Training Neural Networks

Where training goes and its stability

Lecture 18

Pierfrancesco Beneventano
pierb@mit.edu

Massachusetts Institute of Technology

Part 0

Recap from last classes and motivations

The Goal Was

- We have data coming in real life: $X \sim P$
- We need to find the neural network $f_\theta(X)$ that has the right output $\varphi(X)$.
- A measure of the Error $distance(\phi(\,.\,), f_\theta(\,.\,))$.

They exist.

Can we find them?

We minimize the measure of error

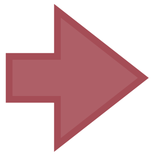
Optimization

Learnability *vs* Optimization

We have:

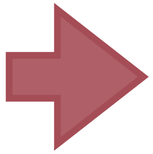
- A dataset.
- Some compute

  (Memory + number of computations constraint).



Do we have enough samples?

Learnability



In case, does the training routine find it within the compute?

Optimization

Compute / Memory

- The dimensionality N of the problem $\left(\theta \in \mathbb{R}^N\right)$ is very high, e
- We have many n loss is an

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

We work with:
(for both learnability and compute)

Mini-batch Gradient Methods

Gradient-based

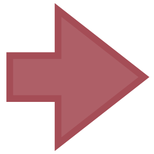
$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k)$$

Mini-batch methods

Gradient Based Methods

Designing:

Gradient-Based Optimizers



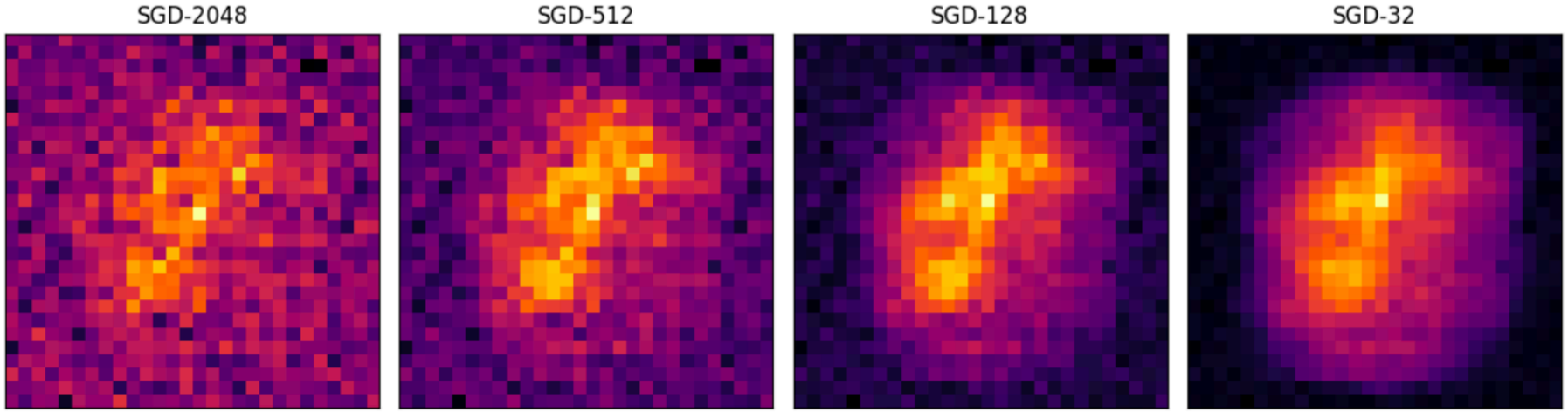
How to go:

from *infinitesimal* **to** *local*

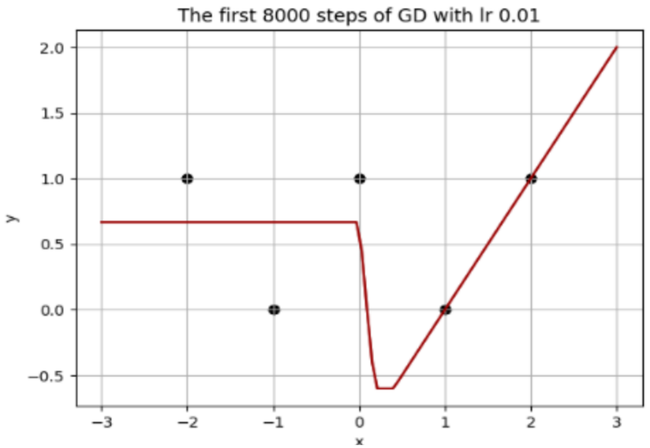
- **Step size = Norm + Trust.**
- **About second derivative.**

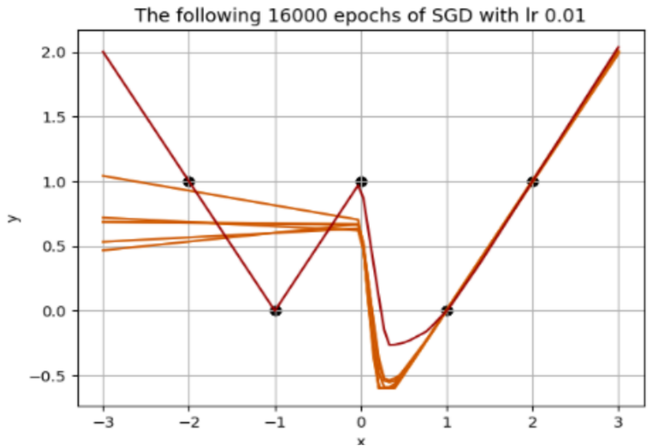
Using history

What is this Picture Missing?



SGD smaller batch size





Location of Convergence

It is not just about if you go down within compute:

- It's also about where you go within compute!
- We discussed how to do your best in the step:

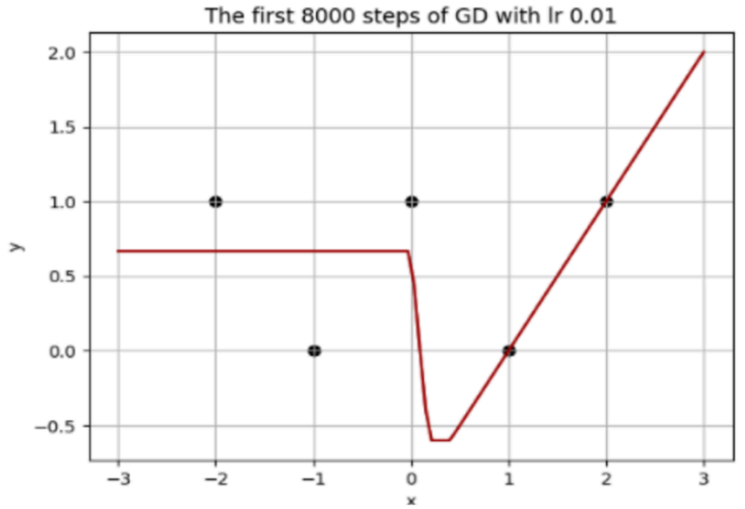
Infinitesimal to *local*

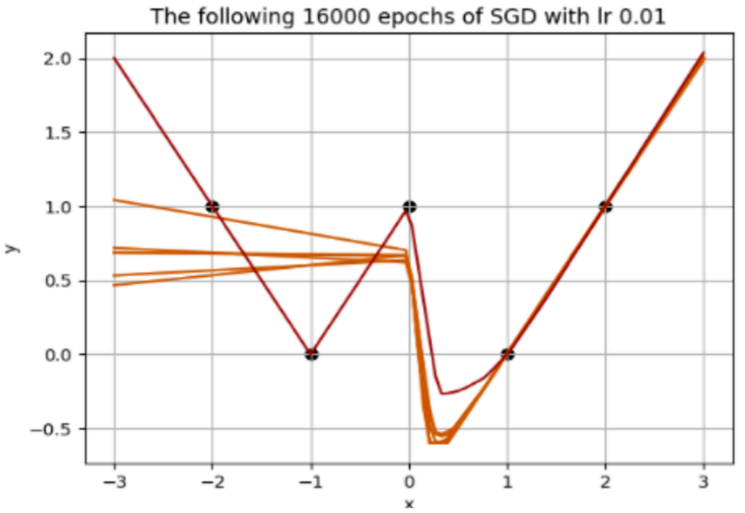
- We did not say anything about:

Local to *Global*

~~Next~~ This class is about local to global:

On the location of convergence and the phases of training

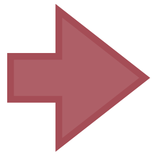




Message 1, Class 17

Designing Gradient

Based Optimizers



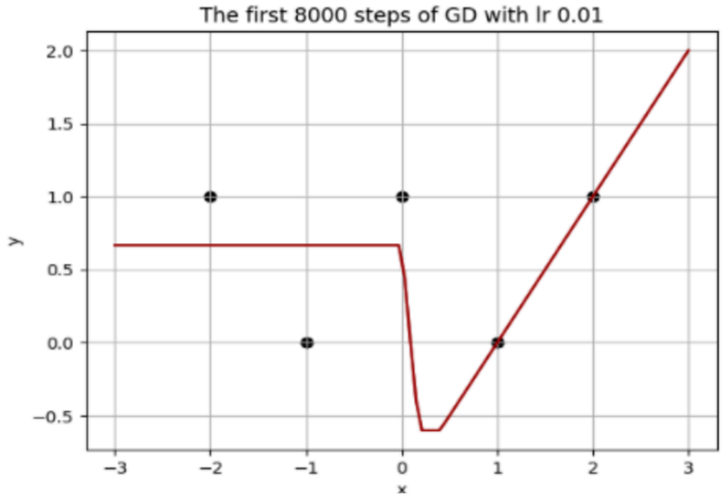
How to go:

from *infinitesimal* to *local*

(TO GLOBAL)

Regime 2

- ***Not enough exact*** data!
- Enough compute



We want a model that:

- Can fit them all.
- It's the simplest such model.

A training pipeline which goes to the best global minima.

Contents

- Location of Convergence
  - Some examples with NN
  - What gradient methods were
- (Stable) Implicit Regularization
- Edge of Stability

The Question

~~How to go optimally from *local* to *global*~~

Where does our training algorithm go in the case of neural networks?

A preliminary

Speaking Math

Where do algorithms go?

This is not an *optimization* question:

It is a dynamical systems question!

Questions are:

- What are the basins of attraction? Do trajectory recur? **(Long-term behavior)**
- Is the invariant set stable **(Stability in the sense of Lyapunov, …)**
- How does the behavior changes with the (hyper)parameters **(Bifurcation Analysis)**

Class 17

Infinitesimally, to go down *optimally* we follow the gradient:

$$\begin{cases} \theta_0 = \text{initialization} \\ \dfrac{d\theta_t}{dt} = -\nabla L(\theta_t) \end{cases}$$

How to pick initialization?

What is the optimal discrete step size version of this?

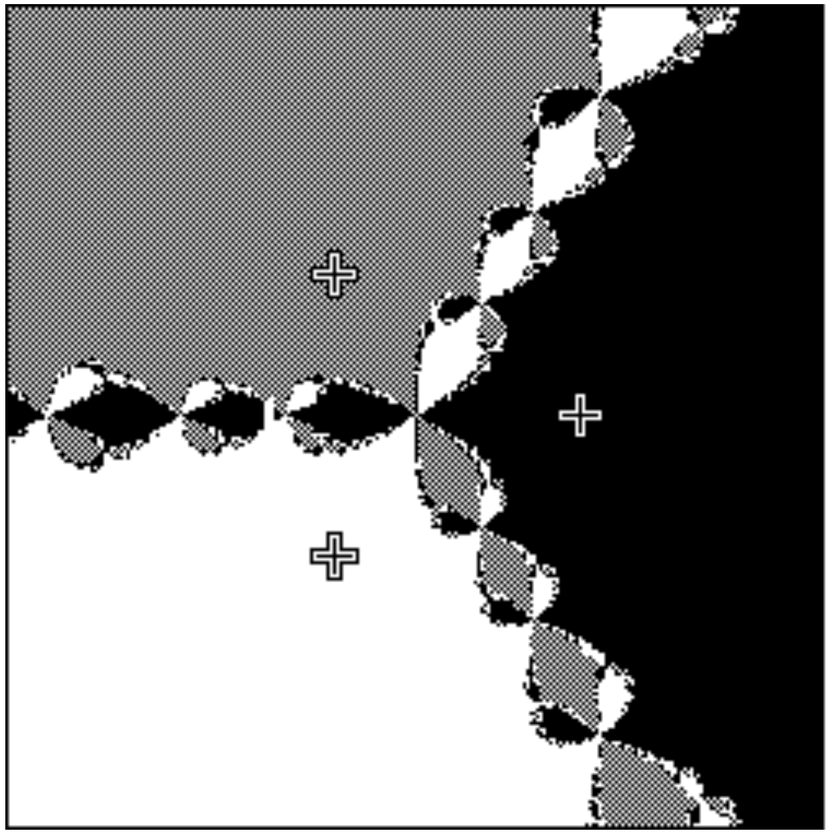
Classical PoV

Classically, initialization is about:

$$\begin{cases} \theta_0 = \text{initialization} \\ \dfrac{d\theta_t}{dt} = -\nabla L(\theta_t) \end{cases}$$

In what basin of attraction you are:

It is about:
which minimum



Basins for Newton on $L(\theta) = \theta^3 - 1$

Message 3, Class 17

- We want to pick the architecture and the initialization to enhance trainability.
- Not really for purposes of approximation or location of convergence!

So let's study the *optimizer* instead of the initialization

Questions

- We would like to design a globally optimal gradient based optimizer.
- We will see that they have been designed for something else:

Convex functions

- We do not have the tools to discuss which one is local to global optimal, yet.
- So we will just explore:

Where they go in NN

What tools can we figure out for the problem

Part 1

What can we expect empirically
&
Refreshing the gradient based methods

Location of Convergence

- **Location of Convergence**
  - **Some examples with NN**
  - What gradient methods were
- (Stable) Implicit Regularization
- Edge of Stability

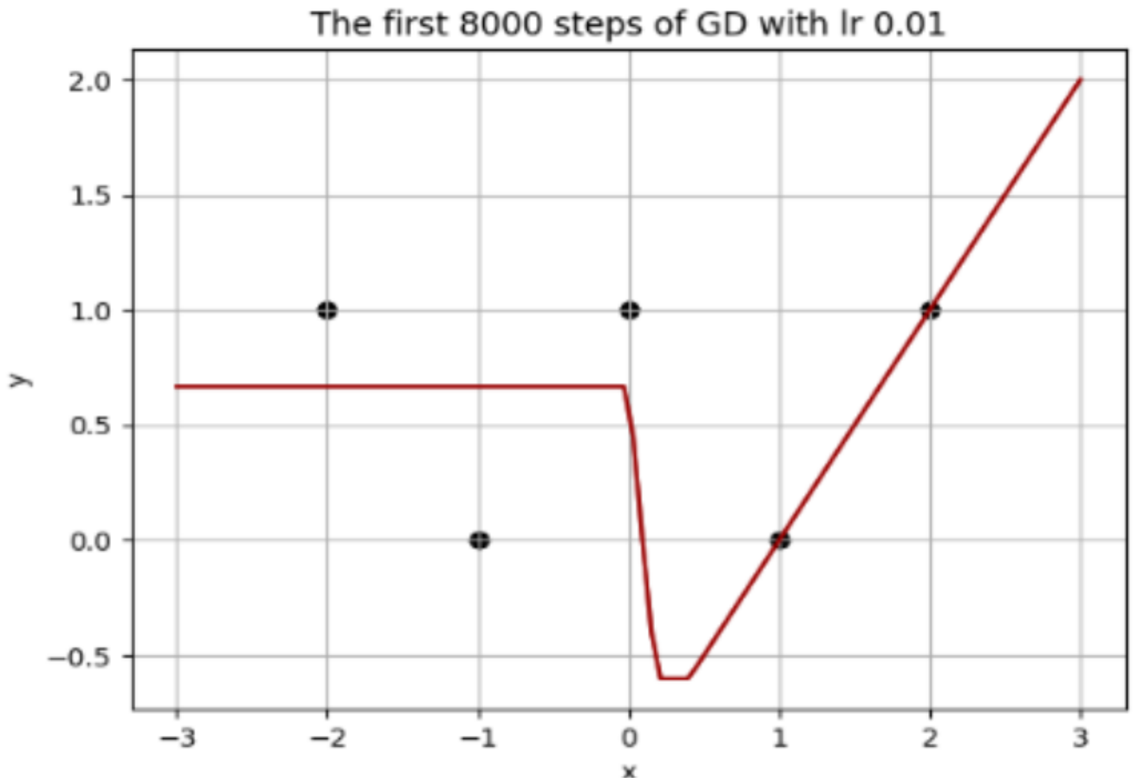
Location of Convergence

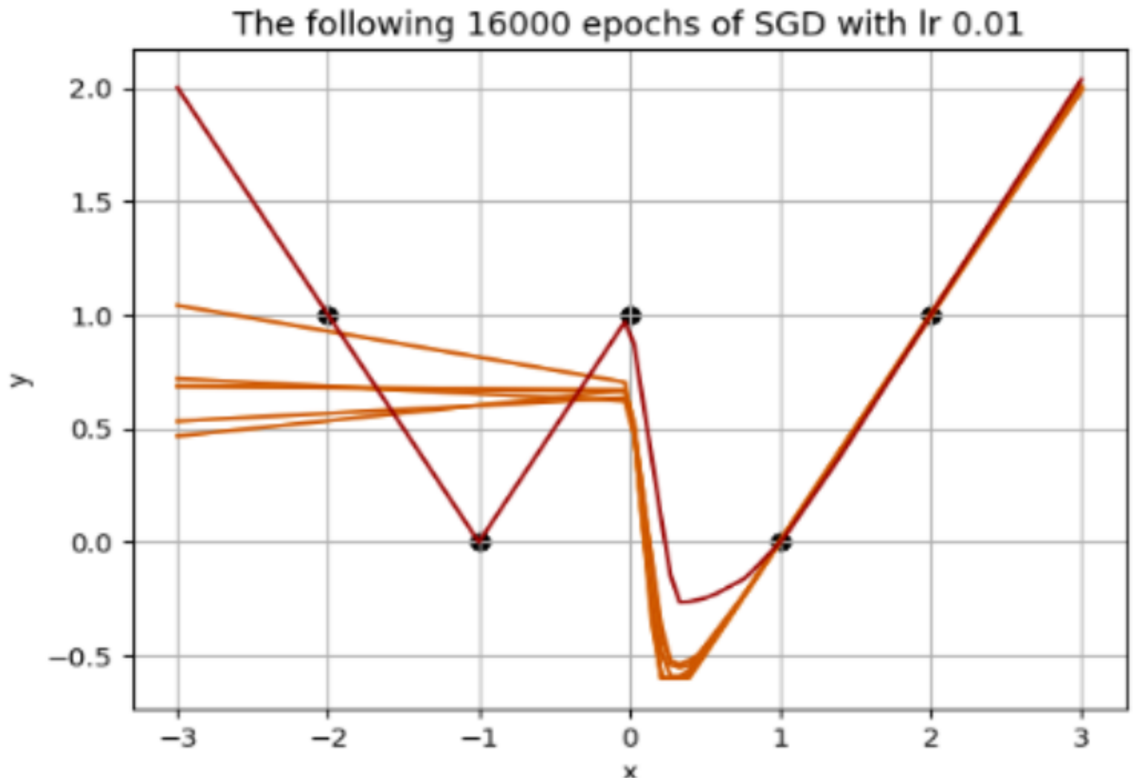
Example 1:

Different algorithms go to minima with different training loss.

SGD *vs* GD

Practical example 1: ReLU network on W-data.

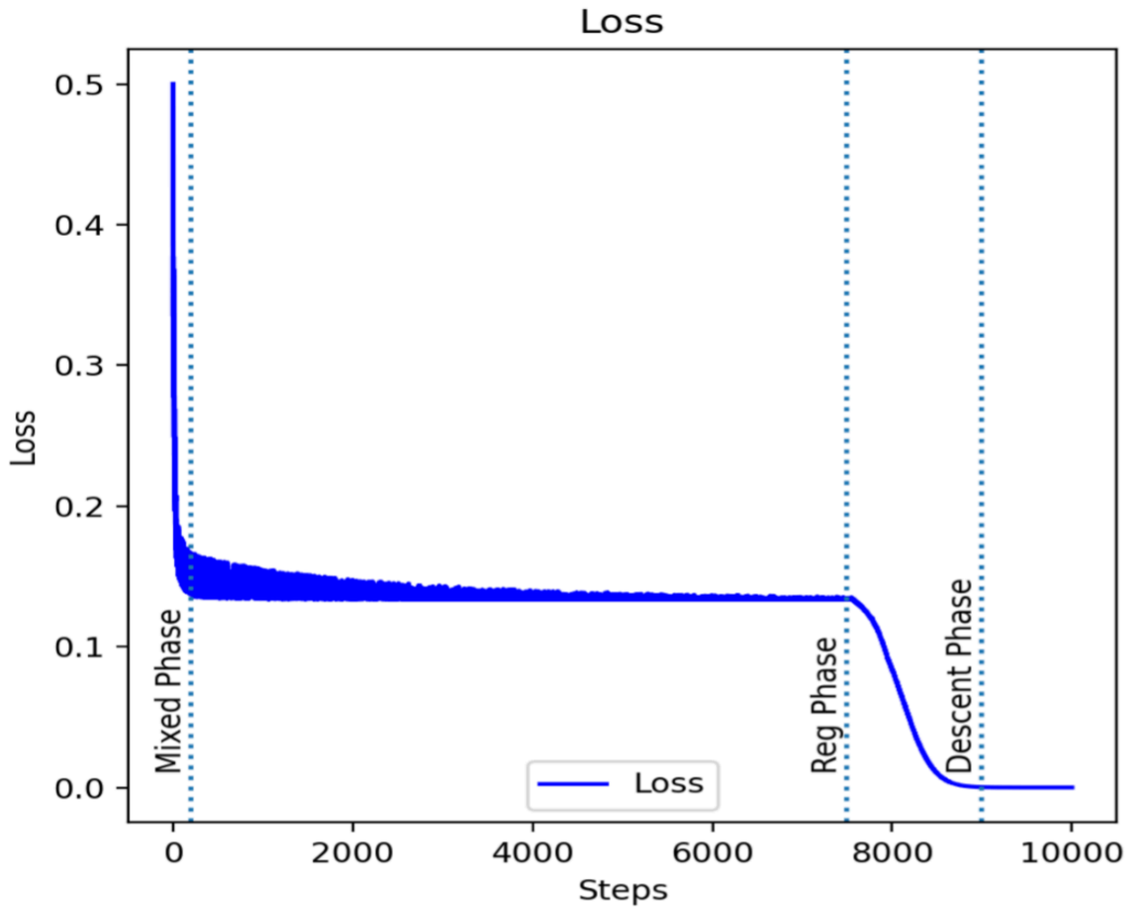


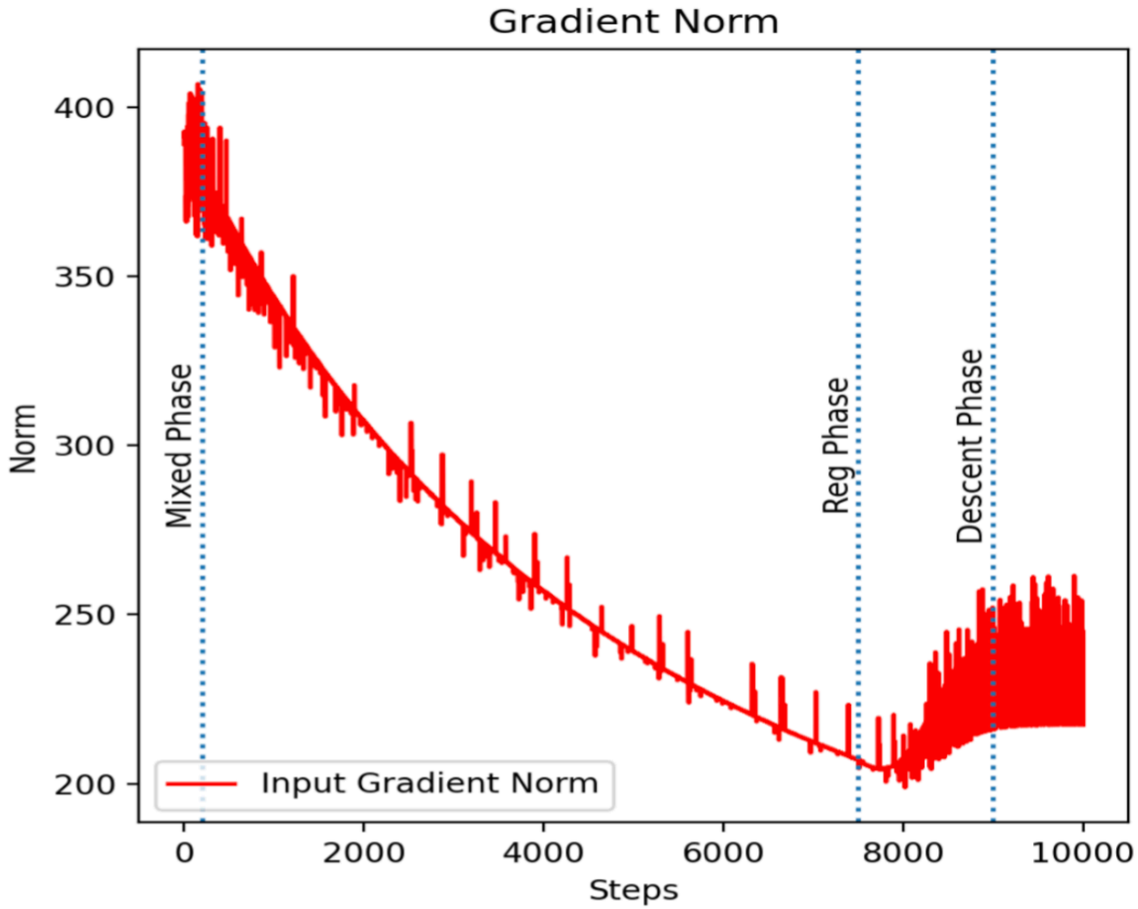


MSE on ReLU Shallow Network with 20 neurons

Implicit Regularization

SGD is cooking something down there





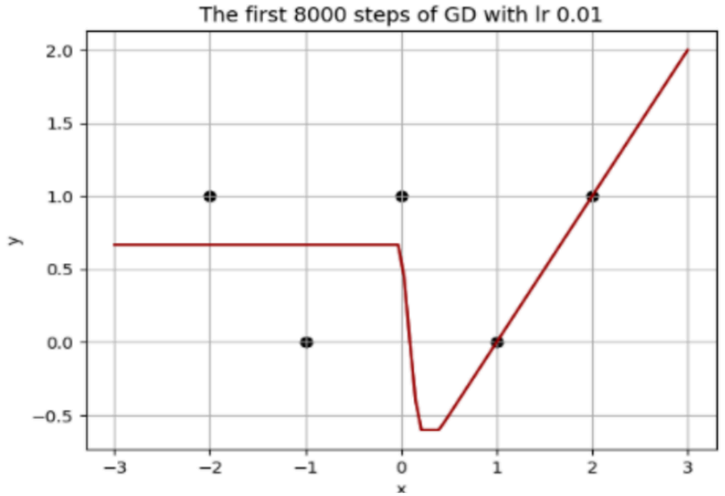
SGD *vs* GD

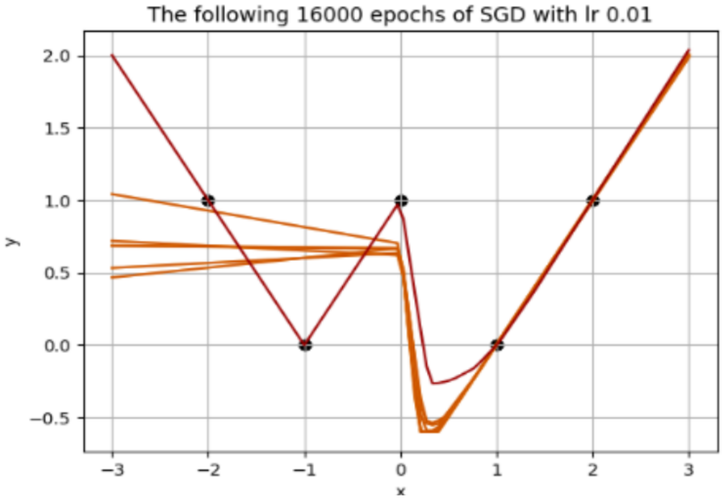
Practical example 1: ReLU network on W-data.

- There are many minima.
- Which one you find matters.

$\Longrightarrow$

- GD stops in a bad one (for the training set).
- SGD escapes to a good one—through flat areas





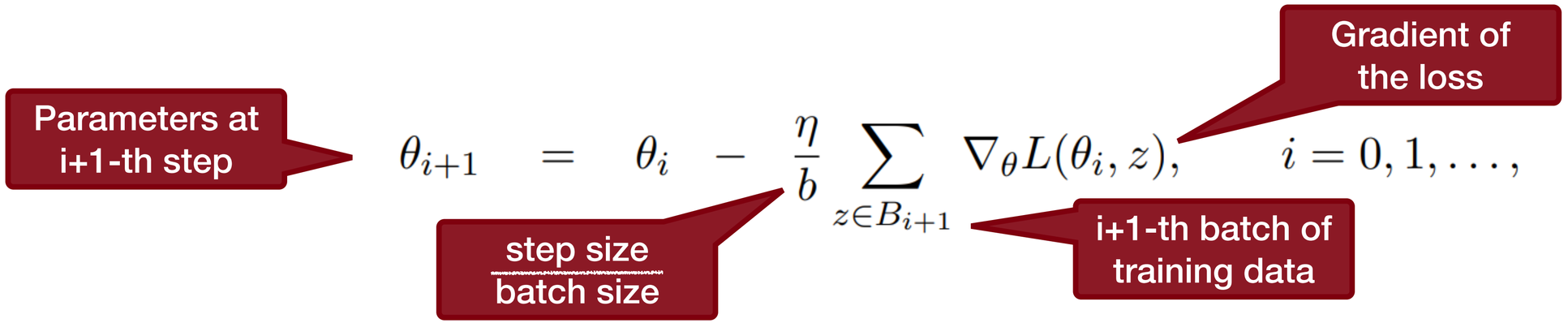
Training Networks

What qualitatively matters empirically:

- Deterministic vs Stochastic

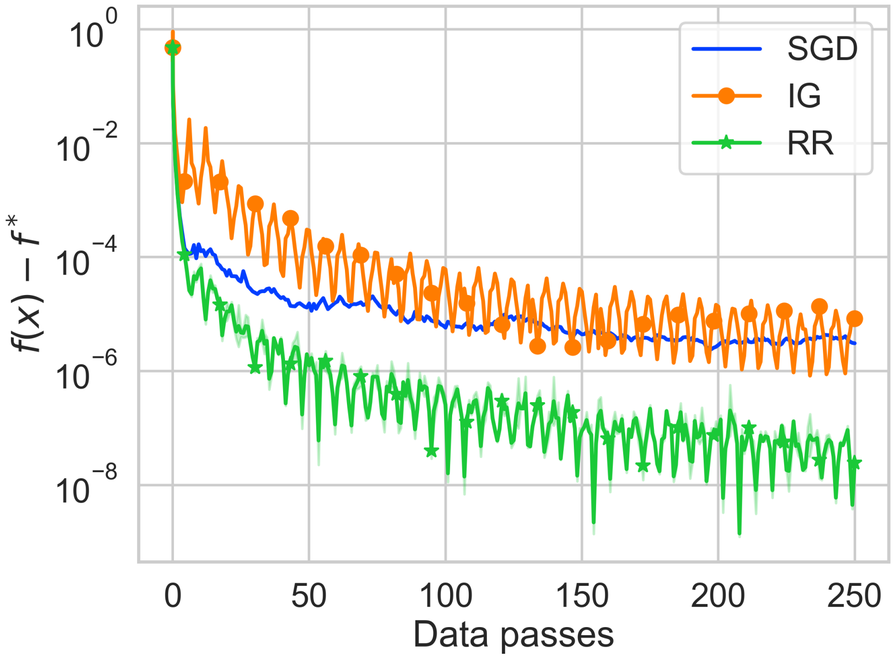
Sampling the Batches

- **With Replacement:** Sample independently at every step b data points.
- **Without Replacement:** Reshuffle the data and partition it in n/b steps.



SGD *is* Without Replacement

SGD without replacement is the standard in Pytorch and TensorFlow



Generated with code based on (*Mishchenko et al., 2020*)

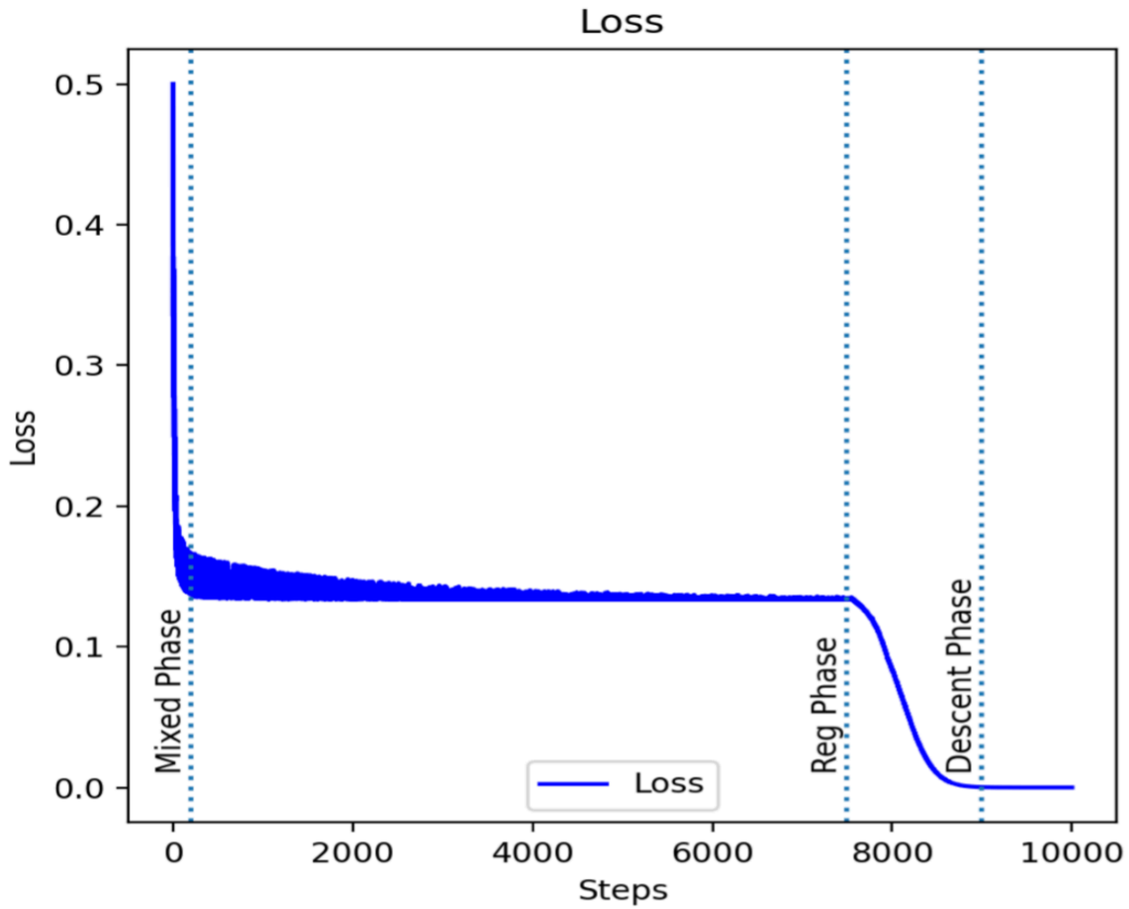
1. It is the one that better generalizes in practice (Bottou, 2009; Recht and Ré, 2013).

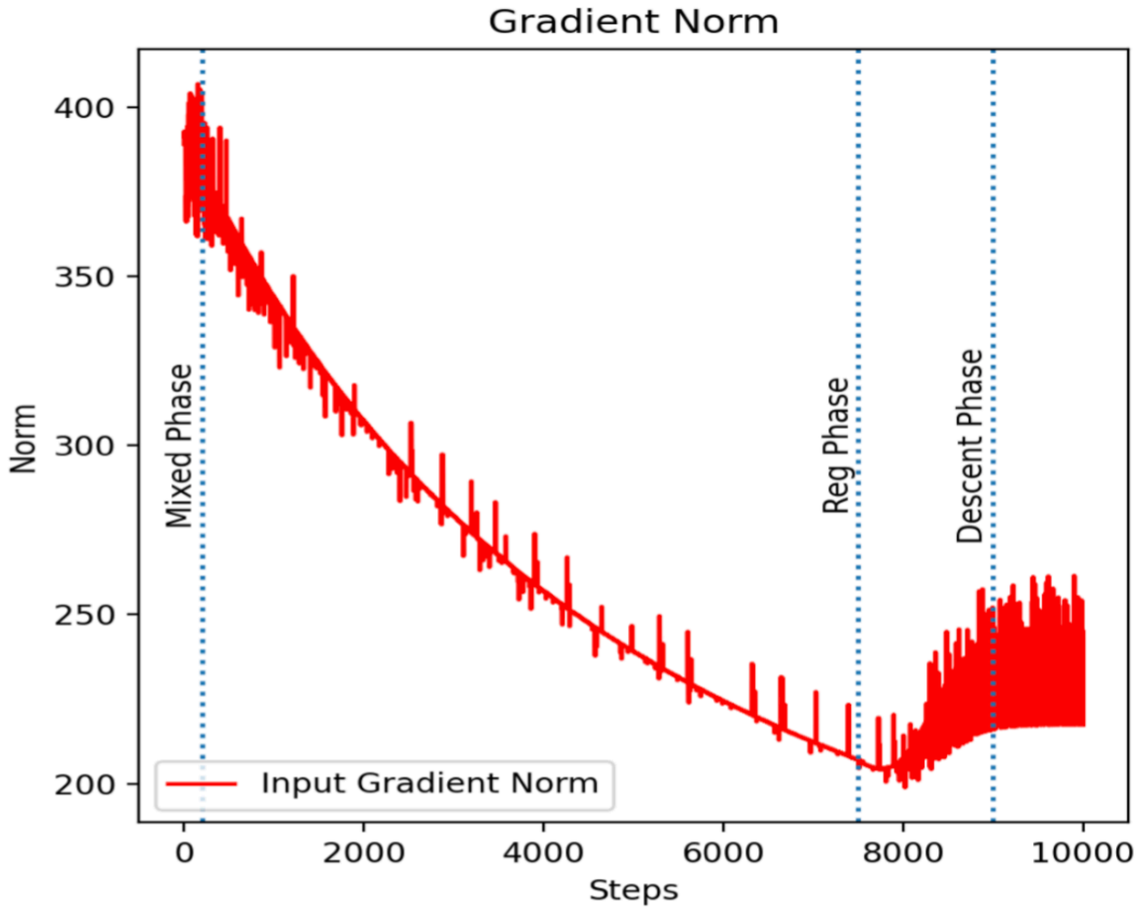
2. It is *faster* (step and convergence) (Bottou, 2009; Bengio, 2012; Mishchenko et al., 2020).

can not assume i.i.d. steps

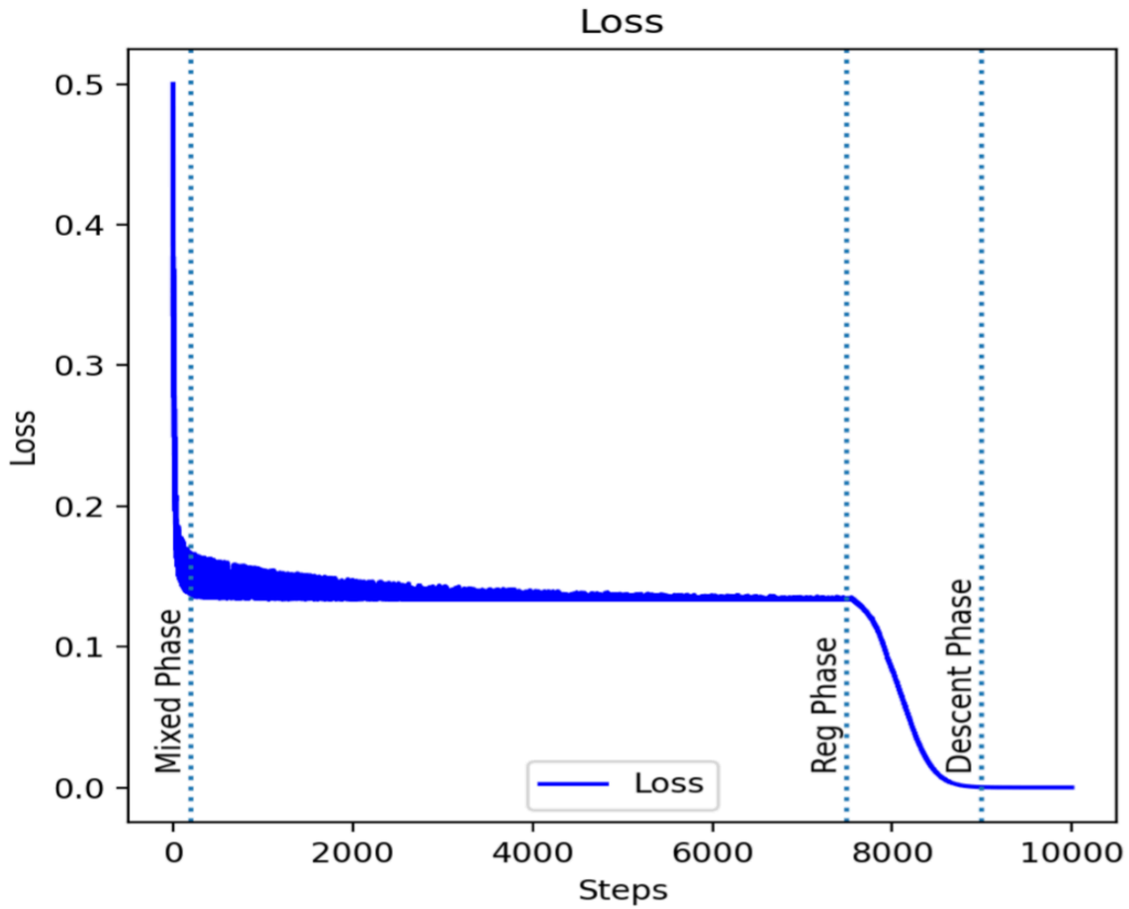
Implicit Regularization

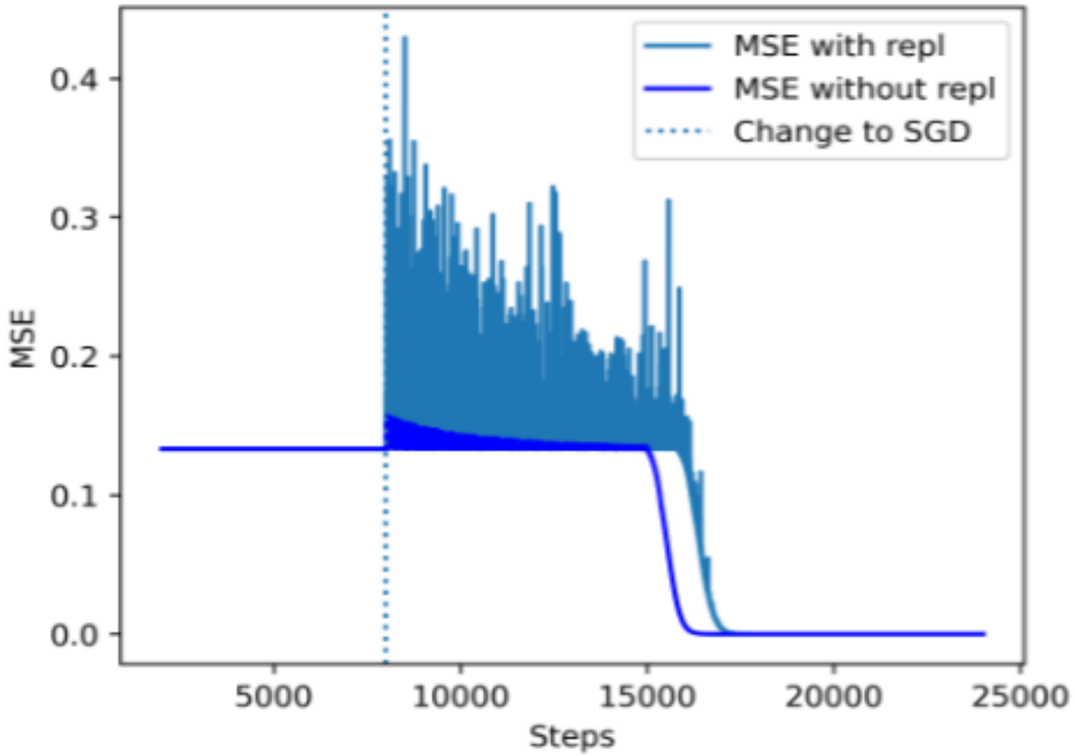
SGD is cooking something down there





Implicit Regularization





Same hyperparameters

Training Networks

What qualitatively matters empirically:

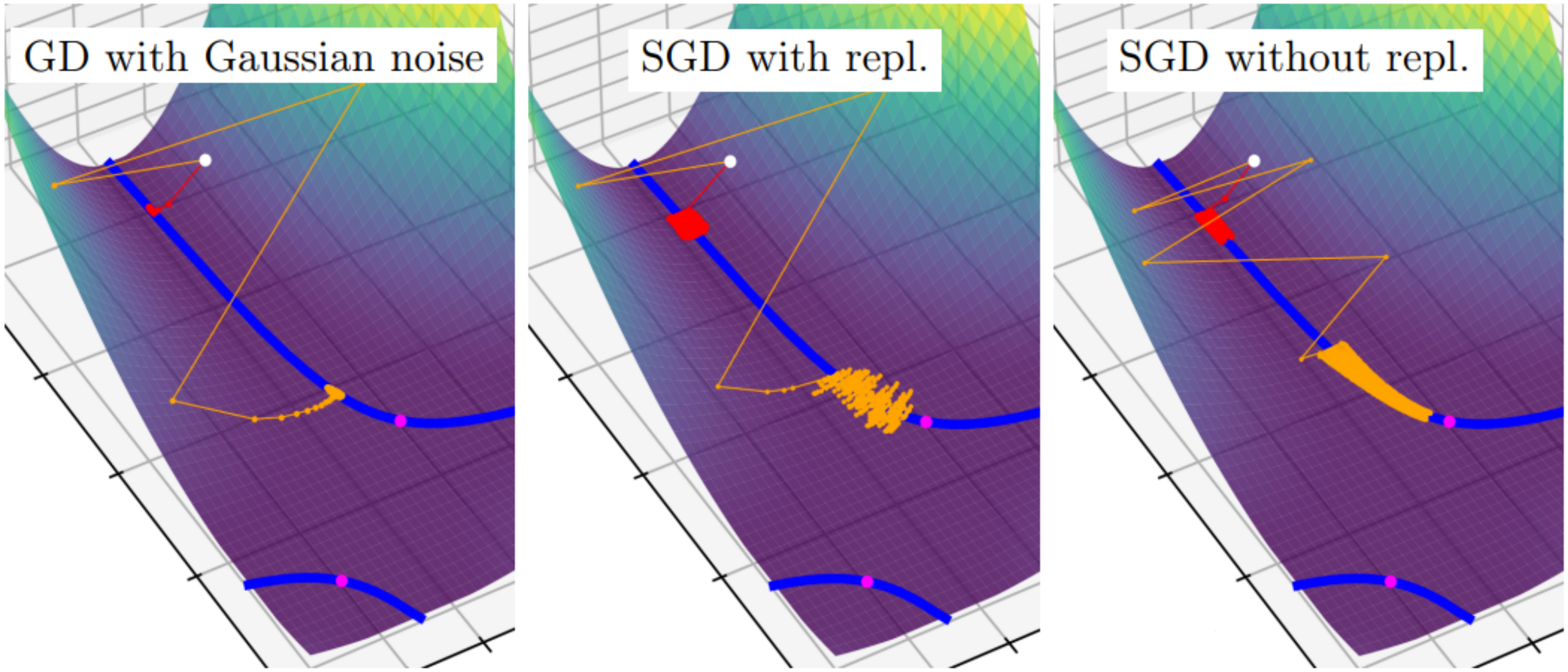
- Deterministic vs Stochastic
- What kind of stochasticity

Location of Convergence

Example 2:

Same function represented but lower norm representation.

Comparing SGDs



$$\frac{1}{2n}\sum_{i=1}^{n}(\theta_1 \cdot \theta_2 \cdot x_i - y_i)^2$$ with dataset $\{(x_i, y_i)\}_{i=1}^{n} \subseteq \mathbb{R} \times \mathbb{R}$ and parameters $(\theta_1, \theta_2) \in \mathbb{R}^2$

Location of Convergence

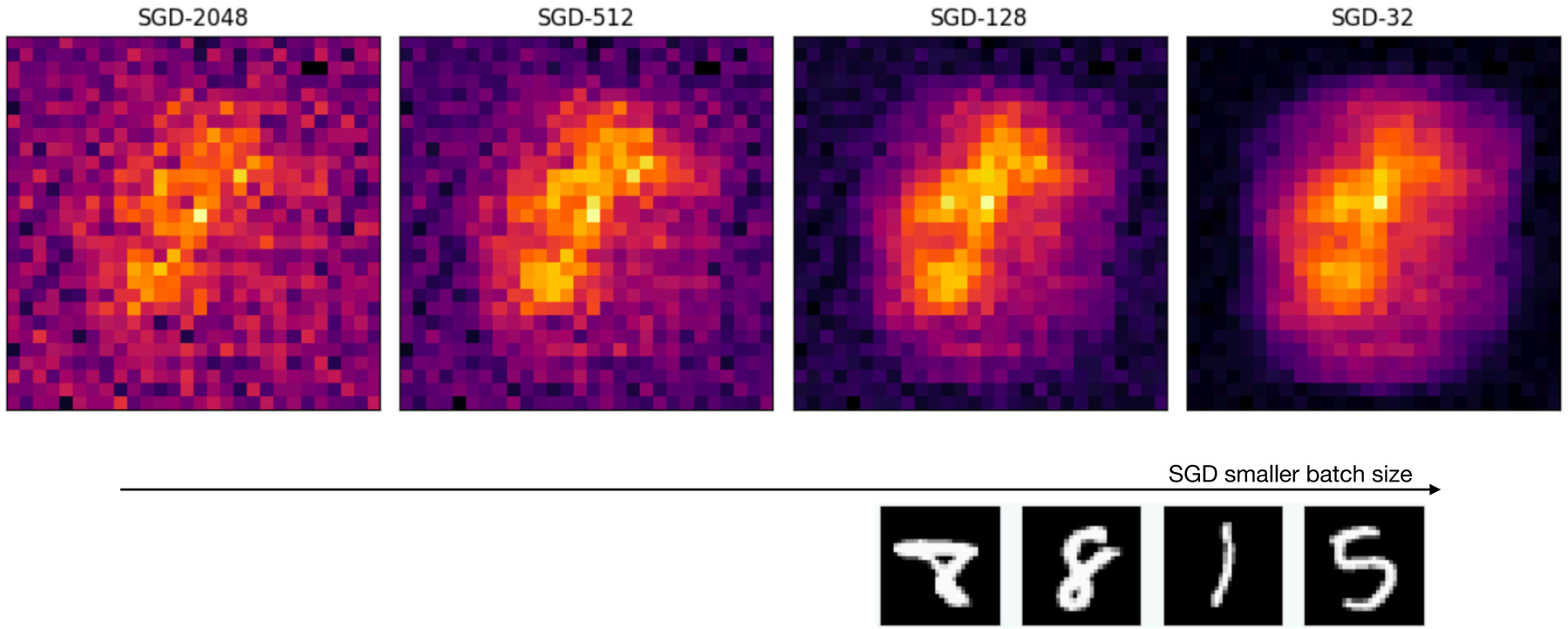
Example 3:

Different algorithms go to minima with the same training loss but different test loss.

Support Selection

SGD identifies support in the first layer, GD does not

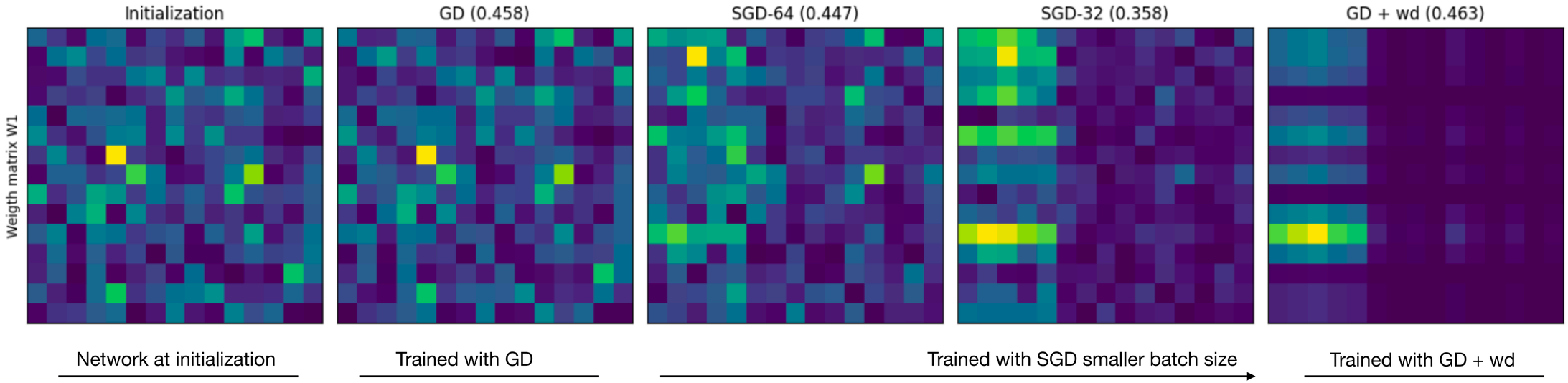
- Size of the first layer weights of an MLP learning MNIST with different optimizers



***B.**, Pinto, Poggio, (2024), ArXiv: https://arxiv.org/pdf/2406.11110, slides inspired to Andrea Pinto's MSc thesis.*

Support Selection

- How neural networks learn features from the data depends on the optimizer.
- Lower batch size yields to more compressible, interpretable, networks.
- Transfer learning may work better $\Longrightarrow$ better generalization.

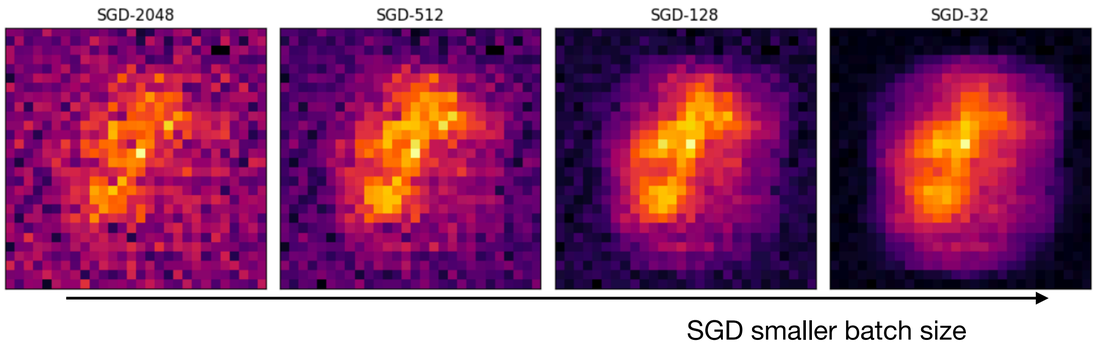


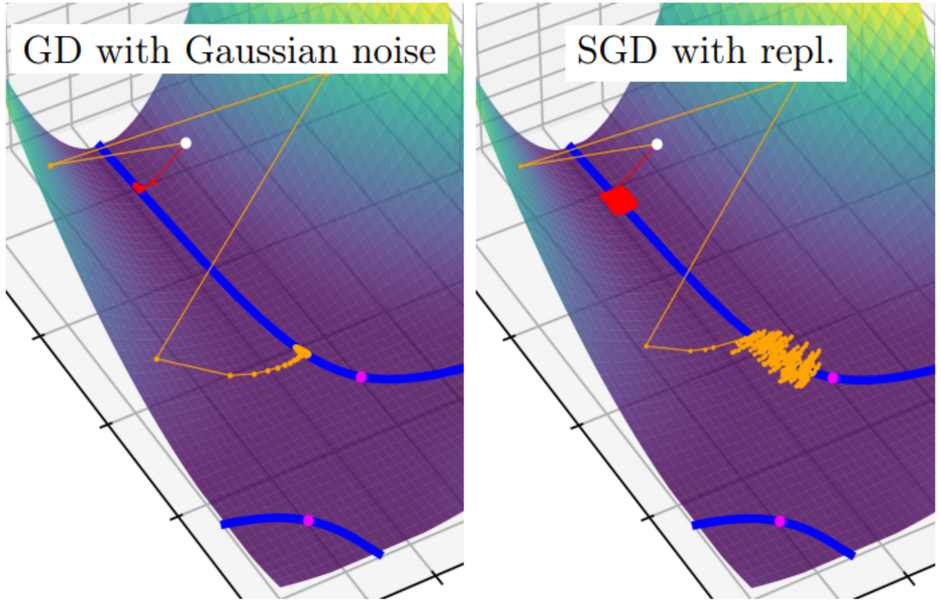
The Motivation

Goal is having best "performance"

There are many different minima ***with different performance***

Different algorithms find different minima, *consistently*





Training Networks

What qualitatively matters empirically *(for performance in Regime 2)*:

- Deterministic vs Stochastic
- What kind of stochasticity
- The choice of hyperparameters

Consistently

Location of Convergence

- **Location of Convergence**
  - Some NN examples
  - **What gradient methods were**
- (Stable) Implicit Regularization
- Edge of Stability

What is GD

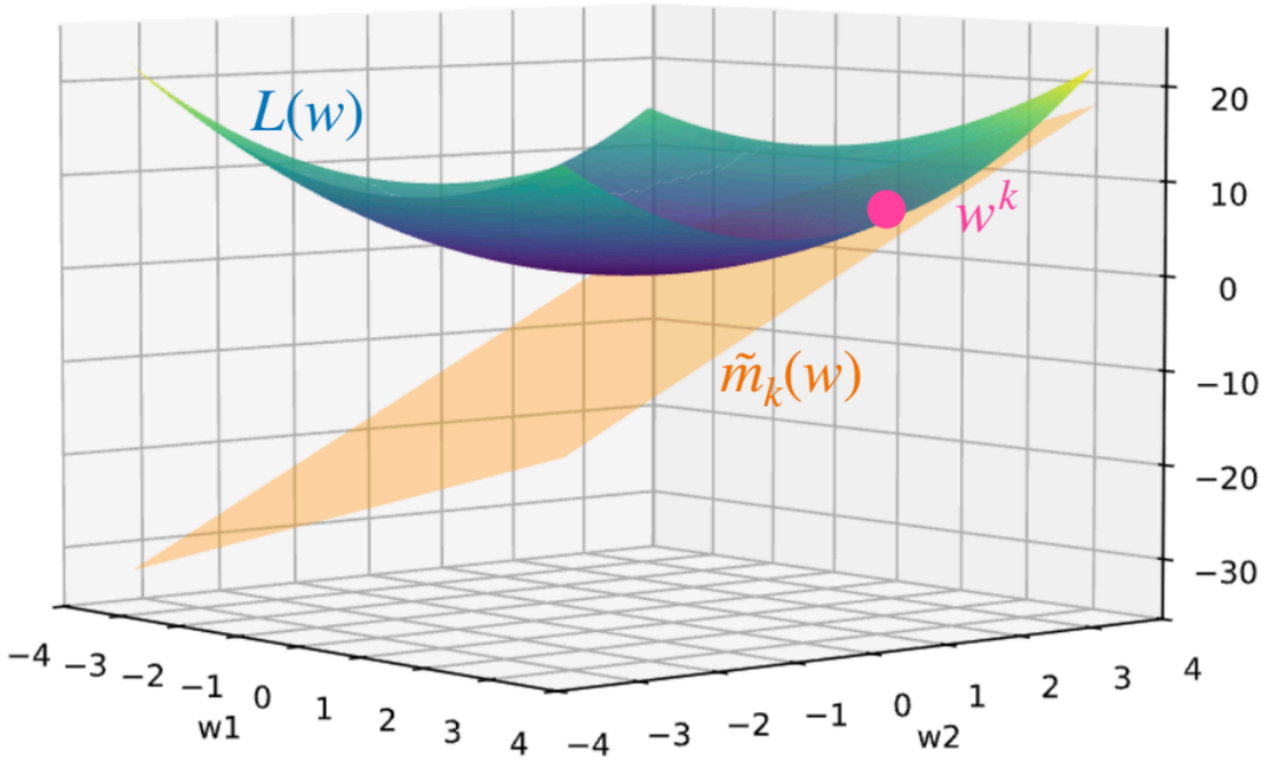
- Locally around θ_t we have the following approx (order 1):

$$\tilde{L}(\vartheta) \;:=\; L(\theta_t) + \nabla L(\theta_t)^\top\,(\vartheta - \theta_t).$$

- What is the min in ϑ of $\tilde{L}(\vartheta)$?

Minus infinity.

This is not an accurate approximation.



Proximal Methods

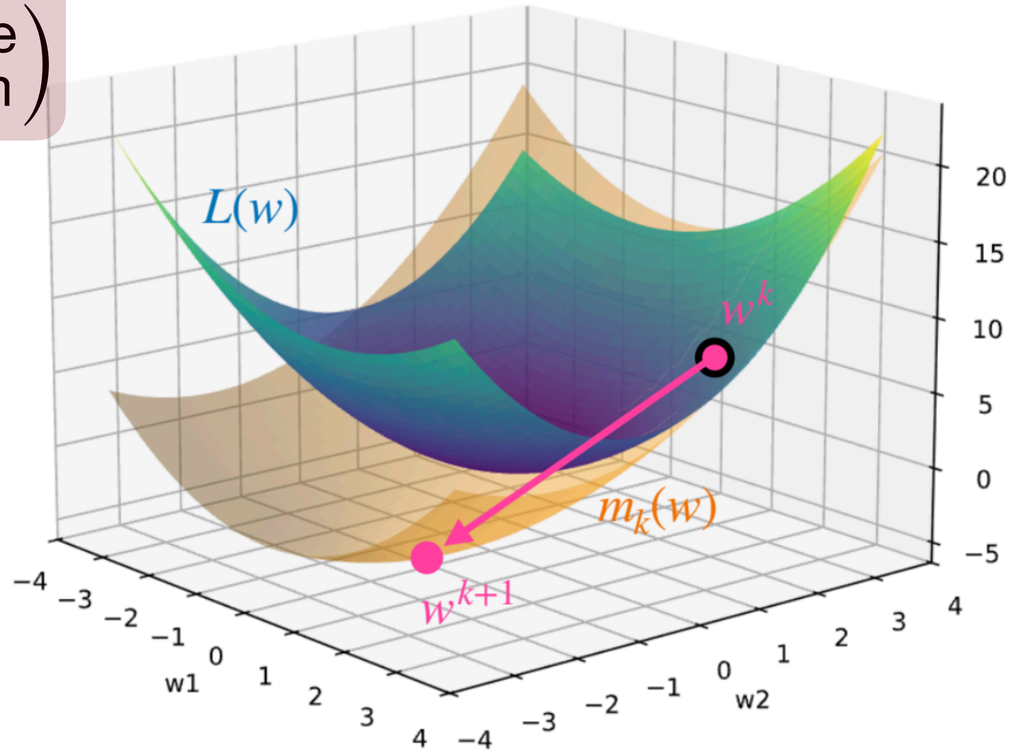
- Let say we trust it "$1/\lambda$", we want to stay closer:

$$\tilde{L}(\vartheta) \;:=\; L(\theta_t) \;+\; \nabla L(\theta_t)^\top (\vartheta - \theta_t) \;+\; \lambda \|\vartheta - \theta_t\|$$

(some norm)

- Minimize $\tilde{L}(\theta_t)$ to obtain θ_{t+1}.

- Any norm.
- Any "trust" $1/\lambda$.



GD *IS A* PROXIMAL METHOD

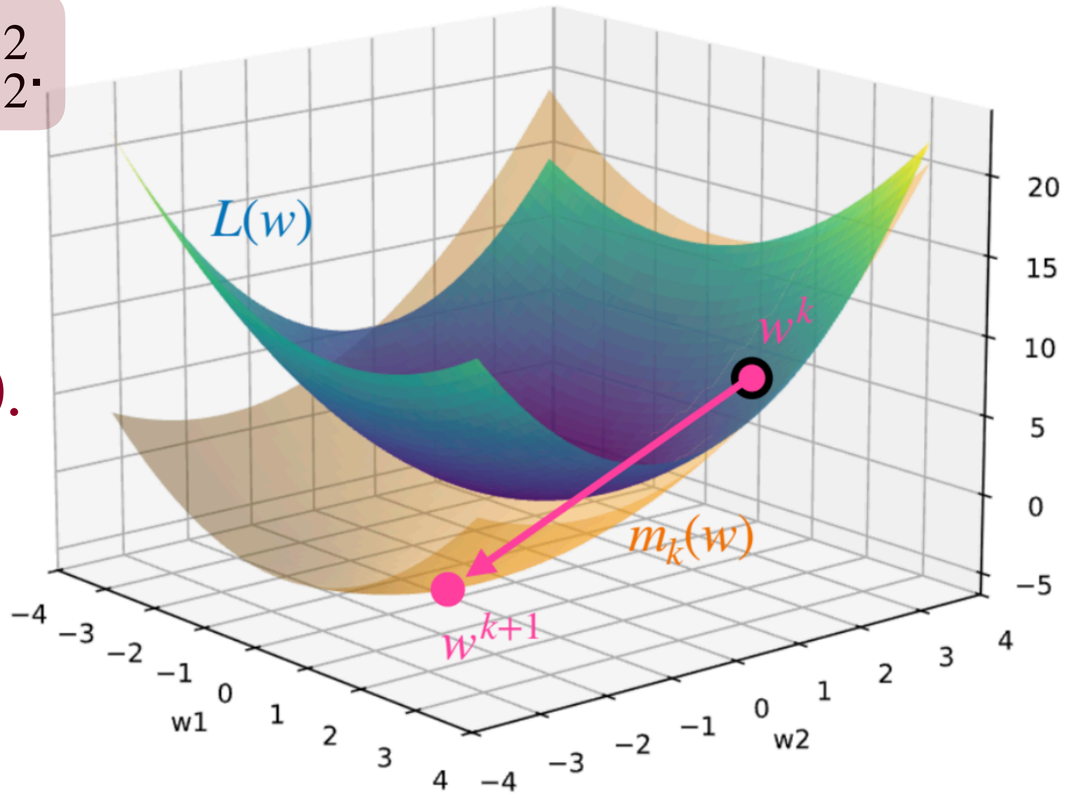
- If we trust it "$1/\lambda$" with Euclidean norm:

$$\tilde{L}(\vartheta) \;:=\; L(\theta_t) \;+\; \nabla L(\theta_t)^\top (\vartheta - \theta_t) \;+\; \lambda \|\vartheta - \theta_t\|_2^2 .$$

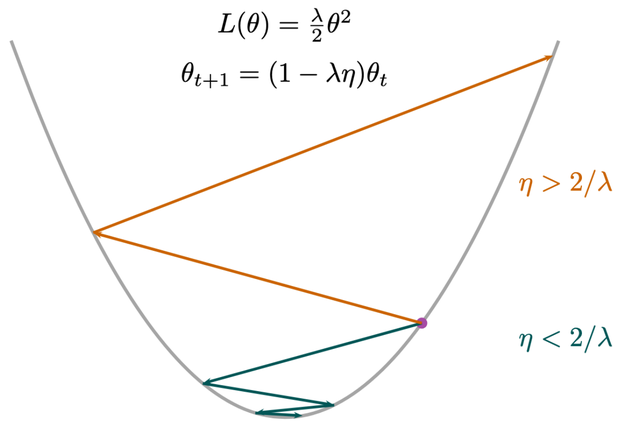
- What is the minimum in ϑ of that?

$$\operatorname{argmin}_\vartheta \left(\tilde{L}(\vartheta) \right) \quad \Longleftrightarrow \quad \nabla L(\theta_t) + 2\lambda \cdot (\vartheta - \theta_t) = 0.$$

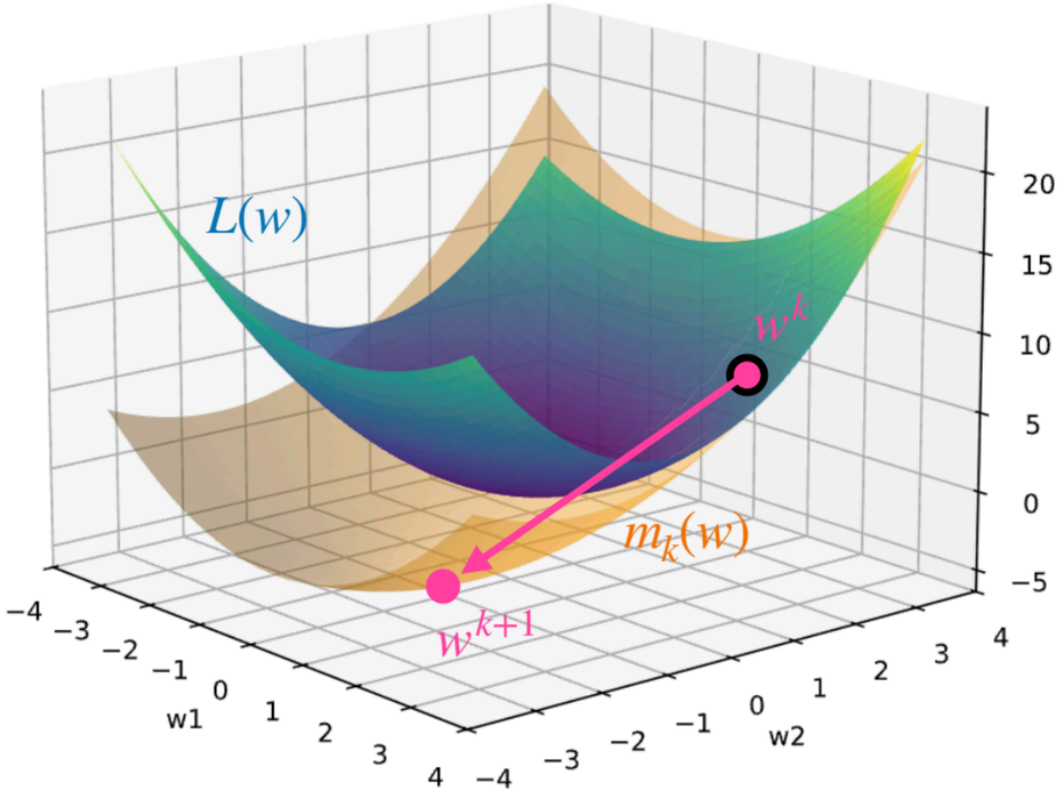
$$\vartheta - \theta_t = -\frac{1}{2\lambda} \nabla L(\theta_t) \, .$$

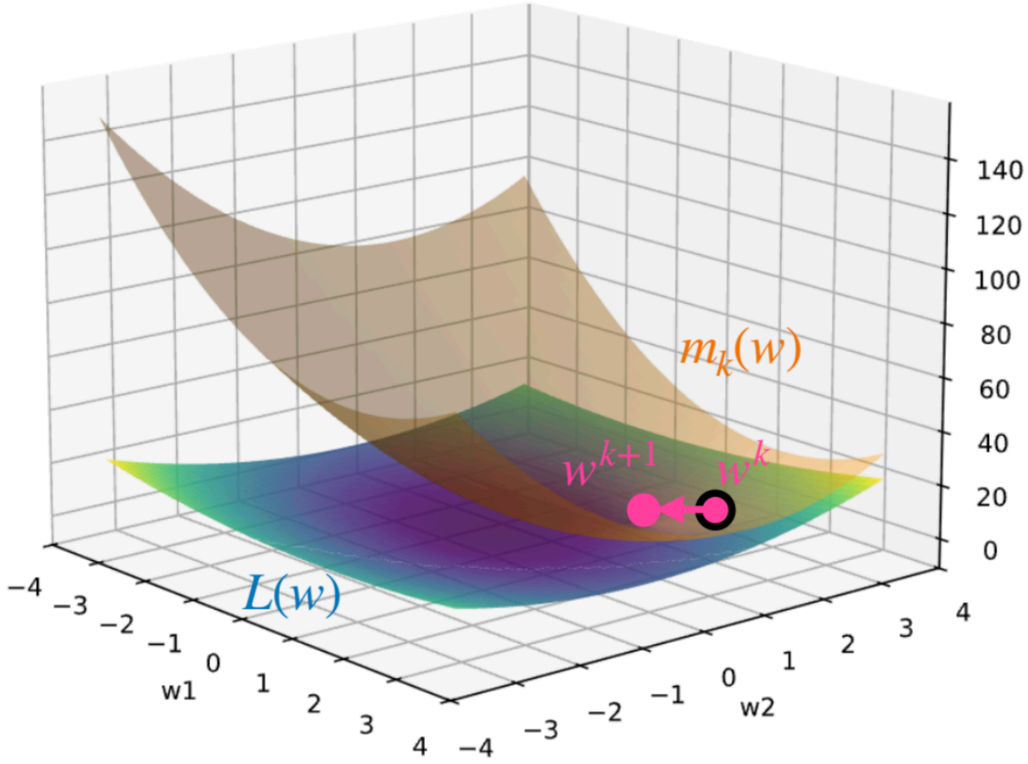


What is GD



Increasing my *trust* $1/\lambda$:





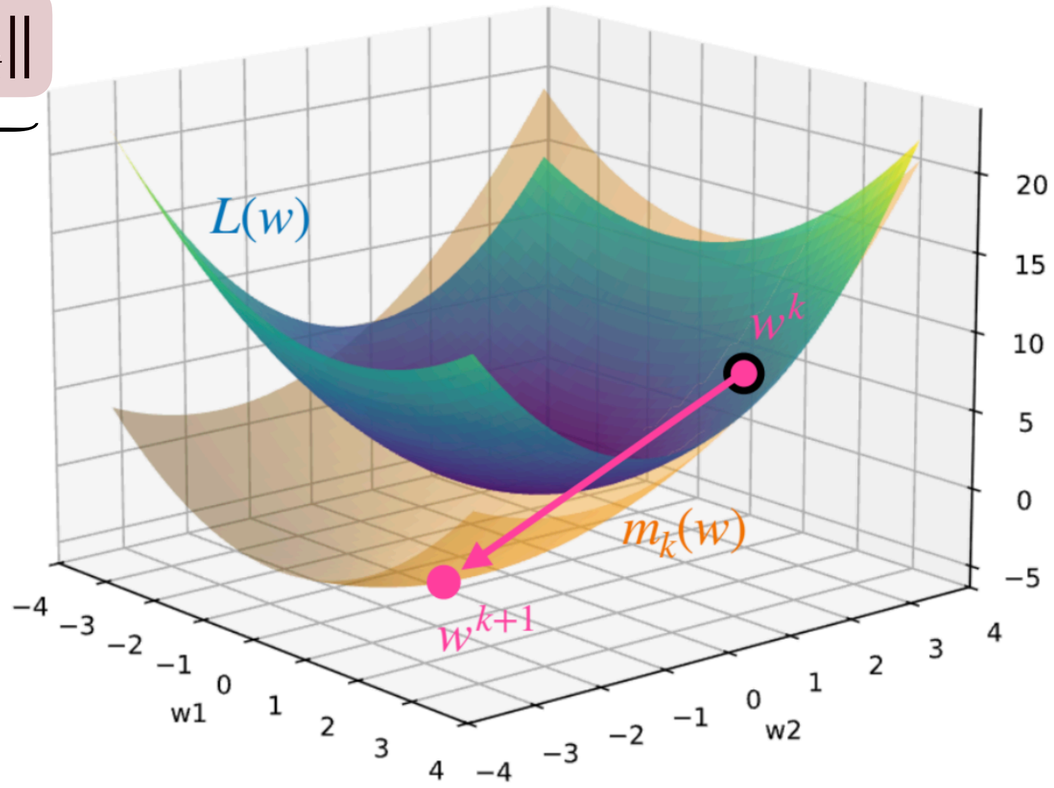
Proximal Methods

- Let say we trust it "$1/\lambda$", we want to stay closer:

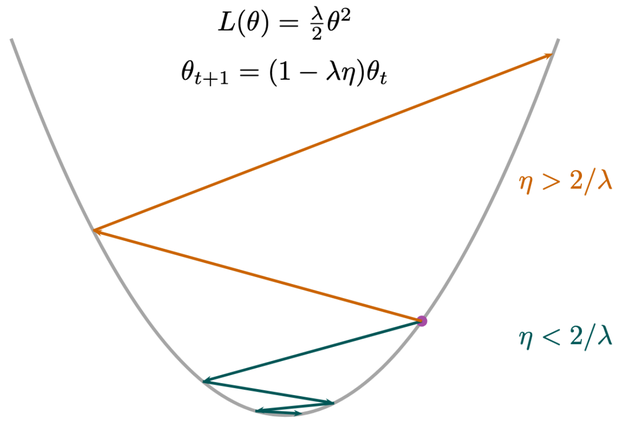
$$\tilde{L}(\vartheta) := \underbrace{L(\theta_t) + \nabla L(\theta_t)^\top (\vartheta - \theta_t) + \lambda \|\vartheta - \theta_t\|}_{\text{Convex Function}}$$

- Minimize $\tilde{L}(\theta_t)$ to obtain θ_{t+1}.

This is a convex relaxation!!

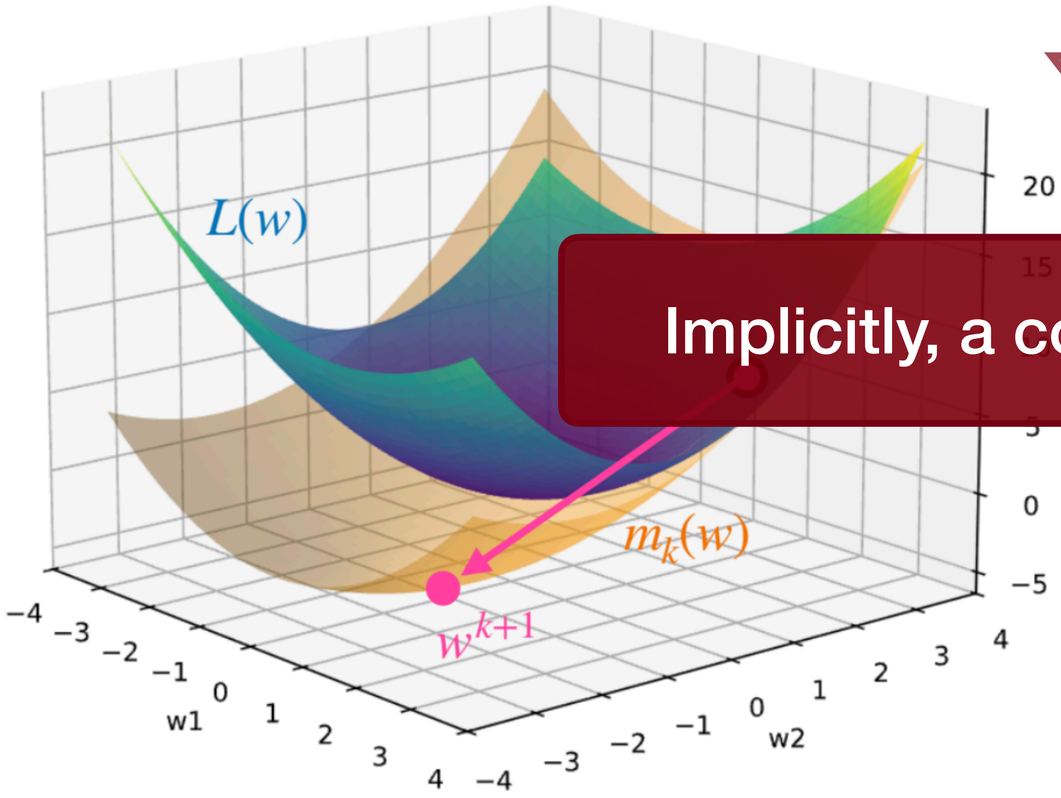


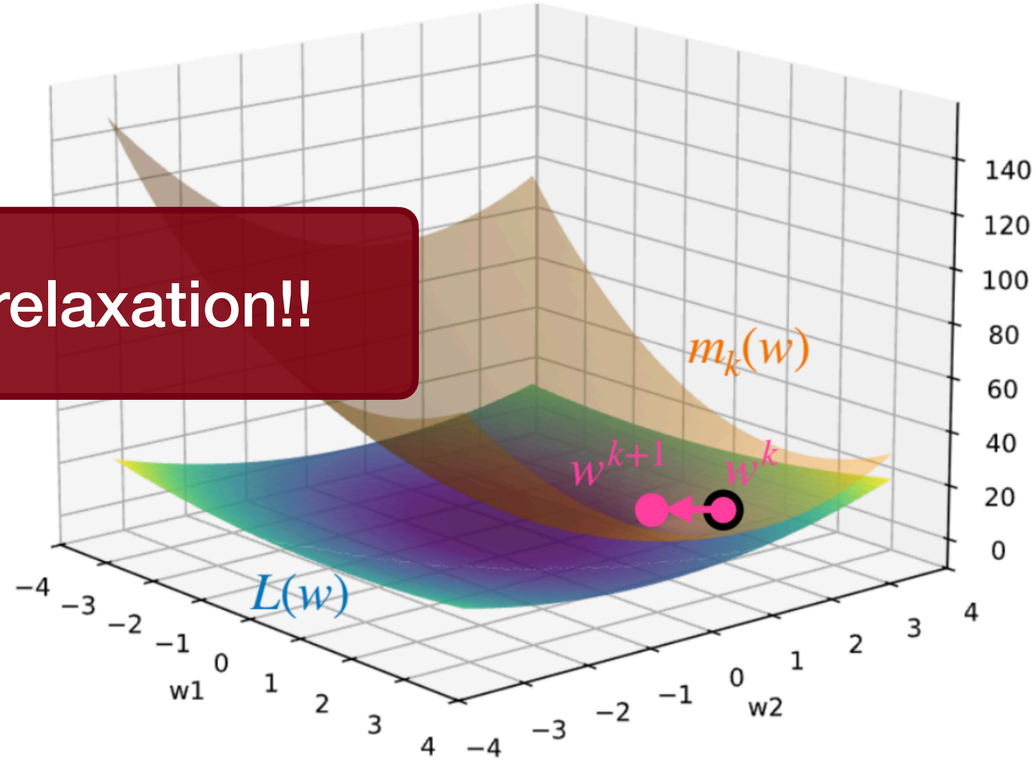
What is GD



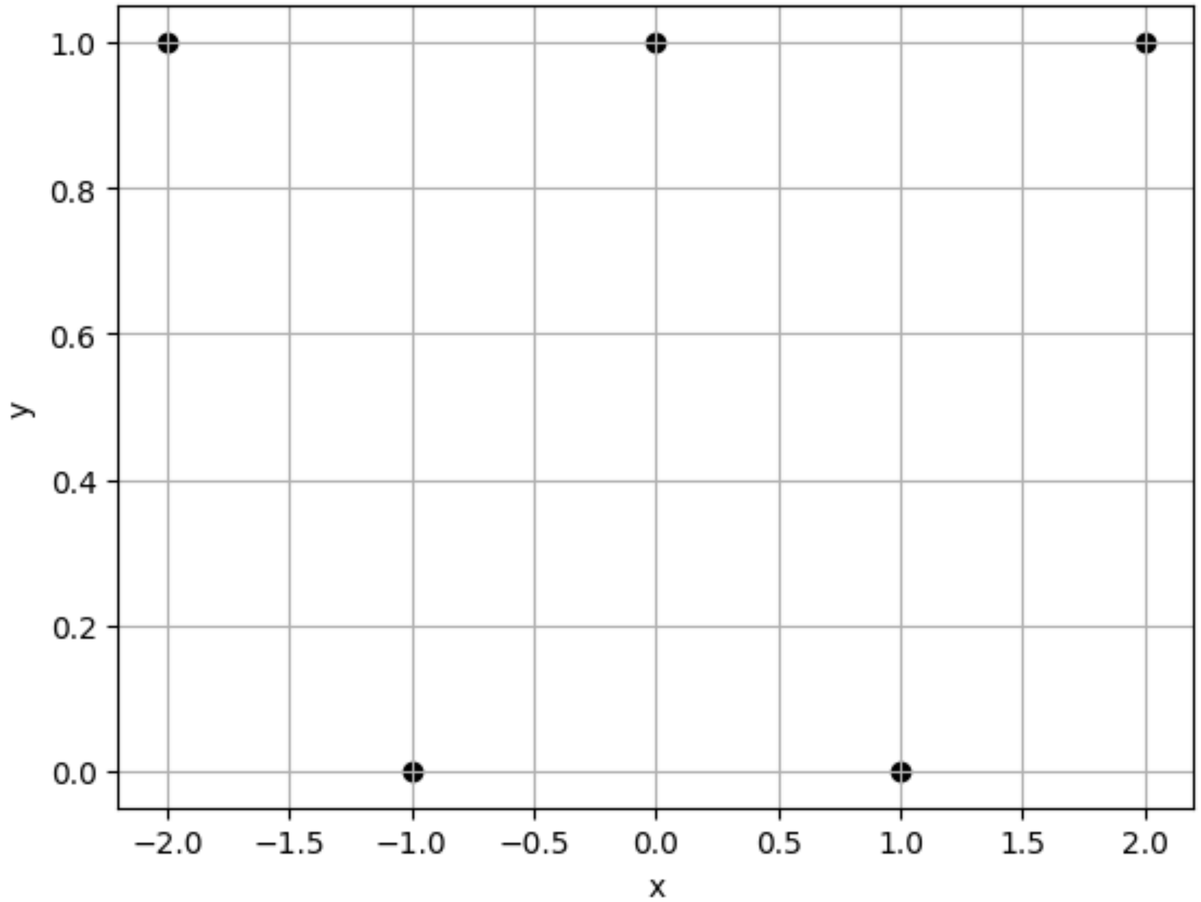
What are we saying here?

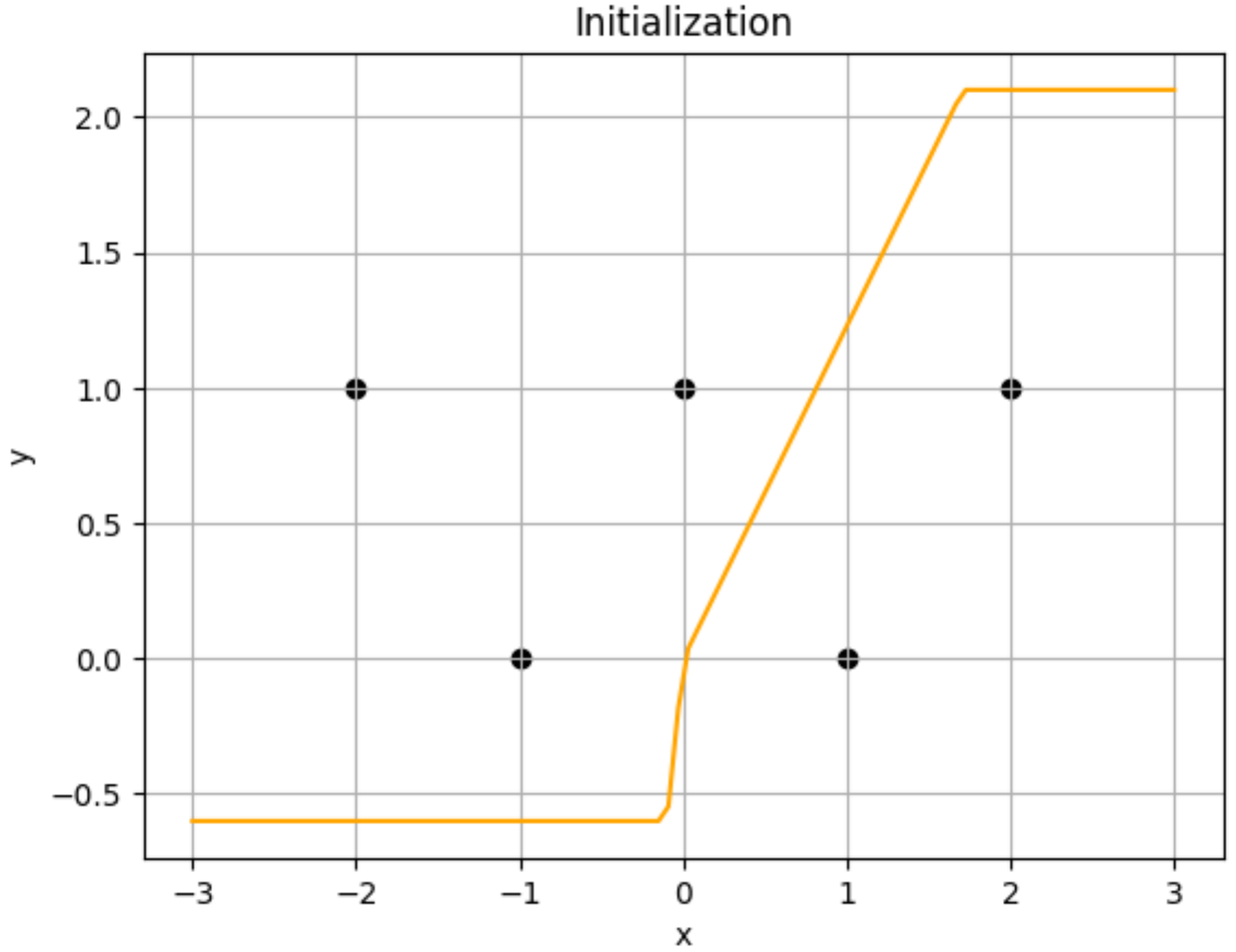
Implicitly, a convex relaxation!!





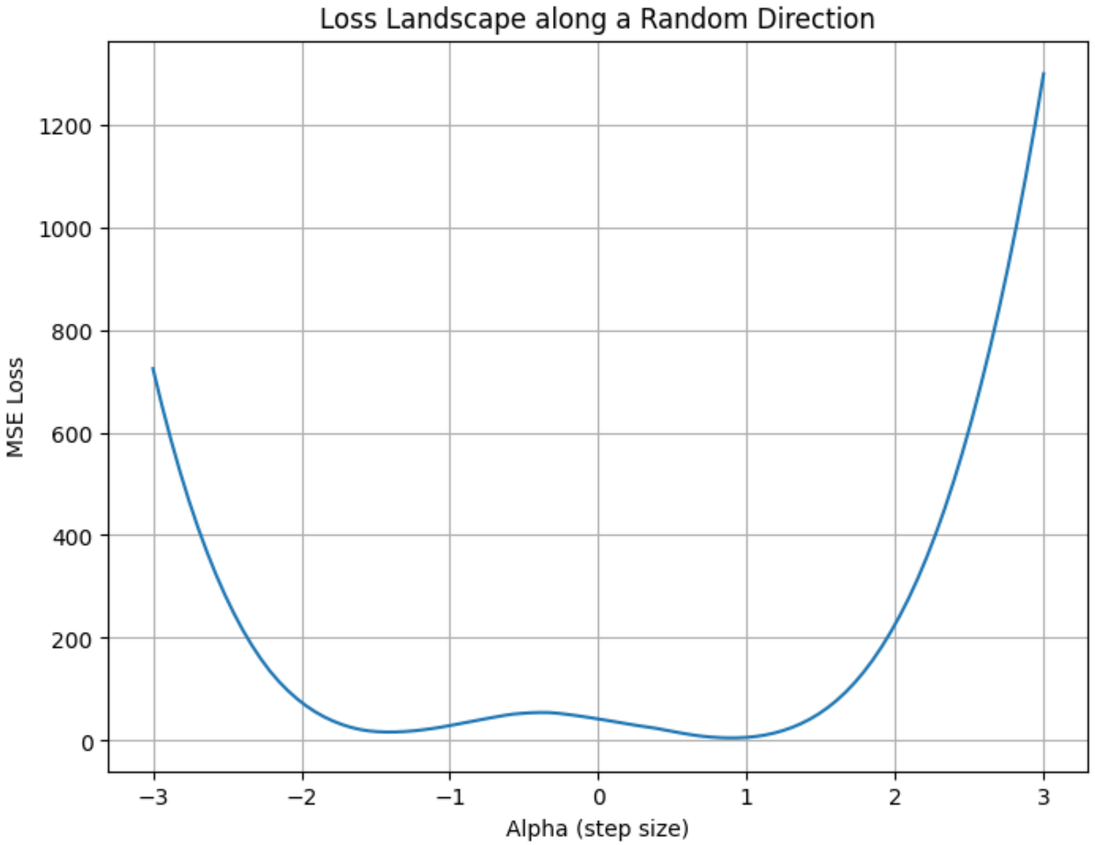
Data on a W

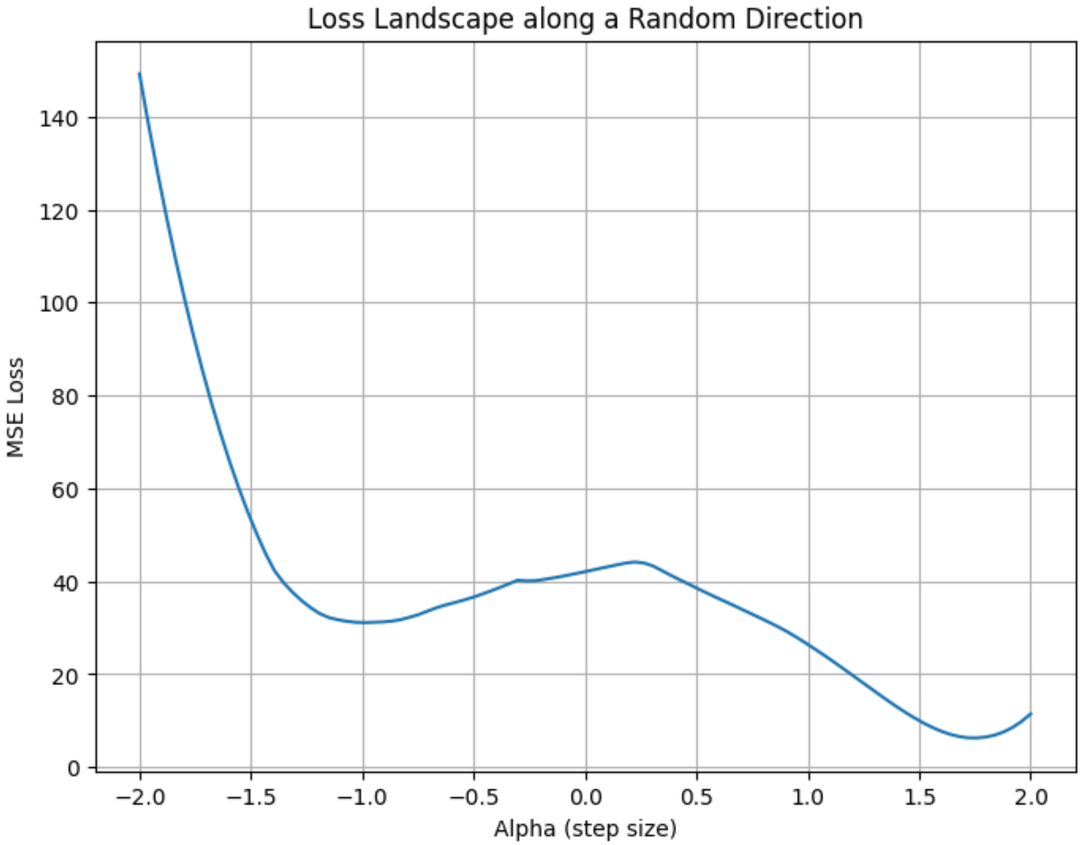




Model $f_{a,b,c}(x) = \sum_{i=1}^{20} a_i \cdot ReLU(b_i x + c_i)$

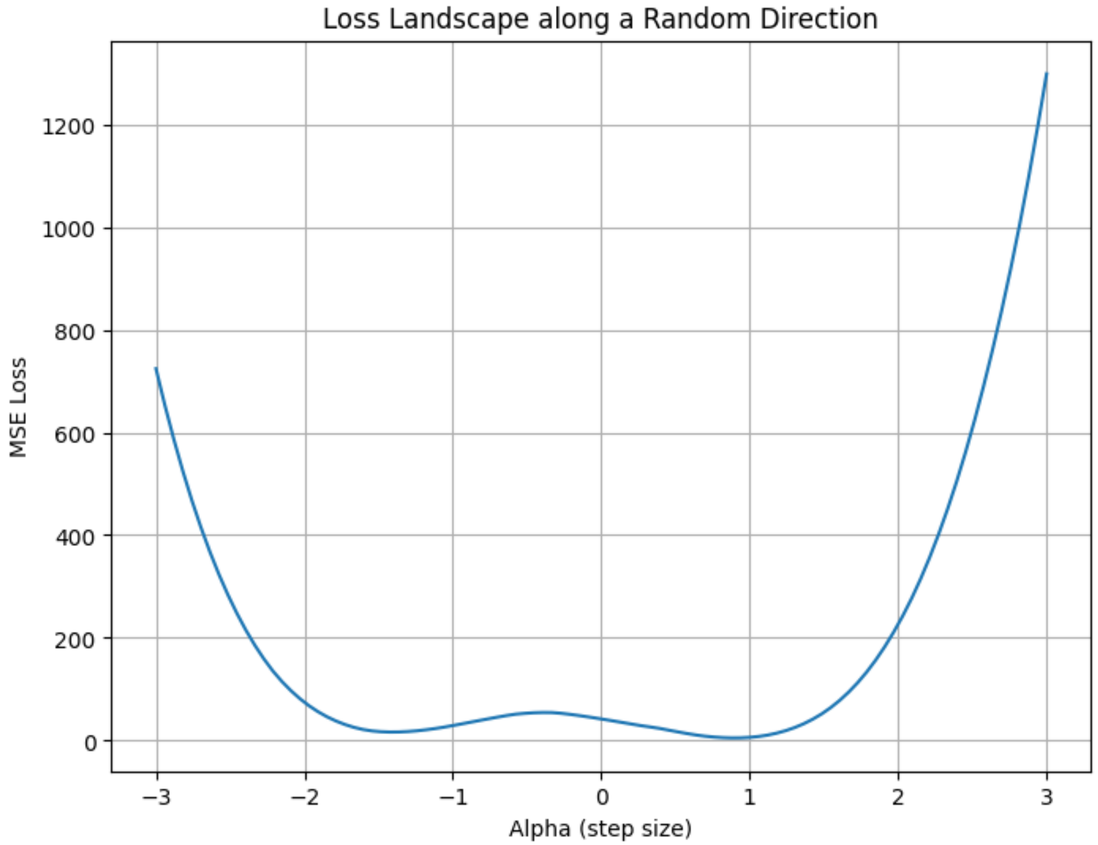
We are training NN this way:

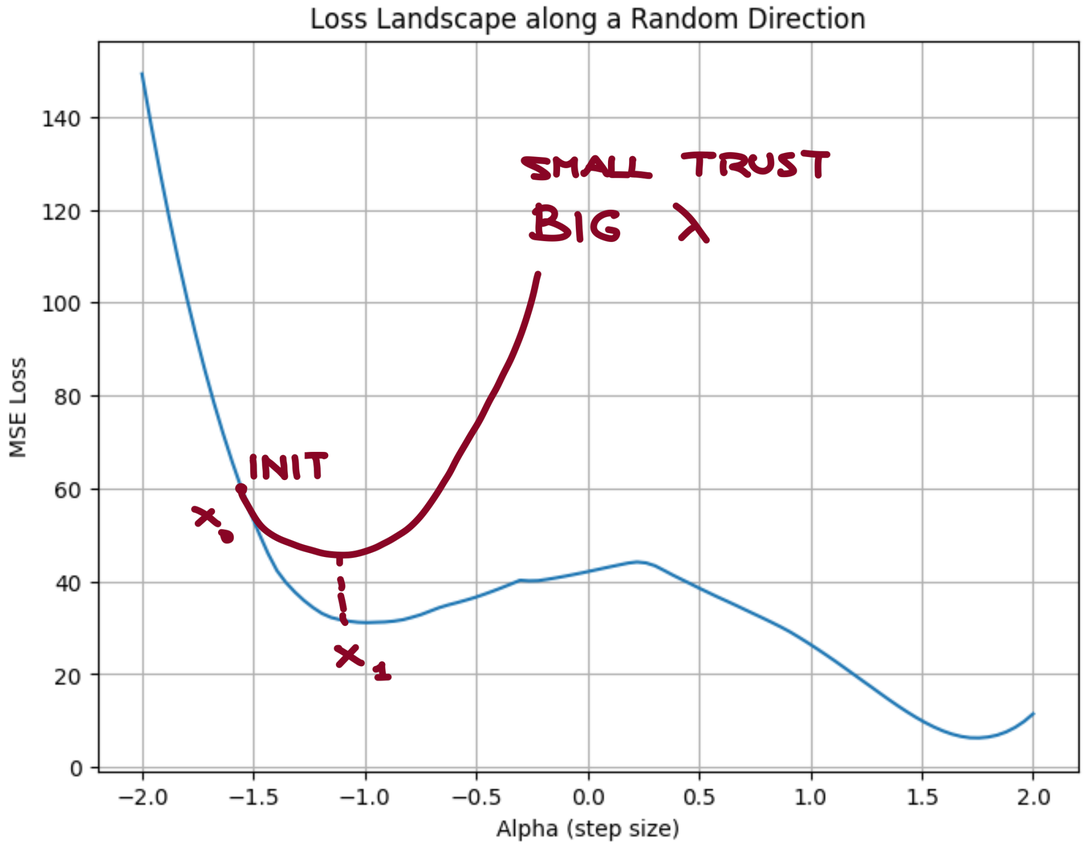




Plot of the loss landscape along a line

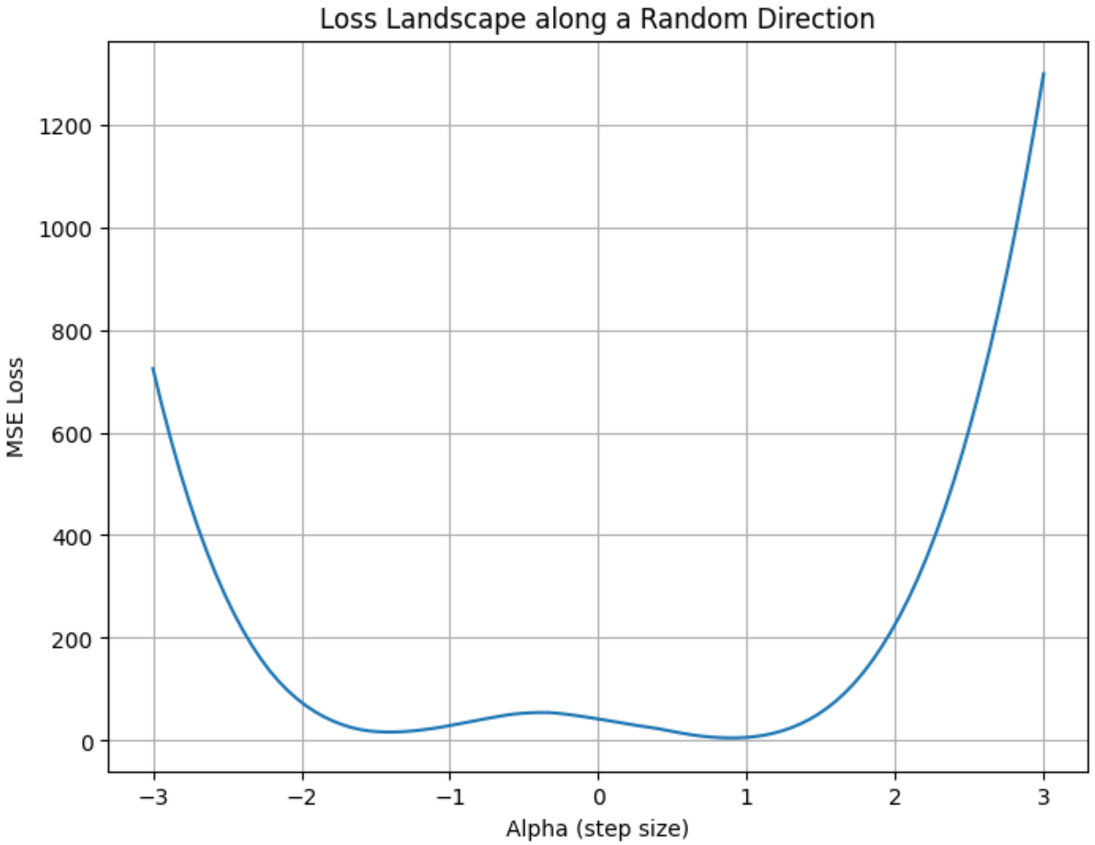
It Should Not Work

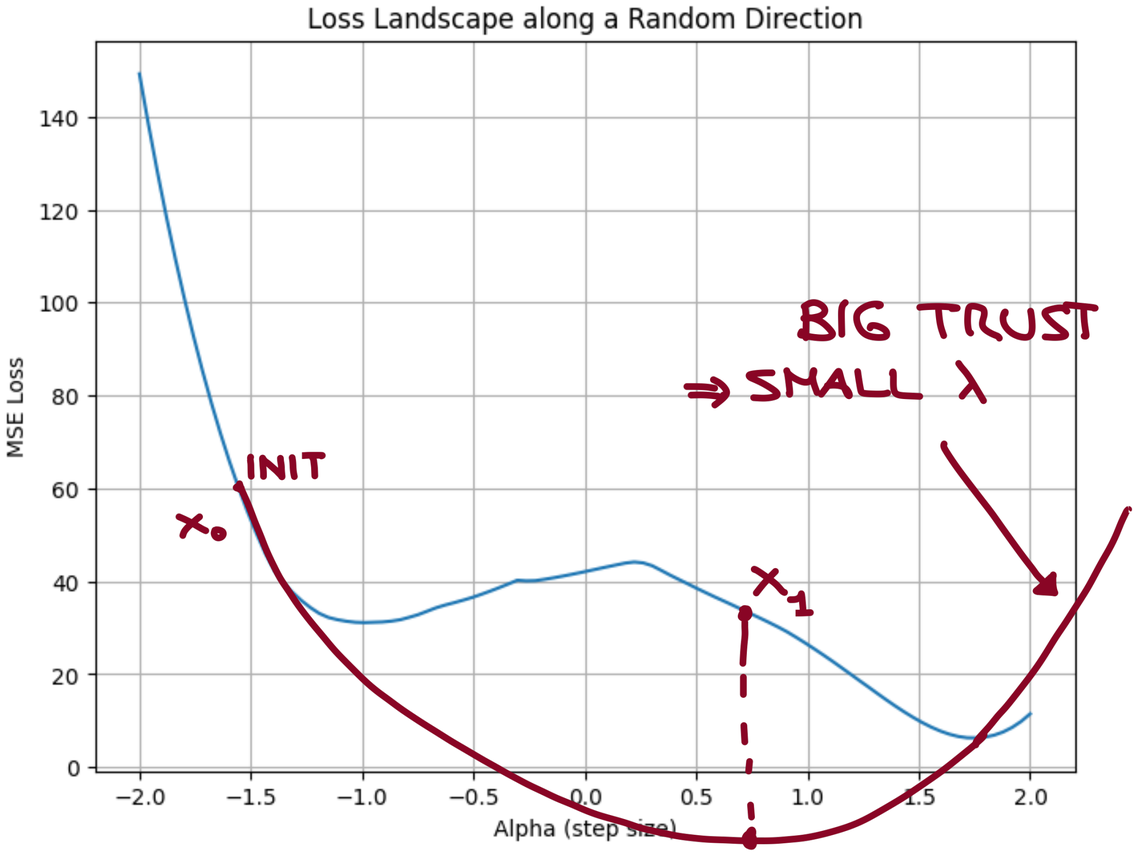




Convex Relaxation

It Should Not Work





Convex Relaxation

Questions

- How well can Gradient Based Methods do?

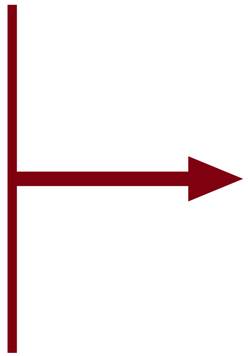
Very well on convex problems…

- How well can these *Iterative Convex Relaxations* do on non-convex problems?

Depends on the geometry of the problem…?

Questions

- How do they escape local minima?
- How do they escape saddles?
- How do they pick the minimum in the high dimensional manifold?



These two are about traveling flat areas!

All things they have *not* been designed for!

Part 2

The naive mathematical approach

Implicit Regularization

- Location of Convergence
  - What gradient methods were
  - Some NN examples
- **(Stable) Implicit Regularization**
- Edge of Stability

Speaking Math

Where do algorithms go?

This is not an *optimization* question:

It is a dynamical systems question!

Questions are:

- What are the basins of attraction? Do trajectory recur? **(Long-term behavior)**
- Is the invariant set stable **(Stability in the sense of Lyapunov, …)**
- How does the behavior changes with the (hyper)parameters **(Bifurcation Analysis)**

The Question

Where do algorithms go?

This is not an *optimization* question:

It is a dynamical systems question!

Previous approaches:
a la Numerical Analysis

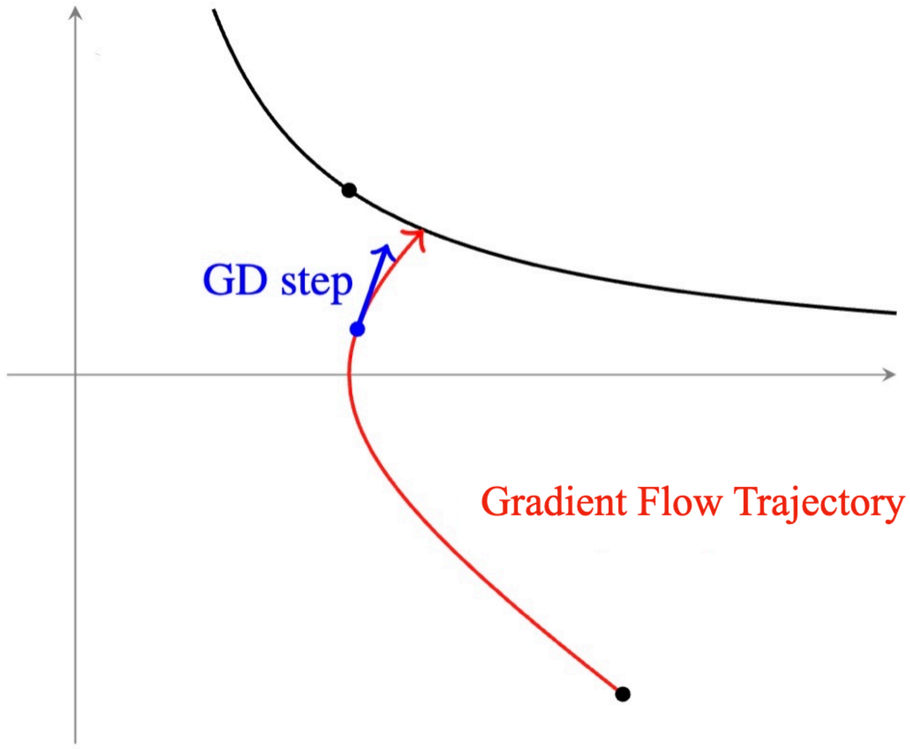
GOAL

What do we want to characterize mathematically

$$\{\theta \text{ s.t. } \nabla L(\theta) = 0\}$$

Not enough!

We want to follow the *trajectory* and check *where* we go



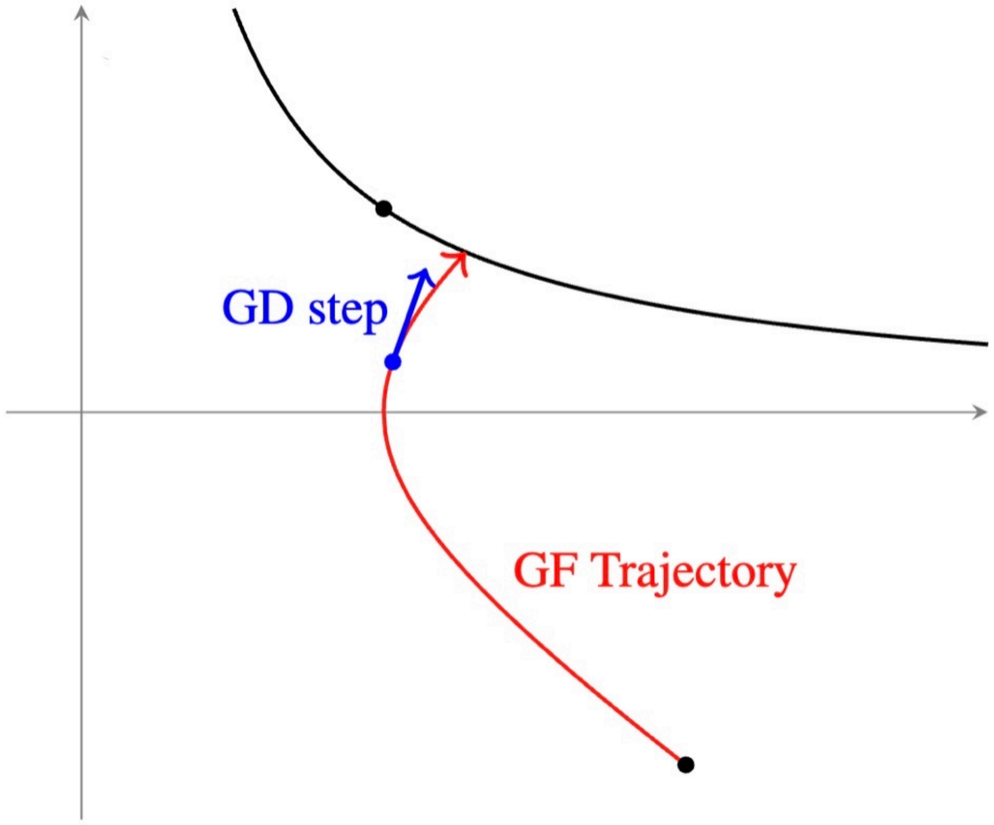
GOAL

What do we want to characterize mathematically

We want to follow the ***trajectory*** and check ***where*** we go

How?

- Take two trajectories
- Compute their difference
  - Expanding in Taylor?



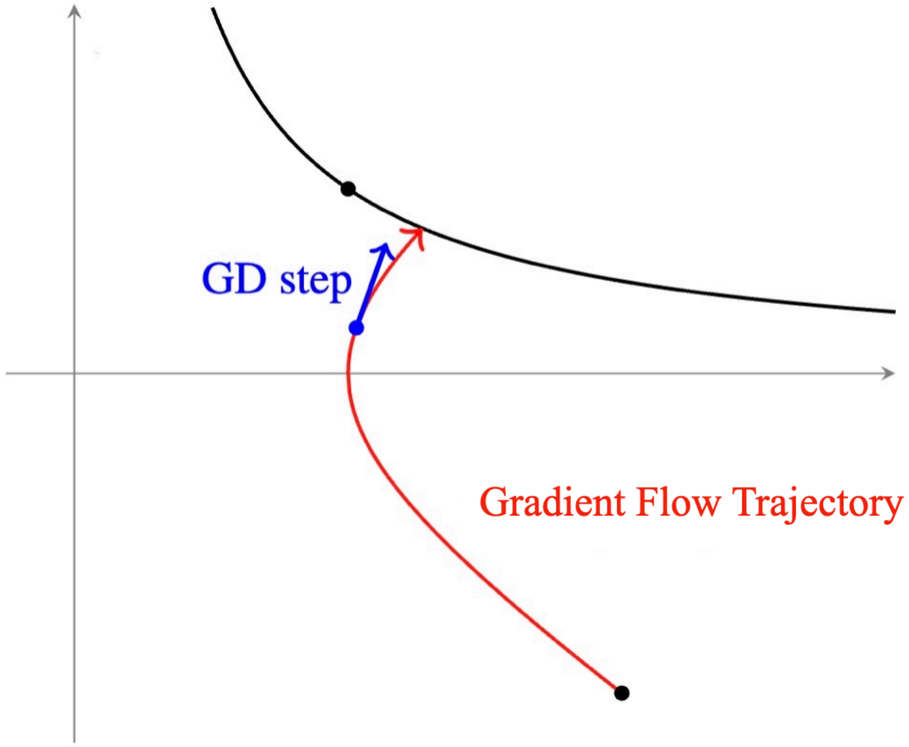
RESULTS

Keywords:
Backward Error Analysis
Implicit Regularization

Discretizing

$\Longrightarrow$

Implicitly adding a penalty!



SGD vs GD

Let us expand k steps of GD one epoch of SGD trajectories in Taylor and check the difference:

$$\theta_2^{GD} = \theta_0 - \eta \nabla L(\theta_0) - \eta \nabla L(\theta_1)$$

$$\theta_2^{GD} = \theta_0 - 2\eta \nabla L + \eta^2 \nabla^2 L \nabla L$$

k

k = EPOCH

$\binom{k}{2} \gamma^2$

Same for SGD with replacement or online.

Comparing SGDs

Let us expand k steps of SGD in Taylor at initialization:

$$\theta^{SGD}(\eta) = \theta - \eta \underbrace{\sum_{B_i \text{ batches}} \nabla L(B_i)}_{\eta k \nabla L} + \eta^2 \underbrace{\sum_{i<j} \nabla^2 L(B_i) \nabla L(B_j) + \dots}_{additional}$$

- Noisy centered for SGD with
- Deterministic for SGD without

- Noisy centered for SGD with
- Biased for SGD without: batches are dependent

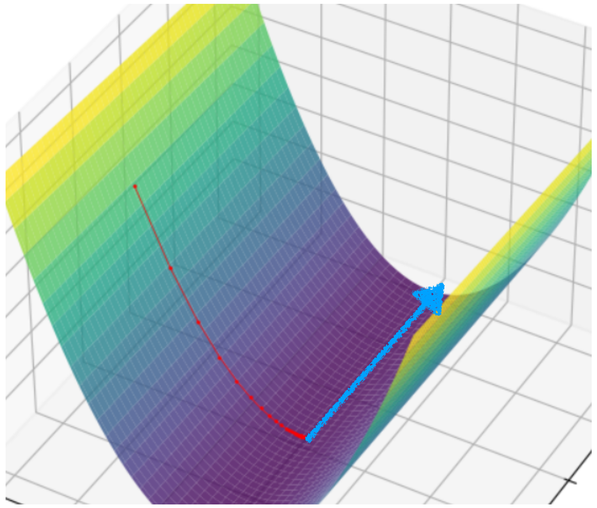
(Smith et al., 2021 and Beneventano, 2023)

SGD Without Replacement

What the behavior is

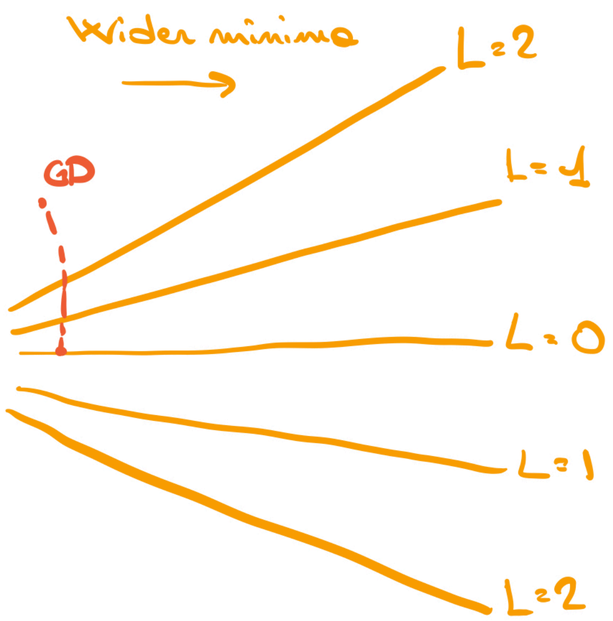
$$SGD = GD + \textbf{step on } Regularizer$$

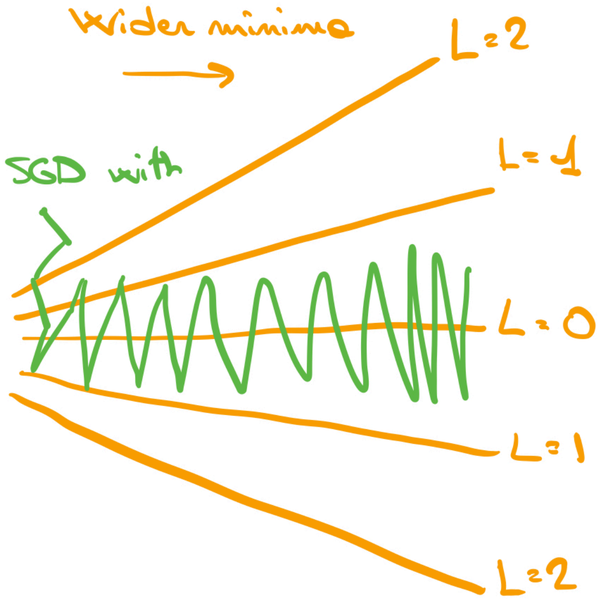
- Flat directions: ***Regularizer***
- Sharp directions: ***Follows SGD with***
- More sharp directions: ***Shoots away***

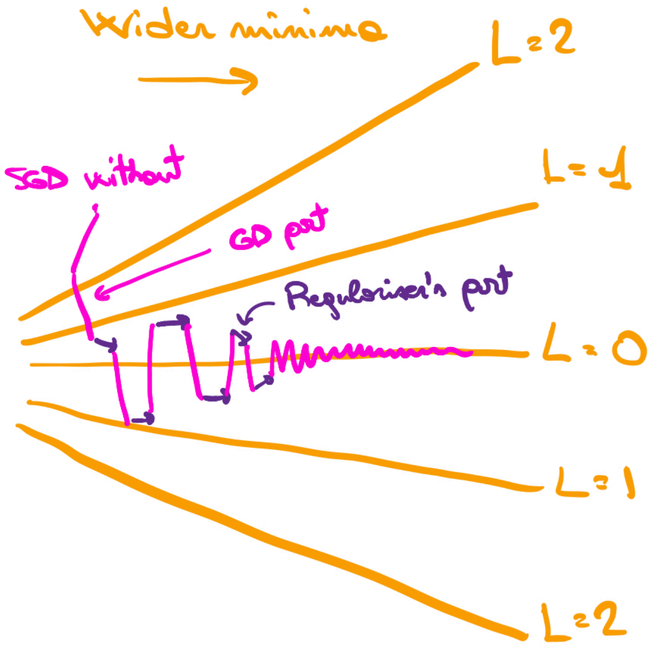


With vs *Without*

The Bias







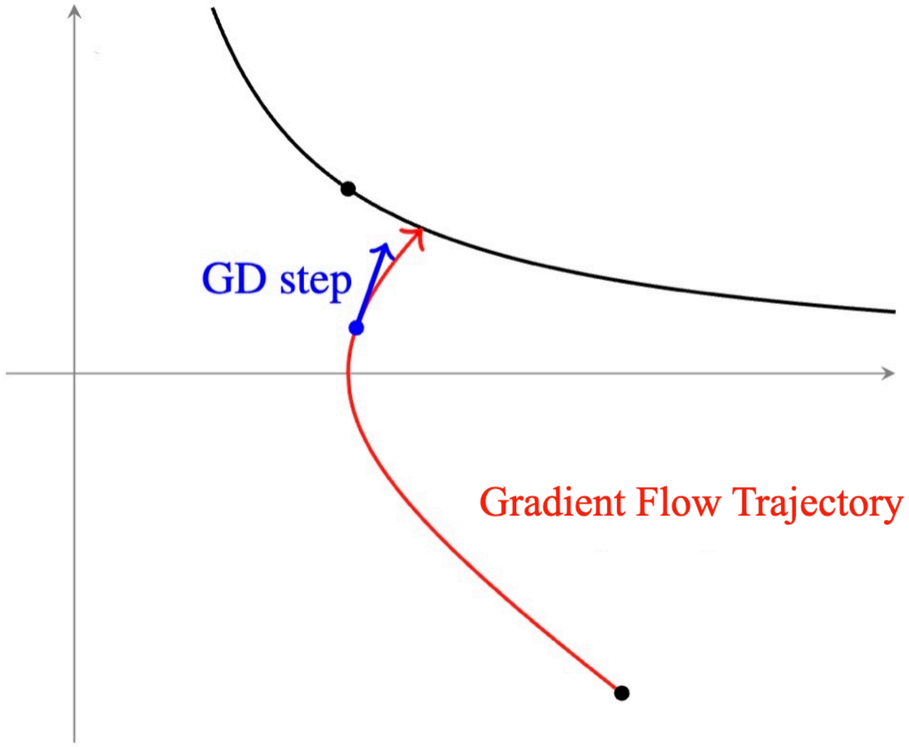
RESULTS

Keywords:
Backward Error Analysis
Implicit Regularization

Discretizing

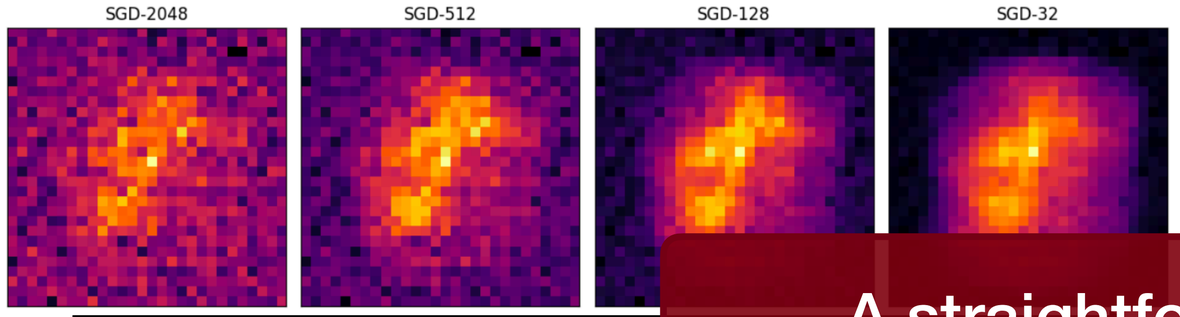
$\Longrightarrow$

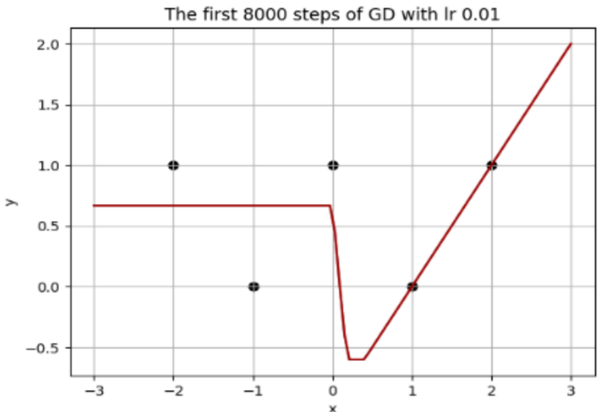
Implicitly adding a penalty!

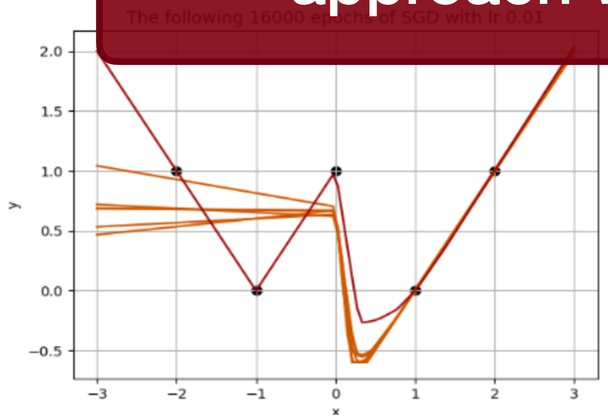


This Approach

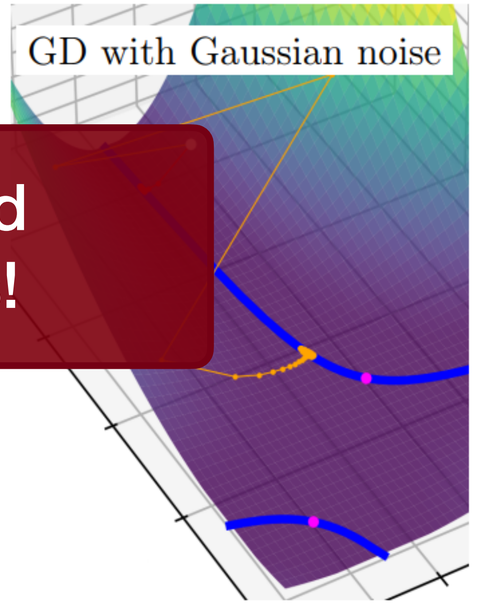
Works in the cases above

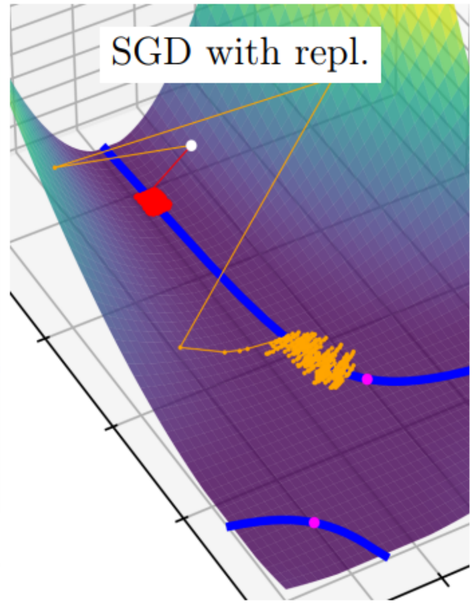






A straightforward approach works!





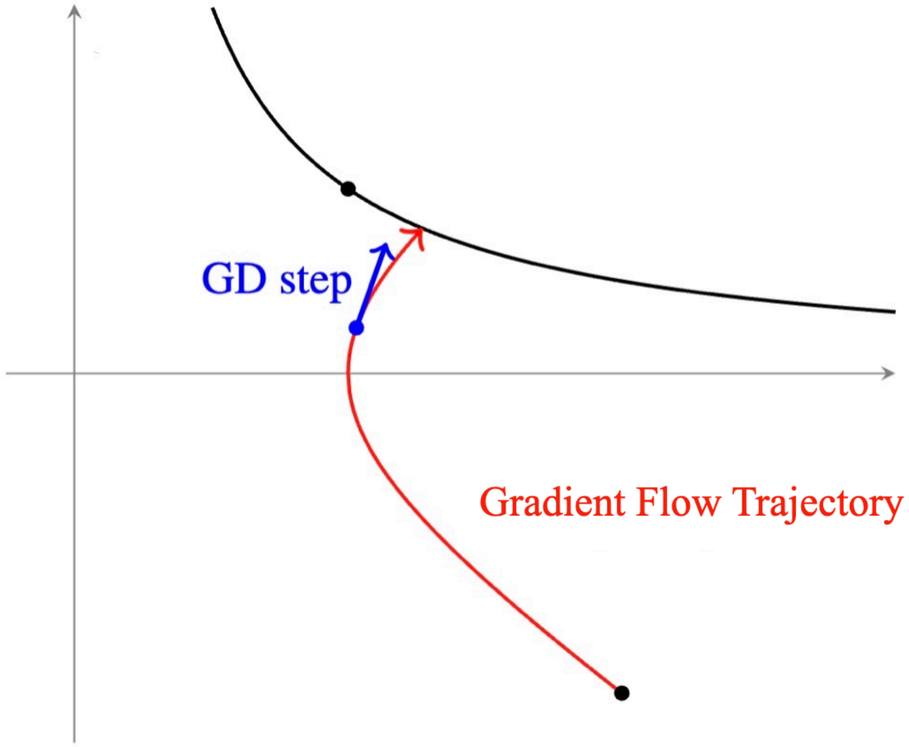
Part 3

When does it break?
To what extent does it explain?
To what extent can we use it to do the next step: optimality?

LIMITATIONS

Main problems

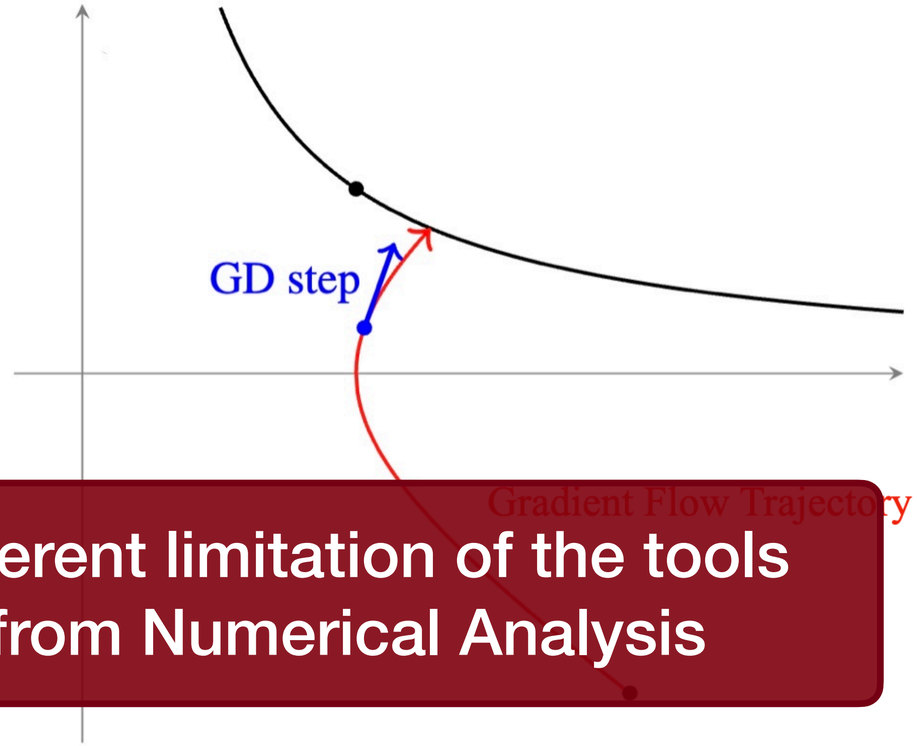
- It works for very small step sizes;
- It may characterize just a few steps;
- It is too complicated to plug in the actual losses and say something sensible;



LIMITATIONS

Main problems

- It works for very small step sizes;
- It may characterize just a few steps;
- It is too complicated to plug in the actual losses and say something sensible;

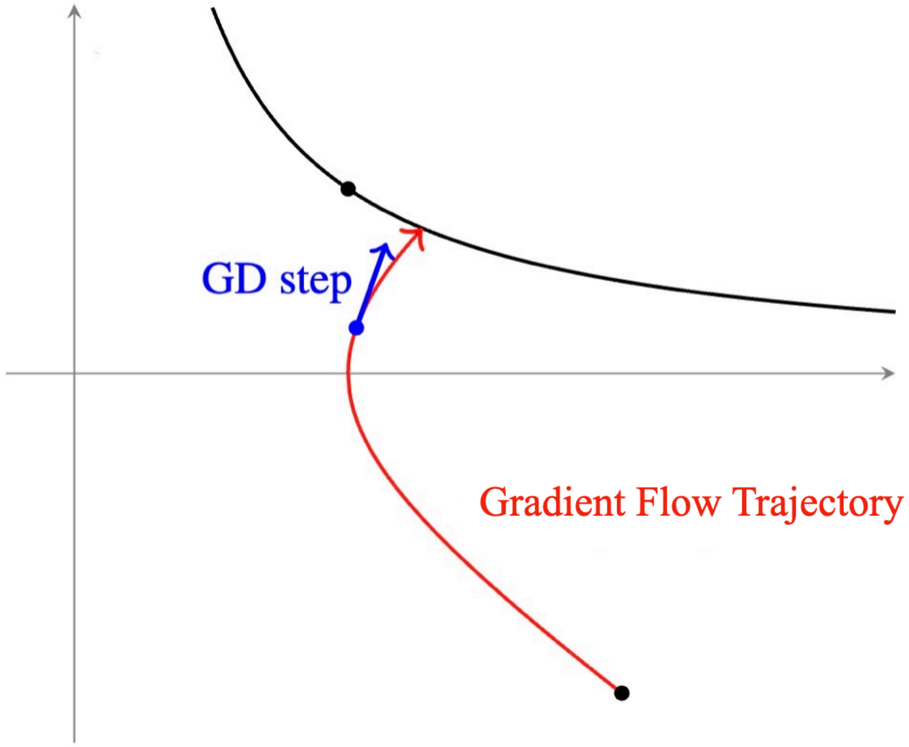


Inherent limitation of the tools from Numerical Analysis

New Set of Questions

- It works for very small step sizes.

- How large are the steps in practice?
- What does it even mean for the step to be large?



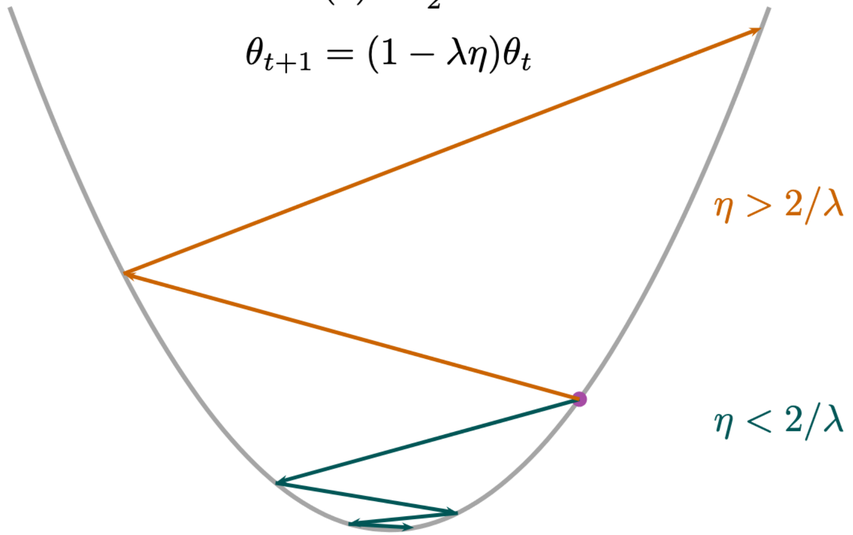
INFINITESIMAL *VS* LOCAL

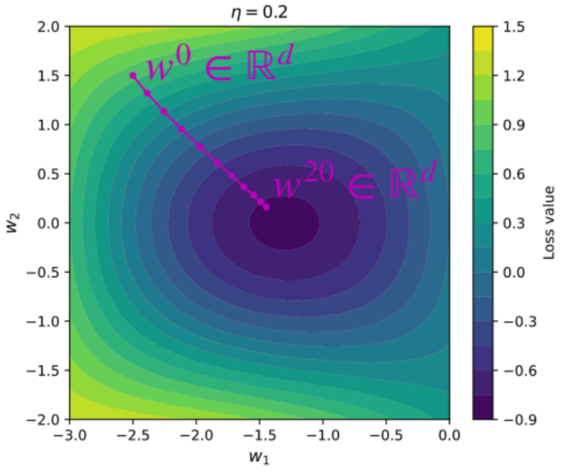
- How much the *direction* of the gradient *changes*.
- How big it needs to be before it *overshoots*.

2^{nd} **Derivative** *(in the neighborhood)*

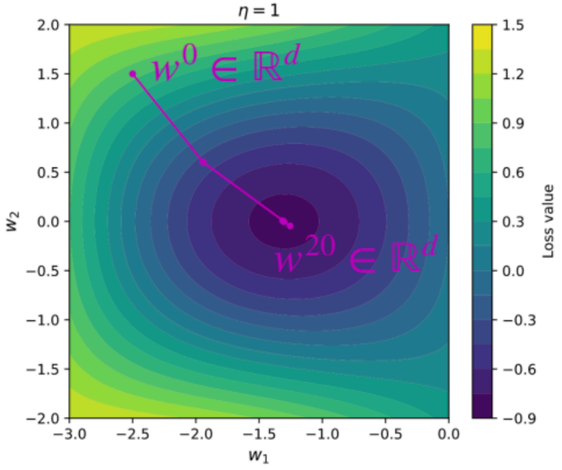
$$L(\theta) = \frac{\lambda}{2}\theta^2$$

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

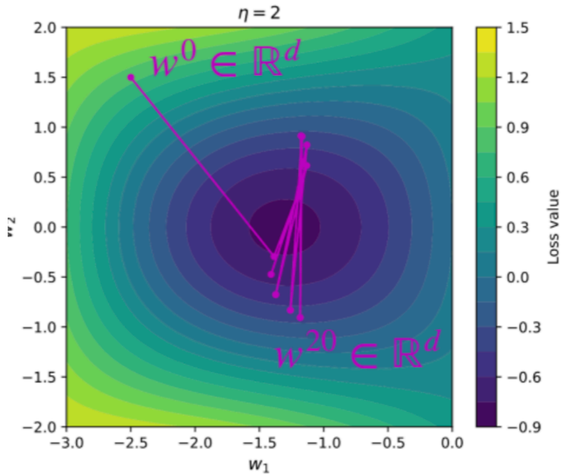




Low η



High η



Too high η

Edge of Stability

- Location of Convergence
  - What gradient methods were
  - Some NN examples
- (Stable) Implicit Regularization
- **Edge of Stability**

Surprising Answer for GD

For Full-Batch GD these Questions have been answered:

Location of convergence becomes:

How does location of convergence depend on the hyper parameters?

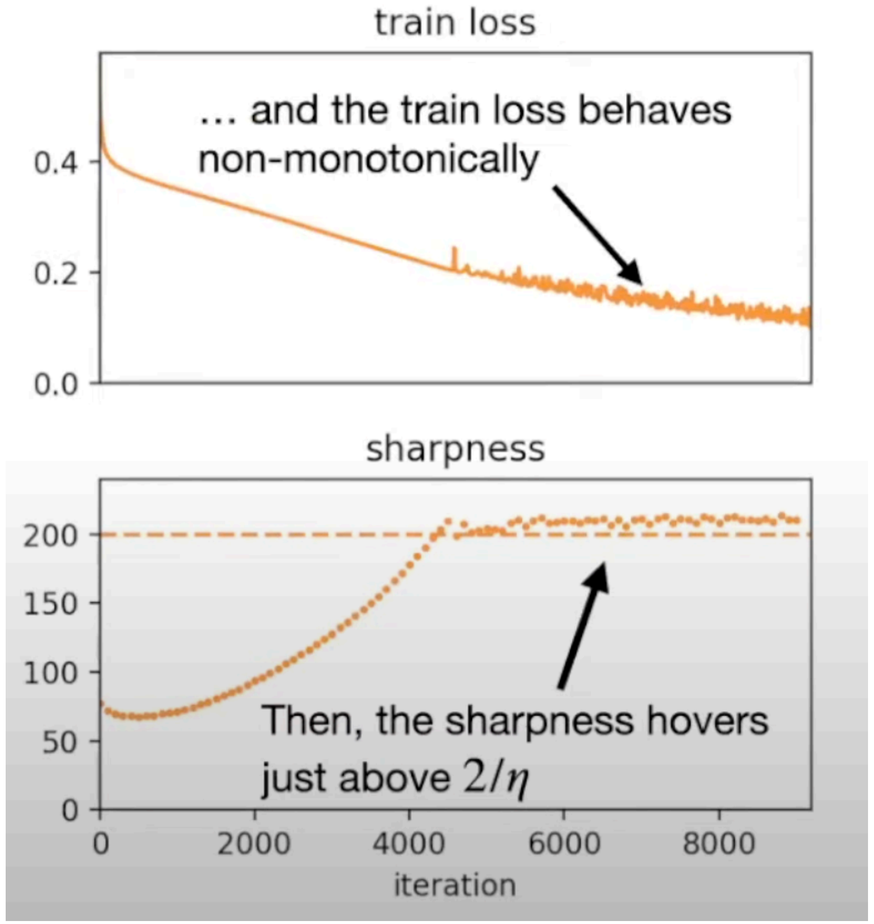
We can answer given this:

Is the step size generally big or small?

The Edge of Stability

What:

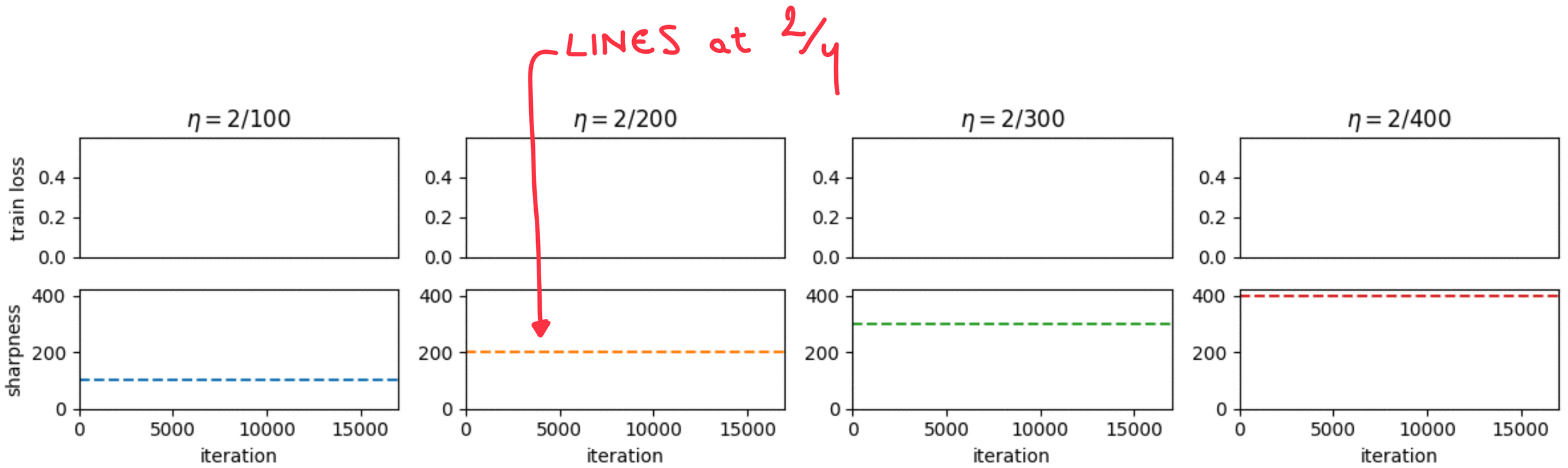
- A *Classification* task.
- Across many different architectures the behavior obeys *a simple rule.*
- The rule is not what you expect.



Plot from the talk above of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

The Edge of Stability

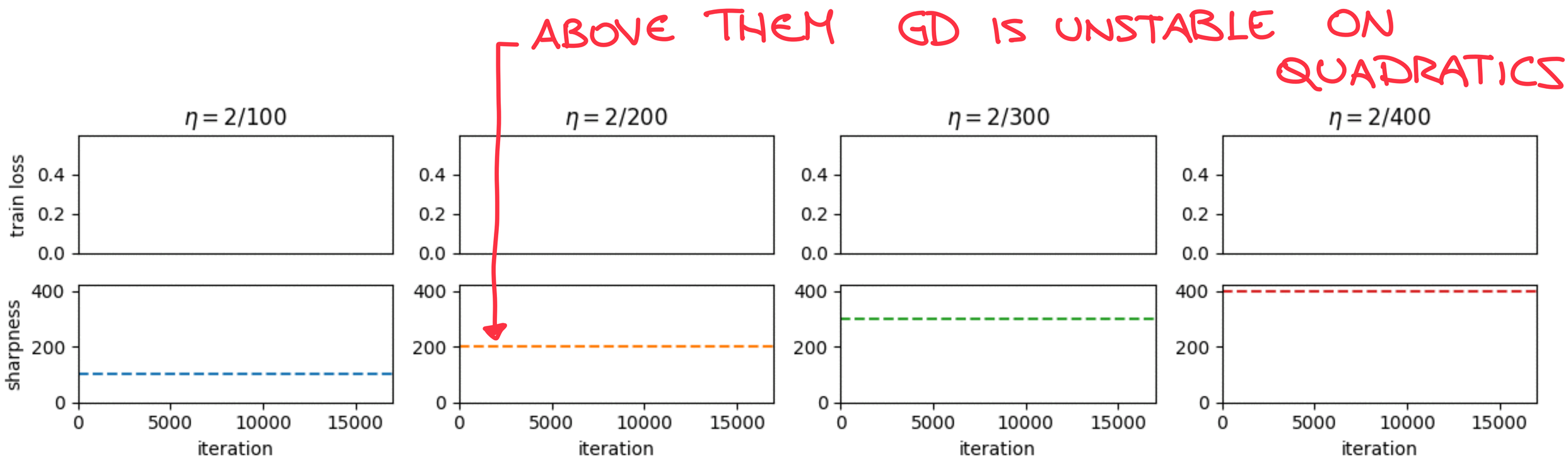
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

The Edge of Stability

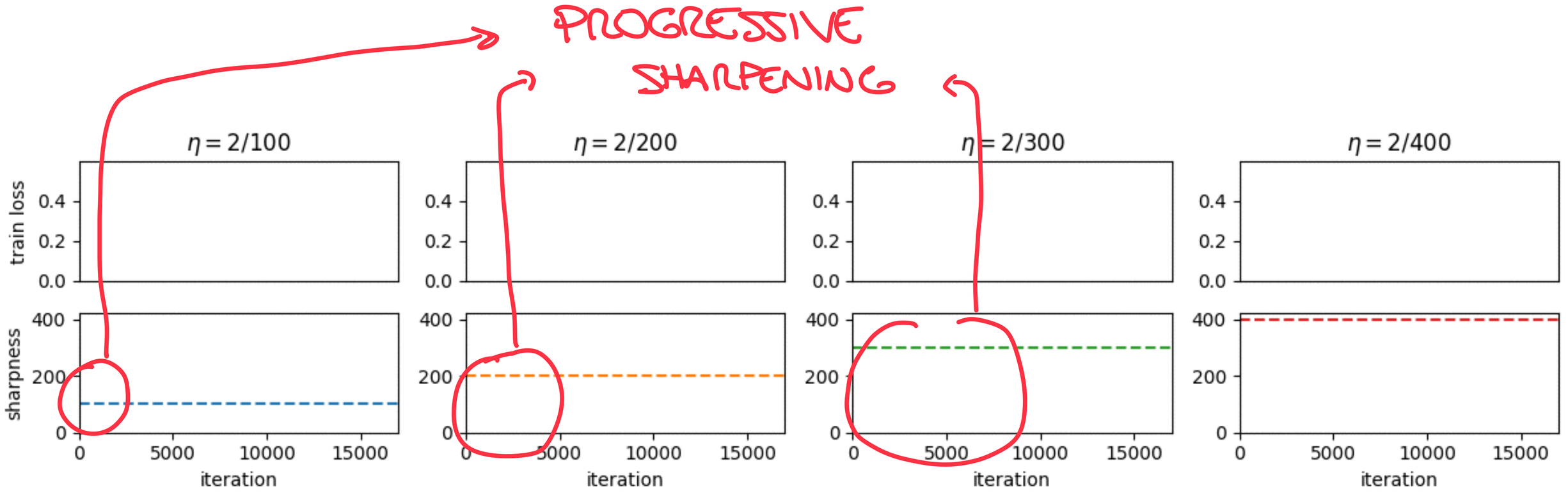
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

The Edge of Stability

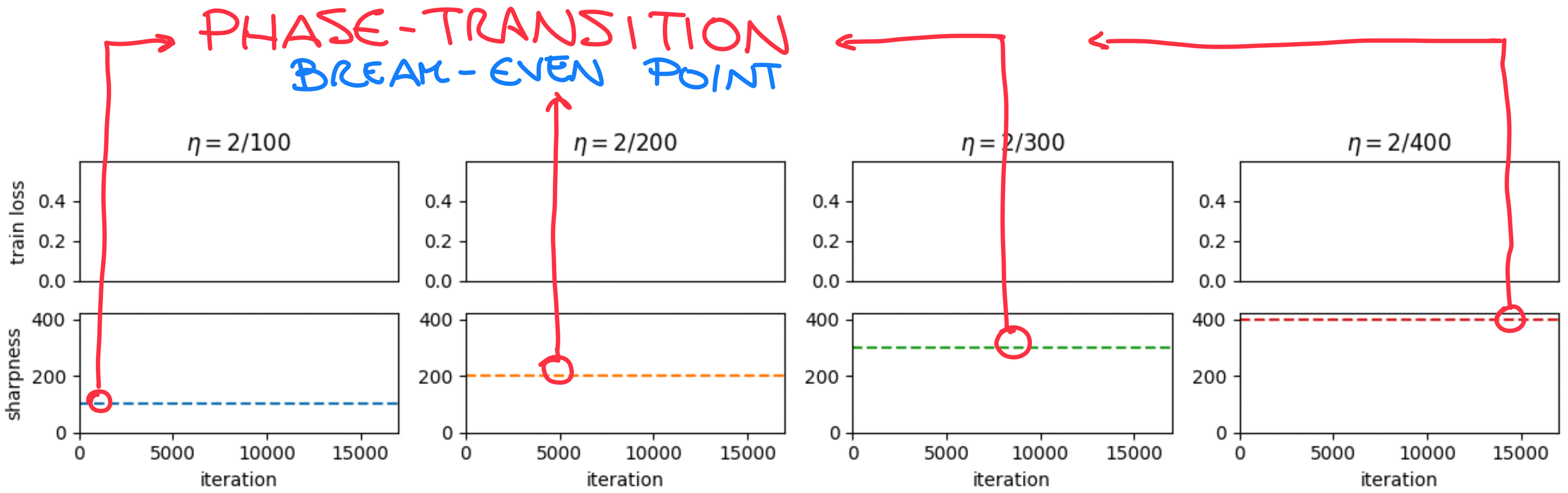
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

The Edge of Stability

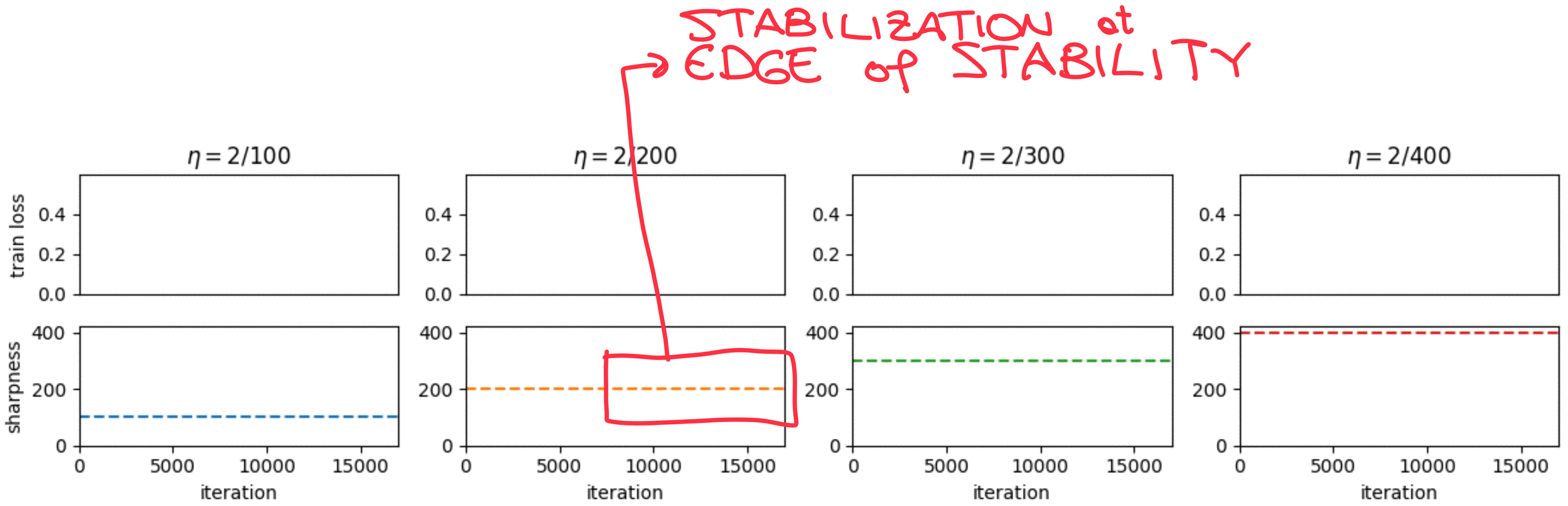
Take the *largest eigenvalue of the Hessian*, call it sharpness, plot it.



Gif from the website of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

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: https://arxiv.org/pdf/2103.00065.

New Set of Questions

For Full-Batch GD all these Questions have been answered:

Location of convergence becomes:

How does location of convergence depend on the hyper parameters?

✓

We can answer given this:

Is the step size generally big or small?

✓

Answered!

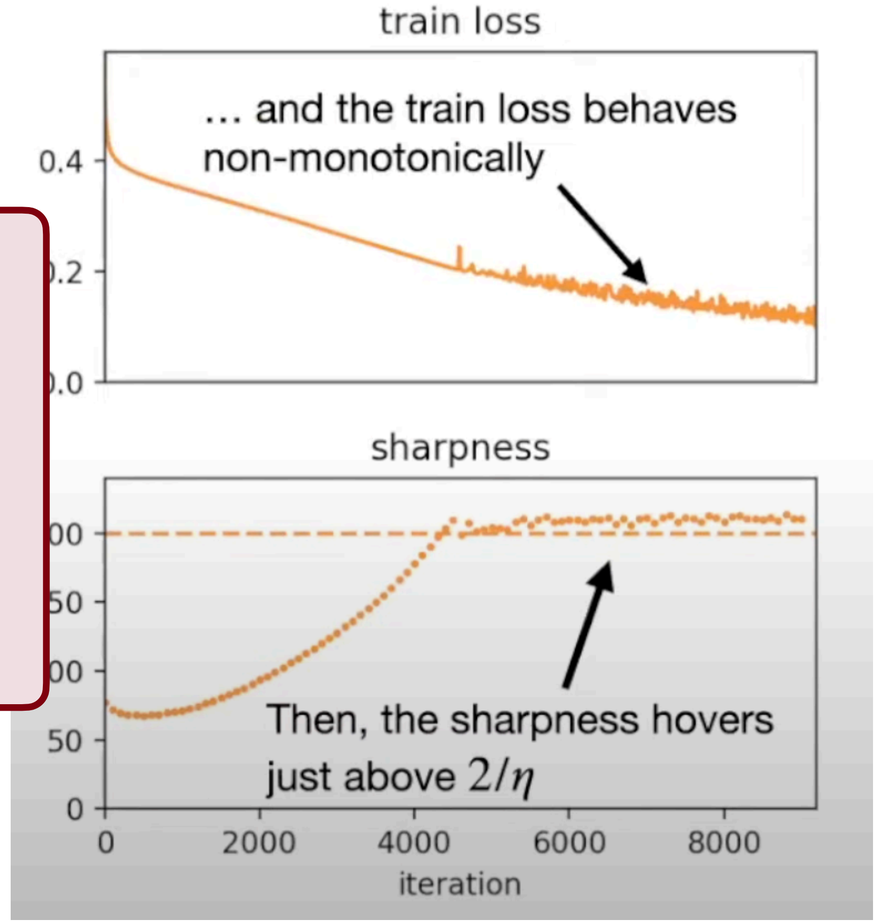
It's just a little heartbreaking that what we did seems useless

The Edge of Stability

Implications:

- Conventional wisdom about step size is wrong — *set $\eta \leq 1/L$ is an impo*
- You have to go deeper than t Taylor approximation when d theory — *it would diverge…*
- GD does not decrease monot loss, *gradient size increases* *Still,* GD is successful.

Not only, same for most the convex optimization



Plot from the talk above of Jeremy Cohen — Cohen et al. (2021), ArXiv: https://arxiv.org/pdf/2103.00065.

The Edge of Stability

Is this the case of SGD?

The Edge of Stability

How about mini-batch SGD?

1000s of works about:
"SGD finding flatter minima"

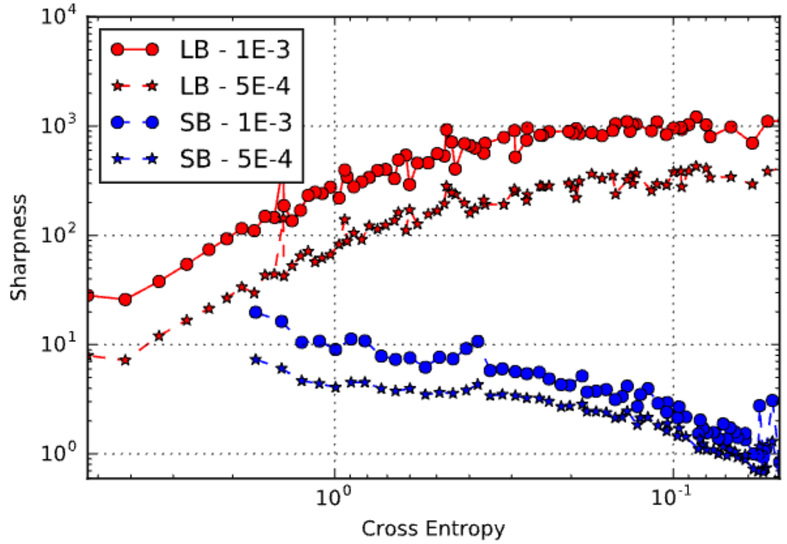
$\lambda_{\max}(H) \leq 2/\eta$ **for SGD, usually!**

SGD $\Longrightarrow$ SMALLER λ_{max}

KESKAR ET AL. (2016) STARTED THE FIELD

- Smaller batch $\Longrightarrow$ Smaller Hessian.
- Smaller batch $\Longrightarrow$ Better Generalization.

- Location Acclimates to Hyperparameters.
- Observed Progressive Sharpening.

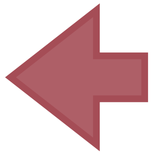


SB	LB
1.23 ± 0.83	205.14 ± 69.52
1.39 ± 0.02	310.64 ± 38.46
28.58 ± 3.13	707.23 ± 43.04

A New Quantity

$$\textit{Batch Sharpness} := \mathbb{E}_{\mathbf{B}}\left[\frac{\nabla L_{\mathbf{B}}^{\top}\mathbf{H}_{\mathbf{B}}\nabla L_{\mathbf{B}}}{\|\nabla L_{\mathbf{B}}\|_2^2}\right]$$

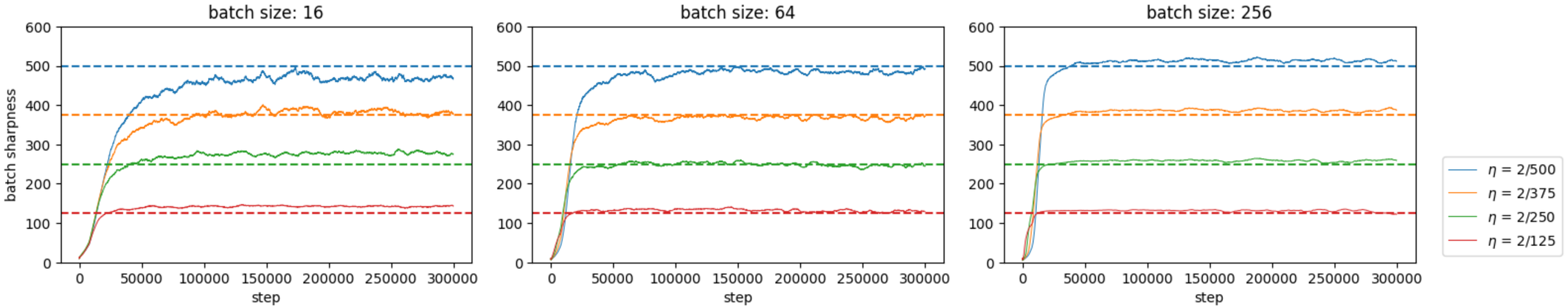
$$\frac{\nabla L_{\mathbf{B}}^{\top}\mathbf{H}_{\mathbf{B}}\nabla L_{\mathbf{B}}}{\|\nabla L_{\mathbf{B}}\|_2^2]} \leq \frac{2}{\eta}$$



This is the EoS threshold *if* I were to do full-batch GD on the dataset B!

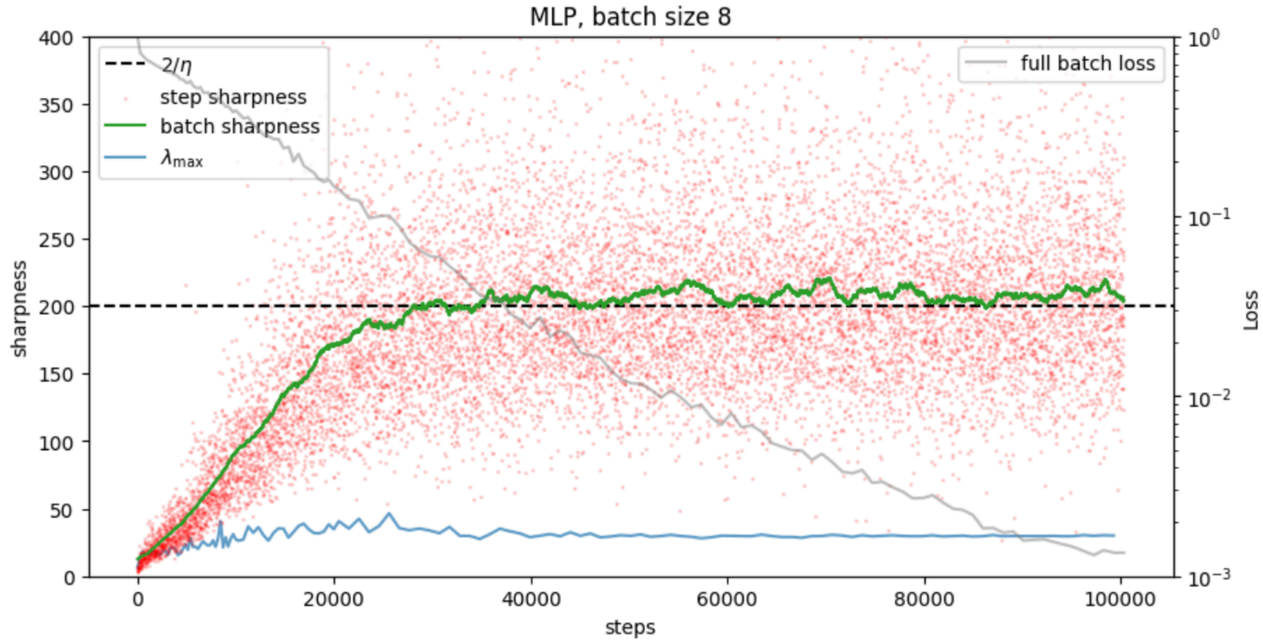
EDGE OF STOCHASTIC STABILITY

$$\textit{Batch Sharpness} := \mathbb{E}_{\mathbf{B}}\left[\frac{\nabla L_{\mathbf{B}}^{\top}\mathbf{H}_{\mathbf{B}}\nabla L_{\mathbf{B}}}{\|\nabla L_{\mathbf{B}}\|_2^2}\right]$$



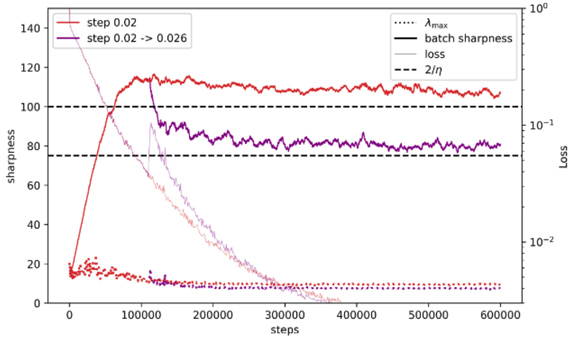
EDGE OF STOCHASTIC STABILITY

- *Batch Sharpness* stabilizes at $2/\eta$ after a long progressive sharpening.

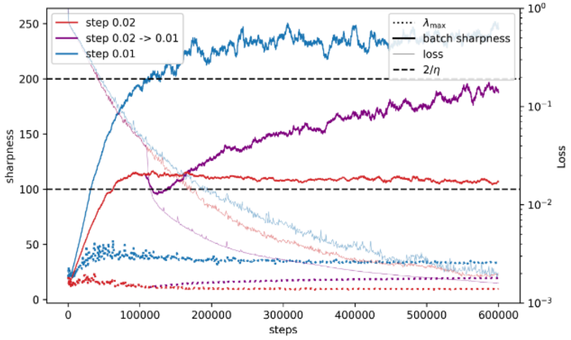


EDGE OF STOCHASTIC STABILITY

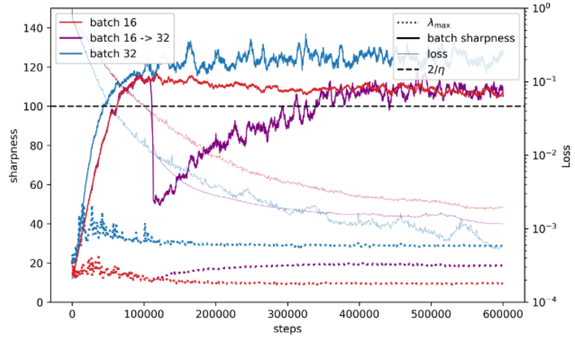
$$\textit{Batch Sharpness} := \mathbb{E}_{\mathbf{B}}\left[\frac{\nabla L_{\mathbf{B}}^{\top}\mathbf{H}_{\mathbf{B}}\nabla L_{\mathbf{B}}}{\|\nabla L_{\mathbf{B}}\|_2^2}\right]$$



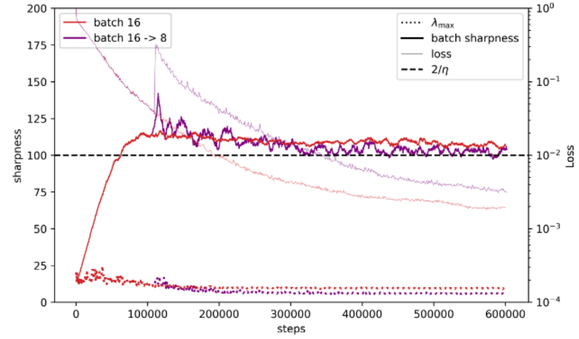
(a) Increasing step size



(b) Decreasing step size



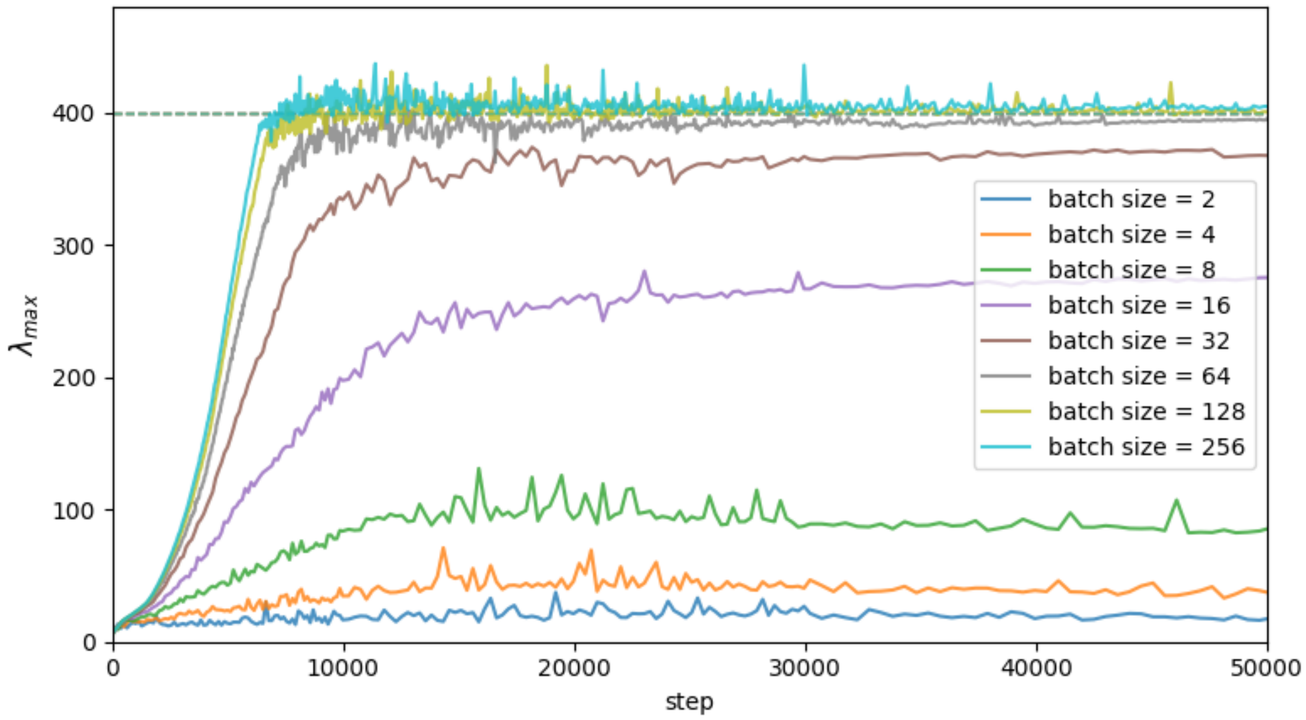
(c) Increasing batch size



(d) Decreasing batch size

EDGE OF STOCHASTIC STABILITY

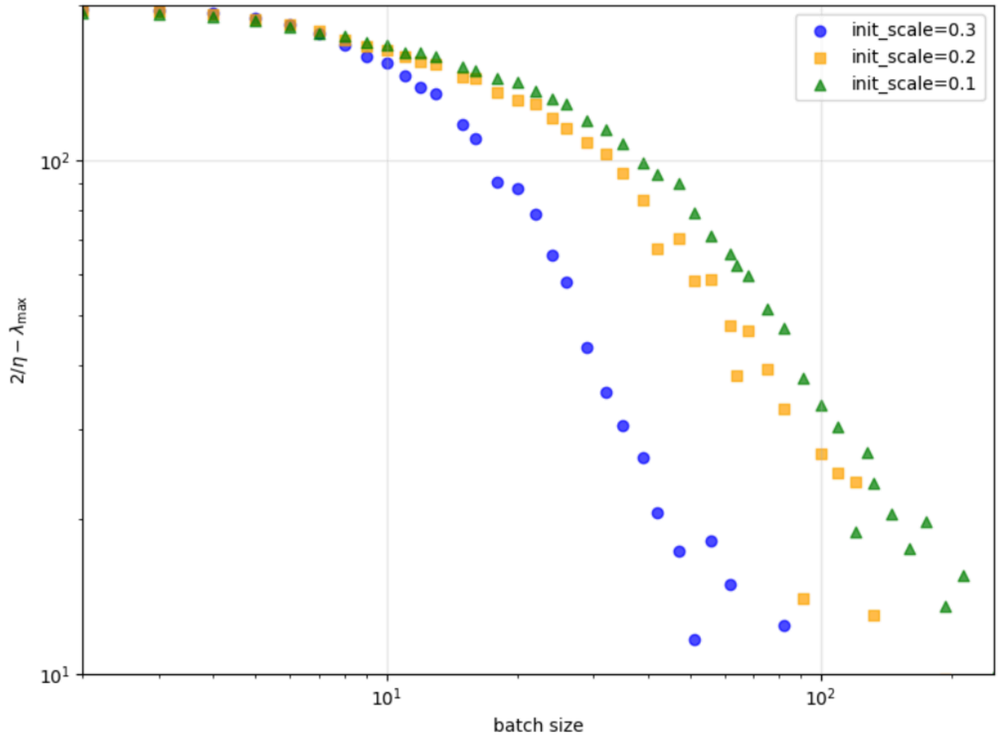
- *Batch Sharpness* stabilizes at $2/\eta$ after a long progressive sharpening.
- Also sharpness $\lambda_{\max} H$ stabilizes *(usually below —lower for smaller BS)*.



EDGE OF STOCHASTIC STABILITY

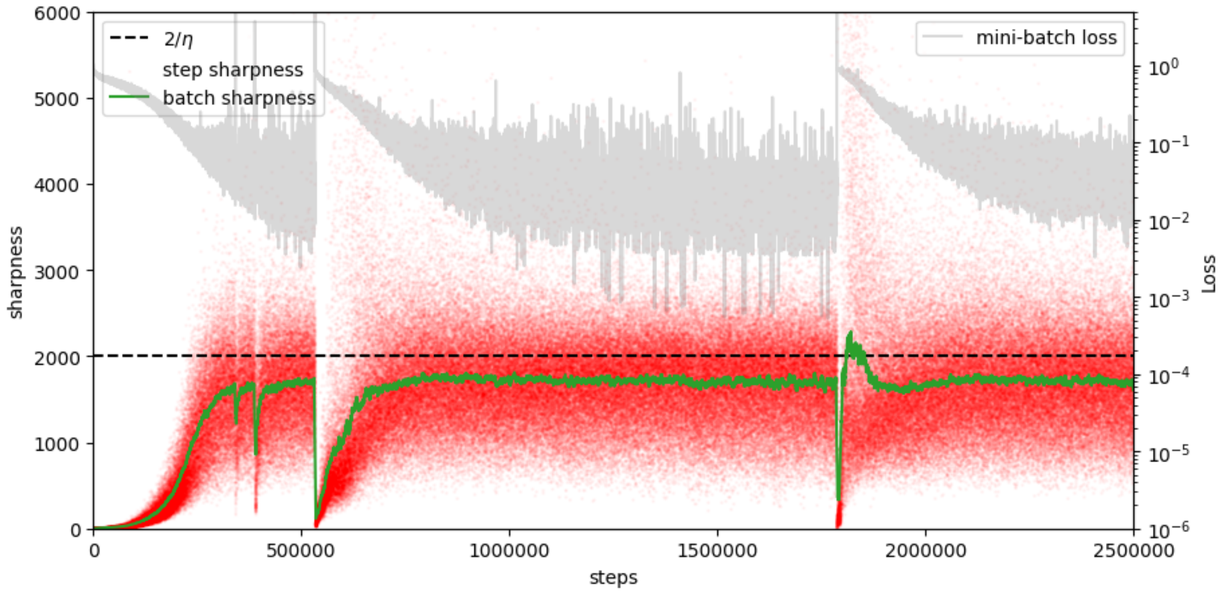
BATCH SHARPNESS, NOT $\lambda_{\max}$!

- ***Batch Sharpness*** **going to $2/\eta$ drives the dynamics**
- **$\lambda_{\max}$ follows its movements**
- **Acts inconsistently**



EDGE OF STOCHASTIC STABILITY

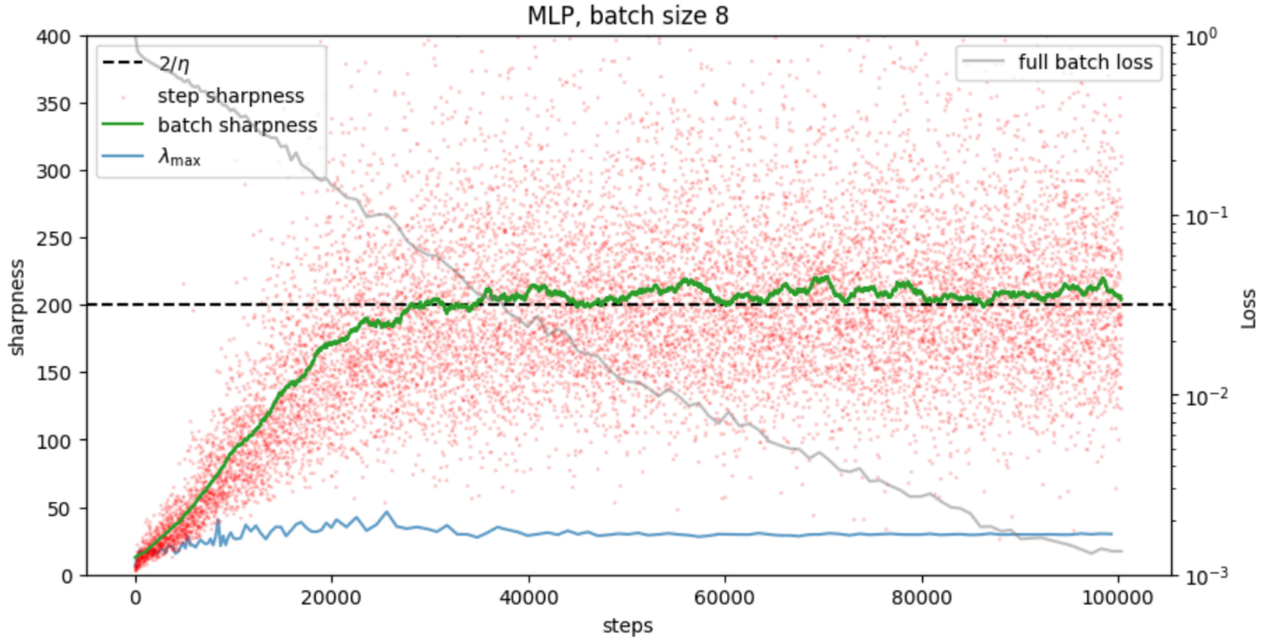
- *Batch Sharpness* stabilizes at $2/\eta$ after a long progressive sharpening.
- Also sharpness $\lambda_{\max} H$ stabilizes.
- The trajectories are subject to catapults when outliers hit!



Implication 1

The ones of EoS translate here:

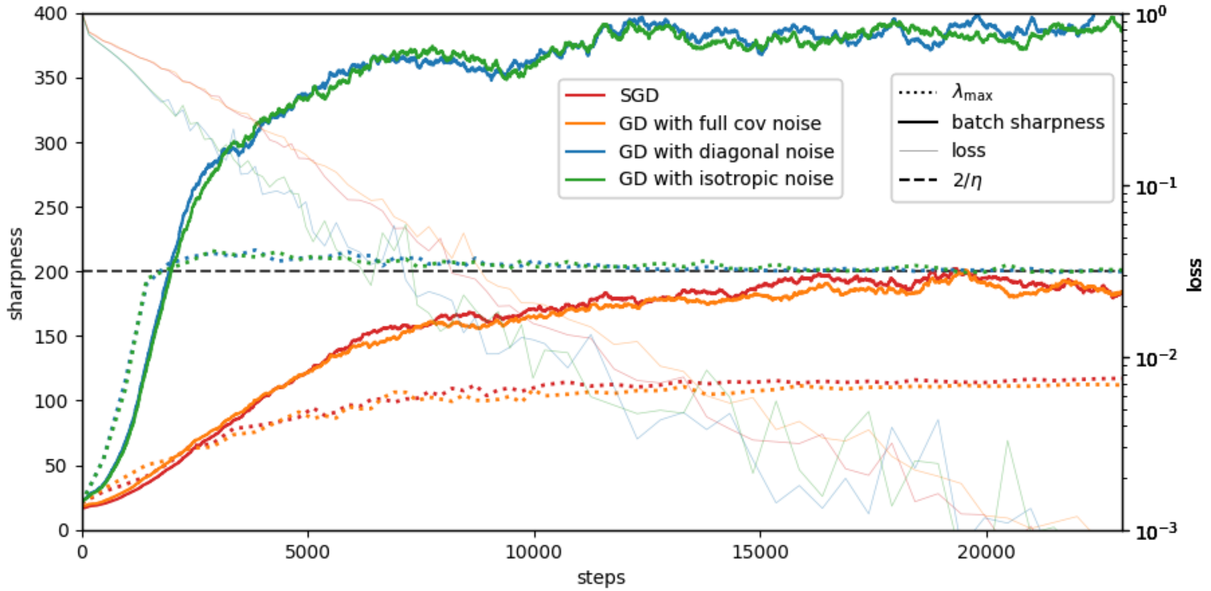
- ✓ Conventional wisdom about step size is wrong —*set $\eta \leq 2/L + \ldots$ is an impossible task!*
- ✓ You have to go deeper than the order-2 Taylor approximation when doing NN theory —*it would diverge…*
- ✓ GD does not decrease monotonically the loss, *gradient size increases.* *Still,* GD is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



IMPLICATION

RANDOM HESSIANS

- **Naive SDE modeling of SGD**
  full-batch Hessian
- **Noise-injected SGD**
  full-batch Hessian
- **Mini-Batch SGD**
  distribution of the Hessians

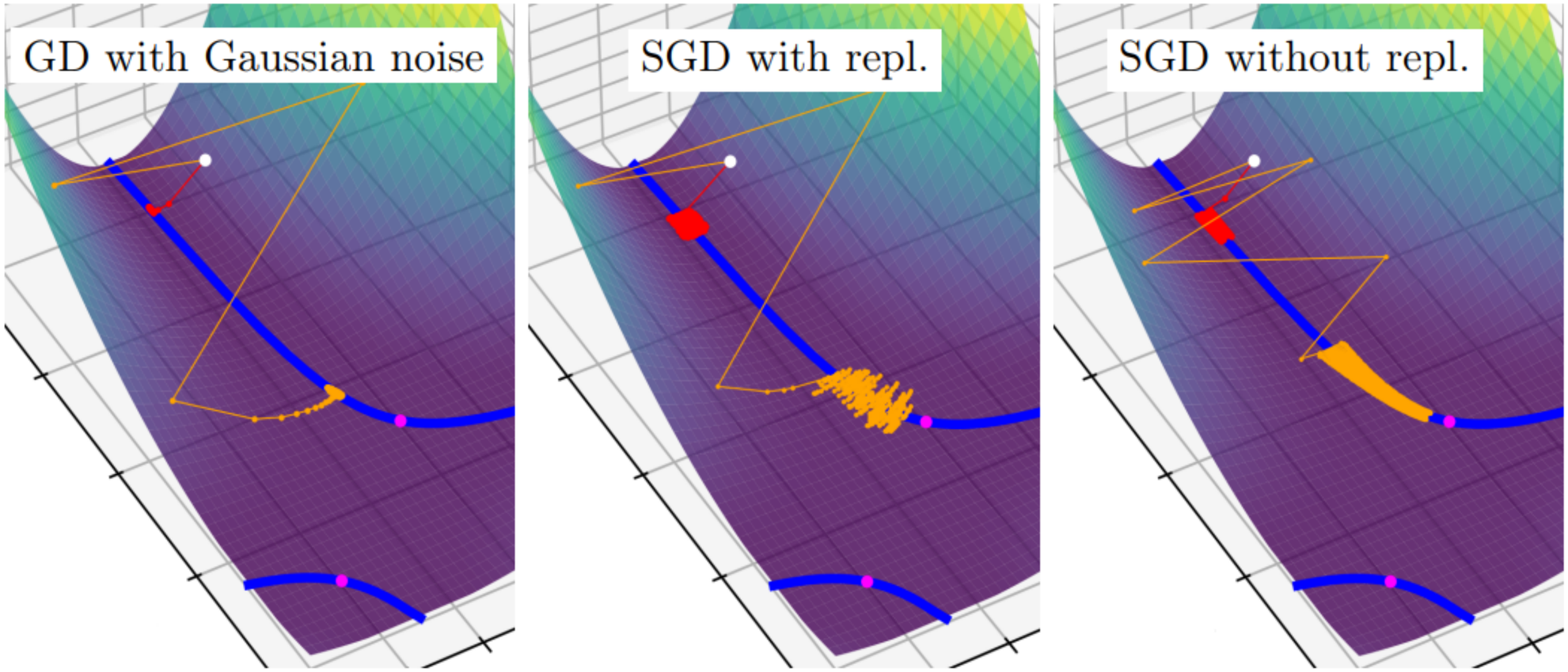


Training Networks

What qualitatively matters empirically:

- Deterministic vs Stochastic
- What kind of stochasticity

Comparing SGDs



$$\frac{1}{2n}\sum_{i=1}^{n}(\theta_1 \cdot \theta_2 \cdot x_i - y_i)^2$$ with dataset $\{(x_i, y_i)\}_{i=1}^{n} \subseteq \mathbb{R} \times \mathbb{R}$ and parameters $(\theta_1, \theta_2) \in \mathbb{R}^2$

CONCLUDING

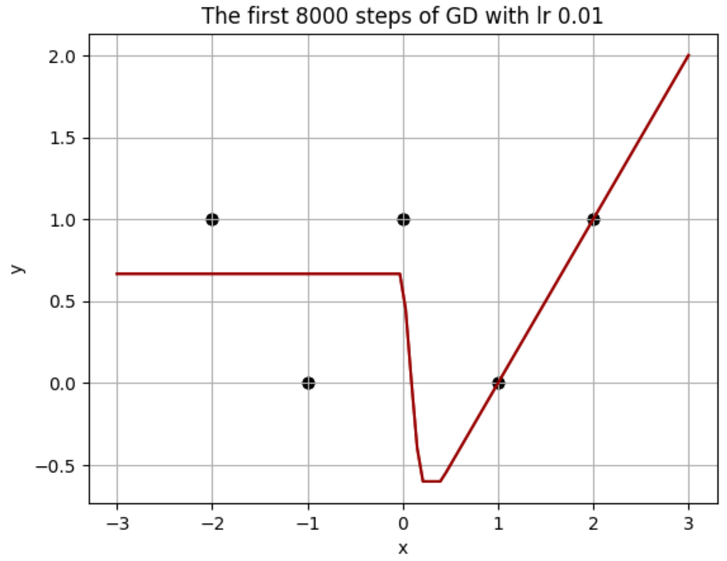
Optimizing *vs* Location

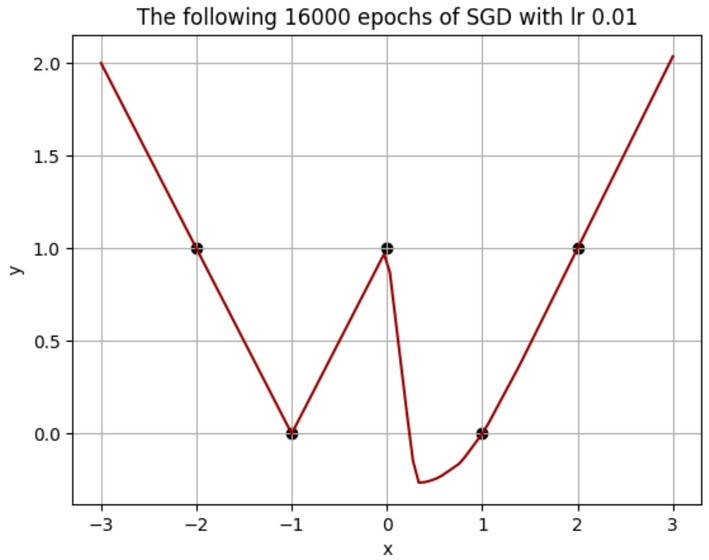
Goal of training neural networks:

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**





The Question

~~How to go optimally from *local* to *global*~~

Where does our training algorithm go in the case of neural networks?

A preliminary

Questions

- We would like to design a globally optimal gradient based optimizer.
- We will see that they have been designed for something else:

Convex functions

- We do not have the tools to discuss which one is local to global optimal, yet.
- So we will just explore:

Where they go in NN

What tools can we figure out for the problem

Speaking Math

Where do algorithms go?

This is not an *optimization* question:

It is a dynamical systems question!

Questions are:

- What are the basins of attraction? Do trajectory recur? **(Long-term behavior)**
- Is the invariant set stable **(Stability in the sense of Lyapunov, …)**
- How does the behavior changes with the (hyper)parameters **(Bifurcation Analysis)**

Message 3, Class 17

- We want to pick the architecture and the initialization to enhance trainability.
- Not really for purposes of approximation or location of convergence!

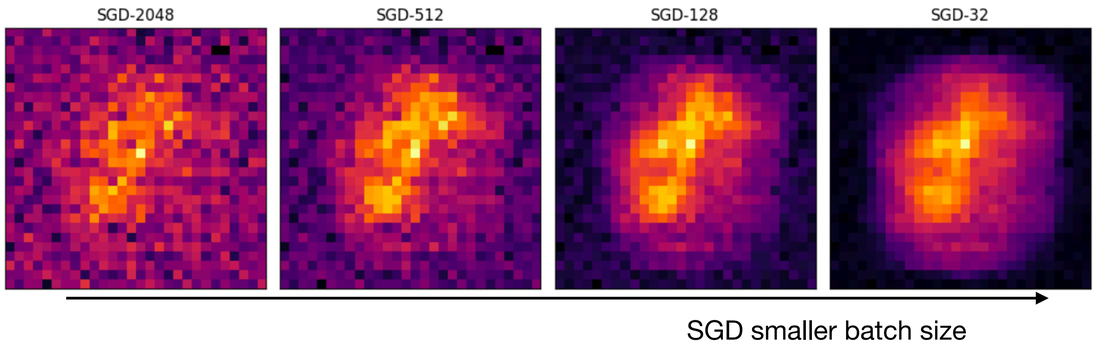
So let's study the *optimizer* instead of the initialization

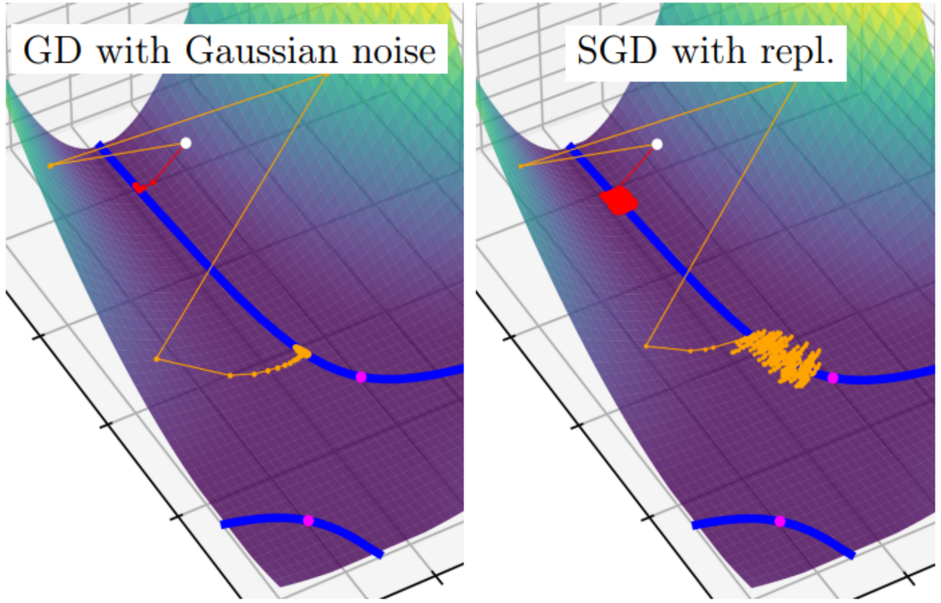
The Motivation

Goal is having best "performance"

There are many different minima ***with different performance***

Different algorithms find different minima, *consistently*





Training Networks

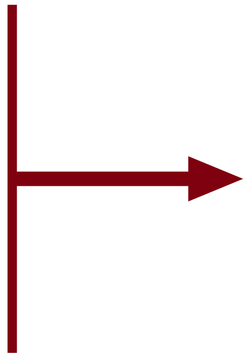
What qualitatively matters empirically *(for performance in Regime 2)*:

- Deterministic vs Stochastic
- What kind of stochasticity
- The choice of hyperparameters

Consistently

Questions

- How do they escape local minima?
- How do they escape saddles?
- How do they pick the minimum in the high dimensional manifold?



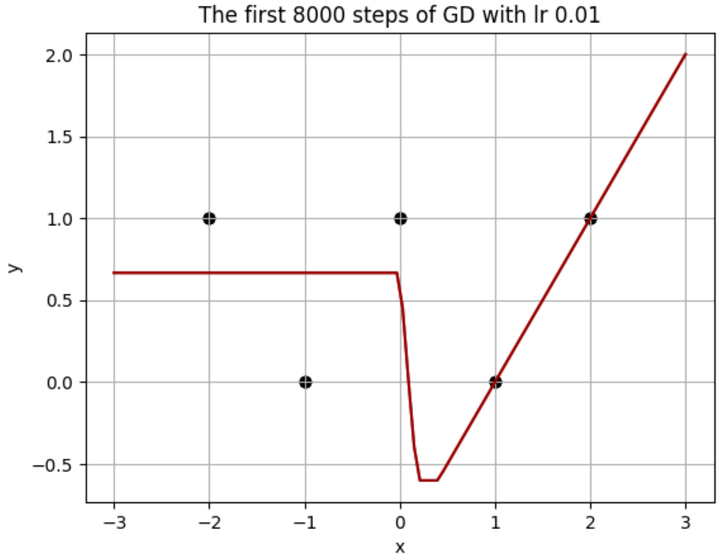
These two are about traveling flat areas!

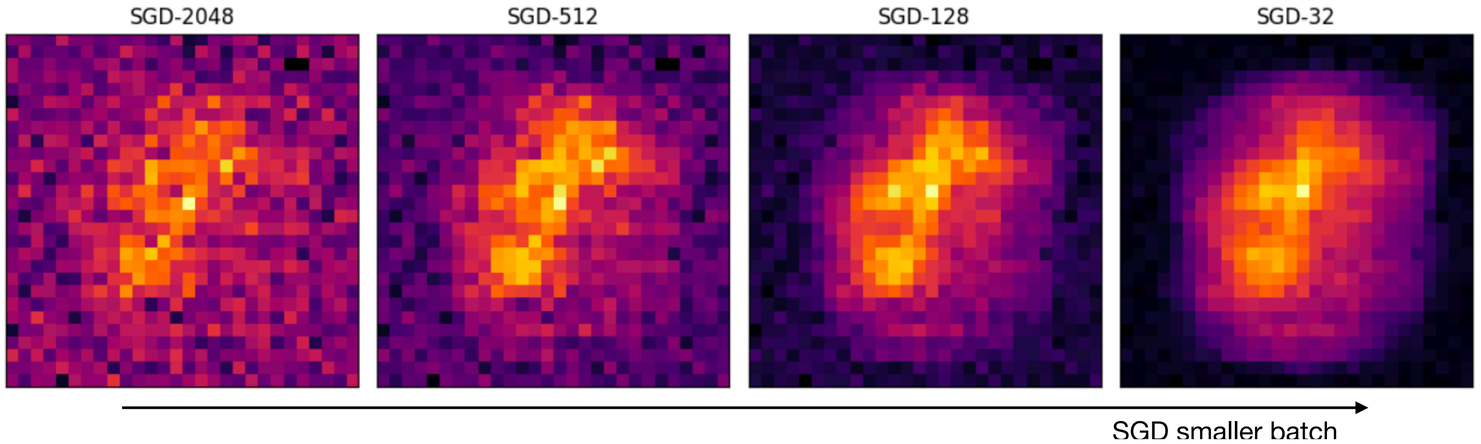
All things they have *not* been designed for!

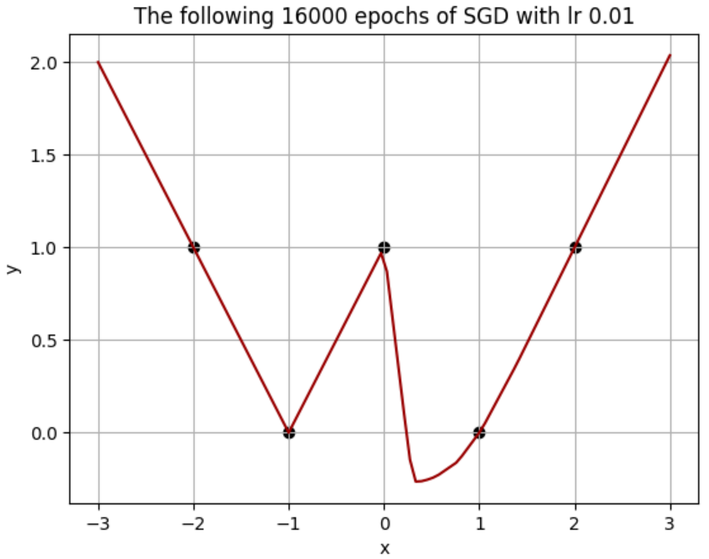
Goal for Mathematicians

Understanding the location of convergence

- Of different algorithms / hyperparameters







Answer

Check the step size / hyper parameters
given some sort of curvature

Answer

Check the step size / hyper parameters given some sort of curvature

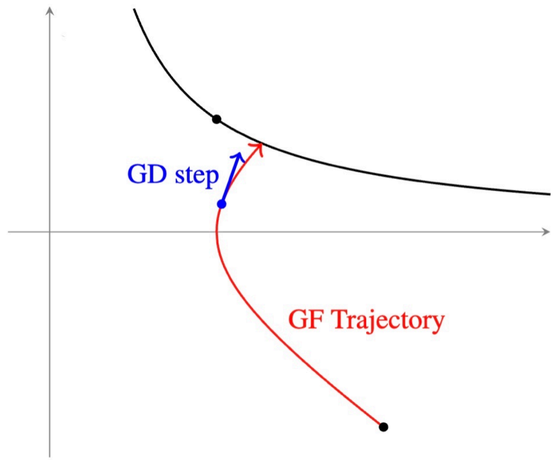
- ***Notions of curvature:***
  are the discriminant for large/small step.

- *You start stable:*
  Large == Progressive Sharpening is there!
  - *Why? To what extent? This is open!*

ANSWER

Check the step size / hyper parameters
given some sort of curvature

Small



The tools are Numerical Analysis-ish

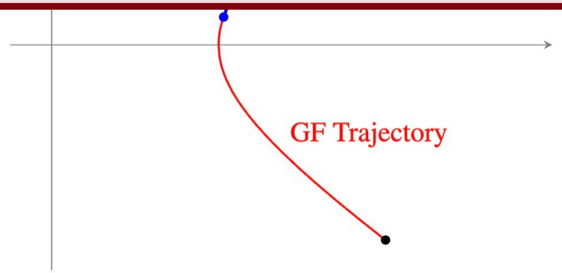
Answer

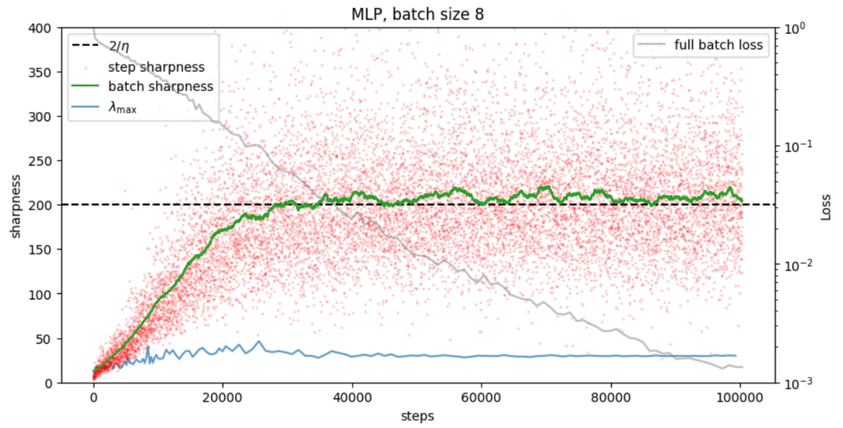
Check the step size / hyper parameters given some sort of curvature

Small

Large

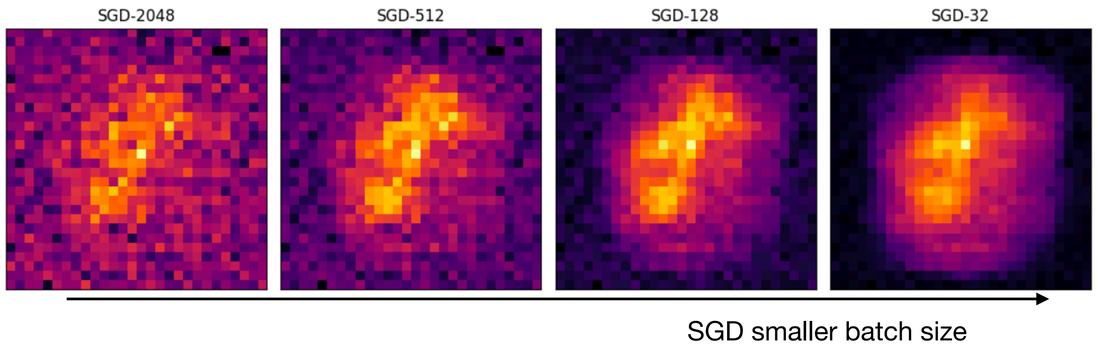
We understood empirically!





What is Missing

- Many things we didn’t discuss as how you escape saddles and local minima.
- How does this translate into what is learned? What do they learn? It changes with the optimizer but how?
- Is there a learning order?



The Broader Question

We have a model (i), some data (ii), a loss (iii), an optimizer (iv) with some hyper parameters (v), a computing budget (vi) and a precise initialization θ_0 (vii):

$\Longrightarrow$

What is the the function represented by the neural network with parameters θ^*

And it's about location of stabilization.