Video 21

Saturday, October 31, 2020 4:22 PM

Activation function is one of the distinctions between deep neural networks (DNNs) vs. Linear models.

DNN

$$\vec{y} = \phi^{(L)}\left(W^{(L)} \vec{h}^{(L-1)}\right)$$

activation function

omit the bias using the dummy variable trick.

$$\vec{h}^{(L-1)} = \phi^{(L-1)}\left(W^{(L-1)} \vec{h}^{(L-2)}\right)$$

$$\vdots$$

$$\vec{h}^{(1)} = \phi^{(1)}\left(W^{(1)} \vec{x}\right)$$

model's input.

If we assume we don't use an activation function

$$\phi^{(L)} = \phi^{(L-1)} = \dots = \phi^{(1)}$$

$$\phi(x) = x \qquad (\text{"Linear" activation function})$$

$$\vec{y} = \underbrace{W^{(L)} W^{(L-1)} \dots W^{(1)}}_{\text{product of weight matrices is itself a single weight matrix}} \vec{x}$$

$\Rightarrow$ a DNN without activation functions is equivalent to a linear model.

Universal approximation theorem:

A single hidden layer neural network with a sufficiently large hidden layer is able to approximate any function arbitrarily well.

with enough neurons.

This universality property was demonstrated for DNNs with various activation functions (threshold, logistic, ReLUs, ...)

Example argument illustrating the universality of DNNs

"Toy" problem space where all inputs are binary

Goal: given a function mapping input vectors to outputs, we need to produce a neural net which matches that function.

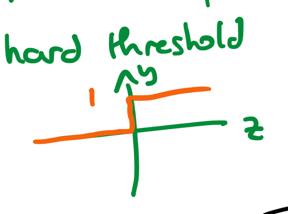
Assume only 3 inputs

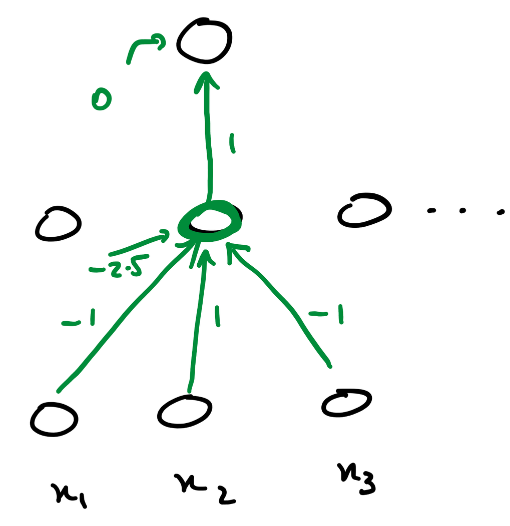
x_1	x_2	x_3	t
	$\vdots$		$\vdots$
-1	-1	1	-1
-1	1	-1	1
-1	1	1	1
	$\vdots$		$\vdots$

inputs: x_1, x_2, x_3; targets/outputs: t

2^D rows; # of inputs: D

Activation function hard threshold





Any input pattern will produce activations at the hidden layer such that exactly one neuron of the hidden layer is active.

While universality is a nice property of DNNs, the DNN required to represent a function may not be compact.

Compact (advantages):

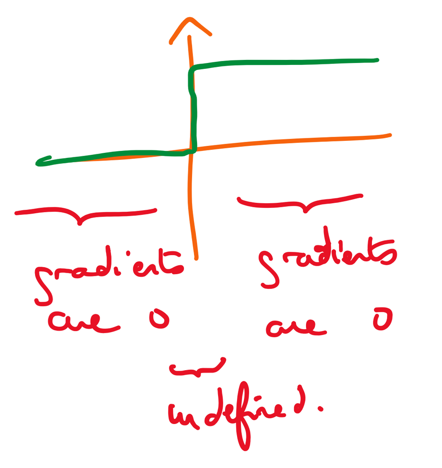
- Easier to compute
- Large (non-compact) models have more parameters (they are more prone to memorization).

Note: in the example above, we used hard threshold activation functions

why? → convenient to design the neural network

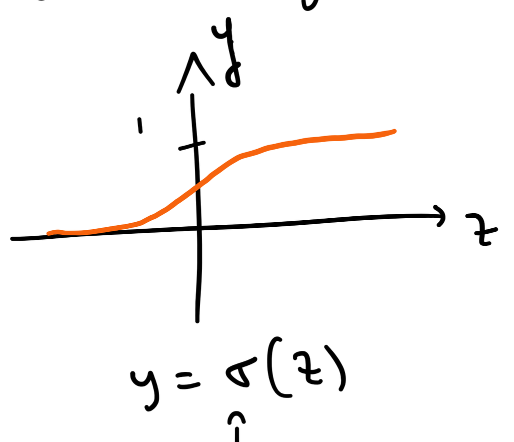
Disadvantage from hard threshold functions:

(We saw previously) hard to train with gradient descent

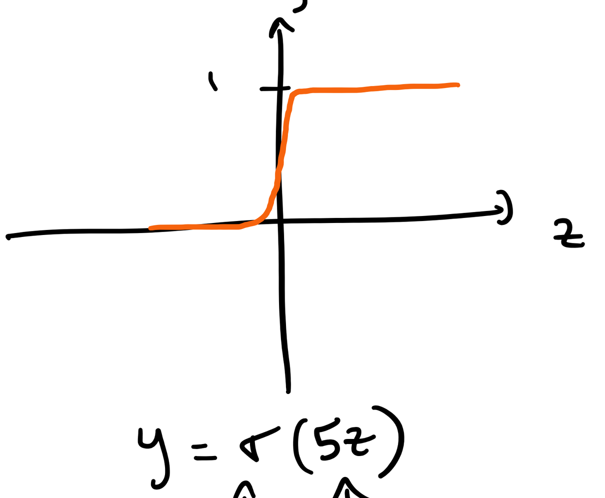


Solution: replace hard threshold functions by other non-linear activation functions.

We could use a logistic activation:



$$y = \sigma(z), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$



$$y = \sigma(5z), \quad \sigma(z) = \frac{1}{1+e^{-z}}$$

scaled weights

If we combine logistic activation functions with gradient descent, we can find values of the weight parameters that yield an activation of the neuron which is close to a hard threshold.

Depth vs width

One of the advantages of deeper neural networks, is that they can represent functions more compactly than shallow neural networks.