DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators

Lu Lu, Pengzhan Jin, George Karniadakis

Division of Applied Mathematics, Brown University

October 18, 2019

BROWN

Introduction

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto 5$

  $\Rightarrow$ Neural network [Universal approximation theorem]

BROWN

Introduction

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

  $\Rightarrow$ Neural network [Universal approximation theorem]

- Operator: function (∞-dim) $\mapsto$ function (∞-dim)

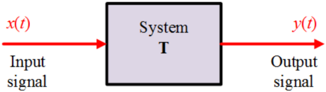
Introduction

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

  $\Rightarrow$ Neural network [Universal approximation theorem]

- Operator: function (∞-dim) $\mapsto$ function (∞-dim)
  e.g., derivative (local): $x(t) \mapsto x'(t)$
  e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$

  e.g., dynamic system: 

BROWN

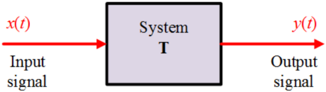
Introduction

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

  $\Rightarrow$ Neural network [Universal approximation theorem]

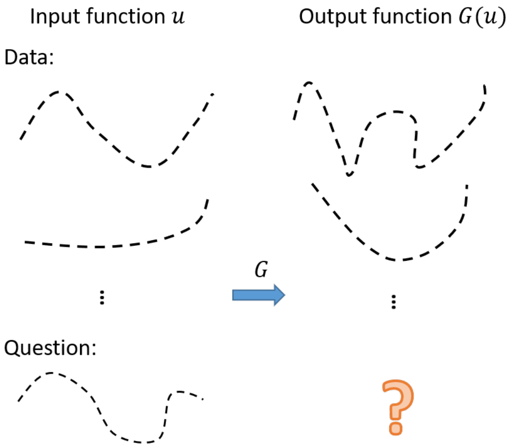
- Operator: function (∞-dim) $\mapsto$ function (∞-dim)
  e.g., derivative (local): $x(t) \mapsto x'(t)$
  e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$

  e.g., dynamic system: 

  $\Rightarrow$ Can we learn operators via neural networks?
  $\Rightarrow$ How?

Problem setup

$$G : u \mapsto G(u)$$
$$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$$



Network inputs and output

$G : u \mapsto G(u)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

- Inputs: u (∞-dim?), $y \in \mathbb{R}^d$
- Output: $G(u)(y) \in \mathbb{R}$

BROWN

Network inputs and output

$G : u \mapsto G(u)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

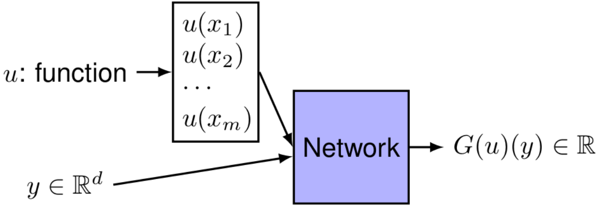
- Inputs: u (∞-dim?), $y \in \mathbb{R}^d$
- Output: $G(u)(y) \in \mathbb{R}$
- u sensors: $x_1, x_2, \ldots, x_m$

BROWN

Network inputs and output

$G : u \mapsto G(u)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

- Inputs: u (∞-dim?), $y \in \mathbb{R}^d$
- Output: $G(u)(y) \in \mathbb{R}$
- u sensors: $x_1, x_2, \ldots, x_m$



Assume $\{x_i\}$ “dense” enough,
is there a universal approximation theorem for operator?

Universal Approximation Theorem for Operator

$G : u \mapsto G(u)$, $G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

Theorem (Chen & Chen, IEEE Trans. Neural Netw., 1995)

Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$, such that

$$\left| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k \cdot y + \zeta_k) \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$.

BROWN

Number of sensors

Consider $G : u(x) \mapsto \boldsymbol{s}(x)$ ($x \in [0, 1]$) by ODE system

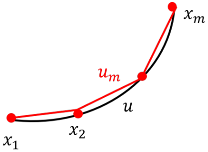
$$\frac{d}{dx}\boldsymbol{s}(x) = \boldsymbol{g}(\boldsymbol{s}(x), u(x), x), \quad \boldsymbol{s}(0) = \boldsymbol{s_0}$$

Number of sensors

Consider $G : u(x) \mapsto \boldsymbol{s}(x)$ ($x \in [0,1]$) by ODE system

$$\frac{d}{dx}\boldsymbol{s}(x) = \boldsymbol{g}(\boldsymbol{s}(x), u(x), x), \quad \boldsymbol{s}(0) = \boldsymbol{s_0}$$

$$u \in V \Rightarrow u_m \in V_m$$

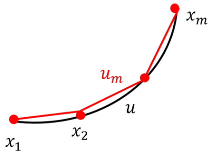


Number of sensors

Consider $G : u(x) \mapsto \boldsymbol{s}(x)$ $(x \in [0,1])$ by ODE system

$$\frac{d}{dx}\boldsymbol{s}(x) = \boldsymbol{g}(\boldsymbol{s}(x), u(x), x), \quad \boldsymbol{s}(0) = \boldsymbol{s_0}$$

$$u \in V \Rightarrow u_m \in V_m$$



Let $\kappa(m, V) := \sup_{u \in V} \max_{x \in [0,1]} |u(x) - u_m(x)|$

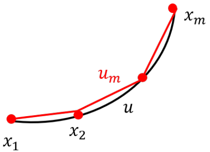
e.g., Gaussian process with kernel $e^{-\frac{\|x_1 - x_2\|^2}{2l^2}}$: $\kappa(m, V) \sim \frac{1}{m^2 l^2}$

Number of sensors

Consider $G : u(x) \mapsto \boldsymbol{s}(x)$ ($x \in [0,1]$) by ODE system

$$\frac{d}{dx}\boldsymbol{s}(x) = \boldsymbol{g}(\boldsymbol{s}(x), u(x), x), \quad \boldsymbol{s}(0) = \boldsymbol{s_0}$$

$$u \in V \Rightarrow u_m \in V_m$$



Let $\kappa(m, V) := \sup_{u \in V} \max_{x \in [0,1]} |u(x) - u_m(x)|$

e.g., Gaussian process with kernel $e^{-\frac{\|x_1 - x_2\|^2}{2l^2}}$: $\kappa(m, V) \sim \frac{1}{m^2 l^2}$

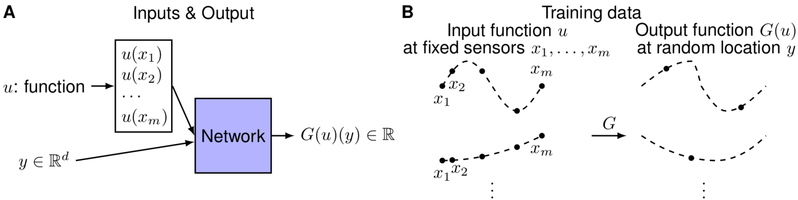
Theorem (informal)

There exists a constant C, such that for any y,

$$\sup_{u \in V} \|G(u)(y) - NN(u(x_1), \ldots, u(x_m), y)\|_2 < C\kappa(m, V).$$

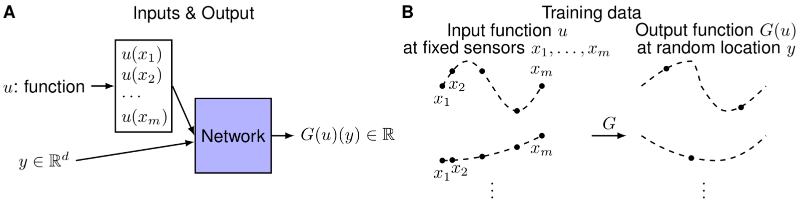
So far, we show

- operators can be approximated by neural networks
- the number of sensors we need



So far, we show

- operators can be approximated by neural networks
- the number of sensors we need



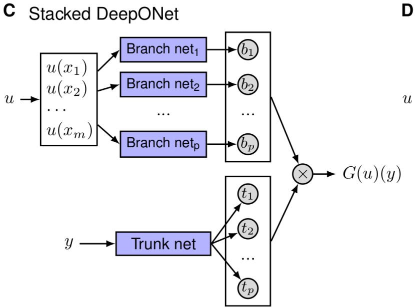
Q: How to design the network?

$\Rightarrow$ Deep operator network (DeepONet)

DeepONet

Recall the Theorem:

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk}$$



BROWN

A simple ODE case

$$\frac{ds(x)}{dx} = u(x), \quad x \in [0,1],$$

with an initial condition $s(0) = 0$.

$$G : u(x) \mapsto s(x) = \int_0^x u(\tau) d\tau$$

BROWN

A simple ODE case

$$\frac{ds(x)}{dx} = u(x), \quad x \in [0, 1],$$

with an initial condition $s(0) = 0$.

$$G : u(x) \mapsto s(x) = \int_0^x u(\tau) d\tau$$

100 u sensors, 10000×1 training points

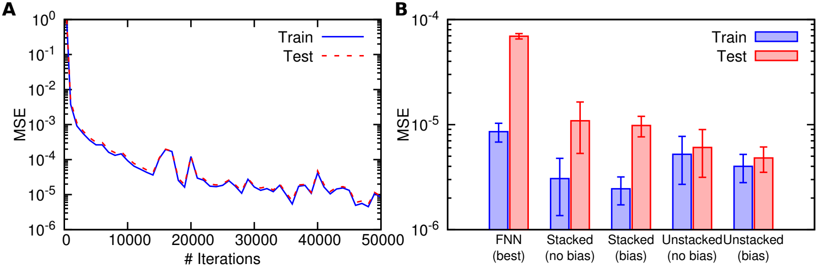
A simple ODE case

$$\frac{ds(x)}{dx} = u(x), \quad x \in [0,1],$$

with an initial condition $s(0) = 0$.

$$G : u(x) \mapsto s(x) = \int_0^x u(\tau) d\tau$$

100 u sensors, 10000×1 training points
Very small generalization error!



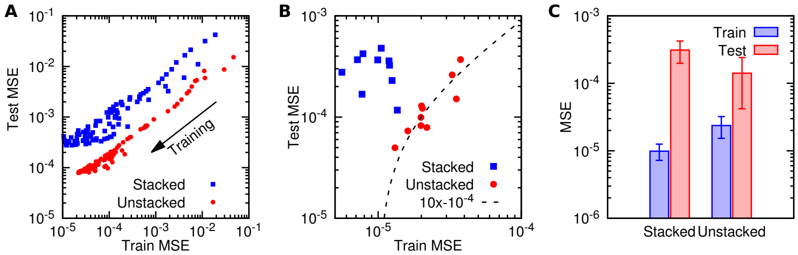
BROWN

A nonlinear ODE case

$$\frac{ds(x)}{dx} = -s^2(x) + u(x)$$

A nonlinear ODE case

$$\frac{ds(x)}{dx} = -s^2(x) + u(x)$$

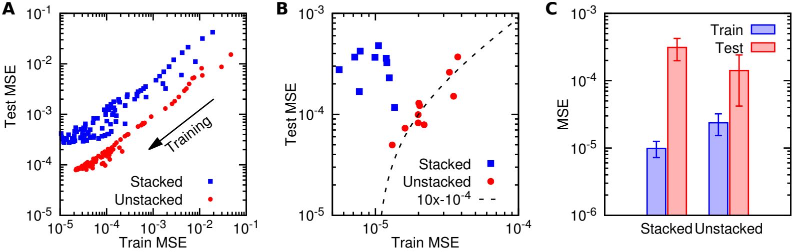


Linear correlation between training and test errors

- A: in one training process

A nonlinear ODE case

$$\frac{ds(x)}{dx} = -s^2(x) + u(x)$$

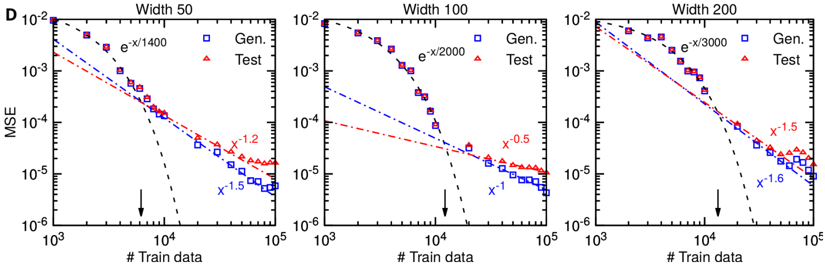


Linear correlation between training and test errors

- A: in one training process
- B: across multiple runs (random dataset and network initialization)

Gravity pendulum with an external force $u(t)$

$$\frac{ds_1}{dt} = s_2, \quad \frac{ds_2}{dt} = -k\sin s_1 + u(t)$$

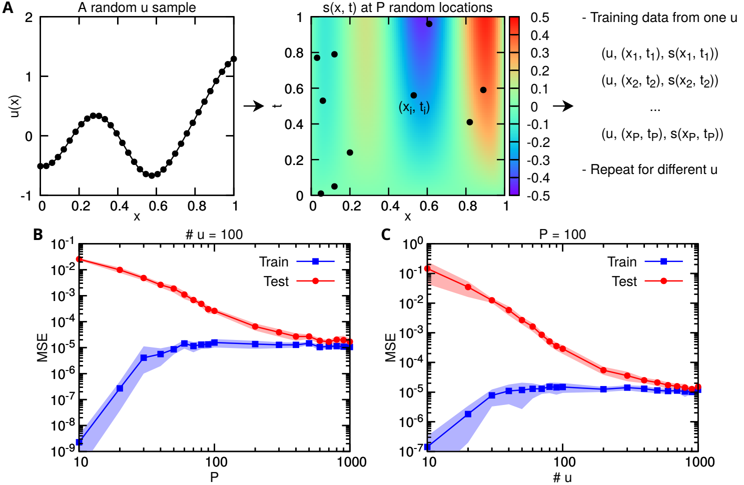


Test/generalization error:

- small dataset: exponential convergence
- large dataset: polynomial rates
- smaller network has earlier transition point

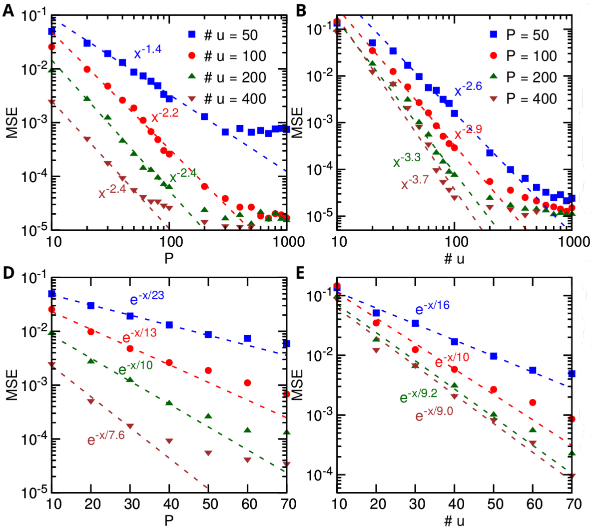
Diffusion-reaction system

$$\frac{\partial s}{\partial t} = D\frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0,1], t \in [0,1]$$



Diffusion-reaction system

exponential/polynomial convergence



Summary

- Number of sensors, $\kappa(m, V)$
- DeepONet
  - 1D ODE (linear, nonlinear), gravity pendulum, diffusion-reaction system (nonlinear)
  - Small generalization error
  - Exponential/polynomial error convergence

- Lu, Jin, & Karniadakis, arXiv:1910.03193, 2019.
- DeepXDE: `https://deepxde.rtfd.io`

BROWN