High Dimensional Numerical Analysis – the Theoretical Foundation of Machine Learning

Weinan E

Joint work with:

Chao Ma, Lei Wu

Jiequn Han, Arnulf Jentzen, Qianxiao Li, Qingcan Wang.

Outline

Outline

Supervised learning: Approximating functions using samples

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

- Strategy: Construct some "hypothesis space" (space of functions) $\mathcal{H}_m$ ($m \sim$ the dimension of $\mathcal{H}_m$). Minimize the "empirical risk":

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

- two important parameters m and n.

Main questions:

1. Optimization: can gradient descent find good minima?
2. Generalization: does the solution found generalize (is the population risk small) ?

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

Key words: High dimension and over-parametrization ($m > n$)

Over-parametrization and curse of dimensionality

$$\mathcal{H}_m = \{f = \sum_{k=1}^{m} a_k \phi_k\}$$

$m > n$, $\{\phi_k\}$ are linearly independent functions.

- Given a data set $\{\boldsymbol{x}_j, y_j; j = 1, \cdots n\}$, one can interpolate the data $(f(\boldsymbol{x}_j) = y_j, j = 1, \cdots, n)$

$$G\boldsymbol{a} = \boldsymbol{y}, \quad G = (\phi_k(\boldsymbol{x}_j))$$

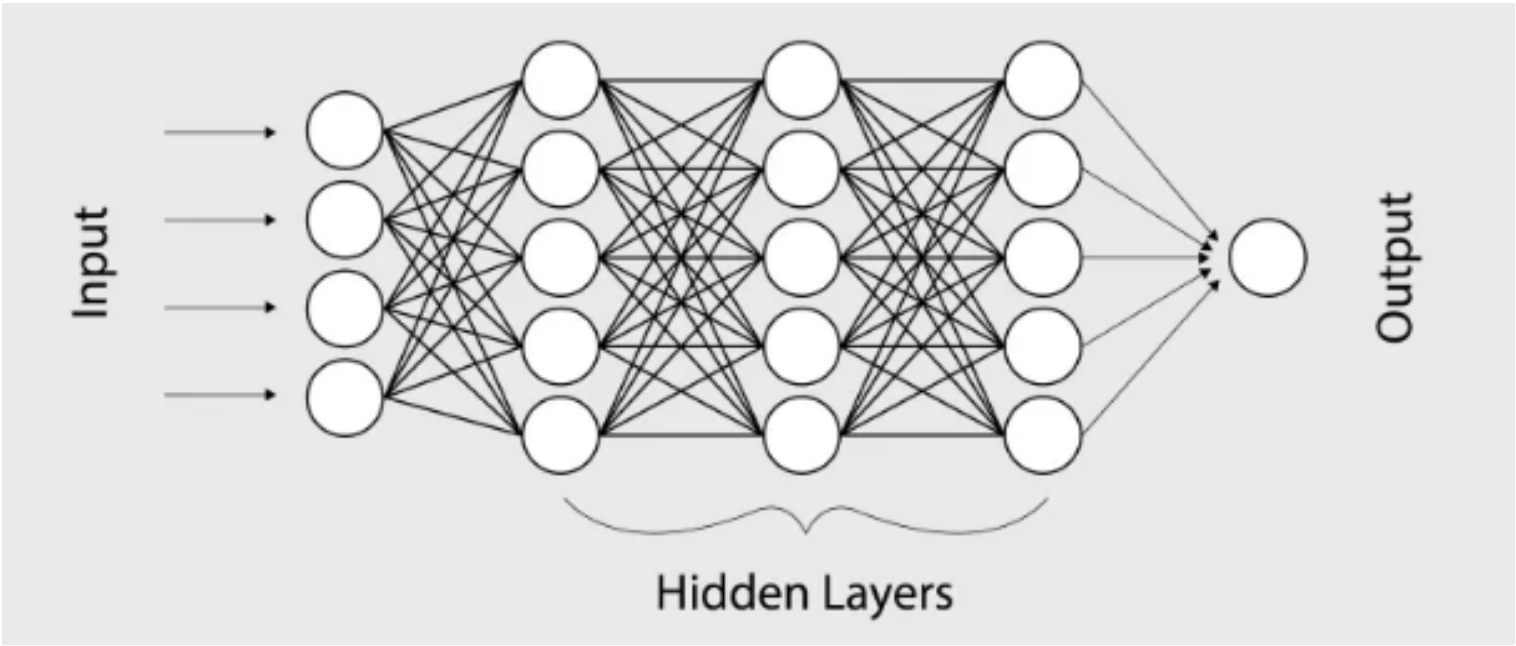
- Yet there is curse of dimensionality

$$\sup_{\|f\|_{\mathcal{B}_1} \leq 1} \inf_{h \in \mathcal{H}_m} \|f - h\|_{L^2(D_0)} \geq \frac{C}{dm^{1/d}}$$

$m \sim \varepsilon^{-d}$

How do we choose the hypothesis space?

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



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

σ is a scalar function.

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

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

Dynamical system (ODE) viewpoint to deep learning

Constructing nonlinear approximations through the **flow map** of a dynamical system (E (2017, Comm Math Stats), Chen et al (NeurIPS 2018, "Neural ODE")):

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

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

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

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

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

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

This is related to residual neural networks.

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

Outline

Solving PDEs in very high dimensions

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

Example: Black-Scholes Equation with Default Risk:

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

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

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

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

Connection to the PDEs (nonlinear Feynman-Kac formula):

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

In other words, given the stochastic process satisfying

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

the solution of PDE satisfies the following SDE

$$u(t, X_t) - u(0, X_0) = -\int_0^t f(u(s, X_s), \nabla u(s, X_s))ds + \int_0^t \nabla u(s, X_s)dW_s.$$

Neural Network Approximation

Deep BSDE method

(Jiequn Han, Arnulf Jentzen and Weinan E (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.

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

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

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

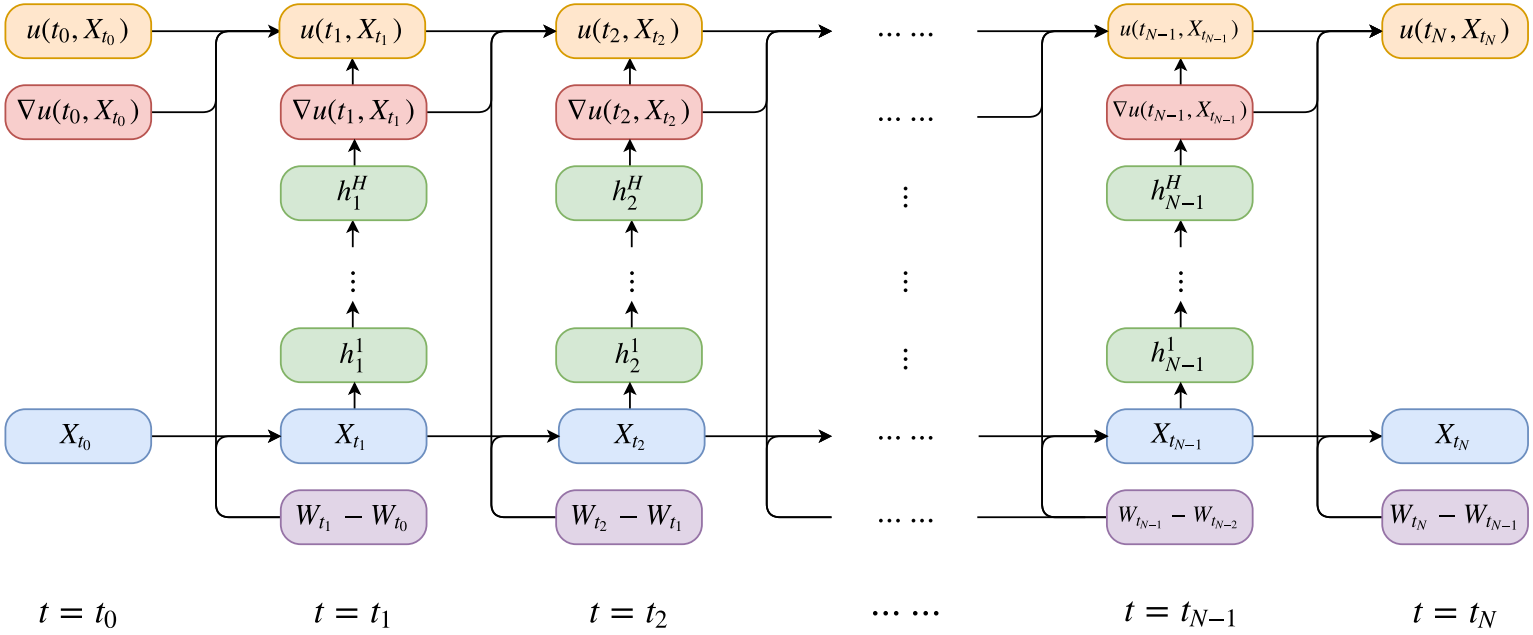


Figure: Each column corresponds to a subnetwork at time $t = t_n$

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

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

Example for d=100

HJB equation:

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

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

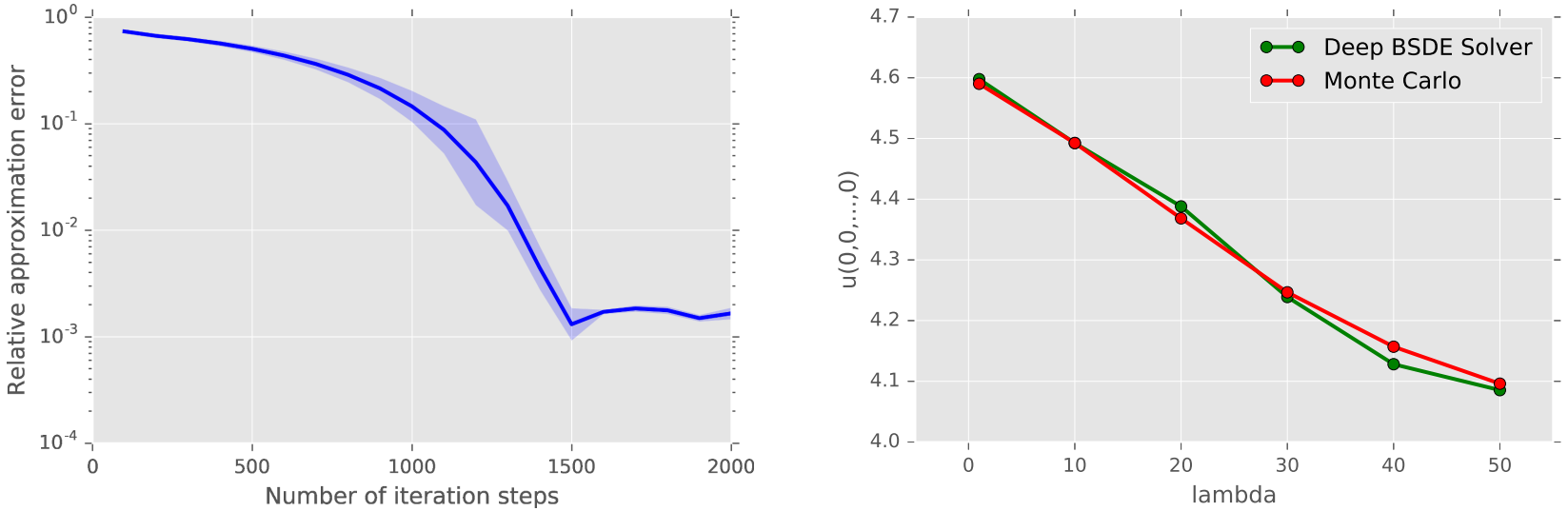


Figure: Left: Relative error of the deep BSDE method for $u(t{=}0, x{=}(0))$ when $\lambda = 1$, which achieves 0.17% in a runtime of 330 seconds. Right: $u(t{=}0, x{=}(0))$ for different λ.

Black-Scholes Equation with Default Risk for d=100

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

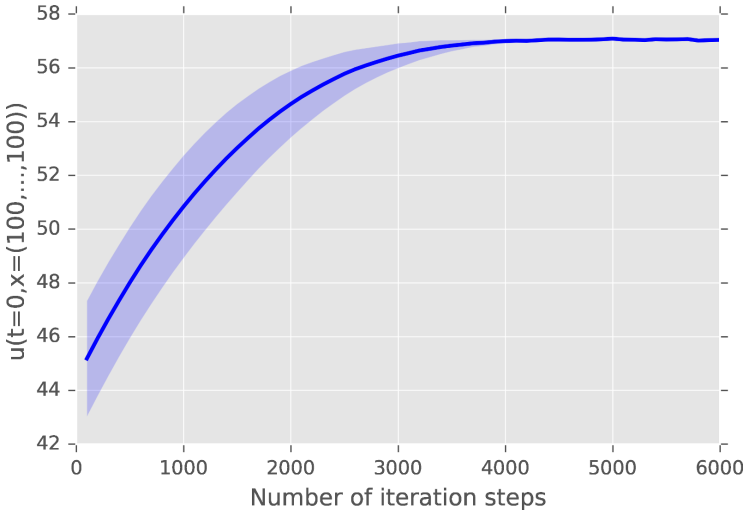


Figure: Approximation of $u(t{=}0, x{=}(100, \dots, 100))$ against number of iteration steps. The deep BSDE method achieves a relative error of size 0.46% in a runtime of 617 seconds.

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

Other applications

- molecular modeling (molecular dynamics, coarse-grained molecular dynamics, reinforced dynamics, ...)
- gas dynamics and kinetic equation
- turbulence models (large eddy simulation)
- image search and identification

Outline

Why neural networks?

Difference between linear and nonlinear approximations:

- Linear: $f(\boldsymbol{x}) \approx f_m(\boldsymbol{x}) = \sum^m a_{\boldsymbol{k}} \cos(2\pi \boldsymbol{k} \cdot \boldsymbol{x}), \boldsymbol{x} \in [0,1]^d$

$$\inf \|f - f_m\|_2 \geq C(f) m^{-1/d}$$

"Curse of dimensionality": number of parameters needed goes up exponentially fast as a function of the accuracy requirement.

- Nonlinear: $f(\boldsymbol{x}) \approx f_m(\boldsymbol{x}) = \sum^m a_{\boldsymbol{k}} \cos(2\pi \boldsymbol{b}_k \cdot \boldsymbol{x}), \boldsymbol{x} \in [0,1]^d$ (two-layer neural network, with $\cos(x)$ as the activation function).

$$\inf \|f - f_m\|_2 \leq C(f) m^{-1/2}$$

This is the best one can hope for.

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, $\alpha = 1/d, s = 2$)

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

  - *A posteriori* estimates (say in finite elements):

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

We will adopt a similar strategy, but aim for high dimensions.

Another benchmark: High dimensional integration

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

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

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

$\mathrm{Var}(g) = \int_X g^2(\boldsymbol{x}) d\boldsymbol{x} - (\int_X g(\boldsymbol{x}) d\boldsymbol{x})^2$

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

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

But we only have finite number of data

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

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

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

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

- 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 $S = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$,

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

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

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: $\{\phi(\cdot, \omega), \omega \in \Omega\}$ is the set of "features". Given any realization $\{\omega_j\}_{j=1}^m$, i.i.d. with distribution π, $\mathcal{H}_m(\{\omega_j\}) = \{f_m(\boldsymbol{x}, \boldsymbol{a}) = \frac{1}{m}\sum_{j=1}^m a_j \phi(\boldsymbol{x}; \omega_j).\}$.
- two layer neural networks $\mathcal{H}_m = \{\frac{1}{m}\sum_{j=1}^m a_j \sigma(\boldsymbol{b}_j^T \boldsymbol{x} + c_j)\}$
- residual neural networks $\mathcal{H}_L = \{f(\cdot, \theta) = \alpha \cdot \boldsymbol{z}_{L,L}(\cdot)\}$

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

Outline

Random feature model

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

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

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

Looking for the right function space: Consider functions of the form

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

This is related to the reproducing kernel Hilbert space (RKHS) with 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

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

Outline

Barron spaces

Two-layer neural networks:

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

Consider the function $f : D_0 = [0,1]^d \mapsto \mathbb{R}$ of the following form

$$f(\boldsymbol{x}) = \int_{\Omega} a\sigma(\boldsymbol{b}^T \boldsymbol{x} + c)\rho(da, d\boldsymbol{b}, dc), \quad \boldsymbol{x} \in D_0$$

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

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

$$\mathcal{B}_p = \{f \in C^0 : \|f\|_{\mathcal{B}_p} < \infty\}$$

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

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

Theorem (Direct Approximation Theorem)

There exists an absolute constant C_0 such that

$$\|f - f_m\|_{L^2(D_0)} \leq \frac{C_0 \|f\|_{\mathcal{B}_2}}{\sqrt{m}}$$

Theorem (Inverse Approximation Theorem)

For $p > 1$, let

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

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 $\boldsymbol{x} \in D_0$, 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 $\boldsymbol{x} \in D_0$.

Complexity estimates

Theorem

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

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

Barron space and RKHS

Equivalent formulation (taking conditional expectation with respect to $\mathbf{w} = (\boldsymbol{b}, c)$):

$$f^*(\boldsymbol{x}) = \int a(\mathbf{w})\sigma(\mathbf{w}^T\boldsymbol{x})\pi(d\mathbf{w}), \quad \boldsymbol{x} = (\boldsymbol{x}, 1)$$

Define:

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

We can write

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

Shallow neural network can be understood as kernel method with adaptive (learned) kernel.

The ability to learn the right kernel is VERY important.
For example, SVM would be perfect if the right kernel was known.

A priori estimates for regularized model

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

where the path norm is defined by:

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

Theorem

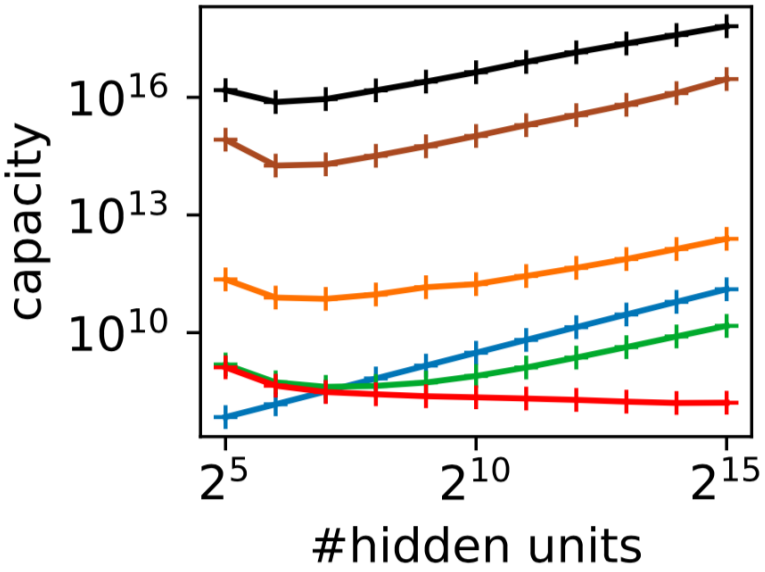
Assume that the target function $f^* : [0,1]^d \mapsto [0,1] \in \mathcal{B}_2$. There exist constants C_0, C_1, C_2, such that for any $\delta > 0$, if $\lambda \geq C_0$, then with probability at least $1-\delta$ over the choice of training set, we have

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

Traditional results: A posteriori estimates

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

where $|||\theta|||$ is some norm of θ (see e.g. Behnam Neyshabur, Zhiyuan Li, et al. *Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks* (2018)).



Outline

Main steps for the analysis

- find probablistic interpretation of the machine learning model (expressed in some law of large numbers)
- prove the corresponding "central limit theorem". This gives the convergence rate for the approximation error.
- study the Rademacher complexity. This gives control for the estimation error.

Compositional law of large numbers

Consider the following compositional scheme:

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

$(\boldsymbol{U}_l, \boldsymbol{W}_l)$ are i.i.d. sampled from a distribution ρ.

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 $L \to +\infty$.

Extension

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:

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

We can also define compositional space for this case, e.g. we define $f_{\alpha,\{\rho_t\},\boldsymbol{V}}(\boldsymbol{x}) = \alpha^T \boldsymbol{z}(\boldsymbol{x}, 1)$ and

$$\begin{aligned} \frac{d}{dt}\mathbf{N}(t) &= \mathbb{E}_{\rho_t} |\boldsymbol{U}||\boldsymbol{W}|\mathbf{N}(t), \\ \mathbf{N}(0) &= \mathbf{I} \\ \|f\|_{\mathcal{D}_1} &= \inf_{f=f_{\alpha,\{\rho_t\},\boldsymbol{V}}} \|\alpha\|_1 \|\mathbf{N}(1)\|_{1,1} \|\boldsymbol{V}\|_{1,1}, \end{aligned}$$

$\|\cdot\|_{\mathcal{D}_2}$ is defined similarly.

Barron space and the compositional function space

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 $f \in \mathcal{B}_2$,

Inverse approximation theorem

Let $f \in L^2(D_0)$. Assume that there is a sequence of residual networks $\{f_L(\boldsymbol{x})\}_{L=1}^{\infty}$ with increasing depth such that $\|f(\boldsymbol{x}) - f_L(\boldsymbol{x})\| \to 0$. Assume further that the parameters are (entry-wise) bounded, then there exists α, $\{\rho_t\}$ and $\boldsymbol{V}$ such that

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

Theorem (Direct approximation theorem)

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

$$\int_{D_0} |f(\boldsymbol{x}) - f_L((\boldsymbol{x},1);\tilde{\theta})|^2 dx \to 0 \lesssim \frac{\|f\|_{c_2}^2}{L}$$

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

$$\int_{D_0} |f(\boldsymbol{x}) - f_L((\boldsymbol{x},1);\tilde{\theta})|^2 dx \lesssim \frac{\|f\|_{\mathcal{D}_2}^2}{L}$$

Complexity control

Rademacher complexity bound for path norm

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

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

Regularized model and a priori estimates

Regularized loss function:

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

Theorem (A-priori estimate)

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 if λ *is larger than some constant, and the depth* L *is sufficiently large, for any* $\delta > 0$, *with probability at least* $1 - \delta$,

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

Outline

"Implicit optimization"

Observations about what happens in practice:

- the kind of regularizations we studied were not used
- the generalization properties depend heavily on parameter tuning (with a lot of parameter tuning, one can get good results)
- over-parametrized networks are easier to train (easier for GD or SGD to converge to global minima)
- SGD solutions usually perform better than GD solutions

Escape phenomenon (Wu and Zhu)

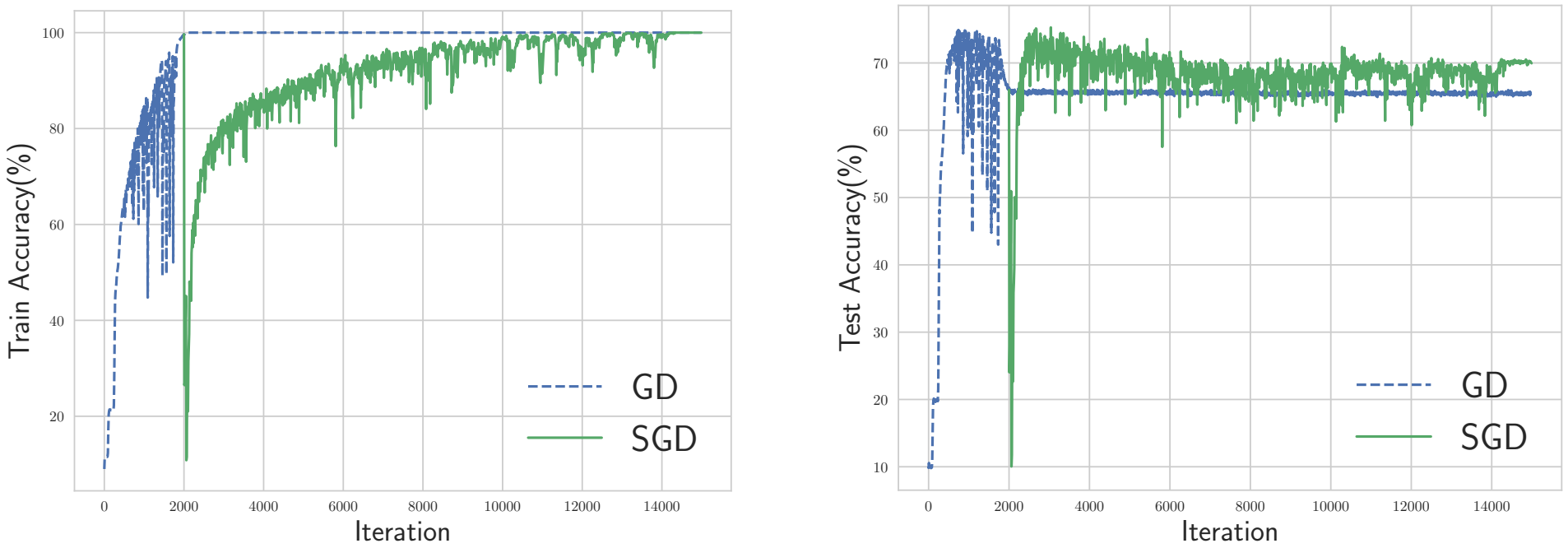


Figure: Fast escape phenomenon in fitting corrupted FashionMNIST. When the optimization algorithm is switched from GD to SGD with the same learning rate, though the GD iterations is reasonably close to a global minimum, one observes a fast escape from that global minimum and subsequent convergence to another global minimum. As shown by the right figure, the ultimate global minimum found by SGD generalizes better for this example than the one that GD was about to converge to.

One-dimensional case

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{1}$$

One-dimensional case

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{1}$$

- The SGD iteration is given by,

$$x_{t+1} = x_t - \eta a_\xi x_t = (1 - \eta a_\xi) x_t, \tag{2}$$

One-dimensional case

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{1}$$

- The SGD iteration is given by,

$$x_{t+1} = x_t - \eta a_\xi x_t = (1 - \eta a_\xi) x_t, \tag{2}$$

- So after one step update, we have

$$\mathbb{E}x_{t+1} = (1 - \eta a)\mathbb{E}x_t, \tag{3}$$
$$\mathbb{E}x_{t+1}^2 = \left[(1 - \eta a)^2 + \eta^2 s^2\right] \mathbb{E}x_t^2, \tag{4}$$

where $a = \frac{1}{n}\sum_{i=1}^{n} a_i$, $s = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2 - a^2}$.
a: sharpness s: non-uniformity.

Over-parametrized two-layer neural networks

- Du et al. It does optimize (GD converges to global min exponentially fast)
- E, Ma and Wu: It does not generalize (the GD path is uniformly close to that of a (linear) random feature model, therefore its generalization properties is no better than that of the random feature model)

To look for "implicit regularization", one has to look at non-over-parametrized regimes.

Convergence behavior

Poisson Equation: A Finite Difference Approach

$$-\partial_x^2 u(x) = g(x), \quad u(0) = u(1) = 0.$$

The finite difference scheme

$$-\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} = g(x_j)$$

yields a linear system

$$Au = g,$$

where

$$A = \frac{1}{h^2}\begin{pmatrix} -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 & -1 \end{pmatrix}.$$

Iterative solver: **lower frequency converges slower.**

Poisson Equation: DNN Approach

A $1d$ Poisson equation on $(0,1)$ with Dirichlet boundary condition,

$$-\Delta u(x) = g(x), \quad u(0) = u(1) = 0.$$

$u(x)$ can be solved by the following variational problem,

$$\min_{u \in H^1(0,1)} \int_0^1 \left(\frac{1}{2} |\partial_x u(x)|^2 - g(x)u(x) \right) \mathrm{d}x + \beta \left(u(0)^2 + u(1)^2 \right).$$

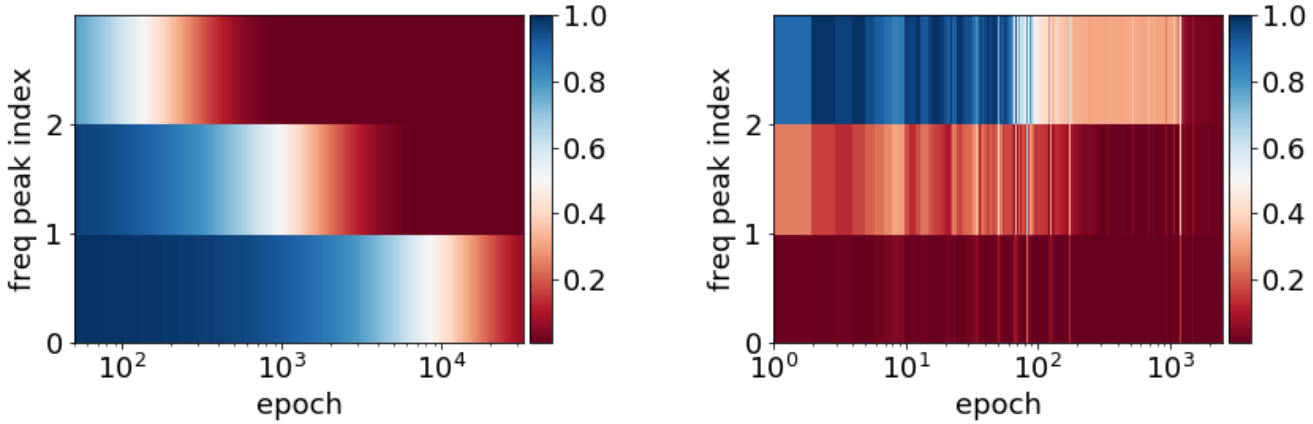
Here we can parametrize $u(x)$ using **deep neural network (DNN)**:

- **Input**: x.
- **Output**: $u(x) = u_{\mathrm{Net}}(x)$.
- **Train**: stochastic gradient decent.

E, Yu, *The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems*, Communications in Mathematics and Statistics, 2018

F-Principle

$$g(x) = \sin(x) + 4\sin(4x) - 8\sin(8x) + 16\sin(24x).$$



(a) Jacobi (b) DNN

F-Principle: *A DNN tends to learn a target function from low to high frequencies during the training.* See (c,d), red: small relative error.

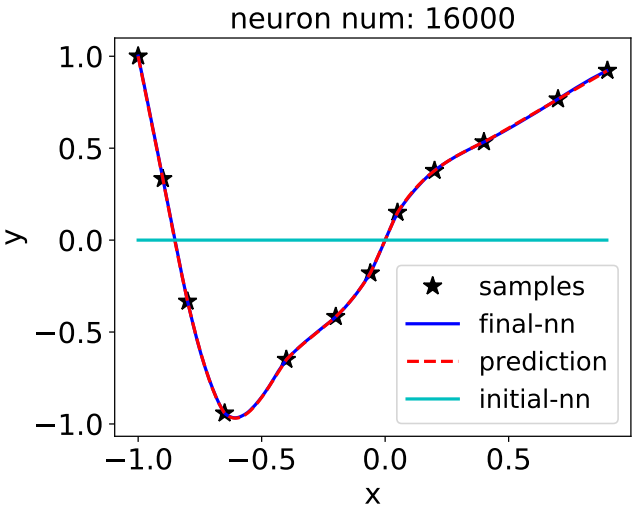
Xu, Zhang, Luo, Xiao, Ma, *Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks*, 1901.06523, 2019

Linear F-Principle Dynamics

$h(x,\boldsymbol{\theta}) = \sum_{i=1}^{m} a_i \mathrm{ReLU}\left(w_i(x+b_i)\right)$ is hypothesis function, $f(x)$ is target function, and $u(x) = h(x,\boldsymbol{\theta}) - f(x)$. In the kernel regime, we can get the following dynamics of u in the frequency domain,

$$\partial_t \hat{u}(\xi,t) = -\left(\frac{\|\boldsymbol{a}\|^2 + \|\boldsymbol{w}\|^2}{|\xi|^4} + 4\pi\frac{\|\boldsymbol{a}\odot\boldsymbol{w}\|^2}{|\xi|^2}\right)\hat{u}_\rho(\xi,t).$$

Here $\odot$ is the Hadamard product, $u_\rho = u\rho$, ρ is the sample distribution.



Zhang, Xu, Luo, Ma, *Explicitizing an Implicit Bias of the Frequency Principle in Two-layer Neural Networks*, 1905.10264, 2019

Outline

Concluding remarks

Analogy with Monte Carlo

$$(I(g) - I_n(g))^2 \sim \frac{\gamma(g)}{n}$$

$\gamma(g) =$ some kind of variance of g, depending on the details of the Monte Carlo algorithm.

For the ML models considered

$$\mathcal{R}(\theta) \leq \frac{\gamma_1(f^*)}{m} + \frac{\gamma_2(f^*)}{\sqrt{n}}$$

Papers can be found on my webpage.

MSML 2020

A new NIPS or ICML style annual conference, which also serves as a venue for publications: 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