2. 신경망 네트워크의 정규화

▼ 목차

정규화

> 🔹 높은 분산으로 신경망이 데이터를 과대적합하는 문제가 의심된다면, 가장 먼저 **정규화**를 시도하자!

- 정규화를 추가함으로써 과대적합을 막고 신경망의 분산을 줄일 수 있음
- 더 많은 훈련 데이터를 얻는 것도 하나의 방법이지만 비용이 많이 들어가게 됨

1️⃣ Regularization in Logistic Regression

- 로지스틱 회귀의 비용함수 J에 정규화 매개변수 λ를 추가한 것은 다음과 같다. (L2 Regularization)

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y^{(i)}}, y^{(i)}) + \frac{\lambda}{2m}\left\|w\right\|_2^2 \quad (+\frac{\lambda}{2m}b^2)$$

- b에 대한 정규화?

  🔷 매개변수 w는 높은 차원의 매개변수 벡터로, 높은 분산을 가질 때 특히 많은 매개변수를 갖는 반면 b는 하나의 숫자이므로 거의 모든 매개변수는 b가 아닌 w에 존재한다. 따라서 b를 정규화하는 것은 실질적인 차이가 없으므로 생략할 수 있다.

🚀 정규화 항

- w norm에 $\frac{\lambda}{2m}$가 곱해진 형태
  - λ는 정규화 매개변수로, 과대적합을 피하기 위해 최적화를 수행해야 하는 하이퍼파라미터 중 하나
  - m 앞에 곱하는 2는 스케일링 상수

💥 L2 Regularization & Euclidean Norm

- 가장 일반적인 정규화로, 네트워크 훈련 시 가장 많이 사용됨

> 정규화 항: $\frac{\lambda}{2m}\left\|w\right\|_2^2$

> 유클리드 노름(L2 노름): $\left\|w\right\|_2 = \sum_{j=1}^{n_x}\sqrt{w_j^2}$
>
> - 따라서 $\left\|w\right\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$로 쓸 수도 있음

💥 L1 Regularization & Manhattan Norm

- w 벡터를 희소하게(벡터 안에 0이 많아짐) 만들 수 있어 모델 압축 시 사용됨

> 정규화 항: $\frac{\lambda}{2m}\left\|w\right\|_1$

> 맨해튼 노름(L1 노름): $\left\|w\right\|_1 = \sum_{j=1}^{n_x}|w_j|$

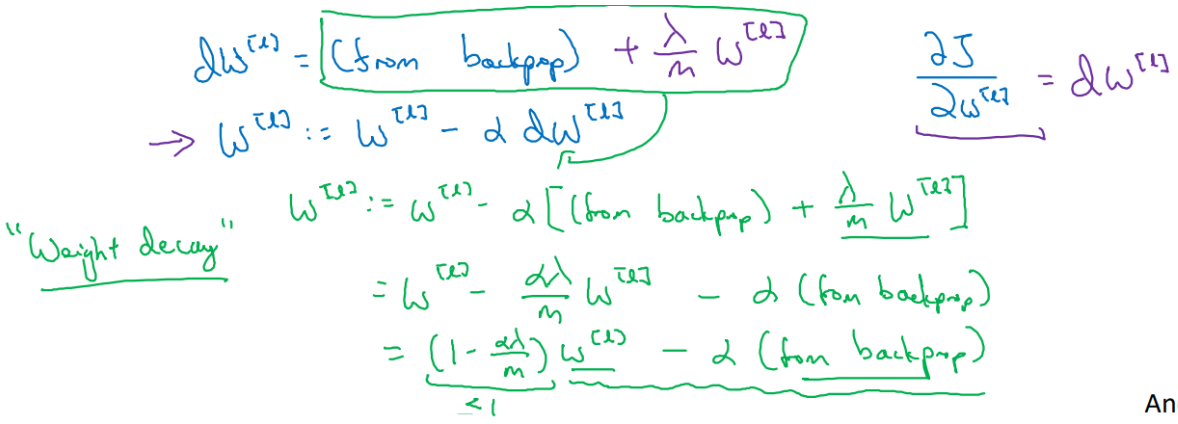
💥 Frobenius Norm

- 행렬의 L2 노름을 이르는 말 (행렬의 원소 제곱의 합)

> 정규화 항: $\frac{\lambda}{2m}\sum_{l=1}^{L}\left\|w^{[l]}\right\|_F^2$

> 프로베니우스 노름: $\left\|w^{[l]}\right\|_F^2=\sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}(w_{ij}^{[l]})^2$

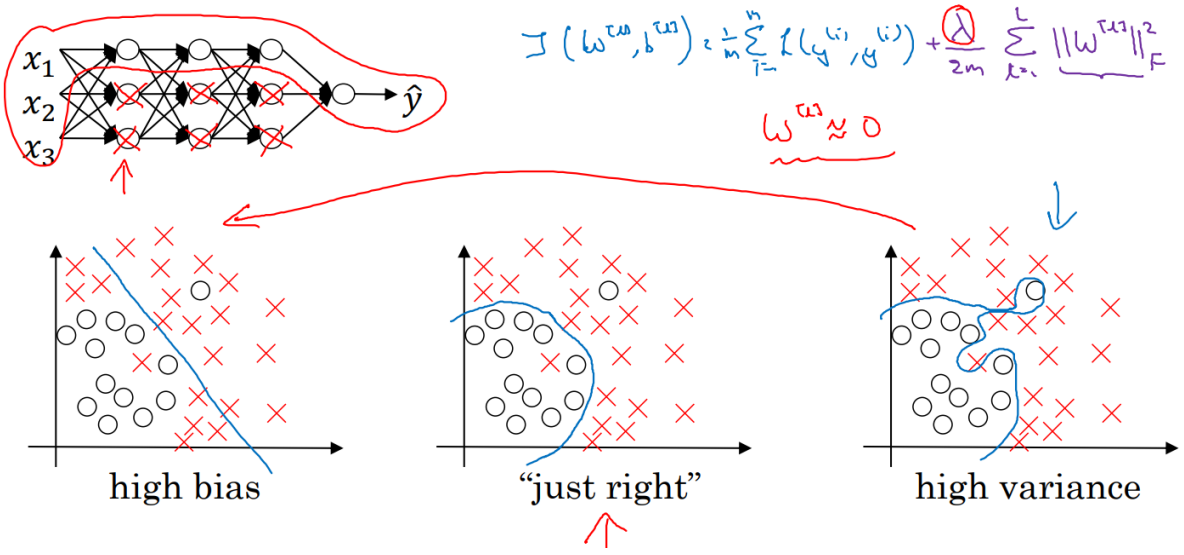
2️⃣ Gradient Descent using Regularization



Andr

왜 정규화는 과대적합을 줄일 수 있을까?

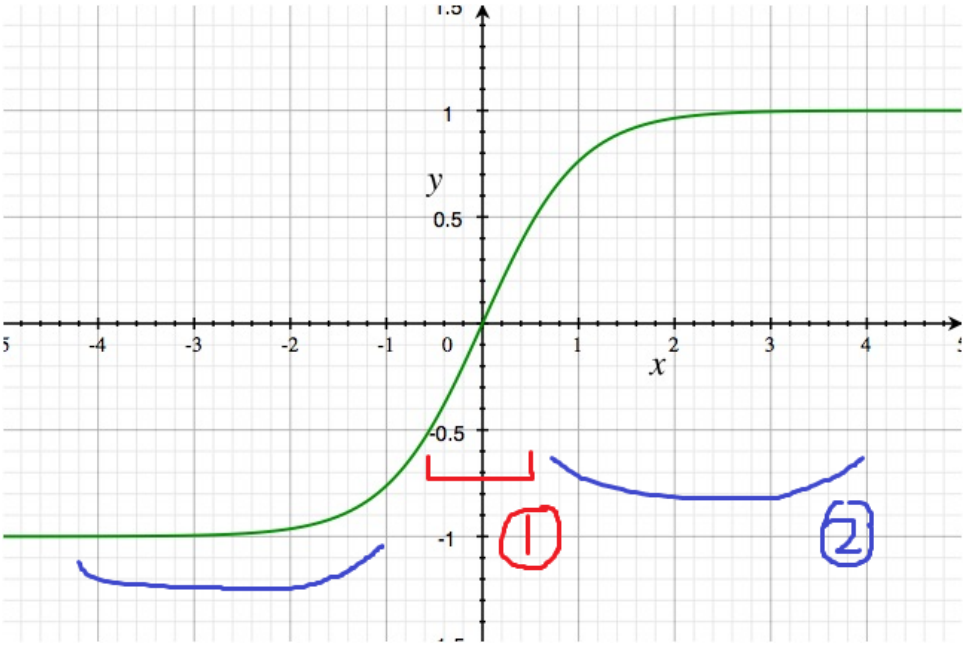
1️⃣ Intuition 1



- 정규화 시 λ를 크게 만듦으로써 가중치 행렬 w를 0에 가깝게 설정할 수 있음
- 많은 은닉 유닛을 0에 가까운 값으로 설정하여 은닉 유닛의 영향력을 줄이게 됨
- 신경망을 훨씬 더 간단하고 작게 만들 수 있음 (로지스틱 회귀 유닛에 가까워짐)
    - high variance → high bias로 만들어주는 것으로 볼 수 있음

2️⃣ Intuition 2

- $\tanh$ 활성화 함수를 사용하는 경우 ($g(z) = \tanh(z)$)



    - z가 아주 작은 경우 (작은 범위의 매개변수를 갖는 경우)
        - $\tanh$ 함수의 선형 영역을 사용하게 됨 (그림의 ①)
    - z의 값이 더 작아지거나 커질 경우
        - 활성화함수는 선형을 벗어나게 됨 (그림의 ②)

정리!

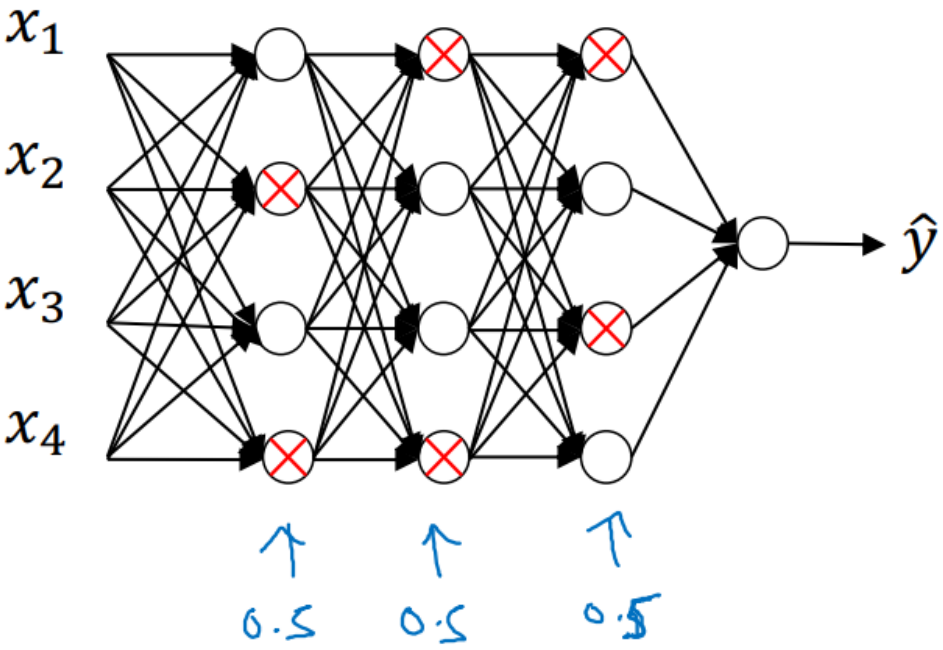
- 정규화 매개변수인 **λ가 커질 때** 비용함수가 커지지 않으려면 상대적으로 **w가 작아짐**
- w가 작으면 **z도 상대적으로 작은 값**을 가지며,
  z가 작은 값을 가질 때 $g(z)$**는 거의 1차원 함수**가 됨
- 따라서 모든 층이 선형 회귀처럼 거의 직선의 함수를 갖게 되며,
  모든 층이 선형이면 전체 네트워크도 선형이 됨

드롭아웃 정규화

- 🔹 드롭아웃 방식은 신경망의 각각의 층에 대해 노드를 삭제하는 확률을 설정하는 것!

1️⃣ Dropout Regularization

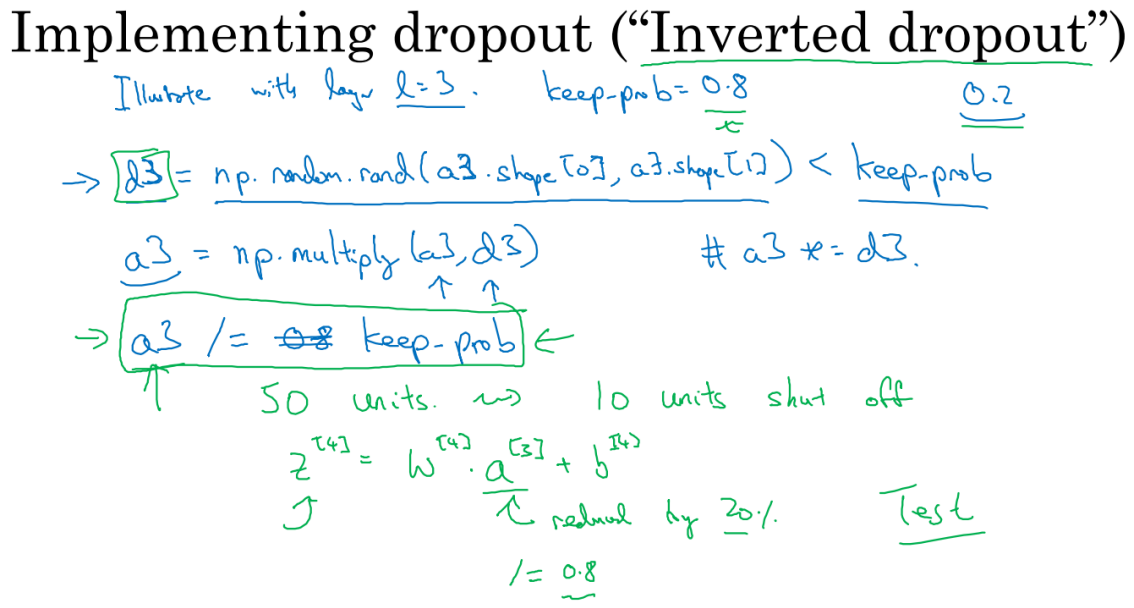
- 각각의 층에 대해 각각의 노드마다 동전을 던지는 경우
  (0.5의 확률로 해당 노드를 유지하고 0.5의 확률로 노드를 삭제)



- 동전을 던진 후 위와 같이 노드를 삭제
- 삭제된 노드의 들어가는 링크와 나가는 링크 또한 삭제
  - 더 작고 간소화된 네트워크가 됨!
- 감소된 네트워크에서 하나의 샘플을 역전파로 훈련시킴
- 다른 샘플에 대해서도 같은 방식을 적용, 각각의 훈련 샘플에 대해서 감소된 네트워크를 사용해 훈련시키게 됨

2️⃣ Implementing Dropout (Inverted Dropout)

- 층이 3인 경우의 역 드롭아웃 예시



- `keep_prob` : 어떤 은닉 유닛이 삭제되지 않을 확률
  - `keep_prob`이 0.8이라는 것은 어떤 은닉 유닛이 삭제될 확률이 0.2라는 것
- 역 드롭아웃의 효과는 테스트에서 드롭아웃을 구현하지 않아도 활성화 기대값의 크기는 변하지 않기 때문에 테스트 할 때 스케일링 매개변수를 추가해주지 않아도 됨

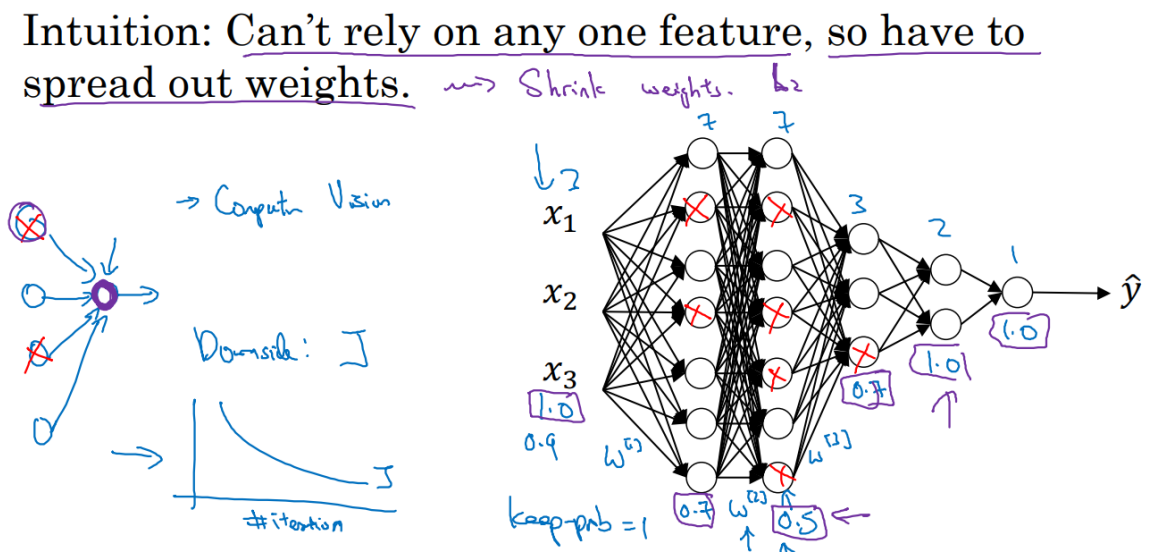
3️⃣ Making Predictions at Test Time

> 🔹 테스트 시에는 드롭아웃을 사용하지 않음!

- 테스트는 예측을 하는 것으로, 결과가 무작위로 나오는 것을 원하지 않으므로 테스트에 드롭아웃을 구현하는 것은 노이즈만 증가시킬 뿐임
- 이론적으로 무작위로 드롭아웃된 서로 다른 은닉 유닛을 예측 과정에서 여러 번 반복해 그들의 평균을 낼 수도 있지만, 컴퓨터적으로 비효율적이며 이 과정과 거의 비슷한 결과를 냄

드롭아웃의 이해

1️⃣ Why does drop-out work?



- 신경망의 유닛을 무작위로 삭제함으로써 모든 반복마다 더 작은 신경망에서 작업하게 되며, 더 작은 신경망을 사용하는 것이 정규화와 비슷한 효과를 주게 됨
- 입력 유닛을 무작위로 삭제함으로써 특성 또는 특성의 고유한 입력이 무작위로 바뀌도록 함
  - 특정 입력에 모든 것을 걸 수 없도록 (큰 가중치를 부여할 수 없도록) 함
  - 나머지 입력 각각에 가중치를 분산시키게 하여 가중치의 노름의 제곱값을 감소시킴

  🔹 결과적으로 드롭아웃은 가중치를 줄이고 과대적합을 막는 데 도움이 되므로 L2 정규화와 비슷한 효과를 갖는다고 볼 수 있음

- 드롭아웃과 L2 정규화의 차이점
  - L2 정규화는 다른 가중치에 적용됨
  - L2 정규화는 서로 다른 크기의 입력에 더 잘 적응함

2️⃣ Implementing Dropout

- 과대적합의 우려가 높은 층일수록 낮은 `keep_prob`를 설정하게 됨
  - 과대적합의 우려가 없는 층은 `keep_prob`을 1로 설정해도 됨
    - 1.0의 값을 갖는 `keep_prob`은 모든 유닛을 유지하며, 해당 층에서는 드롭아웃을 사용하지 않음

    - L2 정규화에서 다른 층보다 더 많은 정규화가 필요한 층에서 매개변수 λ를 증가시키는 것과 유사함
- 이론적으로 드롭아웃을 입력 층에도 적용시킬 수 있음 (자주 사용되지 않음)
    - 0.9 또는 1.0의 `keep_prob`을 주로 사용함

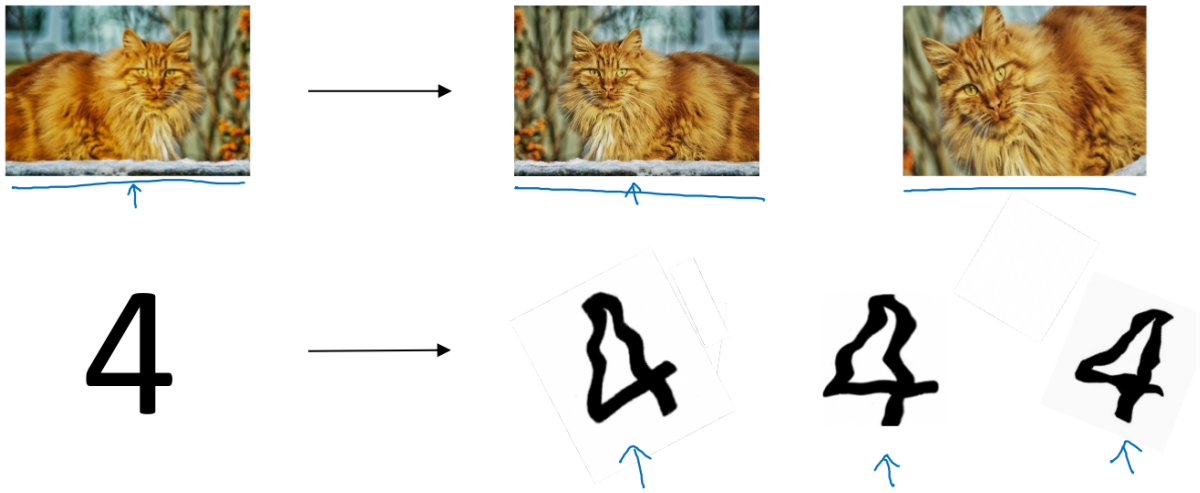
- 주의사항
    - 교차 검증을 위해 더 많은 하이퍼파라미터가 생김
    - 드롭아웃은 과대적합을 막는 데 도움을 주는 정규화 기법이므로, 네트워크 학습 시 과대적합의 문제가 생기기 전까지는 드롭아웃을 사용하지 않는 것이 맞음
    - 드롭아웃을 사용할 경우 **비용함수 J가 잘 정의되지 않음**
        - 모든 반복마다 무작위로 한 뭉치의 노드들을 삭제하게 되므로, 경사하강법의 성능을 이중으로 확인한다면 모든 반복에서 잘 정의된 비용함수 J가 하강하는지 확인하는 것이 어려워짐
        - `keep_prob`을 1로 설정하여, 드롭아웃 효과를 멈추고 코드를 실행시켜 J가 단조감소하는지 확인하는 방법을 활용할 수 있음

다른 정규화 방법들

🔹 L2 정규화, 드롭아웃 정규화와 더불어 신경망의 과대적합을 줄이는 다른 기법을 소개함

- 데이터 증강
- 조기 종료

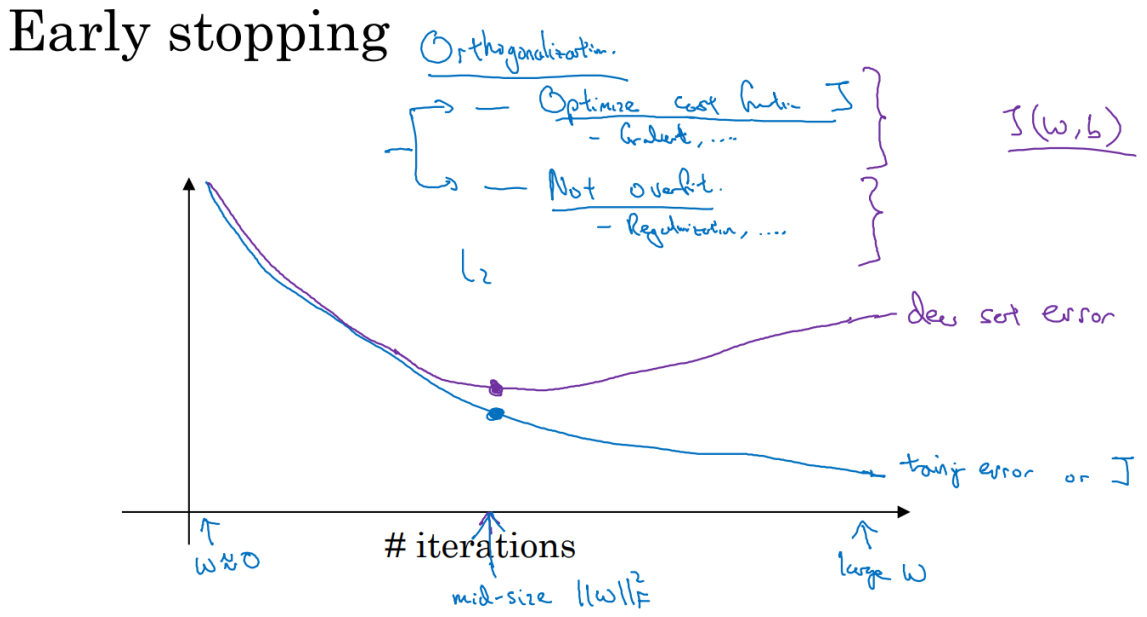
1️⃣ Data Augmentation



- 이미지를 무작위로 편집해 새로운 샘플을 얻음으로써 데이터 세트를 증가시키는 방법!

- 수평 방향으로 뒤집기, 회전시키기, 무작위로 확대하기, 왜곡하기 등
- 중복 샘플이 많아지므로 새로 독립적인 샘플을 얻는 것보다 좋은 방법은 아니지만, 컴퓨터적인 비용을 들이지 않고 데이터 세트를 늘릴 수 있음

2️⃣ Early Stopping



- 경사하강법을 실행하면서 훈련 세트에 대한 분류 오차(훈련 오차)를 그리거나 최적화하는 비용함수 J를 그리는 경우, 개발 세트 오차를 함께 그리게 됨

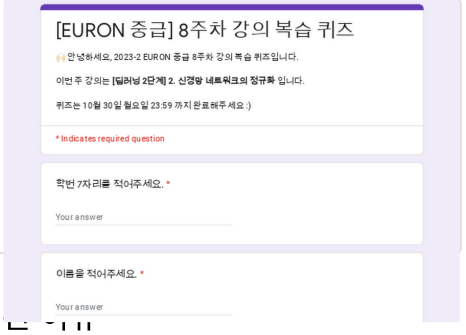
- 개발 세트 오차가 중간에 증가하는 경우 신경망 훈련을 중단, 해당 값을 최적으로 삼음
  - 매개변수 w가 중간 크기의 값을 갖는 상태
- 비용함수 J의 최적화와 과대적합을 방지하는 것을 동시에 수행하게 되므로, 문제를 더욱 복잡하게 만든다는 단점이 있음

출석퀴즈 오답노트

[EURON 중급] 8주차 강의 복습 퀴즈

🙌🏻 안녕하세요, 2023-2 EURON 중급 8주차 강의 복습 퀴즈입니다.
이번 주 강의는 [딥러닝 2단계] 2. 신경망 네트워크의 정규화 입니다.
퀴즈는 10월 30일 월요일 23:59 까지 완료해주세요 :)

https://docs.google.com/forms/d/e/1FAIpQLSeLJf_wDn6375hcTQr_46pJ5JgmY6WbylkFhGZnbZJKRSm4cg/viewform?usp=send_form



▼ 6. Data Augmentation이 효과적이라고 여겨지는 주[illegible]

- 더 많은 학습 데이터를 생성하여 모델의 학습 시간을 늘린다. (X)
- 존재하는 데이터의 다양한 변형을 통해 모델이 과적합을 방지하며 다양한 특징을 학습한다. (O)

▼ 7. Dropout 기법에 대한 설명으로 옳은 것

- 학습 데이터의 일부를 무작위로 제거하여 학습한다. (X)
- 과적합을 방지하기 위해 학습 중 무작위로 뉴런의 부분집합을 선택해 비활성화시킨다. (O)