Tutorial: Beyond Linearization in Deep Learning

Jason D. Lee

Princeton University
Subjective statements are all due to JDL. Theorems are joint work.
Many missing references

July 16, 2021

1. Linearization in Deep Learning
2. Beyond Linearization
3. More Recent Results.
4. Open Directions

Feel free to interrupt with questions.

What I will cover.

Optimization in Deep Learning.

I will focus on **gradient-based** algorithms for minimizing the train loss.

Categorize into three classes:

- Proofs only use gradient non-vanishing.
- Proofs use higher-order derivatives, though the **algorithms do not**.
- Proofs explicitly analyze **sequential order of learning**.

Examples from Machine Learning.

Category 1:

- All of convex optimization.
- Single index/neuron models $E(y - \sigma(w^T x))^2 +$ well-specified[1] .
- One-point convexity.
- Star convexity.
- "Local convergence"

Category 2:

- Matrix factorization/completion[2].
- Tensor decomposition[3].
- Phase retrieval[4].
- Avoiding saddle-points.
- ReLU network with designed loss[5].

[1] Kalai and Sastry, Kakade, Kalai, Kanade and Shamir, Soltanolkotabi, Mei,Bai, Montanari

[2] Ge, Lee, and Ma

[3] Ge, Huang, Jin, and Yuan

[4] Sun, Qu, and Wright

[5] Ge, Lee, and Ma

Examples (continued)

Category 3 (generally require more specific assumptions on $y|x$):

- Label noise SGD[6]
- Hierarchical structures: 3- layer networks, adversarial training, backward feature correction, resnets[7]
- Learning orthogonal ReLU network [8]
- Large learning rate[9]

Category 3 proofs generally look like chaining together steps of category 1 and 2 proofs with careful induction and argument about learning order (alternate between using category 1 and 2).

---

[6] HaoChen, Wei, Lee, and Ma

[7] e.g. first learn hidden layer, then approximately learn next layer, then refine, Allen-Zhu & Li, Chen, Bai, Lee, Zhao et al.

[8] Li, Ma, and Zhang

[9] Li, Wei, and Ma.

Today's Plan

I will cover two general principles: learning as well as first-order information and learning as well as second-order information (roughly Category 1 and Category 2).

- Category 3 will not be explicitly covered, but given a mastery of the techniques in Category 1&2 not too hard to piece together a Category 3 proof.
- Category 1 is the most generally applicable and is the **linearization principle**.

Linearization in Deep Learning

Consider $f_\theta(\cdot)$ is any nonlinear function.

$$f_\theta(x) \approx \underbrace{f_0(x)}_{\approx 0} + \nabla_\theta f_\theta(x)^\top (\theta - \theta_0) + O(\|\theta - \theta_0\|^2),$$

Linearization in Deep Learning

Consider $f_\theta(\cdot)$ is any nonlinear function.

$$f_\theta(x) \approx \underbrace{f_0(x)}_{\approx 0} + \nabla_\theta f_\theta(x)^\top (\theta - \theta_0) + O(\|\theta - \theta_0\|^2),$$

Assumptions:

- Second order term is "negligible".
- f_0 is negligible, which can be argued using initialization+overparametrization.

Interpretation due to:

- Kernel Viewpoint: Jacot et al., (Du et al.)[2], (Arora et al.) [2], and Chizat & Bach.
- Pseudo-network: Li and Liang, Allen-Zhu et al., Zou et al.

Tangent Kernel

Under these assumptions,

$$f_\theta(x) \approx \hat{f}_\theta(x) = (\theta - \theta_0)^\top \nabla_\theta f(\theta_0)$$

- This is a **linear** classifier in θ.
- Feature representation is $\phi(x; \theta_0) = \nabla_\theta f(\theta_0)$.

Tangent Kernel

Under these assumptions,

$$f_\theta(x) \approx \hat{f}_\theta(x) = (\theta - \theta_0)^\top \nabla_\theta f(\theta_0)$$

- This is a **linear** classifier in θ.
- Feature representation is $\phi(x; \theta_0) = \nabla_\theta f(\theta_0)$.

Corresponds to using the kernel

$$K = \nabla f(\theta_0)^\top \nabla f(\theta_0).$$

Interlude: What is this kernel?

Neural Tangent Kernel (NTK)

$$K = \sum_{l=1}^{L+1} \alpha_l K_l \text{ and } K_l = \nabla_{W_l} f(\theta_0)^\top \nabla_{W_l} f(\theta_0)$$

Two-layer

$$K_1 = \sum_j a_j^2 \sigma'(w_j^\top x)\sigma'(w_j^\top x')x^\top x' \text{ and } K_2 = \sum_j \sigma(w_j^\top x)\sigma(w_j^\top x')$$

Interlude: Kernel is initialization dependent

$$K_1 = \sum_j a_j^2 \sigma'(w_j^\top x)\sigma'(w_j^\top x')x^\top x' \text{ and } K_2 = \sum_j \sigma(w_j^\top x)\sigma(w_j^\top x')$$

so how a, w is initialized matters a lot.

- Imagine $\|w_j\|^2 = 1/d$ and $|a_j|^2 = 1/m$, then only $K = K_2$ matters (Daniely, Rahimi-Recht).
- "NTK parametrization": $f_\theta(x) = \frac{1}{\sqrt{m}} \sum_j a_j \sigma(w_j x)$, and $|a_j| = O(1)$, $\|w\| = O(1)$, then

  $$K = K_1 + K_2.$$

  This is what is done in Jacot et al., Du et al, Chizat & Bach
- Li and Liang consider when $|a_j| = O(1)$ is fixed, and only train w,

  $$K = K_1.$$

Interlude: Initialization and LR

Through different initialization/ parametrization/layerwise learning rate, you can get

$$K = \sum_{l=1}^{L+1} \alpha_l K_l \text{ and } K_l = \nabla_{W_l} f(\theta_0)^\top \nabla_{W_l} f(\theta_0)$$

- NTK should be thought of as this family of kernels.
- Rahimi-Recht, Daniely studied the special case where only K_2 matters and the other terms disappear (tangent kernel wrt only output layer).

Interlude: Infinite-width

For theoretical analysis, it is convenient to look at infinite width to remove the randomness from initialization.

Infinite-width

Initialize $a_j \sim N(0, s_a^2/m)$ and $w_j \sim N(0, s_w^2 I/m)$.
Then

$$K_1 = s_a^2 E_w[\sigma'(w_j^\top x)\sigma'(w_j^\top x')x^\top x']$$
$$K_2 = s_w^2 E_w[\sigma(w_j^\top x)\sigma(w_j^\top x')].$$

These have ugly closed forms in terms of $x^\top x', \|x\|, \|x'\|$.

Interlude: Infinite-Width

Deep net Infinite-Width

Let $a^{(l)} = W_l \sigma(a^{(l-1)})$ be the pre-activations with $\sigma(a^{(0)}) := x$. When the widths $m_l \to \infty$, the pre-activations follow a Gaussian process. These have covariance function given by:

$$\Sigma^{(0)} = x^\top x'$$

$$A^{(l)} = \begin{bmatrix} \Sigma^{(l-1)}(x,x) & \Sigma^{(l-1)}(x,x') \\ \Sigma^{(l-1)}(x',x) & \Sigma^{(l-1)}(x',x') \end{bmatrix}$$

$$\Sigma^{(l)}(x,x') = \mathbb{E}_{(u,v)\sim A^{(l)}}[\sigma(u)\sigma(v)].$$

$\lim_{m_l \to \infty} K_{L+1} = \Sigma^{(L)}$ gives us the kernel of the last layer (Lee et al., Matthews et al.).
Define the gradient kernels as $\dot{\Sigma}^{(l)}(x,x') = \mathbb{E}_{(u,v)\sim A^{(l)}}[\sigma'(u)\sigma'(v)]$. Jacot et al. , Lee et al., Du et al., Yang, Arora et al. show

$$K_l(x,x') = \Sigma^{(l-1)}(x,x') \cdot \Pi_{l'=l}^{L} \dot{\Sigma}^{(l')}(x,x')$$

Interlude: Statistical Behavior of the Kernel

TLDR

NTK is roughly the same as Laplace kernel (also similar to Gaussain RBF /polynomial kernel) for sample complexity.

- More precise statements in second half of the talk.
- **Let's turn to Optimization, where the linearization principle shines.**

Linearization as a tool for analyzing Optimization

Let's consider the simplest non-convex setting of

$f(\theta, x) = \sum_{j=1}^{m} a_j \sigma(w_j^\top x)$ and $\ell(y, \hat{y}) = (y - \hat{y})^2$ and only train w_j's

Optimization with Linearized Model

Imagine that $f(\theta, x) = \hat{f}_\theta(x) := (\theta - \theta_0)^\top \nabla f(\theta_0; x)$.

- Loss is convex in θ.
- Satisfies the PL condition iff $\sigma_{\min}(J(\theta)) > 0$ and this ensures linear convergence.

Proof sketch follows Soltanolkotabi, Javanmard, and Lee, which first used the linearization (?).

Proof continued

The two remaining steps:

1. Show that $\hat{f}_\theta(x) \approx f_\theta(x)$ for all θ encountered by GD.
2. Show that $\sigma_{\min}(J(\theta)) > 0$ for all θ encountered by GD.

This can be done via proving the following two intermediate results:

1. Showing $J(\theta)$ is sufficiently Lipschitz, or equivalently $\nabla^2 f_\theta(x)$ is small.
2. $\sigma_{\min}(J(\theta))$ via random matrix theory.

Lipschitz Jacobian Lemma

Lemma (Lemma 7.11 of SJL17)

$$\|J(\bar{W}) - J(W)\|_2 \leq |\sigma'|_\infty \cdot \|x \otimes x\|_2 \cdot \|a\|_\infty \cdot \|W - \bar{W}\|_2$$

- This is a deterministic lemma – no data assumptions used yet.

Minimum Singular Value Lemma

Lemma (Proposition 7.4 of SJL17)

Assume x_i are iid Gaussian, and $m \geq d$. Then

$$\sigma_{\min}(J(W_0)) \gtrsim d \cdot \sigma_{\min}(W_0)\|a\|_\infty.$$

Putting it together.

Theorem (Theorem 2.5 of SJL17 with rescaling)

For Gaussian data, $md \gtrsim n$ and $m \geq d$ and any full rank initialization W_0, gradient descent decreases the loss to zero (and thus finds a global min)

- We do not explicitly construct a global min nearby, but we show the loss goes to 0 and W_t stays near W_0. Together, these imply the existence of a global minimum near W_0.
- The analysis on the required width is sharp up to polylog. $m = n/d$ is necessary.
- Zhong and Montanari 2020 prove the same theorem when $m \leq d$, $md \gtrsim n$, for random initialization.

Comparison

- Compared to the more recent NTK papers, the width is independent of n. This step requires more heavyweight RMT.
- Remark: For any data/width with $\sigma_{\min}(J(W_0)) > 0$, the theorem holds; Gaussian data is not important except that we could give a good quantitative bound on $\sigma_{\min}$. Thus reduces to proving a RMT statement at initialization.
- We **did not** establish that the behavior is approximating the infinite-width kernel. In fact at $m = n/d$, the finite-width kernel is not necessarily close to the infinite-width.

One of the Most "Natural" Questions in DL Theory

Two-layer Teacher Network.

$$f^*(x) = \sum_{j=1}^{r} a_j^* \sigma(w_j^{*\top} x).$$

Problem

Given data from a two-layer teacher, learn to accuracy

$$\mathbb{E}(f^*(x) - f(x))^2 < \epsilon.$$

What is a good result?

Potential results by strength

- Information-theoretic sample complexity is $n \asymp dr$, which is attainable in well-specified case.
- Very good result if computationally efficient: $n \asymp \mathrm{poly}(d, r)$.
- If σ is monomial/polynomial, polynomial kernel will attain $n \asymp d^{\mathsf{deg}(\sigma)} r^2$.

Lower bounds

We shouldn't stop trying because of lower bounds, but we should know what they say.

Lower bounds for discrete distribution (Adam Klivans).

These results rule out any algorithm that does not utilize distribution assumptions/assumptions on W^* that learn in $\mathrm{poly}(d, r)$ (probably even $d^{o(r)}$ is hard).

- Learning intersection of halfspaces (Klivans & Sherstov, Livni et al.)
- Decision trees
- Juntas

Lower bounds II and "breaking" lower bounds

Distribution-specific lower bounds

Even if $p(x)$ is isotropic Gaussian, there are some recent negative results (SQ lower bounds):

- For ReLU teacher network, need d^r queries (Diakonikolas et al., similar result by Goel et al.)

Lower bounds II and "breaking" lower bounds

Distribution-specific lower bounds

Even if $p(x)$ is isotropic Gaussian, there are some recent negative results (SQ lower bounds):

- For ReLU teacher network, need d^r queries (Diakonikolas et al., similar result by Goel et al.)

We should not be discouraged by lower bounds as long as we have algorithmic ideas:

Algorithms (well-specified case, parameter recovery)

- Spectral Methods (Janzamin et al., Zhong et al.): Learn teacher networks if $r \leq d$ and $\sigma_{\min}(W^*) > 0$.
- Tensor method in disguise (GLM, Li et al.): SGD can estimate parameters but under even stronger assumptions.

Functional Analysis Viewpoint

Function spaces

For $f^*(x) = \sum_{j=1}^m a_j \sigma(w_j^\top x)$, write it as

$$f(x) = \int \rho(w)\sigma(w^\top x)dw = \rho^\top \phi(x),$$

where $\phi(x)[w] = \sigma(w^\top x)$ with index set $w \in S^{d-1}$.
Two "natural" function spaces:

- $F_2(B) = \{f : f(x) = \rho^\top \phi(x), \|\rho\|_2^2 \leq B^2\}$ is an ℓ_2 space (RKHS, Rahimi-Recht,Cho and Saul)
- $F_1(B) = \{f : f(x) = \rho^\top \phi(x), \|\rho\|_1 \leq B\}$ is an ℓ_1 sparsity-type space known as convex neural net (Banach, Bengio et al., Bach, ...)

Summary of what is known for two spaces.

F_1

- The global minimum of ℓ_2-norm on all parameters is F_1.
- Mean field aims to learn all of F_1, so does implicit regularization (Chizat-Bach, Nacson et al., Lyu et al., Wei et al.)
- F_1 **adapts to low-dimensional structure**.
  1. $n \asymp d\|f^*\|_{F_1}^2/\epsilon^2$
  2. If $f^*(x) = p(Ux)$, for $U = r \times d$ and $p(\cdot)$ is degree q polynomial, then $n_{F_1}(p(Ux)) = dr^{2q}$.
  3. Learn width r teacher networks in complexity $n_{F_1}(\text{teacher net}) = d \cdot \text{poly}(r)$.
- F_1 almost certainly **cannot be efficiently learned, since it includes two-layer teacher networks of width** r. All the previously mentioned lower bounds apply to F_1.
- Opinion: Unlikely that mean field or any SGD approach will yield polynomial-time learning of F_1.

Summary of what is known for two spaces.

F_2

- Computationally efficient via SGD.
- Does not adapt to low-dimensional structure. If $f^*(x) = p(Ux)$, then $n_{F_2} \asymp d^q$ even if $U = 1 \times d$.
- Teacher network has $\gtrsim e^d$ (probably infinite) F_2 norm (Yehudai and Shamir) and sample complexity $n_{F_2} \gtrsim d^{\text{poly}(1/\epsilon)}$.
- Essentially the same as NTK as far as theoretical results bounds go.

Finding the sweet spot.

Goal.

The goal is to find an in-between space that

- Adapts to low-dimensional structure
- Computationally tractable via SGD.

Finding the sweet spot.

Goal.

The goal is to find an in-between space that

- Adapts to low-dimensional structure
- Computationally tractable via SGD.

Obvious guess is ℓ_p for $1 < p < 2$, but I don't think this leads to tractable algorithms.

Motivation

Monomial Activation

For quadratic activation (and monomial activation),

$$\int \rho(w)(w^\top x)^2 = \langle \int \rho(w) w w^\top, x x^\top \rangle.$$

- F_2 is a frobenius norm inductive bias.
- F_1 is a nuclear norm inductive bias.

Motivation

Monomial Activation

For quadratic activation (and monomial activation),

$$\int \rho(w)(w^\top x)^2 = \langle \int \rho(w) w w^\top, x x^\top \rangle.$$

- F_2 is a frobenius norm inductive bias.
- F_1 is a nuclear norm inductive bias.

- This suggests rank as a measure of "low-dimensional" latent structure.
- For monomial activation, this corresponds to the width of the teacher network.

Low-rank Polynomials

Definition

(Low rank polynomial) f^* is a rank r polynomial of degree p if

$$f^*(x) = \sum_{s=1}^{r} a_s^* (w_s^{*\top} x)^{p_s},$$

where $|a_s^*| \leq 1$, $\mathbb{E}[(w_s^{*\top} x)^{2p_s}] \leq 1$, and $p_s \leq p$.

Low-rank Polynomials

Definition

(Low rank polynomial) f^* is a rank r polynomial of degree p if

$$f^*(x) = \sum_{s=1}^{r} a_s^* (w_s^{*\top} x)^{p_s},$$

where $|a_s^*| \leq 1$, $\mathbb{E}[(w_s^{*\top} x)^{2p_s}] \leq 1$, and $p_s \leq p$.

- Teacher networks with polynomial activation of bounded degree and analytic activation (approximately).
- Constant depth teacher networks with polynomial activation.
- Real reason: I will show you a non-trivial learning guarantee with SGD.

Baseline approaches

- Using F_2 to learn this class needs $\gtrsim d^p$ samples.
- Using F_1 to learn needs at most $d \cdot \mathrm{poly}(r)$ samples (nearly information-theoretic optimal).
- SGD+Signed Dropout needs $d^{p-1} \cdot \mathrm{poly}(r, p)$ samples (via the Quadratic NTK proof technique in Bai and Lee).

Baseline approaches

- Using F_2 to learn this class needs $\gtrsim d^p$ samples.
- Using F_1 to learn needs at most $d \cdot \mathrm{poly}(r)$ samples (nearly information-theoretic optimal).
- SGD+Signed Dropout needs $d^{p-1} \cdot \mathrm{poly}(r, p)$ samples (via the Quadratic NTK proof technique in Bai and Lee).

Still a very large gap between $d \cdot \mathrm{poly}(r)$ and $d^{p-1} \cdot \mathrm{poly}(r, p)$.

Main Theorem

Theorem

SGD+Signed Dropout on a three-layer neural net architecture (polynomial width) learns with $n \asymp d^{p/2} \cdot \frac{\mathrm{poly}(r,p)}{\epsilon^4}$ *in time* $n \cdot \mathrm{poly}(d, r, p, \frac{1}{\epsilon})$.

Assumption:

- Moment assumptions on x. Any multivariate Gaussian or elliptical distribution on sphere is fine.

How to attain this.

Architecture

3-layer network:

$$f(x) = \sum_{j=1}^{m} a_r \sigma(w_r^\top g(x))$$
$$g(x)_l = \sigma(v_l^\top x + b).$$

- We will only train w_r. The a_r, v_l are randomly initialized and fixed.
- It is crucial to have a 3-layer architecture, our results are not attainable with only a two-layer network (lower bound).

Alg: SGD with data-dependent regularizer

Step 1: Estimate covariance $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} g(x_i)g(x_i)^\top$
Step 2: Run SGD +Signed Dropout (AzLL, Bai and Lee) on

$$L(w_1, \ldots, w_m) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_W(x_i), y_i) + \lambda \|W\hat{\Sigma}^{1/2}\|_{2,4}^4$$

- Signed Dropout: Modify the gradient per neuron $z_r \nabla_{w_r} L(W)$ for z_r Rademacher.
- Prevents the linearized model from memorizing (NTK).
- Allows for learning rate $O(m^{0.25})$ larger than NTK.

Why it works.

Proof sketch (assuming I know the input is Gaussian and $f(x) = (\beta^\top x)^p$):

1. Let $g(x)_l$ be Hermite polynomial basis of degree $\frac{p}{2}$ for $1 \leq l \leq D := d^{p/2}$.
2. Via the hermite, we can express $(\beta^\top x)^{p/2} = \theta^\top g(x)$.
3. The second layer input is the hermite polynomials. It needs to learn $f(x) = (\theta^\top g(x))^2$.
4. QNTK is very good at learning quadratic functions of the input

$$(\theta^\top g(x))^2 = \sum_{j=1}^{m} \sigma''(w_{0,j}^\top g(x))(\theta^\top g(x))^2.$$

Potential Improvements

Probably within reach.

Train the first layer v_r's and show this can improve ϵ dependence.
Hope:

$$n \asymp d^{p/2} \cdot \frac{\text{poly}(r,p)}{\epsilon^4} \to n \asymp d^{p/2} \cdot \frac{\text{poly}(r,p)}{\epsilon^2}.$$

- Currently, $\frac{1}{\epsilon^2}$ is due to the first layer only ϵ approximating the basis of degree $p/2$ polynomials.
- We hope that by training the first hidden layer that the approximation error improves from $\epsilon \to 0$.

Bigger Questions

- Is $\mathrm{poly}(d)$ possible?

Bigger Questions

- Is $\mathrm{poly}(d)$ possible?
  A: There are reasons to believe that $d^{p/2}$ is a fundamental limit by analogy to tensor completion/sensing. SOS can get at best $d^{p/2}$ and conditional on hypergraphic planted clique (Luo and Zhang).

Bigger Questions

- What is a learnable function class for deeper teacher networks? I was thinking

$$f^*(x) = \sum_{s=1}^{r} \alpha_s (\beta_s^\top g_s^*(x))^{p_s},$$

where $|\alpha_s| \leq 1$, $\mathbb{E}[(\beta_s^\top x)^{2p_s}] \leq 1$, $p_s \leq p$, and g_s^* is coordinatewise low rank polynomial.

- Easier to learn algorithmically (but I still don't know how to prove):

$$f^*(x) = \sum_{s} v_s^\top g_s^*(x) + \sum_{s=1}^{r} \alpha_s (\beta_s^\top g_s^*(x))^{p_s}$$

Open Directions

Systematic understanding of SGD

- Category 2 & 3 are focused on going beyond linearization/kernel methods, but they are fairly limited in the assumptions on $y|x$ they can exploit. Can we more systematically understand how architecture/sgd exploits $y|x$?
- Category 3 is especially ad-hoc and fragile.
- Most analysis do not use much about $p(x)$ (low-dimensional manifold etc.).
- Interaction of architecture design, algorithm, and regularizer.
- Understanding SGD's ability to learn good representations, not only classifiers.

Thank you for listening.