14
            nn.Conv2d(32,64,3,padding=1), #14
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2,2) # 7
        )
        self.fc_layer = nn.Sequential(
            nn.Linear(64*7*7,100),
            nn.BatchNorm2d(100),
```

BN 층을 하나씩 추가해줌



THANK YOU!

감 사 합 니 다 !