

5. 초기화와 정규화

세종대학교

인공지능데이터사이언스학과

김 정 현

5. 초기화와 정규화

5.1 가중치 초기화

5.2 정규화

5.3 배치 정규화

5.4 가중치 감소

5.5 조기 종료

5.6 데이터 증강

5.7 배깅

5.8 드롭아웃

5.9 잡음주입

5.1 가중치 초기화

■ 가중치 초기화란?

- 신경망을 학습할 때 손실 함수의 어느 위치에서 출발해야 최적해가 있는 곳으로 쉽게 갈 수 있을까?
- 최적해 근처에서 출발할 수 있다면 빠르고 정확하게 최적해를 찾을 수 있겠지만, 최적해가 어디에 있는지 모른다면 어떤 위치에서 출발하는 게 가장 좋을까?
- 신경망을 학습할 때 손실 함수에서 출발 위치를 결정하는 방법이 모델 초기화
- 특히 가중치는 모델의 파라미터에서 가장 큰 비중을 차지하므로 가중치의 초기화 방법에 따라 학습 성능이 크게 달라질 수 있음

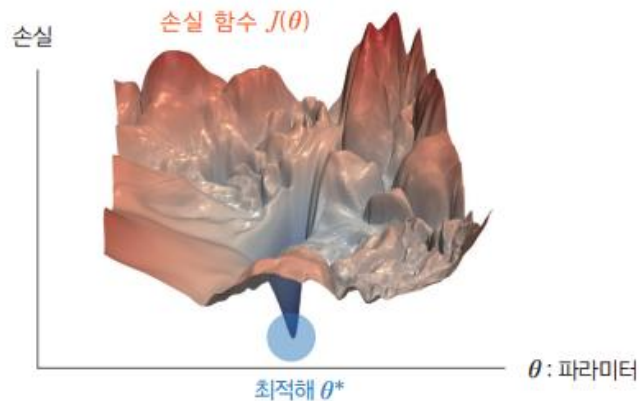


그림 5-1 손실 함수^[20]

5.1 가중치 초기화

■ 상수 초기화

- 최적해에 관한 사전 정보가 없을 때 생각할 수 있는 가중치 초기화 방법 중 하나가 임의의 상수로 초기화하는 것
- 가중치를 0으로 초기화한다면?
 - 신경망의 가중치를 모두 0으로 초기화했다고 해보자.
 - 과연 어떤 일이 일어날까?
 - 뉴런의 가중치가 0이면 가중 합산 결과는 항상 0이 되고, 활성화 함수는 가중 합산 결과인 0을 입력받아서 늘 같은 값을 출력함

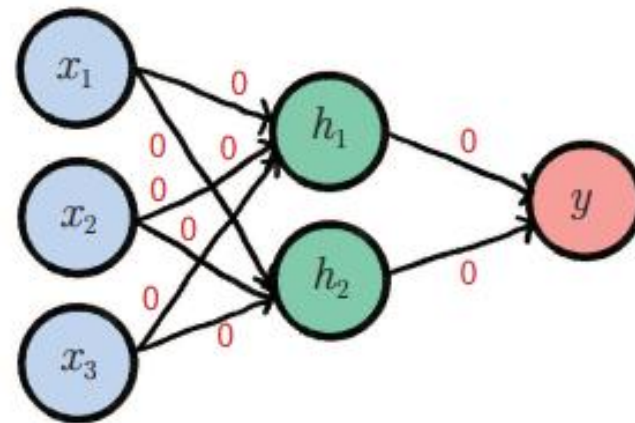


그림 5-2 가중치의 0 초기화

5.1 가중치 초기화

■ 상수 초기화

- 가중치를 0이 아닌 상수로 초기화한다면?
 - 이번에는 가중치를 0이 아닌 다른 상수로 초기화해보자.
 - 다음 그림과 같이 가중치를 0.1로 초기화하면 괜찮을까?

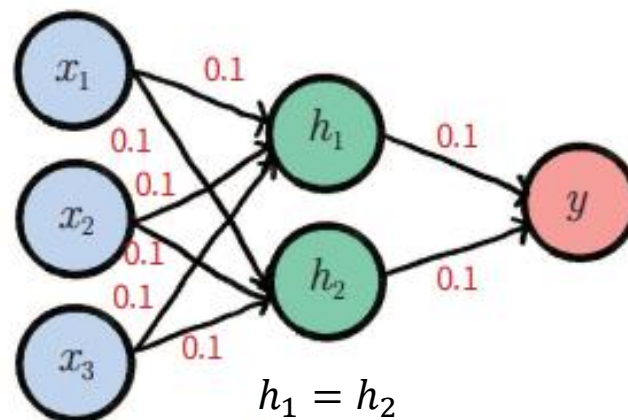


그림 5-3 가중치의 상수 초기화와 신경망의 대칭성

5.1 가중치 초기화

■ 가우시안 분포 초기화

- 대칭성을 피하려면 가중치를 모두 다른 값으로 초기화해야 함
- 이제 가중치를 균등 분포나 가우시안 분포를 따르는 난수를 이용해서 초기화해 보자.
 - 가중치 초기화가 계층별 데이터 분포와 학습에 미치는 현상을 설명하기 위해 다음과 같은 10계층 모델을 가정

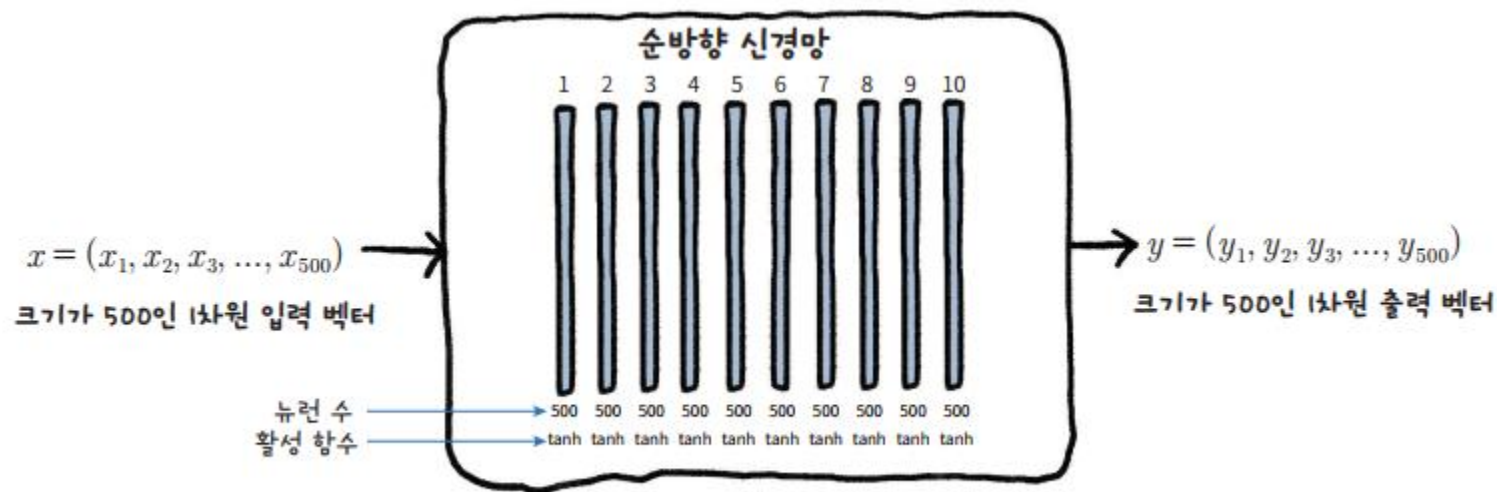


그림 5-4 10계층 순방향 신경망 모델

5.1 가중치 초기화

■ 가우시안 분포 초기화

– 가중치를 아주 작은 난수로 초기화한다면?

- 먼저 모델의 가중치를 가우시안 분포 $N(0, 0.01)$ 을 따르는 난수로 초기화해 보자.
- 가중치가 평균이 0이고 분산이 0.01인 난수로 되어 있기 때문에 아주 작은 값으로 초기화됨
- 신경망에 입력된 데이터는 10개의 계층을 지나면서 다음과 같은 분포로 변화함
- 계층이 깊어질수록 출력이 점점 0으로 변하는 현상 발생

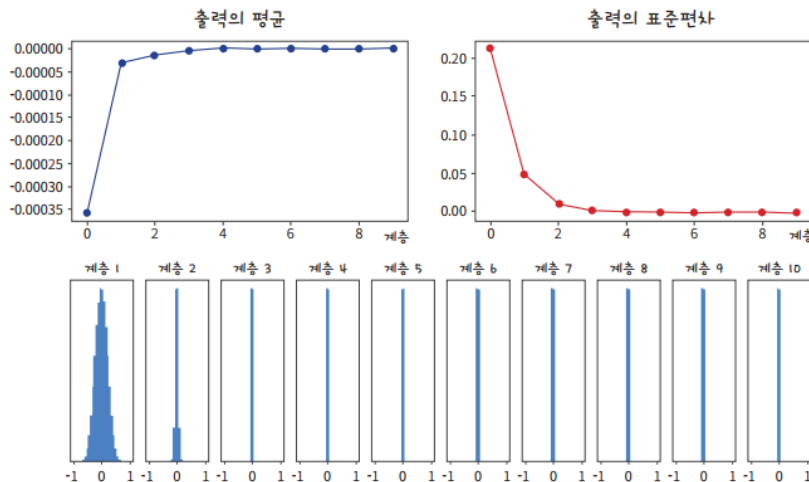
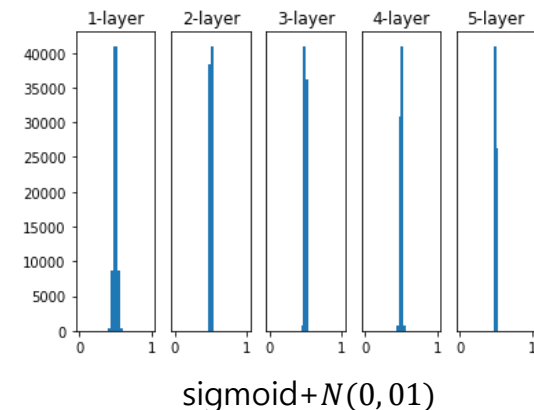


그림 5-5 가중치를 아주 작은 난수로 초기화한 경우

$\tanh + N(0, 0.01)$



5.1 가중치 초기화

■ 가우시안 분포 초기화

– 가중치를 큰 난수로 초기화한다면?

- 가중치가 아주 작을 때 신경망 모델이 정상적으로 학습하지 못한다면, 가중치를 크게 만들면 어떻게 될까?
- 이번에는 가중치를 평균이 0이고 분산이 1인 가우시안 분포 $N(0, 1)$ 로 초기화해 보자.
- 이 경우 다음 그림과 같이 입력 데이터가 각 계층을 지나면서 점점 1이나 -1로 변하는 현상을 확인할 수 있음

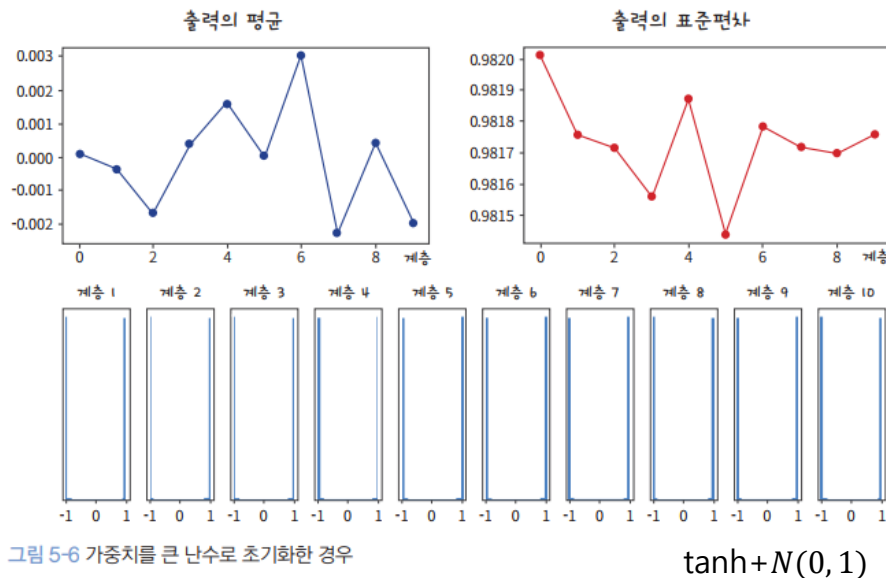
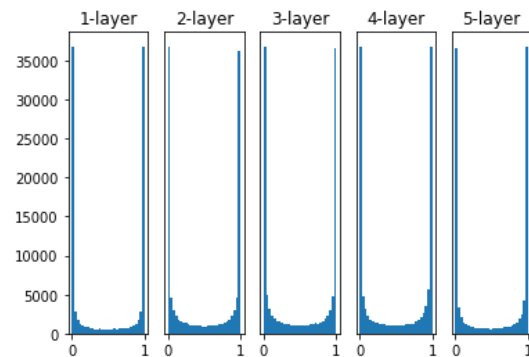


그림 5-6 가중치를 큰 난수로 초기화한 경우



sigmoid+N(0, 1)

5.1 가중치 초기화

- 적절한 가중치는 어떤 값일까?
 - 가중치가 작아도 문제이고 커도 문제라면, 적절한 가중치는 어떤 값이어야 할까?
 - 적어도 데이터의 크기를 점점 작게 만들거나 점점 크게 만들지 않는 값이어야 함
 - 다시 말하면 데이터가 계층을 통과하더라도 데이터의 크기를 유지해주는 가중치로 초기화해야 함
 - 그렇다면 과연 어떤 방법으로 데이터 크기를 유지할 수 있을까?

5.1 가중치 초기화

■ Xavier 초기화

- Xavier 초기화는 시그모이드 계열의 활성화 함수를 사용할 때 가중치를 초기화하는 방법으로, 입력 데이터의 분산이 출력 데이터에서 유지되도록 가중치를 초기화함
- 데이터가 계층을 통과하더라도 같은 크기를 유지하려면 분산이 점점 작아지거나 커지지 않아야 하기 때문

5.1 가중치 초기화

■ Xavier 초기화

- Xavier 초기화 방식의 가정사항
- Xavier 초기화 방식을 유도하기 위한 가정
 - 1. 활성 함수를 선형 함수로 가정하고 입력 데이터는 활성 함수의 가운데 부분을 지난다고 가정

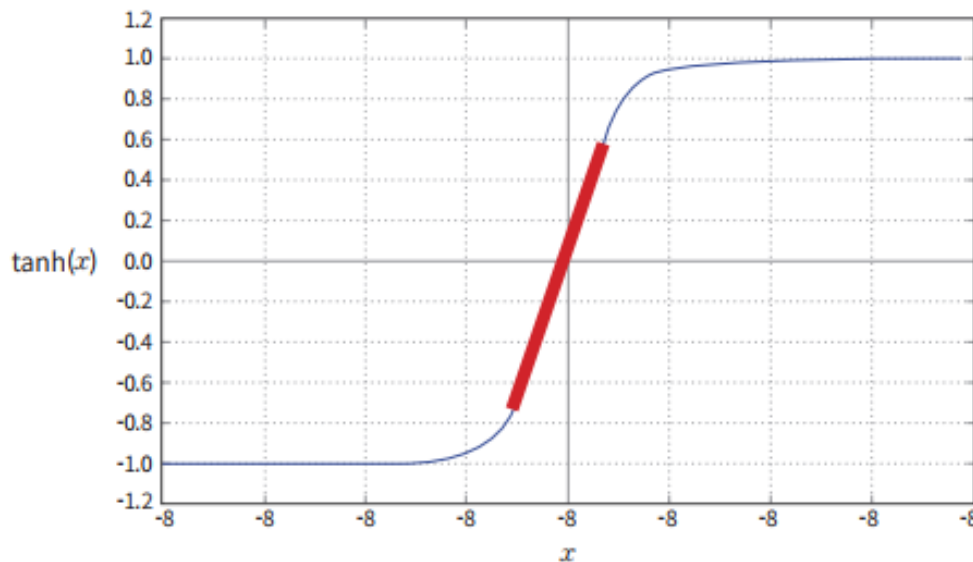


그림 5-7 하이퍼볼릭 탄젠트 함수의 선형성

5.1 가중치 초기화

■ Xavier 초기화

- Xavier 초기화 방식의 가정사항
- Xavier 초기화 방식을 유도하기 위한 가정
 - 2. 입력 데이터와 가중치는 다음과 같은 분포의 성질을 가짐
 - 입력 데이터와 가중치는 서로 독립
 - 입력 데이터의 각 차원 x_i 는 같은 분포이고 서로 독립인 i.i.d를 만족
 - 각중치의 각 차원 w_i 도 같은 분포이고 서로 독립인 i.i.d를 만족
 - 각 x_i 와 w_i 는 평균이 0인 분포를 따름

5.1 가중치 초기화

■ Xavier 초기화

– Xavier 초기화 식의 유도 과정

- 이제 Xavier 초기화 식을 유도해 보자.
- Xavier 초기화는 가중치의 분산이 입력 데이터에 개수 n 에 반비례하도록 초기화하는 방식
- 가중치 분포는 가우시안 분포 또는 균등 분포로 정의할 수 있으며, 다음 그림은 가중치의 분포가 가우시안 분포일 때 정의

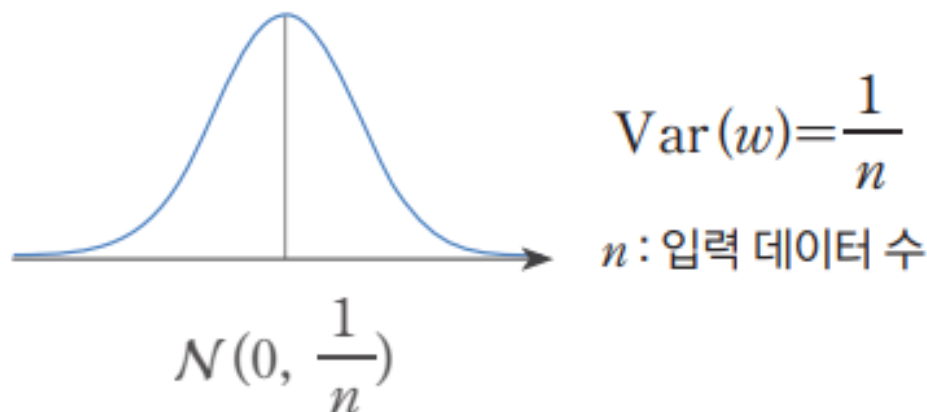


그림 5-8 Xavier 초기화를 위한 가우시안 분포

5.1 가중치 초기화

■ Xavier 초기화

- Xavier 초기화를 이용해서 신경망을 초기화했을 때 실행 결과를 확인
- 이제 입력 데이터가 계층을 여러 번 통과하더라도 분산이 잘 유지되는 것을 확인할 수 있음

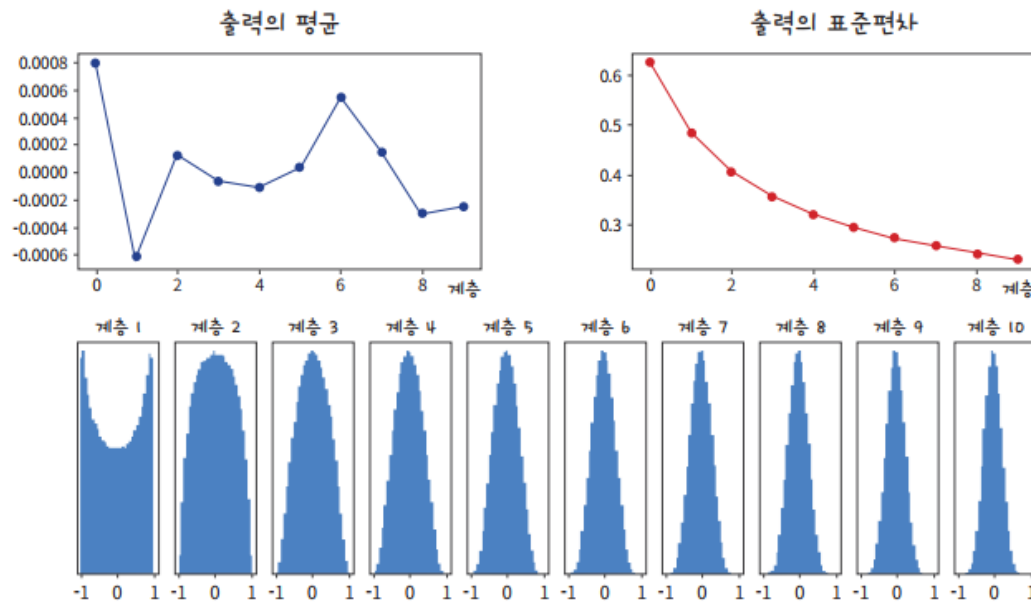
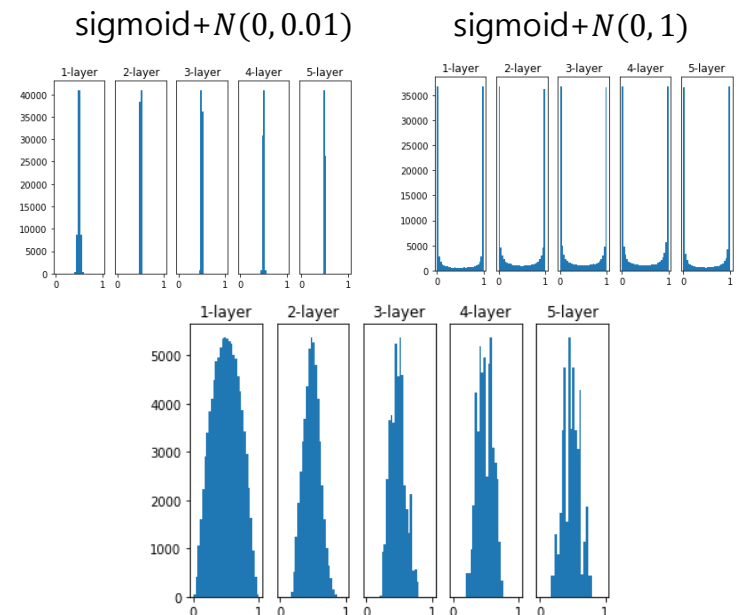


그림 5-9 하이퍼볼릭 탄젠트 사용 시 Xavier 초기화

tanh+Xavier



sigmoid+Xavier

5.1 가중치 초기화

■ He 초기화

- 활성화 함수가 ReLU일 때 Xavier 초기화를 사용하면 데이터의 크기가 점점 작아짐
- 애초에 Xavier 초기화는 시그모이드 계열의 활성화 함수를 사용한다는 전제하에 활성화 함수를 선형 함수로 가정했기 때문
- 반면 ReLU는 양수 구간에서는 이 가정이 유효하지만, 음수 구간에서는 이 가정과 맞지 않음
- 음수 구간에서 입력 데이터가 0으로 변하므로 50%가량의 데이터가 0이 된다면 입력 데이터의 크기, 즉 분산이 절반으로 줄어들기 때문

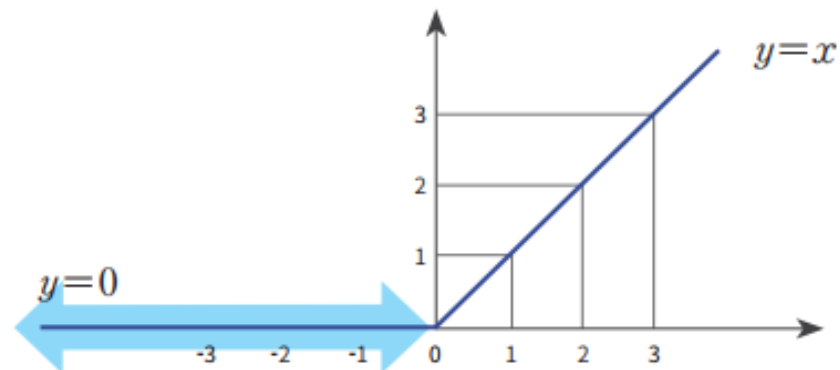


그림 5-10 ReLU의 비활성화 영역

5.1 가중치 초기화

■ He 초기화

- 활성화 함수가 ReLU일 때 Xavier 초기화를 적용해 보면, 입력 데이터가 계층을 통과하면서 분산이 점점 줄어들어 출력이 0이 되는 현상을 확인할 수 있음

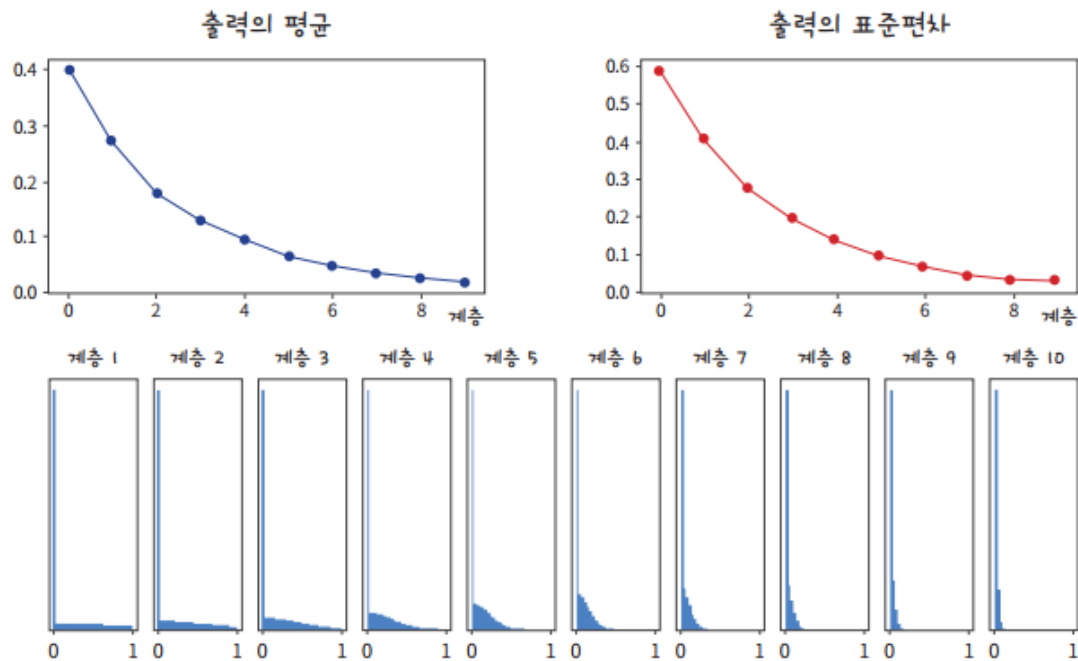


그림 5-11 ReLU 사용 시 Xavier 초기화

5.1 가중치 초기화

■ He 초기화

- 활성화 함수가 ReLU일 때 Xavier 초기화의 한계점을 개선한 방식이 He 초기화
- He 초기화도 Xavier 초기화와 같이 뉴런의 입력 데이터와 출력 데이터의 분산을 같게 만들어줌

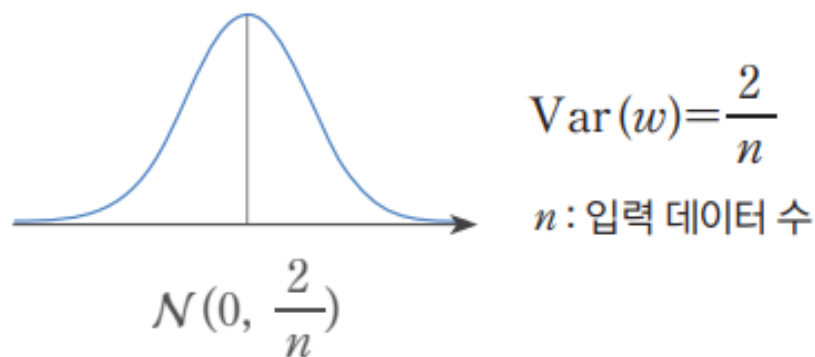


그림 5-12 He 초기화를 위한 정규 분포

5.1 가중치 초기화

■ He 초기화

- 다음 그림과 같이 He 초기화를 적용해 보면 입력 데이터가 계층을 통과하면서 데이터의 분산이 잘 유지되는 것을 확인할 수 있음
- ReLU의 특성상 데이터가 0에 몰려 있지만, 나머지 데이터는 양수 구간에 골고루 퍼져 있는 모습

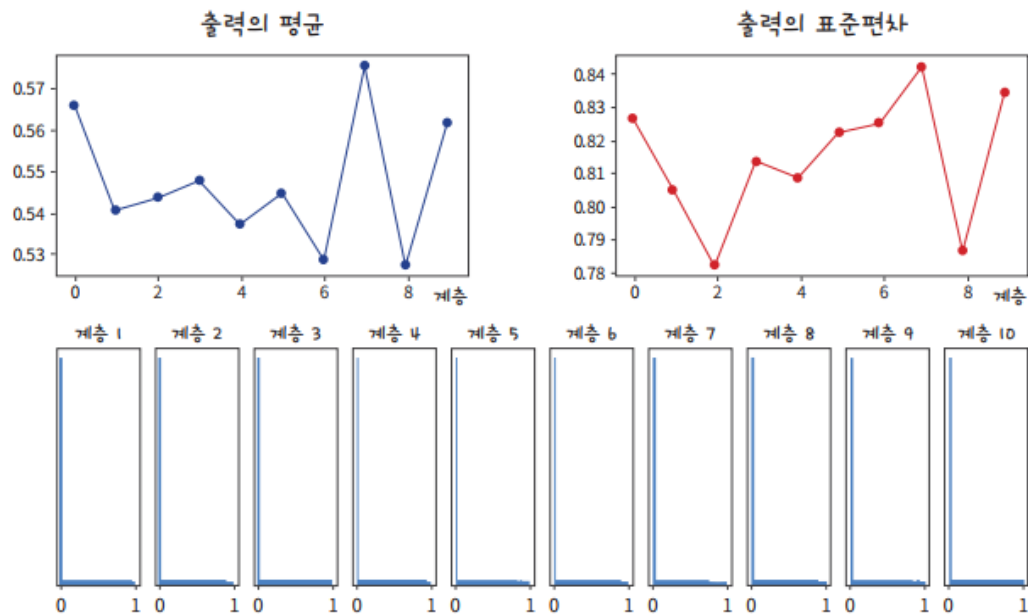


그림 5-13 ReLU 사용 시 He 초기화

5.2 정규화

- 정규화(regularization)란?
 - 배치 정규화(normalization)과 구분하기 위해 규제화라고도 함
 - 신경망을 학습할 때는 최적화에 좋은 위치에서 출발하도록 초기화를 잘하는 것과 더불어, 최적해로 가는 길을 잘 찾을 수 있도록 정규화하는 것이 중요
 - 정규화는 최적화 과정에서 최적해를 잘 찾도록 정보를 추가하는 기법으로, 최적화 과정에서 성능을 개선할 수 있는 포괄적인 기법들을 포함

5.2 정규화

■ 일반화 오류

- 모델의 성능이 좋다는 말은 일반화가 잘 되었다는 의미
- 일반화란 훈련 데이터가 아닌 새로운 데이터에 대해 모델이 예측을 얼마나 잘하는지를 가리킴
- 모델의 훈련 성능과 검증/테스트 성능의 차를 일반화 오류라고 하며 일반화 오류가 적을수록 일반화가 잘 된 모델

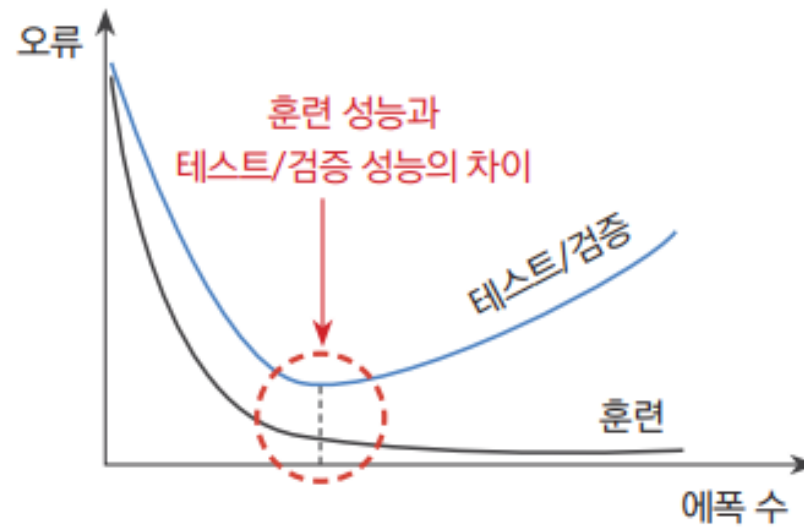


그림 5-14 일반화 오류

5.2 정규화

■ 정규화 접근 방식

- 정규화의 정의가 포괄적인 만큼 정규화 기법도 다양하지만, 기본적인 접근 방법은 다음과 같이 몇 가지로 정리됨
 - 첫째, 모델을 최대한 단순하게 만들기

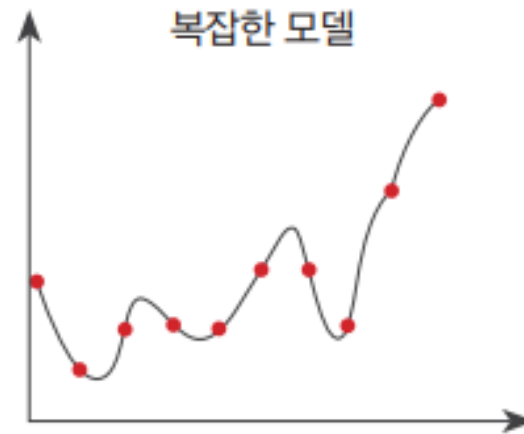
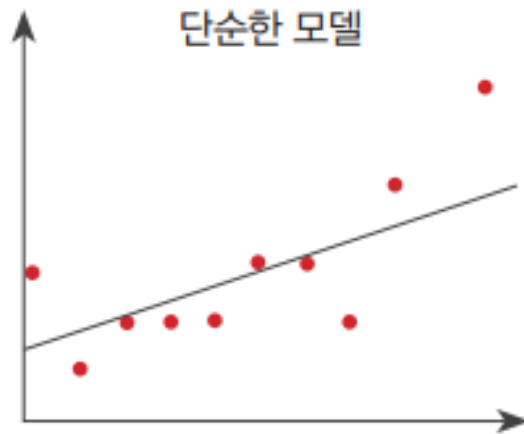


그림 5-15 단순한 모델과 복잡한 모델

5.2 정규화

■ 정규화 접근 방식

- 정규화의 정의가 포괄적인 만큼 정규화 기법도 다양하지만, 기본적인 접근 방법은 다음과 같이 몇 가지로 정리됨
 - 둘째, 사전 지식을 표현해서 최적해를 빠르게 찾도록 함
 - 가중치 감소도 가중치의 사전 분포를 활용한 예
 - 셋째, 확률적 성질을 추가
 - 데이터 증강, 잡음 주입, 드롭아웃 등

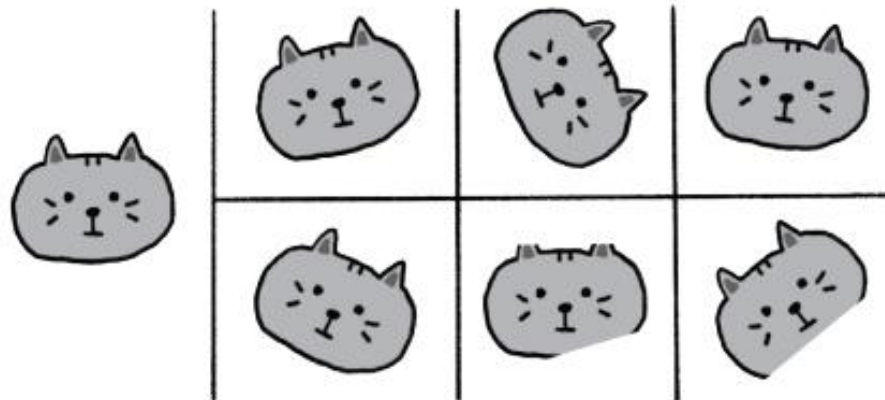


그림 5-16 데이터에 랜덤성을 추가

5.2 정규화

■ 정규화 접근 방식

- 정규화의 정의가 포괄적인 만큼 정규화 기법도 다양하지만, 기본적인 접근 방법은 다음과 같이 몇 가지로 정리됨
 - 넷째, 여러 가설을 고려하여 예측

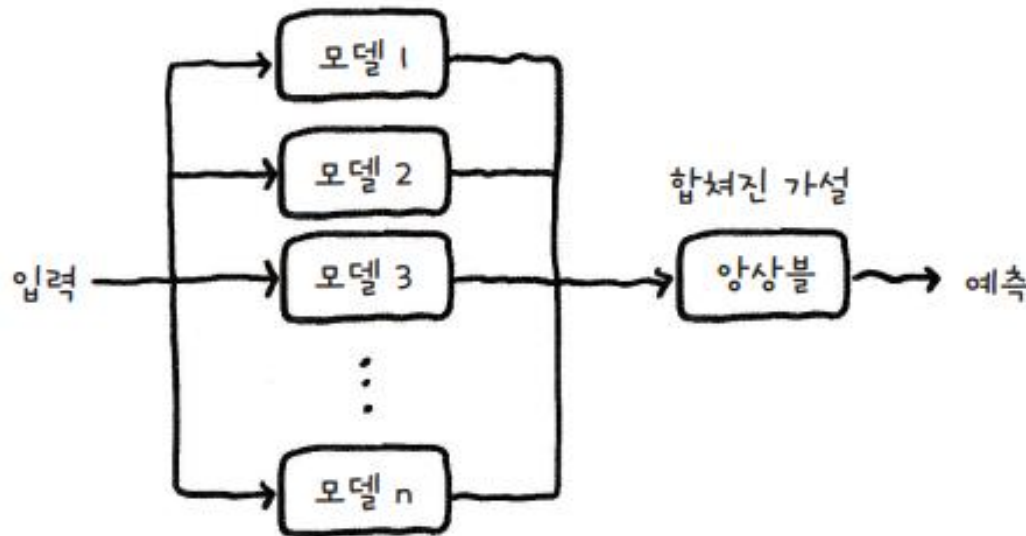
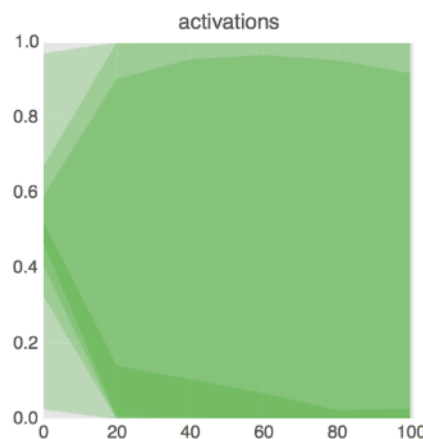
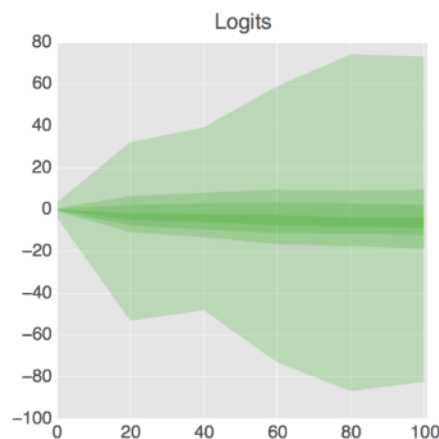


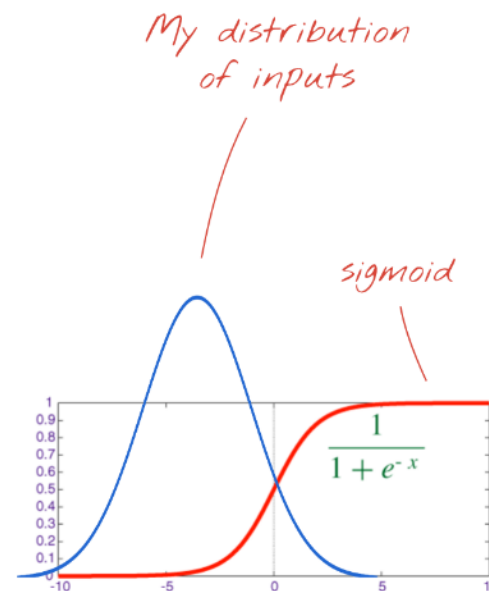
그림 5-17 여러 가설을 함께 고려해서 예측

5.3 배치 정규화

- 배치 정규화(batch normalization) 필요성
 - 신경망 학습이 어려운 이유 중 하나는 계층을 지날 때마다 데이터 분포가 보이지 않는 요인에 의해 조금씩 왜곡되기 때문
 - 데이터 왜곡을 막으려면 가중치 초기화를 잘해야 하고 학습률도 작게 사용해야 하는데 이 경우 학습 속도가 느려지는 문제 발생

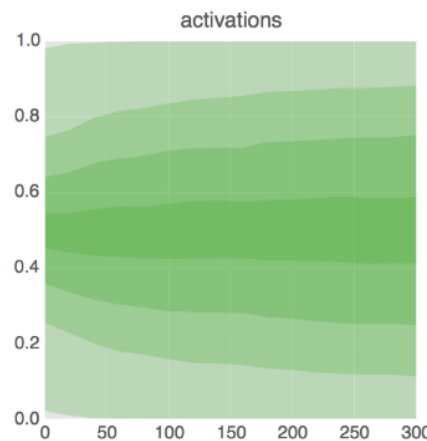
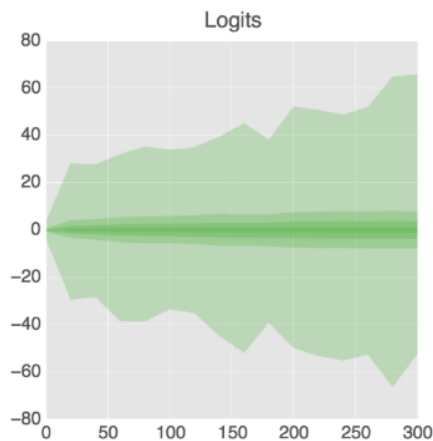


boo-hoo

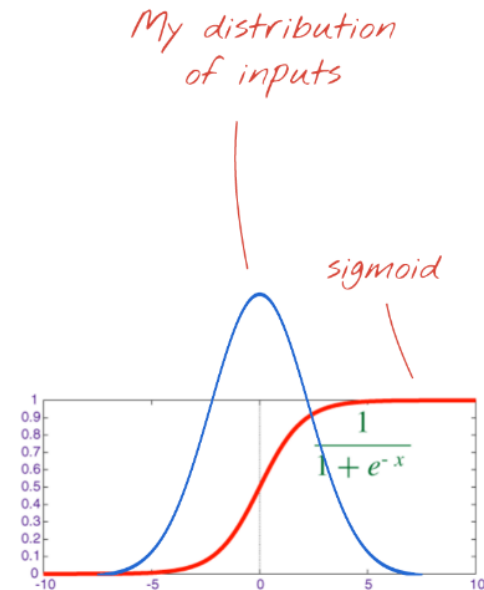


5.3 배치 정규화

- 배치 정규화(batch normalization) 필요성
 - 배치 정규화를 통해 계층 통과시 데이터 왜곡 문제를 해결

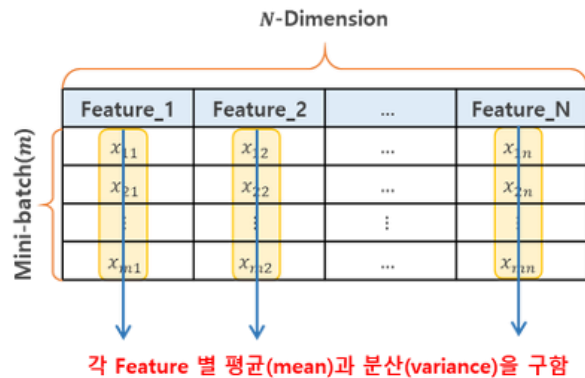


Batch norm



5.3 배치 정규화

- 배치 정규화(batch normalization) 필요성
 - 배치 정규화를 통해 계층 통과시 데이터 왜곡 문제를 해결



Normalize

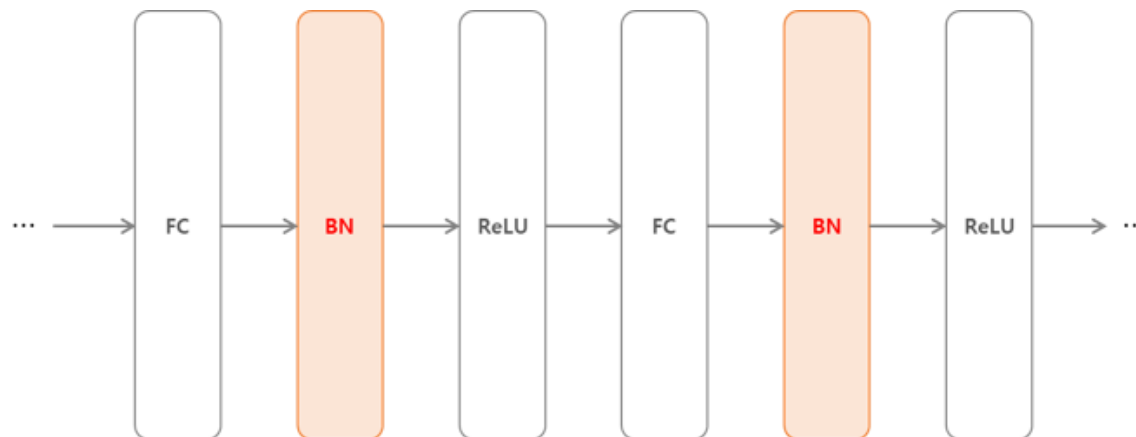
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad // \text{ mini-batch variance}$$

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

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$



5.3 배치 정규화

■ 내부 공변량 변화

- 데이터 분포가 보이지 않는 요인에 의해 왜곡되는 현상을 내부 공변량 변화라고 함
- 분포를 결정하는 보이지 않는 요인을 내부 공변량이라고 하며, 내부 공변량이 바뀌면 다음 그림과 같이 각 계층의 데이터 분포가 원래 분포에서 조금씩 멀어짐
- 그 결과 하위 계층의 작은 변화가 상위 계층으로 갈수록 큰 영향을 미치게 됨

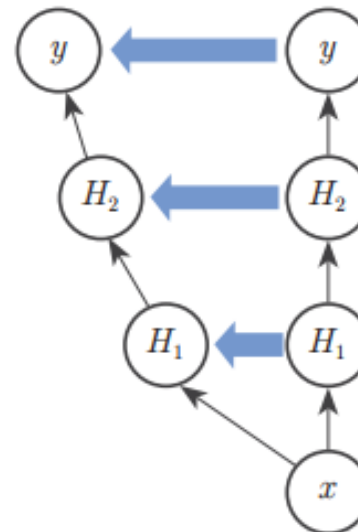


그림 5-18 내부 공변량 변화

5.3 배치 정규화

■ 배치 정규화 단계

- 배치 정규화는 데이터가 계층을 지날 때마다 매번 정규화해서 내부 공변량 변화를 없애는 방법
- 배치 정규화가 기존 데이터 정규화 방식과 다른 점은 모델의 계층 형태로 데이터 정규화를 실행한다는 점
- 따라서 배치 정규화를 하면 모델이 실행될 때마다 해당 계층에서 매번 데이터 정규화가 일어남
- 또한 전체 데이터에 대해 정규화하지 않고 미니배치에 대해 정규화한다는 점도 다름

5.3 배치 정규화

- 표준 가우시안 분포로 정규화
 - 다음과 같이 d 차원의 입력 데이터 $x = (x^{(1)}, x^{(2)}, \dots, x^{(d)})$ 가 있다면, 배치 정규화는 차원별로 평균과 분산을 구해서 표준 가우시안 분포 $N(0, 1)$ 로 정규화함
 - 표준 가우시안 분포로 정규화하므로 데이터의 크기가 작아지면서 내부 공변량의 변화도 작게 만들 수 있음

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}, \quad k=1, 2, \dots, d$$

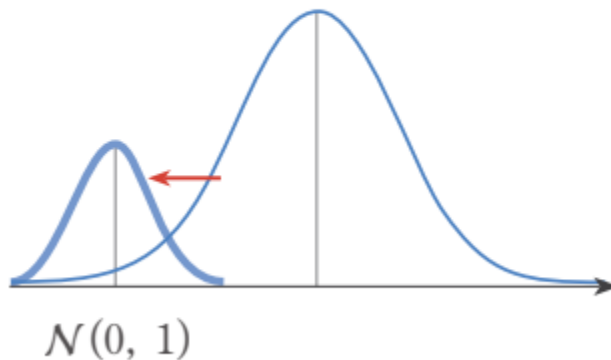


그림 5-19 표준 정규화

5.3 배치 정규화

- 표준 가우시안 분포로 정규화
 - 배치 정규화를 모든 계층에 적용하면 데이터가 계층을 지날 때마다 표준 가우시안 분포로 바뀌고 그에 따라 내부 공변량의 변화를 최소화할 수 있음
 - 원리적으로는 계층을 지나면서 생기는 데이터 오차의 크기를 줄임으로써 누적 오차도 작게 만드는 작업을 한 것

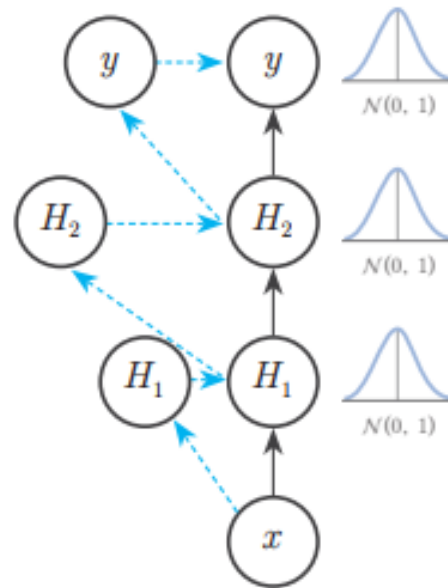


그림 5-20 배치 정규화

5.3 배치 정규화

- 원래 분포로 복구
 - 그런데 데이터를 표준 가우시안 분포로 정규화하면 모델이 표현하려던 비선형성을 제대로 표현할 수 없는 문제가 생김

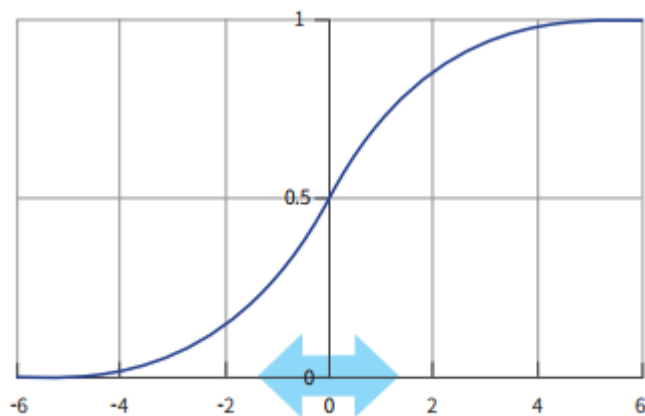


그림 5-21 시그모이드 함수에서 표준 정규화를 할 때 데이터 범위

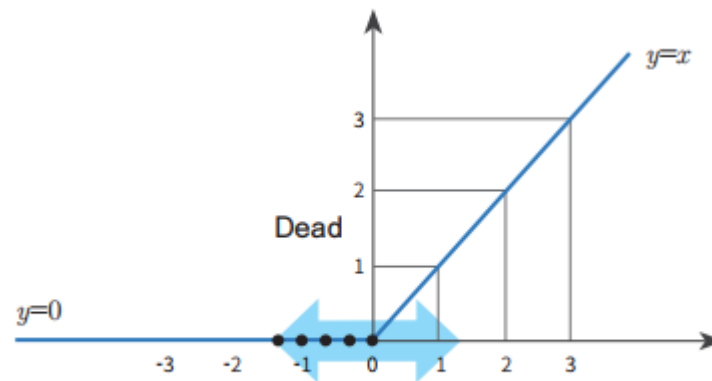


그림 5-22 ReLU에서 표준 정규화를 할 때 데이터 범위

5.3 배치 정규화

■ 원래 분포로 복구

- 따라서 배치 정규화를 하면서 모델의 비선형성을 잘 표현하려면 데이터를 표준 가우시안 분포로 정규화한 뒤 적절한 평균과 분산을 갖도록 재조절하는 과정이 필요함
- 이때 사용되는 파라미터 γ 와 β 는 학습 파라미터로 손실 함수를 최소화하는 방향으로 학습됨
- 이 과정 역시 배치 정규화의 한 단계임

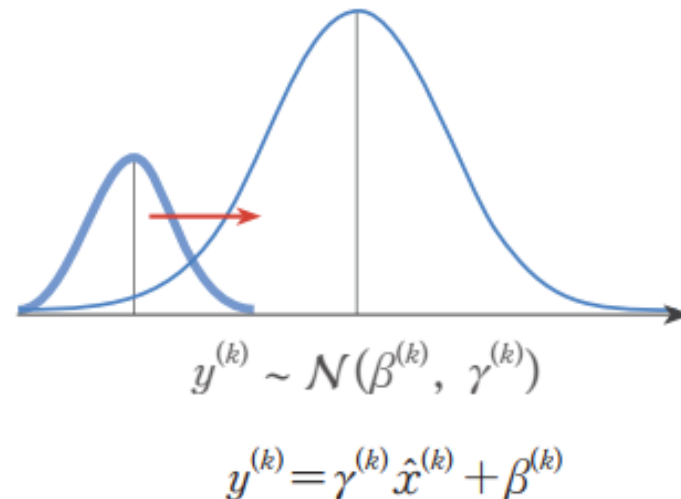


그림 5-23 정규화된 데이터의 선형 변환

5.3 배치 정규화

- 배치 정규화 알고리즘
 - 배치 정규화의 학습 알고리즘

입력: 미니배치: $\mathcal{B} = \{x_1 \dots x_m\}$;

학습 파라미터: γ, β

출력: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{미니배치 평균}$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad // \text{미니배치 분산}$$

$$\hat{x}_i \leftarrow \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} \odot (x_i - \mu_B) \quad // \text{표준 정규화}$$

$$y_i \leftarrow \gamma \odot \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{스케일링 및 시프트}$$

전체 데이터의 평균 $\mathbb{E}[x]$ 과 분산 $\text{Var}[x]$ 은 다음과 같은 식으로 구할 수 있다.

$$\mathbb{E}[x] \leftarrow \mathbb{E}_B[\mu_B]$$

$$\text{Var}[x] \leftarrow \frac{m}{m-1} \mathbb{E}_B[\sigma_B^2]$$

그림 5-25 배치 정규화 학습 알고리즘^[45]

5.3 배치 정규화

■ 배치 정규화 수행 위치

- 그렇다면 배치 정규화는 모델의 어느 위치에서 실행하면 좋을까?
- 배치 정규화를 처음 제안했을 때는 뉴런의 가중 합산과 활성화 함수 사이에서 수행하는 것으로 제안
- 하지만 여러 후속 연구에 따르면 활성화 함수를 실행한 뒤에 배치 정규화를 수행했을 때 더 나은 성능을 보이기도 함
- 일반적으로는 가중 합산한 뒤에 배치 정규화를 적용하지만, 모델의 성능을 세밀하게 개선하려면 활성화 함수 이후에 적용했을 때 성능도 검증해 볼 필요가 있음

5.3 배치 정규화

■ 이미지 정규화 기법

- 배치 정규화를 이미지에 적용할 때는 채널 단위로 정규화를 수행
- 이미지의 경우 배치정규화 이외에 응용에 따라 좀 더 세분된 정규화 방식을 사용

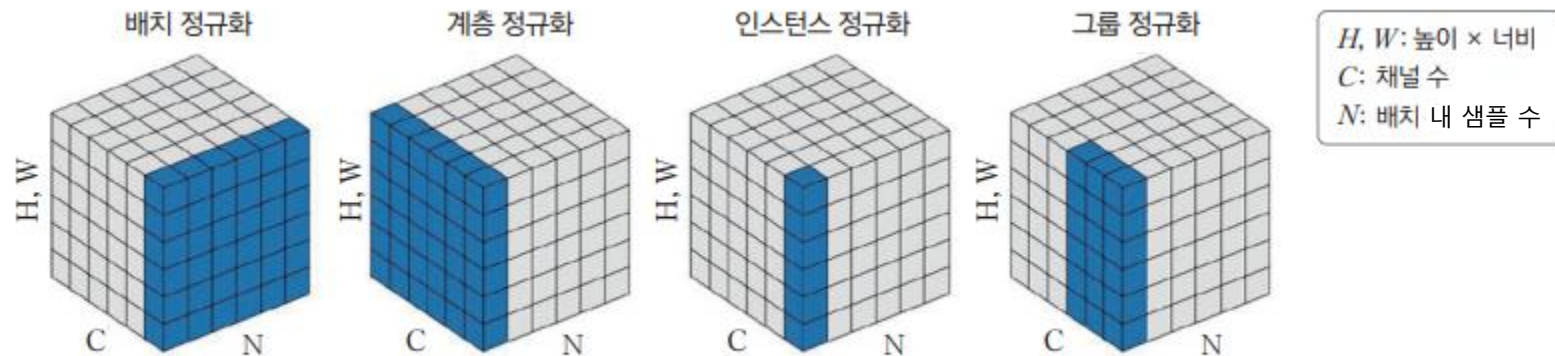


그림 5-26 다양한 이미지 정규화 방법^[52]

5.3 배치 정규화

- 배치 정규화의 우수성
 - 배치 정규화를 하면 내부 공변량 변화가 최소화되므로 그레이디언트의 흐름이 원활해지고 그에 따라 학습이 안정적으로 진행됨
 - 또한 지속적으로 데이터 분포를 유지하기 때문에 초기화 방법에 대한 의존도가 낮아지고 높은 학습률을 사용해도 됨

5.4 가중치 감소

■ 가중치 감소란?

- 직선의 방정식 $w^T x + b = 0$ 의 양변에 2를 곱하면 $2w^T x + 2b = 0$ 이 됨
- 이 두 방정식은 같은 직선을 표현
- 하나의 직선을 표현하는 방정식은 무한히 많은데, 이 중 어떤 방정식을 사용하는 것이 좋을까?
- 최적화할 때는 다루는 숫자의 크기가 작을수록 오차의 변동성이 낮아지므로 파라미터 공간이 원점 근처에 있을 때 정확한 해를 빠르게 찾을 수 있음
- 그래서 직선의 방정식 $w^T x + b = 0$ 식을 표현할 때 가중치와 편향이 작은 게 좋음
- 가중치 감소는 학습 과정에서 작은 크기의 가중치를 찾게 만드는 정규화 기법

5.4 가중치 감소

■ 가중치 감소 적용 방식

- 가중치 감소는 가중치의 크기를 제한하는 제약 조건으로서 손실 함수의 일부 항으로 표현할 수 있음
- 다음 식과 같이 손실 함수로 확장해서 가중치의 크기를 표현하는 정규화 항을 더하면, 최적화 과정에서 원래의 손실 함수와 함께 정규화 항(regularization term)도 같이 최소화되므로 크기가 작은 가중치 해를 구할 수 있음

$$\tilde{J}(w) = \underbrace{J(w)}_{\text{데이터 손실}} + \underbrace{\lambda R(w)}_{\text{정규화}} \quad \lambda: \text{정규화 상수}$$

5.4 가중치 감소

■ 가중치 감소 적용 방식

- 정규화 항 $R(w)$ 은 가중치의 크기를 나타내는 노름으로 정의
- L_2 노름을 사용하면 L_2 정규화라 하고 L_1 노름을 사용하면 L_1 정규화라 함
- 회귀 문제에서는 각각 리지 회귀와 라소 회귀라고 부름

L_2 정규화

$$\tilde{J}(w) = J(w) + \frac{\lambda}{2} \|w\|_2^2$$

리지 회귀

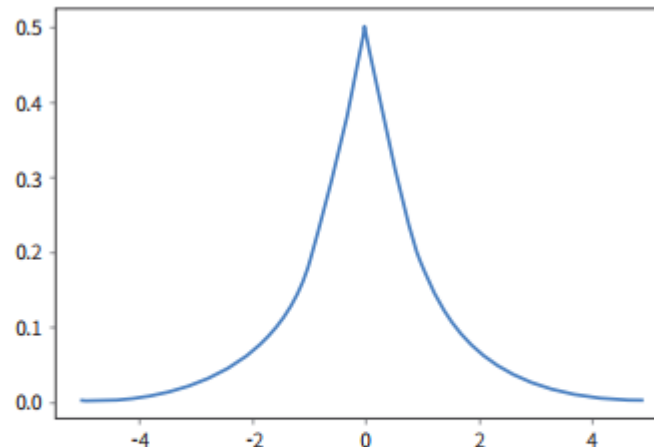
L_1 정규화

$$\tilde{J}(w) = J(w) + \lambda \|w\|_1$$

라소 회귀

5.4 가중치 감소

- 가중치의 사전 분포와 노름
 - 정규화 항에 L_1 노름을 사용할지 L_2 노름을 사용할지는 가중치의 사전 분포에 따라 달라짐
 - 만일 가중치의 사전 분포가 가우시안 분포라면 L_2 노름을 사용하고, 라플라스 분포라면 L_1 노름을 사용함
 - 가중치의 사전 분포를 모른다면 보통 L_2 노름을 사용



$$f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

그림 5-28 라플라스 분포

5.4 가중치 감소

- 가중치 감소 정규화 항의 노름 유도 과정
 - 가중치의 사전 분포에 따른 정규화 항의 노름 유도 과정

가우시안 분포

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{\sqrt{2\pi^D} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

μ : 평균 벡터, Σ : 공분산 행렬, D : 변수 x 의 차원



$$\begin{aligned} -\log \mathcal{N}(w|0, \mathbf{I}) &= -\log \frac{1}{\sqrt{2\pi^D}} e^{-\frac{1}{2}(w-0)^T (w-0)} \\ &= \frac{1}{2} \|w\|_2^2 + \text{const} \end{aligned}$$

라플라스 분포

$$f(x|\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$



$$\begin{aligned} -\log f(w|0, \mathbf{I}) &= -\log \frac{1}{2} e^{-|w-0|} \\ &= \|w\|_1 + \text{const} \end{aligned}$$

5.4 가중치 감소

■ 정규화 효과

- L_2 정규화와 L_1 정규화의 효과는 약간 다름
- 다음 그림에서 왼쪽은 L_2 정규화를, 오른쪽은 L_1 정규화를 보여줌
- 원형 등고선은 원래 손실 함수 $J(w)$ 를 나타내며 등고선의 중심이 원래 손실 함수의 최적해가 있는 부분
- 원점을 중심으로 하는 동그라미와 다이아몬드는 각각 L_2 정규화 항과 L_1 정규화 항을 나타냄

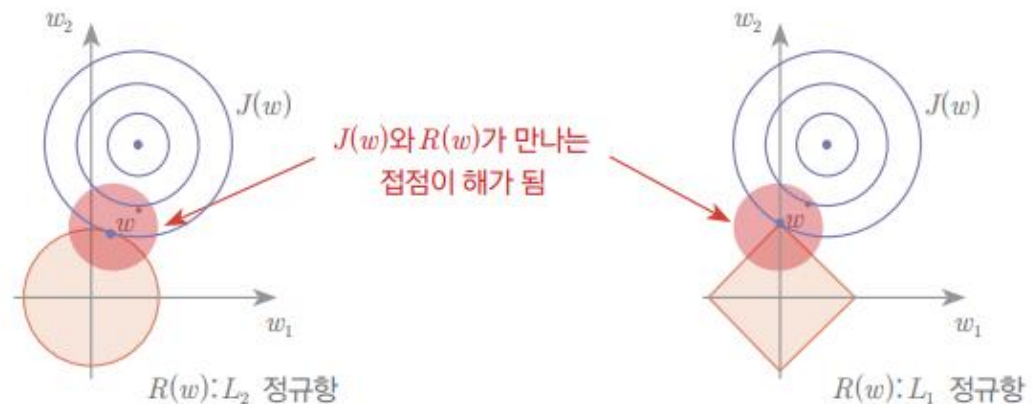


그림 5-29 가중치 감소 정규화를 적용했을 때 해의 위치

5.5 조기 종료

■ 조기 종료란?

- 조기 종료는 모델이 과적합되기 전에 훈련을 멈추는 정규화 기법
- 다음 그림과 같이 훈련 성능과 테스트/검증 성능을 비교해 보면 모델이 과적합되는 걸 알 수 있음
- 과적합이 일어나면 훈련 성능은 계속 좋아지지만 테스트/검증 성능은 좋아지다가 다시 나빠지기 때문

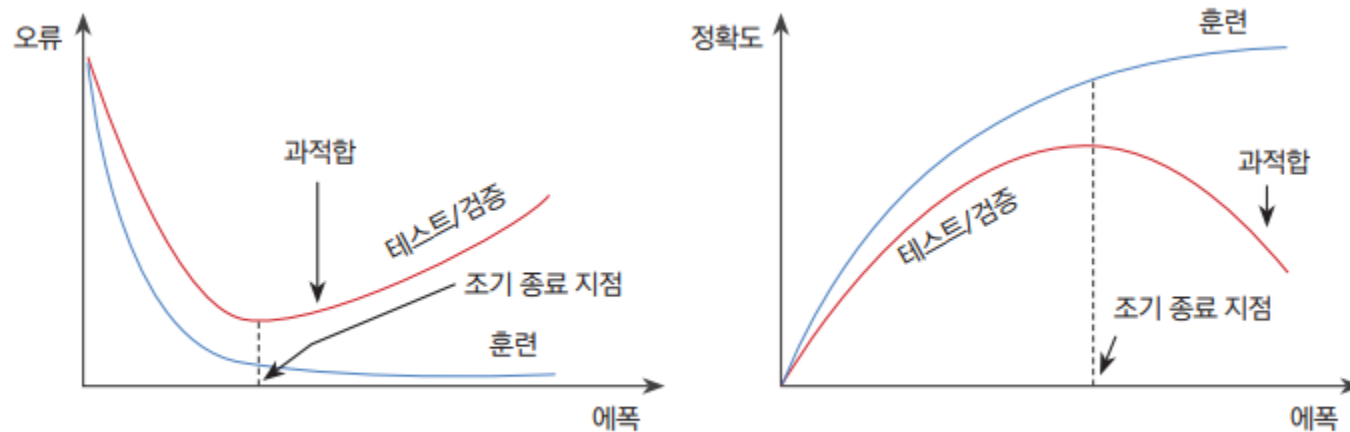


그림 5-30 조기 종료

5.5 조기 종료

■ 조기 종료 기준

- 한 가지 유의해야 할 점은 모델의 성능이 향상하지 않더라도 바로 종료해서는 안 된다는 점
- 신경망을 학습할 때 단계마다 미니배치로 근사한 그레이디언트는 실제 그레이디언트와 차이가 있으므로 성능이 조금씩 좋아졌다 나빠졌다 할 수 있음
- 따라서 일시적인 성능 변동이 아닌, 지속적인 성능의 정체 또는 하락이 판단되면 그때 종료하는 것이 바람직함

5.5 조기 종료

- 조기 종료의 정규화 효과
 - 조기 종료는 파라미터 공간을 작게 만드는 효과
 - 다음 그림과 같이 파라미터 공간에서 초기 파라미터 위치가 w_0 이고 최적화 스텝 수가 τ , 학습률이 a 라면 파라미터 공간은 w_0 를 중심으로 τa 크기의 반경을 갖는 공간으로 제약됨

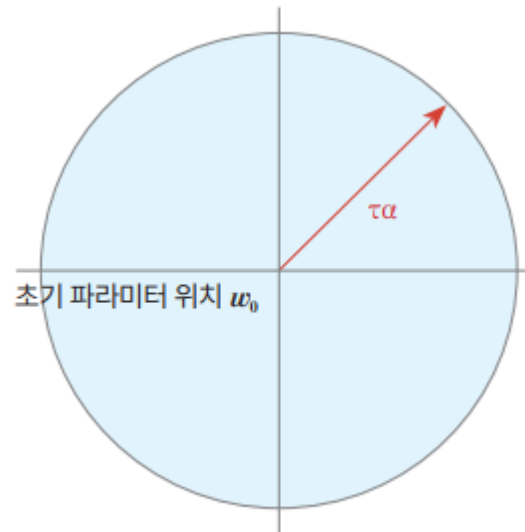


그림 5-31 파라미터 공간에서 조기 종료의 영향

5.5 조기 종료

- 조기 종료와 L_2 정규화의 관계
 - 조기 종료로 파라미터 공간의 크기가 제약되면 L_2 정규화와 동일한 효과
 - 다음 그림의 왼쪽은 조기 종료를 했을 때 최적화 경로를, 오른쪽은 L_2 정규화를 했을 때 최적화 경로를 보여줌

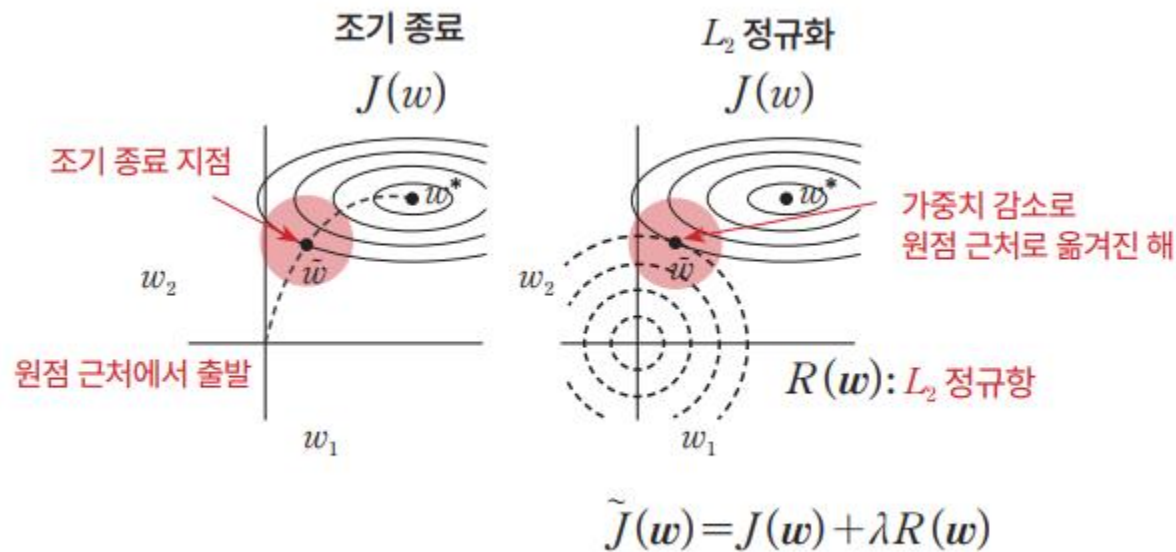


그림 5-32 조기 종료를 L_2 정규화 효과

5.6 데이터 증강

■ 데이터 증강이란?

- 모델은 복잡한데 그만큼 충분한 훈련 데이터가 제공되지 않으면 모델이 데이터를 암기해서 과적합이 생김
- 과적합을 막는 가장 근본적인 방법은 훈련 데이터의 양을 늘리는 것
- 다음 그림을 보면 훈련 데이터셋의 크기가 커질수록 훈련 오류는 증가하고 테스트 오류는 감소하여 두 곡선의 차가 나타내는 일반화 오류 또는 과적합의 정도는 줄어듦
- 따라서 훈련 데이터셋이 커질수록 과적합이 일어나지 않음

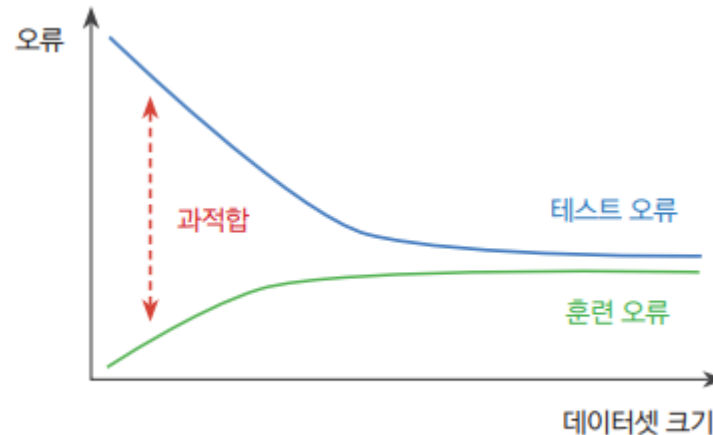


그림 5-33 훈련 데이터셋 크기와 과적합의 관계

5.6 데이터 증강

■ 데이터 증강 기법

- 데이터 증강 기법은 점점 다양해지고 고도화되고 있음
- 가장 기본적인 증강 방법은 훈련 데이터를 조금씩 변형해서 새로운 데이터를 만드는 방법
- 데이터 증강은 어떤 방식으로 실행해야 할까?
- 데이터를 증강해서 미리 훈련 데이터셋에 추가해둘 수도 있겠지만, 데이터를 확률적으로 변형하면 무한히 많은 변형이 생기므로 일반적으로는 훈련 과정에서 실시간으로 데이터를 증강

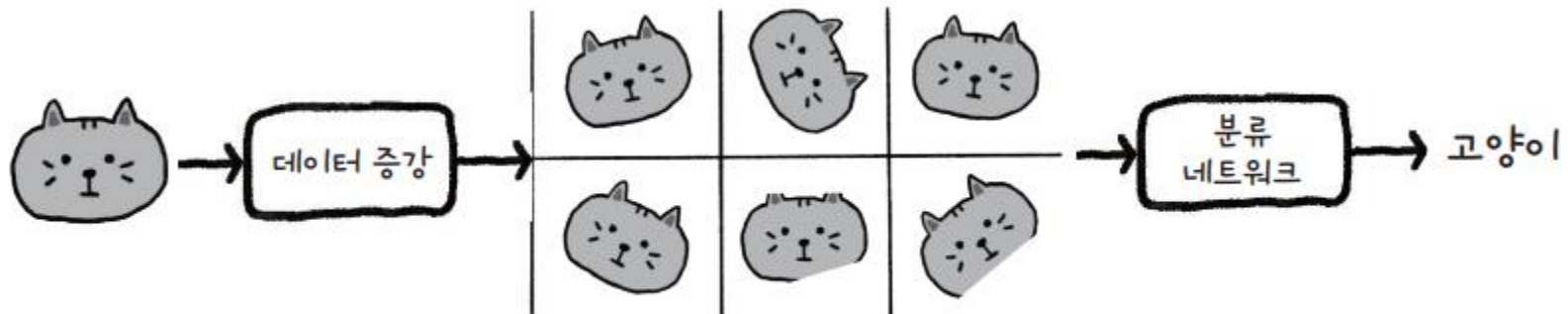


그림 5-34 데이터 증강이 결합된 학습 과정

5.6 데이터 증강

■ 클래스 불변 가정

- 데이터 증강을 할 때는 클래스 불변 가정을 따라야 함
- 클래스 불변 가정은 데이터를 증강할 때 클래스가 바뀌지 않도록 해야 한다는 가정
- 만일 데이터 증강과정에서 클래스의 결정 경계를 넘어서면 다른 클래스로 인식하므로, 각자의 결정 경계 안에서 데이터를 변형해야 함

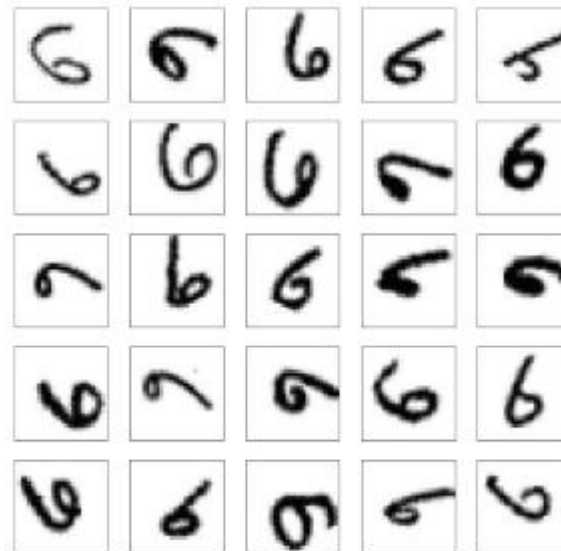


그림 5-35 클래스 6과 클래스 9의 경계가 애매한 경우

5.6 데이터 증강

■ 데이터 증강 방식 선택

- 데이터를 증강하는 방식은 데이터의 종류와 문제에 따라 매우 다양
- 만일 이미지 분류 문제를 푼다면 이미지 이동, 회전, 늘리기, 좌우/상하 대칭, 카메라 왜곡하기, 잡음 추가, 색깔 변환, 잘라내기, 떼어내기와 같은 다양한 이미지 변형 방법을 사용



그림 5-36 이미지 확장 기법들 (좌우 대칭, 색깔 변환)

5.7 배깅

■ 배깅이란?

- 앙상블은 여러 모델을 실행해서 하나의 강한 모델을 만드는 방법
- 개별 모델의 성능은 약하지만, 약한 모델이 모여서 하나의 팀을 이루면 성능이 좋은 강한 모델이 될 수 있음
- 앙상블 기법 중 배깅은 독립된 여러 모델을 동시에 실행한 뒤 개별모델의 예측을 이용해서 최종으로 예측하는 방법
- 배깅이 정규화 방법인 이유는 모델이 서로 독립일 때 예측 오차가 모델 수에 비례해서 줄어들기 때문

5.7 배깅

■ 배깅의 원리

- 배깅은 모델의 종류와 관계없이 다양한 모델로 팀을 구성할 수 있음
- 같은 종류의 모델로 팀을 구성하기도 하고 다른 모델로 팀을 구성하기도 함
- 단, 성능을 높이려면 모델 간에 독립을 보장해야만 함
- 모델의 독립성을 보장하기 위해 훈련 데이터를 부트스트랩하여 모델별로 부트스트랩 데이터를 생성
- 부트스트랩 데이터는 훈련 데이터에서 복원 추출해서 훈련 데이터와 같은 크기로 만듦

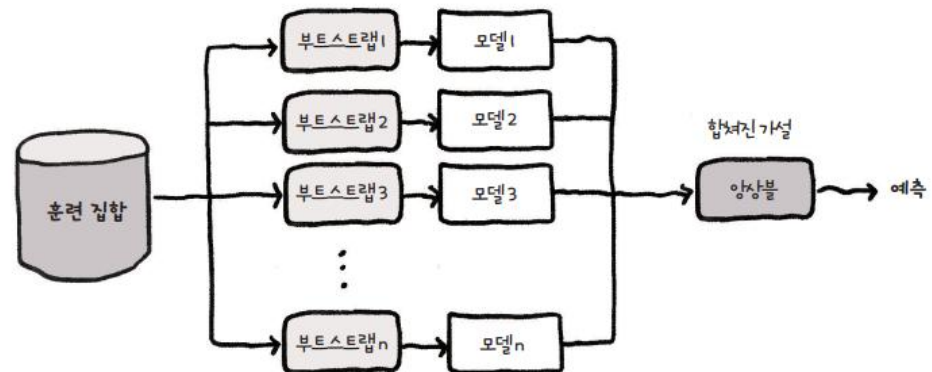


그림 5-37 배깅 과정

5.7 배깅

- 여러 모델의 추론 결과를 이용하는 배깅의 추론 방식
 - 추론 단계에서는 개별 모델의 결과를 집계해서 예측
- 신경망 모델로 배깅할 때 다른 점
 - 신경망 모델로 배깅할 때는 부트스트랩을 사용하지 않아도 됨

5.7 배깅

■ 배깅의 정규화 효과

- 배깅의 정규화 효과를 확인하기 위해 개별 모델의 예측 오차가 배깅에서 어떻게 줄어드는지 살펴보자.
- k 개의 회귀 모델로 배깅한다고 가정

$$\epsilon_i \sim \mathcal{N}(0, \Sigma) (i=1, 2, \dots, k)$$

$$\mathbb{E}[\epsilon_i^2] = v, \mathbb{E}[\epsilon_i \epsilon_j] = c$$

- 회귀 모델에서 배깅의 예측은 개별 모델의 평균으로 계산하므로 배깅의 오차 ϵ_b 는 개별 모델 오차의 평균

$$\epsilon_b = \frac{1}{k} \sum_{i=1}^k \epsilon_i$$

5.7 배깅

- 배깅의 오차 크기
 - 배깅을 해서 오차가 줄었는지 확인하기 위해 오차 ϵ_b 의 분산을 구해 보자.

$$\begin{aligned}\text{Var}(\epsilon_b) &= \mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^k \epsilon_i\right)^2\right] \\ &= \frac{1}{k^2} \mathbb{E}\left[\sum_{i=1}^k \left(\epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j\right)\right] \\ &= \frac{1}{k^2} \left[\sum_{i=1}^k \mathbb{E}(\epsilon_i^2) + \sum_{i=1}^k \sum_{j \neq i} \mathbb{E}(\epsilon_i \epsilon_j) \right] \\ &= \frac{1}{k^2} kv + \frac{1}{k^2} k(k-1)c \\ &= \frac{1}{k} v + \frac{k-1}{k} c\end{aligned}$$

5.7 배경

- 모델이 서로 독립이 아니라면?
 - 먼저 개별 모델 간에 상관성이 커서 공분산과 분산이 같다고 가정
 - 앞의 식에 $c = v$ 를 대입하면 다음과 같이 배경 오차는 단일 모델의 오차와 같은 분산을 가짐
 - 따라서 모델 간의 상관성이 높으면 배경했을 때 오차가 줄어들지 않음

$$\text{Var}(\epsilon_b) = \frac{1}{k}v + \frac{k-1}{k}v = v$$

5.7 배경

- 모델이 서로 독립이라면?
 - 반대로 모델이 서로 독립이라면 공분산이 0이므로 $c = 0$ 가 되어 앙상블 오차는 다음과 같이 모델 수에 비례해서 줄어듦

$$\text{Var}(\epsilon_b) = \frac{1}{k}v + \frac{k-1}{k}0 = \frac{v}{k}$$

5.8 드롭아웃

■ 드롭아웃이란?

- 드롭아웃은 미니배치를 실행할 때마다 뉴런을 랜덤하게 잘라내서 새로운 모델을 생성하는 정규화 방법
- 드롭아웃은 하나의 신경망 모델에서 무한히 많은 모델을 생성하는 배깅과 같음
- 드롭아웃은 계산 시간이 거의 들지 않고 다양한 모델에 쉽게 적용할 수 있는 강력한 정규화 기법

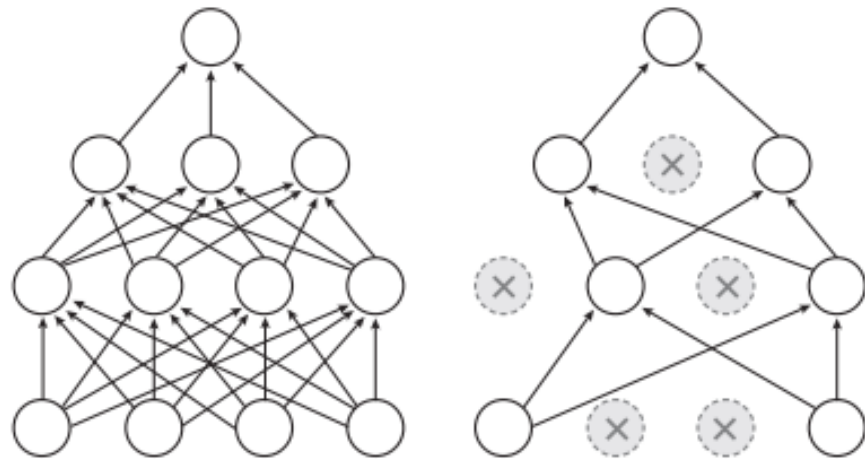


그림 5-39 드롭아웃의 적용^[40]

5.8 드롭아웃

- 드롭아웃은 배깅보다 성능이 좋을까?
 - 드롭아웃으로 무한히 많은 모델을 생성할 수 있다면, 드롭아웃은 배깅보다 성능이 좋을까?
 - 배깅은 서로 독립된 모델을 병렬로 실행해서 예측 오차를 줄이지만, 드롭아웃은 모델 간에 파라미터를 공유하기 때문에 모델 간에 상관성이 생김
 - 따라서 모델 간의 독립성을 전제로 하는 배깅보다 더 좋은 성능을 갖기는 어려움
 - 하지만 드롭아웃은 배깅보다 실용적

5.8 드롭아웃

■ 학습 단계

- 드롭아웃은 미니배치를 실행할 때마다 뉴런을 랜덤하게 잘라내서 매번 다른 모델을 생성
- 뉴런을 드롭아웃할 때는 뉴런의 50% 이상은 유지되어야 함
- 드롭아웃은 입력 계층과 은닉 계층에 적용하며, 뉴런을 유지할 확률은 입력 뉴런은 0.8, 은닉 뉴런은 0.5 정도로 지정
 - 입력 데이터와 신경망 구조에 따라 값은 달라짐

5.8 드롭아웃

- 드롭아웃을 하면 어떤 모델이 생성될까?
 - 다음 그림과 같이 입력 뉴런이 2개, 은닉 뉴런이 2개인 신경망에 드롭아웃을 적용해보자.
 - 뉴런의 50%는 유지하고 나머지 50%는 드롭아웃 한다면 24개에 해당하는 16종류의 모델이 생성될 수 있음

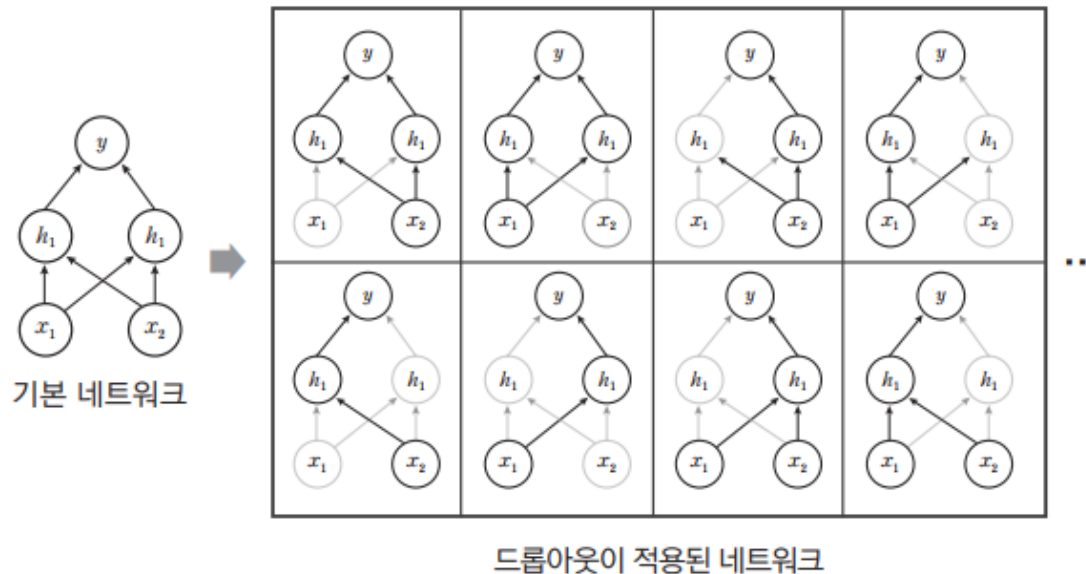


그림 5-40 드롭아웃으로 생성된 다양한 모델

5.8 드롭아웃

- 이진 마스크를 활용한 뉴런의 드롭아웃
 - 훈련 단계에서 드롭아웃을 어떻게 적용하는지 살펴보자.
 - 먼저 미니배치를 실행할 때마다 계층별로 뉴런의 이진 마스크를 생성
 - 이진 마스크는 뉴런별 드롭아웃 여부를 나타내며, 뉴런의 마스크값이 1이면 뉴런은 유지되고 마스크값이 0이면 드롭아웃 됨
 - 계층 l 의 드롭아웃은 다음과 같이 정의

$$\mathbf{a}^{(l)} = \text{activation}(\mathbf{W}^{(l)T} \mathbf{x}^{(l)} + \mathbf{b}^{(l)})$$

$$\mathbf{r}^{(l)} \sim \text{Bern}(p)$$

$$\tilde{\mathbf{a}}^{(l)} = \mathbf{a}^{(l)} \odot \mathbf{r}^{(l)}$$

5.8 드롭아웃

■ 추론 단계

- 추론 단계에서는 뉴런을 드롭아웃하지 않고 훈련 과정에서 확률적으로 생성했던 다양한 모델의 평균을 예측해야 함
- 모델 평균을 어떻게 구하는지 살펴보자.

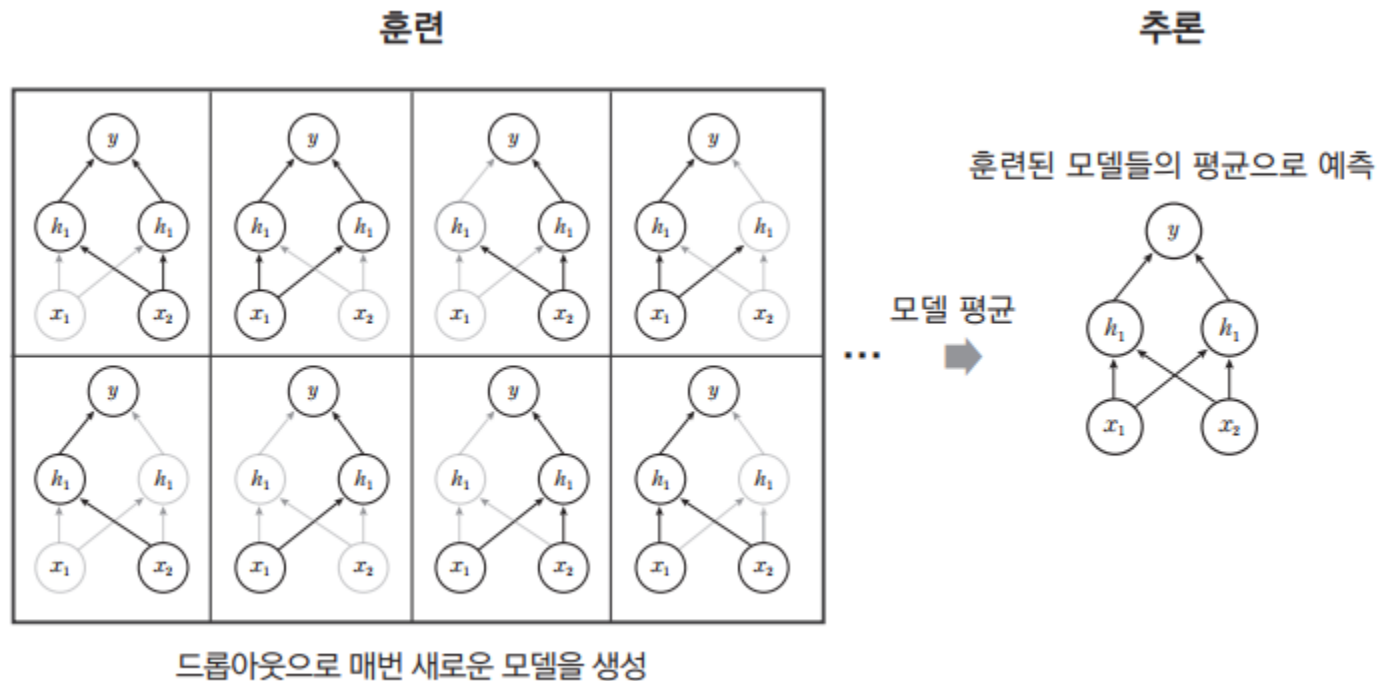


그림 5-41 훈련 과정에서 생성했던 모델들의 평균으로 예측

5.8 드롭아웃

- 무한히 많은 모델의 평균 계산
 - 다음과 같이 입력 뉴런이 2개이고 출력 뉴런이 하나인 신경망에
 - 뉴런을 유지할 확률 $p = 0.5$ 로 드롭아웃을 적용한다고 해보자.

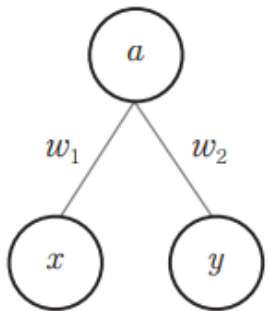


그림 5-42 드롭아웃 예제

$$w_1x + w_2y$$

$$w_1x + 0y$$

$$0x + 0y$$

$$0x + w_2y$$

$$\begin{aligned}\mathbb{E}[a] &= \frac{1}{4}(w_1x + w_2y) + \frac{1}{4}(w_1x + 0y) + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2y) \\ &= \frac{1}{2}(w_1x + w_2y)\end{aligned}$$

$$\mathbf{a}^{(l)} = \text{activation}(\mathbf{W}^{(l)T} \mathbf{x}^{(l)} + \mathbf{b}^{(l)}) \times p$$

5.8 드롭아웃

■ 역드롭아웃

- 뉴런을 유지할 확률 p 를 반드시 추론 시점에 곱해야 할까?
- 훈련 시점에 각 계층의 출력을 미리 p 로 나눠 두면 원래의 추론 코드를 그대로 사용할 수 있음
- 이런 아이디어를 적용한 방법이 역 드롭아웃
- 다음 식과 같이 훈련 단계에서 각 계층 l 의 출력을 p 로 나눠주면 됨

$$\mathbf{a}^{(l)} = \text{activation}(\mathbf{W}^{(l)T} \mathbf{x}^{(l)} + \mathbf{b}^{(l)})$$

$$\mathbf{r}^{(l)} \sim \text{Bern}(p)$$

$$\tilde{\mathbf{a}}^{(l)} = (\mathbf{a}^{(l)} \odot \mathbf{r}^{(l)}) / p$$

5.9 잡음 주입

■ 잡음 주입이란?

- 데이터나 모델을 확률적으로 정의할 수 있다면 더 정확하게 추론할 수 있음
- 하지만 애초에 데이터나 모델이 확률적으로 정의되지 않았다면 간단히 잡음을 넣어서 확률적 성질을 부여할 수 있음

■ 잡음 주입 방식

- 확률적 성질을 부여하고 싶은 대상에 따라서 잡음을 주입하는 형태도 다양함
- 확률적 성질은 입력 데이터, 은닉 계층에서 추출된 특징, 모델 가중치, 레이블 등에 부여할 수 있으며 그만큼 다양한 정규화 기법이 잡음 주입 방식에 포함됨

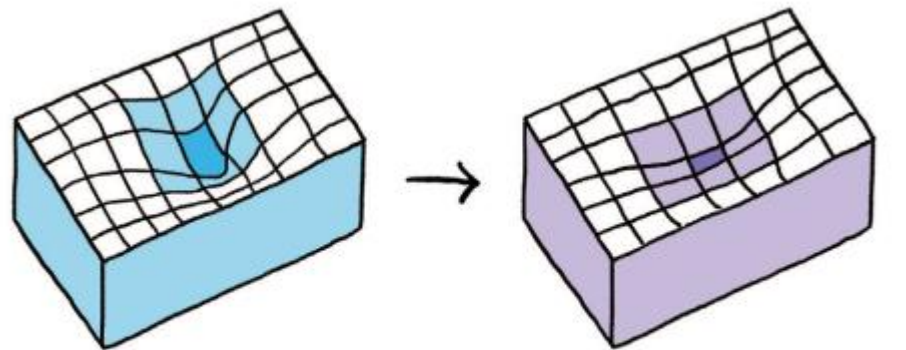
5.9 잡음 주입

- 입력 데이터에 잡음 주입
 - 입력 데이터에 잡음을 추가하는 것은 데이터 증강 기법에 해당
 - 입력 데이터에 아주 작은 분산을 갖는 잡음을 넣으면 가중치 감소와 동일한 정규화 효과를 얻을 수 있음
- 특징에 잡음 주입
 - 은닉 계층에서 추출된 특징에 잡음을 넣는 것은 데이터가 추상화된 상태에서 데이터 증강을 하는 것으로 생각해 볼 수 있음
 - 추상화된 데이터에 확률적 성질을 부여하기 때문에 객체와 같은 상대적으로 의미 있는 단위로 데이터 증강이 일어나서 성능이 크게 향상됨

5.9 잡음 주입

■ 모델 가중치에 잡음 주입

- 가중치에 잡음을 주입하는 대표적인 예가 드롭아웃
- 드롭아웃에서 뉴런을 제거할 때 확률적으로 가중치를 조절하기 때문에 가중치에 잡음을 넣는 것
- 가중치에 잡음을 직접 더해서 가중치에 조금씩 변화를 주기도 함
- 다음 그림과 같이 원래 최소 지점 주변의 경사가 가파르더라도 가중치에 잡음을 넣어주면 최소 지점 주변이 평지로 변해서 새로운 데이터에 대한 일반화 성능이 향상됨



경사가 가파른 위치에 있는 최소

경사가 평평한 위치에 있는 최소

그림 5-43 평평한 위치에 있는 최소^[44]

5.9 잡음 주입

- 최적해가 평지 위에 있으면 좋은 이유
 - 최소 지점이 평지에 있으면 왜 일반화 성능이 향상할까?
 - 다음 그림의 그래프와 같이 2차원 함수로 설명해 보면 직관적으로 이해할 수 있음

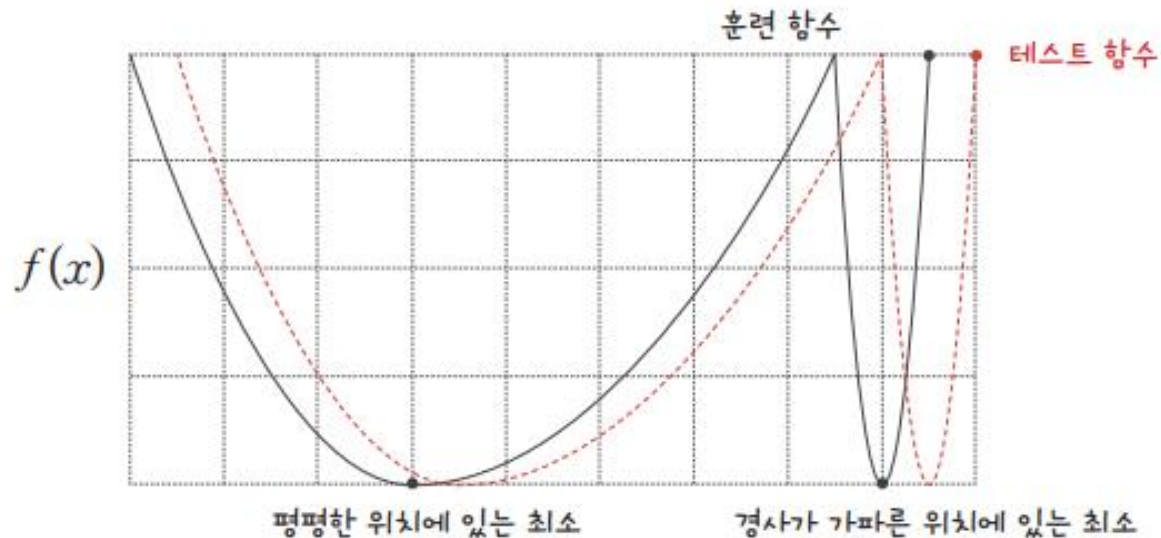


그림 5-44 평지 위의 최소와 일반화 오류의 관계

5.9 잡음 주입

■ 소프트 레이블링

- 분류 문제에서 훈련 데이터의 레이블에 오차가 있다고 해보자.
- 이런 경우 레이블에 일정한 크기의 오차를 반영해 주면 더 정확하게 예측할 수 있음
- 레이블이 정확하지 않다면?
 - 레이블이 정확하지 않다면 어떤 현상이 발생할까?
 - 분류 모델의 학습 과정에서 모델이 타깃 클래스의 확률을 1로, 나머지 클래스의 확률을 0으로 예측하도록 만들 것임
 - 그런데 레이블에 오차가 있다면 모델이 정확히 1이나 0으로 예측하지 못하기 때문에 계속해서 일정량의 손실이 발생하고 최적화가 이루어지지 않을 수 있음

5.9 잡음 주입

■ 소프트 레이블링

– 레이블에 잡음 주입

- 레이블이 정확하지 않다면 ϵ 만큼의 오차가 있다고 가정하고, 타깃 클래스의 확률은 ϵ 만큼 작게 만들고 나머지 클래스들의 확률은 ϵ 을 배분해서 확률을 부여함
- 이런 방식을 소프트 레이블링이라고 함
- 레이블에 오차를 반영한 후 학습하면 모델 성능이 높아짐
- 소프트 레이블링은 80년대부터 지금까지 사용하고 있는 정규화 방법

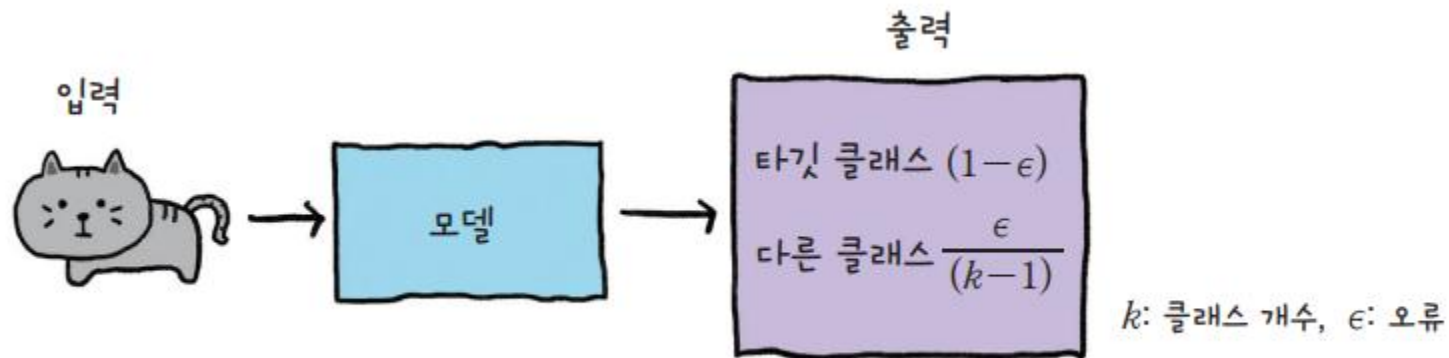


그림 5-45 소프트 레이블링

경청해주셔서 감사합니다.
