Linear clasifier의 문제점.

: template (w)가 class당 1개만 존재해서 여러 모습을 갖는 class에 취약하다.
input → output 형태 자체가 Linear boundary만 생성 가능. (정확도↓)

↳ Linearly seperable 하게 특성 변환을 해주는 방법을 쓸수도 있다.
이미지의 feature들을 뽑아서 분류해보자 ⇒ edge, 색상 list ...) 고전적 방법

아무튼 이미지 전체를 사용하기 보다 이미지의 중요 feature만 사용해서 이미지를 분류가능!
↳ 그래서 feature을 어떻게 만들건데?

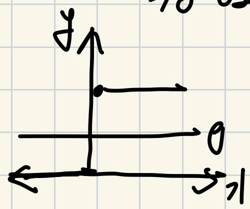
* Model이 스스로 feature를 추출하고 ⇒ 라벨 feature로 예측까지 End-to-End로 설계 하자 라는 아이디어가 deep Learning의 시작이다.
혁명적 접근...!

[End-to-End가 Always 좋을까?]

⇒ 특정 domain의 pattern을 얻기 위해 수많은 resource가 필요. , human이 갖고있는 사전지식인 domain 지식을 주입하면 효율↑ 얻을 수 있다. <사전지식인 domain knowledge이 중요하다>

perceptron ⇒ $y = f(w_1x_1 + w_2x_2)$ 특정 θ보다 크면 1 else 0

$$f(a) = \begin{cases} 0 & (a \geq \theta) \\ 1 & (a < \theta) \end{cases}$$

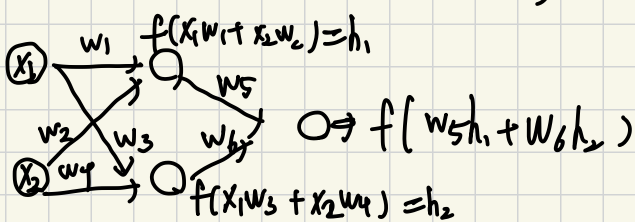


특정 θ를 넘기까지 자극(신호)를 넘겨주는 방법을 그대로 사용.

Single perceptron의 장벽: "XOR"을 만들수가 없다.

↳ Multi perceptron으로 구현 가능!

$$y = f(w_5 f(w_1x_1 + w_2x_2) + w_6 f(w_3x_1 + w_4x_2))$$



MLP (Multi Layer perception)

MLP에서 역할은 ≈ activation function을 사용하지 않으면 Single perceptron과 동일하다.

$$f(W_2(W_1x)) = f((W_2W_1)x) = f(Wx)$$ 가능하기에

그래서 MLP의 핵심은 activation function이다. = non-Linear function.

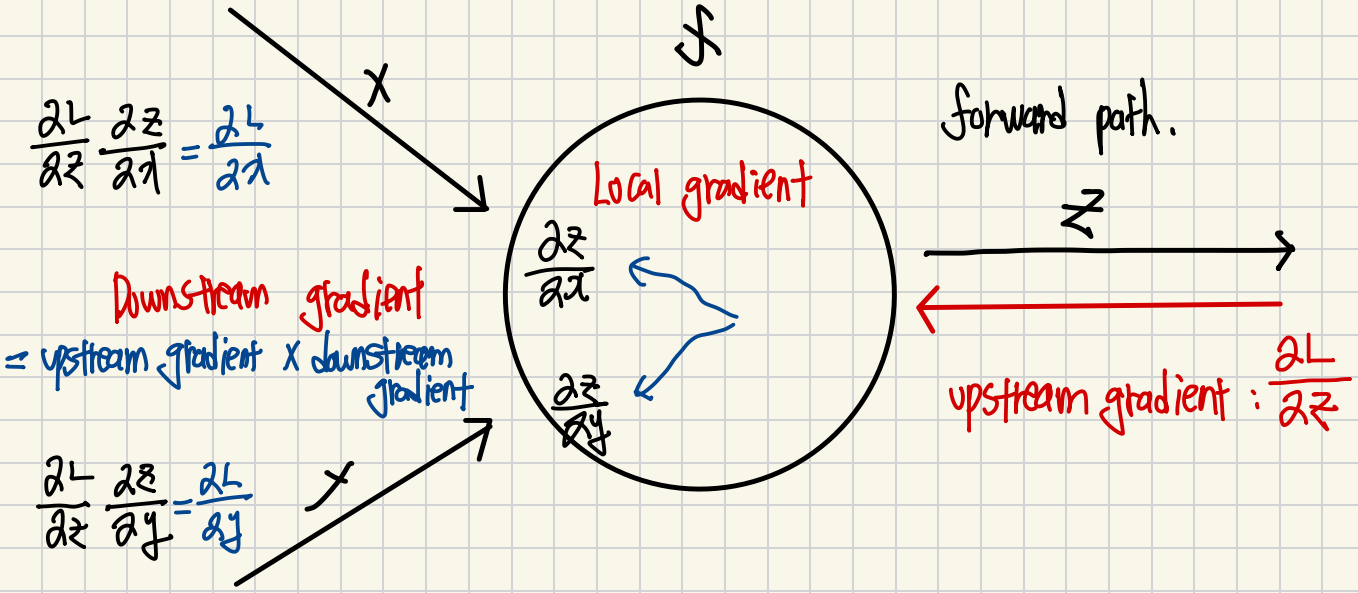
Computing gradients (W 학습하기)

gradient는 Loss function을 minimize 하도록 W를 바꾸자

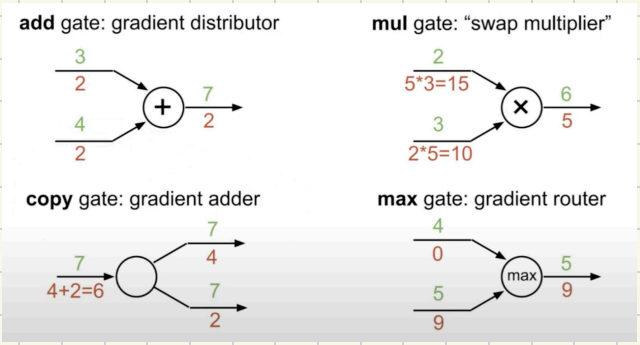
$$\hat{y} = W_2 \cdot \sigma(W_1x_1) \rightsquigarrow \frac{\partial L}{\partial W_1} \& \frac{\partial L}{\partial W_2}$$ 을 계산한다.

Backpropagation

Chain Rule에 의해서 잘게 쪼개진다.



Gradient pattern.



⊕ ⇒ 그냥 gradient를 똑같이 흘려보낸다

⊗ ⇒ gradient를 change해서 보내준다.

(max) ⇒ input이 큰 애한테만 gradient를 보낸다.

copy ⇒ gradient를 sum해서 흘려보낸다.

스칼라는 vector의 일반화 ⇒ vector

vector는 vector의 일반화 ⇒ matrix