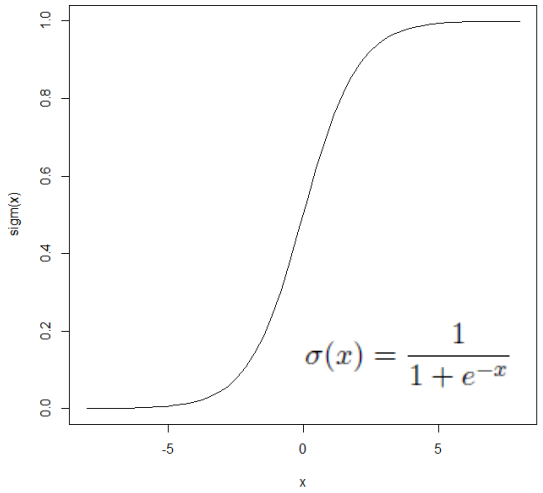
Some discovery of deep learning

서울대학교 통계학과 김동하

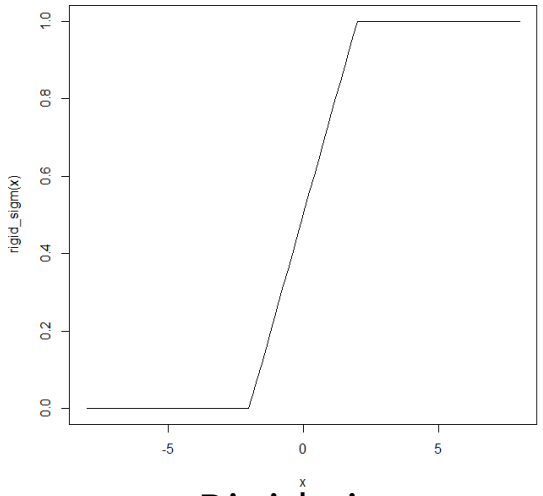
Contents

- Sigmoid function을 사용한 DNN에 대한 이해



Sigm

≈



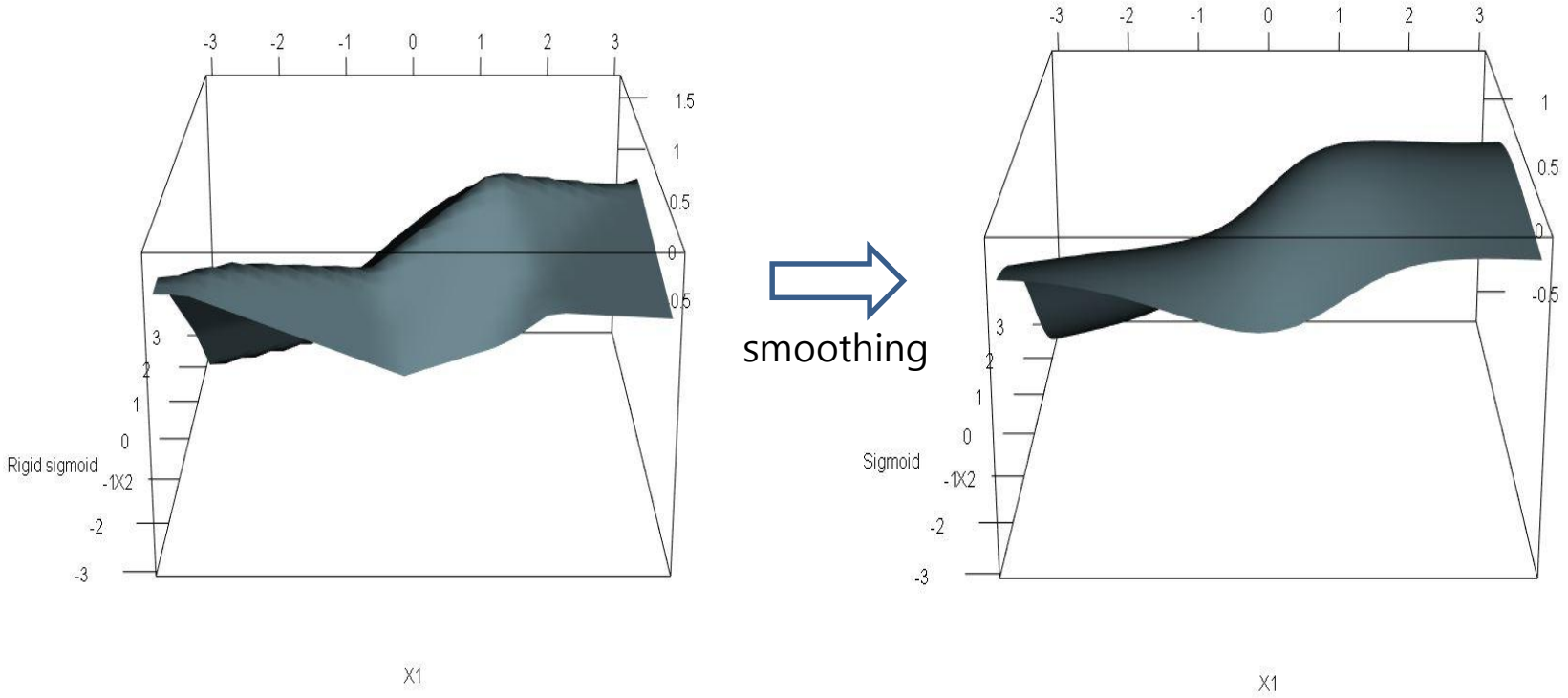
Rigid sigm

- Rigid sigm를 이용하여 만든 DNN : piecewise linear function

✓ Sigm를 이용하여 만든 DNN : piecewise linear function의 smoothing version.

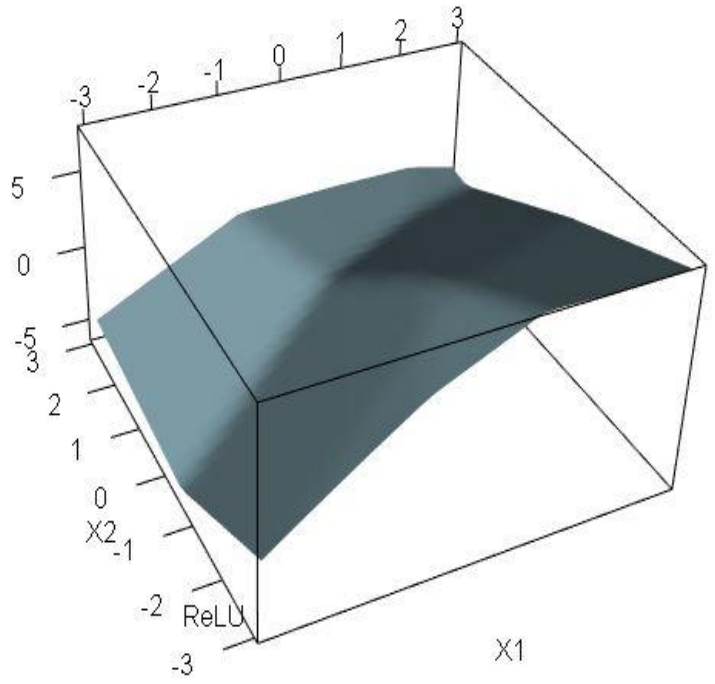
- Sigmoid function을 사용한 DNN plot

- Example
  - 2차원의 input, 5차원의 hidden layer 한 층을 사용한 neural network



- ReLU function을 사용한 DNN plot

- Example



Contents

- Universal approximation theorem (G.Cybenko, 1989)

Theorem 2. *Let σ be any continuous sigmoidal function. Then finite sums of the form*

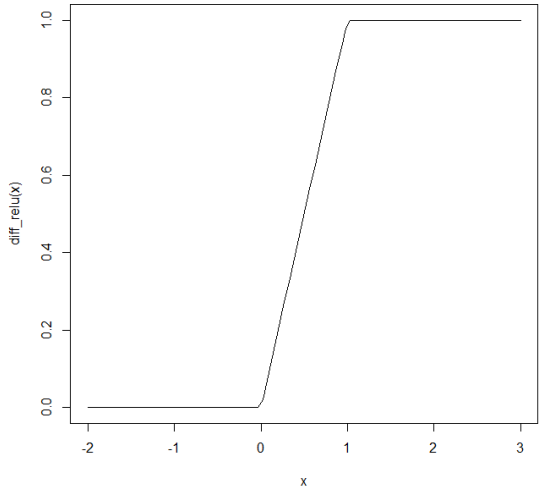
$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^{\mathrm{T}} x + \theta_j)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \varepsilon \qquad \text{for all} \quad x \in I_n.$$

Definition. We say that σ is *sigmoidal* if

$$\sigma(t) \to \begin{cases} 1 & \text{as} \quad t \to +\infty, \\ 0 & \text{as} \quad t \to -\infty. \end{cases}$$



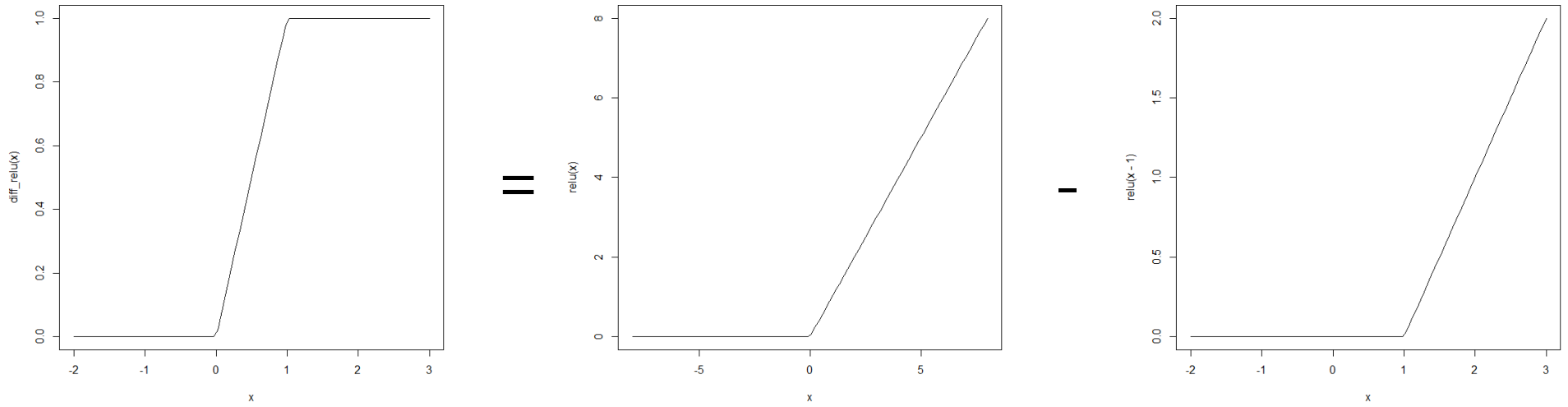
Diff ReLU is also continuous sigmoidal function

- ReLU as universal approximating activation function

- Universal approximation theorem에 의해, diff ReLU를 활성함수로 사용하는 NN은 임의의 연속함수를 근사할 수 있음. (G.Cybenko, 1989)

✓ ReLU와 diff ReLU 사이의 관계

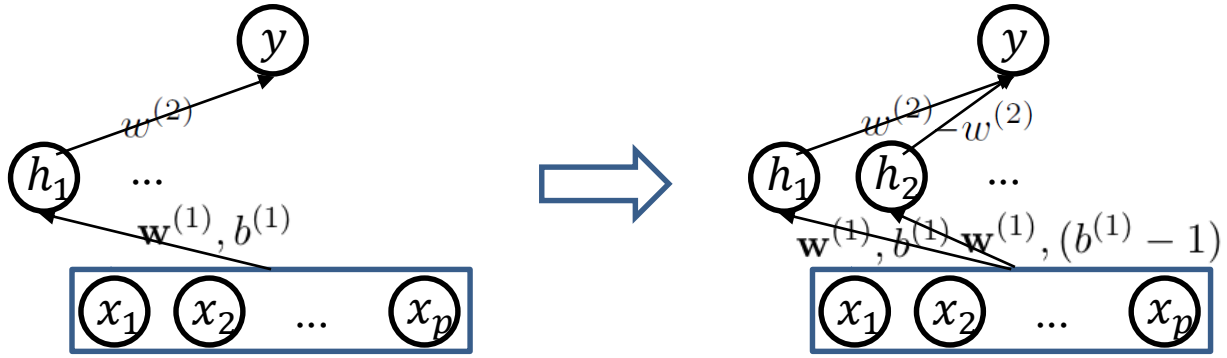
$$diff\,ReLU(x) = ReLU(x) - ReLU(x-1)$$



- 따라서, ReLU를 활성함수로 사용하는 NN 또한 **<u>임의의 연속함수를 근사할 수 있음.</u>**

- ReLU as universal approximating activation function (cont.)

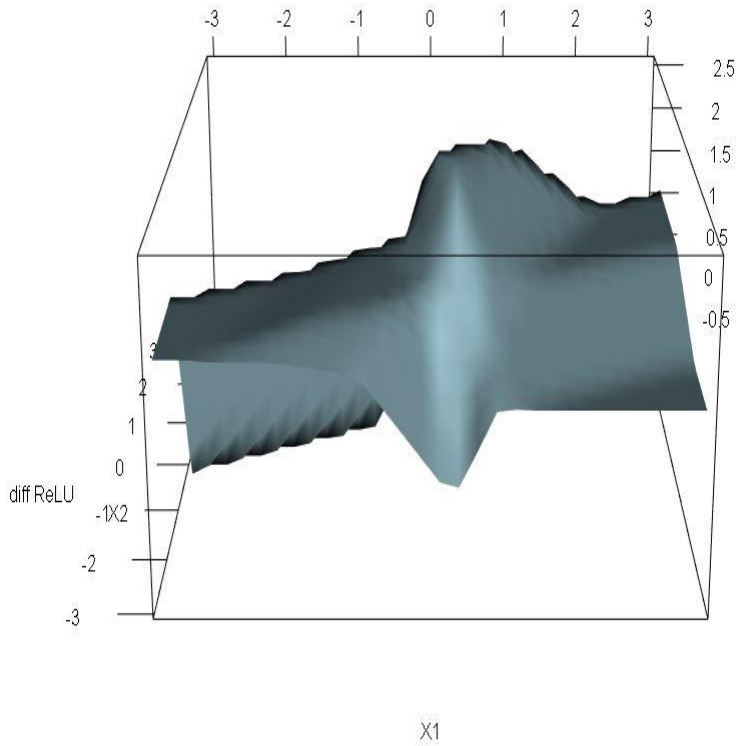
$$w^{(2)} \cdot diff\,ReLU\left(\mathbf{w}^{(1)'}\mathbf{x} + b^{(1)}\right) = w^{(2)} \cdot ReLU\left(\mathbf{w}^{(1)'}\mathbf{x} + b^{(1)}\right) + (-w^{(2)}) \cdot ReLU\left(\mathbf{w}^{(1)'}\mathbf{x} + (b^{(1)} - 1)\right)$$



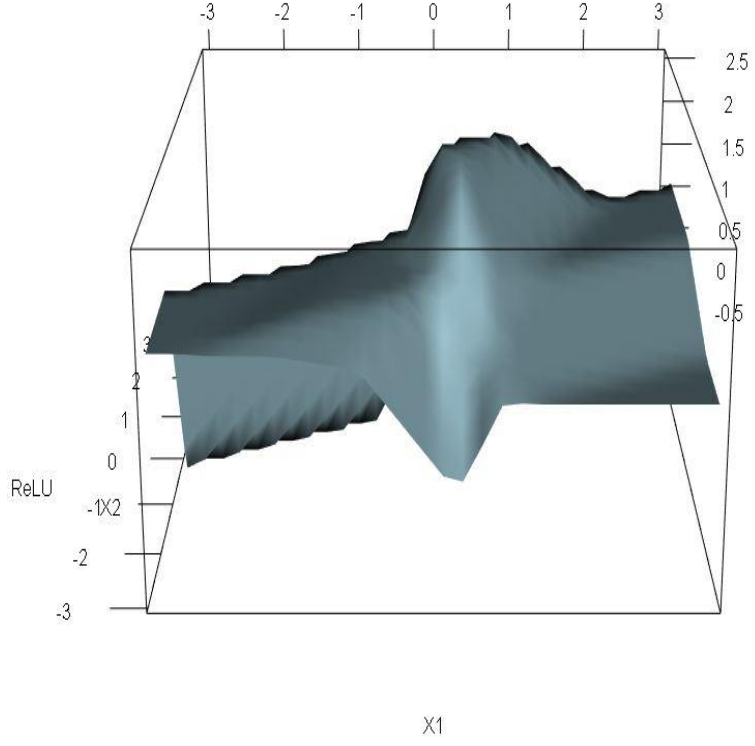
- Diff ReLU와 n_1개의 hidden node를 사용하여 만든 함수를 ReLU와 $2n_1$개의 hidden node를 사용하여 만든 함수로 정확히 표현할 수 있음.

- ReLU and diff ReLU plots

Example



NN using diff ReLU



NN using ReLU

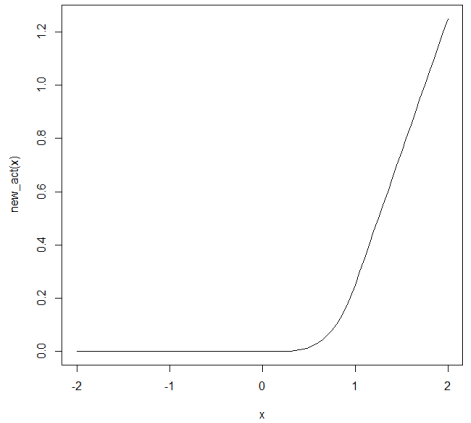
- **고려해볼 사항들.**

- Universal approximation이 가능한 활성함수는 ReLU뿐만 아니라 차분을 했을 때 continuous sigmoidal 조건을 만족하는 함수라면 모두 가능.
  - Leaky ReLU, pReLU, ELU 등..

- S.Sho and N.Murata(2015)에서도 증명이 나왔으나, 증명 과정이 복잡..(추후에 읽어볼 예정)

✓ Unbounded act. ftn이 bounded act. ftn.보다 성능이 좋은 이유.
1. Bounded act. ftn.을 사용할 경우, 좋은 초기값 찾기가 힘듦.
2. Vanishing gradient problem
3. Unbounded act. ftn.을 사용할 경우 deep learning에 필요한 함수를 더 효율적으로 만들 수 있음.

✓ 좋은 활성함수의 조건.
1. Universal approximator
2. SGD의 수렴성 보장 (L.Bottou, 1998)
3. 좋은 성능 (unboundedness와 관련이 있을듯..)
4. Node selection의 기능 (ex. ReLU)



Contents

• Notation

- $\xi_j^{(l)} \sim_{iid} 2 \cdot Ber(0.5), \quad l = 1, ..., L, j = 1, ..., n_l$
- $\boldsymbol{\xi}^{(l)} = (\xi_1^{(l)}, ..., \xi_{n_l}^{(l)})'. \quad l = 1, ..., L$
- $\boldsymbol{\xi} = (\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(L)})'$
- $f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta})$: $f(\mathbf{x}; \boldsymbol{\theta})$의 drop-out version.
- $\hat{P}$: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$의 empirical distribution.

• Drop–out with square loss

- Drop–out learning procedure :

  $\min_{\boldsymbol{\theta}} \mathbb{E}_{\hat{P}} \mathbb{E}_{\boldsymbol{\xi}} [y - f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta})]^2$ 의 SGD version. (S.Wager et al., 2013)

- Sigm function을 활성함수로 사용한 DNN의 경우
  $\mathbb{E}_{\boldsymbol{\xi}}[f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta})] \approx f(\mathbf{x}; \boldsymbol{\theta})$ (P.Baldi and P.J.Sadowski, 2013)

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\xi}}[y - f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta})]^2 &= y^2 - 2y\mathbb{E}_{\boldsymbol{\xi}} f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta}) + \mathbb{E}[f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta})^2] \\ &\approx [y - f(\mathbf{x}; \boldsymbol{\theta})]^2 + Var_{\boldsymbol{\xi}} f_{\boldsymbol{\xi}}(\mathbf{x}; \boldsymbol{\theta}) \end{aligned}$$

- Drop–out with square loss (cont.)

$$\mathbb{E}_{\hat{P}}\mathbb{E}_{\boldsymbol{\xi}}[y - f_{\boldsymbol{\xi}}(\mathbf{x};\boldsymbol{\theta})]^2 \approx \underbrace{\mathbb{E}_{\hat{P}}[y - f(\mathbf{x};\boldsymbol{\theta})]^2}_{\text{Sq. loss}} + \underbrace{\mathbb{E}_{\hat{P}} Var_{\boldsymbol{\xi}} f_{\boldsymbol{\xi}}(\mathbf{x};\boldsymbol{\theta})}_{\text{Penalty term}}$$

- Penalty term :
  - $\boldsymbol{\theta}$ 의 크기가 커질 수록 증가.
  - $\boldsymbol{\theta}$ 의 크기가 작아질 수록 감소. 모두 0이 되면 penalty term 또한 0이 됨.

✓ Linear regression일 경우 위의 식은 정확히 ridge regression이 됨.
✓ Logistic regression일 경우에도 loss function을 negative likelihood로 할 경우 2차 근사하면 ridge regression이 됨. (S.Wager et al., 2013)

• Drop-out loss에 대한 해석.

- Penalty term의 역할
  - θ의 크기를 제한시키는 역할 (ridge penalty처럼..)

✓ 특정 θ의 원소의 크기가 클 경우.
- 특정 node의 영향력이 지나치게 커질 수 있음. (나머지 node들의 효과 미미..)
- 좋지 않은 성능으로 이어질 가능성 증가.

- Drop-out learning procedure는 co-adaptation을 제한한다기보다는 특정 weight 또는 node들의 영향력이 지나치게 커지는 것을 제한하는 학습 방법이 아닐까..

- **고려해볼 사항들.**

- Drop-out을 사용하지 않았을 때와 사용했을 때의 weight들의 크기 비교 및 상관 관계 비교.
- 분류 문제에서의 drop-out learning procedure의 의미 확인.
- Drop-out의 모수 추정값 수렴성 여부 및 수렴 속도 확인.
- 활성함수를 sigm가 아닌 ReLU로 사용하였을 때 같은 논리 전개가 가능한지 확인.