Machine Learning: Mathematical Theory and Scientific Applications

Weinan E

Joint work with:

Jiequn Han, Arnulf Jentzen, Chao Ma, Zheng Ma,
Han Wang, Qingcan Wang, Lei Wu, Linfeng Zhang, Yajun Zhou

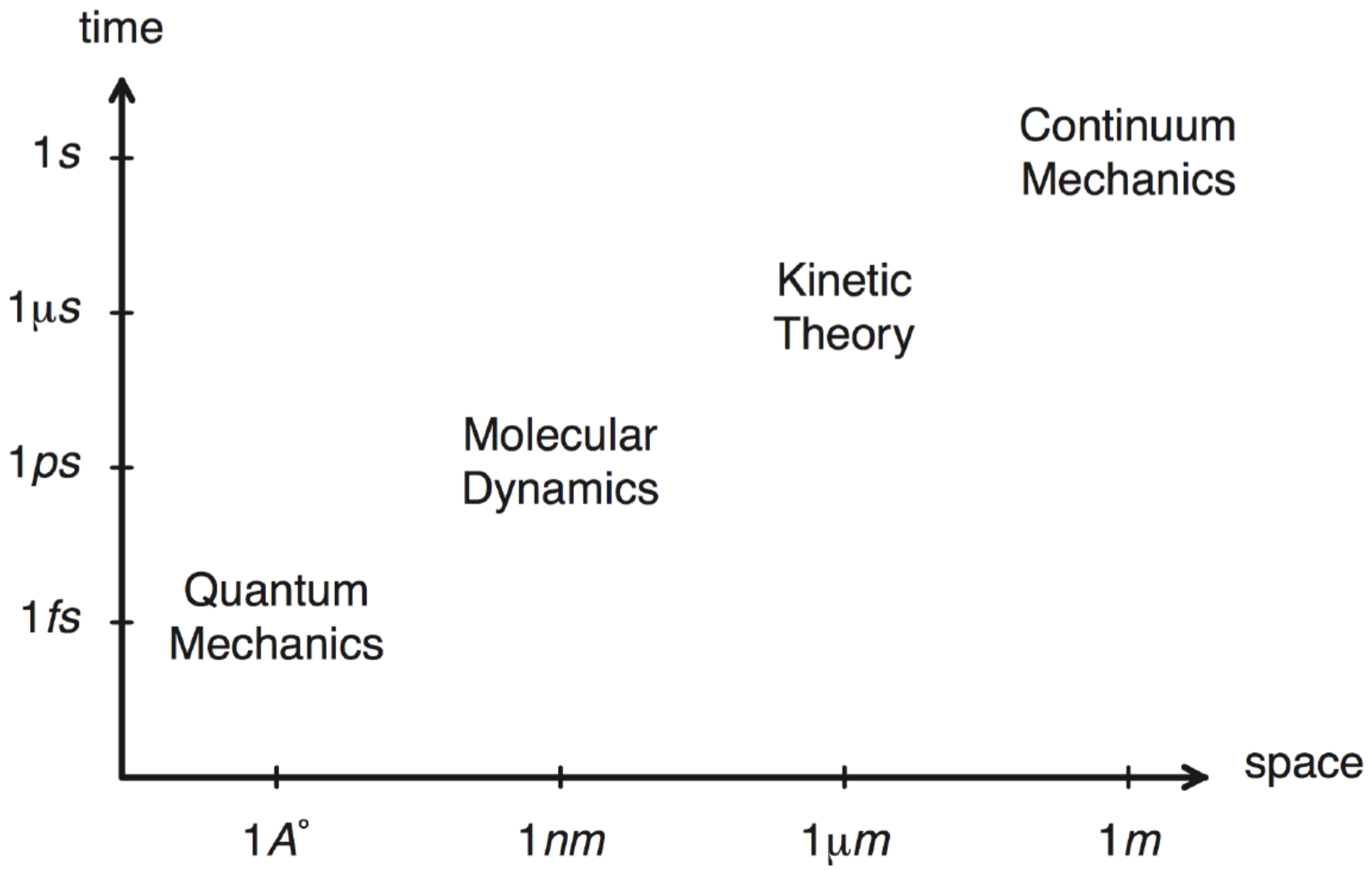
Roberto Car, Wissam A. Saidi

Outline

Outline

PDEs and fundamental laws of physics

- Navier-Stokes equations
- Boltzmann equation
- Schrödinger equation
- ......



Dirac's claim (1929)

"The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. "

Dealing with the mathematical difficulty: Making (drastic/ad hoc) approximations (e.g. density functional theory)

Outline

Numerical methods: 50's-80's

- finite difference
- finite element
- spectral methods
- ......

These have changed the way we do science, and to an even greater extend, engineering.

- gas dynamics/fluid dynamics
- structural analysis
- waves, optics
- control of simple systems
- ......

If the finite difference method were invented today, the shock wave that it would generate would be just as strong as the one generated by deep learning.

Many difficult problems remain

and we still have to rely on ad hoc approximations.

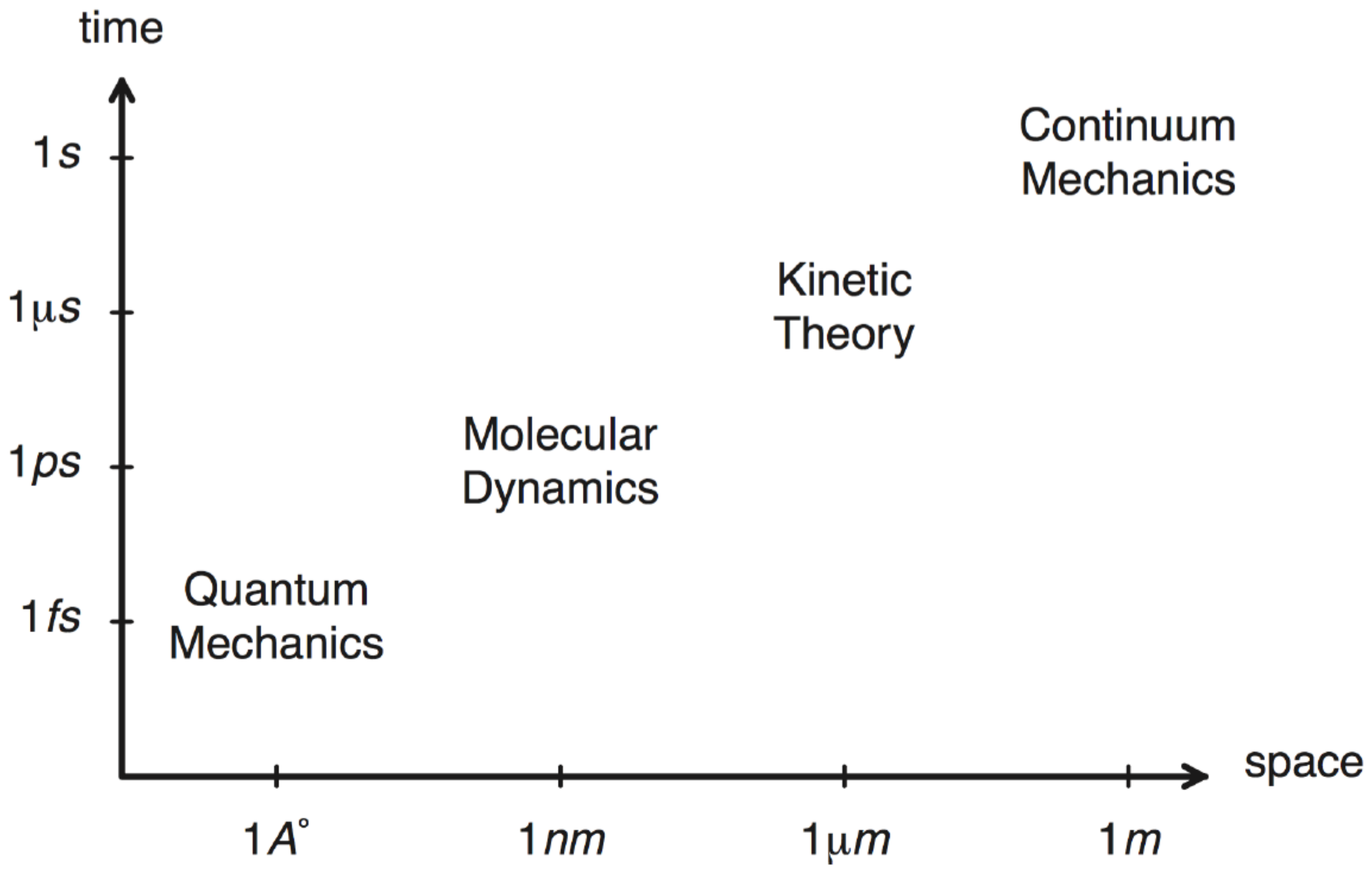
- many-body problems (classical and quantum, in molecular science)
- protein folding
- first principle-based drug and materials design
- coarse-grained molecular dynamics
- polymeric fluids
- transitional flows in gas dynamics
- plasticity
- turbulence, weather forecasting
- ......

Common feature of these problems: Dependence on many variables.

Curse of dimensionality: As the dimension grows, the complexity (or computational cost) grows exponentially.

Outline

Multiscale, multi-physics modeling: 90's till now



- works well when the macro- and micro-scales are very well separated
- not very effective when there are no separation of scales (e.g. turbulence problem)

Status summary

- Solved: low dimensional problems (few dependent variables)
- Unsolved: high dimensional problems (many dependent variables)

Machine learning, particularly deep learning, seems to be a powerful tool for high dimensional problems.

Outline

Supervised learning: Approximating functions using data

- Object of interest: $f^* : \mathbb{R}^d \to \mathbb{R}^1$, with respect to a prob measure μ on $\mathbb{R}^d$.
- Given a set of samples from μ, $\{\boldsymbol{x}_j\}_{j=1}^n$, and $\{y_j = f^*(\boldsymbol{x}_j)\}_{j=1}^n$
- Task: Approximate f^* using $S = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^n$.

Strategy:

- Construct some "hypothesis space" (a space of functions) $\mathcal{H}_m$ ($m \sim$ the dimension of $\mathcal{H}_m$).
  - linear regression: $f(\boldsymbol{x}, \theta) = \beta \cdot \boldsymbol{x} + \beta_0$
  - generalized linear models: $f(\boldsymbol{x}, \theta) = \sum_{k=1}^m c_k \phi_k(\boldsymbol{x})$, where $\{\phi_k\}$ are linearly independent functions.
  - two-layer neural networks: $f(\boldsymbol{x}, \theta) = \sum_k a_k \sigma(\boldsymbol{b}_k \cdot \boldsymbol{x} + c_k)$, where σ is some nonlinear function, e.g. $\sigma(z) = \max(z, 0)$.
  - deep neural networks (DNN) : compositions of functions of the form above.
- Minimize the "empirical risk":

$$\hat{\mathcal{R}}_n(\theta) = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j, \theta) - y_j)^2 = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j, \theta) - f^*(\boldsymbol{x}_j))^2$$

Classical numerical analysis (approximation theory)

- Define a “well-posed” math model (the hypothesis space, the loss function, etc)
    - splines: hypothesis space $= C^1$ piecewise cubic polynomials the data

$$I_n(f) = \frac{1}{n}\sum_{j=1}^{n}(f(x_j) - y_j)^2 + \lambda \int |D^2 f(x)|^2 dx$$

    - finite elements: hypothesis space $= C^0$ piecewise polynomials
- Identify the right function spaces, e.g. Sobolev/Besov spaces
    - direct and inverse approximation theorem (Bernstein and Jackson type theorems): f can be approximated by trig polynomials in L^2 to order s iff $f \in H^s$, $\|f\|_{H^s}^2 = \sum_{k=0}^{s} \|\nabla^k f\|_{L^2}^2$.
    - functions of interest are in the right spaces (PDE theory, real analysis, etc).
- Optimal error estimates
    - A priori estimates (for piecewise linear finite elements, $\alpha = 1/d, s = 2$)

$$\|f_m - f^*\|_{H^1} \le C m^{-\alpha} \|f^*\|_{H^s}$$

    - A posteriori estimates (say in finite elements):

$$\|f_m - f^*\|_{H^1} \le C m^{-\alpha} \|f_m\|_h$$

Key issue: Dependence on dimensionality

Another benchmark: High dimensional integration

Monte Carlo: $X = [0,1]^d, \{\boldsymbol{x}_j, j = 1, \cdots, n\}$ is uniformly distributed in X.

$$I(g) = \int_X g(\boldsymbol{x}) d\mu, \quad I_n(g) = \frac{1}{n} \sum_j g(\boldsymbol{x}_j)$$

$$\mathbb{E}(I(g) - I_n(g))^2 = \frac{1}{n} \text{var}(g)$$

$\text{var}(g) = \int_X g^2(\boldsymbol{x}) d\boldsymbol{x} - (\int_X g(\boldsymbol{x}) d\boldsymbol{x})^2$

The $O(1/\sqrt{n})$ rate is basically the best we can hope for.

However, $\text{var}(g)$ can be very large in high dimension. That's why variance reduction is important!

Second issue: Finite amount of data

Empirical risk vs. population risk:

$$\hat{\mathcal{R}}_n(\theta) = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j, \theta) - y_j)^2 = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j, \theta) - f^*(\boldsymbol{x}_j))^2$$

$$\mathcal{R}(\theta) = \mathbb{E}(f(\boldsymbol{x}, \theta) - f^*(\boldsymbol{x}))^2 = \int_{\mathbb{R}^d} (f(\boldsymbol{x}, \theta) - f^*(\boldsymbol{x}))^2 d\mu$$

Two important parameters: n (the number of data) and m (the number of parameters).
Three regimes:

- $m < n$ under-parameterized regime
- $m > n$ over-parametrized regime
- $m \sim n$ competing regime

Estimating the generalization gap

"Generalization gap" $= \hat{\mathcal{R}}(\hat{\theta}) - \hat{\mathcal{R}}_n(\hat{\theta}) = I(g) - I_n(g), \quad g(\boldsymbol{x}) = (f(\boldsymbol{x}, \hat{\theta}) - f^*(\boldsymbol{x}))^2$

$$I(g) = \int_{X=[-1,1]^d} g(\boldsymbol{x}) d\mu, \quad I_n(g) = \frac{1}{n} \sum_j g(\boldsymbol{x}_j)$$

g is very correlated with the data.

Current strategy: Study the worst case situation.

- For Lipschitz functions (Wasserstein distance)

$$\sup_{\|h\|_{Lip} \leq 1} |I(h) - I_n(h)| \sim \frac{1}{n^{1/d}}$$

- For functions in Barron space, to be defined later

$$\sup_{\|h\|_{\mathcal{B}} \leq 1} |I(h) - I_n(h)| \sim \frac{1}{\sqrt{n}}$$

Rademacher complexity

Let $\mathcal{H}$ be a set of functions, and $S = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ be a set of data points. Then, the Rademacher complexity of $\mathcal{H}$ with respect to S is defined as

$$\hat{R}_S(\mathcal{H}) = \frac{1}{n}\mathbb{E}_\xi\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^n \xi_i h(\boldsymbol{x}_i)\right],$$

where $\{\xi_i\}_{i=1}^n$ are i.i.d. random variables taking values ± 1 with equal probability.

Theorem (Rademacher complexity and the generalization gap)

Given a function class $\mathcal{H}$, for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the random samples $S = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$,

$$\sup_{h\in\mathcal{H}}\left|\mathbb{E}_{\boldsymbol{x}}\left[h(\boldsymbol{x})\right] - \frac{1}{n}\sum_{i=1}^n h(\boldsymbol{x}_i)\right| \leq 2\hat{R}_S(\mathcal{H}) + \sup_{h\in\mathcal{H}}\|h\|_\infty\sqrt{\frac{\log(2/\delta)}{2n}}.$$

$$\sup_{h\in\mathcal{H}}\left|\mathbb{E}_{\boldsymbol{x}}\left[h(\boldsymbol{x})\right] - \frac{1}{n}\sum_{i=1}^n h(\boldsymbol{x}_i)\right| \geq \frac{1}{2}\hat{R}_S(\mathcal{H}) - \sup_{h\in\mathcal{H}}\|h\|_\infty\sqrt{\frac{\log(2/\delta)}{2n}}.$$

- If $\mathcal{H} =$ unit ball in C^0: $\hat{R}_S(\mathcal{H}) \sim O(1)$
- If $\mathcal{H} =$ unit ball in Lipschitz space: $\hat{R}_S(\mathcal{H}) \sim O(1/n^{1/d})$
- If $\mathcal{H} =$ unit ball in Barron space: $\hat{R}_S(\mathcal{H}) \sim O(1/\sqrt{n})$

Two types of machine learning models

(1). Models that suffer from the curse of dimensionality:

$$\text{generalization error} = O(m^{-\alpha/d} + n^{-\beta/d})$$

- piecewise polynomial approximation
- wavelets with fixed wavelet basis

(2). Models that don't suffer from the curse of dimensionality:

$$\text{generalization error} = O(\gamma_1(f^*)/m + \gamma_2(f^*)/\sqrt{n})$$

- random feature models:
- shallow neural networks
- deep residual neural networks

Example 1: Random feature model

$\{\phi(\cdot;\omega)\}$: collection of random features. π: prob distribution of the random variable ω.

Example: $\phi(\boldsymbol{x},\omega) = \sigma(\omega^T \boldsymbol{x})$, $\sigma(z) = \max(z,0)$.

Hypothesis space: Given any realization $\{\omega_j\}_{j=1}^m$, i.i.d. with distribution π

$$\mathcal{H}_m(\{\omega_j\}) = \{f_m(\boldsymbol{x},\boldsymbol{a}) = \frac{1}{m}\sum_{j=1}^m a_j \phi(\boldsymbol{x};\omega_j).\}.$$

Function space

Consider functions of the form

$$\mathcal{H}_\phi = \{f : f(\boldsymbol{x}) = \int a(\omega)\phi(\boldsymbol{x};\omega)d\pi(\omega)\}, \quad \|f\|^2_{\mathcal{H}_\phi} = \mathbb{E}_{\omega\sim\pi}[|a(\omega)|^2]$$

This is the same as the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ for the kernel:

$$k(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\omega\sim\pi}[\phi(\boldsymbol{x};\omega)\phi(\boldsymbol{x}';\omega)]$$

A priori estimates of the regularized model

$$L_n(\theta) = \hat{\mathcal{R}}_n(\theta) + \lambda\sqrt{\frac{\log(2d)}{n}}\|\theta\|_{\mathcal{H}}, \qquad \hat{\theta}_n = \text{argmin } L_n(\theta)$$

where

$$\|\theta\|_{\mathcal{H}} = \left(\frac{1}{m}\sum_{j=1}^{m}|a_j|^2\right)^{1/2}$$

Theorem

Assume that the target function $f^*: [0,1]^d \mapsto [0,1] \in \mathcal{H}_k$. There exist constants C_0, C_1, C_2, such that for any $\delta > 0$, if $\lambda \geq C_0$, then with probability at least $1-\delta$ over the choice of training set, we have

$$\mathcal{R}(\hat{\theta}_n) \leq C_1\left(\frac{\|f^*\|^2_{\mathcal{H}_k}}{m} + \|f^*\|_{\mathcal{H}_k}\sqrt{\frac{\log(2d)}{n}}\right) + C_2\sqrt{\frac{\log(4C_2/\delta)+\log(n)}{n}}.$$

Summary of the general strategy

- express the target function as an expectation
- identify the right function space (including the norm)
- prove direct and inverse approximation theorems
- study the Rademacher complexity
- prove a priori estimates

Example 2: Two-layer neural networks

$$\mathcal{H}_m = \{\frac{1}{m}\sum_{j=1}^{m} a_j \sigma(\boldsymbol{b}_j^T \boldsymbol{x} + c_j)\}$$

Consider $f : X = [-1,1]^d \mapsto \mathbb{R}$ of the following form ("generalized ridgelet transform")

$$f(\boldsymbol{x}) = \int_\Omega a\sigma(\boldsymbol{b}^T \boldsymbol{x} + c)\rho(da, d\boldsymbol{b}, dc) = \mathbb{E}_{(a,\boldsymbol{b},c)\sim\rho} a\sigma(\boldsymbol{b}^T \boldsymbol{x} + c)$$

$\Omega = \mathbb{R}^1 \times \mathbb{R}^d \times \mathbb{R}^1$, ρ is a probability distribution on Ω.

$$\|f\|_{\mathcal{B}_p} = \inf_\rho \left(\mathbb{E}_\rho[|a|^p(\|\boldsymbol{b}\|_1 + |c|)^p]\right)^{1/p}$$

Barron space:

$$\mathcal{B}_p = \text{completion} \quad \text{of} \quad \{f \in \mathcal{S} : \|f\|_{\mathcal{B}_p} < \infty\}, \quad \mathcal{B}_\infty \subset \cdots \mathcal{B}_2 \subset \mathcal{B}_1.$$

Barron space and RKHS

Note that we can rewrite "Barron functions" in the form (now π depends on f).

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d \times \mathbb{R}^1} a(\omega)\sigma(\omega^T \boldsymbol{x}) d\pi(\omega)$$

Define:

$$k_\pi(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\omega \sim \pi} \sigma(\omega^T \boldsymbol{x}) \sigma(\omega^T \boldsymbol{x}')$$

We can write

$$\mathcal{B}_2 = \bigcup_\pi \mathcal{H}_{k_\pi}$$

Two-layer neural network can be understood as random feature model with adaptive (learned) features.

What kind of functions admit such a representation?

Theorem (Barron and Klusowski (2016)): If $\int_{\mathbb{R}^d} \|\omega\|_1^2 |\hat{f}(\omega)| d\omega < \infty$, where $\hat{f}$ is the Fourier transform of f, then f can be represented as

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) - (f(0) + \boldsymbol{x} \cdot \nabla f(0)) = \int_\Omega a\sigma(\boldsymbol{b}^T \boldsymbol{x} + c)\rho(da, d\boldsymbol{b}, dc)$$

where $\sigma(z) = \max(0, z)$. Moreover $f \in \mathcal{B}_\infty$. Furthermore, we have

$$\|\tilde{f}\|_{\mathcal{B}_\infty} \leq 2 \int_{\mathbb{R}^d} \|\omega\|_1^2 |\hat{f}(\omega)| d\omega$$

Direct and inverse approximation theorems

Theorem (Direct Approximation Theorem)

There exists an absolute constant C_0 such that

$$\|f - f_m\|_{L^2(X)} \leq \frac{C_0 \|f\|_{\mathcal{B}_2}}{\sqrt{m}}$$

Theorem (Inverse Approximation Theorem)

For $p > 1$, let

$$\mathcal{N}_{p,C} = \{ \frac{1}{m} \sum_{k=1}^{m} a_k \sigma(b_k^T \boldsymbol{x} + c_k) : \left(\frac{1}{m} \sum_{k=1}^{m} |a_k|^p (\|b_k\|_1 + c_k)^p \right)^{1/p} \leq C, m \in \mathbb{N}^+ \}.$$

Let f^* be a continuous function. Assume there exists a constant C and a sequence of functions $f_m \in \mathcal{N}_{p,C}$ such that

$$f_m(\boldsymbol{x}) \to f^*(\boldsymbol{x})$$

for all $\boldsymbol{x} \in X$. Then there exists a probability distribution ρ on Ω, such that

$$f^*(\boldsymbol{x}) = \int a \sigma(\boldsymbol{b}^T \boldsymbol{x} + c) \rho(da, d\boldsymbol{b}, dc),$$

for all $\boldsymbol{x} \in X$.

Complexity estimates

Theorem

Let $\mathcal{F}_Q = \{f \in \mathcal{B}_1, \|f\|_{\mathcal{B}_1} \le Q\}$. *Then we have*

$$\hat{\mathcal{R}}_n(\mathcal{F}_Q) \le 2Q\sqrt{\frac{2\ln(2d)}{n}}$$

A priori estimates for regularized model

where the path norm is defined by:

$$L_n(\theta) = \hat{\mathcal{R}}_n(\theta) + \lambda\sqrt{\frac{\log(2d)}{n}}\|\theta\|_{\mathcal{P}}, \qquad \hat{\theta}_n = \text{argmin } L_n(\theta)$$

$$\|\theta\|_{\mathcal{P}} = \frac{1}{m}\sum_{k=1}^{m}|a_k|(\|\boldsymbol{b}_k\|_1 + |c_k|) \quad (= \|f(\cdot;\theta)\|_{\mathcal{B}_1})$$

Theorem (Weinan E, Chao Ma, Lei Wu, 2018)

Assume that the target function $f^*: [0,1]^d \mapsto [0,1] \in \mathcal{B}_2$. There exist constants C_0, C_1, C_2, such that for any $\delta > 0$, if $\lambda \geq C_0$, then with probability at least $1-\delta$ over the choice of training set, we have

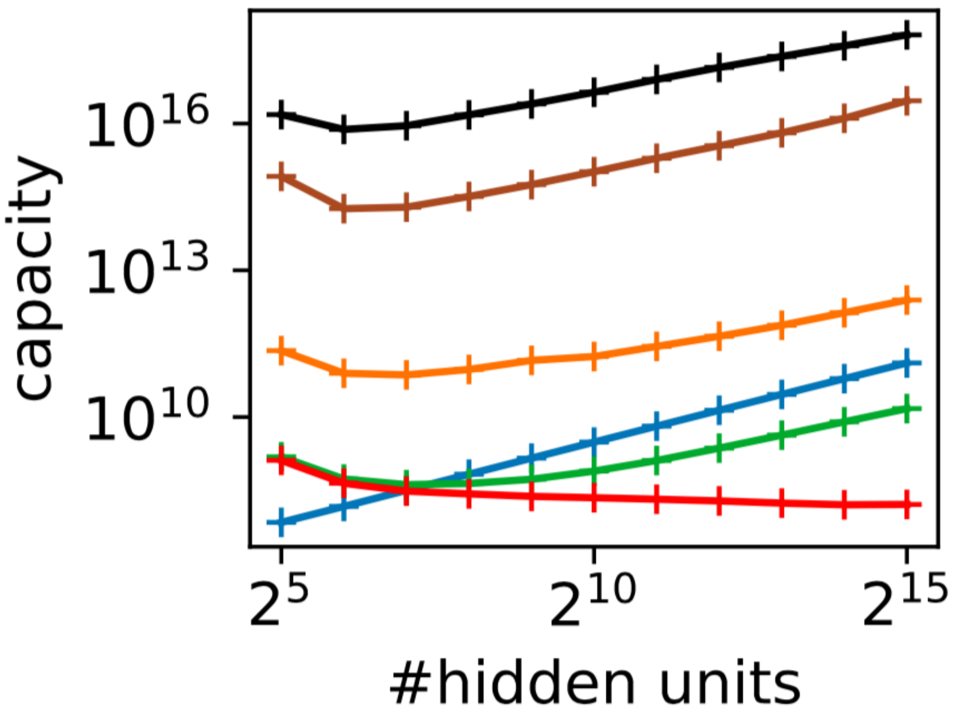
$$\mathcal{R}(\hat{\theta}_n) \leq C_1\left(\frac{\|f^*\|_{\mathcal{B}_2}^2}{m} + \|f^*\|_{\mathcal{B}_2}\sqrt{\frac{\log(2d)}{n}}\right) + C_2\sqrt{\frac{\log(4C_2/\delta)+\log(n)}{n}}.$$

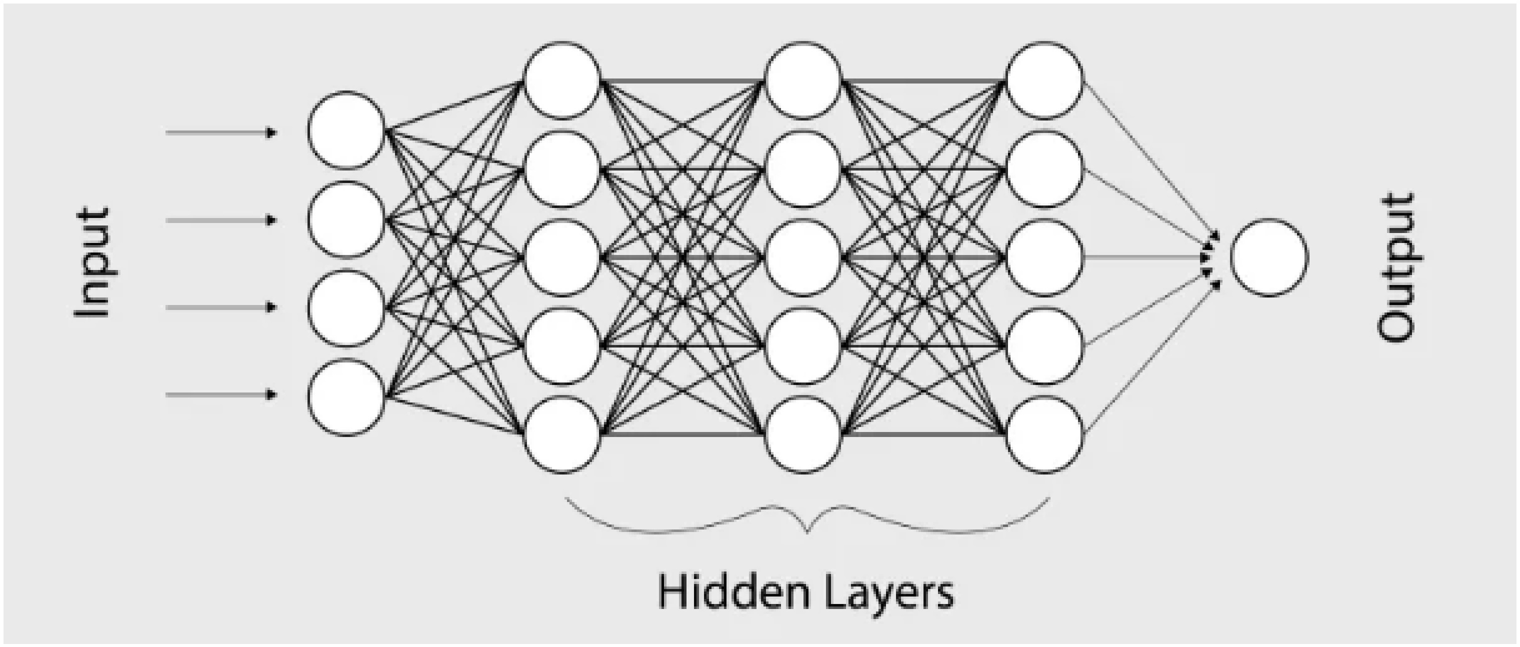
An example of posteriori estimates

Behnam Neyshabur, Zhiyuan Li, et al. (2018):

$$|\mathcal{R}(\theta) - \hat{\mathcal{R}}_n(\theta)| \leq C_1(|||\theta\,||| + 1)\sqrt{\frac{\log(2d)}{n}} + C_2\sqrt{\frac{\log(4C_2(1 + |||\theta|||))^2/\delta)}{n}}$$

where $|||\theta|||$ is some norm of θ





$$f(\boldsymbol{x}, \theta) = \boldsymbol{W}_L \sigma \circ (\boldsymbol{W}_{L-1} \sigma \circ (\cdots \sigma \circ (\boldsymbol{W}_0 \boldsymbol{x}))), \quad \theta = (\boldsymbol{W}_0, \boldsymbol{W}_1, \cdots, \boldsymbol{W}_L)$$

σ is a scalar function.
"$\circ$" means acting on each components, the $\boldsymbol{W}$'s are matrices.

Instability: Exploding and vanishing gradients

$$f(\boldsymbol{x}, \theta) = \boldsymbol{W}_L \sigma \circ (\boldsymbol{W}_{L-1} \sigma \circ (\cdots \sigma \circ (\boldsymbol{W}_0 \boldsymbol{x}))), \quad \theta = (\boldsymbol{W}_0, \boldsymbol{W}_1, \cdots, \boldsymbol{W}_L)$$

Instability of fully connected deep neural networks:

$$\text{``}\nabla_\theta f'' = \boldsymbol{W}_L \cdot \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_0 \sim \kappa^L, \quad L >> 1$$

"Exploding or vanishing gradients problem" (see Hanin (2018)).

Solution: Using residual networks (He et al. (2016))

$$\begin{aligned} \boldsymbol{z}_{0,L}(\boldsymbol{x}) &= \boldsymbol{V}\boldsymbol{x}, \\ \boldsymbol{z}_{l+1,L}(\boldsymbol{x}) &= z_{l,L}(\boldsymbol{x}) + \frac{1}{L}\boldsymbol{U}_l \sigma \circ (\boldsymbol{W}_l z_{l,L}(\boldsymbol{x})), \quad l = 0, 1, \cdots, L-1 \end{aligned}$$

$$f(\boldsymbol{x}, \theta) = \alpha \cdot \boldsymbol{z}_{L,L}(\boldsymbol{x})$$

Dynamical system viewpoint

E (2017, Comm Math Stats), Chen et al (NeurIPS 2018, “Neural ODE”):

Constructing a rich class of nonlinear functions through the **flow map** of a dynamical system with different choices of $\boldsymbol{F}$:

$$\frac{d\boldsymbol{z}(\boldsymbol{x},t)}{dt} = \boldsymbol{F}(\boldsymbol{z}(\boldsymbol{x},t)), \quad \boldsymbol{z}(0,\boldsymbol{x}) = \boldsymbol{V}\boldsymbol{x}$$

The flow map $\boldsymbol{x} \to \boldsymbol{z}(\boldsymbol{x},1)$ is a nonlinear mapping.

Choose the optimal $\boldsymbol{U}, \boldsymbol{W}(\cdot), \alpha$ to approximate f^* by

$$f^*(\boldsymbol{x}) \sim \alpha \cdot \boldsymbol{z}(\boldsymbol{x},1)$$

Simplest choice of (nonlinear) $\boldsymbol{F}$:

$$\boldsymbol{F}(\boldsymbol{z}; \boldsymbol{U}(t), \boldsymbol{W}(t)) = \boldsymbol{U}(t)\sigma \circ (\boldsymbol{W}(t)\boldsymbol{z})$$

Plan of attack for residual networks

$$
\begin{aligned}
\boldsymbol{z}_{l+1,L}(\boldsymbol{x}) &= z_{l,L}(\boldsymbol{x}) + \frac{1}{L}\boldsymbol{U}_l\sigma \circ (\boldsymbol{W}_l z_{l,L}(\boldsymbol{x})), \\
\boldsymbol{z}_{0,L}(\boldsymbol{x}) &= \boldsymbol{V}\boldsymbol{x}, \quad f(\boldsymbol{x},\theta) = \alpha \cdot \boldsymbol{z}_{L,L}(\boldsymbol{x})
\end{aligned}
$$

- express target function as a compositional expectation (compositional law of large numbers)
- compositional function spaces (Barron space is embedded in compositional function spaces)
- direct and inverse approximation theorem
- optimal scaling for the Rademacher complexity
- a priori estimates for the regularized model

Compositional law of large numbers

Consider the scheme (compositions of random near identity maps):

$$\begin{aligned} \boldsymbol{z}_{0,L}(\boldsymbol{x}) &= \boldsymbol{V}\boldsymbol{x}, \\ \boldsymbol{z}_{l+1,L}(\boldsymbol{x}) &= z_{l,L}(\boldsymbol{x}) + \frac{1}{L}\boldsymbol{U}_l \sigma \circ (\boldsymbol{W}_l z_{l,L}(\boldsymbol{x})), \end{aligned}$$

$(\boldsymbol{U}_l, \boldsymbol{W}_l)$ are i.i.d. sampled from a distribution ρ. Then we have the following convergence theorem for $z_{L,L}$.

Theorem

Assume that

$$\mathbb{E}_\rho \| |\boldsymbol{U}| |\boldsymbol{W}| \|_F^2 < \infty$$

where for a matrix $\boldsymbol{A}$, $|\boldsymbol{A}|$ means taking element-wise absolute value for $\boldsymbol{A}$. Define $\boldsymbol{z}(\boldsymbol{x}, t)$ by

$$\begin{aligned} \boldsymbol{z}(\boldsymbol{x}, 0) &= \boldsymbol{V}\boldsymbol{x}, \\ \frac{d}{dt}\boldsymbol{z}(\boldsymbol{x}, t) &= \mathbb{E}_{(\boldsymbol{U},\boldsymbol{W})\sim\rho} U \sigma \circ (\boldsymbol{W}\boldsymbol{z}(\boldsymbol{x}, t)). \end{aligned}$$

Then we have

$$\boldsymbol{z}_{L,L}(\boldsymbol{x}) \to \boldsymbol{z}(\boldsymbol{x}, 1)$$

almost surely as $L \to +\infty$.

Compositional function spaces

Let $\{\rho_t\}$ be a family of prob distributions (for $(\boldsymbol{U}, \boldsymbol{W})$) such that $\mathbb{E}_{\rho_t} g(\boldsymbol{U}, \boldsymbol{W})$ is integrable as a function of t for any continuous function g. Define:

$$\begin{aligned} \boldsymbol{z}(\boldsymbol{x}, 0) &= \boldsymbol{V}\boldsymbol{x}, \\ \frac{d}{dt}\boldsymbol{z}(\boldsymbol{x}, t) &= \mathbb{E}_{(\boldsymbol{U},\boldsymbol{W})\sim\rho_t} \boldsymbol{U}\sigma \circ (\boldsymbol{W}\boldsymbol{z}(\boldsymbol{x}, t)) \end{aligned}$$

Let $f_{\alpha,\{\rho_t\},\boldsymbol{V}}(\boldsymbol{x}) = \alpha^T \boldsymbol{z}(\boldsymbol{x}, 1)$ and define

$$\begin{aligned} \frac{d}{dt}\mathbf{N}_p(t) &= (\mathbb{E}_{\rho_t}|\boldsymbol{U}|^p|\boldsymbol{W}|^p)^{1/p}\mathbf{N}_p(t), \quad \mathbf{N}_p(0) = \mathbf{I} \\ \|f\|_{\mathcal{D}_p} &= \inf_{f=f_{\alpha,\{\rho_t\},\boldsymbol{V}}} \|\alpha\|_p \, \|\mathbf{N}_p(1)\|_{p,p} \, \|\boldsymbol{V}\|_{p,p}, \end{aligned}$$

Barron space and compositional function space

$$f(\boldsymbol{x}) = \int_\Omega a\sigma(\boldsymbol{b}^T\boldsymbol{x} + c)\rho(da, d\boldsymbol{b}, dc)$$

$$\begin{aligned} \frac{d}{dt}z(\boldsymbol{x}, t) &= \mathbb{E}_{\rho(a,\ \boldsymbol{b},\ c)} \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix} \sigma \circ ([0, \boldsymbol{b}^T, c]z(\boldsymbol{x}, t)), \\ z(\boldsymbol{x}, 0) &= \begin{bmatrix} 0 \\ \boldsymbol{x} \\ 1 \end{bmatrix}. \end{aligned}$$

Then, $f(\boldsymbol{x}) = e_1^T z(\boldsymbol{x}, 1)$. In addition, we have

Theorem

$\mathcal{B}_2 \subset \mathcal{D}_2$. There exists constant $C > 0$, such that

$$\|f\|_{\mathcal{D}_2} \leq \sqrt{d+1}\|f\|_{\mathcal{B}_2}$$

holds for any $f \in \mathcal{B}_2$,

Theorem (Inverse approximation theorem)

Let $f \in L^2(D_0)$ be a continuous function. Assume that there is a sequence of residual networks $\{f_L(\cdot)\}_{L=1}^{\infty}$ with increasing depth such that $f_L(\boldsymbol{x}) \to f(\boldsymbol{x})$, for all $\boldsymbol{x} \in D_0$. Assume further that the parameters are (entry-wise) bounded, then there exists α, $\{\rho_t\}$ and $\boldsymbol{V}$ such that $\mathbb{E}_{\rho_t} g(\boldsymbol{U}, \boldsymbol{W})$ is integrable as a function of t for any continuous function g and

$$f(\boldsymbol{x}) = f_{\alpha,\{\rho_t\},\boldsymbol{V}}(\boldsymbol{x}).$$

Theorem (Direct approximation theorem)

Let $f \in L^2(D_0) \cap \mathcal{D}_2$. There exists a residue-type neural network $f_L(\cdot)$ of input dimension $d+1$ and depth L such that $\|f_L\|_P \lesssim \|f\|_{c_1}^3$ and

$$\int_{D_0} |f(\boldsymbol{x}) - f_L((\boldsymbol{x})|^2 dx \to 0$$

Furthermore, if $f = f_{\alpha,\{\rho_t\},\boldsymbol{V}}$ and ρ_t is Lipschitz continuous in t (.....) then

$$\int_{D_0} |f(\boldsymbol{x}) - f_L((\boldsymbol{x})|^2 dx \lesssim \frac{\|f\|_{\mathcal{D}_2}^2}{L}$$

Complexity control

Theorem

Rademacher complexity bound Let $\mathcal{F}_{L,Q} = \{f_L : \|f_L\|_{\mathcal{D}_1} \le Q\}$. *Assume* $\boldsymbol{x}_i \in [-1,1]^d$. *Then, for any data set* $S = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$, *we have*

$$\hat{R}_S(\mathcal{F}_{L,Q}) \le 3Q\sqrt{\frac{2\log(2d)}{n}}.$$

Regularized model and a priori estimates

Regularized loss function:

$$J(\theta) = \hat{L}(\theta) + \lambda(\|\theta\|_{\mathcal{D}_1} + 1)\sqrt{\frac{2\log(2d)}{n}}.$$

$$\|\theta\|_P = \left\| |\alpha|^T \left(I + \frac{1}{L}|\boldsymbol{U}_{L-1}||\boldsymbol{W}_{L-1}|\right) \cdots \left(I + \frac{1}{L}|\boldsymbol{U}_1||\boldsymbol{W}_1|\right) |\boldsymbol{V}| \right\|_1$$

Theorem (Weinan E, Chao Ma and Qingcan Wang, 2018)

Assume that $f^\star : [-1,1]^d \to [-1,1]$ such that $f^* \in \mathcal{D}_2$. Let

$$\hat{\theta} = \textit{argmin}\,_\theta J(\theta)$$

Then there exist fixed constants C_0 and L_0 such that if $\lambda > C_0$ and $L > L_0$, then for any $\delta > 0$, with probability at least $1 - \delta$,

$$L(\hat{\theta}) \lesssim \frac{\|f^*\|_{\mathcal{D}_2}^2}{L} + \lambda(\|f^*\|_{\mathcal{D}_1}^3 + 1)\sqrt{\frac{\log(2d)}{n}} + \sqrt{\frac{\log(1/\delta)}{n}}.$$

Lots of questions remain open:

- do these estimates truly reflect what is going on? are they sharp?
- how do the errors behave in different regimes (under-parametrized, over-parametrized, competing)
- why simple minded optimization algorithms seem to work well (with some tuning)? landscapes, gradient descent dynamics

Outline

Nonlinear parabolic PDEs

$$\frac{\partial u}{\partial t} + \frac{1}{2}\Delta u + \nabla u \cdot \mu + f\big(u, \nabla u\big) = 0.$$

- Terminal condition: $u(T, \boldsymbol{x}) = g(\boldsymbol{x})$.
- To fix ideas, we are interested in the solution at $t = 0$, $\boldsymbol{x} = \xi$ for some vector $\xi \in \mathbb{R}^d$.

Example:

- Black-Scholes Equation with Default Risk and many underlies:

$$f = -(1-\delta)Q(u)u - Ru$$

- Hamilton-Jacobi-Bellman in control theory

$$\frac{\partial u}{\partial t} + \frac{1}{2}\Delta u + H\big(\nabla u\big) = 0.$$

Linear Parabolic PDE and Feynman-Kac Formula

$$\frac{\partial u}{\partial t}(t,\boldsymbol{x}) + \frac{1}{2}\Delta u(t,\boldsymbol{x}) + \nabla u(t,\boldsymbol{x}) \cdot \mu(t,\boldsymbol{x}) + f(t,\boldsymbol{x}) = 0.$$

Terminal condition $u(T,\boldsymbol{x}) = g(\boldsymbol{x})$.

Let

$$dX_t = \mu(t, X_t)\, dt + dW_t,$$

Feynman-Kac formula:

$$u(t,\boldsymbol{x}) = \mathbb{E}[g(X_T) + \int_t^T f(s, X_s) ds | X_t = \boldsymbol{x}].$$

Compute the solution of PDE using Monte Carlo, overcoming the curse of dimensionality.

Nonlinear Feynman-Kac: Connection between PDE and BSDE

Backward stochastic differential equations (Pardoux and Peng 1992): Find an adapted process $\{(X_t, Y_t, Z_t)\}_{t\in[0,T]}$ such that

$$X_t = \xi + \int_0^t \mu(s, X_s)\, ds + W_t$$

$$Y_t = g(X_T) + \int_t^T f(Y_s, Z_s)\, ds - \int_t^T (Z_s)^{\mathrm{T}}\, dW_s$$

Ito's formula gives:

$$u(t, X_t) = g(X_T) + \int_t^T f(u(s, X_s), \nabla u(s, X_s))ds - \int_t^T \nabla u(s, X_s) dW_s.$$

Therefore we have:

$$Y_t = u(t, X_t), \quad Z_t = \nabla u(t, X_t)$$

New formulation of the nonlinear PDE

Consider the variational problem:

$$
\begin{aligned}
&\inf_{Y_0,\{Z_t\}_{0\le t\le T}} \mathbb{E}|g(X_T)-Y_T|^2,\\
s.t.\quad & X_t=\xi+\int_0^t \mu(s,X_s)\,ds+W_t,\\
& Y_t=Y_0-\int_0^t f(Y_s,Z_s)\,ds+\int_0^t (Z_s)^{\mathrm{T}}\,dW_s.
\end{aligned}
$$

The unique minimizer gives the solution to the PDE.

Deep BSDE method (E, Han, Jentzen (2017)

- Key step: approximate the function $x \mapsto \nabla u(t,x)$ at each discretized time step $t = t_n$ by a feedforward neural network (a subnetwork)

$$\nabla u(t_n, X_{t_n}) \approx \nabla u(t_n, X_{t_n} | \theta_n)$$

where θ_n denotes neural network parameters.

$$u(0, \boldsymbol{x}) \approx u(0, \boldsymbol{x} | \theta_0)$$

- Observation: after time discretization, we can stack all the subnetworks together to form a deep neural network (DNN) as a whole:

$$X_{t_{n+1}} - X_{t_n} \approx \mu(t_n, X_{t_n})\, \Delta t_n + \Delta W_n$$

$$u(t_{n+1}, X_{t_{n+1}}) - u(t_n, X_{t_n}) \approx -f(u(t_n, X_{t_n}), \nabla u(t_n, X_{t_n}))\Delta t_n + \nabla u(t_n, X_{t_n})\Delta W_n.$$

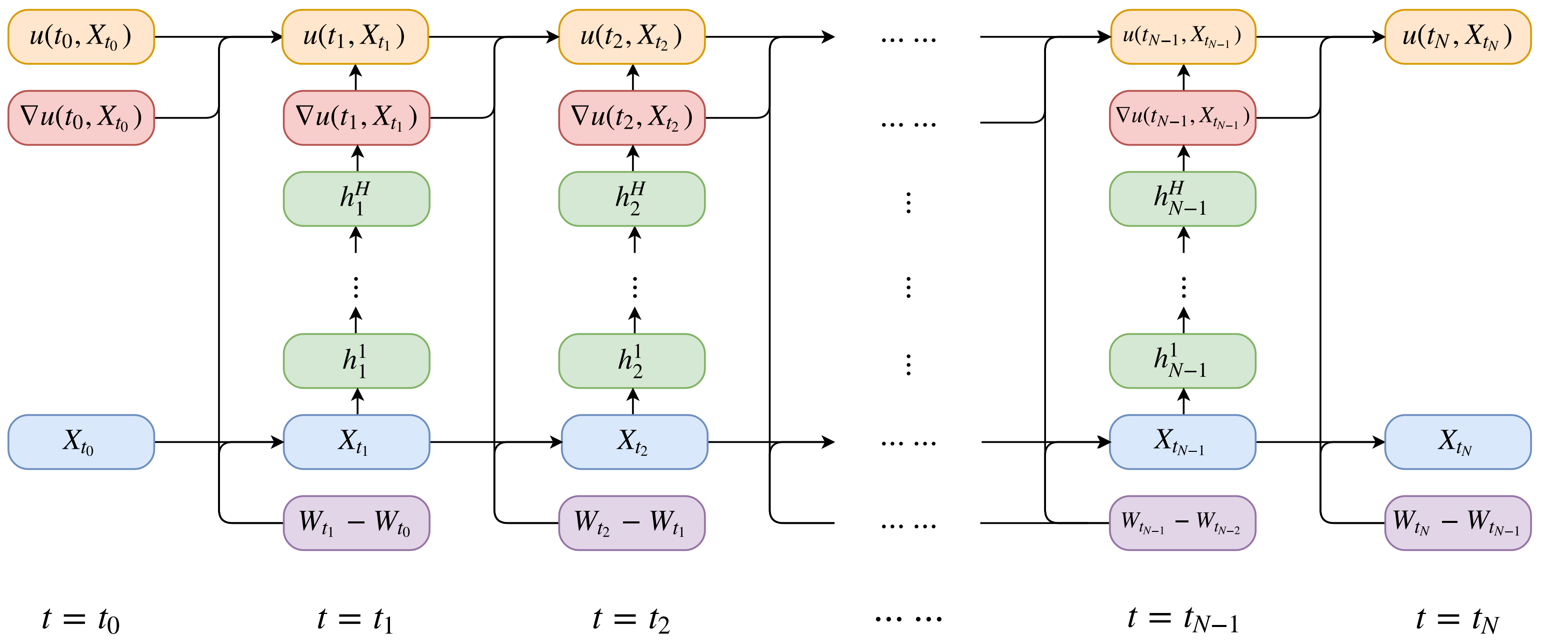


Figure: Each column corresponds to a subnetwork at time $t = t_n$

$$L(\theta) = \mathbb{E}\Big[\Big|g(X_{t_N}) - \hat{u}\big(\{X_{t_n}\}_{0\le n\le N}, \{W_{t_n}\}_{0\le n\le N}\big)\Big|^2\Big].$$

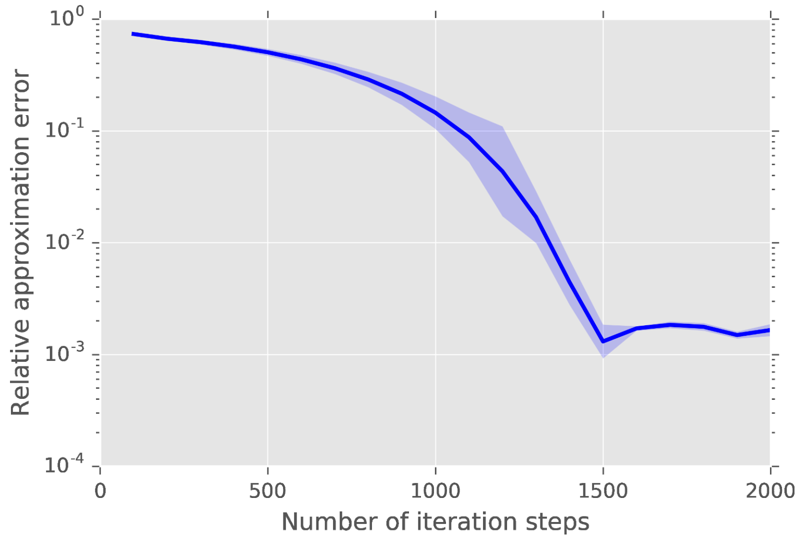
Open-source code on https://github.com/frankhan91/DeepBSDE

LQG (linear quadratic Gaussian) Example for d=100

Hamilton-Jacobi-Bellman equation:

$$\frac{\partial u}{\partial t} + \Delta u - \lambda \|\nabla u\|_2^2 = 0$$

$$u(t, \boldsymbol{x}) = -\frac{1}{\lambda} \ln\Big(\mathbb{E}\Big[\exp\Big(-\lambda g(\boldsymbol{x} + \sqrt{2} W_{T-t})\Big)\Big]\Big).$$



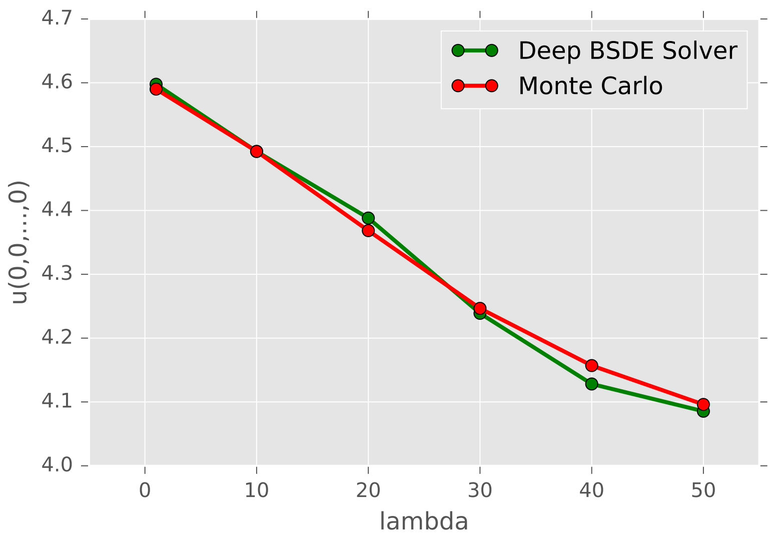


Figure: Left: Relative error of the deep BSDE method for $u(t{=}0, x{=}(0))$ when $\lambda = 1$, which achieves 0.17% in a runtime of 330 seconds. Right: $u(t{=}0, x{=}(0))$ for different λ.

Black-Scholes Equation with Default Risk for d=100

"exact" solution at $t = 0$ $\boldsymbol{x} = (100, \ldots, 100)$ computed by the multilevel Picard method.

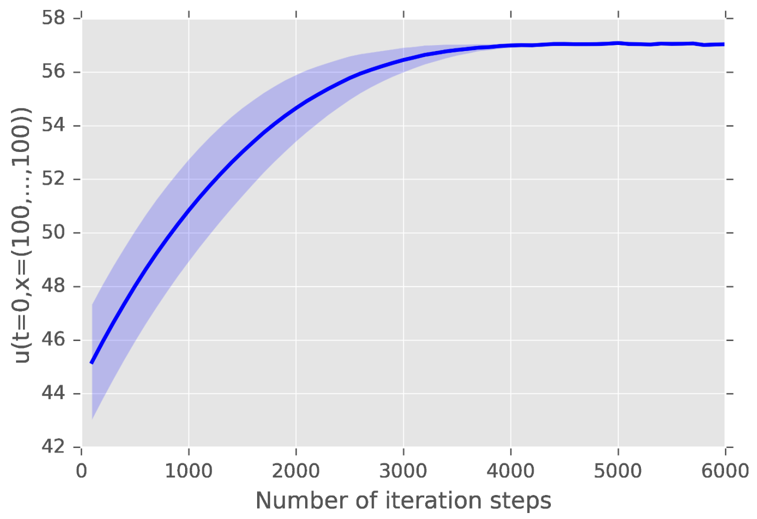


Figure: Approximation of $u(t{=}0, \boldsymbol{x}{=}(100, \ldots, 100))$ against number of iteration steps. The deep BSDE method achieves a relative error of size 0.46% in a runtime of 617 seconds.

Has been applied to the pricing of basket options and path-dependent options.

Outline

Basic setup

- we are given a micro-scale model (quantum mechanics, kinetic theory)
- we would like to construct an effective macro-scale model (classical molecular dynamics, hydrodynamics) with the help of machine learning

Requirements for the macro-scale model:

- obey physical constraints (symmetry, invariants, etc)
- reliable, i.e. "uniformly" accurate
  need the "optimal data set" (as small as possible yet representative enough), generated "on the fly"

The exploration-labeling-training procedure for “on-the-fly” learning

Zhang, Wang and E, J. Chemical Phys., 2017

Start out with no (macro-scale) model, no data; but with a micro-scale model.

Repeat the following steps:

1. exploration: explore the configuration (state) space, and decide which configurations/states need to be labeled.
2. labeling: compute the micro-scale solutions for the configurations that need to be labeled. This is our data set.
3. training: train the macro-scale model, and use it to help the exploration

Indicator: $\epsilon = \max_i \sqrt{\langle \| \boldsymbol{f}_i - \bar{\boldsymbol{f}}_i \|^2 \rangle}, \ \bar{\boldsymbol{f}}_i = \langle \boldsymbol{f}_i \rangle$

Traditional dilemma: accuracy vs cost.

$$E = E(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_N),$$

$$m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} = \boldsymbol{F}_i = -\nabla_{\boldsymbol{x}_i} E.$$

Two ways to calculate E and $\boldsymbol{F}$:

- Computing the inter-atomic forces on the fly using QM, e.g. the Car-Parrinello MD. Accurate but expensive:

$$E = \langle \Psi_0 | H_e^{KS} | \Psi_0 \rangle, \; \mu \ddot{\phi}_i = H_e^{KS} \phi_i + \sum_j \Lambda_{ij} \phi_j.$$

- Empirical potentials: efficient but unreliable. The Lennard-Jones potential:

$$V_{ij} = 4\epsilon[(\frac{\sigma}{r_{ij}})^{12} - (\frac{\sigma}{r_{ij}})^6], \; E = \frac{1}{2} \sum_{i \neq j} V_{ij}.$$

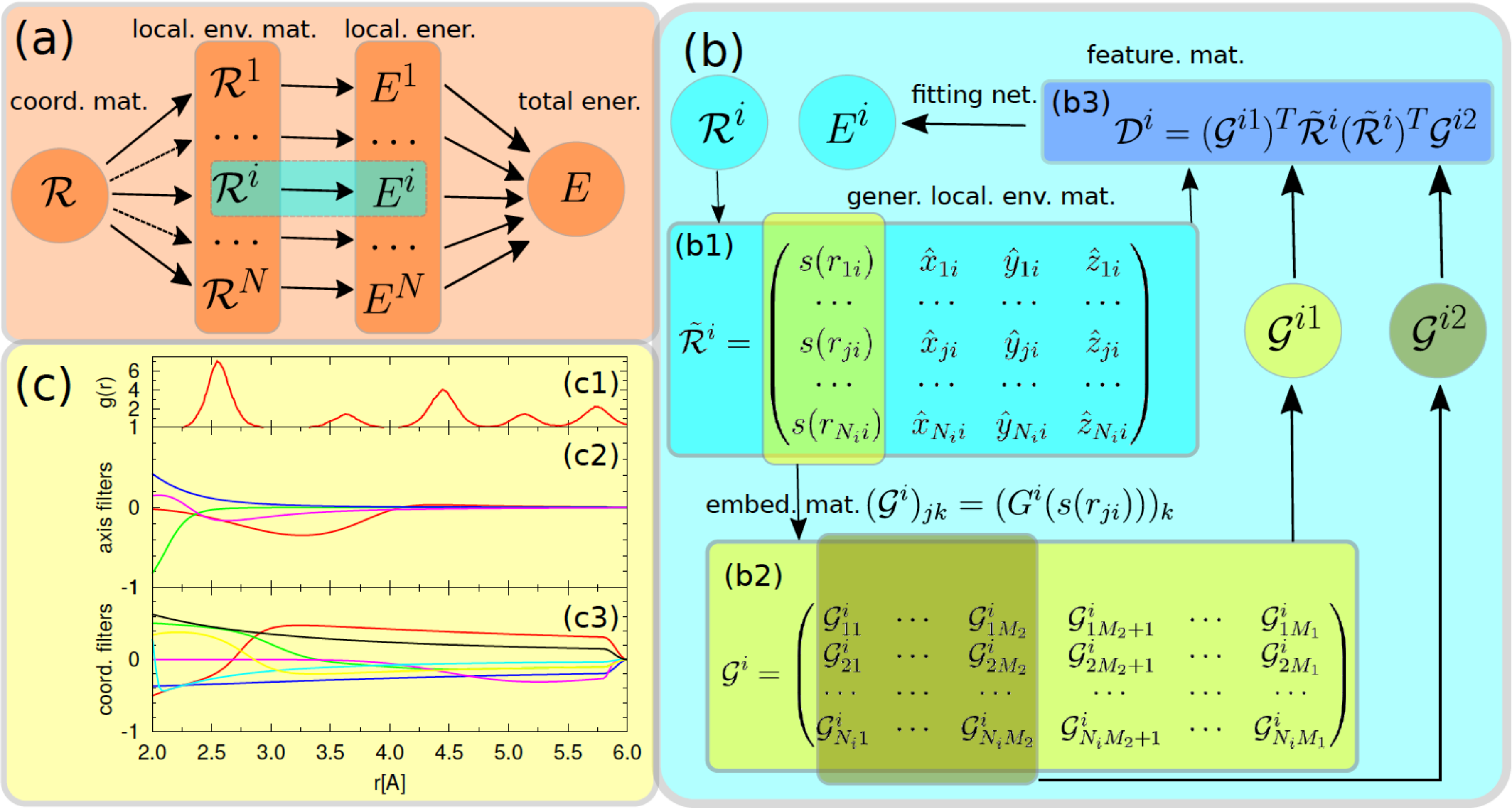
Integrating ML with molecular modeling

New paradigm:

- quantum mechanics model – micro-scale model for data generation
- machine learning – parametrize (represent) the model
- molecular dynamics – simulator

Deep potential

The whole sub-network consists of an encoding net $\mathcal{D}^i(\mathcal{R}^i)$ and a fitting net $E^i(\mathcal{D}^i)$.


(a)
local. env. mat.
local. ener.
coord. mat.
total ener.
$\mathcal{R}$
$\mathcal{R}^1$
$\mathcal{R}^i$
$\mathcal{R}^N$
E^1
E^i
E^N
E
(b)
feature. mat.
fitting net.
(b3) $\mathcal{D}^i = (\mathcal{G}^{i1})^T\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T\mathcal{G}^{i2}$
gener. local. env. mat.
(b1)
$\tilde{\mathcal{R}}^i = \begin{pmatrix} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i i}) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{pmatrix}$
$\mathcal{G}^{i1}$
$\mathcal{G}^{i2}$
embed. mat. $(\mathcal{G}^i)_{jk} = (G^i(s(r_{ji})))_k$
(b2)
$\mathcal{G}^i = \begin{pmatrix} \mathcal{G}^i_{11} & \cdots & \mathcal{G}^i_{1M_2} & \mathcal{G}^i_{1M_2+1} & \cdots & \mathcal{G}^i_{1M_1} \\ \mathcal{G}^i_{21} & \cdots & \mathcal{G}^i_{2M_2} & \mathcal{G}^i_{2M_2+1} & \cdots & \mathcal{G}^i_{2M_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{G}^i_{N_i1} & \cdots & \mathcal{G}^i_{N_iM_2} & \mathcal{G}^i_{N_iM_2+1} & \cdots & \mathcal{G}^i_{N_iM_1} \end{pmatrix}$
(c)
(c1)
(c2)
(c3)
g(r)
axis filters
coord. filters
r[A]
2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0

(Rotation: $\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T$, permutation: $(\mathcal{G}^{i1})^T\tilde{\mathcal{R}}^i$ and $(\tilde{\mathcal{R}}^i)^T\mathcal{G}^{i2}$.)

DeepPot-SE (arxiv: 1805.09003, NIPS 2018), see also Behler and Parrinello, PRL 2007.

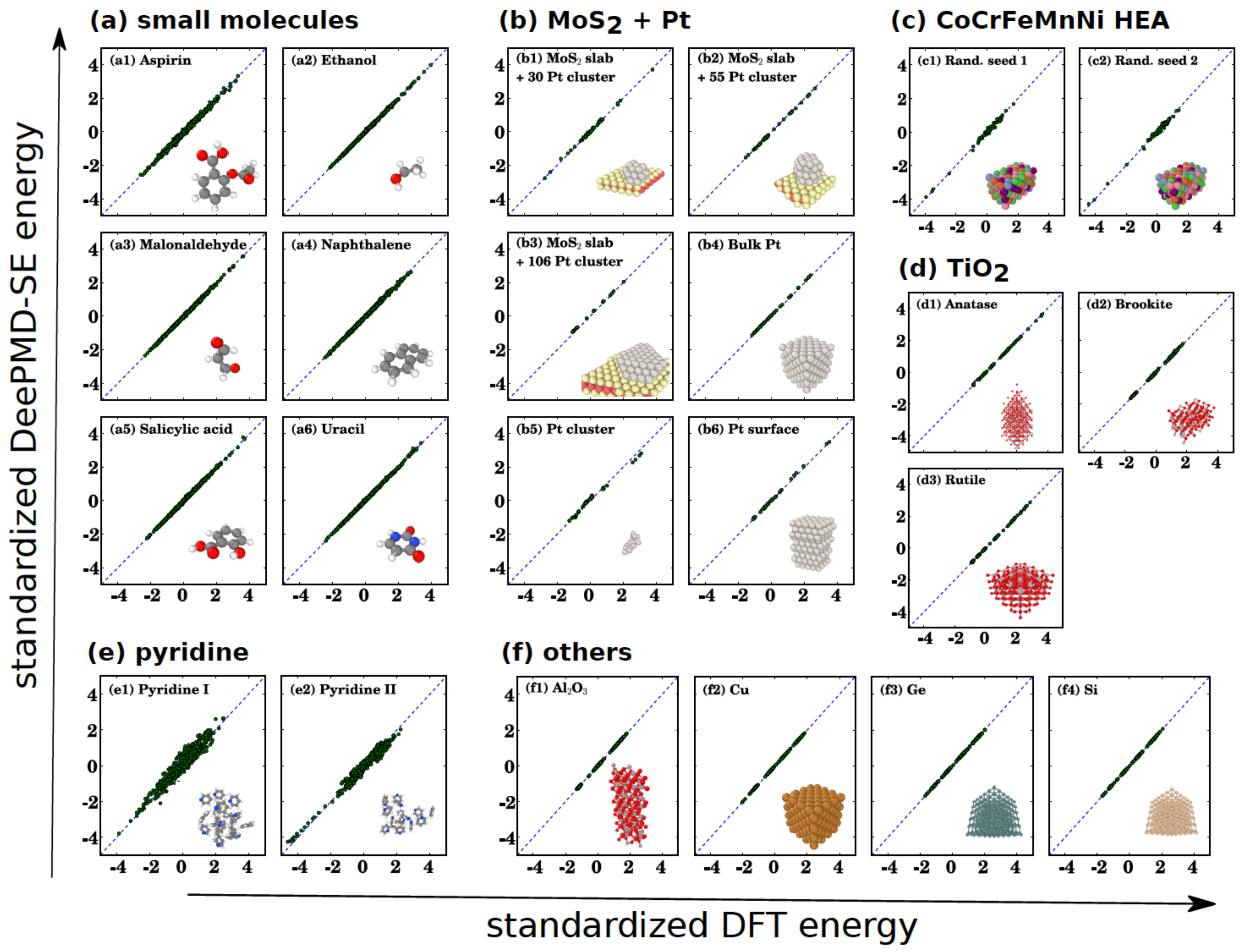
Only a small percentage of data needs to be labeled

Systems			Al		Mg		Al-Mg alloy	
Type	Lattice	#atom	#Confs	#Data	#Confs	#Data	#Confs	#Data
Bulk	FCC	32	15,174,000	1,326	15,174,000	860	39,266,460	7,313
	HCP	16	15,174,000	908	15,174,000	760	18,999,900	2,461
	Diamond	16	5,058,000	1,026	5,058,000	543	5,451,300	2,607
	SC	8	5,058,000	713	5,058,000	234	2,543,940	667
Surface	FCC (100)	12	3,270,960	728	3,270,960	251	62,203,680	1,131
	FCC (110)	16[a],20[b]	3,270,960	838	3,270,960	353	10,744,2720	2,435
	FCC (111)	12	3,270,960	544	3,270,960	230	62,203,680	1,160
	HCP (0001)	12	3,270,960	39	3,270,960	109	62,203,680	176
	HCP $(10\bar{1}0)$	12	3,270,960	74	3,270,960	167	62,203,680	203
	HCP $(11\bar{2}0)$	16[a],20[b]	3,270,960	293	3,270,960	182	107,442,720	501
sum			60,089,760	6,489	60,089,760	3,689	529,961,760	18,654

[a]Pure Al
[b]Mg and Al-Mg alloy

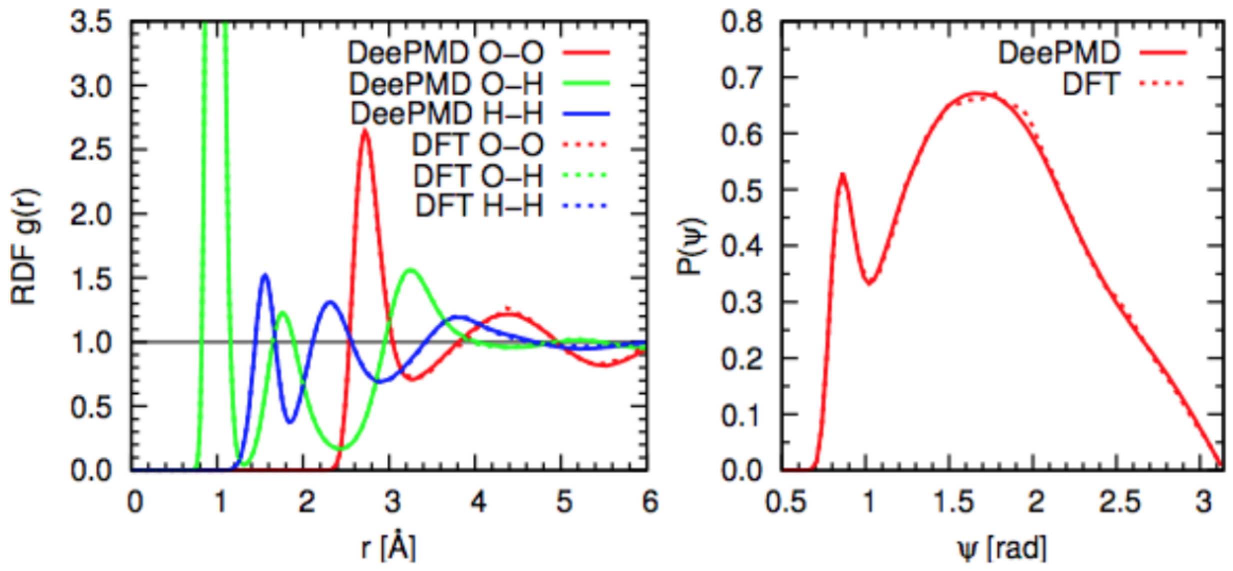
$\sim$0.005% configurations explored by DeePMD are selected for labeling.

Case 1: accuracy is comparable to the accuracy of the data

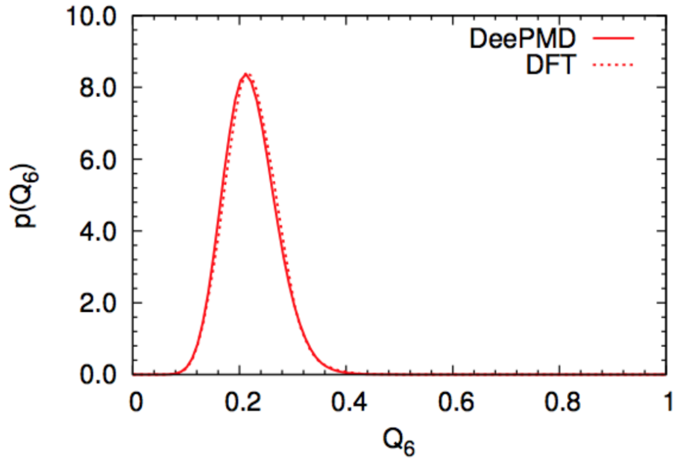


Case 2: structural information of DFT water

Radial and angular distribution function of liquid water (PI-AIMD):

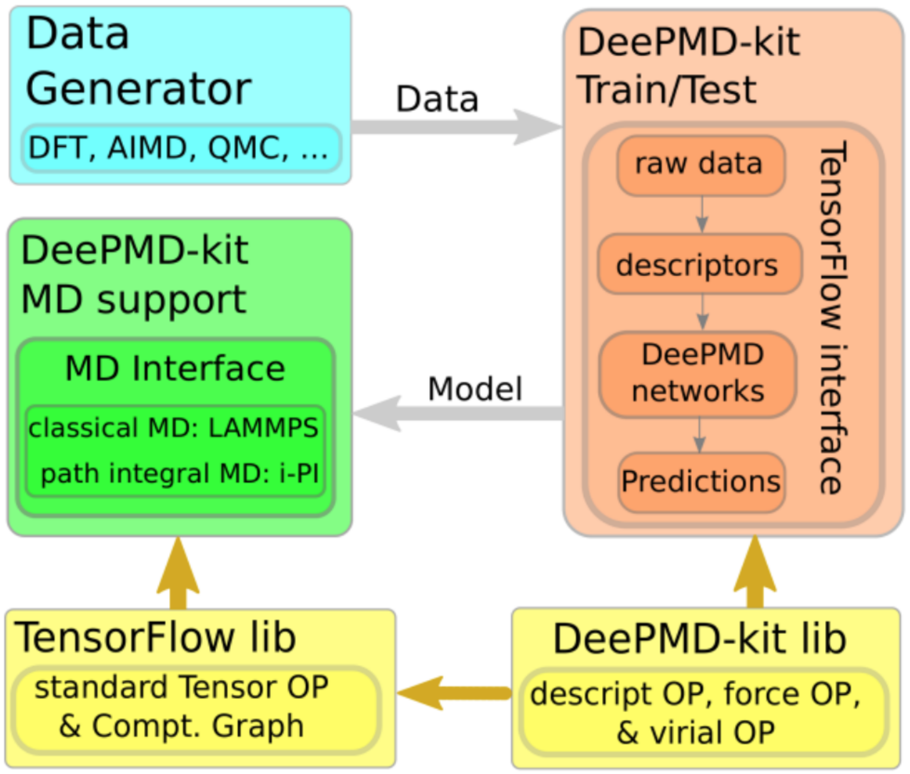


Distribution of the Steinhardt order parameter $\bar{Q}_6$:



DeePMD-kit

Towards realization of a general platform for ML-based PES modeling.



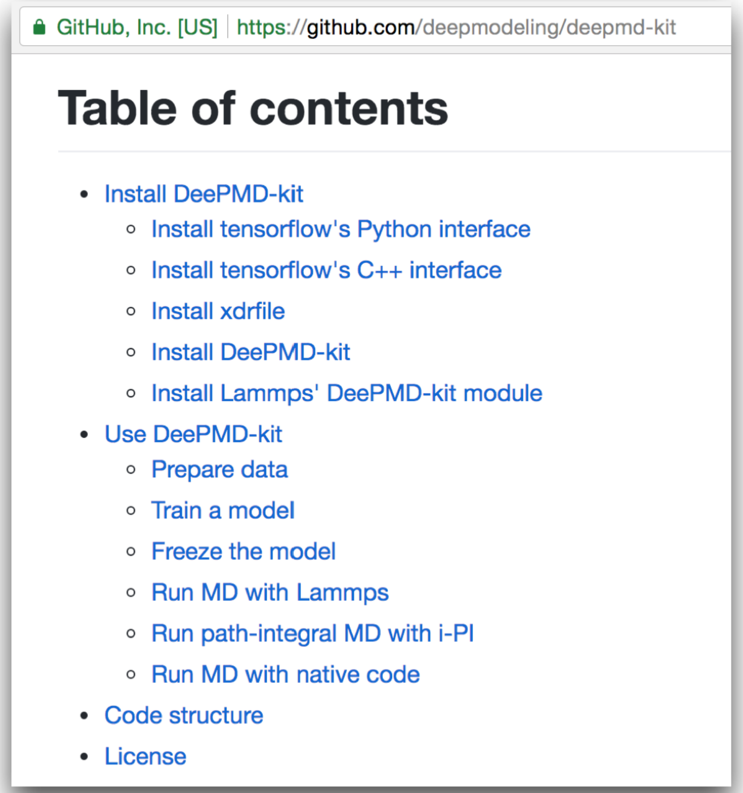

GitHub, Inc. [US] | https://github.com/deepmodeling/deepmd-kit

Table of contents

- Install DeePMD-kit
  - Install tensorflow's Python interface
  - Install tensorflow's C++ interface
  - Install xdrfile
  - Install DeePMD-kit
  - Install Lammps' DeePMD-kit module
- Use DeePMD-kit
  - Prepare data
  - Train a model
  - Freeze the model
  - Run MD with Lammps
  - Run path-integral MD with i-PI
  - Run MD with native code
- Code structure
- License

- interfacing state-of-the-art deep learning and MD packages: TensorFlow, LAMMPS, i-PI;
- parallelization: MPI/GPU support.

Comp. Phys. Comm., 2018: 0010-4655 (https://github.com/deepmodeling/deepmd-kit))

1. physical/chemical problems
   - understanding water (phase diagram of water, including reactive regions; phase transition: ice to water, ionic liquid to super-ionic ice; nuclear quantum effect: collective tunneling, isotope effect; reactive event: dissociation and recombination; water surface and water/TiO_2 interface; spectra: infra-red; Raman; X-ray Absorption; exotic properties: dielectric constant; density anomaly, etc.
   - physical understanding of different systems that require long-time large-scale simulation with high degrees of model fidelity (high-pressure iron: fractional defect; phase boundary; high-pressure hydrogen: exotic phases)
   - catalysis (Pt cluster on MoS_2 surface; CO molecules on gold surface, etc.)
2. materials science problems
   - battery materials (diffusion of lithium in LGePS, LSGeSiPS, etc.; diffusion of Se in Cu_2Se alloy)
   - high entropy/high temperature alloy (CoCrFeMnNi alloy; Ni-based alloy)
3. organic chemistry/bio problems
   - crystal structure prediction of molecular crystals;
   - protein-ligand interaction;
   - protein folding.

Concluding remarks

1. the integration of machine learning and PDE models is a truly very powerful tool
    - Deep potential based molecular dynamics (DPMD) now allows us to perform large scale MD simulations with quantum accuracy.
    - Machine learning-based hydrodynamic models allow us to develop moment closure for realistic kinetic theory models (Maxwell molecules).
    - We can now solve large classes of high dimensional PDEs quite routinely.
    - .....
2. from the viewpoint of machine learning, this provides a framework for "on the fly" learning.
3. high dimensional numerical analysis is a fruitful approach for studying the mathematical theory of machine learning.

MSML 2020

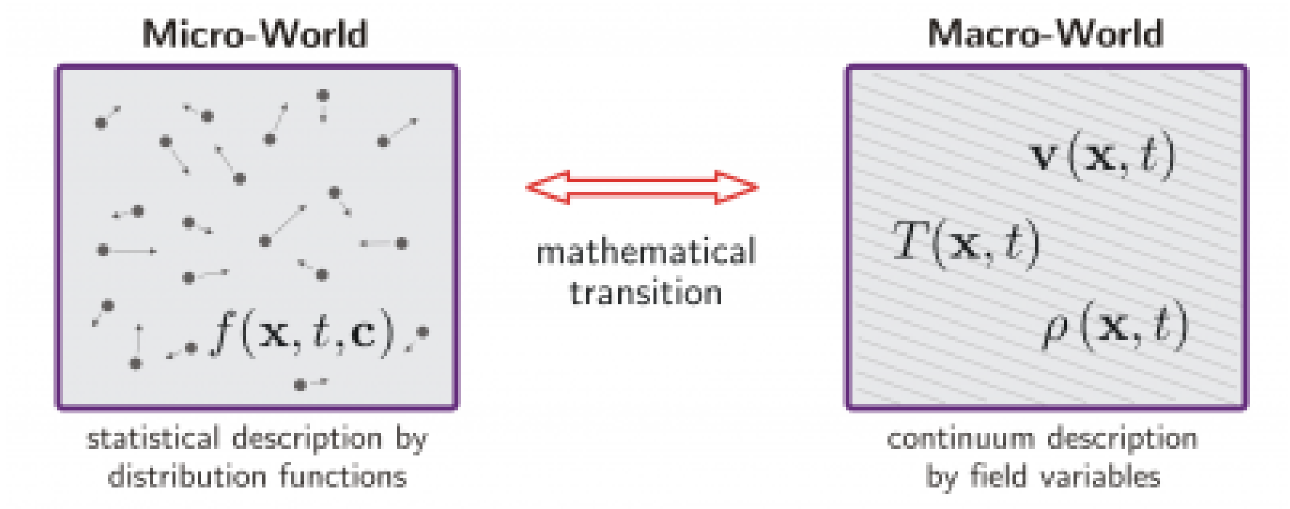
A new annual conference:
Mathematical and Scientific Machine Learning (MSML)

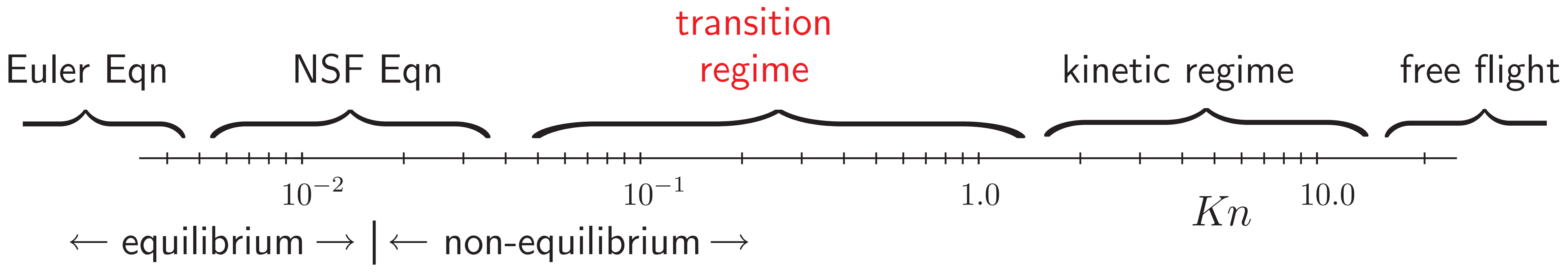
First meeting:

- Program Chairman: Jianfeng Lu (Duke) and Rachel Ward (Univ Texas/Austin)
- Time: July 15-17, 2020
- Location: Princeton
- Submission deadline: November 30, 2019
- website: http://msml-conf.org

Example 2: Modeling gas dynamics

$$Kn = \frac{\text{mean free path}}{\text{macroscopic length}}$$


Micro-World
$f(\mathbf{x}, t, \mathbf{c})$
statistical description by distribution functions
mathematical transition
Macro-World
$\mathbf{v}(\mathbf{x}, t)$
$T(\mathbf{x}, t)$
$\rho(\mathbf{x}, t)$
continuum description by field variables



Boltzmann Equation

One-particle density function $f(\boldsymbol{x}, \boldsymbol{v}, t)$

$$\partial_t f + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = \frac{1}{\varepsilon} Q(f), \quad \boldsymbol{v} \in \mathbb{R}^3, \quad \boldsymbol{x} \in \Omega \subset \mathbb{R}^3,$$

$\varepsilon =$ Knudsen number and Q is the collision operator.

Macroscopic state variables: ρ, $\boldsymbol{u}$ and T (density, bulk velocity and temperature)

$$\rho = \int f \, \mathrm{d}\boldsymbol{v}, \quad \boldsymbol{u} = \frac{1}{\rho} \int f \boldsymbol{v} \, \mathrm{d}\boldsymbol{v}, \quad T = \frac{1}{3\rho} \int f |\boldsymbol{v} - \boldsymbol{u}|^2 \, \mathrm{d}\boldsymbol{v}.$$

When $\varepsilon \ll 1$, Boltzmann can be approximated by Euler:

$$\partial_t \boldsymbol{U} + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{F}(\boldsymbol{U}) = 0,$$

with $p = \rho T$, $E = \frac{1}{2}\rho \boldsymbol{u}^2 + \frac{3}{2}\rho T$,

$$\boldsymbol{U} = (\rho, \rho \boldsymbol{u}, E)^T$$

$$\boldsymbol{F}(\boldsymbol{U}) = (\rho \boldsymbol{u}, \rho \boldsymbol{u} \otimes \boldsymbol{u} + pI, (E + p)\boldsymbol{u})^T$$

Machine learning-based moment method

Objective: construct an uniformly accurate (generalized) moment model using machine learning.

Step 1: Learn the Moments through Autoencoder
Find an encoder Ψ that maps $f(\cdot, \boldsymbol{v})$ to generalized moments $\boldsymbol{W} \in \mathbb{R}^M$ and a decoder Φ that recovers the original f from $\boldsymbol{U}, \boldsymbol{W}$

$$\boldsymbol{W} = \Psi(f) = \int \mathbf{w} f \, \mathrm{d}\boldsymbol{v}, \quad \Phi(\boldsymbol{U}, \boldsymbol{W})(\boldsymbol{v}) = h(\boldsymbol{v}; \boldsymbol{U}, \boldsymbol{W}).$$

The goal is essentially to find optimal $\mathbf{w}$ and h parametrized by neural networks through minimizing

$$\mathop{\mathbb{E}}_{f \sim \mathcal{D}} \|f - \Phi(\Psi(f))\|^2 + \lambda_\eta (\eta(f) - h_\eta(\boldsymbol{U}, \boldsymbol{W}))^2.$$

$\eta(f)$ denotes entropy.

Step 2: Learn the Fluxes and Source Terms in the PDE

Recall the general conservative form of the moment system

$$\begin{cases} \partial_t \boldsymbol{U} + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{F}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon) = 0, \\ \partial_t \boldsymbol{W} + \nabla_{\boldsymbol{x}} \cdot \boldsymbol{G}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon) = \mathbb{R}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon). \end{cases}$$

Rewrite it into (variance reduction)

$$\begin{cases} \partial_t \boldsymbol{U} + \nabla_{\boldsymbol{x}} \cdot [\boldsymbol{F}_0(\boldsymbol{U}) + \tilde{\boldsymbol{F}}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon)] = 0, \\ \partial_t \boldsymbol{W} + \nabla_{\boldsymbol{x}} \cdot [\boldsymbol{G}_0(\boldsymbol{U}) + \tilde{\boldsymbol{G}}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon)] = \mathbb{R}(\boldsymbol{U}, \boldsymbol{W}; \varepsilon). \end{cases}$$

$\boldsymbol{F}_0(\boldsymbol{U}), \boldsymbol{G}_0(\boldsymbol{U})$ are the fluxes of the moments $\boldsymbol{U}, \boldsymbol{W}$ under the Maxwellian distribution.

Our goal is to obtain ML models for $\tilde{\boldsymbol{F}}, \tilde{\boldsymbol{G}}, \mathbb{R}$ from the original kinetic equation.

Issues: (1) physical symmetries (e.g. Galilean invariance); (2) data generation (active learning algorithm); (3) locality vs. non-locality of the model

Numerical results for transitional flows: Maxwell molecules

ε varies from 10^{-3} to 10 in the domain. $\boldsymbol{W} \in \mathbb{R}^9$.

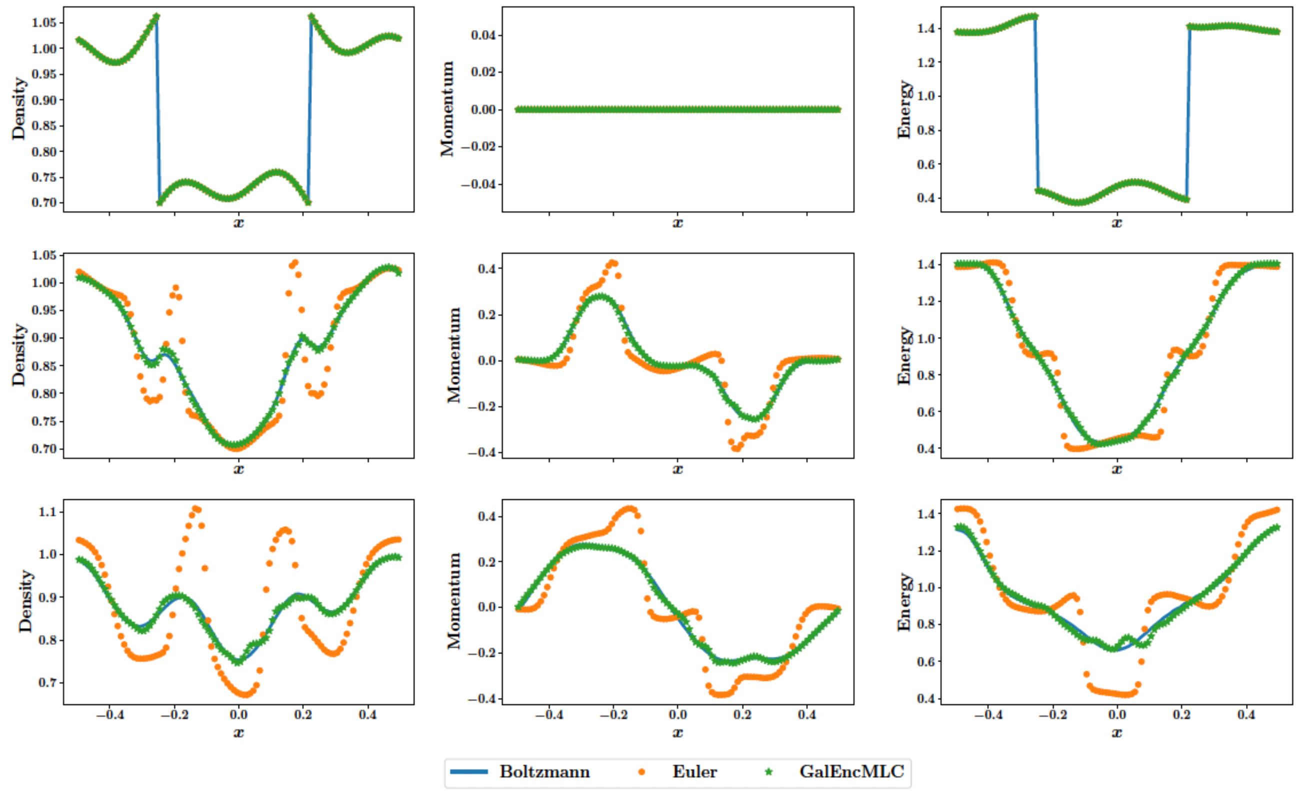
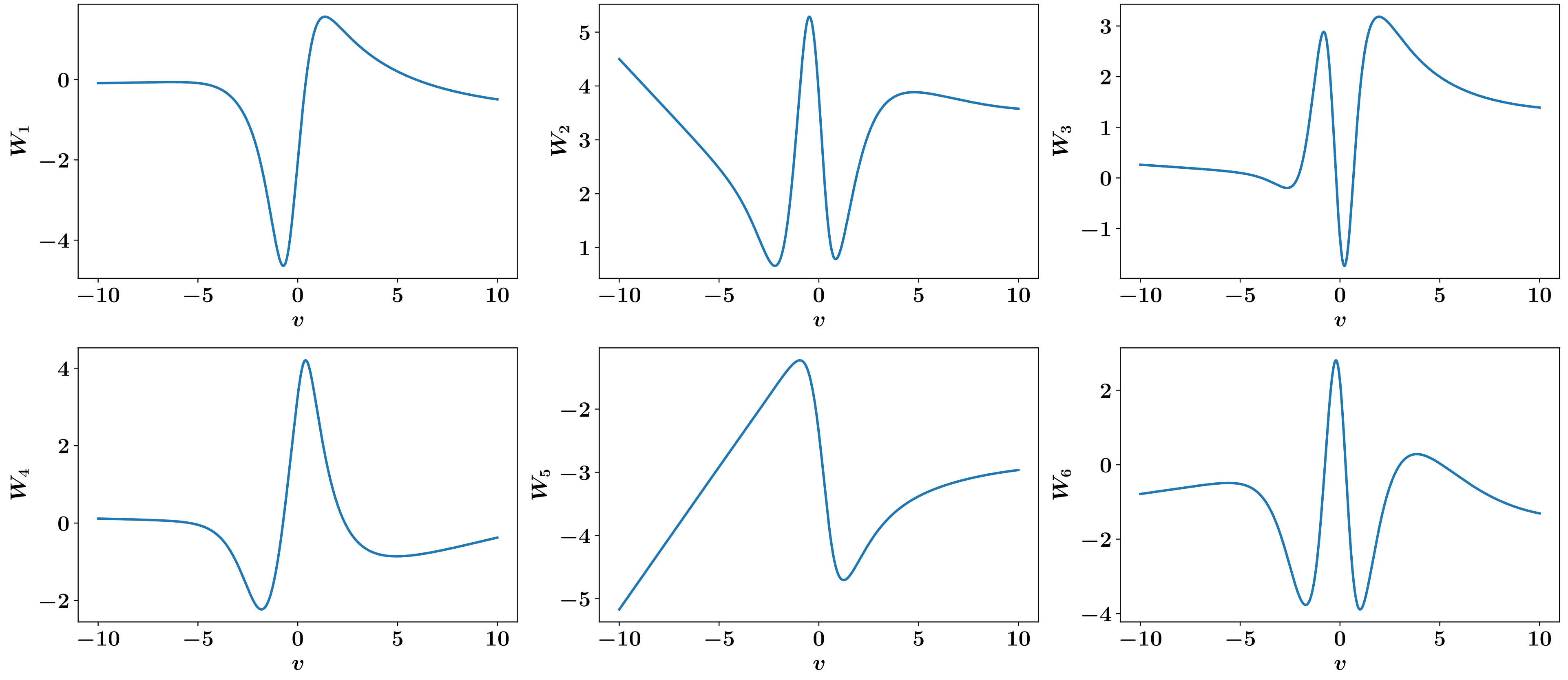


Figure: Profiles of $\rho, \rho\boldsymbol{u}, E$ (from left to right) at $t = 0, 0.05, 0.1$ (from top to bottom)

Numerical results

Learned functions $\mathbf{w}(v)$ as generalized moments



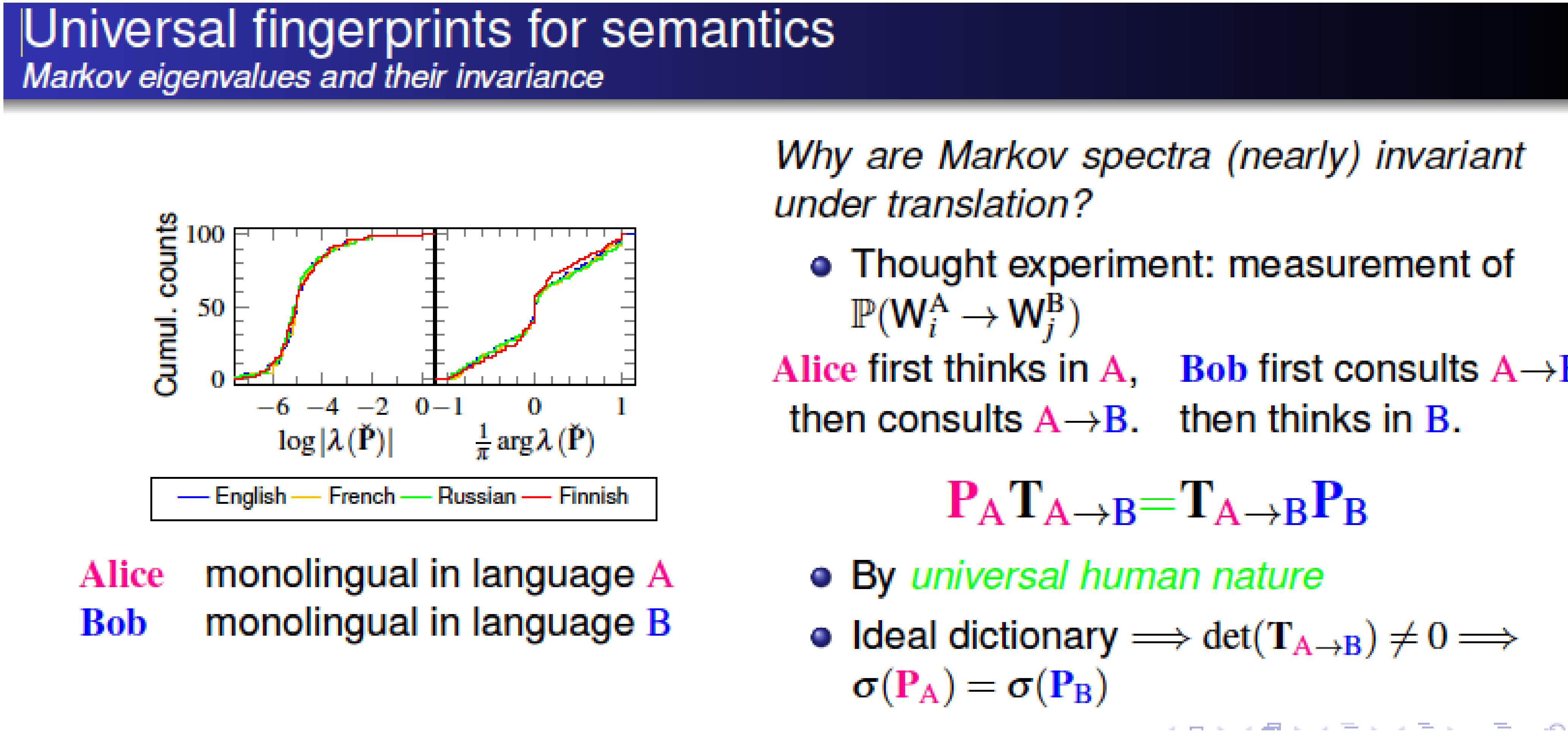
Outline

Scales in natural languages

Diversity and universality of human languages at different scales:

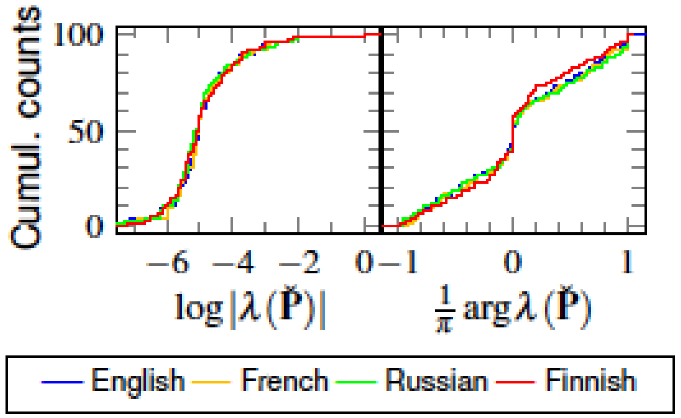
- words
- sentences
- inter-sentences

What is semantics? Semantics are invariants under translation



Universal fingerprints for semantics

Markov eigenvalues and their invariance



Alice	monolingual in language A
Bob	monolingual in language B

Why are Markov spectra (nearly) invariant under translation?

- Thought experiment: measurement of $\mathbb{P}(W_i^A \to W_j^B)$

Alice first thinks in A, then consults A→B.	Bob first consults A→ then thinks in B.

$$\mathbf{P}_A\mathbf{T}_{A\to B}=\mathbf{T}_{A\to B}\mathbf{P}_B$$

- By *universal human nature*
- Ideal dictionary $\Longrightarrow \det(\mathbf{T}_{A\to B}) \neq 0 \Longrightarrow \sigma(\mathbf{P}_A) = \sigma(\mathbf{P}_B)$

Summary and Conclusion

We are at the verge of a new scientific revolution that will impact mathematics and applied mathematics in fundamental ways.

- Integrating machine learning (Keplerian paradigm) with first principle based physical modeling (Newtonian paradigm) opens up a new (and powerful) paradigm for scientific research.
  Applied mathematics is the most natural platform for this integration.

- Theoretical foundation of machine learning = high dimensional numerical analysis

MSML 2020

A new annual conference:
Mathematical and Scientific Machine Learning (MSML)

First meeting:

- Program Chairman: Jianfeng Lu (Duke) and Rachel Ward (Univ Texas/Austin)
- Time: July 15-17, 2020
- Location: Princeton
- Submission deadline: November 30, 2019
- website: http://msml-conf.org