STAT 535

11/28/23

Lecture 17

Wide Neural Networks
as GPs

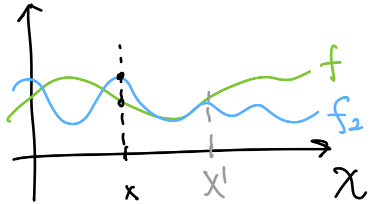
No Q3
No HW7

Gaussian Process → distribution over functions

$\mathcal{X}$ sample space (continuous)

$\{ f : \mathcal{X} \to \mathbb{R} \} = \mathcal{F}$ ← - - - distribution on $\mathcal{F}$ = process

$[f(x)]_{x \in \mathcal{X}}$



distribution of $f(x)$, $f \in \mathcal{F}$

$f(x) \sim N(\mu_x, \sigma_x^2)$ for all $x \in \mathcal{X}$

Gaussian Process G.P.

Gram matrix for SVM

$Cov(f(x), f(x')) = K(x, x') \in \mathbb{R}$

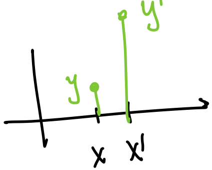
G $\equiv$

$\mathcal{D} = \{x^{1:n}\} \subset \mathcal{X}$

$[K(x^i, x^j)]_{i,j=1:n} = \Sigma \succcurlyeq 0$ for all $\mathcal{D}$

Required ⇕ $K(\,,\,)$ satisfies Mercer condition

$f(x) \perp\!\!\!\perp f(x')$
$\;\;\|\qquad\quad\|$
$\;\;y\qquad\quad y'$



$Cov(y, y') \lessapprox 1 \Rightarrow y \approx y' \Rightarrow f$ smooth

GP is prior

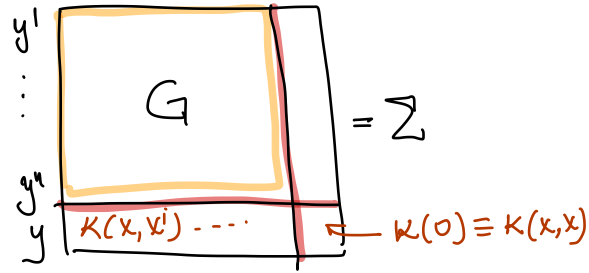
Non-linear
Non-parametric } regression (Bayesian regression)

$\mathcal{D} = \{(x^i, y^i),\ i = 1:n\}$

Prior $\quad GP(0, K)$ ← kernel

Posterior $\quad GP(\mu_{y|x,\mathcal{D}},\ G_{x|\mathcal{D}})$

$\mu_{y|x,\mathcal{D}}$ " $\mu_{x|\mathcal{D}}$ $\qquad G_{x|\mathcal{D}}$ "covariance"



$G = [K(x^i, x^j)]$

Prediction given $x \in \mathcal{X}$

wanted $\quad \Pr[f(x) \mid \mathcal{D}] = N(\mu_{x|\mathcal{D}}, \sigma^2_{x|\mathcal{D}})$

$f(x)$ " y

$=$ conditional distribution ←

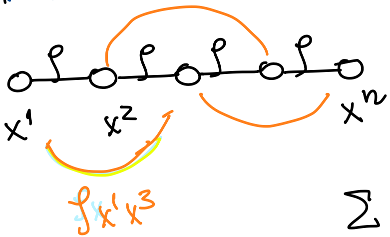
$$\begin{bmatrix} y \\ y^1 \\ \vdots \\ y^n \end{bmatrix} \sim N(0, \Sigma)$$

$n+1$ — joint distribution

$$\mu_{x|\mathcal{D}} \equiv E[y]_{\text{at } x} = [K(x, x^i) \dots]\, G^{-1} \begin{bmatrix} y^1 \\ \vdots \\ y^n \end{bmatrix}$$

$$\sigma^2_{x|\mathcal{D}} = K(x,x) - [K(x, x^i) \dots]\, G^{-1} \begin{bmatrix} K(x, x^i) \\ \vdots \end{bmatrix}$$

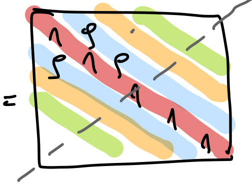
Ex1: Markov chain

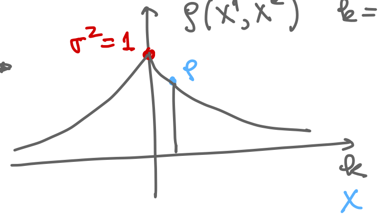


$$\rho = \rho(x^1, x^2) = \rho(x^2, x^3) = \dots$$

$$\sigma^2 = 1 = \mathrm{Var}(x^{1:n})$$

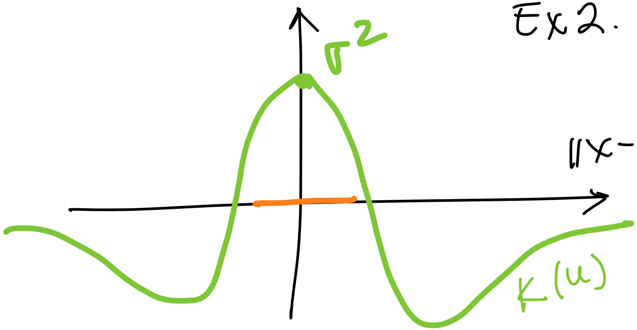
$$\Sigma = \mathrm{Cov}\left(\begin{bmatrix} x^1 \\ \vdots \\ x^n \end{bmatrix}\right) =$$





$n \to \infty$

Ex 2.

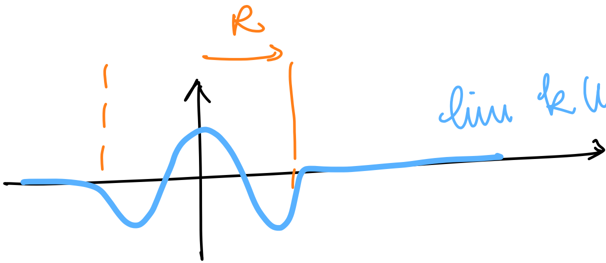


$$\|x - x'\| = u \qquad K(x, x') = K(\|x - x'\|)$$

(time)
(spatially)
invariant kernel

Rem: x, x'

$\|x - x'\| > R \Rightarrow f(x) \perp f(x')$



$\lim k(u) = 0, \; u \to \infty$

R: → smoothing param

supp $K = [-R, R]$

Lecture VI – Wide multilayer networks and the Neural Tangent Kernel (NTK)

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2023

What fctions do wide NN's represent?

(wide $\Leftrightarrow m_\ell \to \infty$ for $\ell = 1:L-1$)

$\mathcal{D} = \{(x^i, y^i)\}$ fixed

What fctions do wide NN's represent?

(wide $\Leftrightarrow m_\ell \to \infty$ for $\ell = 1:L-1$)

The Neural Tangent Kernel (NTK)

any predictor
training

Wide networks and Gaussian Processes

initialization
(before training)

The NTK is constant during training ← because θ doesn't change much

Example – regression and $\mathcal{L}_{\text{LS}}$

Wide and deep networks and classification

Notation

- Neural network predictor $f(x;\theta)$, where $x \in \mathbb{R}^d$
- For each layer $l = 1:L$ of dimension m_l, with $x^0 \equiv x$, and $z^L \equiv f(x)$

$$z^{l+1} = W^{l+1}x^l + b^{l+1} \qquad x^{l+1} = \phi(z^{l+1}) \tag{1}$$

  Here $x^{l,l+1}, z^{l+1}, b^{l+1}$ are column vectors W^{l+1} is a $m_{l+1} \times m_l$ matrix, $\phi()$ is the non-linearity/activation function.
- The weights

$$W^l_{ij} = \sigma_w w^l_{ij}/\sqrt{m_l}, \qquad b^l_j = \sigma_b \beta^l_j, \quad \text{Known as NTK parametrization} \tag{2}$$

- Parameter vector $\theta = \text{vector}\{w^{1:L}, \beta^{1:L}\} \in \mathbb{R}^p$ initialized i.i.d. $\sim N(0,1)$
- $\sigma_{w,b}$ are fixed hyper-parameters, $1/\sqrt{m_l}$ normalizes the expected norm of W^l columns
- Loss $\mathcal{L}(y,f)$

- We want to analize the behavior of this network $f()$ at initialization and during training, when $m_{1:L}$ very large
- Three approximations help analysis
  - (A1) continuous time training, called **gradient flow**
  - (A2) $m_{1:L} \to \infty$ in the wide limit, we can apply the Central Limit Theorem (CLT), and Gaussian Processes
  - (A3) parameters θ do not change much during training, i.e. $\theta_t - \theta_0$ is small

The Gradient Flow

- Assume training by gradient descent on $\hat{\mathcal{L}} = \sum_i \mathcal{L}(y^i, f(x^i))$
- The gradient of $\hat{\mathcal{L}}$

$$\nabla_\theta \hat{\mathcal{L}} = \sum_i \frac{\partial \mathcal{L}}{\partial f}(y^i, f(x^i;\theta))\nabla_\theta f(x^i,\theta) = \nabla_\theta f_{\mathcal{D}} \nabla_f \mathcal{L}_{\mathcal{D}} \quad \in \mathbb{R}^p \tag{3}$$

where $\nabla_f \mathcal{L}_{\mathcal{D}} = [\frac{\partial \mathcal{L}}{\partial f}(y^i, f(x^i;\theta))]_{i=1:n} \in \mathbb{R}^n$, $\nabla_\theta f_{\mathcal{D}} = [\nabla_\theta f(x^i,\theta)]_{i=1:n} \in \mathbb{R}^{p\times n}$

- Assume (A1) gradient descent with infinitezimal time steps. In other words, the parameters evolve by an ordinary differential equation

$$\dot{\theta} = -\eta \nabla_\theta f_{\mathcal{D}} \nabla_f \mathcal{L}_{\mathcal{D}} \quad \in \mathbb{R}^p \tag{4}$$

$$\dot{f} = \sum_{j=1}^{p} \frac{\partial f}{\partial \theta_j}\frac{\partial \theta_j}{\partial t} = (\nabla_\theta f)^T \dot{\theta} \quad \in \mathbb{R} \tag{5}$$

$$\dot{f}_{\mathcal{D}} = -\eta \underbrace{(\nabla_\theta f_{\mathcal{D}})^T \nabla_\theta f_{\mathcal{D}}}_{G} \nabla_f \mathcal{L}_{\mathcal{D}} \quad \in \mathbb{R}^p \tag{6}$$

- $G \equiv \nabla_\theta f_{\mathcal{D}}^T \nabla_\theta f_{\mathcal{D}} \equiv \kappa(\mathrm{X},\mathrm{X})$ is a Gram matrix!
- Therefore, we define the **Neural Tangent Kernel (NTK)** by

$$\kappa(x,x') = \nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta) \tag{7}$$

$$X = \begin{bmatrix} x^1 \\ \vdots \end{bmatrix}_{\mathbb{R}^n} \xrightarrow{\theta} f_\theta = \begin{bmatrix} f_\theta(x^i) \\ \\ \end{bmatrix} \longrightarrow \nabla_\theta f = \begin{bmatrix} \dfrac{\partial f_\theta(x^i)}{\partial \theta_j} \end{bmatrix}^T \in \mathbb{R}^{p \times n} \leadsto G = K(X, X)$$

$$\downarrow$$

$$y = \begin{bmatrix} y^1 \\ \vdots \\ \vdots \end{bmatrix}_{\mathbb{R}^n} \longrightarrow \ell = \begin{bmatrix} \ell(f(x^i), y^i) \\ \vdots \\ \end{bmatrix}_{\mathbb{R}^n}$$

Gradient flow and NTK – summary

$$\dot{\theta} = -\eta \nabla_\theta f_\mathcal{D} \nabla_f \mathcal{L}_\mathcal{D} \quad \in \mathbb{R}^p$$
$$\dot{f}_\mathcal{D} = -\eta G \nabla_f \mathcal{L} \quad \in \mathbb{R}^p$$
$$\kappa_\theta(x, x') = \nabla_\theta f(x)^T \nabla_\theta f(x')$$

- f_X, $\nabla_\theta f_X$, G depend only on the inputs X, θ
- $\nabla_f \mathcal{L}$ depends only on the correct outputs Y, and predicted outputs, i.e. on Y and θ

- This holds for any predictor! So what is special about neural networks?

$\theta \sim N(0, \cdot\cdot)$ iid NN initialization (i)

- First, we will analyze κ for very wide neural networks with random parameters (e.g. at initialization)
- Then, we will analyze what happens during training under assumption (A3) (ii)

Wide NN's Gaussian Process (GP) i

- This is about f_0, a NN initialized with Gaussian independent parameters. For simplicity, we denote it as f.

- Assume $\theta^{1:L-1}$ fixed, only W^L, b^L random as in (2)
- Recall $f(x) = W^L x^{L-1}(x) + b^L$ for any x with $x^{L-1} \in \mathbb{R}^{m_L}$

- $f(x) =$ sum of m_{L-1} i.i.d. random variables, hence $f(x) \sim Normal$ by CLT, for m_{L-1} large
- Randomness is over weights W^L, b^L!!!
- We have $E[f(x)] = 0$ and

$$Cov(f(x), f(x')) = E[(W^L x^{L-1} + b^L)(W^L (x')^{L-1} + b^L)] = \frac{\sigma_w^2}{m_{L-1}} (x^{L-1})^T (x')^{L-1} + \sigma_b^2 \equiv \kappa^L(x \quad (8)$$

where $x^{L-1}, (x')^{L-1} \in \mathbb{R}^{m_{L-1}}$ are the outputs of the $(L-1)$-th layer for inputs x, x'
- κ^L is a positive definite **kernel** Exercise Prove this.
- $f(x)$ is a random function of x
- The distribution of $f(x)$ defined as above, is called a **Gaussian Pocess**

- More generally, it can be shown [Jacot, Gabriel, Hongler, NeurIPS 2018] that, when all θ parameters are sampled as in (**??**), $f_0(x) \sim GP(0, \kappa^L)$

Q1 What is the kernel κ^L of this GP ii

Q2 This is all nice, but θ changes during training. What can we say about θ_t, f_t after training?

Q1: Idea.

- From (8), for layer $l = 1:L$ we have

$$\kappa^l(x,x') \;=\; E[z_j^l(x)z_j^l(x')] \;=\; \frac{\sigma_w^2}{m_{l-1}}(x^{l-1})^T(x')^{l-1} + \sigma_b^2 \tag{9}$$

with $x^{l-1} = \phi(z^{l-1})$. Note also that z_j^l are i.i.d. so it does not matter which j we choose.

- In particular, $\kappa^1(x,x') \;=\; \frac{\sigma_w^2}{m_1}x^Tx' + \sigma_b^2$ is deterministic
- ...and κ^l is random for $l > 1$.
- However, when $m_l \to \infty$, $\frac{1}{m_{l-1}}(x^{l-1})^T(x')^{l-1} \to E[*]$
- More specifically, this expectation can be written as

$$E[*] \;=\; \int\int \phi(z)\phi(z')Normal(\left[\begin{array}{c} z \\ z' \end{array}\right]; 0, \kappa_{x,x'}^{l-1})\, dz\, dz'. \tag{10}$$

In the above z, z' represent the $z^{l-1}(x), z^{l-1}(x')$ variables, sampled from the level l Normal distribution, which has covariance given by κ^{l-1}, namely

$$\kappa_{x,x'}^{l-1} \;=\; \left[\begin{array}{cc} \kappa^{l-1}(x,x) & \kappa^{l-1}(x,x') \\ \kappa^{l-1}(x',x) & \kappa^{l-1}(x',x') \end{array}\right]. \tag{11}$$

- Hence, the limit of $\kappa^l(x,x')$ when $m_{1:l} \to \infty$, is a **deterministic kernel** for all l. [Jacot, Gabriel, Hongler, NeurIPS 2018] derived this recursion (next page).

Q1: A recursive expression for the Neural Tangent Kernel

[Jacot, Gabriel, Hongler, NeurIPS 2018]

- L fixed, $m \to \infty$
- Simplified expression for $m_{0:L} = m$, $\sigma_w = \sigma_b = 1$
- Then the NTK $\kappa \equiv \kappa^L$ is defined recursively by layer

$$\kappa^1(x,x') = \Sigma^1(x,x'), \quad \Sigma^1(x,x') = \frac{1}{m}x^T x' + 1 \tag{12}$$

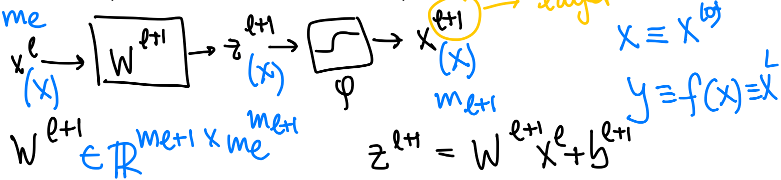
$$\kappa^{l+1}(x,x') = \kappa^l(x,x')\dot{\Sigma}^{l+1}(x,x') + \Sigma^{l+1}(x,x'), \tag{13}$$

with
$$\Sigma^{l+1}(x,x') = \mathrm{L}^{\phi}_{\Sigma^l(x,x')}, \tag{14}$$

$$\dot{\Sigma}^{l+1}(x,x') = \mathrm{L}^{\phi'}_{\Sigma^l(x,x')}, \tag{15}$$

and
$$\mathrm{L}^{\phi}_{\Sigma} = E[\phi(X)\phi(X')] \,\text{with}\, (X,X') \sim N\left(0, \begin{bmatrix} \Sigma(X,X) & \Sigma(X,X') \\ \Sigma(X,X') & \Sigma(X',X') \end{bmatrix}\right) \tag{16}$$

- In other words, at level $l+1$, $X \equiv x^l, X' \equiv (x')^l$ are sampled from a GP with kernel Σ^l, and $\Sigma^{l+1}(x,x')$, $\dot{\Sigma}^{l+1}(x,x')$ represent their (scalar) covariance after passing through the non-linearities ϕ, ϕ' (where ϕ' is the derivative of ϕ)



NN random initialization

$$W^{l+1}_{j,j'} \sim N\left(0, \frac{1}{m_l}\sigma_W^2\right) \quad \text{iid}$$

$$b^{l+1}_j \sim N(0, \sigma_b^2)$$

intuition

$l=0$

first layer

$$z^1 = W^0 x + b \sim N(0, \Sigma_1) \in \mathbb{R}^{m_1}$$

(x ↑ data point)

x, x' data points

1) $j, j' \in 1:m_1$

$$\mathrm{Cov}\left(z^1_j(x), z^1_{j'}(x')\right) = E\left[(W^1_{j:} x + b_j)(W_{j':} x' + b_{j'})\right] = 0 \quad \text{for any } x, x'$$

$b_j, b_{j'}, W_{jk}, W_{j'k'}$ mutually independent for all k, k', j, j'

2) $j = j' \in 1:m_1$

$$\mathrm{Cov}\left(z^1_j(x), z^1_j(x')\right) =$$

$$= E\left[(W^1_{j:} x + b_j)(W^1_{j:} x' + b_j)\right] = $$

$$= E\left[b_j^2 + b_j W^1_{j:}(x' + x) + x^T (W_{j:}^T W_{j:}) x'\right] = \sigma_b^2 + \frac{1}{m_1} x^T x'$$

($E[b_j^2] = \sigma_b^2$; $E[b_j W^1_{j:}(x'+x)] = 0$; $E[W_{j:}^T W_{j:}] = \frac{1}{m_1}\sigma_W^2 I_{m_0}$)

< for any $j = 1:m_1$

depends on x, x' data points

Summary so far

- Now, we understand the random intialization of wide networks, with L layers.

$$f_0 \sim GP(0, \kappa^L) \tag{17}$$

where κ^L is a kernel that depends only on ϕ (and $\sigma^2_{b,w}$)

What next?

- Analysis of training by linearization
- Then, the NTK limit for $L \to \infty$ and its relevance for classification and regression

The Linearized Network f^{lin}

Notation: $\theta_{0,t}, f_{0,t}$ = parameters, predictor at times 0, t

- Here we use (A3), the assumption that the parameters θ change little during training. Extensive evidence supports this assumption.

- First order Taylor expansion of f_t around f_0

$$f_t^{\text{lin}}(x) = f_0(x) + \nabla_\theta f_0(x)^T(\theta_t - \theta_0) \quad (18)$$

non-linear in x, linear in θ

$$\nabla_\theta f_t^{\text{lin}} = \nabla_\theta f_0 \quad (19)$$

$$\kappa(x, x') = \nabla_\theta f_0(x)^T \nabla_\theta f_0(x') = NTK(\theta_0) \quad \textbf{constant during training} \quad (20)$$

$$G_0 \equiv \kappa_{X,X} \quad \text{Gram matrix at } \theta_0 \quad (21)$$

from (4) ⟶

$$\dot{\theta}_t = -\eta \nabla_\theta f_0(X)^T \nabla_f \mathcal{L}(Y, f_t^{\text{lin}}(x)) \quad (22)$$

from (5) ⟶

$$\dot{f}_t^{\text{lin}}(x) = -\eta \underbrace{\kappa(x, X)^T [redacted]}_{\text{depends on } \theta_0} \nabla_f \mathcal{L}(Y, f_t^{\text{lin}}(x)) \quad (23)$$

from (6) ⟶ $\dot{f}_t^{lin}(X) = -\eta \, G_0 \nabla_f \mathcal{L}(Y, f_t^{lin}(X))$

NTK during training – empirical evidence

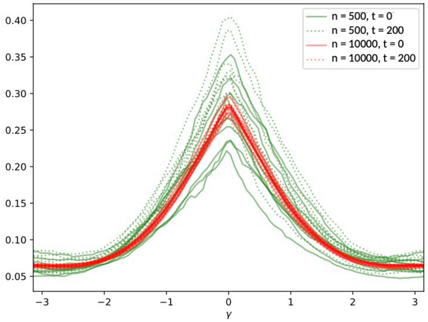


Figure 1: Convergence of the NTK to a fixed limit for two widths n and two times t.

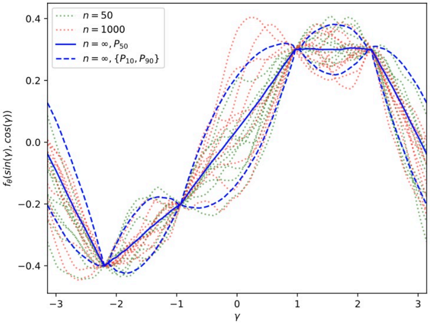


Figure 2: Networks function f_θ near convergence for two widths n and 10th, 50th and 90th percentiles of the asymptotic Gaussian distribution.

Linearized Network dynamics for $\mathcal{L}_{\rm LS}$

- For example, for $\mathcal{L}_{\rm LS}(y, f) = \frac{1}{2}(f - y)^2$, $\nabla_f \mathcal{L}_{\rm LS}(f, y) = f - y$. In this case, equations (22),(23) are a linear system and have an analytic solution.

$$\theta_t - \theta_0 = -\nabla_\theta f_0(\mathrm{X})^T G_0^{-1} \left(I - e^{-\eta G_0 t}\right)(f_0(\mathrm{X}) - \mathrm{Y}) \quad (24)$$

$$f_t^{\rm lin}(\mathrm{X}) = \left(I - e^{-\eta G_0 t}\right)\mathrm{Y} + e^{-\eta G_0 t} f_0(\mathrm{X}) \quad (25)$$

$$f_t^{\rm lin}(x) = \underbrace{\kappa(x, \mathrm{X})^T G_0^{-1}\left(I - e^{-\eta G_0 t}\right)\mathrm{Y}}_{\mu(x)} + \underbrace{f_0(x) - \kappa(x, \mathrm{X})^T G_0^{-1}\left(I - e^{-\eta G_0 t}\right) f_0(\mathrm{X})}_{\gamma(x)} \quad (26)$$

Notes:

- if $G_0 \succ 0$ then $e^{-\eta G_0 t} \to 0$ for $t \to \infty$
- in discrete time $t = 0, 1, 3, \ldots$ replace e^{at} with $(1 - a)^t$.
  Sketch of proof: $\ln(1 - a)^t = t \ln(1 - a) \approx t(-a)$ for a small; therefore $e^{-at} \approx (1 - a)^t$.
- $f_t^{\rm lin}(x) = f_0(x) + \kappa(x, \mathrm{X})^T G_0^{-1}\left(I - e^{-\eta G_0 t}\right)(\mathrm{Y} - f_0(\mathrm{X}))$

Exercise Prove (24),(25),(26) from (22),(23)