9주차

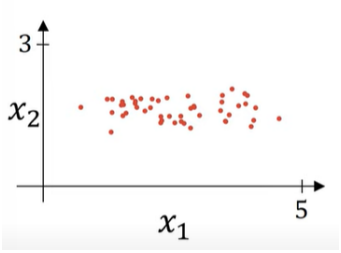
날짜	@2024년 5월 7일
과제	강의 요약 출석 퀴즈
세부내용	[딥러닝 2단계] 3. 최적화 문제 설정
자료	[Week9] 출석퀴즈

최적화 문제 설정

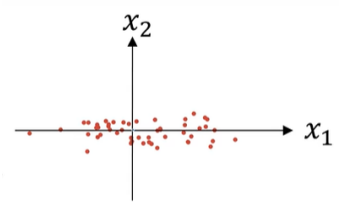
1️⃣ 입력값의 정규화

Nomalizing training sets

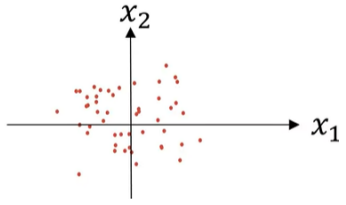
- 입력 피처가 2차원 (x1, x2)인 훈련 세트의 산포도를 정규화



- 1단계 : 평균을 뺌 ⇒ 평균을 0으로 만들기



  - $\mu = \frac{1}{m}\sum_{i=1}^{m} X^{(i)}$ (vector)
  - $X = X - \mu$ ⇒ 0의 평균을 갖도록 훈련 세트를 이동
- 2단계 : 분산을 정규화 ⇒ 분산을 1로 만들기

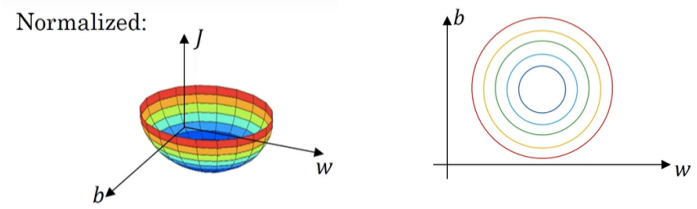


  - $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(X^{(i)})^2$ (vector) → 여기서 $X^{(i)}$는 X에서 평균을 빼 준 값
  - $X/ = \sigma^2$ ⇒ 분산이 모두 1이 되도록 함
- 테스트 세트를 정규화 할 때는 훈련 세트에서와 동일한 μ, σ 사용!

Why normalize inputs?

- 비용함수 정의 : $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$
- 정규화되지 않은 입력 특성을 사용했을 때의 비용함수 분포

  - 활처럼 가늘고 긴 모양
  - 입력 특성의 scale이 매우 다르다면, 파라미터 간 값의 차이도 매우 커짐
  - 경사 하강법 실행 시 매우 작은 학습률 사용 → 최솟값에 도달하기까지 많은 단계 필요
- 정규화된 입력 특성을 사용했을 때의 비용함수 분포



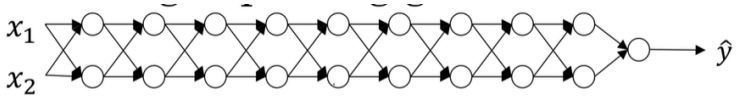
  - 대칭적인 모양
  - 어디에서 시작하든지 최솟값까지 큰 스텝으로 바로 도달할 수 있음

    ⇒ 최적화를 쉽고 빠르도록 함
- 특히 스케일의 차이가 큰 데이터에서 정규화가 중요 !

2️⃣ 경사소실/경사폭발

Vanishing/Exploding Gradients

- 2개의 은닉 유닛을 갖는 신경망 예시



  - 매개변수 : $w^{[1]}\ w^{[2]}\ \cdots\ w^{[L]}$
  - 선형 활성화 함수를 사용하고, $b^{[l]} = 0$ 라고 가정
  - 출력 : $\hat{y} = w^{[L]} \cdot w^{[L-1]} \cdot w^{[L-2]} \cdots w^{[2]} \cdot w^{[1]} \cdot x$

    ⇒ L이 매우 크다면 y의 예측값은 기하급수적으로 커질 것 → Exploding

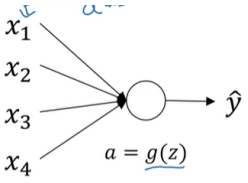
    ⇒ L이 1보다 더 작은 값이라면 y의 예측값은r기하급수적으로 감소 → Vanishing
- 가중치 $w^{[l]}$이 단위행렬보다 조금 더 크다면 깊은 네트워크의 경우 Exploding
- 가중치 $w^{[l]}$이 단위행렬보다 조금 더 작다면 깊은 네트워크의 경우 Vanishing

  ⇒ 경사하강법 시 많은 시간이 걸릴 것

3️⃣ 심층 신경망의 가중치 초기화

Single Neuron Example (단일 뉴런에서의 가중치 초기화 예제)

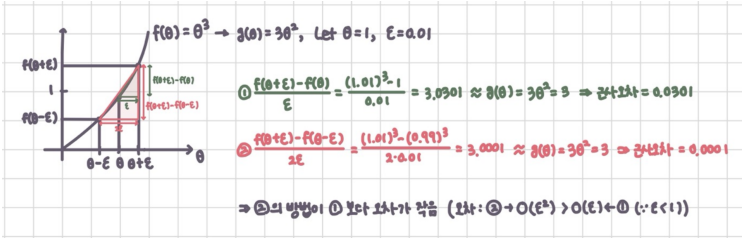
- 기울기 소실과 폭주 문제를 완화하는 방법 → 초기값을 신중하게 선택
- 단일 뉴런



  - $z = w_1x_1 + w_2x_2 + \cdots + w_nx_n$ (b=0이라고 가정)
    → z 값이 너무 크거나 작아지지 않도록 만들어야 함
  - n(입력 특성의 개수)이 클수록 w_i는 작아져야 함
  - 가중치의 분산($var(w_i)$)을 $\frac{2}{n}$로 설정 (using ReLU)
    ⇒ $w^{[l]}$ = np.random.randn(shape) * np.sqrt(2/$n^{[l-1]}$)
- tanh 활성화 함수를 사용한다면?
  - $\frac{1}{n}$을 사용 → np.sqrt(1/$n^{[l-1]}$)를 곱해줌 ⇒ Xavier initianlization
- np.sqrt(1/($n^{[l-1]} + n^{[l]}$)) 을 사용하는 경우도 존재
- 하이퍼파라미터로 고려하여 조정을 시도할 수도 있음

4 기울기의 수치 근사

Checking Your Derivative Computation



5 경사 검사

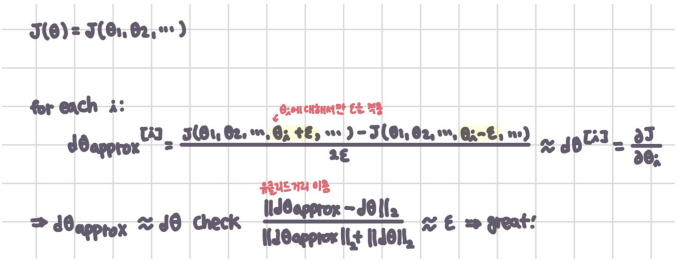
Gradient Check for a Neural Network

① 매개변수들을 하나의 큰 벡터 θ로 바꾸기 ⇒ 비용함수 $J(\theta)$

② dW와 db들을 θ와 같은 차원의 벡터 dθ로 바꾸기

→ dθ가 비용함수 $J(\theta)$의 기울기인가?

Gradient Checking



→ check 결과가 epsilon보다 크다면 버그가 존재할 가능성이 높음

6️⃣ 경사 검사 시 주의할 점

Gradient Checking Implementation Notes

- Don't use in training → only to debug
  - 모든 i에 대해 dθ_approx[i]를 계산하는 것은 시간이 매우 오래 걸림
- If algorithm fails grad check, look at components to try to identify bug
  - dθ_approx[i]와 dθ[i] 값이 매우 다를 때
- Remember regularization
- Doesn't work with dropout
  - 드롭아웃을 끄고(keep_prob=1.0) grad check로 디버깅한 후 드롭아웃 켜기
- Run at random initialization; perhaps again after some training