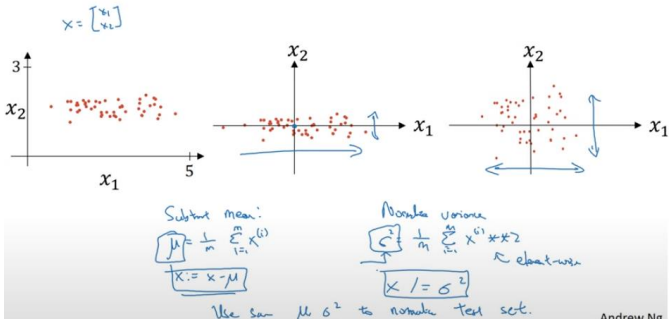
3.최적화 문제 설정

입력값의 정규화

- 정규화: 신경망의 훈련을 빠르게 할 수 있는 기법

<2개의 입력 특성이 있는 training set일 때>



1. 평균을 빼기

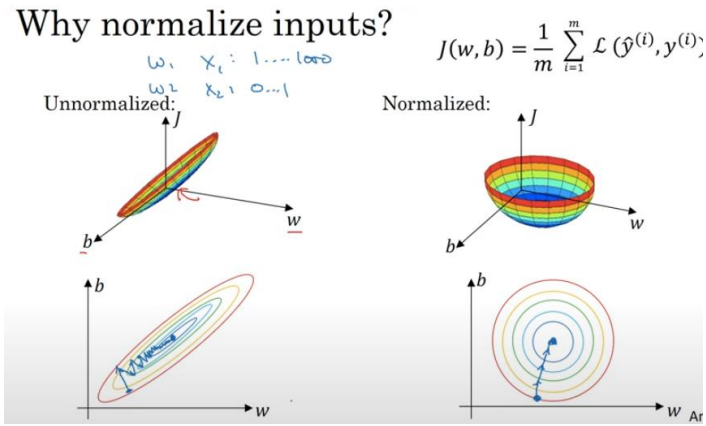
- 0의 평균을 갖게 될 때까지 움직이는 것

2. 분산을 정규화하기 (두 특성의 분산을 정규화하기)

- 테스트 세트를 정규화 할 때에도 사용한다.
- M랑 시그마 제곱을 테스트 세트에도 똑같이 이용한다.

★ 훈련세트와 테스트 세트를 다르게 정규화하면 안된다 ! 같은 변형을 거쳐야 한다

<왜 입력 특성을 정규화할까?>



- **정규화되지 않은 입력 특성**을 사용하면 **비용함수가 가늘고 긴 모양**
- 매우 작은 학습률 사용하게 된다
- 입력 특성이 매우 다른 크기를 가지면 매개변수에 대한 비율, w1과 w2가 다른 범위를 가진다

- **정규화**하면 **비용함수가 평균적으로 대칭적인 모양**

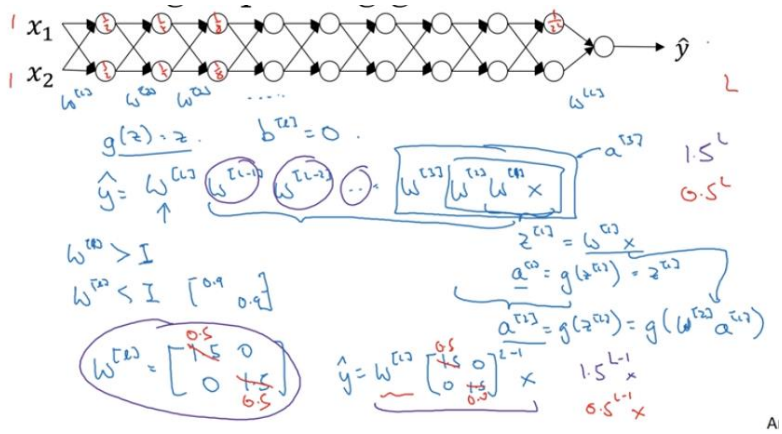
- 경사하강법으로 최솟값을 찾을 수 있음

⇨ **특성이 비슷한 크기를 가질 때 비용함수가 더 둥글고 최적화하기 쉬운 모습이 된다**

경사 소실/ 경사 폭발

- 깊은 신경망 학습시킬 때
- 미분값 기울기가 아주 작아지거나 커지는 경우
- 무작위의 가중치 초기화에 대한 선택으로 부분적 문제 해결 가능

<깊은 네트워크 학습 시키는 경우>



- 매개변수 w[1], ... w[l] & 선형 활성화 함수.

  $g(z)=z, b^{[l]}=0$ 이라고 가정했을 때 $\hat{y}=w^{[l]}w^{[l-1]}\cdots w^{[2]}w^{[1]}x$

  (b가 0이니까 z[1]= w[1]x

  g가 선형함수이므로 a[1]=g(z[1])=z[1]

⇨ 행렬들의 곱이 y의 예측값)

if) 가중치 행렬 w=1.5*단위행렬

- L이 크면 yhat도 커짐 , 1.5^num layers

⇨ 활성화 값 폭발

if) 가중치 행렬 w=0.5*단위행렬

⇨ 활성화 값이 감소

- 가중치가 단위 행렬보다 조금 더 크다면 활성값은 폭발

- 조금 작다면 활성값은 기하급수적 감소
- 경사하강법에서 계산하는 경사가 층의 개수에 대한 함수로 증가 or 감소
- 경사가 기하급수적으로 작으면 경사하강법은 작은 단계만 진행해서 학습 시간이 오래걸림

⇨ 부분적 해결법 : 가중치 초기화에 대한 선택

심층 신경망의 가중치 초기화

ex) 단일 뉴런에 대한 가중치 초기화하는 방법

- z=w1x1+w2x2+...wnxn

⇨ n이 클수록 w_i 가 작아져야함

1. w_i 의 분산을 1/n 으로 설정 (n: 입력특성 개수)

가중치 layer = np.random.randn(shape)*np.sqrt(1/n^[l-1])

2. relu 활성화 함수를 쓰면 분산을 2/n[l-1] 으로 설정

g(z) = relu(z)

가중치 layer = np.random.randn(shape)*np.sqrt(2/n^[l-1])

⇨ 그래디언트 소실 폭주 방지

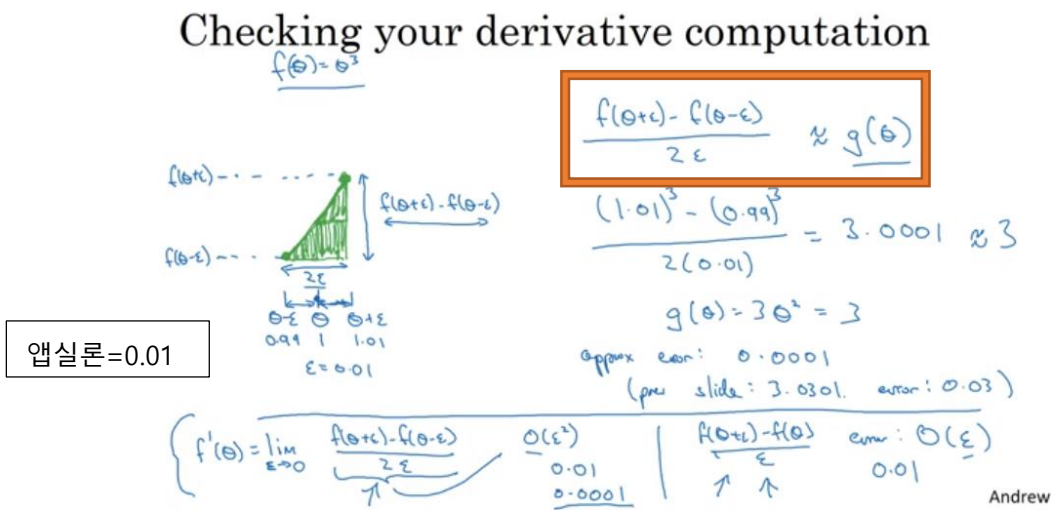
3. tanh 활성화 함수를 쓰면 분산을 1/n[l-1] or 2/(n[l-1]+n[l]) 로 설정

sqrt(1/n[l-1]) => Xavier initiation **세이비어 초기화**

⇨ 가중치 행렬의 초기화 분산에 대한 기본 값을 주는 것

⇨ 이 경우에는 분산 매개변수도 하이퍼 파라미터

기울기의 수치 근사



- 경사검사 : 역전파를 잘 구현했는지 확인하는 방법

  경사의 계산을 수치적으로 근사하는 방법

⇨ **높이 / 너비 를 더 큰 삼각형에서 계산**하면 도함수 근사에 더 효율적

⇨ **양 쪽의 차이**를 이용하는 방법

경사 검사

- 역전파 구현의 버그를 찾는 검사
- 매개변수들을 하나의 큰 벡터로 만들기
- J를 세타 의 함수가 되도록 함
- 벡터가 가까운지 어떻게 정의? => 유클리드 거리 계산

  ||d세타 approx-d세타|| / ||d세타 approx||+||d세타||

~~10^-7 : good

~~10^-3 이면 다시 체크하기

- 원소의 차이가 너무 크다면 버그가 있을 수 있음

경사 검사 시 주의할 점

1. 훈련에서 경사 검사를 사용하지 말고 디버깅을 위해서만 사용하기

I의 값에 대한 d세타 approx. [i] (모든 i에 대해 계산하면 시간 너무 오래 걸림)

d세타를 계산하는 역전파 이용해 도함수 계산

2. 경사 검사의 알고리즘이 실패하면 개별적인 컴포넌트 확인해 버그를 확인하기

d세타approx가 d세타 에서 먼 경우 어떤 d세타approx[i]의 값이 d세타[i]의 값과 매우 다른지 확인하기

3. 경사검사를 할 때 비용함수 J세타가 1/m * 손실함수의 합 + 정규화항

4. 경사검사는 드롭아웃에서 작용하지 않는다

- 모든 반복마다 드롭아웃은 은닉 유닛의 서로다른 부분집합을 무작위로 삭제하기 때문에 드롭아웃이 경사하강법을 시행하는 비용함수를 계산하는 쉬운 방법이 없음

- 드롭아웃의 keep_prop을 1.0으로 설정하고 알고리즘이 최소한 드롭아웃 없이 맞는지 이중검사하기 위해 경사검사를 사용

4. 무작위적인 초기화에서 경사검사를 시행 w와 b가 0에서 멀어질 수 있는 시간을 주기