Neural Tangent Kernel(NTK)

Huihui Xu

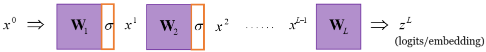
AI Thrust
Information Hub
HKUST (GZ)

2022

Four components

- Dataset: $D = \{(x_i, y_i)\}_N$, **empirical distribution** $p^{in} = \frac{1}{N} \sum_{x \in \mathcal{X}} \delta_x$
  - **input**: $x \in R^d$, **dataset inputs**: $\mathcal{X} = \{x | (x, y) \in D\} \subset R^d$
  - **supervision**: $y \in R^c$, **dataset supervision**: $\mathcal{Y} = \{y | (x, y) \in D\} \subset R^c$
  - For example, image classification dataset: $x \in [0, 1]^{CHW}, y \in \{0, 1\}^c$;
- Model: $f : R^{n_0} \times R^p \to R^{n_L}$, $z^L = f(x^0; \theta)$
  - W.l.o.g, we assume $n^0 = d, n^L = c$

A Feed-forward Network of L layers



- Train: $$\boxed{\min \mathcal{L}(\theta) = \frac{1}{N} \sum_{(x,y) \in D} \ell(f(x; \theta); y) = L(f(\mathcal{X}; \theta); \mathcal{Y})}$$
- Evaluation
  - model architecture, loss function, optimization(algorithm + hyperparameters), train/test asymmetry design
  - data distribution characteristics

Introduction

- Optimization of ANNs are carried out in the parameter space using SGD and its variants;
- The loss function is **nonconvex in the parameter space**;
- But the loss functional, such as the cross-entropy and regression, is generally **convex in the function space**;
- Neural Tangent Kernel [1]: a **tool** to analyse the convergence and generalization of ANNs.

Formulated in Function space

$$\underbrace{\min_{\theta \in R^p} \mathscr{L}(\theta)}_{\text{optimization in parameter space, [non-convex]}} \overset{f_\theta = f(\cdot,\theta)}{\Longleftrightarrow} \underbrace{\min_{f_\theta \in \mathscr{F}} C(f_\theta)}_{\text{optimization in function space, [convex]}}$$

Examples: $g \in \mathscr{F}$ as the **"perfect" mapping**: $\forall (x,y) \in D, g(x) = y$
a semi-inner product in the function space:

$$< f, g >_{p^{in}} = \frac{1}{N} \sum_{(x,y)\in D} f(x)^T g(x)$$

- Regression with L2-norm:

$$\mathscr{L}(\theta) = \frac{1}{N} \sum_{(x,y)\in D} ||f(x;\theta) - y||_2^2 = ||f_\theta - g||_{p^{in}}^2 = C(f_\theta)$$

- Classification (single label) with Cross-Entropy:

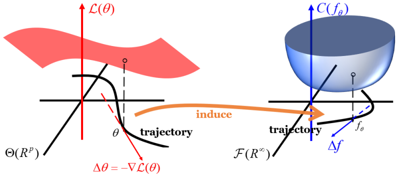
$$\mathscr{L}(\theta) = \frac{1}{N} \sum_{(x,y)\in D} < -log\ softmax f(x;\theta), y >$$

$$= < -log\ softmax f_\theta, y >_{p^{in}} = C(f_\theta)$$

$\Delta\theta \to \Delta f$

- Steepest Gradient Descent in Θ:

$$\Delta\theta = -\nabla_\theta \mathscr{L}(\theta)$$
$$= -\frac{1}{N}\sum_{(x,y)\in D} \nabla_\theta f(x;\theta)\ \nabla_1 \ell(f(x;\theta);y)$$



steepest gradient　　kernel gradient

- Move θ will cause f_θ to move.
  The derivative of the output w.r.t every x induced by $\Delta\theta$ is:

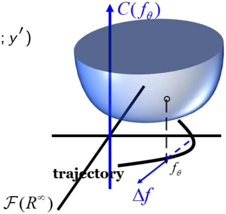
$$\forall x \in R^d : \Delta f(x) = \nabla_\theta f(x;\theta)^T \Delta\theta$$
$$= -\frac{1}{N}\sum_{(x',y')\in D} \underbrace{\nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta)}_{\textbf{NTK}} \nabla_1 \ell(f(x';\theta);y')$$

  - is a **direction in $\mathscr{F}$**
    called Kernel Gradient

$\Delta f \to \Delta C$

- The derivative of C:

$$
\begin{aligned}
\Delta C &=< \nabla C(f_\theta), \Delta f > \\
&= -\frac{1}{N^2} \sum_{(x,y)\in D} \nabla_1 \ell(f(x;\theta);y)^T \sum_{(x',y')\in D} \nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta) \nabla_1 \ell(f(x';\theta);y') \\
&= -\frac{1}{N^2} \sum_{(x,y)\in D} \sum_{(x',y')\in D} \nabla_1 \ell(f(x;\theta);y)^T \underbrace{\nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta)}_{NTK} \nabla_1 \ell(f(x';\theta);y') \\
&= -\frac{1}{N^2} \sum_{(x,y)\in D} \sum_{(x',y')\in D} \nabla_1 \ell(f(x;\theta);y)^T K(x,x';\theta) \nabla_1 \ell(f(x';\theta);y') \\
&= -E_{x,x' \sim p^{in}} \nabla_1 \ell(f(x;\theta);y)^T K(x,x';\theta) \nabla_1 \ell(f(x';\theta);y')
\end{aligned}
$$



kernel gradient

Say ΔC and K using linear algebra language

- Preliminaries
  - In R^n, each linear mapping from R^n to R^n corresponds to an $n * n$ matrix.
  - The set of all linear mappings, i.e. $n * n$ matrices, is denoted as $\mathbf{L}in^n$
  - Each element in $\mathbf{L}in^n$ is called a **second-order tensor**.
  - The **tensor product of two vectors** is a second-order tensor of rank 1, i.e. an $n * n$ matrix, in $\mathbf{L}in^n$:

$$m \otimes n = mn^T \in \mathbf{L}in^n, \ (m \otimes n)p = m(n^T p)$$

Say ΔC and K using linear algebra language

$$\Delta C = -E_{x,x' \sim p^{in}} \nabla_1 \ell(f(x;\theta);y)^T K(x,x';\theta) \nabla_1 \ell(f(x';\theta);y')$$

- Function space (R^∞):
  - We can view $\nabla_{\theta_i} f(\cdot;\theta)$ as a vector in R^∞.
  - Then, $K(\cdot,\cdot;\theta)$ is the summation of tensor products $\sum_{i=1}^p \nabla_{\theta_i} f(\cdot;\theta) \otimes \nabla_{\theta_i} f(\cdot;\theta)$, which is symmetric.
  - Therefore, $K(\cdot,\cdot;\theta)$ can be viewed as a **symmetric matrix** in the function space.
  - Then, ΔC can be viewed as a **quadratic form**:

$$-\underbrace{\nabla_1 \ell(f(\cdot;\theta);y.)^T}_{\text{a vector}} \underbrace{K(\cdot,\cdot;\theta)}_{\text{a symmetric matrix}} \underbrace{\nabla_1 \ell(f(\cdot;\theta);y.)}_{\text{the same vector}}$$

Some Insights about Convergence

$$\underbrace{\min_{\theta \in R^p} \mathscr{L}(\theta)}_{\text{optimization in parameter space, } \textbf{[non-convex]}} \overset{f_\theta = f(\cdot,\theta)}{\Longleftrightarrow} \underbrace{\min_{f_\theta \in \mathscr{F}} C(f_\theta)}_{\text{optimization in function space, } \textbf{[convex]}}$$

- Doing Steepest GD in Θ will cause C to change like a quadratic form in the function space:

$$\Delta C = -\nabla_1 \ell(f(\cdot;\theta); y.)^T \overbrace{K(\cdot,\cdot;\theta)}^{NTK} \nabla_1 \ell(f(\cdot;\theta); y.)$$

  - NTK is changing **changing during training**;
  - $\Delta C < 0 \Longleftrightarrow \Delta f$ is a decreasing direction
  - If we know $\boxed{\forall \theta \in R^p, \text{ NTK is positive definite}}$:
    - **Every update is then a decreasing direction**
    - C is convex $\Longrightarrow$ **converge to global optimal**
  - If we know $\boxed{\text{for the current } \theta, \text{NTK is positive definite}}$, then the next update ΔC is decreasing

infinite-width feed-forward networks

Assumptions:

- The nonlinearity activation σ is Lipschitz, with bounded second derivative.
- Parameters are i.i.d initialized: $W_{ij}^{(l)} \sim N(0, \frac{1}{n_l})$ and $b_j^{(l)} \sim N(0, 1)$

Theorem: In the limit as the layers width $n_1, ..., n_{L-1} \to \infty$, the network 's NTK **converges to be**:

- **deterministic at initialization**, meaning that $K(\cdot, \cdot; \theta)$ is **irrelevant to the initialized** θ and only determined by the model architecture;
- **stay constant during training**.
- **positive definite, for a large family of** p^{in} and σ

infinite-width feed-forward networks

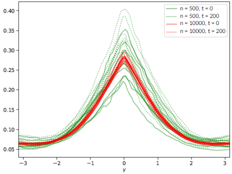


Figure 1: Convergence of the NTK to a fixed limit for two widths n and two times t.

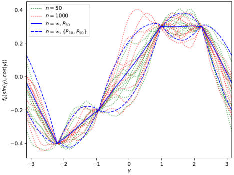


Figure 2: Networks function f_θ near convergence for two widths n and 10th, 50th and 90th percentiles of the asymptotic Gaussian distribution.

- Convergence of NTK
  - At initialization, the wider network's NTK shows less variance.
  - During training, the narrow network's NTK vary more.
- Kernel Regression
  - The NTK can gives good indication of the distribution of the converged function, even for small-width nets, e.g. 50.

Tips

- Linear:

$$K(\cdot,\cdot;\theta) = \sum_{i=1}^{p} \nabla_{\theta_i} f(\cdot;\theta) \otimes \nabla_{\theta_i} f(\cdot;\theta)$$

- To analyse Generalization error:

$$\begin{aligned}\forall x \in R^d : \Delta f(x) &= \nabla_\theta f(x;\theta)^T \Delta\theta \\ &= -\frac{1}{N} \sum_{(x',y')\in D} \underbrace{\nabla_\theta f(x;\theta)^T \nabla_\theta f(x';\theta)}_{\textbf{NTK}} \nabla_1 \ell(f(x';\theta);y')\end{aligned}$$

- To analyse Optimization error:
  stochastic gradient descent, continual learning, transfer learning.

References

A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.