Lecture 4: Introduction to Neural Networks

목차

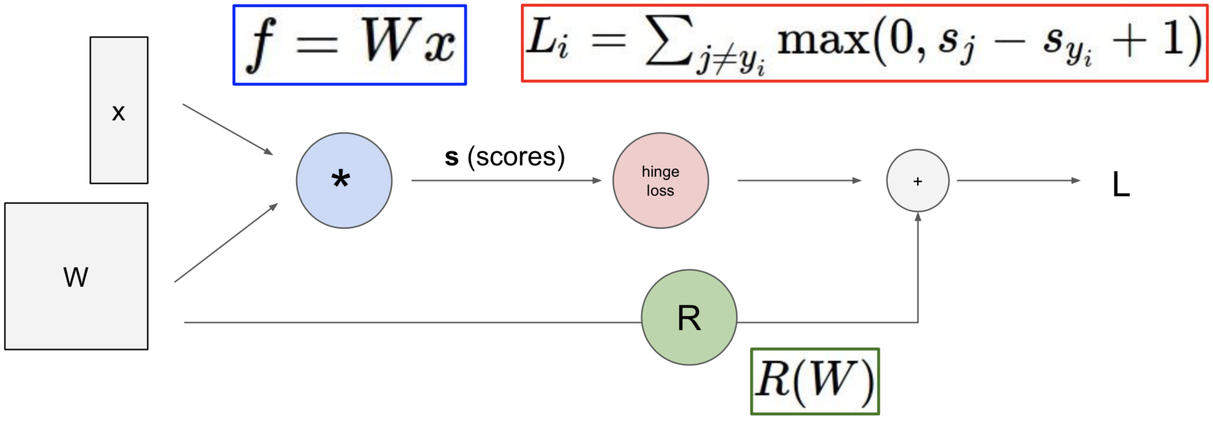
Backpropagation

지금까지 우리는 Gradient 를 구하는 방법에 대해서 알아보고 있다. 그러나 아직 W 를 업데이트 하는 과정을 표현하지는 않았다. ~~근데 이 챕터를 들어도 W 구하는건 아직 안나온다. 언제쯤 배울려나~~

Computational graphs

Computational Graphs 란 일종의 수식의 과정을 그래프화 한 것.
이를 이용하면 가시적으로 수학식을 표현할 수 있을 뿐만 아니라 수식을 코드화함에 있어 단계별로 해야할 일을 표현할 수 있다.

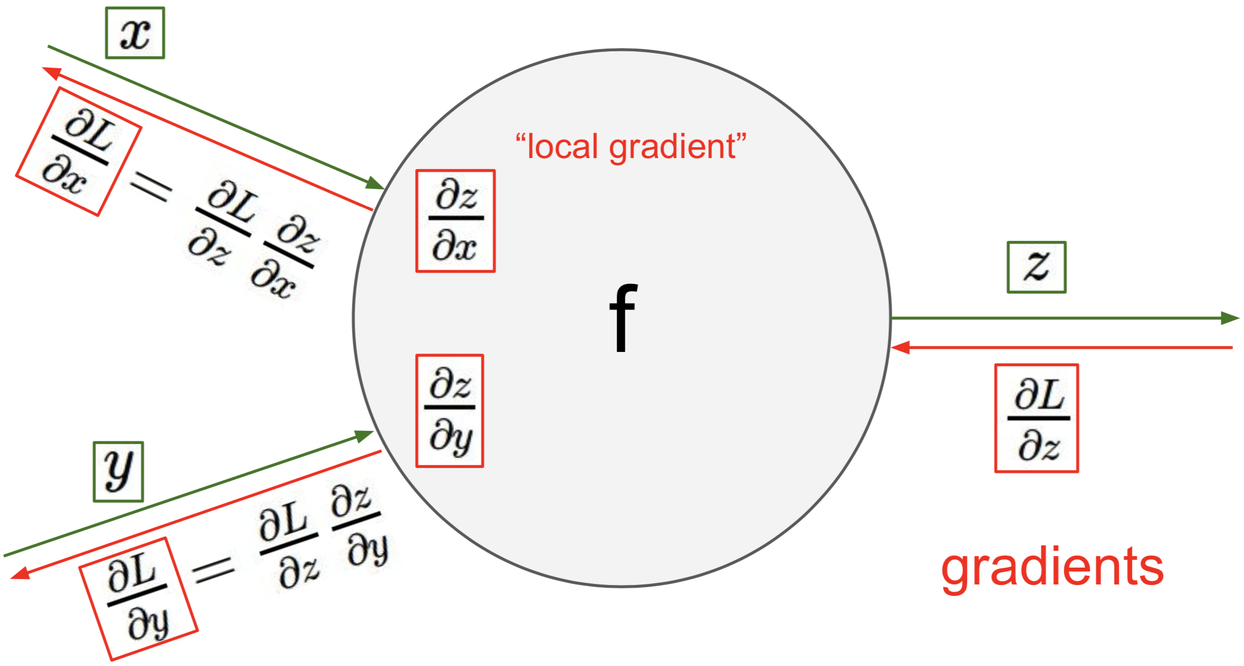
Computational graphs



Computational graph

Backpropagation

우선 우리는 Backpropagation 에 대해서 배울 것이다. 한국어로 역전파라고 하는데 고교 교육과정을 마쳤으면 수월하게 알 수 있는 chain rule 에 대해서 다루고 있다.
간단하게 표현하면 f(x, y, z) =(x + y) * z 라는 수식이 있으면 df/dx, df/dy, df/dz 를 구하기 위해서 chain rule을 이용하는 것이다. 위의 수식이 복잡해지면 chain rule을 이용해 결과부터 표현하는 것이 좋다.

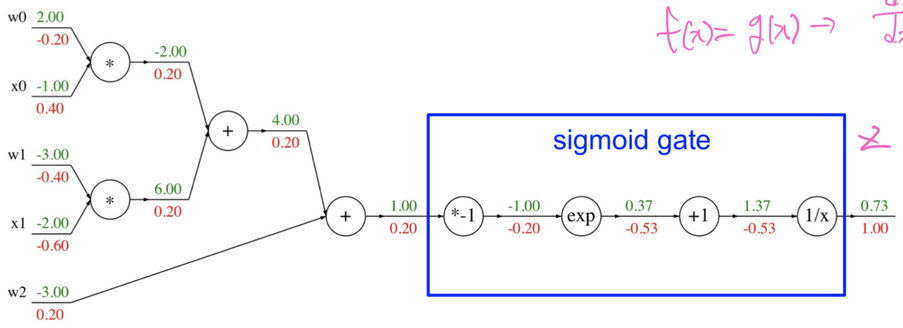


Back propagation

Gradient = local gradient * upstream gradient

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}} \qquad \sigma(x) = \frac{1}{1+e^{-x}} \quad \text{sigmoid function}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$

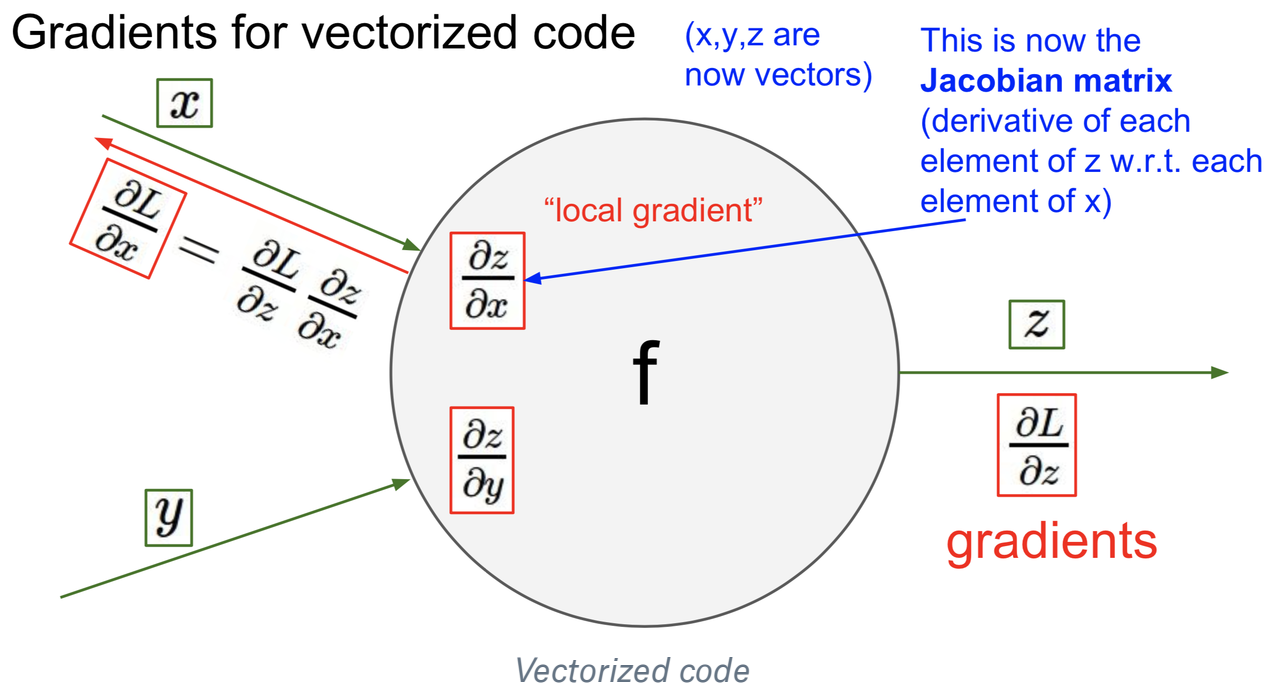


Sigmoid function

위와 같이 여러 수식을 합친 Gate 로 표현해줘도 동일한 값을 나타낸다. 자세한 내용은 강의를 참고.

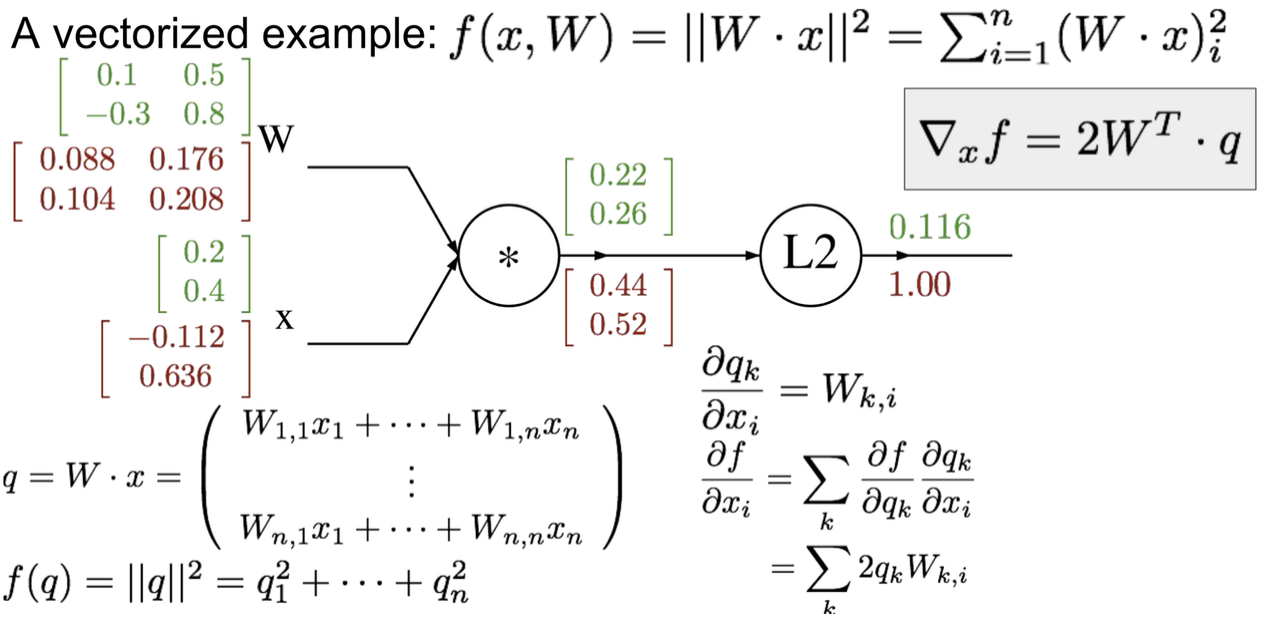
Gradients for vectorized code

위의 식을 딥러닝에서 이용하기 위해선 vectorize 하여 gpu 상에서 연산하는 작업이 필요하다.
이를 vectorized 된 코드로 이용하기 위해선 자코비안 행렬을 이용해야하는데..
분명 선형대수학 시간에 배웠는데 까먹었다. 우선 자코비안 행렬에 대해서 공부하고 가자.



Vectorized code

자코비안_위키백과
간단하게 말하자면 다변수 행렬의 각 성분에 대한 도함수를 나타낸 행렬이다. 따라서 우리는 각 gate 별로 미분을 할 수 있고 위에서 배운 backpropagtion을 적용한 것처럼 chain rule을 이용할 수 있다. 우선은 여기까지만 알아둬도 괜찮을 것 같다.



Slide

위의 슬라이드는 vectorize 된 식을 back propation 을 적용한 방법이다.
조금 의문이 가는 것은 dW 와 dx를 구할 때 df/dq를 곱하는 과정에서 행렬의 순서를 어떻게 정하는지 잘 모르겠다.
~~추측으로는 q_k 와 W_{k,i} 와 관련 있을 것 같은데..~~
우선 이건 자코비안을 제대로 공부하면 알 것 같고 이렇게 넘어가자.

이렇게해서 우리는 back propagation 을 하는 법에 대해서 공부하였다. 강의에서 소개하기로는 이미 많은 deep learning tool 들이 이런 gate 들을 지원하는 api를 이미 만들었다. (foward & backward) 우리는 이런 api 를 이용해 deep learning 에 집중할 수 있다는 말..
그럼 이제는 진짜 deep learning 에 대해서 배우자.

Neural Networks

우리는 지금까지 linear score function 에 대해서 예를 들어 개념을 이해했다(f = W * x) 그런데 W의 결과로 나온 score의 결과를 다시 score function 을 이용해 score를 구하는 방법이 있을 수 있을 것이다.

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

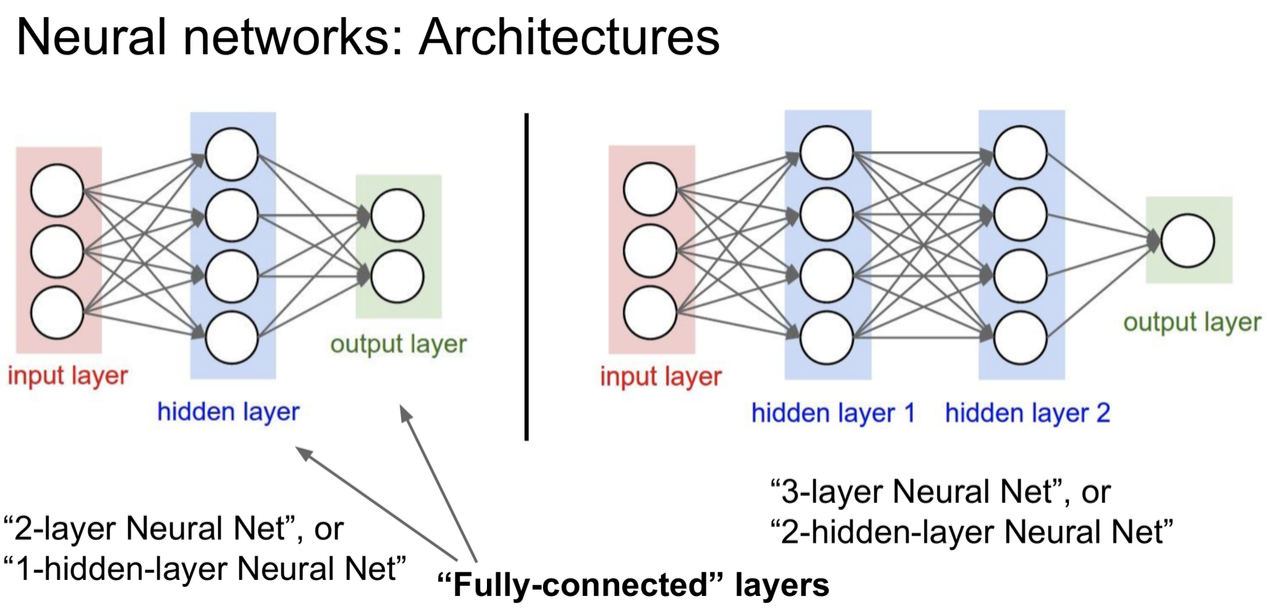
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

multi-layer network

이는 우리의 뇌에서 뉴런이 신호를 전달하는 매커니즘과 같고 이를 이용해서 나온 방식이 Neural Networks 이다.
강의에서 설명하기로는 우리는 딥러닝의 매커니즘을 생물의 특정 영역에서 영감을 받아서 이용하려고 하고 뇌의 기본 단위인 뉴런에서부터 딥러닝의 개념이 나왔다고 한다. ~~라고 내가 말을 지어내고있다.~~

Architecture



이번 강의에서는 Neural networks의 개념에 대해서만 설명하고 있고 다음 강의부터 본격적인 강의가 시작될 듯 하다.

Jun’s Note

1. 무엇을 배웠는지

> *W 계산을 하기위해 필요한 backpropagation 과 neural network 의 기본*

2. 어디에 쓰이는지
   > *위에서 설명.*
3. 수식의 목적을 잘 파악했는지?
   > *Vectorized computation 에 대해 좀 의문이 남는다. 이는 선형대수학을 다시 공부하면서 생각해봐야겠다.*