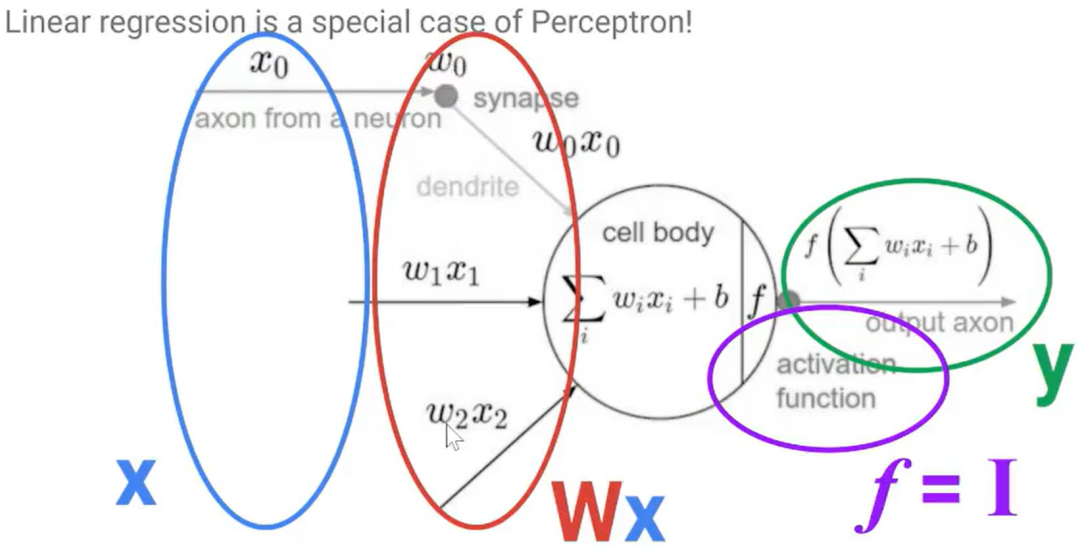
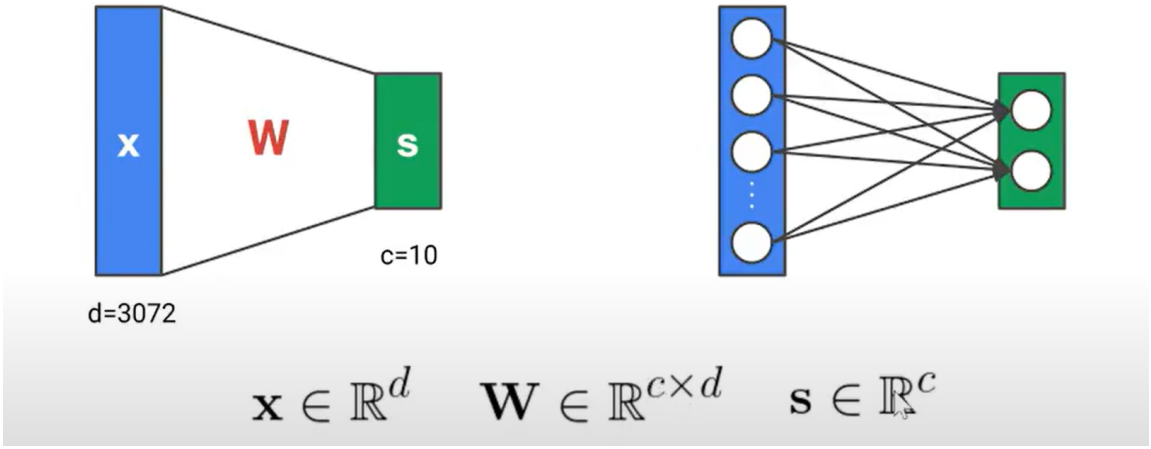
Lecture 4

Neural Network

- Perceptron

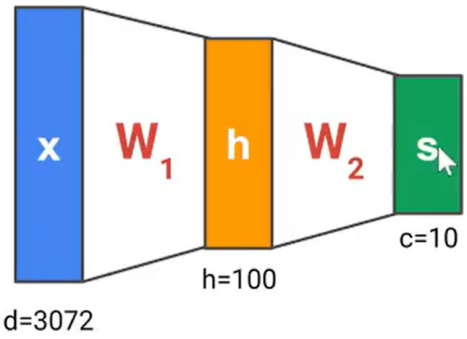


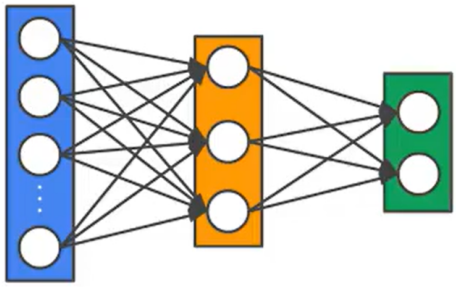
- Neural network with a single layer



- input dimension : 3072
- output class : 10

- Multlayer Perceptron (MLP)





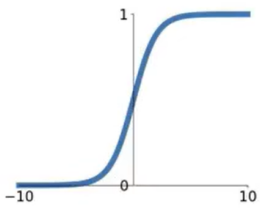
$$\mathbf{x} \in \mathbb{R}^d \quad \mathbf{W}_1 \in \mathbb{R}^{h \times d} \quad \mathbf{W}_2 \in \mathbb{R}^{c \times h} \quad \mathbf{s} \in \mathbb{R}^c$$

- linear model을 중첩시킴
- f(x)=W2(W1x) → 하나의 linear model과 동일한 function

→ 이로 인해 activation function을 만들어 non-linear component를 만든다!

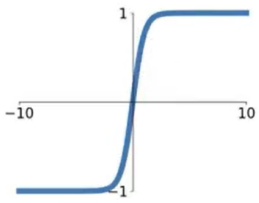
- Activation function
  - 종류

- Sigmoid



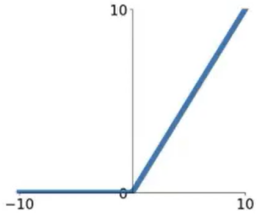
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- tanh



$$\tanh(x)$$

- ReLU



$$\max(0, x)$$

- Relu를 실용적으로 가장 많이 사용함

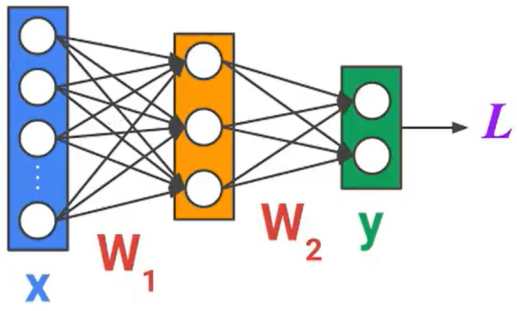
- Computing gradients

What do we need for (Stochastic) Gradient Descent?

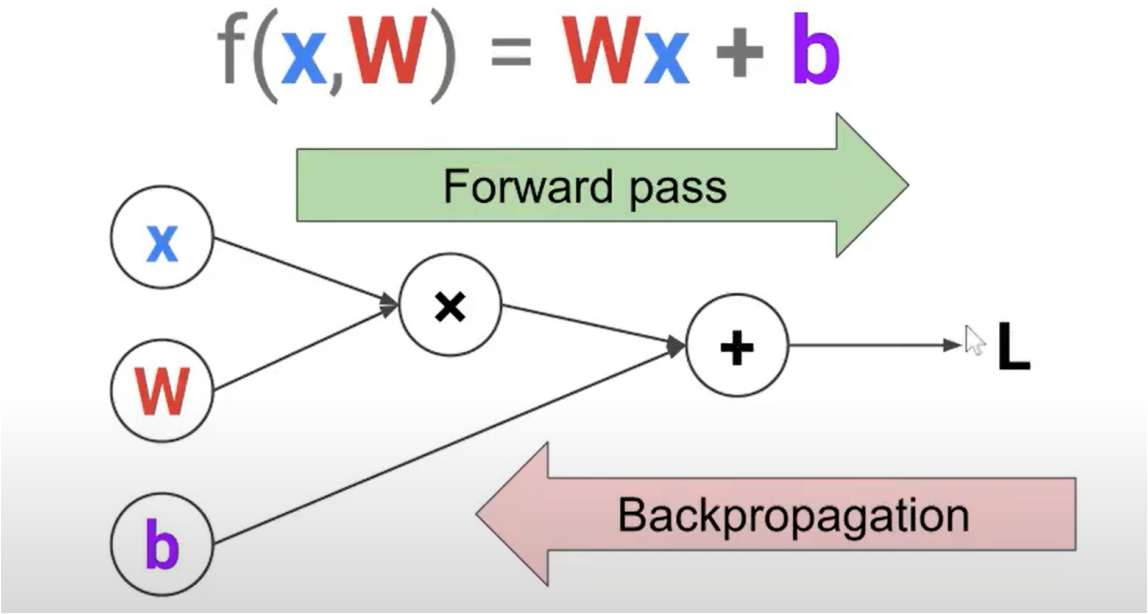
- Gradient of the classification loss w.r.t. **each parameter (weight)**

$$\frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2}$$

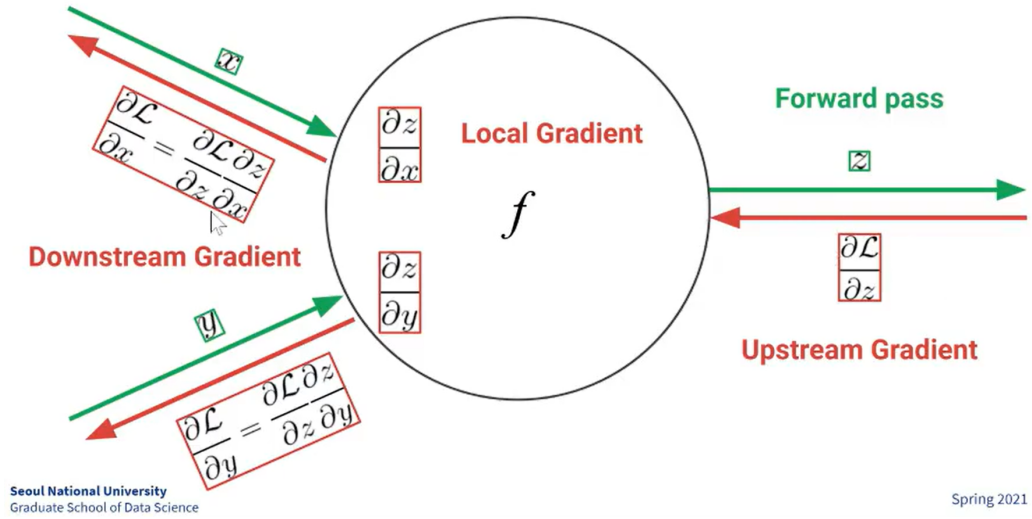
- Each gradient indicates how much that particular weight contributed to the incorrect prediction.



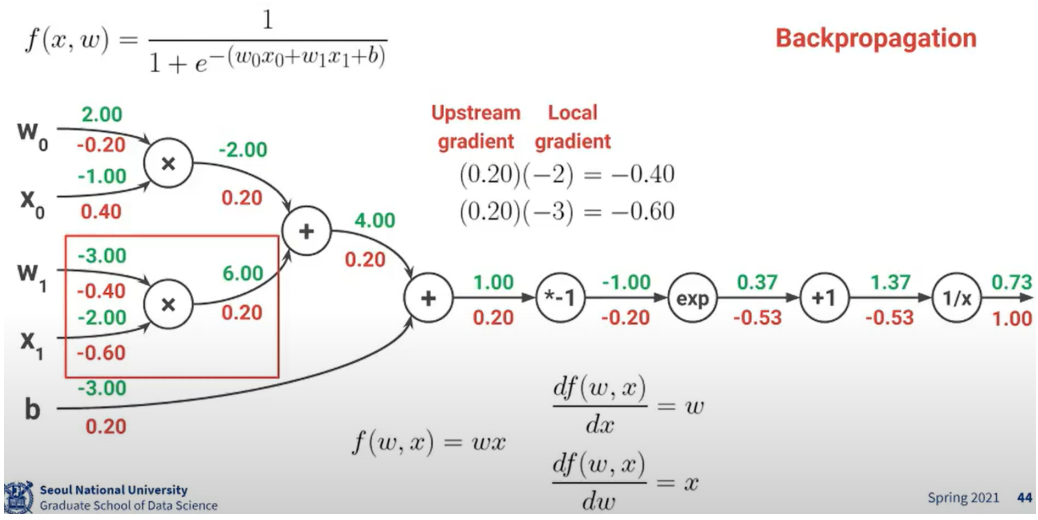
  - loss를 이용해 weight 개선시킴
  - forward pass : 현재 갖고 있는 W,x,b를 이용해 계산한 값



  - back propagation
    - 각각의 parameter들이 loss에 얼마나 기여했는지 계산
    - chain rule 방법



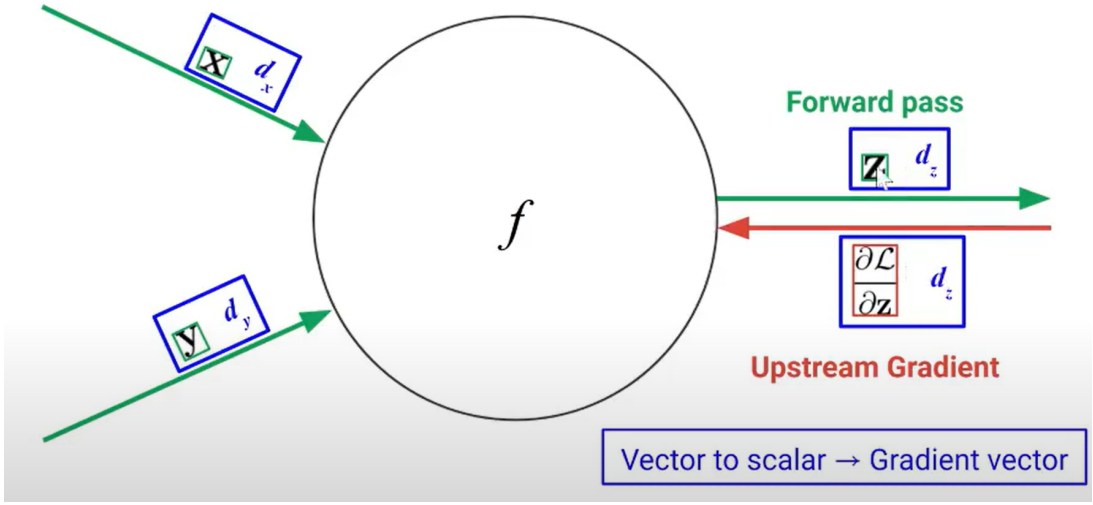
- ex. logistic regression



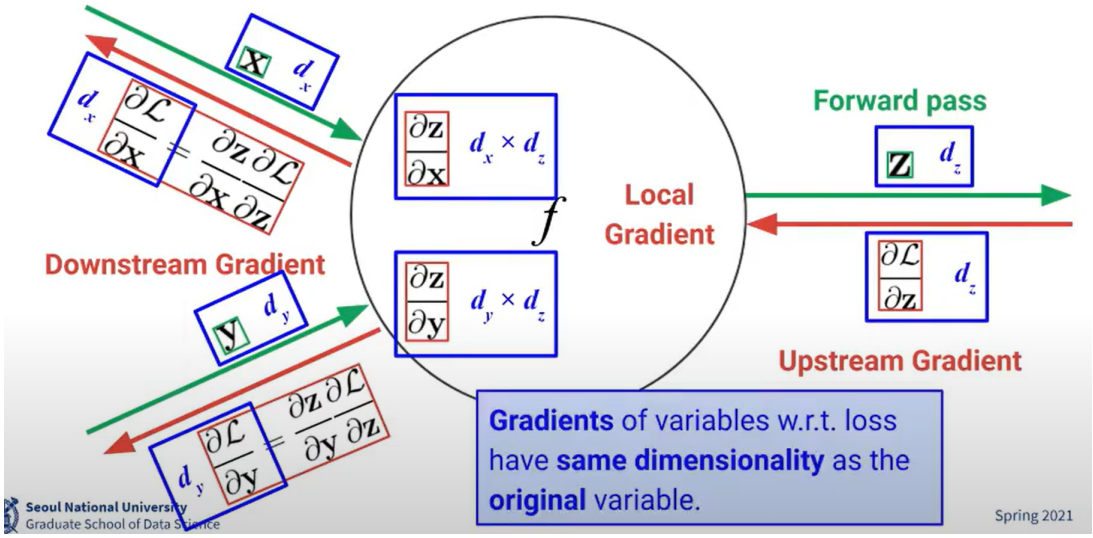
- gate 종류별 back propagation
  - add gate : upstream gradient 그대로 복사
  - mul gate : 상대 노드의 input을 곱해서 out
  - copy gate : upstream gradient들 합쳐서 out
  - max gate : max인 input 값에 upstream gradient 주고 다른 노드에는 gradient가 0

Backpropagation and Vectors and Matrices

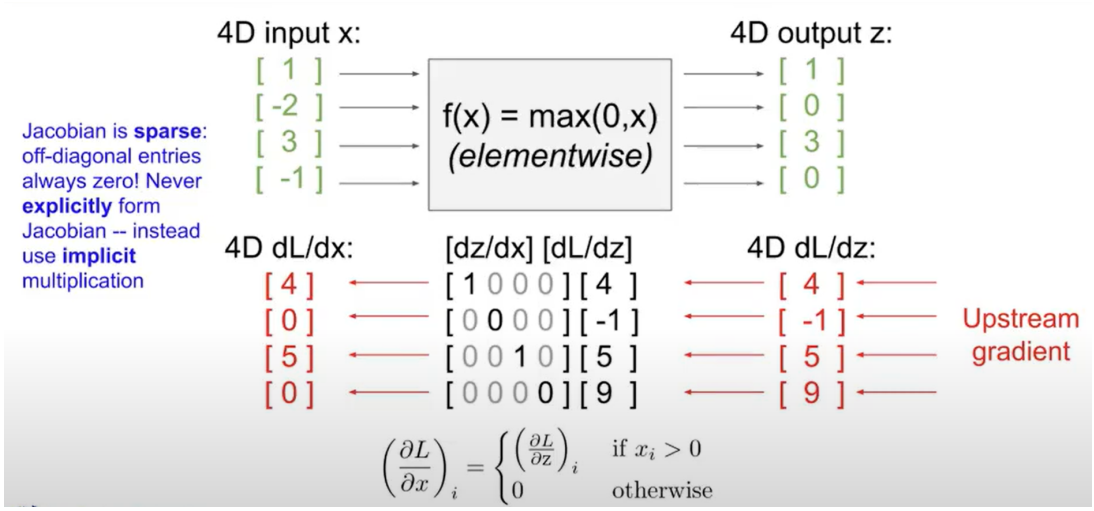
- Backpropagation with vectors
  - loss는 여전히 scalar이기 때문에 vector로 변환해줘야 함.



  - Jacobian matrix 이용해 downstream gradient 계산



  - 이때 dz는 dimension이 1이니까 dx * dz 의 결과 또한 dx의 dimension을 가짐.
  - variable이 갖는 dimension의 크기는 forward, back 항상 같아야 함.
  - example



- Backpropagation with matrices