DL 스터디&프로젝트

[Euron 중급 세션 9주차] 딥러닝 2단계 3. 최적화 문제 설정

by 공부하자_ 2023. 11. 6.

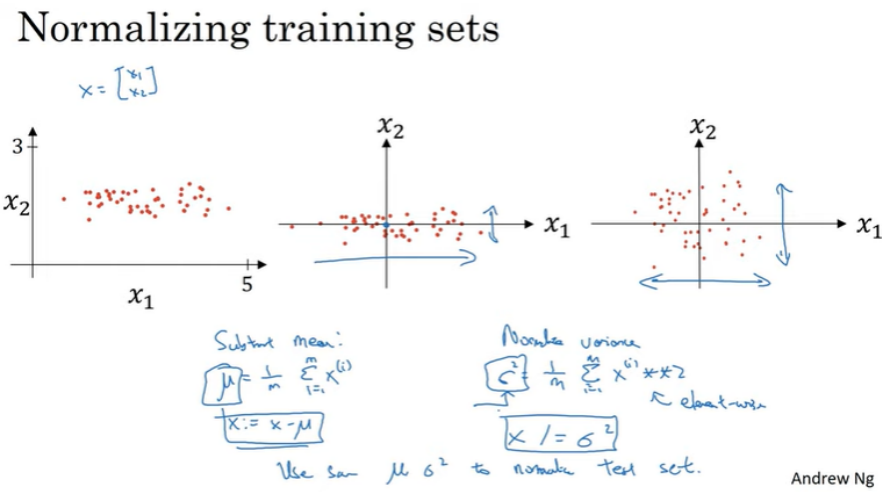
딥러닝 2단계: 심층 신경망 성능 향상시키기

3. 최적화 문제 설정

📌입력값의 정규화(C2W1L09)

핵심어: 정규화(normalizing)

신경망의 훈련을 빠르게할 수 있는 하나의 기법은 입력을 정규화하는 것이다.



두 개의 입력이 있는 훈련 세트가 있다고 치자(입력 특성 x가 2차원). 그리고 훈련 세트의 산포도가 이렇게 존재하며, 여기서 두 단계에 따라서 입력을 정규화하고자 한다. 첫 번째는 평균을 빼는 것, 즉 0으로 만드는 것이다. μ=1/m*(Σi=1부터 m까지 x(i)), x는 모든 훈련 샘플에 대해서 x-μ로 설정된다. 0의 평균을 갖게 될 때까지 훈련 세트를 이동하는 것이다. 두 번째 단계는 분산을 정규화하는 것이다. 위 예시의 경우 특성 x1이 특성 x2보다 더 큰 분산을 가지고 있다. σ^2= 1/m*(Σi=1부터 m까지 x(i)**2), 여기서 **는 요소별 제곱을 나타낸다. σ^2는 각 특성의 분산에 대한 벡터이며, 이미 평균을 뺐기 때문에 x(i) 요소별 제곱이 바로 분산이 되는 것이다. 이제 각각의 샘플을 얻어 σ벡터로 나눠준다. 그림으로 나타내면 세 번째 그림이 된다. x1과 x2의 분산은 이제 모두 1과 같다.

이를 훈련 세트를 확대하는 데 사용한다면, 테스트 세트를 정규화할 때도 같은 μ와 σ를 사용해야 한다. 훈련 세트와 테스트 세트를 다르게 정규화해선 안되기 때문이다. μ와 σ가 어떤 값이든 간에 이를 x=x- μ와 x=x/σ 식에 적용하여 정확히 같은 방식으로 테스트 세트를 확장해야 하는 것이다(훈련 세트와 테스트 세트에서 별개로 μ와 σ^2를 추정하는 것이 아님). 훈련 샘플과 테스트 샘플 모두 훈련 데이터에서 계산된 것과 같은 μ와 σ^2에 의해 정의된 변형을 거치기를 원하기 때문이다.

분류 전체보기
DL 스터디&프로젝트
Data Science 프로젝트
Github 스터디
Data Science 개인 공부
Backend 프로젝트
기타 공부

공지사항

최근글 인기글

[Euron 중급 세션 9주차] 딥러닝 2단계 3…
2023.11.06

백엔드 프로젝트 7주 (스프링 입문)
2023.11.04

[Euron 중급 세션 8주차] 딥러닝 2단계 2…
2023.10.30

자프실 - OOP 개념 정리
2023.10.12

[Euron 중급 세션 5주차] 딥러닝 2단계 1…
2023.10.09

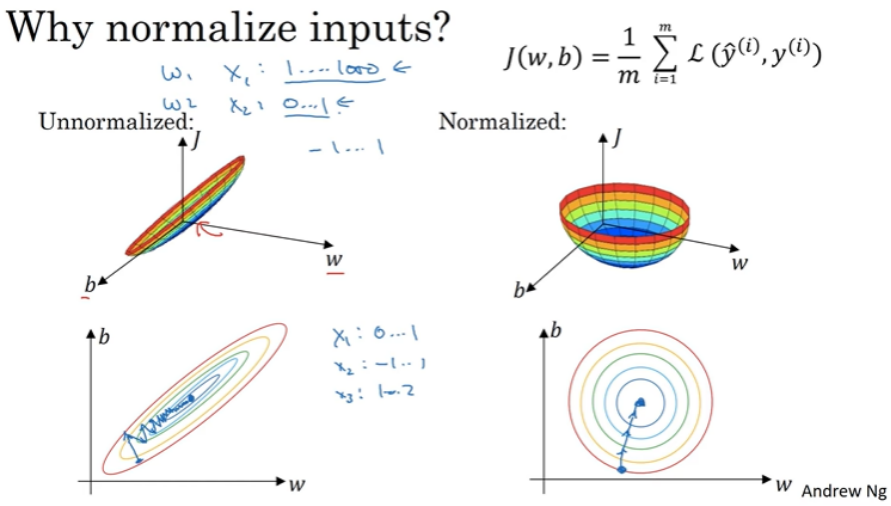
최근댓글

좋은 글 잘 보고 가요! 감사합니다 :)
좋은 글 잘 보고 가요! 감사합니다 :)
잘보고 갑니다!
포스팅 잘보고 갑니다! 응원할게요!

태그

Doit, pandas, 딥러닝스터디, 이지스퍼블리싱, 딥러닝교과서, 판다스, 데이터분석, 데이터사이언스, 판다스입문, bda

전체 방문자



우리는 왜 입력 특성을 정규화하기를 원할까? 비용함수의 정의가 J(w, b) = 1/m(Σi=1부터m까지 L(yhat(i), y(i)))와 같다는 것을 기억해야 한다. 정규화되지 않은 입력 특성을 사용하면 비용함수가 왼쪽 위의 그림과 같은 모습이 된다(매우 구부러진 활처럼 가늘고 긴 모양). 그러나 만약 특성들이 매우 다른 크기를 갖고 있다면(예를 들어 특성 x1이 1~1000, x2가 0~1), 매개변수에 대한 비율, 매개변수 값의 범위는 w1과 w2가 굉장히 다른 값을 갖게 되며, 이 때문에 위와 같은 모양이 되는 것이다. 그 밑에 함수의 등고선을 그려보면 가늘고 긴 함수를 얻을 수 있다.

반면에 특성을 정규화하면 비용함수는 평균적으로 대칭적인 모양을 갖게된다(오른쪽 그림). 왼쪽 함수의 비용함수에 경사하강법을 실행한다면 매우 작은 학습률을 사용하게 될 것이다. 여기서는 경사 하강법이 최종적으로 최솟값에 이르는 길을 찾기 전까지 앞뒤로 왔다갔다 하기 위해 많은 단계가 필요하기 때문이다. 반면 원 모양의 등고선의 경우, 어디서 시작하든 경사 하강법은 최솟값으로 바로 찾아갈 수 있다(왼쪽처럼 왔다갔다 하지 않아도 큰 스텝으로 전진 가능). 실전에서는 w가 높은 차원의 벡터일 것이며 2차원에 그리는 것으로 모든 직관을 올바르게 전달하지는 못한다. 그러나 특성이 비슷한 크기를 가질 때 비용함수가 더 둥글고 최적화하기 쉬운 모습이 된다는 대략적인 직관은 얻을 수 있다. 1~1000, 0~1이 아닌 대부분 -1~1 혹은 서로 비슷한 분산으로 비용함수 J를 최적화하기 쉽고 빠르게 만든다. 실제로 하나의 특성 x1이 0~1, x2가 -1~1, x3가 1~2인 경우에 이들은 상당히 비슷한 범위를 가지므로 1~1000,0~1처럼 극단적으로 다른 범위를 갖는 특성과는 다르게 잘 작동될 것이다. 너무 다른 범위는 최적화 알고리즘에 방해가 되기 때문에, 모든 것을 0의 평균으로 설정하고 앞 슬라이드처럼 모든 특성을 비슷한 크기로 보정할 수 있는 분산으로 설정한다면 학습 알고리즘이 빠르게 실행되는 것을 도울 것이다. 정규화는 어떤 해도 가하지 않기 때문에 이 과정이 알고리즘을 빠르게 훈련하는 데 도움을 줄 수 있을지가 확실하지 않더라도, 되도록이면 진행하는 것을 추천한다.

- 정규화 방법

1. 평균을 0으로 만듭니다.

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

$$x := x - \mu$$

2. 분산을 1으로 만듭니다.

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

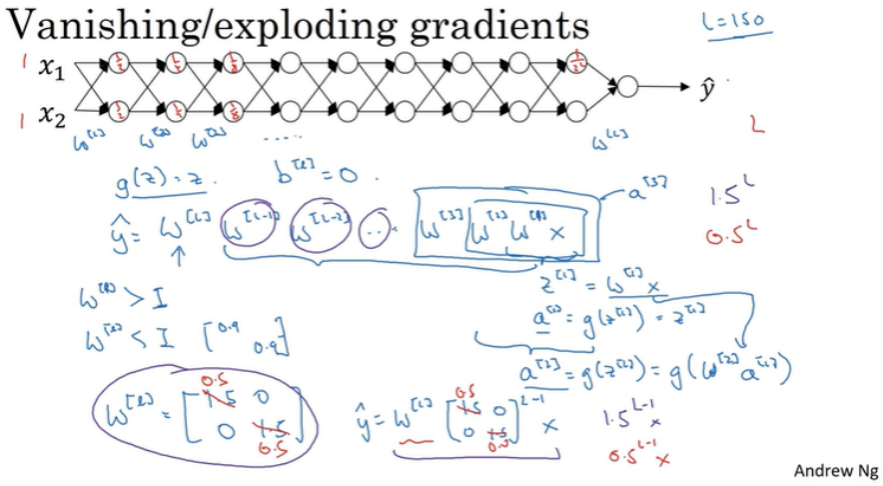
$$x := \frac{x}{\sigma^2}$$

- 테스트 세트 정규화할 때 훈련 데이터에 사용한 μ, σ 를 사용해야합니다.
- 정규화를 통해 비용함수의 모양은 더 둥글고 최적화하기 쉬운 모습이 됩니다. 그로 인해 학습 알고리즘이 빨리 실행됩니다.

x를 σ^2이 아닌 σ로 나눠줘야 분산이 1이 된다.

📌경사소실/경사폭발(C2W1L10)

핵심어: 경사의 소실(vanishing gradients), 경사의 폭발(exploding gradients) 신경망을 훈련시키는 것, 특히 매우 깊은 신경망을 훈련시키는 것의 문제는 경사의 소실과 폭발이다. 매우 깊은 신경망을 훈련시킬 때, 미분값 혹은 기울기가 아주 작아지거나 커질 수 있다. 심지어는 기하급수적으로 작아져 훈련을 어렵게 만들 수도 있다. 이번에는 경사 폭발과 소실의 문제점이 무엇인지 알아보고 무작위의 가중치 초기화에 대한 신중한 선택으로 이 문제를 줄일 수 있는 방법도 함께 다뤄볼 예정이다.



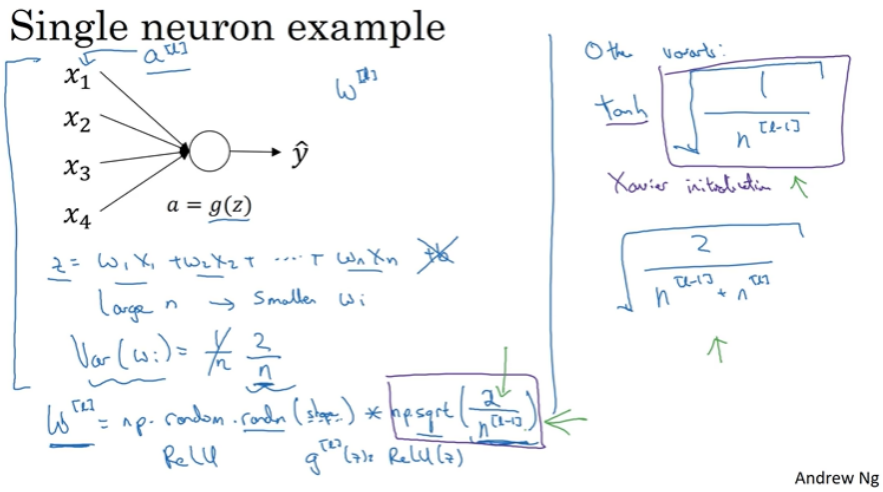
위 그림처럼 매우 깊은 신경망(위의 경우 오직 두 개의 은닉 유닛만을 갖지만 때에 따라 더 많을 수도 있다)을 훈련시키는 경우, 이 신경망의 매개변수는 w[1], w[2], w[3], ..., w[L]까지 갖게된다. 간단하게 보이기 위해 활성화 함수 g(z)가 선형 활성화 함수이고, b[l]은 0이라고 가정하자. 이 경우 출력 y는 y=w[l]*w[l-1]*w[l-2]*...*w[3]*w[2]*w[1]*x 와 같이 된다. 수학적으로 확인하자면 w[1]*x는 z[1]이 될 것이다(b[1]=0이기 때문에 z[1]=w[1]*x+b를 해도 0을 더하는 것과 마찬가지). 또한 a[1]=g(z[1])이 되며, 선형 활성화 함수를 사용하기 때문에 이 값은 z[1]과 같다. 이와 같은 이유 w[1]*x=z[1]=a[1]이므로 w[2]*w[1]*x는 a[2]와 같다(a[2]=g(z[2])=g(w[2]a[1]). 같은 방식으로 w[3]*w[2]*w[1]*x 의 값은 a[3]이 된다. 이런 식으로 계산하면 이 모든 행렬들의 곱은 y의 예측값이 된다. 각각의 가중치 행렬인 w[l]이 1.5, 1.5, 0, 0의 값을 갖는 (대각)행렬이라고 가정하자. 마지막 행렬은 나머지 행렬의 곱에 따라 다른 차원을 갖게 된다. 그럼 y의 예측값은 (다른 차원을 갖는 마지막 가중치 행렬=w[l]) * ((1.5, 0, 0. 1,5)^L-1)*x 이 되는데, 이는 각각의 w[l-1], w[l-2], ... 행렬들이 w[l](1.5*단위행렬)행렬과 같다고 가정했을 때 w[l] * ((1.5, 0, 0. 1,5)^L-1)*x 식의 계산으로 끝나기 때문이다. 따라서 y의 예측값은 1.5^(L-1)*x가 될 것이다. 여기서 만약 깊은 신경망에서 L의 값이 크다면, y의 예측값도 매우 커질 것이며(1.5^L(레이어의 수)만큼 매우 기하급수적으로) 이에따라 매우 깊은 신경망을 갖는다면 y의 값은 폭발할 것이다. 이와 반대로 만약 1.5값을 1보다 작 0.5로 교체한다면, 1.5^L에서 0.5^L이 것이다. 이 행렬은 w[L]을 무시한다면 0.5^(L-1)*x가 될 것이다. 만약 각 행렬이 1보다 작다면, 예를 들어 x1과 x2가 각각 1인 경우에, 활성화 a는 1/2, 1/2, 1/4, 1/4, 1/8, 1/8,... 이런 식으로 가다가 마지막 활성값이 1/2^L이 될 것이다. 따라서 활성화의 값은 네트워크가 매우 깊을 기하급수적으로 감소하게 된다. 따라서 여기서 얻을 수 있는 직관은, 가중치 w[l]이 단위행렬보다 조금 더 크다면(1.5), 매우 깊은 네트워크의 경우 활성값은 폭발할 수 있다는 것이다. 그리고 w[l]이 단위행렬보다 조금 작다면(0.9), 매우 깊은 네트워크의 경우 활성값은 기하급수적으로 감소할 것이다. 현대의 신경망은 보통 L이 150의 값을 갖는다. 그러나 이런 깊은 신경망에서 활성값이나 경사가 L에 대한 함수로 기하급수적으로 증가하거나 감소한다면, 이들의 값들은 아주 커지거나 작아질 수 있으며 그렇게 되면 훈련을 시키는 것이 어려워진다(특히 경사가 기하급수적으로 작은 경우). 이러한 상황에서는 경사 하강법은 아주 작은 단계만을 진행할 것이고 학습시키는데 아주 오랜 시간이 걸릴 것이다.

- 매우 깊은 신경망을 훈련시킬 때 나타나는 문제점은 경사의 소실과 폭발입니다.
- 예를 들어 $g(z) = z, b^{[l]} = 0$ 이라고 가정했을 때 $\hat{y} = w^{[l]}w^{[l-1]}...w^{[2]}w^{[1]}x$ 가 됩니다. 이때 모든 가중치 행렬 $w = 1.5E$ 라고 가정하면(E 는 단위 행렬입니다.) $\hat{y} = 1.5^{(l-1)}Ex$ 가 되고 더 깊은 신경망일수록 $\hat{y}$ 의 값은 기하급수적으로 커집니다. 반대로 모든 가중치 행렬 $w = 0.5E$ 라고 가정하면 $\hat{y} = 0.5^{(l-1)}Ex$ 가 되고 더 깊은 신경망일수록 $\hat{y}$ 의 값은 기하급수적으로 감소합니다. 이를 토대로 생각하면 경사 하강법에서 w의 값이 단위행렬보다 큰 값이라면 경사의 폭발, w 의 값이 단위 행렬보다 작은 값이라면 경사의 소실 문제점이 생깁니다.
- 경사의 소실과 폭발로 인해 학습 시키는데 많은 시간이 걸리기에 가중치 초기화 값을 신중하게 해야합니다. 이는 다음 강의에서 배우도록 하겠습니다.

📌 심층 신경망의 가중치 초기화(C2W1L11)

핵심어: 가중치 초기화(weight initialization)

위에서 아주 깊은 신경망에서 경사 소실과 폭발의 문제가 있다는 점을 살펴볼 수 있었다. 문제를 완전히 해결하진 못하지만 부분적으로 도움을 줄 수 있는 해결법은, 신경망에 대한 무작위의 초기화를 더 신중하게 선택하는 것이다. 먼저 단일 뉴런에 대한 가중치를 초기화하는 예제를 다뤄보자. 그리고 더 깊은 네트워크로 이를 일반화시키고자 한다.



Andrew Ng

하나의 뉴런이 있는 예제를 먼저 살펴보자. 하나의 뉴런에 특성 4개(x1~x4)가 있고, a는 g(z)이고 출력은 y의 예측값이다. 나중에 다룰 더 깊은 망에서 이 입력은 a[l]인 어떤 층이 될 것이다. z는 w1x1 + w2x2 + ... + wnxn의 값을 갖는다(b는 0이라고 생각하고 지금은 무시). 여기서 z의 값이 너무 크거나 작아지지 않도록 해야 한다. n이 커질수록 wi값이 작아지는데, z는 wixi의 합이기 때문에 더 많은 wixi 항들을 더하게 되면 각각의 항이 작아지기를 바랄 것이다. 한 가지 합리적인일은 wi의 분산을 1/n으로 설정하는 것이다(n은 뉴런으로 들어가는 입력 특성의 개수). 따라서 실제로 특정 층에 대한 가중치 행렬 w[l]=np.random.randn(shape)*np.sqrt(1/n[l-1])이라 설정할 수 있다(1/n[l-1]은 층 l의 뉴런에 들어가는 특성의 개수의 역수). ReLU 활성화 함수를 사용하는 경우 분산을 1/n보다 2/n으로 설정하는 것이 더 잘 작동한다. 즉 g[l](z) = ReLU(z)인 경우이다. n[l-1]을 사용하는 이유는, 이 예제의 로지스틱 회귀에서는 n개의 입력 특성을 갖지만, 더 일반적인 경우에 층 l은 해당 층의 각 유닛에 대해 n[l-1]의 입력을 갖기 때문이다. 입력 특성 혹은 활성값의 평균이 대략 0이고 표준편차 1을 갖는다면, z 또한 역시 비슷한 크기를 갖게 될 것이다. 위에서 언급한 문제를 완전히 해결하지는 못하지만, 경사 소실과 폭발 문제에 확실히 도움을 줄 수 있는 것이다. 왜냐하면 각각의 가중치 행렬 w를 1보다 너무 커지거나 너무 작아지지 않게 설정해서, 너무 빨리 폭발하거나 소실되지 않도록 하기 때문이다.

다른 변형을 살펴보면, 이전까지는 ReLU 활성화 함수를 사용하는 상황을 가정했던 반면 이번에 tanh 활성화 함수를 사용한다면 상수 2 대신 상수 1을 사용하라는 논문이 존재한다. 1/n[l-1]의 제곱근을 구해주고, 이 제곱근 항은 nq.sqrt(1/n[l-2])항을 대체하게 된다. tanh 활성화 함수를 사용하는 경우 이 값(1/n[l-1]의 제곱근)은 세이비어 초기화라고 부른다. 또는 2/(n[l-1]+n[l]) 식을 사용하는데, 이 또한 이론적으로 타당한 이유가 있다. 그러나 가장 일반적인 활성화 함수인 ReLU를 사용하는 경우가 많으므로 nq.sqrt(1/n[l-2])를 더 자주 사용한다. 그러나 실제로 이 모든 식들은 그저 시작점을 제공할 뿐이다. 즉 가중치 행렬의 초기화 분산에 대한 기본값만 준다는 것이다. 이와 같은 분산을 원한다면 분산 매개변수는 하이퍼파라미터로 조정할 또 다른 값이 된다. 따라서 이 식(1/n[l-2])에 곱할 매개변수를 가질 수 있고, 하이퍼파라미터 탐색의 일부로 그 곱하는 수를 조정할 수 있다. 가끔 그 하이퍼파라미터를 조정하는 것은 적당한 크기의 효과를 가지며, 가장 먼저 조정을 시도해야 할 하이퍼파라미터는 아니지만 그 조정이 상당한 도움이 되는 경우도 많다.

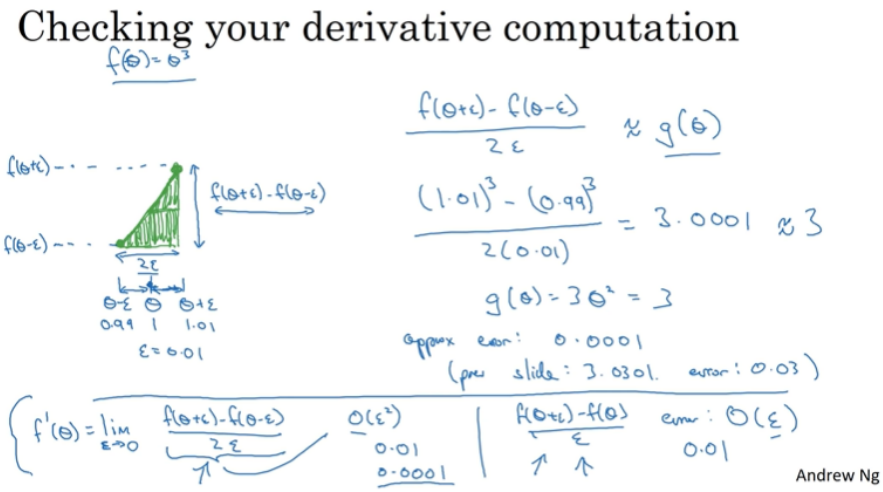
- 가중치 초기화 방법

1. w_i 의 분산을 $\frac{1}{n}$ 으로 설정합니다. (n : 입력 특성의 개수)
2. ReLU 활성화 함수를 사용하는 경우 w_i 의 분산을 $\frac{2}{n^{[l-1]}}$ 으로 설정합니다.
3. tanh 활성화 함수를 사용하는 경우 w_i 의 분산을 $\frac{1}{n^{[l-1]}}$ 또는 $\frac{2}{n^{[l-1]}+n^{[l]}}$ 으로 설정합니다.

📌기울기의 수치 근사(C2W1L12)

핵심어: 경사 검사(gradient checking)

역전파를 구현할 때 경사 검사라는 테스트가 있다. 이는 역전파를 맞게 구현했는지 확인하는 데 도움을 주는데, 왜냐하면 모든 수식을 작성해도 세부 사항이 맞게 작성되었는지는 100% 확신하지 못하기 때문이다. 경사 검사를 구현하기 위해 먼저 경사의 계산을 수치적으로 근사하는 방법을 알아보고, 이후 경사 검사를 구현해 역전파의 구현이 맞는지 확인하는 방법에 대해 다루고자 한다.



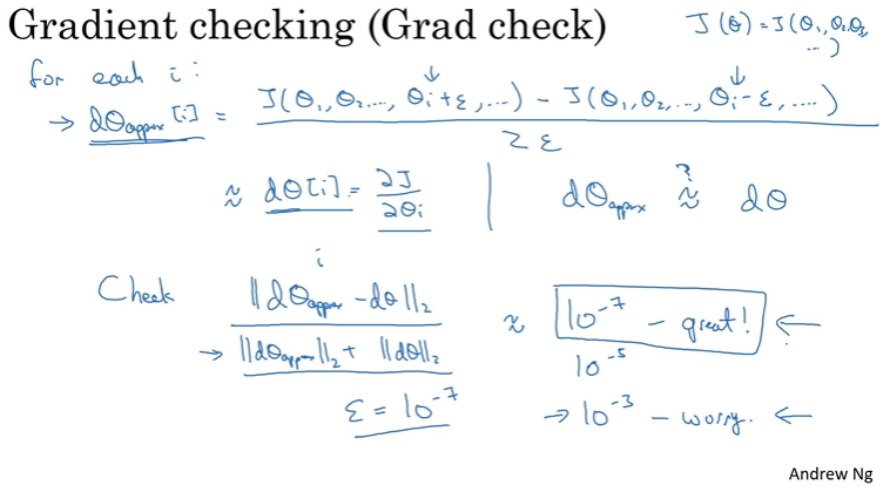
먼저 함수 f를 취해서 그래프로 나타내었다(왼쪽 그림). 여기서 f(θ)= θ^3이며 θ가 1인 경우를 생각해보자. θ+ϵ을 얻기 위해 θ를 오른쪽으로 살짝 이동시키는 대신, 오른쪽과 왼쪽 모두 조금씩 움직여 θ-ϵ을 얻는다. 이 상황에서 θ는 1, 오른쪽은 1.01, 왼쪽은 0.99가 된다(ϵ=0.01). 그래프 위헤 표시한 작은 삼각형을 취해서 (높이/너비)를 얻는 대신, 더 나은 경사 비율을 얻을 수 있다. f가 θ-ϵ인 점과 θ+ϵ인 점을 이어 더 큰 삼각형에서 (높이/너비)를 계산한다. 이렇게 하는 기술적인 이유는 더 큰 삼각형에서 (높이/너비)를 구하는 것이 θ에서의 도함수를 근사하는 데 더 나은 값을 제공하기 때문이다. 이는 위의 작은 삼각형과 아래 작은 삼각형을 모두 고려한다. 한 쪽의 차이만을 사용하지 않고 양 쪽의 차이를 사용하는 것이다. 이제 수학적으로 계산해보면, 큰 초록색 삼각형의 높이는 f(θ+ϵ)-f(θ-ϵ)이며 삼각형의 너비는 2ϵ이다. 따라서 삼각형의 (높이/너비)는 (f(θ+ϵ)-f(θ-ϵ))/2ϵ 이 되는 것이다. 그리고 이 값은 g(θ)와 비슷해야 한다. 값을 대입하기 위해 f(θ)= θ^3를 이용하고 위의 상황에 적용해보면, ((1.01)^3 - (0.99)^3)/2*0.01 이 된다. 이는 3.0001이 되며, g(θ)=3θ^2에서 θ가 1일 때의 이 값은 3이었다. 이 두 값은 매우 유사하며, 이전에 한 쪽의 차이인 θ+ϵ만 사용했을 때 3.0301을 얻었으며 따라서 근사 오차는 0.03이었다(0.0001보다 큰 값). 이와 같이 도함수를 근사하기 위해 양 쪽의 차이를 이용하는 방법을 사용하면, 3에 매우 가까운 값이 나온다는 것을 알 수 있다. 이로 인해 g(θ)가 f의 도함수에 대한 더 올바른 구현이라는 것을 더 확신할 수 있다. 이것을 역전파의 경사 검사에서 사용한다면 한 쪽의 차이만을 사용하는 것보다 두 배는 느리게 실행될 것이지만, 실제로는 이 방법을 사용하는 것이 훨씬 더 정확하기에 가치가 있다고 볼 수 있다.

📌 경사 검사(C2W1L13)

핵심어: 경사 검사(gradient checking)

경사 검사는 시간을 절약하고 역전파의 구현에 대한 버그를 찾는 데 많은 도움을 준다. 어떻게 할 수 있는지 알아보자.

신경망은 매개변수를 W[1], b[1]부터 W[L], b[L]까지 가지고 있다. 경사 검사를 하기 위한 첫 번째 단계는 이 매개변수들을 하나의 큰 벡터 θ로 바꾸는 것이다. 행렬 W[1]을 벡터로 크기를 바꾸며, 모든 W행렬을 받아서 벡터로 바꾸고 모두 연결시킨다. 그 결과 매우 큰 벡터 매개변수 θ를 얻게 될 것이다. 비용함수 J를 W와 b의 함수로 만드는 대신, θ의 함수가 되도록 하는 것이다. W와 b에서 했던 것과 같은 순서로, dW[1], db[1], ... 의 매개변수를 매우 큰 벡터 dθ로 만든다. 그리고 이는 θ와 같은 차원을 가진다. 이전과 같은 방식으로 dW[1]을 벡터로 바꾸고, db[1]은 이미 벡터이므로 둔다(모든 dW는 행렬이다). 이렇게 되면 dW[1]은 W[1]과 같은 차원이고 db[1]은 b[1]과 같은 차원이다. 같은 방식으로 크기를 바꾸고 연결하여 모든 미분값을 매우 큰 벡터 dθ로 바꿀 수 있으며 이는 θ와 같은 차원을 가진다. 여기서 궁금한 점은 dθ가 비용함수 J(θ)의 기울기인지 아닌지이다.



Andrew Ng

이제 경사 검사의 구현 방법을 다룰 것이며, 이는 보통 줄여서 grad check이라고 한다. J는 이제 매우 큰 매개변수 θ에 관한 함수이며 J(θ)를 J(θ1, θ2, θ3, ...)로 확장시킬 수 있다(매우 큰 매개변수 벡터 θ의 차원이 어떻든지 간에). 경사 검사를 구현하기 위해서는 반복문을 구현해야 한다. θ의 요소 각각에 대하여 dθapprox[i]를 계산해보면, 양 쪽 차이를 이용해 J(θ1, θ2, ..., θi)까지 하는데 ϵ를 더해 살짝 이동시킨다. 다른 것은 그대로 두고 θi에만 ϵ을 더해준다. 양 쪽 차이를 취하기 때문에 다른 쪽도 똑같이 해주며 θi-ϵ을 이용한다. θ의 다른 요소들은 마찬가지로 그대로 둔다. 그리고 이 양 쪽 차이를 취한 식의 값을 2ϵ로 나눠준다. 이전에 봤던 것은 이 값이 근사적으로 dθ[i]와 같아야 했다(만약 dθi가 비용함수 J의 도함수라고 했을 때, 함수 J의 θi에 대한 편미분). dθapprox[i] 값을 모든 i의 값에 대해 계산하면, 마지막에 두 개의 벡터로 계산이 마무리된다. dθapprox와 dθ은 모두 θ와 같은 차원이며, 이 두 벡터가 근사적으로 같은지 확인해야 한다.

두 벡터가 꽤 가까운지 어떻게 정의할 수 있을까? 우선 이 두 벡터의 유클리드 거리를 계산한다. 유클리드 거리를 얻기 위해서는 두 원소의 차이를 제곱한 것의 합의 제곱근을 구해야 한다. (dθapprox - dθ)의 L2 norm을 구하고(이 값을 제곱하지 않는다는 것을 명심해야 함), 벡터의 길이로 정규화하기 위해 ||dθapprox|| + ||dθ||의 유클리드 길이로 나눠준다. 이 벡터가 아주 작거나 큰 경우에 대비해, 분모는 이 식을 비율로 바꿔주는 역할을 한다. ϵ=10^-7에서 이 수식은 10^-7이나 그보다 작은 결과가 나오며 이는 근사가 매우 잘 되었다는 뜻이다. 아주 작은 값이기 때문이다.

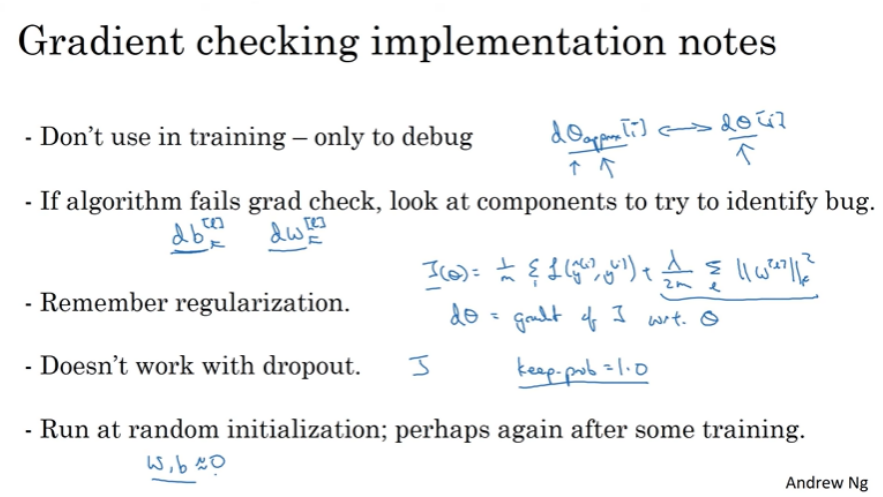
하지만 이제 벡터의 원소를 이중으로 확인해봐야 한다. 너무 큰 원소가 있는 것은 아닌지 확인해야 하며, 원소의 차이가 너무 크다면 버그가 있을 수 있다. 만약 왼쪽 수식이 10^-3을 결과로 내놓는다면 버그의 가능성이 커서 더 자세히 살펴볼 것이다. θ의 개별적인 원소를 신중하게 살펴서 특정 i에 대해 dθapprox[i] - dθ[i]가 심한 값을 추적하고 미분의 계산이 옳지 않은 곳이 있는지 확인해야 한다. 디버깅이 끝나고 10^-7만큼의 작은 값이 나온다면 올바른 구현을 한 것이다. 따라서 신경망의 정방향 전파 혹은 역전파를 구현할 때 경사 검사에서 상대적으로 큰 값이 나온다면 버그의 가능성을 의심해봐야 하며, 디버깅의 과정을 거친 후 경사 검사에서 작은 값이 나온다면 구현에 대해 자신감을 가져도 좋다.

- 우선, 모델 안에 있는 모든 변수(W, b)를 하나의 벡터(θ)로 concatenate합니다.
- 그러면 비용 함수는 J(W, b) 에서 J(θ) 로 변합니다.
- 그후, 수치 미분을 구합니다.
  - $d\theta_{approx}^{[i]} = \frac{J(\theta_1, \cdots, \theta_i + \epsilon, \cdots) - J(\theta_1, \cdots, \theta_i - \epsilon, \cdots)}{2\epsilon}$
- 최종적으로 수치 미분과 일반 미분을 비교합니다.
  - $d\theta_{approx}^{[i]} \approx d\theta$
  - 유사도를 계산하는 방법은 유클리디안 거리를 사용합니다.
  - $\frac{\|d\theta_{approx}^{[i]} - d\theta\|_2}{\|d\theta_{approx}^{[i]}\|_2 + \|d\theta\|_2}$
  - 보통 거리가 10^{-7} 보다 작으면 잘 계산되었다고 판단합니다.

📌 경사 검사시 주의할 점(C2W1L14)

핵심어: 경사 검사(gradient checking)

앞서 경사 검사에 대해 배웠는데, 이 번에는 신경망에서 이를 구현하기 위한 실질적인 방법과 팁을 공유하고자 한다.



첫 번째로 훈련에서 경사 검사를 사용하지 말고 디버깅을 위해서만 사용해야 한다. 모든 i의 값에 대한 dθapprox[i]를 계산하는 것은 매우 느리다. 따라서 경사 검사를 구현하기 위해 dθ를 계산하는 역전파를 이용해 도함수를 계산한다. 디버깅할 때만 dθapprox[i]을 계산하고 dθ에 가까워지게 한다. 이 과정이 끝나면 경사 검사를 끄고 모든 반복마다 실행되지 않도록 한다. 너무 느려질 수 있기 때문이다.

둘째, 만약 경사 검사의 알고리즘이 실패한다면 개별적인 컴포넌트를 통해 버그를 확인해야 한다. dθapprox가 dθ에서 매우 먼 경우 서로 다른 i에 대하여 어떤 dθapprox[i]의 값이 dθ[i]의 값과 매우다른지 확인할 것이다. 예를 들어 어떤 층에서 θ나 dθ의 값이 대응되는 db[l]과 매우 멀지만, 대응되는 dw[l]과는 매우 가까운 경우를 살펴보자. θ의 서로 다른 컴포넌트는 b나 w의 다른 컴포넌트에 대응된다. 이경우 db(매개변수 b에 대응하는 도함수)를 어떻게 계산하느냐에 따라 버그가 발생할 것이며, 비슷한 방식으로 dθapprox의 값이 dθ의 값과 매우 멀고 모든 컴포넌트가 dw 혹은 특정한 층의 dw에서 온 것을 발견한다면 이를 통해 버그의 위치를 알아내는 데 도움을 받을 수 있을 것이다. 이는 항상 버그를 바로 찾을 수 있게 하는 것은 아니지만, 어디서 버그를 추적할 수 있을지에 대한 추측을 제공한다.

셋째, 경사 검사를 할 때 사용하는 정규화 항에 대해, 비용함수 J(θ)가 (1/m)*(ΣL(yhat(i), y(i))+ λ/m(Σ||w[l]||)^2, 즉 이것이 비용함수 J의 정의가 된다. dθ는 θ에 대응하는 J의 경사이며 정규화 항(λ/m(Σ||w[l]||)^2)을 포함한다.

넷째, 경사 검사는 드롭아웃에서 작동하지 않는다. 모든 반복마다 드롭아웃은 은닉 유닛의 서로 다른 부분집합을 무작위로 삭제하기 때문에, 드롭아웃이 경사 하강법을 시행하는 비용함수 J를 계산하는 쉬운 방법이 없다. 드롭아웃은 어떤 비용함수 J를 최적화하는 것처럼 보여질 수 있긴 하다. 그러나 비용함수 J는 어떤 반복에서든지 삭제될 수 있는 기하급수적으로 큰 노드의 부분집합으로 정의되기 때문에 비용함수 J를 계산하는 것은 매우 어렵다. 드롭아웃을 사용하면 매번 다른 부분집합의 노드를 무작위로 삭제하게 되며, 따라서 드롭아웃을 이용한 계산을 이중으로 확인하기 위해 경사 검사를 사용하는 것은 어렵다. 또한 삭제된 노드의 패턴을 수정하거나 삭제된 유닛의 패턴이 맞는지 경사 검사를 확인하는 등 다른 방법도 있다. 좋은 방법은 드롭아웃을 끄고(keep_prob==1) 알고리즘이 드롭아웃 없이 맞는지 이중 검사하기 위해 경사 검사를 사용하고 드롭아웃을 키는 것이다.

마지막으로, 가끔 일어나는 불가능하지 않은 일이다. 무작위적 초기화에서 w와 b가 0에 가까울 때, 경사 하강법의 구현이 맞게 된 경우이다. 그러나 경사 하강법을 실행하면 w와 b가 점점 커지는데, 아마 역전파의 구현이 w와 b가 0에 가까울 때만 맞는 것일 수 있다. 따라서 w와 b가 커지면 그 값은 더 부정확해진다. 이런 상황에서 할 수 있는 방버븐 무작위적인 초기화에서 경사 검사를 실행하는 것이다. 그리고 네트워크를 잠시 동안 훈련해서 w와 b가 0에서 멀어질 수 있는 시간을 주는 것이다(작은 무작위적 초기화 값에서). 그리고 일정 수의 반복을 훈련한 뒤 경사 검사를 한 번 더 실행시킨다.

- 속도가 굉장히 느리기 때문에 훈련시에는 절대 사용 하지 않고 디버깅 할때만 사용합니다.
- 알고리즘이 경사 검사에 실패 했다면, 어느 원소 부분에서 실패했는지 찾아봅니다. 특정 부분에서 계속 실패했다면, 그 경사가 계산된 층에서 문제가 생긴것을 확인 할 수 있습니다.
- $d\theta$ 는 θ 에 대응하는 J의 정규화 항($\frac{d}{dm}\sum_m \|w^{[l]}\|_F^2$)도 포함하기 때문에 경사 검사 계산시 같이 포함해야합니다.
- 드롭아웃에서는 무작위로 노드를 삭제하기 때문에 적용하기 쉽지 않습니다. 따라서 통상은 드롭아웃을 끄고 알고리즘이 최소한 드롭아웃 없이 맞는지 확인하고, 다시 드롭아웃을 켭니다.
- 마지막으로 거의 일어나지 않지만 가끔 무작위 초기화를 해도 초기에 경사 검사가 잘 되는 경우입니다. 이때는 훈련을 조금 시킨 다음에 경사 검사를 다시 해보는 방법이 있습니다.

'DL 스터디&프로젝트' 카테고리의 다른 글

[Euron 중급 세션 8주차] 딥러닝 2단계 2. 신경망 네트워크의 정규화 (1) 2023.10.30