A Mathematical Perspective on Machine Learning

Weinan E

Center for Machine Learning Research and
School of Mathematical Sciences

Peking University

Machine learning has changed the way we do AI.

Recognizing images better than average humans

- Given a set of "labeled" images ("label" = the content of the image), find an algorithm that can automatically tell us the content of similar images.

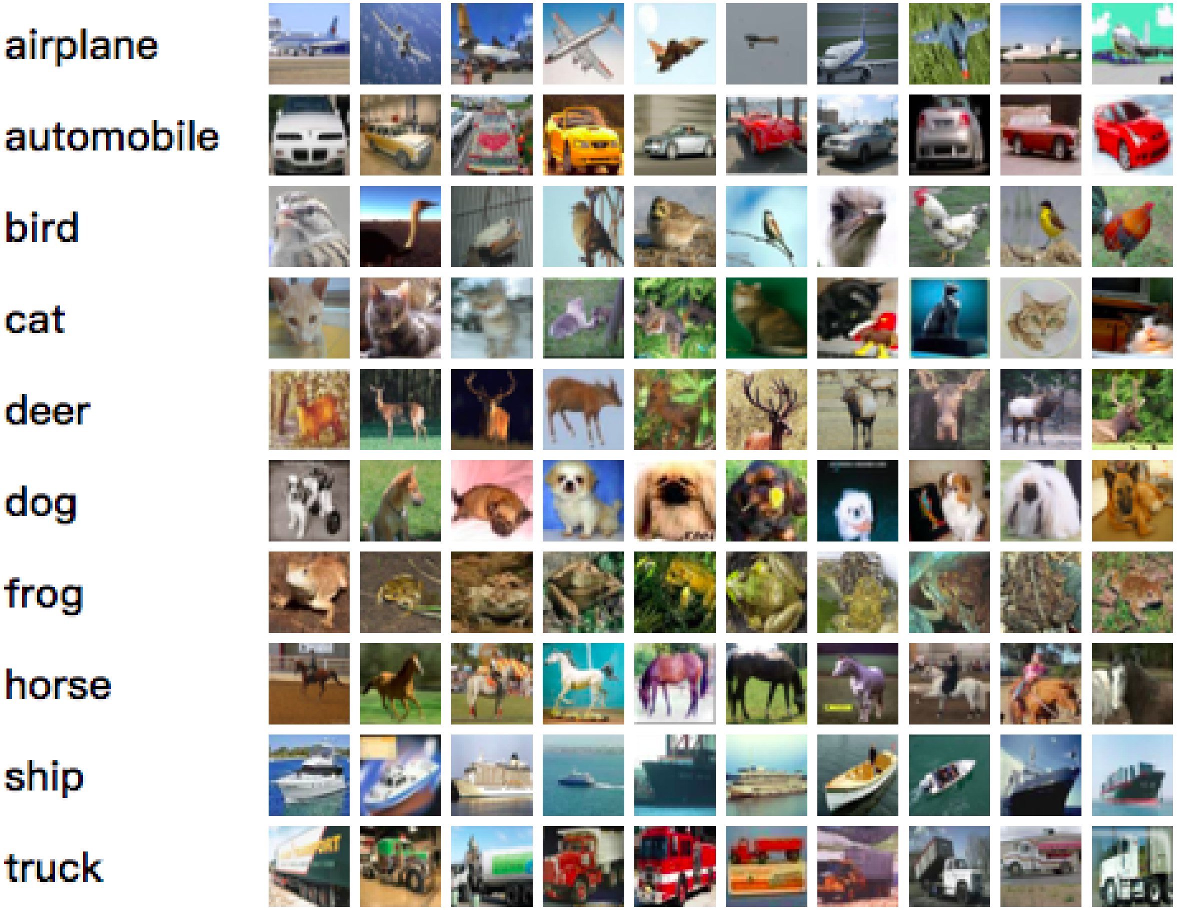


Figure: The Cifar-10 dataset: Each image is assigned a label from the 10 different categories

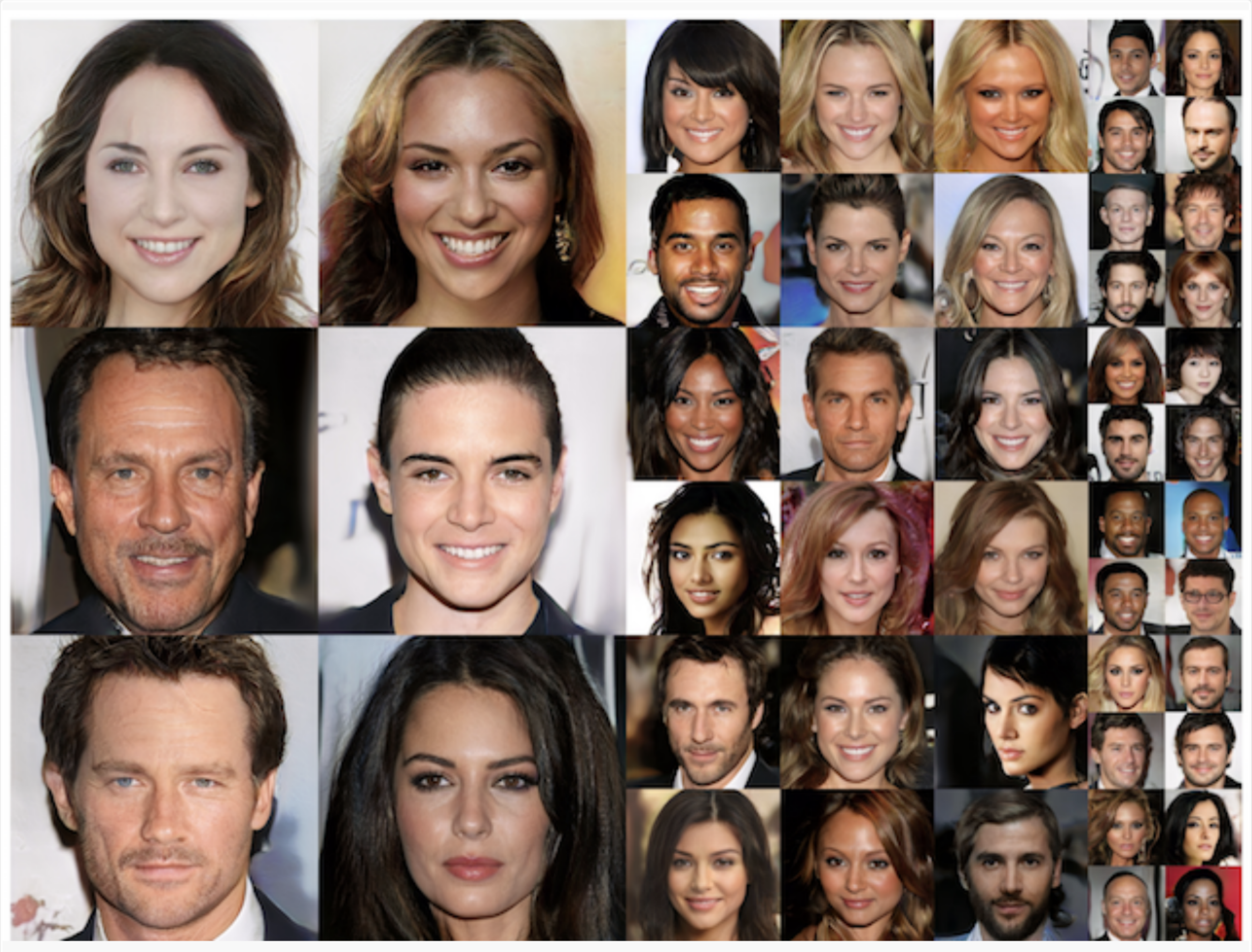
https://www.cs.toronto.edu/~kriz/cifar.html

AlphaGo: Playing Go game better than the best humans!

It was done purely by machine learning!

`https://www.bbc.com/news/technology-35761246`

Generating non-existing data: Pictures of FAKE human faces



https://arxiv.org/pdf/1710.10196v3.pdf

In essence, what's done in all these examples is to solve some standard mathematical problem.

Image classification:

We are interested in the function

$$f^* : \text{image} \rightarrow \text{ its } \ \text{ content (category)}$$

We know the values of f^* on a finite sample (the labeled data). The goal is to find accurate approximation of f^*.

Supervised learning:

Approximating a target function f^* using a finite training set

$$S = \{(\boldsymbol{x}_j, y_j = f^*(\boldsymbol{x}_j)), j \in [n] = \{1, 2, \cdots, n\}\}$$

Generating pictures of fake human faces:

Approximating and sampling an unknown probability distribution.

- Random variable: pictures of human faces
- We don't know its probability distribution
- We do have a finite sample: pictures of real human faces
- We can approximate the unknown probability distribution and produce new samples
- These new samples are pictures of fake human faces.

Unsupervised learning:

Approximating the underlying probability distribution using finite samples.

Playing Go games:

Solving the Bellman equation in dynamic programming.

- Given the strategy of the opponent, the dynamics of the Go game is a dynamic programming problem
- The optimal strategy satisfies a Bellman equation

Reinforcement learning:

Finding the optimal strategy in a *Markov decision process*.

Wait a minute, we have been solving these kinds of problems in (computational) mathematics for a long time!

After all,

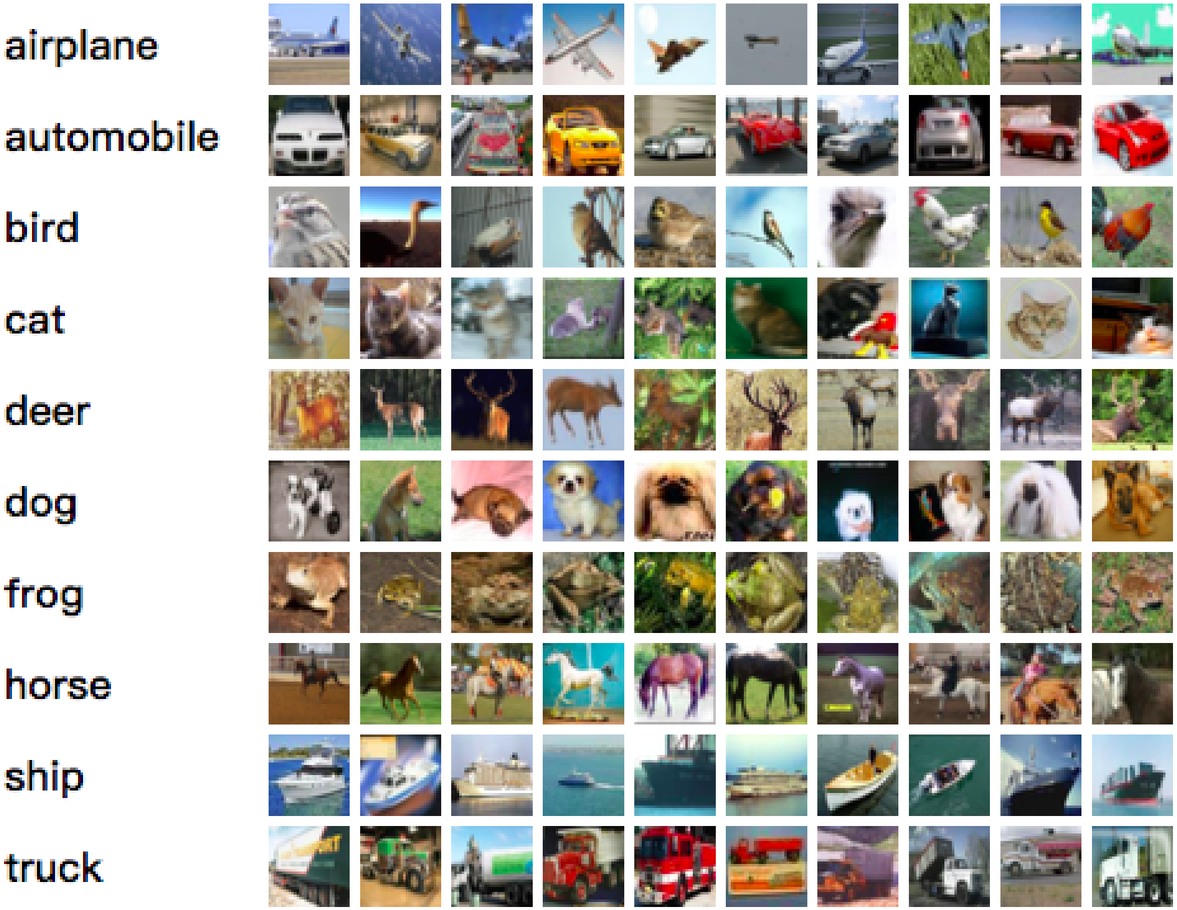
- approximating functions
- approximating and sampling probability distributions
- solving differential and difference equations

are all classical problems in numerical analysis.

So what is really the difference?

Dimensionality!

Dimensionality of the CIFAR-10 problem



Input dimension:

$$d = 32 \times 32 \times 3 = 3072$$

Classical approximation theory

Approximate a function using piecewise linear functions over a mesh of size h:

- d = dimensionality of the problem
- m = the total number of free parameters in the model

$$h \sim m^{-1/d}$$

$$|f^* - f_m| \sim h^2 |\nabla^2 f^*| \sim m^{-2/d} |\nabla^2 f^*|$$

To reduce the error by a factor of 10, we need to increase m by a factor of $10^{d/2}$.

Curse of dimensionality (CoD): As d grows, computational cost grows exponentially fast.

True for all classical algorithms, e.g. approximating functions using polynomials, or wavelets.

Apparently, deep neural networks can do much better in high dimension.

Main content

- **Understanding the magic**: mathematical theory for supervised learning.
- **AI for Science**: application of machine learning to science and scientific computing.
- Skip: “better” machine learning models motivated by ODEs and PDEs.

Neural networks are a special class of functions

Two-layer neural networks:

$$f(\boldsymbol{x}) = \sum_k a_k \sigma(\boldsymbol{w}_k \cdot \boldsymbol{x} + c_k)$$

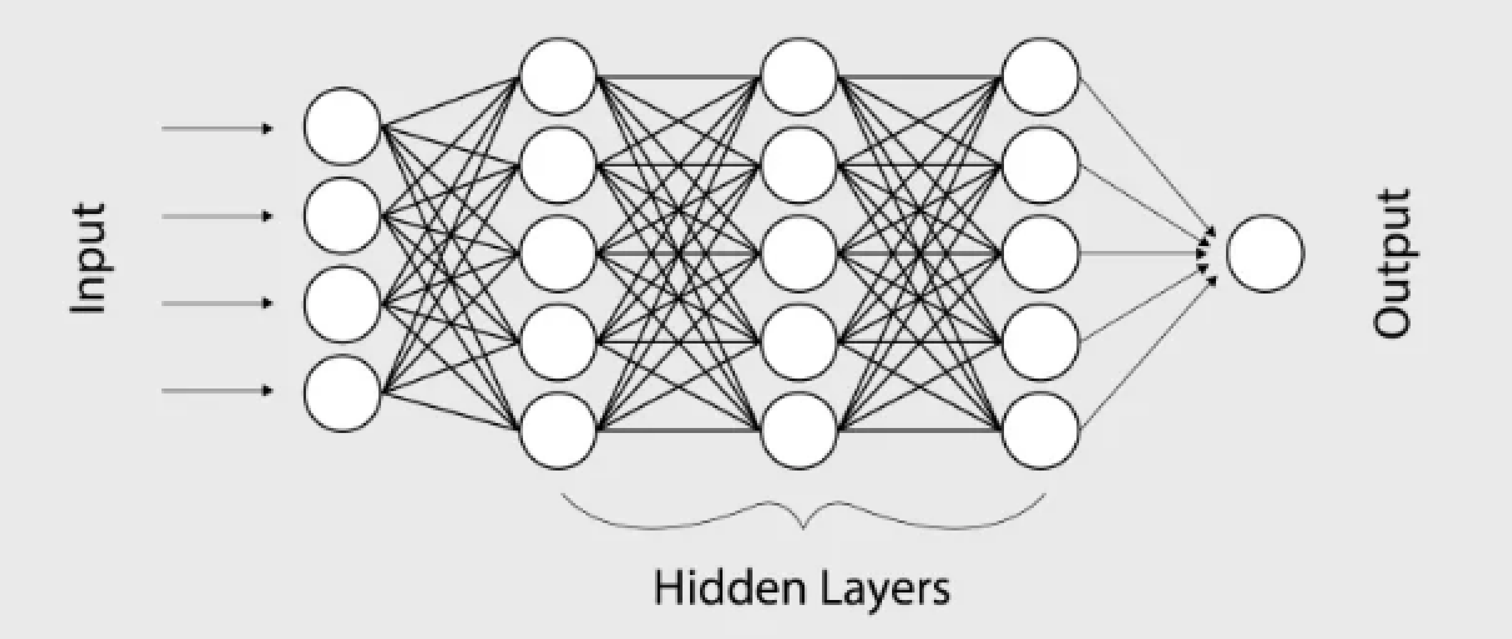
Two set of parameters: $\{a_k\}, \{(\boldsymbol{w}_k, c_k)\}$

- $\sigma(z) = \max(z, 0)$, the ReLU (rectified linear units) function.
- $\sigma(z) = (1 + e^{-z})^{-1}$, the "sigmoid function".

Remark about notation: We will neglect the bias term c_k in the notation.

Deep neural networks: compositions of those

- linear transformations
- scalar nonlinear function
- compositions



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

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

Supervised learning

Knowing the values of f^* on a finite **training dataset**

$$S = \{(\boldsymbol{x}_j, y_j = f^*(\boldsymbol{x}_j)), j \in [n] = \{1, 2, \cdots, n\}\}$$

find accurate approximations of the **target function** f^*.

Our main objective is to:

Minimize the **testing error** ("population risk" or "generalization error"):

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 = \int_X (f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 d\mu$$

where μ is the distribution of $\boldsymbol{x}$ (say on a domain $X \subset \mathbb{R}^d$).

To be specific, we will take $X = [0, 1]^d$.

Standard procedure for supervised learning

1. choose a hypothesis space (a set of trial functions) $\mathcal{H}_m$ ($m \sim \dim(\mathcal{H}_m)$)
   - (piecewise) polynomials, wavelets, ...
   - neural network models
2. formulate an optimization problem, i.e. choose a loss function
   - "empirical risk" (to fit the data)

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

   - add regularization terms
3. training: solve the optimization problem
   - gradient descent (GD):

$$\theta_{k+1} = \theta_k - \eta\nabla\hat{\mathcal{R}}_n(\theta_k) = \theta_k - \eta\frac{1}{n}\sum_{j=1}^{n}\nabla\ell(\theta_k,\boldsymbol{x}_j)$$

   - stochastic gradient descent (SGD):

$$\theta_{k+1} = \theta_k - \eta\nabla\ell(\theta_k,\boldsymbol{x}_{j_k})$$

where j_k is randomly chosen from $\{1, 2, \cdots, n\}$ in some way.

The three main components of the error

The total error: $f^* - \hat{f}$, where $\hat{f}$ = the output of the ML model.

Define:

- $f_m = \text{argmin}_{f \in \mathcal{H}_m} \mathcal{R}(f)$ = best approximation to f^* in $\mathcal{H}_m$
- $\tilde{f}_{n,m} =$ "best approximation to f^* in $\mathcal{H}_m$, using only the dataset S"

Decomposition of the error:

$$f^* - \hat{f} = \underbrace{f^* - f_m}_{\text{appr.}} + \underbrace{f_m - \tilde{f}_{n,m}}_{\text{estim.}} + \underbrace{\tilde{f}_{n,m} - \hat{f}}_{\text{optim.}}$$

- $f^* - f_m$ = *approximation error*, due entirely to the choice of the hypothesis space
- $f_m - \tilde{f}_{n,m}$ = *estimation error* — additional error due to the fact that we only have a finite dataset
- $\tilde{f}_{n,m} - \hat{f}$ = *optimization error* — additional error caused by training

Approximation Error

Benchmark: High dimensional integration

$$I(g) = \int_X g(\boldsymbol{x}) d\mu = \mathbb{E}_{\boldsymbol{x}\sim\mu} g, \quad I_m(g) = \frac{1}{m}\sum_{j=1}^{m} g(\boldsymbol{x}_j)$$

Grid-based quadrature rules (α is some fixed number):

$$I(g) - I_m(g) \sim \frac{C(g)}{m^{\alpha/d}}$$

Curse of dimensionality (CoD)!

Monte Carlo: $\{\boldsymbol{x}_j, j \in [m]\}$ are i.i.d samples of μ

$$\mathbb{E}(I(g) - I_m(g))^2 = \frac{\text{var}(g)}{m}, \quad \text{var}(g) = \mathbb{E}g^2 - (\mathbb{E}g)^2$$

Implications to function approximation

Representation of functions using transforms:

Representing functions using Fourier transform:

$$f^*(\boldsymbol{x}) = \int_{\mathbb{R}^d} \hat{f}(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})} d\boldsymbol{\omega}.$$

Approximate using discrete Fourier transform on uniform grids:

$$f_m(\boldsymbol{x}) = \frac{1}{m} \sum_{j=1}^{m} \hat{f}(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j,\boldsymbol{x})}$$

The error suffers from CoD:

$$f^* - f_m \sim h^\alpha \sim m^{-\alpha/d}$$

“New” approach: Let π be a probability distribution

$$f^*(\boldsymbol{x}) = \int_{\mathbb{R}^d} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega}, \boldsymbol{x})} \pi(d\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\omega} \sim \pi} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega}, \boldsymbol{x})}$$

Let $\{\boldsymbol{\omega}_j\}$ be an **i.i.d. sample of** π, $f_m(\boldsymbol{x}) = \frac{1}{m}\sum_{j=1}^m a(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j, \boldsymbol{x})}$,

$$\mathbb{E}|f^*(\boldsymbol{x}) - f_m(\boldsymbol{x})|^2 = \frac{\mathsf{var}(f)}{m}$$

Note: $f_m(\boldsymbol{x}) = \frac{1}{m}\sum_{j=1}^m a_j \sigma(\boldsymbol{\omega}_j^T \boldsymbol{x})$ = two-layer neural network with $\sigma(z) = e^{iz}$.

Conclusion:
Functions of the this type (i.e. can be expressed as this kind of expectation) can be approximated by two-layer neural networks with a dimension-independent error rate.

Approximation theory for the *random feature model*

- Let $\phi(\cdot;\boldsymbol{w})$ be a <u>feature function</u> parametrized by $\boldsymbol{w} \in \Omega$,
  e.g. $\phi(\boldsymbol{x};\boldsymbol{w}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$.
  We will assume that ϕ is continuous and Ω is compact.
- Let π_0 be a fixed distribution for the random variable $\boldsymbol{w}$.
- Let $\{\boldsymbol{w}_j^0\}_{j=1}^m$ be a set of i.i.d samples drawn from π_0.

The <u>random feature model</u> (RFM) associated with the features $\{\phi(\cdot;\boldsymbol{w}_j^0)\}$ is given by

$$f_m(\boldsymbol{x};\boldsymbol{a}) = \frac{1}{m}\sum_{j=1}^{m} a_j\phi(\boldsymbol{x};\boldsymbol{w}_j^0).$$

What spaces of functions are “well approximated” (say with the same convergence rate as in Monte Carlo) **by the random feature model?**

- In classical approximation theory, these are a the Sobolev or Besov spaces: They are characterized by the convergence behavior for some specific approximation schemes.
- Direct and inverse approximation theorems.

Define the kernel function associated with the random feature model:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w} \sim \pi_0}[\phi(\boldsymbol{x}; \boldsymbol{w})\phi(\boldsymbol{x}'; \boldsymbol{w})]$$

Let $\mathcal{H}_k$ be the reproducing kernel Hilbert space (RKHS) induced by the kernel k.

Probabilistic characterization:
$f \in \mathcal{H}_k$ if and only if there exists $a(\cdot) \in L^2(\pi_0)$ such that

$$f(\boldsymbol{x}) = \int a(\boldsymbol{w})\phi(\boldsymbol{x}; \boldsymbol{w}) d\pi_0(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{w} \sim \pi_0} a(\boldsymbol{w})\phi(\boldsymbol{x}; \boldsymbol{w})$$

and

$$\|f\|_{\mathcal{H}_k}^2 = \int a^2(\boldsymbol{w}) d\pi_0(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{w} \sim \pi_0} a^2(\boldsymbol{w})$$

Direct approximation theorem

Theorem

For any $\delta \in (0,1)$, *with probability* $1-\delta$ *over the samples* $\{\boldsymbol{w}_j^0\}_{j=1}^m$, *we have for any* $f^* \in \mathcal{H}_k$

$$\inf_{a_1,\dots,a_m} \|f^* - \frac{1}{m}\sum_{j=1}^m a_j \phi(\cdot;\boldsymbol{w}_j^0)\|_{L^2(\mu)} \lesssim \frac{\|f^*\|_{\mathcal{H}_k}}{\sqrt{m}}(1+\sqrt{\log(1/\delta)}).$$

Proof uses:

- Duality
- Concentration inequality.

  Example: Hoeffding inequality

  Let $X_1, X_2, \cdots$ be i.i.d random variables with values in $[a,b]$, $S_n = \frac{X_1+\cdots+X_n}{n}$. Then

$$\mathbb{P}(|S_n - \mathbb{E}S_n| \geq t) \leq 2\exp\left(-\frac{nt^2}{(b-a)^2}\right)$$

Inverse approximation theorem

Theorem

Let $(\boldsymbol{w}_j^0)_{j=0}^{\infty}$ be a sequence of i.i.d. random samples drawn from π_0. Let f^* be a continuous function on X. Assume that there exist a constant C and a sequence $\{(a_{j,m}), m \in \mathbb{N}^+, j \in [m]\}$ such that $\sup_{j,m} |a_{j,m}| \leq C$ and

$$\lim_{m\to\infty} \frac{1}{m} \sum_{j=1}^{m} a_{j,m}\phi(\boldsymbol{x}; \boldsymbol{w}_j^0) = f^*(\boldsymbol{x}),$$

for all $\boldsymbol{x} \in X$. Then with probability 1, $f^* \in \mathcal{H}_k$, and there exists a function $a^* \in L^\infty(\pi)$ such that

$$f^*(\boldsymbol{x}) = \int_\Omega a^*(\boldsymbol{w})\phi(\boldsymbol{x}; \boldsymbol{w})d\pi_0(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{w}\sim\pi_0} a^*(\boldsymbol{w})\phi(\boldsymbol{x}; \boldsymbol{w})$$

Moreover, $\|a^*\|_\infty \leq C$.

Conclusion: Roughly speaking, functions that are well approximated by the random feature models are functions which admit the integral representation above.
$\mathcal{H}_k$ is about the right function space associated with the RFM.

Approximation theory for *two-layer neural networks*

Consider "scaled" two-layer neural networks:

$$f_m(\boldsymbol{x};\theta) = \frac{1}{m}\sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x}), \quad \sigma(t) = \max(0,t)$$

What class of functions are well-approximated by two-layer neural networks?

Integral representation: Consider functions $f : X = [0,1]^d \mapsto \mathbb{R}$ of the form

$$f(\boldsymbol{x}) = \int_{\Omega} a\sigma(\boldsymbol{w}^T\boldsymbol{x})\rho(da, d\boldsymbol{w}) = \mathbb{E}_{\rho}[a\sigma(\boldsymbol{w}^T\boldsymbol{x})], \quad \boldsymbol{x} \in X$$

- $\Omega = \mathbb{R}^1 \times \mathbb{R}^{d+1}$ is the parameter space
- ρ is a probability distribution on Ω

The actual values of the weights are not important. What's important is the probability distribution of the weights.

E, Ma and Wu (2018, 2019), (related work in Barron (1993), Klusowski and Barron (2016), Bach (2017), E and Wojtowytsch (2020))

What kind of functions admit such a representation?

Theorem

Given a function $f : X \mapsto \mathbb{R}$. f_e denotes an extension of f to $\mathbb{R}^d$, and $\hat{f}_e$ is the Fourier transform of f_e. If

$$C_f := \inf_{f_e|_X = f} \int_{\mathbb{R}^d} \|\boldsymbol{\omega}\|_1^2 |\hat{f}_e(\boldsymbol{\omega})| d\boldsymbol{\omega} < \infty,$$

then f can be represented as

$$f(\boldsymbol{x}) = f(0) + \boldsymbol{x} \cdot \nabla f(0) + \int_{\Omega} a\sigma(\boldsymbol{w}^T \boldsymbol{x}) \rho(da, d\boldsymbol{w}), \quad \forall \boldsymbol{x} \in X.$$

Furthermore, we have

$$\mathbb{E}_{(a,\boldsymbol{w})\sim\rho} |a| \|\boldsymbol{w}\|_1 \leq 2C_f.$$

Breiman (1993), Barron and Klusowski (2016)

The Barron space

Definition (Barron space)

Consider function $f : X \mapsto \mathbb{R}$. Define the *"Barron norm"*

$$\|f\|_{\mathcal{B}} := \inf_{\rho \in \Psi_f} \mathbb{E}_\rho |a| \|\boldsymbol{w}\|_1.$$

where $\Psi_f = \{\rho : f(\boldsymbol{x}) = \mathbb{E}_\rho a\sigma(\boldsymbol{w}^T \boldsymbol{x})\}$, the set of possible representations for f.
Define the set of *Barron functions*

$$\mathcal{B} = \{f \in C(X) : \|f\|_{\mathcal{B}} < \infty\}$$

E, Chao Ma, Lei Wu (2019)

Structural theorem

Theorem

Let f be a Barron function. Then $f = \sum_{i=1}^{\infty} f_i$ where $f_i \in C^1(\mathbb{R}^d \setminus V_i)$ where V_i is a k-dimensional affine subspace of $\mathbb{R}^d$ for some $0 \leq k \leq d-1$.

As a consequence, distance functions to curved surfaces are not Barron functions.

- $f_1(\boldsymbol{x}) = \text{dist}(\boldsymbol{x}, \mathbb{S}^{d-1})$, then f_1 is not a Barron function.
- $f_2(\boldsymbol{x}) = \|\boldsymbol{x}\|$, f_2 is a Barron function.

E and Wojtowytsch (2020)

Direct approximation theorem

Theorem

For any $f^* \in \mathcal{B}$, there exists a two-layer network $f_m(\cdot;\theta)$ such that

$$\|f^* - f_m(\cdot;\theta)\|_{L^2(\mu)} \lesssim \frac{\|f^*\|_{\mathcal{B}}}{\sqrt{m}}.$$

Moreover,

$$\|\theta\|_{\mathcal{P}} \lesssim \|f^*\|_{\mathcal{B}}$$

Path norm:

$$\|\theta\|_{\mathcal{P}} = \frac{1}{m}\sum_{k=1}^{m} |a_k| \|\boldsymbol{w}_k\|_1,$$

if $f_m(\boldsymbol{x};\theta) = \frac{1}{m}\sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$

– discrete analog of the Barron norm, but for the parameters.

Inverse approximation theorem

Theorem

Let f^* be a continuous function. Assume there exist a constant C and a sequence of two-layer neural networks $\{f_m\}$ such that

$$\frac{1}{m}\sum_{k=1}^{m}|a_k|\|\boldsymbol{w}_k\|_1 \leq C, m \in \mathbb{N}^+,$$

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

for all $\boldsymbol{x} \in X$, then $f^* \in \mathcal{B}$, i.e. there exists a probability distribution ρ^* on Ω, such that

$$f^*(\boldsymbol{x}) = \int_{\Omega} a\sigma(\boldsymbol{w}^T\boldsymbol{x})\rho^*(da, d\boldsymbol{w}) = \mathbb{E}_{\rho^*}a\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

for all $\boldsymbol{x} \in X$ and $\|f^*\|_{\mathcal{B}} \leq C$.

Conclusion: Roughly speaking, functions that are well approximated by two-layer neural networks are functions that admit the above integral representation.
Barron space is the right function space associated with two-layer neural networks.

Other characterizations of Barron space can be found in Kurkova (2001), Bach (2017), Siegel and Xu (2021), etc.

Extensions:

- Extension to residual neural networks (E, Ma and Wu (2019, 2020)):
  - where in place of the Barron space, we have the "*flow-induced function space*".
- Extension to multi-layer neural networks, but results unsatisfactory.
  - Need a natural way of representing "continuous" multi-layer neural networks as expectations over probability distributions on the parameter space, i.e. the analog of:

$$f(\boldsymbol{x}) = \mathbb{E}_\rho[a\sigma(\boldsymbol{w}^T\boldsymbol{x})], \quad \boldsymbol{x} \in X$$

Estimation Error

We are minimizing the **training error**:

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

But what we are really interested in is to minimize the **testing error**:

$$\mathcal{R}(f) = \int_X (f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 d\mu$$

The Runge phenomenon

What happens outside the training dataset?

Example: Polynomial interpolation on equally spaced grid points

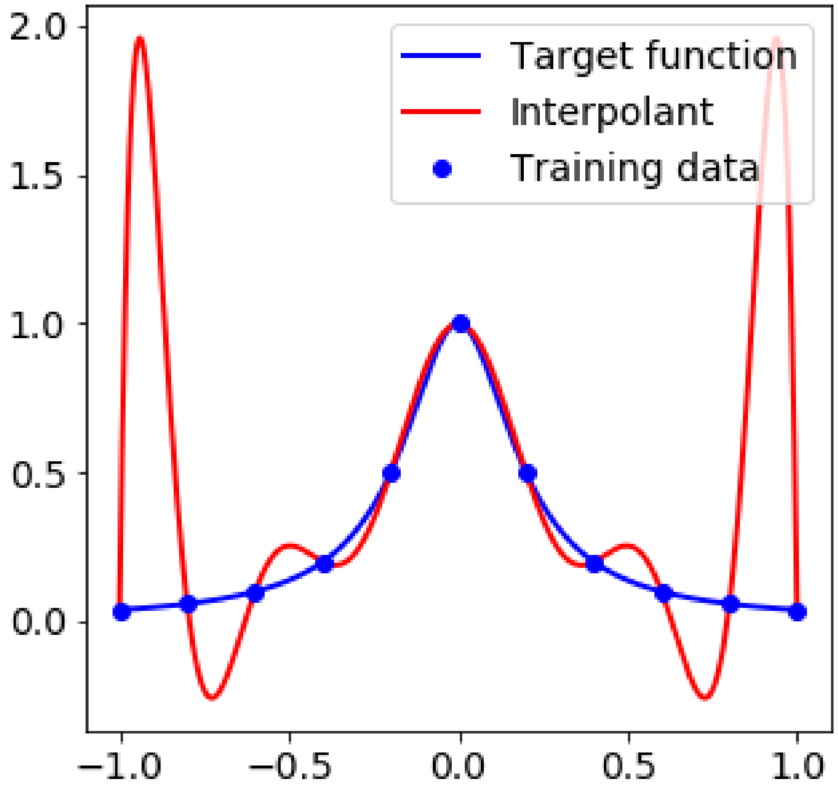


Figure: The Runge phenomenon: $f^*(x) = \frac{1}{1+25x^2}$

Generalization gap = difference between training and testing errors

generalization gap:

$$|\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}_n(\hat{f})| = |\mathbb{E}_{\boldsymbol{x}\sim\mu}\hat{g}(\boldsymbol{x}) - \frac{1}{n}\sum_{j=1}^{n}\hat{g}(\boldsymbol{x}_j)|$$

where $\hat{g}(\boldsymbol{x}) = (\hat{f}(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$.

Naively, one might expect:

$$\mathbb{E}(\text{generalization gap})^2 = O(1/n)$$

This is not necessarily true since $\hat{f}$ is highly correlated with $\{\boldsymbol{x}_j\}$.

Bounding the generalization gap

Use the naive bound:

$$|\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}_n(\hat{f})| \leq \sup_{f \in \mathcal{H}_m} |\mathcal{R}(f) - \hat{\mathcal{R}}_n(f)|$$

Theorem

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

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

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

Rademacher complexity of a function space $\mathcal{H}$

The Rademacher complexity of a function space measures its ability to fit random noise on a set of data points.

Definition: 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. The Rademacher complexity of $\mathcal{H}$ with respect to S is defined as

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

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

- If $\mathcal{H}$ = unit ball in C^0:

$$\widehat{\mathrm{Rad}}_n(\mathcal{H}) \sim O(1)$$

- If $\mathcal{H}$ = unit ball in Lipschitz space:

$$\widehat{\mathrm{Rad}}_n(\mathcal{H}) \sim O(1/n^{1/d})$$

Another form of CoD! (note that n is the size of the training dataset).

As d grows, the size of the training dataset needed grows exponentially fast.

Rademacher complexity of RKHS

Theorem

Assume that $\sup_{\boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{x}) \leq 1$. *Let* $\mathcal{H}_k^Q = \{f : \|f\|_{\mathcal{H}_k} \leq Q\}$. *Then,*

$$\widehat{\mathrm{Rad}}_n(\mathcal{H}_k^Q) \leq \frac{Q}{\sqrt{n}}.$$

Rademacher complexity of Barron space

Theorem

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

$$\widehat{\mathrm{Rad}}_n(\mathcal{F}_Q) \leq 2Q\sqrt{\frac{2\ln(2d)}{n}}$$

Neyshbur et al. (2015), Bach (2017)

Generalization error analysis for two-layer neural networks

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

Theorem

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

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

For Barron functions, not only do good two-layer neural network approximations exist, they can be found using only a finite training dataset (achieves "Monte Carlo error rate").

E, Chao Ma and Lei Wu (2018)

The Training Process

Can we find good solutions efficiently using gradient descent?

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

is a non-convex function in a high dimensional space.

1. Can gradient descent converge fast?

3rd source of **CoD**: the convergence rate.

2. Does the solution we obtain **generalize well (i.e. have small testing error)**?

Hardness of gradient-based training algorithms

- Let $h(\cdot;\theta)$ be any parametric model such that $Q(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mu}\|\nabla_\theta h(\boldsymbol{x};\theta)\|_2^2 < \infty$
- Let $\mathcal{R}^f(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mu}[(h(\boldsymbol{x};\theta) - f(\boldsymbol{x}))^2]$, the loss function associated with f.

Lemma

Let $\mathcal{F} = \{f_1, \ldots, f_M\}$ be an orthonormal class, i.e., $\langle f_i, f_j\rangle_{L^2(\mu)} = \delta_{i,j}$. We have

$$\frac{1}{M}\sum_{i=1}^{M}[\|\nabla\mathcal{R}^{f_i}(\theta) - \overline{\nabla\mathcal{R}^f(\theta)}\|_2^2] \leq \frac{Q(\theta)}{M}.$$

where $\overline{\nabla\mathcal{R}^f(\theta)} = \frac{1}{M}\sum_{j=1}^{M}\nabla\mathcal{R}^{f_j}(\theta)$.

We only have limited ability to distinguish target functions using gradients if there are many orthonormal functions in the function class.

- If $M = \exp(d)$, then the variance of the gradients is exponentially small.
- **The convergence rate for gradient-based training algorithms must suffer from CoD!**

Shamir (2018)

Barron space is such a function class

Lemma

Let $s > 0$ be a fixed number, $\mathcal{B}^s = \{f \in \mathcal{B} : \|f\|_{\mathcal{B}} \lesssim (1+s)^2 d^2\}$. Then $\mathcal{B}^s$ contains at least $(1+s)^d$ orthonormal functions.

Proof:

- Consider the set of orthogonal functions:

$$\mathcal{G}_m = \left\{ \cos(2\pi \mathbf{b}^T \boldsymbol{x}) \, : \, \sum_{i=1}^{d} b_i \leq m, b_i \in \mathbb{N}_+ \right\}.$$

Conclusion: Barron space is the right object for approximation theory, but is too big for training.

Barron (1993)

Some progress made so far

- Highly over-parametrized regime $m \gg n$: degenerate to random feature model
- Mean field regime: gradient flow in Wasserstein metric
- Which global minimum is selected?

Using deep learning to solve other high dimensional problems

- Scientific computing: control problems, PDEs
- **AI for Science**: protein folding, molecular dynamics, quantum many-body problem, multi-scale and multi-physics modeling

This began in 2016......

1. Stochastic control

- Dynamic model:

$$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \mathbf{g}_l(\boldsymbol{z}_l, \boldsymbol{a}_l) + \xi_l,$$

  where $\boldsymbol{z}_l$ = state, $\boldsymbol{a}_l$ = control, ξ_l = noise.

- Objective function:

$$\min_{\{\boldsymbol{a}_l\}_{l=0}^{T-1}} \mathbb{E}_{\{\xi_l\}}\Big\{\sum_{l=0}^{T-1} c_l(\boldsymbol{z}_l, \boldsymbol{a}_l) + c_T(\boldsymbol{z}_T)\Big\},$$

  where $\{c_l\}$ are the running cost, c_T is the terminal cost.

- Look for a feedback control:

$$\boldsymbol{a}_l = \boldsymbol{a}_l(\boldsymbol{z}).$$

The standard approach via solving the Bellman equation suffers from CoD!

Jiequn Han and E (2016)

Why choosing this as the first example?

There is a close analogy between stochastic control and ResNet-based deep learning.

Machine learning approximation:

$$\boldsymbol{a}_l(\boldsymbol{z}) = f(\boldsymbol{z}, \theta_l)$$

	ResNet	Stochastic Control
model	$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \sigma(W_l \boldsymbol{z}_l)$	$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \boldsymbol{g}_l(\boldsymbol{z}_l, f(\boldsymbol{z}_l, \theta_l)) + \xi_l$
loss	$\mathbb{E}\|W_L \boldsymbol{z}_L - f^*\|^2$	$\mathbb{E}\{\sum c_l(\boldsymbol{z}_l, f(\boldsymbol{z}_l, \theta_l)) + c_T(\boldsymbol{z}_T)\}$
data	$\{(\boldsymbol{x}_j, y_j)\}$	$\xi_0, \ldots, \xi_{T-1}$ (noise)
optimization	SGD	SGD

Table: Analogy between ResNet and stochastic control

2. Nonlinear parabolic PDEs

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

Reformulate as a stochastic control problem using backward stochastic differential equations (BSDE, Pardoux and Peng (1990))

$$\begin{aligned} &\inf_{Y_0, \{Z_t\}} \mathbb{E}|g(X_T) - Y_T|^2, \\ s.t. \quad &X_t = X_0 + \int_0^t \mu(s, X_s)\ ds + \int_0^t dW_s, \\ &Y_t = Y_0 - \int_0^t f(Z_s)\, ds + \int_0^t (Z_s)^{\mathrm{T}}\, dW_s. \end{aligned}$$

The unique minimizer is the solution to the PDE with:

$$Y_t = u(t, X_t) \qquad \text{and} \qquad Z_t = \nabla u(t, X_t).$$

E, Han and Jentzen (Comm Math Stats, 2017); Han, Jentzen and E (PNAS, 2018)

LQG (linear quadratic Gaussian) for $d = 100$ with the cost $J = \mathbb{E}(\int_0^T \|\mathbf{m}_t\|_2^2 \, dt + g(X_T))$

$$dX_t = 2\sqrt{\lambda}\,\mathbf{m}_t \, dt + \sqrt{2}\, dW_t,$$

Hamilton-Jacobi-Bellman equation:

$$\partial_t u + \Delta u - \lambda \|\nabla u\|_2^2 = 0, \; u(T, \boldsymbol{x}) = g(\boldsymbol{x})$$

Using Hopf-Cole transform, one obtains the solution:

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

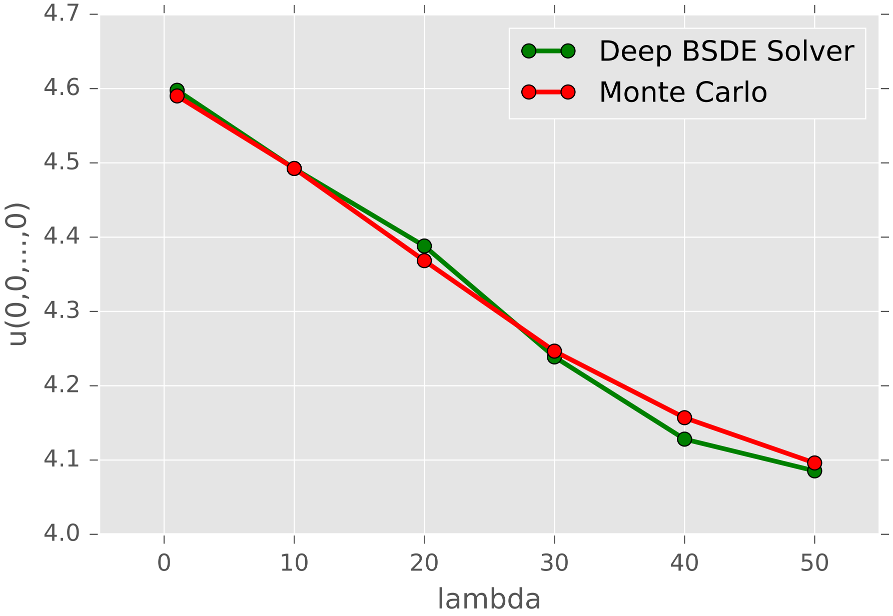
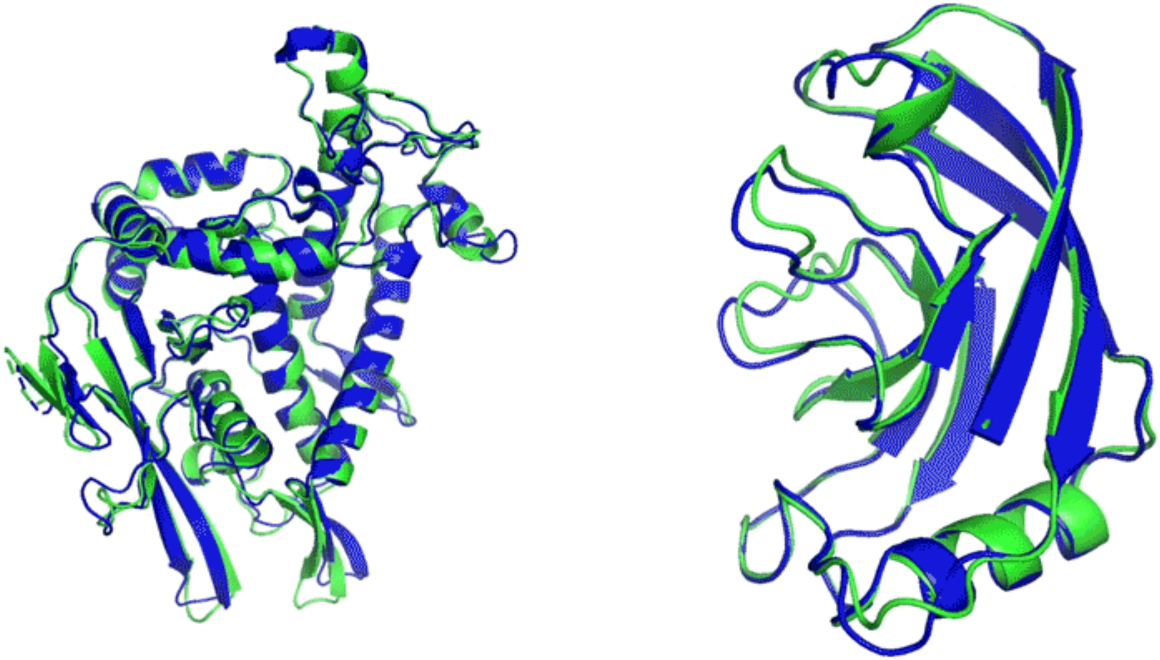


Figure: Optimal cost $u(t{=}0, \boldsymbol{x}{=}(0, \ldots, 0))$ for different values of λ.

3. AlphaFold2: Protein Folding



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

T1049 / 6y4f
93.3 GDT
(adhesin tip)

🟩 Experimental result

🟦 Computational prediction

J. Jumper et al. (2021)

4. DeePMD: Molecular dynamics with *ab initio* accuracy

Modeling the dynamics of atoms in a material or molecule using Newton's equation:

$$m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} = -\nabla_{\boldsymbol{x}_i} V, \quad V = V(\boldsymbol{x}_1, ..., \boldsymbol{x}_N),$$

Key question: $V =?$ The origin of V lies in quantum mechanics (QM).

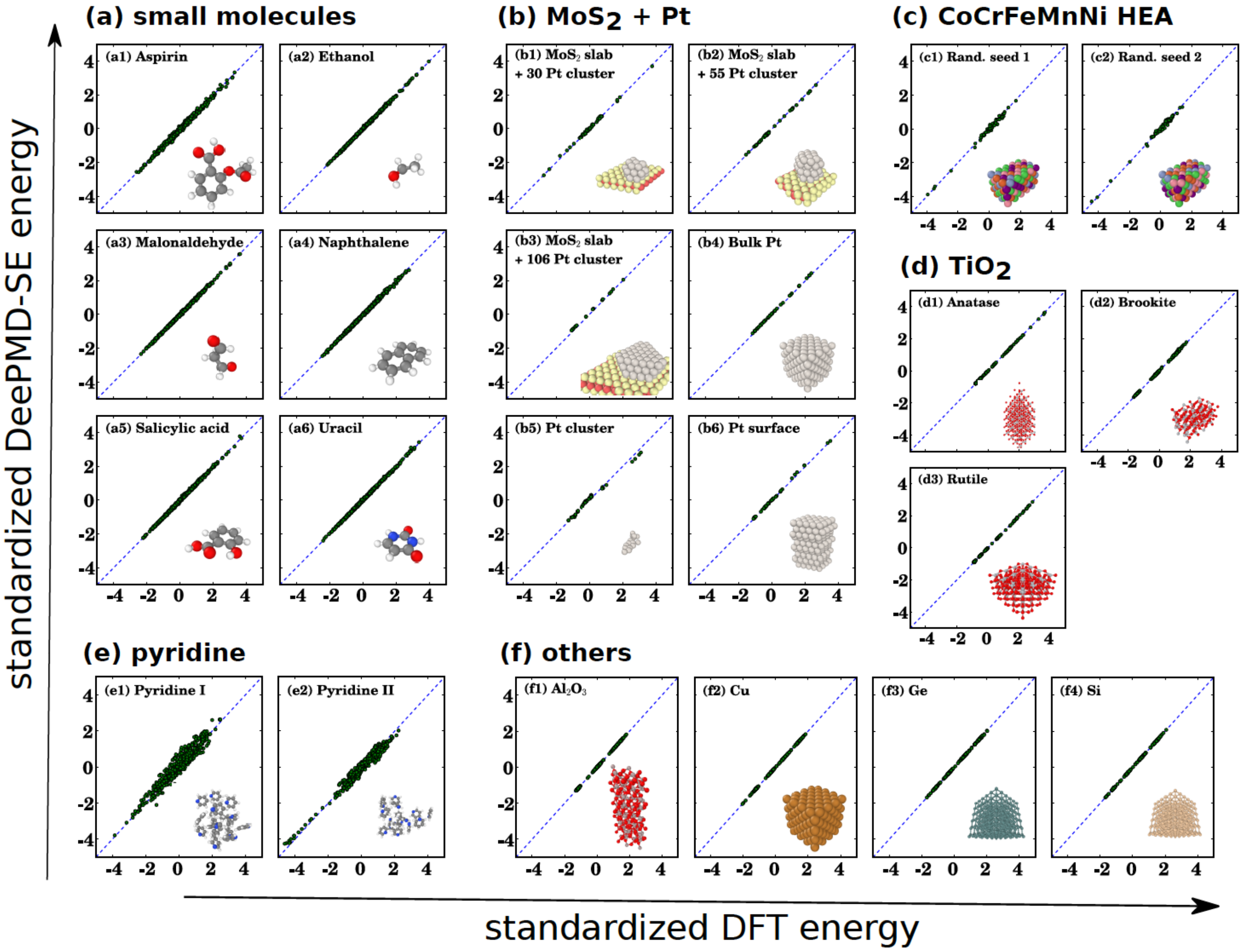
- Empirical potentials: basically guess what V should be.
  Unreliable.
- Compute the forces on the fly using QM models (Car and Parrinello (1985)).
  As reliable as the QM model but expensive (limited to about 1000 atoms).

New paradigm:

- use QM to supply the data
- use neural network model to find accurate approximation of V

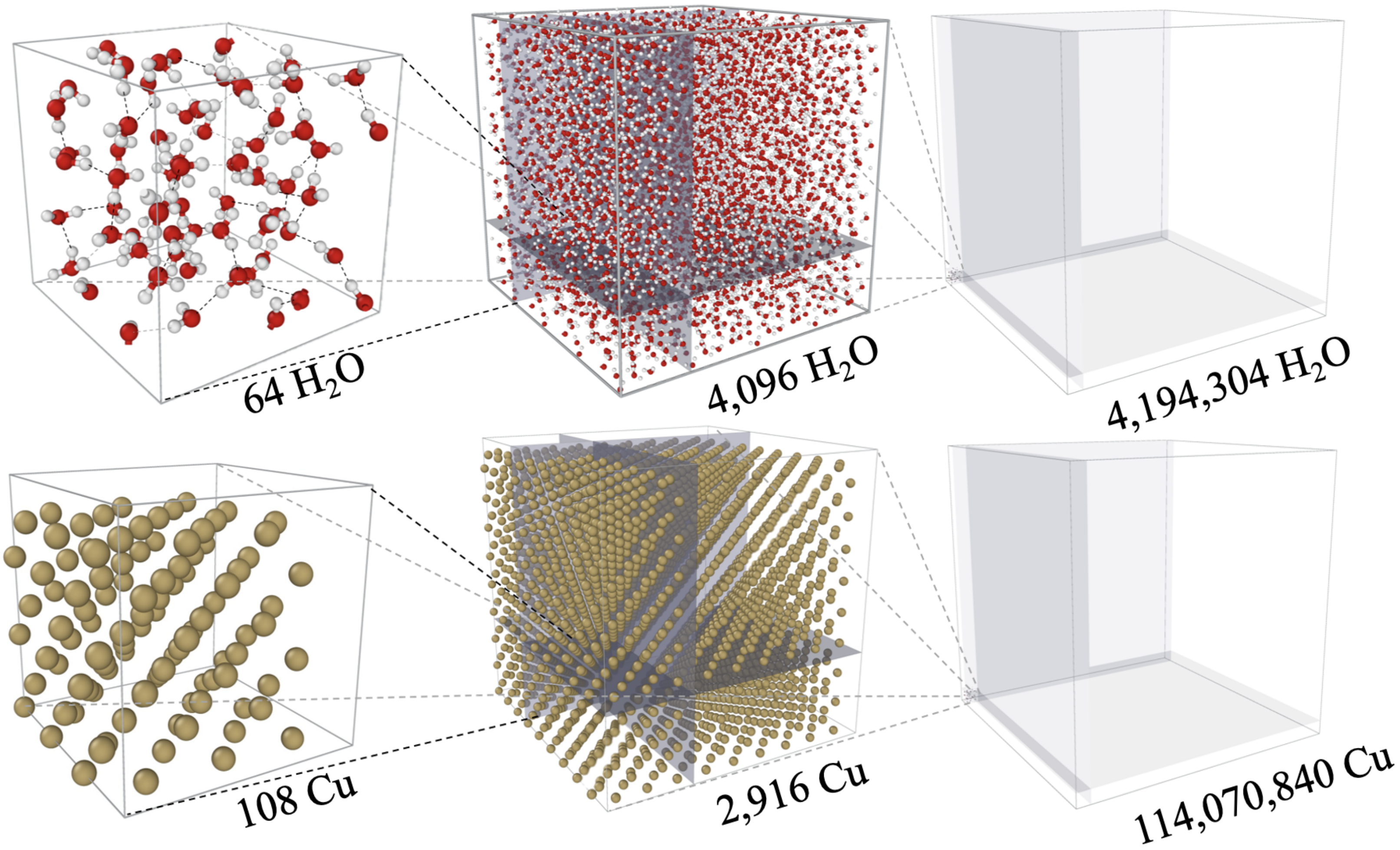
Behler and Parrinello (2007), Jiequn Han et al (2017), Linfeng Zhang et al (2018).

Accuracy comparable to QM for a wide range of materials and molecules



Linfeng Zhang, Jiequn Han, et al (2018)

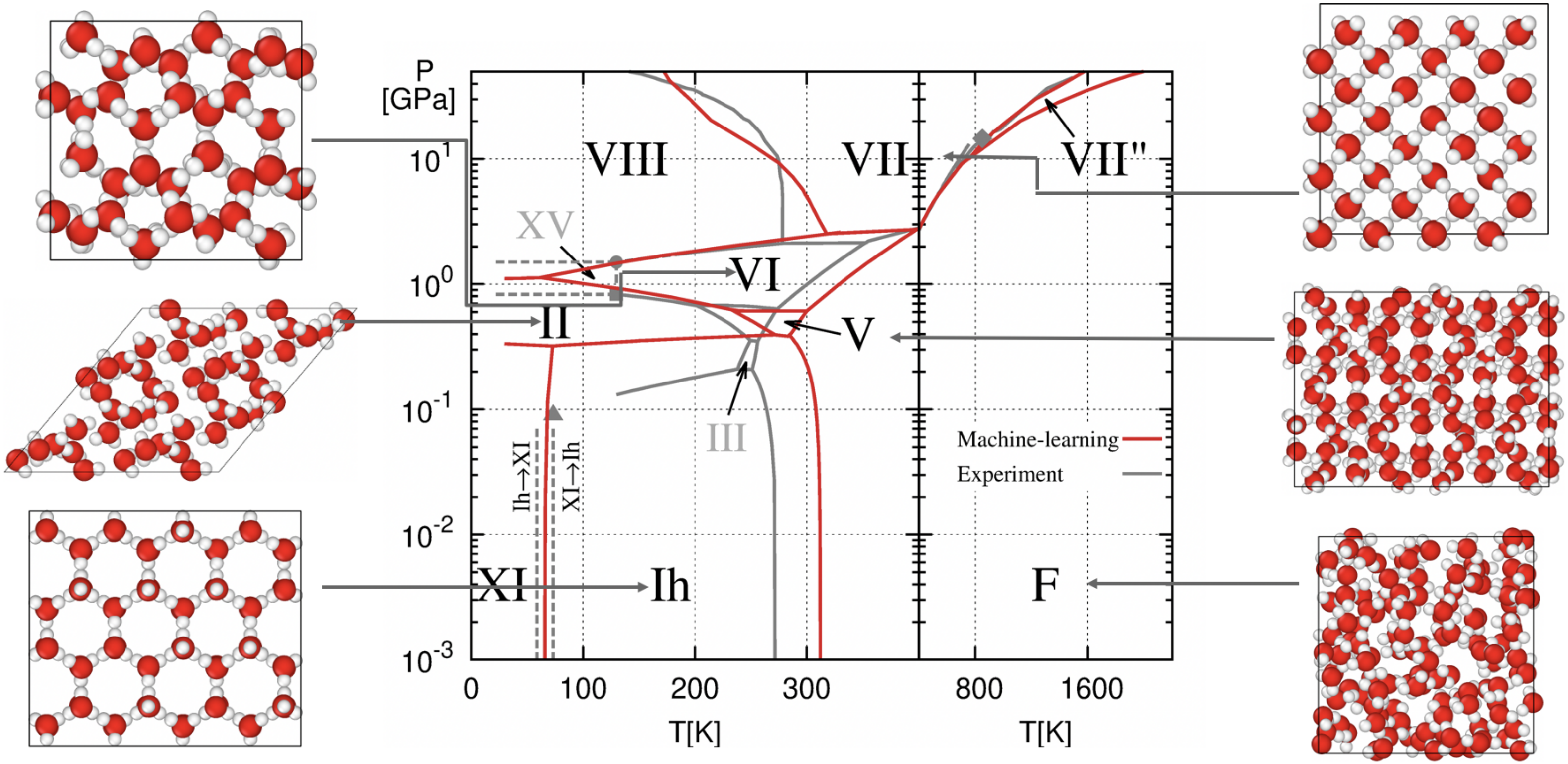
DeePMD simulation of 100M atoms with *ab initio* accuracy



(a) AIMD + HPC; (b) DeePMD+1 GPU @ Home; (c) DeePMD+27360 GPUs @ Summit.

Weile Jia, et al, SC20, 2020 ACM Gordon Bell Prize

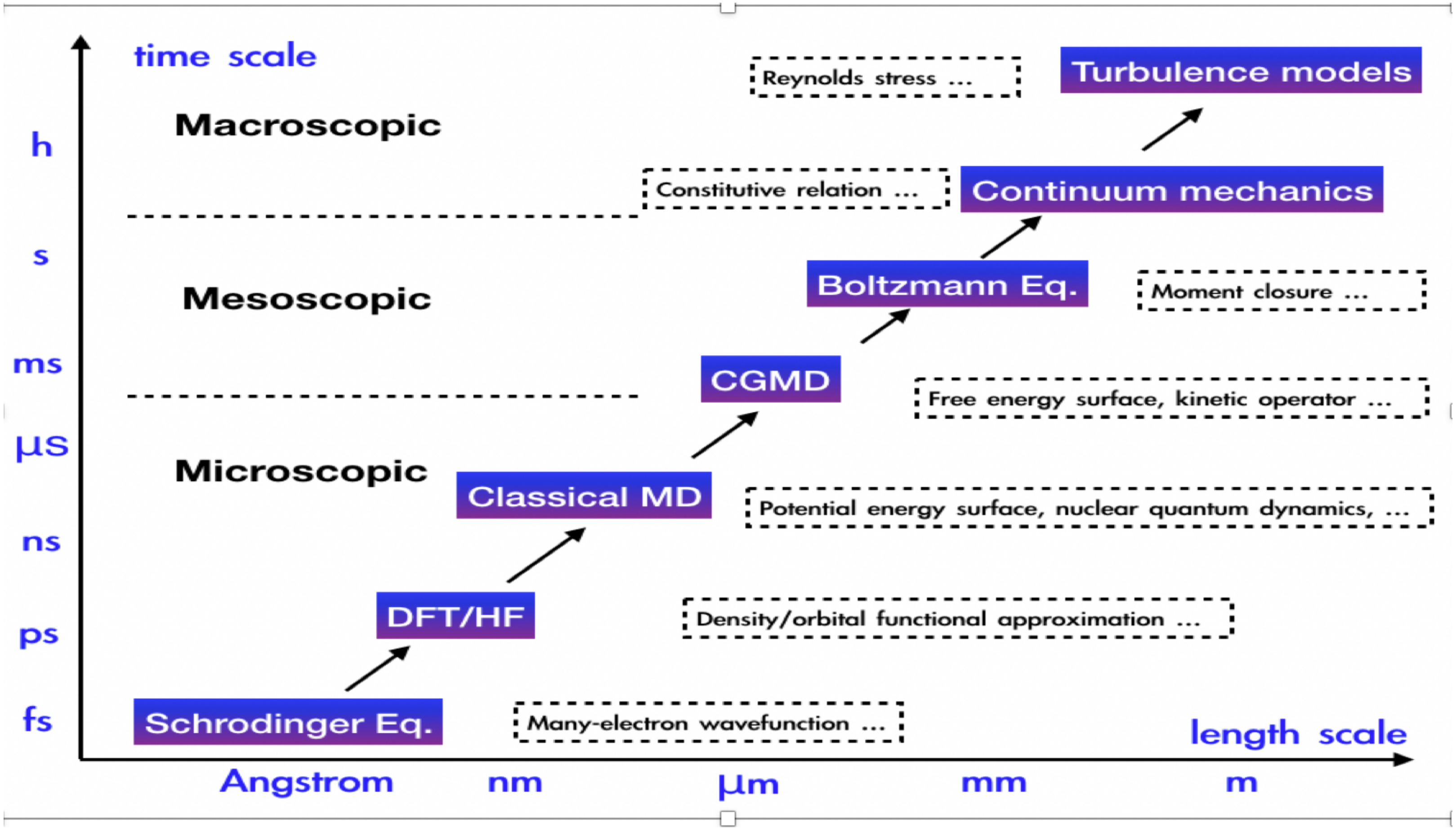
Phase diagram of water



Linfeng Zhang, Han Wang, et al. (2021)

The hierarchy of physical models: The real dilemma

- On one hand, these “first principles” represent the most important results of the whole scientific endeavor.
- On the other hand, to use them, we have to do a lot of fudging.



AI for Science: Making first principles truly reliable and useful

Machine learning provides the missing tool:

- *Quantum many-body problem*: RBM (2017), DeePWF (2018), FermiNet (2019), PauliNet (2019), ...
- *Density functional theory*: DeePKS (2020), NeuralXC (2020), DM21 (2021), ...
- *Molecular dynamics*: DeePMD (2018), ...
- *Coarse-grained molecular dynamics*: DeePCG (2019)
- *Kinetic equation*: machine learning-based moment closure (Han et al. 2019)
- *Continuum mechanics*: DeePN^2 (2020)
- ......

This will change the way we solve many practical problems (drug design, materials, combustion engines, catalysts, etc) from trial and error to first principle-based.

E, Jiequn Han and Linfeng Zhang, Physics Today, 2021.

Concluding remarks: This is all about math in high dimension

- **Compared with polynomials, neural networks provide a much more effective tool for approximating functions in high dimension.**
- Opens up a new subject in mathematics: **high dimensional analysis**.
  - supervised learning: high dimensional functions
  - unsupervised learning: high dimensional probability distributions
  - reinforcement learning: high dimensional Bellman equations
  - time series: high dimensional dynamical systems
- This opens up a lot of new possibilities in science, AI, and technology.

See:

www.math.princeton.edu/ $\sim$ weinan

Training two-layer neural networks under conventional scaling

$$f_m(\boldsymbol{x};\boldsymbol{a},\boldsymbol{W}) = \sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x}) = \boldsymbol{a}^T \sigma(\boldsymbol{W}\boldsymbol{x}),$$

Initialization:

$$a_j(0) = 0, \quad \boldsymbol{w}_j(0) \sim \mathcal{N}(0, I/d), \quad j \in [m]$$

Define the associated Gram matrix $K = (K_{ij})$:

$$K_{i,j} = \frac{1}{n}\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,I/d)}[\sigma(\boldsymbol{w}^T\boldsymbol{x}_i)\sigma(\boldsymbol{w}^T\boldsymbol{x}_j)].$$

The associated random feature model: $\{\boldsymbol{w}_j\} = \{\boldsymbol{w}_j^0\}$ are frozen, only allow $\{a_j\}$ to vary.

Gradient descent dynamics

$$\begin{aligned}\frac{da_j}{dt}(t) &= -\nabla_{a_j}\hat{\mathcal{R}}_n \sim O(\|\boldsymbol{w}_j\|) = O(1)\\ \frac{d\boldsymbol{w}_j}{dt}(t) &= -\nabla_{\boldsymbol{w}_j}\hat{\mathcal{R}}_n \sim O(|a_j|) = O\left(\frac{1}{\lambda_n m}\right)\end{aligned}$$

where $\lambda_n = \lambda_{\min}(K)$

In the “highly over-parametrized regime” (i.e. $m \gg n$), we have time scale separation: the dynamics of $\boldsymbol{w}$ is effectively frozen.

Highly over-parametrized regime

Jacot, Gabriel and Hongler (2018): "**neural tangent kernel**" regime

- Good news: Exponential convergence (Du et al (2018))
- Bad news: converged solution is no better than that of the random feature model (E, Ma, Wu (2019), Arora et al (2019), ...)

Theorem

Denote by $\{f_m(\boldsymbol{x};\tilde{\boldsymbol{a}}(t),\boldsymbol{W}_0))\}$ *the solution of the gradient descent dynamics for the random feature model. For any* $\delta \in (0,1)$*, assume that* $m \gtrsim n^2\lambda_n^{-4}\delta^{-1}\ln(n^2\delta^{-1})$*. Then with probability at least* $1-6\delta$*, we have*

$$\hat{\mathcal{R}}_n(\boldsymbol{a}(t),\boldsymbol{W}(t)) \leq e^{-m\lambda_n t}\hat{\mathcal{R}}_n(\boldsymbol{a}(0),\boldsymbol{W}(0))$$

$$\sup_{\boldsymbol{x}\in\mathcal{S}^{d-1}} |f_m(\boldsymbol{x};\boldsymbol{a}(t),\boldsymbol{W}(t)) - f_m(\boldsymbol{x};\tilde{\boldsymbol{a}}(t),\boldsymbol{W}_0)| \lesssim \frac{(1+\sqrt{\ln(\delta^{-1})})^2}{\lambda_n\sqrt{m}}.$$

This is an effectively linear regime.

Mean-field formulation

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

Let

$$I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) = \hat{\mathcal{R}}_n(f_m), \quad \boldsymbol{u}_j = (a_j, \boldsymbol{w}_j)$$

Lemma:. $\{\boldsymbol{u}_j(\cdot)\}$ is a solution of the gradient descent dynamics

$$\frac{d\boldsymbol{u}_j}{dt} = -\nabla_{\boldsymbol{u}_j} I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m), \quad \boldsymbol{u}_j(0) = \boldsymbol{u}_j^0, \quad j \in [m]$$

if and only if

$$\rho_m(d\boldsymbol{u}, \cdot) = \frac{1}{m}\sum_{j=1}^{m} \delta_{\boldsymbol{u}_j(\cdot)}$$

is a solution of

$$\partial_t \rho = \nabla(\rho \nabla V), \quad V = \frac{\delta \hat{\mathcal{R}}_n}{\delta \rho}$$

Chizat and Bach (2018), Mei, Montanari and Nguyen (2018), Rotskoff and Vanden-Eijnden (2018), Sirignano and Spiliopoulos (2018)

This is the gradient flow under the Wasserstein metric

$$\partial_t \rho = \nabla(\rho \nabla V), \quad V = \frac{\delta \hat{\mathcal{R}}_n}{\delta \rho}$$

Long time decay theorem under the condition of **displacement convexity.**

Unfortunately, in general displacement convexity does not hold in the current setting.

Convergence of gradient flow

If the initial condition $\underline{\rho_0 \text{ has full support}}$ and if the gradient flow dynamics converges, then it must converge to a global minimizer.

Theorem

Let ρ_t be a solution of the Wasserstein gradient flow such that

- *ρ_0 is a density on the cone $\Theta := \{|a|^2 \leq |\boldsymbol{w}|^2\}$.*
- *Every open cone in Θ has positive measure with respect to ρ_0*

Then the following are equivalent.

1. *The velocity potentials $\frac{\delta \mathcal{R}}{\delta \rho}(\rho_t, \cdot)$ converge to a unique limit as $t \to \infty$.*
2. *$\mathcal{R}(\rho_t)$ decays to minimum Bayes risk as $t \to \infty$.*

If either condition is met, the unique limit is zero. If also ρ_t converges in Wasserstein metric, then the limit ρ_∞ is a minimizer.

Chizat and Bach (2018, 2020), Wojtowytsch (2020)