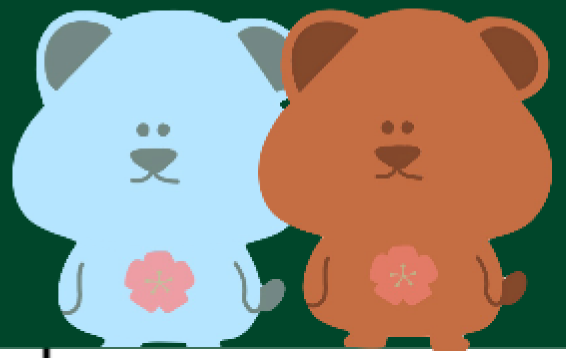
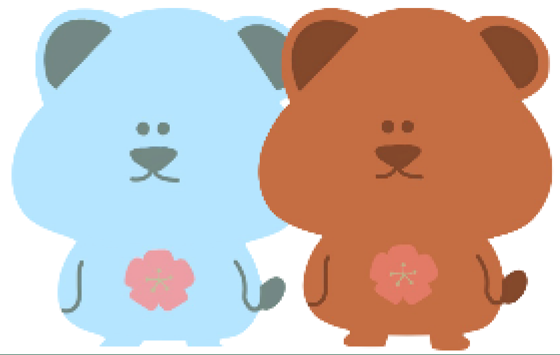
2. 신경망 네트워크의 정규화



Euron 6기 중급팀 강민정

EWHA
EURON

01 정규화



EWHA EURON

정규화, norm 개념

- high variance 해결 방안: training set 늘리기 // But 많은 비용 필요

  ⇒ **정규화 (Regularization)** !

- **norm** : 벡터의 크기(magnitude)의 측정 방법

1. **L1 norm** : 벡터의 모든 성분의 절댓값의 합
2. **L2 norm** : 두 벡터(점) 사이의 직선 거리

e.g. $x = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$

- $\|x\|_1 = |2| + |3| + |4|$
- $\|x\|_2 = \sqrt{(2)^2 + (3)^2 + (4)^2}$

EWHA EURON

로지스틱 회귀에서의 정규화

- 로지스틱 회귀의 (원래) 비용함수 $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

w, b: 매개변수

$w \in \mathbb{R}^{n_x}$: n차원의 매개변수 벡터

$b \in \mathbb{R}$: 실수

- **L2 정규화** (일반적으로 L2 정규화 사용)

  : 기존 비용함수에 L2 norm 추가

  - $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$
  - $\|w\|_2^2$: w^2의 L2 norm = (w의 L2 norm)^2

    $= \sum_{j=1}^{n_x} w_j^2 = w^T w$

- **L1 정규화**

  : 기존 비용함수에 L1 norm 추가

  - W will be sparse(희소해짐) = 0 값 많아짐

    → 모델 압축에 도움

  - $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_1$

    $\|w\|_1$: w의 L1 norm

    $= \sum_{j=1}^{n_x} |w_j|$

<u>⇒ 비용함수 & 가중치(w)를 모두 줄이는 방향으로 학습 진행</u>

- λ : 정규화 매개변수 – 하이퍼 파라미터 (설정 필요)
  - 주로 개발 세트/교차 검증 세트 사용
  - 다양한 값 시도 → 최적 값 찾기

Q. w에 관한 정규화만 시행하는 이유?

A. b에 관한 정규화도 가능하나 주로 생략

(대부분의 매개변수가 w에 존재하기 때문에 실질적 차이 X)

신경망에서의 정규화

- 기존 비용함수에 L2 정규화 추가

$$J(w^{[1]}, b^{[1]}, \dots, w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

- **Frobenius norm** : 행렬의 L2 norm = 행렬의 원소 제곱의 합

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}} \sum_{i=1}^{n^{[l-1]}} (w_{ij}^{[l]})^2$$

경사하강법 구현

- 기존

$$dw^{[l]} = \frac{\partial J}{\partial w^{[l]}}$$ = (from 역전파) : w에 대응하는 J의 편미분 값

$$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$$

- 정규화 추가

$dw^{[l]}$ = (from 역전파) + $\frac{\lambda}{m} w^{[l]}$

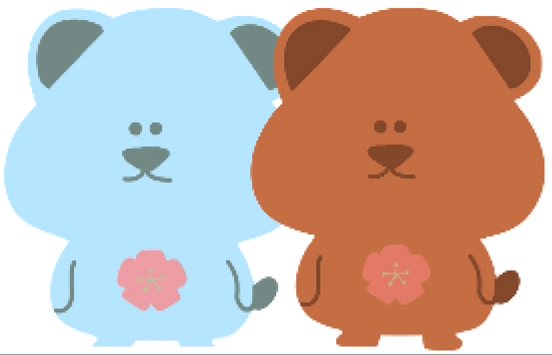
$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$

$= w^{[l]} - \alpha[$(from 역전파)$+ \frac{\lambda}{m} w^{[l]}]$

$= w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]} - \alpha$ (from 역전파)

$= (1 - \frac{\alpha\lambda}{m}) w^{[l]} - \alpha$ (from 역전파) ← weight에 1보다 작은 값 곱해짐: **weight decay**

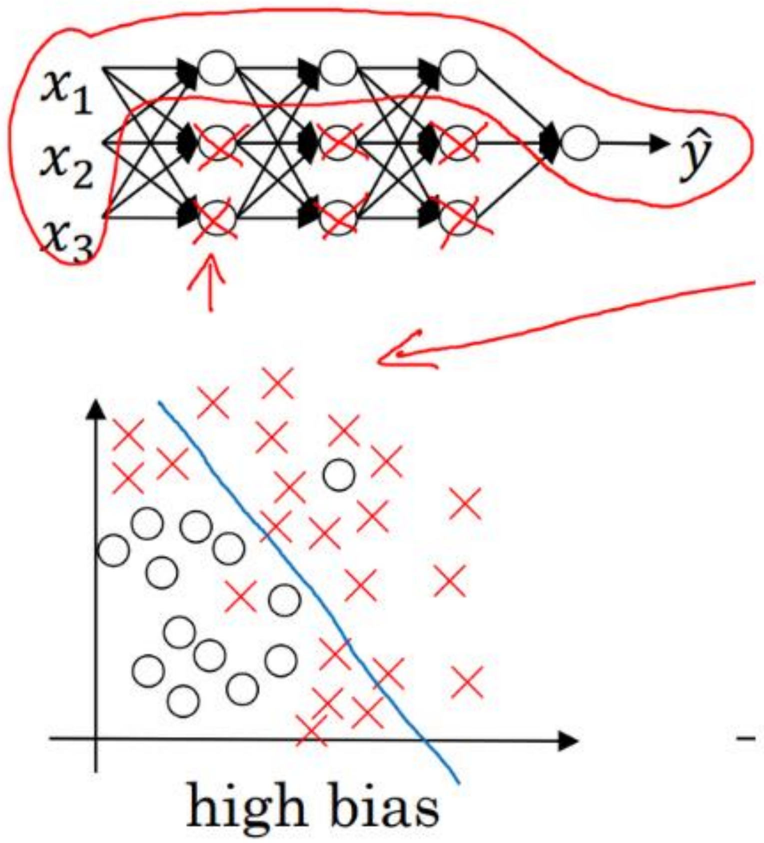
02 정규화가 과대적합 줄이는 이유



정규화가 과대적합 줄이는 이유

1. 가중치 행렬을 0에 가깝게 설정

비용 함수 $J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$



- $\lambda \uparrow \rightarrow w \approx 0$

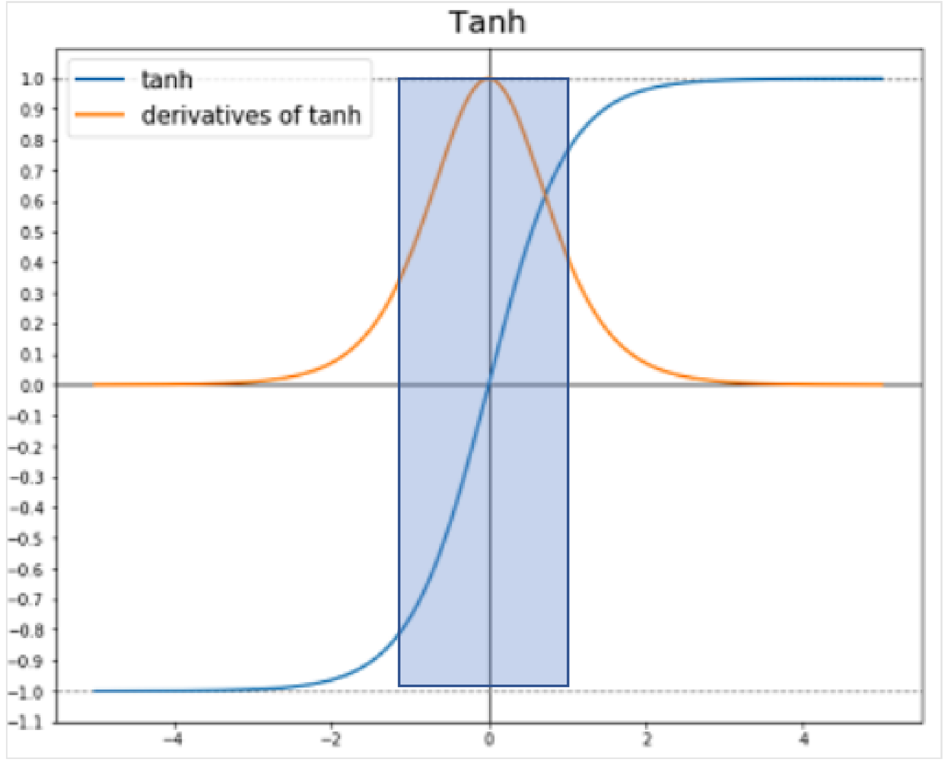
  → 많은 은닉 유닛의 영향력 ⬇ (0에 가깝)

  ⇒ 간단하고 작은 신경망 만들어짐

- over/underfitting 사이의 적절한 λ 찾아야 함!

2. tanh 활성화 함수에서의 선형 네트워크 생성

$g(z) = tanh(z)$



- z가 작으면, tanh의 선형 영역을 사용하게 됨

λ ⬆ → $w^{[l]}$ ⬇ → $z^{[l]}$ ⬇ $(z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]})$

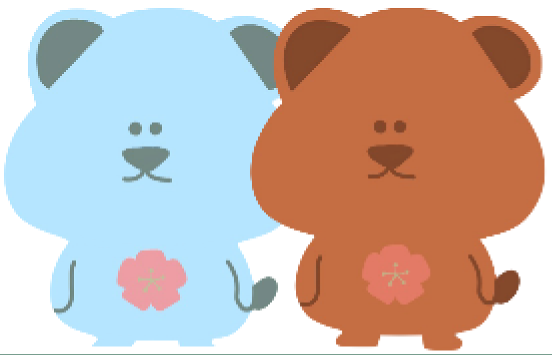
- g(z)가 1차 함수에 가까워짐

→ 모든 층도 선형 회귀에 가까운 거의 직선의 함수

→ 전체 네트워크도 선형 함수만을 계산

⇒ 과대적합과 같이 복잡한 결정 내릴 수 없음

03 드롭아웃 정규화



EWHA EURON

Drop out

- **드롭아웃** : 신경망의 각 층에 대해 노드 삭제할 확률 설정 ⇒ 간소화된 네트워크로 학습

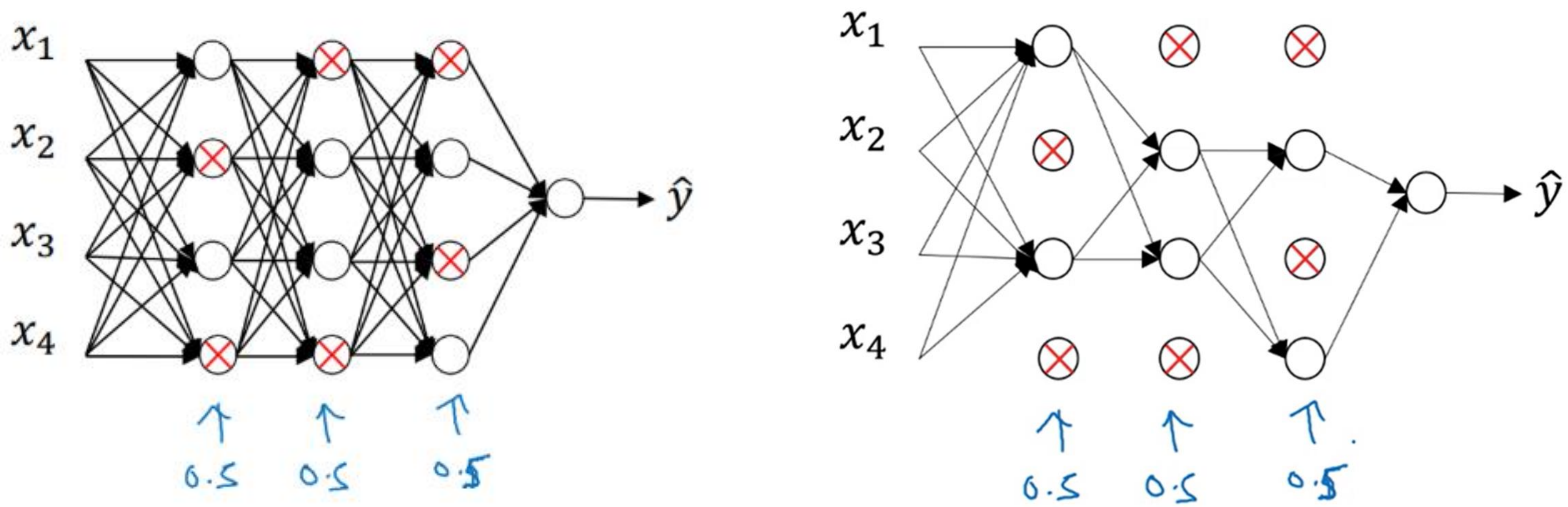
- **과정**

: 무작위로 노드 & 해당 노드의 모든 링크 삭제 ⇒ 네트워크 간소화

→ 하나의 샘플 역전파 훈련

→ 각 훈련 샘플에서 노드 삭제 & 역전파 훈련 반복

⇒ 모든 샘플에서 더 작은 네트워크를 훈련



inverted drop out(역 드롭아웃)

1.
```
# illustrate with layer L=3
keep_prob = 0.8
d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep_porb
```

- `np.random.rand()` : 입력한 shape에 맞는 난수 생성
  - **범위**: 0~1
  - **분포**: uniform
- d3 **벡터**: 어떤 노드를 0으로 만들지 결정 — 정방향/역방향 모두

  `keep_prob` 보다 작으면 True(1) ↔ 크면 False(0)

  ⇒ P(1) = 0.8 ↔ P(0) = 0.2

📌 각 훈련 샘플에서의 반복마다 **0이 되는 은닉 유닛은 무작위로 달라져야** 함

2.
```
a3 = np.multiply(a3, d3) # a3 *= d3
```

- `np.multiply` : array의 elementwise multiply 수행

  a3과 d3의 element끼리 곱셈 (a3 & d3 : 동일 shape)
- d3에서 0 → 대응되는 a3에서도 곱해지면 0 됨

  ⇒ (1 - `keep_prob`) 의 확률로 a3의 element가 0으로 바뀜

3. ⭐ Inverted dropout ⭐

```
a3 /= keep_prob
```

a3 — 50개의 유닛 * **차원**: (50, 1), 벡터화하면 (50, m)

→ 평균적으로 10개 units 삭제 (0의 값 가짐)

→ $z^{[4]} = w^{[4]}a^{[3]} + b^{[4]}$: 그대로 두면 **z의 기댓값 감소**

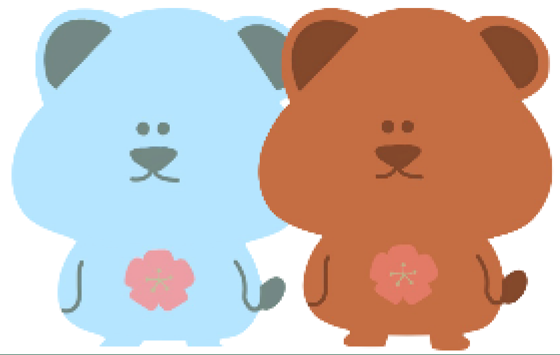
→ z의 기댓값 (+ **a의 기댓값**) **유지**하기 위해 `keep_prob` 으로 나눔

At test time

- 드롭아웃 사용 X (예측해야 하기 때문에 랜덤 결과 X)
- 역 드롭아웃의 효과

  : 테스트에서 스케일링 매개변수 추가할 필요 없어 편리

04 드롭아웃의 이해



드롭아웃의 이해

드롭아웃이 정규화로 잘 작동하는 이유

- dropout에 의해 input이 매번 무작위로 삭제됨

  → unit이 어떤 input feature에도 의존할 수 없음

  = 한 input에 매우 큰 가중치 부여하지 않음

  = 각 input에 가중치 분산

  → W(가중치)의 norm의 제곱값이 감소

  ⇒ overfitting 방지에 도움

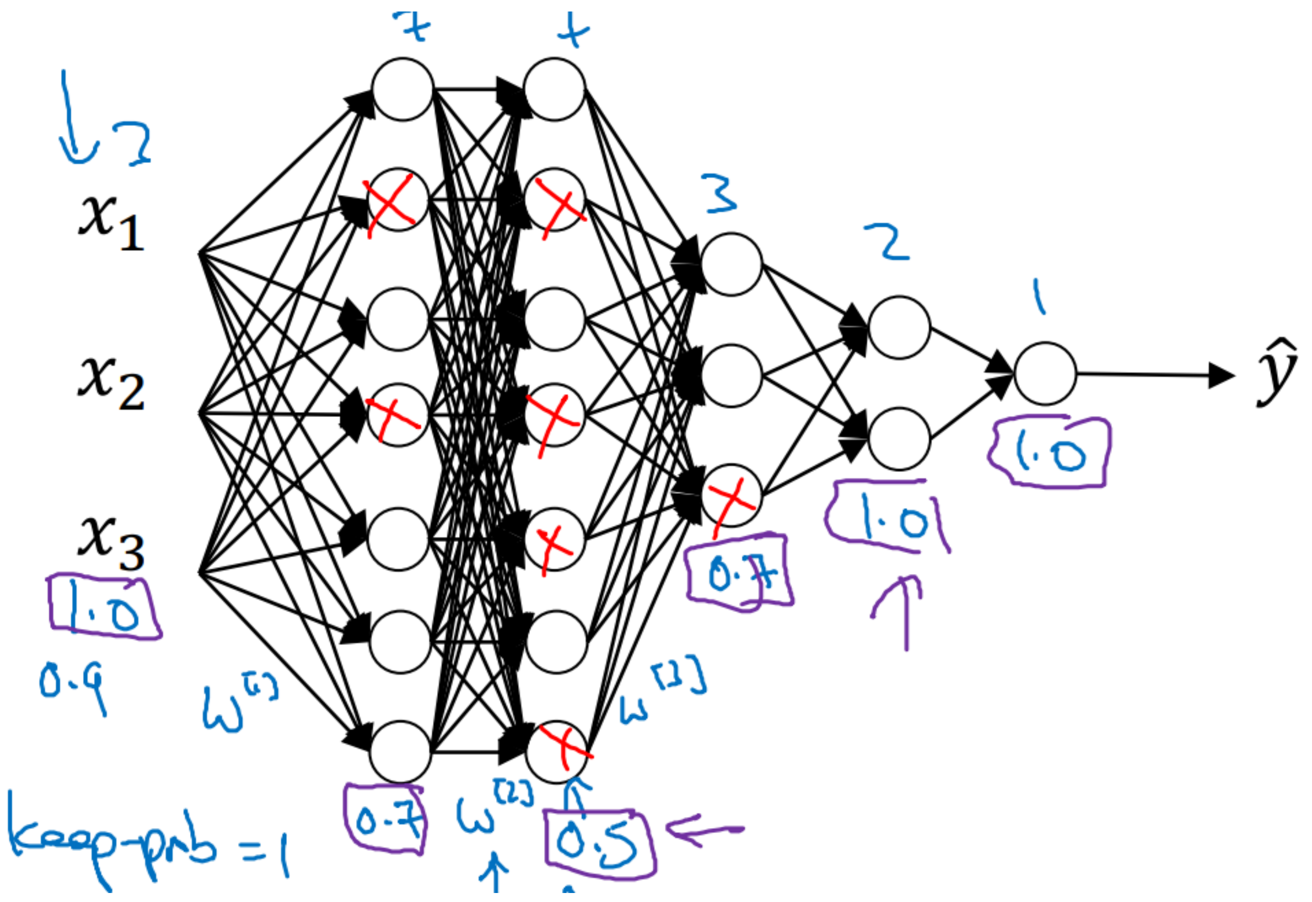
단점

- 드롭아웃 적용 시 비용함수 정의 어려움

⇒ 드롭아웃 효과 없이(keep_prob=1) 코드 실행

J가 단조감소하는지 확인 후 드롭아웃 적용

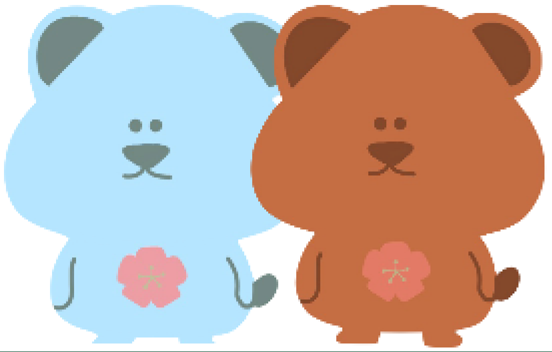
드롭아웃 구현



- **keep_prob**: 각 층마다 다르게 설정 가능
  - 매개변수 많은 층 = overfitting 우려 높음

    : 상대적으로 낮은 확률 부여
  - overfitting 우려 적은 층 : 더 높은 값 설정 가능
  - input layer에도 설정 가능하나, 거의 하지 않음

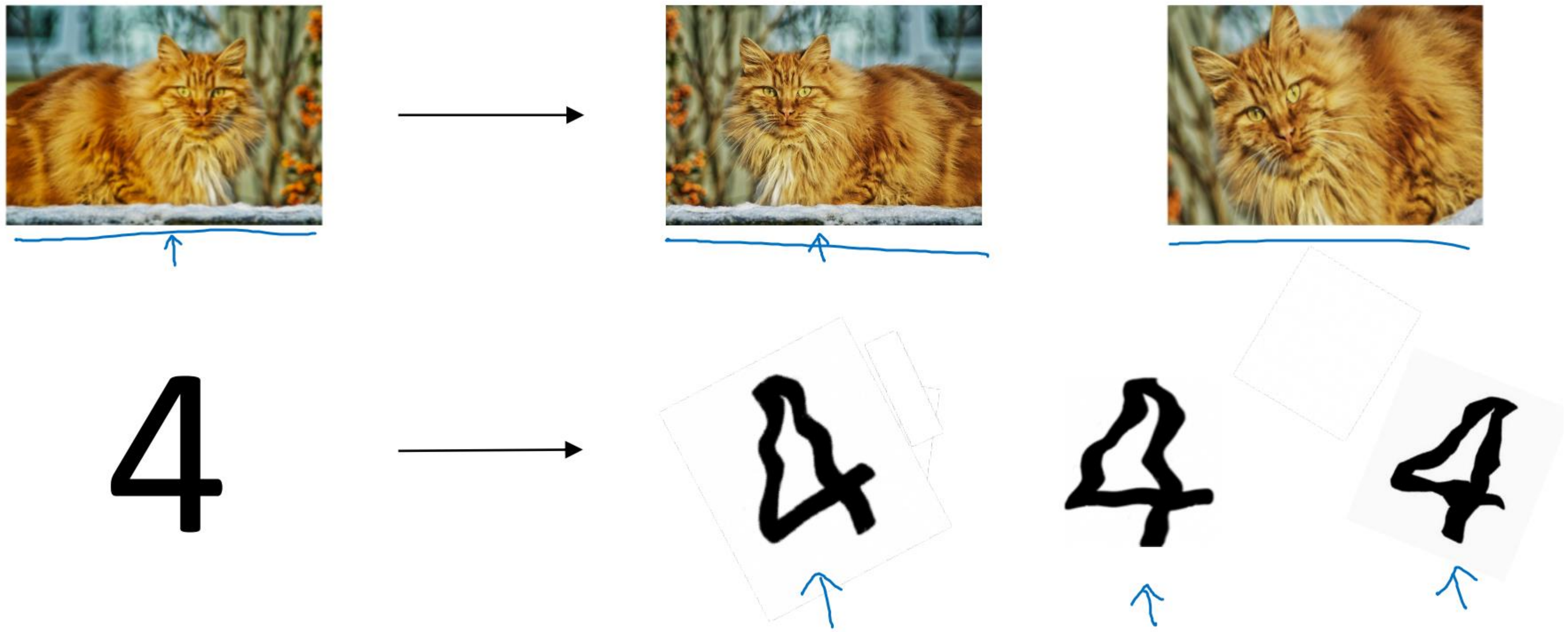
05 다른 정규화 방법들



데이터증식 (Data augmentation)

e.g. 이미지 데이터 부족한 경우

- 기존 이미지의 대칭, 확대, 왜곡, 회전 등 ⇒ 무작위적인 이미지 편집
    - (-) 중복 샘플 많아져 new & independent 샘플보다 적은 정보만 추가
    - (+) 적은 비용으로 데이터 수집 가능



조기종료 (Early stopping)

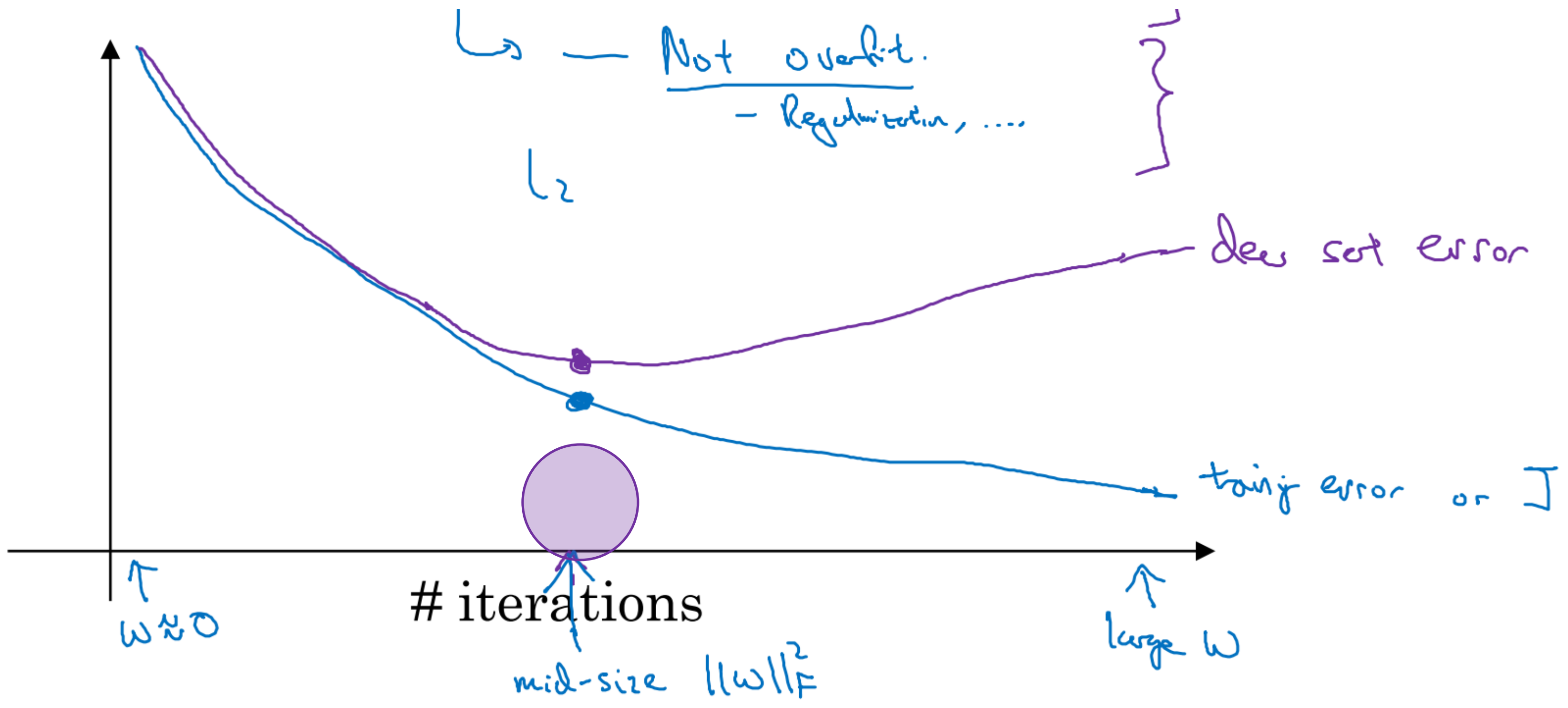
경사하강법 plot

- <u>**훈련오차/비용함수**</u> : 단조 감소 형태로 그려져야 함

  \+ (조기종료) <u>**dev set error**</u>도 확인

- **dev set error** : 감소하다가 증가 → 최저점에서 신경망이 가장 잘 작동

  ⇒ 훈련 멈추고 해당 지점을 최적으로 설정



- **가중치 w** : (초기) 0에 가까운 작은 값 → 점점 커짐

  ⇒ 조기종료로 멈추면 w가 중간 값 가짐

- L2 정규화와 유사
- w에 대해 더 작은 norm 가지는 신경망 선택

  ⇒ less overfitting 만듦

조기종료 (Early stopping)

단점

- ML의 2가지 작업
  ① 비용함수 최적화
  ② overfitting 방지
- 두 가지는 별개의 작업, 별개의 도구 필요
  = Orthogonalization (직교화)

장점

- 한 번의 시도만으로 큰/중간/작은 w값을 얻을 수 있음

But, 조기종료는 두 작업을 섞음 ⇒ 독립적으로 수행 X

경사하강법 일찍 중단 (작업 2) → 비용함수 최적화 멈춤 (작업 1)

⇒ 대안: L2 정규화 사용

퀴즈 리뷰

퀴즈 9

✓ 9. L1 정규화와 L2 정규화의 가장 큰 차이점은 무엇인가요? * 1/1

- ○ L1은 모델의 가중치에 대한 제곱의 합에 페널티를 부여하고, L2는 가중치의 절대값의 합에 페널티를 부여한다.
- ◉ L1은 특정 가중치를 0으로 만들 수 있어 sparse하기 때문에 피처 선택의 효과가 있다. ✓
- ○ L1과 L2 모두 모델의 가중치를 0으로 만들어, 모델의 복잡도를 감소시킨다.
- ○ L2는 모든 가중치를 동시에 0으로 만든다.

① L1과 L2 설명 반대

1. L1 norm : 벡터의 모든 성분의 절댓값의 합
2. L2 norm : 두 벡터(점) 사이의 직선 거리

e.g. $x = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$

- $\|x\|_1 = |2|+|3|+|4|$
- $\|x\|_2 = \sqrt{(2)^2 + (3)^2 + (4)^2}$

L1 정규화

- 정규화 추가 비용함수 : $C = C_0 + \frac{\lambda}{n}\sum_w |w|$
- 미분 → 상수값

⇒ 업데이트 시 <u>w의 크기와 상관없이</u> 상수 빼면서 진행

- 작은 가중치들은 0으로 수렴. 중요한(큰) 가중치만 남음.

= sparse

L2 정규화

- 가중치 업데이트 시 가중치 크기가 영향 미침
  - 큰 값 : 강하게 규제 ↔ 작은 값 : 약하게 규제

⇒ 완전히 0으로 없애지는 않음

$dw^{[l]}$ = (from 역전파) + $\frac{\lambda}{m} w^{[l]}$

$$\begin{aligned} w^{[l]} &:= w^{[l]} - \alpha dw^{[l]} \\ &= w^{[l]} - \alpha[\text{(from 역전파)} + \tfrac{\lambda}{m} w^{[l]}] \\ &= w^{[l]} - \tfrac{\alpha\lambda}{m} w^{[l]} - \alpha \text{ (from 역전파)} \\ &= (1 - \tfrac{\alpha\lambda}{m}) w^{[l]} - \alpha \text{ (from 역전파)} \end{aligned}$$

퀴즈 10

✓ 10. 정규화 시 람다(lambda)값과 모델 학습에 대한 설명으로 옳은 것을 **모두** 골라주세요. *1/1

- [x] 람다 값이 작을수록 모델은 데이터에 대해 더 복잡하게 학습하며, 오버피팅의 위험이 증가한다. ✓
- [ ] 람다 값이 작을수록 모델의 가중치에 대한 페널티가 증가하여, 오버피팅을 방지하는데 도움이 된다.
- [ ] 람다 값이 클수록 모델은 데이터에 대해 더 복잡하게 학습하며, 오버피팅의 위험이 증가한다.
- [x] 람다 값이 클수록 모델의 가중치에 대한 페널티가 증가하며, 모델의 복잡도를 제한하여 오버피팅의 위험이 감소한다. ✓

$$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$$
$$= w^{[l]} - \alpha\left[(\text{from 역전파}) + \frac{\lambda}{m} w^{[l]}\right]$$
$$= w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]} - \alpha\,(\text{from 역전파})$$
$$= \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha\,(\text{from 역전파})$$

- <u>람다 값 커지면</u>

→ weight에 곱해지는 값이 1보다 매우 작아짐

→ 가중치가 더 작은 값으로 업데이트 (= 패널티 증가)

⇒ 모델 복잡도 제한됨 = 오버피팅 위험 감소

- <u>람다 값 작아지면</u>

→ weight에 곱해지는 값이 1보다 약간 작아짐 (1과 비슷)

→ 가중치 업데이트 느림

⇒ 더 복잡하게 학습 = 오버피팅 위험 증가

THANK YOU