Neural Tangent Kernel

Shihua Zhang

December 1, 2021

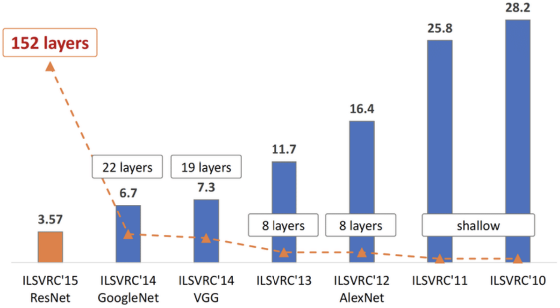
Overview

Rise of Deep Learning: Algorithm and Data

- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Krizhevsky et al., 2012].
- When most AI research focused on models and algorithms, Fei-Fei Li wanted to expand the data available to train AI alg.

Rise of Deep Learning: Algorithm and Data

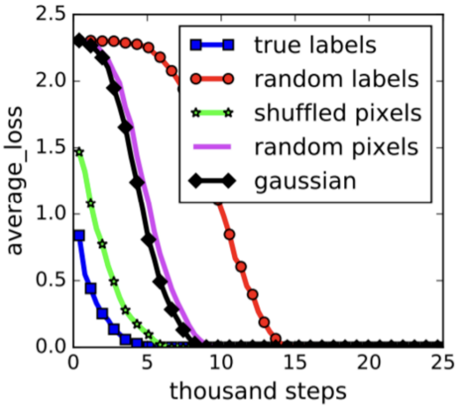
- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Krizhevsky et al., 2012].
- When most AI research focused on models and algorithms, Fei-Fei Li wanted to expand the data available to train AI alg.



- The evolution of the winning entries on the ImageNet.

What do DNNs Learn?

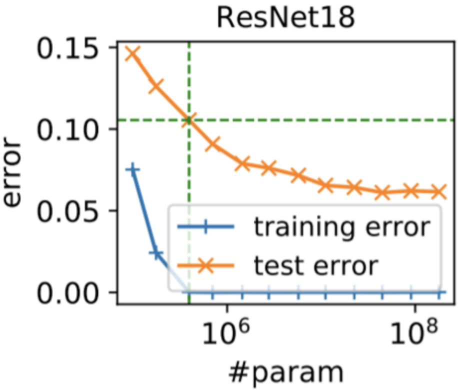
- DNNs are often highly over-parameterized.
- Fit random labels and random pixels [Zhang et al., 2021].



- On CIFAR10

What do DNNs Learn?

- Training ResNet of different sizes [Zhang et al., 2021].



- On CIFAR10

Principled Understanding is still Lacking

- **Questions:**
  - Why over-parameterized DNNs trained by gradient descent can fit training data with arbitrary labels?
  - Why DNNs can generalize in the over-parameterized regime?

- **Challenges:**
  - The objective function is highly non-convex and/or non-smooth.
  - Conventional optimization theory can only guarantee finding stationary points.

Principled Understanding is still Lacking

- **Questions:**
  - Why over-parameterized DNNs trained by gradient descent can fit training data with arbitrary labels?
  - Why DNNs can generalize in the over-parameterized regime?

- **Challenges:**
  - The objective function is highly non-convex and/or non-smooth.
  - Conventional optimization theory can only guarantee finding stationary points.

  - Traditional generalization bounds often require the number of parameters is much smaller than the number of data points.
  - These bounds become vacuous in the over-para. regime.

Today's Theme

- What is NTK theory [Jacot et al., 2018]?
A class of infinitely wide (or ultra-wide) NNs trained by gradient descent $\Leftrightarrow$ kernel regression with a fixed kernel.

- Neural tangent kernel (NTK) is a kernel that describes the evolution of DNNs during their training by gradient descent.

Today's Theme

- What is NTK theory [Jacot et al., 2018]?
  A class of infinitely wide (or ultra-wide) NNs trained by gradient descent $\Leftrightarrow$ kernel regression with a fixed kernel.

- Neural tangent kernel (NTK) is a kernel that describes the evolution of DNNs during their training by gradient descent.

- Can we use make this theory practical?
  - to understand optimization and generalization in NNs.
  - to design new algorithms inspired by the NTK theory.

Overview

Setting: Supervised Learning

- Given data: $\{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^d \times \mathbb{R}$.
- A neural network: $f(w, x)$.
- Loss function:

$$\ell(w) = \frac{1}{2} \sum_{i=1}^{n} \left(f\left(w, x_i\right) - y_i\right)^2$$

- Algorithm: minimize training loss $\ell(w)$ by gradient descent.

Setting: Supervised Learning

- Given data: $\{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^d \times \mathbb{R}$.
- A neural network: $f(w, x)$.
- Loss function:

$$\ell(w) = \frac{1}{2} \sum_{i=1}^{n} \left(f\left(w, x_i\right) - y_i\right)^2$$

- Algorithm: minimize training loss $\ell(w)$ by gradient descent.
- Tool: gradient flow, a.k.a., gradient decent with infinitesimally learning rate.
- The dynamics can be described by an ODE:

$$\frac{dw(t)}{dt} = -\nabla \ell(w(t))$$

Evolving Equation on Predictions

- The following lemma describes the dynamics of the predictions on training data points.

Lemma 1

Let $u(t) = (f(w(t), x_i))_{i\in[n]} \in \mathbb{R}^n$ *be the network outputs on all* x_i*'s at time* t*, and* $y = (y_i)_{i\in[n]}$ *be the labels. Then* $u(t)$ *follows the following evolution, where* $H(t)$ *is an* $n \times n$ *positive semidefinite matrix whose* (i, j)*-th entry is* $\left\langle \frac{\partial f(w(t),x_i)}{\partial w}, \frac{\partial f\left(w(t),x_j\right)}{\partial w} \right\rangle$ *:*

$$\frac{du(t)}{dt} = -H(t) \cdot (u(t) - y)$$

Evolving Equation on Predictions

Proof.

The parameters w evolve according to the differential equation

$$\frac{dw(t)}{dt} = -\nabla \ell(w(t)) = -\sum_{i=1}^{n} \left(f\left(w(t), x_i\right) - y_i\right) \frac{\partial f\left(w(t), x_i\right)}{\partial w}.$$

According to the chain rule, we have:

$$\frac{df\left(w(t), x_i\right)}{dt} = -\sum_{j=1}^{n} \left(f\left(w(t), x_j\right) - y_j\right) \left\langle \frac{\partial f\left(w(t), x_i\right)}{\partial w}, \frac{\partial f\left(w(t), x_j\right)}{\partial w} \right\rangle$$

Consider the definition of $H(t)$ and $u(t)$, we have:

$$\frac{du(t)}{dt} = -H(t) \cdot (u(t) - y)$$

Evolving Equation on Predictions

The evolving equation on predictions:

$$\frac{du(t)}{dt} = -H(t) \cdot (u(t) - y)$$

Remarks:

- The matrix $H(t)$ remains constant during training, i.e., equals $H(0)$ when width is allowed to go to infinity.
- Under a random initialization of parameters, the random matrix $H(0)$ converges in probability to a certain deterministic kernel matrix H^* as the width goes to infinity.

Coupling Ultra-wide NNs and NTK

Consider a simple two-layer neural network:

$$f(a, W, x) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(w_r^\top x\right)$$

where $\sigma(\cdot)$ is the activation function.

Assumptions:

- $|\dot{\sigma}(z)|$ and $|\ddot{\sigma}(z)|$ are bounded by 1 for all $z \in \mathbb{R}$.
- Any input x has Euclidean norm 1, i.e., $\|x\|_2 = 1$.
- Use random initialization $w_r(0) \sim N(0, I)$ and $a_r \sim \text{Unif}\,[\{-1, 1\}]$.
- Only optimize the first layer, i.e., $W = [w_1, \ldots, w_m]$.
- Note: this is still a non-convex optimization problem.

Coupling Ultra-wide NNs and NTK

First calculate $H(0)$.
Show as $m \to \infty$, $H(0)$ converges to a fixed matrix H^*.

Note $\frac{\partial f(a,W,x_i)}{\partial w_r} = \frac{1}{\sqrt{m}} a_r x_i \sigma'\left(w_r^\top x_i\right)$.
Each entry of $H(0)$ admits the formula

$$
\begin{aligned}
[H(0)]_{ij} &= \sum_{r=1}^{m} \left\langle \frac{\partial f\left(a, W(0), x_i\right)}{\partial w_r(0)}, \frac{\partial f\left(a, W(0), x_j\right)}{\partial w_r(0)} \right\rangle \\
&= \sum_{r=1}^{m} \left\langle \frac{1}{\sqrt{m}} a_r x_i \dot{\sigma}\left(w_r(0)^\top x_i\right), \frac{1}{\sqrt{m}} a_r x_j \sigma'\left(w_r(0)^\top x_i\right) \right\rangle \\
&= x_i^\top x_j \cdot \frac{\sum_{r=1}^{m} \sigma'\left(w_r(0)^\top x_i\right) \sigma'\left(w_r(0)^\top x_j\right)}{m}
\end{aligned}
$$

Coupling Ultra-wide NNs and NTK

Lemma 2 (Perturbation on the Initialization, [Du et al., 2019b])

Fix some $\epsilon > 0$. *If* $m = \Omega\left(\epsilon^{-2} n^2 \log(n/\delta)\right)$, *then with probability at least* $1 - \delta$ *over* $w_1(0), \cdots, w_m(0)$, *we have*

$$\|H(0) - H^*\|_2 \leqslant \epsilon$$

Note that:

- $[H(0)]_{ij} = x_i^\top x_j \cdot \frac{\sum_{r=1}^m \sigma'\left(w_r(0)^\top x_i\right) \sigma'\left(w_r(0)^\top x_j\right)}{m}$.
- $H_{ij}^* \triangleq x_i^\top x_j \cdot \mathop{\mathbb{E}}_{w \sim N(0,I)} \left[\sigma'\left(w^\top x_i\right) \sigma'\left(w^\top x_j\right)\right]$.

Proof of Lemma 2

We first fixed an entry (i, j). Note

$$\left| x_i^\top x_j \sigma' \left(w_r(0)^\top x_i \right) \sigma' \left(w_r(0)^\top x_j \right) \right| \leqslant 1$$

Applying Hoeffding inequality, we have with probability $1 - \frac{\delta}{n^2}$,

$$\left| [H(0)]_{i,j} - H^*_{i,j} \right| \leqslant \left(\frac{2}{m} \log \left(2n^2/\delta \right) \right)^{1/2} \leqslant 4 \left(\frac{\log(n/\delta)}{m} \right)^{1/2} \leqslant \frac{\epsilon}{n}$$

Next, applying the union bound over all pairs $(i, j) \in [n] \times [n]$, we have for all (i, j), $\left| [H(0)]_{i,j} - H^*_{i,j} \right| \leqslant \frac{\epsilon}{n^2}$. Then:

$$\begin{aligned} \left\| H(0) - H^* \right\|_2 &\leqslant \left\| H(0) - H^* \right\|_F \\ &= \left(\sum_{ij} \left| [H(0)]_{i,j} - H^*_{i,j} \right|^2 \right)^{1/2} \\ &\leqslant \left(n^2 \cdot \frac{\epsilon^2}{n^2} \right)^{1/2} = \epsilon \end{aligned}$$

Coupling Ultra-wide NNs and NTK

We proceed to show during training, $H(t)$ is close to $H(0)$.

Lemma 3

Assume $y_i = O(1)$ for all $i = 1, \cdots, n$. Given $t > 0$, suppose that for all $0 \leqslant \tau \leqslant t$, $u_i(\tau) = O(1)$ for all $i = 1, \cdots, n$. If $m = \Omega\left(\frac{n^6 t^2}{\epsilon^2}\right)$ we have

$$\|H(t) - H(0)\|_2 \leqslant \epsilon$$

Proof of Lemma 3

The first key idea is to show that every weight vector only moves little if m is large.

$$
\begin{aligned}
\|w_r(t) - w_r(0)\|_2 &= \left\| \int_0^t \frac{dw_r(\tau)}{d\tau} d\tau \right\|_2 \\
&= \left\| \int_0^t \frac{1}{\sqrt{m}} \sum_{i=1}^n (u_i(\tau) - y_i)\, a_r x_i \dot{\sigma}\left(w_r(\tau)^\top x_i\right) d\tau \right\|_2 \\
&\leqslant \frac{1}{\sqrt{m}} \int \left\| \sum_{i=1}^n (u_i(\tau) - y_i)\, a_r x_i \dot{\sigma}\left(w_r(\tau)^\top x_i\right) \right\|_2 d\tau \\
&\leqslant \frac{1}{\sqrt{m}} \sum_{i=1}^n \int_0^t \left\| u_i(\tau) - y_i a_r x_i \dot{\sigma}\left(w_r(\tau)^\top x_i\right) \right\|_2 d\tau \\
&\leqslant \frac{1}{\sqrt{m}} \sum_{i=1}^n \int_0^t O(1) d\tau = O\left(\frac{tn}{\sqrt{m}}\right)
\end{aligned}
$$

Proof of Lemma 3

Next, we show this implies the kernel matrix $H(t)$ is close $H(0)$.

$$
\begin{aligned}
& [H(t)]_{ij} - [H(0)]_{ij} \\
= & \left| \frac{1}{m} \sum_{r=1}^{m} \left(\dot{\sigma}\left(w_r(t)^\top x_i \right) \dot{\sigma}\left(w_r(t)^\top x_j \right) - \dot{\sigma}\left(w_r(0)^\top x_i \right) \dot{\sigma}\left(w_r(0)^\top x_j \right) \right) \right| \\
\leqslant & \frac{1}{m} \sum_{r=1}^{m} \left| \max_r \dot{\sigma}\left(w_r(t)^\top x_i \right) \|x_i\|_2 \|w_r(t) - w_r(0)\|_2 \right| \\
& + \frac{1}{m} \sum_{r=1}^{m} \left| \max_r \dot{\sigma}\left(w_r(t)^\top x_i \right) \|x_i\|_2 \|w_r(t) - w_r(0)\|_2 \right| \\
= & \frac{1}{m} \sum_{r=1}^{m} O\left(\frac{tn}{\sqrt{m}} \right) = O\left(\frac{tn}{\sqrt{m}} \right).
\end{aligned}
$$

Several Remarks about Lemma 3

- **Remark 1:** The assumption that $y_i = O(1)$ is mild because in practice most labels are bounded by an absolute constant.
- **Remark 2:** The assumption on $u_i(\tau) = O(1)$ for all $\tau \leqslant t$ and m's dependency on t can be relaxed. This requires a more refined analysis (see [Du et al., 2019b]).
- **Remark 3:** One can generalize the proof for multi-layer neural network (see [Arora et al., 2019]).
- **Remark 4:** While we only prove the continuous time limit, it is not hard to show with small learning rate (discrete time) gradient descent, $H(t)$ is close to H^* (see [Du et al., 2019b]).

Equivalence to Kernel Regression

From the above lemmas, when the width goes to infinity, we have:

- $H(t)$ remains constant during training, i.e., equals $H(0)$.
- $H(0)$ converges in probability to a certain deterministic kernel matrix H^*, which is the Neural Tangent Kernel.

If $H(t) = H^*$ for all t, then we have:

$$\frac{du(t)}{dt} = -H^* \cdot (u(t) - y)$$

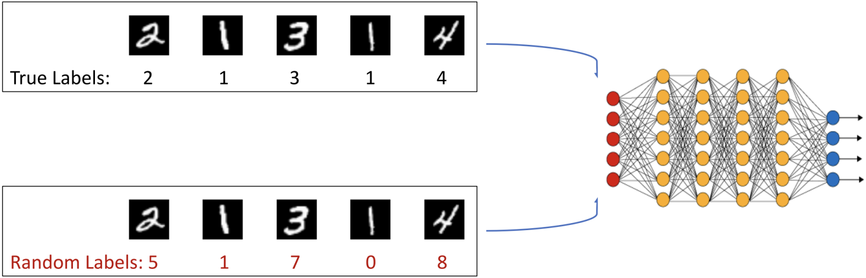
Note: this dynamics is identical to the dynamics of kernel regression under gradient flow, for which at time $t \to \infty$ the final prediction function is (assuming $u(0) = 0$)

$$f^*(x) = (k(x, x_1), \cdots, k(x, x_n)) \cdot (H^*)^{-1} y.$$

Overview

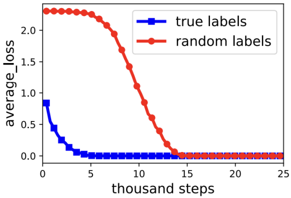
True vs Random Labels

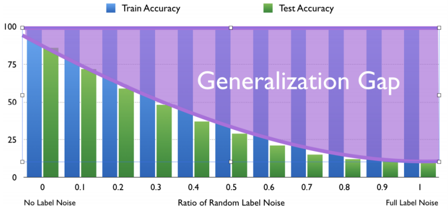
"Understanding deep learning requires rethinking generalization" [Zhang et al., 2017].



Phenomena

- Faster convergence with true labels than random labels.
- Good generalization with true labels, poor generalization with random labels.





Understanding Optimization

The dynamics of $u(t)$ that follows

$$\frac{du(t)}{dt} = -H^* \cdot (u(t) - y).$$

We denote the eigenvalue decomposition of H^* as

$$H^* = \sum_{i=1}^{n} \lambda_i v_i v_i^\top,$$

where $\lambda_1 \geqslant \ldots \geqslant \lambda_n \geqslant 0$ are eigenvalues and $v_1, \ldots, v_n$ are eigenvectors.
Consider the dynamics of $u(t)$ on each eigenvector separately,

$$\begin{aligned} \frac{dv_i^\top u(t)}{dt} &= -v_i^\top H^* \cdot (u(t) - y) \\ &= -\lambda_i \left(v_i^\top (u(t) - y)\right). \end{aligned}$$

Understanding Optimization

The ODE admits an analytical solution:

$$v_i^\top (u(t) - y) = \exp(-\lambda_i t) \left(v_i^\top (u(0) - y) \right)$$

Each component $v_i(u(t) - y)$ converges to 0 at a different rate.

Concretely,

Theorem 4 (Informal, [Arora et al., 2019])

Training loss at time t is:

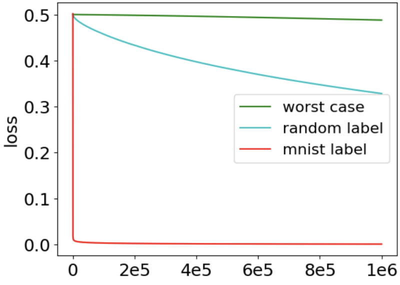
$$\sum_{i=1}^{n} (f(w(t), x_i) - y_i)^2 \approx \sum_{i=1}^{n} e^{-2\lambda_i t} \cdot \langle v_i, y \rangle^2$$

Understanding Optimization

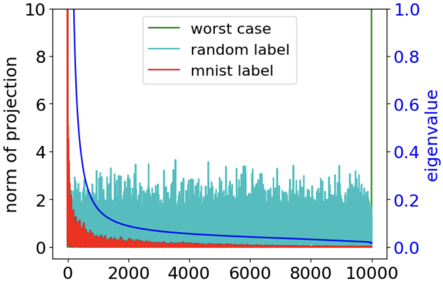
- Components of y on larger eigenvectors of H^* converge faster than those on smaller eigenvectors.
- If y aligns well with top eigenvectors, then GD converges quickly.
- If y's projections on eigenvectors are close to being uniform, then GD converges slowly.

Understanding Optimization

MNIST (2 Classes)



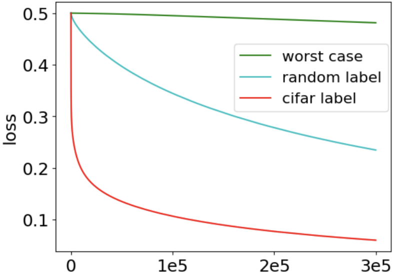
Convergence Rate



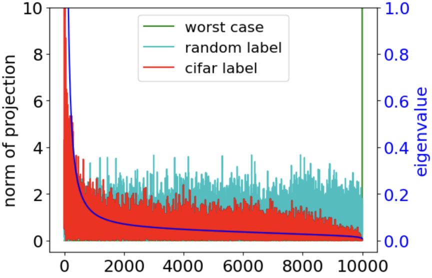
Projections

Understanding Optimization

CIFAR-10 (2 Classes)



Convergence Rate



Projections

Understanding Generalization

- It suffices to analyze generalization for $f_{\mathrm{ker}}(x) = (k(x, x_1), \cdots, k(x, x_n))(H^*)^{-1} y$
- For $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$, its RKHS norm is $\|f\|_{\mathcal{H}} = \sqrt{\alpha^\top H^* \alpha} \Rightarrow \|f_{\mathrm{ker}}\|_{\mathcal{H}} = \sqrt{y^\top (H^*)^{-1} y}$

- From the Rademacher complexity bound, the population loss bound for f_{ker} [Bartlett and Mendelson, 2002]:

$$\mathbb{E}_{(x,y)\sim D} |f_{\mathrm{ker}}(x) - y| \leqslant \frac{2\sqrt{y^\top (H^*)^{-1} y}\sqrt{\operatorname{tr}[H^*]}}{n} + \sqrt{\frac{\log(1/\delta)}{n}}$$

- $\operatorname{tr}[H^*] = O(n) \Rightarrow \mathbb{E}_{(x,y)\sim D} |f_{\mathrm{ker}}(x) - y| \leqslant O\left(\sqrt{\frac{y^\top (H^*)^{-1} y}{n}}\right)$

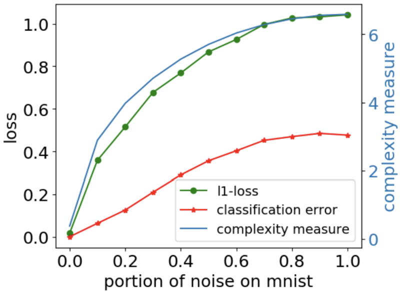
Understanding Generalization

- Consider binary classification $(y_i = \pm 1)$.
- We have the bound on classification error:

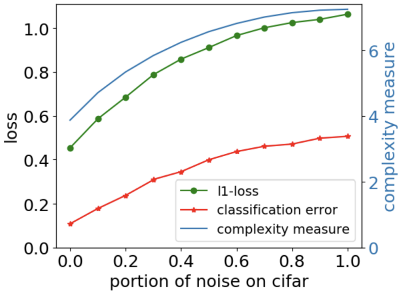
$$\Pr_{(x,y)\sim D}\left[\mathrm{sign}\left(f_{\mathrm{ker}}(x)\right) \neq y\right] \leqslant \sqrt{\frac{2y^\top (H^*)^{-1} y}{n}}$$

- This bound is a priori (can be evaluated before training).
- Can this bound distinguish true and random labels?

Understanding Generalization



MNIST

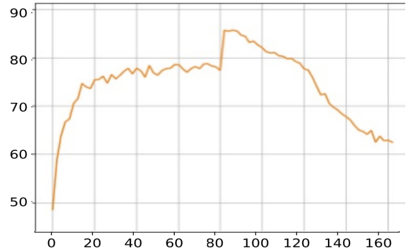


CIFAR-10

Overview

Training with Noisy Labels

- Noisy labels lead to degradation of generalization

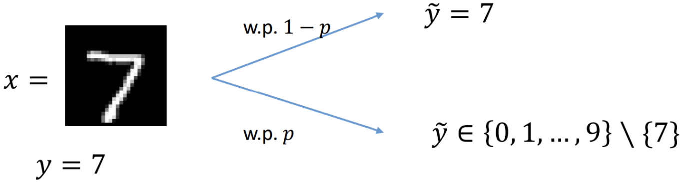


Test accuracy curve on CIFAR-10 trained with 40% label noise

- Early stopping is useful empirically
- But when to stop? (need access to clean validation set)

Setting

- In general, there is a transition matrix P such that $P_{i,j} = \Pr[\,\text{class } j \to \text{class } i\,]$



- Goal: Given a noisily labeled dataset $S = \{(x_i, \widetilde{y}_i)\}_{i=1}^{n}$, train a NN $f(w, \cdot)$, to get small loss on the clean data distribution $\mathcal{D}$

$$L_{\mathcal{D}}(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\,\ell(f(w,x), y)$$

NTK Viewpoint

- $\ell(w) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(w, x_i\right) - \tilde{y}_i\right)^2 \longleftrightarrow$
  $f_{\text{ker}}(x) = \left(k\left(x, x_1\right), \cdots, k\left(x, x_n\right)\right) \cdot \left(H^*\right)^{-1} \cdot \tilde{y}$

- Kernel ridge regression was shown to perform comparably to early-stopped gradient descent [Wei et al., 2017].

- What kinds of loss functions correspond to the kernel ridge regression?

$$f_{\text{ridge}}\left(x\right) = \left(k\left(x, x_1\right), \ldots, k\left(x, x_n\right)\right) \cdot \left(\left(H^*\right) + \lambda I\right)^{-1} \cdot \tilde{y}$$

NTK Viewpoint

- Method 1: Regularization using Distance to Initialization (RDI).

$$\ell_{\lambda}^{\mathrm{RDI}}(w) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(w, x_i\right) - \tilde{y}_i\right)^2 + \frac{\lambda}{2}\|w - w(0)\|^2$$

NTK Viewpoint

- Method 1: Regularization using Distance to Initialization (RDI).

$$\ell_{\lambda}^{\mathrm{RDI}}(w) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(w, x_i\right) - \tilde{y}_i\right)^2 + \frac{\lambda}{2}\|w - w(0)\|^2$$

- Method 2: Adding an AUXiliary variable for each training example (AUX).

$$\ell_{\lambda}^{\mathrm{AUX}}(w, b) = \frac{1}{2}\sum_{i=1}^{n}\left(f\left(w, x_i\right) + \sqrt{\lambda} b_i - \tilde{y}_i\right)^2$$

Theorem 5 (Informal, [Hu et al., 2020])

For infinitely wide NN, both methods lead to kernel ridge regression.

Generalization Bound

- $y \in \{\pm 1\}^n$: true labels
- $\tilde{y} \in \{\pm 1\}^n$: observed labels, each label being flipped w.p. $p(p < 1/2)$
- Goal: analyze generalization of $f_{\text{ridge}}(x) = (k(x, x_1), \ldots, k(x, x_n)) \cdot (H^* + \lambda I)^{-1} \tilde{y}$ on the clean distribution $\mathcal{D}$

Theorem 6 (Informal, [Hu et al., 2020])

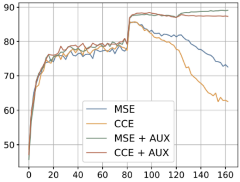
$$\Pr_{(x,y)\sim D}[\operatorname{sign}(f_{ridge}(x)) \neq y] \leqslant \frac{1}{1-2p} O\left(\sqrt{\lambda}\sqrt{\frac{y^\top K^{-1} y}{n}} + \frac{1}{\sqrt{\lambda}}\right)$$

- Set $\lambda = n^\alpha$

Experiments

Noise level	0	0.2	0.4	0.6
CCE normal (early stop)	94.05	89.73	86.35	79.13
MSE normal (early stop)	93.88	89.96	85.92	78.68
CCE+AUX	94.22	92.07	87.81	82.60
MSE+AUX	**94.25**	**92.31**	**88.92**	**83.90**
[Zhang and Sabuncu. 2018]	-	89.83	87.62	82.70

Test accuracy of ResNet-34 using different methods at various noise levels



Test accuracy curve for 40% label noise

- Training with AUX achieves very good accuracy, better than normal training with early stopping.
- Training with AUX doesn't over-fit (no need to stop early)

NTK on Small to Medium Scale Datasets

- Tested the NTK classifier on 90 small to medium scale datasets from UCI database [Arora et al., 2020].
- NTK can beat NNs.
- NTK can beat other kernels like Gaussian and the best previous classifier – random forest.

Classifier	Friedman Rank	Average Accuracy	P90	P95	PMA
NTK	**28.34**	**81.95%±14.10%**	**88.89%**	**72.22%**	**95.72% ±5.17%**
NN (He init)	40.97	80.88%±14.96%	81.11%	65.56%	94.34% ±7.22%
NN (NTK init)	38.06	81.02%±14.47%	85.56%	60.00%	94.55% ±5.89%
RF	33.51	[illegible]1.56% ±13.90%	85.56%	67.78%	95.25% ±5.30%
Gaussian Kernel	35.76	81.03% ± 15.09%	85.56%	**72.22%**	94.56% ±8.22%
Polynomial Kernel	38.44	78.21% ± 20.30%	80.00%	62.22%	91.29% ±18.05%

Table 1: Comparisons of different classifiers on 90 UCI datasets. P90/P95: the number of datasets a classifier achieves 90%/95% or more of the maximum accuracy, divided by the total number of datasets. PMA: average percentage of the maximum accuracy.

Graph NTK and CNTK

- For every NN architecture, one can derive a corresponding kernel function.

- Graph NTK (GNTK) has been derived for graph classification tasks [Du et al., 2019a].
- GNTK can outperform graph neural networks on diverse appls.

- Convolutional NTK (CNTK) was derived for convolutional neural networks [Arora et al., 2019].

References I

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019).
On exact computation with an infinitely wide neural net.
In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8139–8148.

Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., and Yu, D. (2020).
Harnessing the power of infinitely wide deep nets on small-data tasks.
In *International Conference on Learning Representations*.

Bartlett, P. L. and Mendelson, S. (2002).
Rademacher and gaussian complexities: Risk bounds and structural results.
Journal of Machine Learning Research, 3(Nov):463–482.

Du, S. S., Hou, K., Salakhutdinov, R. R., Poczos, B., Wang, R., and Xu, K. (2019a).
Graph neural tangent kernel: Fusing graph neural networks with graph kernels.
Advances in Neural Information Processing Systems, 32:5723–5733.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2019b).
Gradient descent provably optimizes over-parameterized neural networks.
In *International Conference on Learning Representations*.

References II

Hu, W., Li, Z., and Yu, D. (2020).
Simple and effective regularization methods for training on noisily labeled data with generalization guarantee.
In *International Conference on Learning Representations*.

Jacot, A., Gabriel, F., and Hongler, C. (2018).
Neural tangent kernel: Convergence and generalization in neural networks.
In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems, 25:1097–1105.

Wei, Y., Yang, F., and Wainwright, M. J. (2017).
Early stopping for kernel boosting algorithms: A general analysis with localized complexities.
arXiv preprint arXiv:1707.01543.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017).
Understanding deep learning requires rethinking generalization.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

References III

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021).
Understanding deep learning (still) requires rethinking generalization.
Communications of the ACM, 64(3):107–115.