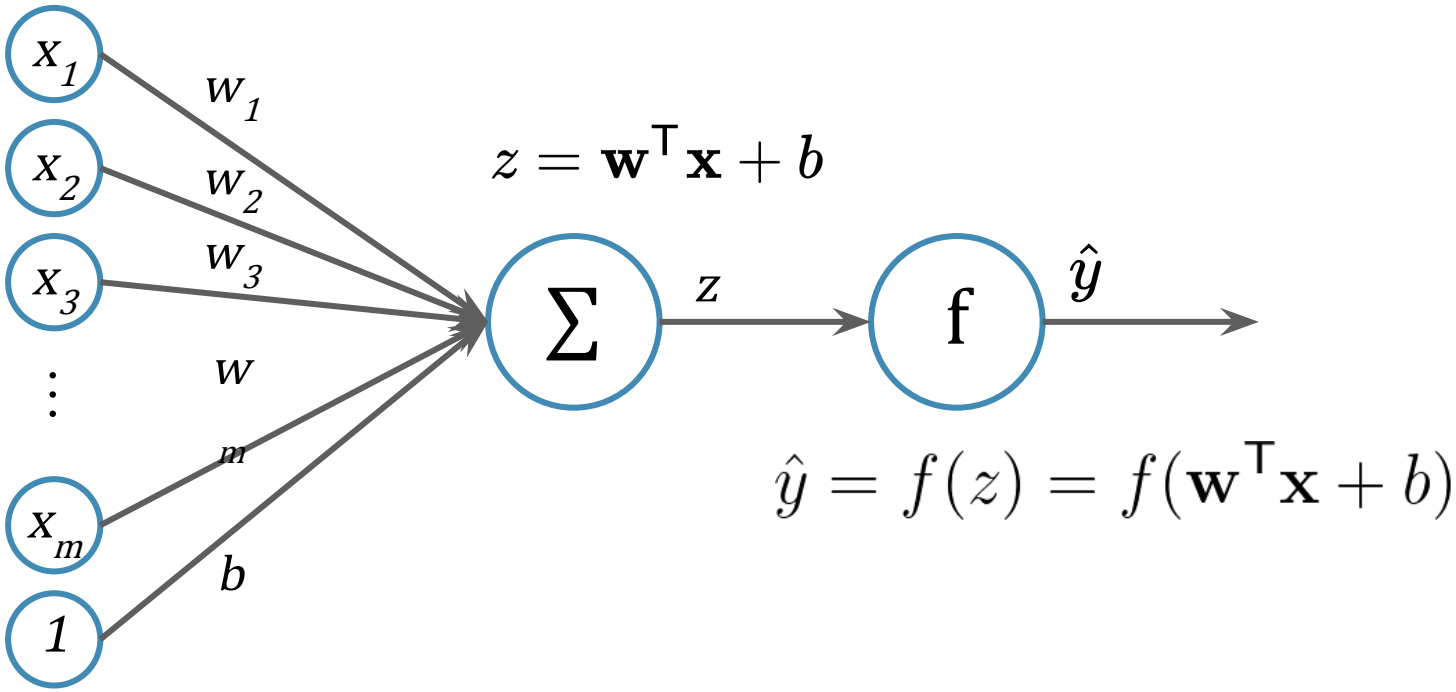
인공지능 개론
L09 Deep Neural Network

국민대학교
소프트웨어융합대학원
박하명

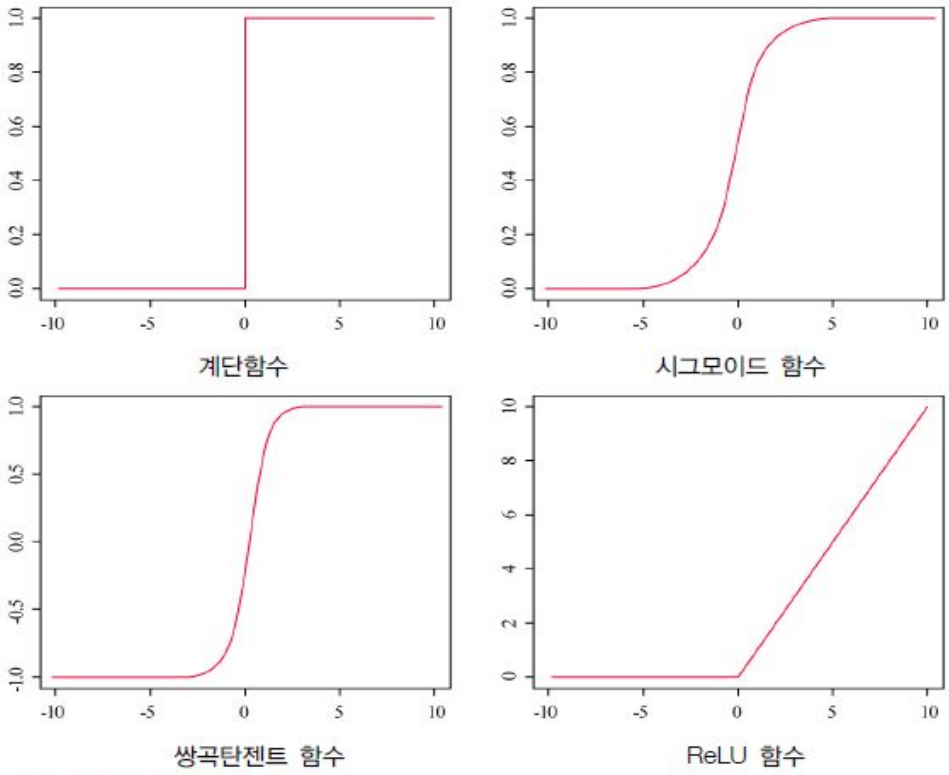
인공 뉴런 (복습)

- **인공 뉴런**: 인공신경망의 기본 정보 처리 단위
- 여러개의 입력 신호에 가중치를 곱하여 합하고 활성함수에 넣는다.
- 활성함수의 결과는 다른 인공뉴런의 입력이 되거나 출력된다.



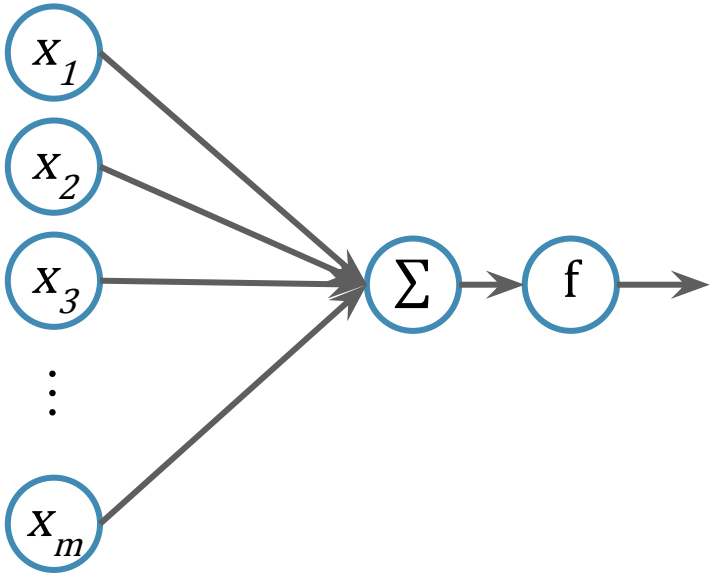
활성함수 Activation function (복습)

- 활성함수로 어떤 함수든 사용 가능
- 예) 계단함수, 시그모이드 함수, 쌍곡탄젠트 함수,
  ReLU (Rectified Linear Unit) 함수



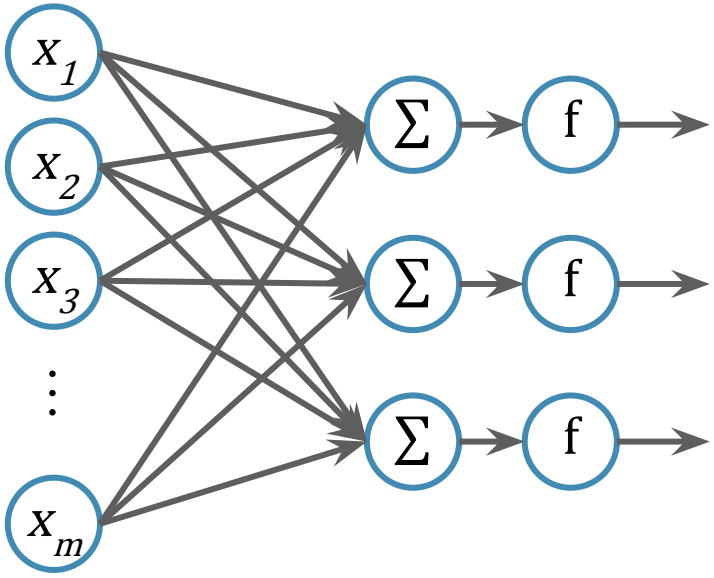
인공신경망: 여러개의 인공 뉴런 (복습)

- 인공 뉴런 하나가 단독으로 사용되어 학습할 수 있다.
  - 예1) Linear Regression (활성함수가 시그모이드 함수인 인공 뉴런)
  - 예2) 로젠블랫의 퍼셉트론 (활성함수가 계단함수)



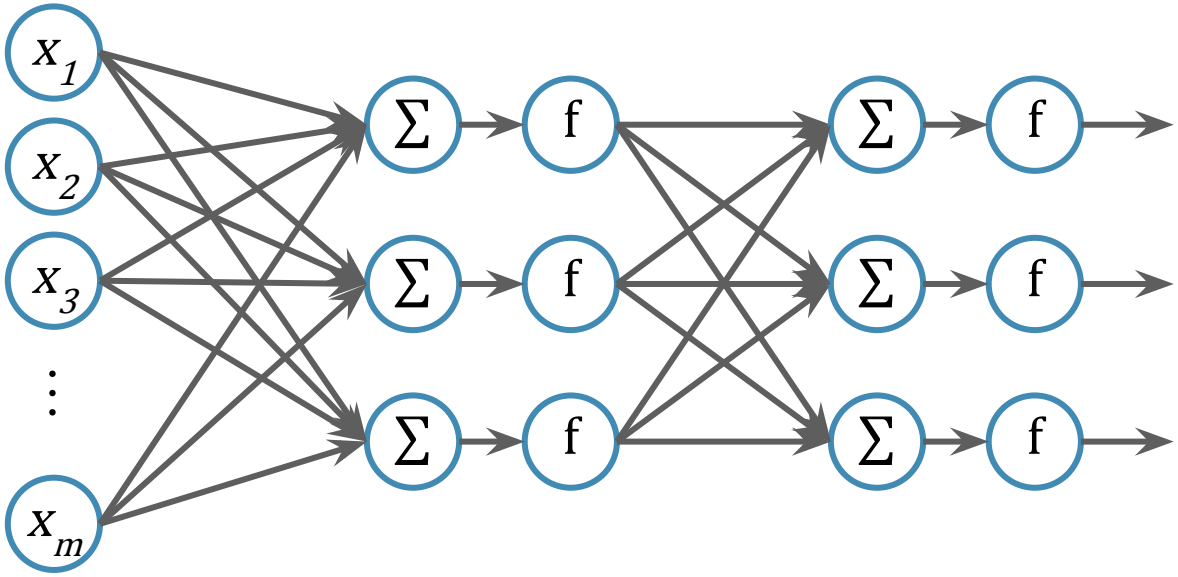
인공신경망: 여러개의 인공 뉴런 (복습)

- 인공 뉴런이 **병렬적으로 연결**되어 여러 개의 출력을 가질수도 있다.
  - 예) Softmax Regression



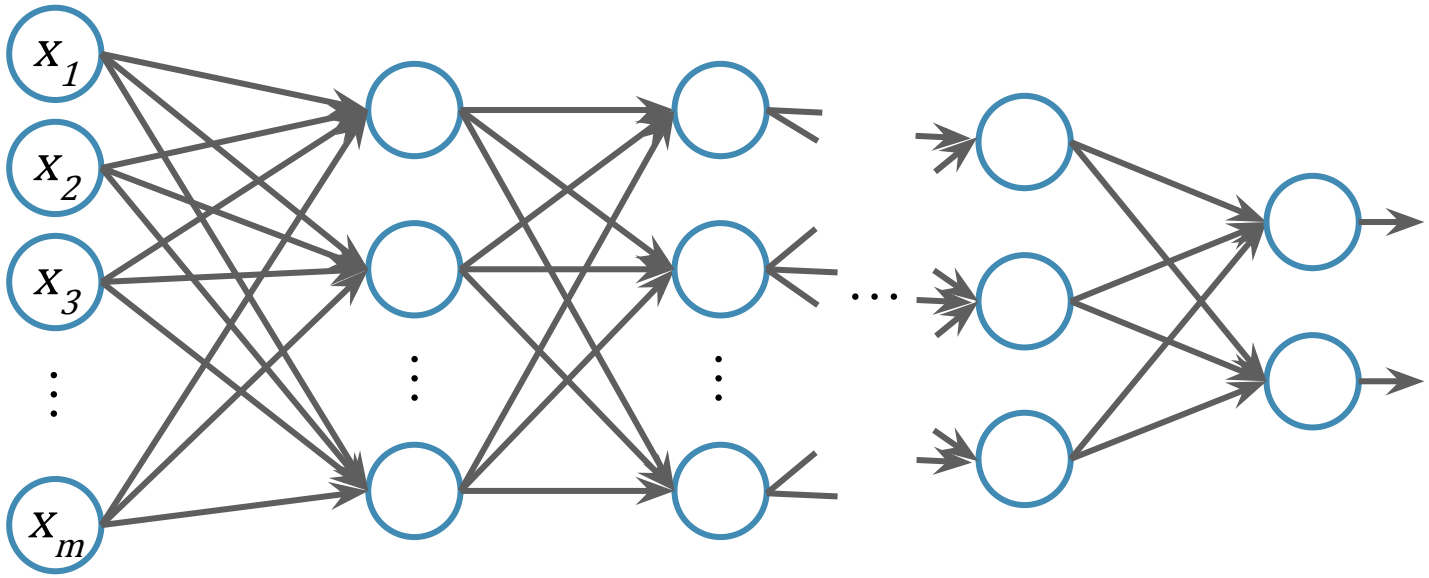
인공신경망: 여러개의 인공 뉴런 (복습)

- 인공 뉴런의 출력은 다른 인공 뉴런의 입력으로 활용될 수 있다.
  - 보다 복잡한 문제를 해결할 수 있다.
  - 예) 1개의 층으로 된 인공신경망은 xor 문제를 학습할 수 없다..!



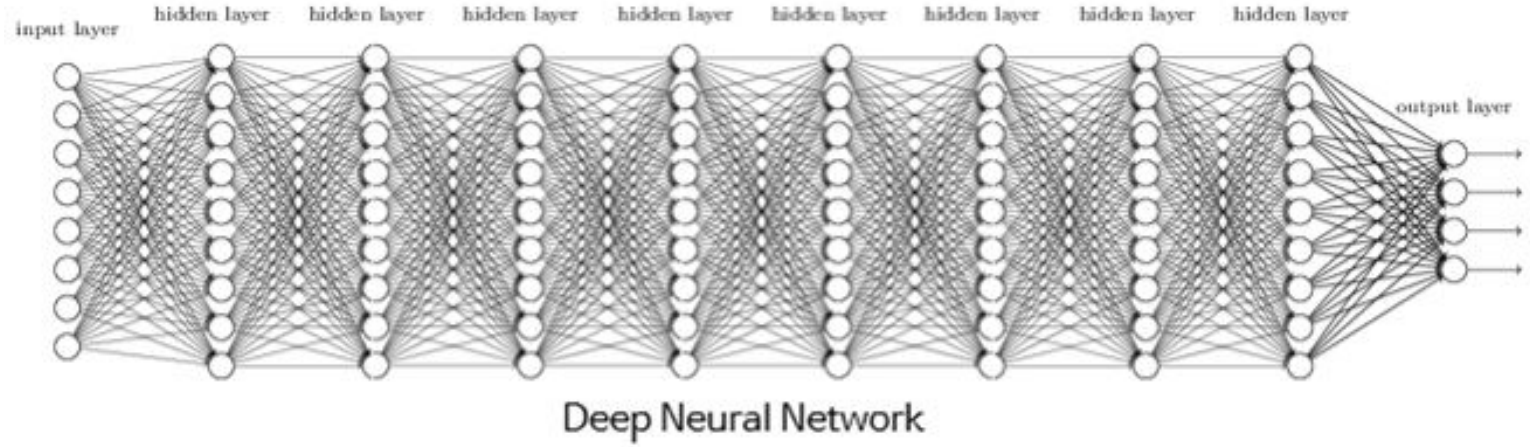
Deep Neural Networks (복습)

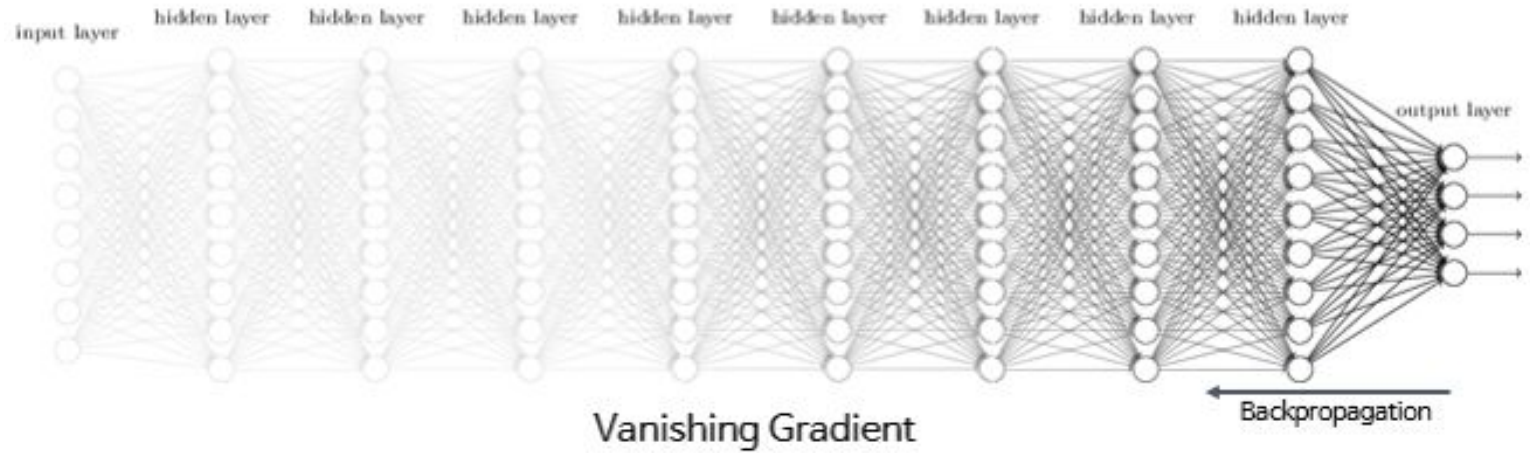
- 층이 많아질 수록 더욱 더 복잡한 문제를 해결 가능하다…!
  - 하지만… 계산 복잡도가 크게 증가하여 학습이 오~래걸린다..!
    ⇒ GPU 등의 고성능 컴퓨팅 자원을 활용하여 해결~!



Vanishing Gradient Problem

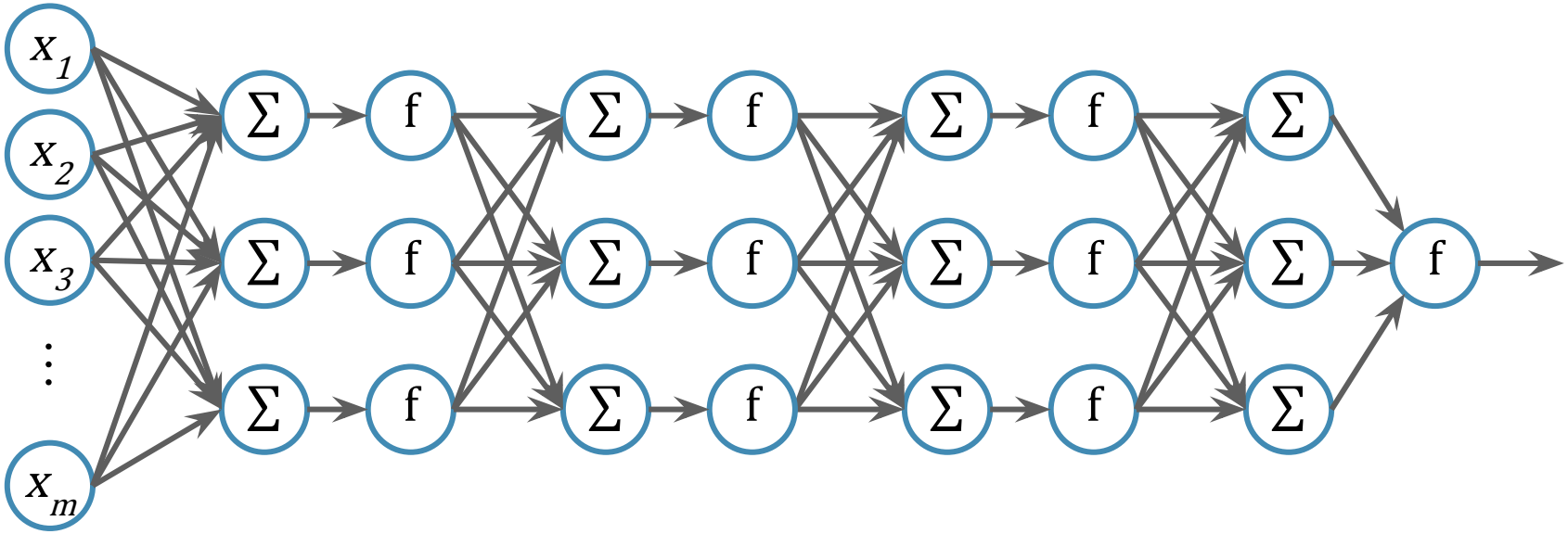
- 깊은 신경망의 더 큰 문제는, **학습이 안될 수 있다는 것…**





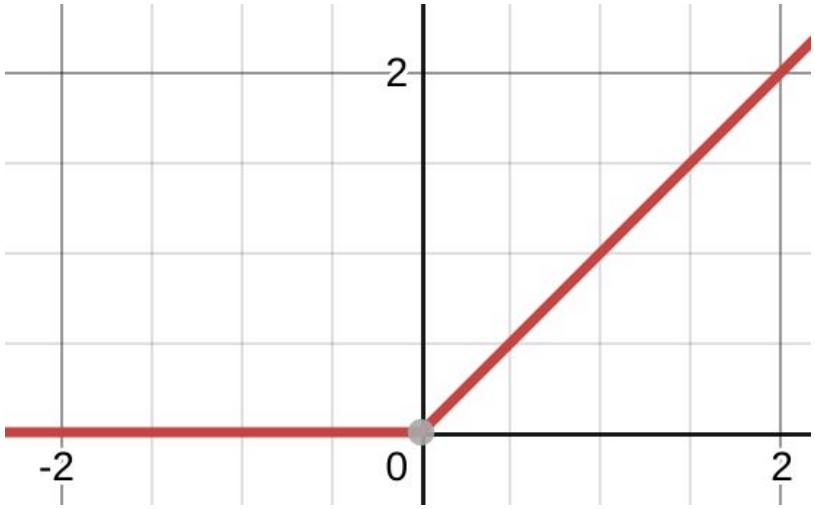
Vanishing Gradient Problem

- 문제는 활성함수가 **sigmoid**라는 것
  - sigmoid를 거치면 출력이 0~1사이로 좁아진다!
  - 앞쪽의 변수들은 최종 출력에 영향을 거~의 주지 못한다.
    - 기울기를 계산해보면 거~~의 0이다.



ReLU

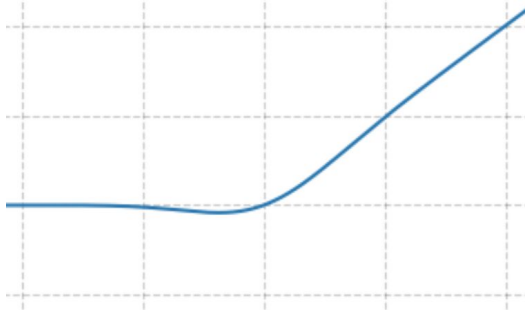
- 해결방법: 활성함수를 **ReLU** (Rectified Linear Unit)으로 바꾼다!
  - ReLU는 입력이 음수일때 0을 내보낸다… → Dead node 발생 가능
  - 질문! 그냥 활성함수를 없애면 안될까요..?



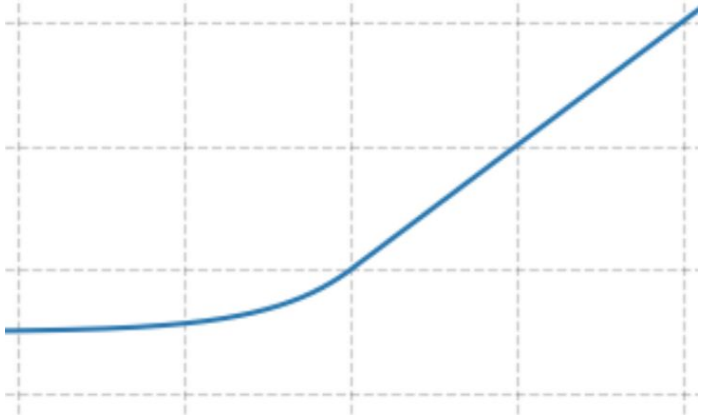
$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU의 변형들

LeekyReLu(x) = max(0.01x, x)



GeLU(x) = x *Φ(x)

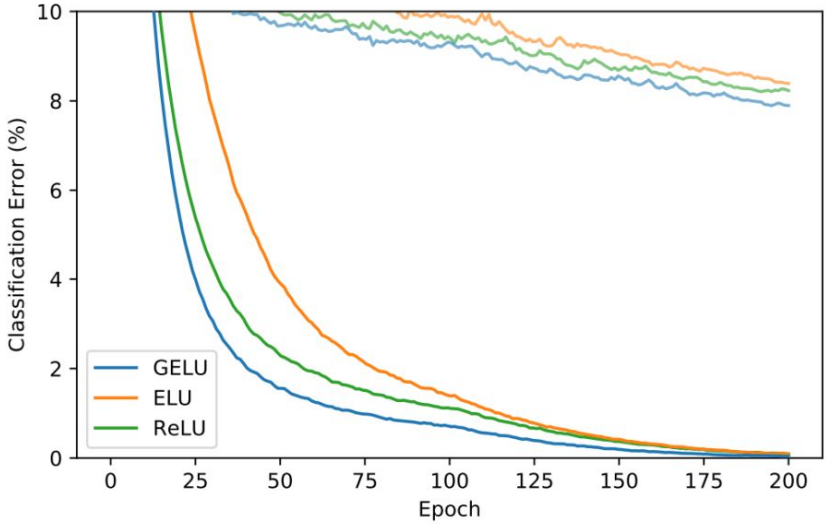


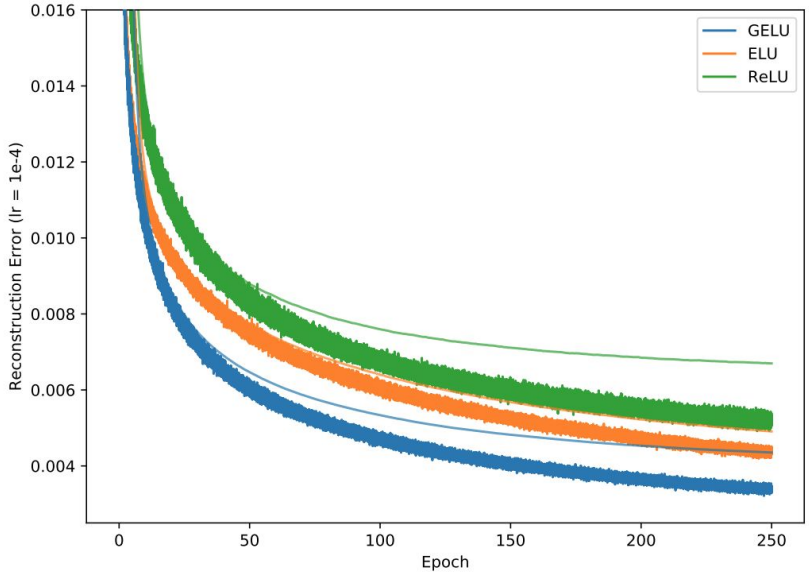
ELU(x)=max(0,x)+min(0,α∗(exp(x)−1))

CeLU
SeLU
RReLU
PReLU
Softplus
tanh
arctan
…

ReLU의 변형들

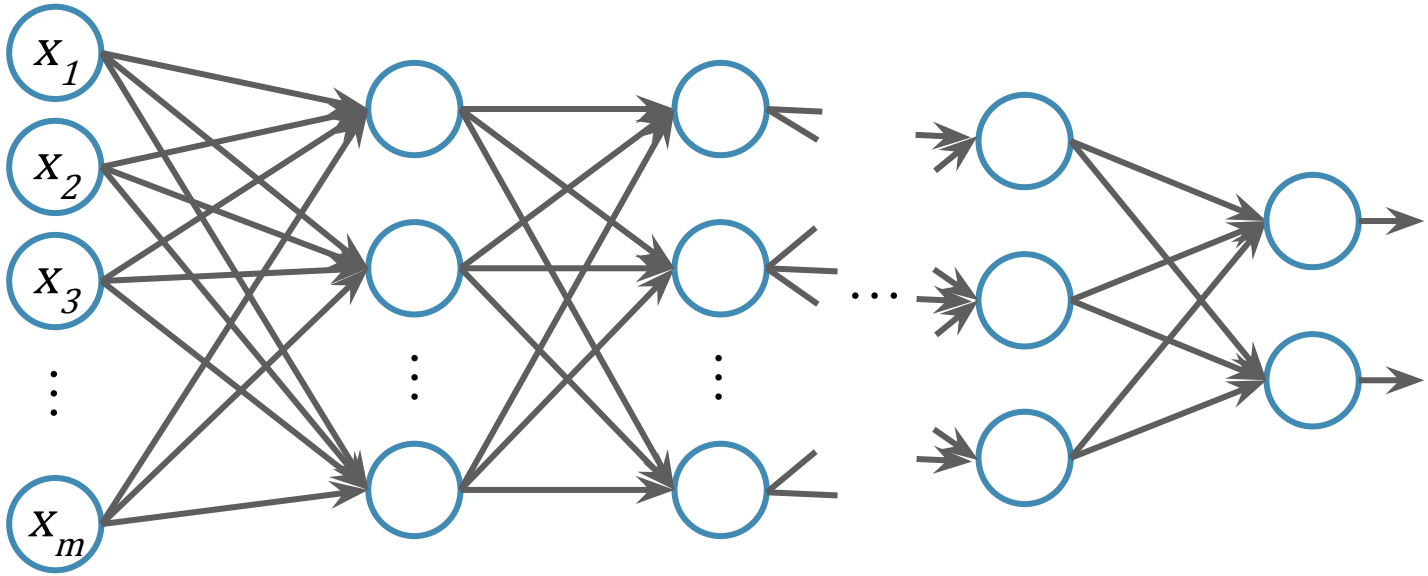
- GeLU가 젤루 좋다(고 주장)
  https://arxiv.org/pdf/1606.08415.pdf (2018)





초기값 설정

- 가중치의 초기값을 모두 0 으로 설정한다면..?
  ⇒ 학습이 안된다. Why?
- 랜덤하게 설정한다면? 적당히 괜찮을지도...?
  - 더 좋은 방법: Xavier Initialization, He Initialization



초기값 설정

Xavier Init.:
Glorot, X. & Bengio, Y. (2010)

$$\mathcal{U}(-a, a) \qquad a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

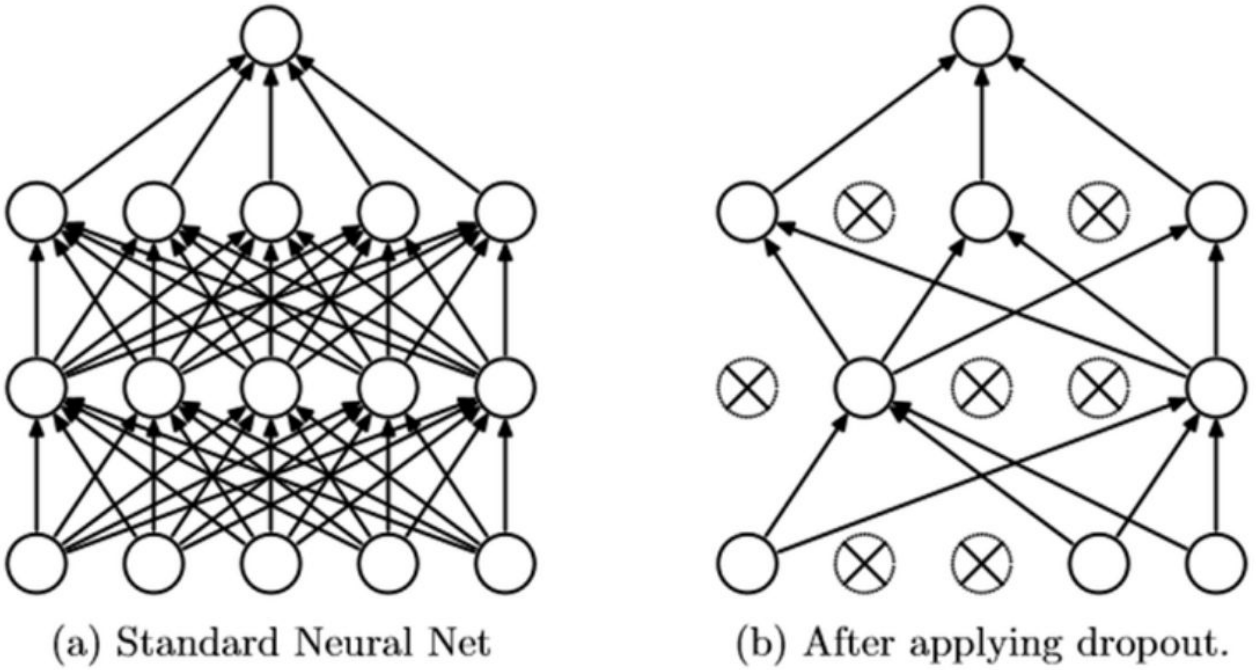
He Init.:
He, K. et al. (2015)

$$\mathcal{U}(-\text{bound}, \text{bound})$$

$$\text{bound} = \text{gain} \times \sqrt{\frac{3}{\text{fan\_mode}}}$$

Dropout: 과적합 방지

- **학습할 때** 내부 노드를 **랜덤하게 몇 개 없애자**
  - 이게 왜 잘될까..?



(a) Standard Neural Net

(b) After applying dropout.

Question?