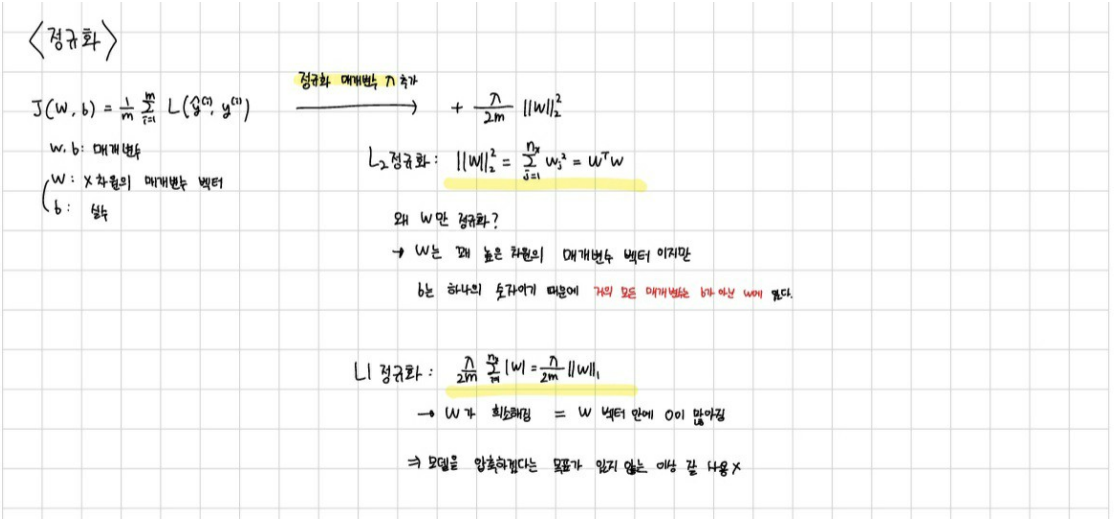
2. 신경망 네트워크의 정규화

1. 정규화

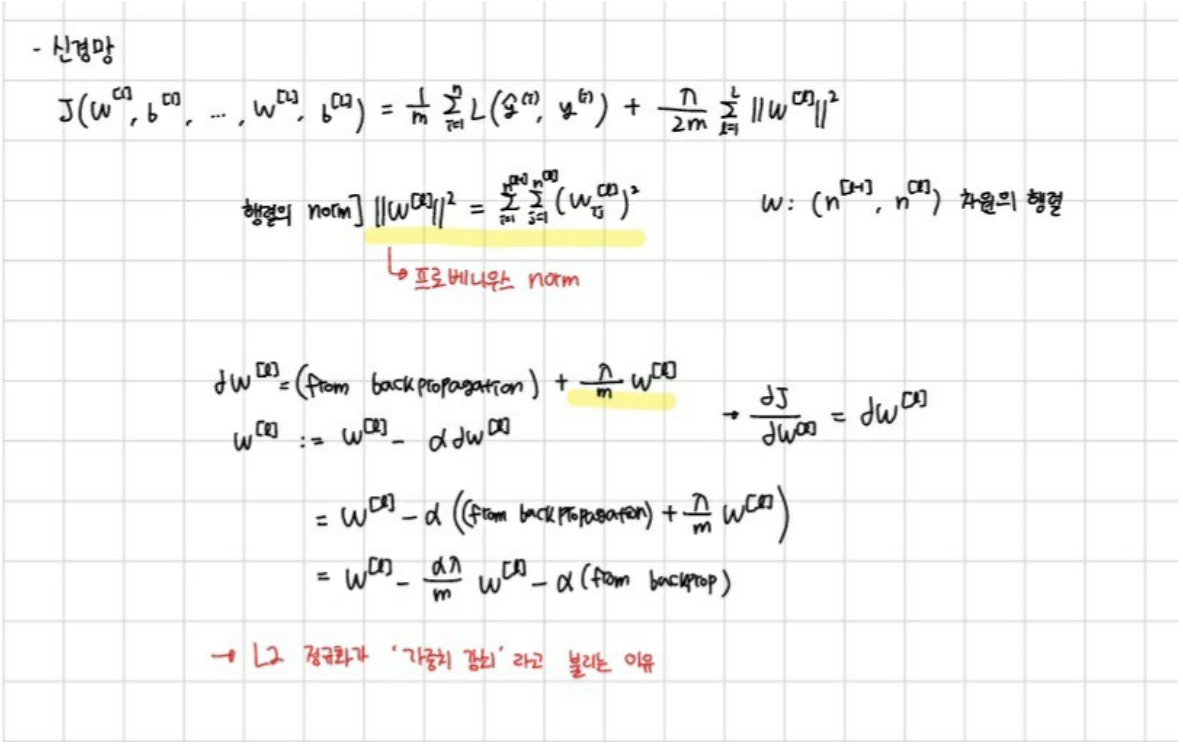
- 높은 분산으로 신경망이 데이터를 과대적합하는 문제가 의심되면

→ **정규화 시도**
→ 더 많은 훈련 데이터 얻기 : 비용 많이 필요.

- **정규화 작동 방법**
  - 로지스틱 회귀: 비용함수 J를 최소화하는 것(훈련 샘플의 개별적인 예측의 손실에 관한 함수)



- 신경망



2. 왜 정규화는 과대적합을 줄일 수 있을까요?

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

λ↑ : 가중치 행렬 W를 0에 상당히 가깝게 설정 가능

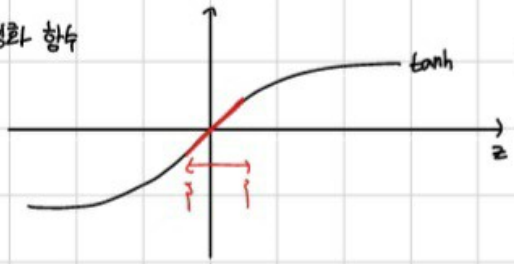
⇒ 은닉 유닛의 영향력↓

⇒ 더 간단하고 작은 신경망

⇒ 로지스틱 회귀 유닛에 가깝

간단한 NW : 과대적합 문제가 덜 발생

tanh 활성화 함수



λ↑ : 비용함수가 커지려면 상대적으로 $w^{[l]}$↓

$z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]}$

작은 범위에서 z가 작은 값 가지면 : g(z)는 거의 1차원 함수

≈ 모든 층이 선형 = 전체 NW도 선형

→ 과대 적합 가능성↓

- 구현 Tip
  - 경사 하강법의 반복 수에 대한 함수로 비용함수 설정하기
    → 비용함수 J가 경사 하강법의 반복마다 단조감소하기를 원함.

3. 드롭아웃 정규화

1. L2 외의 또 다른 정규화 기법: **드롭아웃**

- ex1. **과적합 신경망 훈련**
  - 드롭아웃 방식: 신경망의 각 층에 대해 노드를 삭제하는 확률 설정하는 것
    → 더 작은 네트워크가 된다. : 정규화의 효과를 주는 것처럼 보임
    = 각각의 샘플에서 더 작은 네트워크를 훈련시키는 방식

2. 구현 방식

- **역 드롭아웃**(가장 일반적인 기법)
  ex) keep_prob = 0.8로 설정(: 어떤 은닉 유닛이 삭제될 확률이 0.2)
  - **층 3에 대한 드롭아웃 벡터 d3 설정:**

  → d3 = np.random.rand(a3.shap3[0], a3.shape[1]) < keep_prob

  → 0.8의 확률로 True(1), 0.2의 확률로 False(0)

  - **층 3에 대한 활성화 a3 설정**:

  → a3 = np.multiply(a3,d3) # a3 *= d3
  → a3 /= keep_prob

  - 세 번째 은닉층에 50개의 유닛(뉴런)이 있다고 가정
    - a3차원 = (50,1)
    - 평균적으로 10개의 유닛이 삭제됨.
    - z^[4] = w^[4]*a^[3]+b^[4]
      → z^[4]의 기댓값을 줄이지 않기 위해 a^[3] 값을 0.8로 나눠줘야 함.
      : 필요한 20% 정도의 값을 다시 원래대로 만들 수 있기 때문

    = 기존에 삭제하지 않았을 때 활성화 값의 기대값으로 맞춰주기 위함

  - d벡터로 서로 다른 훈련 샘플마다 다른 은닉 유닛들을 0으로 만들게 됨.

3. **Test Time**
   : 명시적으로 드롭아웃을 사용X
   → 테스트에 드롭아웃 구현하는 것은 노이즈만 증가시킴

- 샘플 X(예측용) = a^[0]
- z^[1] = w^[1]*a^[0] + b^[1]
- a^[1] = g^[0]
- z^[2] = w^[2]*a^[1] + b^[2]

4. 드롭아웃의 이해

- 드롭아웃
  - 랜덤으로 노드를 삭제 시키기 때문에, 하나의 특성에 의존 하지 못하게 만듦으로서 **가중치를 다른 곳으로 분산 시키는 효과**
  - 입력 각각에 가중치 분산시키기 = 가중치의 norm 제곱값이 줄어들게 됨. = **과대적합 방지** 가능
    → L2 정규화와 비슷한 효과
- **keep_prob를 층마다 바꿀 수 있다.**
  - 과대적합의 우려가 큰 층(매개변수가 많은 층): keep_prob를 작게
  - 과대적합의 우려가 적은 층(매개변수가 적은 층) : keep_prob를 크게
  - 과대적합의 우려가 없는 층: keep_prob를 1로 설정해도 됨.
  - keep_prob = 1: 모든 유닛을 유지하고, 해당 층에서는 드롭 아웃을 사용X
  - 단점: 교차 검증을 위해 더 많은 하이퍼파라미터가 생김, 비용함수 J가 제대로 작동 X
- 구현 Tip
  - Computer Vision: 데이터가 부족한 경우가 많아서 드롭아웃 매우 자주 사용됨.
  - 경사하강법의 성능을 이중으로 확인하면, 비용함수 J가 하강하는지 확인하기가 어렵
    → **keep_prob를 1로 설정하여 드롭아웃 효과를 멈추고, 코드를 실행시켜 J가 단조감소하는지 확인**

5. 다른 정규화 방법들

1. **Data augmentation**

- 고양이 사진 분류기 예제
  - 더 많은 훈련 데이터가 과대 적합 해결 가능 → 비싼 비용 & 불가능
  - 수평 방향으로 뒤집은 이미지를 훈련 세트에 추가 → 훈련 세트 늘리기

  → 중복된 샘플이 많아짐.
  - 무작위로 이미지를 편집하여 새로운 샘플 얻기(회전, 확대, etc..)

  → 완전히 새로운 독립적인 고양이 사진 샘플보다 많은 정보를 추가하기는 어렵

- 데이터 증가는 정규화 기법과 비슷하게 사용될 수 있음

2. **Early stopping(조기 종료)**

- 개발 세트 오차가 감소하다가 증가하는 부분에서 신경망을 훈련시키는 것을 멈추고, 그 값을 최적으로 삼는다.
    - 개발 세트의 오차가 증가하기 시작하는 부분: 과대적화가 시작되는 시점.
    - 신경망이 개발 세트의 오차 저점 부근(=가장 잘 작동하는 점)에서 훈련 멈추기
- 신경망에서 많은 훈련을 시키지 않은 경우, 매개변수 w는 0에 가깝.
- L2 정규화와 유사하게, 매개변수 w에 대해 더 작은 norm 값을 가지는 신경망 선택.
- 단점
    - 비용함수를 최적화 시키는 작업과 과대적합하지 않게 만드는 작업을 섞어버려서 독립적이지 못하게 한다.
    - 최적의 조건을 찾지 못할 수 있다.
- 장점
    - 경사 하강법 과정을 1번만 실행하여, 작은 w, 중간 w, 큰 w의 값을 얻을 수 있다.