DEEP LEARNING JP
[DL Papers]

“Neural Tangent Kernel: Convergence and Generalization in Neural Networks”

Kensuke Wakasugi, Panasonic Corporation.

書誌情報

タイトル：
Neural Tangent Kernel: Convergence and Generalization in Neural Networks （NIPS2018）[1]

著者：
Jacot, A., Gabriel, F., & Hongler, C （スイス連邦工科大学ローザンヌ校）

選書理由：
最近の深層学習の理論研究について興味があったため.
引用数437(2020/10/01時点)で，盛んに研究されていると思われるため.

※特に断りがない限り，本資料の図・表・式は上記論文より引用したものです.
※Neural Tangent Kernel(NTK)の理解にあたっては，下記ページがとても勉強になりました.
Rajat's Blog Understanding the Neural Tangent Kernel (https://rajatvd.github.io/NTK/)

[1] Jacot, A., Gabriel, F., & Hongler, C. (2018). In *Advances in neural information processing systems* (pp. 8571-8580).

背景

- NNは多様なタスクで高い汎化性能を示しているが，
  その理由を理論的に説明することができていない．

- 先行研究：
  ・隠れ層のwidth→∞のとき，NNがガウス過程とみなせる[2]
  ・初期化時（ランダムな重み）のLossの形状についての解析[3]

⇨ 任意の入力値に対し，
中心極限定理（width→∞のとき ）によって，層毎の共分散行列を数式で扱い，
パラメータ空間における，NNの出力値または損失関数の形状を解析

- 課題：
  ・学習中の挙動について扱えない
  学習が進むにつれ，重みがガウス分布に従うといった仮定がおけなくなる

[2] J. H. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. ICLR, 2018.
[3] R. Karakida, S. Akaho, and S.-i. Amari. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. jun 2018.

全体像

lossを最小化する際の各変数/関数の軌跡を考える

NNパラメータθ ⇒ 関数f ⇒ 出力y ⇒ 損失関数loss

学習データ$\bar{x}$ ⇒ 関数f

学習データ$\bar{y}$ ⇒ 損失関数loss

1、NNパラメータの更新式

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial loss}{\partial \boldsymbol{\theta}}$$

2、微分方程式とみなすと

$$\frac{\partial \boldsymbol{\theta}}{\partial t} = -\frac{\partial loss}{\partial \boldsymbol{\theta}} = -\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \overline{\boldsymbol{y}})$$

※ $\boldsymbol{y}$は複数の学習データを並べてベクトル化している

3、出力yの変化

$$\frac{\partial \boldsymbol{y}}{\partial t} = \frac{\partial \boldsymbol{y}^T}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial t} = -\frac{\partial \boldsymbol{y}^T}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \overline{\boldsymbol{y}})$$

4、Neural Tangent Kernel

$$\boldsymbol{\phi} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}, \boldsymbol{K} = \frac{\partial \boldsymbol{y}^T}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}$$

※ ϕはカーネル法でいうところの
高次元特徴量空間への写像関数

5、width→∞でK→const

$$\frac{\partial \boldsymbol{y}}{\partial t} = -\boldsymbol{K}(\boldsymbol{y} - \overline{\boldsymbol{y}})$$

6、 $\boldsymbol{d} = \boldsymbol{y} - \overline{\boldsymbol{y}}$について

$$\frac{\partial \boldsymbol{d}}{\partial t} = -\boldsymbol{K}\boldsymbol{d}$$

$$\boldsymbol{d}(t) = \boldsymbol{d}(0)e^{-\boldsymbol{K}t}$$

※$\boldsymbol{K}$は正定値行列で，固有値は収束の速さに対応する

Contribution

1. 勾配降下法がカーネルを用いて表現でき，
   このとき，NN関数f_θがNNの層数，非線形関数，
   初期化の分散のみに依存する、こと示した

2. NNの収束性が、NTKの正定性で議論できるようにした.

3. 二乗損失の場合、f_θが線形微分方程式に従い、
   ヤコビアンの固有値が収束性を表す.
   すなわち，固有関数ごとに収束性が異なることを示した.
   これは，early-stoppingを支持する結果.

4. 人工データとMNISTで、数値実験を実施.

$$\frac{\partial \boldsymbol{y}}{\partial t} = -\boldsymbol{K}(\boldsymbol{y} - \bar{\boldsymbol{y}})$$

$$\boldsymbol{K} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}^T \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{d}(t) = \boldsymbol{d}(0)e^{-\boldsymbol{K}t}$$

準備

一般的な形式でNNを記述

・seminorm

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x\sim p^{in}}\left[f(x)^T g(x) \right].$$

$$\langle f, g \rangle_K := \mathbb{E}_{x,x'\sim p^{in}}\left[f(x)^T K(x,x') g(x') \right].$$

※二つの関数間の距離のようなもの
カーネル法の文脈で登場している？
p^{in}は入力データの分布で，実際は学習データの経験分布を使う
期待値はΣになるか，ベクトルのノルムで置き換わる.

・NNの定式化

$$f_\theta(x) := \tilde{\alpha}^{(L)}(x;\theta)$$

$$\alpha^{(0)}(x;\theta) = x$$
$$\tilde{\alpha}^{(\ell+1)}(x;\theta) = \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x;\theta) + \beta b^{(\ell)}$$
$$\alpha^{(\ell)}(x;\theta) = \sigma(\tilde{\alpha}^{(\ell)}(x;\theta)),$$

※ $\tilde{\alpha}$は理論系の論文でよく見かける．中心極限定理はここで議論

準備：Kernel gradient

カーネルを用いて，損失関数Cの時間発展を記述

- 損失関数Cの微分をカーネルで表現

$$\nabla_K C|_{f_0}(x) = \frac{1}{N}\sum_{j=1}^{N} K(x, x_j) d|_{f_0}(x_j).$$

- この時，Cの時間発展（最小化の更新計算を時間とみなす）は下記のようになる

$$\partial_t C|_{f(t)} = -\left\langle d|_{f(t)}, \nabla_K C|_{f(t)} \right\rangle_{p^{in}} = -\left\| d|_{f(t)} \right\|_K^2 .$$

⇨ 仮にカーネルが正定値で，定数であれば，Cの時間発展はt→無限で0に収束

※この時点でNNは登場していないが，NNの最終層に関して同様の論理展開となる
また，width→無限で定数という議論が出てくる

$$\frac{\partial loss}{\partial \boldsymbol{\theta}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \overline{\boldsymbol{y}})$$

$$\frac{\partial loss}{\partial t} = \frac{\partial loss}{\partial \boldsymbol{\theta}}\frac{\partial \boldsymbol{\theta}}{\partial t}$$

$$= -\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \overline{\boldsymbol{y}})\right)^T \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \overline{\boldsymbol{y}})\right)$$

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x\sim p^{in}}\left[f(x)^T g(x)\right].$$

$$\langle f, g \rangle_K := \mathbb{E}_{x,x'\sim p^{in}}\left[f(x)^T K(x, x') g(x')\right].$$

Random functions approximation

Kernel gradientとNNの関係性についての例示

- 出力関数を任意関数の和で表現されるとする

$$\theta \mapsto f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p f^{(p)}.$$

※NNの最終層のイメージ.
最終層のパラメータ$\boldsymbol{\theta}_p$のみが学習対象
fはランダムにサンプリングされた関数

- この時の出力関数の微分

$$\partial_t f_{\theta(t)}^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \partial_t \theta_p(t) f^{(p)} = -\frac{1}{P} \sum_{p=1}^{P} \left\langle d|_{f_{\theta(t)}^{lin}}, f^{(p)} \right\rangle_{p^{in}} f^{(p)},$$

- 上式は，カーネルを下記で定義した場合のKernel gradientに対応

$$\tilde{K} = \sum_{p=1}^{P} \partial_{\theta_p} F^{lin}(\theta) \otimes \partial_{\theta_p} F^{lin}(\theta) = \frac{1}{P} \sum_{p=1}^{P} f^{(p)} \otimes f^{(p)}.$$

※補足

These functions define a random linear parametrization $F^{lin} : \mathbb{R}^P \to \mathcal{F}$

$$\theta \mapsto f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p f^{(p)}.$$

$$f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p f^{(p)}.$$

$$\frac{\partial \boldsymbol{y}}{\partial t} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial t}$$

$$= -\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}^T \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}} (\boldsymbol{y} - \overline{\boldsymbol{y}})$$

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x \sim p^{in}} \left[f(x)^T g(x) \right].$$

Neural tangent kernel

多層の場合も同様の形式で，カーネルで記述

- 前述の内容と同様に，勾配法がカーネルで記述される．

$$\partial_t f_{\theta(t)} = -\nabla_{\Theta^{(L)}} C|_{f_{\theta(t)}}$$

$$\Theta^{(L)}(\theta) = \sum_{p=1}^{P} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta).$$

ただし， Fがθに依存する（学習の進捗で変化する）

⇨ widthの無限極限においては，Fがコンスタントとみなせる

Initialization

初期化時のカーネルは，ガウス過程近似における共分散行列の漸化式から算出

- 深層学習のガウス過程近似

$$\Sigma^{(1)}(x,x') = \frac{1}{n_0} x^T x' + \beta^2$$
$$\Sigma^{(L+1)}(x,x') = \mathbb{E}_{f \sim \mathcal{N}\left(0,\Sigma^{(L)}\right)}[\sigma(f(x))\sigma(f(x'))] + \beta^2,$$

- カーネルの計算に発展

$$\Theta_\infty^{(1)}(x,x') = \Sigma^{(1)}(x,x')$$
$$\Theta_\infty^{(L+1)}(x,x') = \Theta_\infty^{(L)}(x,x')\dot{\Sigma}^{(L+1)}(x,x') + \Sigma^{(L+1)}(x,x'),$$

⇨ 初期化時のカーネル計算は可能

$$\frac{\partial \boldsymbol{y}^{L+1}}{\partial \boldsymbol{\theta}^{L+1}} = \frac{\partial \boldsymbol{w}^L \sigma(\boldsymbol{y}^L)}{\partial \boldsymbol{\theta}^{L+1}}$$
$$= \boldsymbol{w}^L \underbrace{\frac{\partial \sigma}{\partial \boldsymbol{y}^L}}_{\dot{\Sigma}} \frac{\partial \boldsymbol{y}^L}{\partial \boldsymbol{\theta}^L} + \underbrace{\sigma(\boldsymbol{y}^L)}_{\Sigma}$$

※最初の式を変更すると対応

$$\frac{\partial \boldsymbol{y}^{L+1}}{\partial \boldsymbol{\theta}^{L+1}} \to \frac{\partial \boldsymbol{y}^{L+1}}{\partial \boldsymbol{\theta}^{L+1}}^T \frac{\partial \boldsymbol{y}^{L+1}}{\partial \boldsymbol{\theta}^{L+1}}$$

Training

無限極限では，学習中のカーネルは定数とみなせる

➢ 無限極限ではカーネルの時間に依存しなくなるため
初期化時に計算したカーネルを利用できる

$$\Theta^{(L)}(t) \to \Theta_\infty^{(L)} \otimes Id_{n_L}.$$

$$\partial_t f_{\theta(t)} = \Phi_{\Theta_\infty^{(L)} \otimes Id_{n_L}}\left(\langle d_t, \cdot \rangle_{p^{in}}\right).$$

※ Φに関する本文中の記載

order to represent μ as $\langle d, \cdot \rangle_{p^{in}}$. Using the fact that the partial application of the kernel $K_{i,\cdot}(x, \cdot)$ is a function in $\mathcal{F}$, we can define a map $\Phi_K : \mathcal{F}^* \to \mathcal{F}$ mapping a dual element $\mu = \langle d, \cdot \rangle_{p^{in}}$ to the function $f_\mu = \Phi_K(\mu)$ with values:

$$f_{\mu,i}(x) = \mu K_{i,\cdot}(x, \cdot) = \langle d, K_{i,\cdot}(x, \cdot) \rangle_{p^{in}}.$$

※ Appendixより，下記式で定義される値Aがn_L→∞で0に収束するとのこと

$$A(t) = ||\alpha_i^{(L)}(0)||_{p^{in}} + c\left\|\tilde{\alpha}_i^{(L)}(t) - \tilde{\alpha}_i^{(L)}(0)\right\|_{p^{in}} + ||W_i^{(L)}(0)||_2 + \left\|W_i^{(L)}(t) - W_i^{(L)}(0)\right\|_2.$$

$$A(t) \le A(0) \exp\left(\frac{\max\{c^2 \|\Theta_\infty^{(L)}\|_{op}, 1\}}{\sqrt{n_L}} \int_0^t \|d_s\|_{p^{in}} ds\right).$$

$$\frac{\partial \boldsymbol{y}}{\partial t} = -\boldsymbol{K}(\boldsymbol{y} - \bar{\boldsymbol{y}})$$

※基本的には上式に対応していると
思われるが，対応関係を追いきれません
でした.

Least-squares regression

二乗損失を考えて，具体的に計算．訓練誤差は指数関数的に減少する

- 一般的な二乗損失

$$C(f) = \frac{1}{2}||f - f^*||^2_{p^{in}} = \frac{1}{2}\mathbb{E}_{x\sim p^{in}}\left[\|f(x) - f^*(x)\|^2\right].$$

- 学習による関数f_tの更新

$$\partial_t f_t = \Phi_K\left(\langle f^* - f, \cdot\rangle_{p^{in}}\right).$$

- 微分方程式として関数f_tを解く

$$f_t = f^* + \Delta_f^0 + \sum_{i=1}^{Nn_L} e^{-t\lambda_i}\Delta_f^i,$$

where Δ_f^0 is in the kernel (null-space) of Π and $\Delta_f^i \propto f^{(i)}$.

※ Δ_f^0 の意味合いが分からなかった・・・

⇨ λはカーネルの固有値であり，固有値の大きい次元から順に収束する
early-stoppingを支持する結果とのこと

$$\frac{\partial \boldsymbol{y}}{\partial t} = -\boldsymbol{K}(\boldsymbol{y} - \bar{\boldsymbol{y}})$$

$$\boldsymbol{d} = \boldsymbol{y} - \bar{\boldsymbol{y}}$$

$$\frac{\partial \boldsymbol{d}}{\partial t} = -\boldsymbol{K}\boldsymbol{d}$$

$$\boldsymbol{d}(t) = \boldsymbol{d}(0)e^{-\boldsymbol{K}t}$$

Numerical experiments

widthの増大/時間発展に伴い，収束することを確認

➤ カーネルの収束（左図）と出力関数の収束（右図）

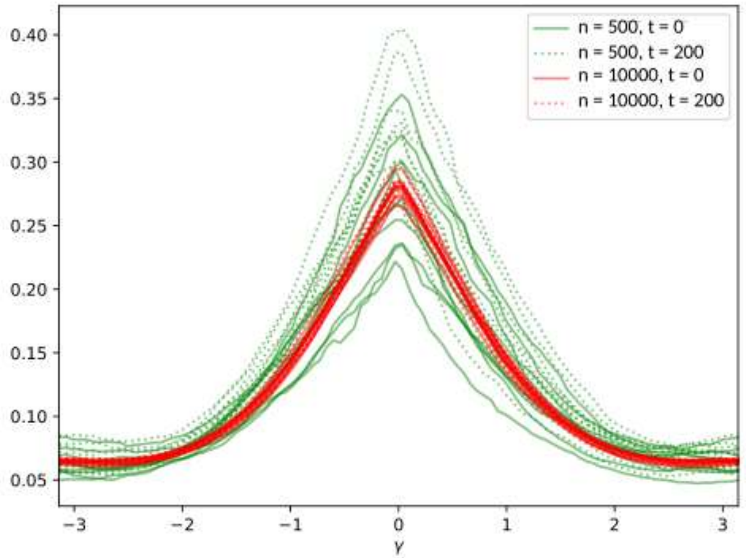


Figure 1: Convergence of the NTK to a fixed limit for two widths n and two times t.

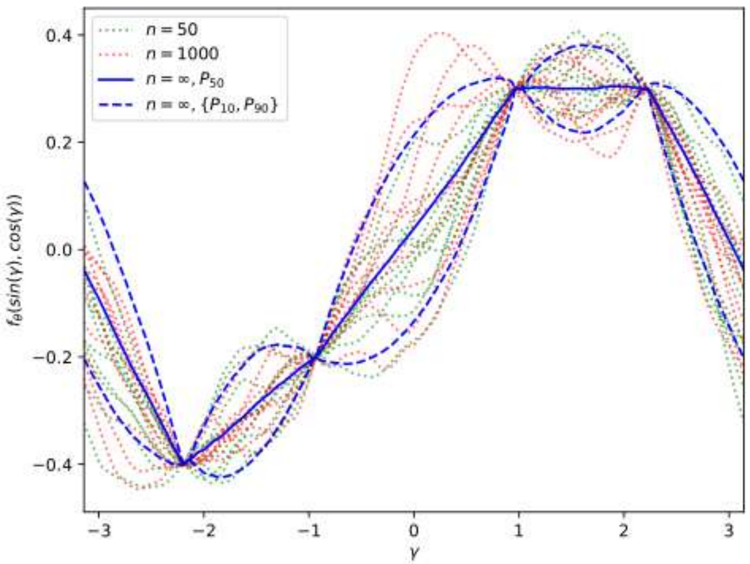


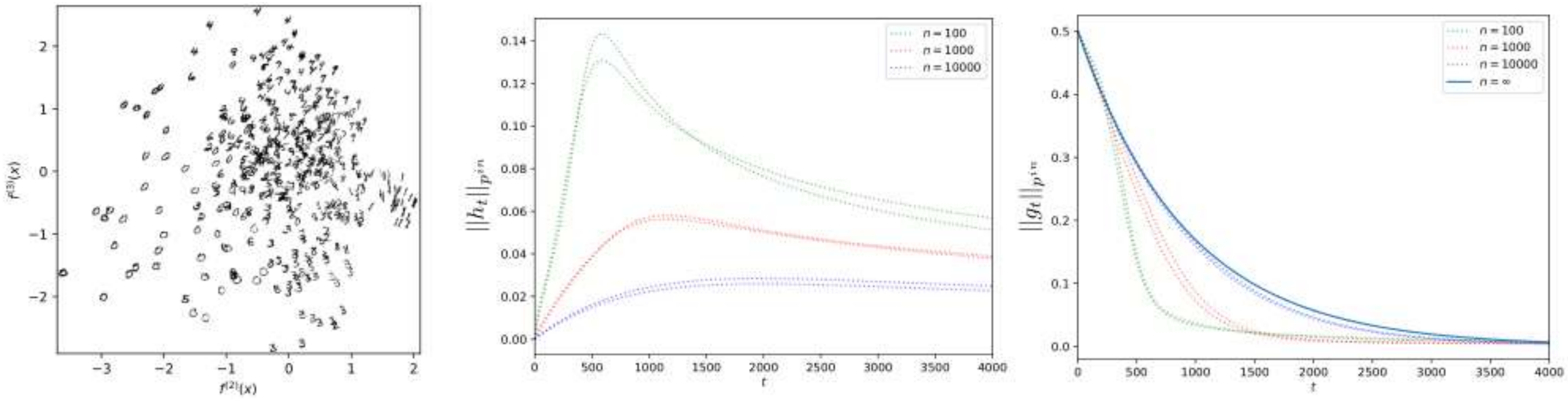
Figure 2: Networks function f_θ near convergence for two widths n and 10th, 50th and 90th percentiles of the asymptotic Gaussian distribution.

※学習データはunit circle(二次元)上の点．4層のNN

Numerical experiments

widthの増大に伴い，学習が安定化（勾配≒定数）することを確認

➢ 学習データの可視化（左図），収束点方向に垂直な方向の誤差（中央），収束点方向への誤差移（右図）



(a) The 2nd and 3rd principal components of MNIST.

(b) Deviation of the network function f_θ from the straight line.

(c) Convergence of f_θ along the 2nd principal component.

Figure 3

※正解を下記のように設定し，$f^{(2)}$方向と直交成分を観察

$$f^* = f_{\theta(0)} + 0.5 f^{(2)}$$

前スライドより

$$f_t = f^* + \Delta_f^0 + \sum_{i=1}^{Nn_L} e^{-t\lambda_i} \Delta_f^i,$$

➢ n=10000のときのカーネルを使ったPCA上位3成分への写像
➢ nが大きいほど，$f^{(2)}$方向の誤差が指数関数的に減少．
直交成分へのブレも最も少ない
➢ 一方，nが小さいほうが収束自体は早い．
学習係数とも相補的になっているため，考察は難しいが・・・

Conclusion

- Neural Tangent Kernel による学習過程の記述を行い，
  width→∞で，カーネルが定数となり，学習過程の解析を可能にした

- カーネルが定数になることは数値実験で確認できたが，
  widthが小さいほうが収束が早いという現象が見られた．

その後の研究

- On lazy training in differentiable programming[4]
  関数fを定数倍することで，width→∞と同様の性質が得られることを示した.
  CNNで学習すらうまくいかないケースも

- Enhanced Convolutional Neural Tangent Kernels[5]
  CIFAR-10でSOTAに対し-7%程度の性能（Alexnet相当）

※最新の識別性能を達成できてはいないが,
　それなりに高性能な予測器を定数カーネルの元で学習できたということらしい

[4] Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. In Advances in Neural Information Processing Systems (pp. 2937-2947).
[5] Li, Z., Wang, R., Yu, D., Du, S. S., Hu, W., Salakhutdinov, R., & Arora, S. (2019). Enhanced convolutional neural tangent kernels. arXiv preprint arXiv:1911.00809.

感想

- 現時点で，今後の研究に対する示唆（予測性能向上に向けた知見など）ができる段階までは到達できていないようだが，汎化性能の条件などについての整理が進み，性能向上に寄与することを期待したい

- データそのものに関する性質の理論解析もあればよいように思うが，
やはり難しいか・・・