Machine Learning and Dynamical Systems

Approximation, Optimization and Beyond

Qianxiao Li

Email: qianxiao@nus.edu.sg

Website: https://blog.nus.edu.sg/qianxiaoli

April, 2021

1 Background and Motivation

2 Machine Learning via Dynamical Systems

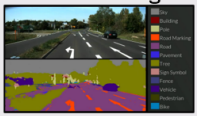
3 Machine Learning for Dynamical Systems

4 Machine Learning of Dynamical Systems

Background and Motivation

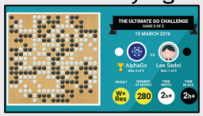
Deep Learning: Theory vs Practice

Self-Driving



Machine Translation

Game-Playing

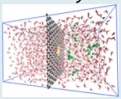


Practical Success

Emerging Applications

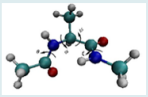
Science and Engineering

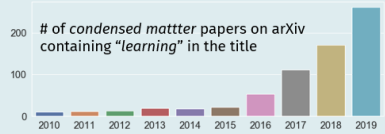
Molecular Dynamics



Phase Transitions

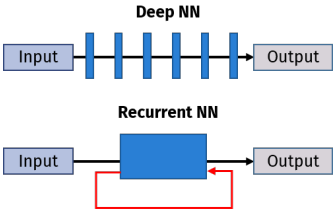
Conformational Changes





Theoretical Mystery

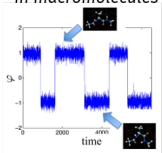
Models



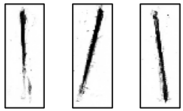
Data

Data in Scientific Applications

Conformational changes in macromolecules

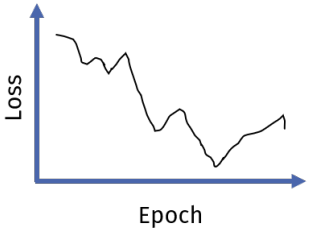


Damage evolution In self-healing materials



Algorithms

Stochastic Gradient Algorithm



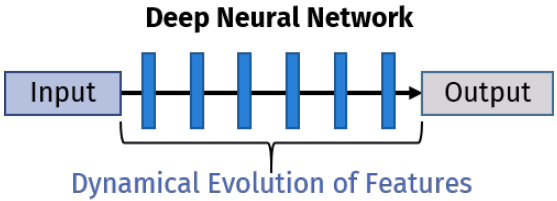
Composition is Dynamics

Compositional function

$$y = F_T \circ F_{T-1} \circ \cdots \circ F_0(x)$$

Dynamics

$$y = x_{T+1}, \qquad x = x_0$$
$$x_{t+1} = F_t(x_t) \qquad t = 0, 1, \ldots, T$$



Three Categories of Interactions

Machine Learning via Dynamical Systems

- Dynamical formulation of deep learning
- Dynamical analysis of learning algorithms
- Numerical analysis and architecture design

Machine Learning for Dynamical Systems

- Time series forecasting
- Sequence classification
- Sequence to Sequence models

Machine Learning of Dynamical Systems

- Learning dynamical models from trajectories
- Learning reduced order models

Machine Learning via Dynamical Systems

Machine Learning via Dynamical Systems

Background on Supervised Learning

Supervised learning is about making predictions

Dataset: $\mathcal{D} = \{x^i \in \mathcal{X}, y^i \in \mathcal{Y}\}_{i=1}^N$, Target Function: $F^* : \mathcal{X} \to \mathcal{Y}$, $y^i = F^*(x^i)$

$x^1 =$ 

$x^2 =$ 

F^*

$y^1 = \text{Cat} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$y^2 = \text{Dog} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Goal: Learn/Approximate F^* using information from $\mathcal{D}$

Hypothesis Space

To approximate F^*, we first define a hypothesis space $\mathcal{H}$ consisting of candidate functions

- Linear models for regression and classification

$$\mathcal{H} = \left\{ \sum_j a_j \varphi_j(x) : a_j \in \mathbb{R} \right\}$$

Hypothesis Space

To approximate F^*, we first define a hypothesis space $\mathcal{H}$ consisting of candidate functions

- Linear models for regression and classification

$$\mathcal{H} = \left\{ \sum_j a_j \varphi_j(x) : a_j \in \mathbb{R} \right\}$$

- Shallow neural networks

$$\mathcal{H} = \left\{ \sum_j a_j \sigma(w_j^T x + b_j) : w_j \in \mathbb{R}^d, a_j, b_j \in \mathbb{R} \right\}$$

σ is the activation function, e.g. $\sigma(z) = \max(0, z)$ (ReLU); $\sigma(z) = \tanh(z)$ (Tanh)

Hypothesis Space

To approximate F^*, we first define a hypothesis space $\mathcal{H}$ consisting of candidate functions

- Linear models for regression and classification

$$\mathcal{H} = \left\{ \sum_j a_j \varphi_j(x) : a_j \in \mathbb{R} \right\}$$

- Shallow neural networks

$$\mathcal{H} = \left\{ \sum_j a_j \sigma(w_j^T x + b_j) : w_j \in \mathbb{R}^d, a_j, b_j \in \mathbb{R} \right\}$$

  σ is the activation function, e.g. $\sigma(z) = \max(0, z)$ (ReLU); $\sigma(z) = \tanh(z)$ (Tanh)

- Deep neural networks

$$\mathcal{H} = \{ F_T \circ F_{T-1} \circ \cdots \circ F_0(x) : \text{ Each } F_t \text{ is a shallow network} \}$$

Empirical and Population Risk Minimization

Learning/approximation involves picking out the "best" $F \in \mathcal{H}$ so that it makes similar predictions to F^*

- Empirical risk minimization (ERM)

$$\hat{F} \longleftarrow \arg\min_{F \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \Phi(F(x^i), \underbrace{y^i}_{F^*(x^i)}) \qquad (\Phi \text{ is a loss function})$$

Empirical and Population Risk Minimization

Learning/approximation involves picking out the "best" $F \in \mathcal{H}$ so that it makes similar predictions to F^*

- Empirical risk minimization (ERM)

$$\hat{F} \longleftarrow \arg\min_{F \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \Phi(F(x^i), \underbrace{y^i}_{F^*(x^i)}) \qquad (\Phi \text{ is a loss function})$$

- Population risk minimization (PRM)

$$\tilde{F} \longleftarrow \arg\min_{F \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mu^*} \Phi(F(x), y) \qquad (\mu^* \text{ is the input-output distribution})$$

Empirical and Population Risk Minimization

Learning/approximation involves picking out the "best" $F \in \mathcal{H}$ so that it makes similar predictions to F^*

- Empirical risk minimization (ERM)

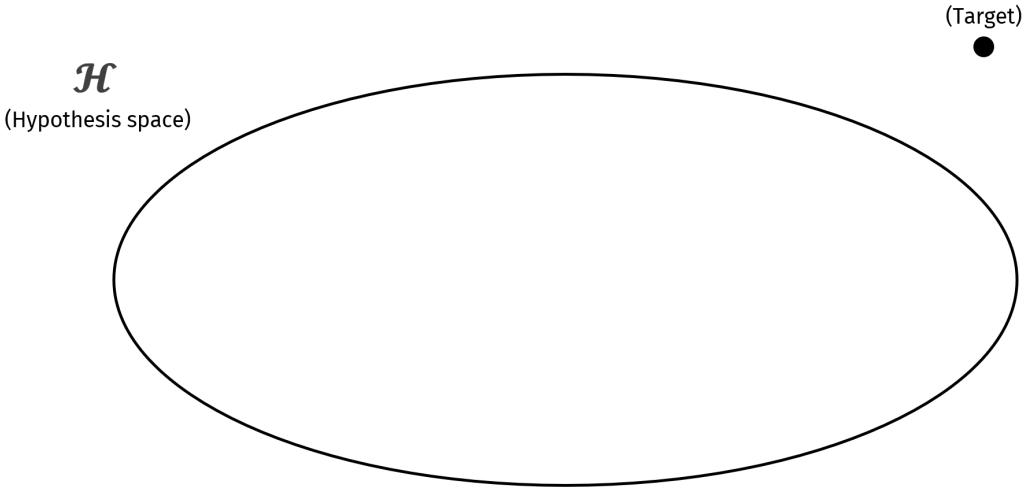
$$\hat{F} \longleftarrow \arg\min_{F \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \Phi(F(x^i), \underbrace{y^i}_{F^*(x^i)}) \qquad (\Phi \text{ is a loss function})$$

- Population risk minimization (PRM)

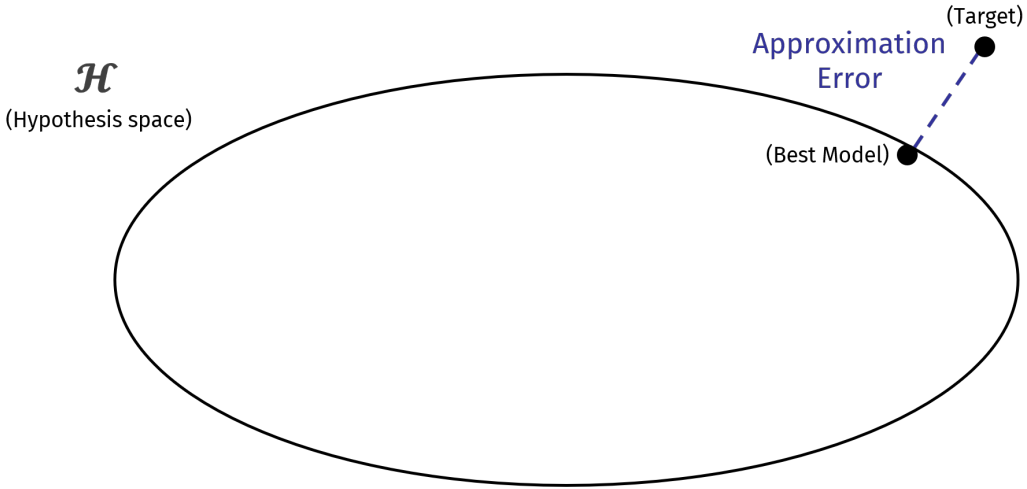
$$\tilde{F} \longleftarrow \arg\min_{F \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mu^*} \Phi(F(x), y) \qquad (\mu^* \text{ is the input-output distribution})$$

- We want to solve PRM, but we often can only perform ERM. The gap between $\hat{F}$ and $\tilde{F}$ is the problem of generalization.

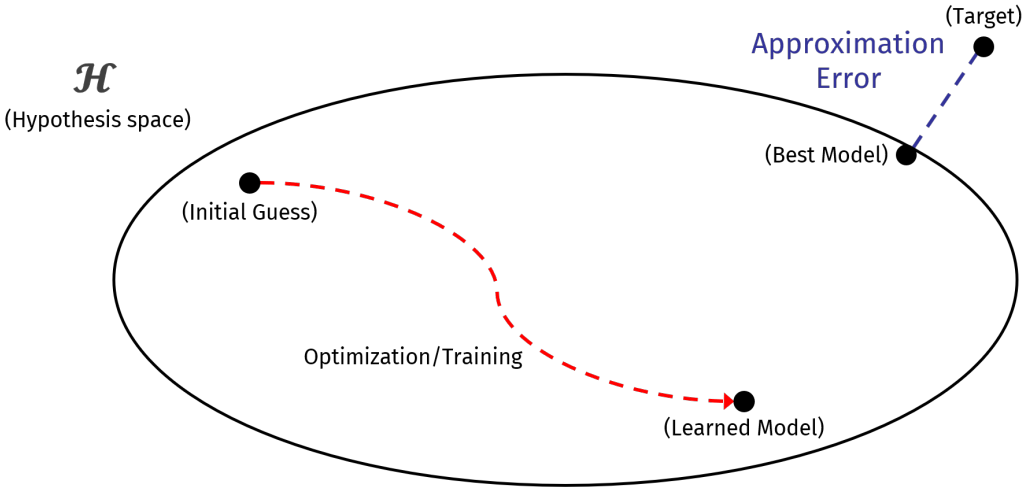
Three Paradigms of Supervised Learning



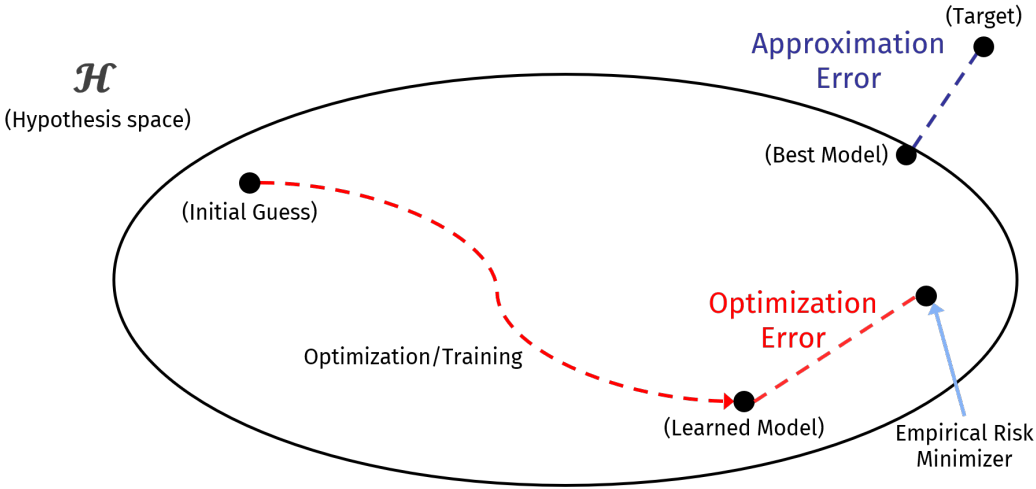
Three Paradigms of Supervised Learning



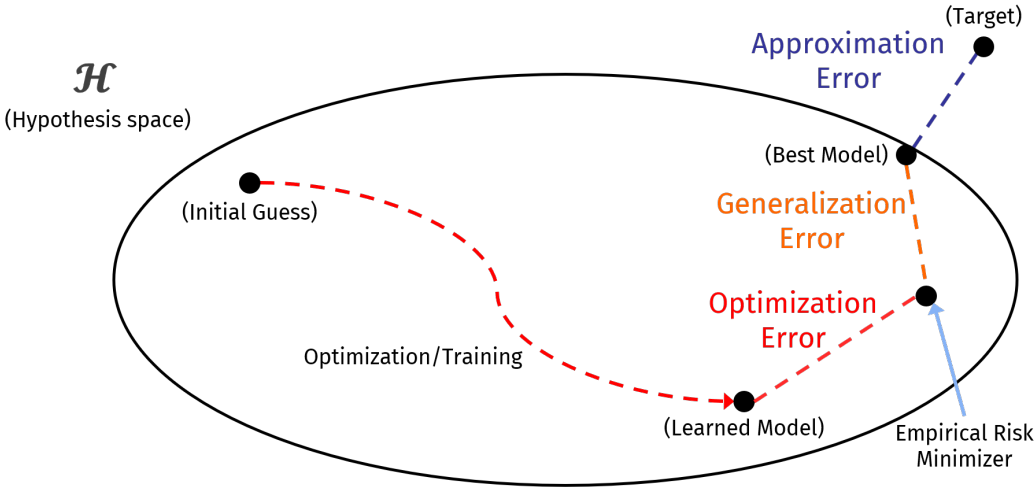
Three Paradigms of Supervised Learning



Three Paradigms of Supervised Learning



Three Paradigms of Supervised Learning



Goal: study new phenomena that arise in deep learning with respect to approximation, optimization and generalization

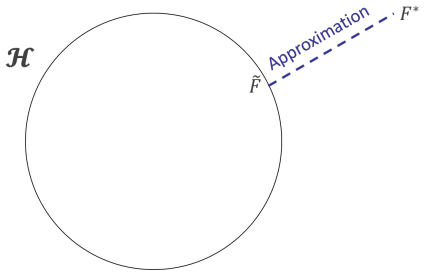
Machine Learning via Dynamical Systems

Approximation Theory for Continuous-time ResNets

Universal Approximation Property

Given a target $F^* \in \mathcal{C}$ and $\epsilon > 0$, does there exists a $\widetilde{F} \in \mathcal{H}$ such that

$$\|F^* - \widetilde{F}\| \leq \epsilon?$$

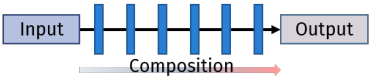


Key question: how to achieve approximation via composition?

The Continuum Idealization of Residual Networks

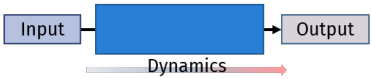
Deep Neural Network

$$x_{k+1} = x_k + f_k(x_k, \theta_k)$$



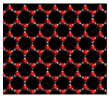
Continuous-time Dynamical System

$$\dot{x}(t) = f\big(t, x(t), \theta(t)\big)$$



Continuum Idealization

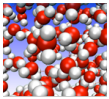
Solids



Linear Elasticity Equations

$$\nabla \cdot \sigma + F = \rho \ddot{u}$$

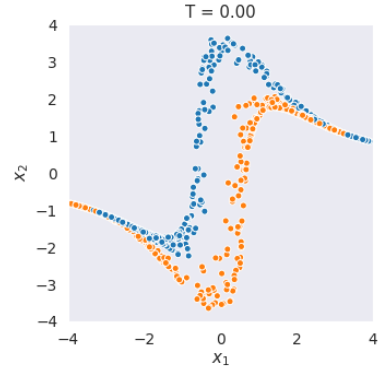
Fluids



Navier Stokes Equations

$$\dot{u} + (u \cdot \nabla)u = -\rho^{-1}\nabla P + \nu \nabla^2 u$$

Example: Flow Maps as Function Approximators



Binary Classification Problem

Not linearly separable!

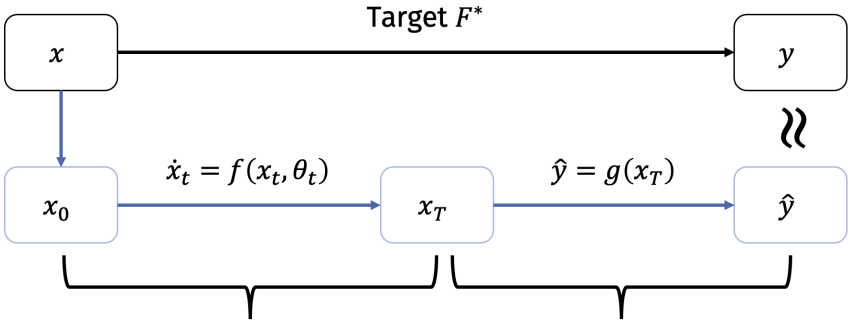
Evolve with the ODE

$$\dot{x}_{t,1} = -x_{t,2}\sin(t) \qquad x_{0,1} = z_1$$
$$\dot{x}_{t,2} = -\frac{1}{2}(1 - x_{t,1}^2)x_{t,2} + x_{t,1}\cos(t) \qquad x_{0,2} = z_2$$

Classify using linear classifier at the end:

$$g(x_T) = \mathbf{1}_{x_{T,1}>0}$$

Approximating Functions by Dynamics



Compositional Hypothesis Space

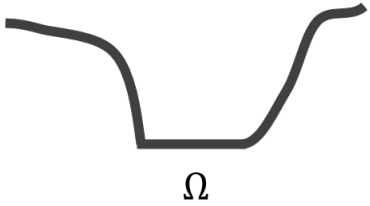
$\mathcal{H}(\mathcal{F}, \mathcal{G}) = \cup_{\{T \geq 0\}} \{g \circ \varphi : g \in \mathcal{G}, \varphi \in \Phi(\mathcal{F}, T)\}$

What conditions on $\mathcal{F}, \mathcal{G}$ are sufficient to induce the universal approximation property on $\mathcal{H}(\mathcal{F}, \mathcal{G})$?

Universal Approximation by Flows

- Sufficient conditions for universal approximation by flows [L, Lin & Shen, 19]
- In dimension ≥ 2, always possible under mild conditions

  1. $\mathcal{G}$ covers range of F^*
  2. $\mathcal{F}$ is restricted affine invariant
  3. $\overline{\mathrm{Conv}}(\mathcal{F})$ contains a well function



- In dimension 1, only increasing functions if $\mathcal{G} = \{\mathrm{id}\}$
- Approximation rates can be characterized for $d = 1$. For $d \geq 2$, a general characterization is open.
- Connections with controllability [CLT, 19; TG, 20]

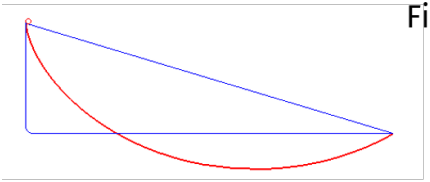
Q. Li, T. Lin, and Z. Shen, "Deep Learning via Dynamical Systems: An Approximation Perspective," en, *To appear in J. Eur. Math. Soc. (JEMS), 2021*, 2019. arXiv: 1912.10382

C. Cuchiero, M. Larsson, and J. Teichmann, "Deep neural networks, generic universal interpolation, and controlled ODEs," en, *arXiv:1908.07838 [math]*, 2019. arXiv: 1908.07838

P. Tabuada and B. Gharesifard, "Universal Approximation Power of Deep Residual Neural Networks via Nonlinear Control Theory," *arXiv:2007.06007*, 2020. arXiv: 2007.06007

Machine Learning via Dynamical Systems

Deep Learning and Optimal Control

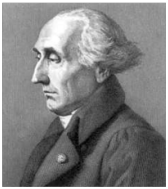


Find curve connecting A→B so that ball rolls down in shortest time
- Johann Bernoulli (1696)

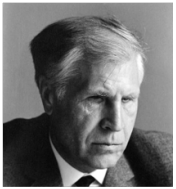
Answer: Cycloids

This led to the development of **calculus of variations** and **optimal control**


Euler


Lagrange


Weierstrass


Pontryagin


Bellman

The PRM problem using $\mathcal{H}(\mathcal{F}, \mathcal{G})$ can be formulated as

$$\inf_{\theta \in L^\infty([0,T],\Theta)} J(\theta) := \mathbb{E}_{\mu^*}\left[\Phi(x_T, y) + \int_0^T \underbrace{R(x_t, \theta_t)}_{\text{Regularizer}}\, dt\right]$$

$$\dot{x}_t = f(x_t, \theta_t) \qquad 0 \le t \le T \qquad (x_0, y) \sim \mu^*$$

This is a mean-field optimal control problem, because we need to select θ that controls not one, but an entire distribution of inputs and outputs

Key questions:

- Theoretical: Necessary and sufficient conditions for optimality
- Practical: Understanding, improving learning algorithms

Mathematical Results and Numerical Algorithms

- Necessary conditions
  - Mean-field Pontryagin's maximum principle (PMP) [E, Han & L, 19]
- Necesssary and Sufficient Conditions
  - Mean-field Hamilton Jacobi Bellman equations (HJB) [E, Han & L, 19]
- Algorithms
  - Training algorithms without gradient descent (based on PMP, generalizes back propagation) [L, Chen, Tai & E, 18]
  - Training algorithms for quantized networks [L and Hao, 18]

W. E, J. Han, and Q. Li, "A mean-field optimal control formulation of deep learning," *Research in the Mathematical Sciences*, vol. 6, no. 1, p. 10, 2019, ISSN: 2522-0144

Q. Li, L. Chen, C. Tai, and W. E, "Maximum principle based algorithms for deep learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5998–6026, 2017

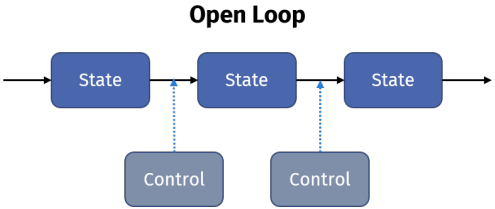
Q. Li and S. Hao, "An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 2985–2994

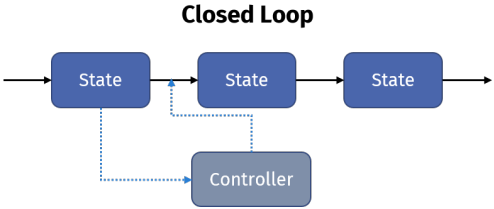
Open Loop and Closed Loop Control

Controlled dynamics:

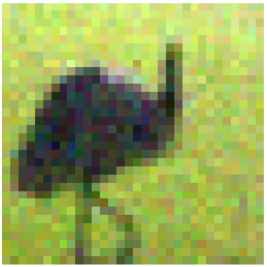
$$\dot{x}_t = f(x_t, \theta_t).$$

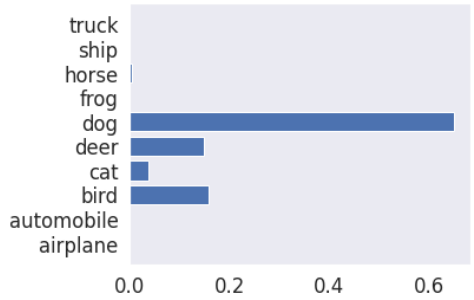
- Open-loop optimal control: θ_t^* a deterministic function of t
  - Works for fixed x_0
  - Unstable to perturbations on x_0
- Close-loop optimal control: $\theta_t^* = \xi(x_t^*, t)$
  - Feed-back based on current state
  - Stable to perturbations on x_0 (or any x_t)





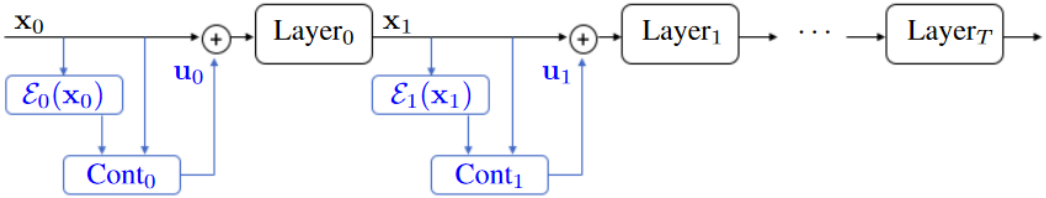
Adversarial Examples





Adversarial Defense based on Closed Loop Optimal Control

We can use closed loop control to stabilize propagation of features through a deep network



This leads to increased adversarial robustness without the need to retrain the model [Chen, L and Zhang, 21]

Z. Chen, Q. Li, and Z. Zhang, "Towards robust neural networks via close-loop control," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=2AL06y9cDE-

Machine Learning via Dynamical Systems

Dynamical Analysis of Learning Algorithms

Empirical risk

$$J(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\sim\mathcal{D}} \Phi(F(x;\theta), y)$$

Simplest learning (optimization) algorithm is the stochastic gradient desecent

$$\theta_{k+1} = \theta_k - \underbrace{\eta}_{\text{learning rate}} \nabla_\theta J(\theta; \mathcal{D}_k)$$

where $\mathcal{D}_k$ is a random subset of $\mathcal{D}$

Convergence Bounds vs Modelling Dynamics

Convergence bounds:

$$\mathbb{E}\|\theta_k - \theta^*\| \leq r(k) \qquad (r(k) \to 0)$$

Dynamical models:

$$\max_k \mathbb{E}\|\theta_k - s_{k\eta}\| \leq r(\eta) \qquad (r(\eta) \to 0)$$

$\{s_t\}$ is a continuous-time dynamical system

Such techniques dates back to the analysis of finite-difference algorithms using modified equations

R. Warming and B. Hyett, "The modified equation approach to the stability and accuracy analysis of finite-difference methods," *Journal of computational physics*, vol. 14, no. 2, pp. 159–179, 1974

Stochastic Modified Equations

SGD:

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta J(\theta; \mathcal{D}_k)$$

Stochastic modified equations

$$ds_t = -\nabla_\theta J(s_t; \mathcal{D})dt + \sqrt{\eta}\Sigma(s_t)^{\frac{1}{2}} \underbrace{dW_t}_{\text{Wiener process}}$$

where

$$\Sigma(s) = \mathrm{Cov}_{\mathcal{D}_k}(\nabla J(s; \mathcal{D}_k)) \qquad \text{(Gradient covariance)}$$

- Weak approximation of $\{\theta_k\}$ by $\{s_{k\eta}\}$ [L, Tai & E, 17,19]
- Extensions
  - Higher order
  - SGD variants (adaptive SGD, momentum SGD)

Q. Li, C. Tai, and W. E, "Stochastic modified equations and adaptive stochastic gradient algorithms," in *International Conference on Machine Learning*, 2017

Q. Li, C. Tai, and W. E, "Stochastic Modified Equations and Dynamics of Stochastic Gradient Algorithms I: Mathematical Foundations," *Journal of Machine Learning Research*, vol. 20, no. 40, pp. 1–47, 2019

Applications of Dynamical Modelling of Optimization

Many Applications

- Tuning learning rates can be formulated as a (stochastic) optimal control problem [L, Tai & E, 17]
- Quantitative analysis of batch normalization [Cai, L & Shen, 19]
- Analysis and improvement of adversarial training [Ye, L, Zhou & Zhu, 21]

Q. Li, C. Tai, and W. E, "Stochastic modified equations and adaptive stochastic gradient algorithms," in *International Conference on Machine Learning*, 2017

Y. Cai, Q. Li, and Z. Shen, "A quantitative analysis of the effect of batch normalization on gradient descent," in *International Conference on Machine Learning*, PMLR, 2019, pp. 882–890

N. Ye, Q. Li, X.-Y. Zhou, and Z. Zhu, "Amata: An annealing mechanism for adversarial training acceleration," *AAAI 2021, [arXiv preprint arXiv:2012.08112]*, 2021

Machine Learning for Dynamical Systems

Machine Learning for Dynamical Systems

Supervised Learning of Dynamical Relationships

Learning Dynamic Relationships

Often, supervised learning has to be performed on the dynamic setting

- Price forecasting: given past prices, predict tomorrow's price
- Machine translation: given context, translate a sentence
- Learning response systems: given temporal forcing, predict temporal response

Learning Dynamic Relationships

Often, supervised learning has to be performed on the dynamic setting

- Price forecasting: given past prices, predict tomorrow's price
- Machine translation: given context, translate a sentence
- Learning response systems: given temporal forcing, predict temporal response

Common features

- Each output depends on sequence of inputs
- Need to model a sequence of input-output relationships

Learning Dynamic Relationships

Often, supervised learning has to be performed on the dynamic setting

- Price forecasting: given past prices, predict tomorrow's price
- Machine translation: given context, translate a sentence
- Learning response systems: given temporal forcing, predict temporal response

Common features

- Each output depends on sequence of inputs
- Need to model a sequence of input-output relationships

Deep Learning on temporal/sequential data

↓

Interactions between dynamical structures in model and data

Static setting

(input)	$x \in \mathcal{X} = \mathbb{R}^d$
(output)	$y \in \mathcal{Y} = \mathbb{R}^n$
(target)	$y = F^*(x)$

Modelling Static vs Dynamic Relationships

Static setting

(input)	$x \in \mathcal{X} = \mathbb{R}^d$
(output)	$y \in \mathcal{Y} = \mathbb{R}^n$
(target)	$y = F^*(x)$

Dynamic setting

(input)	$\boldsymbol{x} = \{x_k \in \mathbb{R}^d\} \in \mathcal{X}$
(output)	$\boldsymbol{y} = \{y_k \in \mathbb{R}^n\} \in \mathcal{Y}$
(target)	$y_k = H_k^*(\boldsymbol{x}) \quad \forall \quad k$

Modelling Static vs Dynamic Relationships

Static setting

(input)	$x \in \mathcal{X} = \mathbb{R}^d$
(output)	$y \in \mathcal{Y} = \mathbb{R}^n$
(target)	$y = F^*(x)$

Goal of supervised learning

- Static: learn/approximate F^*
- Dynamic: learn/approximate $\{H_k^*\}$

Dynamic setting

(input)	$\boldsymbol{x} = \{x_k \in \mathbb{R}^d\} \in \mathcal{X}$
(output)	$\boldsymbol{y} = \{y_k \in \mathbb{R}^n\} \in \mathcal{Y}$
(target)	$y_k = H_k^*(\boldsymbol{x}) \quad \forall \quad k$

Some Examples of Learning Temporal Relationships

- Time series classification / regression

$$y_k = H_k^*(\boldsymbol{x}) = \sum_{j \leq k} \alpha_k x_k \qquad \text{(Adding Problem)}$$

- Time series forecasting

$$y_k = x_{k+1} = H_k^*(\{x_j : j \leq k\})$$

- Sequence to sequence models

$$y_k = H_k^*(\{x_j : j \in \mathbb{Z}\})$$

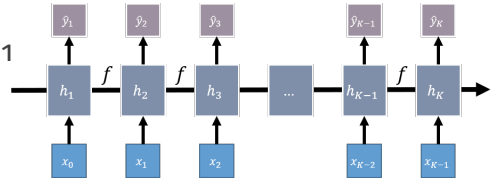
Machine Learning for Dynamical Systems

Approximation Theory of RNNs

The Recurrent Neural Network Hypothesis Space

The recurrent neural network (RNN) architecture

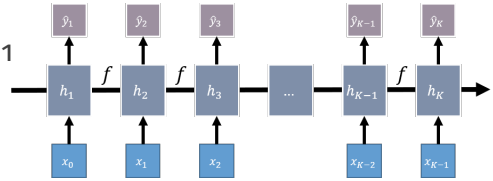
$$h_{k+1} = \sigma(Wh_k + Ux_k), \qquad k = 0, 1, \ldots, K-1$$
$$h_0 = 0$$
$$\hat{y}_k = c^\top h_K$$



The Recurrent Neural Network Hypothesis Space

The recurrent neural network (RNN) architecture

$$h_{k+1} = \sigma(Wh_k + Ux_k), \qquad k = 0, 1, \ldots, K-1$$
$$h_0 = 0$$
$$\hat{y}_k = c^\top h_K$$



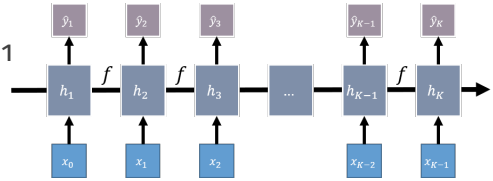
The RNN parametrizes a sequence of functions for $k = 1, 2, \ldots$

$$\hat{H}_k = \{x_0, \ldots, x_{k-1}\} \mapsto \hat{y}_k$$

The Recurrent Neural Network Hypothesis Space

The recurrent neural network (RNN) architecture

$$h_{k+1} = \sigma(Wh_k + Ux_k), \qquad k = 0, 1, \ldots, K-1$$
$$h_0 = 0$$
$$\hat{y}_k = c^\top h_K$$



The RNN parametrizes a sequence of functions for $k = 1, 2, \ldots$

$$\hat{H}_k = \{x_0, \ldots, x_{k-1}\} \mapsto \hat{y}_k$$

Training RNN: adjust (c, W, U) so that

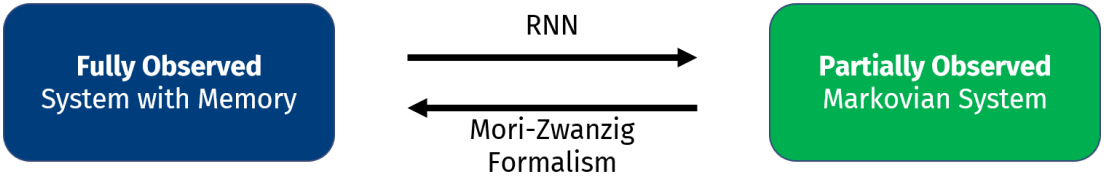
$$\hat{H}_k \approx H_k^*, \qquad k = 1, 2, \ldots, K$$

The Recurrent Neural Network Hypothesis Space

- From the outset, the RNN approach parametrizes a sequence of high-dimensional functions together

The Recurrent Neural Network Hypothesis Space

- From the outset, the RNN approach parametrizes a sequence of high-dimensional functions together
- This is akin to a reverse of the Mori-Zwanzig formalism



C. Ma, J. Wang, and W. E, "Model reduction with memory and the machine learning of dynamical systems," *Communications in Computational Physics*, vol. 25, no. 4, pp. 947–962, 2018, ISSN: 1991-7120. DOI: https://doi.org/10.4208/cicp.OA-2018-0269

R. Zwanzig, *Nonequilibrium statistical mechanics*. Oxford University Press, 2001

Empirically, it is found that memory affects RNN performance. Can we rigorously investigate this phenomena?

We take a continuum approach:

- Continuous-time approximation:

$$h_{k+1} = \underbrace{h_k + \delta}_{\text{residual variant}} \sigma(Wh_k + Ux_k) \qquad \to \qquad \dot{h}_t = \sigma(Wh_t + Ux_t)$$

- Linear activations: $\sigma(z) = z$

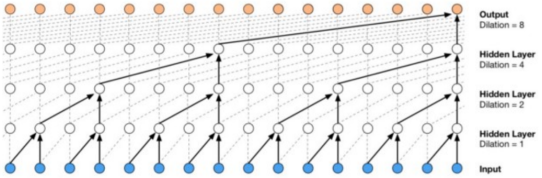
This allows us to prove precise approximation results [Li, Han, E & L, 21]

- Efficient approximation if and only if there is exponential memory decay in $\{H_k^*\}$
- If there is no exponential decay, then one may require exponentially large number of parameters for approximation - curse of memory

Z. Li, J. Han, W. E, and Q. Li, "On the curse of memory in recurrent neural networks: Approximation and optimization analysis," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=8Sqhl-nF50

Extension to Convolution-based Methods

We can extend the previous approach to analyze CNN based methods (e.g. WaveNet) for time-series



Key question: how is RNN and CNN different?

- RNN approximation depends on memory decay [Li, Han, E & L, 21]
- CNN approximation depends on spectral regularity (sparsity) [Jiang, Li & L, 21]

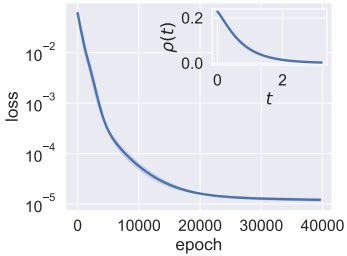
Z. Li, J. Han, W. E, and Q. Li, "On the curse of memory in recurrent neural networks: Approximation and optimization analysis," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=8Sqhl-nF50

H. Jiang, Z. Li, and Q. Li, "Approximation theory of convolutional architectures for time series modelling," *Submitted*, 2021

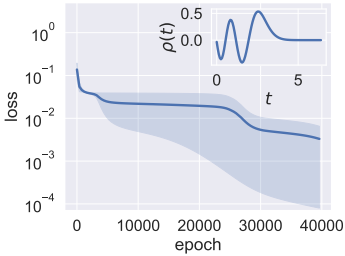
Machine Learning for Dynamical Systems

Optimization Dynamics of RNNs

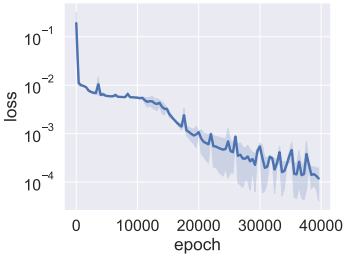
RNN is notoriously hard to train for long sequences



(a) Exponential sum target



(b) Airy function target



(c) Lorenz dynamics target

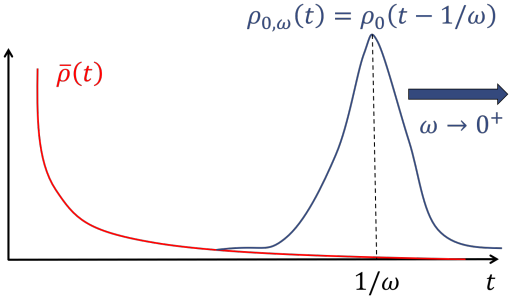
When and why does the training plateau?

What is the source of the plateaus?

Building Memory Explicitly into the Target Functional

Let us consider a target functional that builds explicitly the effect of memory

$$H_T^*(\boldsymbol{x}) = \int_0^\infty x_{T-t}\rho(t)dt \quad \text{(Riesz representation)} \qquad \rho(t) = \bar{\rho}(t) + \rho_{0,\omega}(t)$$

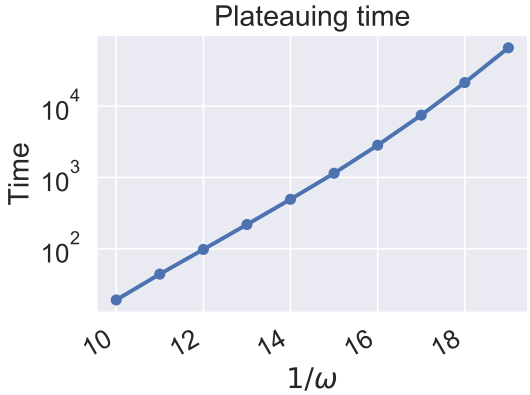


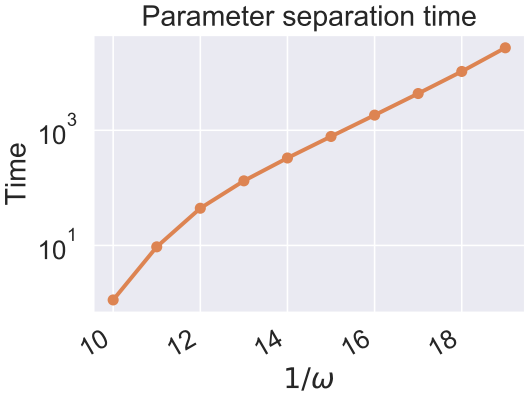
We can prove that as memory increases, the training stalls exponentially, with escape time estimates [Li, Han, E & L, 21]

Z. Li, J. Han, W. E, and Q. Li, "On the curse of memory in recurrent neural networks: Approximation and optimization analysis," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=8Sqhl-nF50

Plateauing Time Scale

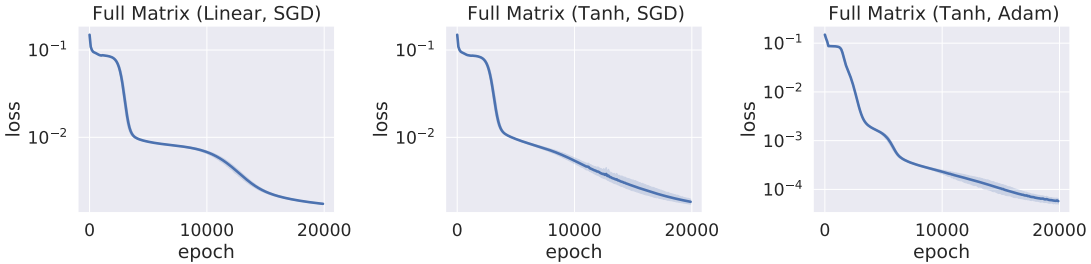
Predicted: log[Plateauing time] $\propto c/\omega$ for ω small





Plateauing for General Cases

Although our analysis is on simplified setting, the plateauing occurs generally

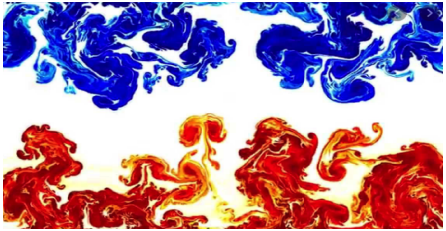


Machine Learning of Dynamical Systems

Machine Learning of Dynamical Systems

Learning Stable and Interpretable Dynamics

Motivation



Rayleigh-Bénard convection
(High-dimensional)

Lorenz ‘63 model
(Low-dimensional)

Data-driven method to characterize high-dimensional dynamics?

- Learning reduced coordinates
- Learning dynamics on these coordinates

The Basic Setup

Consider a high-dimensional dynamical system

$$\dot{x} = F(x), \qquad x(0) \in \mathbb{R}$$

from which we get observation data

$$\mathcal{D} = \{x(t)^i : t \in [0, T], i = 1, \ldots, N\}$$

Goals:

- Learn a function $\varphi : \mathbb{R}^d \to \mathbb{R}^m$
- Learn a function $f : \mathbb{R}^m \to \mathbb{R}^m$ such that the reduced dynamics

$$\dot{h} = f(h), \qquad h(0) = \varphi(x(0))$$

satisfy $h(t) \approx \varphi(x(t))$ for all $t \in [0, T]$

Structure

The reduction function φ and the vector field f should satisfy some structural constraints

Reduction function

- m should be much smaller than d
- φ should be approximately invertible

Vector field

- The dynamics $\dot{h} = f(h)$ should be stable
- We should be able to interpret the physical structure of the learned f

Learning dynamics

Two classes of approaches to learn dynamics from data

- Unstructured approaches: sparse identification, direct NN fitting, ...
  - Accurate over short times
  - Flexible
  - Unstable
- Structured approaches: Hamiltonian, gradient, ...
  - Stable
  - interpretable
  - Limited approximation power

Can we get the best of both worlds?

Onsager principle

A general dynamical model for near-equilibrium process

$$M\dot{h} = -\nabla V(h)$$

- Second law (dissipative system)

$$u \cdot Mu \geq 0 \qquad \forall u \in \mathbb{R}^m \implies \frac{d}{dt}V \leq 0$$

- Onsager's reciprocal relations (Onsager, 1931)

$$M_{ij} = M_{ji}$$

But, limited to linearized regime.

L. Onsager, "Reciprocal relations in irreversible processes. i.," *Physical review*, vol. 37, no. 4, p. 405, 1931
L. Onsager, "Reciprocal relations in irreversible processes. ii.," *Physical review*, vol. 38, no. 12, p. 2265, 1931

We generalise the Onsager principle

$$[M(h) + W(h)]\,\dot{h} = -\nabla V(h) + g(h)$$

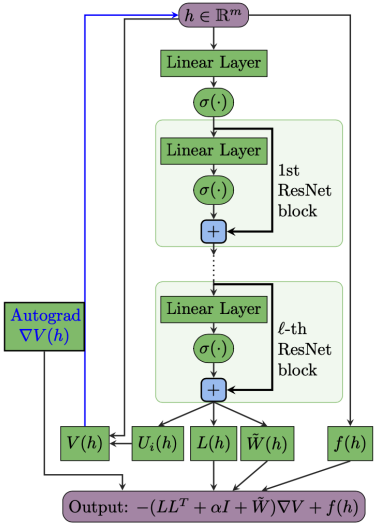
- M is symmetric positive semi-definite (dissipative, reciprocal relations)
- W is anti-symmetric (conservative)
- V is lower-bounded, sufficient growth (potential, free energy, -entropy)
- g is a simple function (external force, control)

There are two main challenges

1. How to find a good set of generalized coordinates $h = \varphi(x)$
2. How to determine M, W, V, g

Data-driven approach [Yu, Tian, E & L, 20]

1. Use *PCA / near-isometric auto-encoder* to parametrize φ
2. Use *suitably designed neural networks* to parametrize M, W, V, g (OnsagerNet)



H. Yu, X. Tian, W. E, and Q. Li, "Onsagernet: Learning stable and interpretable dynamics using a generalized onsager principle," *arXiv preprint arXiv:2009.02327*, 2020

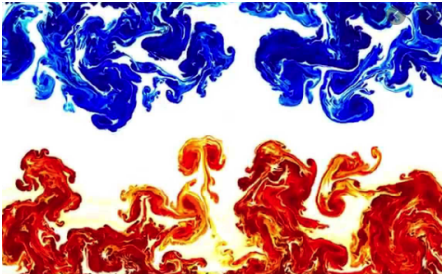
Application: Rayleigh-Bénard convection (RBC)

The RBC equations for $x = (u, \theta)$

$$\frac{du}{dt} + (u \cdot \nabla)u = \nu \Delta u - \nabla p + \alpha_0 g \theta$$

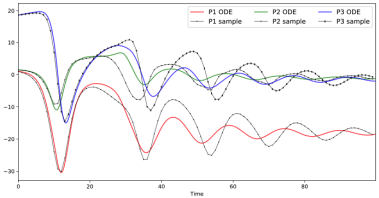
$$\nabla \cdot u = 0$$

$$\frac{d\theta}{dt} + u \cdot \nabla \theta = \kappa \Delta \theta + u_y \Gamma$$

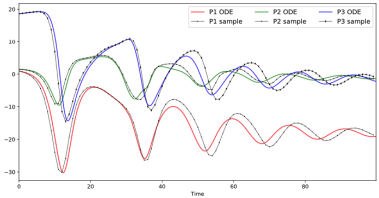


- u: velocity field
- θ: temperature profile

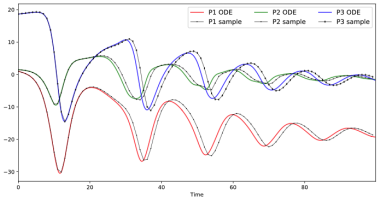
Lorenz '63 model is a reduction of the RBC equations by keeping the 3 most unstable Fourier modes, but it is quantitatively inaccurate



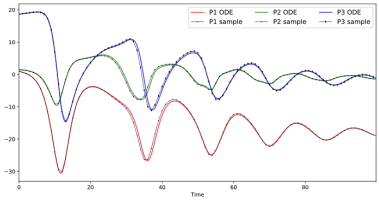
(c) $m = 3$



(d) $m = 5$



(e) $m = 7$



(f) $m = 9$

Figure: Results using different numbers of principle components.

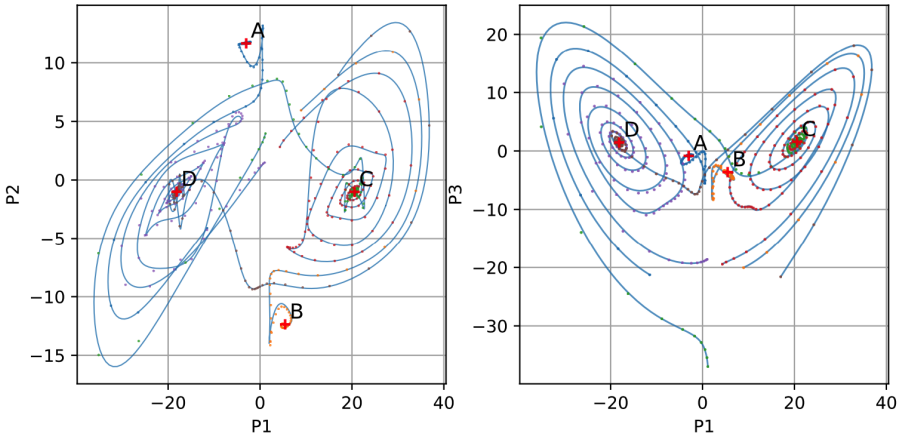
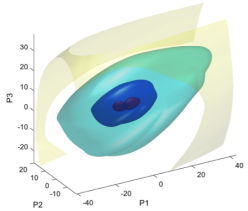
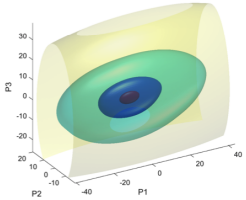


Figure: Trajectories simulated used learned ODE for the RBC problem with $r = 28$, $m = 9$. Solid curves: learned ODE. Dots: sample data

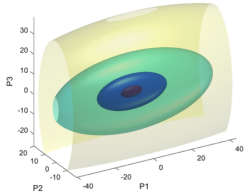
Visualizing learned energy function



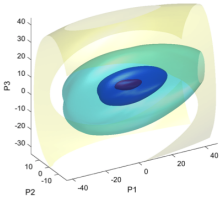
(a) PCA $m = 3$



(b) PCA $m = 5$



(c) PCA $m = 9$



(d) AE $m = 3$

Machine Learning of Dynamical Systems

Machine Learning for Rare Events

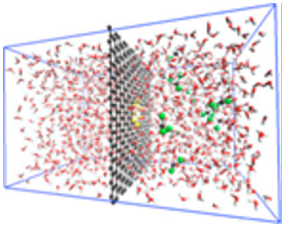
Rare Events

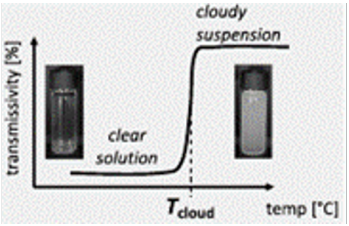
Many important applications

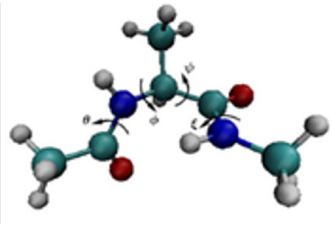
- Structure of macro-molecules
- Chemical reactions
- Nucleation in phase transitions

Quantities of interest

- Transition rates
- Potential landscapes
- Invariant distributions





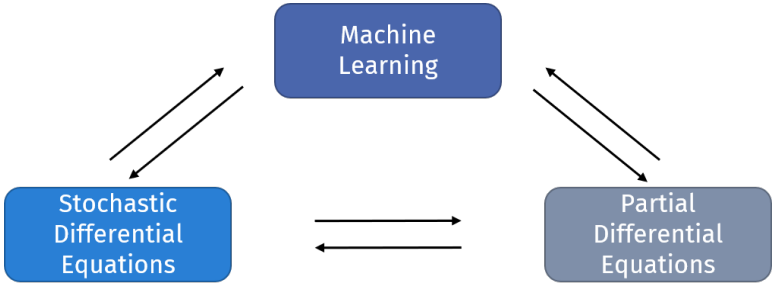


Challenge: high dimensions!

Machine Learning for Rare Events

Machine learning can deal with high-dimensions in a data-driven manner

- Computing commitor functions (reaction rates) in high-dimensions [L, Lin & Ren, 19]
- Computing quasipotentials in high-dimensions [Lin, L & Ren, 20]



Q. Li, B. Lin, and W. Ren, "Computing committor functions for the study of rare events using deep learning," *The Journal of Chemical Physics*, vol. 151, no. 5, p. 054 112, 2019

B. Lin, Q. Li, and W. Ren, "A data driven method for computing quasipotentials," *Mathematical and Scientific Machine Learning, MSML 2021 [arXiv preprint arXiv:2012.09111]*, 2020

Some Open Problems

- ML via DS
  - Theory for continuum limits
  - HJB based learning algorithms
- ML for DS
  - Non-linear extensions
  - Analysis of encoder-decoder models
  - Analysis of attention models
- ML of DS
  - Approximation theory for structured models
  - Data-dependent generalization results

More detailed account of ML via DS: lecture notes at https://blog.nus.edu.sg/qianxiaoli/teaching/

Thank you!