Universal Approximation Theorem 이자

Cybenko theorem

Theorem 9.3.6 (Cybenko, 1989) *Let σ be an arbitrary continuous sigmoidal function in the sense of Definition 2.2.1. Then the finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + \theta_j), \qquad w_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}$$

are dense in $C(I_n)$.

dense 하다.

모든 카페가 아메리카노를 취급한다.

충분한 시간과 노력을 들인다면

나에게 딱 맞는 아메리카노를 만드는 카페를

찾을 수 있을 것이다.

Definition 2.2.1 *A function $\sigma : \mathbb{R} \to [0,1]$ is called sigmoidal if*

$$\lim_{x \to -\infty} \sigma(x) = 0, \qquad \lim_{x \to +\infty} \sigma(x) = 1.$$

ReLU, softplus

해당 안되지만

Summary에 의하면

되긴 된다고 한다.

Heaviside는 될까?

9.7 Summary

The chapter answers the question of what kind of functions can be approximated using multiple layer networks and is based on the approximation theory results developed by Funahashi, Hornik, Stinchcombe, White, and Cybenko in the late 1980s.

The results proved in this chapter show that a one-hidden layer neural network with sigmoid neurons in the hidden layer and linear activation in the output layer can learn continuous functions, integrable functions, and square integrable functions, as well as measurable functions. The quality of neural networks to be potential universal approximators is not the specific choice of the activation function, but rather the feedforward network architecture itself.

The price paid is that the number of neurons in the hidden layer does not have an a priori bound. However, there is a result stating that for each extra digit of accuracy gain in the target approximation, the number of hidden neurons should increase 100-fold.

The approximation results work also if the activation function is a ReLU or a Softplus. However, the proof is not a straightforward modification of the proofs provided in this chapter. The trade-off between depth, width and approximation error is still an active subject of research. For instance, in 2017, Lu et al. [79] proved an universal approximation theorem for width-bounded deep neural networks with ReLU activation functions that can approximate any Lebesgue integrable function on n-dimensional input space. Shortly after, Hanin [52] improved the result using ReLU-networks to approximate any continuous convex function of n-dimensional input variables.

Neural networks can also be used to solve numerically first-order differential equations with initial conditions.

Continuous 말고

다른 조건의 함수도

U.A.T. 적용가능한가보다.

Width-bounded deep neural network

with ReLU activation function

특정 조건 함수 U.A.T. 적용가능!

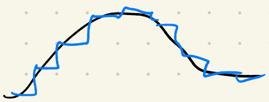
신기한 이론의 세계

CODE HW

Heaviside function

Y아님!

$$\sum c_i H(x - b_i) \qquad H = \begin{cases} 1 & x > b_i \\ \text{지정하기 나름 여기선 } 0 & x = b_i \\ 0 & x < b_i \end{cases}$$



이렇게 만들어보려 했으나...

error : Back propagation 할 때

$$\frac{\partial L}{\partial b_i} \simeq \frac{H(x-b_i)}{\partial b_i}$$ 이 계산이 안돼서 그런것 같다.

Heaviside function 에 derivative 계산 지원을 안하는것같다.

$H = 0$ or 1 로 $\frac{\partial H}{\partial b_i} = 0$ 라 한다 치면 b_i는 업데이트가 안된다.
의미가 없다.

Sigmoid, ReLU 다 잘 되는 것 같다.

Softplus 는 iteration을 더 많이 해줘야 어느정도 근사가 이루어진다.

U.A.T is super cool !!