9.520/6.7910: Statistical Learning Theory and Applications

- Class: **Tue, Thu 11:00 am - 12:30 pm**, 46-3002
- Office Hours: **Tue, Thu 1:30pm - 3:00pm, 46-5155D**
- Web: https://poggio-lab.mit.edu/9-520/
- Contact: 9.520@mit.edu, pierb@mit.edu
- 9.520/6.7910 will use Canvas: https://canvas.mit.edu/courses/34219
- Also check Canvas announcements for updates

9.520: the parts of the course

The first part is about:

- classical regularization and regularized least squares + kernel machines, SVM
- logistic regression, square and exponential loss
- large margin (minimum norm)
- stochastic gradient methods,
- overparametrization, implicit regularization
- approximation/estimation errors.

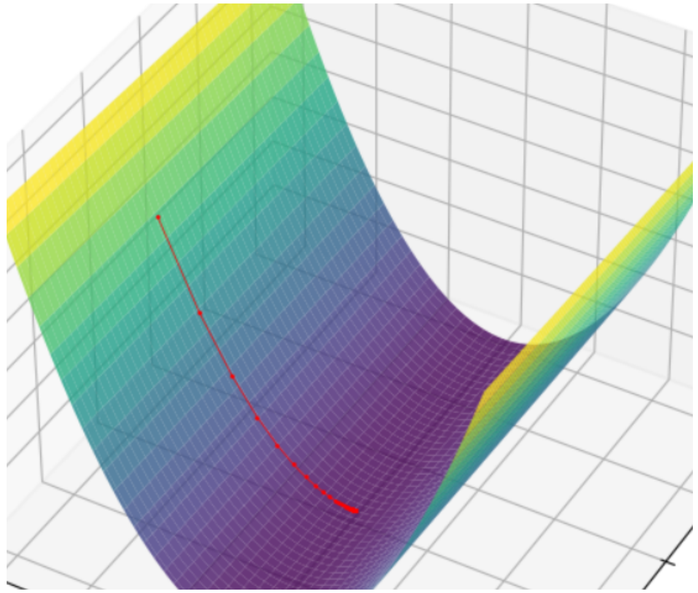
The second part is about deep networks and, particular, about:

- approximation theory -- why deep networks avoid the curse of dimensionality and are a universal parametric approximator s
- optimization theory -- how weights and activation evolve in time and across layers during training with SGD
- learning theory -- how generalization in deep networks can be explained in terms of the complexity control implicit in regularized (or unregularized) SGD and in terms of the details of the sparse compositional architecture itself.

- **This is not a class where we teach material.**
- **Today we *pitch* what comes next.**
- **Now we care about** *Neural Networks.*
- **Many of the things we saw with Lorenzo are** *not that easy* **anymore:**

They break!

Today we see what some of them are!



vs

In these modern times…
the theory we need should be concerned with

fundamental principles of intelligence,

not just theorems for the sake of math

The stunning success of deep learning (DL) over the past decade is a puzzling natural phenomenon of great scientific interest. This success has been amply demonstrated through a plethora of convincing experiments, and yet it is crying out for a thorough scientific understanding through foundational research (C.P.)

We need a theory of deep learning

As we have discussed, current attempts to understand, predict and control behaviors of machine learning models are not based on fundamental principles and are unlikely to be successful. On the other hand, for the reasons given above, there is a real hope that such fundamental principles may exist and can be discovered. This is evidenced by recent progress in the theory of deep learning… These considerations call for a major expansion in the effort to understand fundamental principles of deep learning. …. **Understanding the fundamental theory of deep learning and applying its analyses to real systems is a compelling and urgent need.**

Misha Belkin, Jul 30, 2023

Conceptual Framework of Machine Learning

1. choose $\mathbb{H}_\lambda$, a space $\mathbb{H}_\lambda$ of functions with parameters λ - in which to search for f. The functions in $\mathbb{H}_\lambda$ must have a *non-exponential* number of parameters (in the effective dimensionality) AND be powerful enough to approximate a broad range of target functions (realm of approximation theory);

2. optimize the values of the parametric approximator by minimizing a loss function on a training set (realm of optimization theory);

3. study generalization properties of the trained representation (domain of learning theory).

Conceptual Framework of Machine Learning

Note that the functions in $\mathbb{H}_\lambda$ should have a *non-exponential* number of parameters (in the effective dimensionality). This means that if I can find such a space AND if $\mathbb{H}_\lambda$ can approximate a broad range of target functions then I have made not only *approximation possible* but I have made also *optimization easier* and *generalization more likely*. The reason is # *parameters.*

In the modern era.. “new” puzzles about approximation, generalization, optimization

- No curse of dimensionality
- VC bounds overparametrization
- Optimization unreasonably easy, non-convex non-PL but optimization works

In reality…

The puzzles about *overparametrization* and *generalization* are not new (remember lorenzo with norms not # parameters)

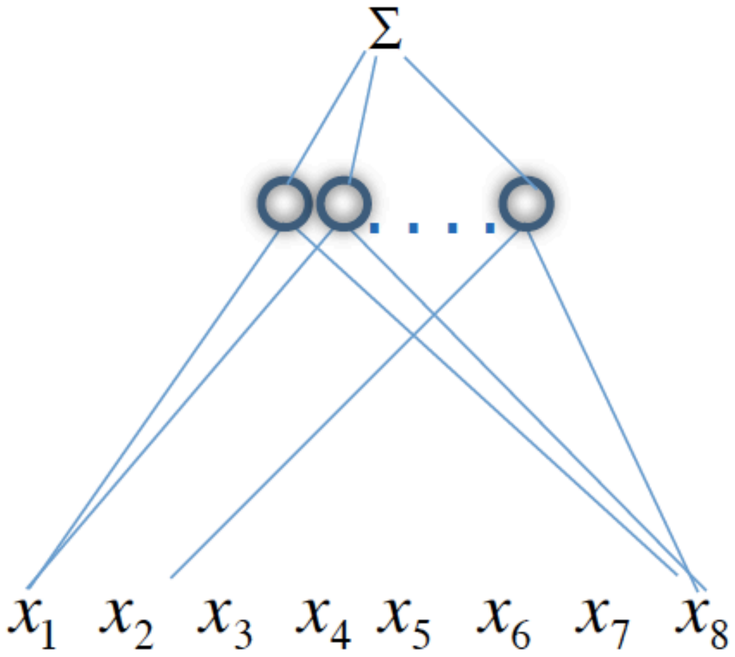
Multiple layers ==> *compositional sparsity* is new

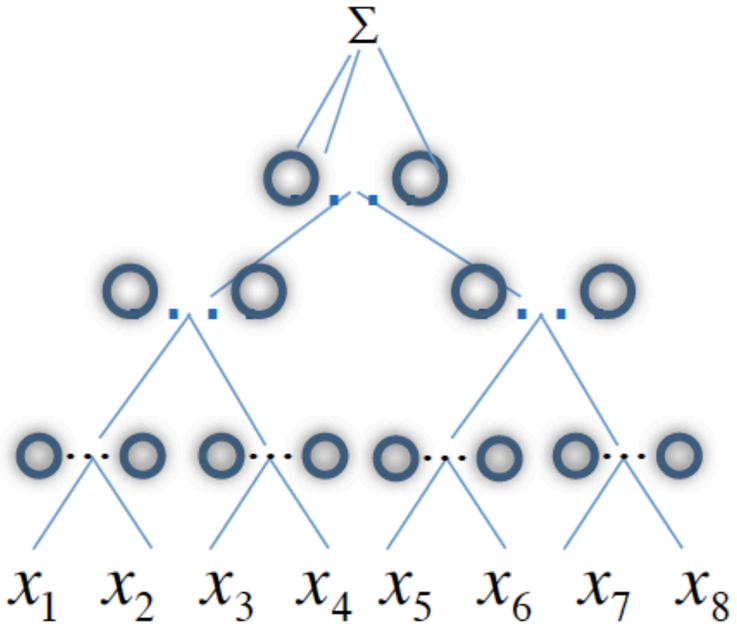
In the modern era.. “new” puzzles about approximation, generalization, optimization

- No curse of dimensionality
- VC bounds overparametrization
- Optimization unreasonably easy, non-convex non-PL but optimization works relaxing adding neurons

Deep and shallow networks: approximation universality

Theorem *Assume as target continuous functions on a compact domain. Shallow, one-hidden layer networks with a a nonlinear $\sigma(x)$ (which is not a polynomial) can approximate such targets arbitrarily well. Deep networks with a nonlinear $\sigma(x)$ (including polynomials) can also approximate such target functions arbitrarily well.*





$$\phi(x) = \sum_{i=1}^{r} c_i \left| < w_i , x > + b_i \right|_+$$

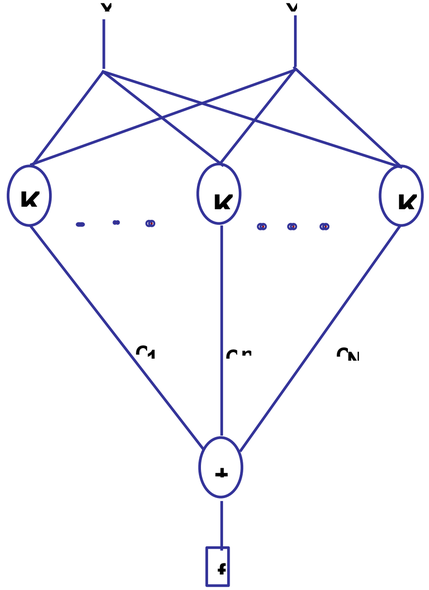
Cybenko, Girosi, ….

Typical shallow networks: Kernel Machines

$$\min_{f \in H} \left[\frac{1}{\ell} \sum_{i=1}^{\ell} V(f(x_i) - y_i) + \lambda \left\| f \right\|_K^2 \right]$$

implies

$$f(\mathbf{x}) = \sum_i^l \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

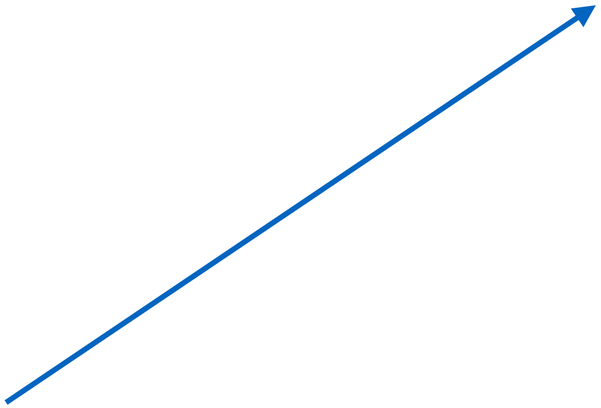


Equation includes splines, Radial Basis Functions and Support Vector Machines (depending on choice of V).

Degree (or rate) of approximation (theorem)

$$f(x_1, x_2, ..., x_8)$$

Both shallow and deep network can approximate a function of d variables equally well (density). The number of parameters in both cases depends exponentially on d as $N = O(\epsilon^{-\frac{d}{s}})$ or $\epsilon = O(N^{-\frac{s}{d}})$



Curse of dimensionality

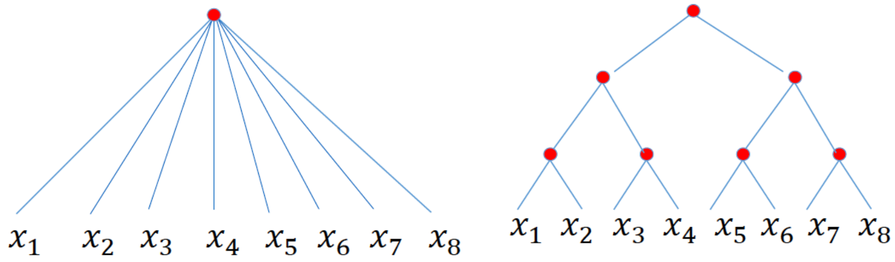
Mhaskar, Poggio, Liao, 2016

Key *sparse target function* property of the world

A generic function associated to the CIFAr problem $f : \mathscr{R}^{1000} \rightarrow \mathscr{R}$ requires a very large memory (10 bins per coordinate corresponds to 10^{1000}, numbers which is $>> N_{Edd} \approx 10^{80}$ the number of all the protons in the universe…there are ~ 10^{15} synapses in the human brain).

Kernel method *cannot* exploit compositional sparsity

For one-layer kernel methods the ambient dimension D (8 in the figure) controls (with smoothness) the sample complexity irrespectively of any (compositionally sparse) structure of f



$$f(\mathbf{x}) = \sum_i^l \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

Note: HyperBF can exploit sparsity $f(x) = \sum_i c_i K(\|x - x_i\|_M)$

One of these old questions: why hierarchies (in deep nets, in primate visual cortex)?

In Poggio and Smale (2003) we wrote "*A comparison with real brains offers another, and probably related, challenge to learning theory. The ``learning algorithms'' we have described in this paper correspond to one-layer architectures. Are hierarchical architectures with more layers justifiable in terms of learning theory?* Twenty years later, a most interesting theoretical question, both for machine learning and neuroscience, is indeed *why hierarchies, why depth.*

Outline: puzzles about approximation, optimization, generalization

- No Curse dimensionality (puzzle for approx/optimization)
- VC bounds are “very” vacuous for overparametrization
- Optimization unreasonably easy, non-convex non-PL but optimization works

Generalization bounds

In general generalization bounds are *quantitatively* meaningless but *qualitatively* useful.

- Quantitatively meaningless: because it is easier and better to to do empirical estimates of test error on data not used in training

- Qualitatively useful because they often provide an insight into mechanisms of generalization

Rademacher

Rademacher Complexity *Let $\mathbb{H}$ be a set of real-valued functions $h : \mathcal{X} \to \mathbb{R}$ defined over a set $\mathcal{X}$. Given a fixed sample $S \in \mathcal{X}^m$the empirical Rademacher complexity of $\mathbb{H}$ is defined as follows:*

$$\mathscr{R}_S(\mathbb{H}) := \frac{2}{m}\mathbb{E}_\sigma \left[\sup_{h\in\mathbb{H}} \left| \sum_{i=1}^{m} \sigma_i h(x_i) \right| \right] .$$

The expectation is taken over $\sigma = (\sigma_1, \ldots, \sigma_m)$, where, $\sigma_i \in \{\pm 1\}$are i.i.d. and uniformly distributed samples.

From Class 6

$$\mathbb{E}_S\left[\sup_{f\in\sqrt{G_0}}\left|\frac{1}{n}\sum_i \ell(y_i,f(x_i)) - \mathbb{E}_{x,y}\left[\ell(y,f(x))\right]\right|\right]$$

$$\leq 2\,\mathbb{E}_{S,\sigma}\left[\sup_{f\in\sqrt{G_0}}\left|\frac{1}{n}\sum_i \sigma_i\,\ell(y_i,f(x_i))\right|\right]$$

Rademacher Complexity of $\ell\circ f$

Expected error

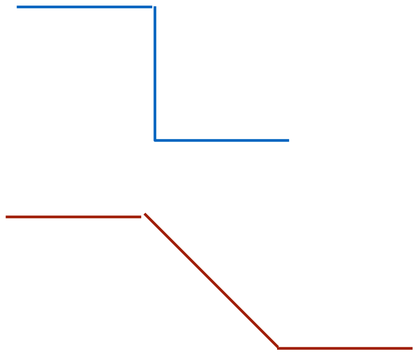
A typical generalization bound that holds with probability at least $(1-\delta)$, $\forall f \in \mathbb{F}$ has the form:

$$|L(g) - \hat{L}(g)| \leq \frac{1}{\gamma}\mathbb{R}_N(\mathbb{G}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}$$

where $L(g) = \mathbf{E}[\ell_{gamma}(g(x), y)]$ is the expected loss and $\hat{L}(g) = \frac{1}{N}\sum_n \ell_{gamma}(g(x_n), y_n)$ is the empirical loss

with

$$\ell_{gamma}(y, y') = \begin{cases} 1, & \text{if} \quad yy' \leq 0, \\ 1 - \frac{yy'}{\gamma}, & \text{if} \quad 0 \leq yy' \leq \gamma, \\ 0, & \text{if} \quad yy' \geq \gamma\,. \end{cases}$$



Rademacher Bounds

Theorem *Let* P *be a distribution over* $\mathbb{R}^d \times \{\pm 1\}$. *Let* $\mathbb{F} = \{f_W \mid \prod_{i=1}^{L} \|W_i\| \leq 1\}$. *Let* $g(x) = \rho f(x)$ *and* $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N}$ *be a dataset of i.i.d. samples selected from* P. *Then, with probability at least* $1 - \delta$ *over the selection of* $\mathcal{S}$, *for any* g_W, *we have*

$$|L(g) - \hat{L}(g)| \leq 2\mathscr{R}_{\mathcal{S}}(\mathbb{G}) + O\sqrt{\frac{1}{\delta N}}$$

Since $\mathscr{R}(\mathbb{G}) = \rho\mathscr{R}(\mathbb{F}) = \rho\dfrac{\mathscr{R}(\mathbb{F})}{N}$ *because* $g = \rho f$ *we have*

$$|L(g) - \hat{L}(g)| \leq 2\rho\mathscr{R}_{\mathcal{S}}(\mathbb{F}) + O\sqrt{\frac{1}{\delta N}}$$

Parenthesis

Covering numbers and bits

Let $(\mathcal{F}, \|\cdot\|)$ be a function class equipped with a norm or metric.
The *covering number* $N(\varepsilon, \mathcal{F}, \|\cdot\|)$ is the minimal number of balls of radius ε required to cover $\mathcal{F}$.
In other words, it measures how many "representative functions" are needed to approximate any $f \in \mathcal{F}$ within ε. Geometrically, it quantifies how large or rich the function class is at precision ε. N_ϵ measures *how large or rich* the function class is at precision ϵ

We need $log_2 N_\epsilon$ bits to represent a function in the space. In fact the metric entropy is

$$H(\varepsilon, \mathcal{F}, \|\cdot\|) := \log N(\varepsilon, \mathcal{F}, \|\cdot$$

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

CBMM Memo No. 91 **July 11, 2018**

Classical generalization bounds are surprisingly tight for Deep Networks

Qianli Liao[1], **Brando Miranda**[1], **Jack Hidary**[2] **and Tomaso Poggio**[1]
[1]Center for Brains, Minds, and Machines, MIT
[2]Alphabet (Google) X

Main points of Zhang et al. paper

Classical learning theory (VC dimension, Rademacher complexity, uniform convergence) suggests that generalization depends on capacity control relative to sample size. Yet modern deep networks have enormous capacity, often far exceeding the number of training samples — and still generalize well in practice.

The authors designed experiments to probe this paradox:

- *Random Labels Experiments*
  Deep networks can fit completely random labels on standard datasets (CIFAR-10, ImageNet). They achieve near-zero training error even when labels are permuted or when inputs are replaced with random noise. This shows networks have more than enough capacity to memorize arbitrary data.
- *Classical Complexity Measures Fail*
  Measures like VC dimension, Rademacher complexity, stability, etc., are too loose: they do not explain why the same architecture both memorizes random data and generalizes on real data.

Main claims (not completely correct) of paper

1. Deep nets can memorize random data. So their effective capacity is huge.
2. Standard generalization bounds are vacuous in practice — they don't distinguish between learning structure vs. memorization.
3. New theory is required. We need explanations that go beyond uniform capacity control, perhaps focusing on implicit bias, algorithmic stability of SGD, margin-based perspectives, compression schemes, or data-dependent priors.

This paper:

- Broke the assumption that explicit regularization was the main reason for generalization.
- Pushed the field toward more refined complexity measures (e.g. norm-based, compression-based, margin-based).

But….

…we would like to show that classical generalization theory -- in the form of norm-based Rademacher complexity bounds -- can predict qualitatively the generalization error of these overparametrized models in the case of random labels and in the case of natural labels. In particular, some of the claims in the paper are wrong (in an interesting way), and the test performance can be predicted by data collected in the training stage.

The key observation is that the Rademacher complexity satisfies the property:

$\mathbb{R}_N(\mathbb{G}) = \rho_1 \cdots \rho_K \mathbb{R}_N(\mathbb{F})$ where $g(x)$ is the function represented by the network and $f(x)$ is the function with normalized weight matrices.

Then, for the unnormalized cross-entropy loss the bound gives

$$L(g) \leq \prod_{k=1}^{K} \rho_k \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2N}},$$

since $\hat{L}(g) \approx 0$, whereas for the normalized cross-entropy loss it yields

$$L(f) \leq \hat{L}(f) + \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2N}}$$

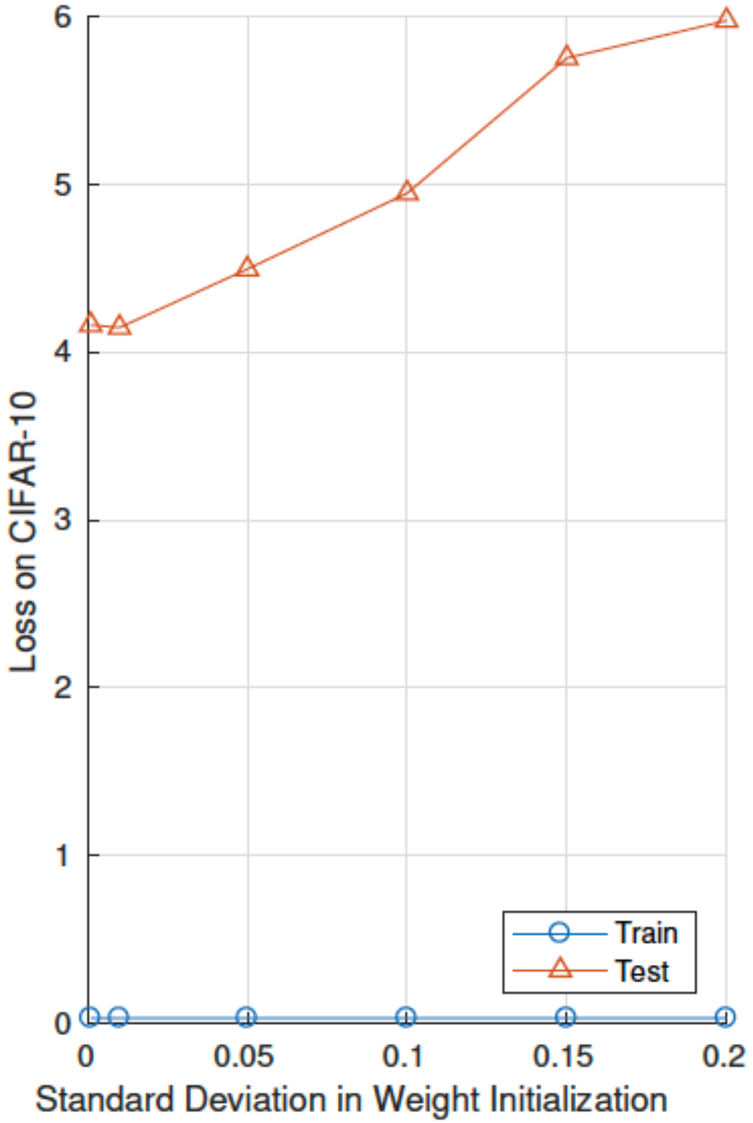
A “lemma” on expected error

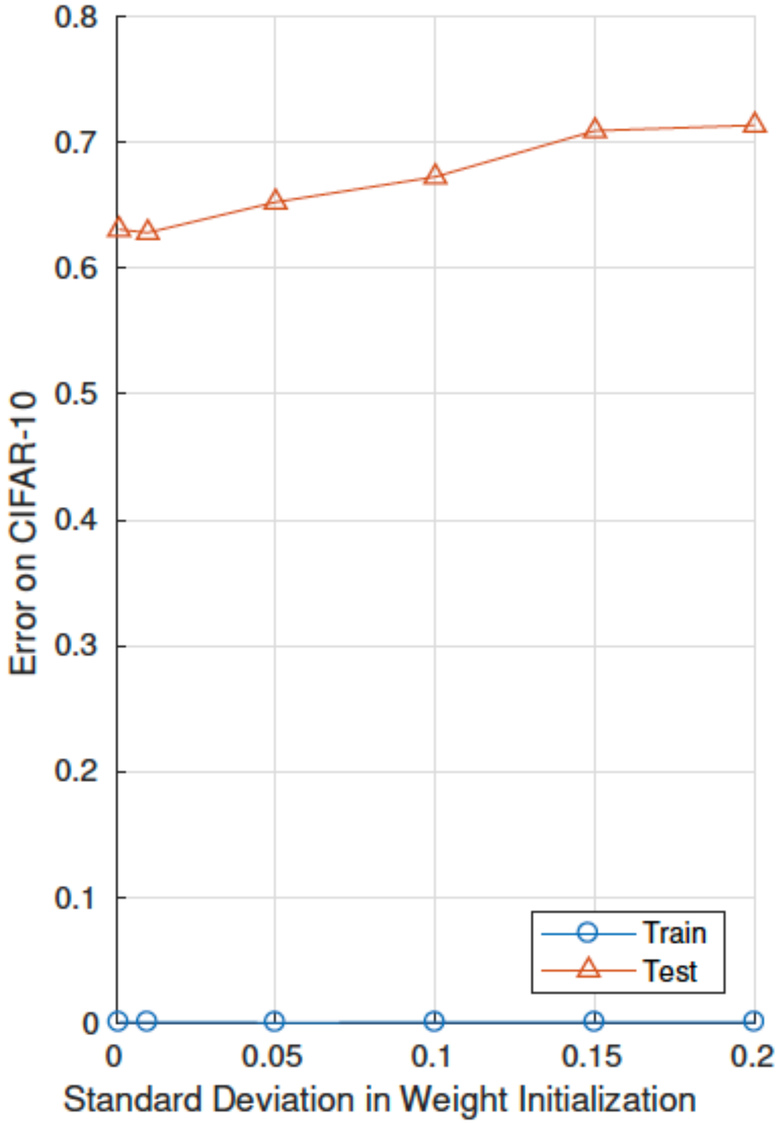
Solutions with smaller ρ_{eq} have a better upper bound on the expected error.

Consider two solutions with zero empirical loss of the square loss regression problem obtained with the same RELU deep network and corresponding to two different minima with two different ρ’s. Let us call them $\rho_a f^a(x)$ and $\rho_b f^b(x)$ and assume that $\rho_a < \rho_b$.
The bounds have the form $L_0(f^a) \leq A\rho_a + \epsilon$ and $L_0(f^b) \leq A\rho_b + \epsilon$.

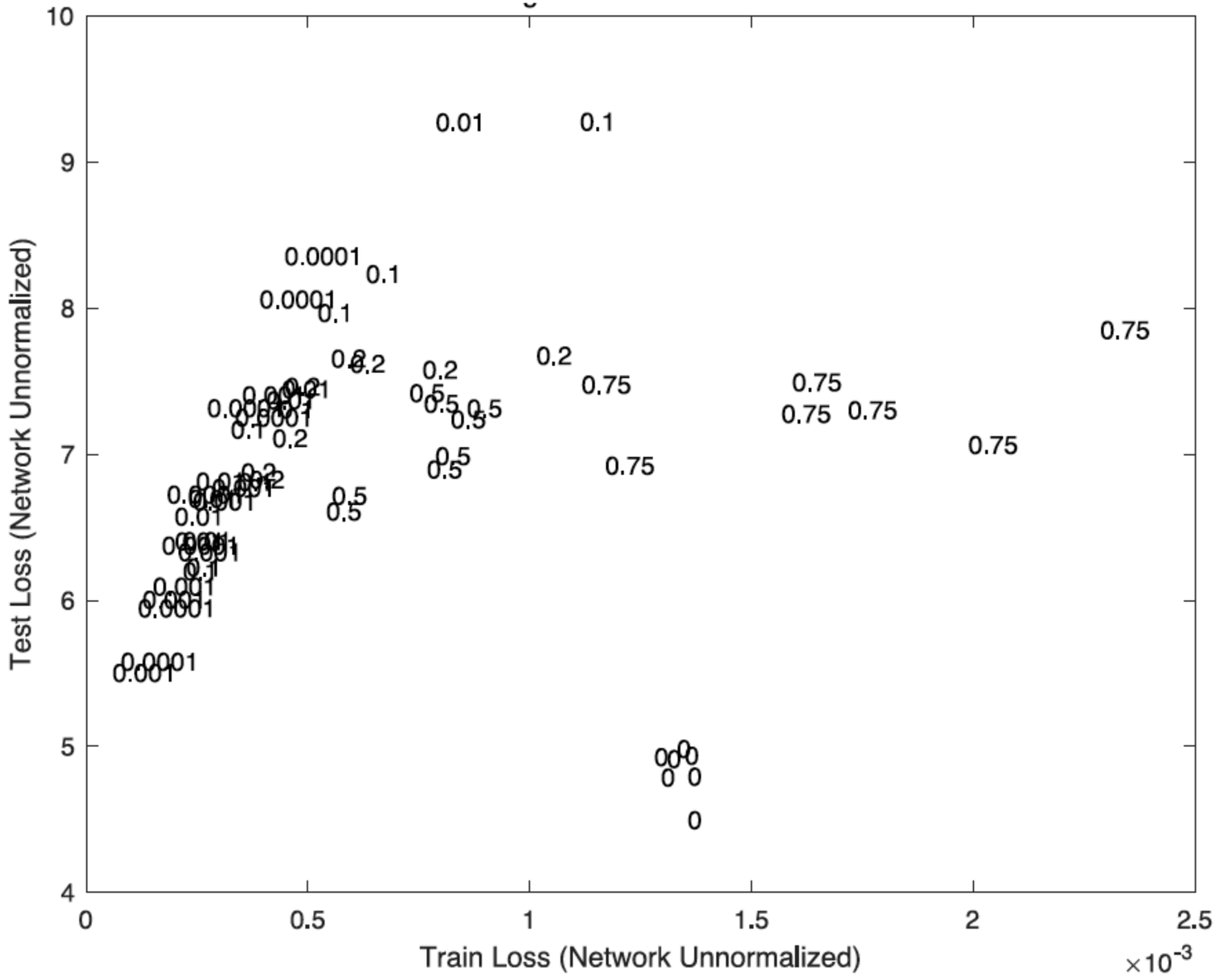
Thus *the upper bound for the expected error $L_0(f^a)$ is better than the bound for $L_0(f^b)$*

How to generate minima with zero training error but different test errors

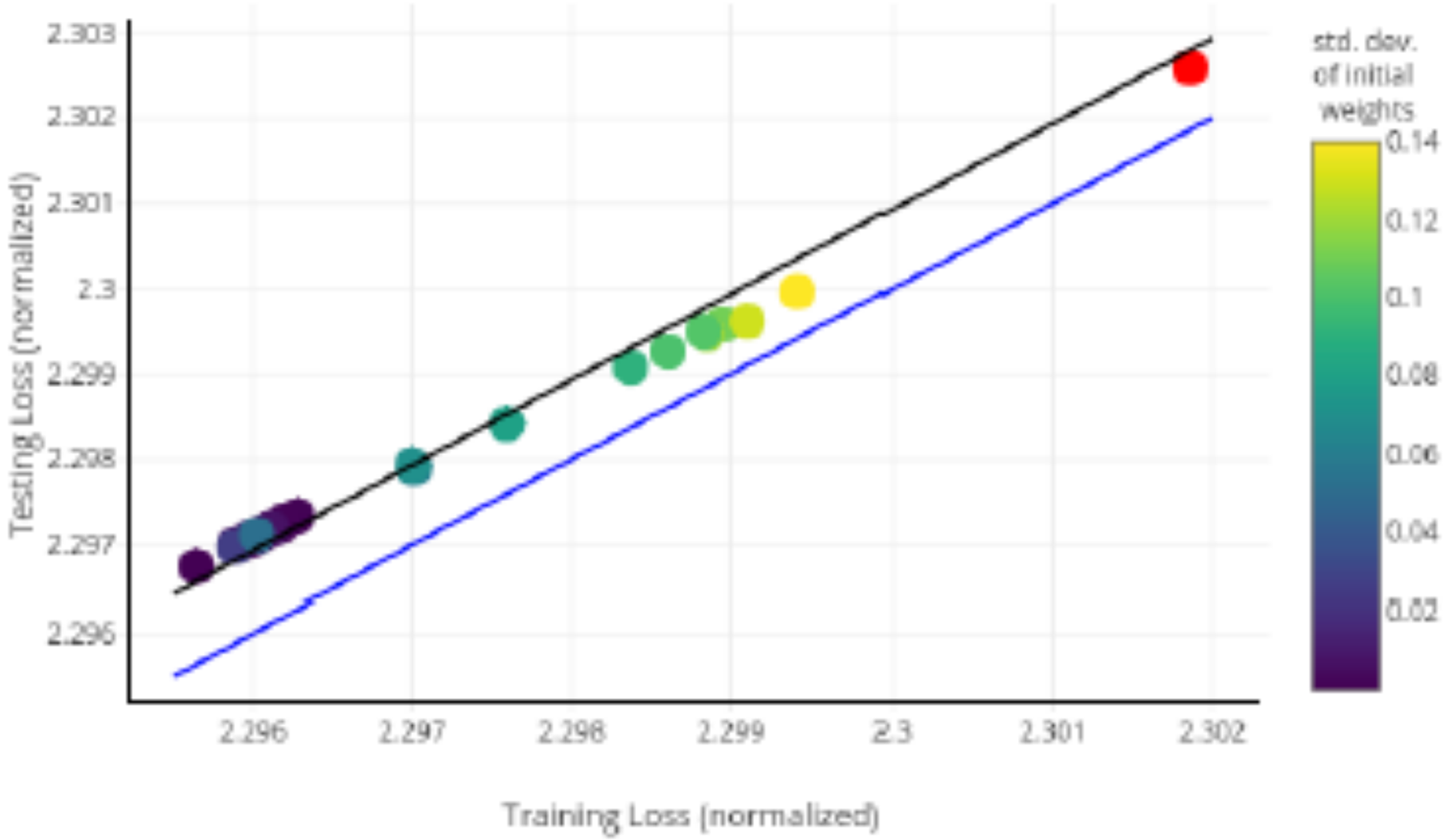




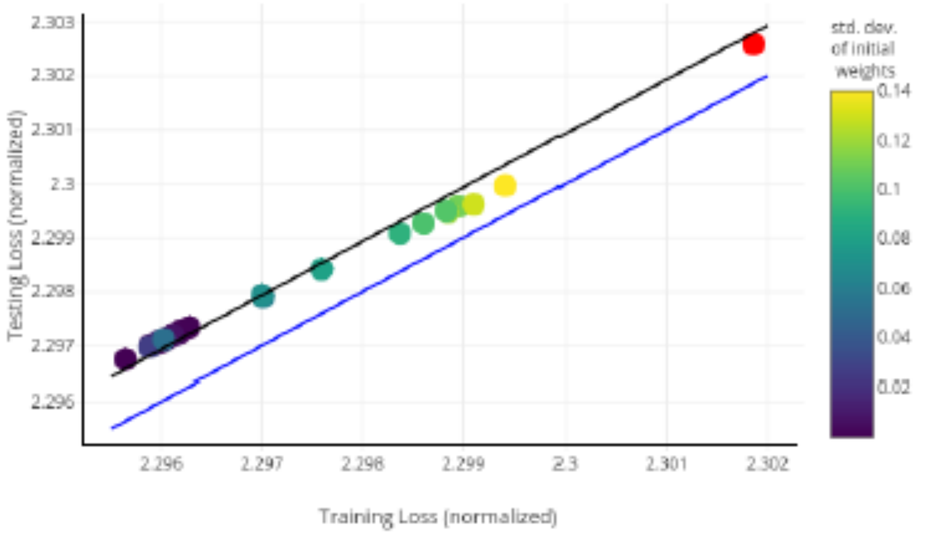
Unnormalized mess



Predicting test performance from training performance



$$L(f) \leq \hat{L}(f) + \mathbb{R}_N(\mathbb{F}) + c_2\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2N}}$$



In our experiments the terms $\mathbb{R}_N(\mathbb{F}) + c_2\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{2N}}$ are small, implying that the normalized test loss is very close to the normalized train loss. Thus $L(f)$ is bounded by $\hat{L}(f)$, predicting a bound in terms of a linear relationship with slope one and a small offset between $L(f)$ and $\hat{L}(f)$. The prediction is verified experimentally (see Figure). Thus the equation shows that the training cross-entropy loss is a very good proxy of the cross-entropy loss at testing, implying generalization for a relatively small number of examples N.

So where do we stand re: Zhang et al. paper

- The classical bounds qualitatively *work:* one can predict the test error from measurements at training (e.g the “norm” of the weights of a deep net)

- The classical bounds are very loose (so much that there are claims we need a different theory): we will show (in Class 14) that sparsity implies much tighter bounds

- Zhang et al. are still correct that we cannot predict how the Rademacher complexity depends on network architecture and optimization. This is still an open question.

Parenthesis

Covering numbers and bits

Let $(\mathcal{F}, \|\cdot\|)$ be a function class equipped with a norm or metric.
The *covering number* $N(\varepsilon, \mathcal{F}, \|\cdot\|)$ is the minimal number of balls of radius ε required to cover $\mathcal{F}$.
In other words, it measures how many "representative functions" are needed to approximate any $f \in \mathcal{F}$ within ε. Geometrically, it quantifies how large or rich the function class is at precision ε. N_ϵ measures *how large or rich* the function class is at precision ϵ

We need $log_2 N_\epsilon$ bits to represent a function in the space. In fact the metric entropy is

$$H(\varepsilon, \mathcal{F}, \|\cdot\|) := \log N(\varepsilon, \mathcal{F}, \|\cdot$$

Relation Between Them

VC dimension controls growth of Rademacher complexity.

- If $|\mathscr{H}| < \infty$:

$$\mathfrak{R}_n(\mathscr{H}) \leq \sqrt{\frac{2\log|\mathscr{H}|}{n}}.$$

- If $\mathrm{VC}(\mathscr{H}) = d < \infty$:

$$\mathfrak{R}_n(\mathscr{H}) \leq C\sqrt{\frac{d\log n}{n}}.$$

Thus: finite VC dimension *implies* vanishing Rademacher complexity.

Generalization Bounds (VC)

With probability $1-\delta$:

$$\sup_{h\in\mathscr{H}}\left|\mathbb{E}[h(X)]-\frac{1}{n}\sum_{i=1}^{n}h(x_i)\right| \leq C\sqrt{\frac{d\log(n/d)+\log(1/\delta)}{n}}.$$

Distribution-free, worst-case.

Generalization Bounds (Rademacher)

With probability $1-\delta$

$$\sup_{h\in\mathscr{H}}\left|\mathbb{E}[h(X)]-\frac{1}{n}\sum_{i=1}^{n}h(x_i)\right| \leq 2\hat{\mathfrak{R}}_S(\mathscr{H})+3\sqrt{\frac{\log(2/\delta)}{2n}}.$$

Adaptive to sample S.

Comparison

- VC bound: combinatorial, worst-case;
- Rademacher bound: data-dependent, usually tighter;
- Relation: $\mathbf{VC}(\mathscr{H}) = d$ implies $\mathfrak{R}\_n(\mathscr{H}) = O(\sqrt{d \log n / n})$.

VC Dimension of Deep Networks

* Neural networks with W parameters and piecewise-linear activations (e.g. ReLU):

$$\mathrm{VC}(\mathscr{H}) = \Theta(W \log W).$$

* Sample complexity:

$$n = O\left(\frac{W \log W + \log(1/\delta)}{\epsilon^2}\right).$$

Upper direction: metric entropy $\Rightarrow$ **small chains** $\Rightarrow$ **small \Rad (Dudley).**

Lower direction: large **\Rad forces large entropy at some scale (Sudakov).**

Hence, **they bound each other** up to constants

Summary

- VC dimension: worst-case combinatorial capacity;
- Rademacher complexity: data-dependent complexity;
- Both give uniform convergence guarantees;
- For neural nets: $\mathbf{VC}(\mathscr{H}) = \Theta(W \log W)$, so sample complexity scales quasi-linearly with number of parameters.

Another footnote

Outline: puzzles about approximation, optimization, generalization

- No Curse dimensionality (puzzle for approx/optimization)
- VC bounds overparametrization (puzzle for generalization)
- Optimization should not work this way:
  - Non-Convexity
  - Instability
  - About Location

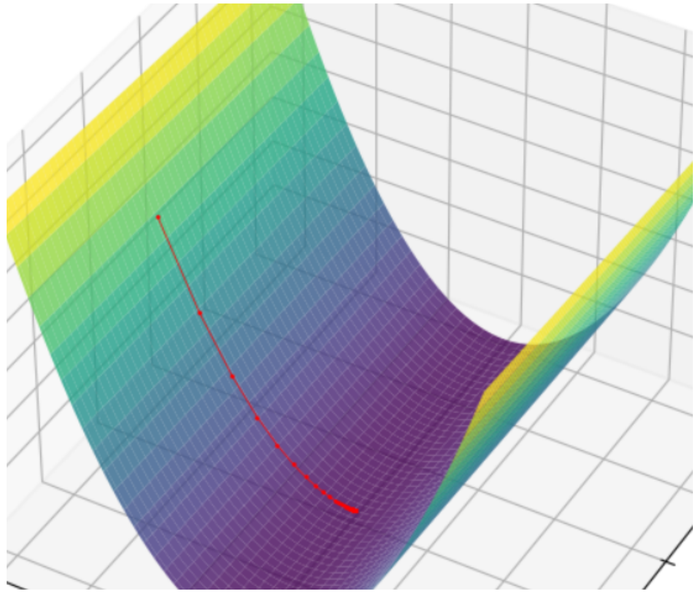
Training is a minimization

In Class 4 we saw when optimization algorithm are meant to work and why

- **This is not a class where we teach material.**
- **Today we *pitch* what comes next.**
- **Now we care about** *Neural Networks.*
- **Many of the things we saw with Lorenzo are** *not that easy* **anymore:**

They break!

Today we see what some of them are!

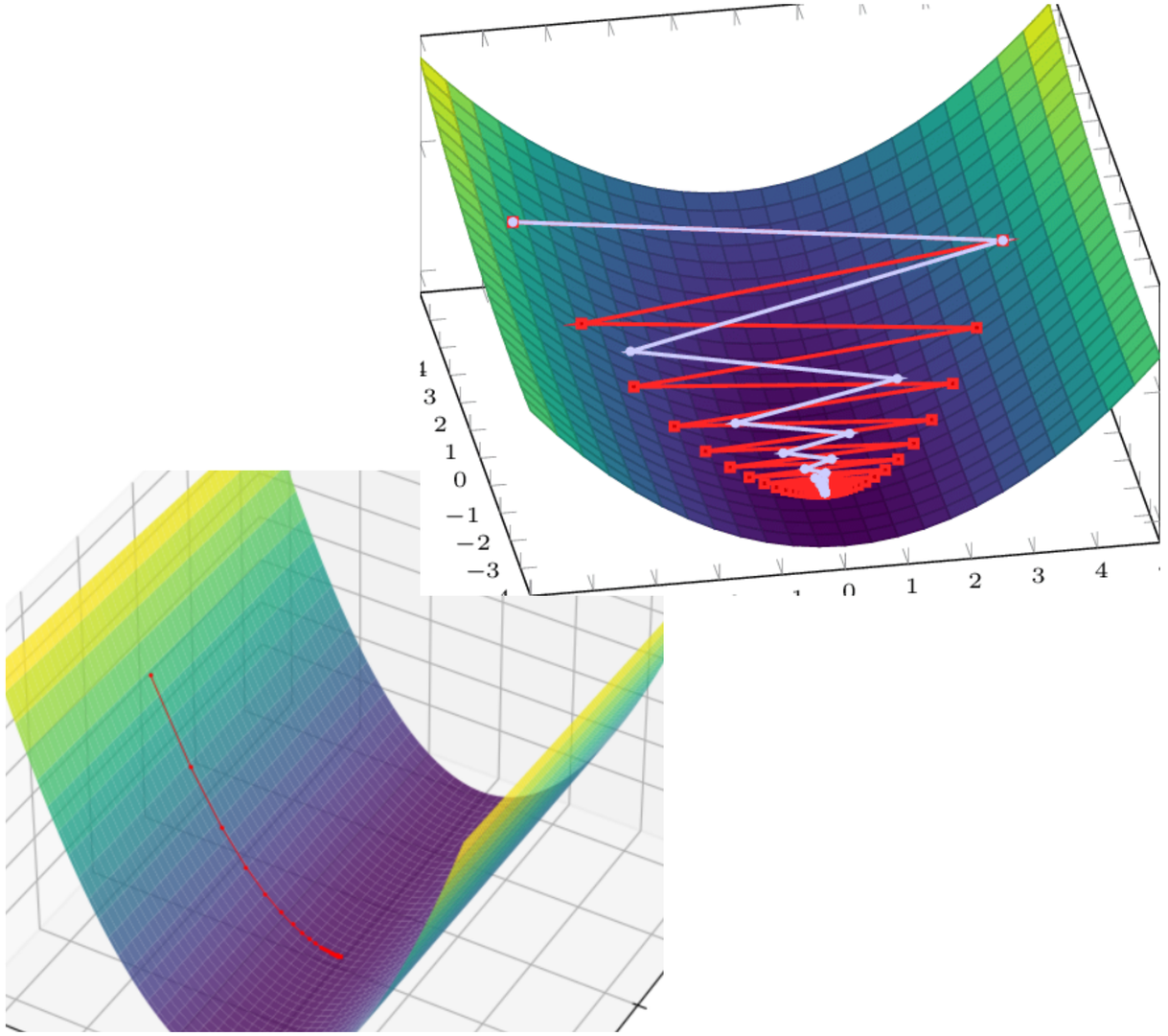


vs

Back to optimization

- Let $\theta = (\theta_1, \theta_2, \ldots, \theta_d) \in \mathbb{R}^d$
- Let $L\colon \mathbb{R}^d \to \mathbb{R}$
- I want $\theta \in \mathbb{R}^d$ that minimizes $L(\theta)$:

$$\min_{\theta=(\theta_1,\theta_2,\ldots)} L(\theta)$$



What algorithms

- The dimensionality d of the problem $\left(\theta \in \mathbb{R}^d\right)$ is very high, e.g. 1.7 trillion for GPT-4.
- We have many n data points $x \in D$ and the loss is an average over them:

$$\frac{1}{n}\sum_{x \in D} L(\theta, x)$$

$\Longrightarrow$ **Gradient-based**

$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k)$$

Let us work with

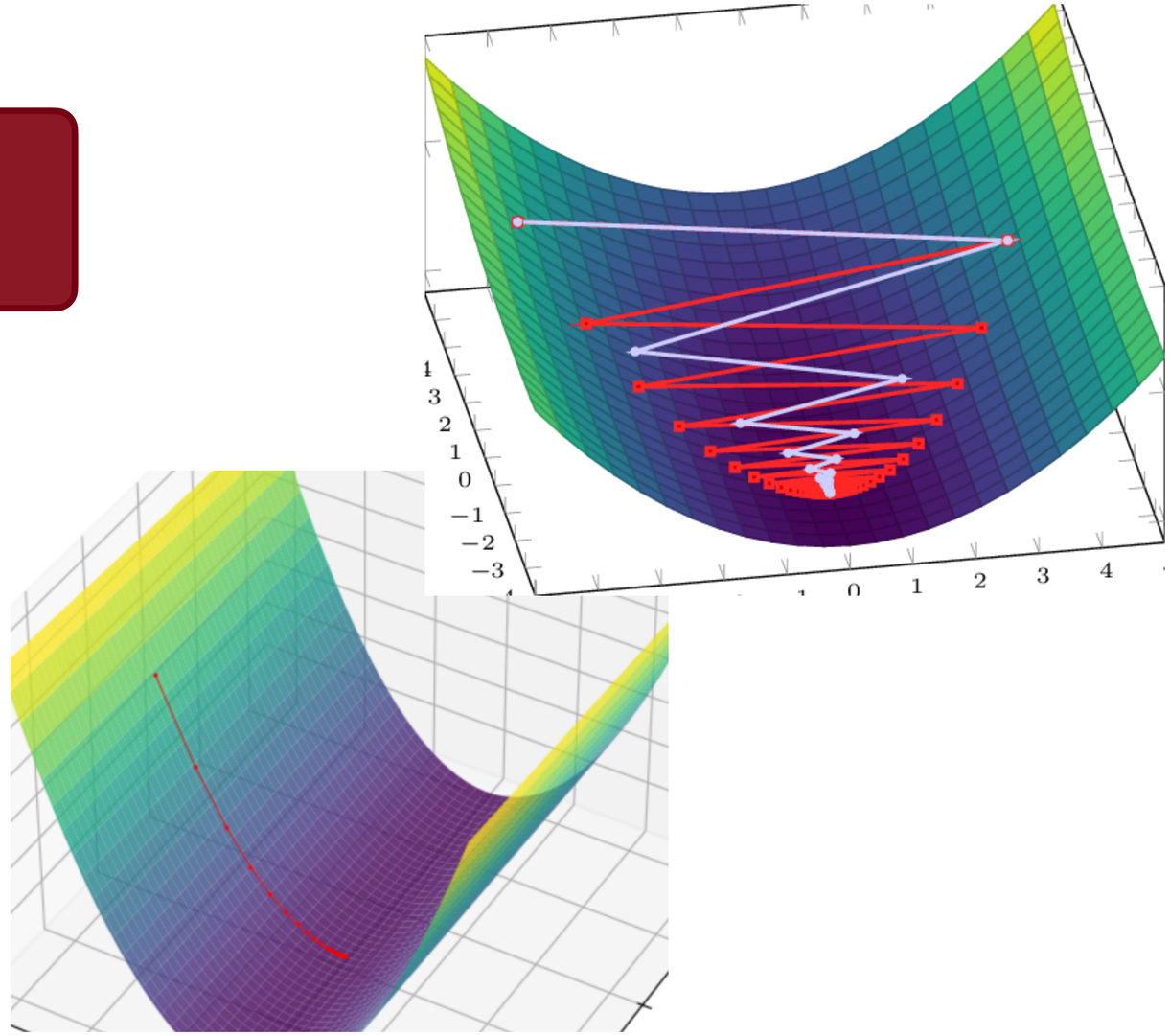
Mini-batch gradient-based methods

$\Longrightarrow$ **Mini-batch methods**

When Do They Work?

The classical setting:

- The function is convex.
- The step size is small.



Both these two things break for Neural Networks:

- The convexity.
- The existence of B.

Minimization

From Class 4

Assume ℓ convex and continuous, let

$$\widehat{L}_\lambda(w) = \widehat{L}(w) \; + \; \lambda \|w\|^2.$$

θ is now w

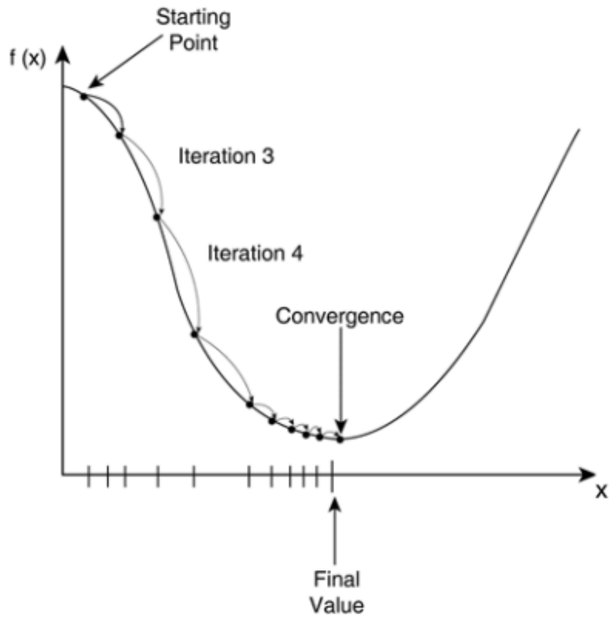
- Coercive[1], continuous, strongly convex functional
  $\Rightarrow$ a minimizer exists and is unique.

- Computations depends on the considered loss.

[1] $\lim_{\|w\| \to \infty} \widehat{L}_\lambda(w) = \infty$.

Gradient descent

From Class 4



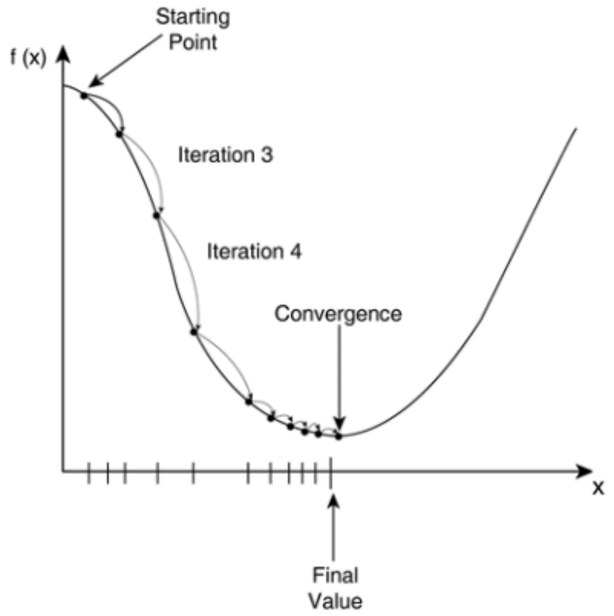
Let $F : \mathbb{R}^d \to \mathbb{R}$ differentiable,

$$w_0 = 0, \quad w_{t+1} = w_t - \eta \nabla F(w_t),$$

θ is now w
L is F

Gradient descent

From Class 4



Let $F : \mathbb{R}^d \to \mathbb{R}$ differentiable, convex and such that

$$\|\nabla F(w) - \nabla F(w')\| \leq B\|w - w'\|$$

(e.g. $\sup_w \|\underbrace{\nabla^2 F(w)}_{\text{Hessian}}\| \leq B$)

Then

$$w_0 = 0, \quad w_{t+1} = w_t - \frac{1}{B}\nabla F(w_t),$$

converges to a minimizer w^* of F.

γ STEP SIZE

$\gamma = 1/B$

θ is now w

B will be $\lambda_{\max}$

$\eta = 1/B$ here!

Function value rates

From Class 4

Rates can be obtained under different assumptions. We consider two case:

Convex + B-smooth:

$$F(w_t) - F(w^\star) \leq \frac{B\|w_0 - w^\star\|^2}{2t}.$$

θ is now w

B will be $\lambda_{\max}$!

GD rate: convex + smooth

From Class 4

If F is convex and B-smooth, then gradient descent with step size $\eta = \frac{1}{B}$ satisfies

$$F(w_t) - F(w^\star) \leq \frac{B\|w_0 - w^\star\|^2}{2t}.$$

Proof steps:

1. Descent lemma
2. Convexity
3. Combining
4. Telescoping
5. Final bound

θ is now w
B will be $\lambda_{\max}$
$\eta = 1/B$ here!

Descent lemma

From Class 4

If F is B-smooth, then for all u, v, the following quadratic bound holds

$$F(v) \leq F(u) + \nabla F(u)^\top (v - u) + \frac{B}{2}\|v - u\|^2.$$

- Expand F in Taylor
- Bound with smoothness the 2^{nd} term
- Show that if $\eta \leq 2/B$ we go down.

Descent lemma (cont.)

From Class 4

$$F(v) \leq F(u) + \nabla F(u)^\top (v-u) + \frac{B}{2}\|v-u\|^2.$$

In particular, for GD $w_{t+1} = w_t - \frac{1}{B}\nabla F(w_t)$, taking $v = w_{t+1}$, $u = w_t$

$$F(w_{t+1}) \leq F(w_t) - \frac{1}{2B}\|\nabla F(w_t)\|^2.$$

θ is now w

B will be $\lambda_{\max}$

$\eta = 1/B$ here!

GD rate: convex + smooth

From Class 4

If F is convex and B-smooth, then gradient descent with step size $\eta = \frac{1}{B}$ satisfies

$$F(w_t) - F(w^\star) \leq \frac{B\|w_0 - w^\star\|^2}{2t}.$$

Proof steps:

1. Descent lemma
2. Convexity
3. Combining
4. Telescoping
5. Final bound

Convexity

From Class 4

If F is convex, then for all u, v,

$$F(u) - F(v) \leq \nabla F(u)^\top (u - v).$$

In particular, with $u = w_t$ and $v = w^\star$:

$$F(w_t) - F(w^\star) \leq \nabla F(w_t)^\top (w_t - w^\star).$$

- Let us now see *how:*

Both these two things break for Neural Networks:

- **The convexity.**
- **The existence of B.**

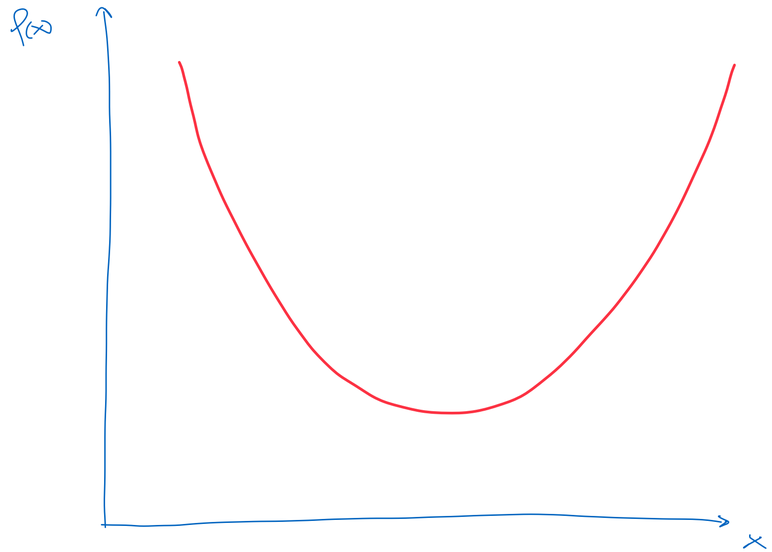
Outline: puzzles about approximation, optimization, generalization

- No Curse dimensionality (puzzle for approx/optimization)
- VC bounds overparametrization (puzzle for generalization)
- Optimization should not work this way:
  - Non-Convexity
  - Instability
  - About Location

Convex functions

- A convex function is a function that is curved up-ward at every point.

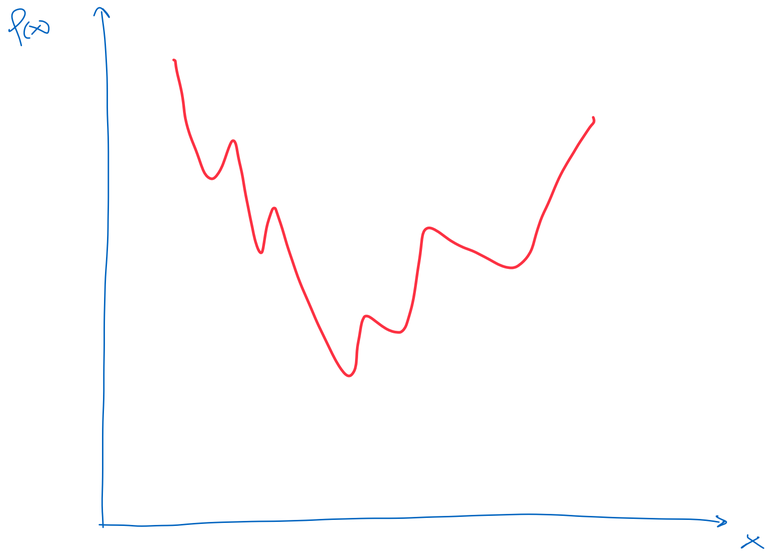
- A convex function is a function in which if you go down you go to a minimum.



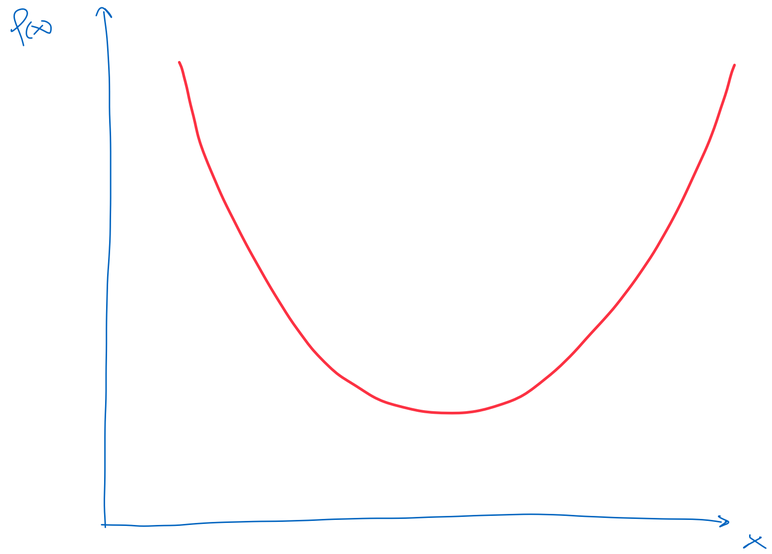
Convexity is about:
the local steps
work out globally

Convex

Non Convex

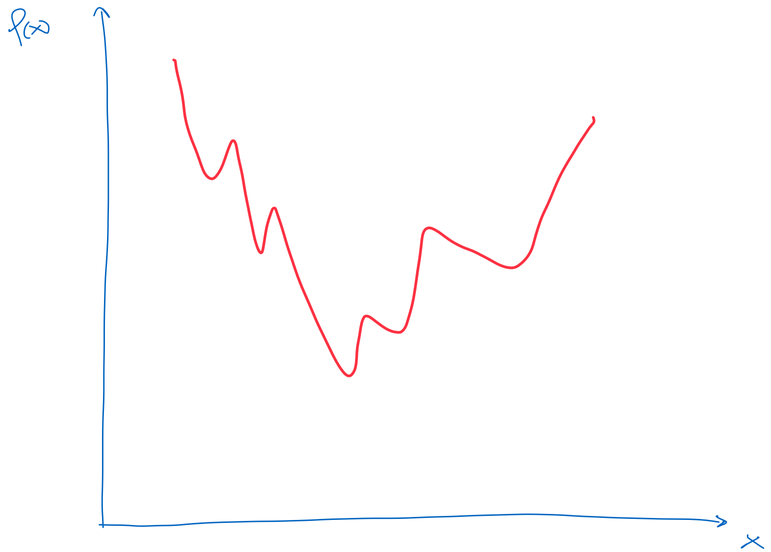


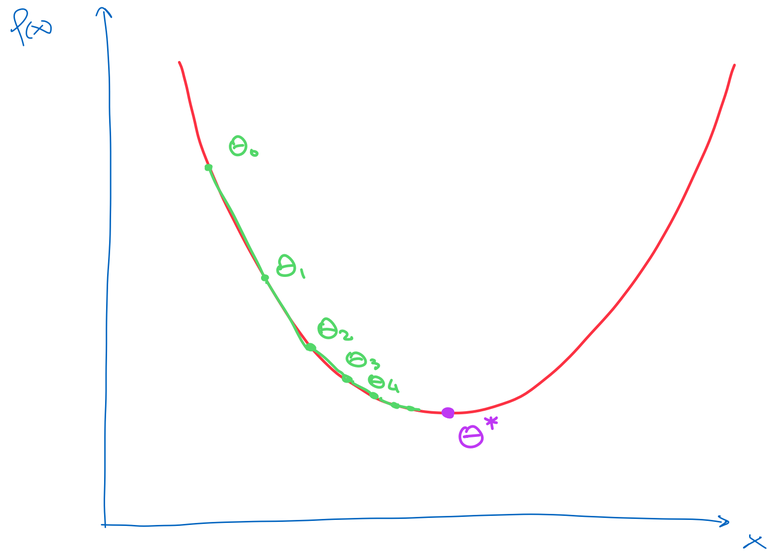
Non-Convex



Convex

What do we expect from a non convex function

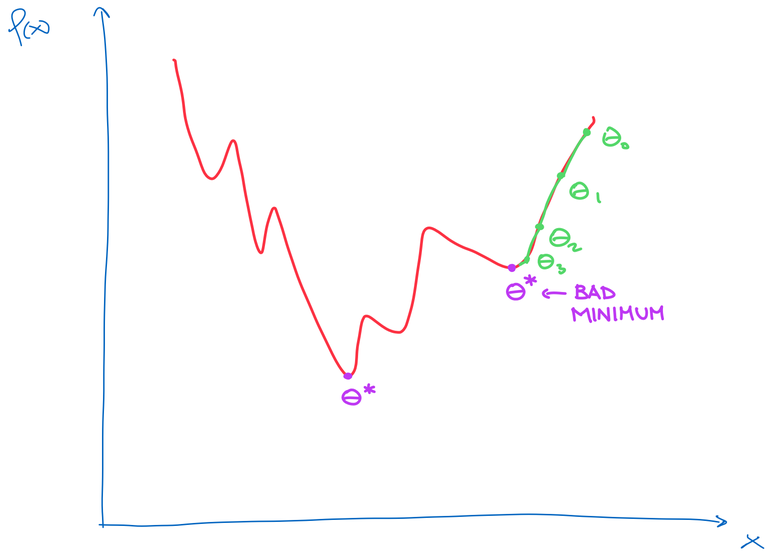




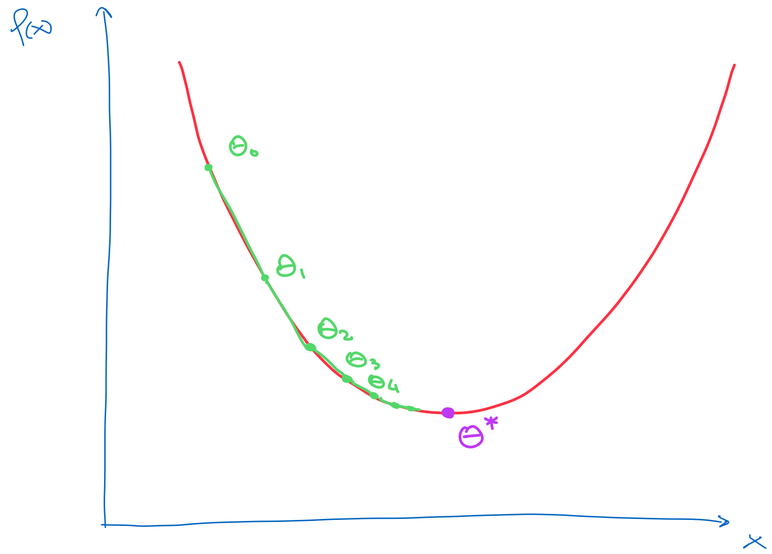
Non-Convex

Convex

What do we expect from a non convex function



Non-Convex

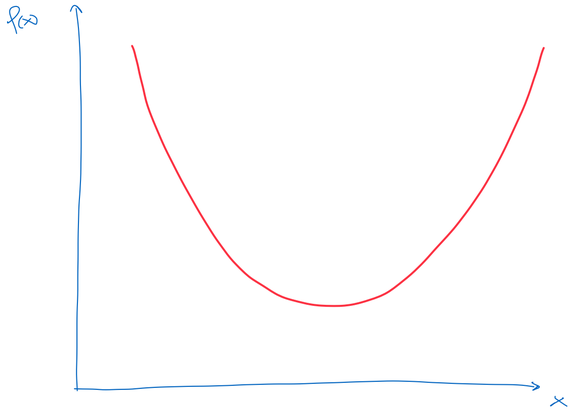


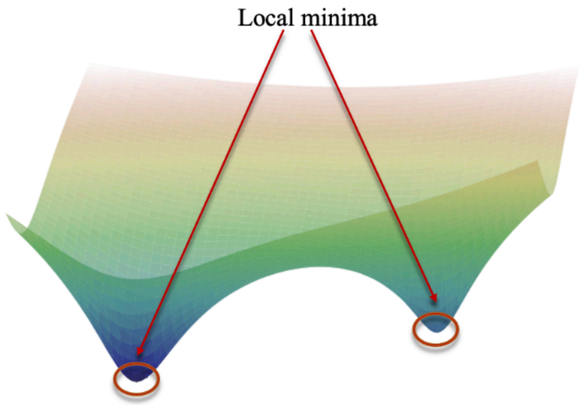
Convex

Convex functions

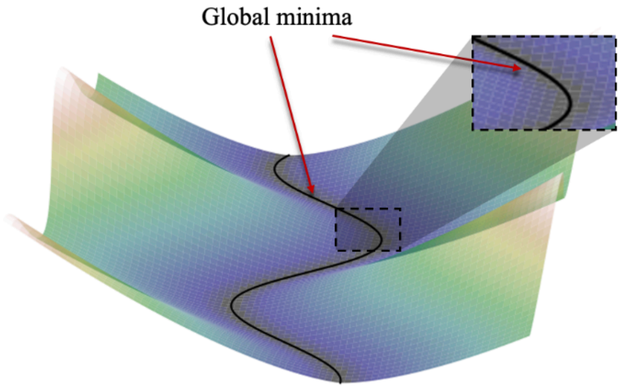
Is Neural Networks' training loss convex?

Let's play with an example





(a) Loss landscape of under-parameterized models

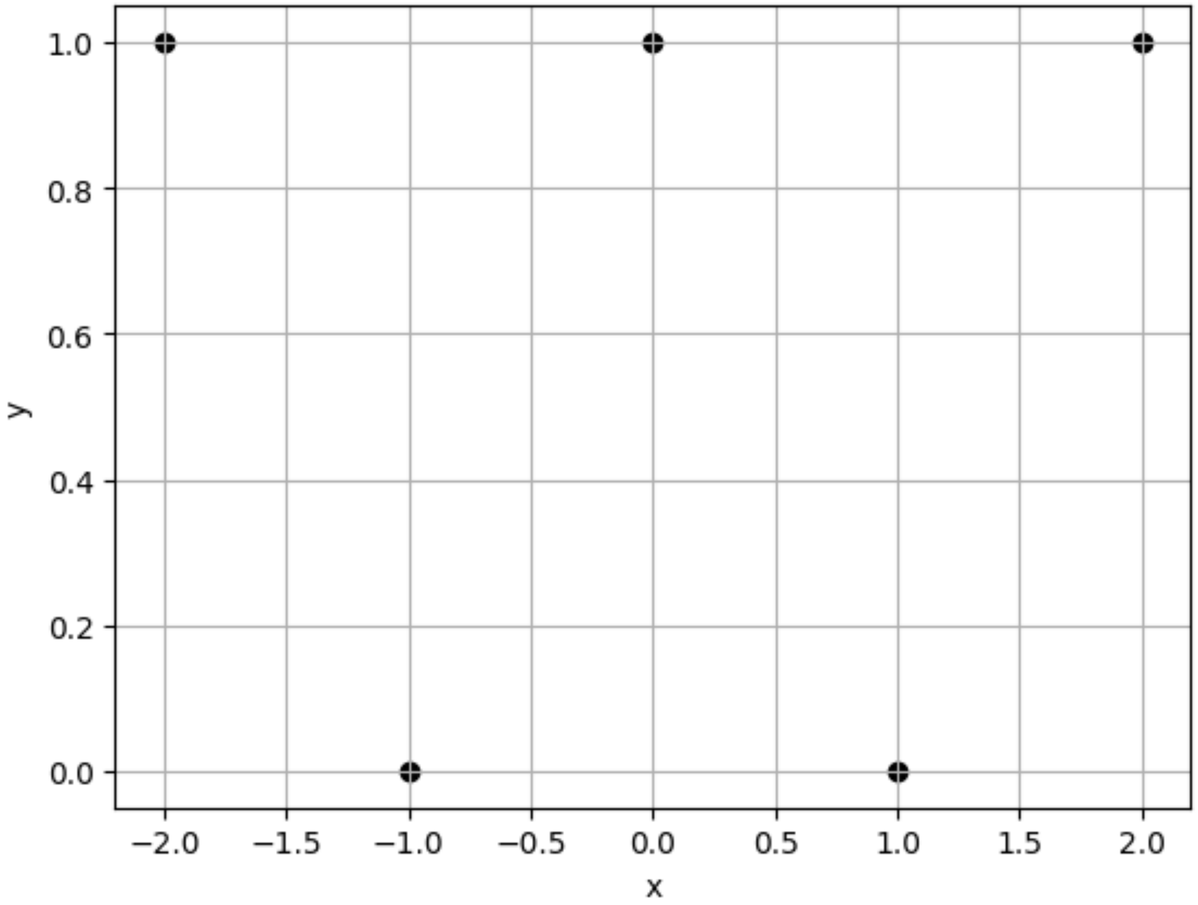


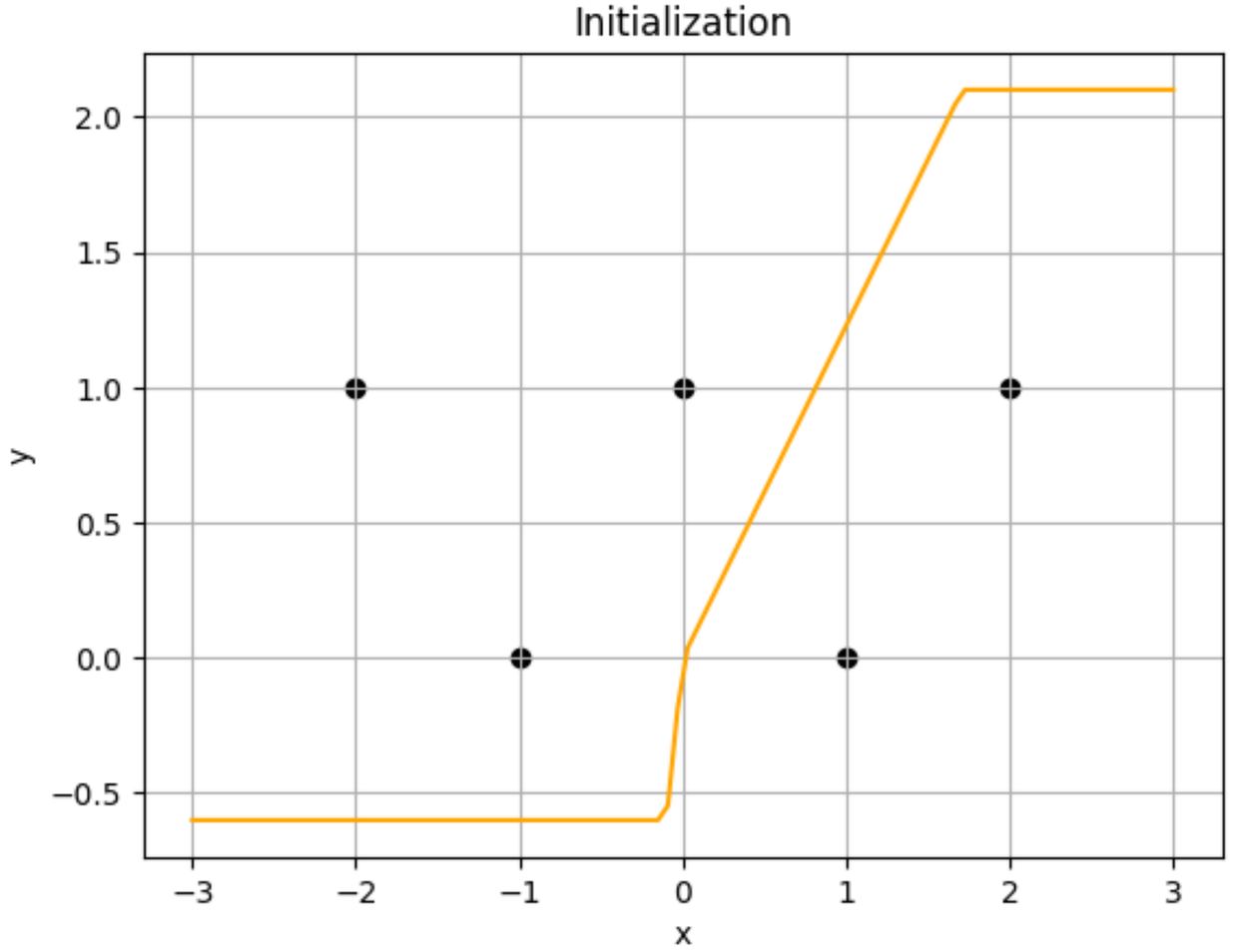
(b) Loss landscape of over-parameterized models

Liu, Zhu, Belkin, (2020), Arxiv link.

Landscape for Neural Networks

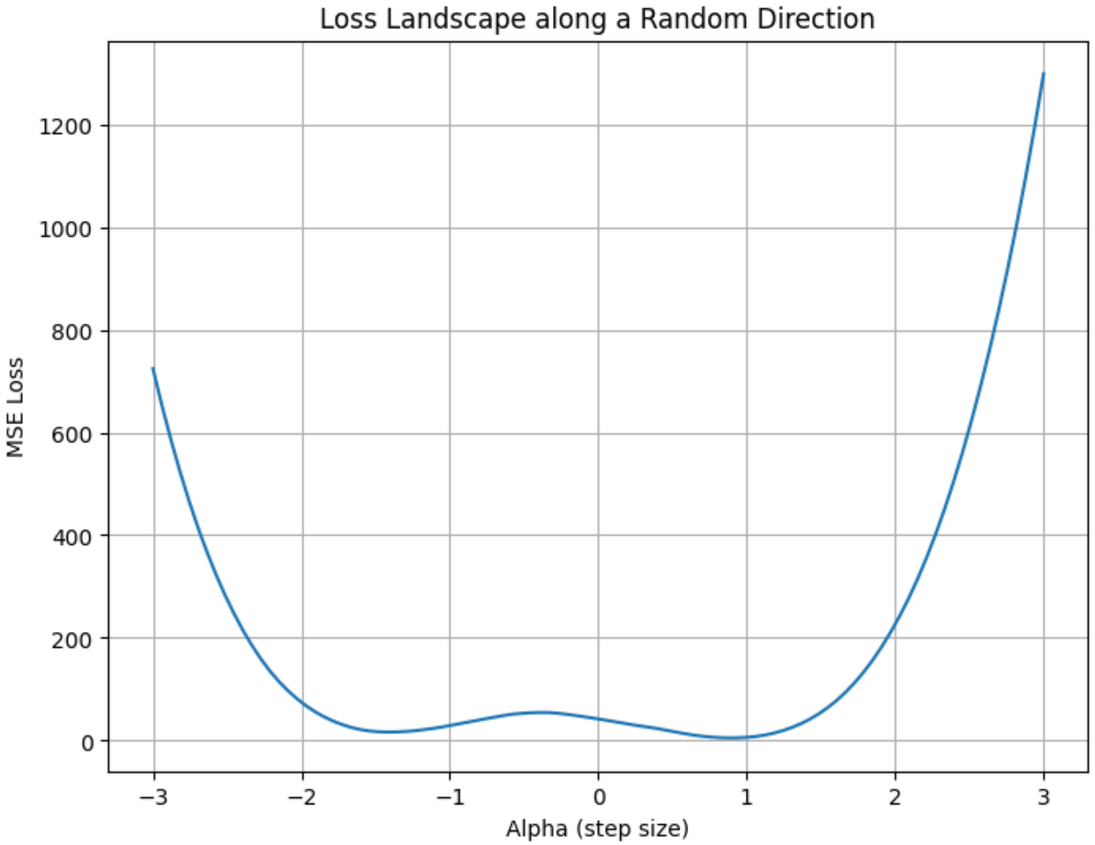
Data on a W

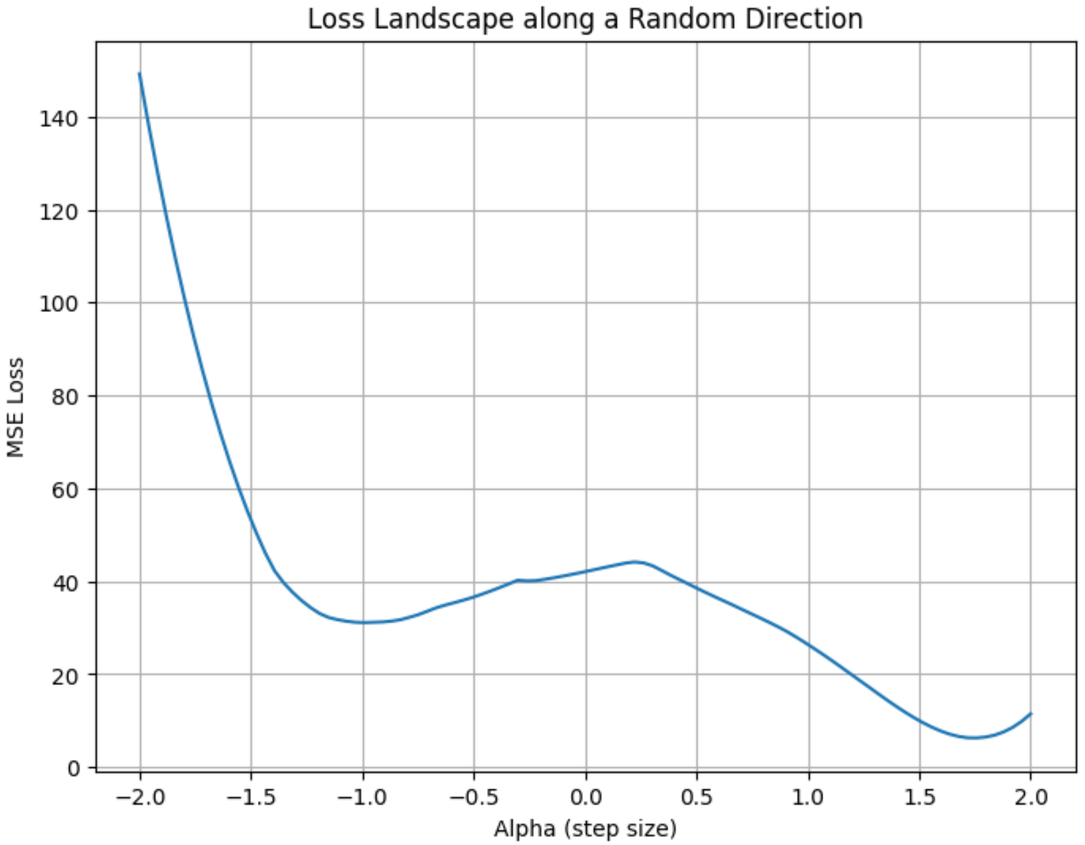




Model $f_{a,b,c}(x) = \sum_{i=1}^{20} a_i \cdot ReLU(b_i x + c_i)$

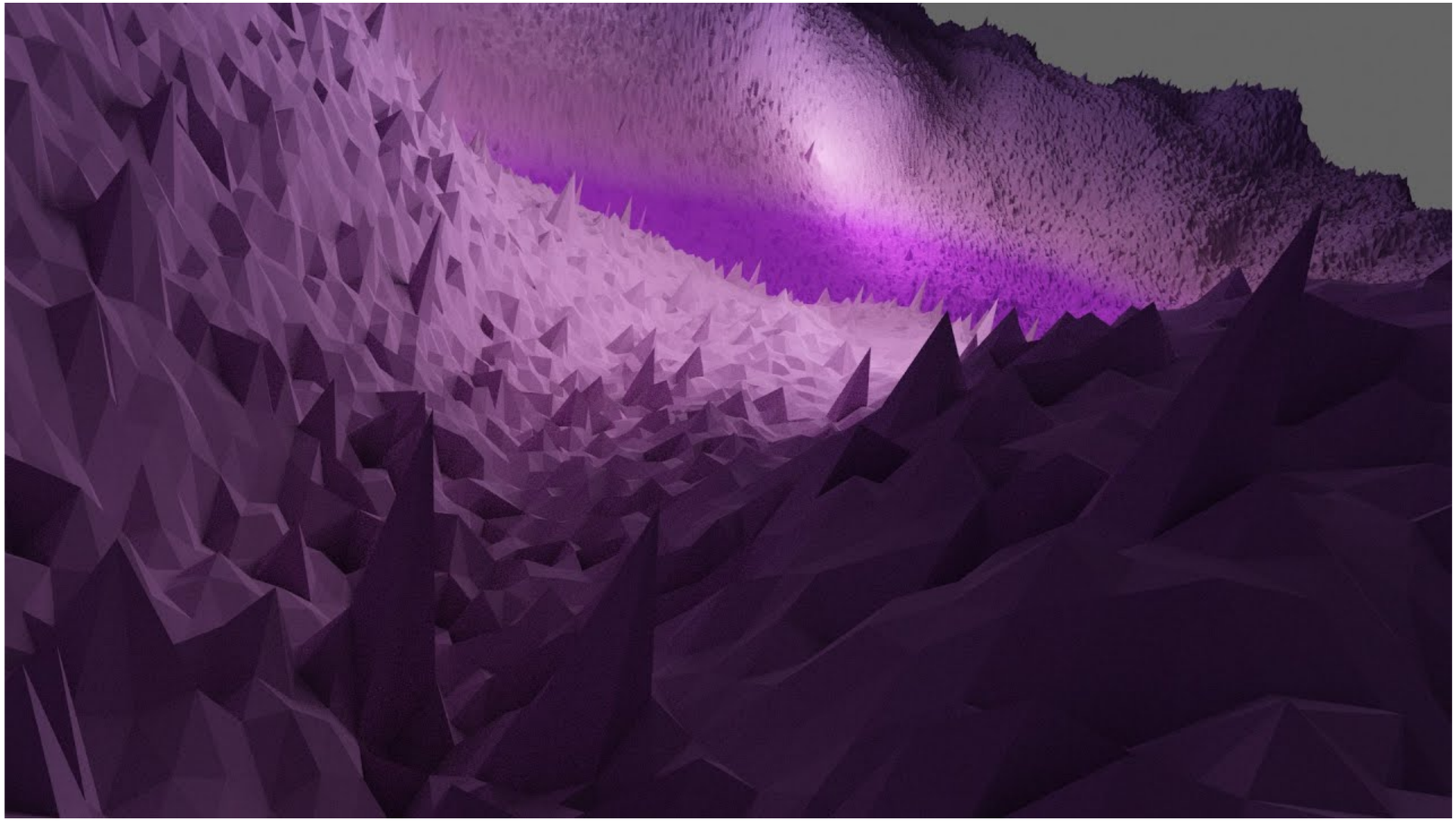
Landscape for Neural Networks



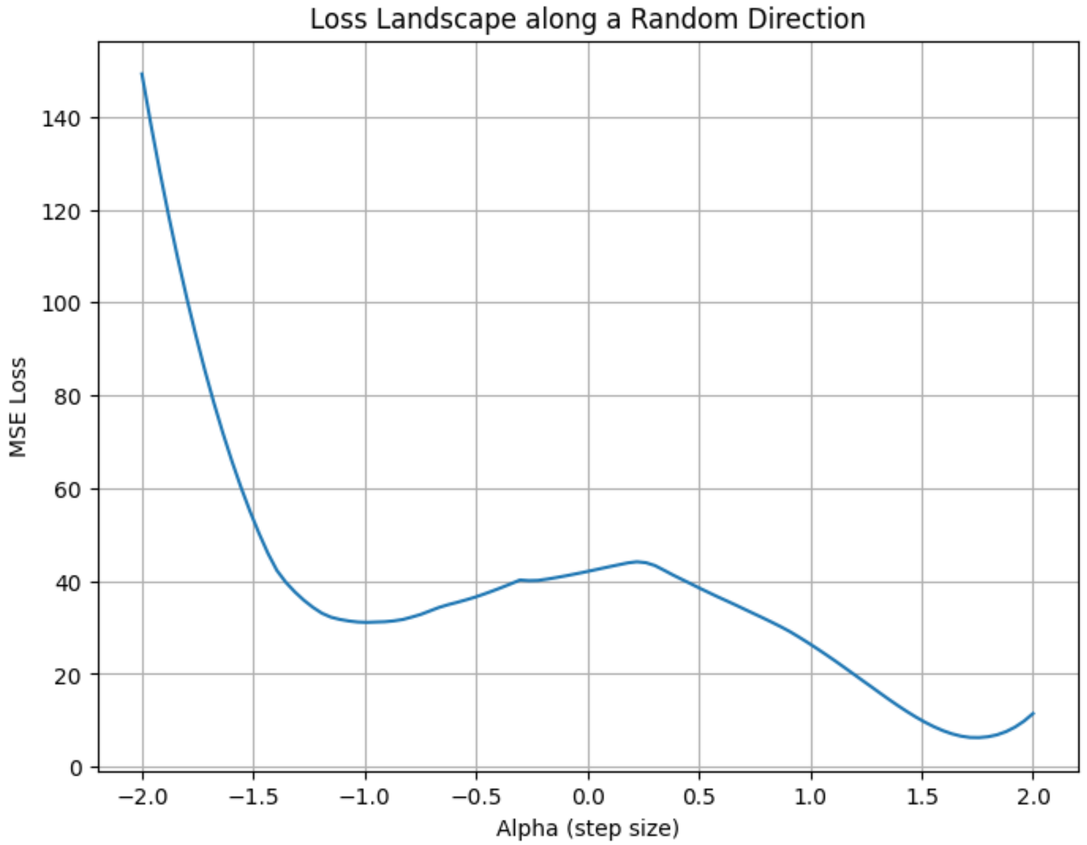


Plot of the loss landscape along a line

Landscape for Neural Networks



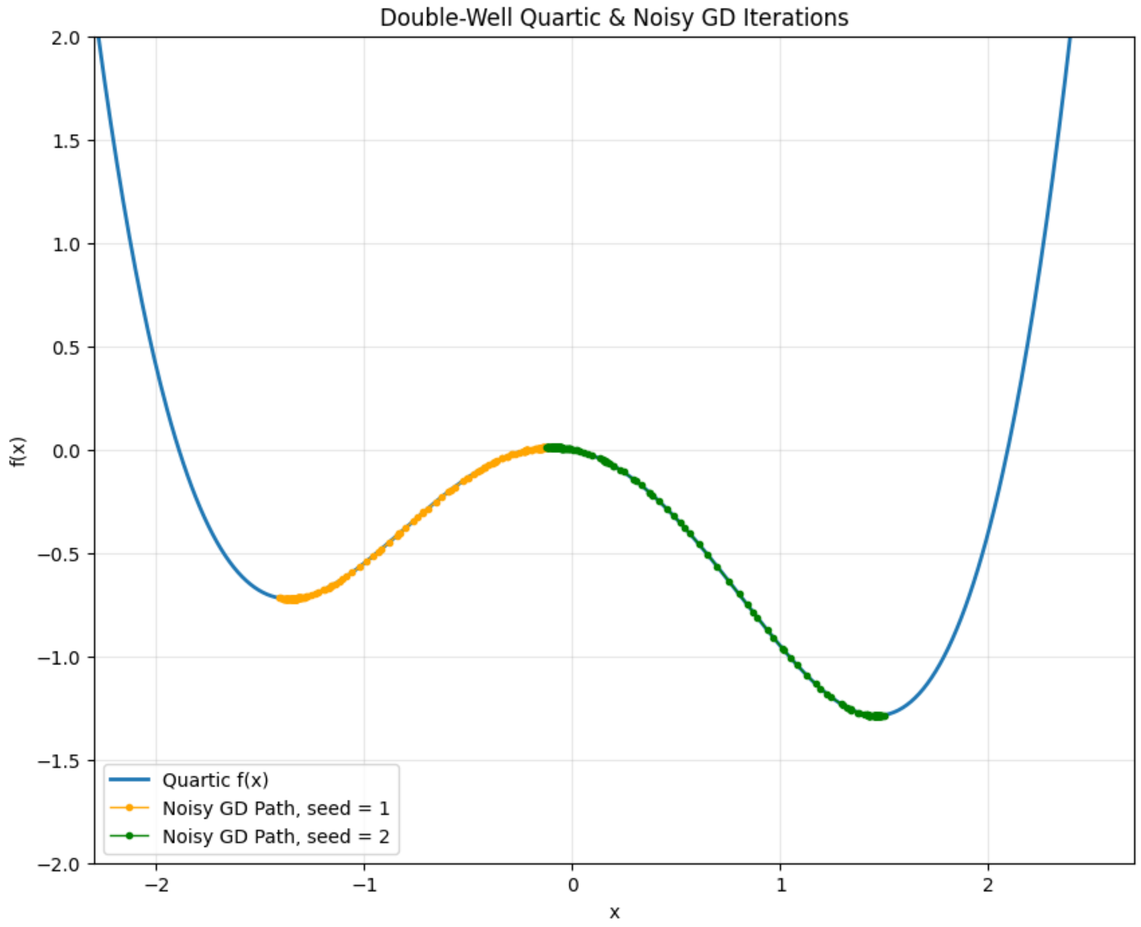
What do we expect from a non convex function



$$f_{a,b,c}(x) = \sum_{i=1}^{20} a_i \cdot ReLU(b_i x + c_i)$$

$(a_i b_i x - a_i c_i - y)^2$ is a quartic

What do we expect from a non convex function

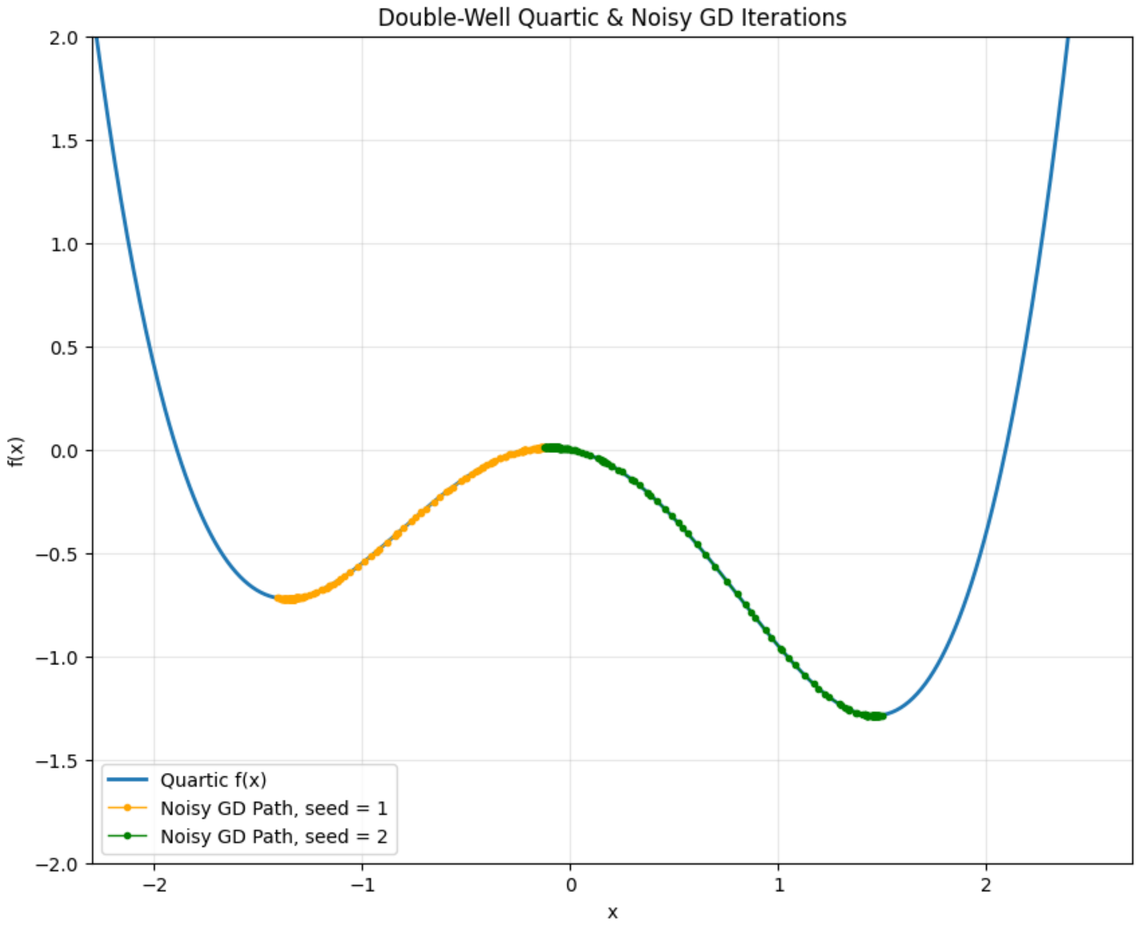


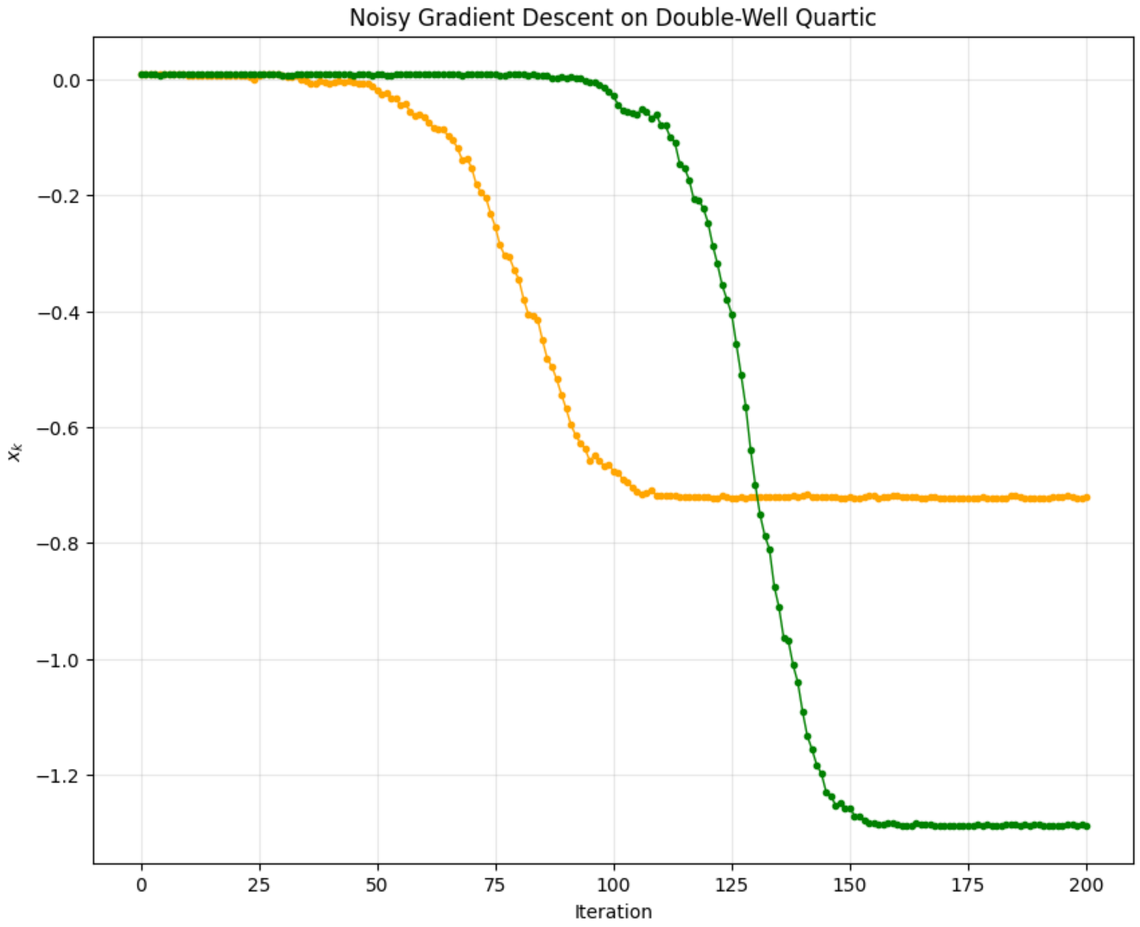
$$f_{a,b,c}(x) = \sum_{i=1}^{20} a_i \cdot ReLU(b_i x + c_i)$$

$(a_i b_i x - a_i c_i - y)^2$ is a quartic

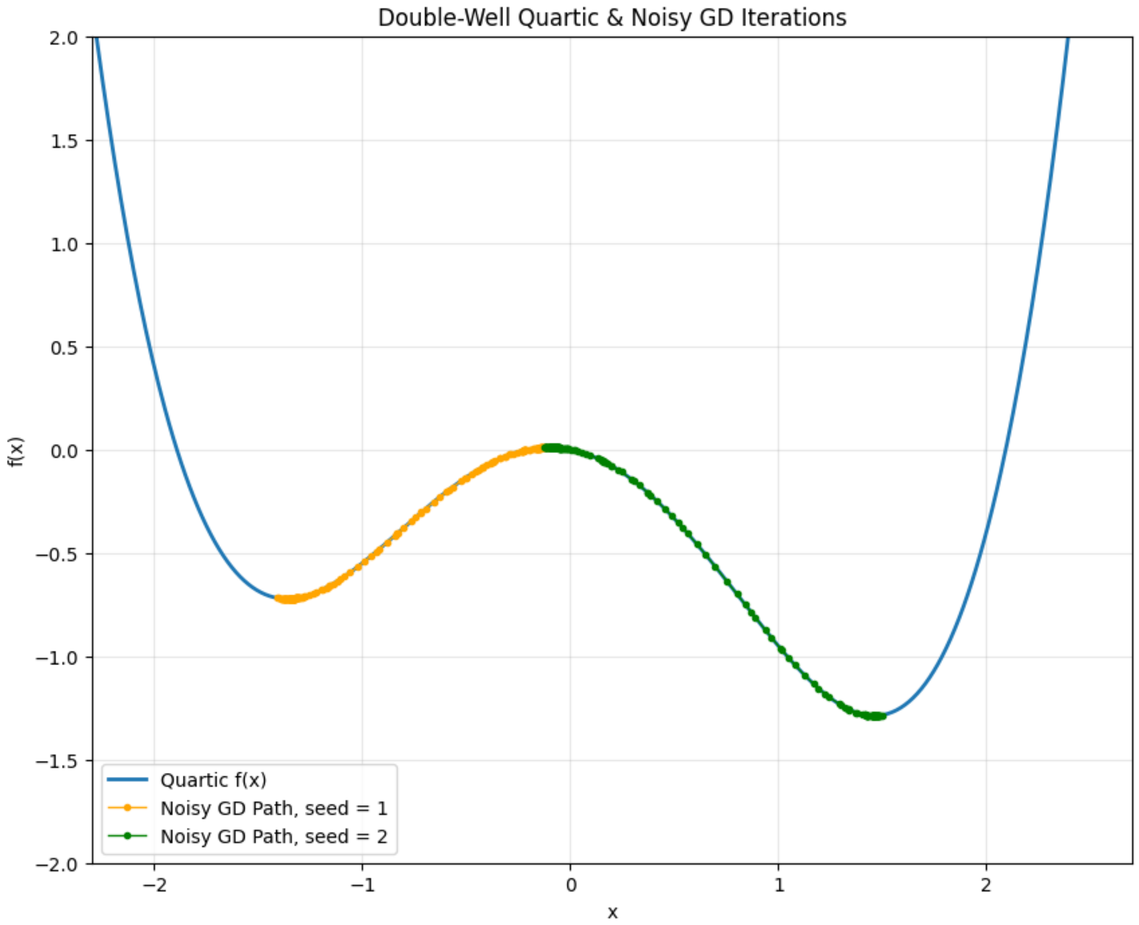
$$f(x) = \frac{1}{4}x^4 - x^2 - \frac{1}{5}x$$

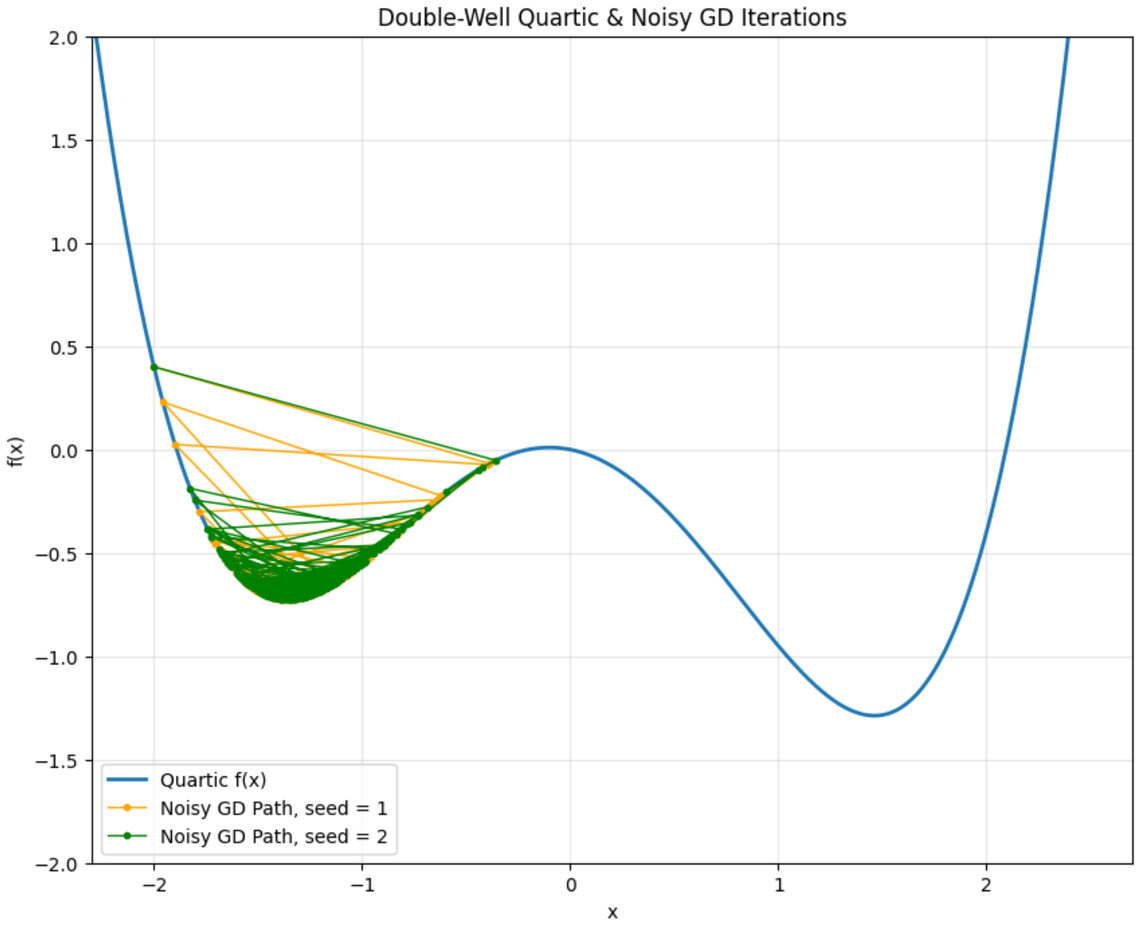
What do we expect from a non convex function





What do we expect from a non convex function



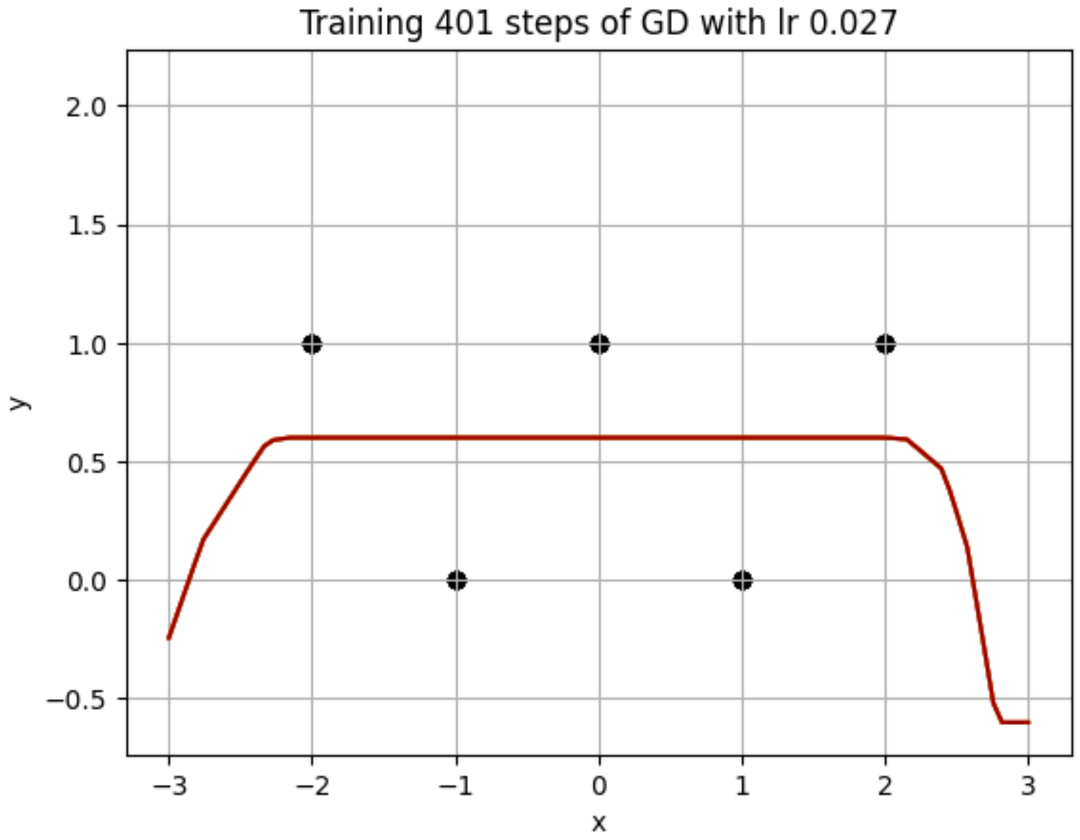


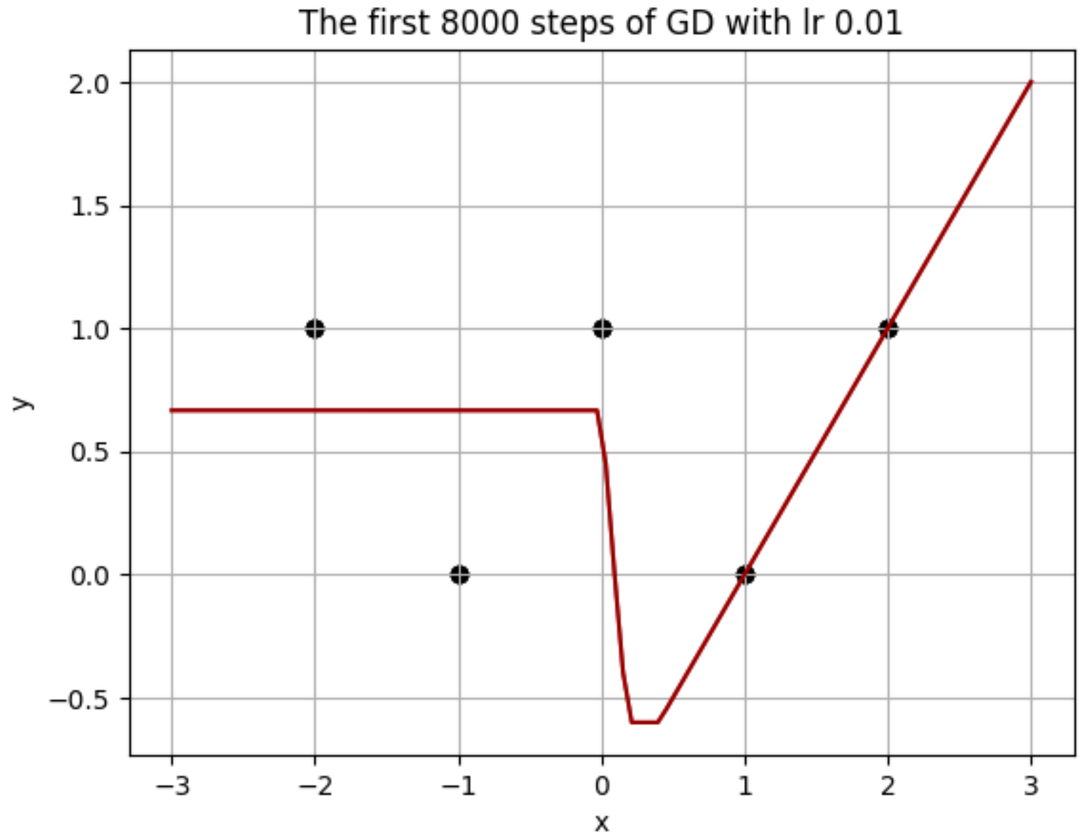
The landscape is non-convex:

Optimization is not meant to work

Landscape for Neural Networks

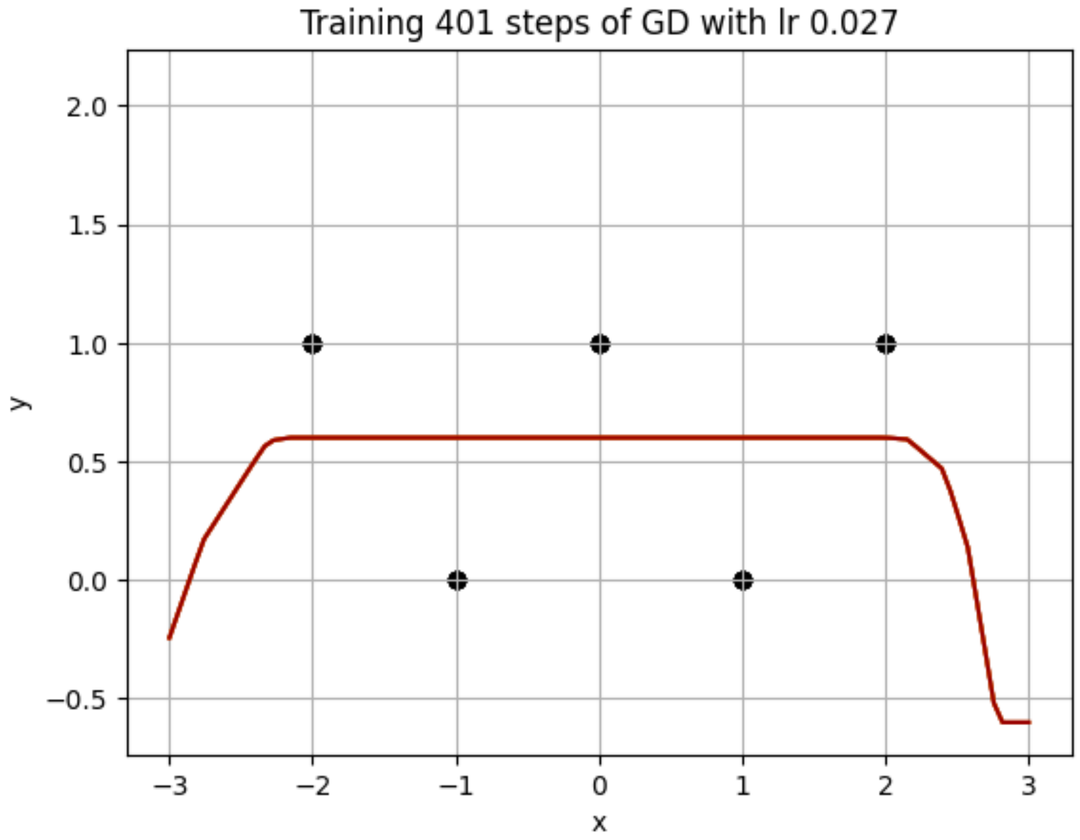
Does GD work?

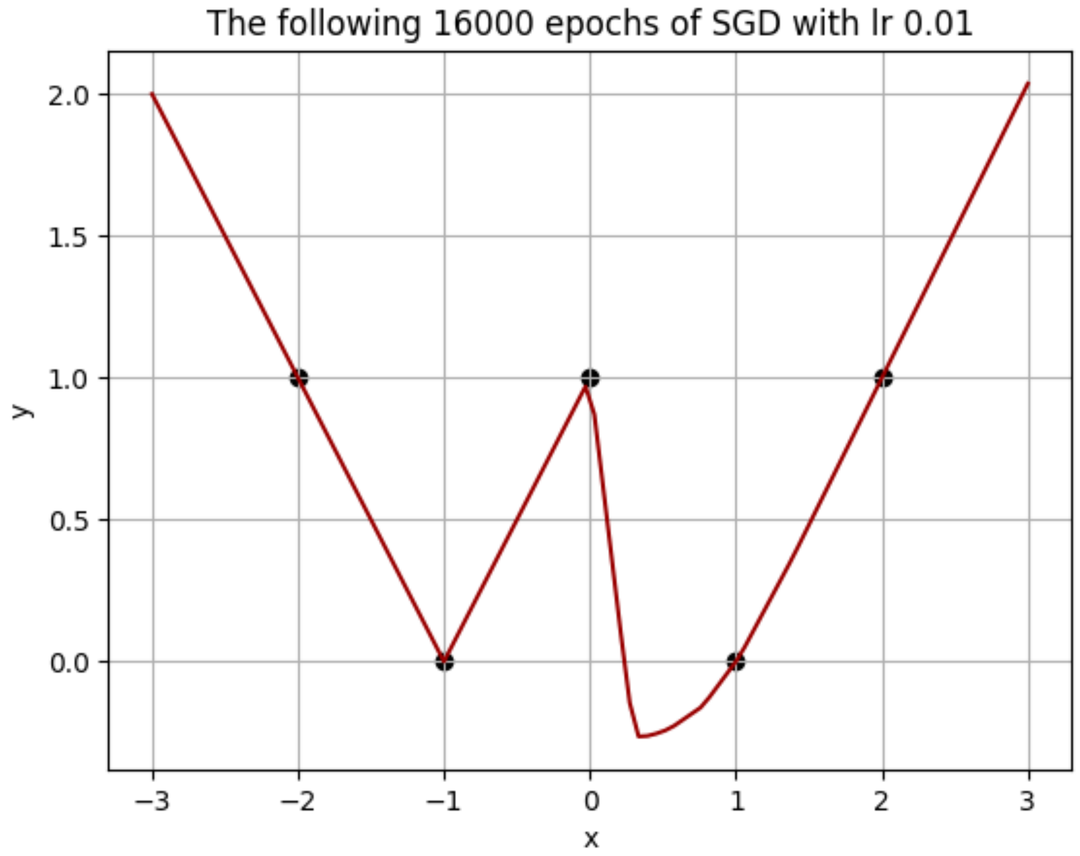




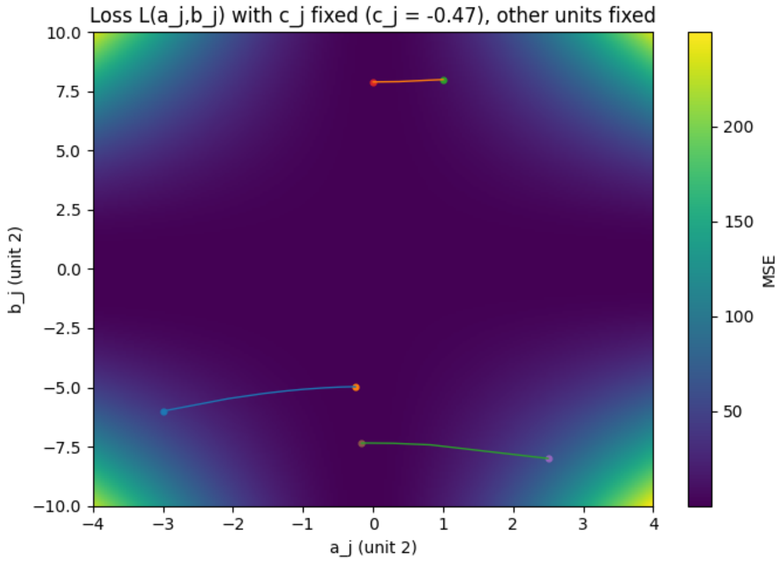
Landscape for Neural Networks

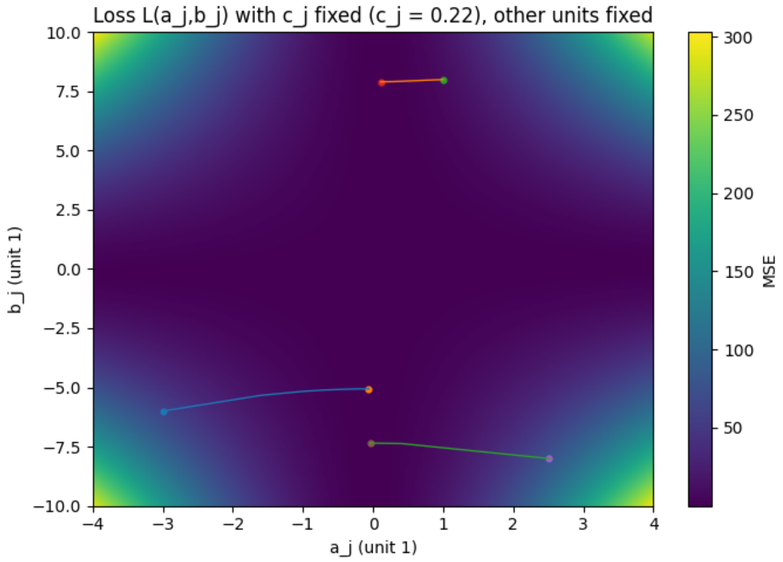
SGD or Adam do!

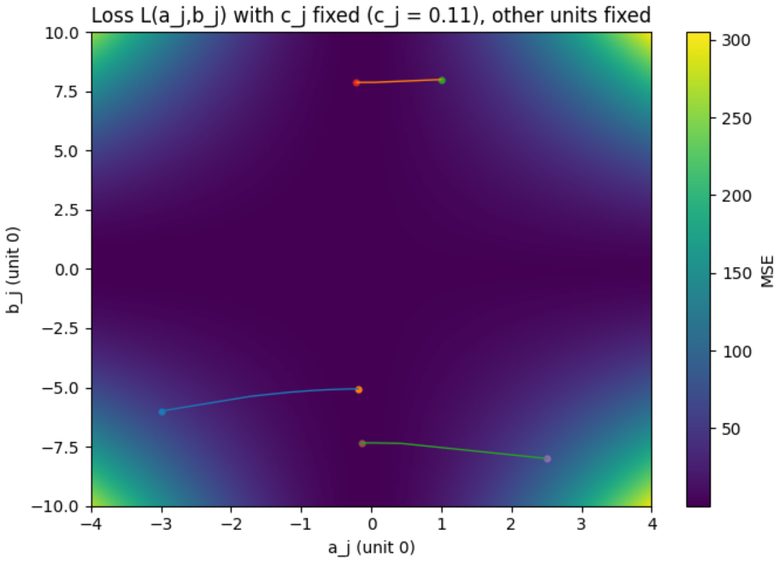




Landscape for Neural Networks

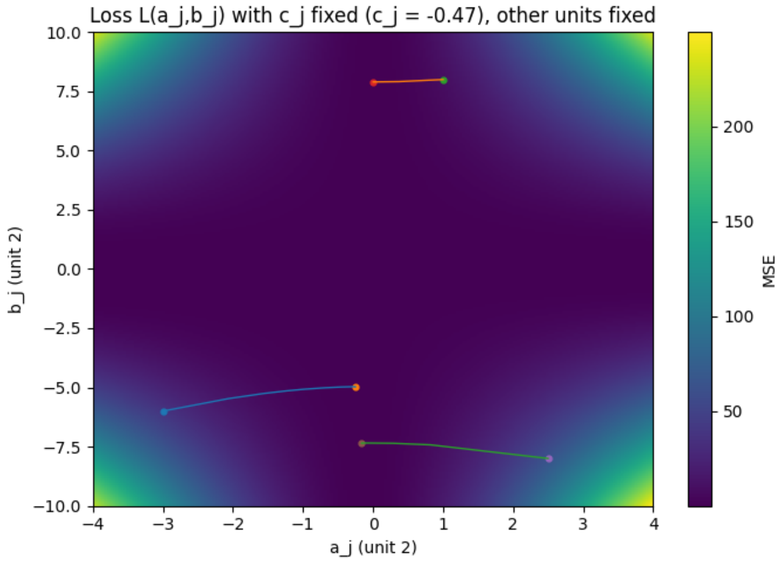


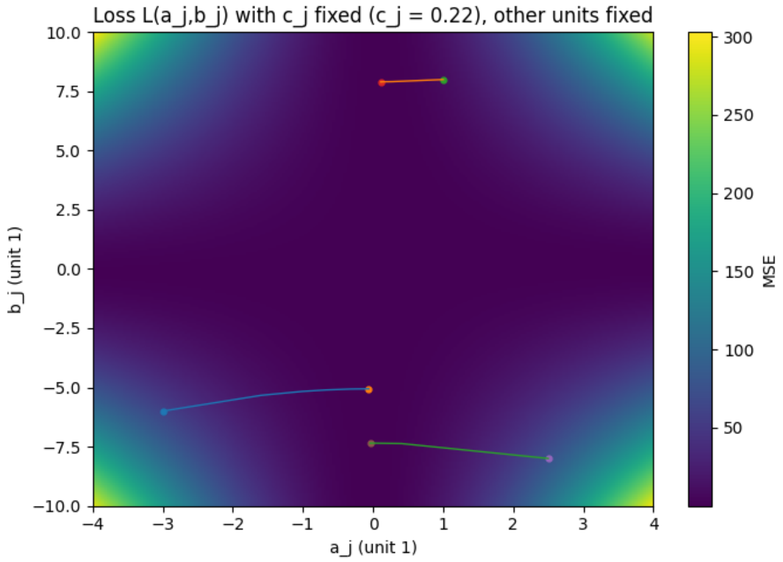


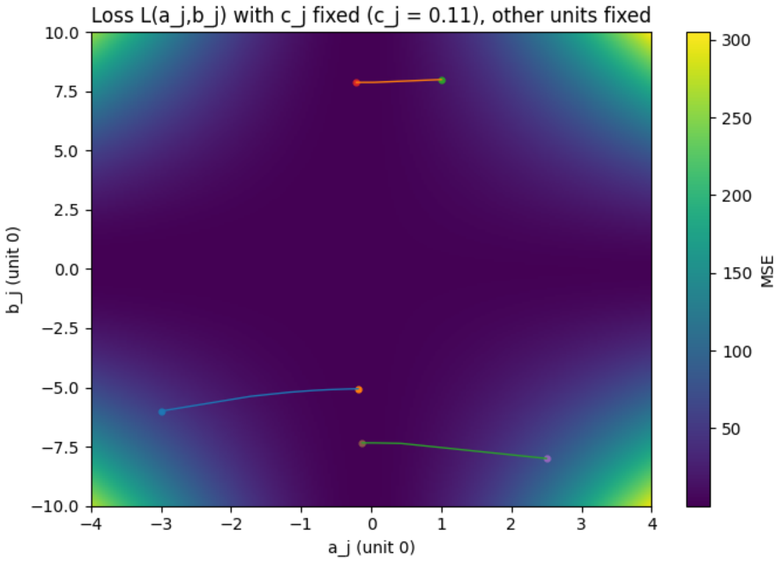


But the neurons move a little bit in a certain direction and then you learned!

Landscape for Neural Networks

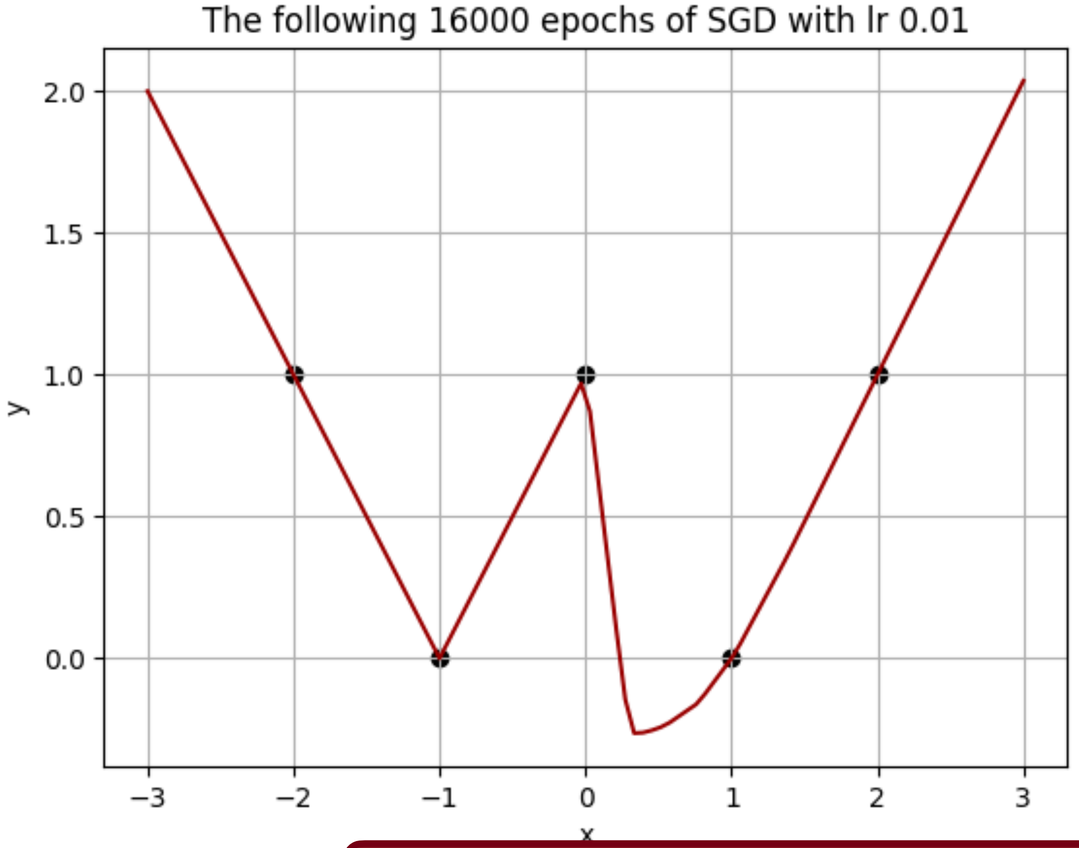


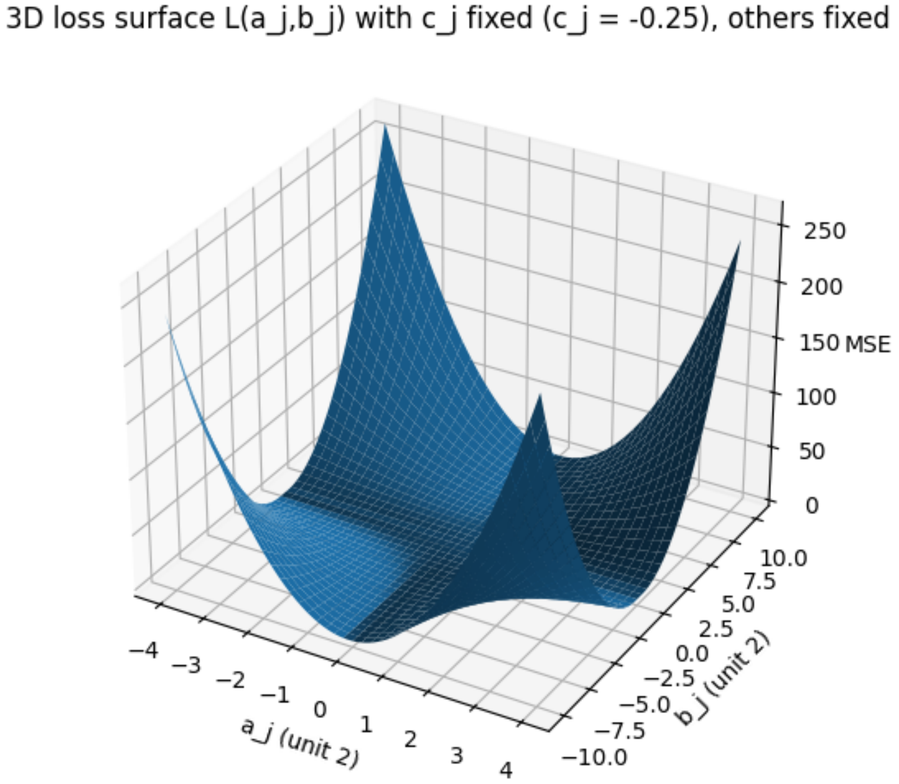




It should not be, but it is still easy!

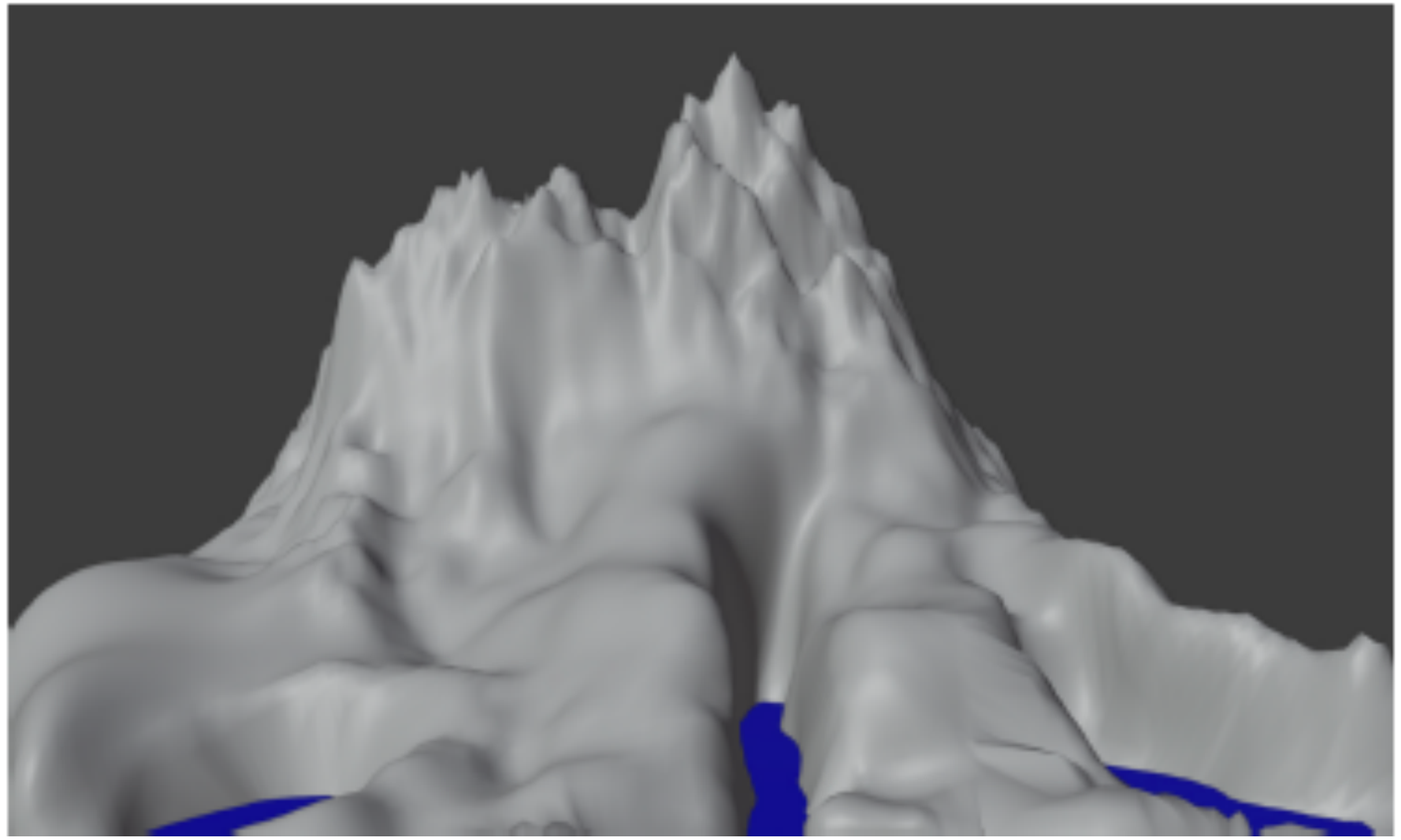
Landscape for Neural Networks

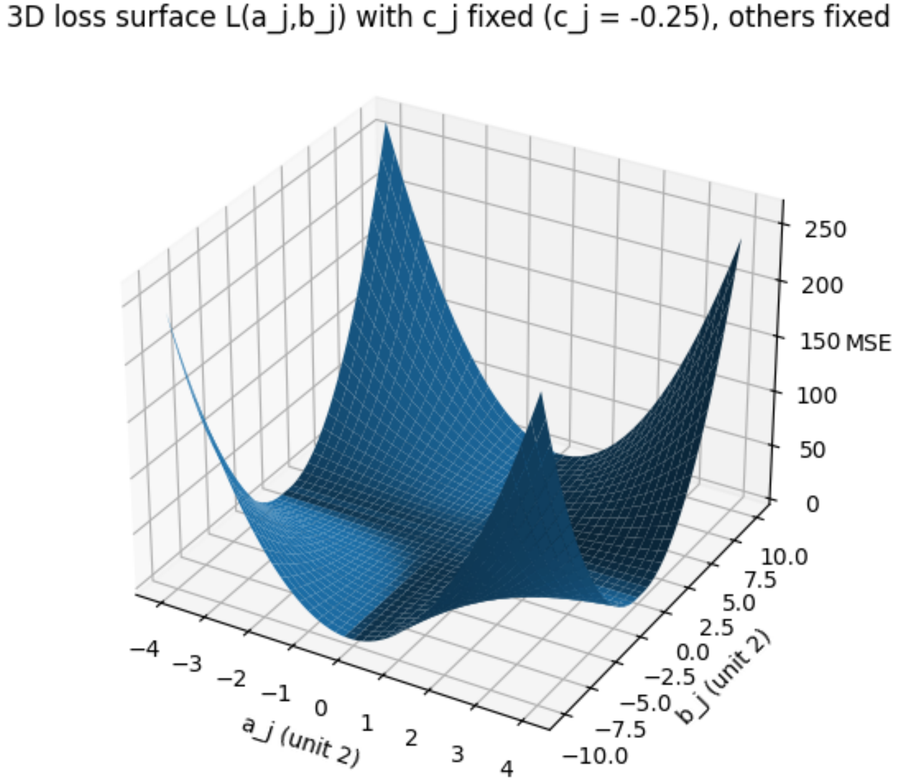




It should not be, but it is still easy!

Landscape for Neural Networks



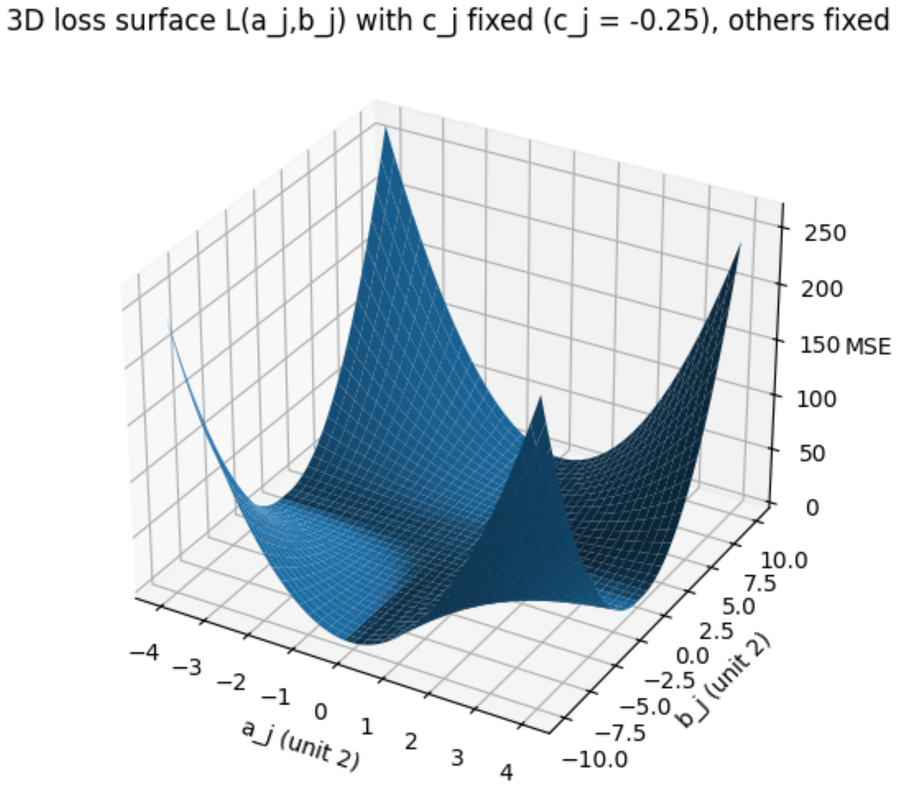


It is non convex but it’s easy to go down!

Questions

It is non convex but if you go down, you are good!

- Is this actually the case?
- Why?
- Can we say something theoretical about this?
- Can we propose a better algorithm?



Work on the Topic

A number of works tried to make sense of this:

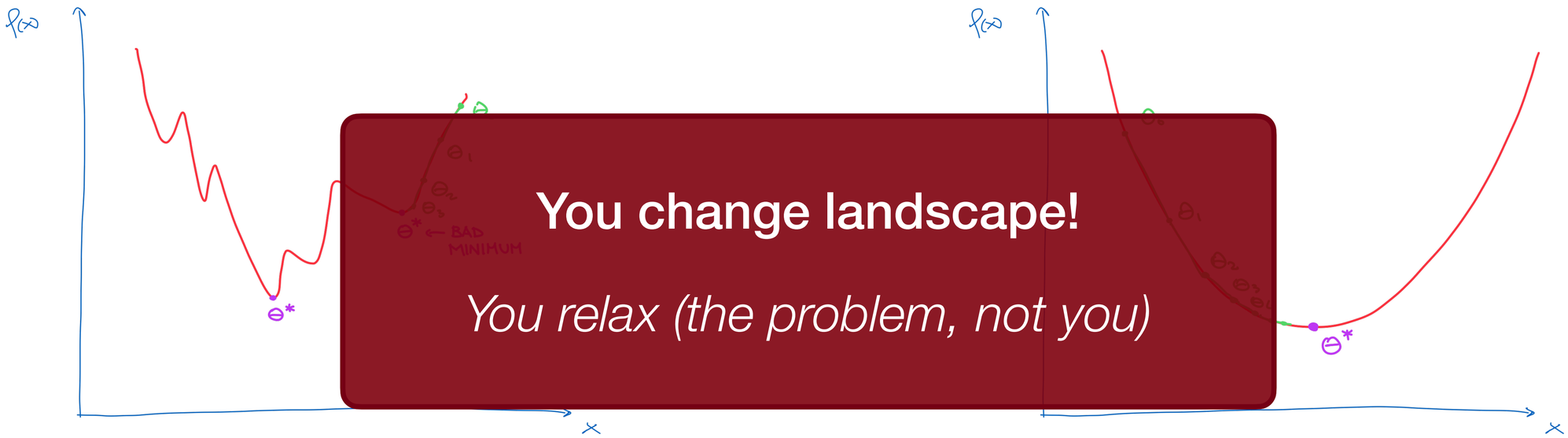
- **All the sublevel sets are connected (overparam)**
  (Nguyen, 2019)

- **Overparameterized networks are essentially PL**
  (Allen Zhu et al., 2019 ++)

- **You may escape saddle points** *(Lee et al., 2016)*

But the problem is open

How do you deal with a bad landscape?



You change landscape!

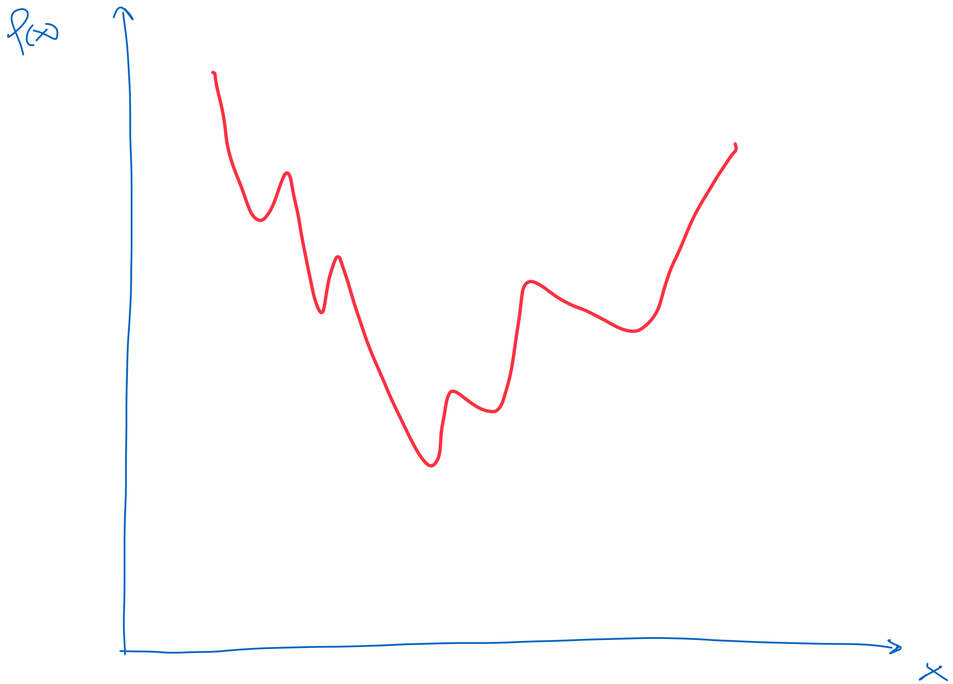
You relax (the problem, not you)

Non-Convex

Convex

A Breakthrough from the Community

- When you have a non-convex problem in applied math: ***you convexify it.***
- The neural network community of the 80s, decided to try making the network bigger, train it with GD, then prune it.
- This was the first such time in which a field *went against convexity.*
- We still do not understand why relaxation by capacity, instead of by convexity works!



No Convexity: Recap

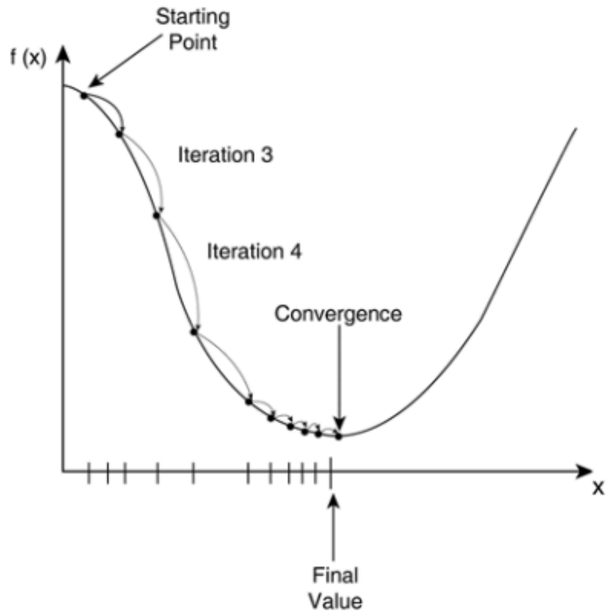
- The problem is non-convex $\Longrightarrow$ should not work.
- Optimization still works, but we do not understand why and how.
- Neural networks were the first case in which this did not stop researchers who *relaxed by capacity.* What does that mean!

Outline: puzzles about approximation, optimization, generalization

- No Curse dimensionality (puzzle for approx/optimization)
- VC bounds overparametrization (puzzle for generalization)
- Optimization should not work this way:
  - Non-Convexity
  - Instability
  - About Location

Gradient descent

From Class 4



Let $F : \mathbb{R}^d \to \mathbb{R}$ differentiable, convex and such that

$$\|\nabla F(w) - \nabla F(w')\| \leq B\|w - w'\|$$

(e.g. $\sup_w \| \underbrace{\nabla^2 F(w)}_{\text{Hessian}} \| \leq B$)

Then

$$w_0 = 0, \quad w_{t+1} = w_t - \frac{1}{B}\nabla F(w_t),$$

converges to a minimizer w^* of F.

θ is now w

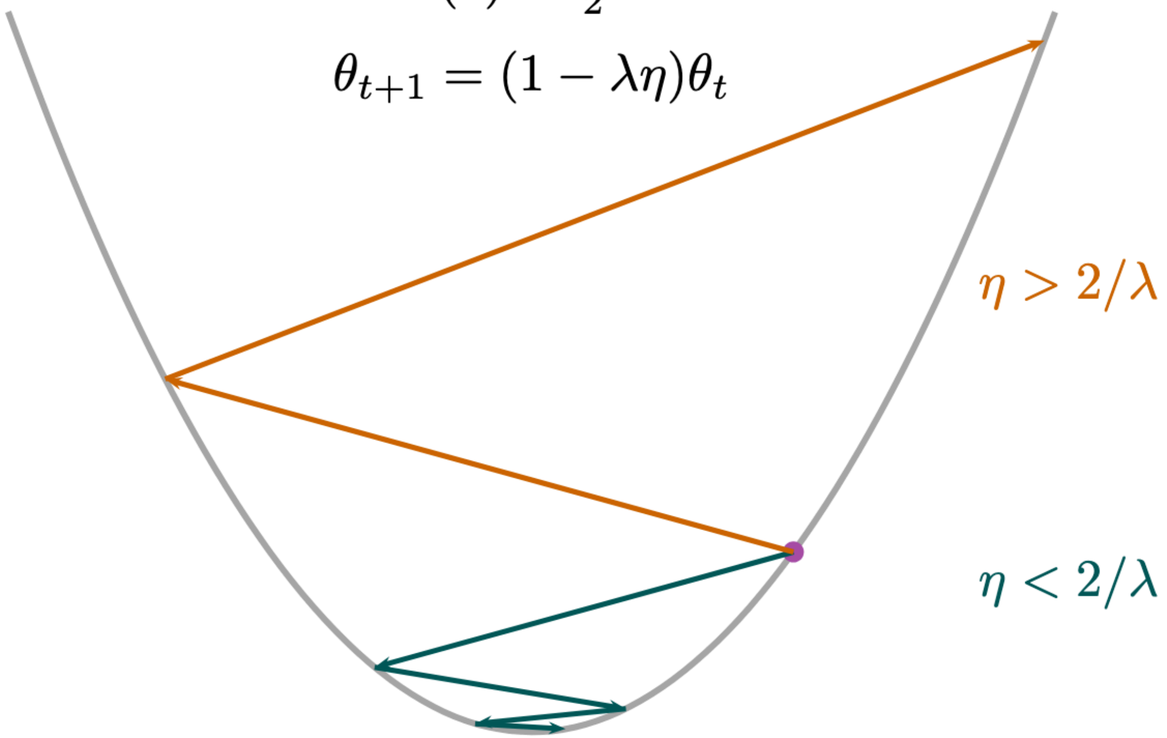
B will be $\lambda_{\max}$

$\eta = 1/B$ here!

Stability for Quadratics

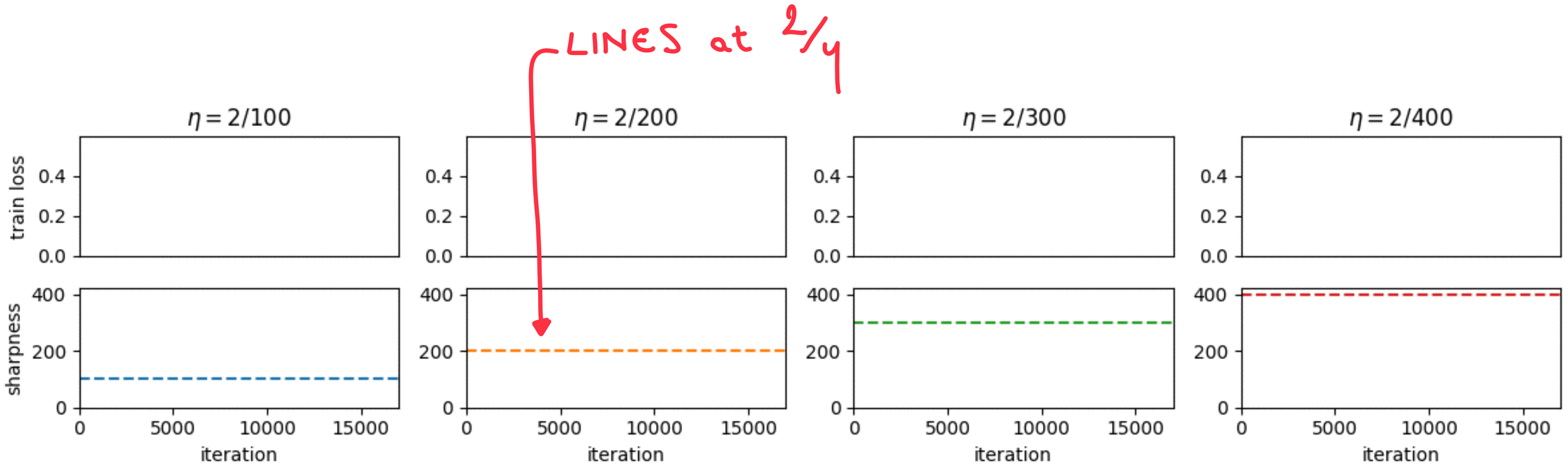
$$L(\theta) = \frac{\lambda}{2}\theta^2$$

$$\theta_{t+1} = (1 - \lambda\eta)\theta_t$$



The Edge of Stability

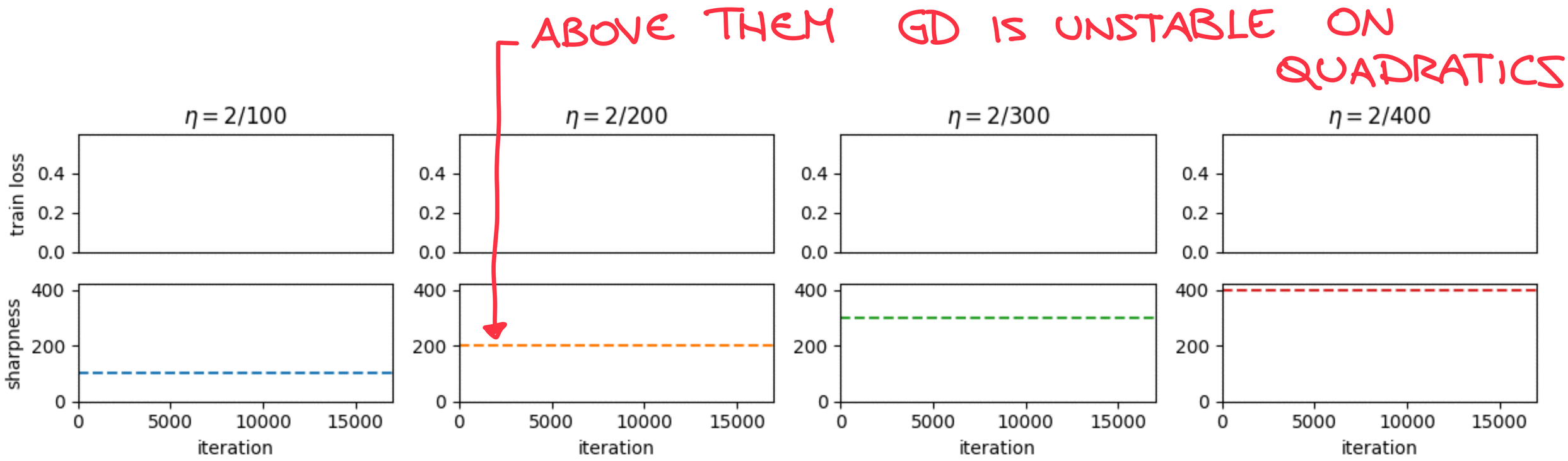
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv:

The Edge of Stability

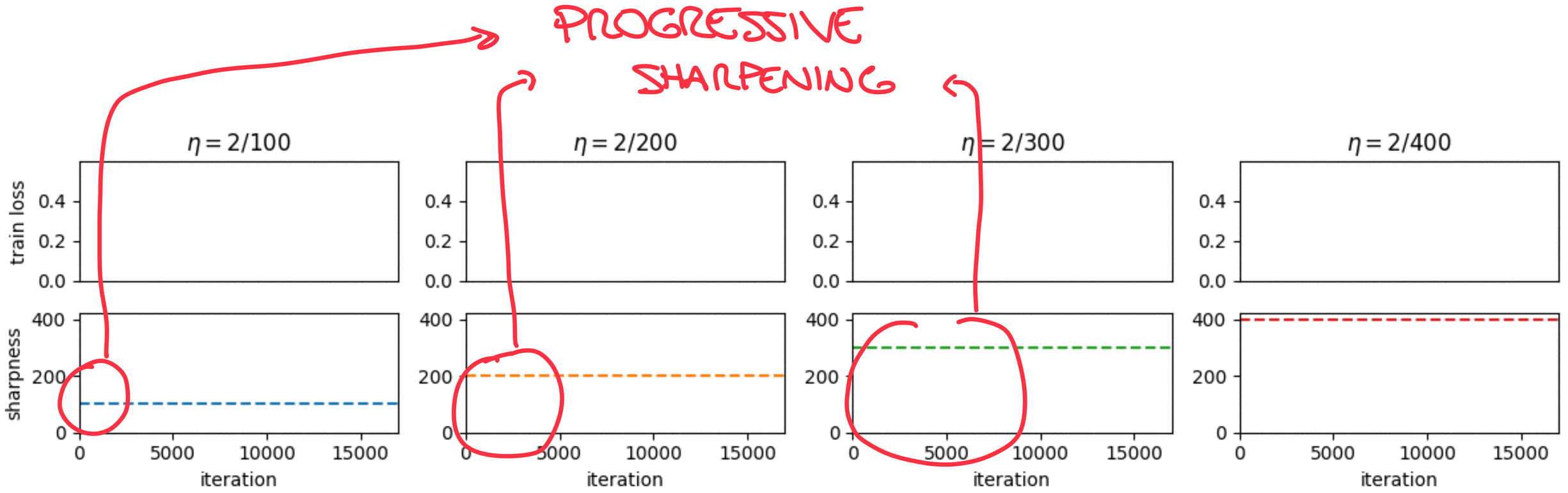
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv:

The Edge of Stability

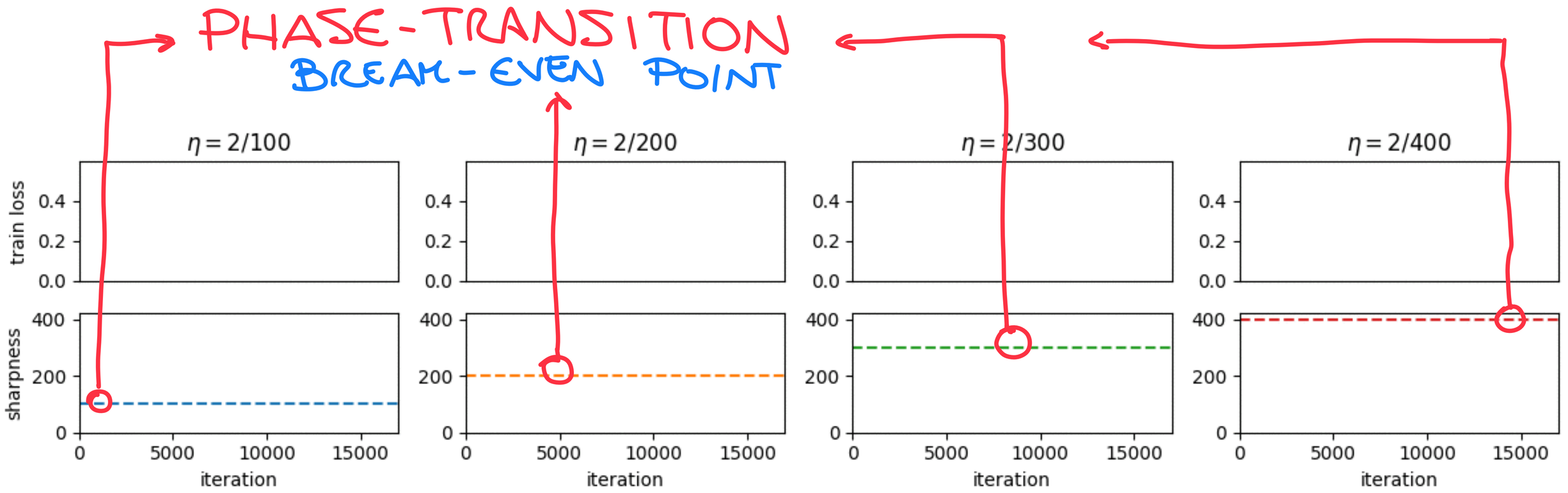
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv:

The Edge of Stability

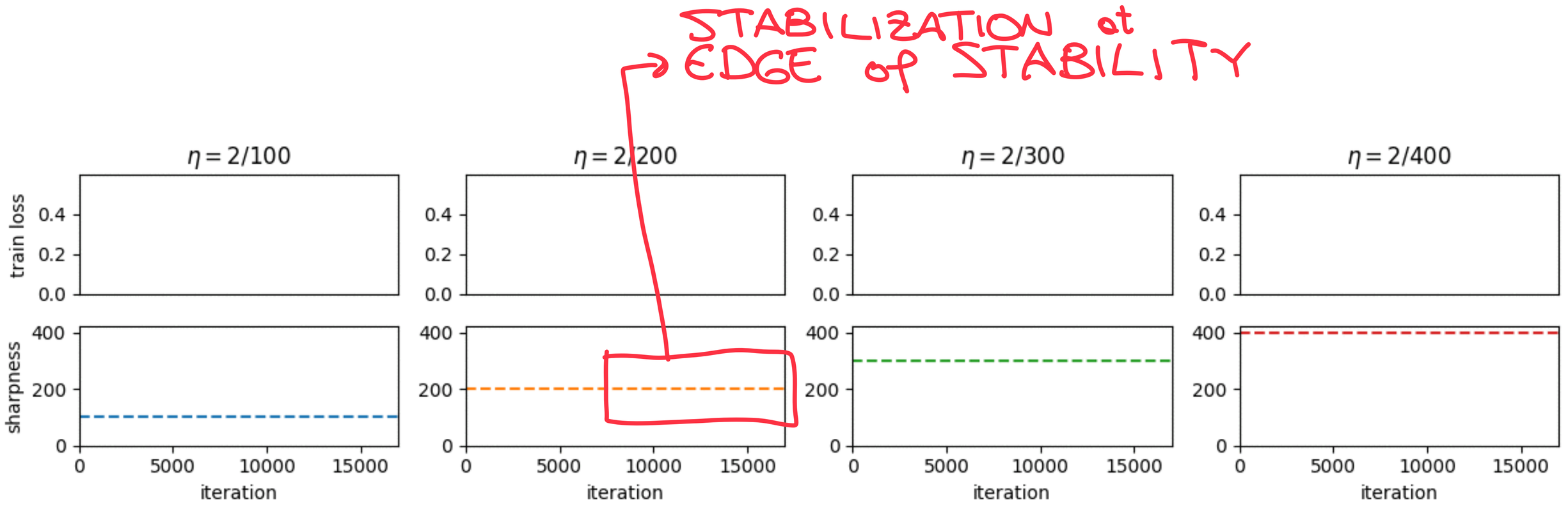
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv:

The Edge of Stability

Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



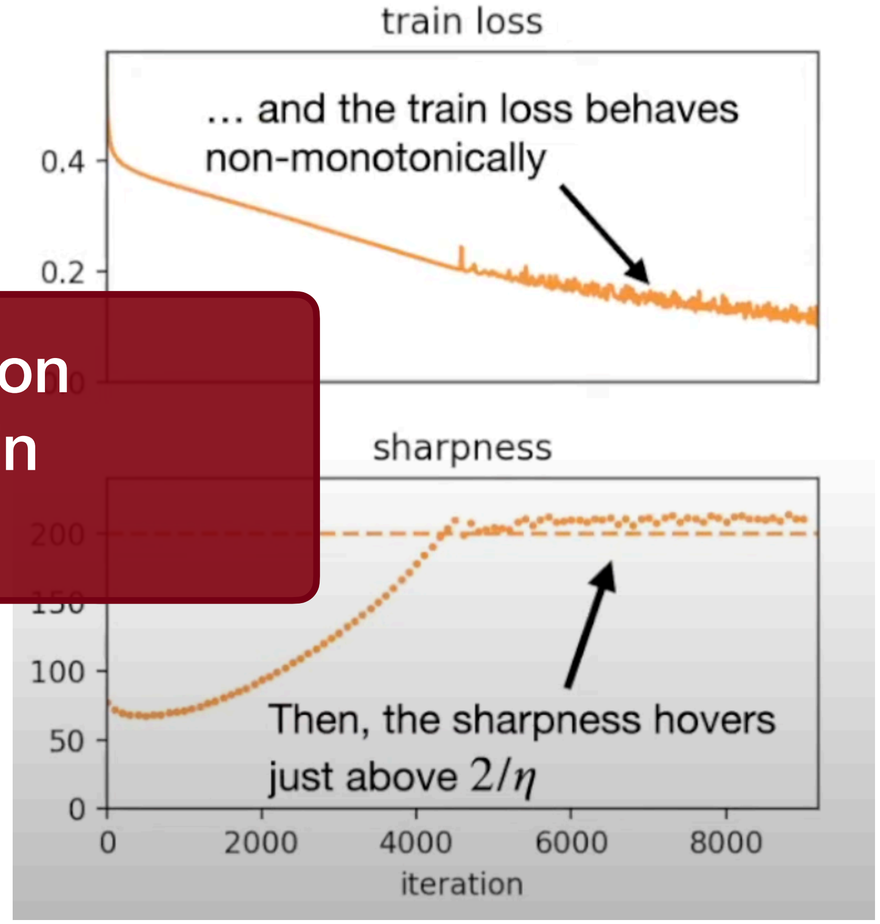
Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv:

The Edge of Stability

Implications:

- Conventional wisdom about step size is wrong — *set $\eta \leq 1/B$ is an impossible task!*
- You have to go deeper than the order-2 Taylor approximation when doing NN theory — *it would diverge…*
- GD does not decrease monotonically the loss, *gradient size increases*. *Still,* GD is successful.

Classical optimization tools do not explain what happens



Plot from the talk of Jeremy Cohen — Cohen et al. (2021), ArXiv:

Questions

What is going on here?

- Is this phenomenon beneficial or not?
- Is this the case of practical optimizers?
- Can we still prove that we converge?

Instability: Recap

- Step size is too large $\Longrightarrow$ the step should be (is) counterproductive: you go up!
- Optimization still works, but we do not understand why and how.
- The whole machinery we used to prove that you go down is broken.
- Not just to prove, to give principles on how to optimize!

Outline: puzzles about approximation, optimization, generalization

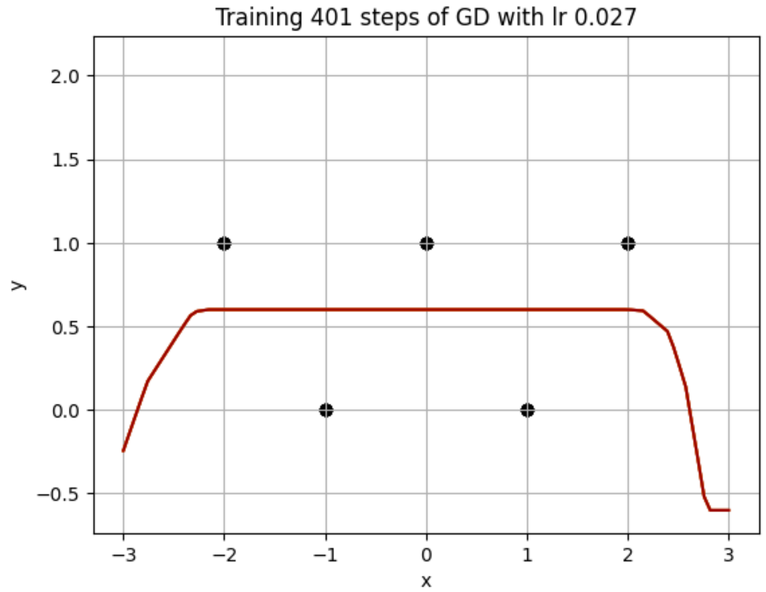
- No Curse dimensionality (puzzle for approx/optimization)
- VC bounds overparametrization (puzzle for generalization)
- Optimization should not work this way:
  - Non-Convexity
  - Instability
  - About Location

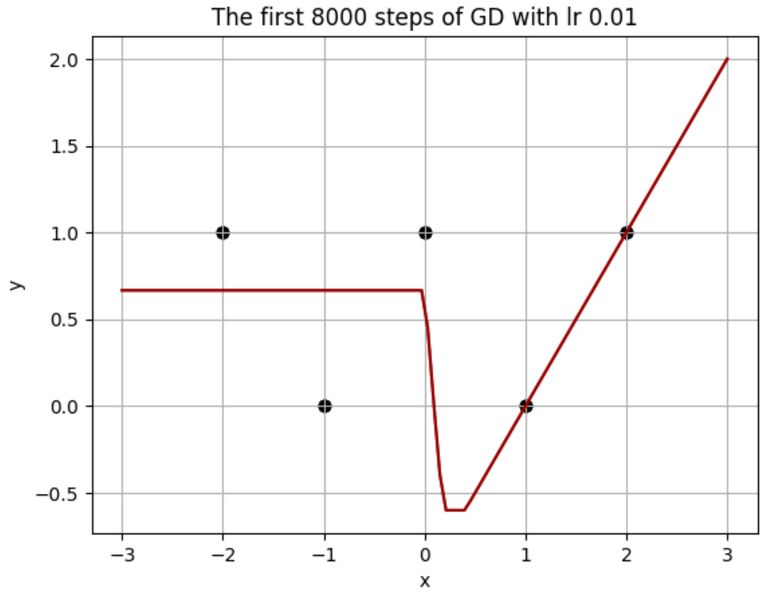
Goal of Training

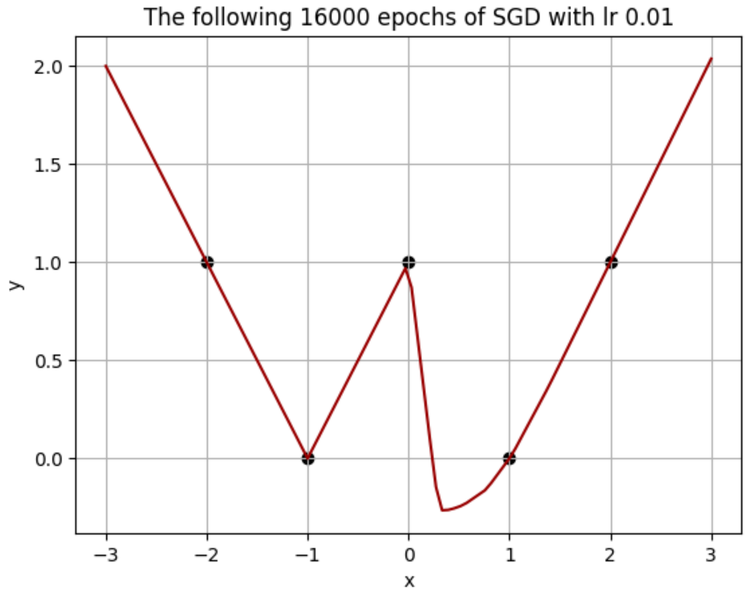
Think like a *practitioner*

Goal of many classical optimization problems:

- **Find a minimum**
- **With minimal compute budget**







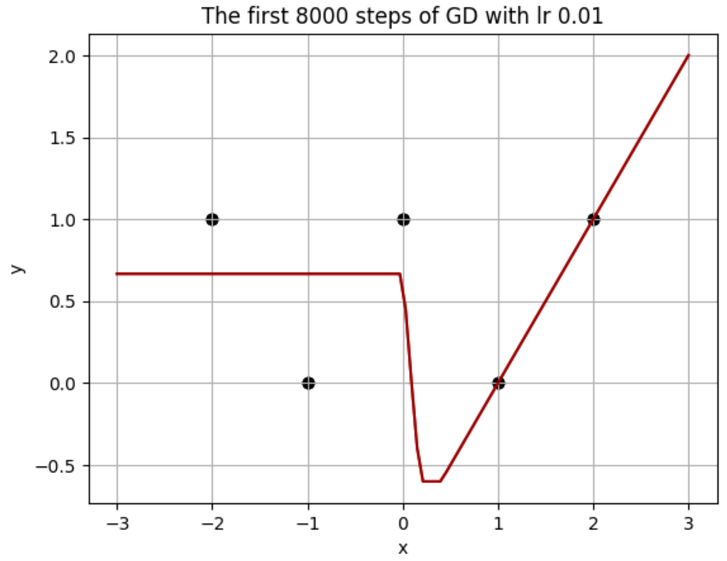
Paradigm Shift

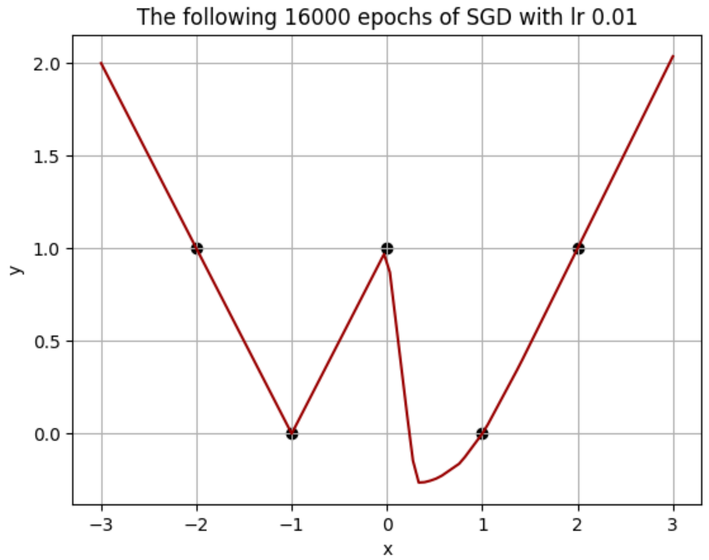
Goal of training neural networks:

Find the one which
Performs best on your task

What we can work with:

- The data we got
- The measure of error $L(\theta)$
- (Within) a compute budget

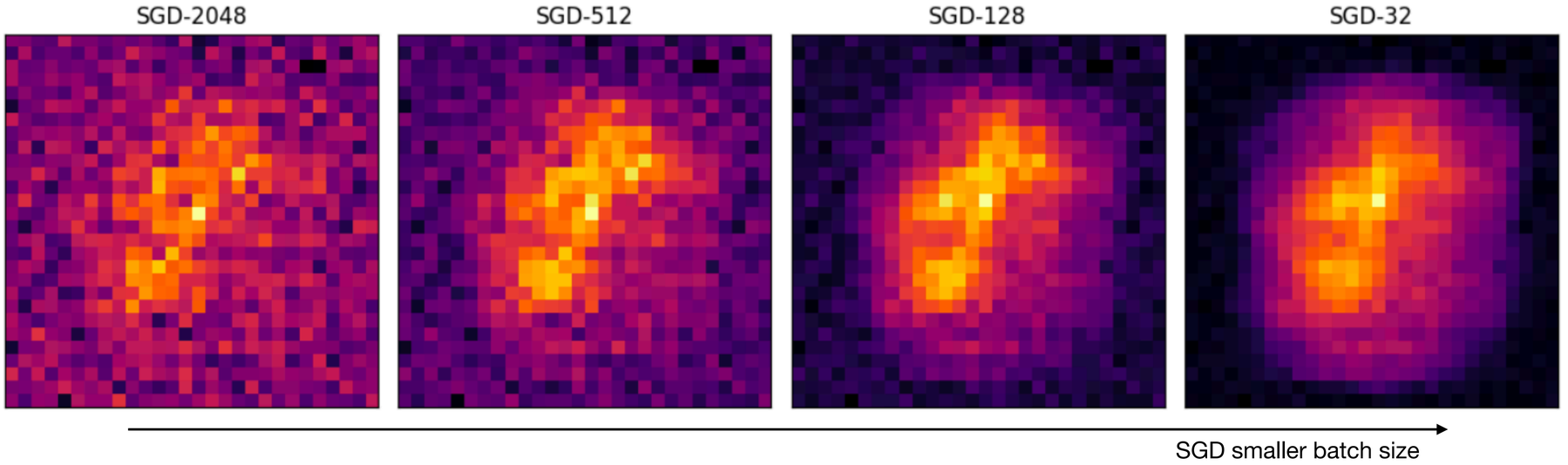




HYPER PARAMETERS

- Size of the first layer weights of an MLP learning MNIST with different batch sizes

VANILLA NEURAL NETWORK



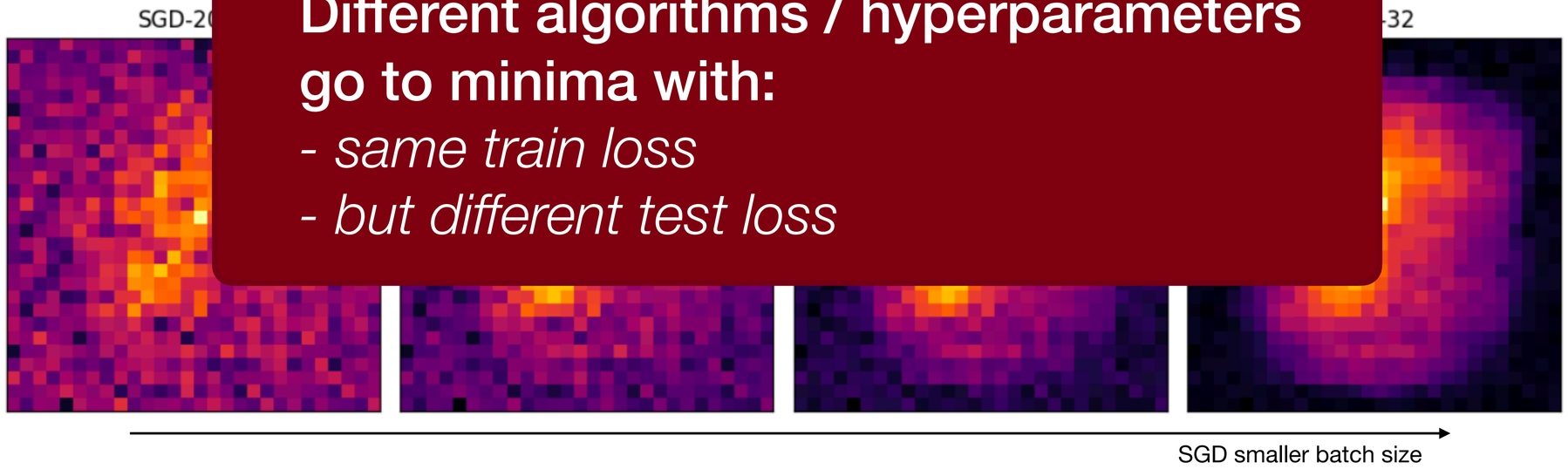
Beneventano*, *Pinto**, *Poggio, (2024),* .

HYPER PARAMETERS

- Size of the first layer weights of an MLP learning MNIST with different batch sizes

VANILLA NEURAL NETWORK

Different algorithms / hyperparameters go to minima with:
- same train loss
- but different test loss



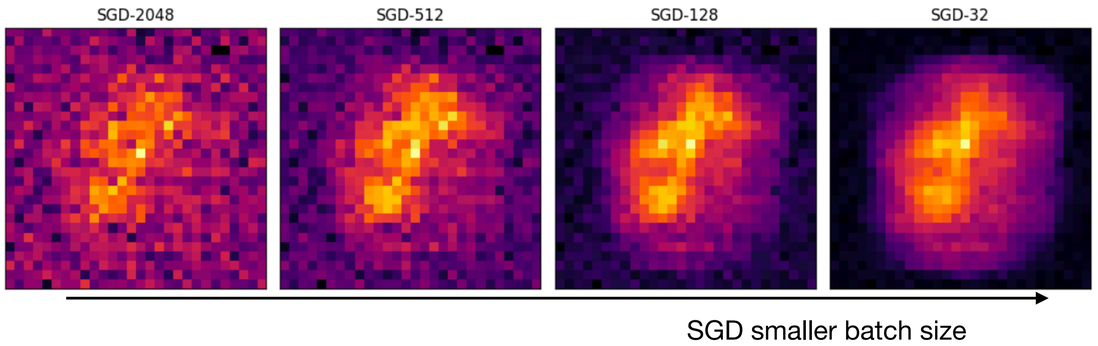
Beneventano*, Pinto*, Poggio, (2024),* .

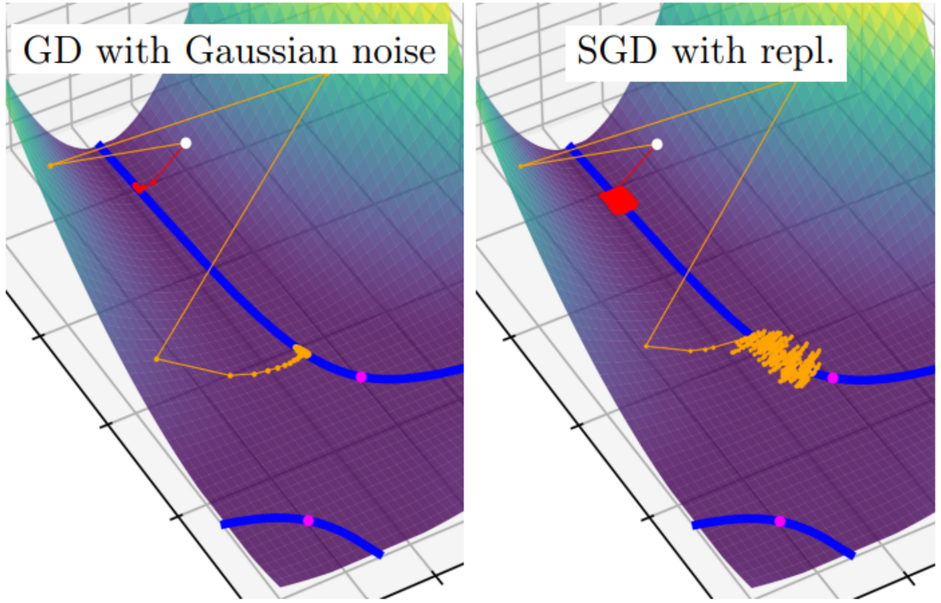
RECAP

Goal is having best "performance"

There are many different minima *with different performance*

Different algorithms find different minima, *consistently*





Location

- It’s not obvious that you just want to go to a global minimum.
- You want to go to a good minimum for your *real* measure of performance.
- It may be unrelated to train loss!

You care only about *real world* performance

Location

You care only about *real world* performance

- Maybe it is fine even if you do not converge on the train loss.
- You care about what model you find, not the fact that it is a minimum.

You may want to study *location* of convergence, *for instance*

Questions

- Why/When is training successful?
- What does successful mean?
- Can we prove you converge to a *good* model?
- What neural networks do we see?

Optimization?

- Do we actually still care about training loss going to zero?
- Maybe we care about what training algorithm does what?
- Maybe the fact that optimization is rigged does not matter because we did not care about optimizing that super precisely?

**We have to restart from scratch,
thinking what theory we want, if any.**

Recap

- We lost the properties that allowed us to say:
  If you go down you converge.
- We lost the properties that allowed us to say:
  After one step you go down.
- Maybe we have to step back and rethink deeply about what we care about?