Understanding Stability in Neural Network Training Dynamics

Shreyas Kalvankar, Advised by: David Tax

Why this is interesting

- **Scaling works**, but mechanisms remain partly opaque.
- **Infinite width** is a solvable baseline: randomness averages out.
- **Finite width** is where modern models live: features move.
- Understanding the finite-width *training dynamics* may expose **stability points** and regimes that predict when features start to learn.

Setup

Supervised data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, network $f(\boldsymbol{x}; \boldsymbol{\theta})$, squared loss $L(\boldsymbol{\theta}) = \frac{1}{2}\sum_i (f(\boldsymbol{x}_i; \boldsymbol{\theta}) - y_i)^2$.

Linearization at init:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) \approx f(\boldsymbol{x}; \boldsymbol{\theta}_0) + \underbrace{\nabla_{\boldsymbol{\theta}} f^\top(\boldsymbol{x}; \boldsymbol{\theta}_0)}_{\phi^\top(\boldsymbol{x})}(\boldsymbol{\theta} - \boldsymbol{\theta}_0).$$

NTK at init: $K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$ (Jacot et al., 2018).

Training dynamics: from constant to drifting kernels

Start from gradient flow on the loss

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t), \quad L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{j=1}^{n} \big(f(x_j; \boldsymbol{\theta}) - y_j\big)^2.$$

The gradient of L with respect to the parameters is

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) = \sum_{j=1}^{n} \big(f_t(x_j) - y_j\big)\, \nabla_{\boldsymbol{\theta}} f(x_j; \boldsymbol{\theta}_t).$$

Using the chain rule for the network outputs $f_t(x_i) := f(x_i; \boldsymbol{\theta}_t)$,

$$\frac{df_t(x_i)}{dt} = \nabla_{\boldsymbol{\theta}} f(x_i; \boldsymbol{\theta}_t)^\top \frac{d\boldsymbol{\theta}_t}{dt} = -\sum_{j=1}^{n} \underbrace{\nabla_{\boldsymbol{\theta}} f(x_i; \boldsymbol{\theta}_t)^\top \nabla_{\boldsymbol{\theta}} f(x_j; \boldsymbol{\theta}_t)}_{K_t(x_i, x_j)} \big(f_t(x_j) - y_j\big).$$

Here K_t is the (time-dependent) Neural Tangent Kernel (NTK).

Constant-kernel (lazy) regime: if $K_t \approx K_0 \equiv K$ (as in the infinite-width limit),

$$\frac{df_t}{dt} = -K\,(f_t - y) \quad \Rightarrow \quad f_t = y + e^{-Kt}(f_0 - y).$$

Finite width breaks this constant-kernel assumption: during training the features and kernel evolve, producing measurable *kernel/feature drift*.

Stability and fixed points: linking to linear systems

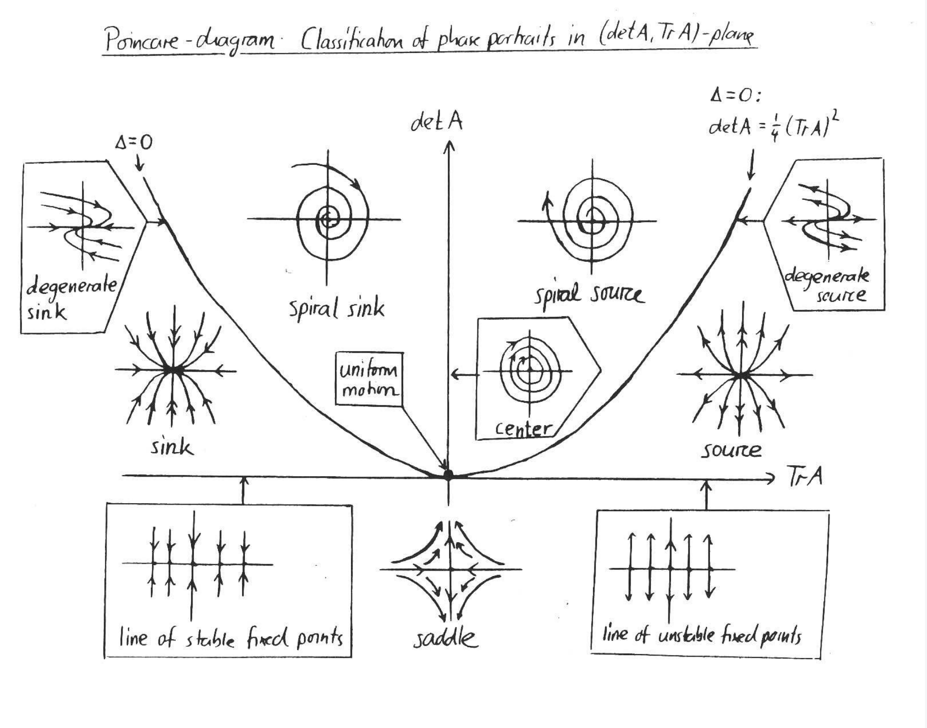
Start with a linear, homogeneous system (in residual space). Let $r_t := f_t - y$. Under squared loss and gradient flow, and in the constant–kernel (lazy/infinite-width) regime,

$$\frac{dr_t}{dt} = \frac{d}{dt}(f_t - y) = \frac{df_t}{dt} = -K(f_t - y) = -K\, r_t \quad \Longrightarrow \quad r_t = e^{-Kt}\, r_0.$$

This is a linear system because the change in r_t depends only on its current value, scaled by a fixed matrix K.

Eigenvalues govern stability. Each eigenvalue of K describes how quickly a particular direction in the residual space changes during training:

- $\lambda_i > 0$: exponential decay $\Rightarrow$ **stable (sink)** — errors along this direction shrink quickly.
- $\lambda_i \approx 0$: very slow change $\Rightarrow$ **flat / plateau** — these errors hardly change unless features move.
- Mixed signs (in general systems): **saddle-like** behavior — some directions improve, others worsen.



Phase portraits of $\dot{\mathbf{x}} = A\mathbf{x}$ illustrate these cases: eigenvalues determine whether trajectories converge, stall, or diverge. Here, $A = -K$ plays the same role for the residual dynamics $r_t = f_t - y$.
(Image credit: Douglas Hundley, Differential Equations Math 244, Spring 2025)

Finite width $\Rightarrow$ time-varying K_t. As features change, K_t and its eigenvalues drift. Stable directions can weaken, flat ones can become learnable, the stability picture itself evolves during training.

Key messages (TL;DR)

- Infinite width gives a **clean, predictive baseline** (kernel regression).
- Finite width introduces **feature learning** captured by kernel/feature drift.
- **Stability points** = regions where drift is small (lazy) vs. large (adaptive).

References

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

Open questions for discussion

- How does K_t evolve during realistic training?
- Could stability analysis guide when to leave or re-enter the lazy regime?