گزارش پروژه

دانشکده مهندسی کامپیوتر

نام و نام خانوادگی دانشجو: سید احسان حسن بیگی

استاد: دکتر نجفی

پاییز ۱۴۰۳

فهرست مطالب

۱. مقدمه

در این گزارش به بررسی مقاله‌ی

Neural Tangent Kernel: Convergence and Generalization in Neural Networks [1]

می‌پردازیم.

استفاده از کرنل، یکی از رویکرد های مؤثری بود که قبل از اوج گرفتن شبکه های عصبی مورد استفاده قرار می‌گرفت و در کاربرد هایی از قبیل classification و regression قابل استفاده می‌باشد. در این روش ابتدا فضای ویژگی ها توسط یک تبدیل غیر خطی به یک فضای Hilbert، معمولا با بُعد بالاتر، انتقال پیدا می‌کند و سپس در آن فضا توسط یک مدل خطی به حل مسئله می‌پردازیم. مزیت استفاده از کرنل آن است که ممکن است داده های ورودی به صورت خام در فضای ویژگی ها جدایی پذیر نباشند اما در فضای با بُعد بالاتر به راحتی تفکیک شوند. همچنین بنا به استفاده از kernel trick، روش هایی از قبیل kernel SVM، kernel regression نیازی به ملاقات مستقیم فضای Hilbert ندارند و تنها ارائه‌ی یک کرنل معتبر (PDS kernel) برای انجام محاسبات کافی است. با این حال ایراد اصلی این روش که باعث می‌شود در عمل، نسبت به مدل های بر پایه‌ی شبکه عصبی ضعیف تر عمل کند آن است که کرنل مورد استفاده، باید به صورت ثابت و از پیش تعیین شده بر مسئله اعمال شود.

یک تعبیر شهودی از روند کار شبکه های عصبی به این صورت است که لایه های میانی شبکه، معادل تبدیلی غیر خطی اند که ورودی را به فضای Hilbert می‌برند و سپس لایه‌ی آخر حکم تفکیک خطی در فصای Hilbert را دارد. به این صورت مشاهده می‌شود که گویا شبکه های عصبی نیز دیدگاه کرنل را اعمال می‌کنند اما بر خلاف روش های قبل، انتخاب کرنل به صورت adaptive و بر اساس داده های ورودی، معماری شبکه و روند آموزش صورت می‌گیرد. پیش از این مقاله، تعبیر هایی مشابه آن چه ذکر شد برای ارتباط میان شبکه های عصبی و روش های بر پایه‌ی کرنل وجود داشت اما با توجه به تغییرات شدید شبکه های عصبی در طول فرآیند آموزش، تئوری دقیقی برای پیدا کردن کرنل متناظر با هر شبکه عصبی ارائه نشده بود.

این مقاله از محدود کار هایی است که توانسته تئوری مناسبی برای شبکه های عصبی ارائه دهد. به طور دقیق تر، برای شبکه‌های fully connected نشان می‌دهد که اگر عرض شبکه به بی‌نهایت میل کند و مقدار دهی اولیه‌ی وزن ها بر اساس توزیع گاوسی باشد، آنگاه شبکه معادل یک فرآیند تصادفی گاوسی می‌باشد. حال بر اساس کرنل (کوواریانس) این فرآیند تصادفی کرنلی به نام neural tangent kernel (NTK) ارائه می‌کنند که متناظر شبکه عصبی است و می‌توان توسط آن، رفتار شبکه را تحلیل کرد. ابتدا نشان داده می‌شود که این کرنل در مقدار دهی اولیه به یک مقدار deterministic میل می‌کند و سپس نشان داده می‌شود که حتی در فرآیند آموزش نیز این مقدار ثابت می‌ماند. در حقیقت اعمال gradient descent بر روی پارامتر های شبکه معادل اعمال kernel gradient descent نسبت به این کرنل می‌باشد و همگرا شدن شبکه بستگی به مثبت معین بودن ماتریس این کرنل دارد. در نهایت نشان داده می‌شود که استفاده از شبکه‌ی عصبی طبق قید های ذکر شده معادل حل کردن یک مسئله‌ی ridge-less kernel regression توسط NTK می‌باشد.

۲. روش ارائه شده

۲.۱. الگوریتم Gradient Flow

از پیش با الگوریتم gradient descent آشنایی داریم و می‌دانیم که روند به روز رسانی پارامتر های شبکه در قالب difference equation زیر صورت می‌گیرد:

$$\theta^{(0)} \leftarrow \text{init}$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta\, \nabla_\theta L\big(f(\theta)\big)\big|_{\theta=\theta^{(t)}}$$

$$\frac{\theta^{(t+1)} - \theta^{(t)}}{\eta} = -\nabla_\theta L\big(f(\theta)\big)\big|_{\theta=\theta^{(t)}}$$

حال اگر اندازه‌ی گام را به صفر میل دهیم خواهیم داشت:

$$\frac{d\theta^{(t)}}{dt} = -\nabla_\theta L\big(f(\theta)\big)\big|_{\theta=\theta^{(t)}}$$

که به آن gradient flow می‌گوییم. به عبارت دیگر، gradient flow معادل gradient descent است که در زمان پیوسته انجام شود و در عمل برای استفاده از این روند، همان gradient descent را با اندازه‌ی گام کوچک اعمال می‌کنیم.

۲.۲. تعریف صورت مسئله

توزیع ورودی را p^{in} می‌نامیم و داریم:

$$p^{in} = \frac{1}{N}\sum_{i=0}^{N} \delta_{x_i}$$

همچنین ضرب داخل و شبه نورم p^{in} را بر اساس

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

تعریف می‌کنیم.

تمام صحبت های این مقاله بر روی شبکه های fully connected می‌باشد. عمق شبکه را L فرض کرده و تعداد نورون های هر لایه را n_l می‌نامیم. تابع شبکه را به صورت

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

تعریف می‌کنیم و داریم:

$$\alpha^{(0)}(x;\theta) \coloneqq x$$

$$\tilde{\alpha}^{(l+1)}(x;\theta) \coloneqq \frac{1}{\sqrt{n_L}} W^{(l)} \alpha^{(l)}(x;\theta) + \beta b^{(l)}$$

$$\alpha^{(l)}(x;\theta) \coloneqq \sigma\left(\tilde{\alpha}^{(l)}(x;\theta)\right)$$

بنابراین تعداد پارامتر های شبکه را نیز به صورت زیر خواهیم داشت:

$$P = \sum_{l=0}^{L-1} (n_l + 1) n_{l+1}$$

ضریب $\frac{1}{\sqrt{n_L}}$ و متغیر β برای تعیین میزان اثر بایاس اضافه شده است تا اثر بایاس توسط وزن ها مغلوب/قالب نشود. همچنین تابع فعال سازی σ به صورت element-wise اعمال می‌شود. به صورت رسمی می‌توان گفت که تابع f_θ عضو فضای توابع $\mathcal{F}$ است و نسبت به یک functional cost مانند $C: \mathcal{F} \to \mathbb{R}$ بهینه سازی می‌شود.

۲.۳. مفهوم Kernel Gradient Descent

مفهومی تحت عنوان kernel gradient descent وجود دارد که به جای بهینه سازی تابع $f(x;\theta)$ در فضای پارامتر ها، به طور مستقیم f را نسبت به تابع هزینه، در فضای توابع بهینه می‌کند. می‌دانیم که تابع هزینه نسبت به پارامتر های شبکه به شدت non-convex است و بنابراین در صورت استفاده از gradient descent، تحلیل روند آموزش برای شبکه های عصبی بسیار سخت می‌شود. مزیت رویکرد kernel gradient آن است که می‌توان فرآیند آموزش را به طور مستقیم در فضای توابع بررسی کرد که تابع هزینه نسبت به آن convex می‌باشد.

در setup ذکر شده C تنها به مقادیر $f \in \mathcal{F}$ در نقاط مربوط به نمونه های ورودی وابسته است و بنابراین مشتق C در نقطه‌ی $f_0 \in \mathcal{F}$ عضوی از فضای dual خانواده‌ی $\mathcal{F}$ یعنی $\mathcal{F}^*$ می‌باشد و آن را به صورت $\partial_f^{in} C|_{f_0}$ نمایش می‌دهیم و به ازای یک عضو dual مانند $d|_{f_0}$ خواهیم داشت:

$$\partial_f^{in} C|_{f_0} = \langle d|_{f_0}, \cdot \rangle_{p^{in}}$$

حال kernel gradient برای کرنل K به صورت زیر تعریف می‌شود:

$$\nabla_K C|_{f_0} \coloneqq \Phi_K\left(\partial_f^{in} C|_{f_0}\right)$$

که Φ_K تبدیل متناظر کرنل K است.

تابعی مانند $f(t)$ مسیر kernel gradient descent را دنبال می‌کند اگر معادله‌ی دیفرانسیل زیر برقرار باشد:

$$\partial_t f(t) = -\nabla_K C|_{f(t)}$$

و تابع هزینه‌ی C نیز مطابق:

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

تغییر می‌کند. بنابراین اگر کرنل K نسبت به نورم p^{in} مثبت معین باشد، همگرا شدن به critical point تابع C گارانتی می‌شود و اگر C محدب بوده و از پایین محدود باشد، به ازای $t \to \infty$ تابع $f(t)$ به بهینه سراسری همگرا می‌شود. حال می‌توان نشان داد هنگامی که عرض لایه‌های شبکه به بی‌نهایت میل کند، الگوریتم gradient descent بر روی پارامتر های شبکه، به الگوریتم kernel gradient descent همگرا می‌شود. در ادامه به معرفی این کرنل می‌پردازیم.

۲.۴. تعریف Neural Tangent Kernel

مطابق آن چه گفته شد، این مقاله بر اساس پیش فرض های زیر صورت مسئله را تعریف می‌کند:

- شبکه‌ی عصبی $f(x;\theta)$ یک شبکه‌ی fully connected با عمق L است
- وزن های شبکه را توسط توزیع گاوسی initialize می‌کنیم
- برای به روز رسانی وزن ها از gradient flow استفاده می‌کنیم
- عرض شبکه را به بی‌نهایت میل می‌دهیم

همچنین در این گزارش در رابطه با حالتی صحبت می‌کنیم که $n_L = 1$ باشد. برای حالت هایی که خروجی شبکه بیش از یک بُعد داشته باشد می‌توان لایه های مخفی شبکه را به همراه هر نورون خروجی، معادل یک شبکه مستقل در نظر گرفت. تابع هزینه را نیز تابع مربعات خطا در نظر می‌گیریم. مطابق gradient flow خواهیم داشت:

$$\frac{d\theta^{(t)}}{dt} = -\nabla_\theta C\left(f\left(\cdot\,;\theta^{(t)}\right)\right)|_{\theta=\theta^{(t)}}$$

$$= -\nabla_\theta\left[\frac{1}{2}\sum_{j=1}^{N}\left(f\left(x_j;\theta^{(t)}\right)-y_j\right)^2\right]|_{\theta=\theta^{(t)}}$$

$$= -\sum_{j=1}^{N}\underbrace{\left(f\left(x_j;\theta^{(t)}\right)-y_j\right)}_{\Delta_j^{(t)}}\nabla_\theta f\left(x_j;\theta^{(t)}\right) \qquad (*)$$

در حقیقت $\Delta_j^{(t)}$ فاصله‌ی بین خروجی شبکه و جواب درست می‌باشد و معادل جهتی است که شبکه باید در گام t طی کند. طبق قاعده‌ی زنجیره‌ای خواهیم داشت:

$$\frac{\partial\Delta_j^{(t)}}{\partial t} = \frac{\partial\left(f\left(x_j;\ \theta^{(t)}\right)-y_j\right)}{\partial\theta_1}\frac{\partial\theta_1}{\partial t}+\cdots+\frac{\partial\left(f\left(x_j;\ \theta^{(t)}\right)-y_j\right)}{\partial\theta_P}\frac{\partial\theta_P}{\partial t}$$

که فرم برداری آن به صورت زیر است:

$$= \nabla_\theta\left[f\left(x_j;\theta^{(t)}\right)-y_j\right]^T\frac{\partial\theta^{(t)}}{\partial t}$$

با جاگذاری معادله‌ی (*) در عبارت فوق خواهیم داشت:

$$\frac{\partial \Delta_i^{(t)}}{\partial t} = -\sum_{j=1}^{N} \Delta_j^{(t)} \nabla_\theta f\big(x_j; \theta^{(t)}\big)^T \nabla_\theta f\big(x_i; \theta^{(t)}\big)$$

که فرم ماتریسی این معادله‌ی دیفرانسیل به صورت زیر است:

$$\frac{\partial}{\partial t}\Delta^{(t)} = -K\,\Delta^{(t)}$$

$$K_{ij} = \big\langle \nabla_\theta f\big(x_i; \theta^{(t)}\big), \nabla_\theta f\big(x_j; \theta^{(t)}\big) \big\rangle$$

به کرنل به دست آمده neural tangent kernel (NTK) می‌گوییم. در واقع دستاورد بزرگ این کار آن است که این کرنل، تنها به معماری شبکه (f) و دیتاست آموزش (x_i) بستگی دارد و به θ وابسته نیست. حتی اثبات می‌شود که NTK در طول فرایند آموزش نیز ثابت می‌ماند. پاسخ معادله‌ی دیفرانسیل فوق به صورت $\Delta(0)$ $\Delta^{(t)}= e^{-tK^{(t)}}$ خواهد بود و با فیکس کردن $t \in [0, T]$ ماتریس $K^{(t)}$ ثابت خواهد ماند.

بنابراین تعریف NTK به صورت زیر است:

$$\Theta^{(L)}(x, x'|\theta) \coloneqq \langle \nabla_\theta f(x;\theta), \nabla_\theta f(x';\theta) \rangle$$

در ادامه به شرح مقدار NTK در فاز مقدار دهی اولیه و همچنین تغییرات آن در حین فرآیند آموزش می‌پردازیم.

۲.۴.۱. NTK در مقدار دهی اولیه

بر اساس پیش فرض های صورت مسئله که پیش تر ذکر شد، مقدار دهی اولیه به وزن ها بر اساس توزیع گاوسی صورت می‌گیرد. حال ثابت می‌شود که اگر عرض لایه ها به بی‌نهایت میل کند، pre-activation های تمامی نورون های تمامی لایه ها به صورت فرآیند تصادفی گاوسی در فضای ورودی $\mathbb{R}^{n_0}$ خواهند بود. اثبات کامل این قضیه در appendix مقاله آورده شده است اما صرفا برای گاوسی بودن pre-activation نورون ها می‌توان گفت حاصل pre-activation هر نورون به صورت جمع حاصل ضرب post-activation نورون های لایه‌ی قبل در وزن های متناظر آن ها می‌باشد. با توجه به این که مقدار دهی اولیه وزن ها از توزیع گاوسی بوده است این مقدار معادل ترکیب خطی از تعدادی سمپل گاوسی (وزن ها) می‌باشد که می‌دانیم توزیع آن گاوسی خواهد بود.

حال می‌دانیم که فرآیند تصادفی گاوسی را می‌توان صرفا با میانگین و کرنل (ماتریس کوواریانس) آن نمایش داد. به طور دقیق تر طبق proposition 1 مقاله، هر کدام از نورون های لایه‌ی آخر شبکه یک centered gaussian process با میانگین صفر و کوواریانس $\Sigma^{(L)}$ می‌باشد که $\Sigma^{(L)}$ به صورت بازگشتی و طبق فرمول زیر به دست می‌آید:

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$

$$\Sigma^{(L+1)}(x,x') = \mathbb{E}_{f\sim N(0,\Sigma^{(L)})}\left[\sigma\big(f(x)\big)\,\sigma\big(f(x')\big)\right] + \beta^2$$

حال طبق قضیه‌ی ۱ مقاله، در صورت برقرار بودن فرض های ذکر شده NTK به یک مقدار deterministic همگرا می‌شود که آن را $\Theta_\infty^{(L)}: \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ می‌نامیم و مقدار آن نیز به صورت بازگشتی و طبق فرمول زیر به دست می‌آید:

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

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

$$\dot{\Sigma}^{(L+1)}(x,x') \coloneqq \mathbb{E}_{f\sim N(0,\Sigma^{(L)})}\left[\sigma'\big(f(x)\big)\,\sigma'\big(f(x')\big)\right]$$

که σ'، مشتق σ می‌باشد.

بنابراین همان طور که پیش تر نیز ذکر شده بود، مقداری که NTK به آن میل می‌کند، یعنی $\Theta_\infty^{(L)}$، تنها به انتخاب σ، عمق شبکه (L) و واریانس پارامتر ها در مقدار دهی اولیه (که برابر یک می‌باشد) بستگی دارد و از روی کرنل فرآیند تصادفی گاوسی قابل محاسبه است.

۲.۴.۲. NTK در فرآیند آموزش

نتیجه‌ی دوم مقاله آن است که در صورت برقرار بودن پیش فرض های صورت مسئله، علاوه بر موارد ذکر شده در زمان مقدار دهی اولیه، در حین آموزش نیز NTK ثابت می‌ماند. به طور دقیق تر طبق قضیه‌ی ۲ مقاله، تابع فعال سازی σ باید Lipschitz و twice differentiable باشد و مشتق دوم آن نیز محدود باشد. همچنین به ازای هر تعداد گام T ثابت، اگر جهت تغییرات در gradient descent را d_t بنامیم، باید در حالی که عرض لایه ها به بی‌نهایت میل می‌کند، مقدار $\int_0^T \|d_t\|_{p^{in}}\, dt$ محدود باشد (برای حالتی که تابع خطا، مجموع مربعات باشد این شرط برقرار است). در این صورت به ازای هر $t \in [0,T]$ خواهیم داشت:

$$\Theta^{(L)}(t) \to \Theta_\infty^{(L)}$$

بنابراین $\Theta_\infty^{(L)}$ کرنلی است که رفتار شبکه‌ی عصبی را در خود دارد و همان طور که برای kernel gradient descent گفته شد، اگر ماتریس این کرنل مثبت معین باشد، به نقطه‌ی بهینه‌ی تابع خطا خواهیم رسید.

۲.۵. Kernel Regression

در مسئله‌ی ridge-less kernel regression تابع هزینه به صورت زیر است:

$$C = \frac{1}{2}\sum_{j=1}^{N}\left(W^T\Phi(x_j) - y_j\right)^2$$

بنابراین خواهیم داشت:

$$\frac{\partial \Delta_i^{(t)}}{\partial t} = -\sum_{j=1}^{N} \underbrace{\Phi(x_i)^T \Phi(x_j)}_{K(x_i, x_j)} \underbrace{\left(W^T \Phi(x_j) - y_j\right)}_{\Delta_j}$$

که نشان می‌دهد اگر NTK را داشته باشیم، آموزش شبکه‌ی عصبی در اصل معادل حل کردن یک ridge-less kernel regression توسط NTK است.

۲.۶. Early Stopping

مطابق تعاریفی که در بخش kernel gradient descent داشتیم:

$$\partial_t f(t) = -\Phi_K\left(\partial_f^{in} C|_{f(t)}\right)$$

حال برای حالتی که C مجموع مربعات خطا باشد داریم:

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

پاسخ این معادله‌ی دیفرانسیل به صورت زیر خواهد بود:

$$f_t = f^* + e^{-t\Pi}(f_0 - f^*)$$

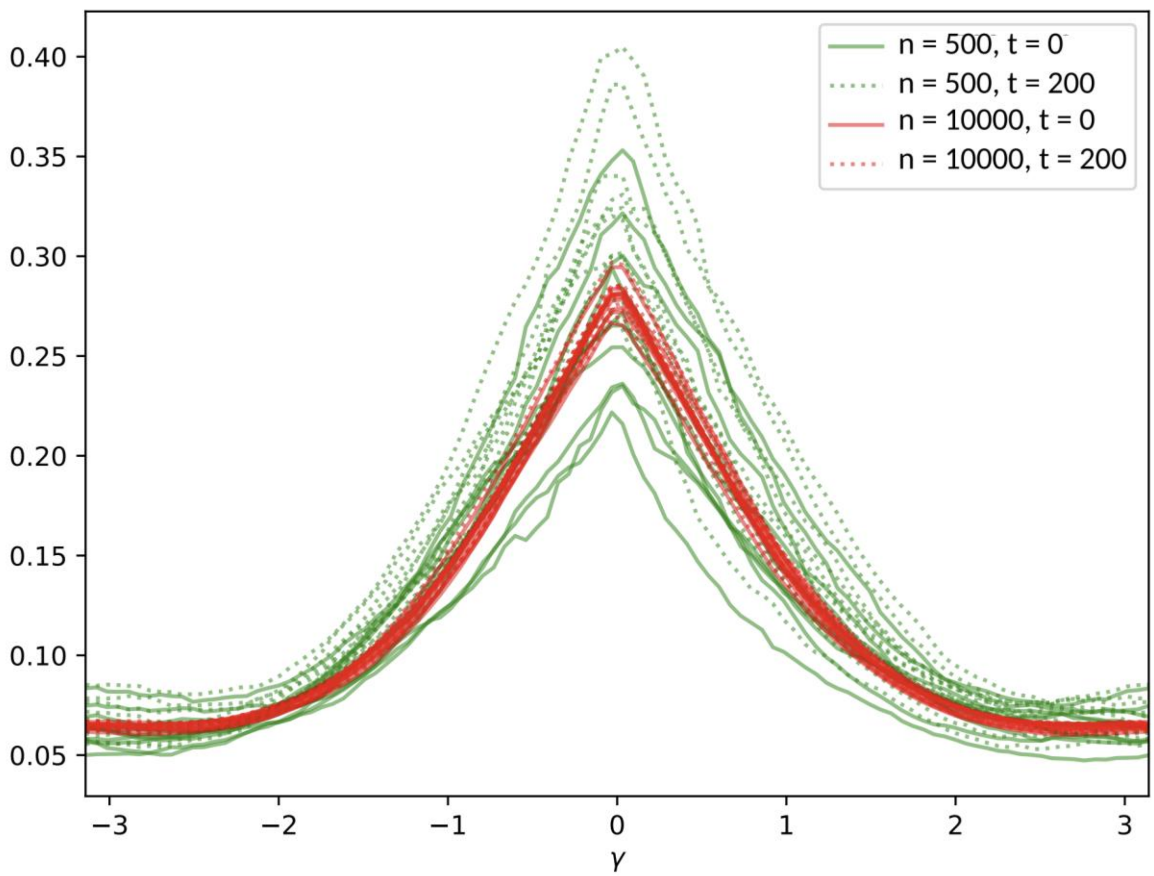
$$\Pi: f \to \Phi_K\left(\langle f, \cdot \rangle_{p^{in}}\right)$$

حال اگر Π دارای جفت های eigen function, eigen value به صورت $f^{(i)}, \lambda_i$ باشد، آنگاه $e^{-t\Pi}$ دارای همان eigen function ها اما با مقدار ویژه های $e^{-t\lambda_i}$ می‌باشد. حال اگر $f^* - f_0$ را بر اساس این eigen function ها decompose کنیم، نرخ همگرایی در جهت $f^{(i)}$ هایی که λ_i بزرگتری دارند بیشتر است. بنابراین اعمال تکنیک early stopping باعث می‌شود در جهت principal component های کرنل حرکت بیشتری شود و سایر جهات که نماینده‌ی نویز داده ها می‌باشند، تاثیر کمتری داشته باشند.

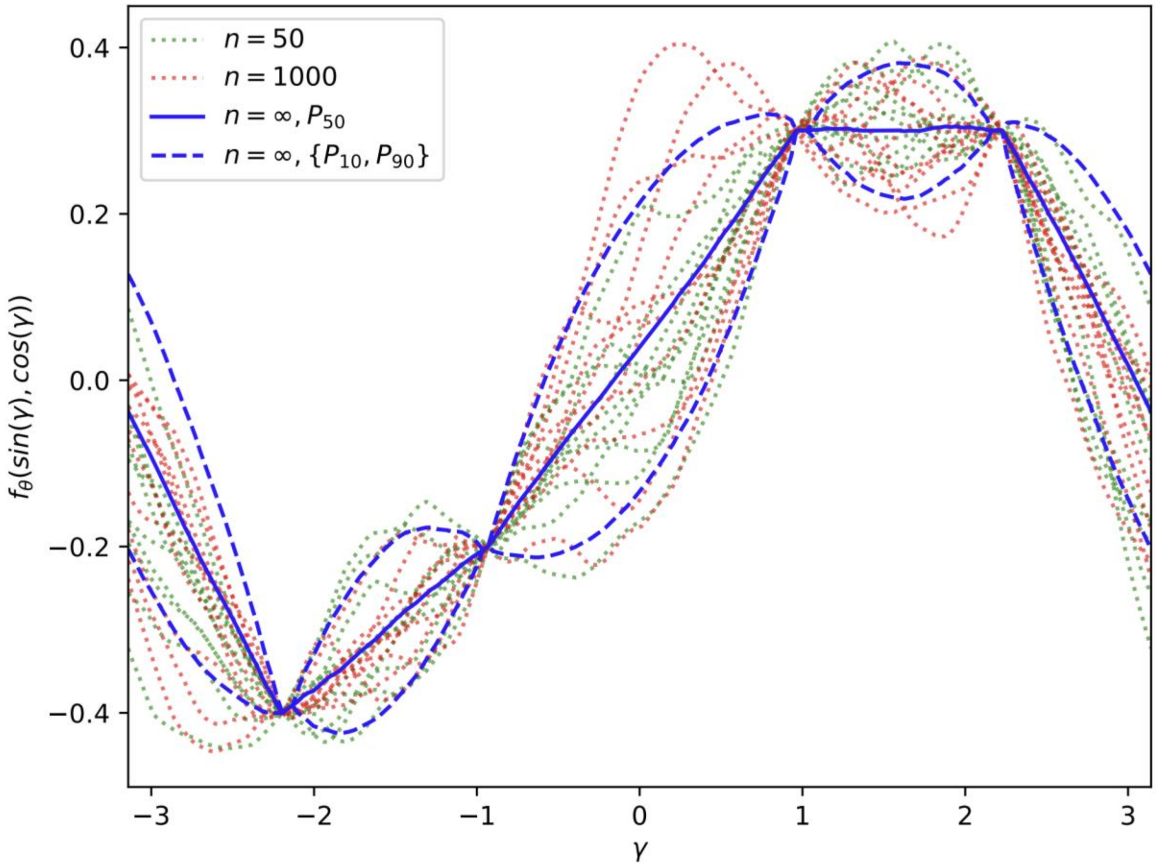
۳. آزمایش عملکرد

به منظور آزمایش روش ارائه شده، آزمایش‌های زیر انجام شده است:

- ابتدا همگرایی NTK بررسی می‌شود. شبکه‌ای با عمق $L = 4$ را برای دو عرض لایه‌ی مختلف $n = 500$ و $n = 10000$ در نظر گرفته و مقدار $\Theta^{(4)}(x_0, x)$ به ازای $x_0 = (1,0)$ و $x = (cos(\gamma), sin(\gamma))$ رسم شده است. همان طور که در تصویر زیر مشاهده می‌شود، با افزایش عرض شبکه، واریانس NTK کمتر بوده و هموار تر است. همچنین مشاهده می‌شود که با افزایش گام ها، نمودار ملتهب می‌شود اما اثر این اتفاق برای شبکه‌ی با عرض بیشتر کمتر می‌باشد. نکته‌ی حائز اهمیت آن است که میانگین برای تمام حالات مشابه هم می‌شود.



- همان طور که پیش تر ذکر شد، خروجی شبکه (یعنی $f_{\theta(t)}$) در حالتی که عرض شبکه به بی‌نهایت میل کند و تابع هزینه، جمع مربعات خطا باشد، یک فرآیند تصادفی گاوسی می‌باشد. در تصویر زیر توزیع گاوسی تئوری در $t \to \infty$ با توزیع خروجی شبکه به ازای $T = 1000$ مقایسه شده است. مشاهده می‌شود که توزیع خروجی شبکه به ازای هر دو عرض $n = 500$ و $n = 10000$ مشابه است و همچنین میانگین و واریانس آن مشابه حالت تئوری است.



$n = 50$
$n = 1000$
$n = \infty, P_{50}$
$n = \infty, \{P_{10}, P_{90}\}$
$f_\theta(sin(\gamma), cos(\gamma))$
γ
0.4
0.2
0.0
−0.2
−0.4
−3
−2
−1
0
1
2
3

۴. نتیجه گیری

به دلیل تغییرات زیاد در روند آموزش، تحلیل تئوری روش های بر پایه‌ی شبکه‌های عصبی کار پیچیده ای محسوب می‌شود. در این مقاله با دیدگاه جدیدی برای بررسی این شبکه ها آشنا شدیم و دیدیم با اعمال چند پیش فرض، رفتار شبکه های fully connected را می‌توان در یک کرنل به نام NTK خلاصه کرد. نشان داده می‌شود که این کرنل در مقدار دهی اولیه به مقداری deterministic میل کرده و در فرآیند آموزش نیز ثابت می‌ماند. این مقدار تنها به عمق شبکه، انتخاب توابع فعال سازی و واریانس پارامتر ها در مقدار دهی اولیه بستگی دارد و اعمال gradient descent بر روی پارامتر ها معادل اعمال kernel gradient descent بر حسب NTK می‌باشد و همگرایی الگوریتم به مثبت معین بودن ماتریس این کرنل بستگی دارد. حال دلیل نام گذاری این کرنل مشخص شده است. neural به دلیل اختصاص کار به شبکه های عصبی و tangent نیز به دلیل وجود گرادیان در تابع تبدیل کرنل می‌باشد.

بدیهی است که پیش فرض های ذکر شده خانواده‌ی بزرگ تمامی مسائل و معماری ها را شامل نمی‌شود اما همچنان دیدگاه ارائه شده، گام بزرگی در راستای تفسیر پذیری شبکه های عصبی به حساب می‌آید.

۵. مراجع

[1] Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, *31*.