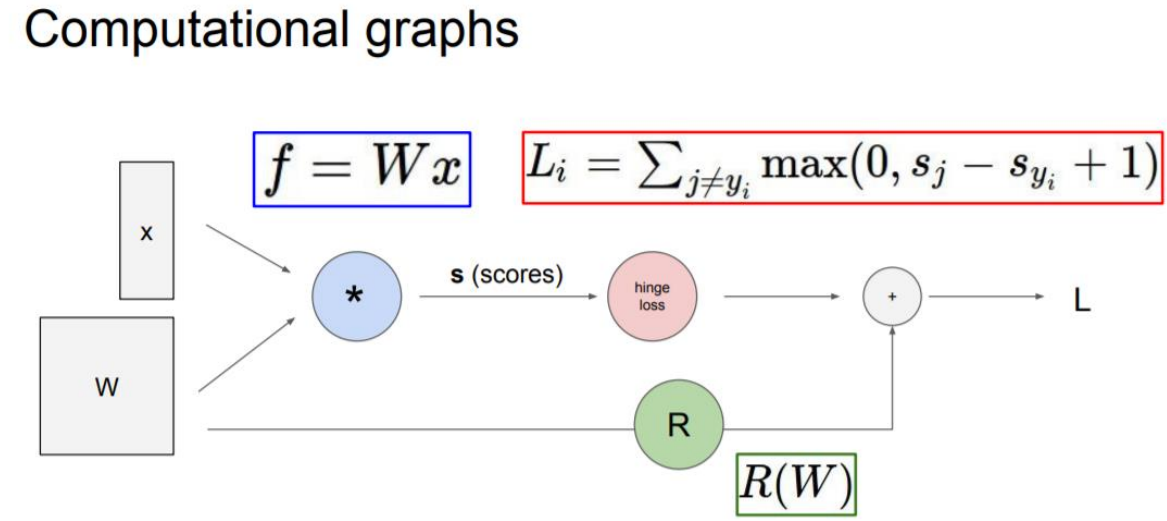
Lec4. Introduction to Neural Networks

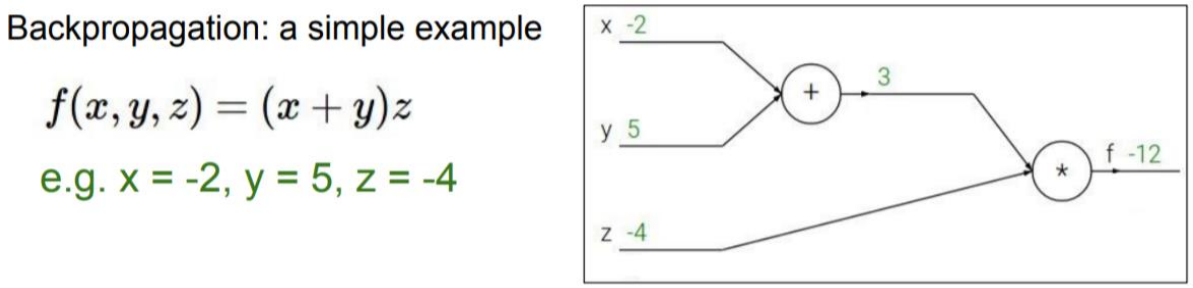
Backpropagation: computational graph 를 통해 이용 가능함

- parameter update 를 위해서는 gradient 를 구해야하는데, 복잡한 모델에서는 이를 구하기가 쉽지 않음. 이 때 이용하는 것이 backpropagation
- gradient 를 얻기 위해 computational graph 내부의 모든 변수에 대해 chain rule 을 재귀적으로 사용
- 복잡한 함수를 이용하여 작업할 때 유용하게 쓰임



- computational grah 를 이용하면 backpropagation 진행을 더 빠르게 할 수 있고, 미분도 빠르게 할 수 있음
- 각 노드는 연산 단계를 나타냄

아래는 예제



- forward pass: 순차적으로 함수값을 구하는 것
- backward pass: 역방향으로 차례로 미분하여 gradient 를 구하는 것 (backpropagation)

- ∂f/∂f, ∂f/∂z, ∂f/∂q 는 쉽게 구할 수 있음. 하지만 ∂f/∂y 를 한 번에 구하기에는 중간의 노드가 있어 조금 불편함
- 이는 chain rule 을 통해 구할 수 있음

Chain rule:

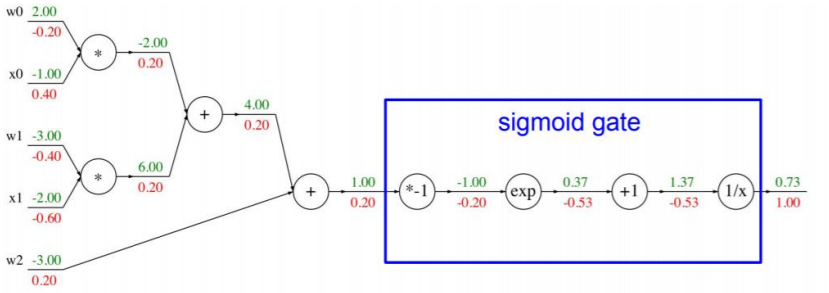
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

- chain rule 은 구하려는 미분값을 분해하여 구할 수 있다는 것을 의미함
- 우리가 구하고자 하는 global gradient 를 local gradients 의 곱으로 구할 수 있음
- local gradient 는 forward pass 과정에서 구할 수 있음
- 모듈화; 아래의 사진에서처럼, 직접 미분할 수 있는 sigmoid gate 를 module 로 묶어 직접 미분하여 구현한다면 좀 더 빠르게 backpropagation 을 진행할 수 있음

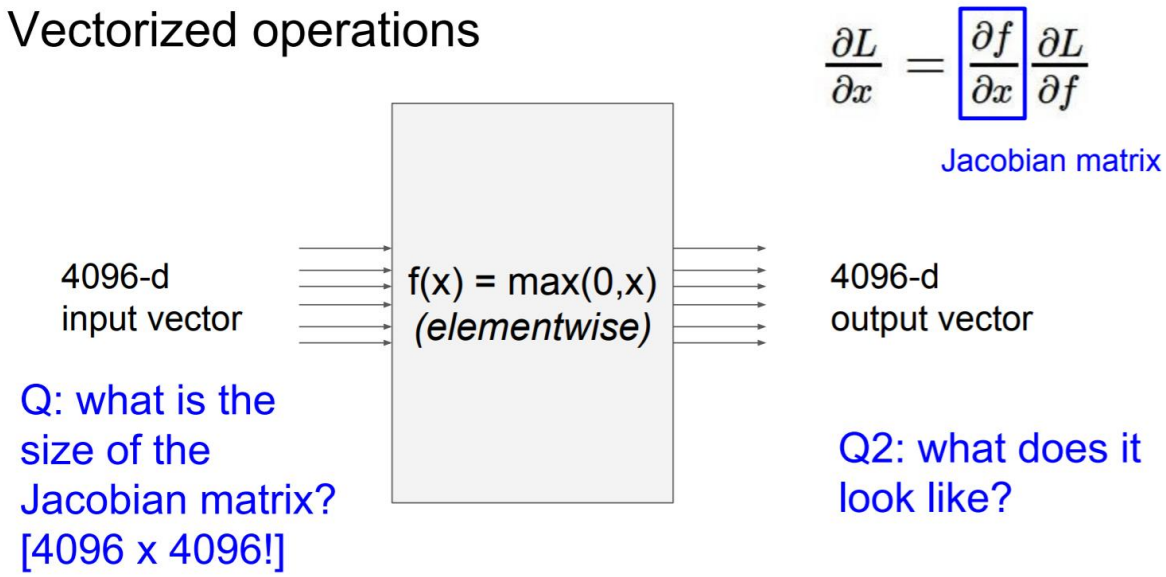
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$ sigmoid function

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



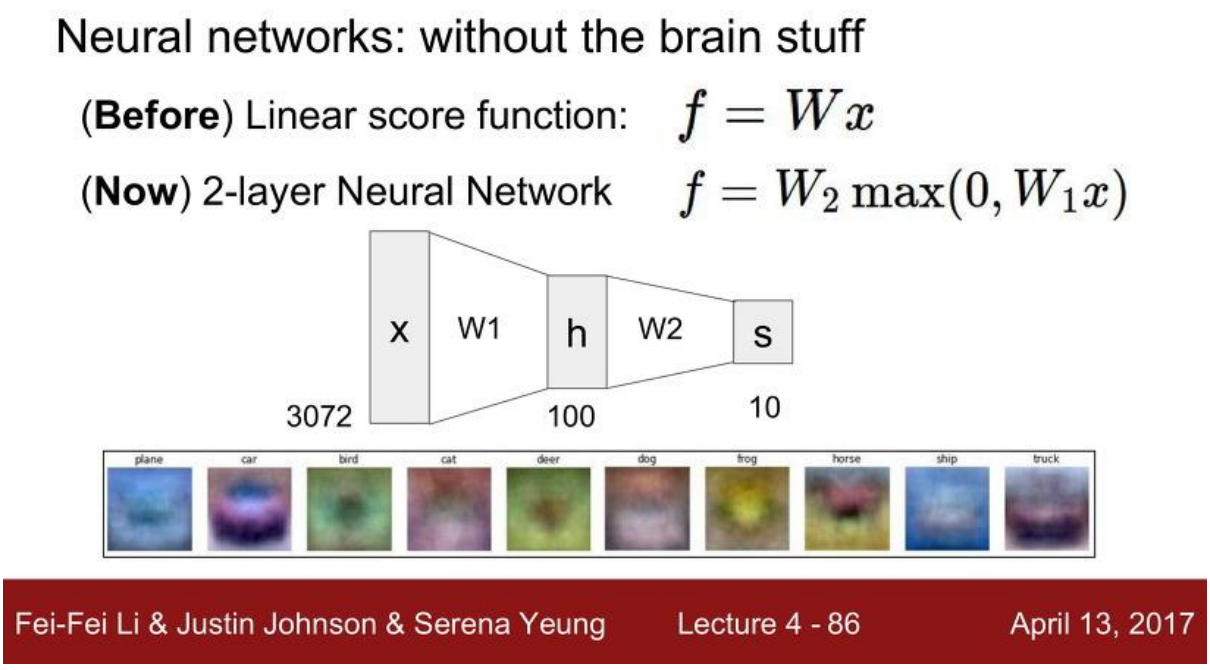
vectorized form: Jacobian matrix



- problem: elementwize 함수에서는 Jacobian matrix 의 크기가 variable x varaible 만큼임
- minibatch 를 이용한다면 그의 제곱배만큼 처리를 해야할 수도 있음
- 이 경우, diagonal matrix 만 이용하면 됨
- 가능한 이유: 입력의 각 요소는 출력의 해당 요소에만 영향을 미침
- 벡터화된 gradient 계산은 아래와 같이 하면 됨. 단, variable 과 결과값의 shape 이 같다는 것을 유념해야 함

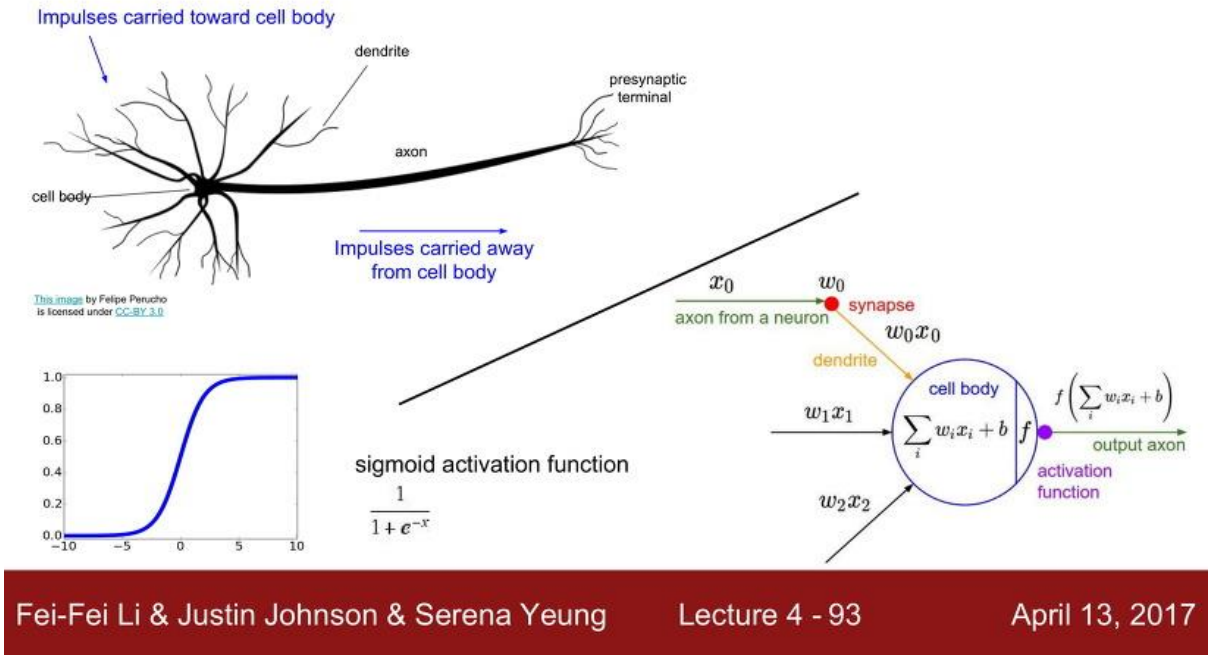
$$\nabla_w f = 2q \cdot x^T$$

Neural Networks: deep learning 에서 기본적으로 사용하는 frame 으로, 생물학의 neural network 에서 따온 모델

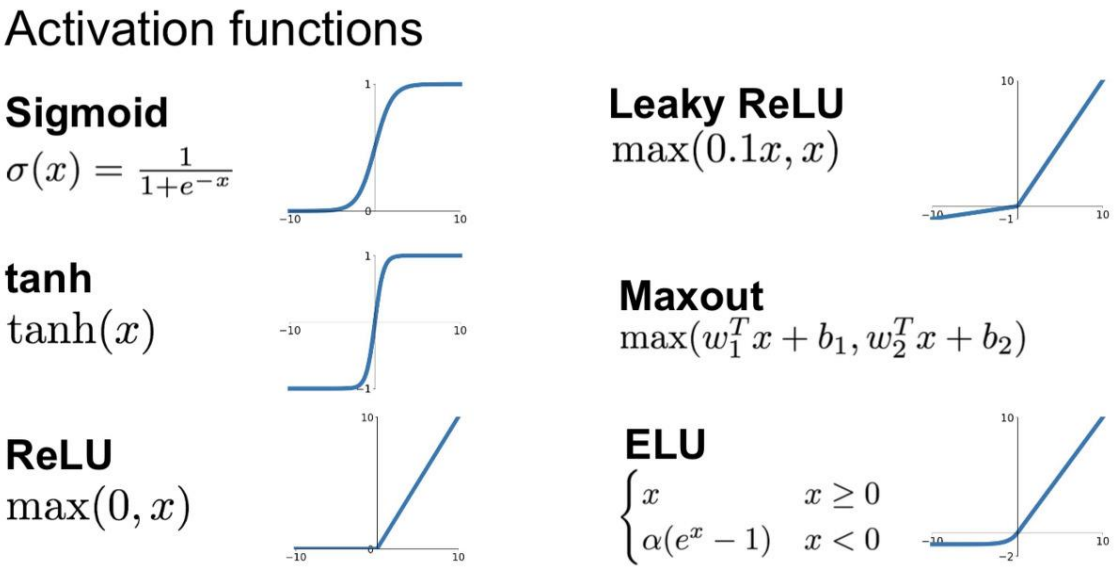


- linear function 을 non-linear 한 다른 함수에 넣어 layer 를 쌓는 방식
- 지난 시간에 배웠던 것으로는 red car 에 대해서만 분류가 되었는데, multi layer 를 이용하면 yellow/blue car 등 다양한 종류의 자동차가 동일한 자동차 class 로 분류될 수 있음
- 각 layer 마다 non-linear function 이 들어가는데, 이는 이후 강의에서 설명해준다고 함
- 여기서는 max function 이 쓰였는데, 이는 ReLU function 임
- W1 은 input 과 직접적으로 연결되어 있고, W2 는 h 의 score 라 생각하면 됨
- layer 를 하나씩 더 쌓으려면, 점점 ReLU function 을 위에 덧씌우면 됨

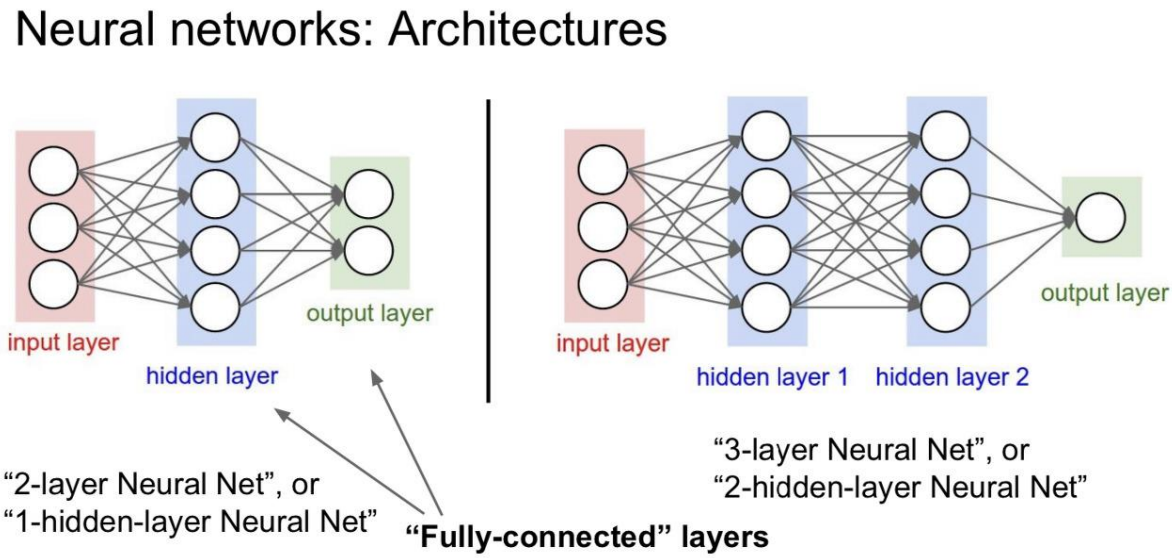
인간의 신경망과의 비교



- x0 는 인간 신경망에서의 입력 신호와 같음
- 시냅스처럼 입력신호가 cell body 와 연결됨
- 인간 신경망의 cell body 에서는 들어오는 신호를 종합하는 역할을 하는데, 인공 신경망의 cell body 에서는 weighted sum 을 구하고 bias term 을 더하는 역할을 함
- 생물학적 뉴런은 일정 임계치를 넘으면 다음 뉴런으로 신호를 보내는데, 인공 신경망은 activation function 을 통해 활성/비활성을 표시함



주의) biological 뉴런과 인공 뉴런의 메커니즘은 비슷하지만, loose inspiration 만 있기에 실제 뉴런은 이보다 복잡함. 이는 수많은 non-linear function 으로 이루어지기 때문



- 2 layer 의 경우, input layer 에서 입력값이 주어지고, hidden layer 에서 Wx 가 들어가며, output layer 에서는 ReLU function 의 결과값이 들어가게 됨
- 3 layer 도 동일한 방식