3. 최적화 문제 설정

1. 입력값의 정규화

1. **입력 정규화**

- 평균 빼기(0으로 만들기): 0의 평균을 갖게 될 때까지 훈련 세트 이동

1. 평균을 0으로 만듭니다.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$x := x - \mu$$

2. **분산 정규화**

- 분산을 1로 만들기

2. 분산을 1으로 만듭니다.

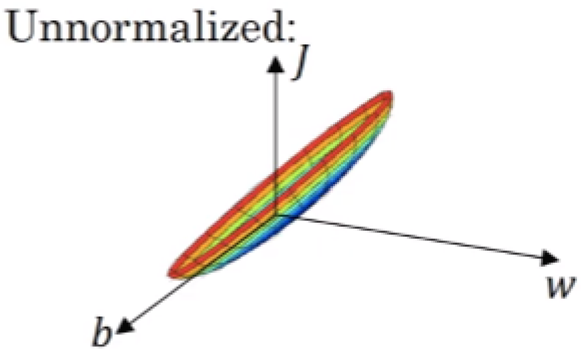
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} x^{(i)2}$$

$$x := \frac{x}{\sigma^2}$$

→ 훈련 데이터를 확대하는 데 사용하려면, 테스트 세트를 정규화할 때도 같은 평균, sigma 사용하기.

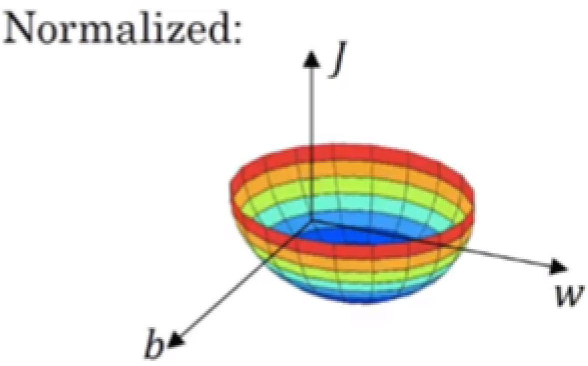
3. **왜 입력 특성을 정규화?**

- 정규화 되지 않은 입력 특성 사용: 매우 구부러진 활처럼 가늘고 긴 모양의 비용 함수가 됨.



→ 경사하강법 실행: **매우 작은 학습률(왔다갔다 이동이 많이 필요)**

- 정규화 된 입력 특성 사용: 평균적으로 대칭적인 모양의 비용 함수를 가짐.



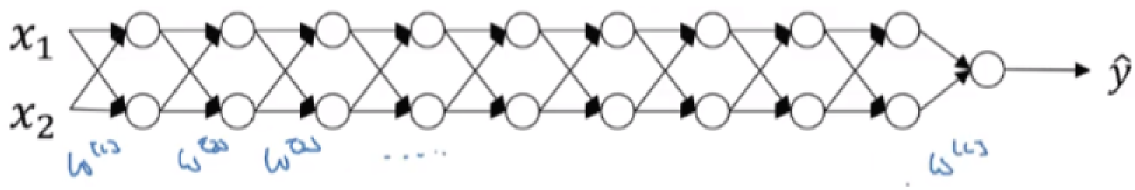
→ 경사하강법 실행: **최솟값으로 바로 갈 수 있다.**

입력 특성이 매우 다른 크기를 갖는다면, 특성을 정규화하는 것이 중요하다.

2. 경사 소실 / 경사 폭발

: 매우 깊은 신경망을 훈련시키는 것의 문제점

1. **2개의 은닉 유닛을 가지는 매우 깊은 신경망을 훈련시키는 경우**



- 활성화 함수 g(z)가 선형 활성화 함수를 사용한다고 가정
- b^[l]=0라고 가정
- 출력 y = w^[l] * w^[l-1] * w^[l-2] * ... * w^[3] * w^[2] * w^[1] * x
- 모든 행렬들의 곱 = y의 예측값

$g(z) = z. \quad b^{[l]} = 0.$

$$\hat{y} = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots \underbrace{\boxed{w^{[3]} \boxed{w^{[2]} w^{[1]} x}}}_{a^{[3]}}$$

$$z^{[1]} = w^{[1]} x$$

$$a^{[1]} = g(z^{[1]}) = z^{[1]}$$

$$a^{[2]} = g(z^{[2]}) = g(w^{[2]} a^{[1]})$$

- 모든 가중치 행렬 w = 1.5E(E: 단위 행렬)라고 가정하면, yhat = 1.5^(l-1)Ex
  - 더 깊은 신경망일수록 yhat 값이 기하급수적으로 증가.
- w = 0.5E라고 가정하면, yhat = 0.5^(l-1)Ex
  - 더 깊은 신경망일수록 yhat 값이 기하급수적으로 감소.

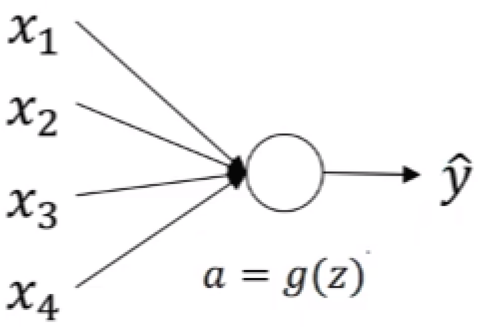
→ w의 값 > 단위 행렬: 경사의 폭발

→ w의 값 < 단위 행렬: 경사의 소실

- 깊은 신경망에서 활성값, 경사가 기하급수적으로 증가/감소 하면, 훈련 시키기 어렵.

3. 심층 신경망의 가중치 초기화

1. **단일 뉴런에 대한 가중치 초기화 예제**



- 더 깊은 망에서 입력은 a^[l]인 어떤 층이 될 것
- z=w_1x_1 + w_2x_2 + … + x_nx_n의 값을 갖는다.(b=0)
- n의 값이 클수록 w_i의 값이 작아져야 한다.

→ z는 w_ix_i의 합이므로, 이를 다 더하면 각각의 항이 작아지기를 바라기 때문

- **w_i의 분산을 1/n으로 설정**
  - 특정 층에 대한 가중치 행렬 w^[l]: `w^[l] = np.random.rand(shape)*sp.sqrt(1/n^[l-1])`

- **ReLU 활성화 함수를 사용하는 경우, w_i의 분산을 2/n^[l-1]로 설정**
  - 입력 특성 혹은 활성 값의 평균이 대략 0이고 표준편차 1을 갖는다면, 이 역시 비슷한 크기를 갖는다.

  → 경사 소실과 폭발 문제에 도움을 줄 수 있다.

  - `w^[l] = np.random.rand(shape)*sp.sqrt(2/n^[l-1])`

- **tanh 활성화 함수를 사용하는 경우, w_i의 분산을 1/n^[l-1] 또는 2/n^[l-1]+n[l] 으로 설정**

```
w^[l] = np.random.rand(shape)*sp.sqrt(1/n^[l-1])
w^[l] = np.random.rand(shape)*sp.sqrt(2/n^[l-1]+n^[l])
```

→ 위의 식들은 **가중치 행렬의 초기화 분산에 대한 기본값을 줄 뿐**

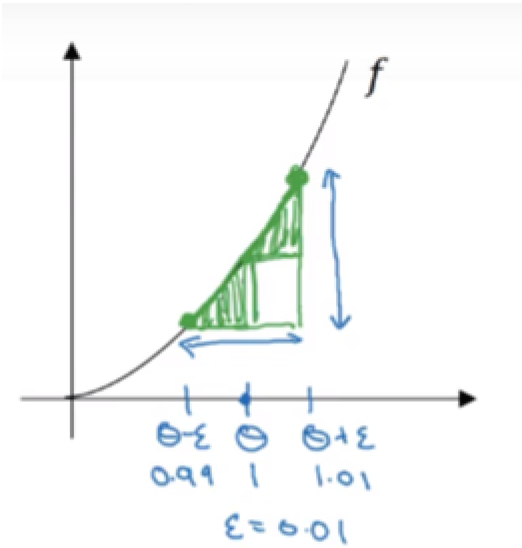
4. 기울기의 수치 근사

1. **경사 검사**

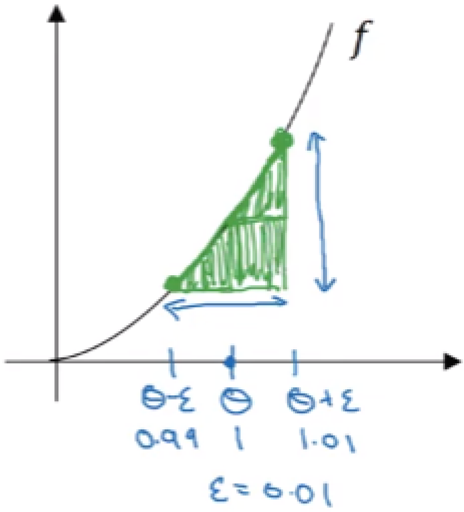
- 역전파를 알맞게 구현했는지 확인하는데 이용

→ 모든 수식을 작성해도 세부 사항이 올바른지 100% 확신할 수 없기 때문

2. **경사의 계산을 수치적으로 근사하는 방법**



- f가 θ-ε 인 점과 θ+ε 인 점에서 높이/너비를 더 큰 삼각형에서 계산한다면, 더 나은 경사 비율을 얻을 수 있다



- (큰 초록 삼각형의 높이) = f(θ+ε) - f(θ-ε)
- (큰 초록 삼각형의 너비) = 2ε

→ (f(θ+ε) - f(θ-ε))/ 2는 g(θ)와 비슷해야한다

$$\frac{f(\theta+\varepsilon)-f(\theta-\varepsilon)}{2\varepsilon} \approx \underline{g(\theta)}$$

$$\frac{(1.01)^3-(0.99)^3}{2(0.01)} = 3.0001 \approx 3$$

$$g(\theta) = 3\theta^2 = 3$$

approx error: 0.0001

(prev slide: 3.0301. error: 0.03)

5. 경사 검사

1. **매개변수들을 하나의 큰 벡터 θ로 바꾸기**

- 신경망은 매개변수 W^[1], b^[1]부터 W^[L], b^[L]까지 가진다.
- 행렬 W^[1] → 벡터의 크기로
- 모든 W 행렬을 받아서 벡터로 바꾸고 연결

→ **비용함수 J(W, b)는 J(θ)로 변한다.**

2. **dW^[1], db^[1], ...의 매개변수를 매우 큰 벡터 dθ로 만들기**

- dW^[1]: 벡터로 바꾸기
- db^[1]: 이미 벡터
- 모든 dW는 행렬
- dW^[1]은 W^[1]과 같은 차원이고 db^[1]은 b^[1]과 같은 차원
- 같은 방식으로 크기를 바꾸고 연결해 모든 미분값을 매우 큰 벡터 dθ로 바꾸기

→ **dθ가 비용함수 J(θ)의 기울기?**

3. **경사 검사의 구현 방법(Grad Check)**

- J: 이제 매우 큰 매개변수 θ에 관한 함수.
- J(θ) = J(θ1, θ2, θ3, ...)
- 경사 감소를 위해 반복문을 구현

  - 그후, 수치 미분을 구합니다.
    - $d\theta^{[i]}_{approx} = \frac{J(\theta_1, \cdots, \theta_i + \epsilon, \cdots) - J(\theta_1, \cdots, \theta_i - \epsilon, \cdots)}{2\epsilon}$
  - 최종적으로 수치 미분과 일반 미분을 비교합니다.
    - $d\theta^{[i]}_{approx} \approx d\theta$
    - 유사도를 계산하는 방법은 유클리디안 거리를 사용합니다.
    - $\frac{||d\theta^{[i]}_{approx} - d\theta||_2}{||d\theta^{[i]}_{approx}||_2 + ||d\theta||_2}$
    - 보통 거리가 10^{-7} 보다 작으면 잘 계산되었다고 판단합니다.

- dθapprox[i]은 근사적으로 dθ[i]와 같아야 함.(θapprox와 dθ가 같은 차원)
- 두 벡터의 유사도: 유클리드 거리를 계산하면 알 수 있음
- dθapprox - dθ의 L2 노름을 구한다. (제곱하지 않음에 주의)
- 벡터의 길이로 정규화하기 위해 ||dθapprox|| + ||dθ||의 유클리드 길이로 나누어 주기.
- 신경망의 정방향 전파 또는 역전파를 구현할 때 경사검사에서 상대적으로 큰 값이 나온다면 버그의 가능성을 의심해봐야 한다.

→ **디버깅의 과정을 거친 후 경사검사에서 작은 값이 나온다면 구현이 잘 된 것**

6. 경사 검사 시 주의할 점

1. **훈련에서 경사 검사를 사용하지 말고, 디버깅을 위해서만 사용하기**

- 모든 i의 값에 대해 dθapprox[i]를 계산하는 것은 느리기 때문에 훈련에서 사용하지 않고 디버깅을 위해서만 사용
- dθ에 가까워지면, 경사 검사를 끄고 모든 반복마다 실행되지 않도록 한다.

2. **경사 검사의 알고리즘이 실패한다면, 개별적이 컴포넌트를 확인하여 버그 체크**

- dθapprox가 dθ에서 매우 먼 경우, 서로 다른 i에 대하여 어떤 dθapprox[i]의 값이 dθ[i]의 값과 매우 다른지 확인
- 어디서 버그를 추적할 수 있을지에 대한 약간의 추측 제공

3. **정규화 항 기억하기**

- dθ는 θ에 대응하는 J의 정규화 항도 포함하기 때문에, 경사 검사 계산시 같이 포함해야 한다.

4. **경사 검사는 드롭아웃에서 작동하지 않는다.**

- 드롭아웃은 모든 반복마다 은닉 유닛의 서로 다른 부분 집합을 무작위로 삭제하기 때문에 적용하기 쉽지 않다.

→ 드롭아웃을 이용한 계산을 이중으로 확인하기 위해 경사검사를 확인하기는 어렵

- 드롭아웃을 끄고 알고리즘이 최소한 드롭아웃 없이 맞는지 확인하고, 다시 드롭아웃을 켠다.

5. **Run at random initialization, perhaps again after some training**

- 가끔 일어나는 무작위 초기화(w와 b가 0에 가까울 때 경사하강법의 구현에 맞게된 경우)에서 경사 검사를 실행하는 것
- 네트워크를 잠시 동안 훈련해서 w와 b가 0에서 멀어질 수 있는 시간을 준다.
- 일정 수의 반복을 훈련한 뒤에 경사 검사를 다시 해보는 방법