Training neural networks beyond the kernel regime

Instructor: Lei Wu

PKU Summer School, 2021

A mean-field perspective

Scaled two-layer neural networks

Consider the two-layer neural network under the mean-field (MF) scaling:

$$f_m(x;\theta) = \frac{1}{m}\sum_{j=1}^{m} a_j \sigma(w_j^T x) = \int a\sigma(w^T x)\,\mathrm{d}\pi(a,w),$$

where

$$\pi(a,w) = \frac{1}{m}\sum_{j=1}^{m} \delta(a-a_j)\delta(w-w_j).$$

Observation:

- The above representation holds for any $m \in \mathbb{N}_+$.
- The represented function only depends on the distribution π.
- We only need to track the evolution of π instead of individual particles $\{(a_j, w_j)\}$.

Gradient descent at the continuous level

- Consider a general two-layer neural network:

$$f_\pi(x) = \int \psi(x; w)\, \mathrm{d}\pi(w).$$

  For two-layer neural network, $\psi(x; w) = a\sigma(b^T x)$.
- The risk functional is given:

$$\mathcal{R}(\pi) = \mathbb{E}_x[(f_\pi(x) - y)^2].$$

- Gradient flow:

$$\partial_t \pi_t = -\mathsf{Grad}\, \mathcal{R}(\pi_t).$$

- What is the Grad operator?

The continuity equation

Consider a general problem:

$$\min_{\pi \in \mathcal{P}(\Omega)} J(\pi).$$

- Wasserstain gradient flow is given by the continuity equation:

$$\partial_t \pi_t = \nabla \cdot \left(\pi_t \nabla \frac{\delta J}{\delta \pi} \right), \tag{0.1}$$

where $\frac{\delta J}{\delta \pi}$ is the variational derivative defined by

$$\langle \frac{\delta J}{\delta \pi}, \delta \pi \rangle = \lim_{t \to 0} \frac{J(\pi + t\delta\pi) - J(\pi)}{t},$$

where the inner product is with respect to the L^2 metric.

Wasserstain gradient flow

Definition 1

For any $p_1, p_2 \in \mathcal{P}(\Omega)$, the 2-Wasserstain metric is defined by

$$W_2(p_1, p_2) = \min_{\rho \in \Gamma(p_1, p_2)} \int \|x - x'\|_2^2 \, \mathrm{d}\rho(x, x'), \tag{0.2}$$

where $\Gamma(p_1, p_2)$ denotes the set of joint distribution such that the marginal distributions are p_1 and p_2, respectively.

- Define the following proximal point iteration (minimizing movements scheme):

$$\pi_{k+1} = \operatorname{argmin} J(\pi) + \frac{W_2(\pi, \pi_k)}{2\eta} \tag{0.3}$$

- It is well-known that (0.3) converges to the continuity equation (0.1) as $\eta \to 0$. This is why (0.1) is the Wasserstain gradient flow for minimizing $J(\cdot)$.

Properties of the Wasserstain gradient flow

- Preserve the measure:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int \pi_t(w)\,\mathrm{d}w = \int \nabla\cdot\left(\pi_t\nabla\frac{\delta J}{\delta\pi}\right)\mathrm{d}w = 0.$$

- Energy dissipation:

$$\frac{\mathrm{d}J(\pi_t)}{\mathrm{d}t} = \int \frac{\delta J}{\delta\pi}\partial_t\pi_t\,\mathrm{d}w = \int \frac{\delta J}{\delta\pi}\nabla\cdot\left(\pi_t\nabla\frac{\delta J}{\delta\pi}\right)\mathrm{d}w = -\int\left\|\nabla\frac{\delta J}{\delta\pi}\right\|^2\mathrm{d}\pi_t(w).$$

Wasserstain gradient flow for two-layer networks

- For two-layer neural network, $\frac{\delta \mathcal{R}}{\delta \pi} = \mathbb{E}_x[(f_\pi(x) - y)\psi(x; w)]$.
- Hence, the GD flow becomes

$$\partial_t \pi_t = \nabla \cdot (\pi_t v(w; \pi_t)) \,, \tag{0.4}$$

where the velocity field is given by

$$v(w; \pi) = \mathbb{E}_x[(f_\pi(x) - y)\nabla_w \psi(x; w)].$$

- This is a McKean-Vlasov type equation. The PDE (0.4) should be understood in a weak sense. The weak formulation of (0.4) is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} \int g(w) \, \mathrm{d}\pi_t(w) = \int \nabla \cdot (\pi_t v(\pi_t, w)) g(w) \, \mathrm{d}w = - \int \langle v(\pi_t, w), \nabla g(w) \rangle \, \mathrm{d}\pi_t(w), \tag{0.5}$$

where g is a test function.

Finite-width networks

Lemma 2

Given a set of initial data $\{w_j^0 : j \in [m]\}$. *The solution of* (0.4) *with the initial distribution* $\pi_0 = \frac{1}{m}\sum_{j=1}^m \delta_{w_j^0}$ *is given by*

$$\pi_t = \frac{1}{m}\sum_{j=1}^m \delta(w_j(t) - \cdot),$$

where $\{w_j(\cdot) : j \in [m]\}$ *solves the following ODE:*

$$\frac{\mathrm{d}w_j(t)}{\mathrm{d}t} = -v(w_j; \pi_t), \quad w_j(0) = w_j^0, \quad j \in [m].$$

Proof

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int g(w)\,\mathrm{d}\pi_t(w) &= \frac{1}{m}\sum_{j=1}^{m}\frac{\mathrm{d}g(w_j(t))}{\mathrm{d}t} = \frac{1}{m}\sum_{j=1}^{m}\langle\nabla g(w_j(t)),\dot{w}_j(t)\rangle \\ &= -\frac{1}{m}\sum_{j=1}^{m}\langle\nabla g(w_j(t)), v(w_j(t);\pi_t)\rangle \\ &= -\int\langle v(w;\pi),\nabla g(w)\rangle\,\mathrm{d}\pi_t(w).\end{aligned}$$

π_t satisfies the weak form (0.5). Hence, π_t is a solution of the PDE (0.4).

The dynamics of $\{w_j(t) : j \in [m]\}$ can be explicitly written as

$$\dot{w}_j(t) = -\mathbb{E}_x[(f(x;\theta) - f^*(x))\nabla_w \psi(x;w)] = \nabla_{w_j}\mathcal{R}(\theta), \tag{0.6}$$

where

$$f(x;\theta) = \frac{1}{m}\sum_{j=1}^{m} \psi(x;w_j). \tag{0.7}$$

We can see that (0.6) is exactly the GD flow for minimizing the two-layer neural network (0.7).

Remarks

- GD flow for the **scaled** two-layer networks can be equivalently written as a McKean-Vlasov PDE, which holds even when $m = \infty$.
- The McKean-Vlasov PDE is exact for the finite-width case, if the initial condition is an empirical measure. In other words, there does not exist so called **mean-field "limit"**.
- The "mean-field" means that each neuron interacts with other neurons through the distribution formed by all the neurons: $\pi_t = \frac{1}{m}\sum_{j=1}^{m}\delta(w_j - \cdot)$, which is the "mean-field". See (0.6) and the RHS only depends on π_t instead of individual $\{w_j\}$.

Benefit of the perspective of Wasserstain gradient flow

- The risk functional is convex with respect to L^2 metric:

$$\mathcal{R}(\pi) = \mathbb{E}_{x,y}[\int \psi(x;w)\,\mathrm{d}\pi - y]^2 \tag{0.8}$$

$$= \int k(w,w')\pi(w)\pi(w')\,\mathrm{d}w\,\mathrm{d}w' - \int g(w)\pi(w)\,\mathrm{d}w + C, \tag{0.9}$$

  where $k(w,w') = \mathbb{E}_x[\psi(x;w)\psi(x;w)], g(w) = \mathbb{E}_{x,y}[\psi(x;w)y], C = \mathbb{E}_y[y^2]$.

- The McKean-Vlasov PDE is a GD flow with respect to the 2-Wasserstain metric. However, $\mathcal{R}(\cdot)$ is only convex with L_2 metric. Hence, the gradient flow structure is not really helpful. Only for some very special case, $\mathcal{R}(\cdot)$ is displacement convex (`Javanmard, arXiv:1901.01375`).
- **The continuous PDE might be helpful if considering the continuous case, π_0 and π_t are not singular.**

Convergence

Theorem 3 (Informal, Chizat and Bach, NeurIPS 2019)

Suppose the initialization and activation function satisfy some very technical conditions. Let $(\pi_t)_{t\geq 0}$ be a Wasserstain gradient flow of $\mathcal{R}(\cdot)$. If $(\pi_t)_t$ converges to π_∞ in W_2 metric. Then, π_∞ is a global minimizer of $\mathcal{R}(\cdot)$ over $\mathcal{P}_2(\Omega)$.

The above technical condition is hard to verify.

Convergence

Theorem 3 (Informal, Chizat and Bach, NeurIPS 2019)

Suppose the initialization and activation function satisfy some very technical conditions. Let $(\pi_t)_{t\geq 0}$ be a Wasserstain gradient flow of $\mathcal{R}(\cdot)$. If $(\pi_t)_t$ converges to π_∞ in W_2 metric. Then, π_∞ is a global minimizer of $\mathcal{R}(\cdot)$ over $\mathcal{P}_2(\Omega)$.

The above technical condition is hard to verify.

Theorem 4 (Informal, E, Ma, Wu 2019)

Let $f^*(x) = \int \sigma(w^T x)\,\mathrm{d}\pi^*(w)$ and $w \in \mathbb{S}^1$. Assume π^* is the uniform distribution and the initialization π_0 has a differentiable density function. Then, we have

$$\lim_{t\to\infty} W_2(\pi_t, \pi^*) = 0.$$

Convergence in the teacher-student setting

Consider the problem:

$$J(\pi) = \mathbb{E}_x \left(\int a\sigma(w^T x)\, \mathrm{d}\pi(a,w) - f^*(x) \right)^2 + \int |a|\|w\|\, \mathrm{d}\pi(a,w),$$

where σ is ReLU and the target function is finite neurons:

$$f^*(x) = \frac{1}{m} \sum_{j=1}^{m} a_j^* \sigma({w_j^*}^T x).$$

Theorem 5 (Akiyama and Suzuki, ICML 2021)

Assume $\hat{\mathcal{R}}_n$ has a global minimizer π^*. Let π_t be the solution of a variant of GD. Under some smooth technical assumptions, and $\pi_0 = Unif(\mathbb{S}^{d-1})$

- *Global exploration: there exist k_0 such that*

$$J(\pi_k) - J(\pi^*) \leq J(\pi_0) - J(\pi^*).$$

- *Local convergence: There exist $\tau > 1$ such that for any $k \geq k_0$, it holds that*

$$J(\pi_k) - J(\pi^*) \leq \tau^{-(k-k_0)} (J(\pi_0) - J(\pi^*)).$$

Multilayer networks: Propagation of chaos

$$
\begin{aligned}
z_0 &= Vx + b \\
z_i^{l+1} &= \frac{1}{m}\sigma(\sum_{j=1}^{m} w_{i,j}^{l+1} z_j^l) \\
f_{m,L}(x;\theta) &= \frac{1}{m}\sum_{j=1}^{m} a_j z_j^L.
\end{aligned}
\tag{0.10}
$$

Let $(\theta_t)_{t\geq 0}$ be the solution of a (rescaled) GD flow with the initialization:

$$w_{i,j}^l \overset{iid}{\sim} \pi^l.$$

Theorem 6 (Informal: (Araujo, Oliveira et al. arXiv:1906.00193))

As $m \to \infty$, the GD solution $f_{m,L}(x;\theta_t)$ converges to a network with $L \leq 5$.

Intuitive explanation

Let $\delta_i^l = \frac{\partial \mathcal{R}}{\partial z_i^l}$. Then,

$$\delta_j^l = \frac{1}{m} \sum_{j=1}^{m} \frac{\partial \mathcal{R}}{\partial z_j^{l+1}} \frac{z_j^{l+1}}{\partial z_i^l} = \frac{1}{m} \sum_{j=1}^{m} w_{j,i}^{l+1} \delta_j^{l+1}$$

$$\frac{\partial \mathcal{R}}{\partial w_{i,j}^{l+1}} = z_i^l \delta_j^{l+1}.$$

In the limit $m = \infty$, $\frac{\partial \mathcal{R}}{\partial w_{i,j}^{l+1}}$ becomes independent of i, j.

Remarks

- The $1/m$ scaling results in the depth collapse.
- The $1/\sqrt{m}$ scaling results in the lazy training.
- The $1/m$ scaling + deep ResNet works pretty well in the limit $m \to \infty$ and $L \to \infty$.

Implicit Biases of SGD

Motivation

- Modern neural networks usually work in the over-parameterized regime.

ImageNet	# train 1.2×10^6
AlexNet	6.1×10^7
VGG19	1.43×10^8
ResNet-152	6.0×10^7

Motivation

- Modern neural networks usually work in the over-parameterized regime.

ImageNet	# train 1.2×10^6
AlexNet	6.1×10^7
VGG19	1.43×10^8
ResNet-152	6.0×10^7

- To avoid overfitting, one may think that we must rely on the **explicit regularization**, such as weight decay, dropout, batch normalization, etc.

Motivation

- Modern neural networks usually work in the over-parameterized regime.

ImageNet	# train 1.2×10^6
AlexNet	6.1×10^7
VGG19	1.43×10^8
ResNet-152	6.0×10^7

- To avoid overfitting, one may think that we must rely on the **explicit regularization**, such as weight decay, dropout, batch normalization, etc.
- Surprisingly, practitioners often find that optimizers can find good solutions without the need of any **explicit regularizations**.

$$\hat{R}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i).$$

Motivation

- Modern neural networks usually work in the over-parameterized regime.

ImageNet	# train 1.2×10^6
AlexNet	6.1×10^7
VGG19	1.43×10^8
ResNet-152	6.0×10^7

- To avoid overfitting, one may think that we must rely on the **explicit regularization**, such as weight decay, dropout, batch normalization, etc.
- Surprisingly, practitioners often find that optimizers can find good solutions without the need of any **explicit regularizations**.

$$\hat{R}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i).$$

- There must exist some **implicit biases/regularizations** mechanism at work for optimizers.

A large-scale example

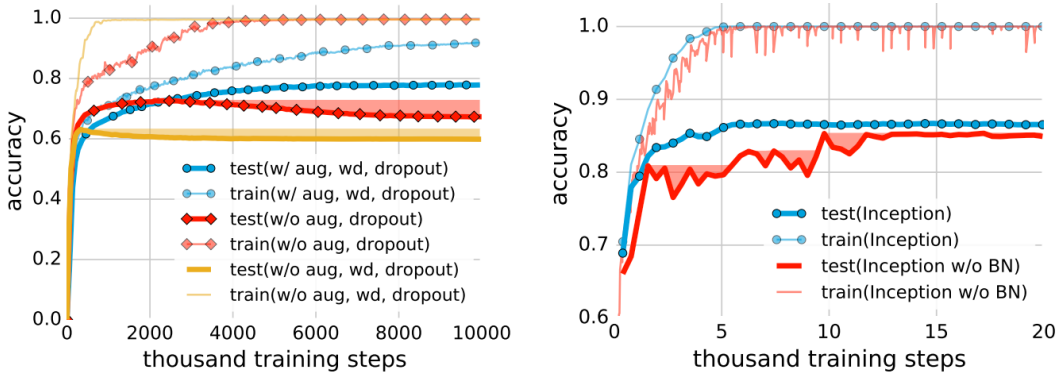


Figure 1: Effects of implicit regularizations. (Taken from (Zhang et al, ICLR2017))

A large-scale example

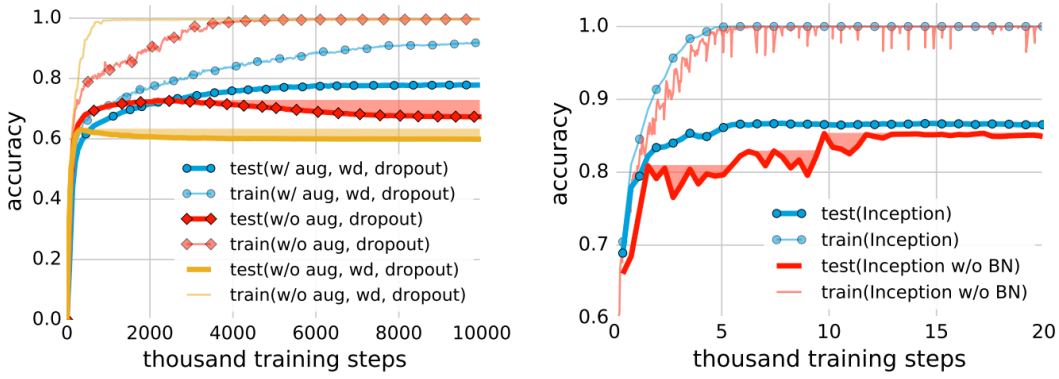


Figure 1: Effects of implicit regularizations. (Taken from (Zhang et al, ICLR2017))

- For CIFAR10, the implicit regularizations account for $85\%+$ test accuracy. Explicit regularizations only improve less than 5% accuracy.

A large-scale example

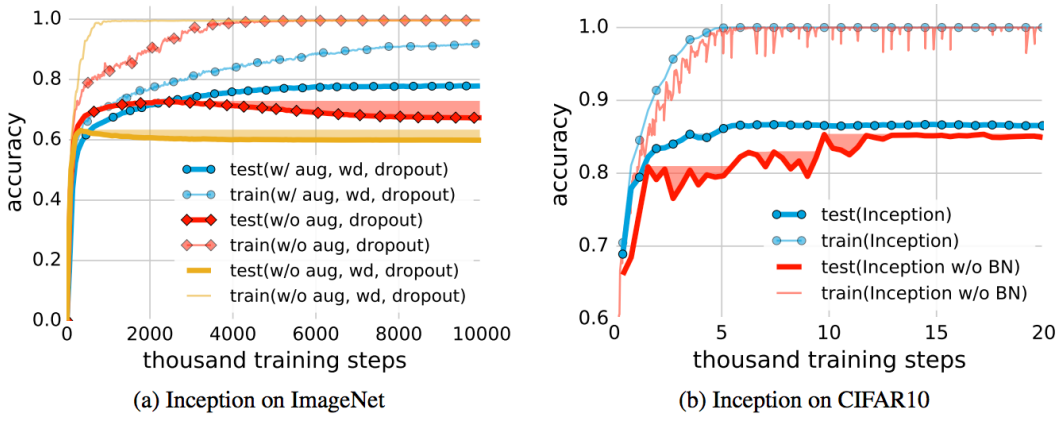


Figure 1: Effects of implicit regularizations. (Taken from (Zhang et al, ICLR2017))

- For CIFAR10, the implicit regularizations account for $85\%+$ test accuracy. Explicit regularizations only improve less than 5% accuracy.
- For ImageNet, explicit regularizations are more important but still not as crucial as the implicit regularizations.

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Many factors affect the implicit regularizations:

- The initialization.

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Many factors affect the implicit regularizations:

- The initialization.
- The structure of model $f(\cdot; \theta)$.

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Many factors affect the implicit regularizations:

- The initialization.
- The structure of model $f(\cdot; \theta)$.
  - For convex models, the pictures of implicit regularizations are rather clear.

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Many factors affect the implicit regularizations:

- The initialization.
- The structure of model $f(\cdot; \theta)$.
  - For convex models, the pictures of implicit regularizations are rather clear.
  - For NN models, not too much progress due to the non-convexity nature [✓].

Implicit regularizations of SGD

SGD:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} \nabla \ell(f(x_i; \theta_t), y_i), \tag{0.11}$$

where $|S_t| = B$ is the batch size.

- GD: When $B = n$, $\theta_{t+1} = \theta_t - \eta \nabla \hat{R}_n(\theta_t)$.
- GD flow: When $\eta \to 0$, $\dot{\theta}_t = -\nabla \hat{R}_n(\theta_t)$ [✓].

Many factors affect the implicit regularizations:

- The initialization.
- The structure of model $f(\cdot; \theta)$.
  - For convex models, the pictures of implicit regularizations are rather clear.
  - For NN models, not too much progress due to the non-convexity nature [✓].
- The learning rate and batch size [✓].

A systematic study of the teacher-student setting

Setup: two-layer neural networks (2LNNs) in the teacher-student setting.

- Model:

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

where $\sigma(z) = \max(0, z)$ is the ReLU function.

A systematic study of the teacher-student setting

Setup: two-layer neural networks (2LNNs) in the teacher-student setting.

- Model:

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

  where $\sigma(z) = \max(0, z)$ is the ReLU function.

- Target functions:

$$y = f^*(\boldsymbol{x}) = \frac{1}{M} \sum_{j=1}^{M} a_j^* \sigma(\boldsymbol{b}_j^{*T} \boldsymbol{x}).$$

  We will consider the **over-realized** case, i.e., $m > M$.

A systematic study of the teacher-student setting

Setup: two-layer neural networks (2LNNs) in the teacher-student setting.

- Model:

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

  where $\sigma(z) = \max(0, z)$ is the ReLU function.

- Target functions:

$$y = f^*(\boldsymbol{x}) = \frac{1}{M} \sum_{j=1}^{M} a_j^* \sigma(\boldsymbol{b}_j^{*T} \boldsymbol{x}).$$

  We will consider the **over-realized** case, i.e., $m > M$.

- Initialization:

$$a_j = 0, \boldsymbol{b}_j \sim \mathcal{N}(0, I/d).$$

A systematic study of the teacher-student setting

Setup: two-layer neural networks (2LNNs) in the teacher-student setting.

- Model:

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

  where $\sigma(z) = \max(0, z)$ is the ReLU function.

- Target functions:

$$y = f^*(\boldsymbol{x}) = \frac{1}{M} \sum_{j=1}^{M} a_j^* \sigma(\boldsymbol{b}_j^{*T} \boldsymbol{x}).$$

  We will consider the **over-realized** case, i.e., $m > M$.

- Initialization:

$$a_j = 0, \boldsymbol{b}_j \sim \mathcal{N}(0, I/d).$$

- The empirical risk

$$\hat{R}_n(\boldsymbol{a}, \boldsymbol{B}) = \frac{1}{n} \sum_{i=1}^{m} (f_m(\boldsymbol{x}_i; \boldsymbol{a}, \boldsymbol{B}) - f^*(\boldsymbol{x}_i)).$$

A systematic study of the teacher-student setting

Setup: two-layer neural networks (2LNNs) in the teacher-student setting.

- Model:

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

  where $\sigma(z) = \max(0, z)$ is the ReLU function.

- Target functions:

$$y = f^*(\boldsymbol{x}) = \frac{1}{M} \sum_{j=1}^{M} a_j^* \sigma(\boldsymbol{b}_j^{*T} \boldsymbol{x}).$$

  We will consider the **over-realized** case, i.e., $m > M$.

- Initialization:

$$a_j = 0, \boldsymbol{b}_j \sim \mathcal{N}(0, I/d).$$

- The empirical risk

$$\hat{R}_n(\boldsymbol{a}, \boldsymbol{B}) = \frac{1}{n} \sum_{i=1}^{m} (f_m(\boldsymbol{x}_i; \boldsymbol{a}, \boldsymbol{B}) - f^*(\boldsymbol{x}_i)).$$

- The associated random feature model (RFM): $f_m(\cdot; \boldsymbol{a}, \boldsymbol{B}_0)$. Here $\boldsymbol{B}_0$ is fixed after the random initialization and only $\boldsymbol{a}$ is learnable.

The highly over-parameterized regime

Theorem 7 (Informal, (E, Ma, Wu, 2019))

Let $\tilde{\boldsymbol{a}}(t)$ denote the GD solution of RFM. For any $\delta \in (0,1)$, if $m \geq \mathrm{poly}(n, \log(1/\delta))$, with probability $1-\delta$, we have

$$\sup_{\boldsymbol{x}\in\mathbb{S}^{d-1}, t\in[0,\infty)} |f_m(\boldsymbol{x}; \boldsymbol{a}(t), \boldsymbol{B}(t)) - f_m(\boldsymbol{x}; \tilde{\boldsymbol{a}}(t), \boldsymbol{B}_0)| \leq \frac{\mathrm{poly}(n, \log(1/\delta))}{\sqrt{m}}.$$

- $\|\boldsymbol{B}(t) - \boldsymbol{B}(0)\| \leq \mathrm{poly}(n)/m \ll 1.$

The highly over-parameterized regime

Theorem 7 (Informal, (E, Ma, Wu, 2019))

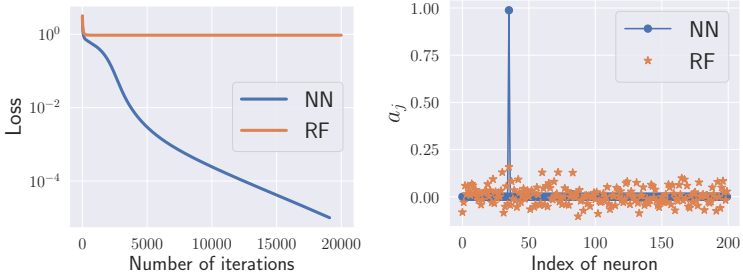
Let $\tilde{\boldsymbol{a}}(t)$ denote the GD solution of RFM. For any $\delta \in (0,1)$, if $m \geq \mathrm{poly}(n, \log(1/\delta))$, with probability $1-\delta$, we have

$$\sup_{\boldsymbol{x}\in\mathbb{S}^{d-1}, t\in[0,\infty)} |f_m(\boldsymbol{x}; \boldsymbol{a}(t), \boldsymbol{B}(t)) - f_m(\boldsymbol{x}; \tilde{\boldsymbol{a}}(t), \boldsymbol{B}_0)| \leq \frac{\mathrm{poly}(n, \log(1/\delta))}{\sqrt{m}}.$$

- $\|\boldsymbol{B}(t) - \boldsymbol{B}(0)\| \leq \mathrm{poly}(n)/m \ll 1$.
- This result sheds no light on the origin of improved performance of NNs over kernel methods. Does there exist strong implicit regularizations when NNs are less over-parameterized?

GD for the population risk

Consider the target function is a single neuron: $f_1^*(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{*T}\boldsymbol{x})$. Here, $m = 100, d = 200$.



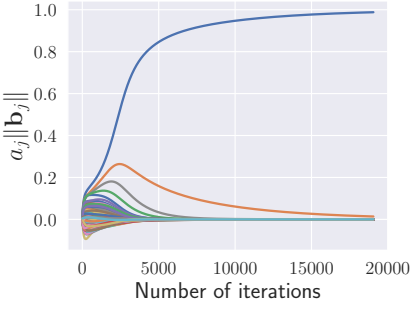
- RFM and 2LNN are close initially due to the time-scale separation.

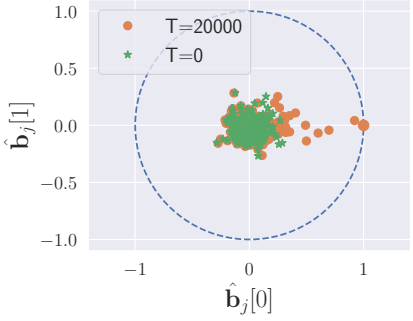
$$\dot{a}_j(t) = O(\|\boldsymbol{b}_j\|) = O(1), \qquad \dot{\boldsymbol{b}}_j(t) = O(|a_j|) = o(1).$$

- GD for 2LNN is "implicitly" biased to pick sparse solutions, although there exist many other solutions.

A self-sparsification behavior

Let us take a closer look at the dynamics of each neuron.





- Initially, all the neurons increase the magnitude due to the closeness to RFM.
- After the initial phase, one "activated" neuron keeps increasing the magnitude, while all the other neurons are gradually "quenched".

More examples

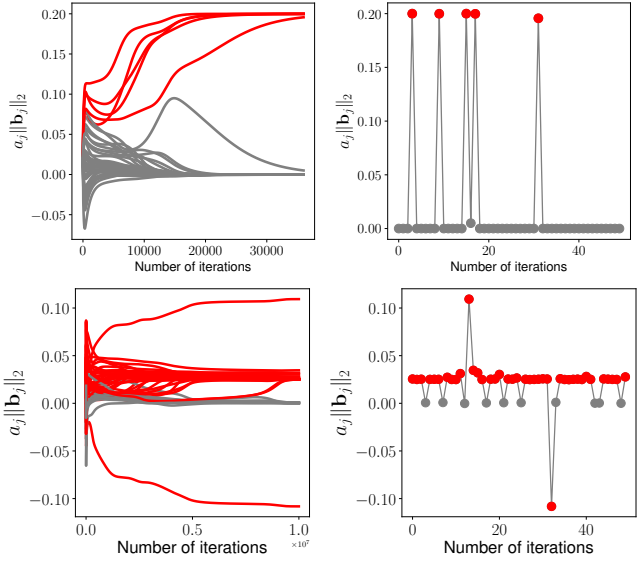


Figure 2: Upper: $m = 50, M = 5$; **Bottom:** $m = 50, M = 40$.

What happens for the finite sample case?

- Highly over-parameterized regime: $m \gg n$. It is clear that NN $=$ RFM.

What happens for the finite sample case?

- Highly over-parameterized regime: $m \gg n$. It is clear that NN $=$ RFM.
- Highly under-parameterized regime: $m \ll n/d$. GD should exhibit the quenching-activation behavior and tend to pick up sparse solutions.

What happens for the finite sample case?

- Highly over-parameterized regime: $m \gg n$. It is clear that NN $=$ RFM.
- Highly under-parameterized regime: $m \ll n/d$. GD should exhibit the quenching-activation behavior and tend to pick up sparse solutions.
- **Mildly over- and under-parameterized regime:** $n/d \lesssim m \lesssim n$. It is unclear!

GD for the empirical risk

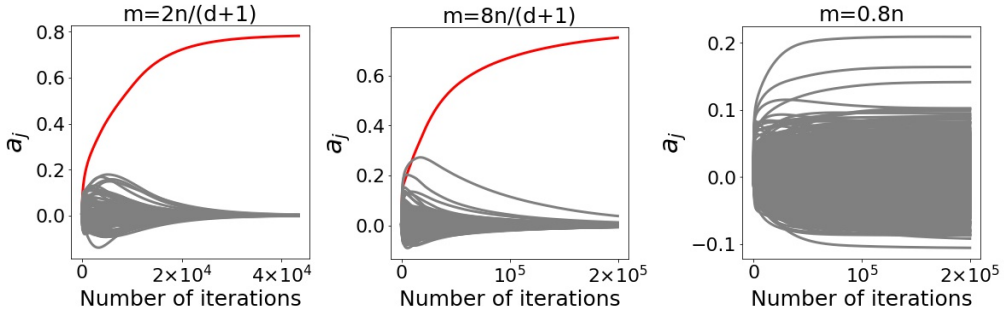


Figure 3: The target function is a single neuron.

It suggests that the implicit biases of GD exhibit a transition when increasing the width from $m = n/(d+1)$ to $m = n$.

Test error and path norm curves

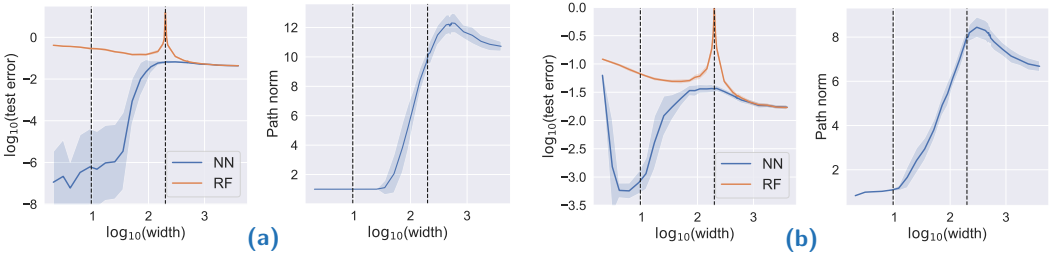


Figure 4: (a) Single neuron. (b) Circle neuron $f^*(x) = \mathbb{E}_{b\sim\pi}[\sigma(b^T x)]$ with π be the uniform distribution over the unit circle $\Omega = \{b \in \mathbb{S}^{d-1} : b_1^2 + b_2^2 = 1\}$. Two dashed lines correspond to $m = n/(d+1)$ and $m = n$, respectively.

Test error and path norm curves

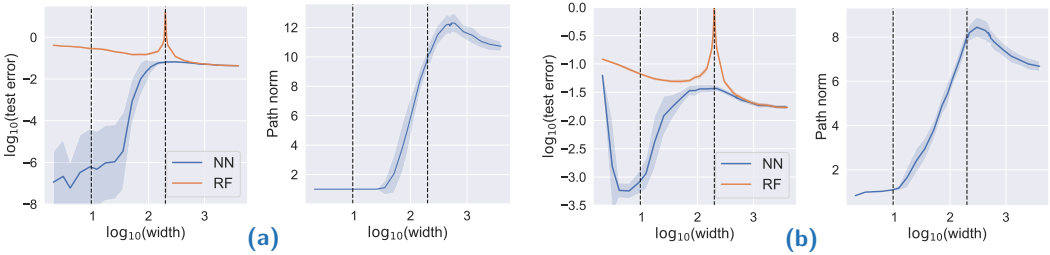


Figure 4: (a) Single neuron. (b) Circle neuron $f^*(x) = \mathbb{E}_{b\sim\pi}[\sigma(b^T x)]$ with π be the uniform distribution over the unit circle $\Omega = \{b \in \mathbb{S}^{d-1} : b_1^2 + b_2^2 = 1\}$. Two dashed lines correspond to $m = n/(d+1)$ and $m = n$, respectively.

- Test error peaks around $m \approx n$ due to the closeness of NN and RFM at the early stage. When $m = n$, the norm of RFM blows up, and GD for NN is not strong enough to cure it.

Test error and path norm curves

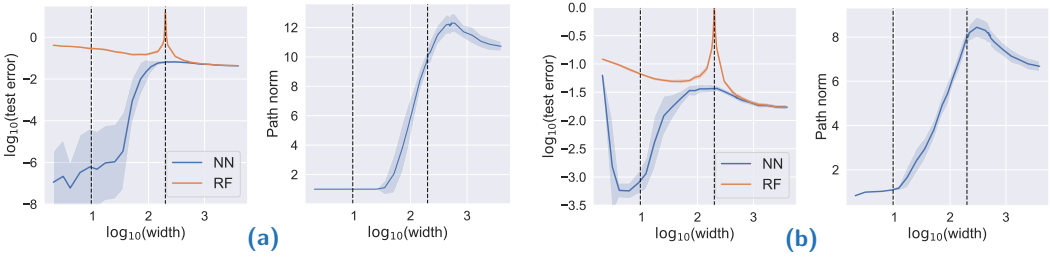


Figure 4: (a) Single neuron. (b) Circle neuron $f^*(x) = \mathbb{E}_{b\sim\pi}[\sigma(b^T x)]$ with π be the uniform distribution over the unit circle $\Omega = \{b \in \mathbb{S}^{d-1} : b_1^2 + b_2^2 = 1\}$. Two dashed lines correspond to $m = n/(d+1)$ and $m = n$, respectively.

- Test error peaks around $m \approx n$ due to the closeness of NN and RFM at the early stage. When $m = n$, the norm of RFM blows up, and GD for NN is not strong enough to cure it.
- The test errors of NN exhibits a "double descent". The first descent is due to the decrease of approximation error of NN; the second descent is due to the decrease of approximation error of RFM. The mechanism is very different from that of RFM.

2LNN under the mean-field scaling

In theoretical studies, it is common to consider the 2LNN with the *mean-field*(MF) scaling:

$$f_m(x; a, B) = \frac{1}{m} \sum_{j=1}^{m} a_j \sigma(b_j^T x). \tag{0.12}$$

2LNN under the mean-field scaling

In theoretical studies, it is common to consider the 2LNN with the *mean-field*(MF) scaling:

$$f_m(x; a, B) = \frac{1}{m} \sum_{j=1}^{m} a_j \sigma(b_j^T x). \tag{0.12}$$

The GD dynamics (after rescaling the time by $t \to mt$) is given by

$$\dot{a}_j(t) = -\sum_{i=1}^{n} e_i \sigma(b_j^T x_i) \tag{0.13}$$

$$\dot{b}_j(t) = -\sum_{i=1}^{n} e_i a_j \sigma'(b_j^T x_i) x_i. \tag{0.14}$$

For this *scaled* 2LNN, $\dot{a}_j(t) \sim \|b_j\| = O(1)$ and $\dot{b}_j(t) \sim |a_j| = O(1)$. No time-scale separation. Hence, we expect that all the neurons behave similarly. We will call (0.13) GD-MF.

Implicit biases of GD for the scaled 2LNN

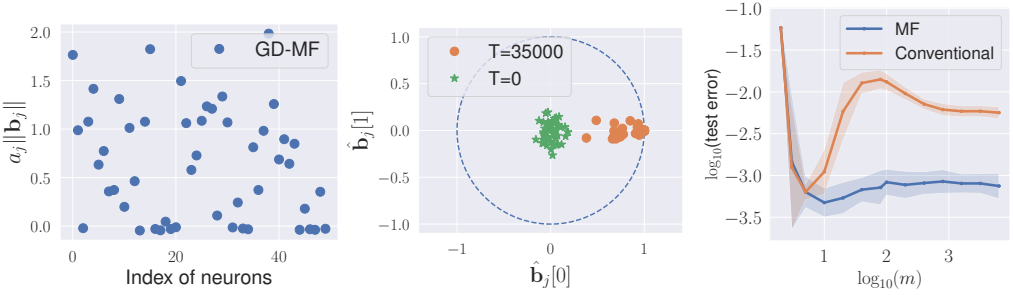


Figure 5: GD-MF for learning the single neuron. Here $m = 50, d = 100, n = \infty$. **Left:** The "magnitude" of each neuron for the converged solution. **Middle:** The projection to the first two coordinates of $\hat{\boldsymbol{b}}$ for each neuron. **Right:** The test error curves of the scaled 2LNN and the conventional one.

Comparison between the scaled and unscaled 2LNNs

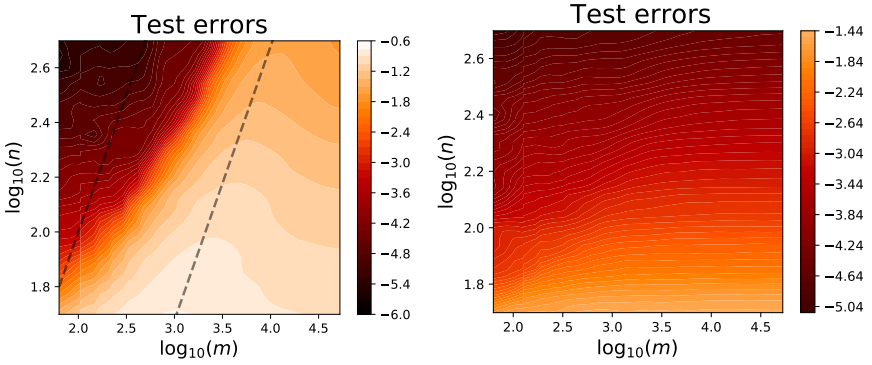


Figure 6: Heatmap of test errors of GD solutions. **Left:** GD solutions for the conventional unscaled 2LNN. **Right:** GD solutions for the scaled 2LNN.

Comparison between the scaled and unscaled 2LNNs

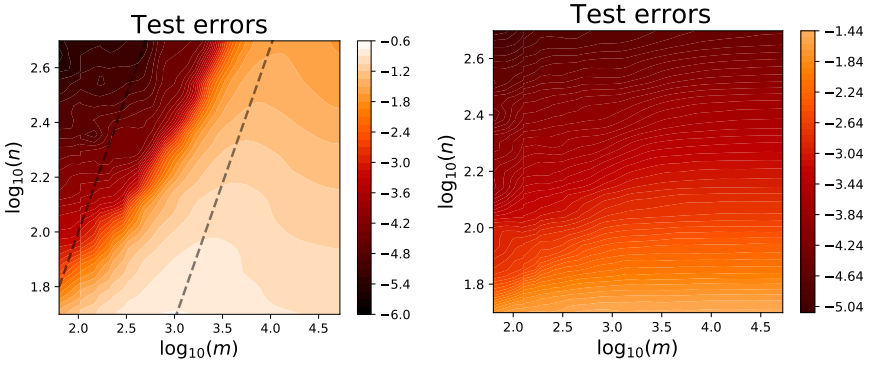


Figure 6: Heatmap of test errors of GD solutions. **Left:** GD solutions for the conventional unscaled 2LNN. **Right:** GD solutions for the scaled 2LNN.

- Test errors of the unscaled 2LNNs are sensitive to the network width.

Comparison between the scaled and unscaled 2LNNs

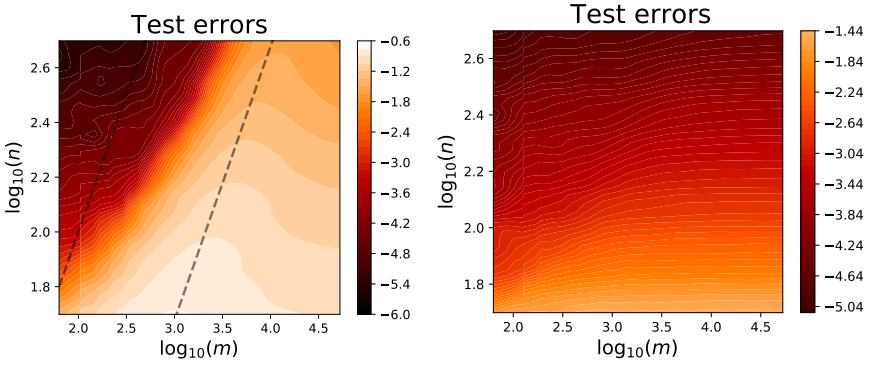


Figure 6: Heatmap of test errors of GD solutions. **Left:** GD solutions for the conventional unscaled 2LNN. **Right:** GD solutions for the scaled 2LNN.

- Test errors of the unscaled 2LNNs are sensitive to the network width.
- Test errors of the scaled 2LNNs are more smoothly with the increase of network width. There is almost no deterioration of performance even when the network is highly over-parameterized.

SGD vs. GD

- SGD is originally suggested to speed up GD. Surprisingly, it is often observed that SGD solutions generalize better than GD solutions.
- How can we characterize the difference between the solutions found by the two optimizers? Why SGD generalizes better than GD?

Flatness hypothesis

The famous **flatness** hypothesis (Hochreiter and Schmidhuber, 1995; Keskar et al., 2016):

SGD converges to flatter solutions and flatter solutions generalize better.

Flatness hypothesis

The famous **flatness** hypothesis (Hochreiter and Schmidhuber, 1995; Keskar et al., 2016):

SGD converges to flatter solutions and flatter solutions generalize better.

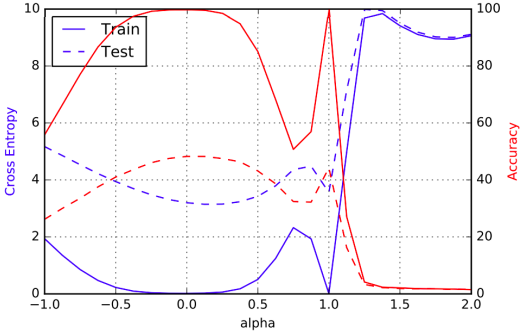


Figure 7: The landscape for for $\theta(\alpha) := \alpha\theta_{SGD} + (1-\alpha)\theta_{GD}$. Taken from (Keskar et al., 2016).

Flatness hypothesis

The famous **flatness** hypothesis (Hochreiter and Schmidhuber, 1995; Keskar et al., 2016):

SGD converges to flatter solutions and flatter solutions generalize better.

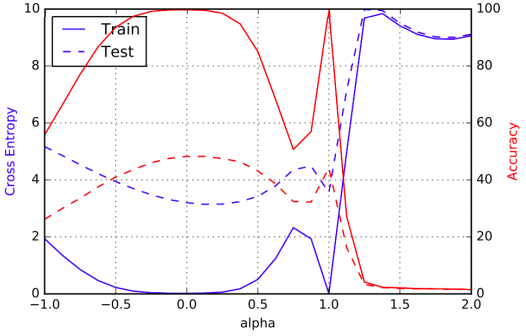


Figure 7: The landscape for for $\theta(\alpha) := \alpha\theta_{SGD} + (1-\alpha)\theta_{GD}$. Taken from (Keskar et al., 2016).

- Does SGD really prefer flat solutions? Why? [✓]
- Why does the flat solutions generalize better than the sharp solutions? [✗]

Escape phenomenon

SGD = GD + noise. Let $g_i = \nabla \ell(f_i, f), g = \mathbb{E}_i[g_i]$.

$$\theta_{t+1} = \theta_t - \eta \nabla g(\theta_t) + \frac{\eta}{B} \xi(\theta_t),$$

where $\mathbb{E}[\xi] = 0, \mathbb{E}[\xi \xi^T] = \frac{1}{n} \sum_{i=1}^{n} (g_i - g)(g_i - g)^T$.

Escape phenomenon

SGD = GD + noise. Let $g_i = \nabla \ell(f_i, f), g = \mathbb{E}_i[g_i]$.

$$\theta_{t+1} = \theta_t - \eta \nabla g(\theta_t) + \frac{\eta}{B} \xi(\theta_t),$$

where $\mathbb{E}[\xi] = 0, \mathbb{E}[\xi \xi^T] = \frac{1}{n} \sum_{i=1}^{n} (g_i - g)(g_i - g)^T$.

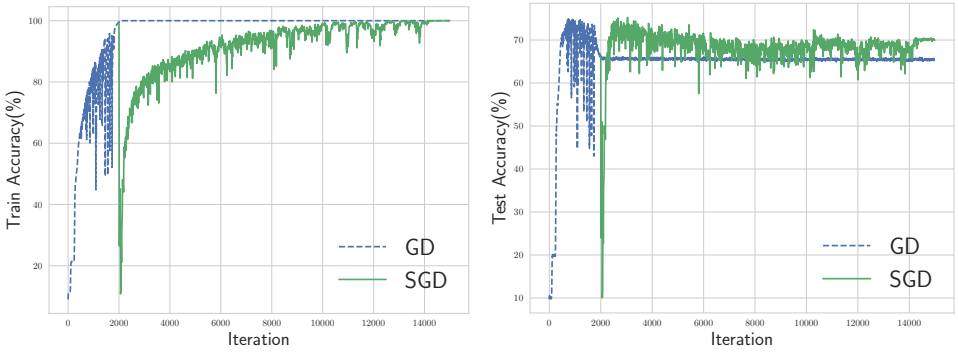


Figure 8: Fast escape phenomenon in fitting corrupted FashionMNIST. This escape phenomenon shows that the GD solutions are unstable for SGD dynamics.

An illustrative example

Consider the target function $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with

$$f_1(x) = \min(x^2, 0.1(x-1)^2), \qquad f_2(x) = \min(x^2, 1.9(x-1)^2)$$

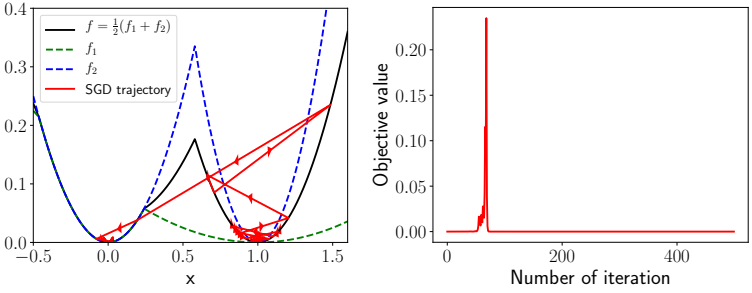


Figure 9: SGD with $\eta = 0.7, x_0 = 1 - \varepsilon$ with $\varepsilon = 1e-5$.

Observation:

- Flatness cannot fully characterize the difference between SGD and GD, in particular the escape phenomenon.

A linear stability analysis

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{0.15}$$

A linear stability analysis

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{0.15}$$

- The SGD iteration is given by,

$$x_{t+1} = x_t - \eta a_\xi x_t = (1 - \eta a_\xi) x_t, \tag{0.16}$$

A linear stability analysis

- Consider the one-dimensional problem:

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2, \quad a_i \geq 0 \ \forall i \in [n] \tag{0.15}$$

- The SGD iteration is given by,

$$x_{t+1} = x_t - \eta a_\xi x_t = (1 - \eta a_\xi) x_t, \tag{0.16}$$

- So after one step update, we have

$$\mathbb{E}\, x_{t+1} = (1 - \eta a)\, \mathbb{E}\, x_t, \tag{0.17}$$

$$\mathbb{E}\, x_{t+1}^2 = \left[(1 - \eta a)^2 + \eta^2 s^2\right] \mathbb{E}\, x_t^2, \tag{0.18}$$

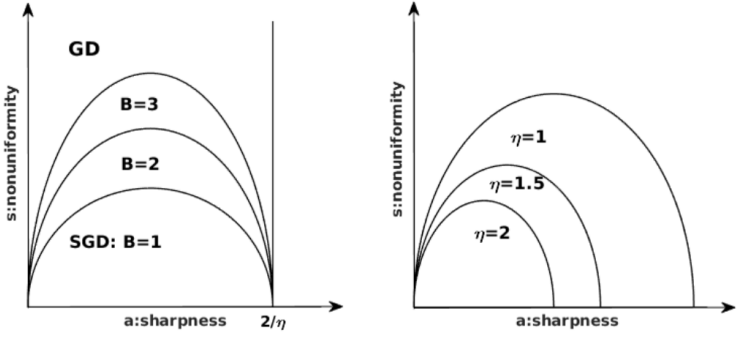
where $a = \frac{1}{n} \sum_{i=1}^{n} a_i, s = \sqrt{\frac{1}{n} \sum_{i=1}^{n} a_i^2 - a^2}$. We call a: sharpness b: non-uniformity.

Linear stability condition

- Global minimum $x^* = 0$ is stable for SGD with batch size B, if

$$(1 - \eta a)^2 + \frac{\eta^2 (n - B)}{B(n - 1)} s^2 \leq 1, \quad s \geq 0. \tag{0.19}$$

- Otherwise, a small perturbation will lead SGD to escape from 0.
- Diagram:



- The learning rate and batch size play different roles in the global minima selection.

Remarks

- Similar analyses can be extended for high-dimensional cases

$$\lambda_{max}\left\{(I-\eta H)^2+\frac{\eta^2(n-B)}{B(n-1)}\Sigma\right\}\leq 1.$$

  Let $a=\lambda_{max}(H)$, $s^2=\lambda_{max}(\Sigma)$, then a necessary condition is

$$0\leq a\leq\frac{2}{\eta},\quad 0\leq s\leq\frac{1}{\eta}\sqrt{\frac{B(n-1)}{n-B}}\approx\frac{\sqrt{B}}{\eta}.$$

- The dynamical stability perspective is applicable for general optimizers and "definition" of stability.
- In practice, SGD for NNs lives at the **edge of stability**.

Learning rate is crucial for the flatness of the selected minima

Table 1: Sharpness of the solutions found by GD with different learning rates. Each experiment is repeated for 5 times with independent random initialization.

η	0.01	0.05	0.1	0.5	1
FashionMNIST	53.5 ± 4.3	39.3 ± 0.5	19.6 ± 0.15	3.9 ± 0.0	1.9 ± 0.0
CIFAR10	198.9 ± 0.6	39.8 ± 0.2	19.8 ± 0.1	3.6 ± 0.4	-
prediction $2/\eta$	200	40	20	4	2

The selection mechanism

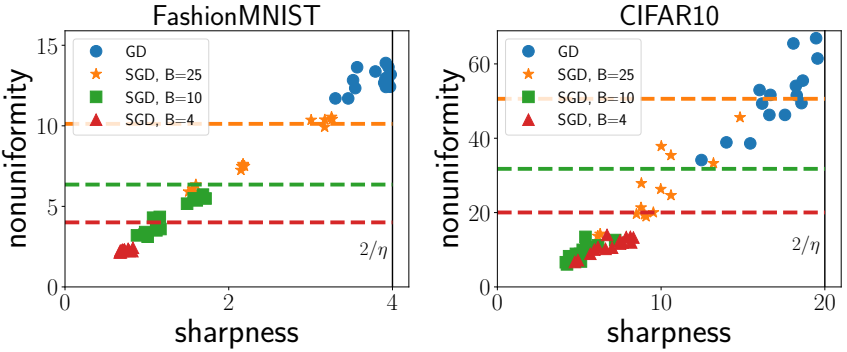


Figure 10: The sharpness-non-uniformity diagram for the minima selected by SGD.

- SGD prefer uniform solutions. [✓]
- Non-uniformity is nearly proportional to the sharpness. [✗]
- Combining them together, SGD is biased to solutions with small sharpness, i.e., flat solutions.

Non-uniformity is strongly correlated to sharpness

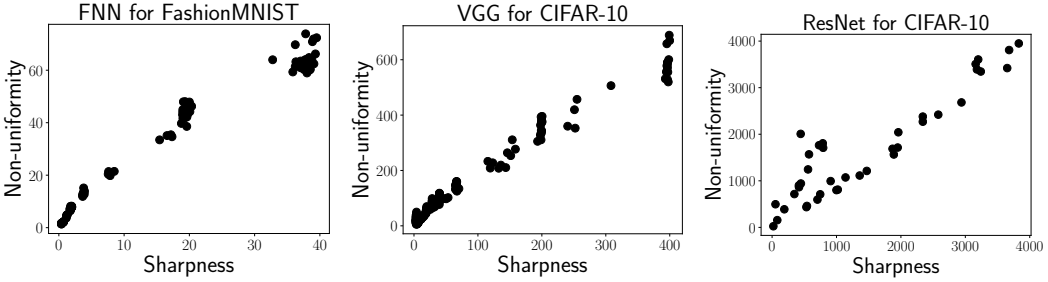


Figure 11: Scatter plot of sharpness and non-uniformity. For each case, we trained about 500 models with different initializations, learning rates, batch sizes, etc.

Why the non-uniformity is strongly correlated with the sharpness?

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN $=$ kernel.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN $=$ kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN $=$ kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

GD vs. SGD

- Minima selection via the (linear) dynamical stability.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

GD vs. SGD

- Minima selection via the (linear) dynamical stability.
- SGD favors solutions with large uniformity, while GD does not.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

GD vs. SGD

- Minima selection via the (linear) dynamical stability.
- SGD favors solutions with large uniformity, while GD does not.
- The uniformity is strongly correlated with the flatness.

Summary

GD for 2LNN

- In the highly over-parameterized regime, NN = kernel.
- In the less over-parameterized regime, **bias to sparse solutions** if the model is over-realized.
- This implicit regularizations is **sensitive** to the change of network width. A sharp transition happens within the midly over-parameterized regime.
- For **scale networks**, the implicit regularization is much more **robust**.

GD vs. SGD

- Minima selection via the (linear) dynamical stability.
- SGD favors solutions with large uniformity, while GD does not.
- The uniformity is strongly correlated with the flatness.
- Hence, SGD prefers flatter solution than GD.

Missing topics

What I should have covered:

- Refined analyses of RFM and KRR: implicit biases, double descent, and benign overfitting.
- (Deep) matrix factorization and linear networks.
- SDE-based interpretations of differences between SGD and GD.
- Dropout, batch normalization, weight decay, distillation, etc.

Missing topics

What I should have covered:

- Refined analyses of RFM and KRR: implicit biases, double descent, and benign overfitting.
- (Deep) matrix factorization and linear networks.
- SDE-based interpretations of differences between SGD and GD.
- Dropout, batch normalization, weight decay, distillation, etc.

Less explored topics:

- RNN, LSTM, Transformer, CNN.
- GAN, Auto-encoder, normalized flow, etc.
- Self-supervised (contrastive) learning.