Towards a Mathematical Understanding of Supervised Learning: what we know and what we don't know

Weinan E

Princeton University

Joint work with:

Chao Ma, Stephan Wojtowytsch, Lei Wu

www.math.princeton.edu/~weinan

What constitutes a mathematical understanding of ML?

ML is both very powerful and very subtle.

Our general goal:

- understand the reasons behind
- propose better (more robust, more flexible) machine learning models

Strategies:

- theorems
- careful numerical experiments
- simple model problems to gain insight

Purpose of this talk:

- For regression problems, such an understanding is starting to take some shape.
- What are the crucial missing pieces of the puzzle?

Outline

Outline

Basic problem of supervised learning

Given $S = \{(\boldsymbol{x}_j, y_j = f^*(\boldsymbol{x}_j)), j \in [n]\}$, learn f^*.

1. This is a problem about function approximation.
2. Based on finite pieces of "labeled" data
   - regression (f^* is continuous) vs classification (f^* is discrete)
   - will neglect measurement noise (not crucial for the talk)
   - assume $\boldsymbol{x}_j \in X = [0,1]^d$. d is typically quite large
   - notation: $\mu =$ the distribution of $\{\boldsymbol{x}_j\}$

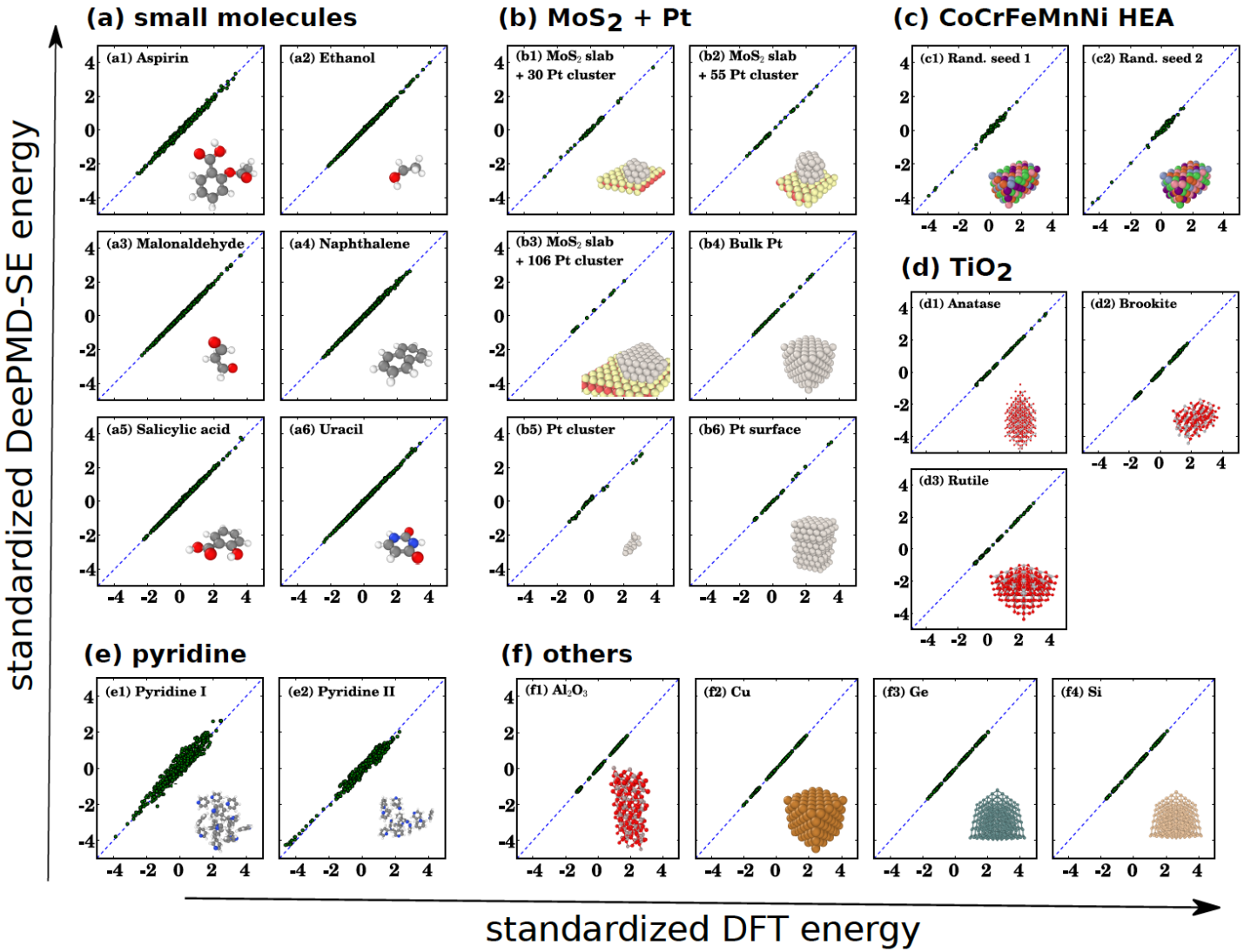
In practice, one divides S into two subsets, a training set and a testing set.

Will focus on the regression problem

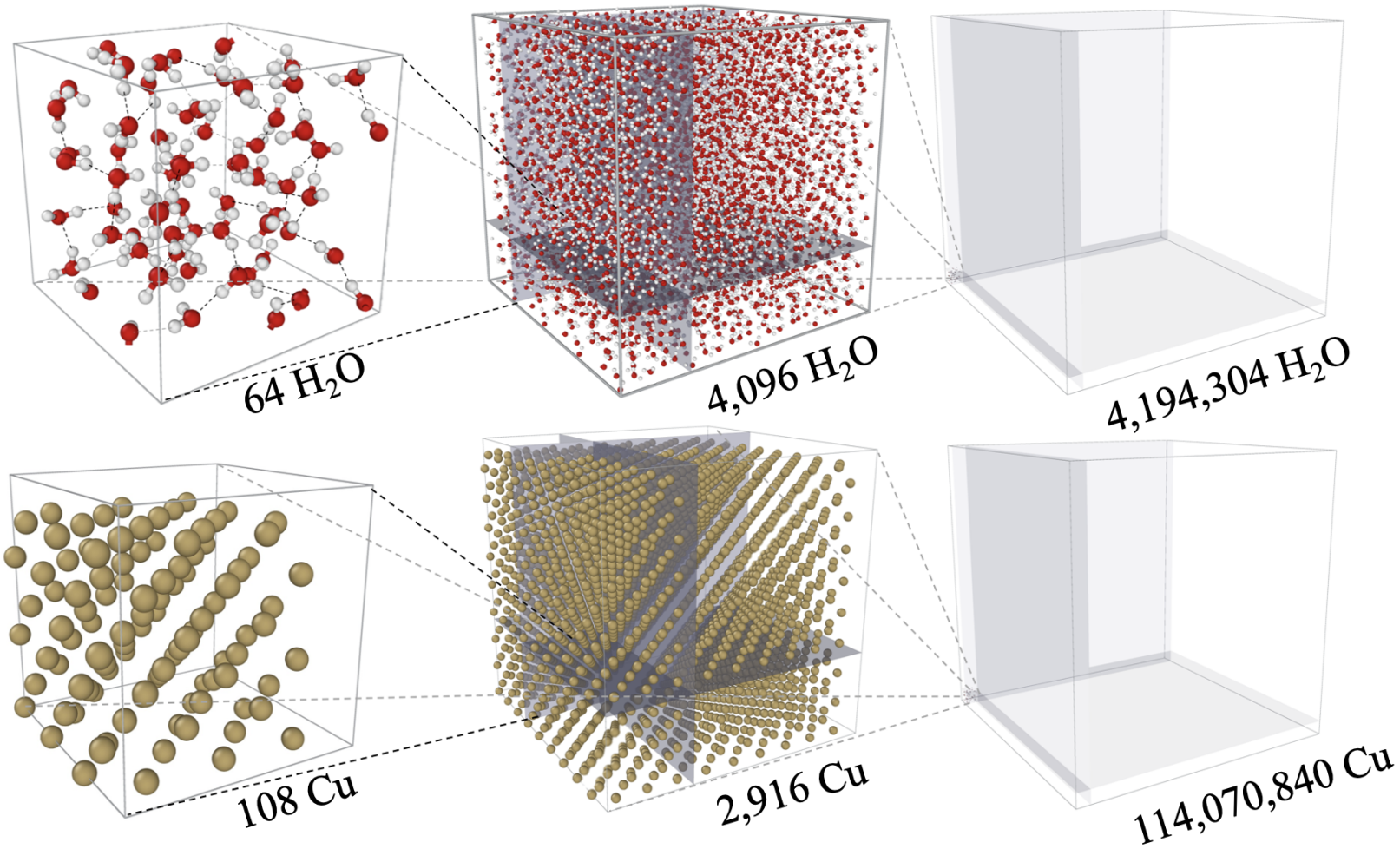
DeePMD: Inter-atomic potential in molecular dynamics

http://www.deepmd.org/database/deeppot-se-data/ (Linfeng Zhang et al)

$$V = V(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$$



DeePMD simulation of 100M atoms



(a) AIMD + HPC;

(b) DeePMD+1 GPU @ Home;

(c) DeePMD+27360 GPUs @ Summit.

D. Lu, et al, arXiv: 2004.11658; W. Jia, et al, arXiv: 2005.00223

Main puzzles

- Why is it so successful?
  - computer vision and natural language processing
  - reinforcement learning (games, robots)
  - molecular dynamics
  - solving high ($100 \sim 10,000$) dimensional partial differential equations

  These are VERY high dimensional problems.
- Why is it so fragile?
  - choice of network architecture (in particular, more layers seem to perform better but too many layers is not good either)
  - training process can be problematic (spikes, plateaus, oscillations, sensitive dependence on choice of algorithm, initialization, learning rates)
  - difference between training and testing accuracies – the generalization gap (sometimes big, sometimes small)

Standard procedure for supervised learning

1. choose a hypothesis space (set of trial functions) $\mathcal{H}_m$ ($m \sim \dim(\mathcal{H}_m)$)
   - (piecewise) polynomials, wavelets, ...
   - neural network models
2. choose a loss function (to fit the data)
   - "empirical risk"
   $$\hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j) - y_j)^2 = \frac{1}{n}\sum_j (f(\boldsymbol{x}_j) - f^*(\boldsymbol{x}_j))^2$$
   - add regularization
3. choose an optimization algorithm and parameters
   - gradient descent (GD), stochastic gradient descent (SGD), ADAM, RMSprop, ...
   - hyperparameters (initialization, step size=learning rate, ...)

Objective: Minimize the "population risk" (also known as the "generalization error")

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

Important parameters: m, n, t, d (typically interested in the case: $m, n, t \to \infty, d \gg 1$).

Classical numerical analysis vs machine learning

- trial functions (hypothesis space)
  - splines: C^1 piecewise cubic polynomials
  - machine learning: neural network functions $= \{\sum a_j \sigma(\boldsymbol{b}_j^T \boldsymbol{x})\}$
- variational problem to be solved (loss function)
  - splines $I_n(f) = \frac{1}{n}\sum_{j=1}^n (f(x_j) - y_j)^2 + \lambda \int |D^2 f(x)|^2 dx$
  - machine learning: $I_n(f) = \frac{1}{n}\sum_{j=1}^n (f(\boldsymbol{x}_j) - y_j)^2 +$ regularization
- optimization algorithm (training)
  - splines: conjugate gradient
  - machine learning: gradient descent, stochastic gradient descent

Key differences:
(1) sheer size of the problem;
(2) importance of nonlinearity

Classical theory

- Error estimates (a priori and a posteriori)

$$\|f^* - f_m\|_{L^2(X)} \leq Cm^{-\alpha/d}\|f^*\|_{H^\alpha(X)}, \quad \|f^* - f_m\|_{L^2(X)} \leq Cm^{-\alpha/d}\|f_m\|_h$$

  $H^\alpha(X)$ is α-th order Sobolev space over the domain X
  Curse of dimensionality (CoD) !

- Optimization (basically linear):

$$\|\theta_k - \theta^*\| \leq C\left(1 - \frac{1}{\kappa}\right)^k \|\theta_0 - \theta^*\|$$

  where κ is the condition number.

Benchmark: High dimensional integration

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

Grid-based quadrature rules:

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

Appearance of $1/d$ in the exponent of m: **CoD**!

Monte Carlo: $\{\boldsymbol{x}_j, j \in [m]\}$ is uniformly distributed in X.

$$\mathbb{E}(I(g) - I_m(g))^2 = \frac{\mathsf{var}(g)}{m}, \quad \mathsf{var}(g) = \int_X g^2(\boldsymbol{x}) d\boldsymbol{x} - \left(\int_X g(\boldsymbol{x}) d\boldsymbol{x} \right)^2$$

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

However, $\mathsf{var}(g)$ can be very large in high dimension. Variance reduction!

Approximation and estimation errors

Want to understand $f^* - \hat{f}$, where $\hat{f}$ = the output of the ML model.

f_m = argmin $_{f \in \mathcal{H}_m} \mathcal{R}(f)$ = "best approx" to f^* in $\mathcal{H}_m$.

Decomposition of the error:

$$f^* - \hat{f} = f^* - f_m + f_m - \hat{f}$$

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

Estimation error in classical numerical analysis: Runge phenomenon

Since we can only work with a finite dataset, what happens outside of the dataset?

Example: Polynomial interpolation on equally spaced grid points

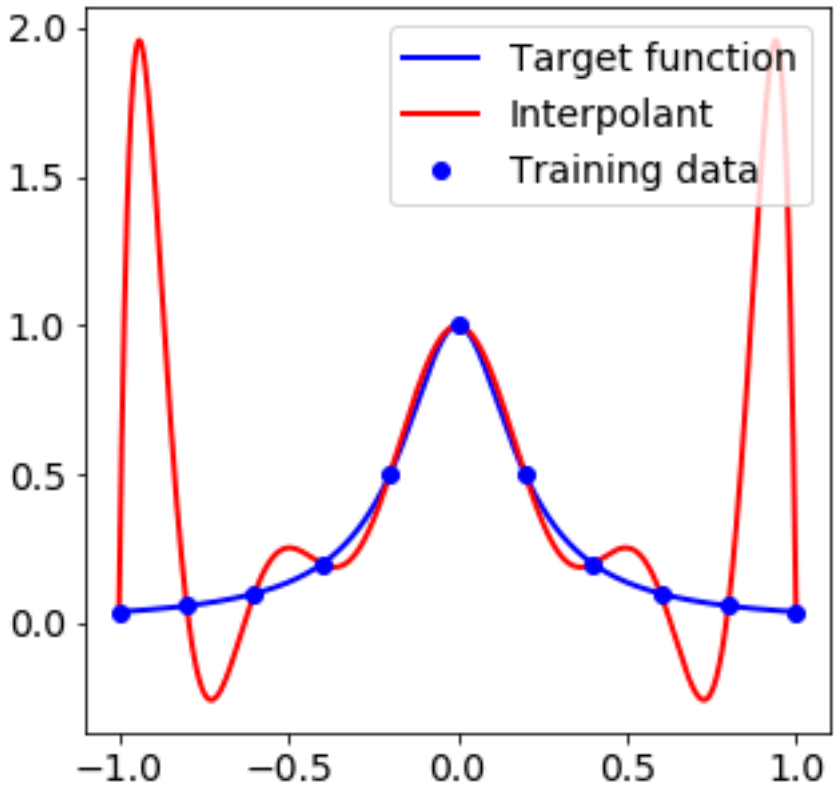


Figure: The Runge phenomenon: $f^*(x) = \frac{1}{1+25x^2}$

Generalization gap = difference between training and testing errors

$$\text{"Generalization gap"} = |\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}_n(\hat{f})| = |I(g) - I_n(g)|$$

$$\mathcal{R}(f) = \int_{\mathbb{R}^d} (f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 d\mu, \quad \hat{\mathcal{R}}_n(f) = \frac{1}{n} \sum_j (f(\boldsymbol{x}_j) - f^*(\boldsymbol{x}_j))^2$$

$$I(g) = \int g(\boldsymbol{x}) d\mu, \quad I_n(g) = \frac{1}{n} \sum_j g(\boldsymbol{x}_j), \quad g(\boldsymbol{x}) = (\hat{f}(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$$

Difficulty: $\hat{f}$ is highly correlated with $\{\boldsymbol{x}_j\}$.

Should NOT expect: generalization gap $= O(1/\sqrt{n})$ to hold automatically.

If hypothesis space = space of all Lipschitz-1 functions:

$$\text{generalization gap} \sim \frac{1}{n^{1/d}}$$

This gives rise to **CoD for the size of the dataset**.

Best that we can hope for: generalization gap $\lesssim \frac{\|\theta\|_*}{\sqrt{n}}$

- Does this hold?
- Identify the corresponding norm, which should depend on the particular machine learning model
- Is this norm controlled during training dynamics?

The exponent $1/2$ might be improved to $1/2 + O(1/d)$ using things like "local Rademacher complexity" (see work of Bartlett et al).

CoD in training dynamics

Theorem (Wojtowytsch and E (2020))

Let σ be a Lipschitz-continuous activation function. Consider the risk functional

$$\mathcal{R}\big((a_i)_{i=1}^m, (\boldsymbol{w}_i)_{i=1}^m\big) = \frac{1}{2}\int_X (f_{a,w} - f^*)^2(\boldsymbol{x})d\boldsymbol{x}$$

where

$$f_{a,w} = \frac{1}{m}\sum_{i=1}^m a_i\,\sigma(\boldsymbol{w}_i^T\boldsymbol{x})$$

and f^* is a Lipschitz-continuous target function. There exists f^* with Lipschitz constant and L^∞-norm bounded by 1 such that parameters $a_i, \boldsymbol{w}_i$ evolving by the gradient flow of $\mathcal{R}$ satisfy

$$\limsup_{t\to\infty}\Big[t^\gamma\,\mathcal{R}\big((a_i(t))_{i=1}^m, (\boldsymbol{w}_i(t))_{i=1}^m\big)\Big] = \infty$$

for all $\gamma > \frac{4}{d-2}$. The result is independent of the network width m and holds also for infinitely wide networks (where the parameter distributions evolve by a Wasserstein gradient flow).

Intuitively: $\mathcal{R}\big((a_i(t))_{i=1}^m, (\boldsymbol{w}_i(t))_{i=1}^m\big) \geq c\,t^{-\frac{4}{d-2}}$ for large t. **CoD!**

Outline

General procedure

Given a machine learning model:

- Approximation error:

$$\inf_{f\in\mathcal{H}_m} \mathcal{R}(f) = \inf_{f\in\mathcal{H}_m} \|f - f^*\|^2_{L^2(d\mu)} \lesssim \frac{\|f^*\|^2_*}{m}$$

  Identify $\|f^*\|_*$.

  - Direct and inverse approximation theorems

- Estimation error (generalization gap): $\mathcal{H}_Q = \{f, \|f\|_* \leq Q\}$. Then we have

$$\mathrm{Rad}_S(\mathcal{H}_Q) \lesssim \frac{Q}{\sqrt{n}}$$

Combined: Up to log terms, we have

$$\mathcal{R}(\hat{f}) \lesssim \frac{\|f^*\|^2_*}{m} + \frac{\|f^*\|_*}{\sqrt{n}}$$

Two-layer neural network model: Barron spaces

E, Ma and Wu (2018, 2019) [i]

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

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

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

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

$$\|f\|_{\mathcal{B}} = \inf_{\rho \in P_f} \left(\mathbb{E}_\rho[a^2 \|\boldsymbol{w}\|_1^2]\right)^{1/2}$$

where $P_f := \{\rho : f(\boldsymbol{x}) = \mathbb{E}_\rho[a\sigma(\boldsymbol{w}^T\boldsymbol{x})]\}$.

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

[i] Related work in Barron (1993), Klusowski and Barron (2016), Bach (2017), E and Wojtowytsch (2020)

Theorem (Direct Approximation Theorem)

There exists an absolute constant C_0 such that

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

Theorem (Inverse Approximation Theorem)

Let

$$\mathcal{N}_C \stackrel{def}{=} \{ \frac{1}{m} \sum_{k=1}^{m} a_k \sigma(\boldsymbol{w}_k^T \boldsymbol{x}) : \frac{1}{m} \sum_{k=1}^{m} |a_k|^2 \|\boldsymbol{w}_k\|_1^2 \leq C^2, m \in \mathbb{N}^+ \}.$$

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

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

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

Complexity estimates

Theorem (Bach, 2017)

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

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

A priori estimates for regularized model

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

where the path norm is defined by:

$$\|\theta\|_{\mathcal{P}} = \left(\frac{1}{m}\sum_{k=1}^{m}|a_k|^2\|\boldsymbol{w}_k\|_1^2\right)^{1/2}$$

Theorem (E, Ma, Wu, 2018)

Assume that the target function $f^* : X \mapsto [0,1] \in \mathcal{B}$. 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

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

Typical results in ML literature: a posteriori estimates

$$\mathcal{R}(\hat{\theta}_n) - \hat{\mathcal{R}}_n(\hat{\theta}_n) \lesssim \frac{\|\hat{\theta}_n\|}{\sqrt{n}}$$

Other models

1. Random feature model

$\{\phi(\cdot;\boldsymbol{w})\}$: collection of random features, e.g. $\phi(\boldsymbol{x},\boldsymbol{w}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$.
π: prob distribution of the random variable $\boldsymbol{w}$.

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

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

Corresponding function space: reproducing kernel Hilbert space (RKHS) with kernel:

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

2. ResNets

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

Corresponding function space: "flow-induced function space" (E, Ma and Wu (2019))
Regularized loss function:

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

Up to logarithmic terms, we have:

$$\mathcal{R}(\hat{\theta}) \lesssim \frac{\|f^*\|_{\mathcal{D}_2}^2}{L} + \lambda\frac{\|f^*\|_{\mathcal{D}_1}^2}{\sqrt{n}}$$

Variance reduction:

$$\|f^*\|_{\mathcal{D}} \leq \|f^*\|_{\mathcal{B}} \leq \|f^*\|_{\mathcal{H}}$$

3. Multilayer networks

$$f(x) = \sum_{i_L=1}^{m_L} a_{i_L}^L \sigma \left(\sum_{i_{L-1}=1}^{m_{L-1}} a_{i_L i_{L-1}}^{L-1} \sigma \left(\dots \sigma \left(\sum_{i_1=1}^{m_1} a_{i_2 i_1}^1 \, \sigma \left(\sum_{i_0=1}^{d+1} a_{i_1 i_0}^0 \, x_{i_0} \right) \right) \right) \right)$$

Corresponding function space: "multilayer space" (E and Wojtowytsch (in preparation))

- Rademacher complexity/generalization gap: Same as for Barron space. Let $\mathcal{F}_Q = \{f \in C^0, \|f\|_{\text{multi-layer}} \leq Q\}$. Then $\mathrm{Rad}_S(\mathcal{F}_Q) \leq 2Q\sqrt{\frac{2\ln(2d+2)}{n}}$
- Inverse approximation theorem holds.
- Direct approximation theorem: holds, but not with Monte-Carlo rate. For f^* in multi-layer space and $m \in \mathbb{N}$, there exists a network f with layers of width $m_\ell = m^{L-\ell+1}$ such that

$$\|f - f^*\|_{L^2(\mathbb{P})} \leq \frac{2^L \, \|f^*\|_{\text{multi-layer}}}{\sqrt{m}}.$$

  Note that the network has $O(m^{2L-1})$ parameters! (but dimension-independent)

Outline

Issues

- The optimization problem: Can we make the training error small? How fast?
  The loss function is non-convex.
  Possible CoD in time?
- The generalization problem: Can we make the testing error small?
  Global optimum is non-unique
  Cooper: Global minimum forms a manifold of dimension $m - n$
  Which one is picked? Does it generalize well?
  Implicit regularization?

Resonance and slow deterioration [a]

[a] Ma, Wu and E (MSML (2020))

Random feature model:

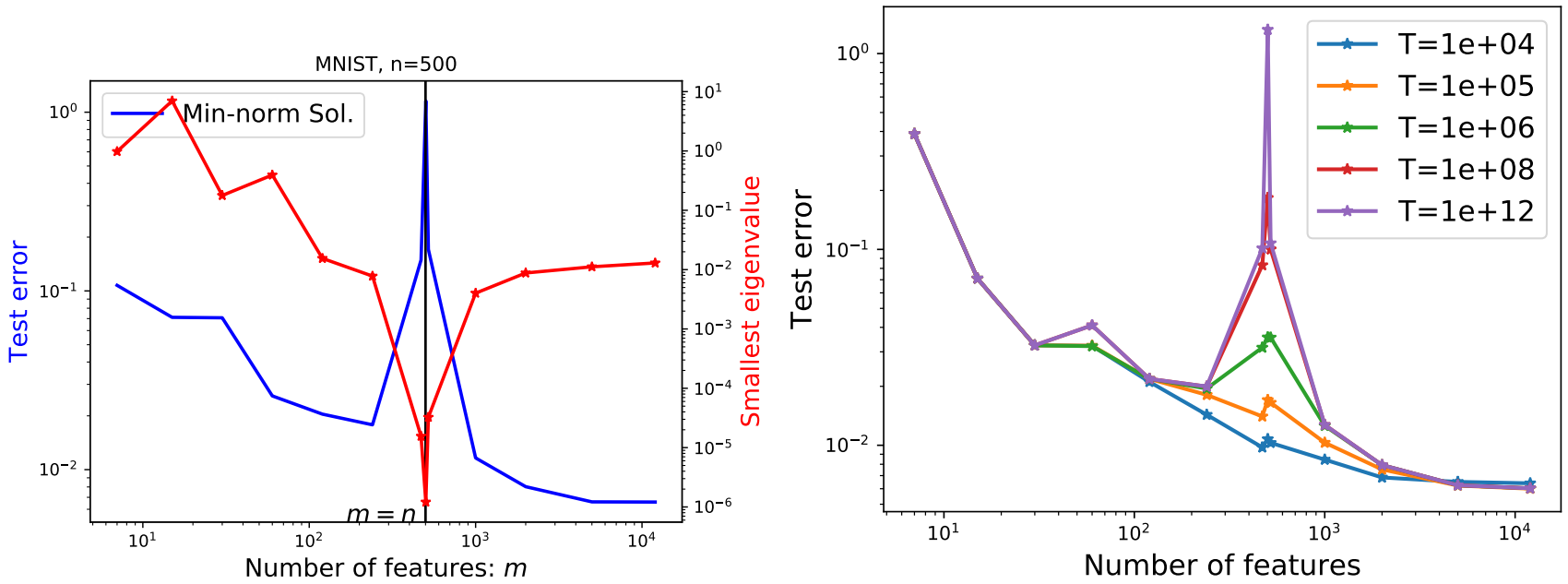


Figure: **Left:** Test error as a function of m for $n = 500$. **Right:** The test error of GD solutions obtained by running different number of iterations.

The "double descent" phenomenon (Saxe, Belkin et al)

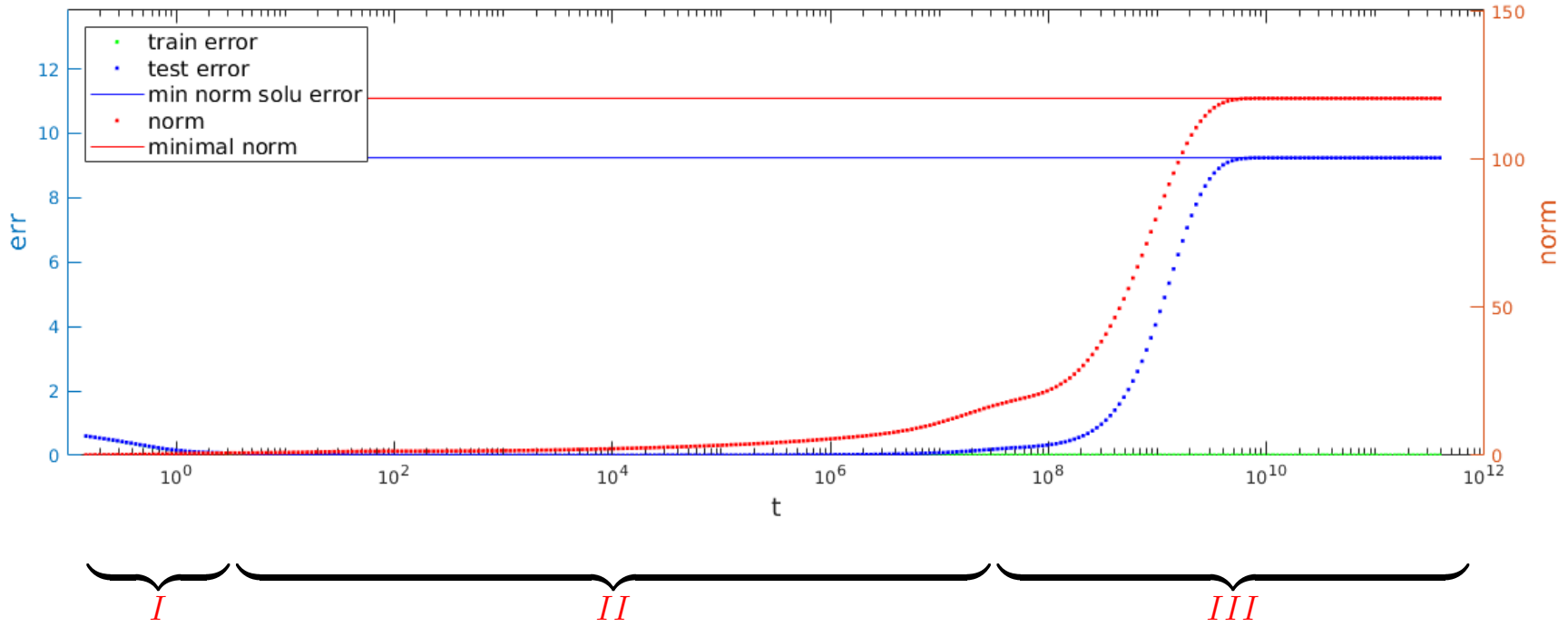
The large test error is caused by very small eigenvalues of the Gram matrix, which also lead to slow convergence.

Theorem (Ma, Wu and E (MSML(2020)))

Let $\Phi = \sigma(X^T B)$, $\{\lambda_i\}$ *be the singular values of* Φ, *and* $\hat{\lambda}_i = \frac{\lambda_i}{n}$. *Let* $f^*(\boldsymbol{x}) = \psi_1(\boldsymbol{x})$, *the first eigenfunction of the kernel operator. There exists constant* C *s.t., for any* $\delta > 0$, *with prob no less than* $1 - \delta$,

$$\|\hat{f}_t - f^*\| \leq e^{-\hat{\lambda}_1^2 t} + C\left(1 + \log\frac{1}{\delta}\right)(1 + Md(t))\, n^{-\frac{1}{4}},$$

where $M = \frac{1}{m}\int \|\sigma(B^T\boldsymbol{x})\|^2 \pi(d\boldsymbol{x})$, *and* $d(t) = \min\left\{\sqrt{t}, \hat{\lambda}_{\lfloor\sqrt{n}\rfloor+1} t, \hat{\lambda}_n^{-1}\right\}$.



Training two-layer neural networks

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

Xavier-like initialization:

$$a_j(0) \sim \mathcal{N}(0, \beta^2), \qquad \boldsymbol{b}_j(0) \sim \mathcal{N}(0, I/d)$$

$\beta = 0, = 1/\sqrt{m}$

The associated random feature model: $\{\boldsymbol{b}_j\}$ frozen, only train $\{a_i\}$
Define Gram matrix $K = (K_{ij})$:

$$K_{i,j} = \frac{1}{n} \mathbb{E}_{\boldsymbol{b} \sim \pi_0} [\sigma(\boldsymbol{x}_i^T \boldsymbol{b}) \sigma(\boldsymbol{x}_j^T \boldsymbol{b})].$$

Highly over-parametrized 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

Let $\lambda_n = \lambda_{\min}(K)$ *and assume* $\beta = 0$. *Denote by* $f_m(\boldsymbol{x}; \tilde{\boldsymbol{a}}(t), \boldsymbol{B}_0))$ *the solutions of GD 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*

$$\hat{\mathcal{R}}_n(\boldsymbol{a}(t), \boldsymbol{B}(t)) \leq e^{-m\lambda_n t} \hat{\mathcal{R}}_n(\boldsymbol{a}(0), \boldsymbol{B}(0)) \tag{1}$$

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

In particular, there is no "implicit regularization"!

Observation: Time scale separation

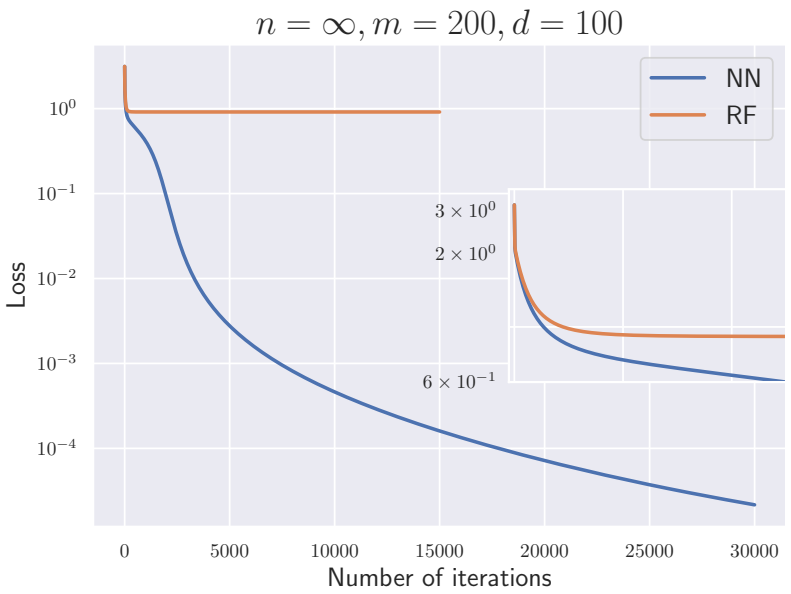
$$\dot{a}_j(t) \sim O(\|\boldsymbol{b}_j\|) = O(1) \tag{3}$$

$$\dot{\boldsymbol{b}}_j(t) \sim O(|a_j|) = O\left(\frac{1}{\lambda_n m}\right) \tag{4}$$

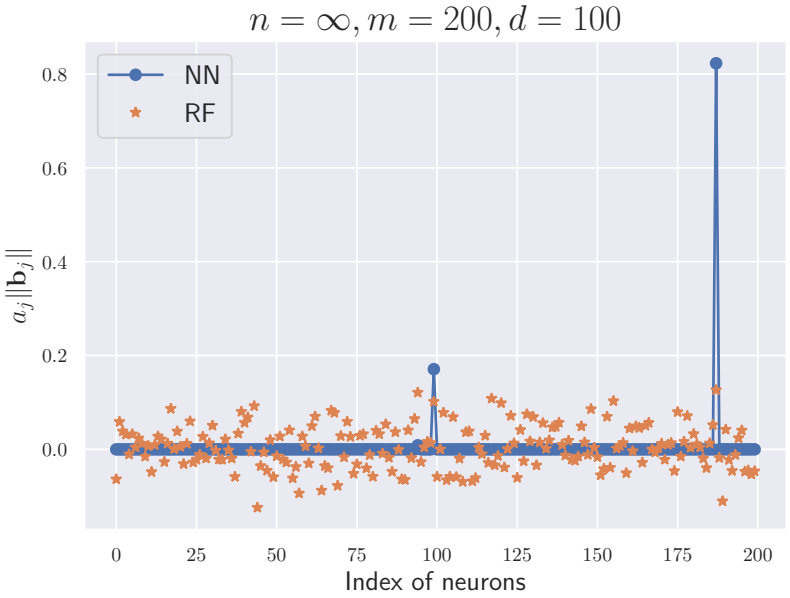
The dynamics of $\boldsymbol{b}$ is effectively frozen.

An example: The single neuron target function

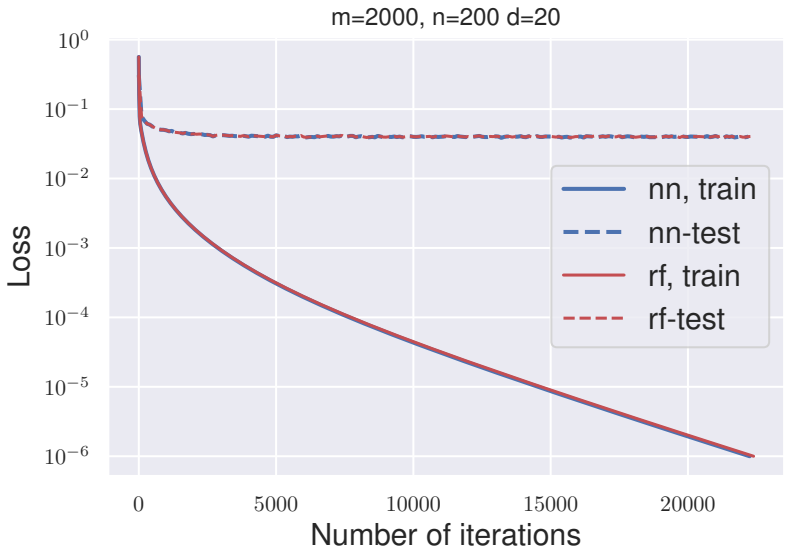
$f_1^*(\boldsymbol{x}) = \sigma(\boldsymbol{b}^* \cdot \boldsymbol{x}), \quad \boldsymbol{b}^* = \boldsymbol{e}_1$. See Ma, Wu and E (2020) for more results.

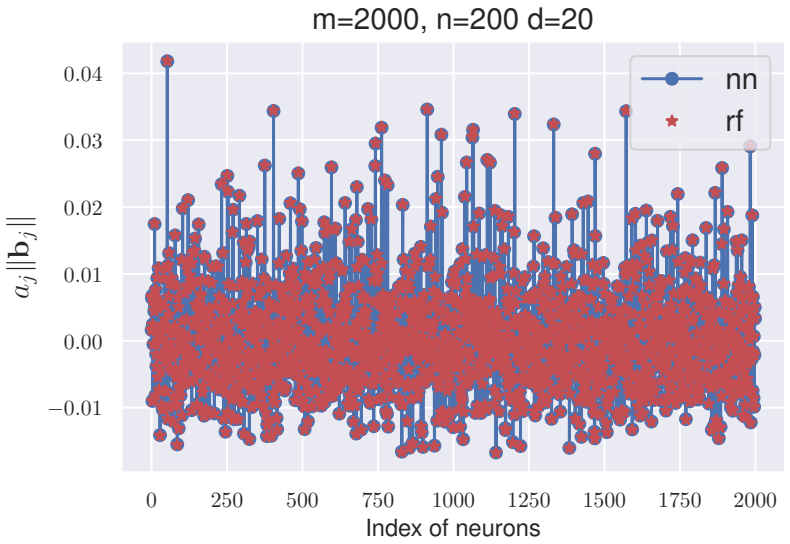


(a)



(b)





Neural network-like vs. random feature-like behavior

Consider target function which can be accurately approximated by a small number of neurons (effectively "over-parametrized")

- Neural network-like behavior
  - two phases: initial random feature-like phase, followed by a second phase with quenching and activation
    Neurons are divided into two groups: activated ones and background neurons
  - testing error continues to decay after the first phase
- Random feature-like behavior
  - simpler dynamics (e.g. no division into two groups)
  - testing error saturates quickly while training error continues to decay

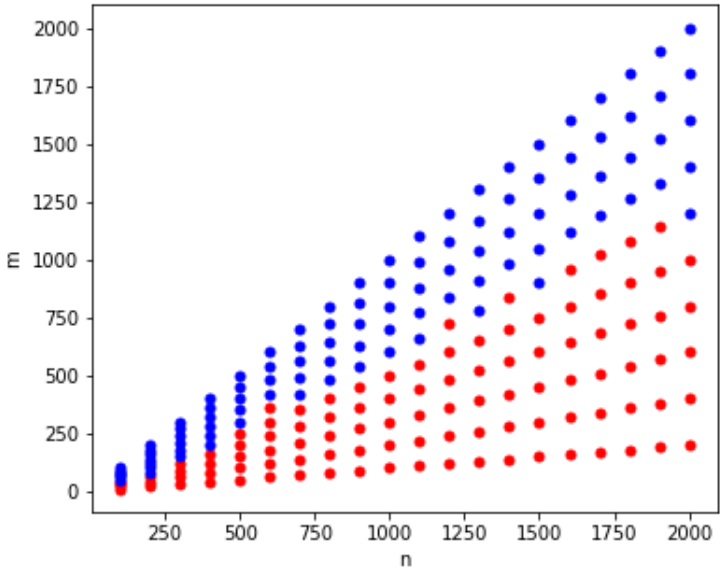
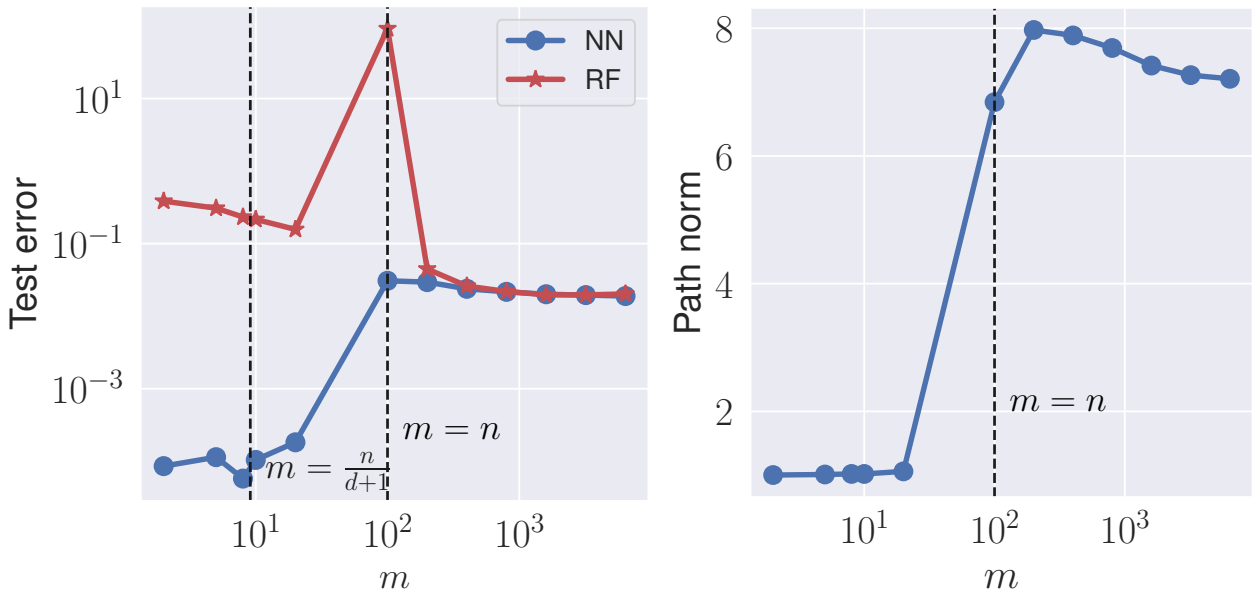


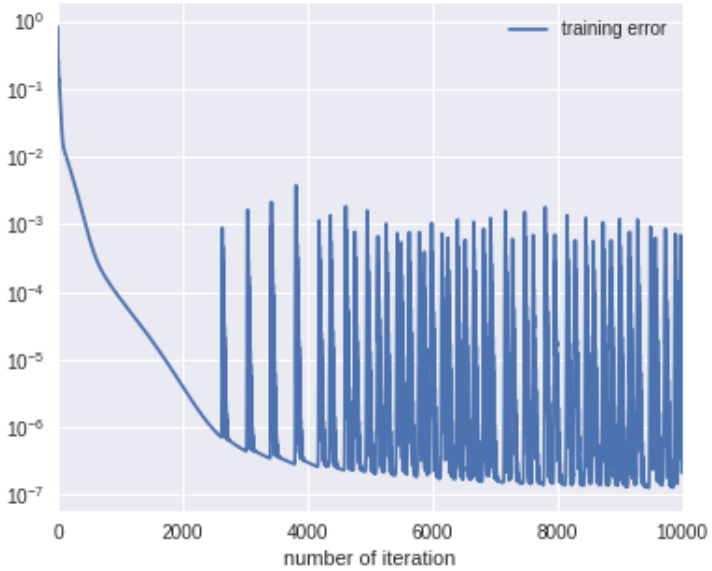
Figure: The red and blue dots represent the NN-like and RF-like behavior.

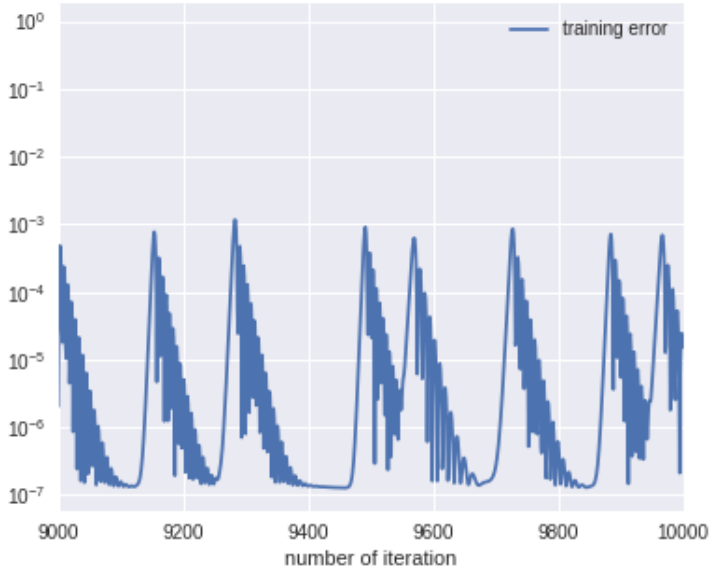


The different regimes of Adam

Adam algorithm and its ODE limit with $\eta \to 0$, $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$,

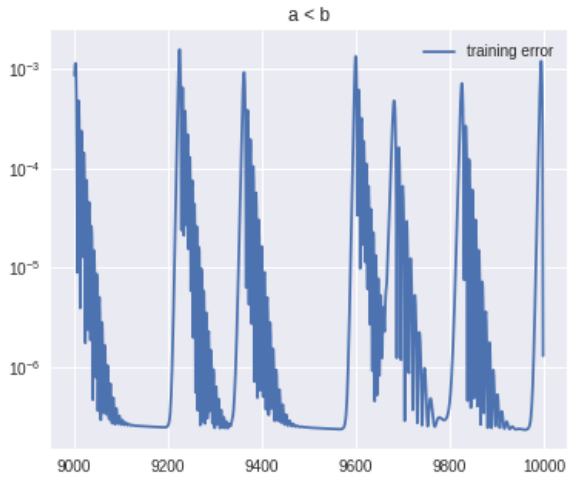
$$\begin{aligned} v_t &= \alpha v_{t-1} + (1-\alpha) g_t^2 \\ m_t &= \beta m_{t-1} + (1-\beta) g_t \\ \theta_{t+1} &= \theta_t - \eta \frac{m_t/(1-\beta^t)}{\sqrt{v_t/(1-\alpha^t)} + \epsilon} \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{v} &= a(\nabla F(x)^2 - v) \\ \dot{m} &= b(\nabla F(x) - m) \\ \dot{x} &= -\frac{(1-e^{-bt})^{-1} m}{\sqrt{(1-e^{-at})^{-1} v} + \epsilon} \end{aligned}$$

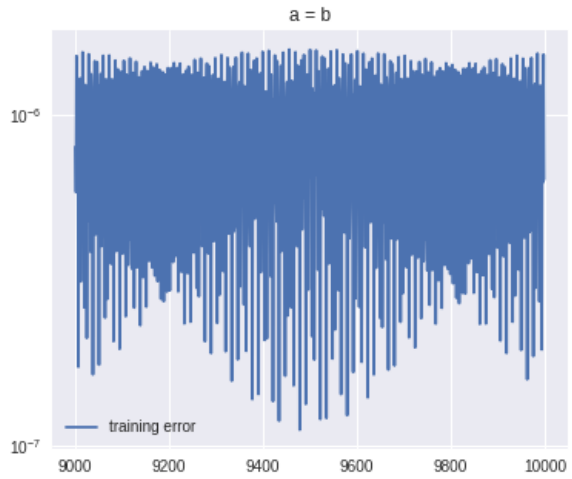


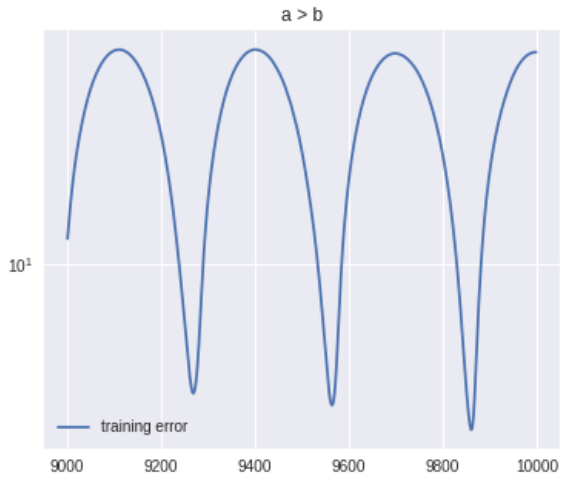


- Fast initial convergence; Spikes; Small oscillations

The influence of a and b







- $a < b$: spikes
- $a \approx b$: small, fast oscillations
- $a > b$: large, slow oscillations; large training error

The influence of a and b

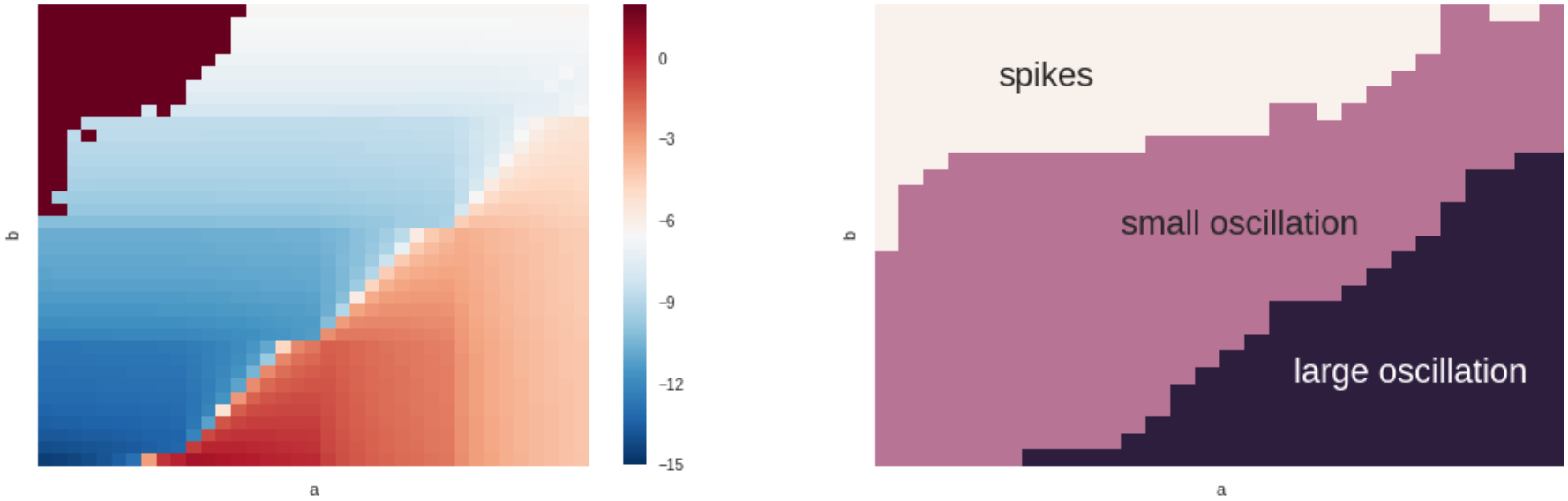


Figure: **(Left)** Average training loss of the last 1000 iterations in logarithm scale. a and b range from 0.01 to 100 in logarithm scale. **(Right)** The classification of the three categories.

GD and SGD select different global minima

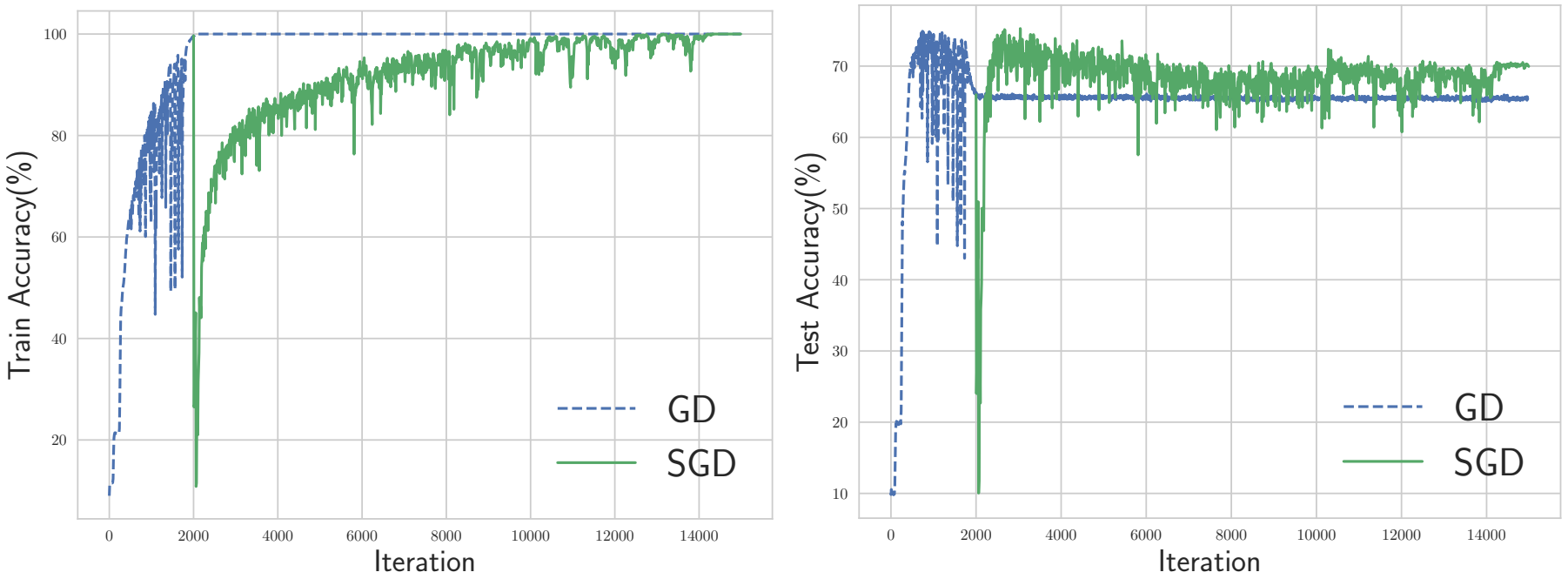


Figure: Escape phenomenon in fitting corrupted FashionMNIST. This escape phenomenon shows that the GD solutions are unstable for SGD dynamics (Wu, Ma and E (2018)).

- GD and SGD converge towards different solutions.
- This is because that their stability conditions are different: SGD requires more stringent stability conditions

Other subtleties (or black magic)

Other things that make deep learning TRICKY:

- batch normalization
- dropout
- performance of particular network architecture

  In particular, the advantage and disadvantage of deep (or multi-layer) neural network models.

Outline

Advantage of deeper network models: Approximation property

Theorem (E and Wojtowytsch (2020))

Let $f \in \mathcal{B}(X)$. We can decompose $f = \sum_{i=0}^{\infty} f_i$ in such a way that f_0 is C^1-smooth and the singular set Σ_i of f_i is a k_i-dimensional affine subspace of $\mathbb{R}^d$ for some $0 \leq k_i \leq d-1$.

Barron functions cannot have **curved** singular sets of co-dimension 1.

$$f_1(x) = \text{dist}(x, S^{d-1}), f_2(x) = \text{dist}(x, B_1(0)) \notin \mathcal{B}(X)$$

Both functions are in three-layer space because

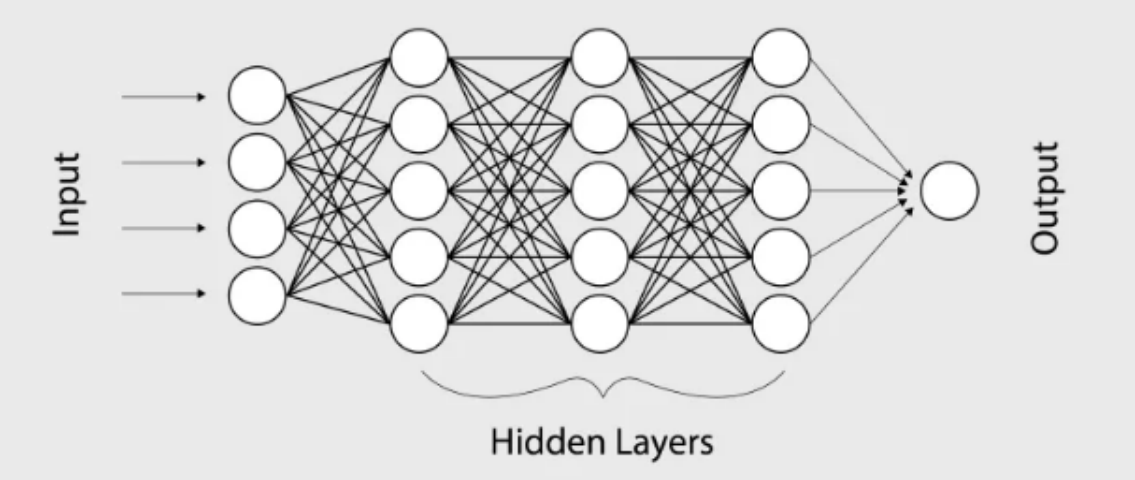
$$f_1 = g_1 \circ g_2, \quad f_2 = \tilde{g}_1 \circ g_2, \qquad g_2(x) = |x| \text{ and } g_1(z) = |z-1|, \;\; \tilde{g}_1(z) = \sigma(z-1).$$

g_2 is in Barron space with norm $\sim \sqrt{d}$.

Advantage of deeper network models: Generalization property

More subtle than expected.

Disadvantage of deeper network models: Training



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

σ is a scalar function (the activation function), e.g. $\sigma(x) = \max(x, 0)$, ReLU

Numerical instability: Exploding gradients (see Hanin (2018))

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

Outline

Looking for "well-posed" formulations of machine learning

Start with a "nice" continuous problem and then discretize to get practical ML algorithms

- representation of functions: integral transform representation and flow-based representation
- the variational problem for minimizing the population risk (loss function)
- gradient flow for the variational problem (training, a PDE-like problem)

Key: The variational problem should be "nice".

Similar philosophy in image processing

Example: Denoising

- constructing various special purpose filters, e.g. wavelet-based, curvelet-based
- start with a continuous mathematical problems (Mumford-Shah, Rudin-Osher-Fatemi), and then discretize and optimize

$$I(u) = \int_\Omega ((u-f)^2 + \lambda|\nabla u|)d\boldsymbol{x}$$

f =original image, u= denoised image.

Representing functions: An illustrative example

Traditional approach:

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})} d\boldsymbol{\omega}, \quad f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j a(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j,\boldsymbol{x})}$$

$\{\boldsymbol{\omega}_j\}$ is a fixed grid, e.g. uniform.

$$\|f - f_m\|_{L^2(X)} \leq C_0 m^{-\alpha/d} \|f\|_{H^\alpha(X)}$$

"New" approach:

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

$\pi =$ probability measure on $\mathbb{R}^d$. Let $\{\boldsymbol{\omega}_j\}$ be an i.i.d. sample of π.

$$\mathbb{E}|f(\boldsymbol{x}) - \frac{1}{m} \sum_{j=1}^m a(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j,\boldsymbol{x})}|^2 = \frac{\mathsf{var}(f)}{m}$$

where

$$\mathsf{var}(f) = \mathbb{E}_{\boldsymbol{\omega}\sim\pi} |a(\boldsymbol{\omega})|^2 - f(\boldsymbol{x})^2$$

$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})} \pi(d\boldsymbol{\omega})$: continuous version of two-layer neural network function with activation function σ defined by $\sigma(z) = e^{iz}$.

Integral transform-based representation and the variational problem

Consider the (parametric) representation:

$$f(\boldsymbol{x},\theta)=\int_{\mathbb{R}^d} a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})\pi(d\boldsymbol{w})=\mathbb{E}_{\boldsymbol{w}\sim\pi}a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

$\theta=$ parameter: $\theta=\{a(\cdot)\}$ or $\theta=\{a(\cdot),\pi(\cdot)\}$.
Equivalently:

$$f(\boldsymbol{x},\theta)=\mathbb{E}_{(a,\boldsymbol{w})\sim\rho}a\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

e.g. $\rho(da,d\boldsymbol{w})=\pi(d\boldsymbol{w})\delta(a-a(\boldsymbol{w}))da$. In this case, $\theta=\rho(\cdot)$.

Given a target function f^*, the variational problem for minimizing the population risk becomes: $\min\mathcal{R}$ where

$$\min_{\theta}\mathcal{R},\quad \mathcal{R}(\theta)=\mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x},\theta)-f^*(\boldsymbol{x}))^2$$

Conjecture: This is a "nice (convex-like)" variational problem.

The continuous formulation: Gradient flows

Recall the population risk: $\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 =$ "free energy"

$$f(\boldsymbol{x}) = \int a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})\pi(d\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{w}\sim\pi}a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

Follow Halperin and Hohenberg (1977)

- $a =$ non-conserved, use "model A" dynamics (Allen-Cahn):

$$\frac{\partial a}{\partial t} = -\frac{\delta \mathcal{R}}{\delta a}$$

- $\pi =$ conserved (probability density), use "model B" (Cahn-Hilliard):

$$\frac{\partial \pi}{\partial t} + \nabla \cdot \mathbf{J} = 0$$

$$\mathbf{J} = \pi \boldsymbol{v},\ \boldsymbol{v} = -\nabla V,\ V = \frac{\delta \mathcal{R}}{\delta \pi}.$$

Gradient flow for the feature-based model

Fix π, optimize over a.

$$\partial_t a(\boldsymbol{w}, t) = -\frac{\delta \mathcal{R}}{\delta a}(\boldsymbol{w}, t) = -\int a(\tilde{\boldsymbol{w}}, t) K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) \pi(d\tilde{\boldsymbol{w}}) + \tilde{f}(\boldsymbol{w})$$

$$K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) = \mathbb{E}_{\boldsymbol{x}}[\sigma(\boldsymbol{w}^T \boldsymbol{x}) \sigma(\tilde{\boldsymbol{w}}^T \boldsymbol{x})], \quad \tilde{f}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}}[f^*(\boldsymbol{x}) \sigma(\boldsymbol{w}^T \boldsymbol{x})]$$

This is an integral equation with a symmetric positive definite kernel.

Decay estimates due to convexity: Let $f^*(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{w} \sim \pi} a^*(\boldsymbol{w}) \sigma(\boldsymbol{w}^T \boldsymbol{x})$,

$$I(t) = \frac{1}{2} \|a(\cdot, t) - a^*(\cdot)\|^2 + t(\mathcal{R}(a(t)) - \mathcal{R}(a^*))$$

Then we have

$$\frac{dI}{dt} \leq 0, \quad \mathcal{R}(a(t)) \leq \frac{C_0}{t}$$

Conservative gradient flow

$$f(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{u}\sim\rho}\phi(\boldsymbol{x}, \boldsymbol{u})$$

Example: $\boldsymbol{u} = (a, \boldsymbol{w}), \phi(\boldsymbol{x}, \boldsymbol{u}) = a\sigma(\boldsymbol{w}^T\boldsymbol{x})$

$$V(\boldsymbol{u}) = \frac{\delta\mathcal{R}}{\delta\rho}(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{x}}[(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))\phi(\boldsymbol{x}, \boldsymbol{u})] = \int K(\boldsymbol{u}, \tilde{\boldsymbol{u}})\rho(d\tilde{\boldsymbol{u}}) - \tilde{f}(\boldsymbol{u})$$

$$\partial_t\rho = \nabla(\rho\nabla V)$$

- This is the mean-field equation derived by Chizat and Bach (2018), Mei, Montanari and Nguyen (2018), Rotskoff and Vanden-Eijnden (2018), Sirignano and Spiliopoulos (2018), by studying the continuum limit of two-layer neural networks.
- It is the gradient flow of $\mathcal{R}$ under the Wasserstein metric.
- $\mathcal{R}$ is convex but NOT displacement convex.

Discretizing the gradient flows

- Discretizing the population risk (into the empirical risk) using data
- Discretizing the gradient flow
  - particle method – the dynamic version of Monte Carlo
  - smoothes particle method – analog of vortex blob method
  - spectral method – very effective in low dimensions

Particle method for the feature-based model

$$\partial_t a(\boldsymbol{w}, t) = -\frac{\delta \mathcal{R}}{\delta a}(\boldsymbol{w}) = -\int a(\tilde{\boldsymbol{w}}, t) K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) \pi(d\tilde{\boldsymbol{w}}) + \tilde{f}(\boldsymbol{w})$$

$$\pi(d\boldsymbol{w}) \sim \frac{1}{m} \sum_j \delta_{\boldsymbol{w}_j}, a(\boldsymbol{w}_j, t) \sim a_j(t)$$

Discretized version:

$$\frac{d}{dt} a_j(t) = -\frac{1}{m} \sum_k K(\boldsymbol{w}_j, \boldsymbol{w}_k) a_k(t) + \tilde{f}(\boldsymbol{w}_j)$$

This is exactly the GD for the random feature model.

$$f(\boldsymbol{x}) \sim f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$$

Discretization of the conservative flow

$$\partial_t \rho = \nabla(\rho \nabla V)$$

$$\rho(da, d\boldsymbol{w}) \sim \frac{1}{m} \sum_j \delta_{(a_j, \boldsymbol{w}_j)}$$

$$I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) = \mathcal{R}(f_m), \quad \boldsymbol{u}_j = (a_j, \boldsymbol{w}_j), \quad f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$$

Lemma: Given a set of initial data $\{\boldsymbol{u}_j^0 = (a_j^0, \boldsymbol{w}_j^0), j \in [m]\}$. The solution of (56) with initial data $\rho(0) = \frac{1}{m} \sum_{j=1}^m \delta_{\boldsymbol{u}_j^0}$ is given by

$$\rho(t) = \frac{1}{m} \sum_{j=1}^m \delta_{\boldsymbol{u}_j(t)}$$

where $\{\boldsymbol{u}_j(\cdot), j = \in [m]\}$ solves the following systems of ODEs:

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

Note that this is exactly the GD dynamics for two-layer neural networks.

Flow-based representation

Dynamical system viewpoint (E (2017), Haber and Ruthotto (2017), "Neural ODEs" (Chen et al (2018)), ...)

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbf{g}(\tau, \boldsymbol{z}),\ \boldsymbol{z}(0) = \tilde{\boldsymbol{x}}$$

The flow-map at time 1: $\boldsymbol{x} \to \boldsymbol{z}(1)$.

Hypothesis space (or trial functions):

$$f = \alpha^T \boldsymbol{z}(1)$$

The correct form of $\mathbf{g}$ (E, Ma and Wu, 2019):

$$\mathbf{g}(\tau, \boldsymbol{z}) = \mathbb{E}_{\boldsymbol{w}\sim\pi_\tau} \boldsymbol{a}(\boldsymbol{w}, \tau)\sigma(\boldsymbol{w}^T \boldsymbol{z})$$

where $\{\pi_\tau\}$ is a family of probability distributions.

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbb{E}_{\boldsymbol{w}\sim\pi_\tau} \boldsymbol{a}(\boldsymbol{w}, \tau)\sigma(\boldsymbol{w}^T \boldsymbol{z})$$

As before, we can also use the model:

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbb{E}_{(\boldsymbol{a},\boldsymbol{w})\sim\rho_\tau} \boldsymbol{a}\sigma(\boldsymbol{w}^T \boldsymbol{z})$$

$$f(\boldsymbol{x}) = \alpha^T \boldsymbol{z}(\boldsymbol{x}, 1)$$

Discretize: We obtain the residual neural network model:

$$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \frac{1}{LM}\sum_{j=1}^{M} \boldsymbol{a}_{j,l}\sigma(\boldsymbol{z}_l^T \boldsymbol{w}_{j,l}), l = 1, 2, \cdots, L-1, \quad \boldsymbol{z}_0 = V\tilde{\boldsymbol{x}}$$

$$f_L(\boldsymbol{x}) = \alpha^T \boldsymbol{z}_L$$

Similarly, **particle discretization of the continuous GD flow equation recovers GD for ResNets.**

Mean-field vs continuous formulation

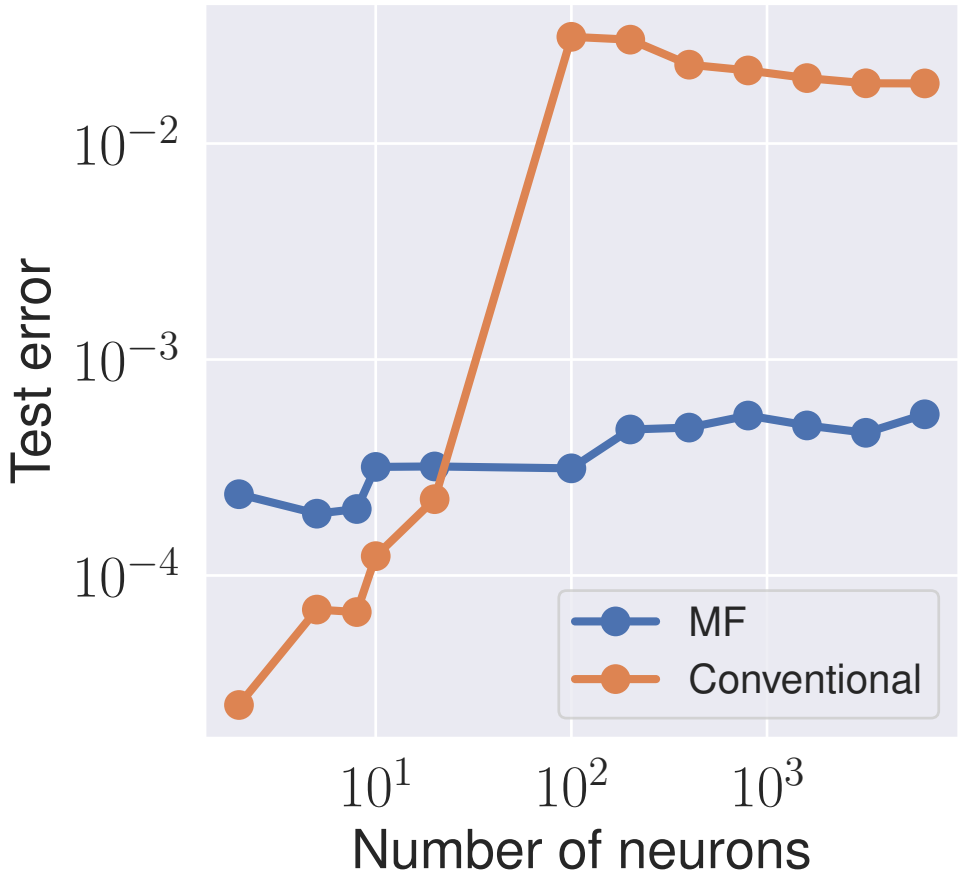
- mean field: discrete $\to$ continuous by taking the limit (more like interacting particles in stat phys)
  The crucial technical work is in the dynamics.
- continuous formulation: continuous $\to$ discrete by discretization (more like the usual numerical analysis situation)
  The crucial technical work is in the statics (representation of functions).

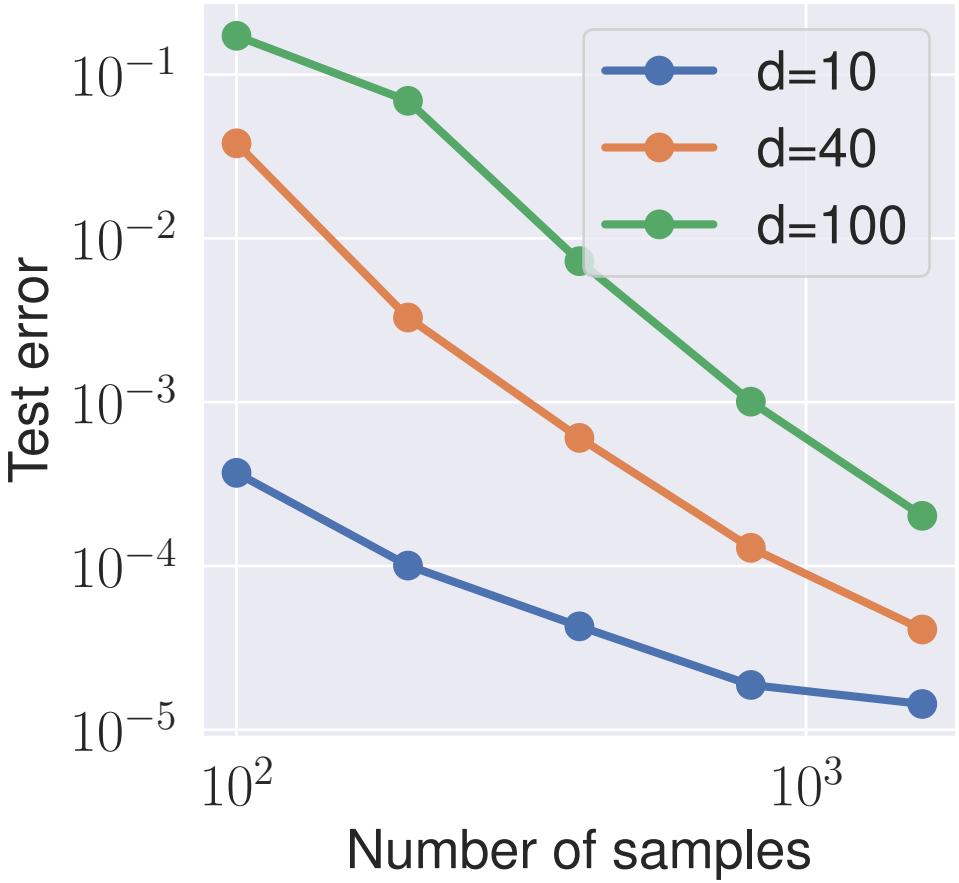
Remarks:

- In a way, neural network models are also very natural: They arise naturally when considering continuous formulations in high dimensions.
- Other discretizations are indeed possible: Continuous formulation allows us to think about ML "outside the box".

Why is "mean-field" or "continuous formulation" better?

Their performance is more robust.





Outline

Success and fragility

- **Why are neural network models successful?**
  - approximation property: overcomes CoD (like Monte Carlo for integration)
  - generalization property: Rademacher complexity controlled
  - optimization: landscape is "nice". Some (not all) implicit regularization mechanisms (e.g. quenching-activation) intuitively understood.
- **Why are neural networks fragile?**. Sensitive dependence on lots of things
  - network architecture (may overfit, maybe unstable)
  - optimization algorithm (different algorithm selects different global min)
  - hyper-parameter (e.g. different training regimes for Adam).

Most important puzzle: Why can we train NN models? (Gradient descent would be disastrous in structural optimization in molecular modeling)

- Chizat and Bach '18 & '20; Wojtowytsch '20; Lu, Ma, Lu, Lu and Ying '20; Agazzi and Lu '20: for certain initial conditions, mean-field training dynamics cannot get stuck at local min
- E, Ma and Wu '19: global and local convergence in the case of simple one-dimensional continuous models

Summary

- A reasonable (although far from being complete) mathematical picture has emerged for regression problems
- Should put more emphasis on better formulation of machine learning, such as the continuous formulation
- Other machine learning problems (classification, GAN, RNN, RL) much less clear

A roadmap for "high dimensional analysis"

- integration $\rightarrow$ Monte Carlo
- functions $\rightarrow$ regression
- probability distributions $\rightarrow$ classification, GAN
- dynamical systems $\rightarrow$ RNN
- PDEs $\rightarrow$ RL