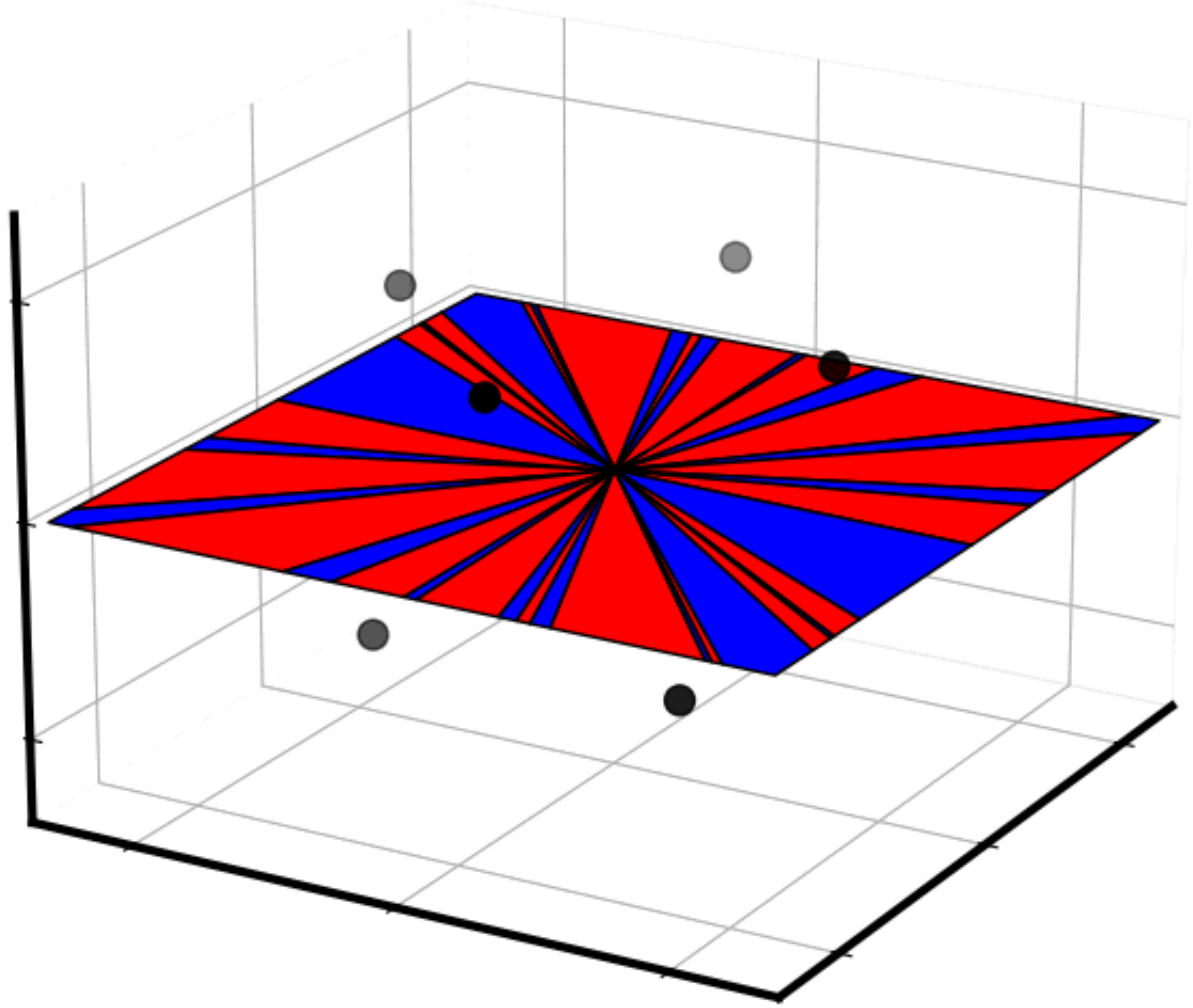
An Introduction to the Learning Dynamics of Neural Networks:

Conservation Laws, Implicit Biases, and Feature Learning



Daniel Kunin

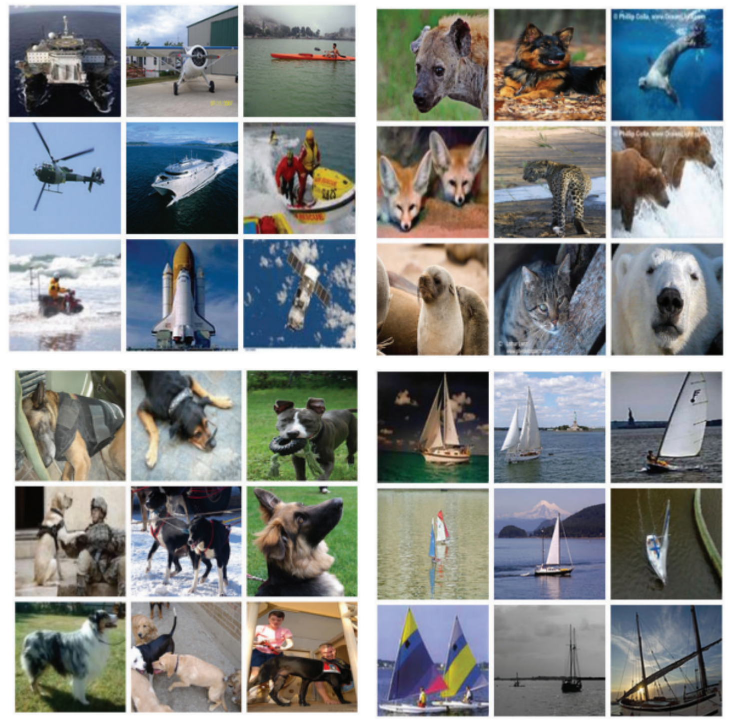
PhD Stanford University woking with Surya Ganguli

Incoming Miller Fellow UC Berkeley

Physics for AI Workshop, Oxford, March 19 2025

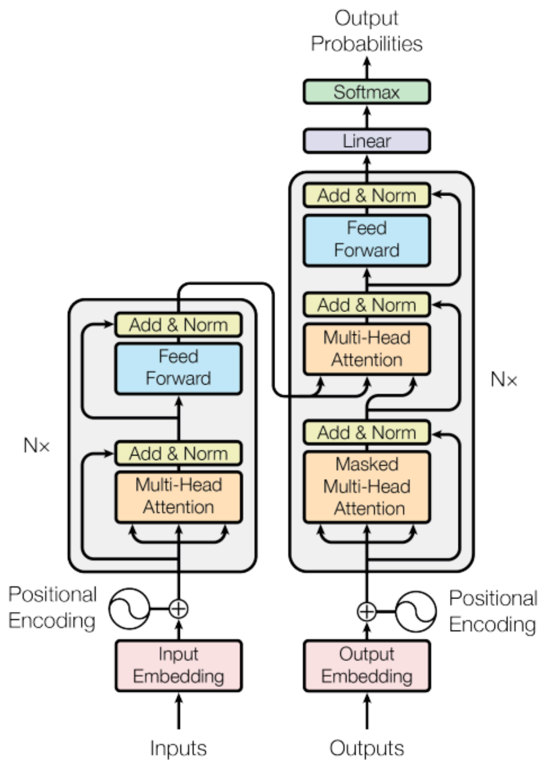
Deep learning has revolutionized AI:

Computer Vision



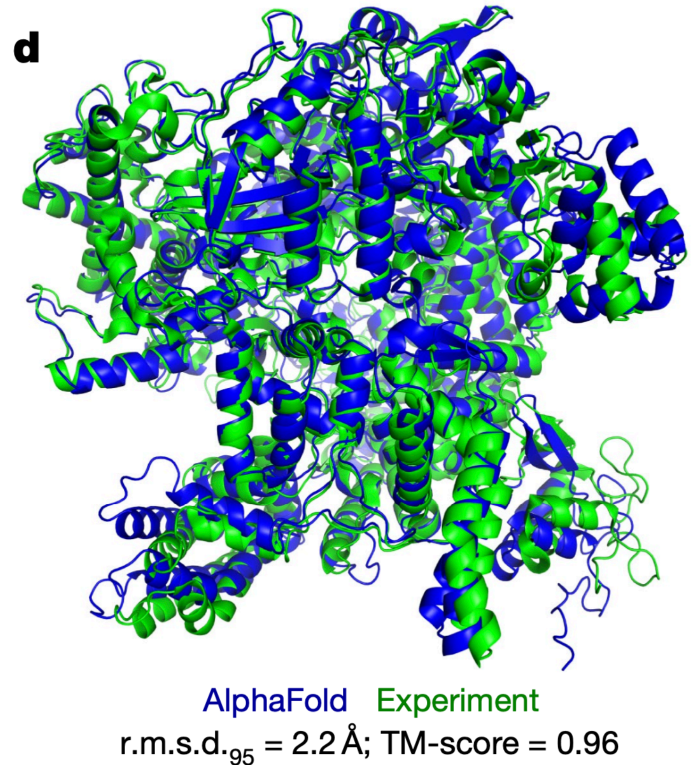
Deng et al., 2009

Natural Language Processing



Vaswani et al., 2017

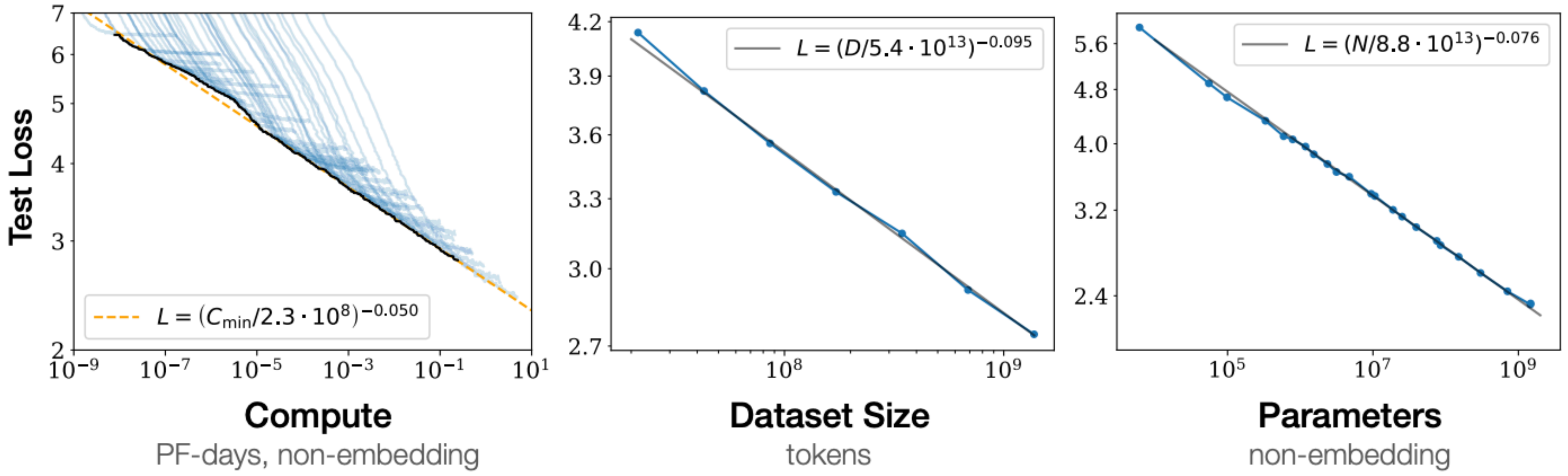
Scientific Modeling



Jumper et al., 2021

These advances have largely been driven by <u>increasing computational scale</u>:

Kaplan et al., 2020



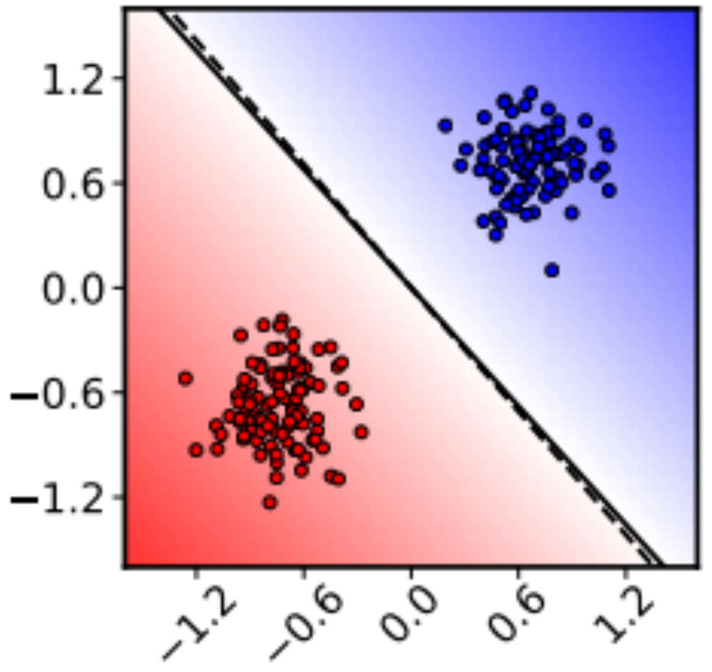
Despite the success, a **fundamental question** remains:

What are the mathematical principles governing a neural network’s ability to generalize?

Two prevailing perspectives

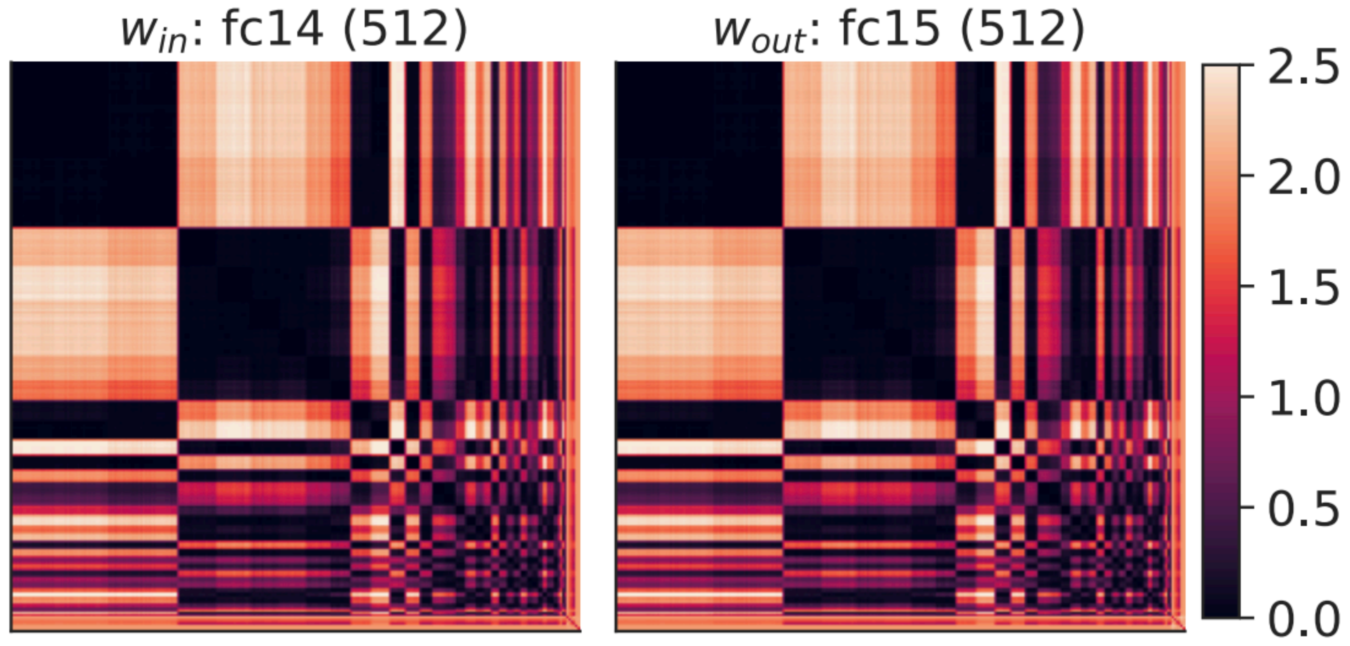
Implicit Bias Perspective: Overparameterized neural networks can memorize their training data, but the architecture and training process implicitly bias them toward simple generalizing solutions.

Maximum-Margin



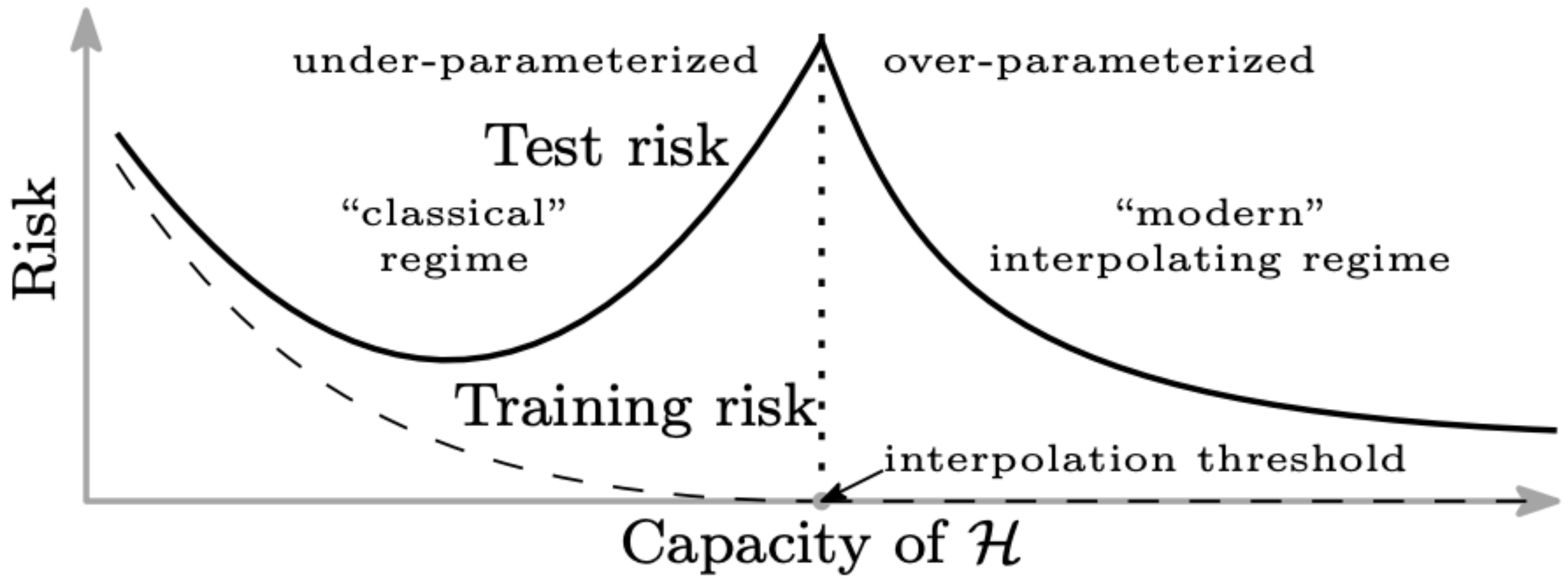
Soudry et al., 2018

Low-rank / Sparse / Low-entropy



Chen et al., 2024

Benign Overfitting

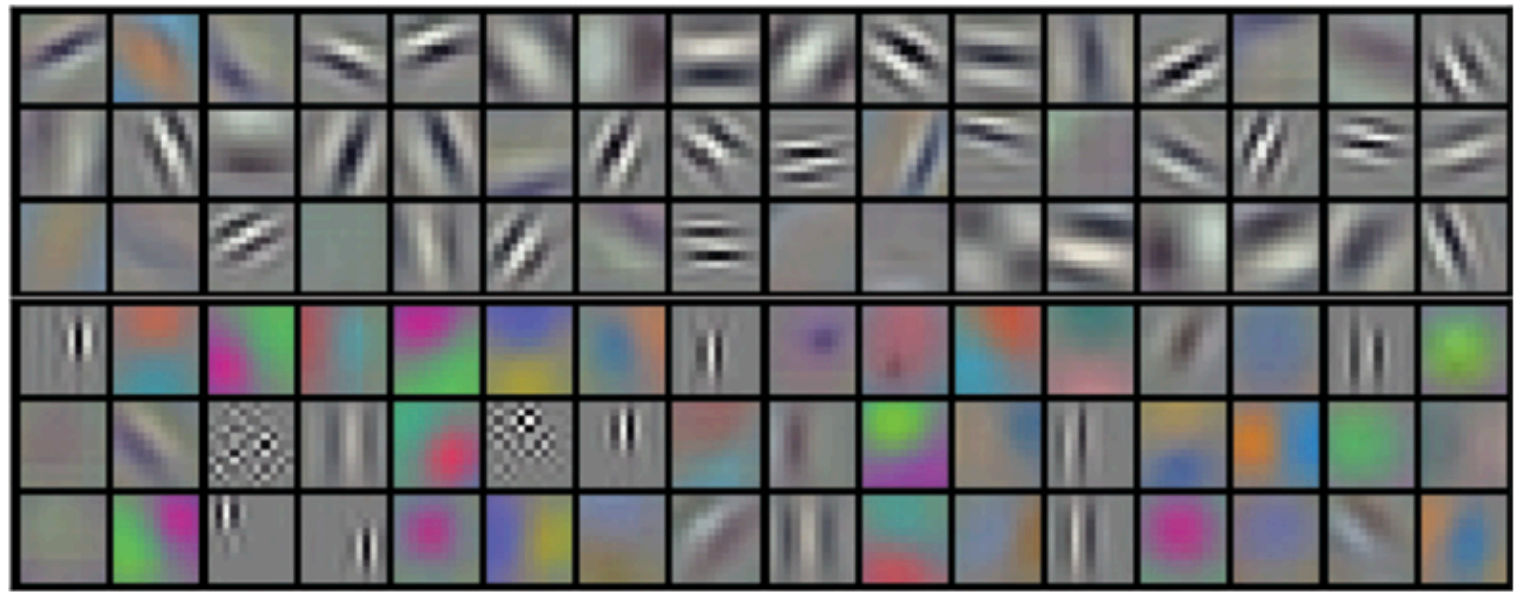


Belkin et al., 2019

Two prevailing perspectives

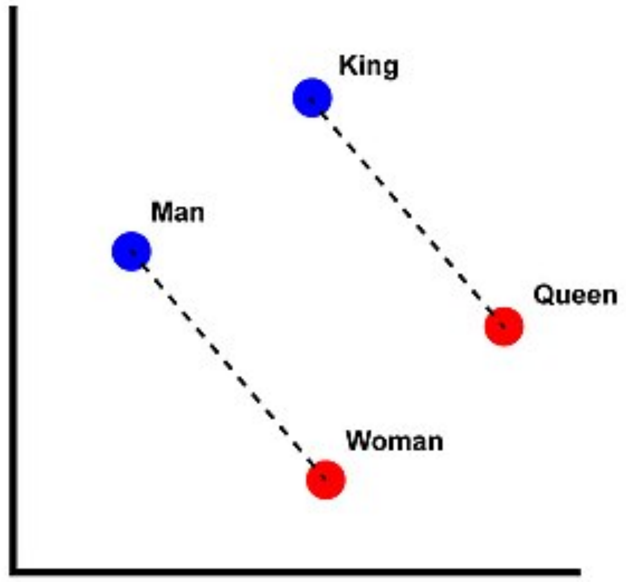
Feature Learning Perspective: A neural network's performance depends on its <u>architecture's</u> ability to efficiently extract and compose task-relevant <u>features from data</u>.

<u>Gabor Filters in First-layer Weights</u>



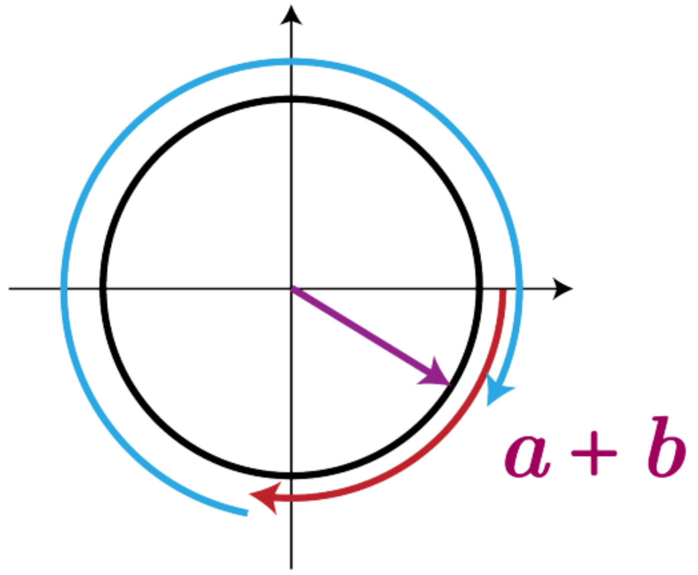
Krizhevsky et al. (2012)

<u>Meaningful word embeddings</u>



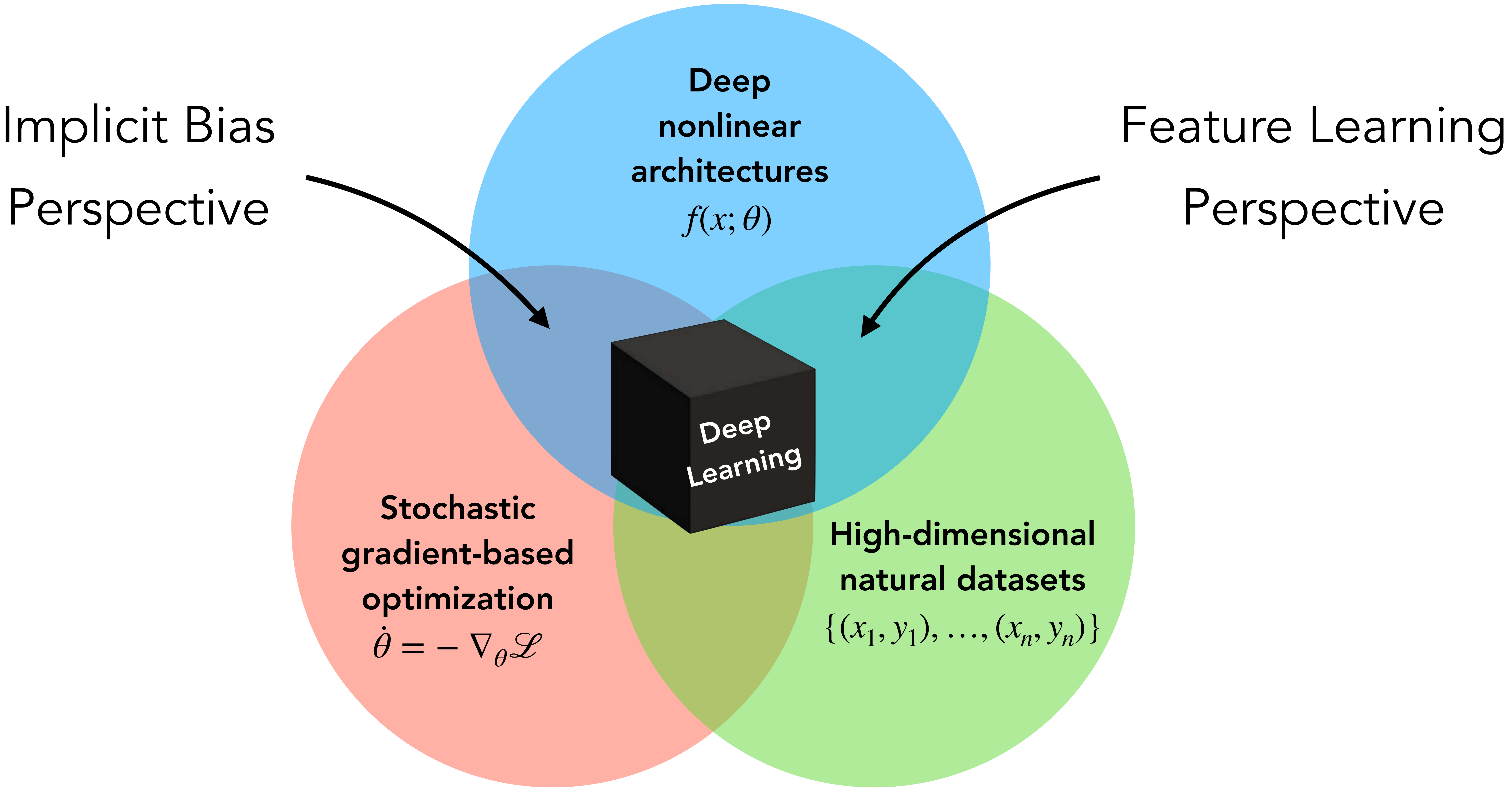
Mikolov et al. (2013)

<u>Mechanistic Interpretability</u>



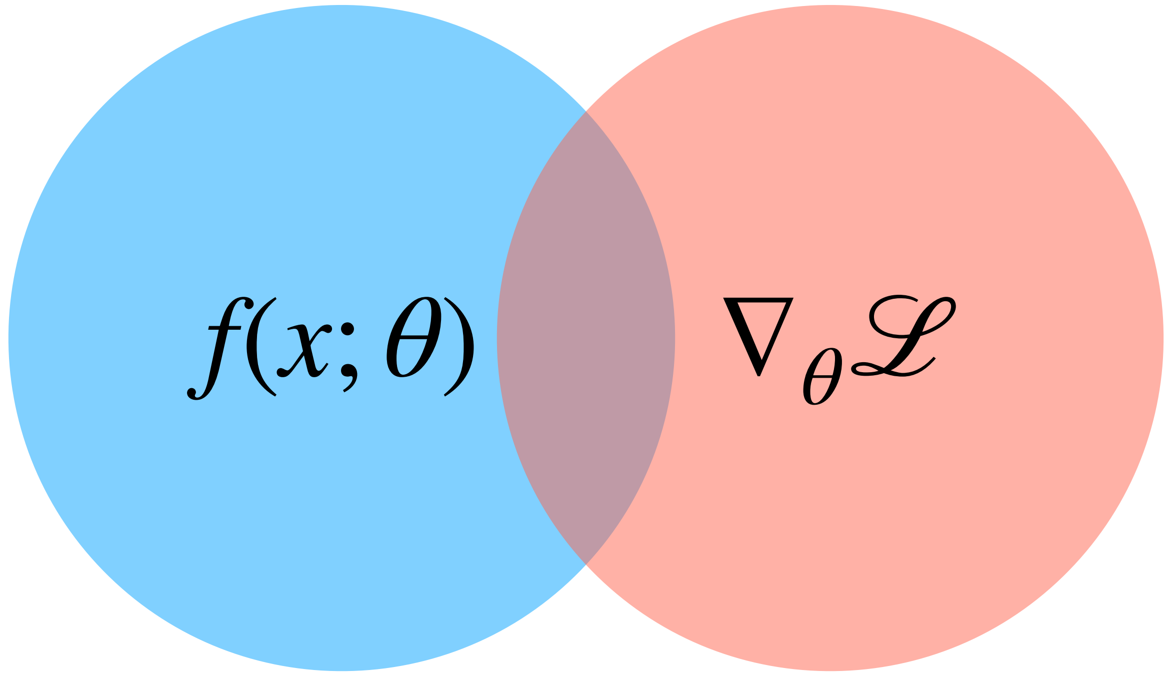
Nanda et al. (2023)

Unraveling a complex interaction of three ingredients



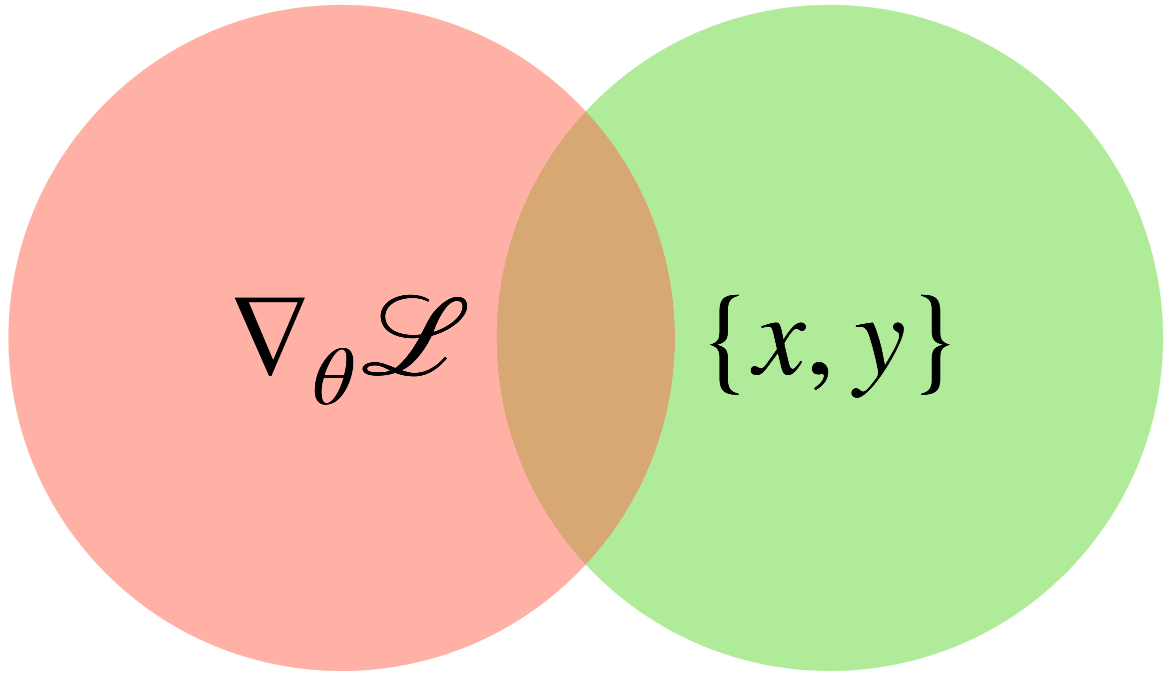
Q1: What conditions enable feature learning?

When and why does feature learning emerge.



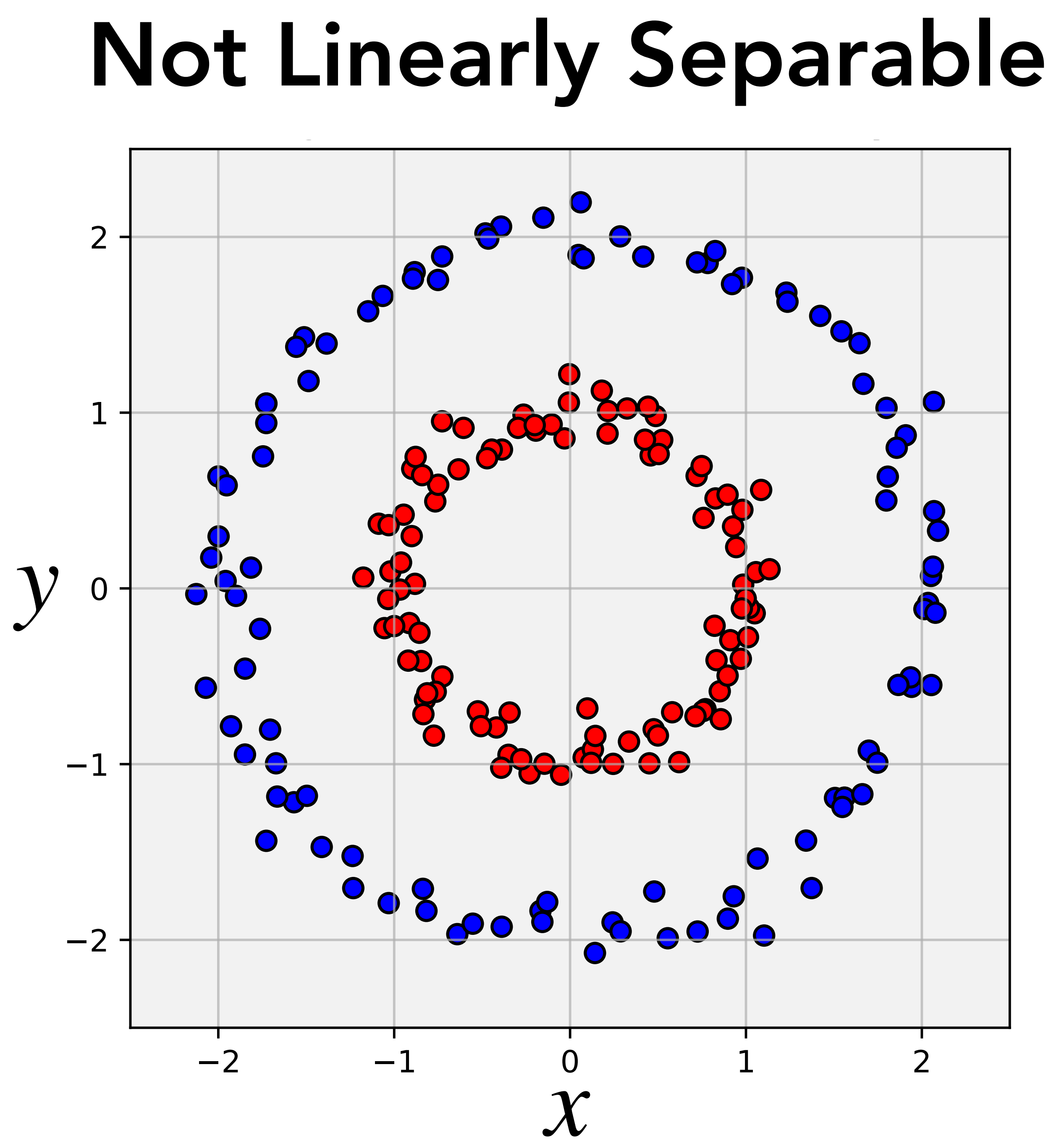
Q2: What mechanisms drive feature learning?

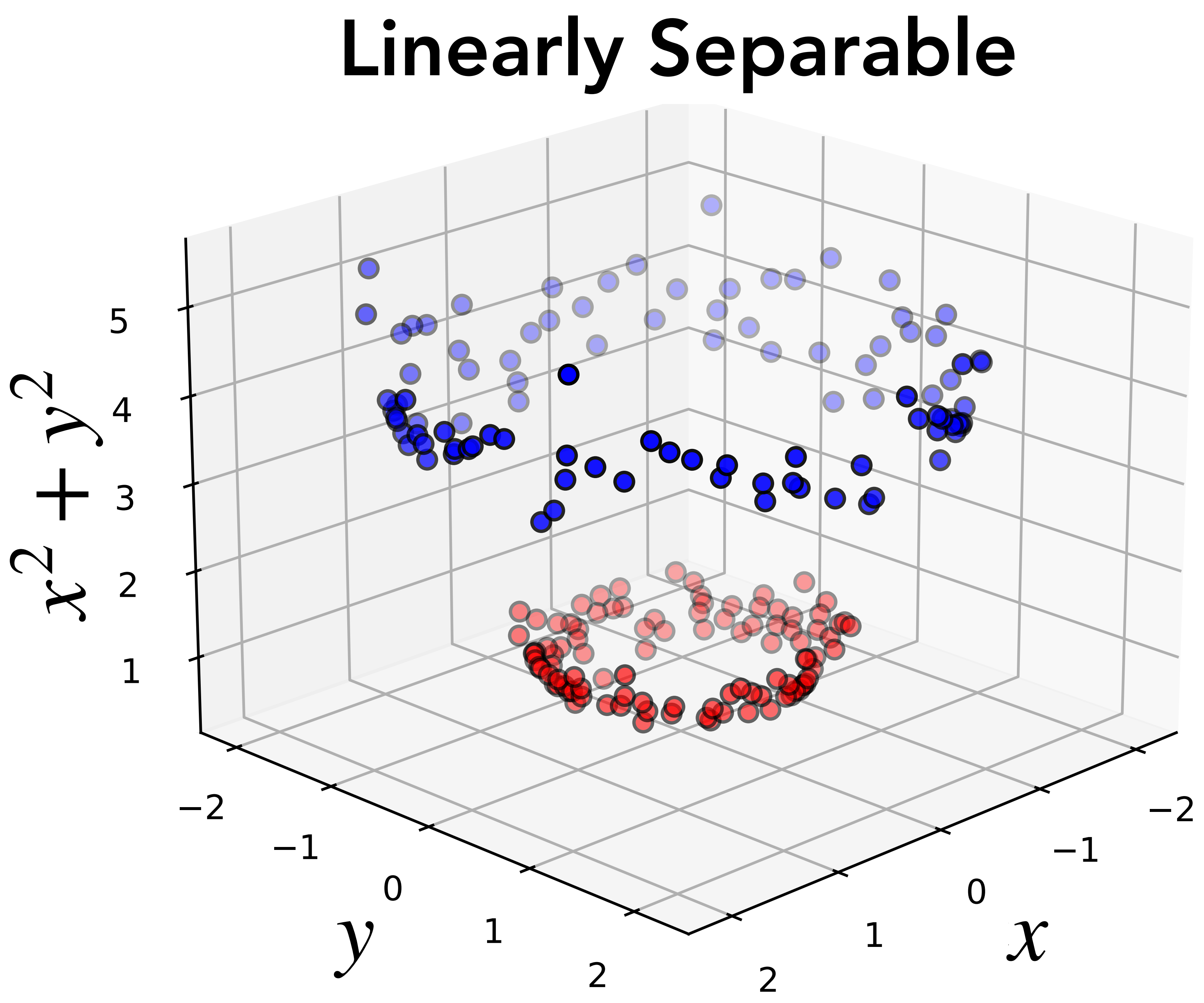
What features do neural networks learn, and how.



What is a feature?

Informally, a **feature** is a *representation* of data that a model uses to make predictions.

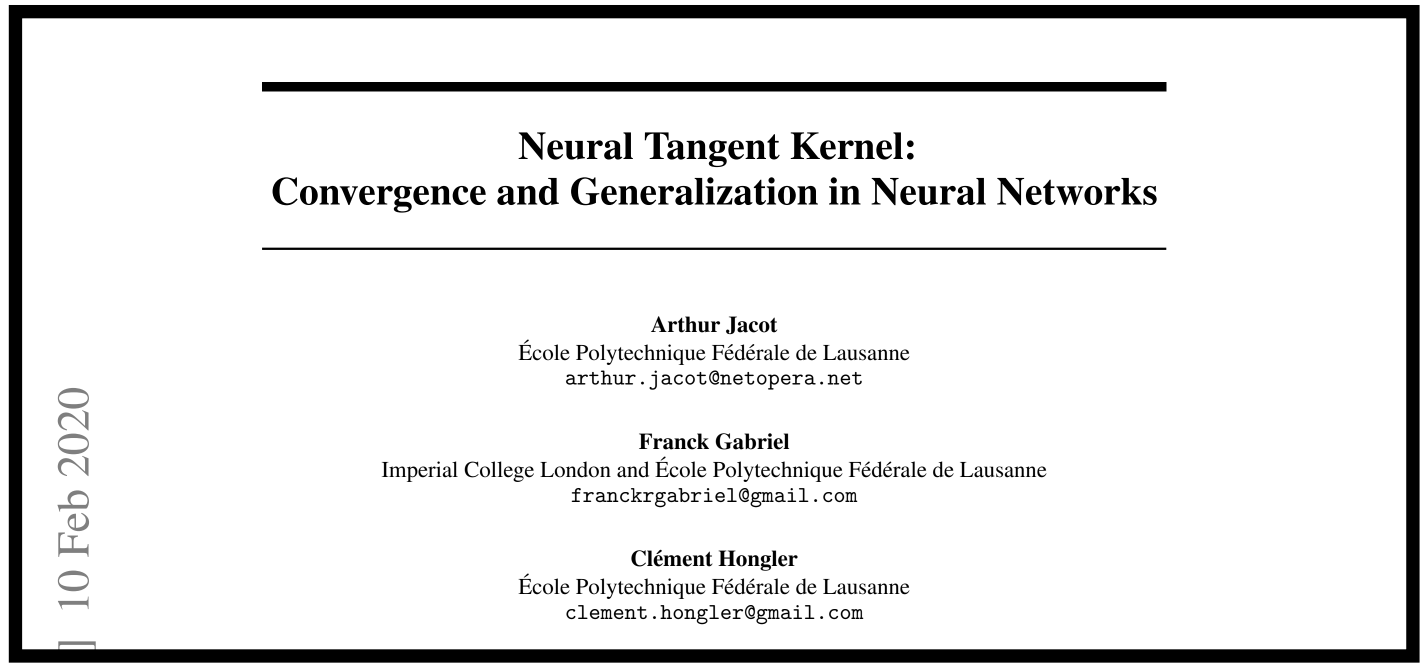




Learning the right features is critical. Traditional ML relied on hand-crafted feature selection, while neural networks automatically learn features

When Feature Learning ***Doesn't*** Happen

When a network becomes linear in θ: $$f(x;\theta) \to f(x;\theta_0) + (\theta - \theta_0)\overbrace{\nabla_\theta f(x;\theta_0)}^{\text{feature map}}$$



Neural Tangent Kernel:
Convergence and Generalization in Neural Networks

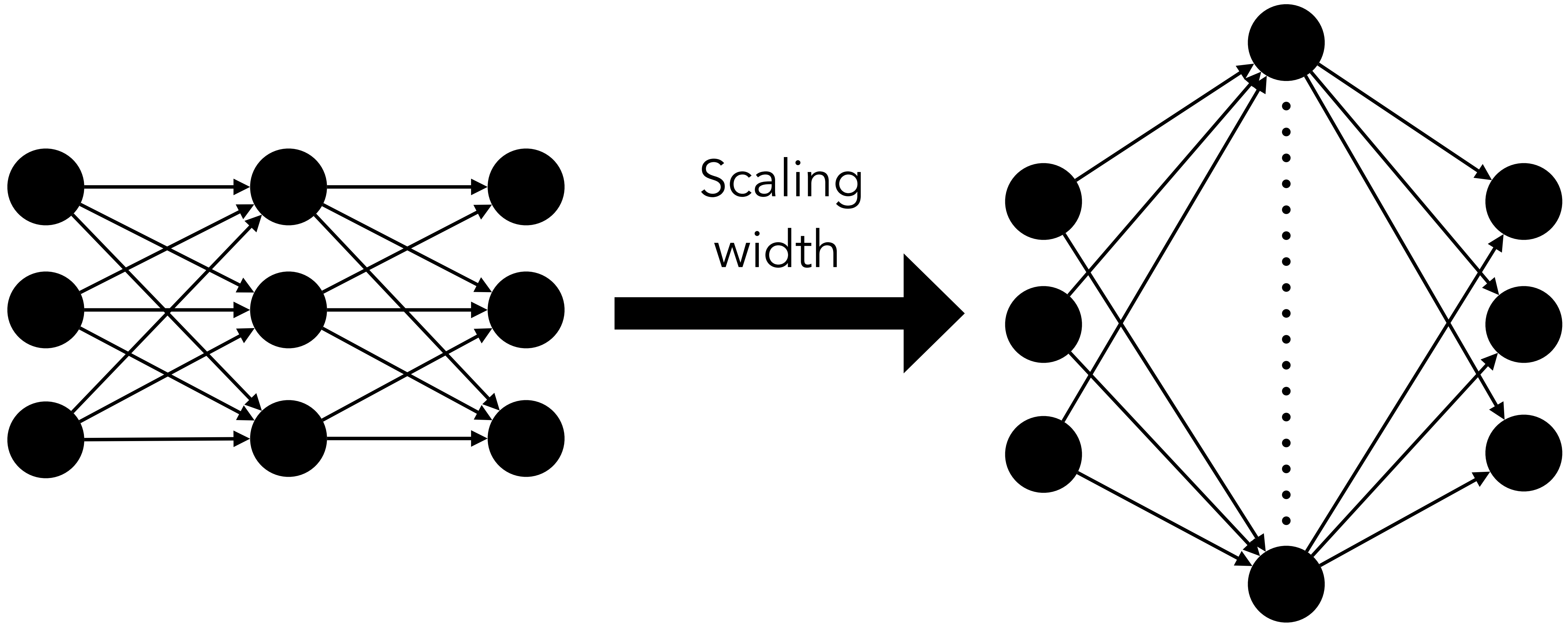
Arthur Jacot
École Polytechnique Fédérale de Lausanne
arthur.jacot@netopera.net

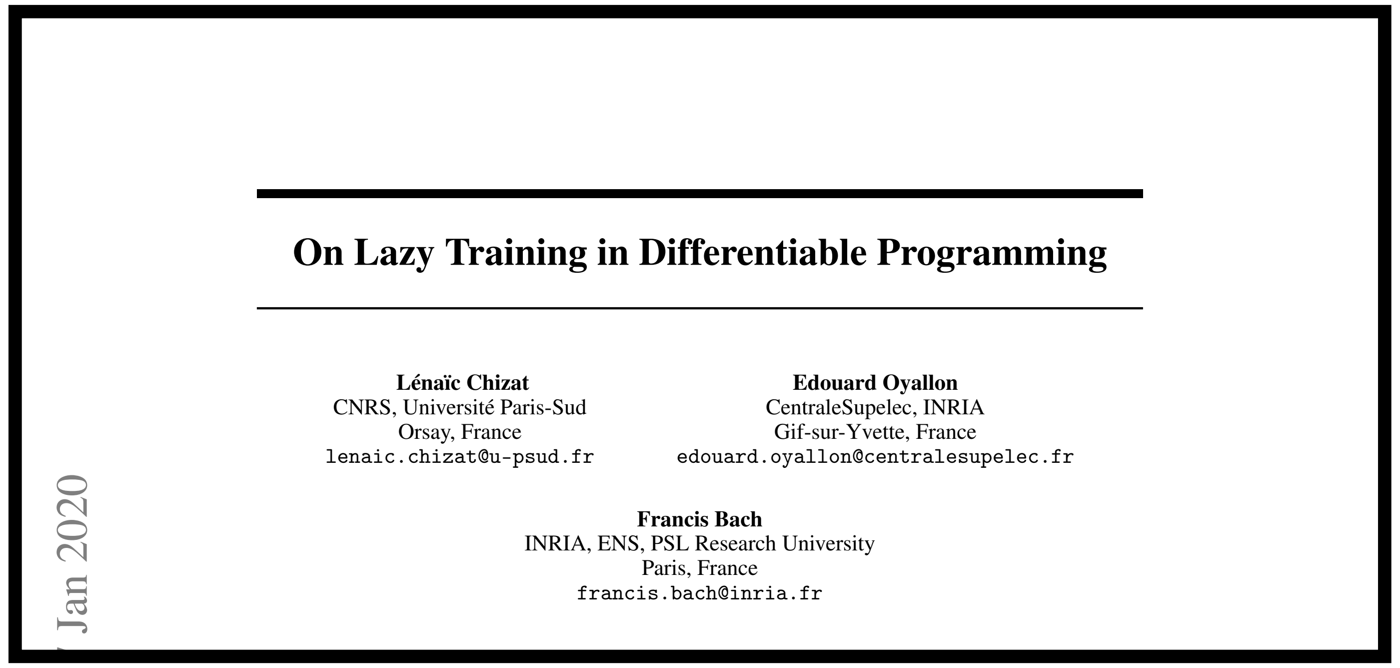
Franck Gabriel
Imperial College London and École Polytechnique Fédérale de Lausanne
franckrgabriel@gmail.com

Clément Hongler
École Polytechnique Fédérale de Lausanne
clement.hongler@gmail.com

10 Feb 2020

Jacot et al. (2018)





On Lazy Training in Differentiable Programming

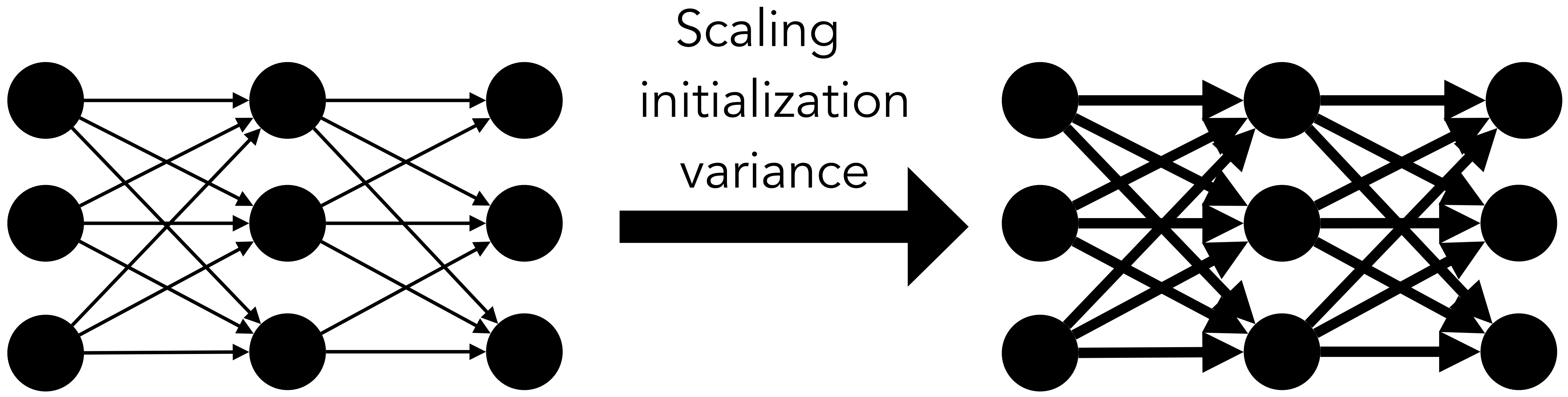
Lénaïc Chizat
CNRS, Université Paris-Sud
Orsay, France
lenaic.chizat@u-psud.fr

Edouard Oyallon
CentraleSupelec, INRIA
Gif-sur-Yvette, France
edouard.oyallon@centralesupelec.fr

Francis Bach
INRIA, ENS, PSL Research University
Paris, France
francis.bach@inria.fr

Jan 2020

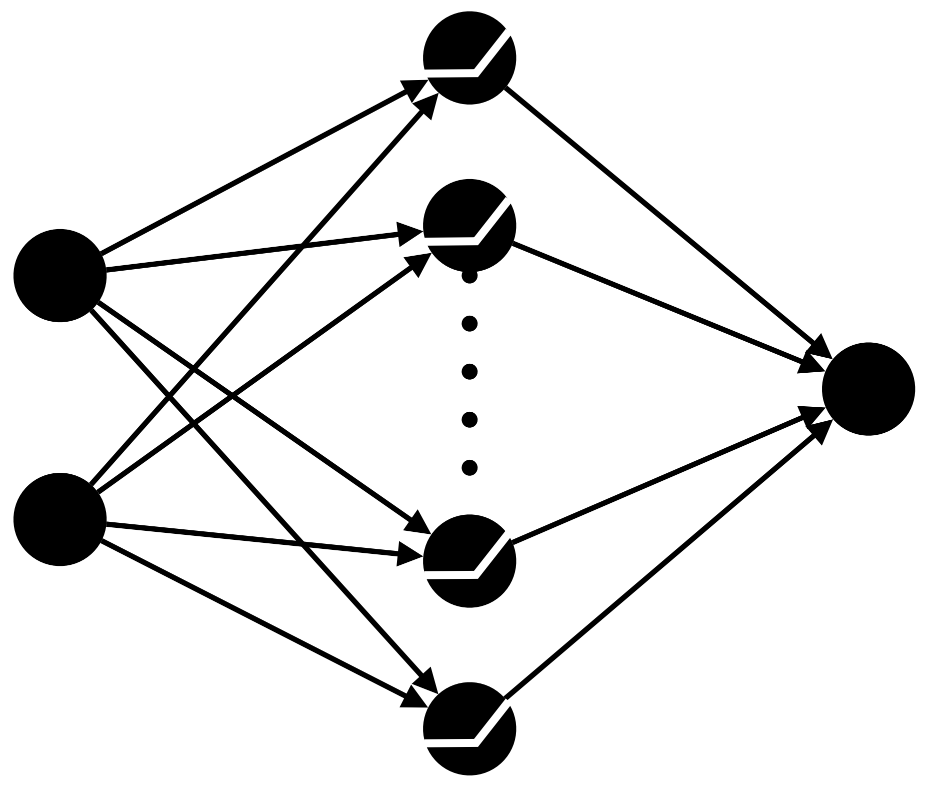
Chizat et al. (2019)



*sufficiently wide for a finite training dataset and $f(x;\theta_0) = 0$

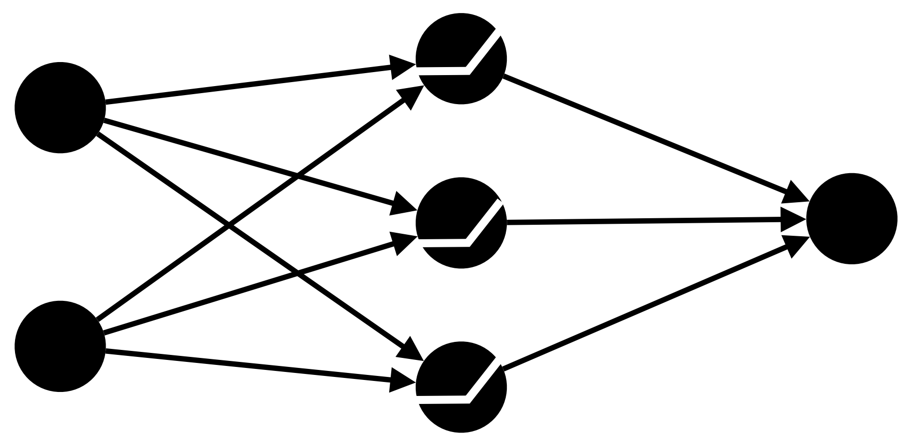
A revealing experiment inspired by Chizat et al. (2019)

Wide two-layer student ReLU Network

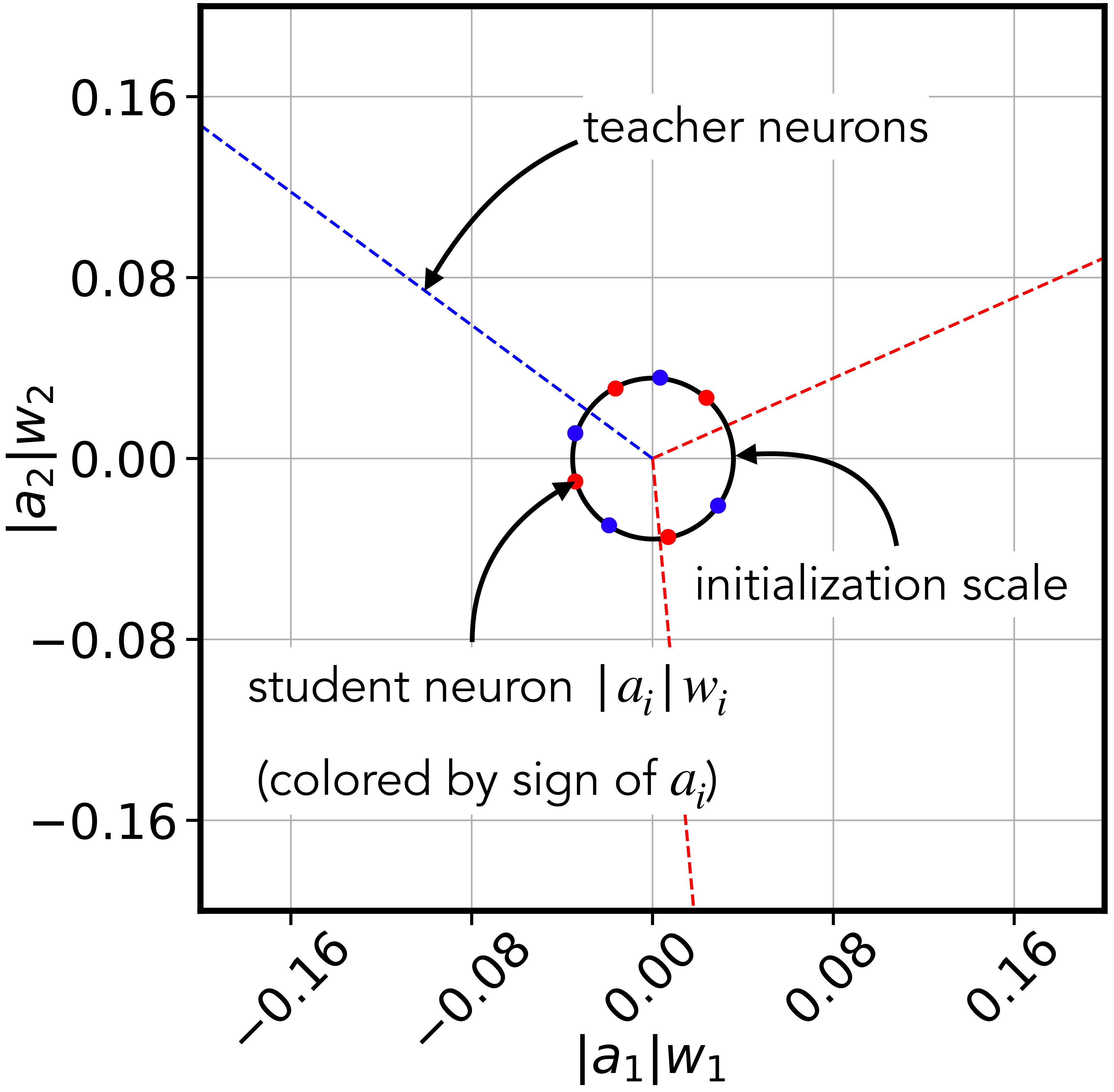


$$f(x;\theta) = \sum_{k=1}^{h} a_k \max(w_k^\intercal x, 0)$$

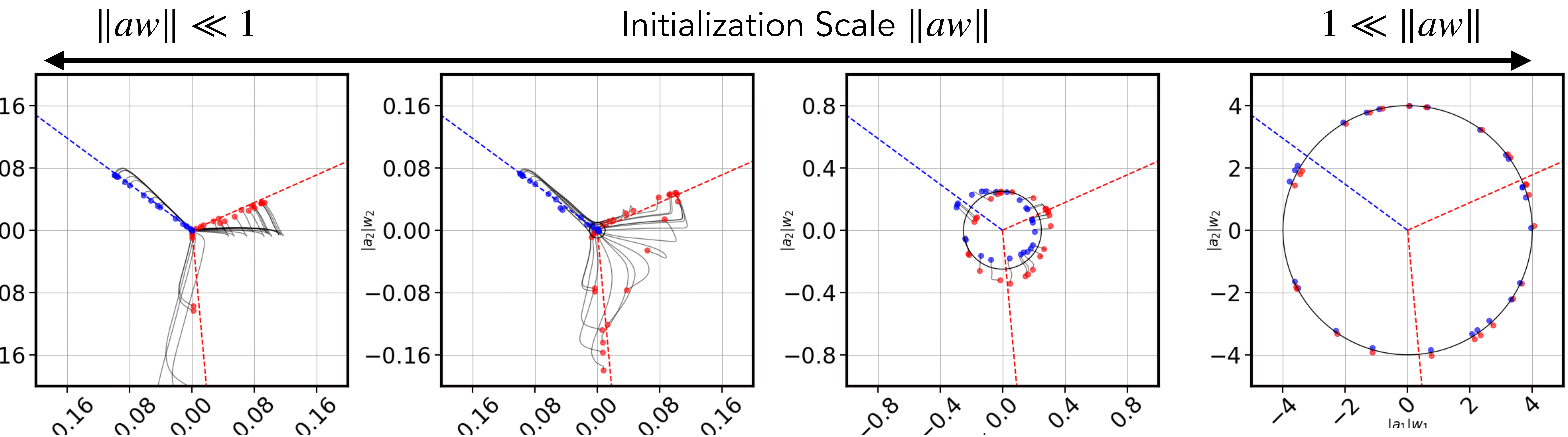
Narrow two-layer teacher ReLU Network



$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

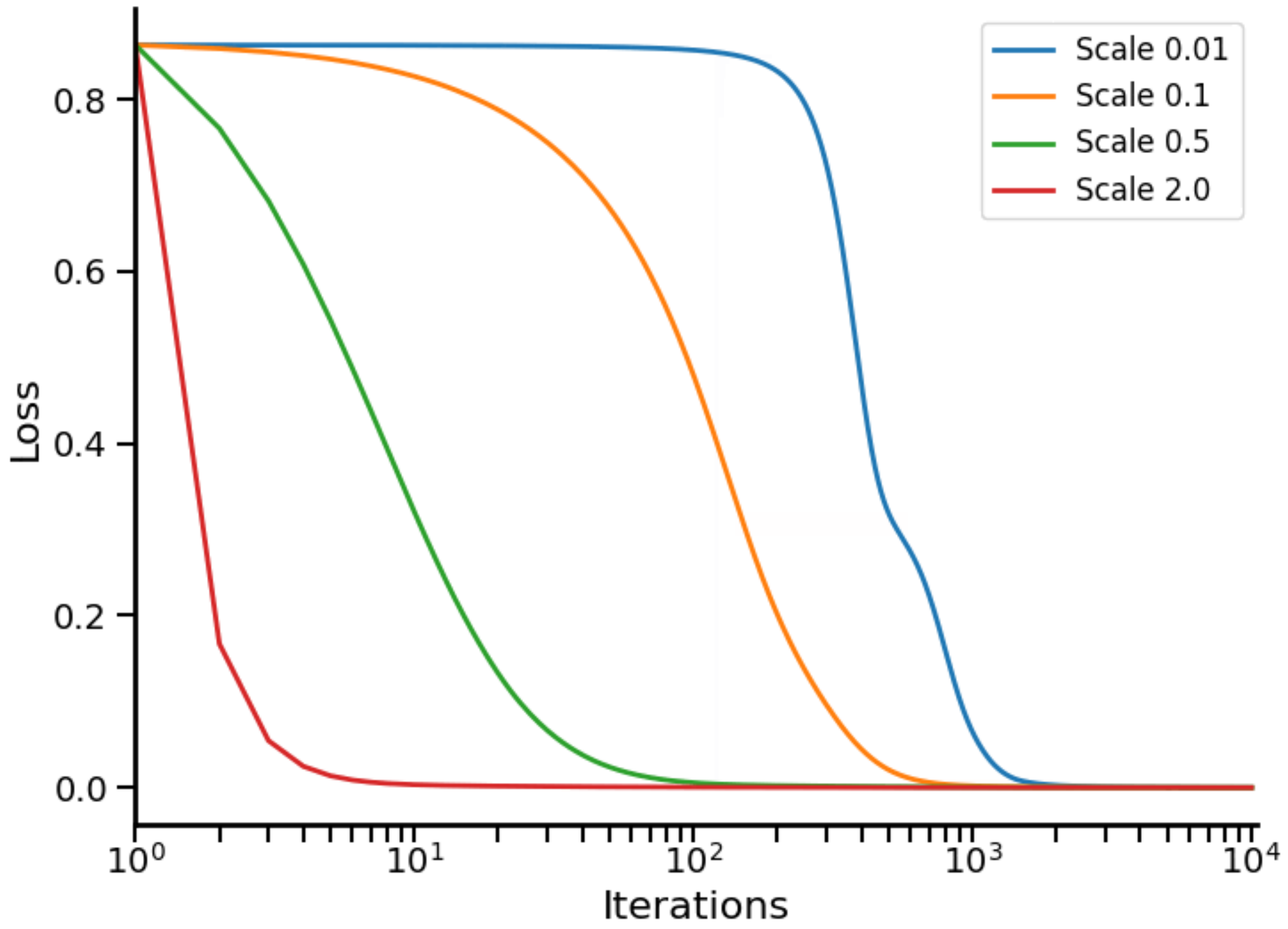


Feature learning = student neurons aligning to teacher

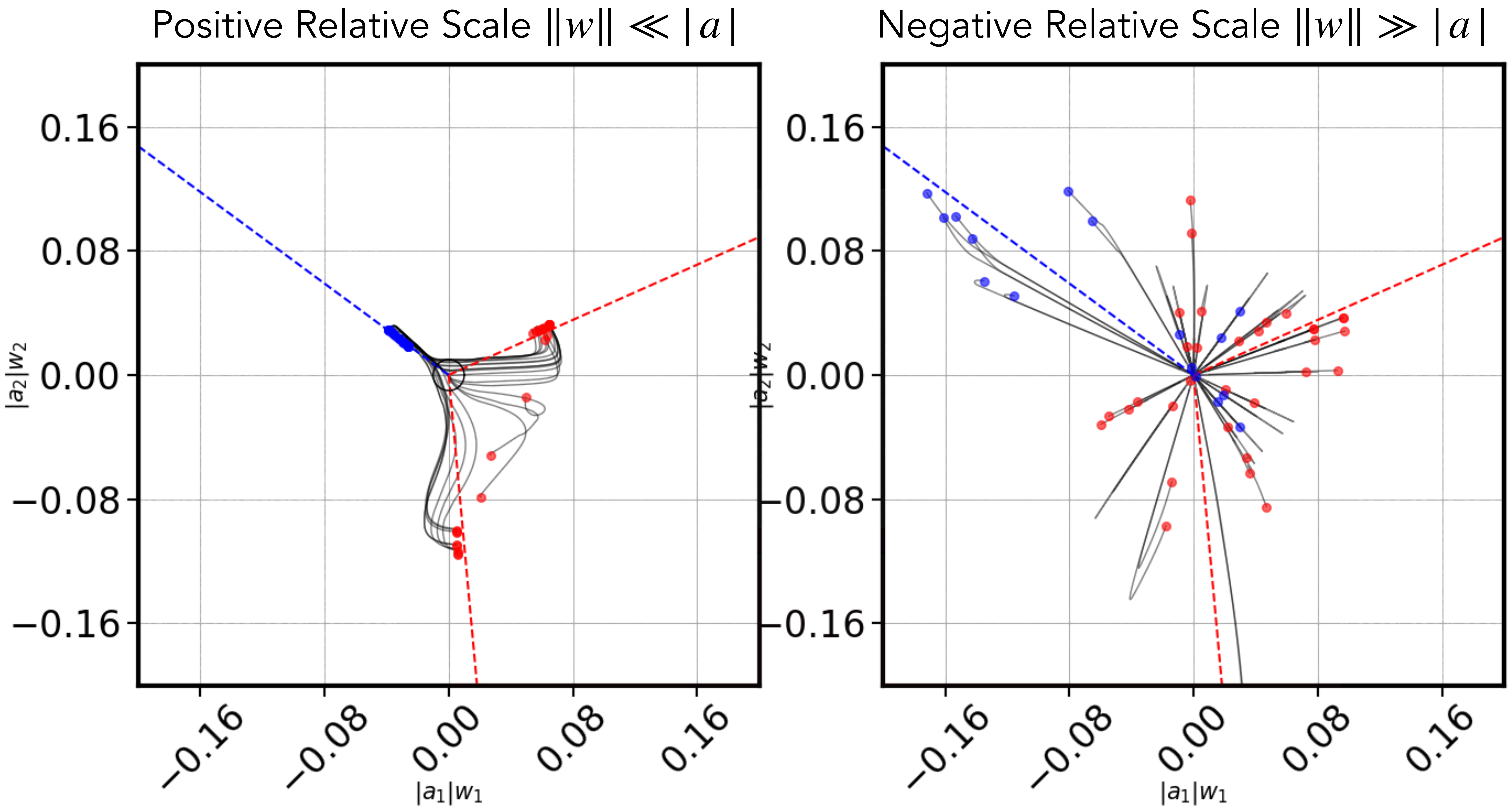


1. *At large-scale initialization, the student moves very little to quickly fit the data.*

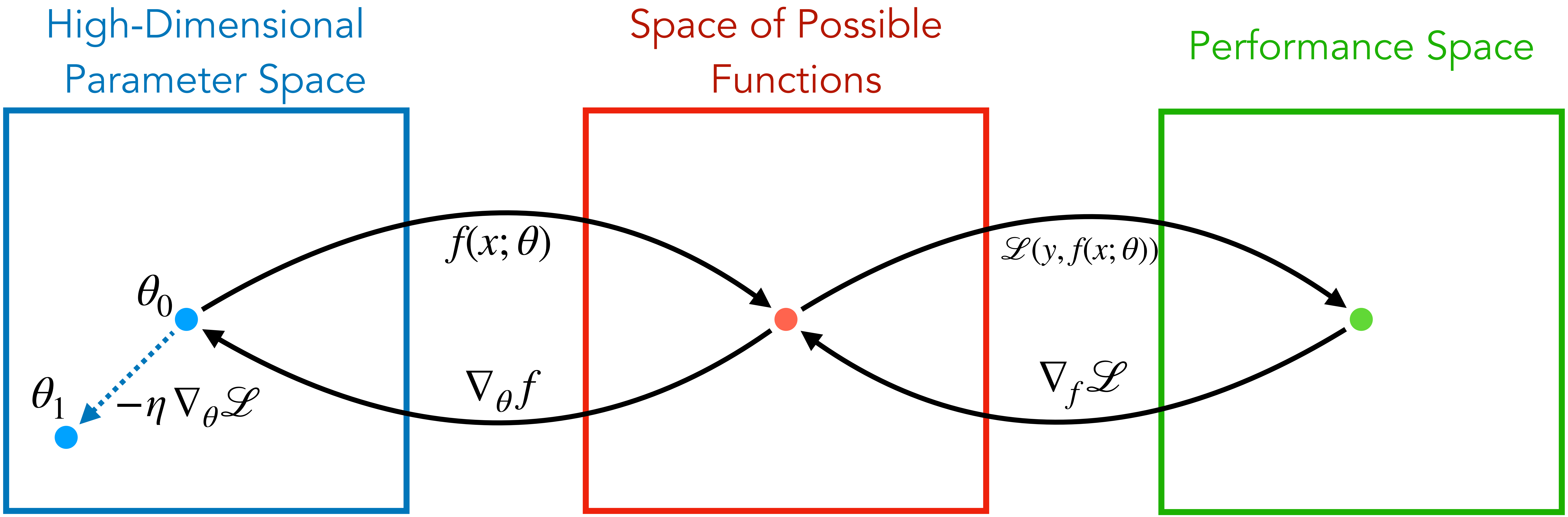
2. *At small-scale initialization, the student aligns to teacher, during which the loss has long plateaus.*



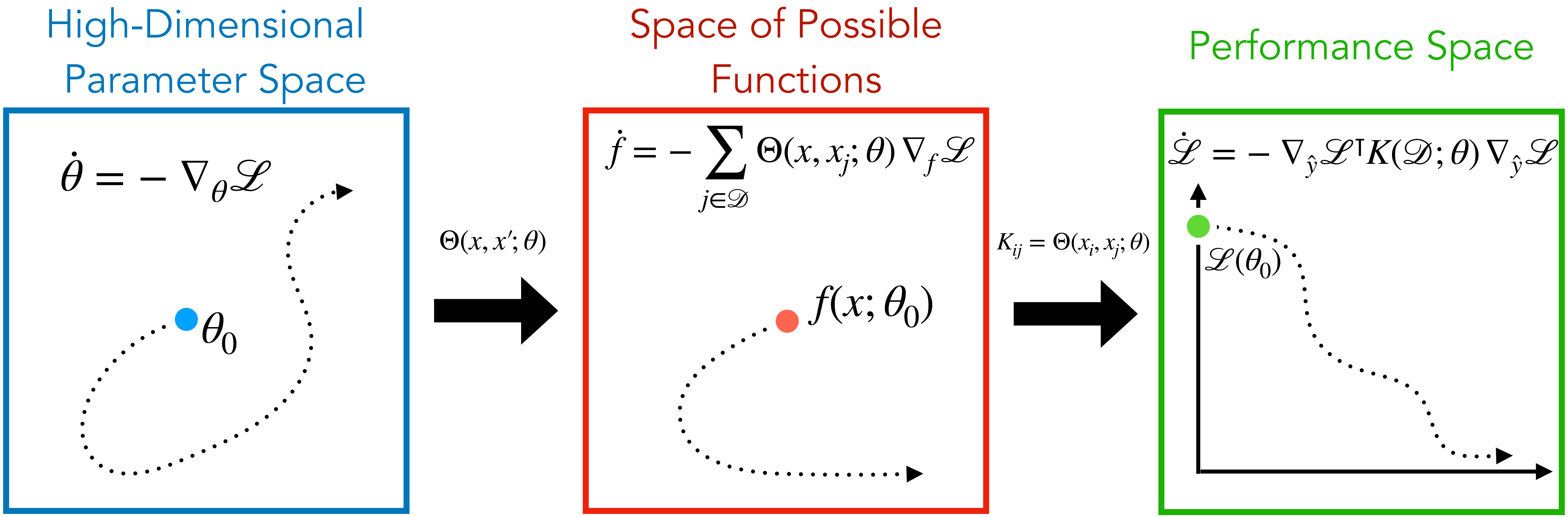
The relative scale also influences feature learning



The different spaces where learning occurs



The different spaces where learning occurs



Neural Tangent Kernel

$$\Theta(x, x'; \theta) = \langle \nabla_\theta f(x), \nabla_\theta f(x') \rangle$$

The NTK quantifies how one gradient step with data point x' affects the evolution of the networks's output evaluated at another data point x.

If there is feature learning, then the NTK must change!

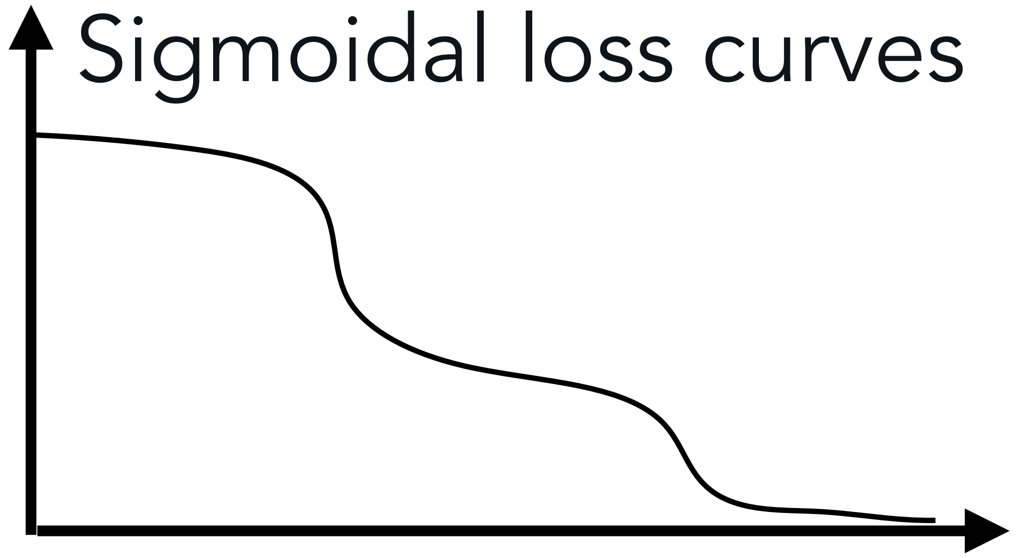
<u>When</u> does feature learning emerge?

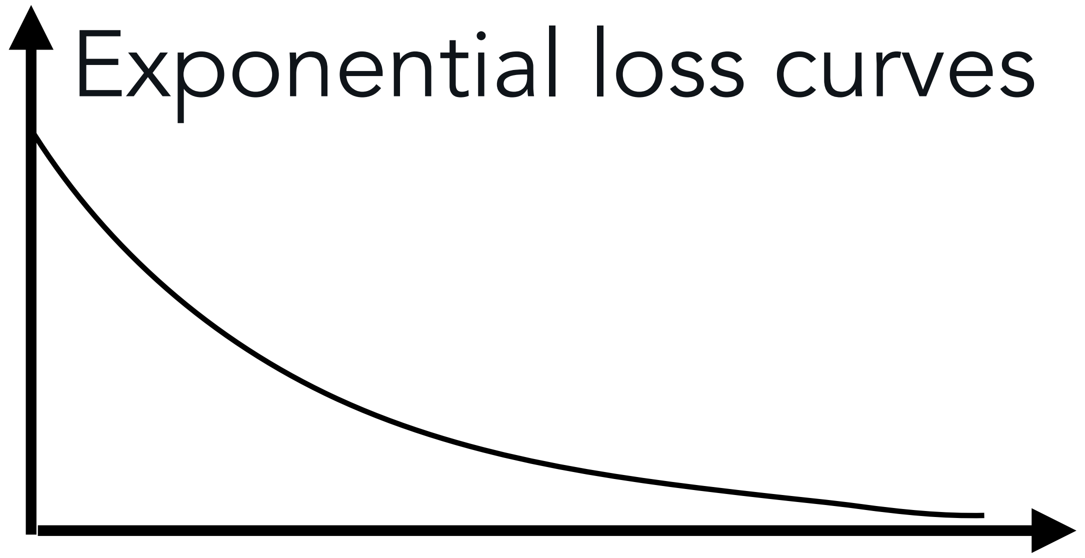
Feature Learning
(Rich Learning)

Kernel Learning
(Lazy Learning)

Feature learning Spectrum

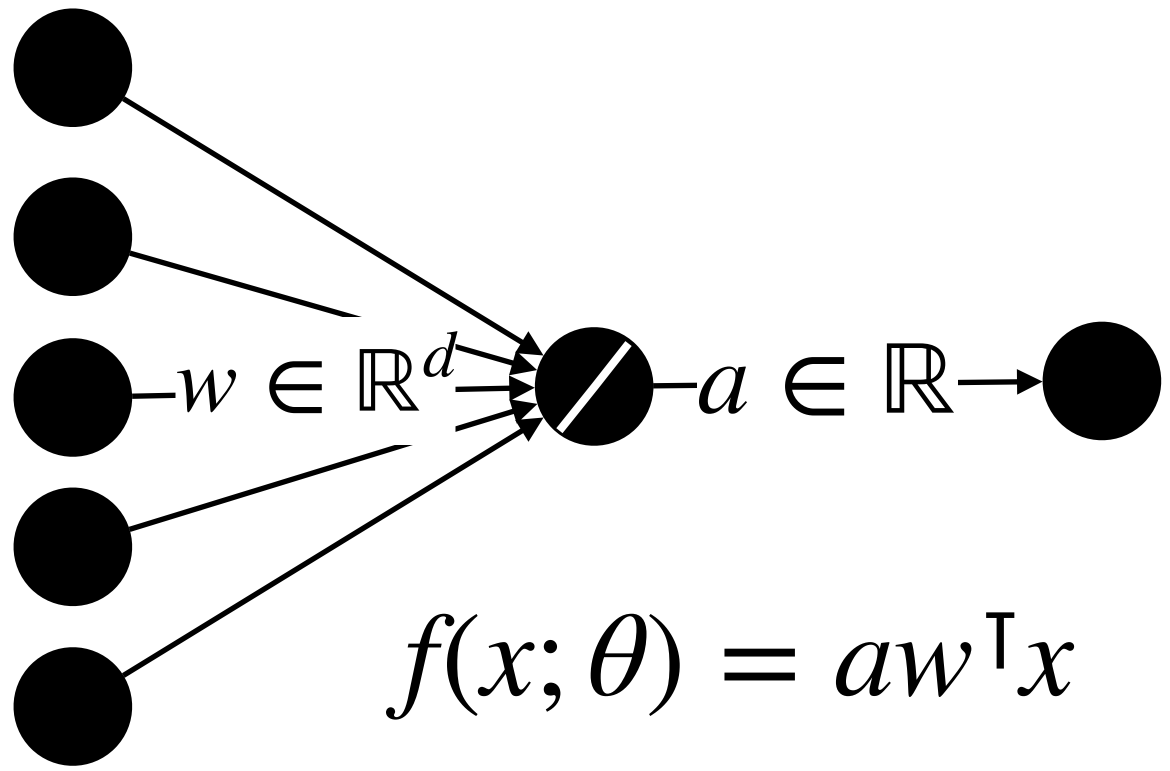
Feature Learning	Kernel Learning
The NTK **does** evolve	The NTK **does not** evolve
Small-scale initialization	Large-scale initialization
Sigmoidal loss curves	Exponential loss curves





<u>Why</u> are small-scale initializations and sigmoidal loss curves related to the emergence of learning?

We will consider a minimal model — a single linear neuron



$$f(x;\theta) = a w^\intercal x$$

Linear networks — linear in input x, but nonlinear in parameters θ

Neural Networks, Vol. 2, pp. 53–58, 1989
0893-6080/89 $3.00 + .00
Printed in the USA. All rights reserved.
Copyright © 1989 Pergamon Press plc

ORIGINAL CONTRIBUTION

Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima

PIERRE BALDI AND KURT HORNIK*
University of California, San Diego

(*Received* 18 May 1988; *revised and accepted* 16 *August* 1988)

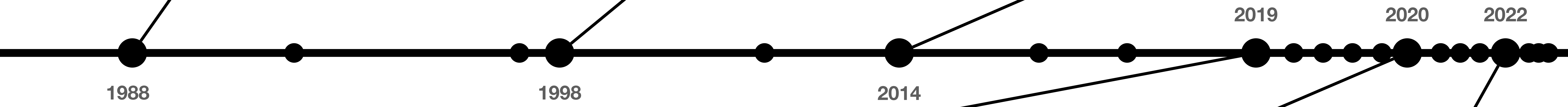
Submitted to ICONIP'98

EFFECT OF BATCH LEARNING IN MULTILAYER NEURAL NETWORKS

Kenji Fukumizu
Email:fuku@brain.riken.go.jp
Lab. for Information Synthesis, RIKEN Brain Science Institute
Hirosawa 2-1, Wako, Saitama, 351-0198, Japan

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks

Andrew M. Saxe (asaxe@stanford.edu)
Department of Electrical Engineering
James L. McClelland (mcclelland@stanford.edu)
Department of Psychology
Surya Ganguli (sganguli@stanford.edu)
Department of Applied Physics
Stanford University, Stanford, CA 94305 USA



Implicit Regularization in Deep Matrix Factorization

Sanjeev Arora
Princeton University and Institute for Advanced Study
arora@cs.princeton.edu
Nadav Cohen
Tel Aviv University
cohennadav@cs.tau.ac.il
Wei Hu
Princeton University
huwei@cs.princeton.edu
Yuping Luo
Princeton University
yupingl@cs.princeton.edu

Published as a conference paper at ICLR 2020

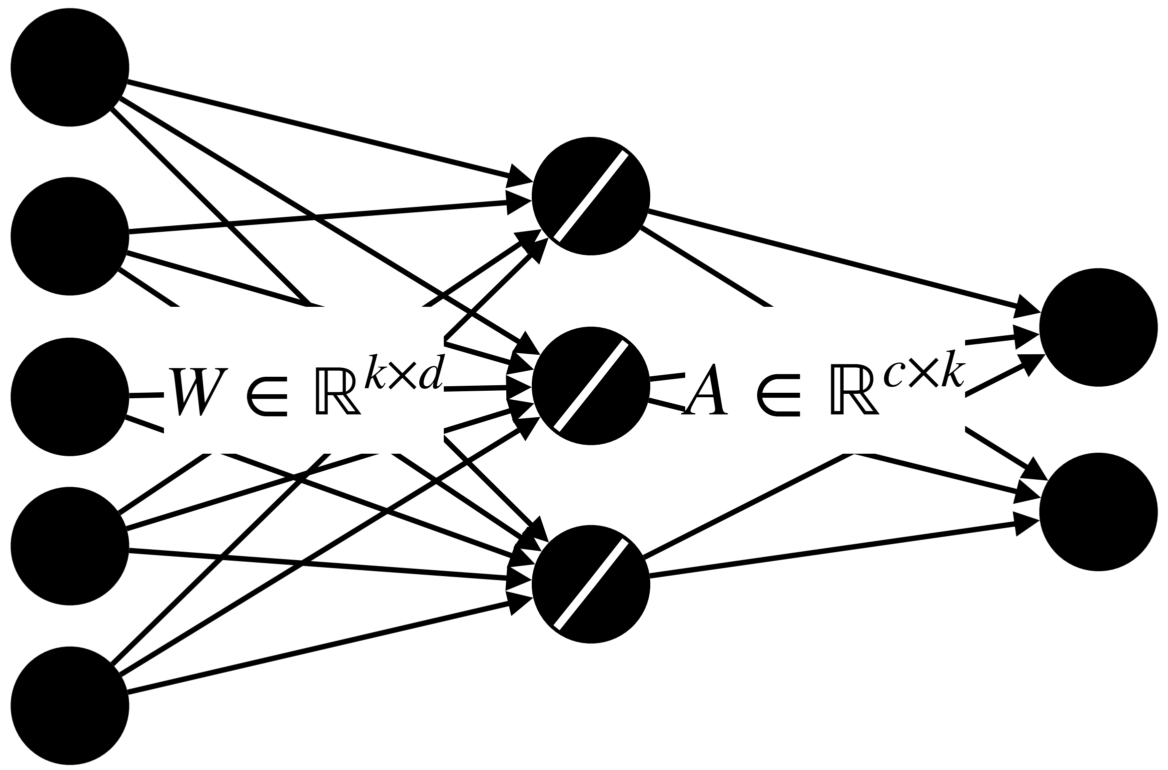
THE IMPLICIT BIAS OF DEPTH: HOW INCREMENTAL LEARNING DRIVES GENERALIZATION

Daniel Gissin, Shai Shalev-Shwartz, Amit Daniely
School of Computer Science
The Hebrew University
Jerusalem, Israel
{daniel.gissin,shais,amit.daniely}@mail.huji.ac.il

Saddle-to-Saddle Dynamics in Deep Linear Networks: Small Initialization Training, Symmetry, and Sparsity

Arthur Jacot [1] François Ged [1] Berfin Şimşek [1] Clément Hongler [1] Franck Gabriel [1]

A rescale symmetry — something all linear network analysis share



$$f(x;\theta) = AWx$$

$f(x;\theta)$ is symmetric under the action of $GL_k(\mathbb{R})$ defined by:

$$(A, W) \mapsto (AG^{-1}, GW)$$

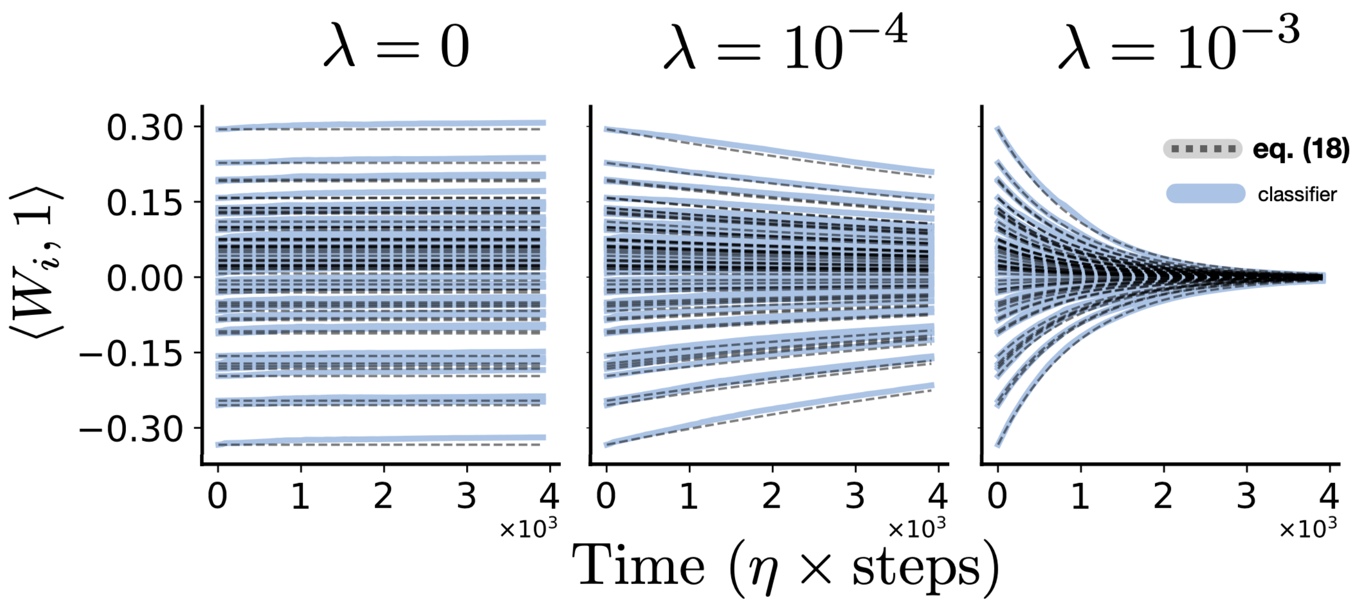
Implications of this "*rescale*" symmetry:

- Minima of the loss are on smooth submanifolds
- Gradient property $A^\intercal \dfrac{\partial \mathscr{L}}{\partial A} = \dfrac{\partial \mathscr{L}}{\partial W} W^\intercal$.
- Conserved quantity under gradient flow:

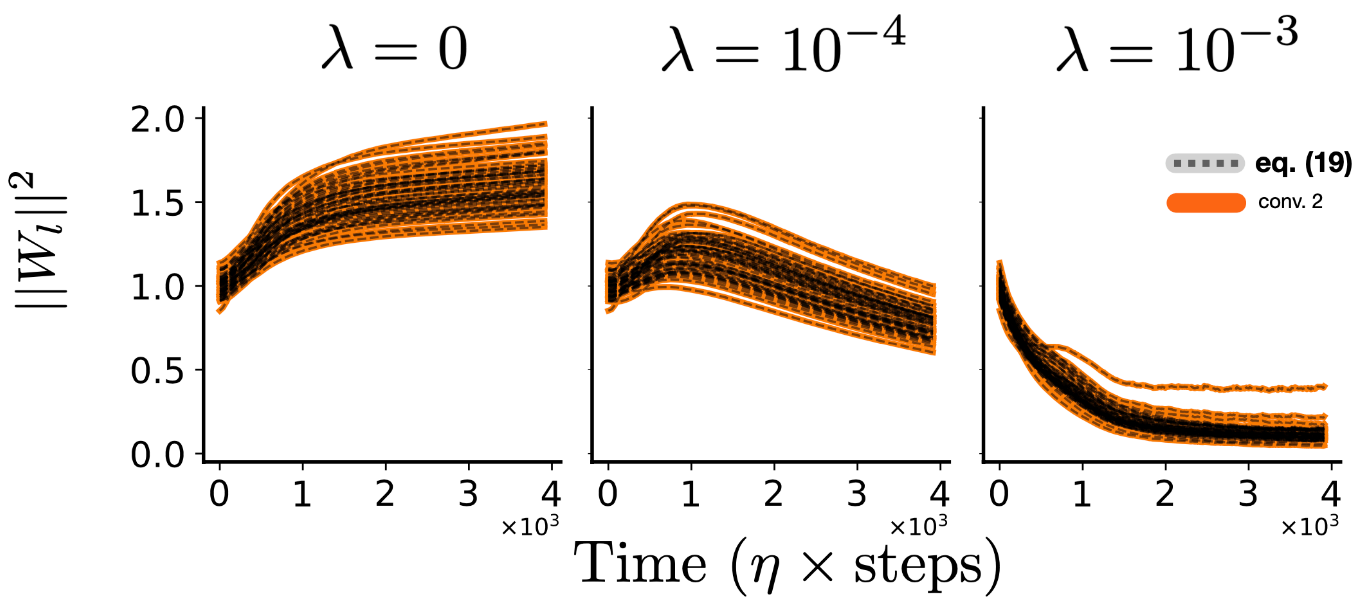
$$\frac{d}{dt}\left(A^\intercal A - WW^\intercal\right) = 0$$

“Conserved” quantities exist throughout deep learning

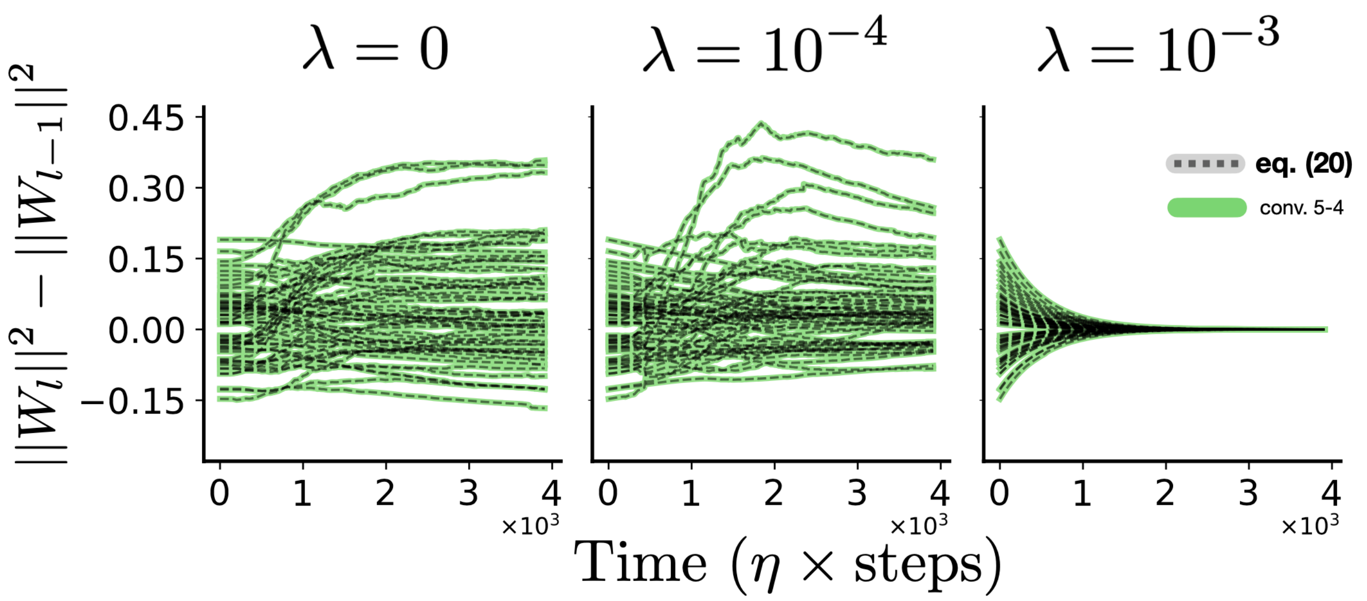
Translation symmetry
in weights before softmax



Scale symmetry
in weights before normalization



Rescale symmetry
between layers with ReLU



Emmy Noether (1882 - 1935)

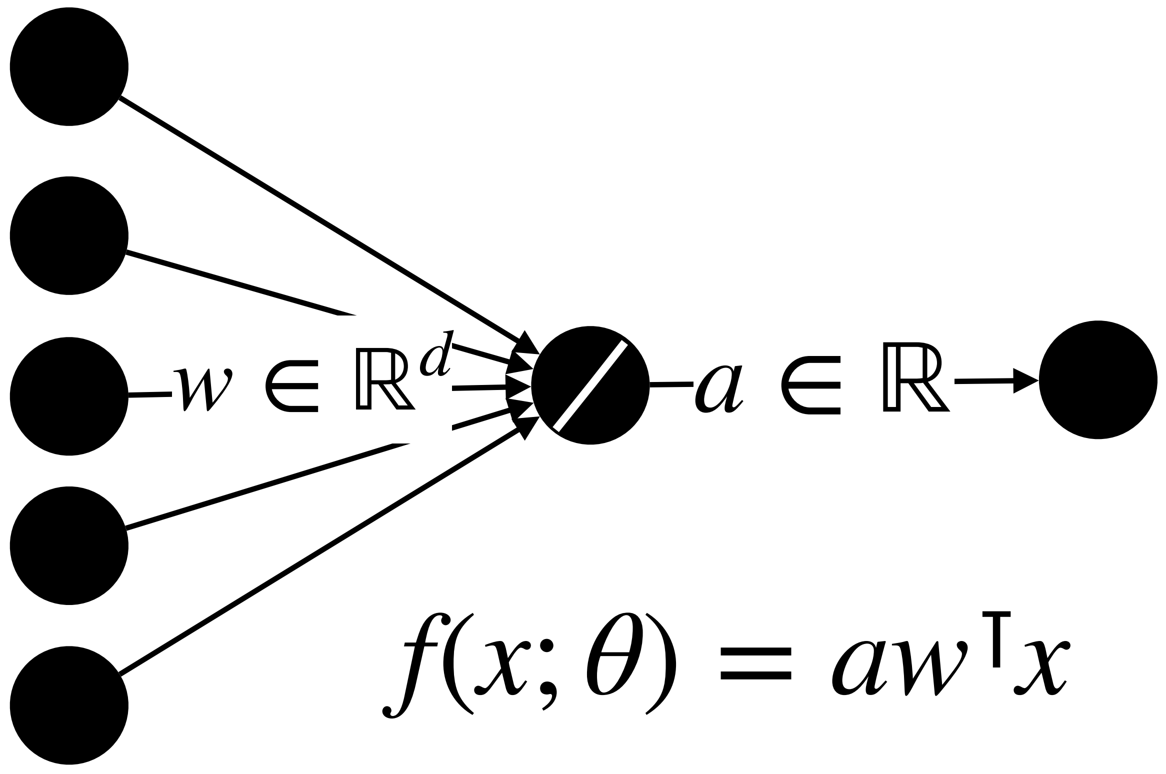
A version of Noether’s Theorem: Every continuous symmetry* of a network architecture has a corresponding conserved quantity under gradient flow. Projecting the gradient flow dynamics onto the vector field generating symmetry gives an ODE, whose solution is a conservation law.

$$\frac{d}{dt}\left\langle \theta, \partial_\alpha \psi \right\rangle = 0$$

*satisfying a mild assumption

This analogy is explored in Kunin et al. 2020 and Tanaka and Kunin, 2021.

Minimal model that transitions between rich & lazy



$$f(x;\theta) = aw^\intercal x$$

A single neuron trained on MSE $\mathscr{L} = \dfrac{1}{2}\|y - aw^\intercal X^\intercal\|^2$ by gradient flow:

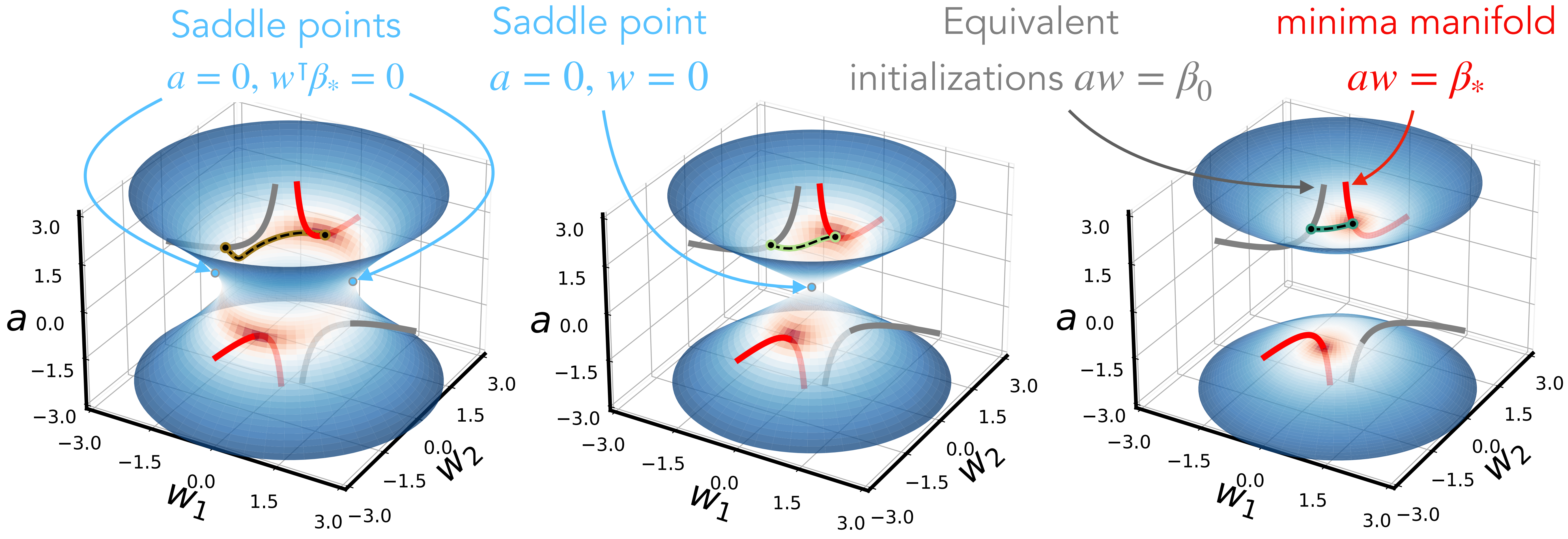
$$\dot{a} = -\, w^\intercal \left(X^\intercal X a w - X^\intercal y\right), \qquad a(0) = a_0,$$

$$\dot{w} = -\, a \left(X^\intercal X a w - X^\intercal y\right), \qquad w(0) = w_0\,.$$

The initialization determines

$$\delta = a_0^2 - \|w_0\|^2,$$

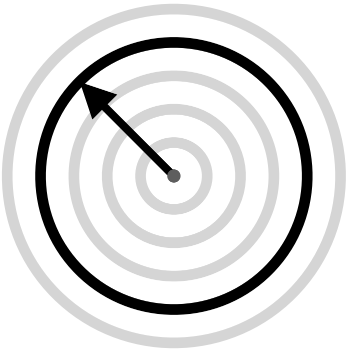
which constrains trajectories.



Downstream $\delta < 0$ **Balanced $\delta = 0$** **Upstream $\delta > 0$**

We can derive exact solutions when we assume whitened input $X^\intercal X = \mathbf{I}_d$.

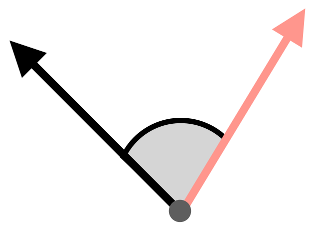
Norm $\mu = a\|w\|$



$$\dot{\mu} = \sqrt{\delta^2 + 4\mu^2}(\phi\|\beta_*\| - \mu)$$

Bernoulli ODE

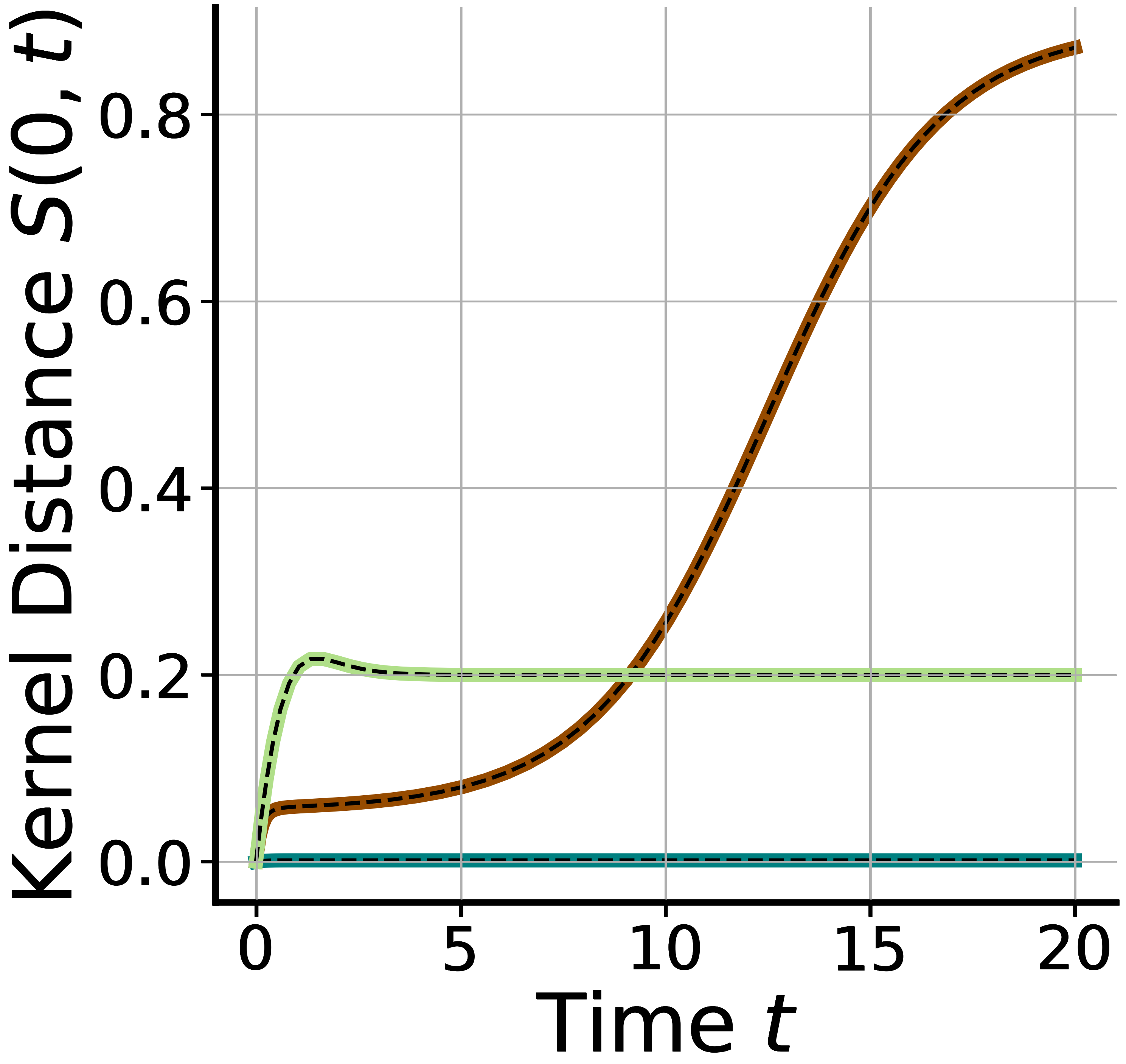
Cosine Angle $\phi = \dfrac{w^\intercal \beta_*}{\|w\|\|\beta_*\|}$



$$\dot{\phi} = \frac{2\mu\|\beta_*\|}{\sqrt{\delta^2 + 4\mu^2} - \delta}(1 - \phi^2)$$

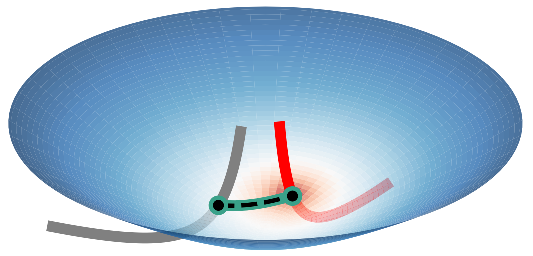
Riccati ODE

Limitation — all initializations converge to the same solution.

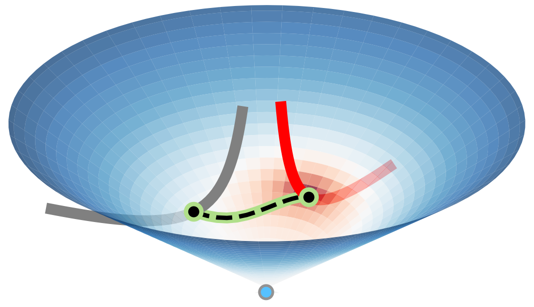


Derive dynamics of NTK $K = X(a^2\mathbf{I}_d + ww^\intercal)X^\intercal$:

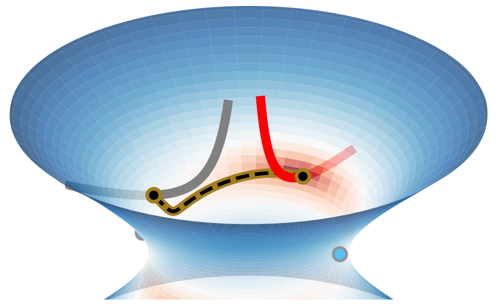
Lazy — when $\delta \gg 0$, essentially initialized at the minima manifold



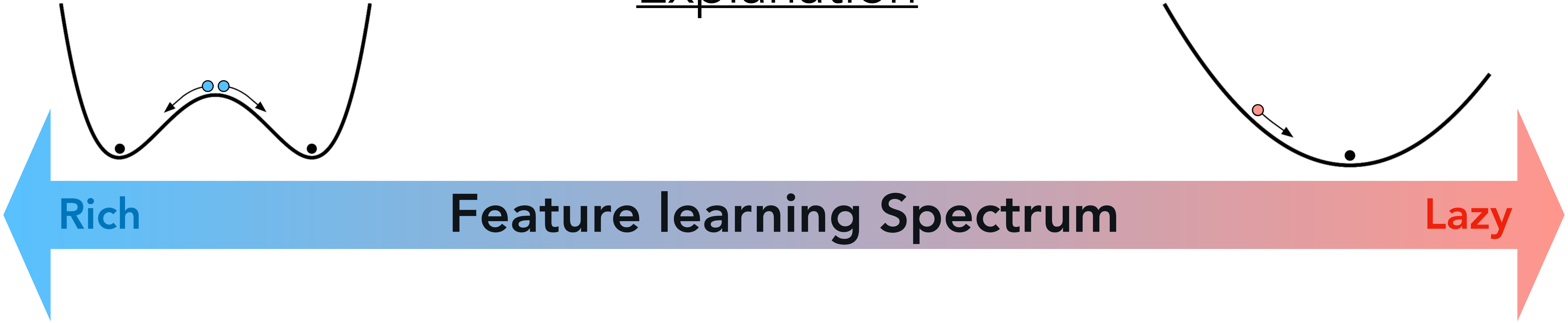
Rich — when $\delta = 0$ and small norm, then initialized near the saddle at origin.



Delayed rich — when $\delta \ll 0$ and $\phi(0) \approx 0$, the trajectory goes near the saddle $w^\intercal \beta_* = a = 0$.



Parameter Space Explanation



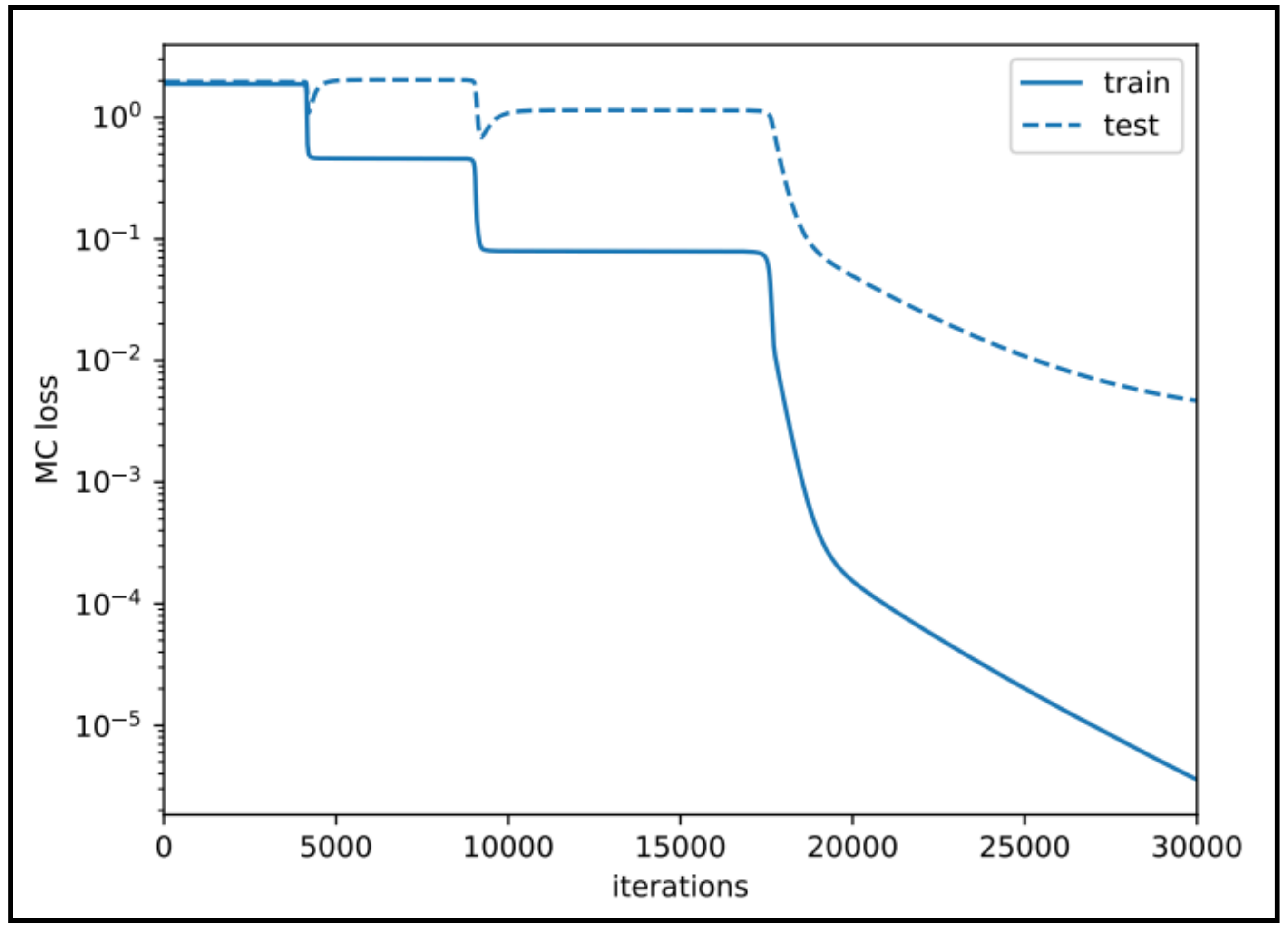
Parameter Space: Saddle-to-Saddle Dynamics

Jacot et al., 2022, study deep linear networks $f(x;\theta) = W_L \dots W_1 x$ with width w initialized with variance $\sigma^2 = w^{-\gamma}$. They observe a phase transition in γ as $w \to \infty$:

$\gamma > 1$	$\gamma = 1$	$1 - 1/L \leq \gamma < 1$
Saddle-to-Saddle	Mean Field Limit	NTK Regime
(θ_0 is close to saddle and far from minima)		(θ_0 is close to minima and far from saddle)

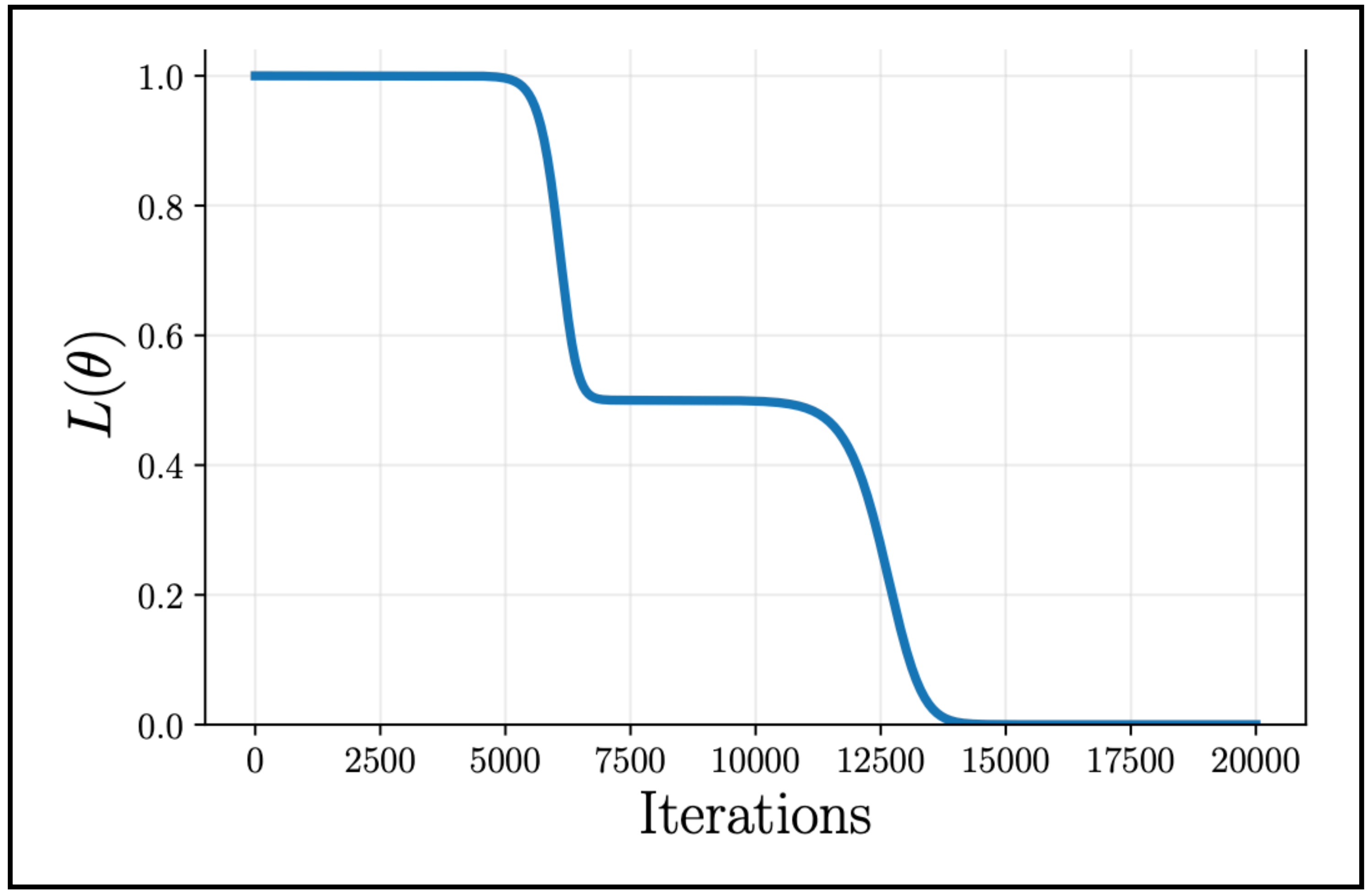
Saddle-to-Saddle conjecture: In the vanishing initialization limit, gradient descent visits a sequence of saddles, each corresponding to linear maps of increasing rank, until reaching a sparse global minimum.

Linear Networks



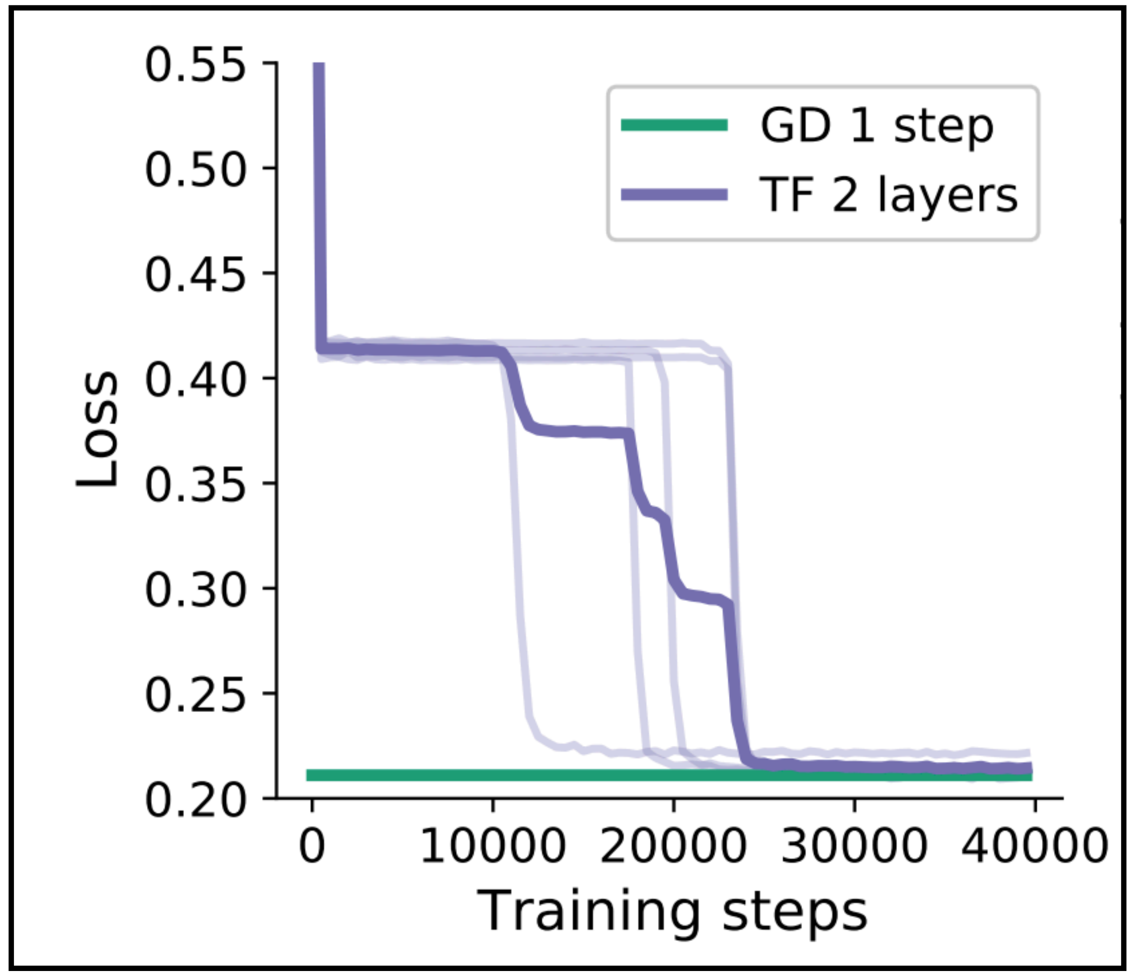
Jacot et al., 2022

Two-layer ReLU Networks



Boursier et al., 2022

Attention-only Transformer



von Oswald et al., 2023

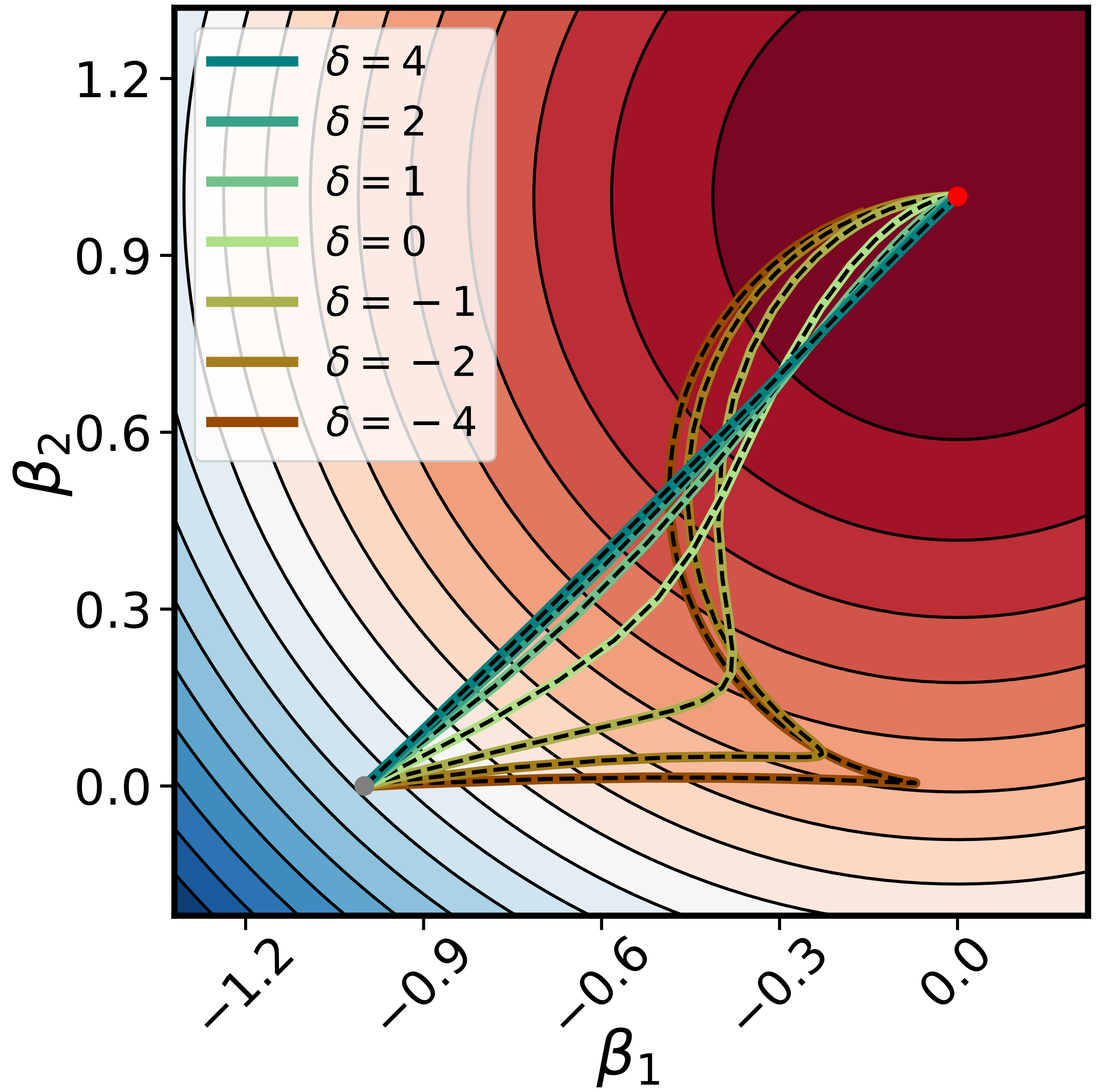
A function space perspective of our minimal model

We can derive a self-consistent equation for the dynamics of $\beta = aw$,

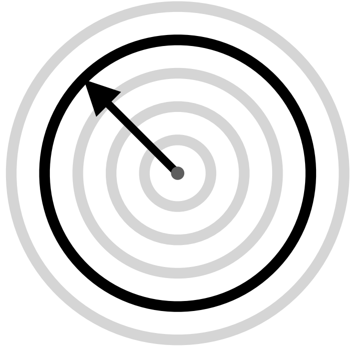
$$\dot{\beta} = -\underbrace{\left(\frac{\sqrt{\delta^2 + 4\|\beta\|^2} + \delta}{2}\mathbf{I}_d + \frac{\sqrt{\delta^2 + 4\|\beta\|^2} - \delta}{2}\frac{\beta\beta^\intercal}{\|\beta\|^2}\right)}_{M}\nabla_\beta \mathscr{L},$$

which is a preconditioned gradient flow. The NTK matrix $K = XMX^\intercal$.

Strength — this holds even when $X^\intercal X$ is low-rank.

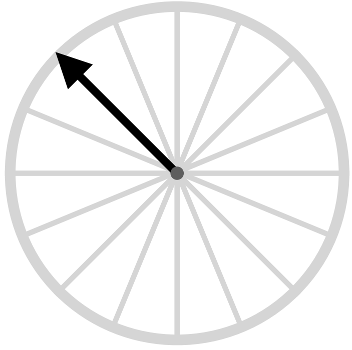


Radial $\|\beta\|$



$$\|\dot{\beta}\| = -\sqrt{\delta^2 + 4\|\beta\|^2}\langle\hat{\beta}, \nabla_\beta \mathscr{L}\rangle$$

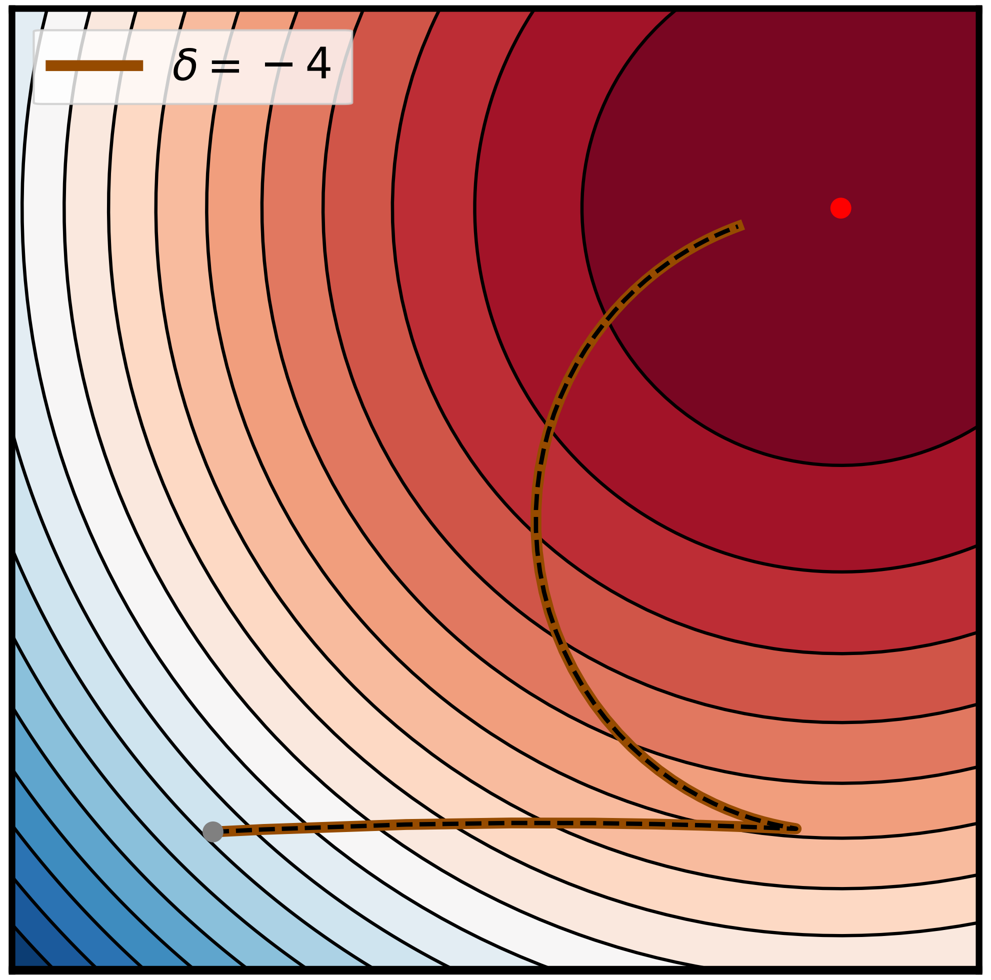
Directional $\hat{\beta}$



$$\dot{\hat{\beta}} = -\frac{\sqrt{\delta^2 + 4\|\beta\|^2} + \delta}{2\|\beta\|}\mathbf{P}_\beta^\perp \nabla_\beta \mathscr{L}$$

A separation of the timescales in dynamics

	$\delta \ll 0$	$\delta = 0$	$\delta \gg 0$
$\|\dot{\beta}\|$	Fast	$\propto \|\beta\|$	Fast
$\dot{\hat{\beta}}$	Slow	$O(1)$	Fast



Radial dominated dynamics

Lazy — when $\delta \gg 0$, $M \approx \delta \mathbf{I}_d$, akin to linear regression.

Directional dominated dynamics

Rich — when $\delta = 0$, $M = \sqrt{\eta_a \eta_w} \|\beta\| (\mathbf{I}_d + \frac{\beta\beta^\intercal}{\|\beta\|^2})$, akin to to silent alignment (Atanasov et al. 2021).

Delayed rich — when $\delta \ll 0$, $M \approx |\delta| \frac{\beta\beta^\intercal}{\|\beta\|^2}$, projected gradient descent — initially lazy followed by rich

Among the many interpolating solutions which one do we converge to?

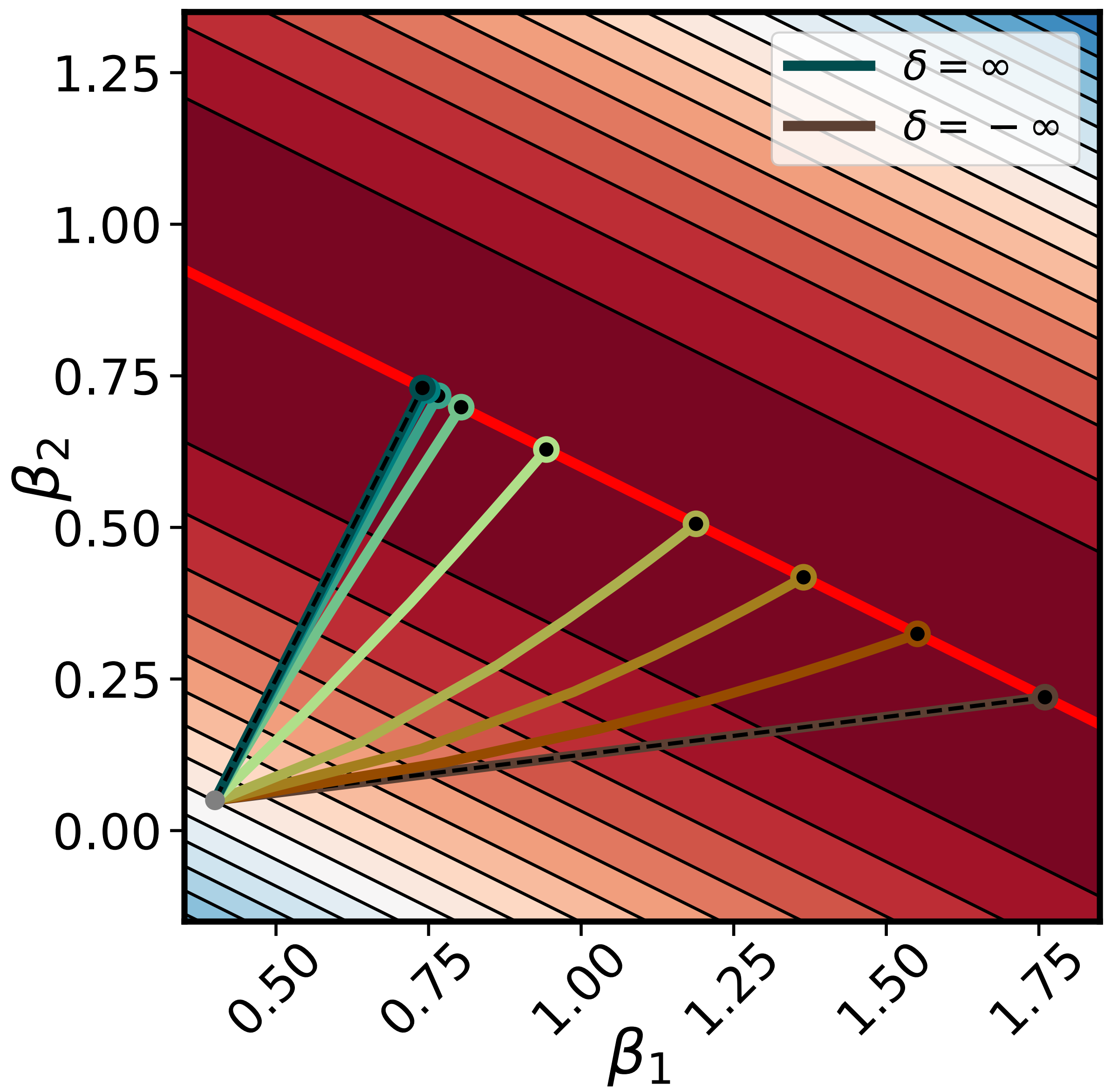
Theorem 3.1 (Extending Theorem 2 in Azulay et al. [9]). *For a single hidden neuron linear network, for any $\delta \in \mathbb{R}$, and initialization β_0 such that $\beta(t) \neq 0$ for all $t \geq 0$, if the gradient flow solution $\beta(\infty)$ satisfies $X\beta(\infty) = y$, then,*

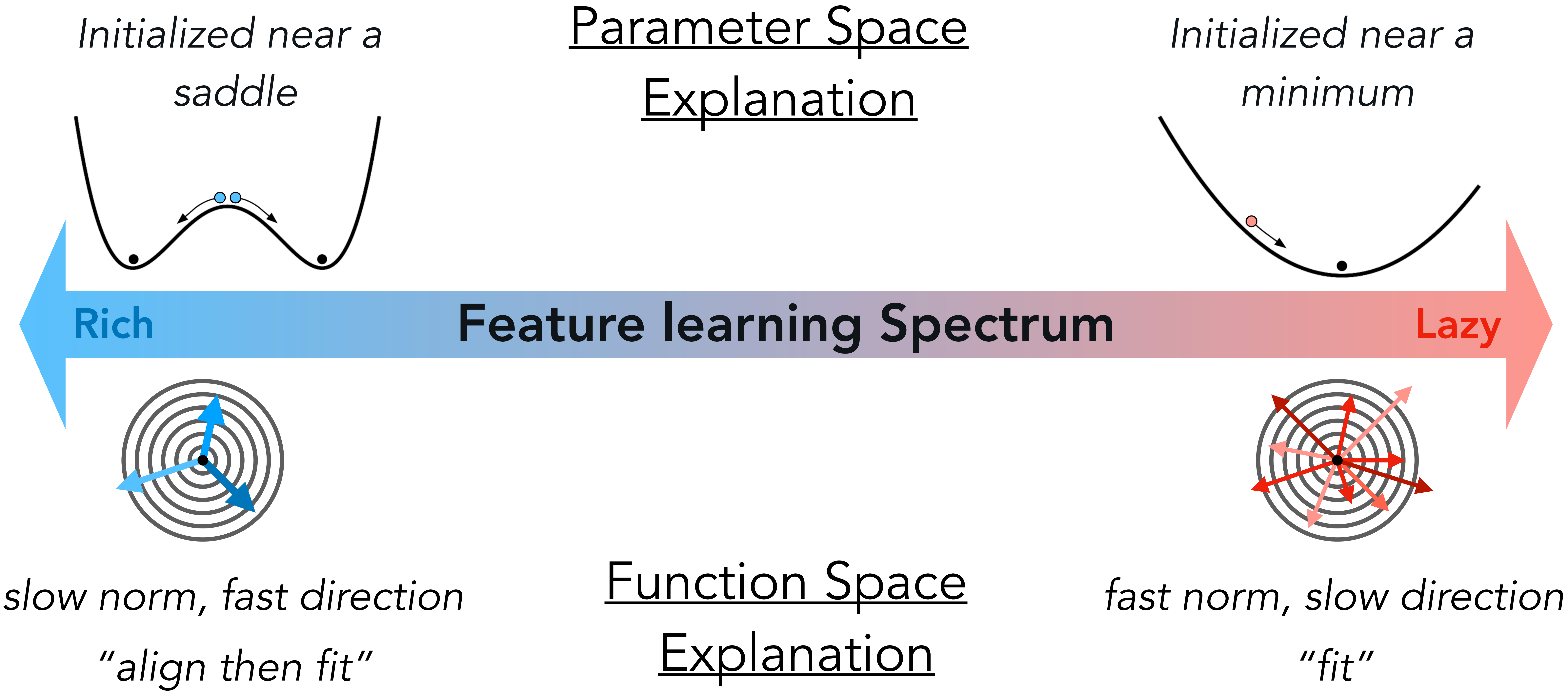
$$\beta(\infty) = \arg\min_{\beta \in \mathbb{R}^d} \Psi_\delta(\beta) - \psi_\delta \frac{\beta_0}{\|\beta_0\|}^\intercal \beta \quad \text{s.t.} \quad X\beta = y \tag{5}$$

where $\Psi_\delta(\beta) = \frac{1}{3}\left(\sqrt{\delta^2 + 4\|\beta\|^2} - 2\delta\right)\sqrt{\sqrt{\delta^2 + 4\|\beta\|^2} + \delta}$ *and* $\psi_\delta = \sqrt{\sqrt{\delta^2 + 4\|\beta_0\|^2} - \delta}$.

encourage minimum norm

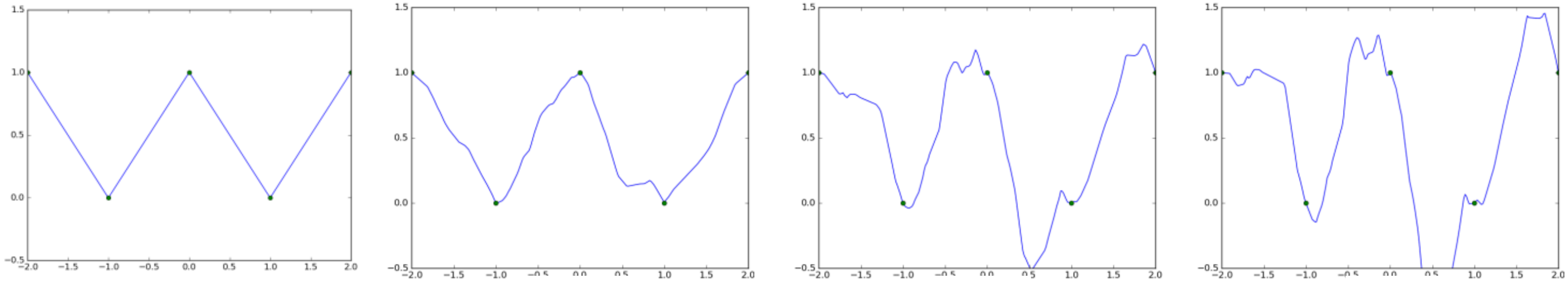
preserve initialization β_0




Initialized near a saddle
Parameter Space Explanation
Initialized near a minimum
Rich
Feature learning Spectrum
Lazy
slow norm, fast direction
"align then fit"
Function Space Explanation
fast norm, slow direction
"fit"

Function Space: Quantization in ReLU Networks

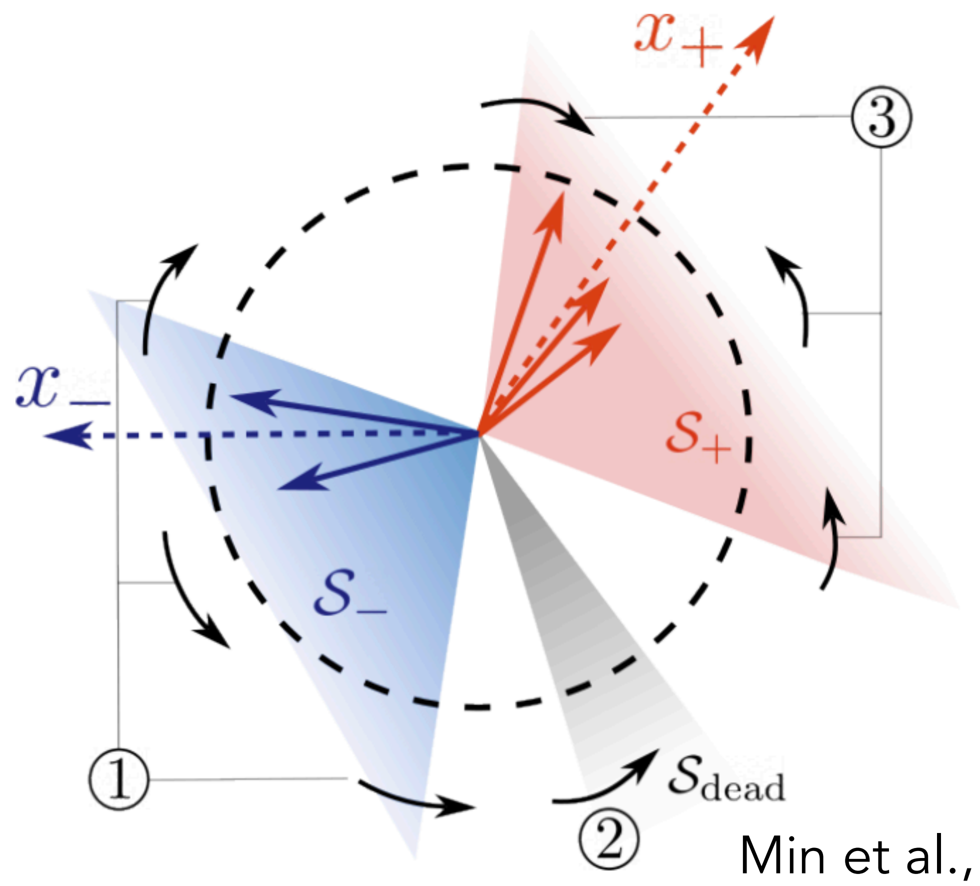
Maennel et al., 2018 observed that two-layer ReLU networks from small initializations ($\alpha \ll 1$), the first-layer weights concentrate along fixed directions determined by the training data, irrespective of network width.



$\alpha = 0.1$ $\alpha = 0.3$ $\alpha = 0.5$ $\alpha = 0.7$

"Empirical observation in 1d is that the ReLU kinks move while the training loss stays approximately constant, and they align with each other." Maennel et al., 2018

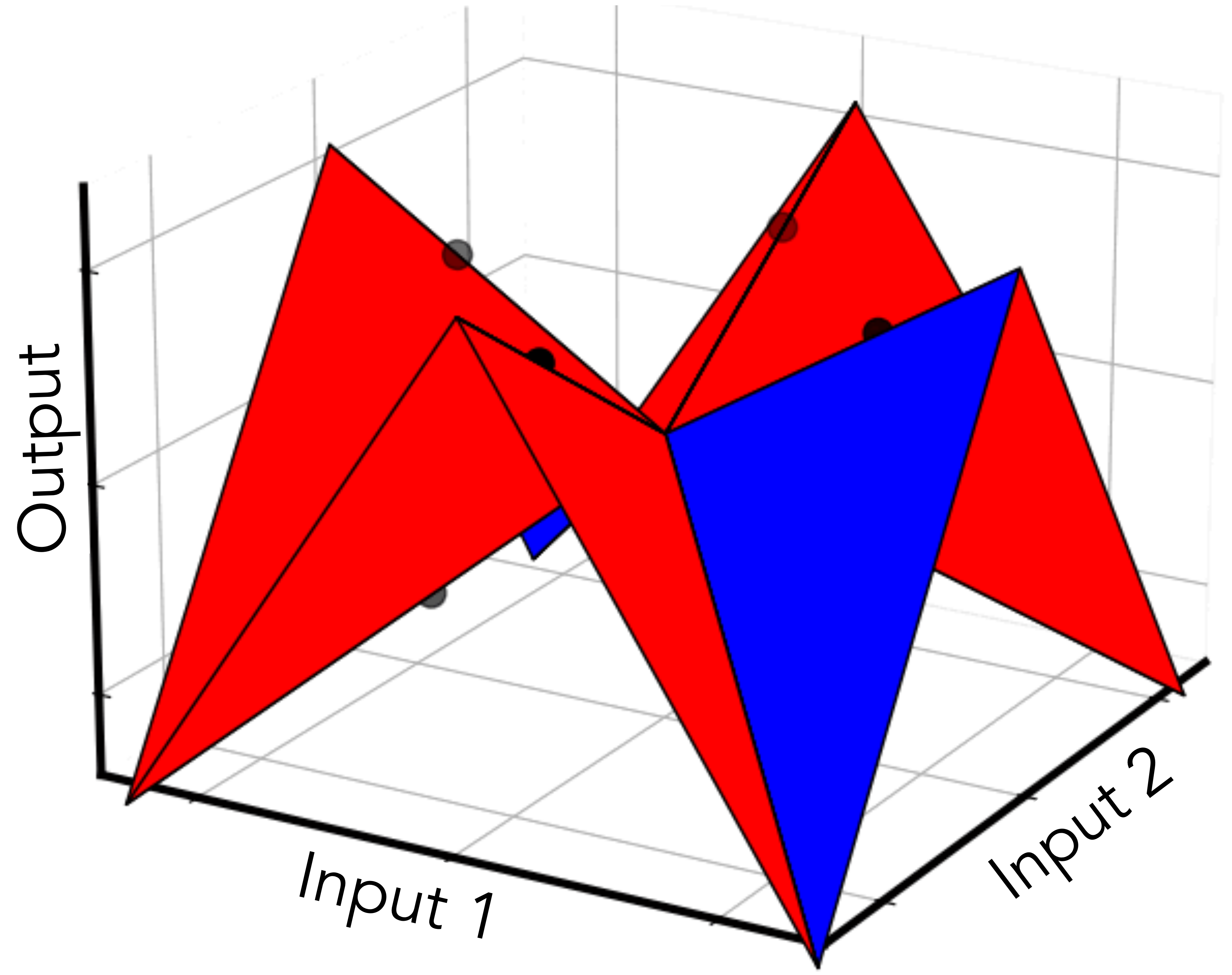
Many subsequent studies (Phuong and Lampert, 2020, Lyu et al., 2021, Boursier et al., 2022, Min et al., 2023, Wang and Ma, 2024) have observed distinct <u>alignment</u> and <u>fitting</u> phases.



Min et al., 2023

2D analog of Maennel et al. observation

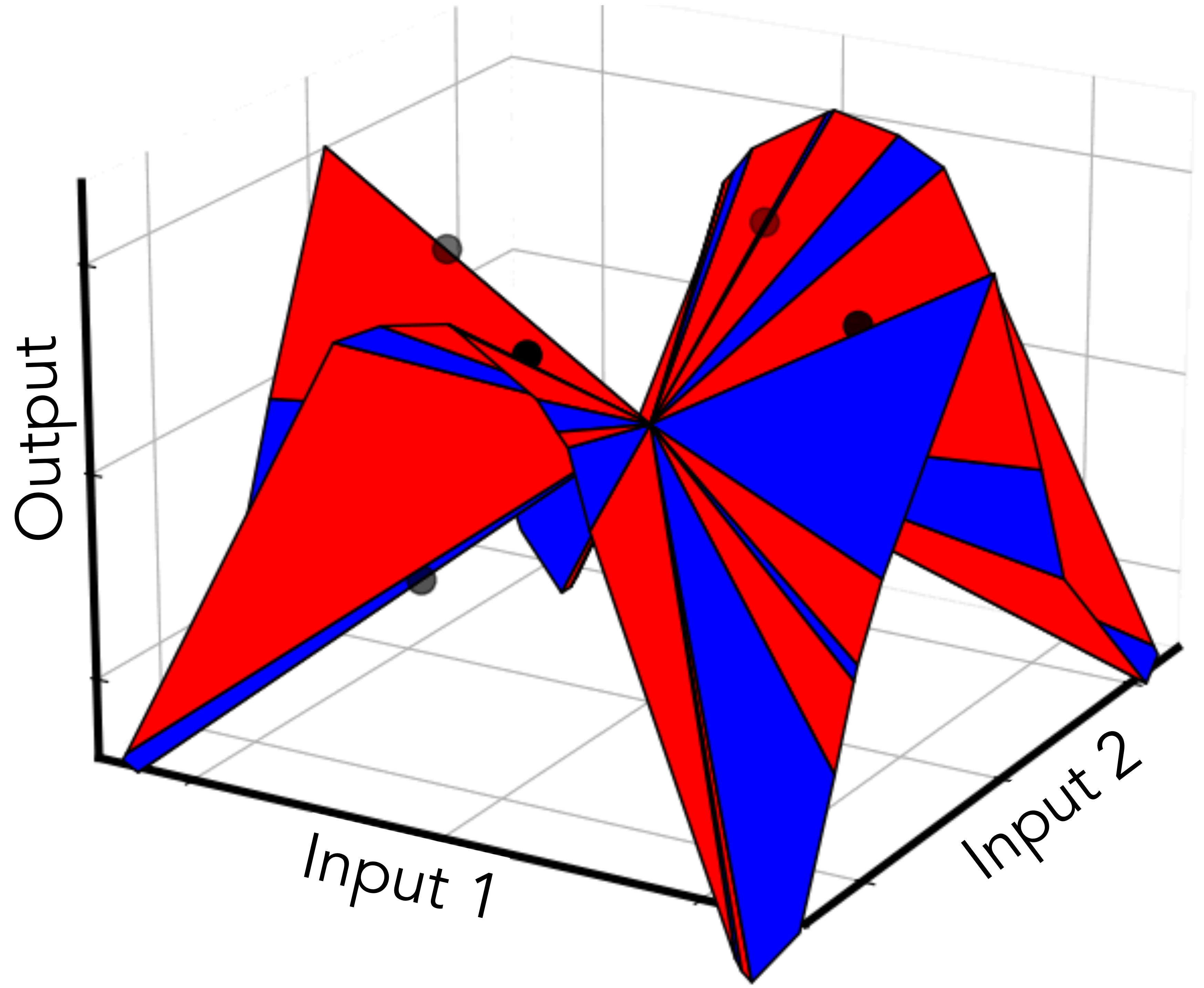
<u>Feature Learning</u>



norm is slow & direction is fast

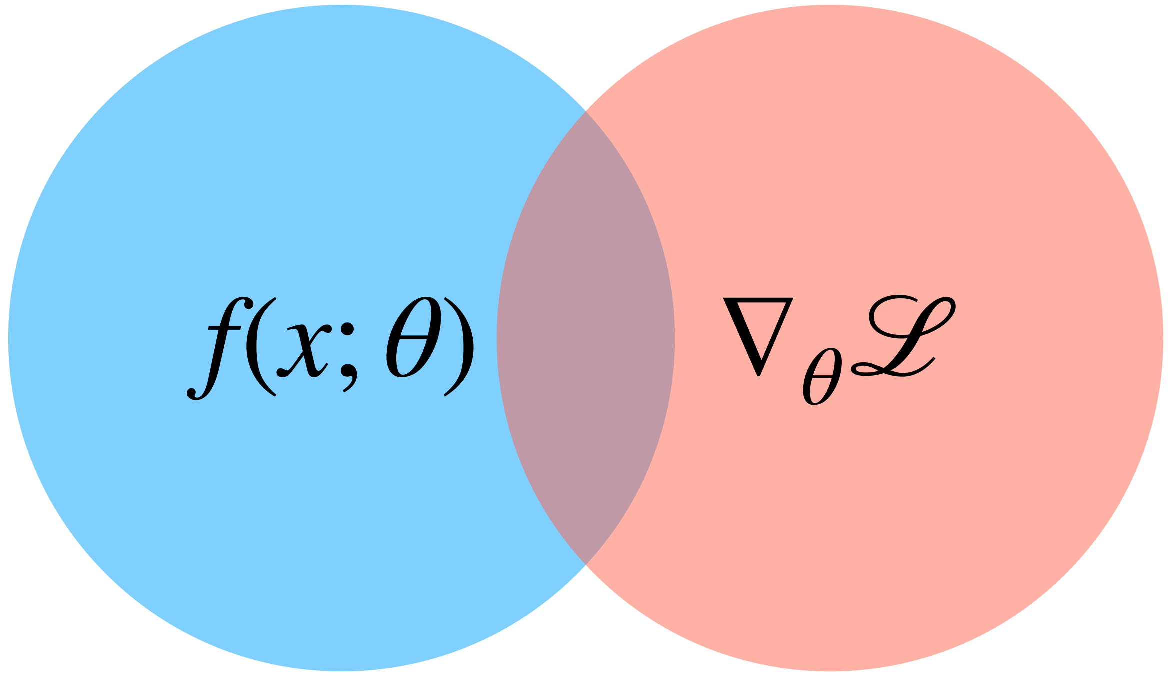
“align then fit”

<u>Kernel Learning</u>



norm is fast & direction is slow

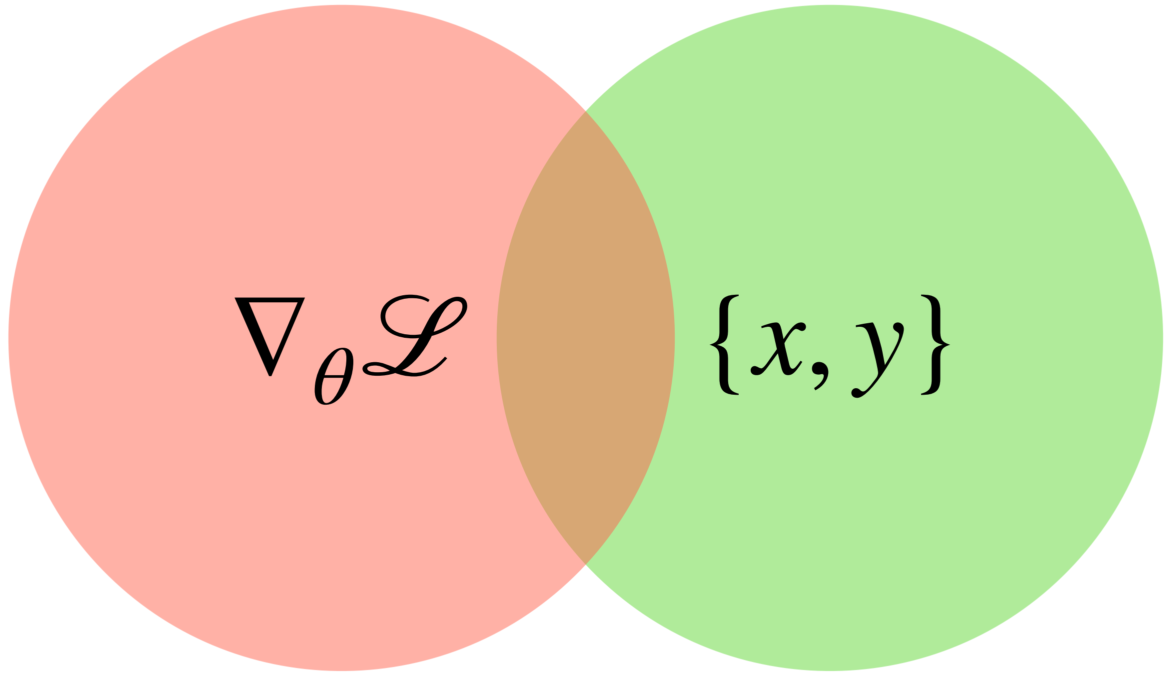
“fit”



Part 1: What conditions enable feature learning?
When and why does feature learning emerge.

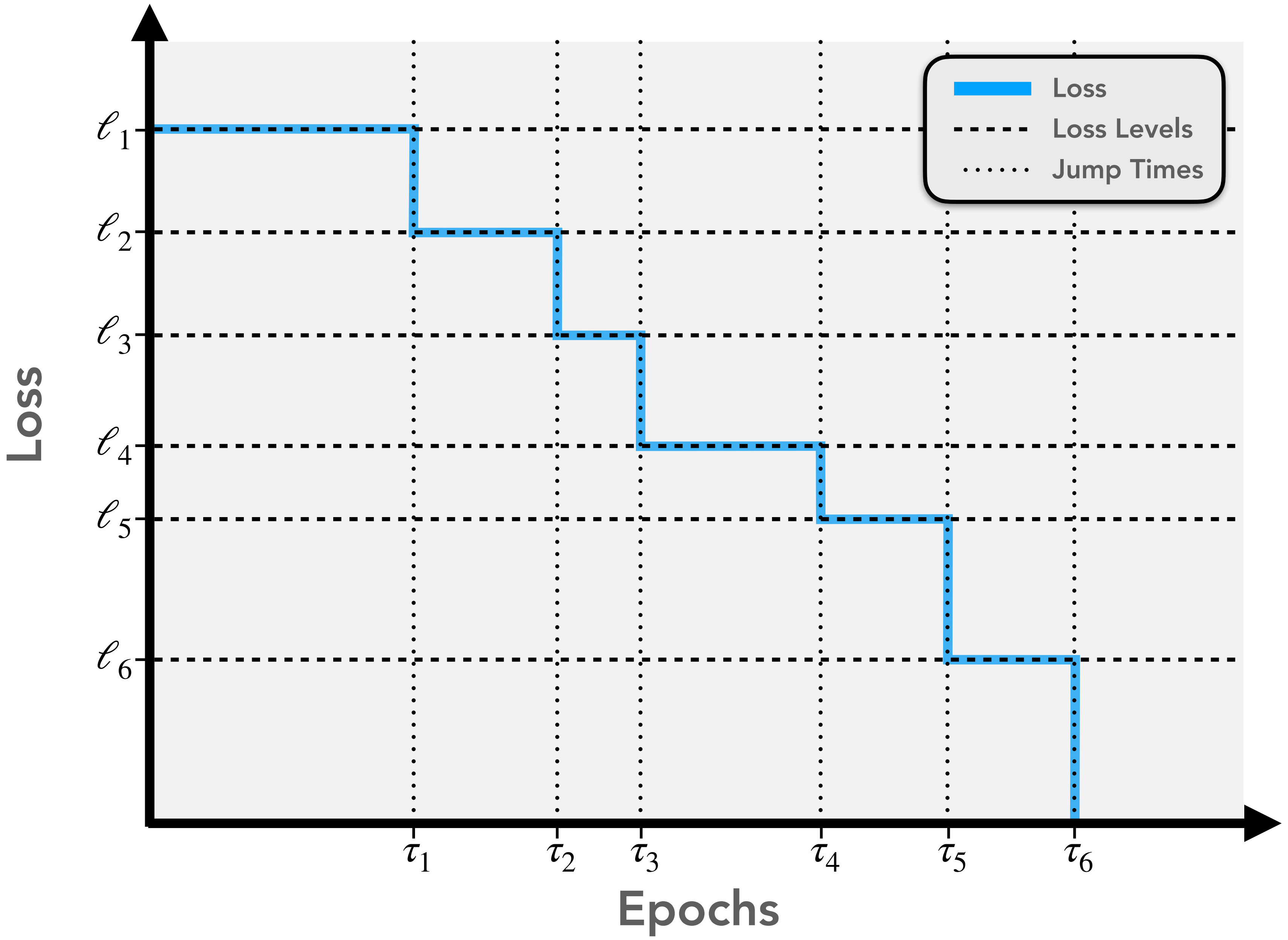
When: Small-scale initializations where the NTK evolves

Why: Saddle-to-saddle dynamics with fast directions and slow norm



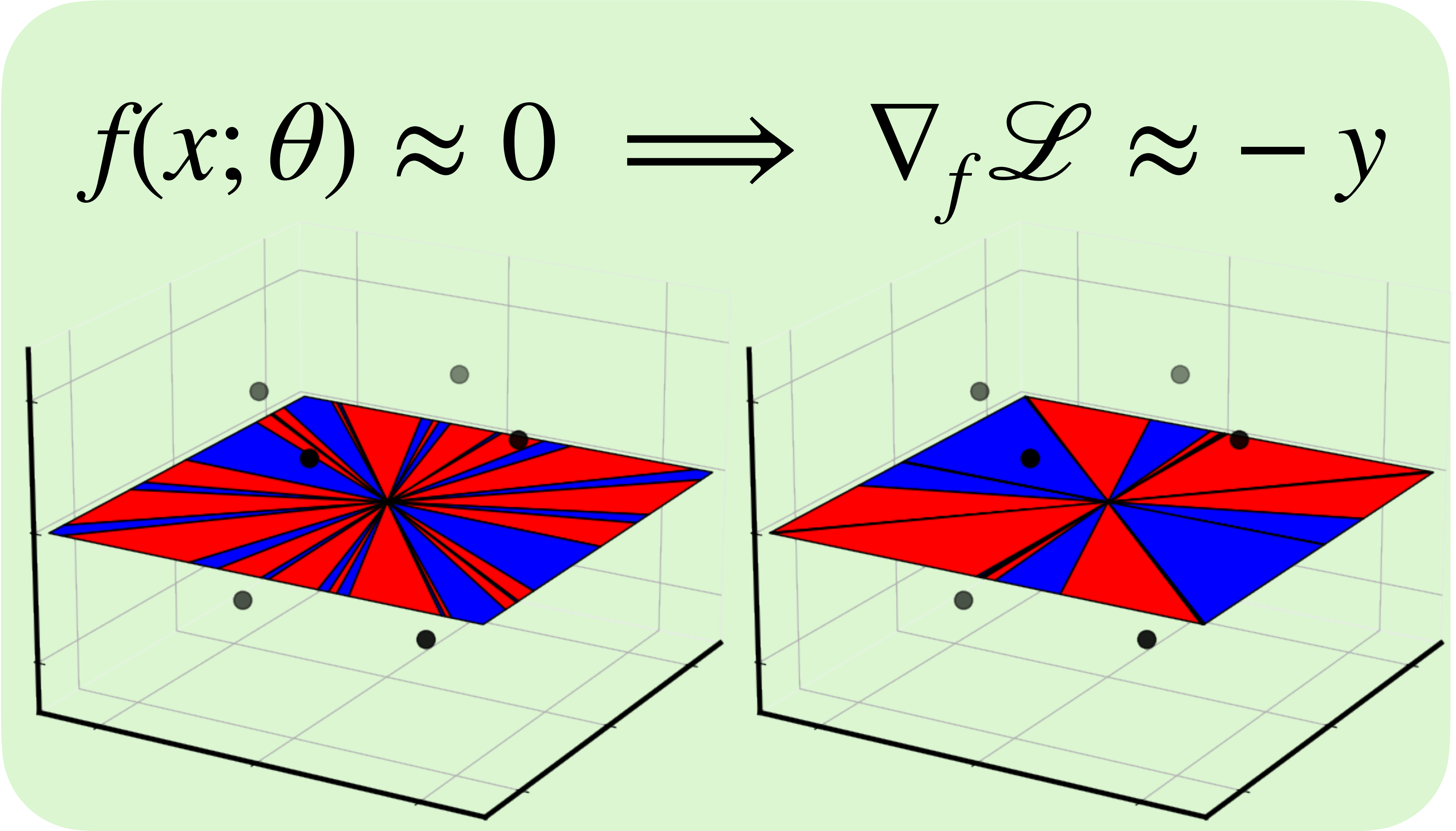
Part 2: What mechanisms drive feature learning?
What features do neural networks learn, and how.

Parameter Space



Can we predict loss levels and jump times of saddle-to-saddle?

Function Space



Can we determine the fixed points of alignment from data?

Alternating Gradient Flows: A Framework for Feature Learning in Neural Networks

Daniel Kunin
Stanford University

Giovanni Luca Marchetti
KTH

Feng Chen
Stanford University

Dhruva Karkada
UC Berkeley

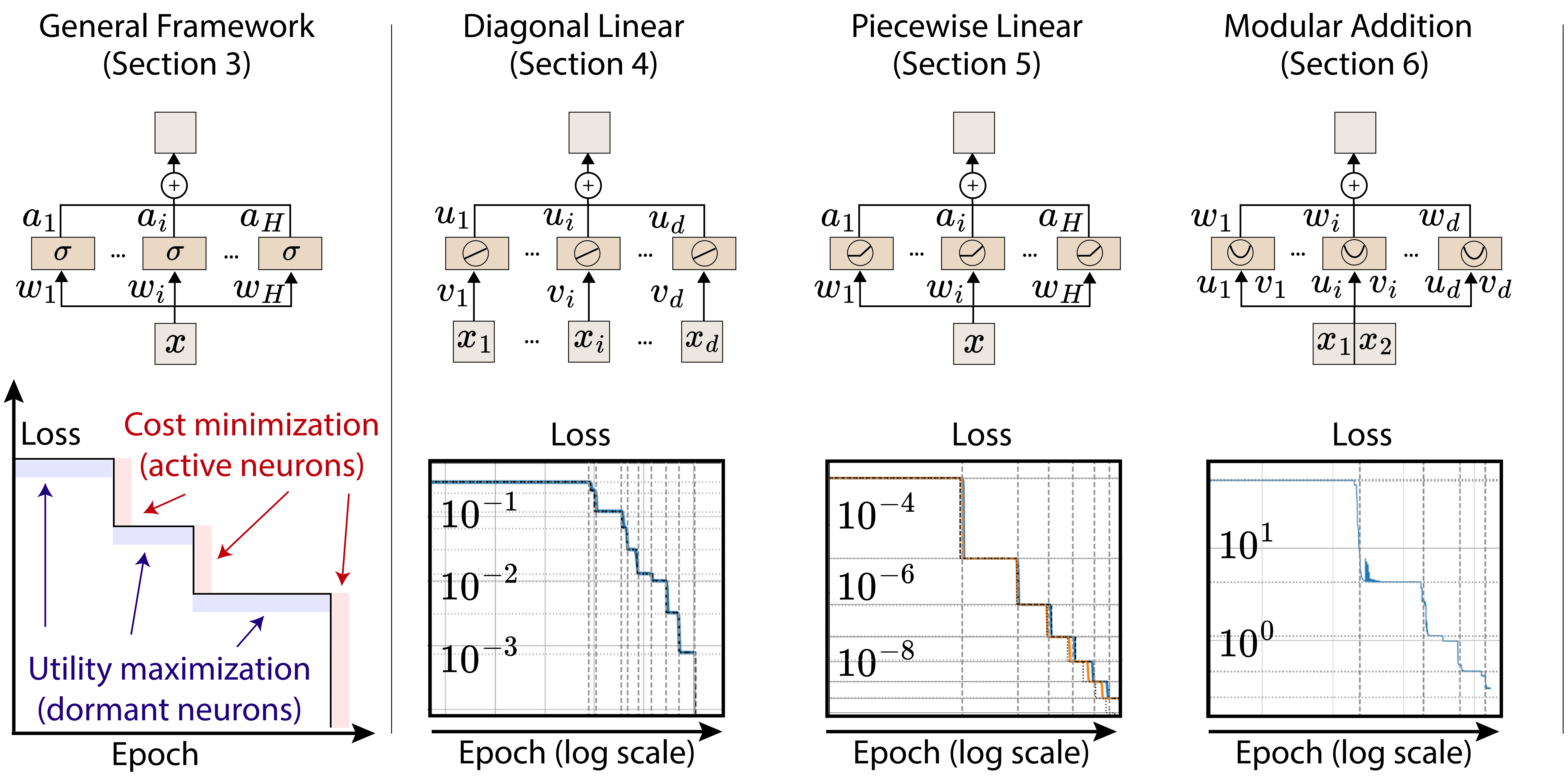
James B. Simon
UC Berkeley

Michael R. DeWeese
UC Berkeley

Surya Ganguli
Stanford University

Nina Miolane
UC Santa Barbara

TL;DR: We conjecture that two-layer neural networks with a vanishing initialization alternates between maximizing a utility function over dormant neurons and minimizing a cost function over active neurons

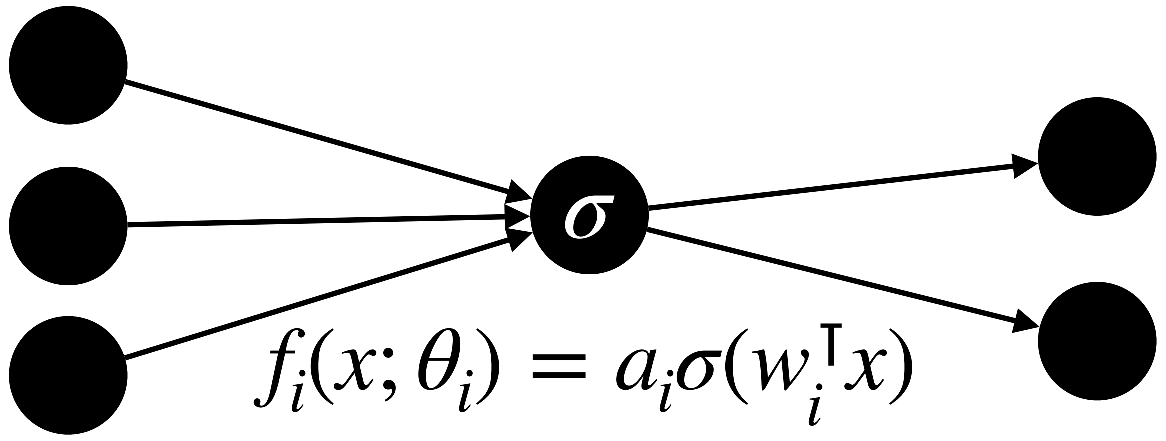


General Framework (Section 3)
Diagonal Linear (Section 4)
Piecewise Linear (Section 5)
Modular Addition (Section 6)
a_1 a_i a_H
σ … σ … σ
w_1 w_i w_H
x
u_1 u_i u_d
v_1 v_i v_d
x_1 … x_i … x_d
w_1 w_i w_d
u_1 v_1 u_i v_i u_d v_d
x_1 x_2
Loss
Cost minimization (active neurons)
Utility maximization (dormant neurons)
Epoch
Loss
10^{-1}
10^{-2}
10^{-3}
Epoch (log scale)
Loss
10^{-4}
10^{-6}
10^{-8}
Epoch (log scale)
Loss
10^{1}
10^{0}
Epoch (log scale)

Alternating Gradient Flows

A *two-layer homogeneous* ($\sigma(\alpha z) = \alpha^{\kappa-1}\sigma(z)$) *network composed as a sum of "neurons":*

$$f(x;\Theta) = \sum_{i=1}^{H} f_i(x;\theta_i)$$



$$\theta_i = (w_i, a_i)$$

By the homogeneity of $f(x;\Theta)$, we can express it as the norm weighted sum

$$f(x;\Theta) = \sum_{i=1}^{H} \|\theta_i\|^\kappa f_i\left(x; \frac{\theta_i}{\|\theta_i\|}\right)$$

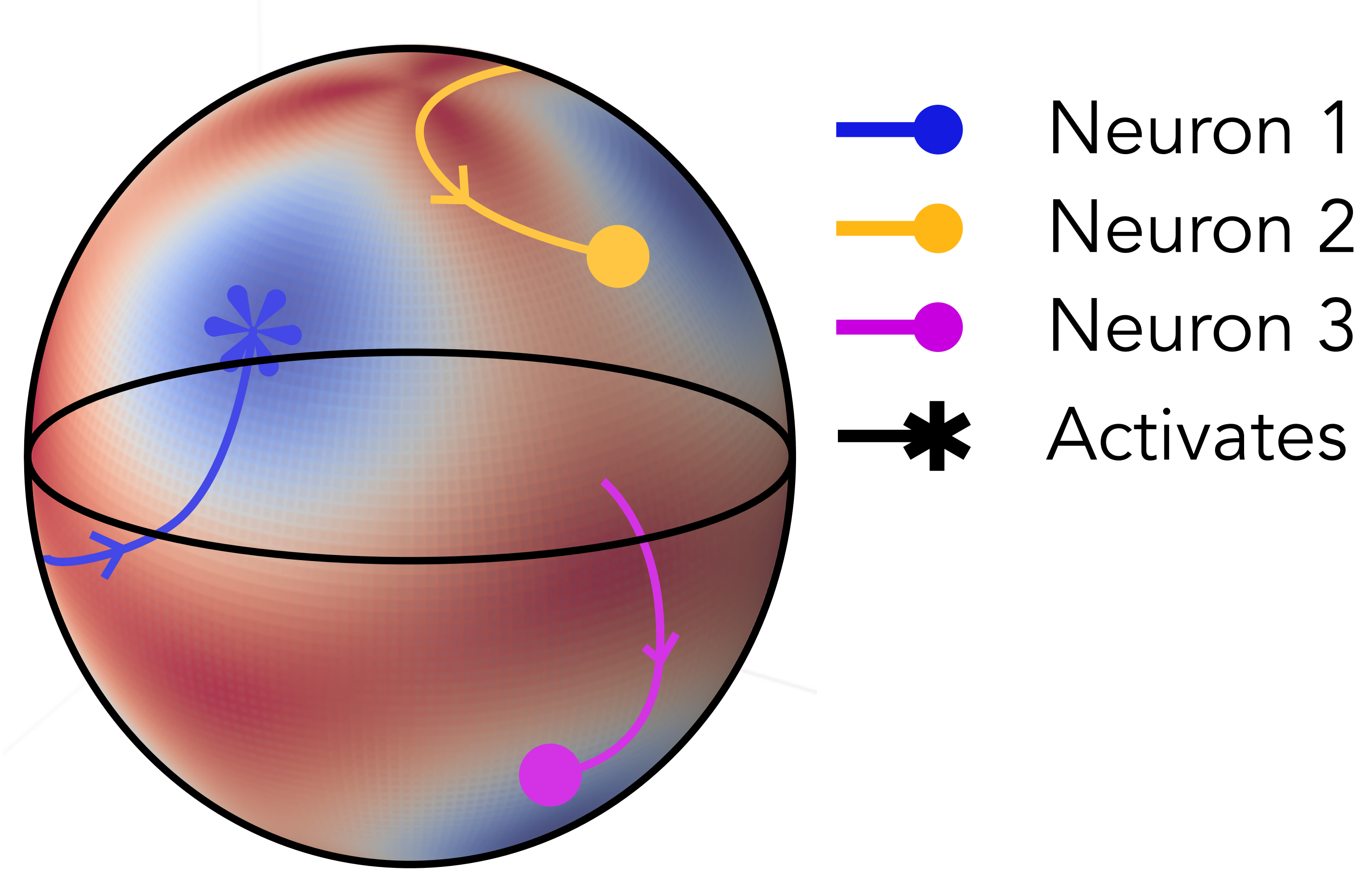
The neurons only "interact" through the **residual**:

$$r(x;\Theta) = y(x) - f(x;\Theta) \in \mathbb{R}^c,$$

In the vanishing limit (because norm dynamics are slow) the residual becomes piecewise constant

$$r(x;\Theta) \to y(x)$$

For a fixed residual $r(x)$, we define the **normalized utility function**, $\mathscr{U} : \mathbb{R}^m \to \mathbb{R}$, for each neuron as:



$$\hat{\mathscr{U}}_i(\theta; r) = \mathbb{E}_x \left[\left\langle f_i\left(x; \frac{\theta}{\|\theta\|}\right), r(x) \right\rangle \right]$$

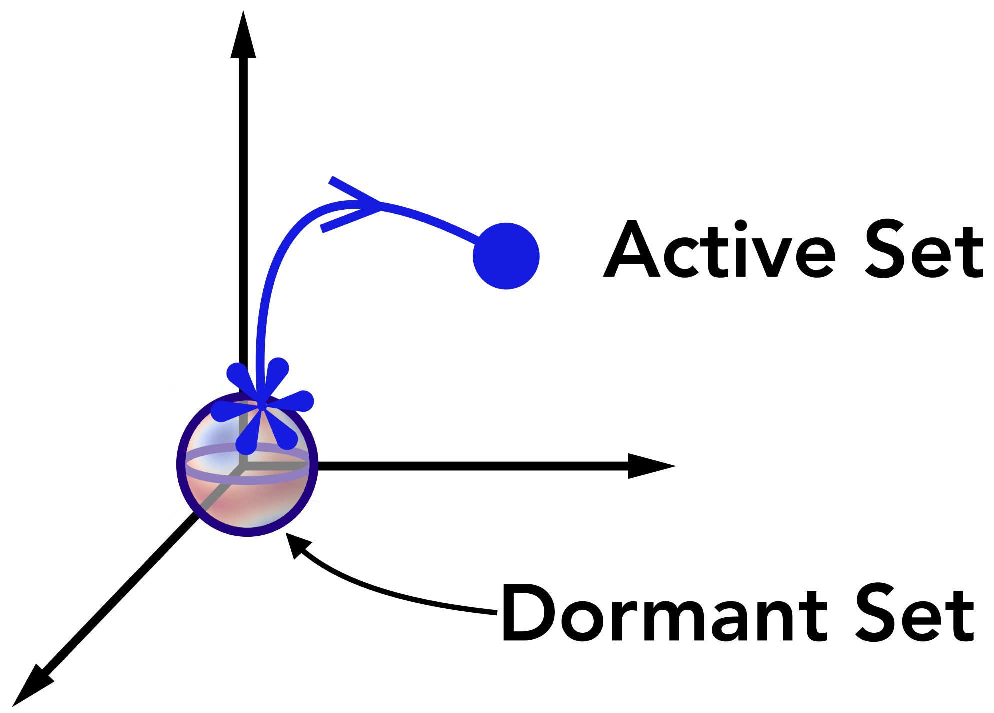
which drives the directional and radial dynamics for each neuron independently

$$\frac{d}{dt}\frac{\theta_i}{\|\theta_i\|} = \|\theta_i\|^{\kappa-2}\mathbf{P}^{\perp}_{\theta_i}\nabla_\theta \hat{\mathscr{U}}_i, \quad \frac{d}{dt}\|\theta_i\| = \kappa\|\theta_i\|^{\kappa-1}\hat{\mathscr{U}}_i\,.$$

*These dynamics breaks down as soon as a neuron **activates** by crossing $\|\theta_i\| = \Theta(1)$ — the residual cannot be constant. For each neuron, we can solve for its **jump time** τ_i, using a path integral of its **accumulated utility** $\mathscr{S}_i(t)$:*

$$\tau_i = \inf\left\{ t > 0 \;\middle|\; \mathscr{S}_i(t) = \begin{cases} -\log\|\theta_i(0)\| & \text{if } \kappa = 2, \\ -\frac{\|\theta_i(0)\|^{2-\kappa} - 1}{2-\kappa} & \text{if } \kappa > 2. \end{cases} \right\} \quad \text{where} \quad \mathscr{S}_i(t) = \int_0^t \kappa \hat{\mathscr{U}}_i(s)ds$$

We partition the set of neurons into **dormant set** (still near the origin) and **active set** (left the neighborhood of the origin).



The active neurons are all "aware" of each other and work collectively to quickly minimize the loss restricted to the active set:

$$\mathscr{L}_{\mathscr{A}}(\theta_{\mathscr{A}}) = \frac{1}{2}\mathbb{E}_x\left[\left\| y(x) - \sum_{i\in\mathscr{A}} f_i(x;\theta_i)\right\|^2\right]$$

*It is possible for an active neuron to become dormant again when minimizing the restricted loss.

The active set equilibrates at a critical point $\theta^*_{\mathscr{A}}$, then a new residual is computed, which is a saddle point of the original loss.

$$r(x) = y(x) - \sum_{i\in\mathscr{A}} f_i(x;\theta^*_i) \qquad\qquad \theta^*_{\mathscr{A}} \in \mathrm{Crit}(\mathscr{L}_{\mathscr{A}}) \implies (\theta^*_{\mathscr{A}}, 0) \in \mathrm{Crit}(\mathscr{L})$$

The process repeats until all dormant neurons are either active or the residual is zero.

Alternating Gradient Flows Framework

Initialize: $\mathscr{D} = [H], \mathscr{A} = \{\}, r(x) = y$

Utility Maximization:

For each $i \in \mathscr{D}$, $\quad$ maximize $\mathscr{U}_i(\theta_i, r)$ $\quad$ subject to $\|\theta_i\| = 1.$

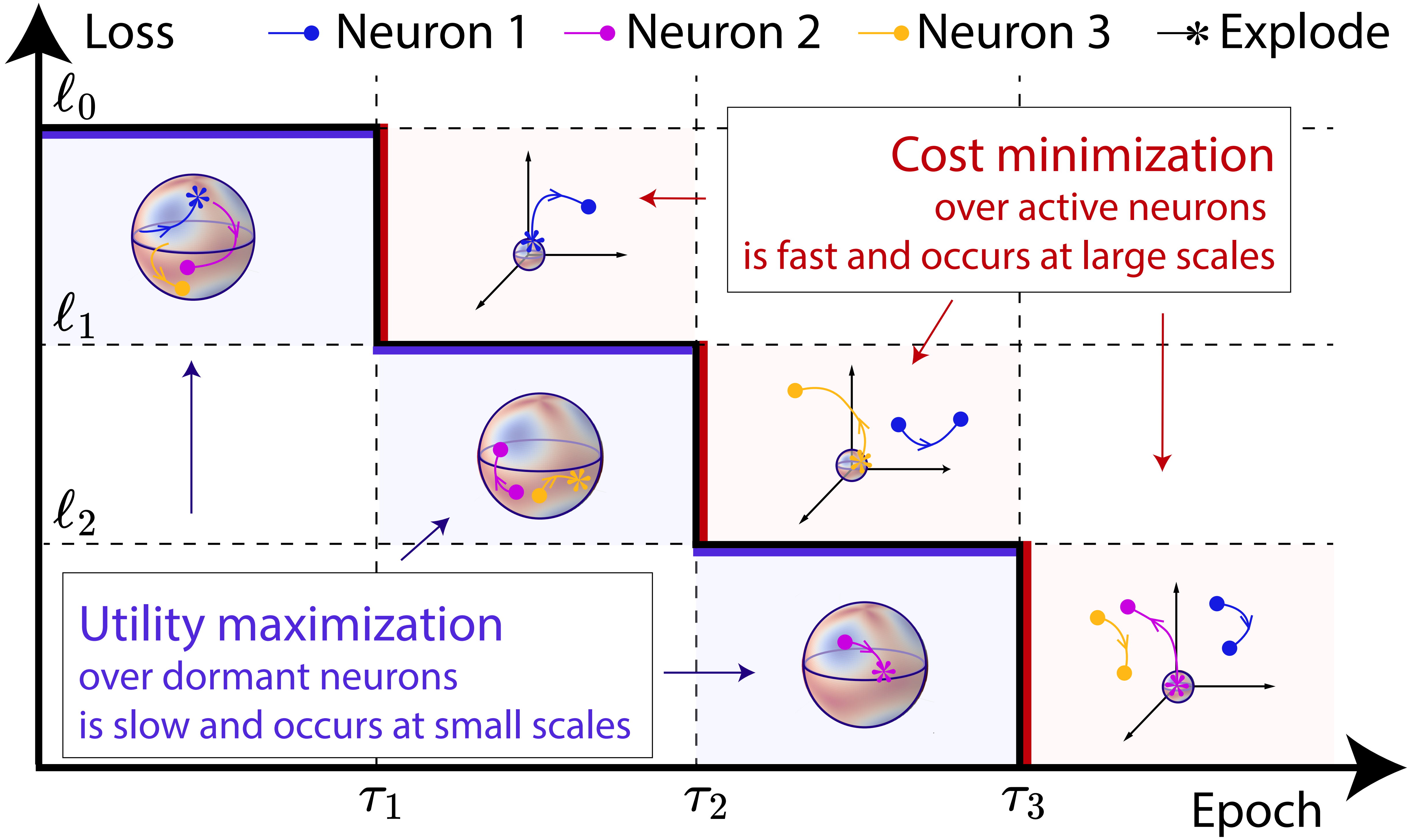
Transition neuron with smallest jump time τ_i to active set.

Cost Minimization:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(\Theta) \\ \text{subject to} \quad & \|\theta_i\| = 0, \quad \forall i \in \mathscr{D}, \\ & \|\theta_i\| \geq 0, \quad \forall i \in \mathscr{A}. \end{aligned}$$

Update residual and if necessary return neurons to dormant set.

Termination: When $\mathscr{D} = \{\}$ or $r(x) = 0$

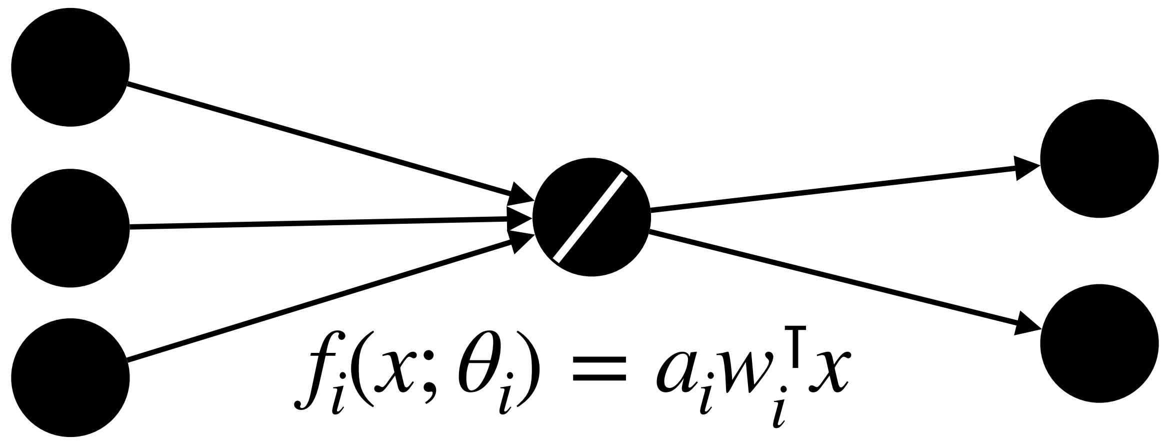


Conjecture: *With vanishing initialization, gradient flow in two-layer homogeneous networks follows a discrete trajectory transitioning between saddles, with the order and time determined by AGF.*

1: AGF for linear networks

A *two-layer linear network as a sum of* H *neurons:*

$$f(x;\Theta) = \sum_{i=1}^{H} f_i(x;\theta_i)$$



$$\theta_i = (a_i, w_i)$$

Let's assume $X^\intercal X = \mathbf{I}_d$ such that its only the input-output cross-covariance $X^\intercal Y$ that need to be learned

At initialization, the residual is $y(x)$, and then the utility is the bi-linear product

$$\mathcal{U}_i(\theta; r) = \mathbb{E}_x\left[\left\langle a_i w_i^\intercal x, r(x)\right\rangle\right] = a_i^\intercal \mathbb{E}_x\left[yx^\intercal\right] w_i = a_i^\intercal Y^\intercal X w_i$$

Utility maximization is a Rayleigh quotient problem, resulting in the top left/right singular vector of $Y^\intercal X = U\Sigma V^\intercal$

$$\begin{aligned} \text{maximize} \quad & a_i^\intercal Y^\intercal X w_i \\ \text{subject to} \quad & \|a_i\|^2 + \|w_i\|^2 = 1. \end{aligned} \implies (a_i^*, w_i^*) \propto (u_1, v_1)$$

We can estimate the accumulated utility $\mathcal{S}_i(t) \approx \kappa \mathcal{U}_i^* t$, and thus the jump time as

$$\tau_i \approx -\frac{\log(\|\theta_i(0)\|)}{\sigma_1}.$$

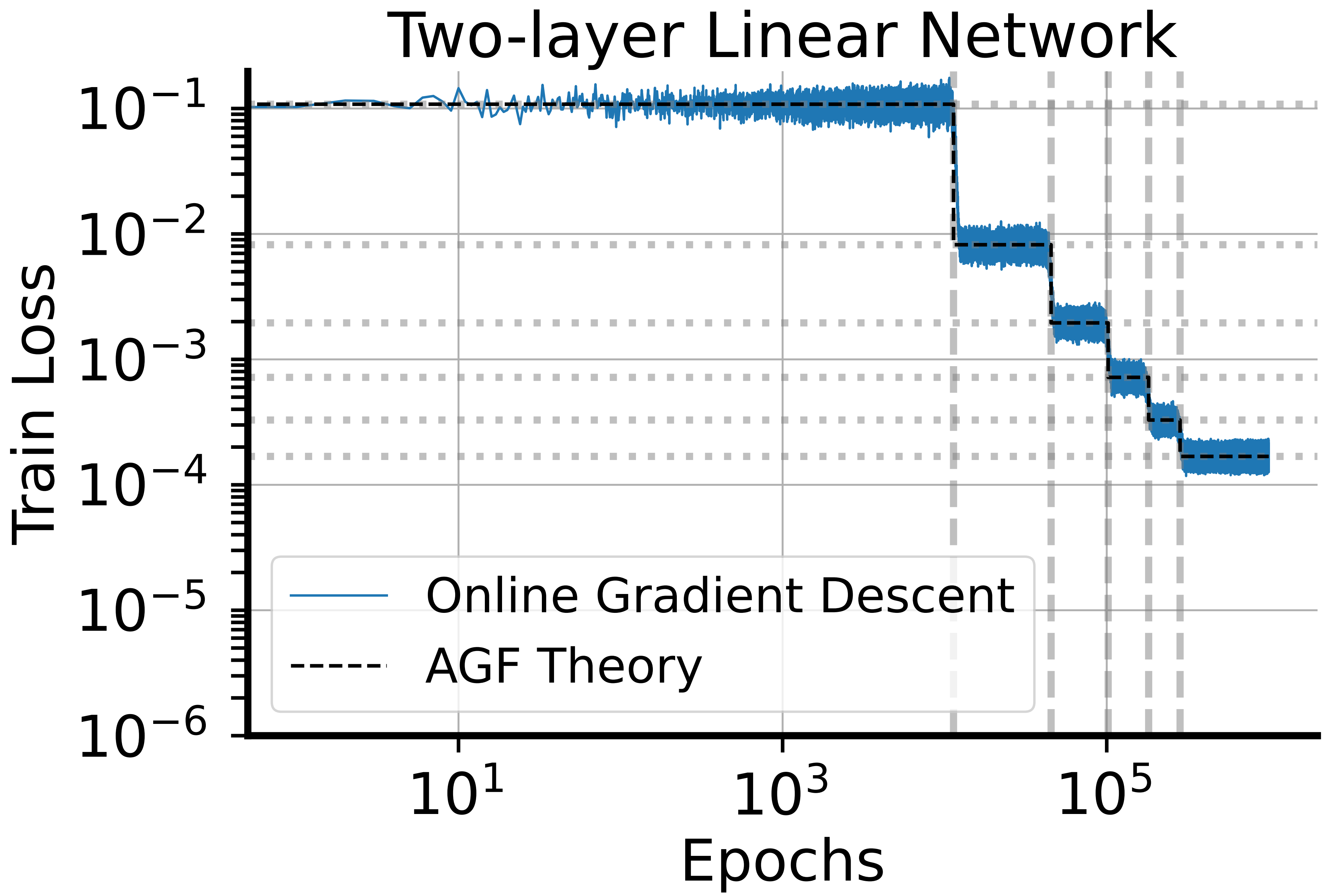
After cost minimization, $f(x;\Theta) = u_1 \sigma_1 v_i^\intercal x$, thus the new utility will be computed with the matrix,

$$Y^\intercal X - u_1 \sigma_1 v_1^\intercal,$$

then the second top singular vectors are learned and it repeats recursively.

Algorithm 3: Greedy Low-Rank Learning Li et al. [12]

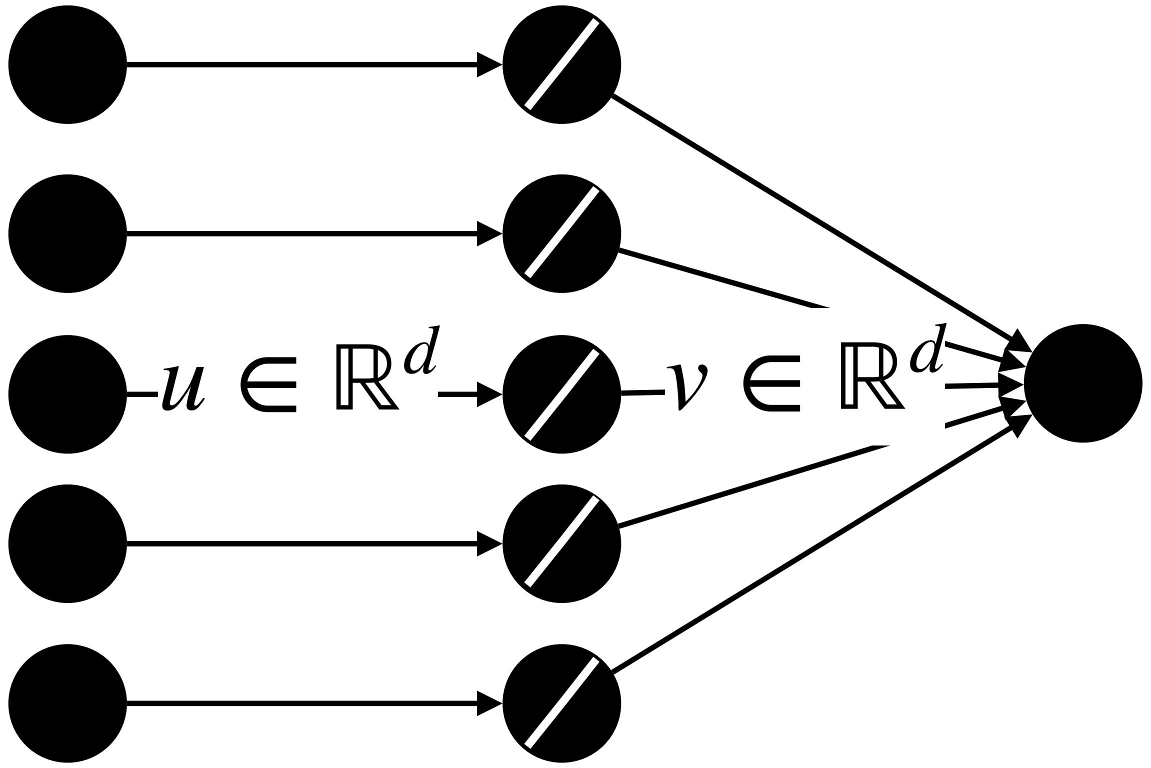
Input: step $\eta > 0$, iterations $T \in \mathbb{N}$, perturbation $\varepsilon > 0$
Initialize: $r \leftarrow 0$, $W_0 \leftarrow 0 \in \mathbb{R}^{d\times d}$, $U_0(\infty) \in \mathbb{R}^{d\times 0}$
while $\lambda_1(-\nabla\mathcal{L}(W_r)) > 0$ **do**
  $r \leftarrow r + 1$
  $u_r \leftarrow$ unit top eigenvector of $-\nabla\mathcal{L}(W_{r-1})$
  $U_r(0) \leftarrow [U_{r-1}(\infty) \quad \sqrt{\varepsilon} u_r] \in \mathbb{R}^{d\times r}$
  for $t = 0, 1, \ldots, T$ **do**
    $U_r(t+1) \leftarrow U_r(t) - \eta\nabla\mathcal{L}(U_r(t))$
  $W_r \leftarrow U_r(\infty) U_r^\top(\infty)$
return Sequence of W_r



Li et al. 2020 proved that all two-layer matrix factorization problems, i.e. $X^\intercal X \neq \mathbf{I}_d$, gradient flow with infinitesimal initialization is mathematically equivalent to a simple heuristic rank minimization algorithm.

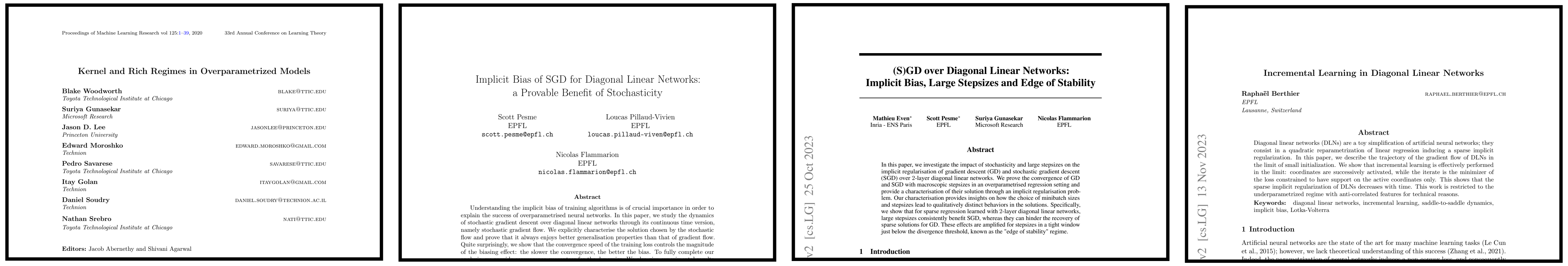
The AGF framework is this algorithm, once you replace the top eigenvector calculation as maximizing utility.

2. AGF for diagonal linear networks



$$f(x;\theta) = (u \odot v)^\intercal x$$

A diagonal linear network trained on MSE $\mathscr{L} = \frac{1}{2}\|y - (u \odot v)^\intercal X^\intercal\|^2$



Gissin et al. 2019, Woodworth et al. 2020, Pesme et al. 2021, Even et al. 2023, Berthier 2023, Papazov et al. 2024

We consider gradient flow dynamics from initialization $u = \alpha\mathbf{1}, v = 0$.

The utility for each coefficient is:

$$\hat{\mathscr{U}}_i(u_i, v_i; r) = \frac{1}{n}\sum_{j=1}^{n} u_i v_i X_{ij} r_j = -u_i v_i \nabla_{\beta_i}\mathscr{L}(\beta_\theta)$$

We can get exact expression for accumulated utility

$$\mathcal{S}_i(t+\tau^*) = \frac{1}{2}\log\left(\frac{\cosh(4\mathscr{U}_i^*(t)\tau^* + c_i(t))}{\cosh(c_i(t))}\right)$$

In the limit $\alpha \to 0$, AGF converges to the same sequence found by Pesme and Flammarion, 2023.

Algorithm 2: Pesme and Flammarion [22]

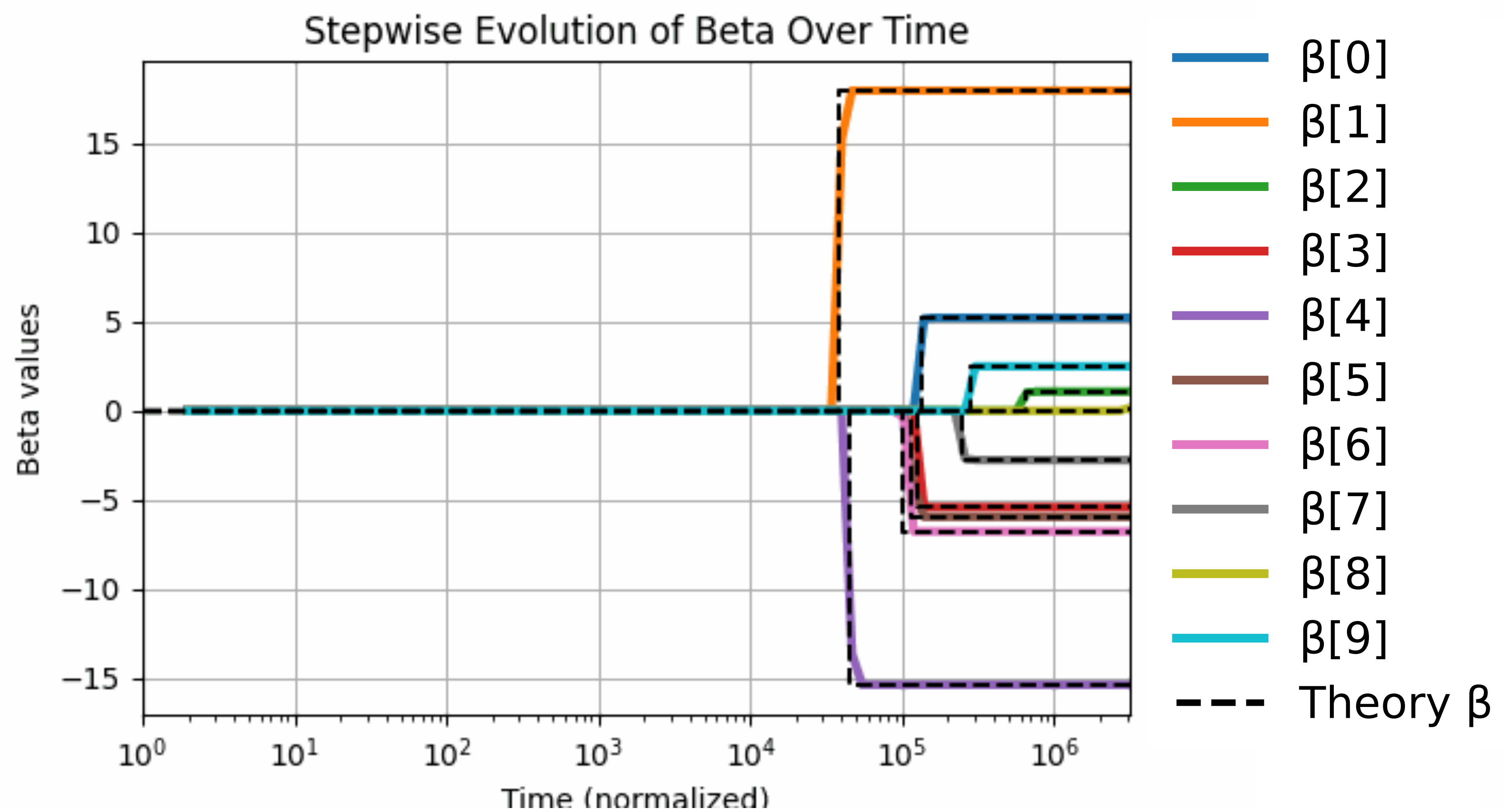
Initialize: $t \leftarrow 0, \beta \leftarrow 0 \in \mathbb{R}^d, \mathcal{S} \leftarrow 0 \in \mathbb{R}^d$
while $\nabla\mathcal{L}(\beta) \neq 0$ **do**

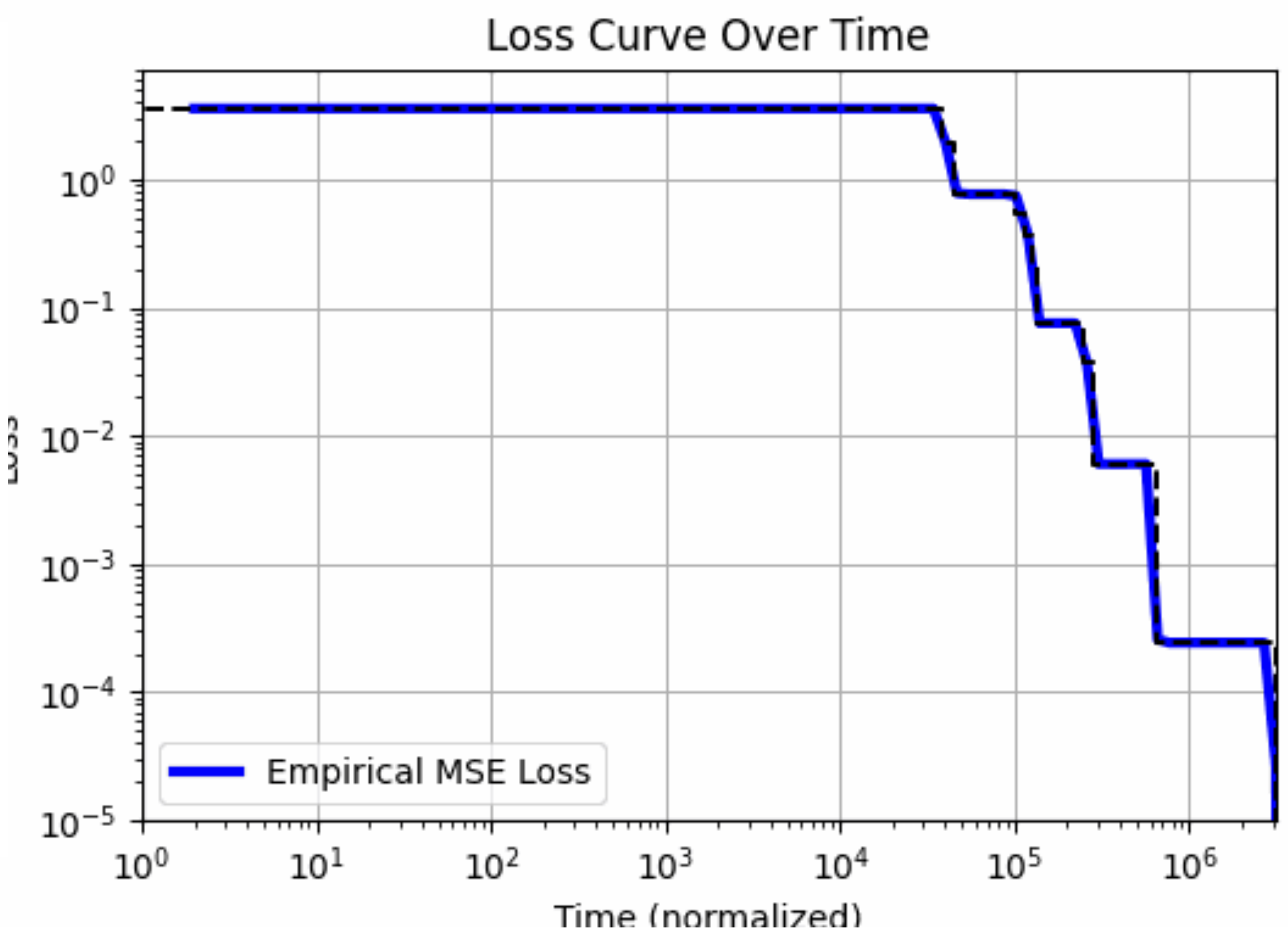
$\mathcal{D} \leftarrow \{j \in [d] \mid \nabla\mathcal{L}(\beta)_j \neq 0\}$
$\tau^* \leftarrow \inf\{\tau_i > 0 \mid \exists i \in \mathcal{D},\ \mathcal{S}_i - \tau_i\nabla\mathcal{L}(\beta)_i = \pm 1\}$
$t \leftarrow t + \tau^*, \quad \mathcal{S} \leftarrow \mathcal{S} - \tau^*\nabla\mathcal{L}(\beta)$

$$\beta = \arg\min \mathcal{L}(\beta) \text{ where } \beta \in \left\{\beta \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \beta_i \geq 0 & \text{if } \mathcal{S}_i = +1, \\ \beta_i \leq 0 & \text{if } \mathcal{S}_i = -1, \\ \beta_i = 0 & \text{if } \mathcal{S}_i \in (-1, 1) \end{array}\right\}$$

return Sequence of (β, t)

Coordinate-aligned $X^\intercal X$

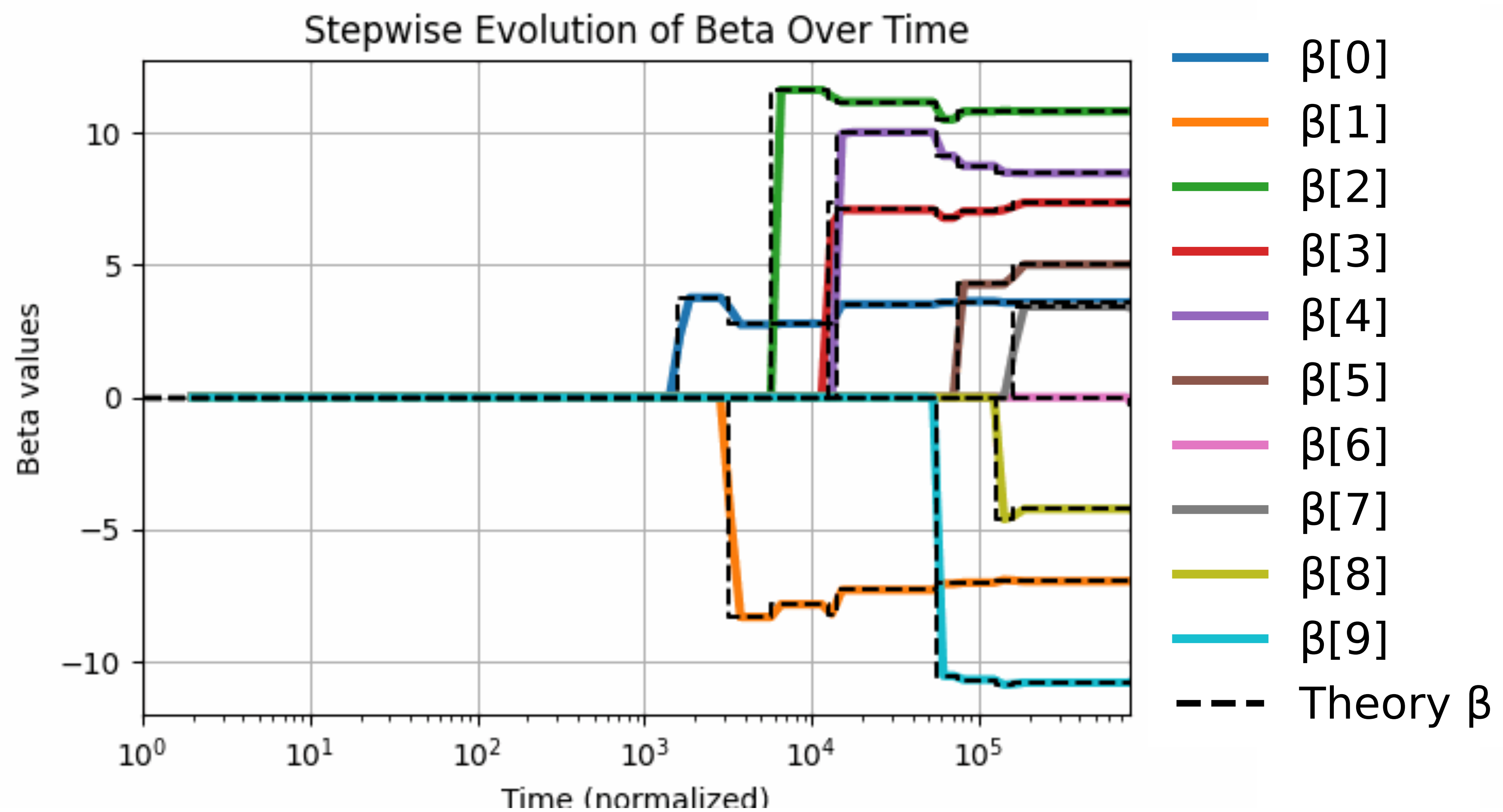


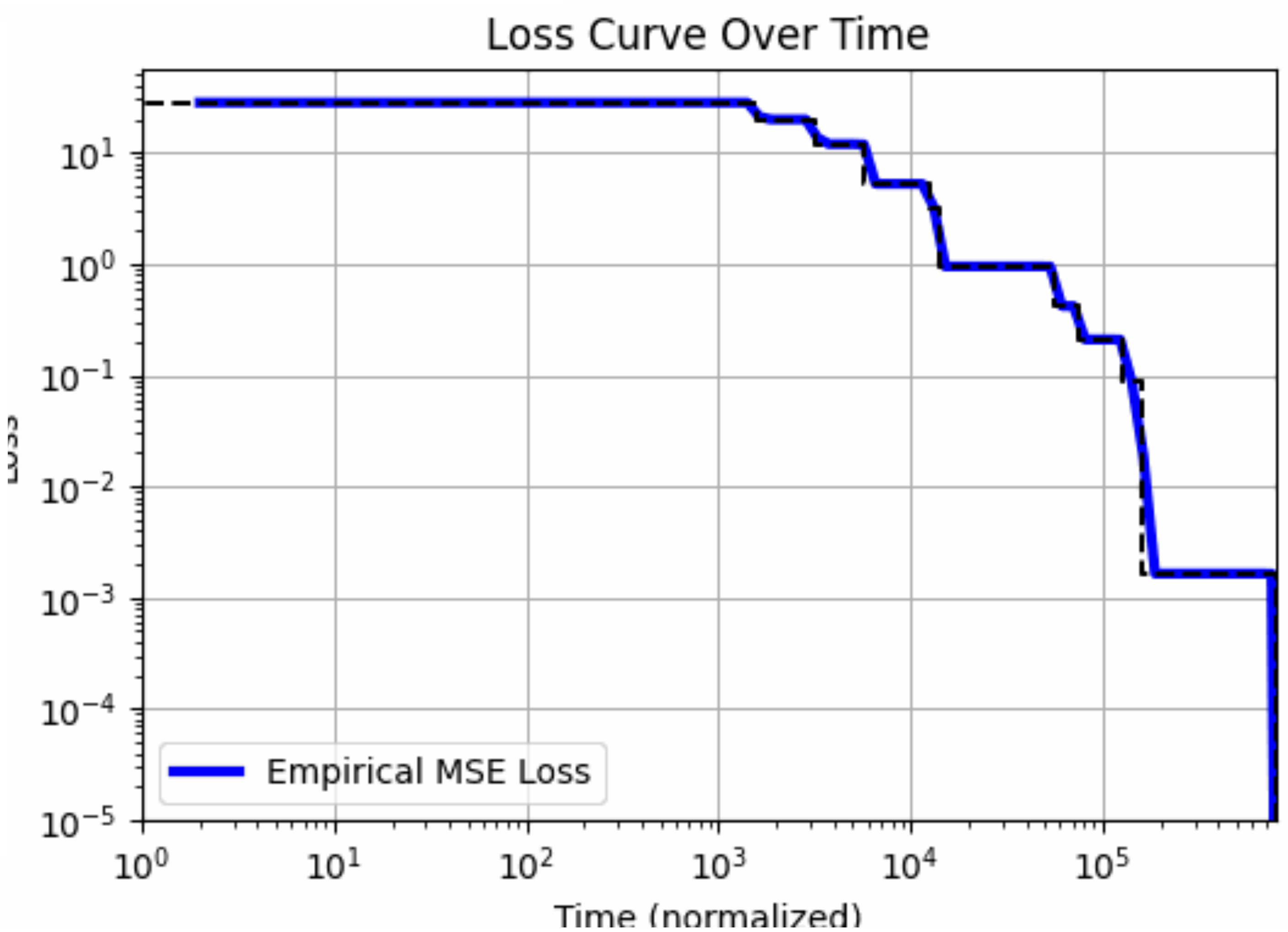


...... = theory

PF 2023	AGF
0.000000	0.000000
384.197942	384.197942
448.504742	448.504742
1018.934863	1018.934863
1159.597081	1159.597081
1274.067146	1274.067146
1323.114193	1323.114193
2492.848996	2492.848996
2769.748739	2769.748739
6422.194344	6422.194343
31358.671215	31358.671212

Unaligned $X^\intercal X$





...... = theory

PF 2023	AGF
0.000000	0.000000
15.674629	15.674629
31.729039	31.729039
57.702840	57.702840
127.497782	127.497782
142.665350	142.665351
554.688721	554.688721
758.349115	758.349115
1276.582141	1276.582141
1603.605670	1603.605669
7982.029052	7982.029053

3. AGF for quadratic networks trained on modular addition

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$$a + b \mod 5$$

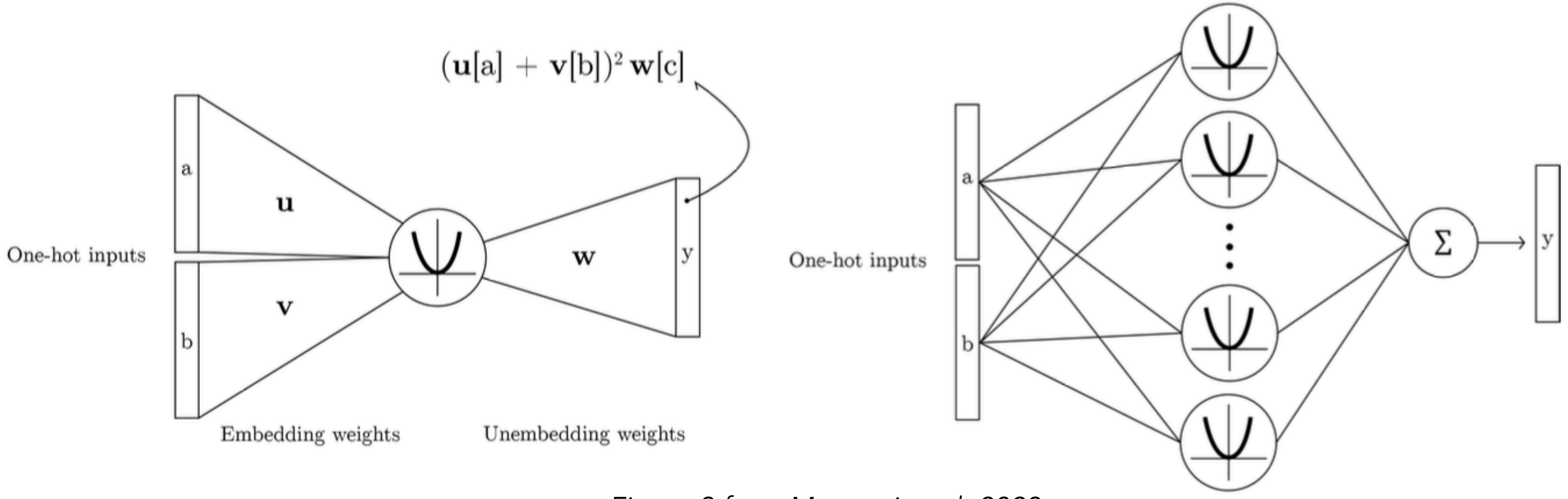


Figure 2 from Morwani et al. 2023

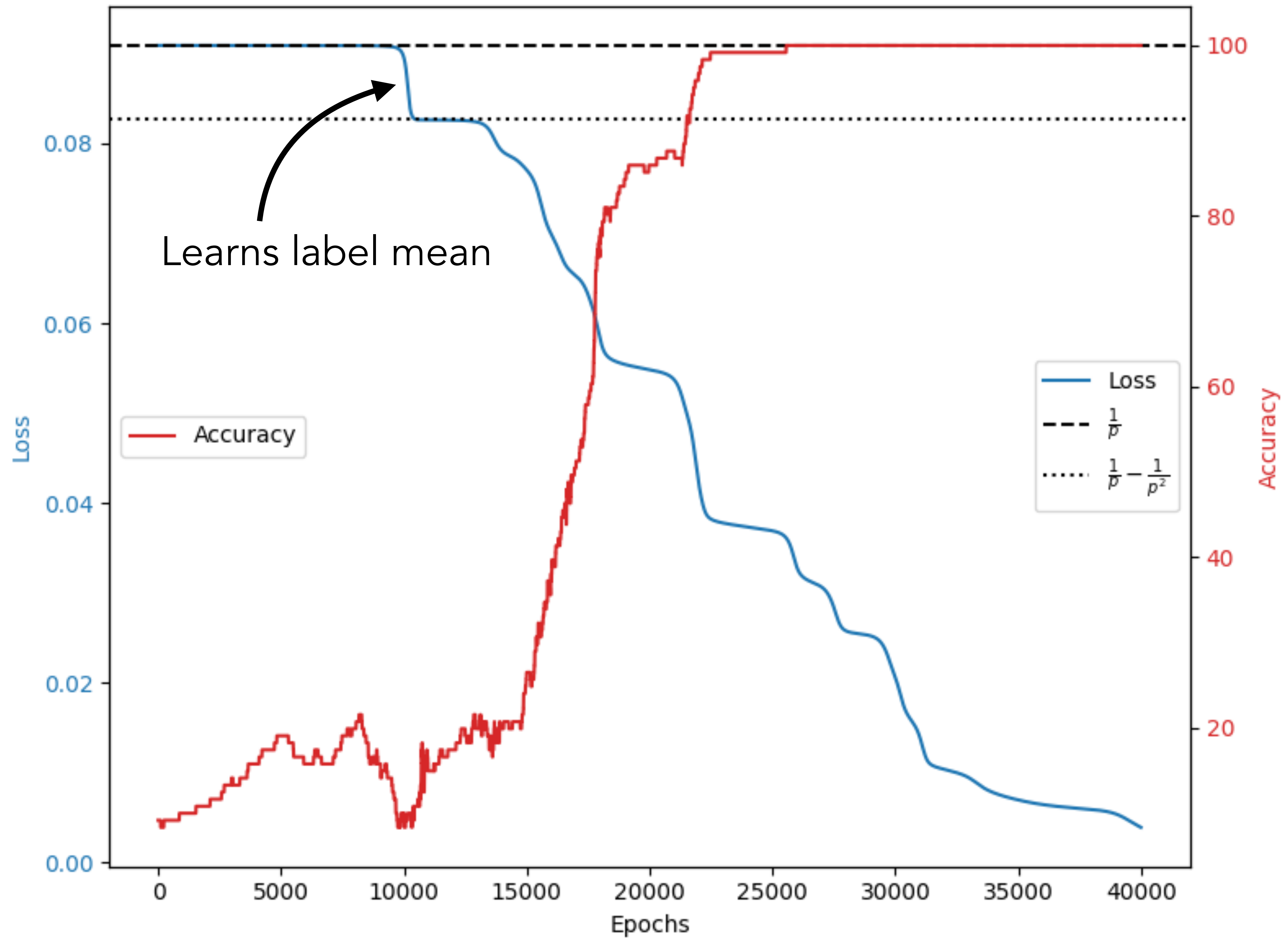
Grokking modular arithmetic

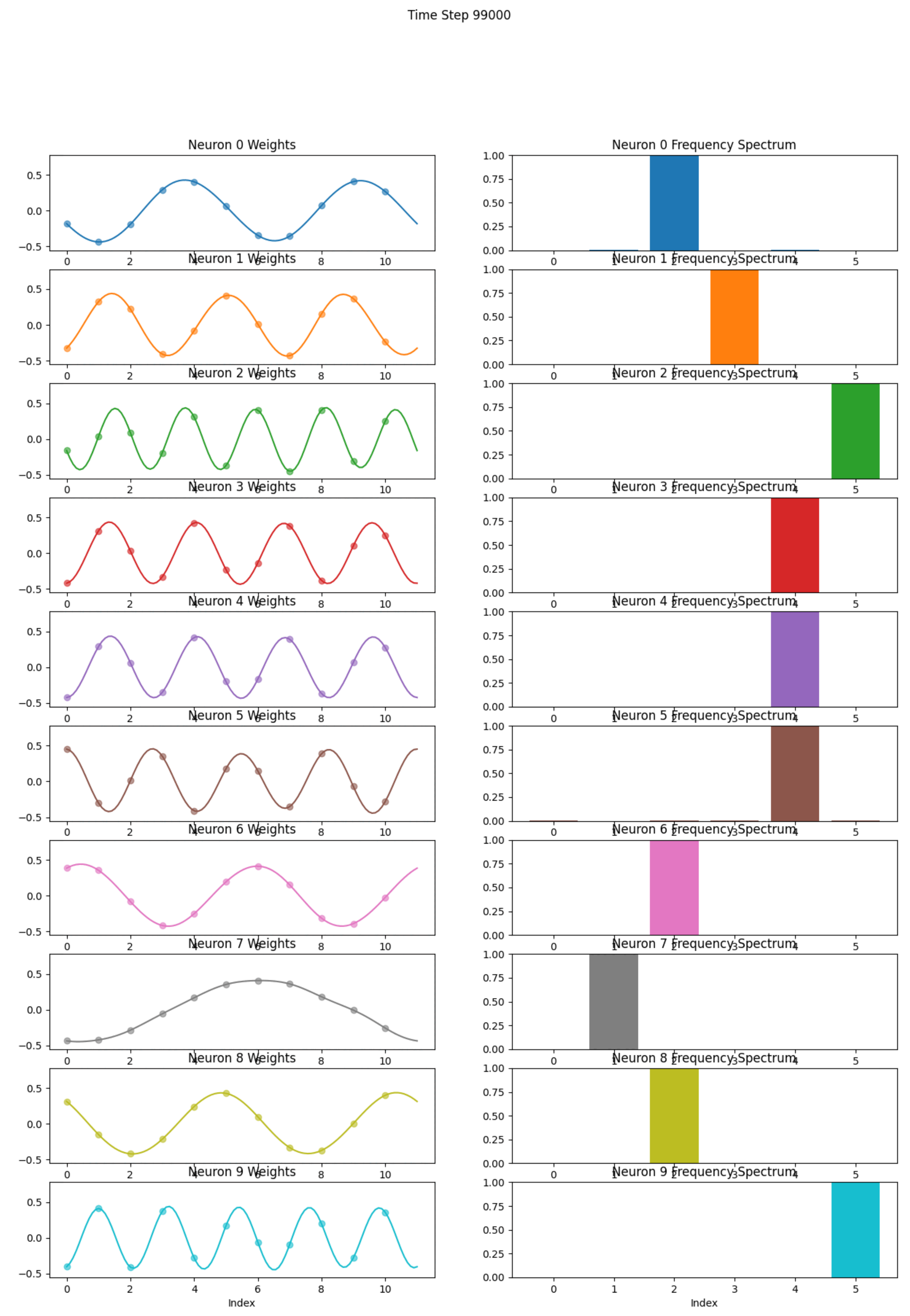
Andrey Gromov
Meta AI
Meta Platforms, Inc.
Menlo Park, California 94025
&
Department of Physics,
Condensed Matter Theory Center,
University of Maryland
College Park, Maryland 20740
gromovand@meta.com

For $a, b \in \mathbb{Z}_p$ *and* $k \in \mathbb{Z}_p \backslash \{0\}$, *then*

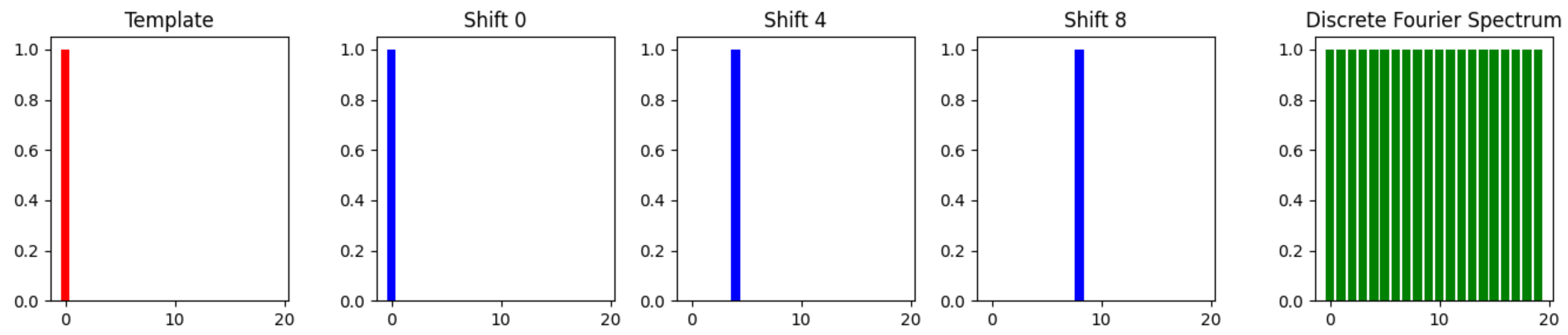
$$(a+b) \mod p = \arg\max_{c \in \mathbb{Z}_p} \left\{ \cos\left(2\pi k \frac{a+b-c}{p}\right) \right\}$$

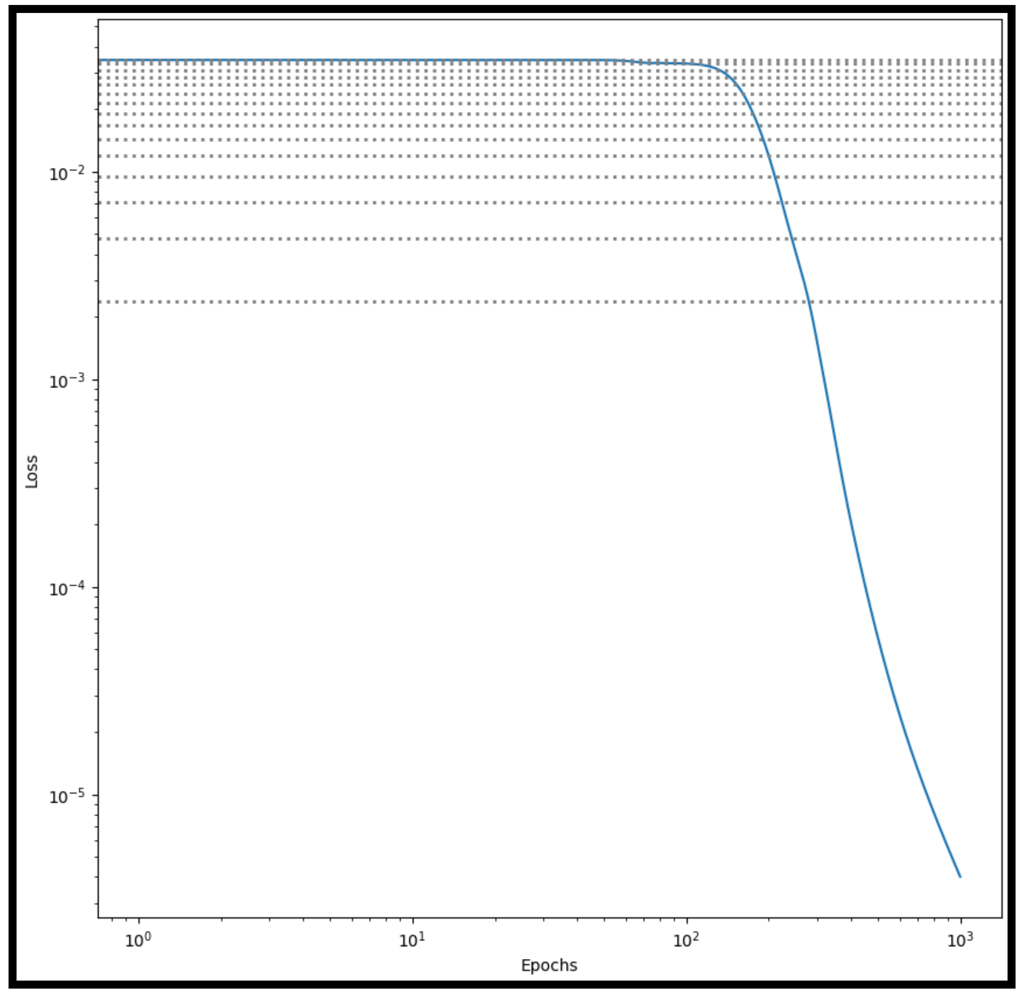
Neurons specialize to a specific frequency



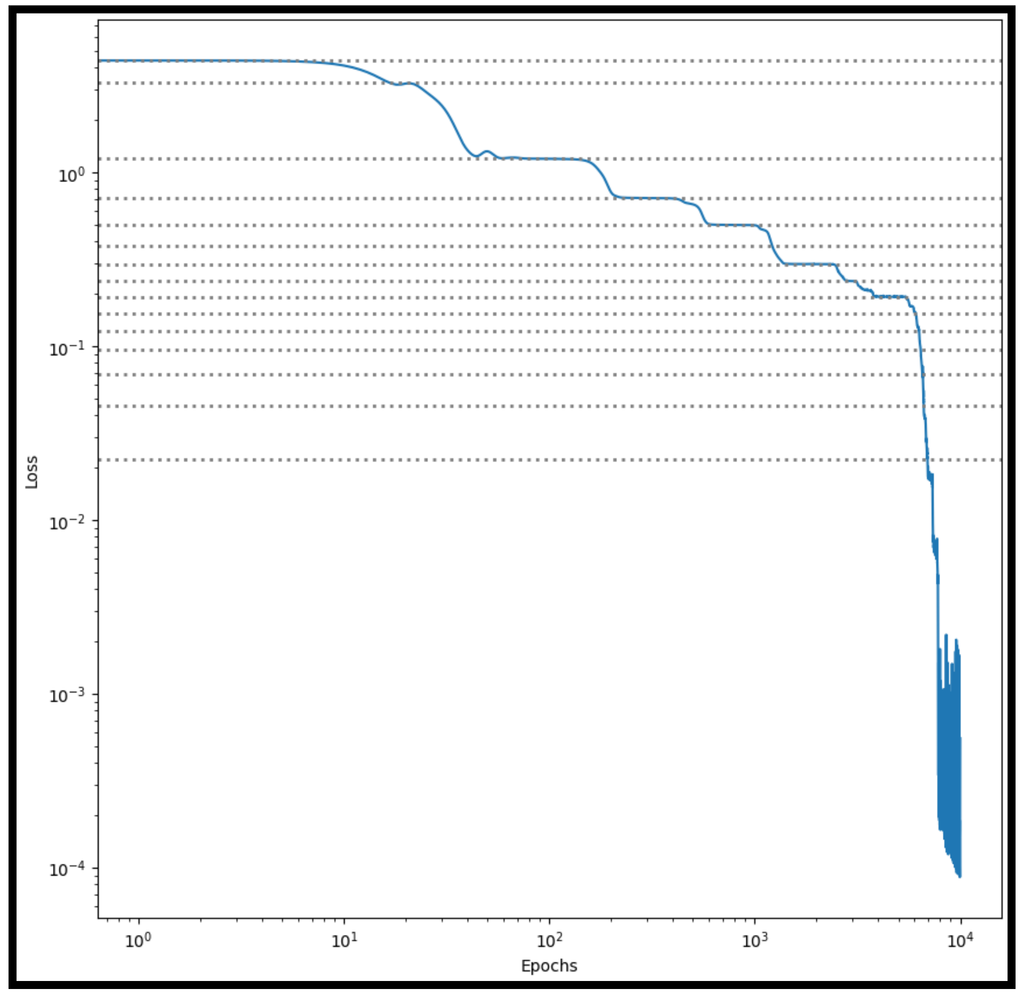


No saddles with a one-hot encoding:





Saddles with a correlated encoding:



Cosine waves maximize the utility function

We prove that the utility-maximizing unit vectors are cosine waves at the dominant frequency of the encoding vector.

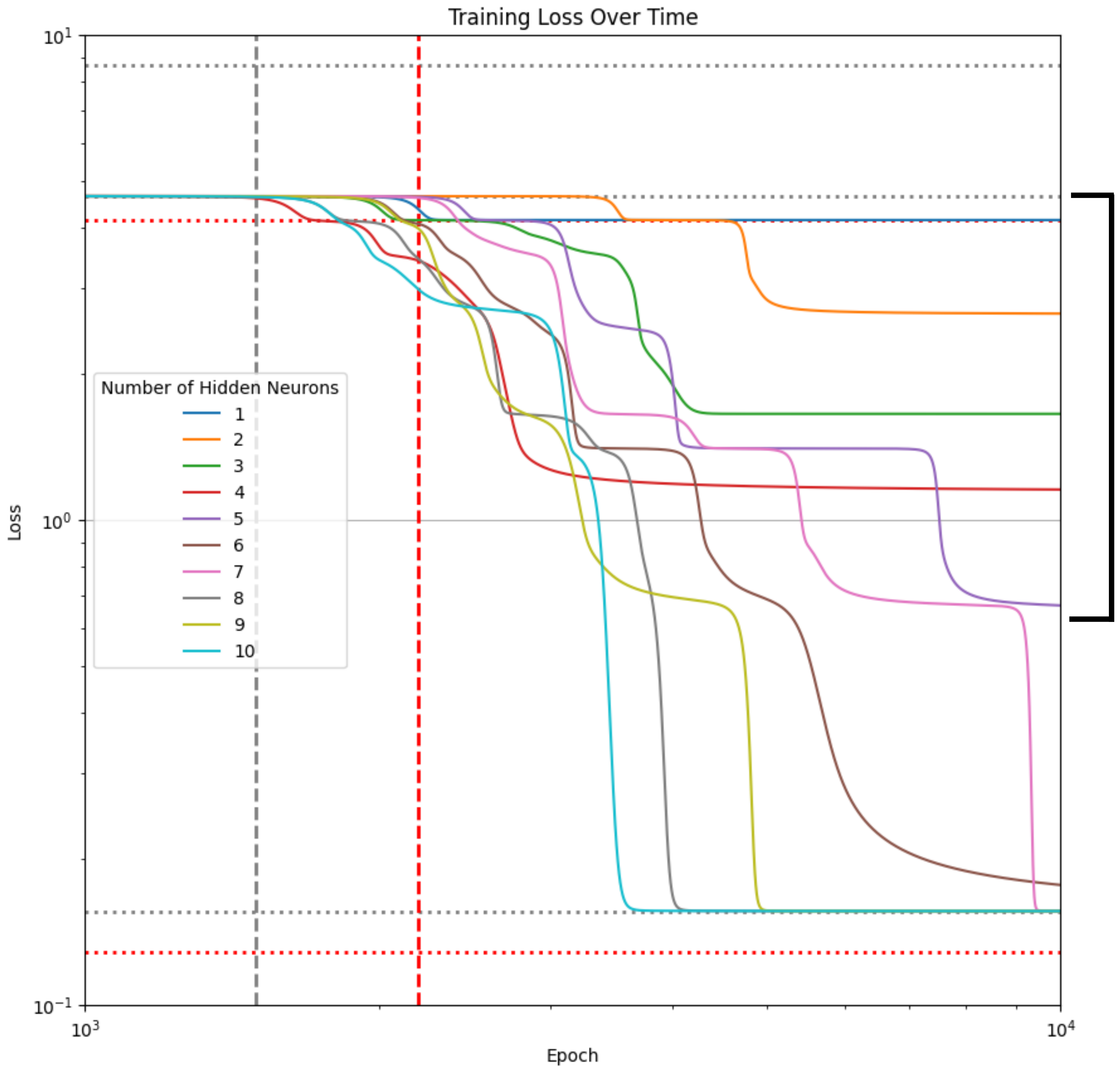
Theorem 6.2. *Let ξ be a frequency that maximizes $|\hat{x}[k]|$, $k = 1, \ldots, p-1$, and denote by s_x the phase of $\hat{x}[\xi]$. Then the unit vectors $\theta_* = (u_*, v_*, w_*)$ that maximize the utility function $\mathcal{U}_0(\theta)$ take the form*

$$\begin{aligned} u_*[a] &= \sqrt{\frac{2}{p}} \cos\left(2\pi \frac{\xi}{p} a + s_u\right) \\ v_*[b] &= \sqrt{\frac{2}{p}} \cos\left(2\pi \frac{\xi}{p} b + s_v\right) \\ w_*[c] &= \sqrt{\frac{2}{p}} \cos\left(2\pi \frac{\xi}{p} c + s_w\right), \end{aligned} \tag{19}$$

where $a, b, c \in \{0, \ldots, p-1\}$ are indices and $s_u, s_v, s_w \in \mathbb{R}$ are phase shifts satisfying $s_u + s_v \equiv s_w + s_x \pmod{2\pi}$. They achieve a maximal value of $\mathcal{U}_* = \sqrt{2p}|\hat{x}[\xi]|^3$.

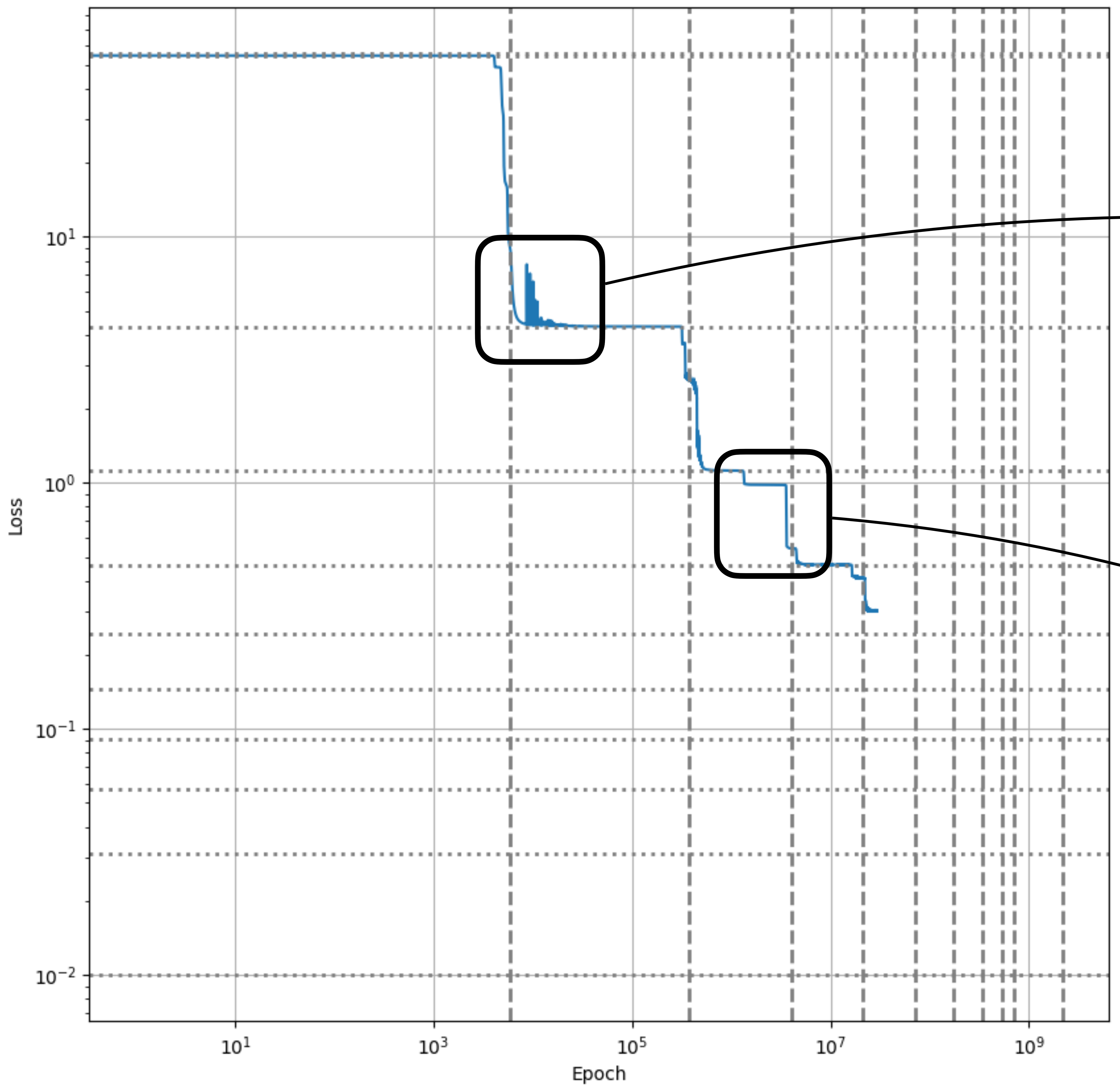
During cost minimization, multiple neurons collaborate

During the cost minimization step, the neurons grow in norm and specialize in their phase shifts.



With < 6 neurons the network cannot learn to remove the dominant frequency from the residual

Putting it together



1. Often large spikes and instability follow cost minimization — Adam smooths them.

2. Often the network consistently uses 6 neurons per frequency — hinting at a sparsity bias.

Future directions for AGF

0. Finish the pre-print.

1. Consider other theoretical settings for AGF, such as Multi-index model.

2. Extending AGF to deeper networks.

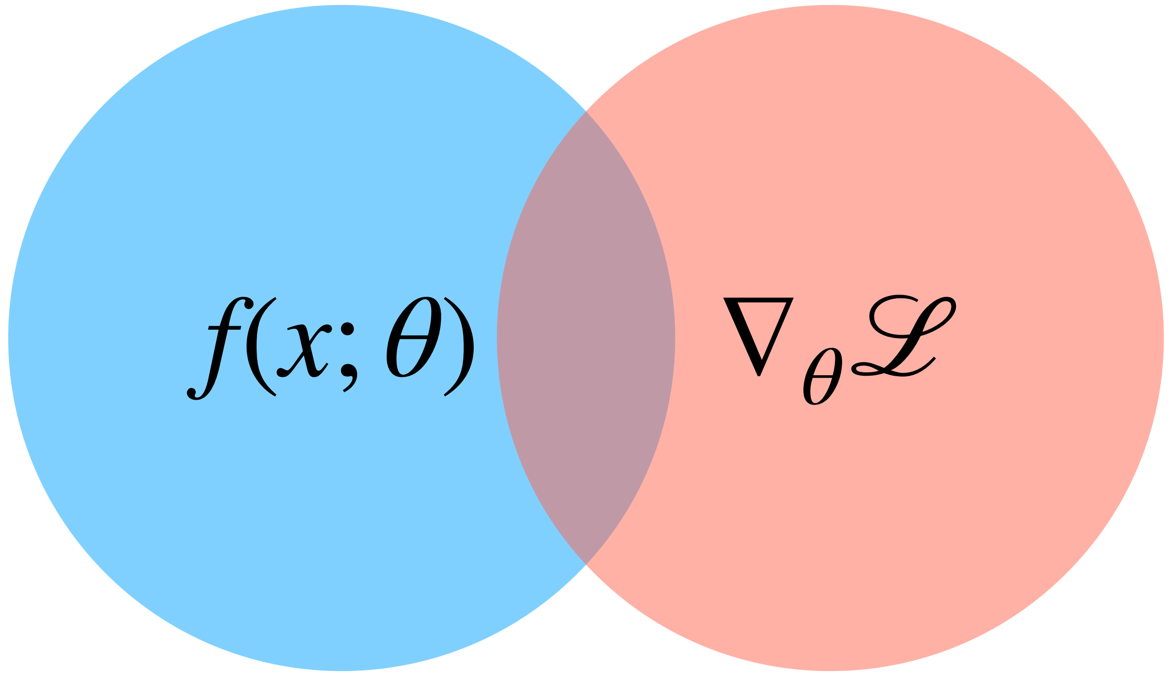
3. Using AGF as an alternative optimizer to SGD.

Part 1: What conditions enable feature learning?

When and why does feature learning emerge.

When: Small-scale initializations where the NTK evolves

Why: Saddle-to-saddle dynamics with fast directions and slow norm

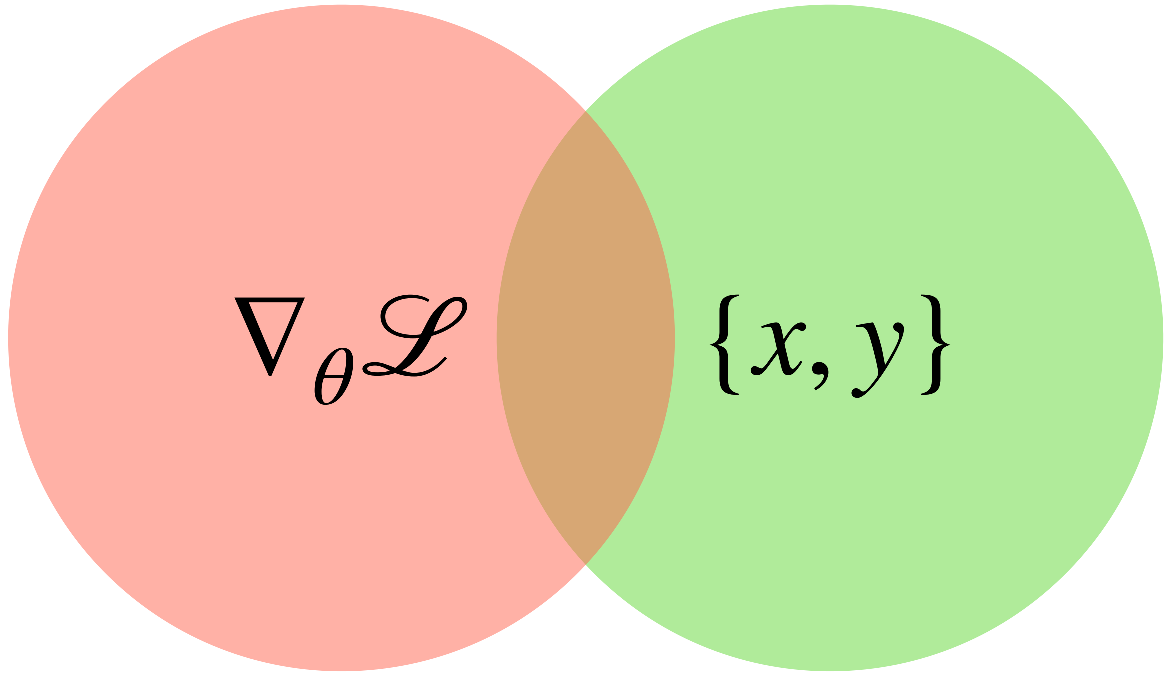


Part 2: What mechanisms drive feature learning?

What features do neural networks learn, and how.

What: Directions that maximize the utility function.

How: Through an iterative maximization-minimization process



Thank you!

kunin@stanford.edu