Student-T Process Instead of Gaussian Process for NTK

資工21 周聖諺

Motivation

According to the paper **"Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel"** (presented by 袁哥),

- Problem: Infinite-width NTK gives a **poor prediction** on the loss of the **finite-width neural network during training**
- Reason 1: **Final basin(of the loss surface)** chosen by a NN is highly **sensitive to SGD noise** and **is determined in the early stage of training**.
- Reason 2: The empirical NTK of a NN **involves very rapidly in the early stage of training** and becomes stable after the final basin fate is determined.

My Idea: Model the **noise** to the NTK with **Student-T process**

To argue that NTK is sensitive to SGD noise and changes rapidly in the early stage of training, recall

- Hierarchical Exploration of Loss Landscape through Parents and Children
- Visualization of The Function Space Motion During Training
- Error Barrier Between Spawned Children During Training
- Kernel Distance During Training

Hierarchical Exploration of Loss Landscape through Parents and Children

- In this process, a parent network is trained from initialization to a spawning time t_s, yielding a parent weight trajectory $\{w_t\}_{t=0}^{t_s}$.
- At the spawn time t_s, several copies of the parent network are made, and these so-called children are then training with independent minibatch stochasticity, yielding different child weight trajectories $\{w_t^{t_s,a}\}_{t=t_s}^{T}$, where a indexes the children, and T is the final training time.

Visualization of The Function Space Motion During Training

Function Distance

- To compute the distance between the two functions f_w and $f_{w'}$, parameterized by weights w and w_0, we would ideally like to calculate the **degree of disagreement between their outputs averaged over the whole input space** x.
- Let S test denote the test set. Then,

$$||f_w(x) - f_{w'}(x)||_{S^{test}} = \frac{1}{Z|S_x^{test}|} \sum_{x \in S_x^{test}} (f_w(x) \neq f_{w'}(x))1$$

Where S_x^{test} are test inputs and Z is normalizing constant.

- T-SNE visualization of parent and children evolution in the function space with different spawn epoch.
- The trajectories of the children are highly **sensitive to SGD noise**

ResNet20 on CIFAR100, total training epoch: 200

Error Barrier Between Spawned Children During Training

- Compute the error barrier between children along a linear path interpolating between them in weight space.
- Let $w_t^\alpha = \alpha w_t + (1-\alpha)w_t'$ where w_t and w_t' are the weight of 2 children networks, spawn from some iteration t_s, and $\alpha \in [0,1]$.
- At various t_s we compute $\max_{\alpha\in[0,1]} \hat{R}_S(w_t^\alpha) - \frac{1}{2}(\hat{R}_S(w_t) + \hat{R}_S(w_t'))$, which we call the **error barrier**.

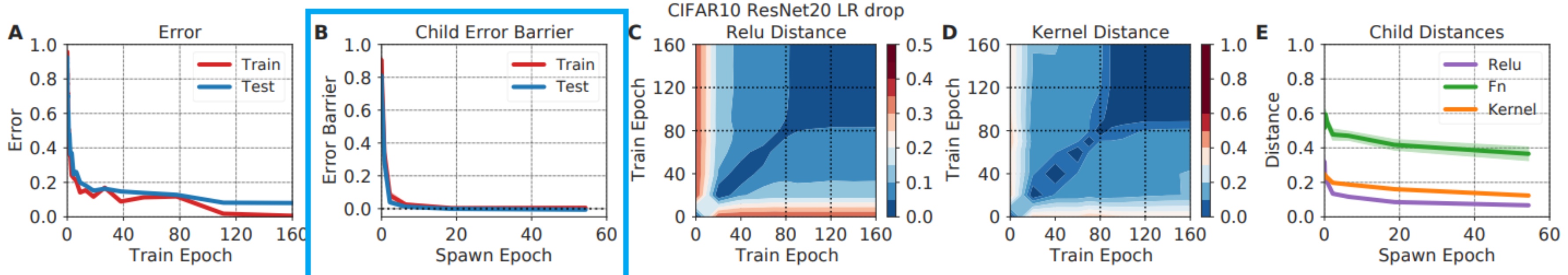

Figure 2: SOTA ResNet20 trained on CIFAR10 using SGD with momentum and learning rate drops.

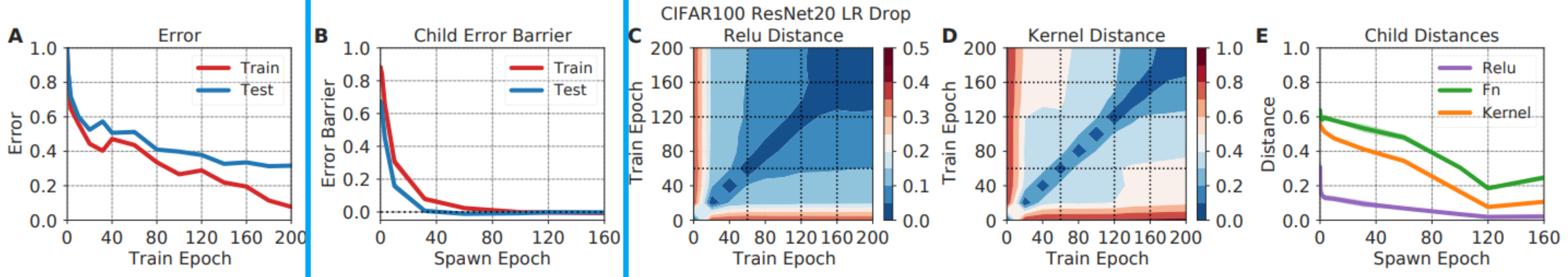

Figure 3: ResNet20 trained on CIFAR100 using SGD with momentum and learning rate drops.

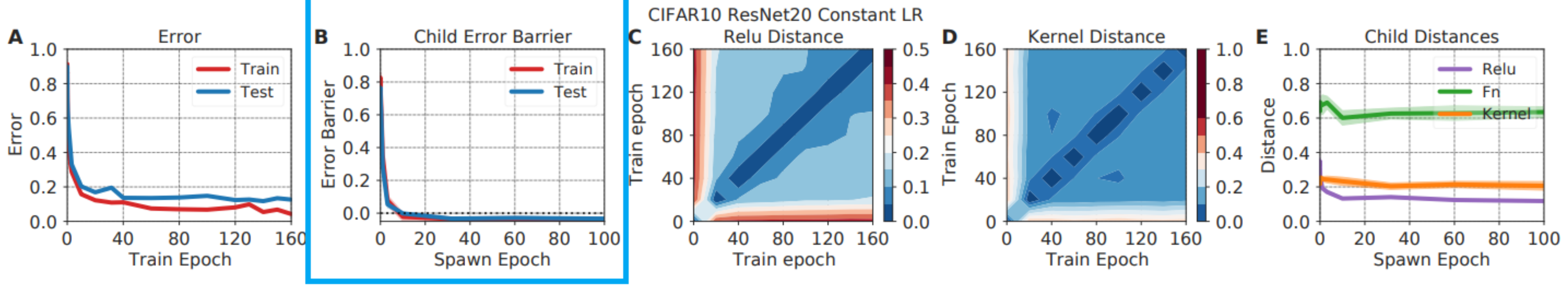

Figure 4: ResNet20 trained on CIFAR10 using SGD with momentum and constant learning rate.

the error barrier drops rapidly within a few epochs in panel B,

Kernel Distance During Training

Kernel Distance

For finite width networks, the kernel $\kappa_t(S) = \kappa_{w_t}(S)$ changes with training time t.

Define the kernel distance as

$$S(w, w') = 1 - \frac{Tr(\kappa_w(S)\kappa_{w'}^T(S))}{\sqrt{Tr(\kappa_w(S)\kappa_w^T(S))}\sqrt{Tr(\kappa_{w'}(S)\kappa_{w'}^T(S))}}$$

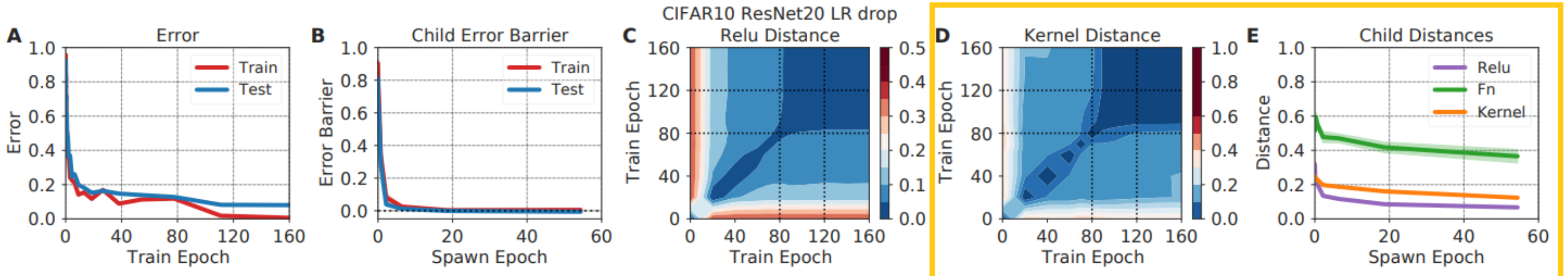

Figure 2: SOTA ResNet20 trained on CIFAR10 using SGD with momentum and learning rate drops.

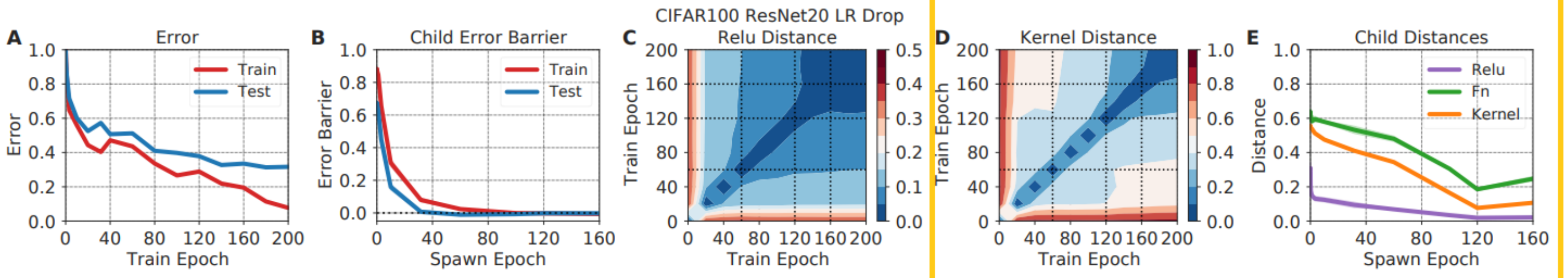

Figure 3: ResNet20 trained on CIFAR100 using SGD with momentum and learning rate drops.

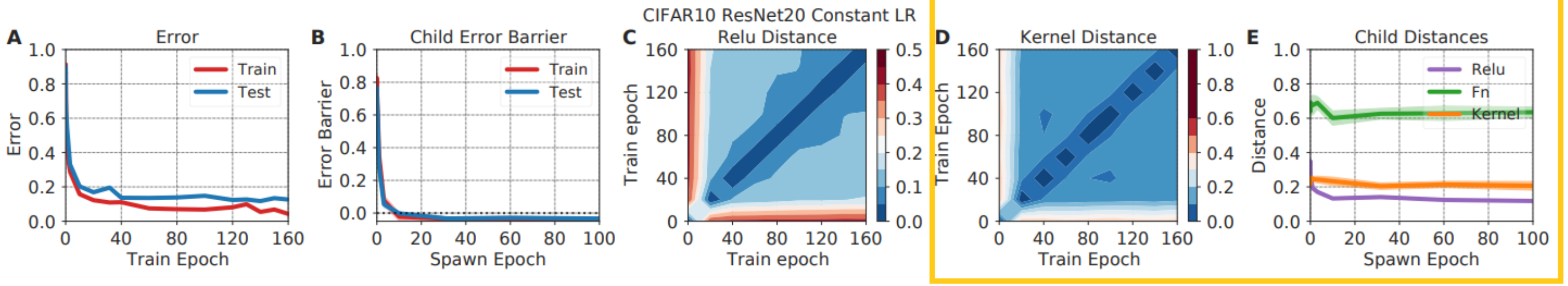

Figure 4: ResNet20 trained on CIFAR10 using SGD with momentum and constant learning rate.

Panel E shows that function, kernel and ReLU distances between children at the end of training also drop as a function of spawn time.

Main Idea

Why not just add some random noise to the NTK, to simulate the unstable NTK?

With **inverse Wishart distribution** $\Sigma \sim \mathcal{IW}(\nu, K)$ where the degree of freedom $\nu > 2$ and K is positive definite, we can generate a random matrix Σ with expectation $\mathbb{E}[\Sigma] = \frac{K}{\nu - 2}$.

Inverse Normal-Wishart Distribution

Consider the **randomness of the kernel matrix**, with Bayesian rule, we can model the kernel matrix with **Inverse Wishart distribution**.

$$p(y|\phi, \nu, K) = \int p(y|\phi, \Sigma)p(\Sigma|\nu, K)d\Sigma = \mathcal{TP}(\phi, K, \nu)$$

Denote Inverse Wishart distribution as $\Sigma|\nu, K \sim \mathcal{IW}(\nu, K) = p(\Sigma|\nu, K)$ and the Gaussian process as $y|\phi, \Sigma \sim \mathcal{GP}(\phi, \Sigma) = p(y|\phi, \Sigma)$. Note that the random matrix $\Sigma \in \mathbb{R}^{N\times N}$ is generated by Inverse Wishart. The hyperparameters are $K \in \mathbb{R}^{N\times N}$, $\nu \in \mathbb{R}$, $\phi \in \mathbb{R}^N$.

Finally, $\mathcal{TP}(\phi, K, \nu)$ is the **Student-T process**.

Student-T Process

The Student-T process can be written as

$$y \sim \mathcal{TP}(\phi, K, \nu)$$

The hyperparameter ν is called **degrees of freedom**, it can control the covariance of the output $cov(y) = \frac{\nu}{\nu-2}K$. Thus, when $\nu \to \infty$, $\mathcal{TP}$ will converge to $\mathcal{GP}$ in distribution.

$$P(X) = \lim_{\nu \to \infty} P(Y)$$

$$X \sim \mathcal{GP}(\phi, K), \quad Y \sim \mathcal{TP}(\phi, K, \nu),$$

Student-T Process

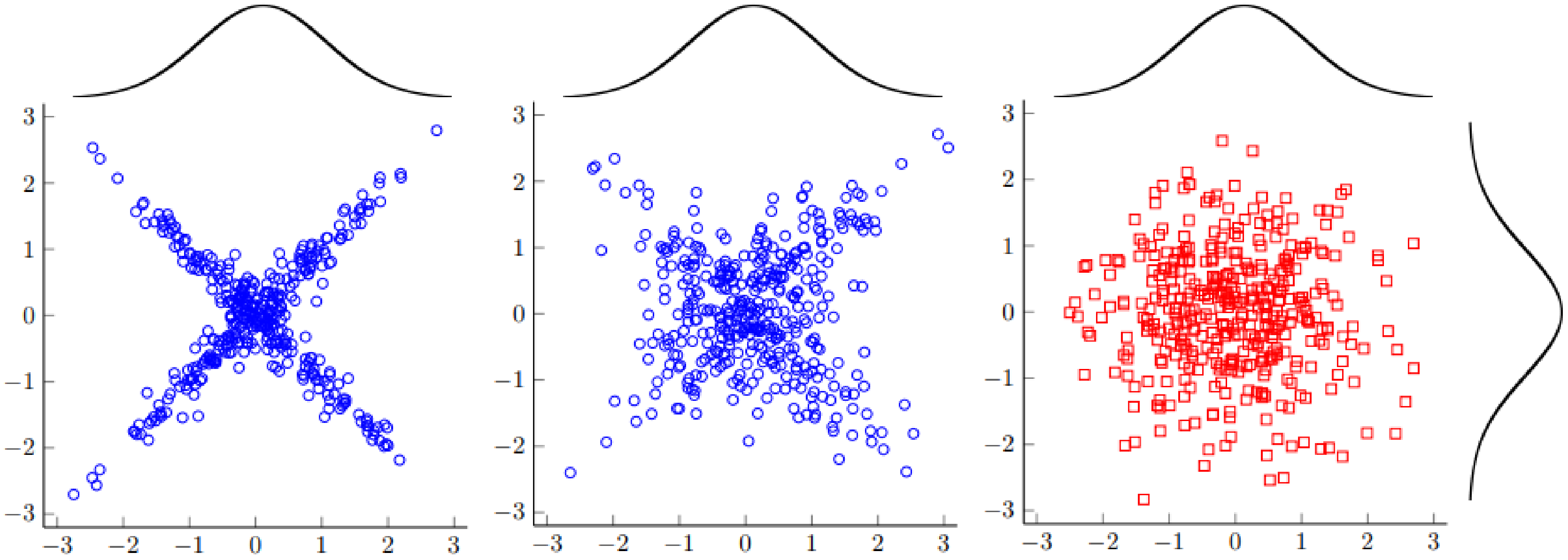

Figure 2: Uncorrelated bivariate samples from a Student-t copula with $\nu = 3$ (left), a Student-t copula with $\nu = 10$ (centre) and a Gaussian copula (right). All marginal distributions are $\mathrm{N}(0, 1)$ distributed.

Degrees of freedom is the number of values in the final calculation of a statistic that are free to vary.

Student-T Process Posterior

Given a training dataset $\{X_1, Y_1\}$ with n_1 samples and a testing dataset $\{X_2, Y_2\}$ with n_2 samples where $X_1 \in \mathbb{R}^{n_1 \times d}$, $Y_1 \in \mathbb{R}^{n_1}$, and $X_2 \in \mathbb{R}^{n_2 \times d}$, $Y_2 \in \mathbb{R}^{n_2}$.

Denote the mean function as z and the kernel function as k. Thus, $\phi_i = z(X_i)$ and $K_{ij} = k(X_i, X_j)$

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \sim \mathcal{TP}(\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}, \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}, \nu)$$

Student-T Process Posterior

The posterior is

$$y_2|y_1 \sim \mathcal{TP}(\hat{\phi}, \frac{\nu + \beta + -2}{\nu + n_1 - 2}\hat{K}_{22}, \nu + n_1)$$

Where

$\hat{\phi} = K_{21}K_{11}^{-1}(Y_1 - \phi_1) + \phi_2$

$\beta = (Y_1 - \phi_1)^\top K_{11}^{-1}(Y_1 - \phi_1)$

$\hat{K}_{22} = K_{22} - K_{21}K_{11}^{-1}K_{12}$

Experiment

width:600px

We've already known, if $\nu = \infty$, the student-T process will converge to Gaussian process. In the above figure, we compare the result of **fitting a $\sin()$ function with $\mathcal{TP}$ (blue) and $\mathcal{GP}$(yellow) respectively**. The **left** part of above figure shows the **training/testing progress of fitting**. The right part is the value of ν during training progress.

As the **blue line** shows above(left part), as the **training progress goes**, **the ν gets lower**. It shows that we can **control ν of $\mathcal{TP}$ to achieve a better fitting(closer to black line, real NN training)**.

Conclusion

- During **training progress**, the ν **gets lower**.
- By controlling the value of ν of $\mathcal{TP}$, we can get a **more accurate prediction on the training loss** rather than $\mathcal{GP}$
- If the NTK follows the $\mathcal{TP}$, the **bigger training dataset**, the **larger degree of freedom of the posterior**.

Thanks For Listening