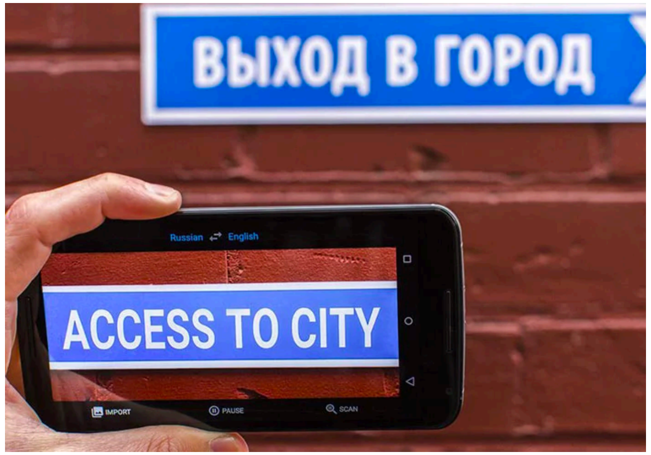
Understanding Neural Networks and its Roles in Prioritized Search

Yuandong Tian

Research Scientist and Manager
Facebook AI Research

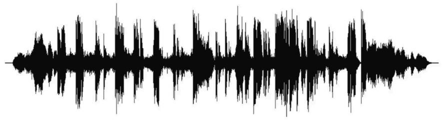
facebook Artificial Intelligence

Great Empirical Success from Deep Models

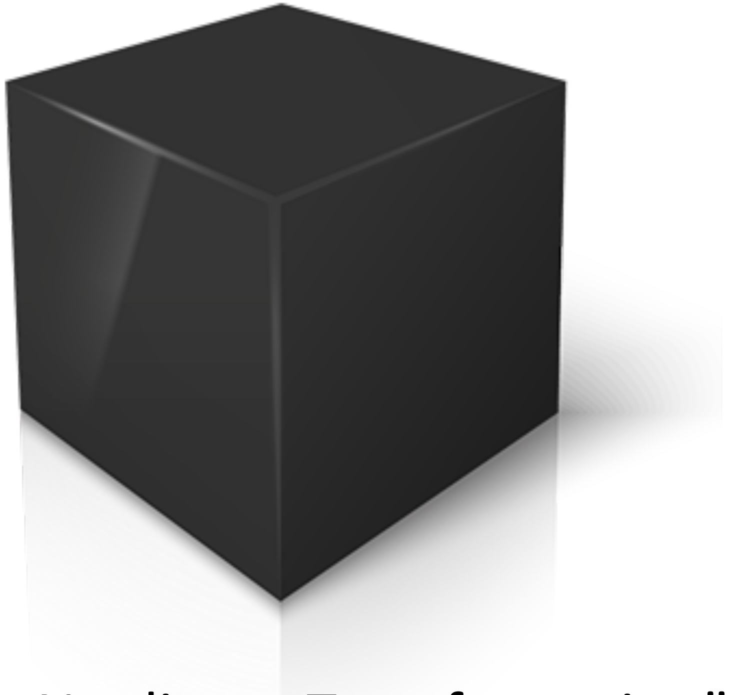


How do deep models work?

Input



This is an apple



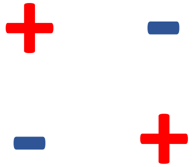
"Some Nonlinear Transformation"

Output

Three Major Problems

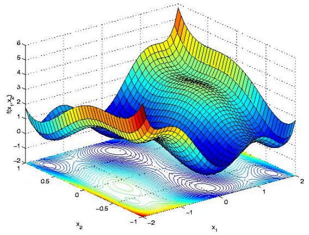
Understanding how Deep Models work

Expressibility



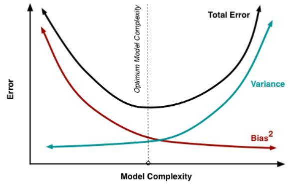
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

Optimization



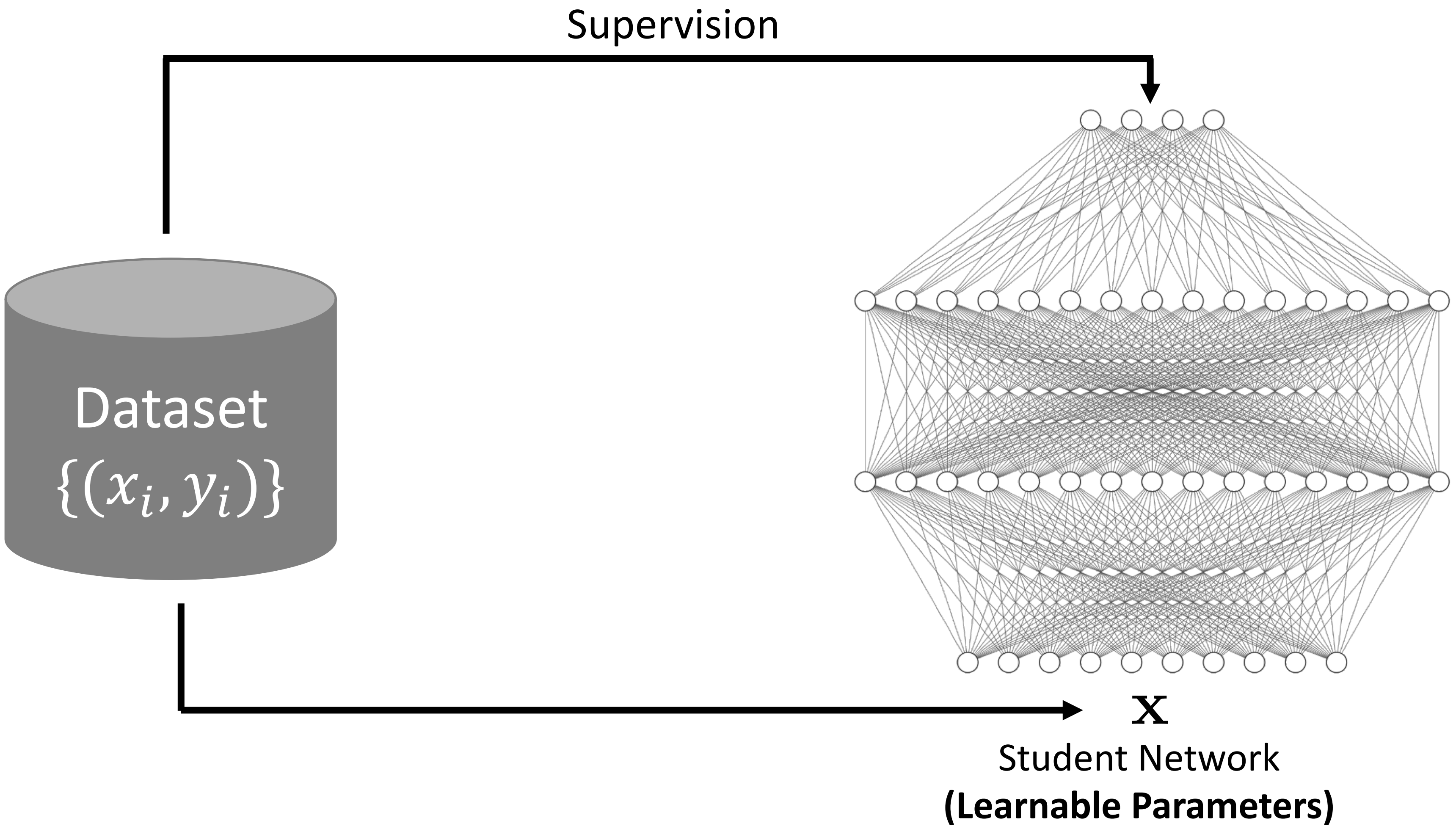
"Gradient vanishing/exploding"
"Gradient Descent might get stuck at saddle point / local minima"
"Can GD/SGD go to global optima? How fast?"

Generalization



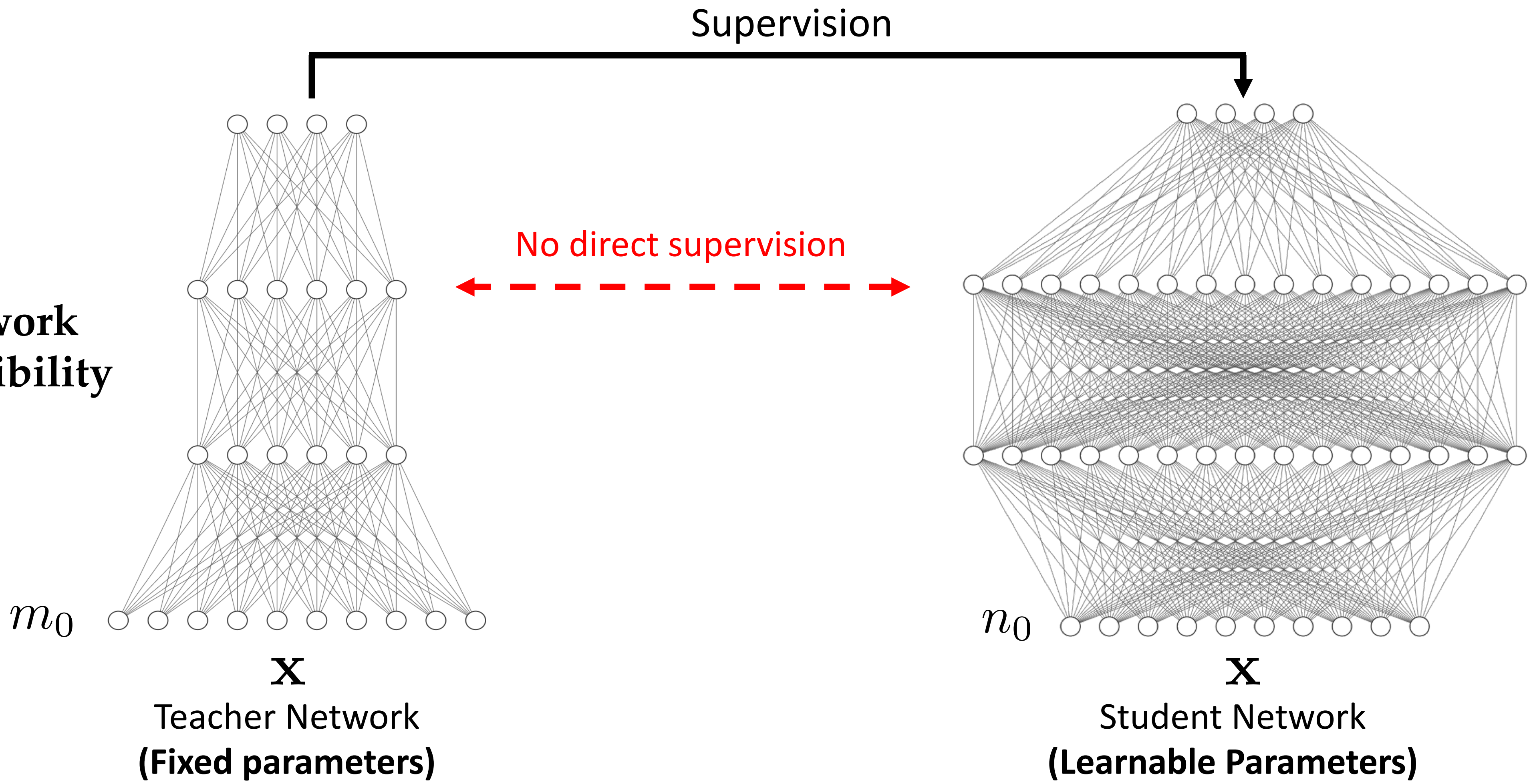
"Does zero training error often lead to overfitting?"
"More parameters might lead to overfitting."

Supervised Learning



Student-Teacher Setting

By Network Expressibility



Why Student-Teacher Setting?

Understanding how Deep Models work

Expressibility

Provide a target function with bounded complexity

Optimization

Study fine dynamics behaviors by comparing with teacher

Generalization

Weight alignment with the teacher yields generalization

Old History of Teacher-Student Setting

$$\epsilon(\boldsymbol{J}) = \frac{1}{2}\langle |f(\boldsymbol{J},\boldsymbol{\xi}) - f(\boldsymbol{B},\boldsymbol{\xi})|^2\rangle_{\xi} \qquad f(\boldsymbol{J},\boldsymbol{\xi}) = \sum_{i=1}^{K} \sigma(\boldsymbol{J}_i \cdot \boldsymbol{\xi})$$

Study when the input dimension $n_0 = m_0 \to +\infty$ (i.e., **thermodynamics limits**)

In some situations, student nodes are "***specialized***" to teacher node

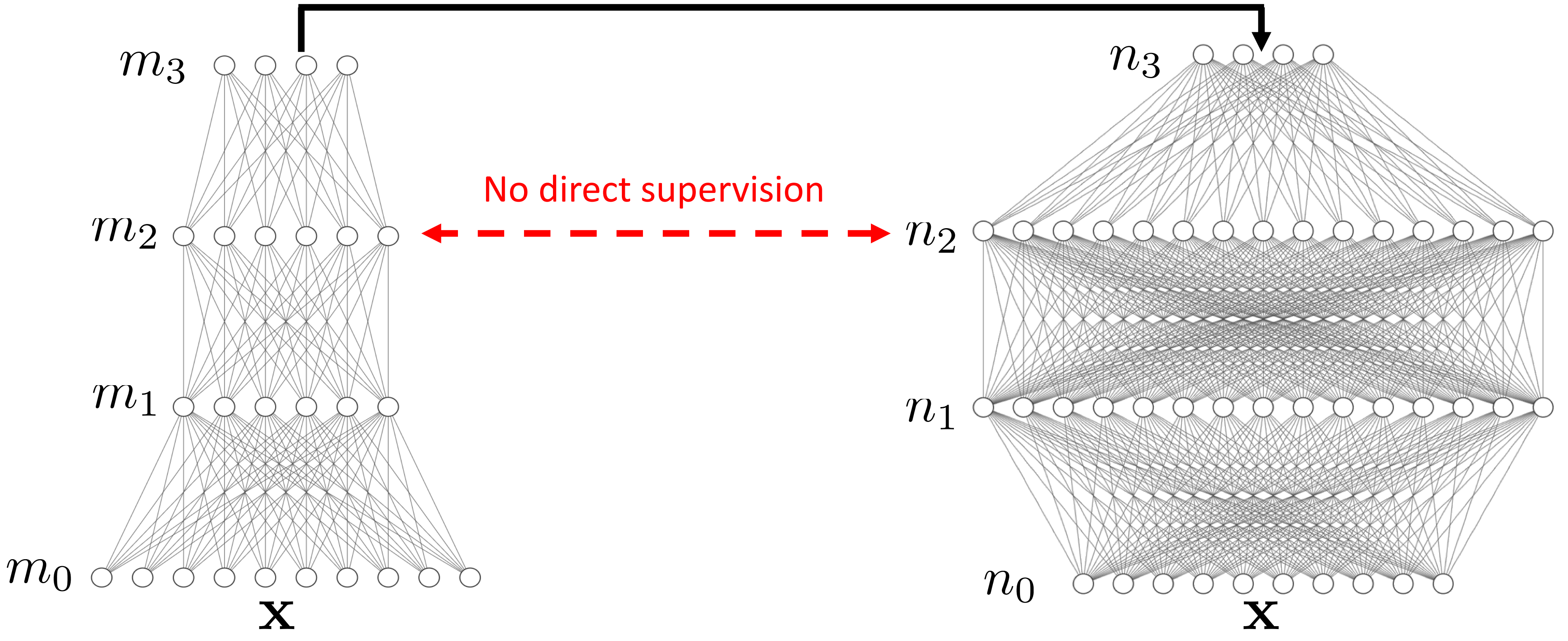
One layer of trainable parameters
Nonlinear function $\sigma(x) = \mathrm{erf}(x\ /\ 2)$
Locally linearized analysis around symmetry breaking plane and final solution

[On-line learning in soft committee machines, Saad & Solla, Phys. Rev 1995]

Proposed Setting

1. Finite m_0 and n_0
2. Works for $n_i \geq m_i$
(no crazy overparameterization)

$$\min_{\mathcal{W}} J(\mathcal{W}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}}\left[\|\mathbf{f}_L^*(\mathbf{x}) - \mathbf{f}_L(\mathbf{x})\|^2\right]$$



Teacher Network
(Fixed parameters)

(Over-parameterized) Student Network
(Learnable Parameters)

Arxiv: https://arxiv.org/abs/1909.13458

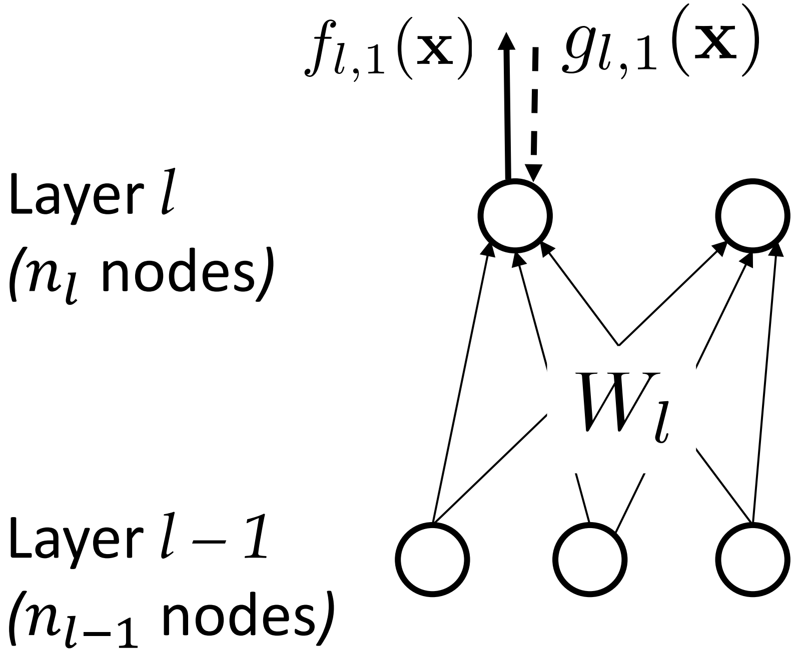
Main Question

Question: With over-parameterized student network:

Small gradient during training ⟶ ? ? ? ⟶ Student aligns with the teacher

→ Small training error potentially leads to good generalization

Notation



Activation

$$\mathbf{f}_l(\mathbf{x}) = \begin{bmatrix} f_{l,1}(\mathbf{x}) \\ f_{l,2}(\mathbf{x}) \end{bmatrix}$$

Gradient

$$\mathbf{g}_l(\mathbf{x}) = \begin{bmatrix} g_{l,1}(\mathbf{x}) \\ g_{l,2}(\mathbf{x}) \end{bmatrix}$$

Weight update rule: $\dot{W}_l = \mathbb{E}_{\mathbf{x}}\left[\mathbf{f}_{l-1}(\mathbf{x})\mathbf{g}_l^{\intercal}(\mathbf{x})\right]$

GD: expectation taken over the entire dataset

SGD: expectation taken over a batch

Lemma1: Recursive Gradient Rule

For layer l, there exists $A_l(x)$ and $B_l(x)$ so that:

$$\underbrace{\mathbf{g}_l(\mathbf{x})}_{\text{Student gradient}} = \underbrace{D_l(\mathbf{x})}_{\text{Student gating}} \left[\underbrace{A_l(\mathbf{x})\mathbf{f}_l^*(\mathbf{x})}_{\text{Teacher mixture}} - \underbrace{B_l(\mathbf{x})\mathbf{f}_l(\mathbf{x})}_{\text{Student mixture}} \right]$$

$A_l(x)$ and $B_l(x)$ are **piece-wise constant.**

Lemma1: Recursive Gradient Rule

$$D_l(\mathbf{x}) \in \mathbb{R}^{n_l \times n_l}$$
$$A_l(\mathbf{x}) \in \mathbb{R}^{n_l \times m_l}$$
$$B_l(\mathbf{x}) \in \mathbb{R}^{n_l \times n_l}$$

For layer l, there exists $A_l(x)$ and $B_l(x)$ so that:

n_l: number of student nodes at layer l

m_l: number of teacher nodes at layer l

$$\mathbf{g}_l(\mathbf{x}) = D_l(\mathbf{x})\left[A_l(\mathbf{x})\mathbf{f}_l^*(\mathbf{x}) - B_l(\mathbf{x})\mathbf{f}_l(\mathbf{x})\right]$$

Student gradient

Student gating

Teacher mixture

Student mixture

$$\mathbf{f}_l^*(\mathbf{x}) \in \mathbb{R}^{m_l}$$
$$\mathbf{f}_l(\mathbf{x}) \in \mathbb{R}^{n_l}$$
$$\mathbf{g}_l(\mathbf{x}) \in \mathbb{R}^{n_l}$$

$A_l(x)$ and $B_l(x)$ are **piece-wise constant.**

$V_l(\mathbf{x}) \in \mathbb{R}^{C\times n_l}$

$V_l^*(\mathbf{x}) \in \mathbb{R}^{C\times m_l}$

C: output dimension

Recursive Formula for $A_l(x)$ and $B_l(x)$

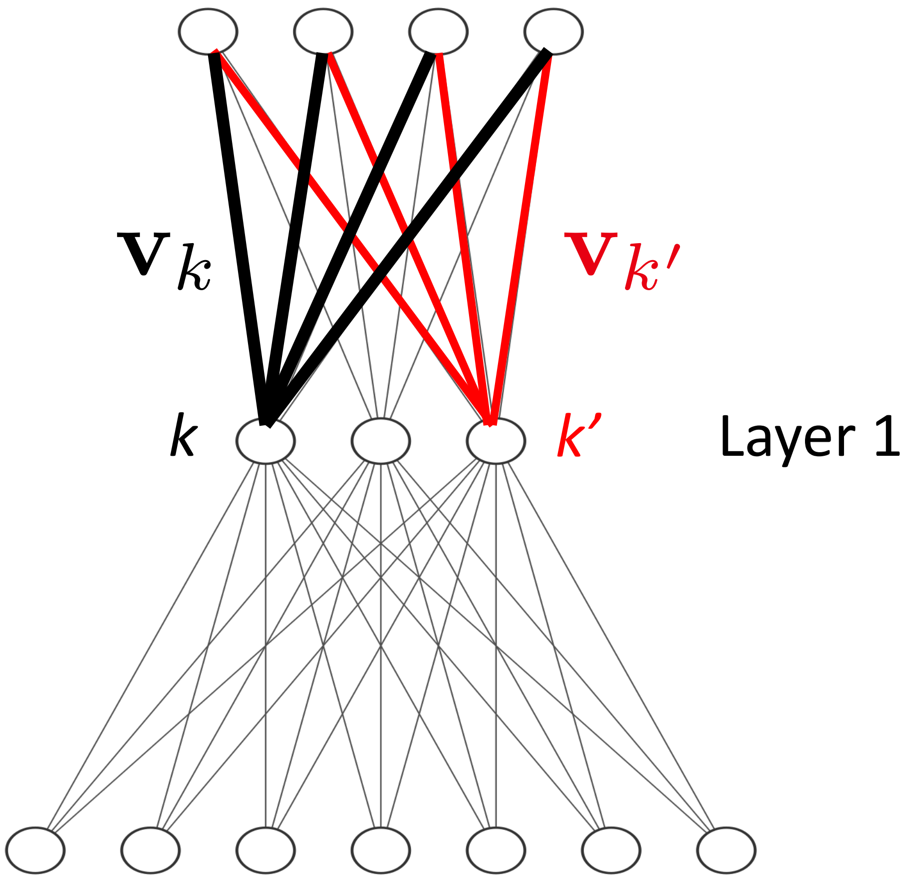
$$A_l(\mathbf{x}) = V_l^{\intercal}(\mathbf{x})V_l^*(\mathbf{x})$$
$$B_l(\mathbf{x}) = V_l^{\intercal}(\mathbf{x})V_l(\mathbf{x})$$

Recursive Formula for *V*:

$$V_{l-1}^*(\mathbf{x}) = V_l^*(\mathbf{x})D_l^*(\mathbf{x})W_l^{*\intercal}$$
$$V_{l-1}(\mathbf{x}) = V_l(\mathbf{x})D_l(\mathbf{x})W_l^{\intercal}$$

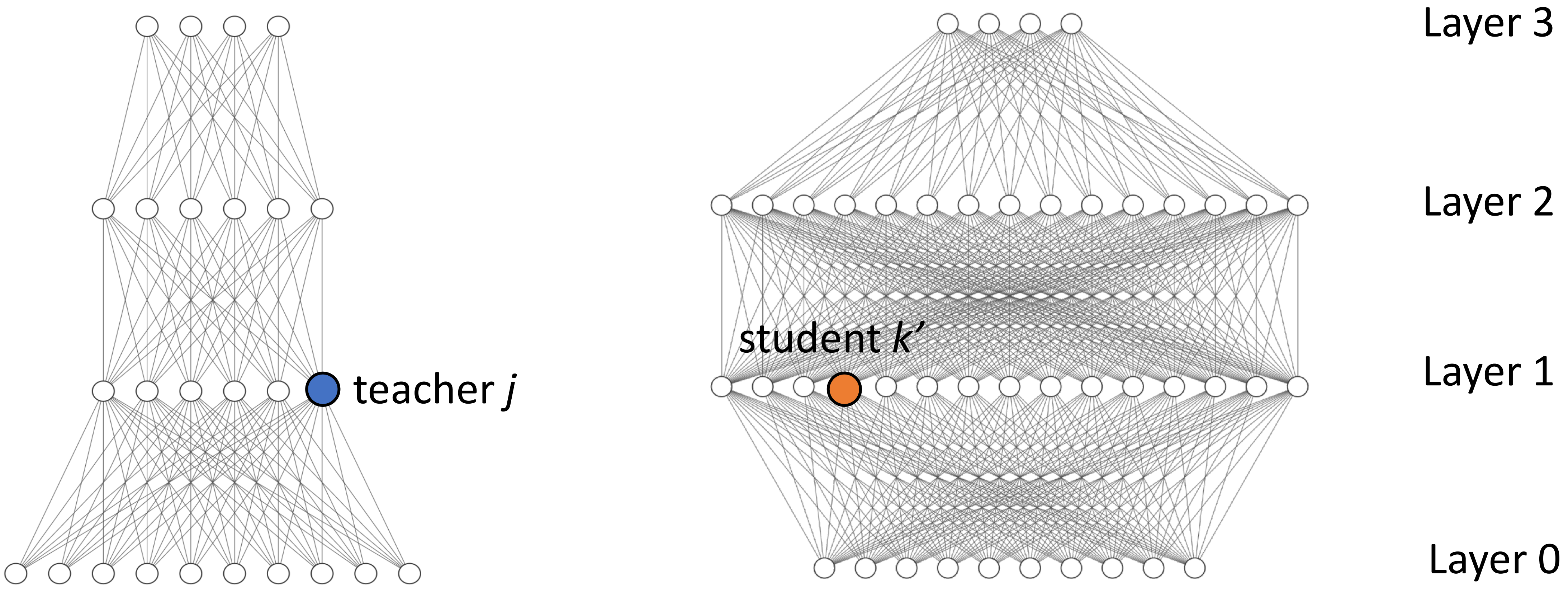
Base case:

$$V_L(\mathbf{x}) = V_L^*(\mathbf{x}) = I_{C\times C}$$

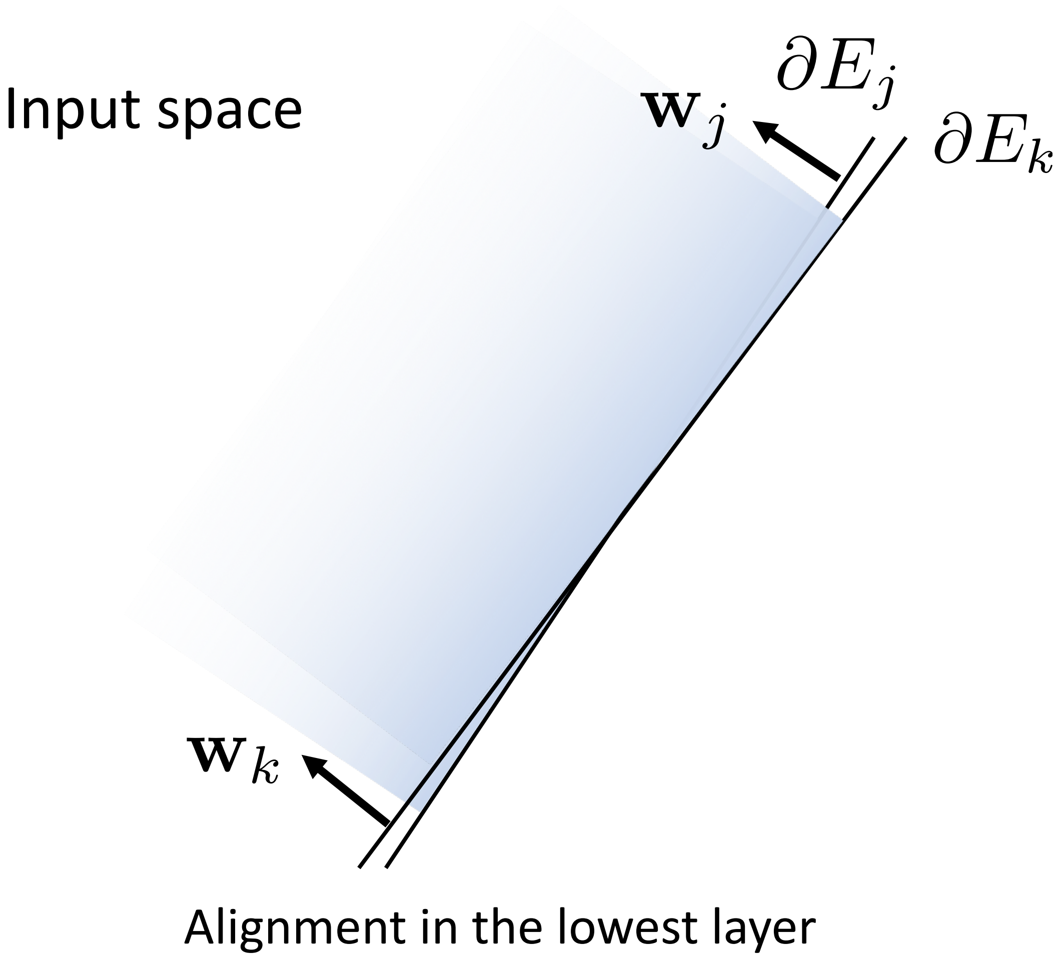


$$V_1 = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_1}]$$

Main results: Alignment could happen!



Definition of Alignment

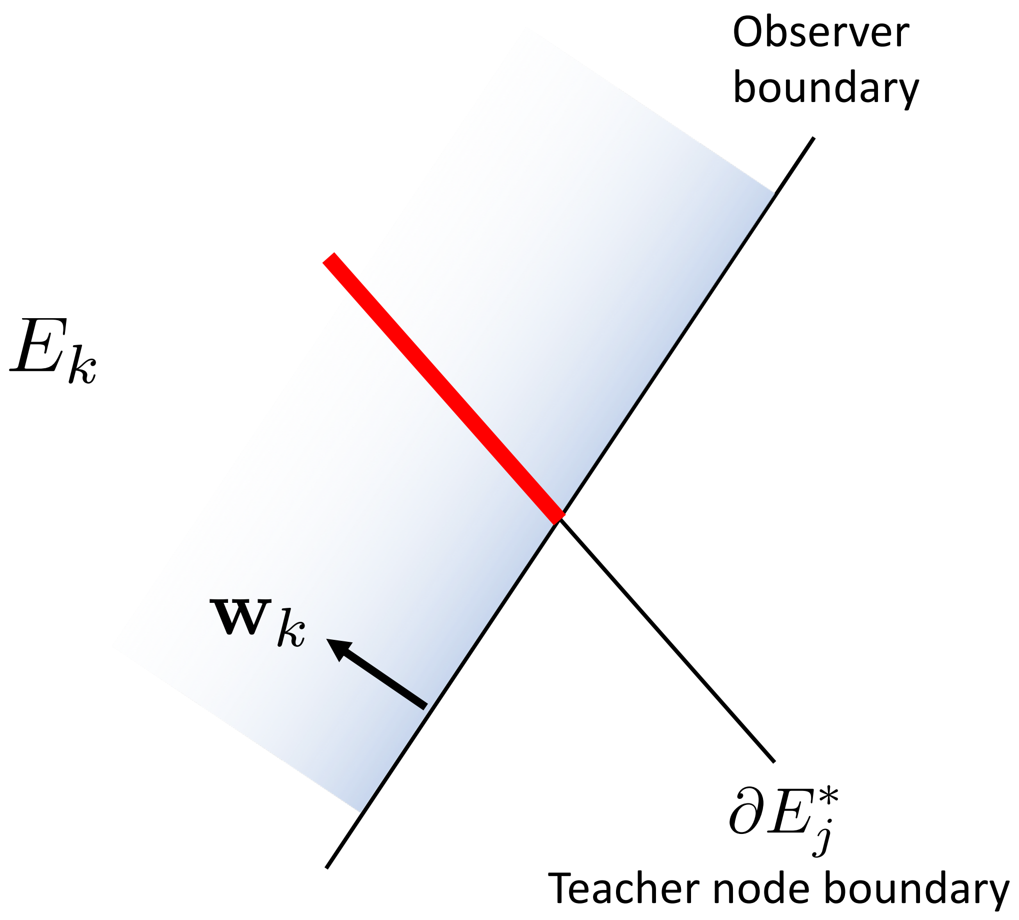


E_j Activated Region of node *j*

∂E_j Boundary of node *j*

∂E_k Boundary of node *k*

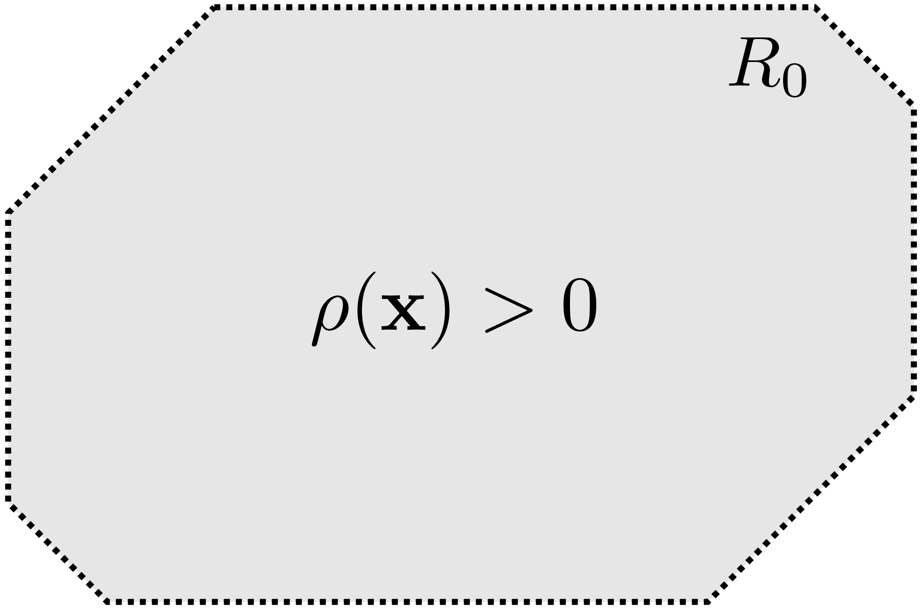
Definition of “Observation”



$$\partial E_j^* \cap E_k \neq \emptyset$$

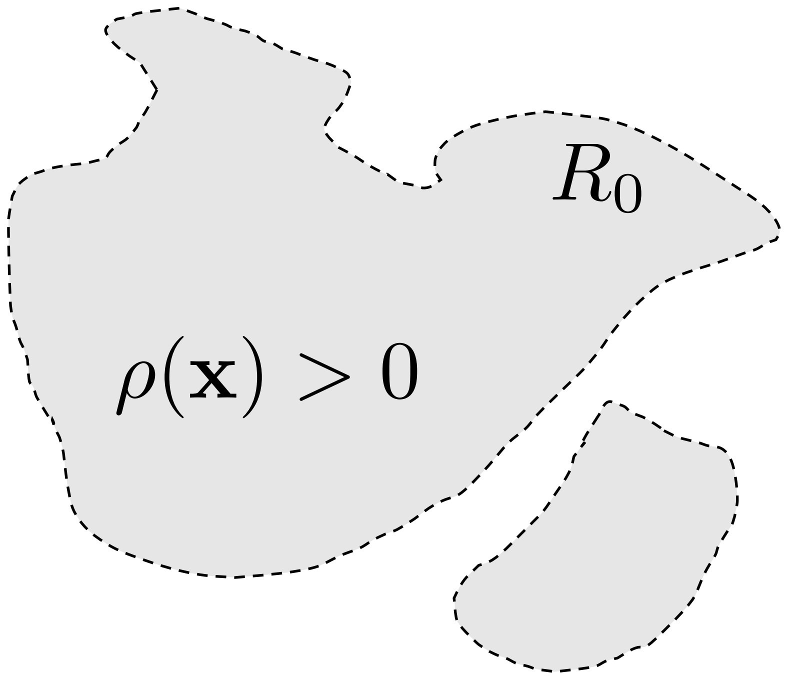
Teacher *j* is **observed** by a student *k*

Assumption of the dataset



Infinite dataset!

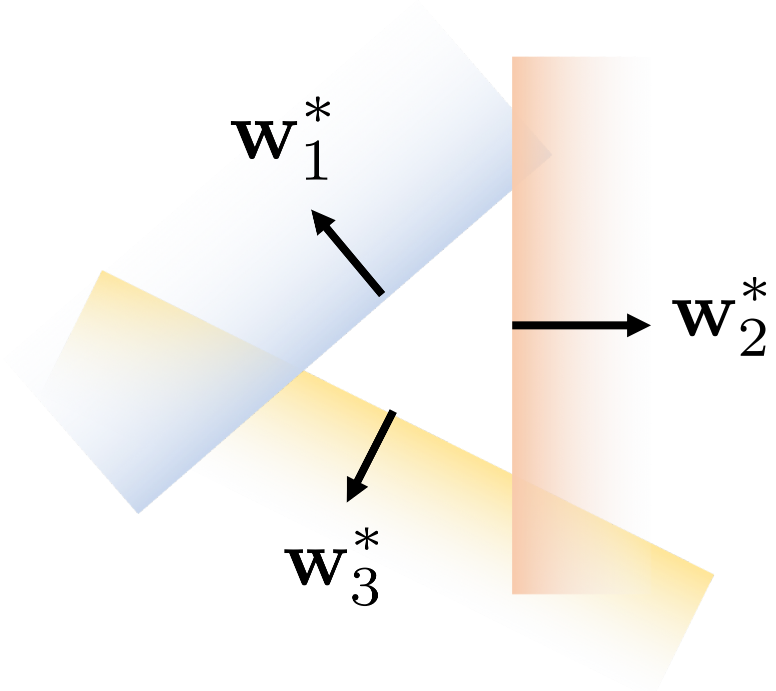
Assumption of the dataset



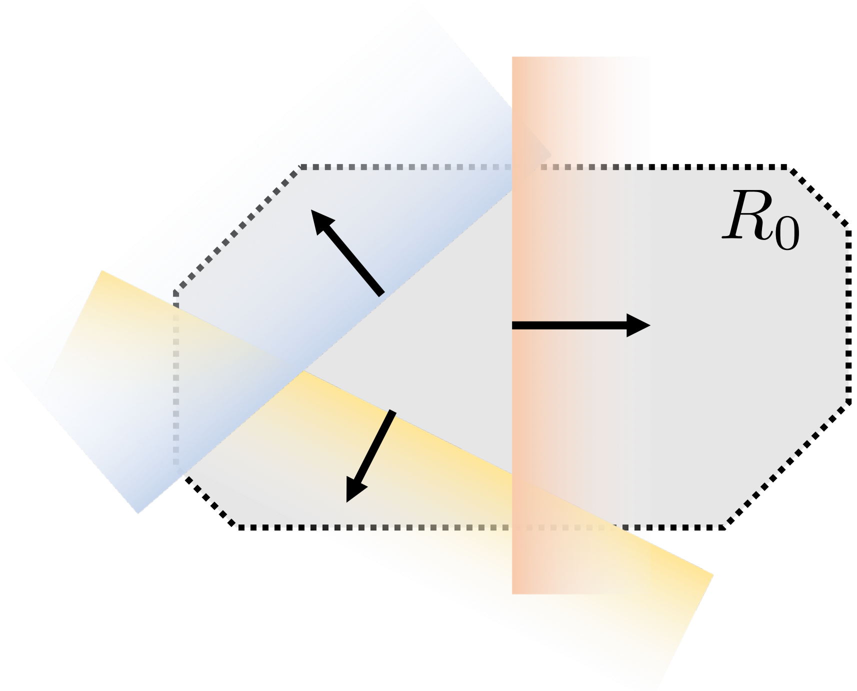
Infinite dataset! (Region needs to have interiors)

Assumptions on Teacher Network

- Cannot reconstruct arbitrary teachers
  - e.g., all ReLU nodes are dead

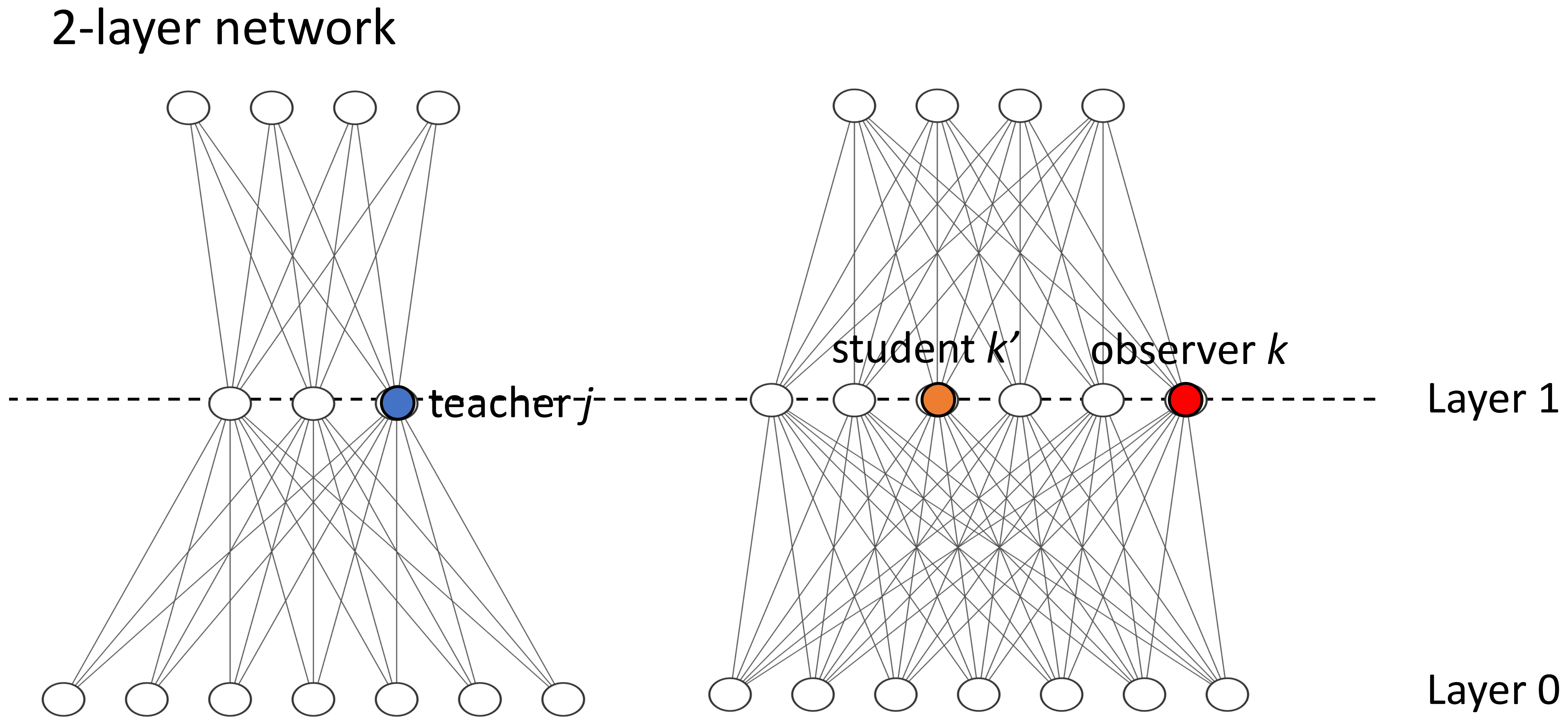


Distinct teacher nodes



Teacher's boundary are visible in the dataset

Main results: Alignment could happen!



Main results: Alignment could happen!

At the lowest layer:

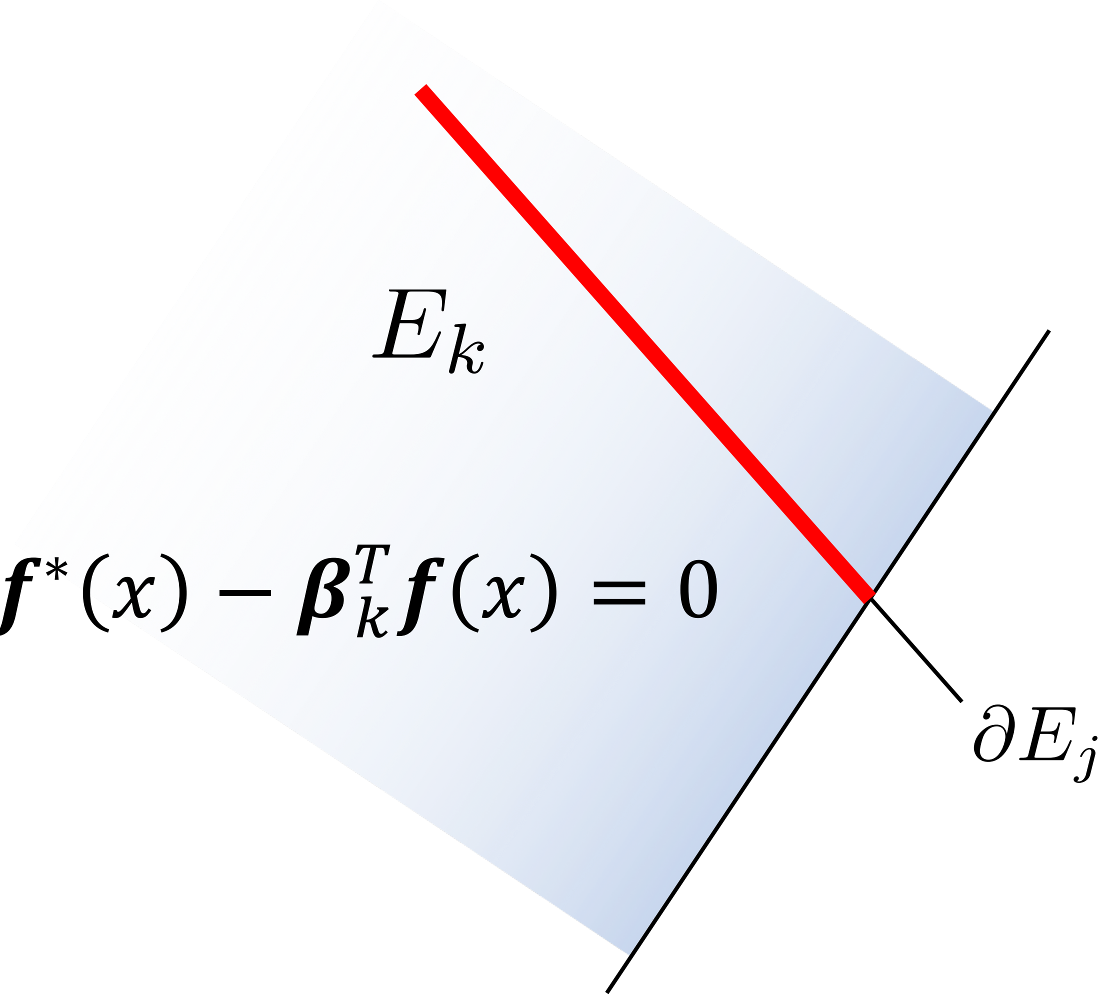
$\boldsymbol{g}_1(x) = \mathbf{0}$ for all $x \in R_0$
(all input gradients at layer 1 is *zero* everywhere)

Teacher node *j* is **observed** by a student node *k*

Teacher *j* is **aligned with** at least one student *k'*

Why?

The gradient of observer *k* is 0:



From Lemma 1, $g_k(x) = \boldsymbol{\alpha}_k^T \boldsymbol{f}^*(x) - \boldsymbol{\beta}_k^T \boldsymbol{f}(x) = 0$

If $x \in E_k$

Why?

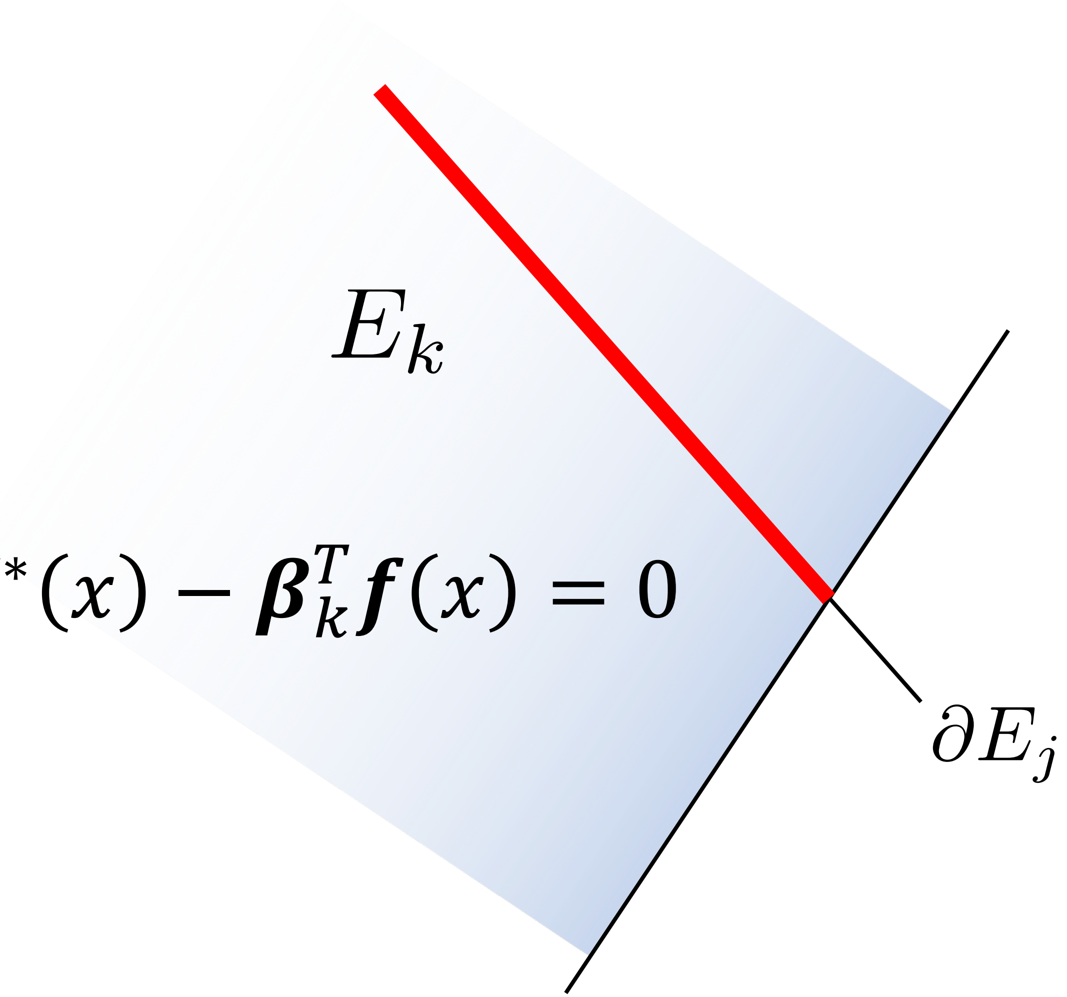
The gradient of observer *k* is 0:

From Lemma 1, $g_k(x) = \boldsymbol{\alpha}_k^T \boldsymbol{f}^*(x) - \boldsymbol{\beta}_k^T \boldsymbol{f}(x) = 0$

If $x \in E_k$

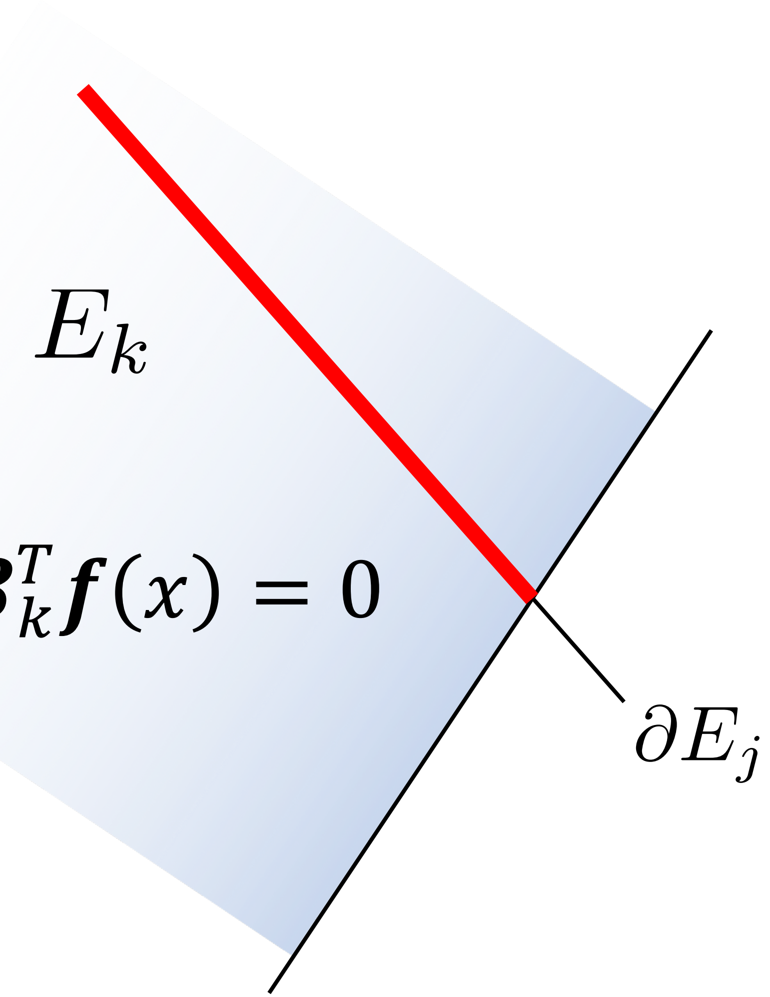
ReLUs are linear independent!

Coefficients for teacher *j* direction must be 0



Why?

The gradient of observer *k* is 0:



From Lemma 1, $g_k(x) = \boldsymbol{\alpha}_k^T \boldsymbol{f}^*(x) - \boldsymbol{\beta}_k^T \boldsymbol{f}(x) = 0$

If $x \in E_k$

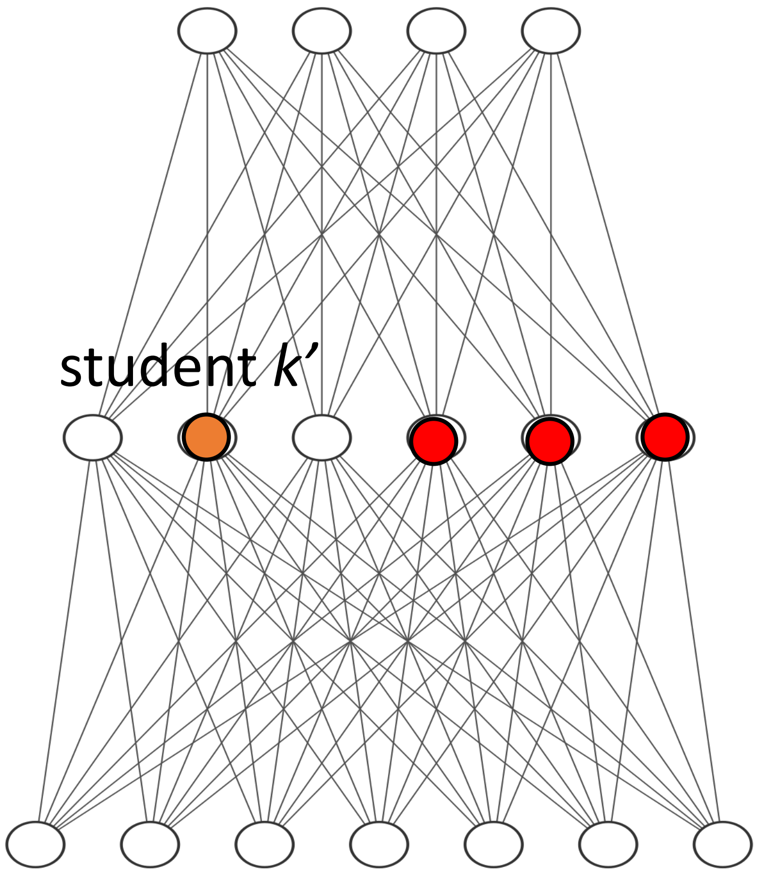
ReLUs are linear independent!

Coefficients for teacher *j* direction must be 0

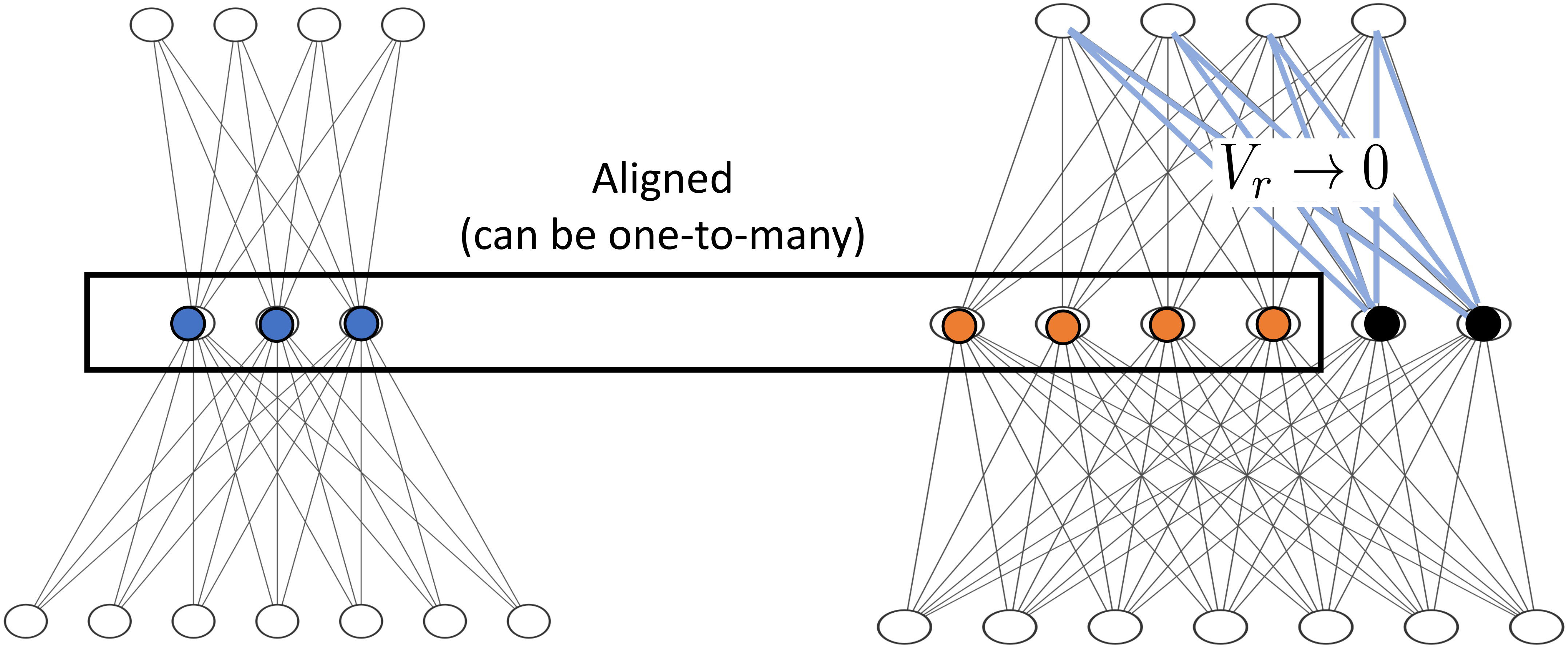
Teacher *j* is aligned with at least one student *k'* (sum of coefficients = 0)

Why Over-parameterization helps?

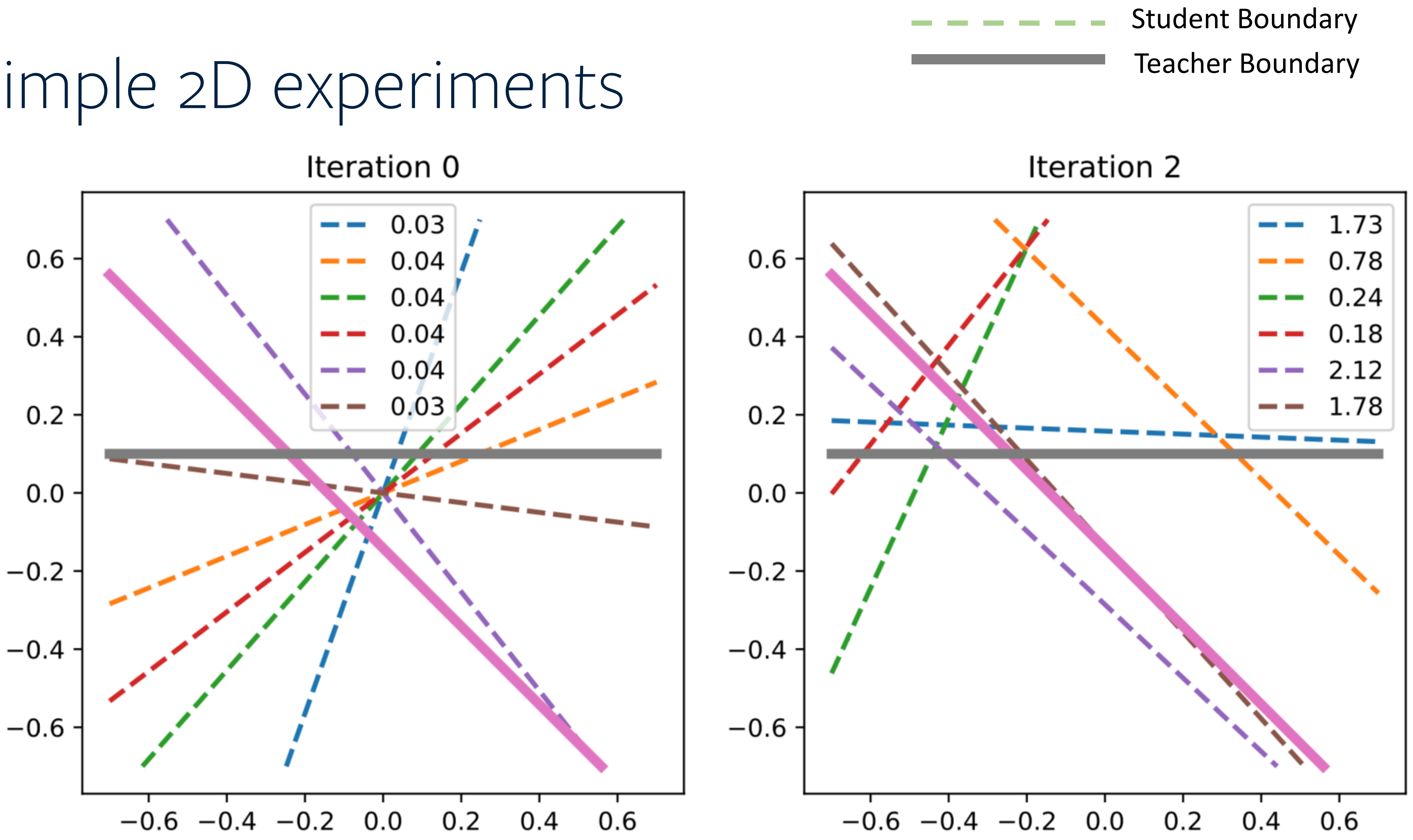
More observers!



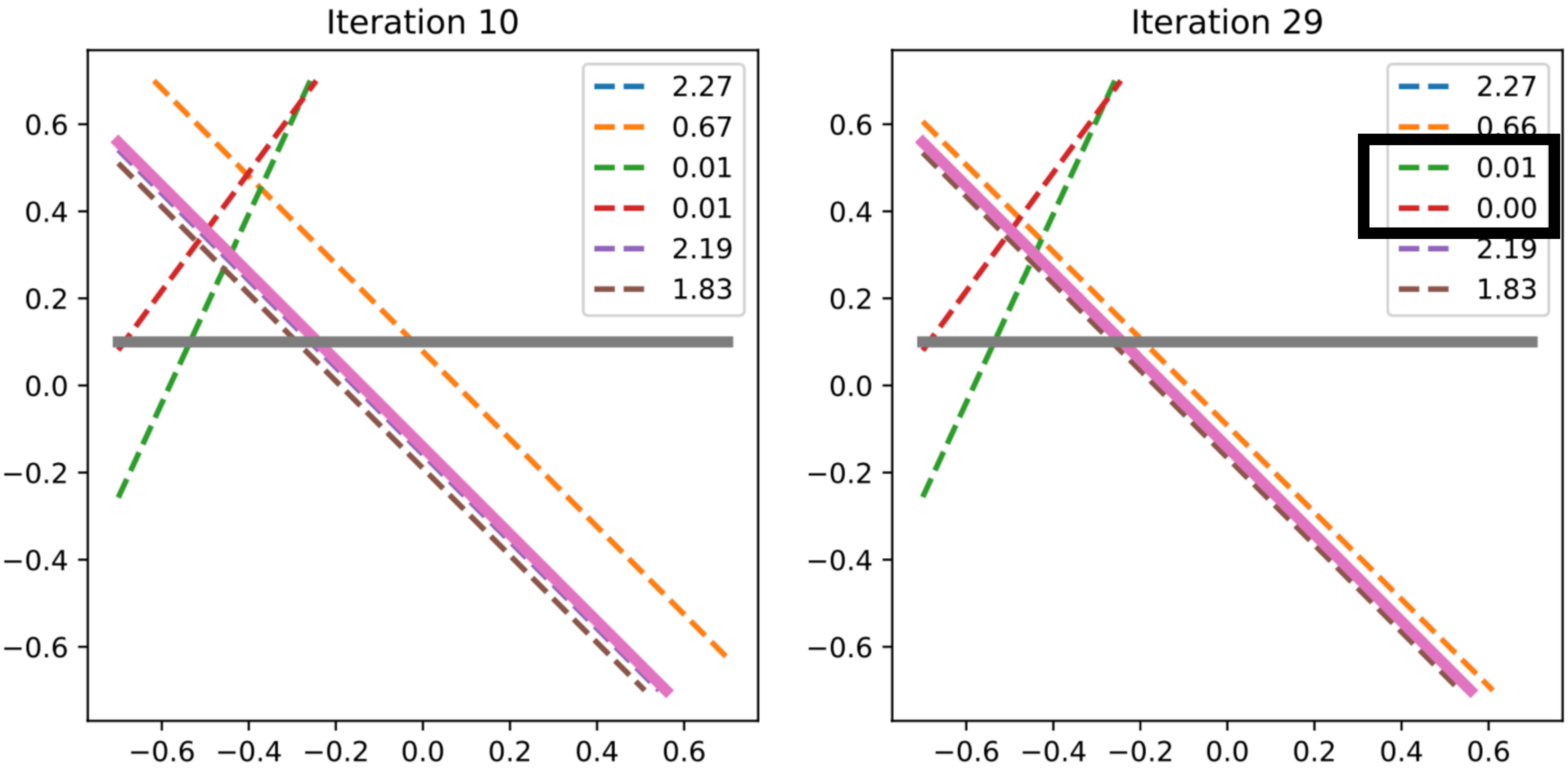
What happens to unaligned students?



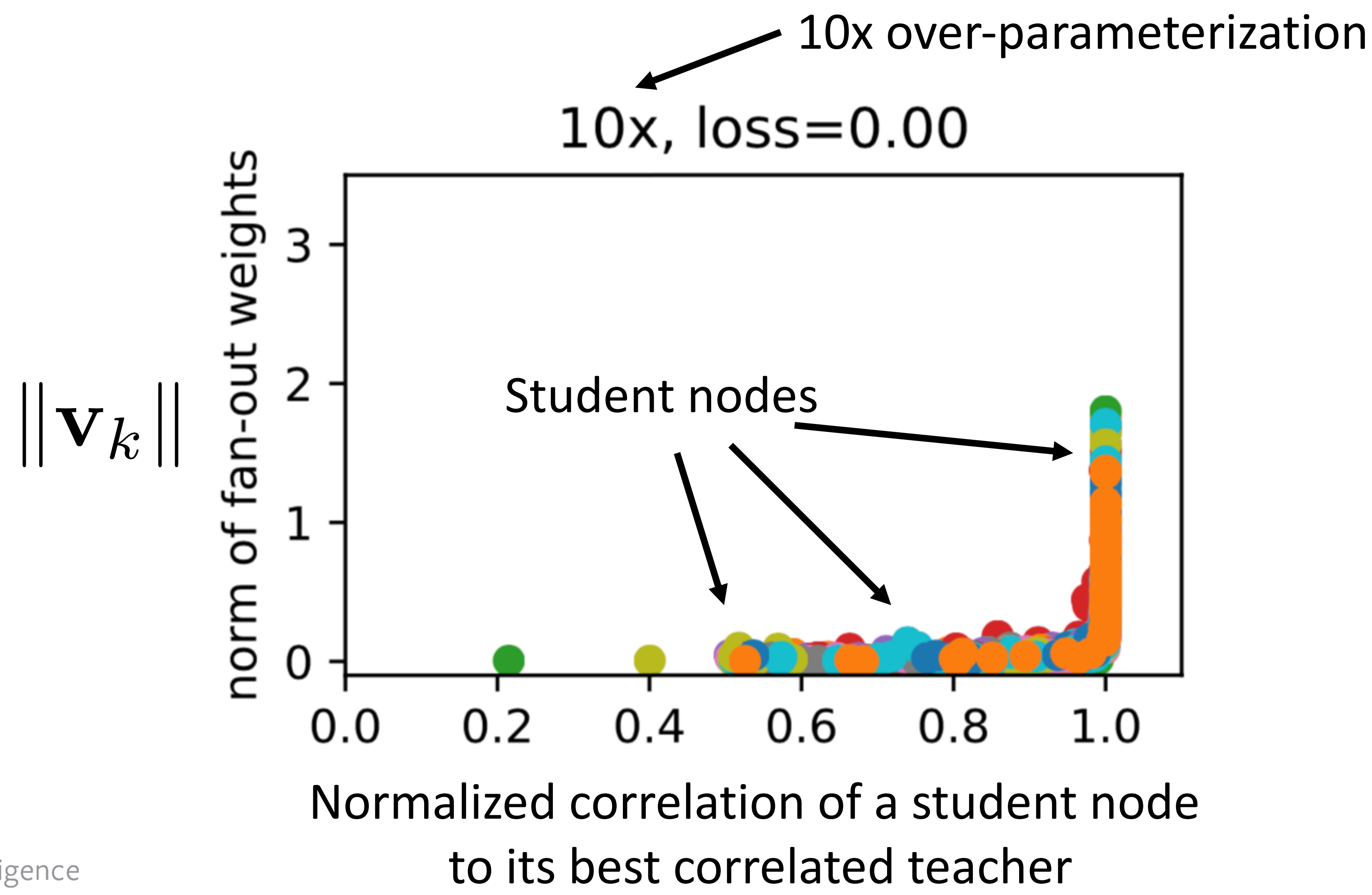
Simple 2D experiments



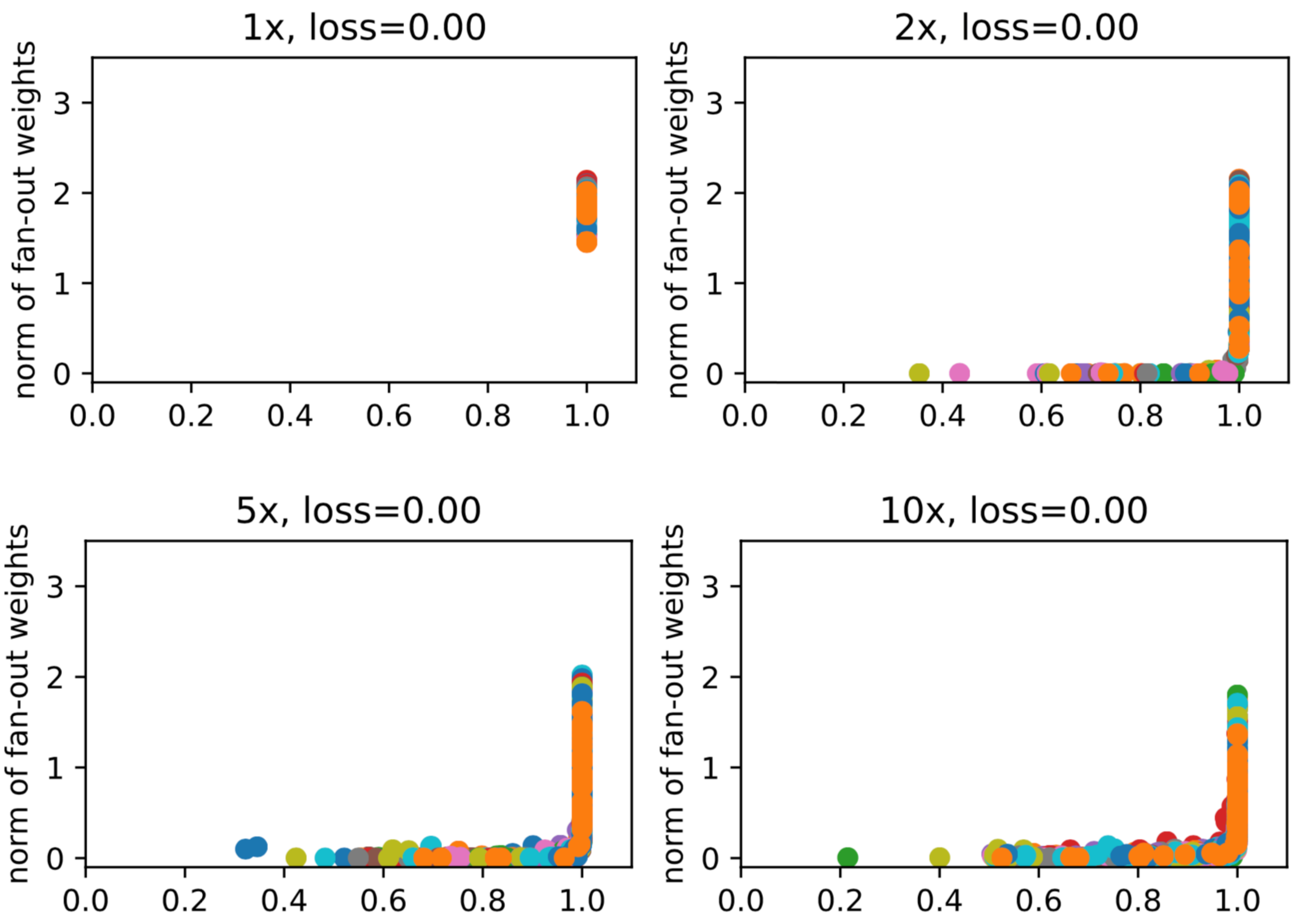
Simple 2D experiments



L-shape curve at convergence



L-shape curve at convergence

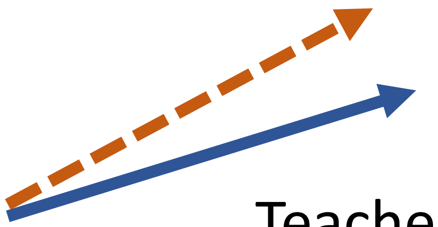


Noisy Case $\|\mathbf{g}_1(\mathbf{x}; \mathcal{W})\|_\infty \leq \epsilon$

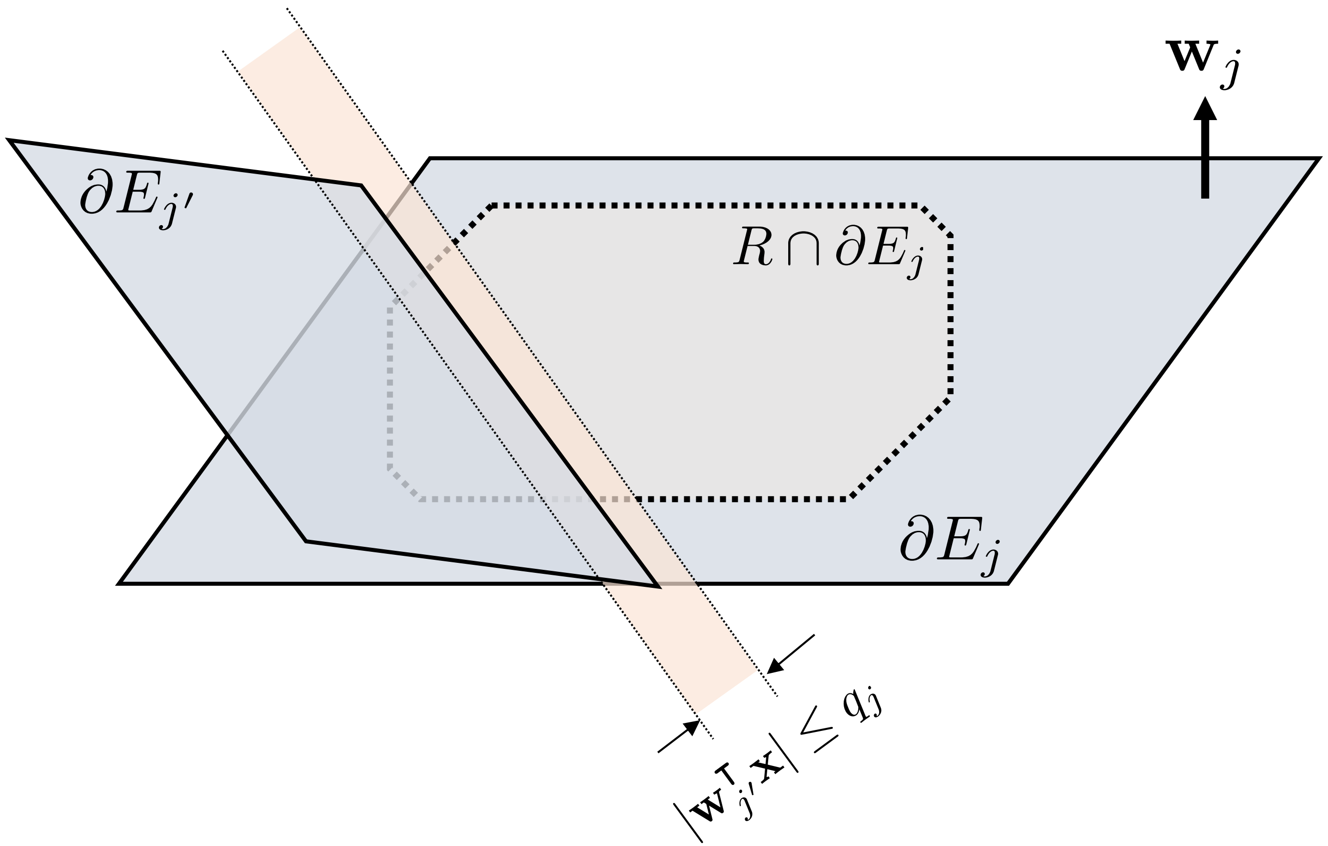
For teacher *j*, there exists student *k'*:

weights $$\sin\theta_{jk'} = \mathcal{O}\left(\frac{\epsilon^{1-\delta}}{|\alpha_{kj}|}\right)$$

bias $$|b_j^* - b_{k'}| = \mathcal{O}\left(\frac{\epsilon^{1-2\delta}}{|\alpha_{kj}|}\right)$$

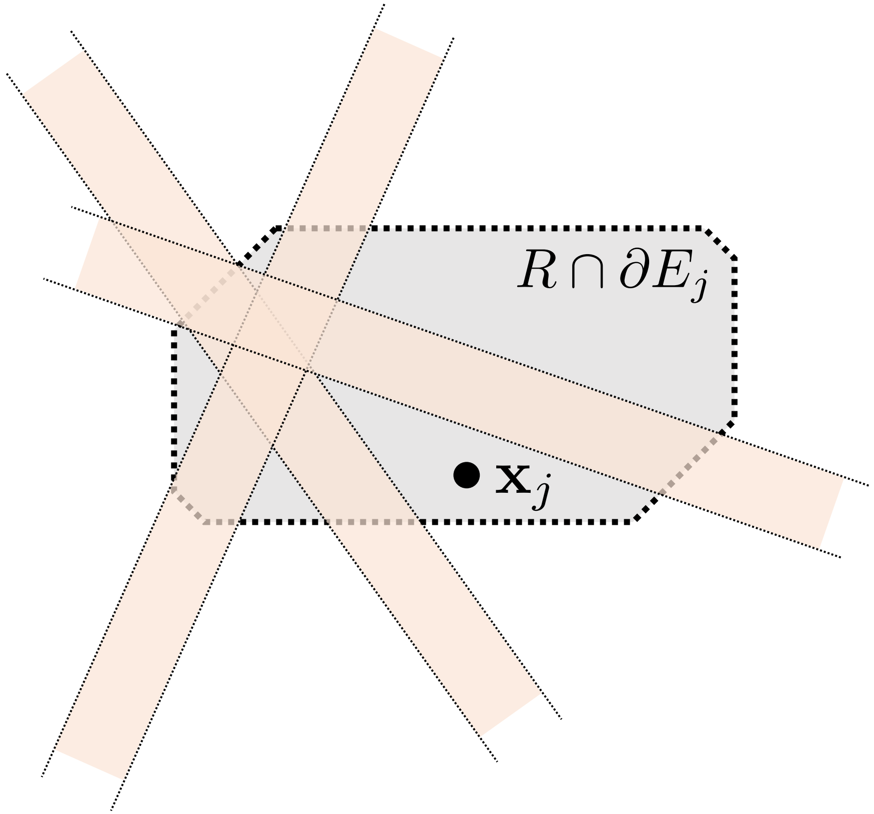


How to Prove?



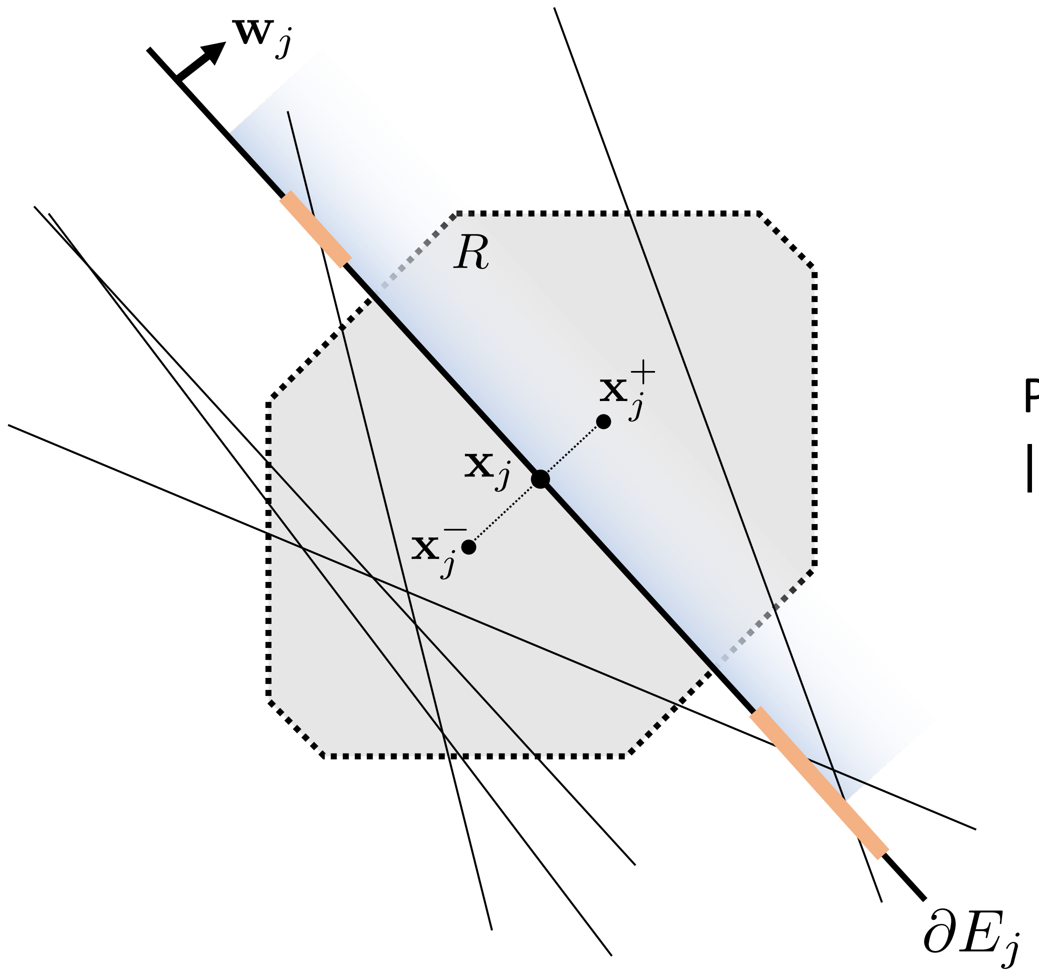
Misalignment leads to small overlap

How to Prove?



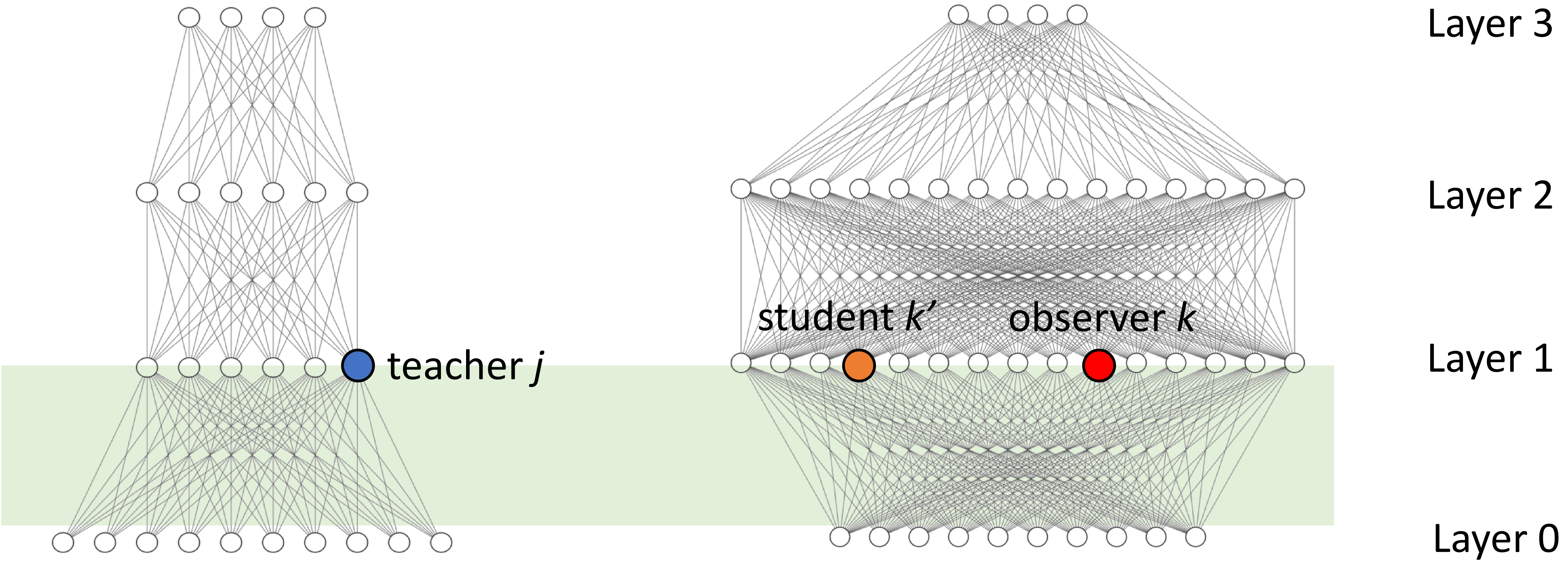
Small overlap → There exists a datapoint that is far away from all boundaries.

How to Prove?



Pick three points x_j, x_j^+, x_j^- and there will be one with $|g_j(x)| > \epsilon$, which is a contradiction.

Multi-Layer case: Alignment could happen!



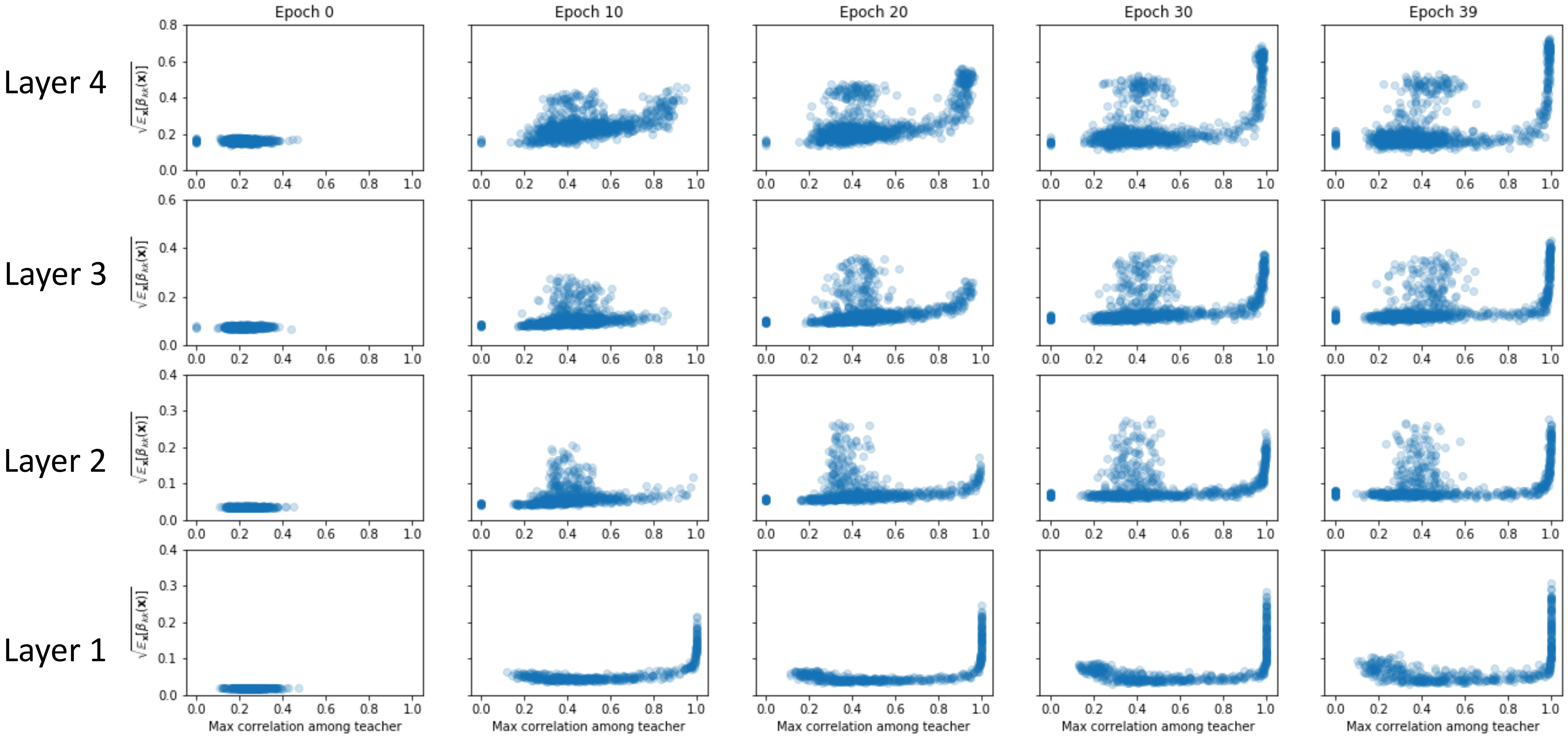
$$\boldsymbol{\alpha}_k^\intercal(\mathbf{x})\mathbf{f}^*(\mathbf{x}) - \boldsymbol{\beta}_k^\intercal(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

Piece wise constant, apply the same logic **per region!**

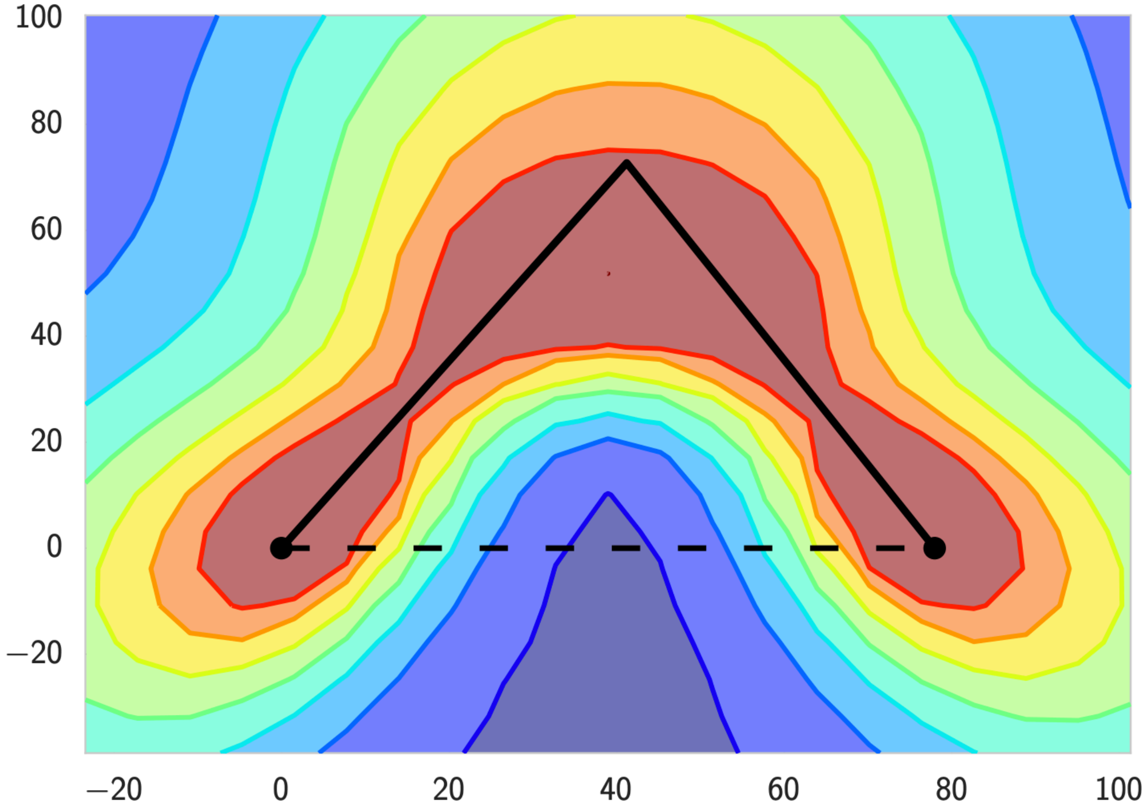
For 2-layer:

$$\sqrt{\mathbb{E}_{\mathbf{x}}\left[\beta_{kk}(\mathbf{x})\right]} = \|\mathbf{v}_k\|$$

Training Progresses



Solutions can be connected by line segments

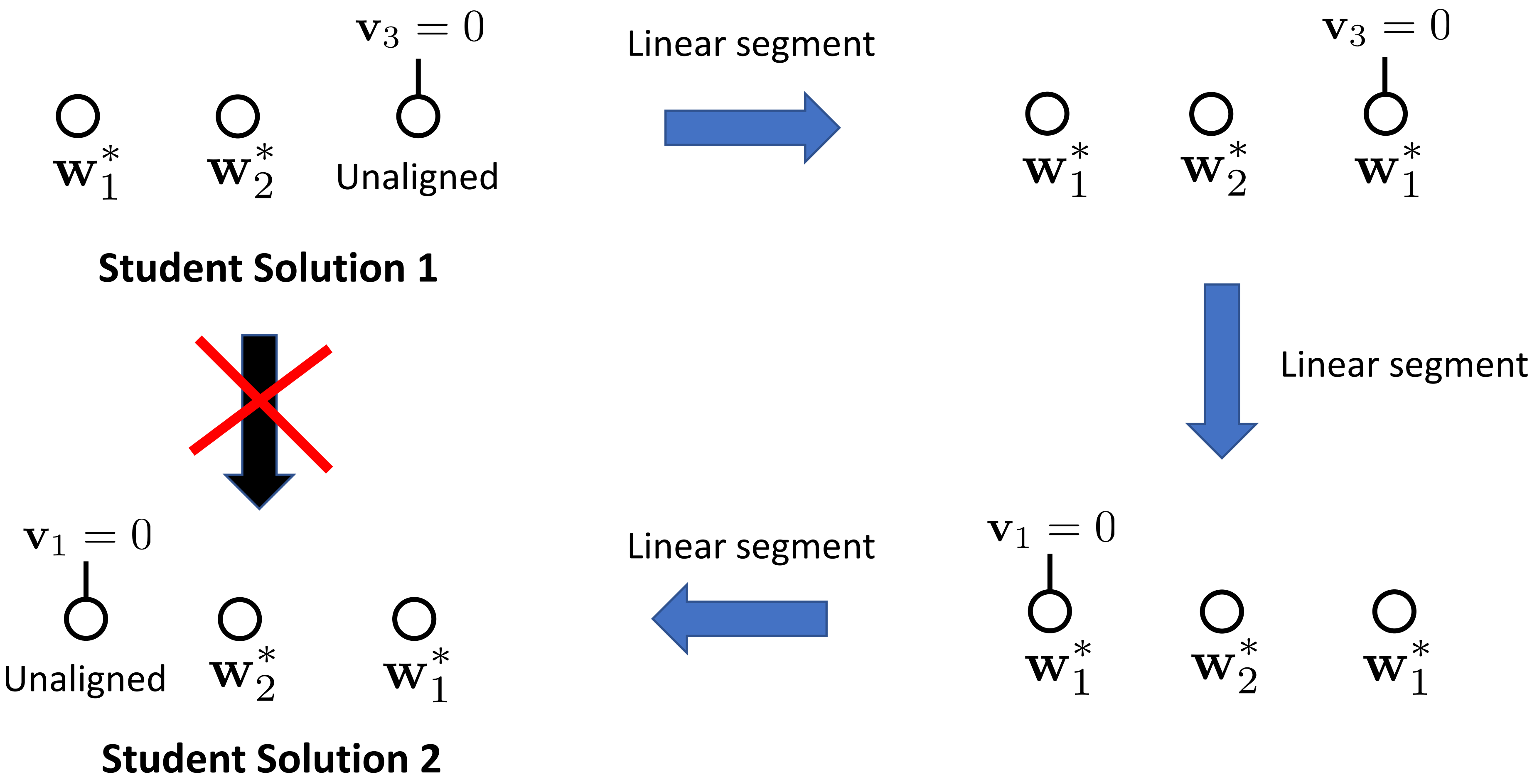


[Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, Garipov et al. NeurIPS 2018]

[Essentially No Barriers in Neural Network Energy Landscape, Draxler et al, 2018]

[Explaining Landscape Connectivity of Low-cost Solutions for Multilayer Nets, Kuditipudi et al, 2019]

Our Explanation



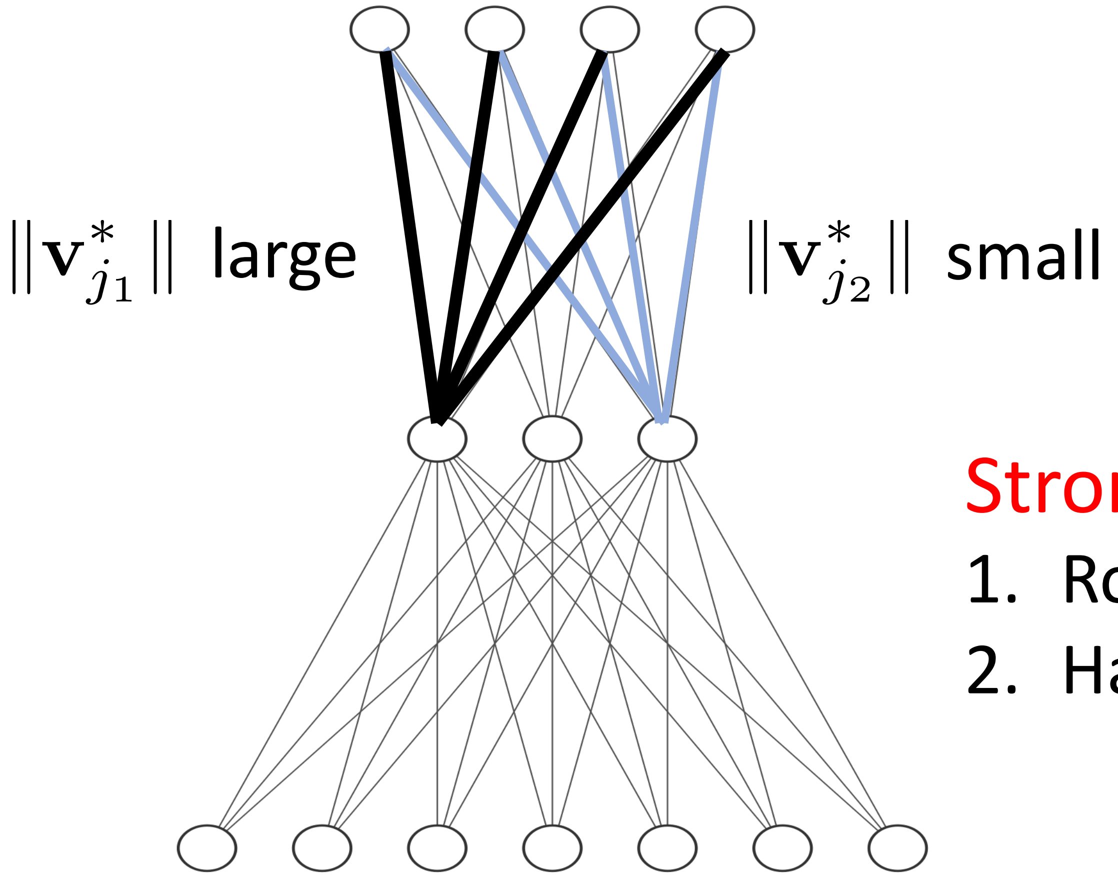
Training Dynamics

Critical Points have nice properties!

Can we achieve that via training with SGD?

Not Easy

Strong/weak teacher nodes

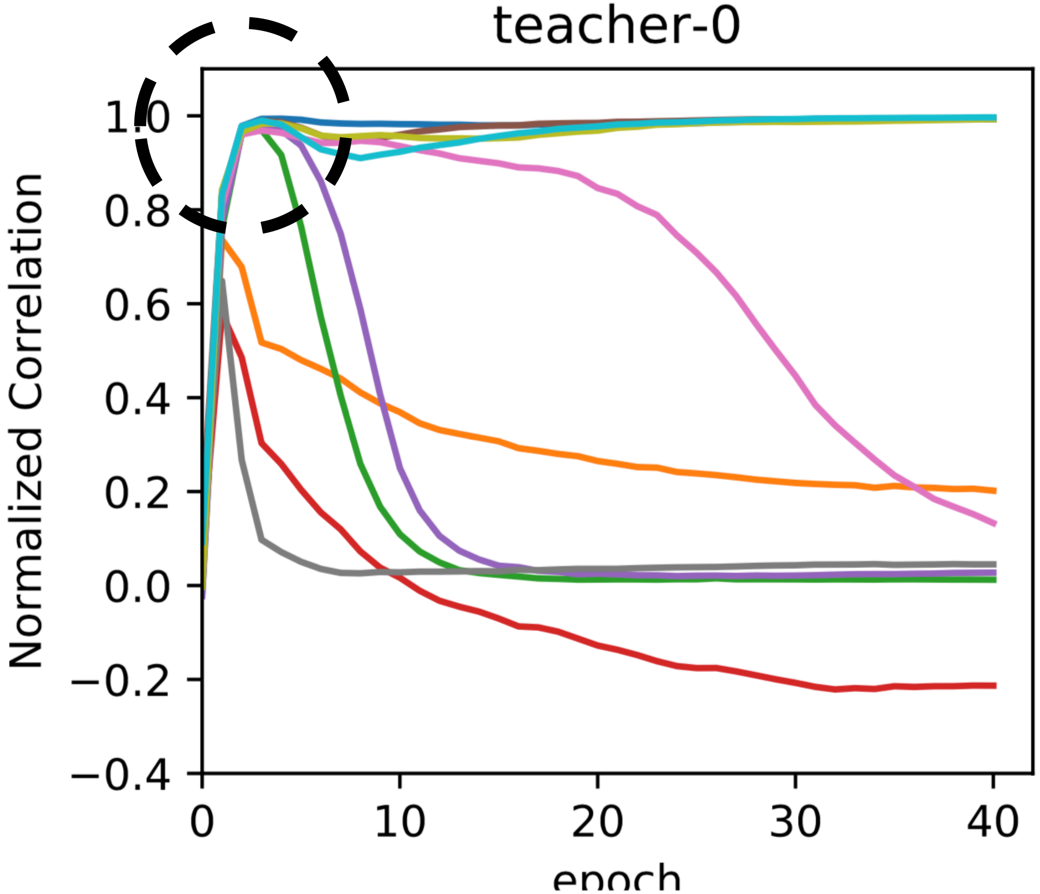


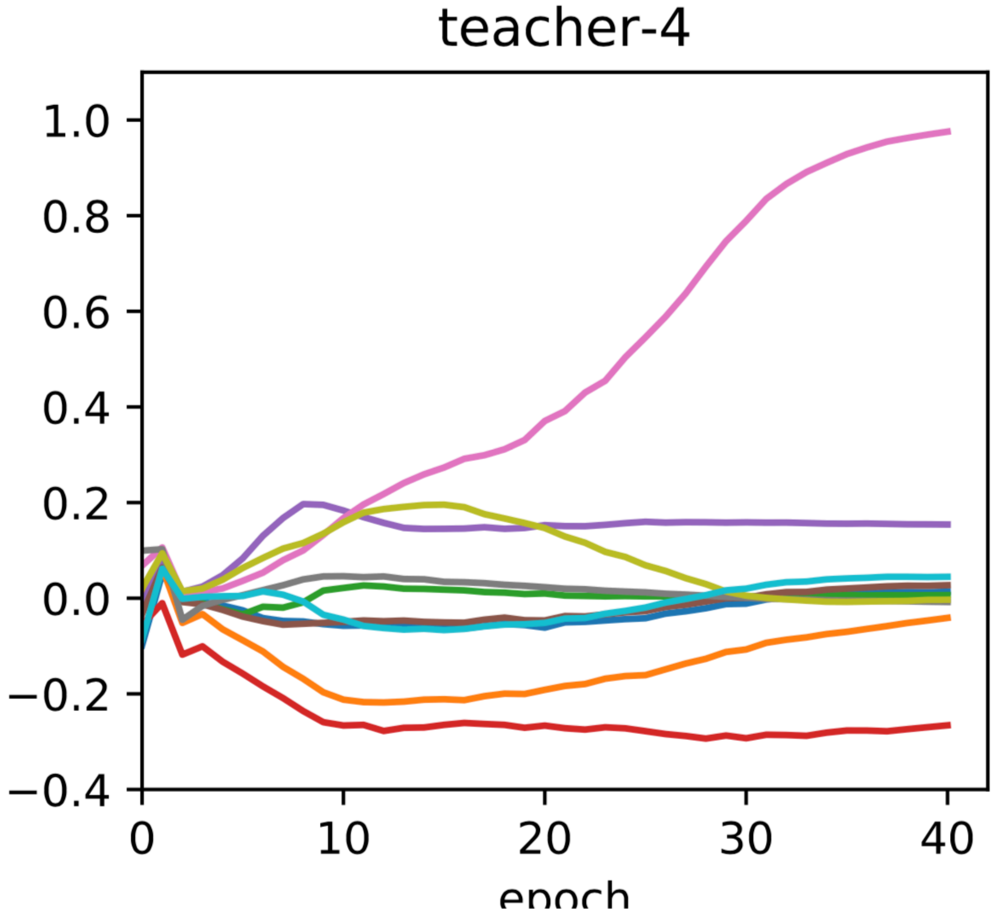
Strong teacher nodes are learned faster

1. Robust to Noise! 😃
2. Hard to learn weak teacher nodes 😢

Training Dynamics

Teacher *j*: $\|\mathbf{v}_j^*\| \propto 1/j^p$

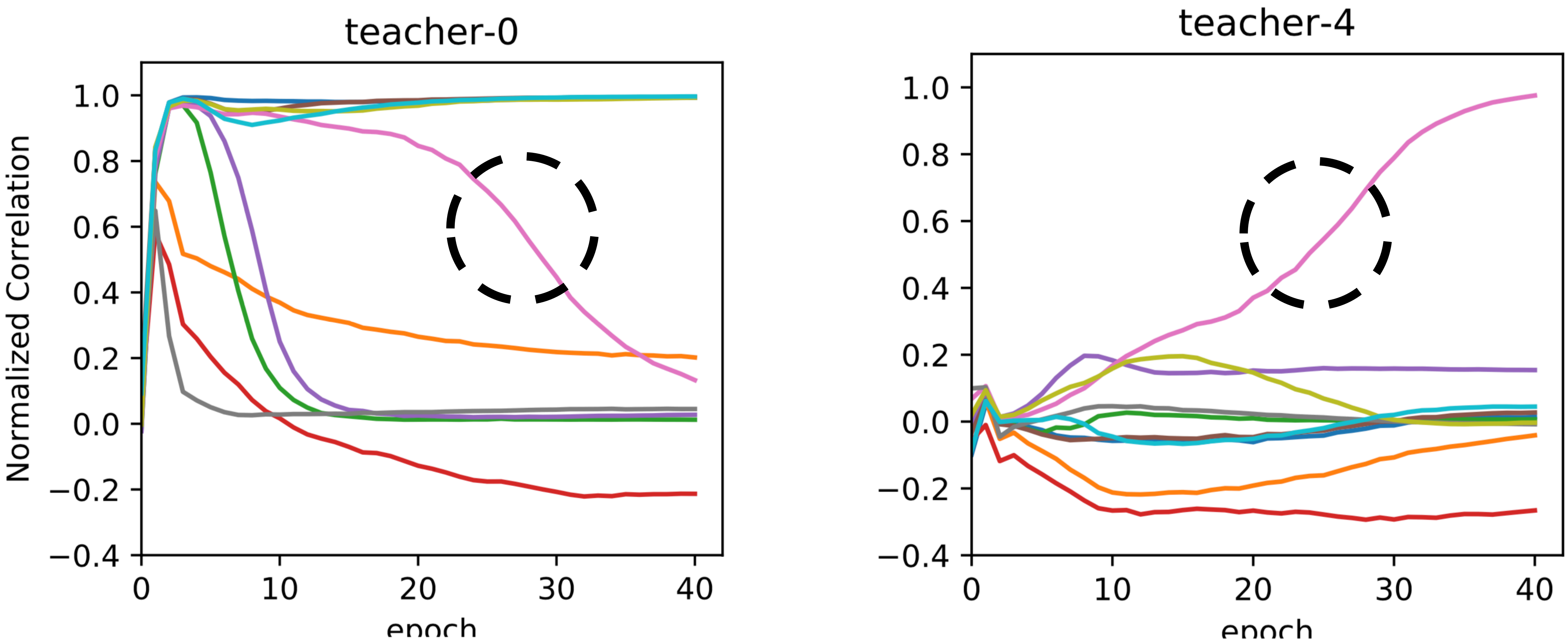




Strong teacher node attracts many students!

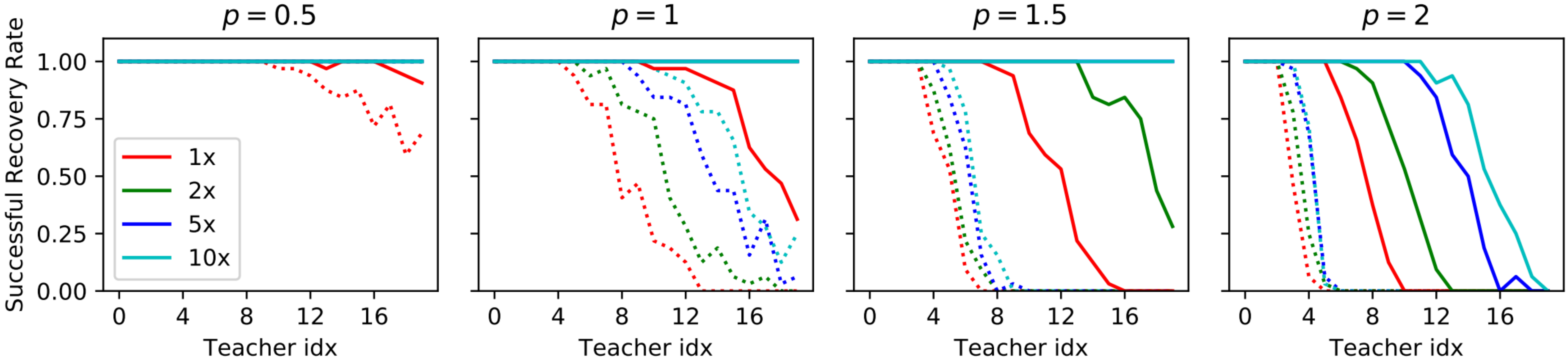
Training Dynamics

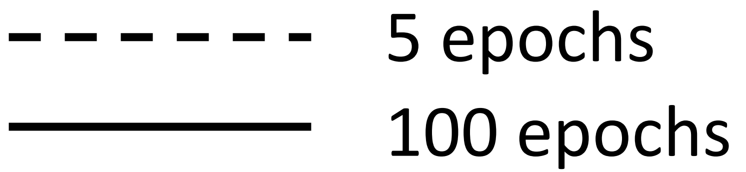
Teacher *j*: $\|\mathbf{v}_j^*\| \propto 1/j^p$



Losing student node shifts focus.

Successful Rate of Teacher Node Reconstruction





Teacher *j*: $\|\mathbf{v}_j^*\| \propto 1/j^p$

Future Directions

- Training Dynamics
- Generalization Bound
- Landscape
- ResNet / DenseNet / Network with Attention
- Adversarial Samples

Understand the Role Played by Neural Network in Prioritized Search

Carrie Wu[1], Lexing Ying[1], **Yuandong Tian[2]**

[1]Stanford University, [2]Facebook AI Research

AlphaGo Series

DeepMind

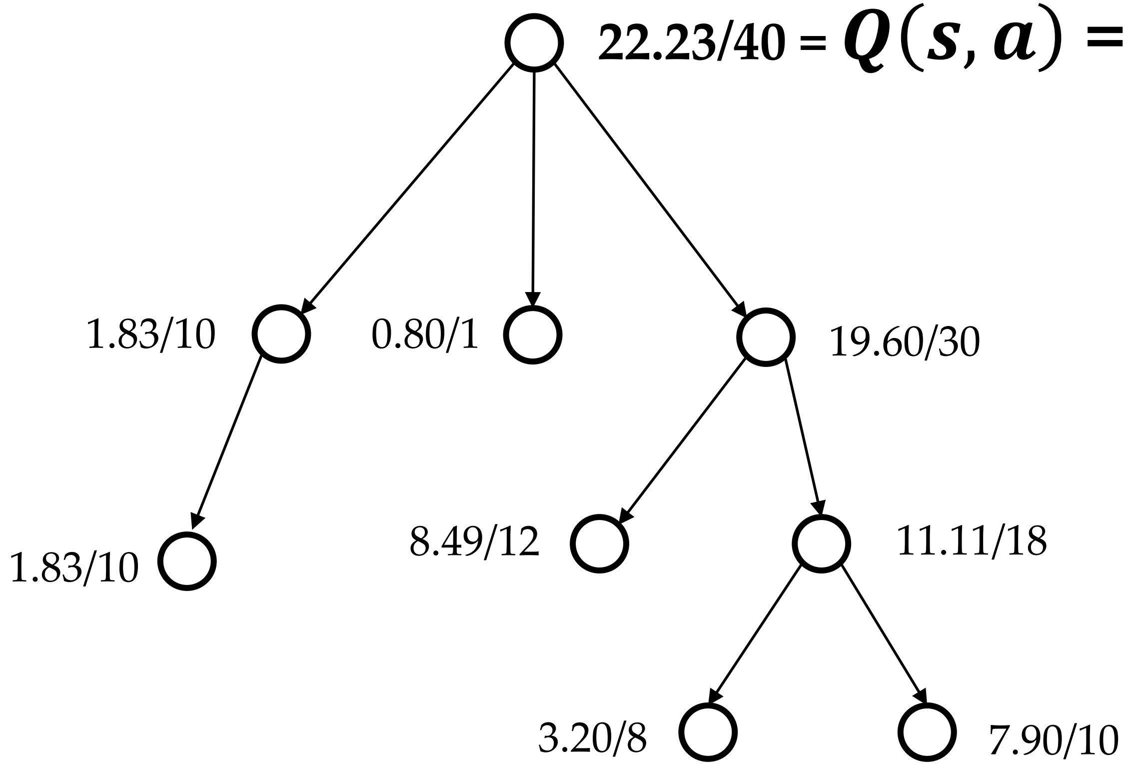
AlphaGo Lee
(Mar. 2016)

AlphaGo Master
(May. 2017)

AlphaGo Zero
(Oct. 2017)

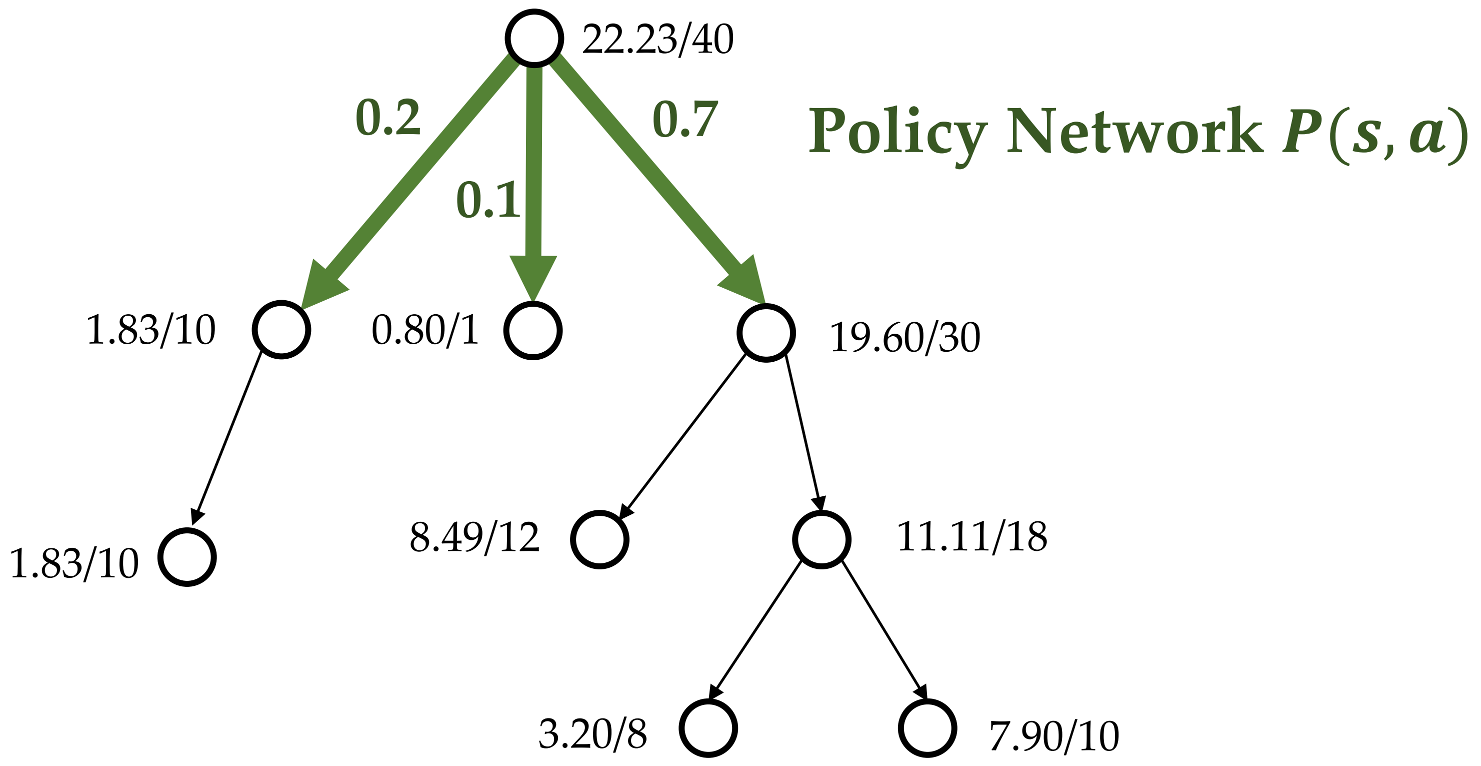
Monte Carlo Tree Search with Networks

Aggregate win rates, and search towards the good nodes.



$$22.23/40 = \boldsymbol{Q(s,a)} = \boldsymbol{\tilde{Q}(s,a)\ /\ N(s,a)}$$

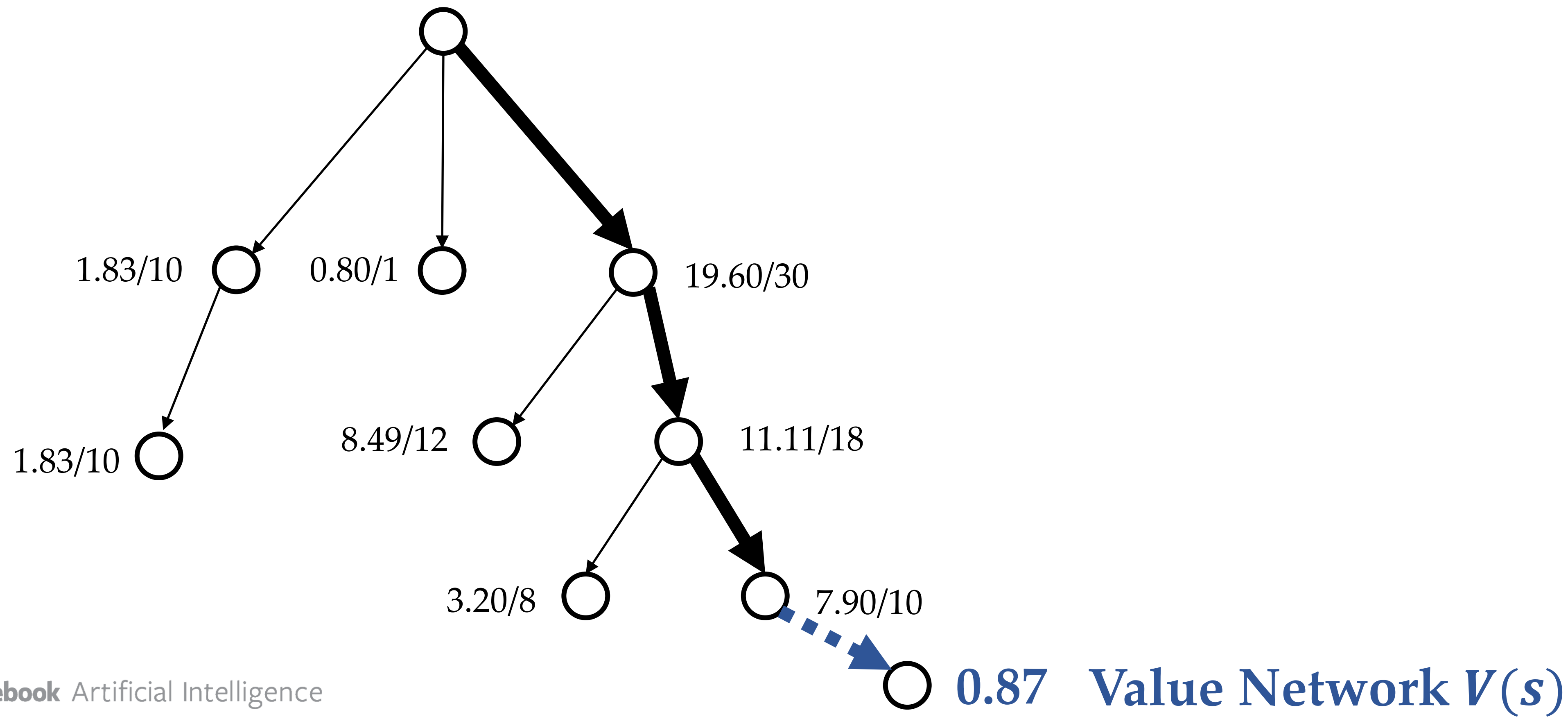
Monte Carlo Tree Search with Networks



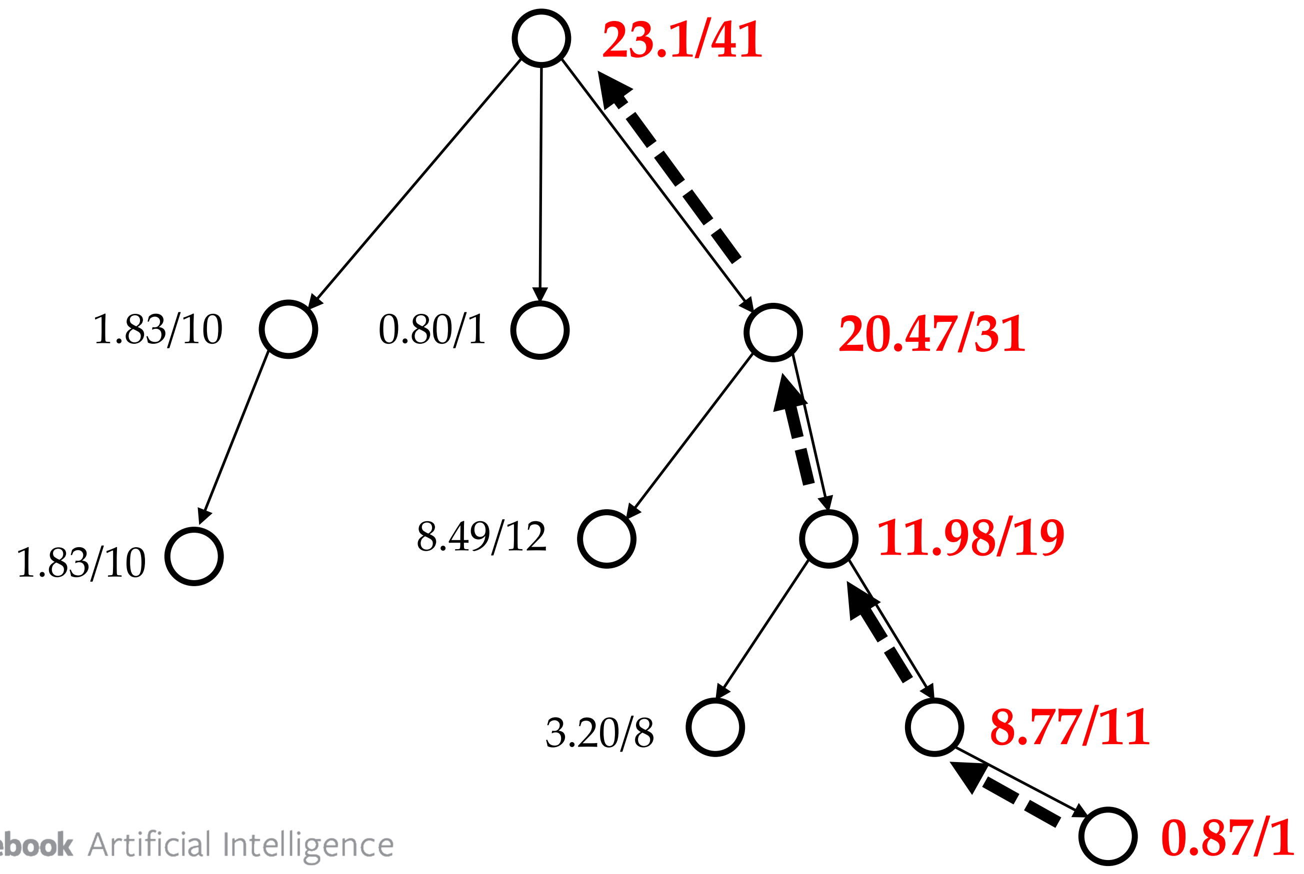
$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t,a) + u(s_t,a))$$

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$ PUCT

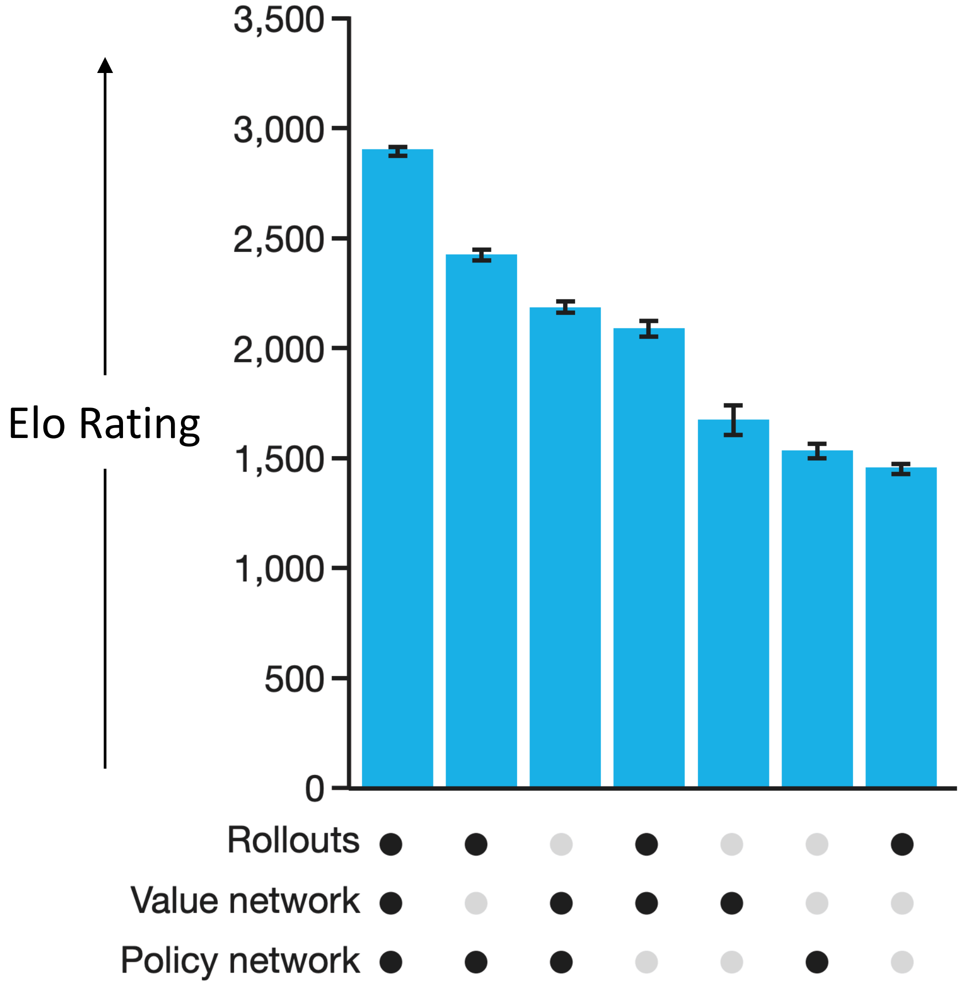
Monte Carlo Tree Search with Networks



Monte Carlo Tree Search with Networks



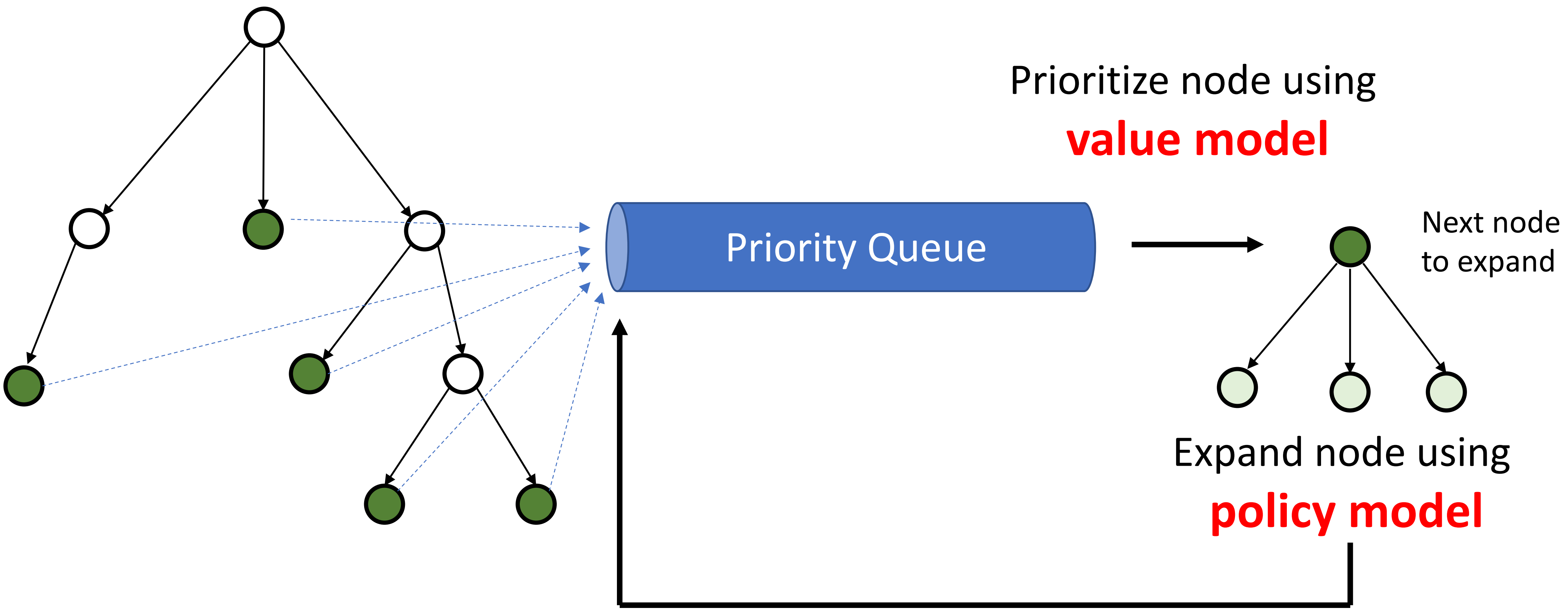
Monte Carlo Tree Search with Networks



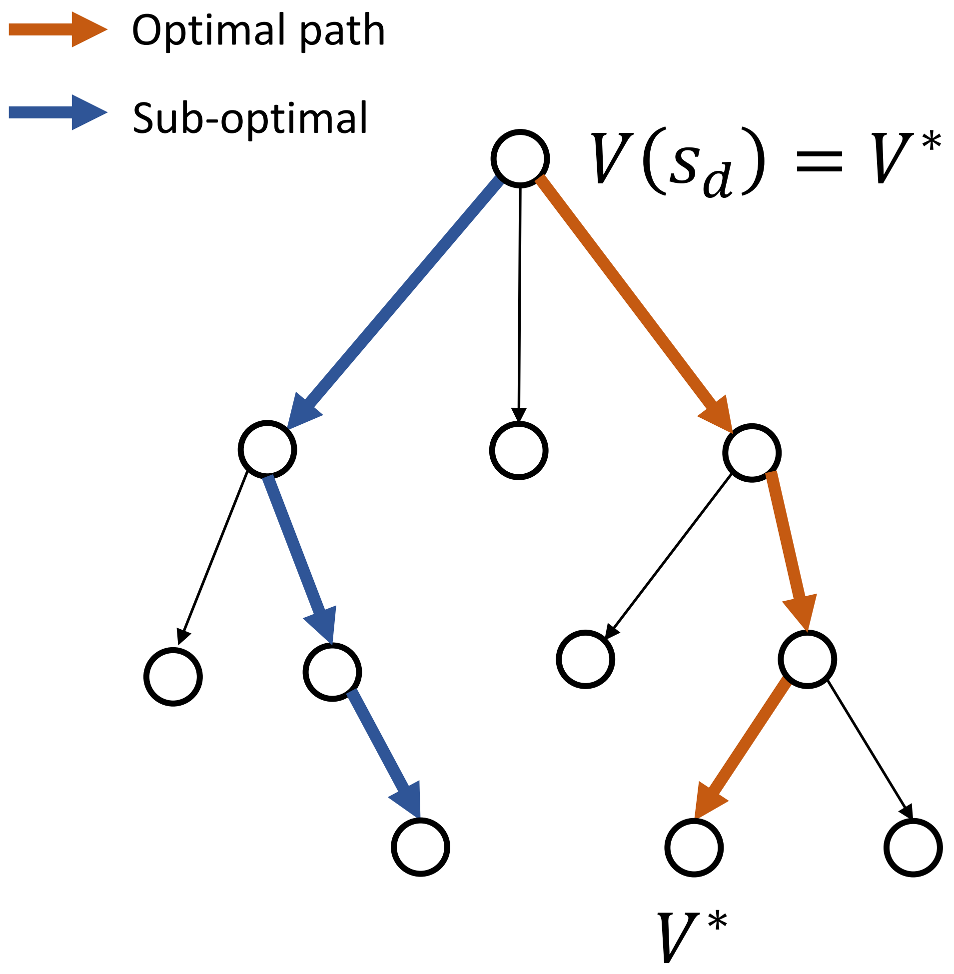
How Policy Network and Value Network improves Search Efficiency?

[Mastering the game of Go with deep neural networks and tree search, D. Silver et al. Nature 2016]

A Simple A^* Model



Notations



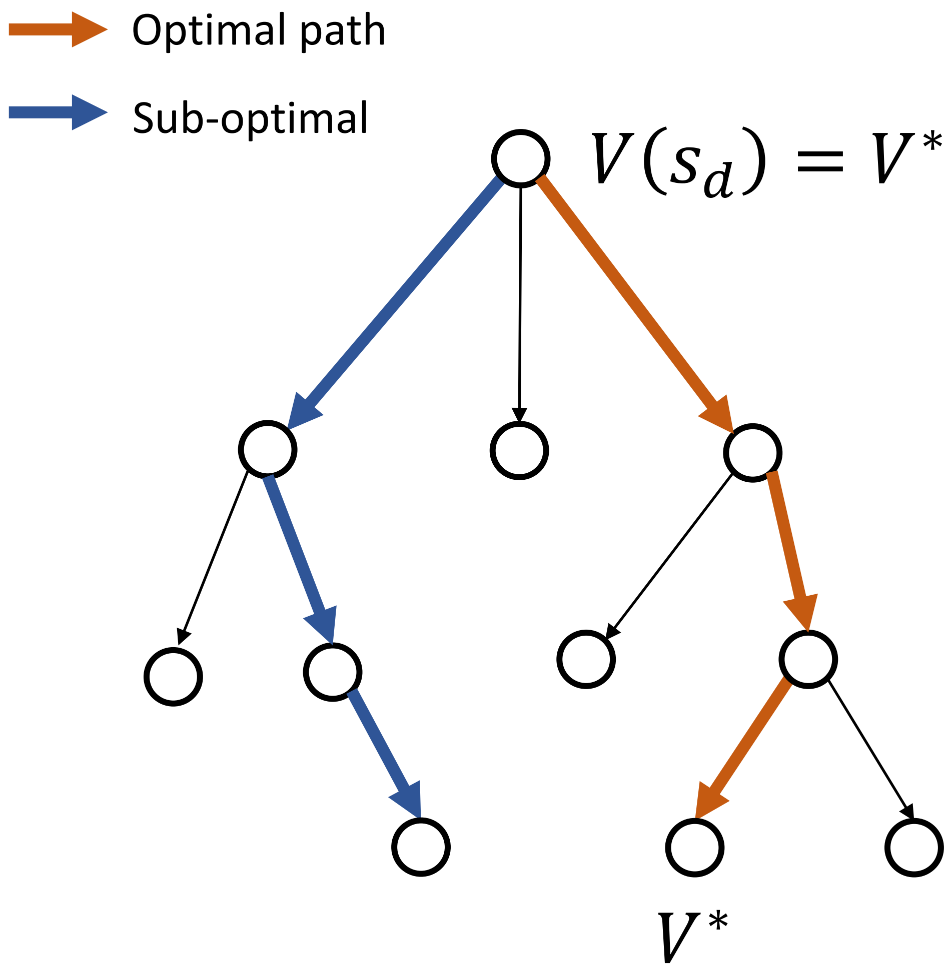
K: Branching factor

$V(s_d)$: True value of state s_d at depth d

$\Delta(s_d) = V^* - V(s_d)$: Gap to optimal value

$U(s_d)$: Predicted **deterministic** value of state s_d by value net

Notations



$X_d = V(s_d) - U(s_d)$:

i.i.d zero-mean random variable at depth d

σ_d: standard deviation

σ_d decays over depth

Set $c_d = 5\sqrt{d}\sigma_d$

$|X_d| \leq c_d$ with high probability

$U(s_d) + c_d$: Priority value

Value Network Only

A sub-optimal node is chosen if the heuristic value is **over-estimated**:

$$U(s_d) + c_d \geq V^* \quad \textbf{or} \quad e(s_d) \equiv V^* - U(s_d) - c_d = \Delta(s_d) - X(s_d) - c_d \leq 0$$

Expected Sample Complexity:

$$\mathbb{E}[N] = K\left[D + \sum_{s_d \notin \mathcal{L} \cup \mathcal{A}(l^*)} \mathbb{P}\left(e(s_d) \leq 0 \bigcap_{s_{d'} \in \mathcal{A}(s_d)} e(s_{d'}) \leq 0\right)\right]$$

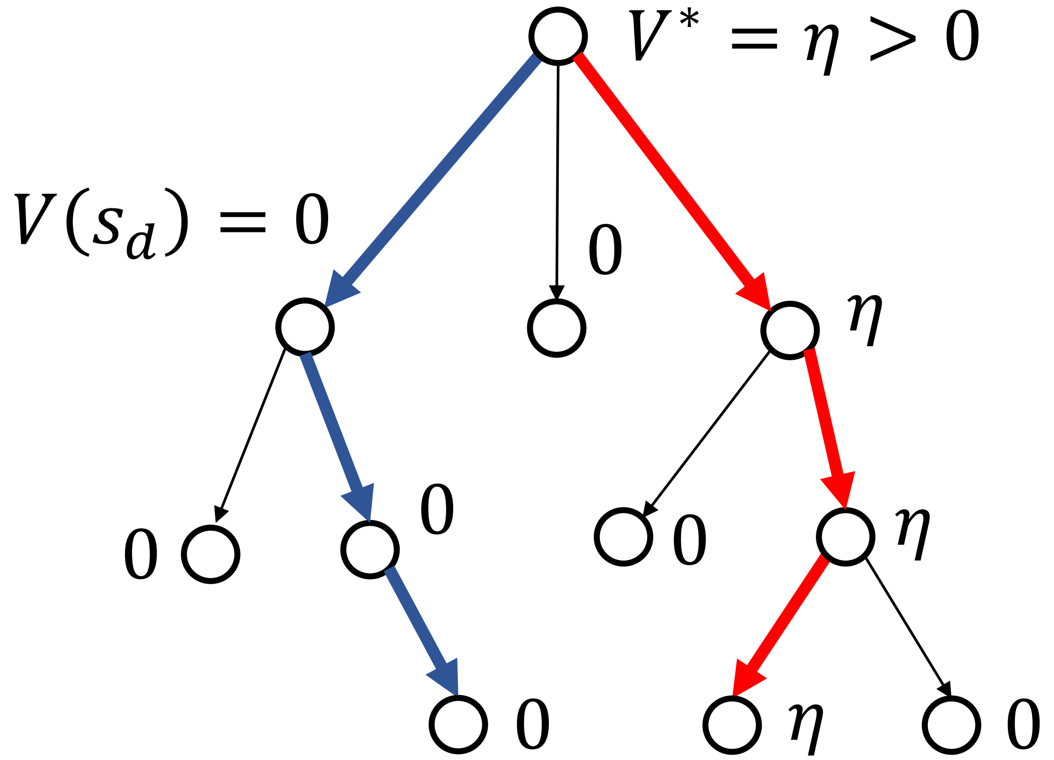
Fixed node expand cost

Optimal search path

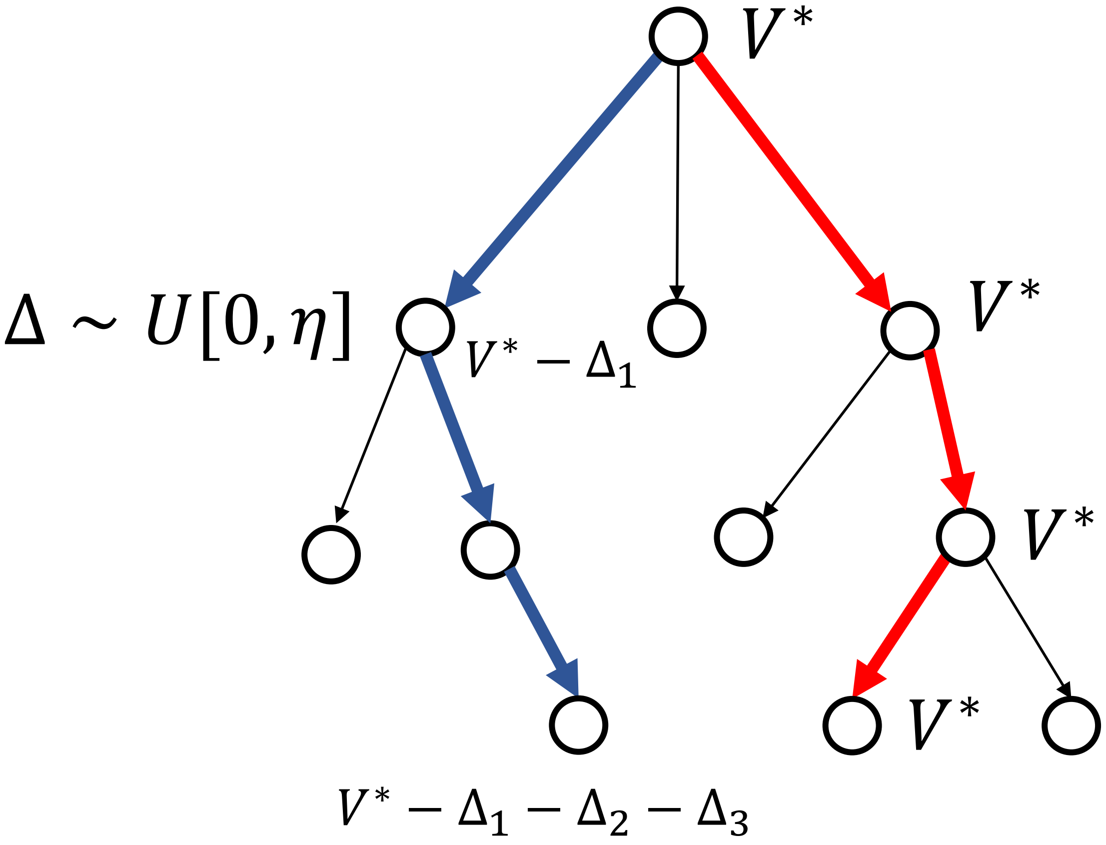
Sub-optimal path

Neural Network Models

Constant Gap Models.



Generative Models.



Value Network Only (Constant Gap Model)

Sample Complexity (#calls of value functions):

$$\mathbb{E}[N] = KD + D^2(K-1)K^c$$

for some c so that $\frac{\eta}{\sigma_c} - \sqrt{c} \geq \sqrt{2\log K}$

$\sigma_d = O(d^{-0.5-\delta})$ ➔ ***Polynomial*** sample complexity

Value Network Only (Generative Model)

Sample Complexity (#calls of value functions):

$$\mathbb{E}[N] = KD + \sum_{d=1}^{D} K^{T(d)}$$

where $T(d) = \frac{2}{\eta}\left(\sqrt{2\log K} + 1\right)\sqrt{d}\sigma_d$

$\sigma_d = O(d^{-0.5-\delta})$ ➔ ***Polynomial*** sample complexity

Success Rate at 20k expansion

Constant Gap

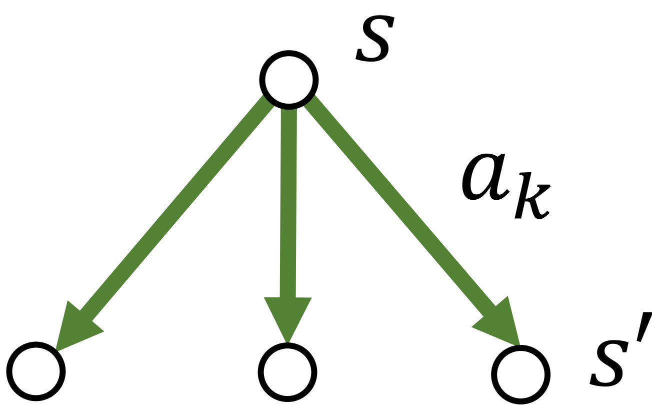
	Polynomially Decaying Noise				Exponentially Decaying Noise			
	$\gamma = 1.3$		$\gamma = 1.5$		$\alpha = 1.3$		$\alpha = 1.5$	
	Alg 1	MCTS	Alg 1	MCTS	Alg 1	MCTS	Alg 1	MCTS
$\eta = 1$	**1**	0.51	**1**	0.695	**1**	0.355	**1**	0.605
$\eta = 0.5$	**1**	0.38	**1**	0.435	**0.65**	0.265	**1**	0.4

Generative Model

	Polynomially Decaying Noise				Exponentially Decaying Noise			
	$\gamma = 1.3$		$\gamma = 1.5$		$\alpha = 1.3$		$\alpha = 1.5$	
	Alg 1	MCTS	Alg 1	MCTS	Alg 1	MCTS	Alg 1	MCTS
$\eta = 1$	**1**	0.895	**1**	0.895	**1**	0.9	**1**	0.895
$\eta = 0.5$	**1**	0.865	**1**	0.865	0.825	**0.865**	**0.995**	0.87

Polynomial: $X_d \sim N(0, d^{-2\gamma})$, Exponential: $X_d \sim N(0, \alpha^{-2d})$

Adding Policy Networks

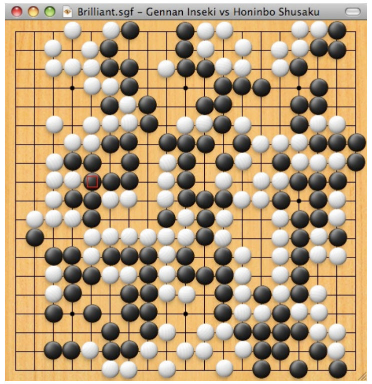


$$P(s, a_k) = \frac{\exp(U^\pi(s, a_k))}{\sum_{k=1}^{K} \exp(U^\pi(s, a_k))}$$

Assume $U^\pi(s, a_k) = V(s'(s, a_k)) + X_d^\pi$:

X_d^π is i.i.d zero-mean random variable at depth d
σ_d^π: standard deviation

Adding Policy Networks



s

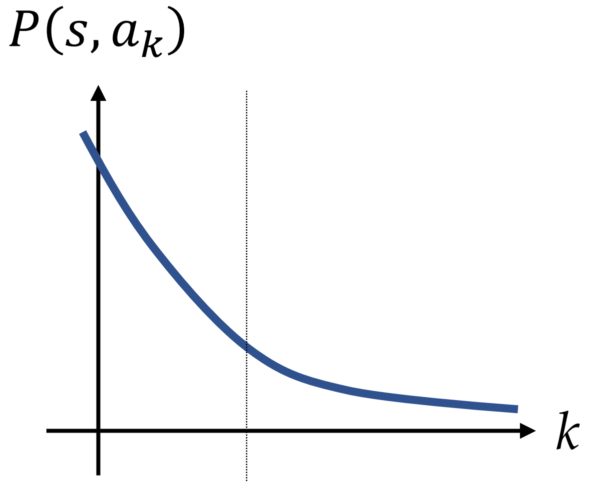
Trunk

$P(s, a)$ One forward yields ***many*** values.

$U(s)$ One forward yields a single value

Sort $P(s, a)$ so that $P(s, a_1) \geq P(s, \mathrm{a}_2) \geq \cdots \geq P(s, a_K)$

If $\log P(s, a_1) - \log P(s, a_k) \geq 2c_d^{\pi}$, **stop expanding** now.



Value and Policy Networks

Sample Complexity (#calls of neural networks):

$$\mathbb{E}[N] \leq \sum_{s_d \notin \mathcal{L}} \left(2 + \sum_{k=2}^{K-1} \mathbb{P}\left(U^\pi(s_d, a_1) - U^\pi(s_d, a_k) \leq 2c^\pi_{d+1} \right) \right) \cdot$$

$$\mathbb{P}\left(e(s_d) \leq 0 \bigcap_{s_{d'} \in \mathcal{A}(s_d)} e(s_{d'}) \leq 0 \right)$$

No fixed *K* expansions anymore

Value + Policy (*Success Rate at 20k expansion*)

Constant Gap

	Polynomially Decaying Noise				Exponentially Decaying Noise			
	$\gamma = 1.3$		$\gamma = 1.5$		$\alpha = 1.3$		$\alpha = 1.5$	
	Alg 2	PUCT	Alg 2	PUCT	Alg 2	PUCT	Alg 2	PUCT
$\eta = 1$	1	1	1	1	1	1	1	1
$\eta = 0.5$	**1**	0.885	**1**	0.92	0.685	**.705**	**1**	0.875

Generative Model

	Polynomially Decaying Noise				Exponentially Decaying Noise			
	$\gamma = 1.3$		$\gamma = 1.5$		$\alpha = 1.3$		$\alpha = 1.5$	
	Alg 2	PUCT	Alg 2	PUCT	Alg 2	PUCT	Alg 2	PUCT
$\eta = 1$	**1**	0.94	**1**	0.94	**0.99**	0.92	**1**	0.935
$\eta = 0.5$	**1**	0.935	**1**	0.92	0.82	**0.9**	**1**	0.92

Future Work

- PUCT (MCTS + Policy Network) becomes much more efficient, why?
- Visitation counts (memory)
- Max versus Average, which one is better in which situations
- Test it in real games/environment.

Thanks!