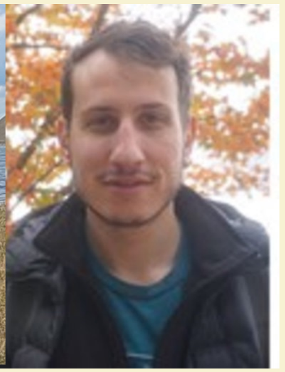
CS 229br: Foundations of Deep Learning

Lecture 5: Training Dynamics Part 1

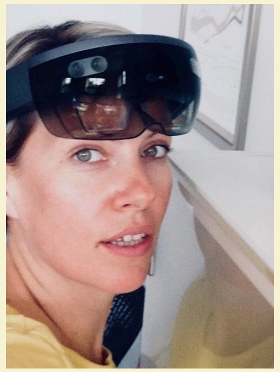
Boaz Barak

Gustaf Ahdritz

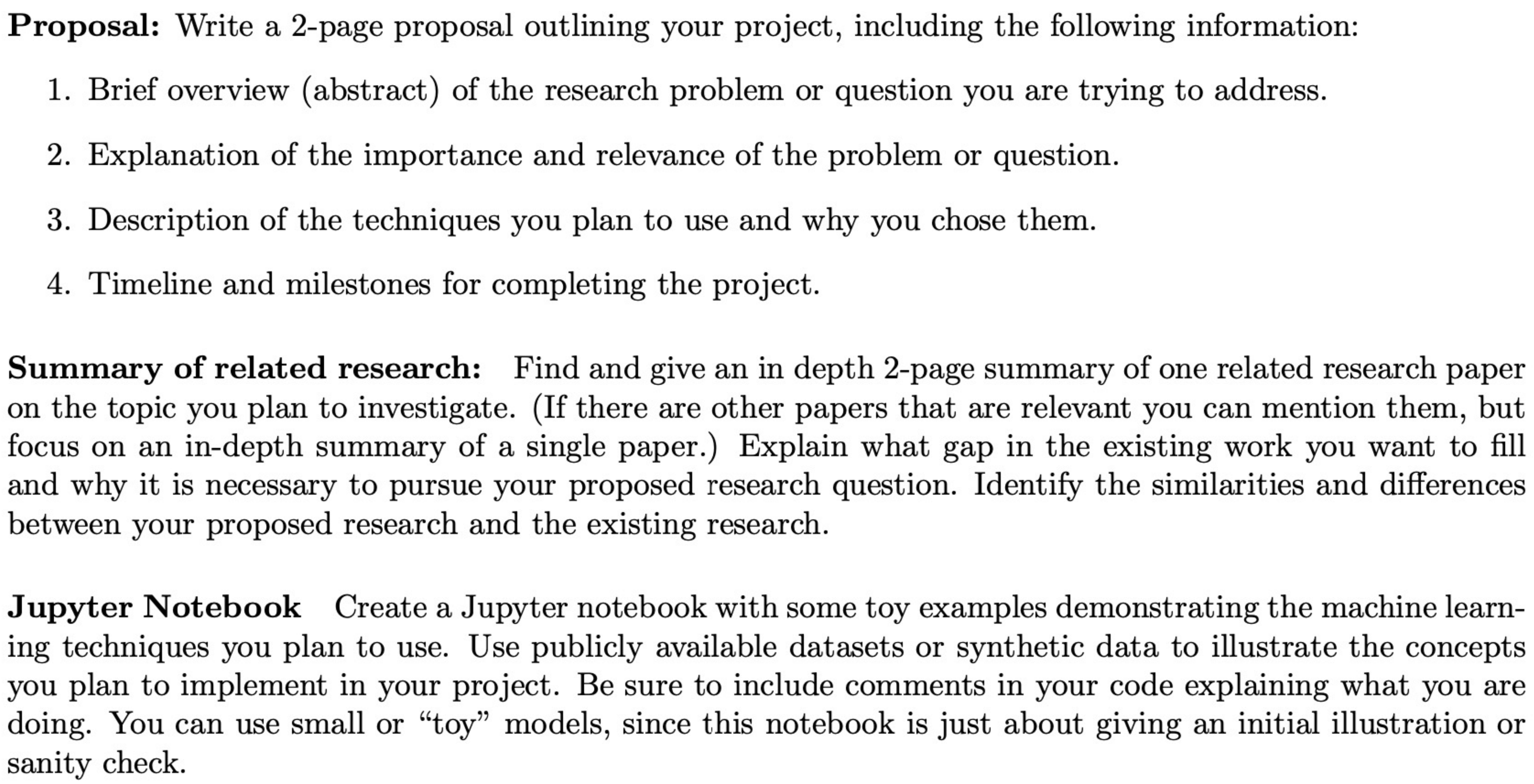
Gal Kaplun



Zona Kostic



Project proposal due 3/20



Proposal: Write a 2-page proposal outlining your project, including the following information:

1. Brief overview (abstract) of the research problem or question you are trying to address.
2. Explanation of the importance and relevance of the problem or question.
3. Description of the techniques you plan to use and why you chose them.
4. Timeline and milestones for completing the project.

Summary of related research: Find and give an in depth 2-page summary of one related research paper on the topic you plan to investigate. (If there are other papers that are relevant you can mention them, but focus on an in-depth summary of a single paper.) Explain what gap in the existing work you want to fill and why it is necessary to pursue your proposed research question. Identify the similarities and differences between your proposed research and the existing research.

Jupyter Notebook Create a Jupyter notebook with some toy examples demonstrating the machine learning techniques you plan to use. Use publicly available datasets or synthetic data to illustrate the concepts you plan to implement in your project. Be sure to include comments in your code explaining what you are doing. You can use small or "toy" models, since this notebook is just about giving an initial illustration or sanity check.

Yann LeCun’s Response

In the history of science and technology, the engineering artifacts have almost always preceded the theoretical understanding: the lens and the telescope preceded optics theory, the steam engine preceded thermodynamics, the airplane preceded flight aerodynamics, radio and data communication preceded information theory, the computer preceded computer science.

Yann LeCun's Response

It is this kind of attitude that led the ML community to abandon neural nets for over 10 years ... with their non-convex loss functions, [NNs] had no guarantees of convergence (though they did work in practice then, just as they do now).

So people threw the baby with the bath water and focused on "provable" convex methods or glorified template matching methods ...

Sticking to a set of methods just because you can do theory about it, while ignoring a set of methods that empirically work better just because you don't (yet) understand them theoretically is akin to looking for your lost car keys under the street light knowing you lost them someplace else.

Roger Grosse’s comments

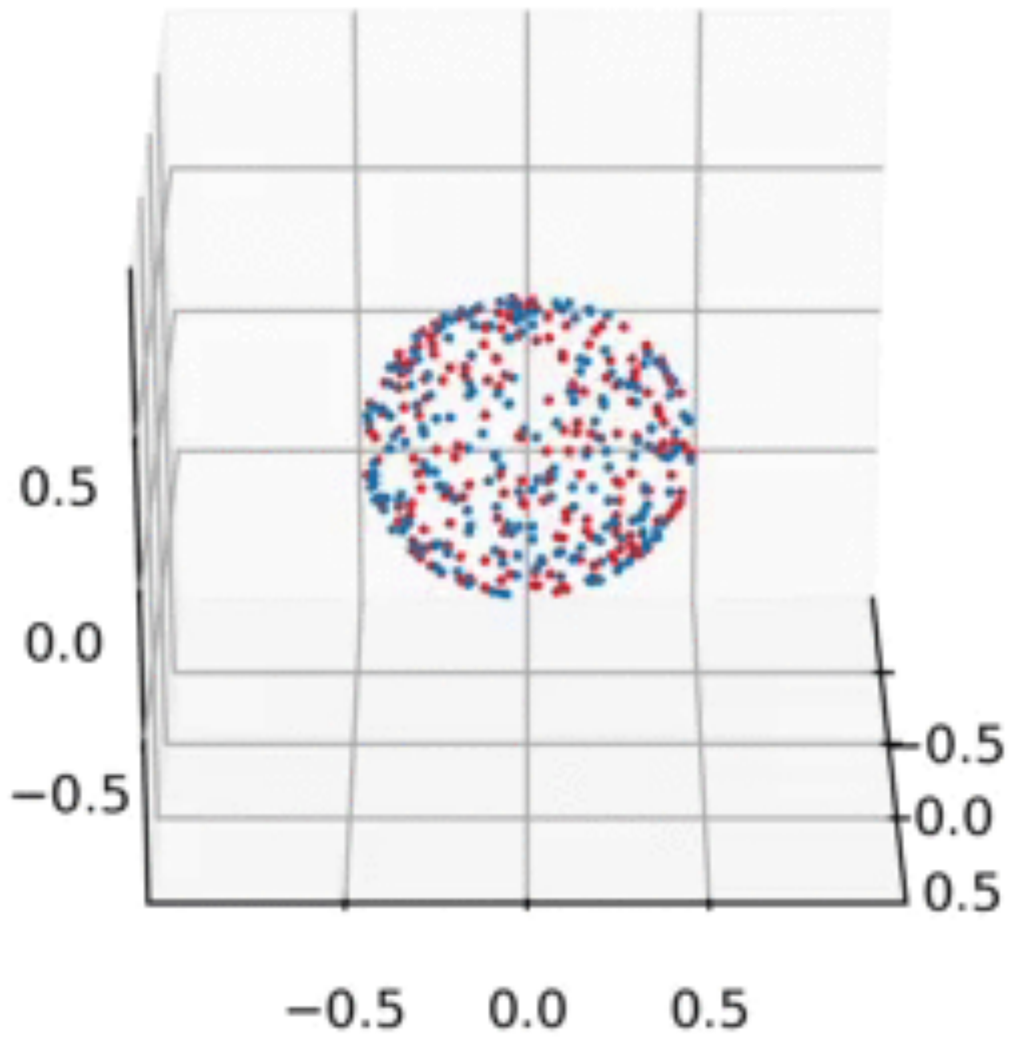
Asking “why does Batch Norm help?” is like asking an organic chemist “what does Nitrogen do?”

What are the mysteries of deep learning?

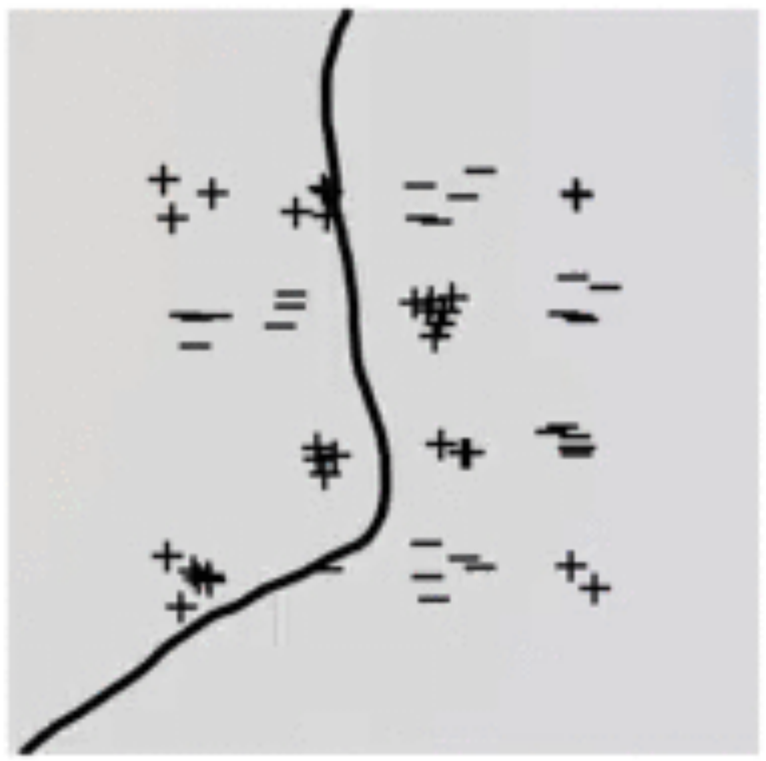
(discussion)

Training Dynamics

Parameters

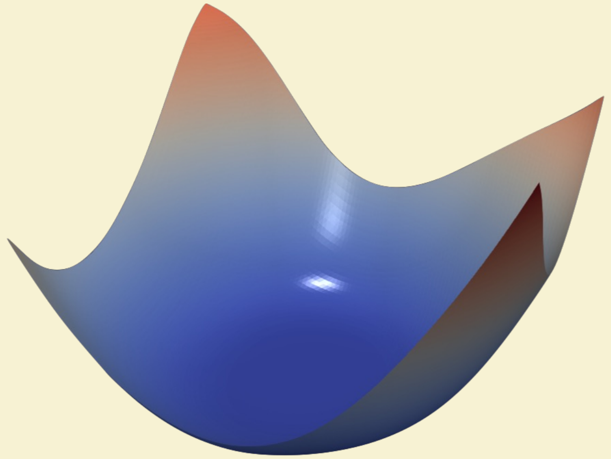


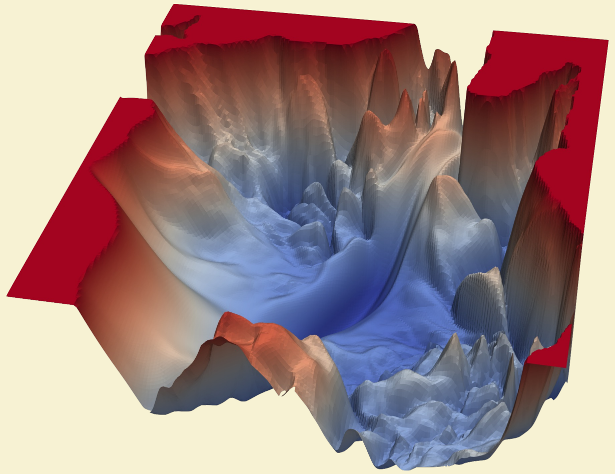
Outputs



Training depth-2 ReLU on exp loss - Lénaïc Chizat

Optimization





$$\min_{\theta \in C} L(\theta)$$

*Images: Tom Goldstein lab

$$\min_{\theta \in \mathcal{C}} L(\theta) \qquad L: \mathbb{R}^N \to \mathbb{R}$$

Or estimators

Access to L:

- Value oracle $L: \mathbb{R}^N \to \mathbb{R}$
- Gradient: $\nabla L: \mathbb{R}^N \to \mathbb{R}^N$
- Hessian: $\nabla_2 L: \mathbb{R}^N \to \mathbb{R}^{N^2}$
- Restricted: Experts / Bandits / RL

Thm: If L (strongly) convex and $\mathcal{C}$ convex then following are equivalent:

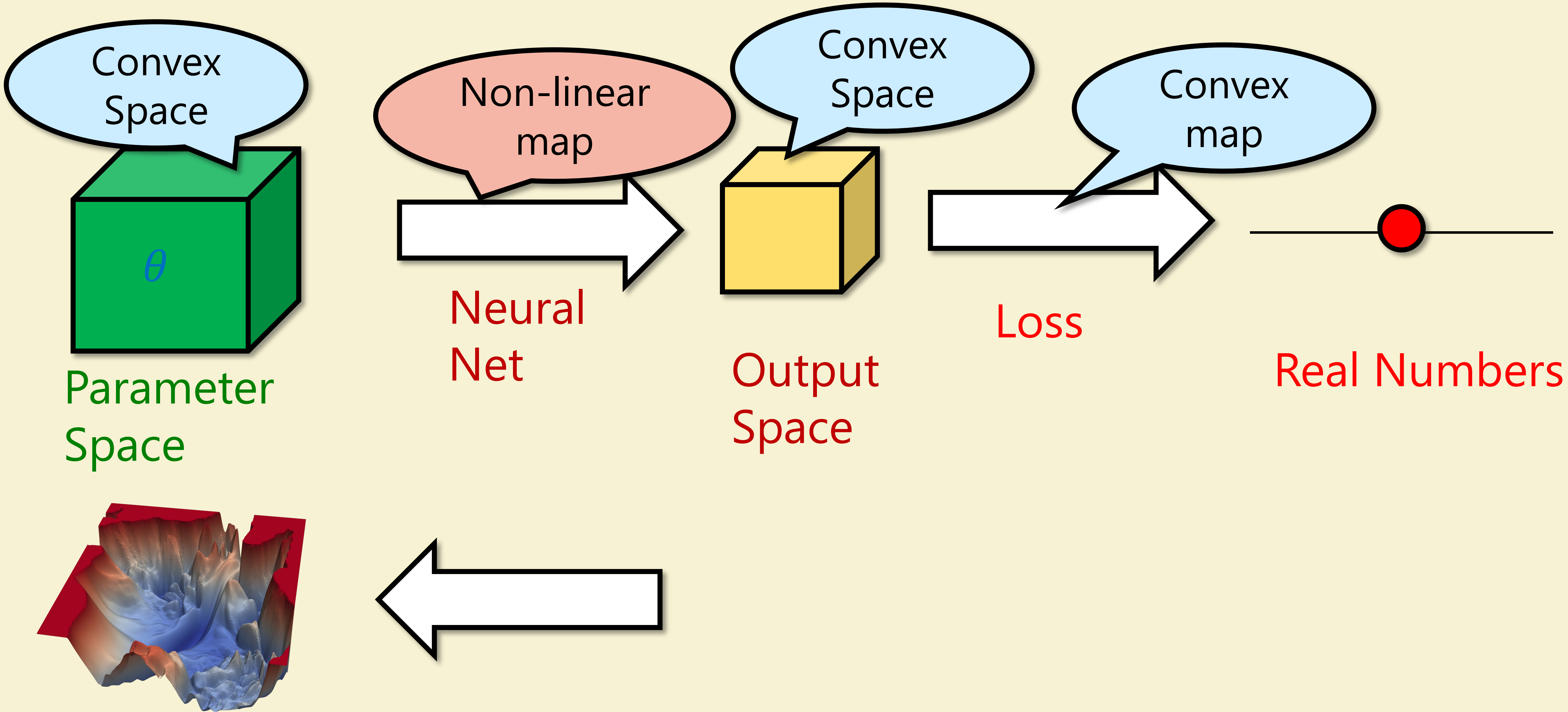
- θ is global minimum
- θ is local minimum
- $\nabla \mathrm{L}(\theta) = 0^N$

Constraints:

- Explicit
- Implicit: membership or separation oracle
- Can replace with regularizer $\lambda \cdot b(\theta)$

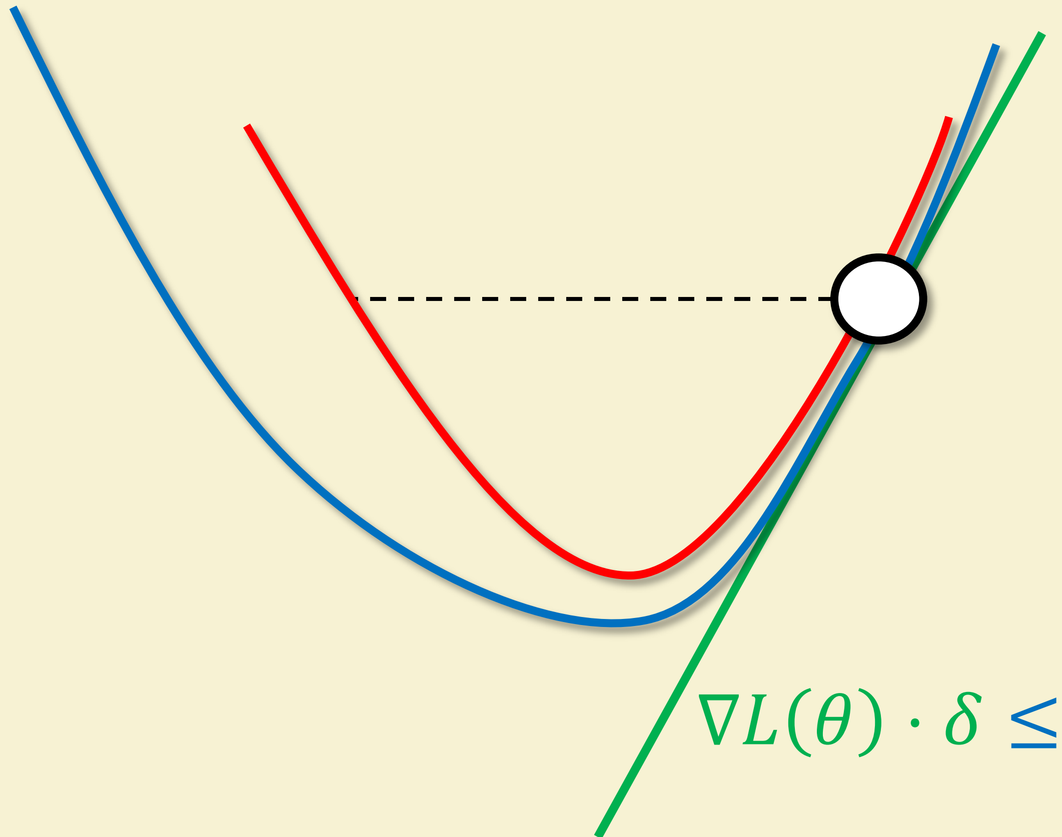
Optimization in (self) supervised learning

Distribution $x \sim X$, Loss $L(\theta) = \mathbb{E}_{x \sim X}[L_x(\theta)]$



Iterative optimization algorithms

- $\theta^0 =$ something
- $\theta^{t+1} = \theta^t + \delta$ for δ "small" and (on average) $L(\theta^{t+1})$ "smaller" than $L(\theta^t)$



$$\nabla L(\theta) \cdot \delta \leq L(\theta + \delta) - L(\theta) \approx \nabla L(\theta) \cdot \delta + \frac{1}{2}\delta^\top \nabla_2(\theta)\delta$$

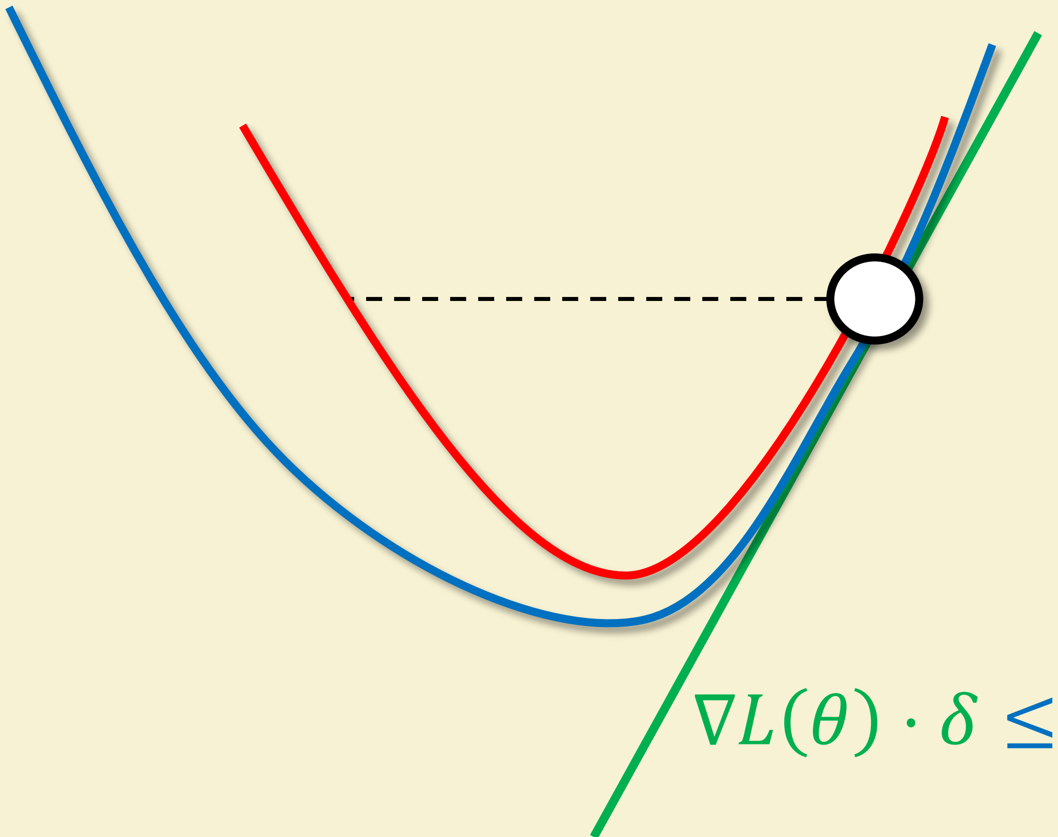
Linear Regression

“half of neural net phenomena can be explained by reasoning about linear networks” Roger Grosse

Move to whiteboard

Iterative optimization algorithms

- $\theta^0 =$ something
- $\theta^{t+1} = \theta^t + \delta$ for δ "small" and (on average) $L(\theta^{t+1})$ "smaller" than $L(\theta^t)$



$$\nabla L(\theta) \cdot \delta \leq L(\theta + \delta) - L(\theta) \approx \nabla L(\theta) \cdot \delta + \frac{1}{2}\delta^\top \nabla_2(\theta)\delta$$

- $\theta^0 =$ something
- $\theta^{t+1} = \theta^t + \delta$ for δ "small" and (on average) $L(\theta^{t+1})$ "smaller" than $L(\theta^t)$

Gradient Descent: $\theta^{t+1} = \theta^t - \eta \cdot I \cdot \nabla L(\theta)$

Trust Region: $\theta^{t+1} = \arg \min_{d(\theta,\theta^t)\leq\eta} L(\theta)$

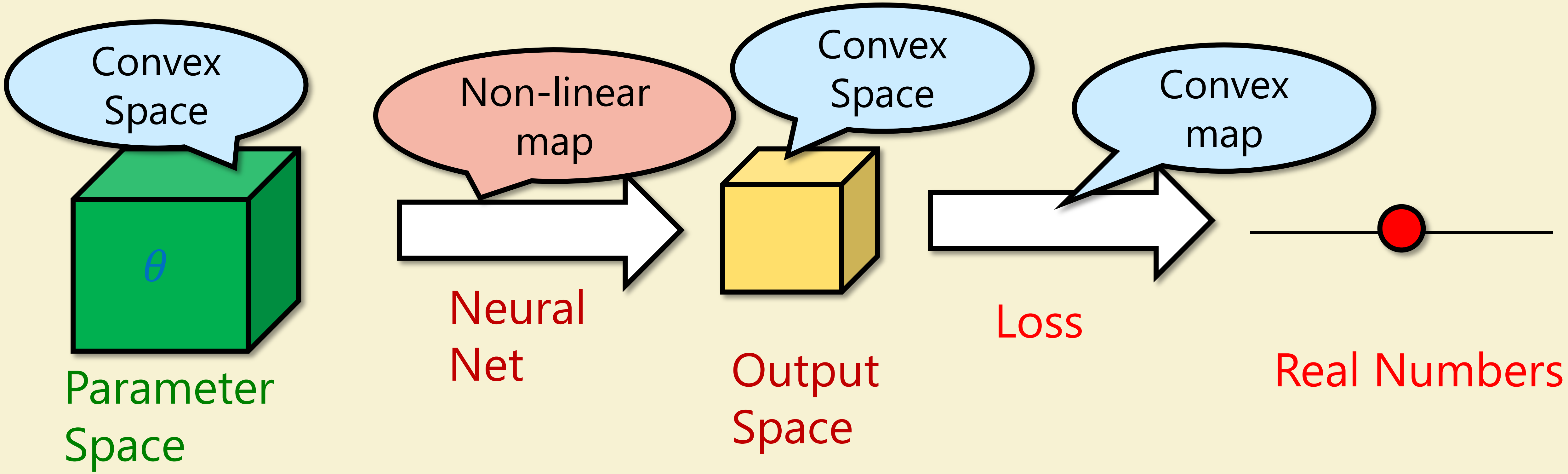
$\eta \to 0 \Leftrightarrow \lambda \to \infty$

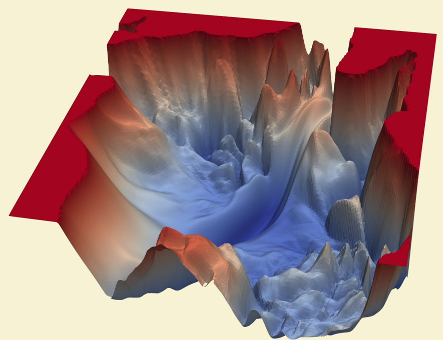
Proximal: $\theta^{t+1} = \arg\min L(\theta) + \lambda \cdot d(\theta, \theta^t)$

Projected GD: $\theta^{t+1} = Proj_{\{\theta | d(\theta,\theta^t) \leq \eta\}} \theta - \nabla \mathrm{L}(\theta)$

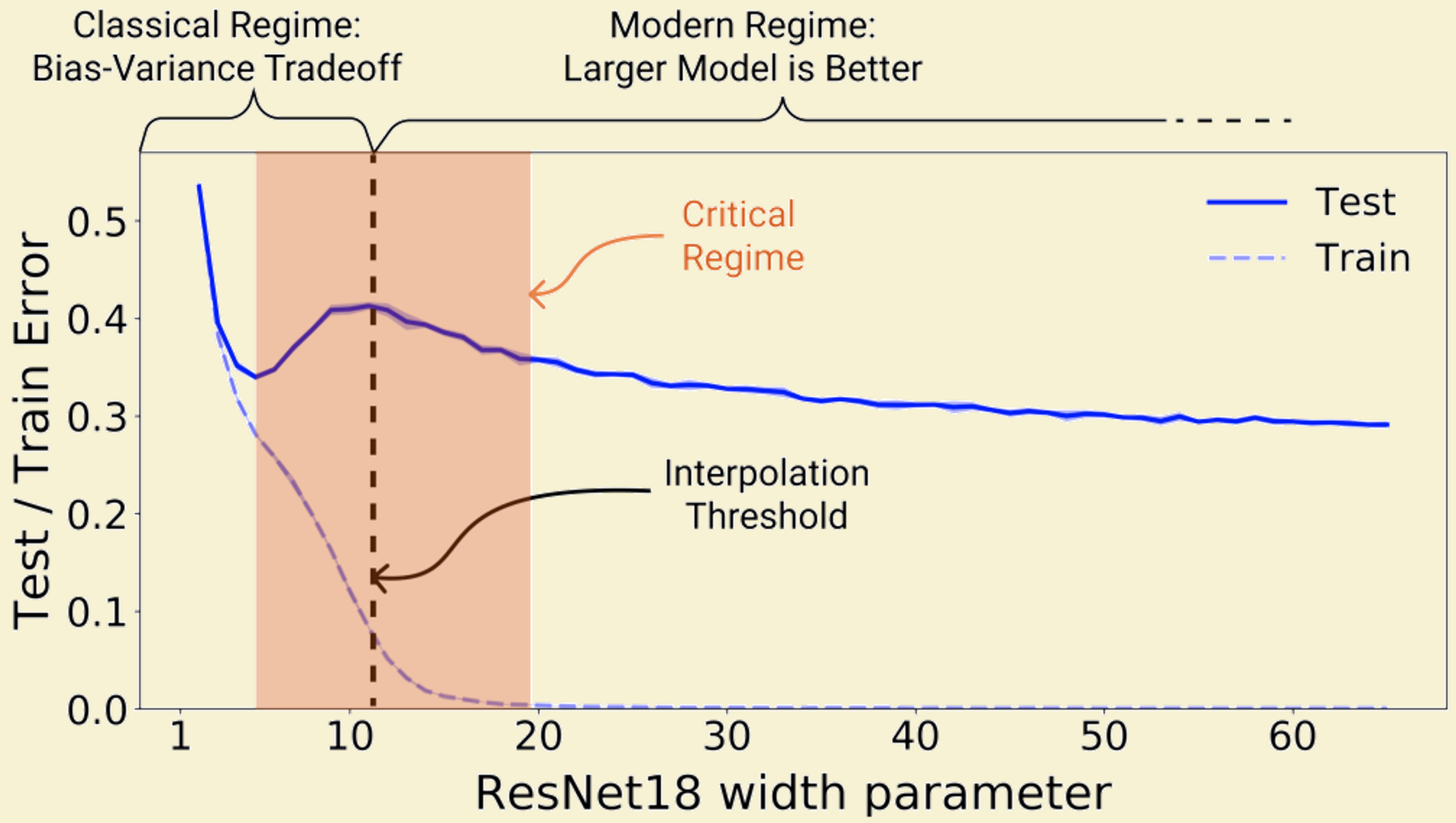
Optimization in (self) supervised learning

Distribution $x \sim X$, Loss $L(\theta) = \mathbb{E}_{x \sim X}[L_x(\theta)]$





Extra: Double Descent

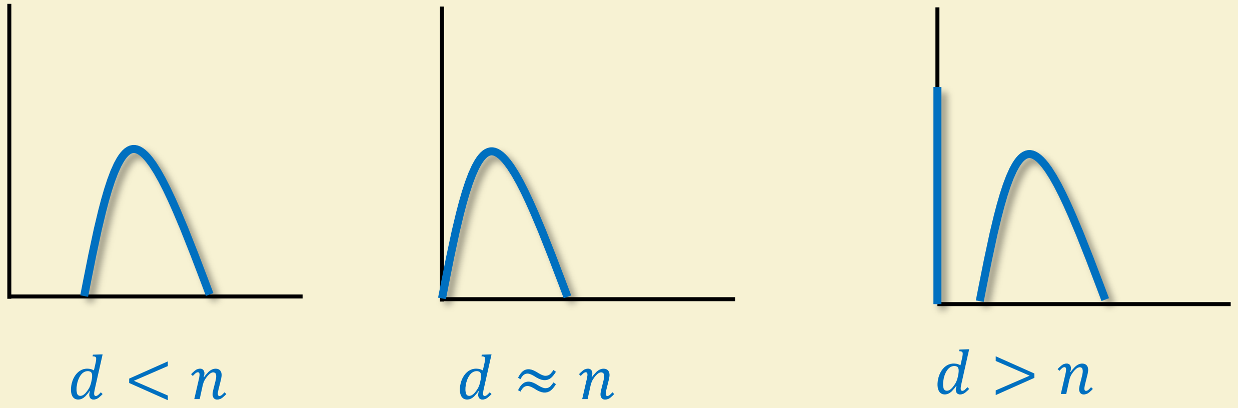


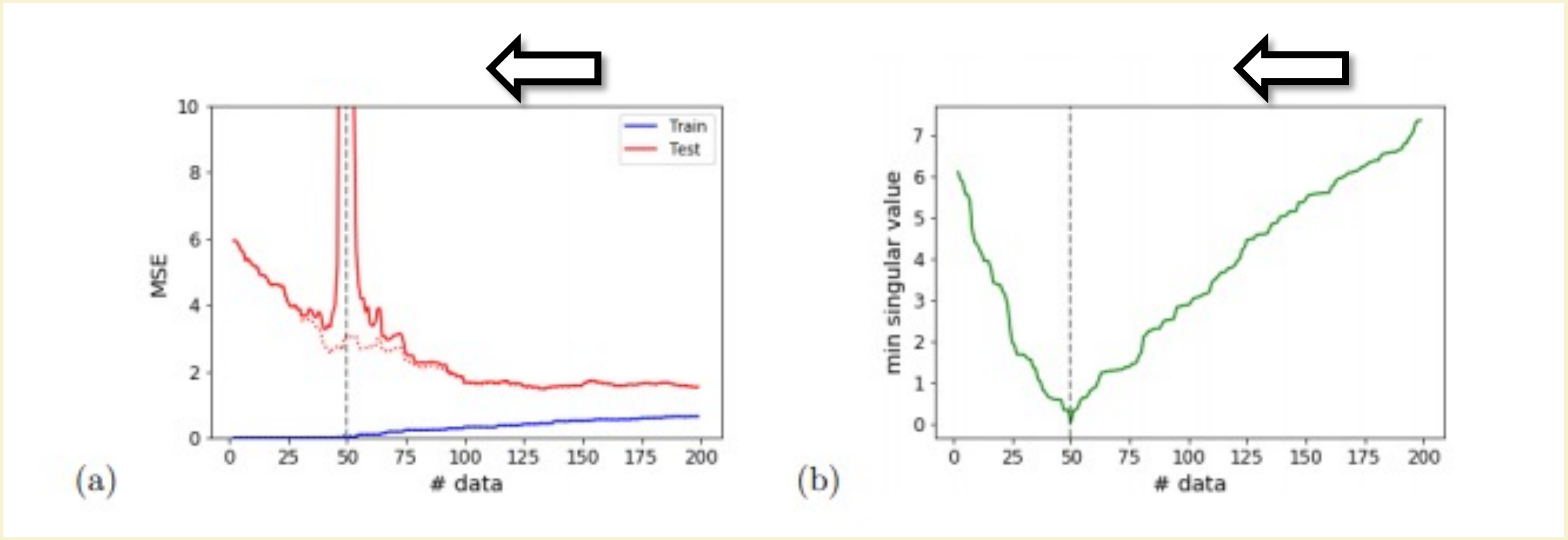
Nakkiran, Kaplun, Bansal, Yang, Barak, Sutskever

Extra: Double Descent

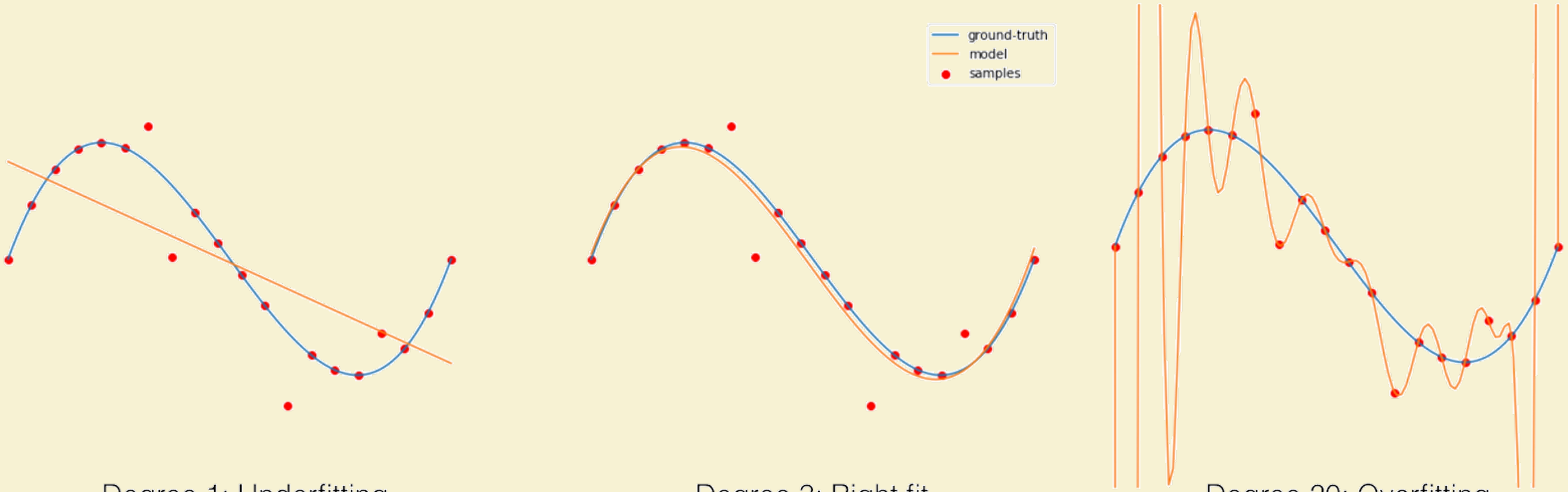
$$X^\top X = \begin{pmatrix} \lambda_1 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \lambda_d \end{pmatrix}$$

Random X:





Extra: Double Descent

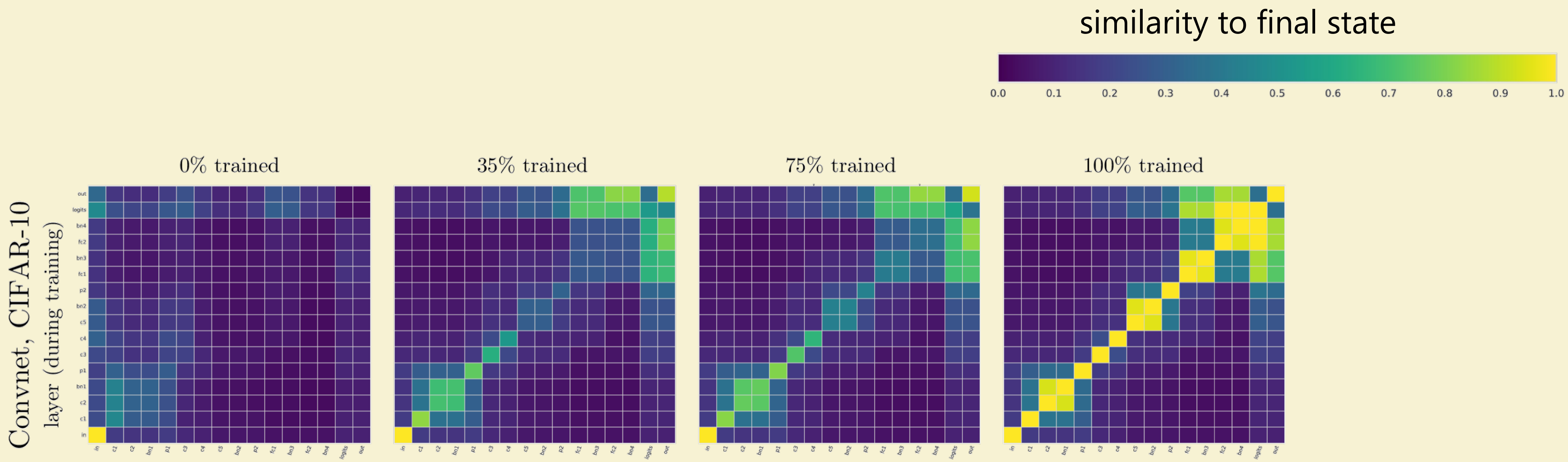


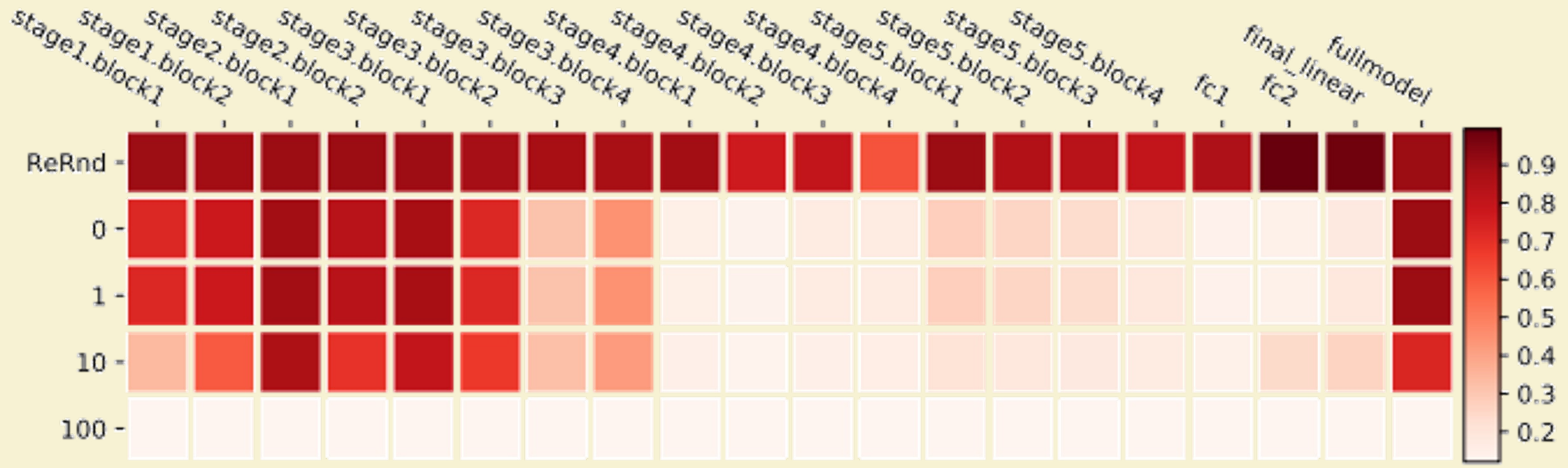
Degree 1: Underfitting

Degree 3: Right fit

Degree 20: Overfitting

Layers





randomness just for symmetry breaking!

Raghu, Gilmer, Yosinski, Sohl-Dickstein, 2017

Zhang, Bengio, Singer 2019