Oxford Machine Learning Summer School 2022

Optimisation for Machine Learning

Yali Du
Assistant professor
Department of Informatics, King's College London

People

Instructor Yali Du
yali.du@kcl.ac.uk

KCL Cristina Gava
cristina.gava@kcl.ac.uk

KCL Ziyan Wang:
wangziyan1998@gmail.com

CAS Xue Yan
yanxuesdu@gmail.com

CAS Runji Lin:
linprophet@icloud.com

Content

Part 0: Introduction to introduction …

Part 1: Theory of Convexity

Part 2: Gradient Descent and convergence

Part 3: Stochastic Gradient Descent, Non-Convex Optimization, Adam, and Machine Learning Applications

Practical work

https://colab.research.google.com/drive/1r-FTu17Utca7JjblyubSnFEC0n5ph3V6?usp=sharing

Optimization

General optimization problem (unconstrained minimization)

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{with} && \mathbf{x} \in \mathbb{R}^d \end{aligned}$$

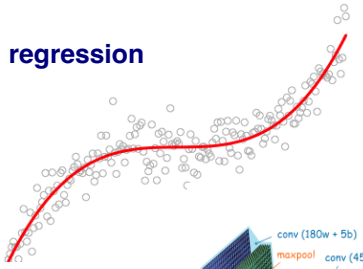
candidate solutions, variables, parameters $\mathbf{x} \in \mathbb{R}^d$

objective function $f : \mathbb{R}^d \mapsto \mathbb{R}$

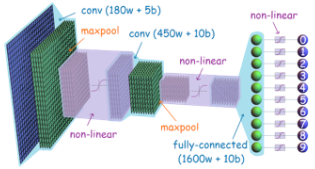
typically: technical assumption: f is continuous and differentiable

Why optimization?

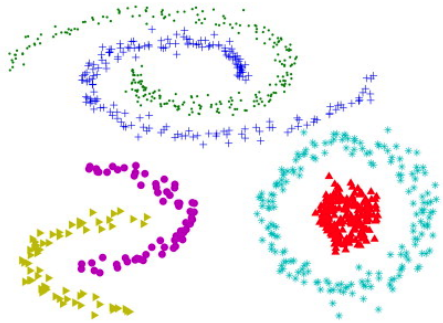
regression



classification



clustering/density estimation



computer games

Supervised Learning

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}\left(\mathbf{x}, \mathbf{a}_i, b_i\right)$$

Unsupervised Learning

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}\left(\mathbf{x}, \mathbf{a}_i\right)$$

Reinforcement Learning

$$\max_{\mathbf{x}\in\mathbb{R}^d} \mathbb{E}_{\tau\sim p_{\mathbf{x}(\tau)}}[R(\tau)]$$

… all these involve a minimisation of some function ...

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$$

Optimization for machine learning

Mathematical Modeling:

- defining & and measuring the machine learning model

Computational Optimization:

- learning the model parameters

Theory vs. practice:

- libraries are available, algorithms treated as “black box” by most practitioners
- Not here: we look inside the algorithms and try to understand why and how fast they work!

Solving optimization problems

Optimization at large scale:

Main approaches:

- Gradient Descent
- Stochastic Gradient Descent (SGD)
- …

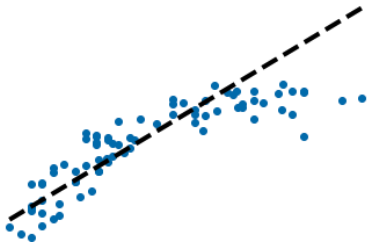
History:

- 1847: Cauchy proposes gradient descent
- 1950s: Linear Programs, soon followed by non-linear, SGD
- 1980s: General optimization, convergence theory
- 2005-2015: Large scale optimization (mostly convex), convergence of SGD
- 2015-today: Improved understanding of SGD for deep learning

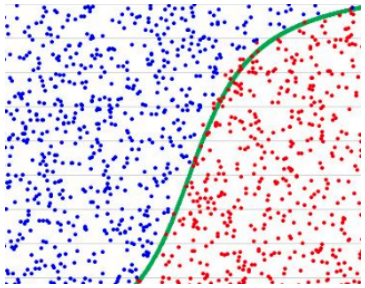
Function types, and what one can hope for …

… optimising for unknown parameters depends on the type of function under study ...

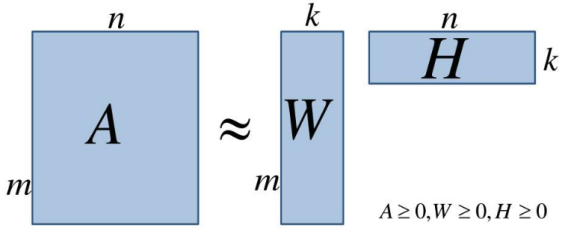
Convex $f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}), \lambda \in [0,1], \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$



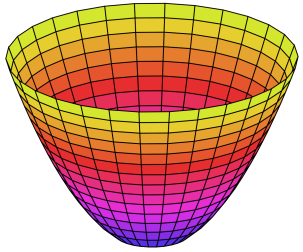
Linear Regression



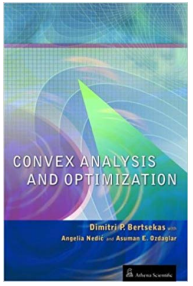
Classification with Hinge Loss



Non-Negative Matrix Factorisation



Unique global minimum

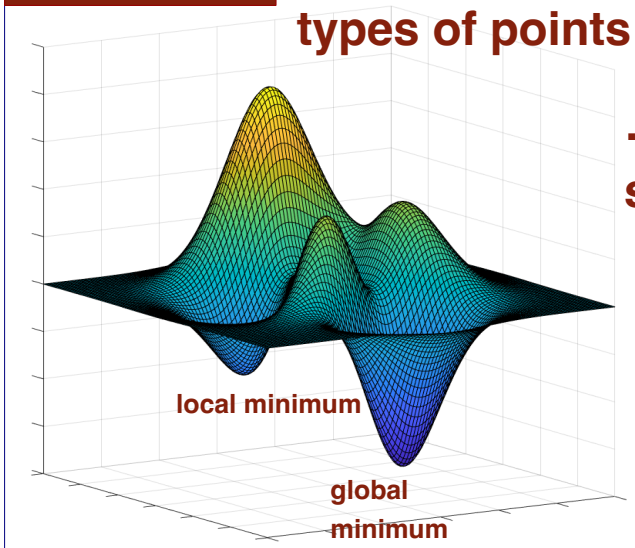


… admits polynomial time algorithms

Function types, and what one can hope for …

… optimising for unknown parameters depends on the type of function under study ...

Non-Convex **… global and local minima (checking) are NP-Hard, we look for other types of points …**



$$\nabla_{\mathbf{x}} f(\mathbf{x}_{\text{stationary}}) = 0$$

… so instead, the community is fetching for stationary points …

1. **ϵ-First-Order-Stationary Point (FOSP):** $\|\nabla_{\mathbf{x}} f(\mathbf{x}_{\text{FOSP}})\|_2 \leq \epsilon$

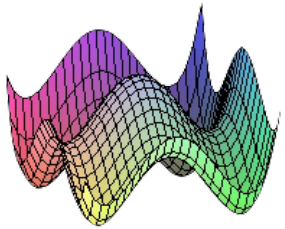
   [e.g., all global and local minima, saddle points, plateau points]

2. **ϵ- Second-Order-Stationary Point (SOSP):**

   $\|\nabla_{\mathbf{x}} f(\mathbf{x}_{\text{SOSP}})\|_2 \leq \epsilon$ and $\lambda_{\min}(\nabla^2_{\mathbf{x}} f(\mathbf{x}_{\text{SOSP}})) \geq -\sqrt{\epsilon}$

   [e.g., all global and local minima, plateau points]

Algorithms vary in type of information used …



Second-Order Methods

- Newton Method
- Regularised Newton
- Stochastic Quasi-Newton

First-Order Methods

- GD
- ADAM
- AdaGrad
- SGD
- NAGD
- RMSProp

Zero-Order Methods

- Bayesian Optimisation
- StoSOO
- StroquOOL
- ?

Non-Convex Optimisation

References

This tutorial is based on

Books:

- Convex Optimization by Stephen Boyd & Lieven Vandenberghe, 2003
- Convex Optimization: Algorithms and Complexity, by Sébastien Bubeck 2015

Course:

- Lecture notes: Optimization for Machine Learning, by Martin Jaggi & Nicolas Flammarion
- UTDallas Course - Convex Optimization- CS 7301, by Nicholas Ruozzi
- ADAM's Story and Proof - Machine Learning and AI Academy, by Haitham Bou-Ammar

Notes on Notations

$\mathbb{R}$: reals

$\mathbb{R}^d$: d-dimensional Euclidean space

$\mathbf{x} \in \mathbb{R}^d$: the variable or unknown parameters

$f : \mathbb{R}^d \mapsto \mathbb{R}$ the objective or cost function, and $\mathbb{R}^d$ is called the domain of f, or $\text{dom}(f) = \mathbb{R}^d$

$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x}_1)}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x}_d)}{\partial x_d} \right)^T$: the vector of partial derivatives, also the gradient of f at $\mathbf{x}$

Content

Part 0: Introduction to introduction …

Part 1: Theory of Convexity

Part 2: Gradient Descent and convergence

Part 3: Stochastic Gradient Descent, Non-Convex Optimization, Adam, and Machine Learning Applications

Practical work

https://colab.research.google.com/drive/1r-FTu17Utca7JjblyubSnFEC0n5ph3V6?usp=sharing

Theory of convexity

Convex Sets

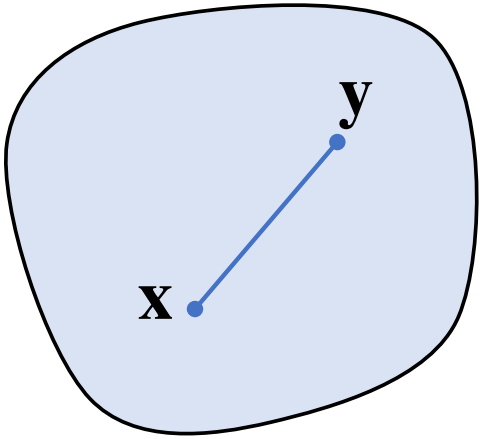
line segment between $\mathbf{x}_1$ and $\mathbf{x}_2$: all points

$$\mathbf{x} = t\mathbf{x}_1 + (1-t)\mathbf{x}_2$$

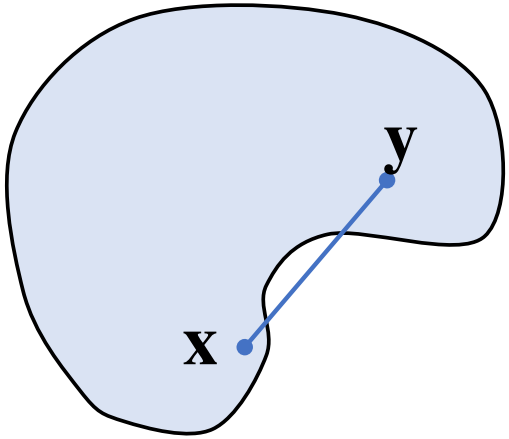
with $0 \leq t \leq 1$.

convex set: contains line segment between any two points in the set

That is, if $\mathbf{x}, \mathbf{y} \in C$, then $t\mathbf{x} + (1-t)\mathbf{y} \in C$ for all $t \in [0,1]$



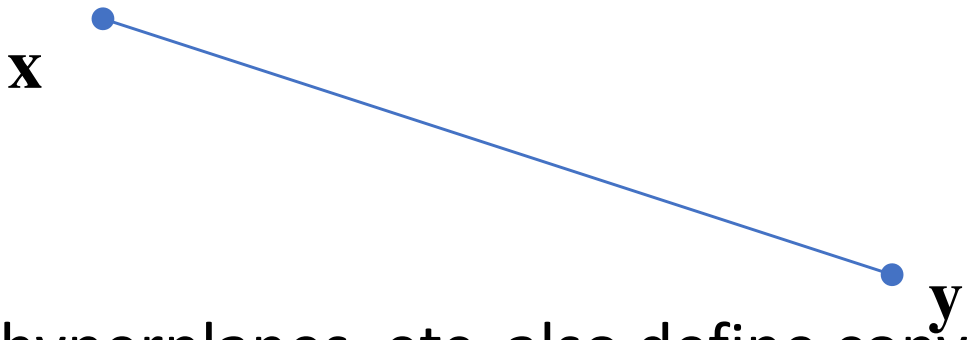
Convex Set



Not a Convex Set

Examples of Convex Sets

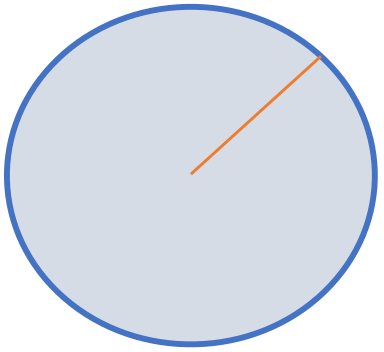
Line Segments: $C = \{t\mathbf{x} + (1-t)\mathbf{y} : t \in [0,1]\}$ for some $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$



Lines, planes, hyperplanes, etc. also define convex sets

Balls of Radius ϵ: $C = \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq \epsilon\right\}$ for some $\epsilon \geq 0 \in \mathbb{R}$

$$\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}$$



Properties of Convex Sets

Intersections of convex sets are convex

Observation. Let $C_i, i \in I$ be convex sets, where I is a (possibly infinite) index set. Then $C = \cap_{i \in I} C_i$ is a convex set.

Projections onto convex sets are unique, and often efficient to compute

$$P_C(\mathbf{x}) := \operatorname{argmin}_{\mathbf{y} \in C} \|\mathbf{y} - \mathbf{x}\|$$

Convex functions

Convex Functions

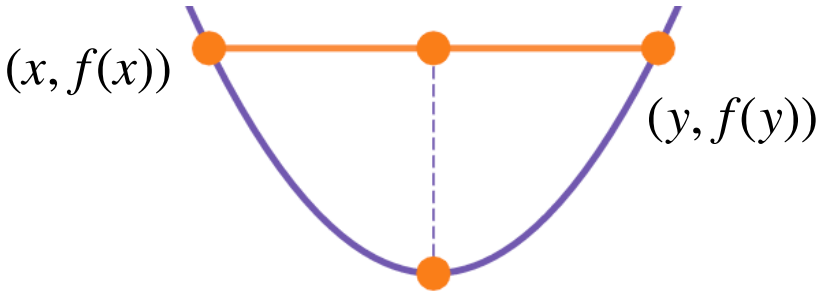
A **function** $f: C \to \mathbb{R}^n$ is **convex** if C is a convex set and

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in C$ and $\lambda \in [0,1]$

Geometrically: The line segment between $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ lies above the graph of f.

f is called **concave** if $-f$ is convex



Smiley face

Examples

convex:

- affine function: $\mathbf{a}^T\mathbf{x} + b$
- exponential: e^{ax}, for any $a \in \mathbb{R}$
- powers: x^α on $\mathbb{R}_+$, for $\alpha \geq 1$ or $\alpha \leq 0$
- negative entropy: $x\log x$ on $\mathbb{R}_+$

concave:

- affine: $\mathbf{a}^T\mathbf{x} + b$
- powers: x^αon $\mathbb{R}_+$, for $0 \leq \alpha \leq 1$
- logarithm: $\log x$ on $\mathbb{R}_+$

Note:

- $\mathbb{R}_+$ indicate positive reals.
- affine functions are convex and concave.

First-order condition

f is differentiable if C is open and the gradient

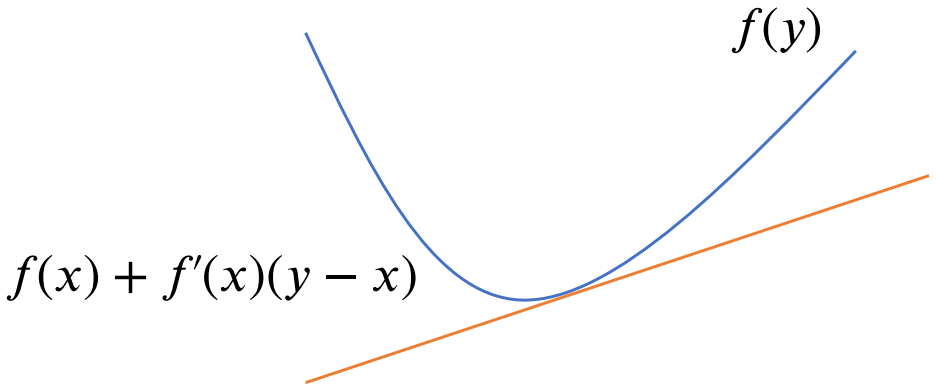
$$\nabla f(\mathbf{x}) := \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_d}(\mathbf{x}) \right)^T$$

exist at each $\mathbf{x} \in C$

1st-order condition: differentiable f with convex domain is convex iff

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \quad \mathbf{x}, \mathbf{y} \in \mathrm{dom}(f)$$

i.e. first-order approximation of f is global underestimator



First-order condition

i.e. first-order approximation of f is global underestimator

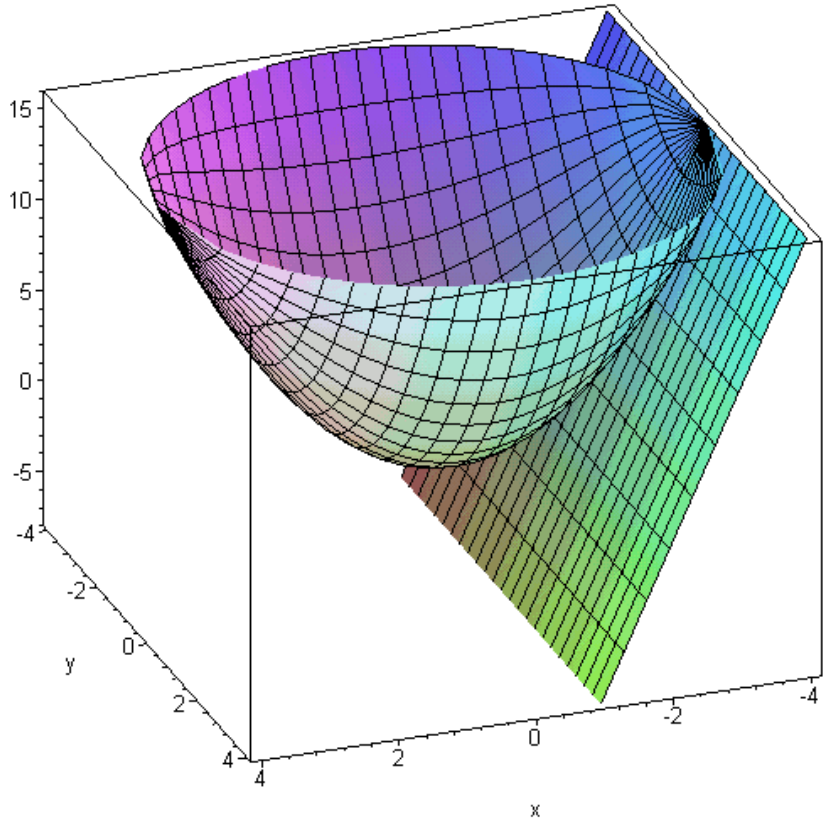


Image: Lane Vosbury, Seminole State College

Second-order condition

f is twice differentiable if C is open and the Hessian $\nabla^2 f(\mathbf{x})$,

$$\nabla^2 f(\mathbf{x})_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}),\ i, j = 1,\ldots,d$$

exist at each $\mathbf{x} \in C$

2nd-order conditions: for twice differentiable f with convex domain

f is convex if and only if

$$\nabla^2 f(\mathbf{x}) \succcurlyeq 0 \text{ for all } \mathbf{x} \in C$$

i.e. the hessian is positive semi-definite (all of the eigenvalues are nonnegative).

Second-order conditions

Special case when $C \subseteq \mathbb{R}$

A twice continuously differentiable **function** $f: C \to \mathbb{R}$ with $C \subseteq \mathbb{R}$ is **convex** if C is a convex set and

$$\frac{d^2 f}{dx^2}(x) \geq 0$$

for all $x \in C$

Operations Preserving Convexity

Nonnegative weighted sums of convex functions are convex, i.e., if $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ are convex functions and $c_1, c_2 \geq 0$, then

$$g(\mathbf{x}) = c_1 f_1(\mathbf{x}) + c_2 f_2(\mathbf{x})$$

is a convex function.

Proof:

Let $\mathbf{z} = t\mathbf{x} + (1-t)\mathbf{y}$,

$$\begin{aligned} g(\mathbf{z}) &= c_1 f_1(\mathbf{z}) + c_2 f_2(\mathbf{z}) \\ &\leq c_1(t f_1(\mathbf{x}) + (1-t) f_1(\mathbf{y})) + c_2(t f_2(\mathbf{x}) + (1-t) f_2(\mathbf{y})) \\ &= t(c_1 f_1(\mathbf{x}) + c_2 f_2(\mathbf{x})) + (1-t) c_1(f_1(\mathbf{y})) + c_2(f_2(\mathbf{y})) \\ &= t g(\mathbf{x}) + (1-t) g(\mathbf{y}) \end{aligned}$$

The statement follows.

Operations Preserving Convexity

Composition with an affine function, i.e., if $f: \mathbb{R}^n \to \mathbb{R}$ is a convex function and $M \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$, then $g: \mathbb{R}^m \to \mathbb{R}$ given by

$$g(\mathbf{x}) = f(M\mathbf{x} + b)$$

is a convex function.

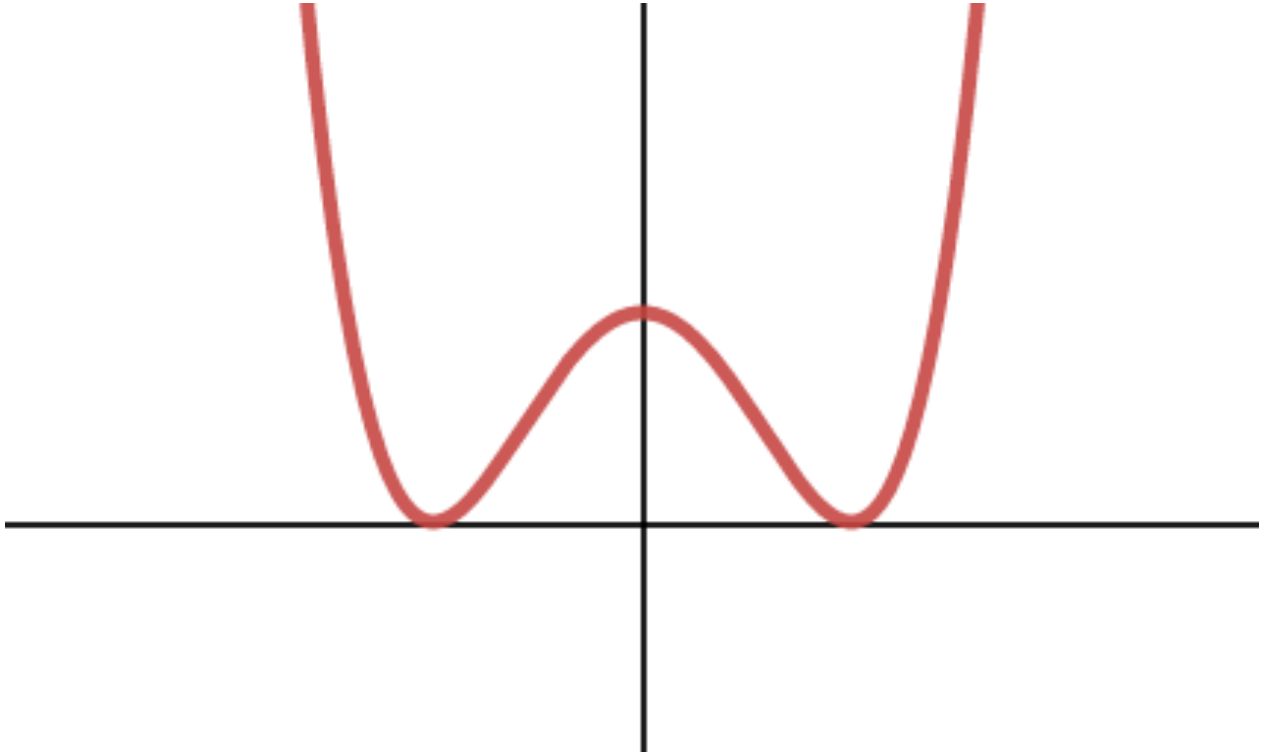
Questions:

Would any composition of two convex functions be convex?

Counterexample: composite function

Counterexample: Consider two convex functions $f(x) = x^2$ and $g(x) = x^2 - 1$. Then, the composition

$$(f \circ g)(x) = x^4 - 2x^2 + 1.$$



Operations Preserving Convexity

Pointwise maximum of convex functions are convex, i.e., if $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ are convex functions, then

$$g(\mathbf{x}) = \max(f_1(\mathbf{x}), f_2(\mathbf{x}))$$

is a convex function.

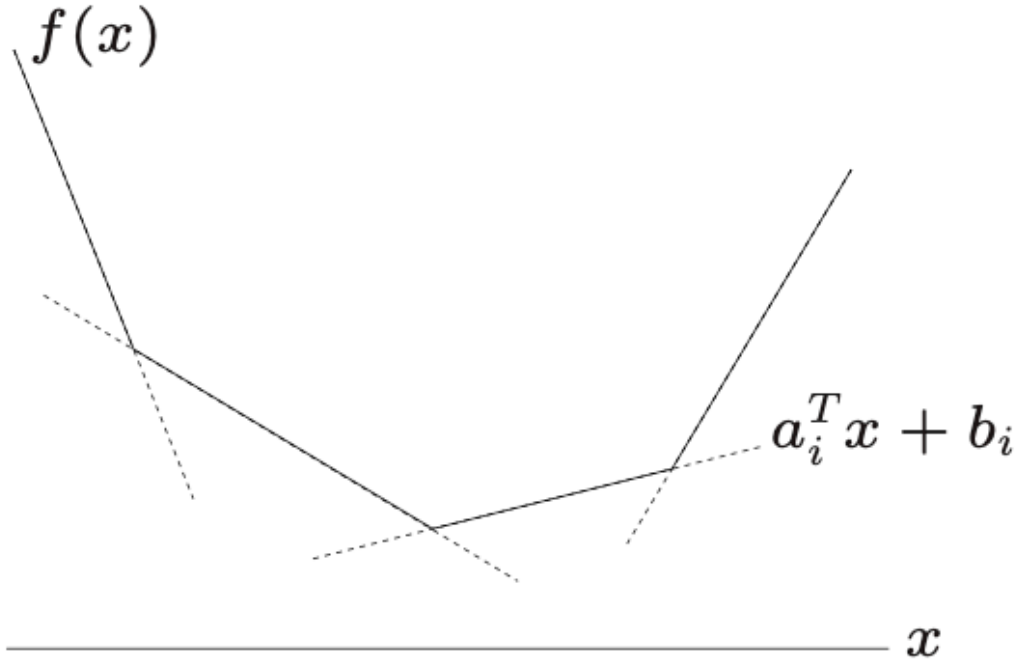


Image: L. Vandenberghe

Convex optimisation

Optimization

Convex Optimization Problem is of the form

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in X \end{aligned}$$

Where both

- f is a convex function
- $X \subseteq \text{dom}(f)$ is a convex set

Crucial Property of Convex Optimization Problems

- Every local minimum is a **global minimum**, see later...

Why do we favor convex optimization ?

For convex optimization problems, all algorithms

- Coordinate Descent, Gradient Descent, Stochastic Gradient Descent, Projected and Proximal Gradient Descent

do **converge** to the global optimum! (assuming f differentiable)

Example Theorem: The convergence rate is proportional to $\frac{1}{t}$,

i.e.

$$f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \frac{c}{t}$$

(where $\mathbf{x}^\star$ is some optimal solution to the problem.)

Meaning: **Approximation error** converges to 0 over time.

Local Minima are Global Minima

Definition

A local minimum of $f : \text{dom}(f) \mapsto \mathbb{R}$ is a point $\mathbf{x}$ such that there exists $\epsilon > 0$ with

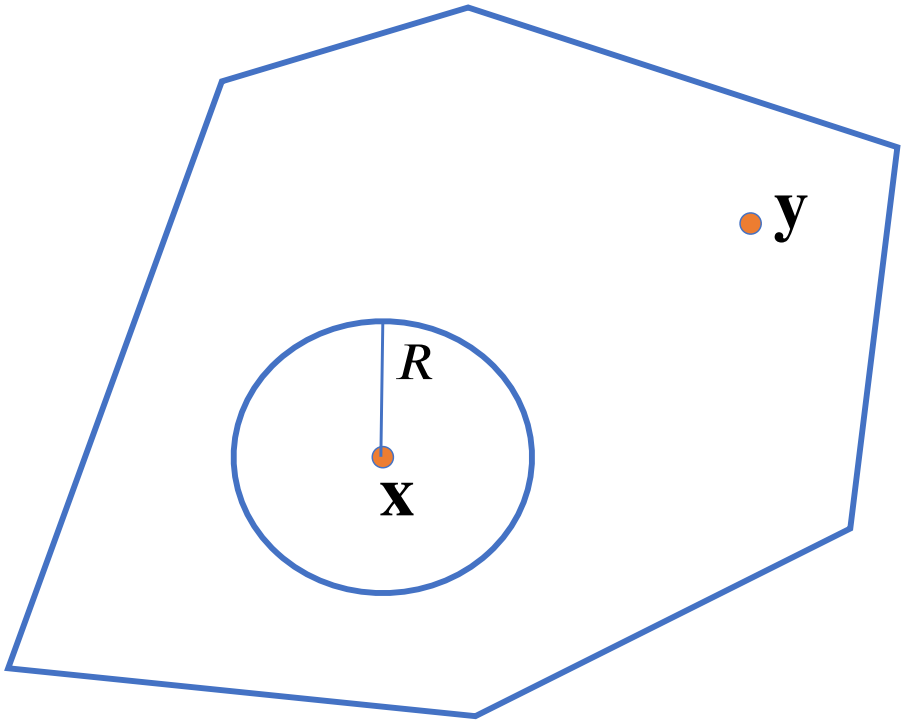
$$f(\mathbf{x}) \leq f(\mathbf{y}) \ \forall \mathbf{y} \in \text{dom}(f) \text{ satisfying } \|\mathbf{y} - \mathbf{x}\| < \epsilon$$

Lemma: Let $\mathbf{x}^\star$ be a local minimum of a convex function $f : \text{dom}(f) \mapsto \mathbb{R}$. Then $\mathbf{x}^\star$ is a global minimum, meaning that $f(\mathbf{x}^\star) \leq f(\mathbf{y}) \ \forall \mathbf{y} \in \text{dom}(f)$.

Local Minima are Global Minima

Proof.

Let $\mathbf{x}$ be a local minimum, assume there exists $\mathbf{y} \in \text{dom}(f)$ such that $f(\mathbf{x}) > f(\mathbf{y})$



$\mathbf{x}$ is a local minimum implies that there exists a radius R such that for all points $\mathbf{x}'$ within the ball of radius R of $\mathbf{x}$,

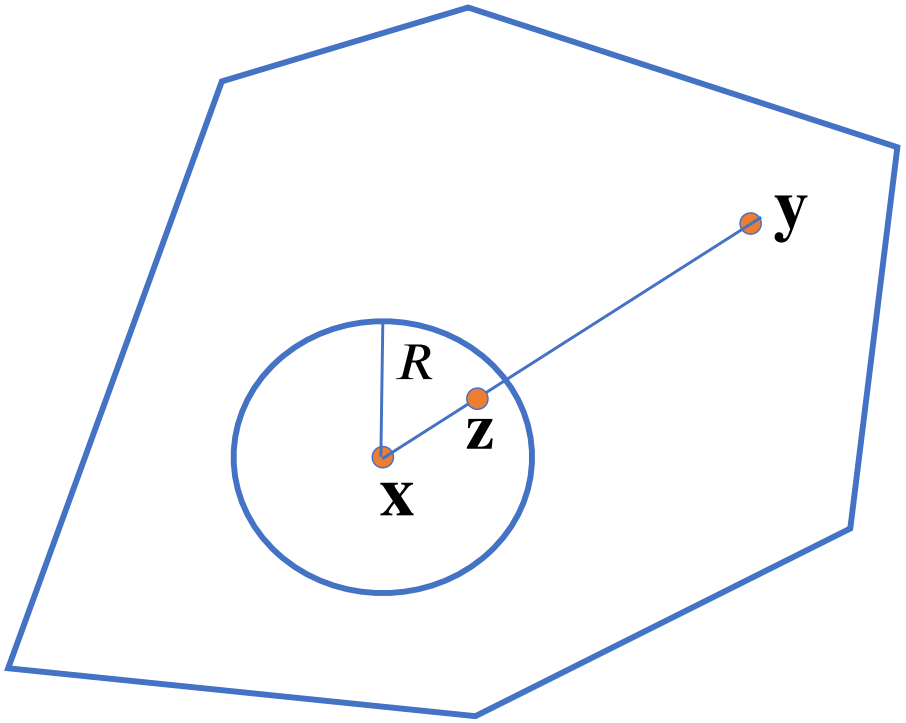
$$f(\mathbf{x}) \leq f(\mathbf{x}')$$

Local Minima are Global Minima

Let $\mathbf{z} = \theta\mathbf{y} + (1-\theta)\mathbf{x}$,

(1) By convexity,

$$f(\mathbf{z}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}) < f(\mathbf{x})$$



(2) Let $\theta = \dfrac{R}{2\|\mathbf{x}-\mathbf{y}\|_2}$,

Then

$$\|\mathbf{x}-\mathbf{z}\|_2 = \theta\|\mathbf{x}-\mathbf{y}\|_2 = \frac{R}{2} < R$$

This implies that $f(\mathbf{x}) \leq f(\mathbf{z})$.

(1) and (2) cannot hold at the same time, which is a contradiction to $\mathbf{x}$ being a local minimum.

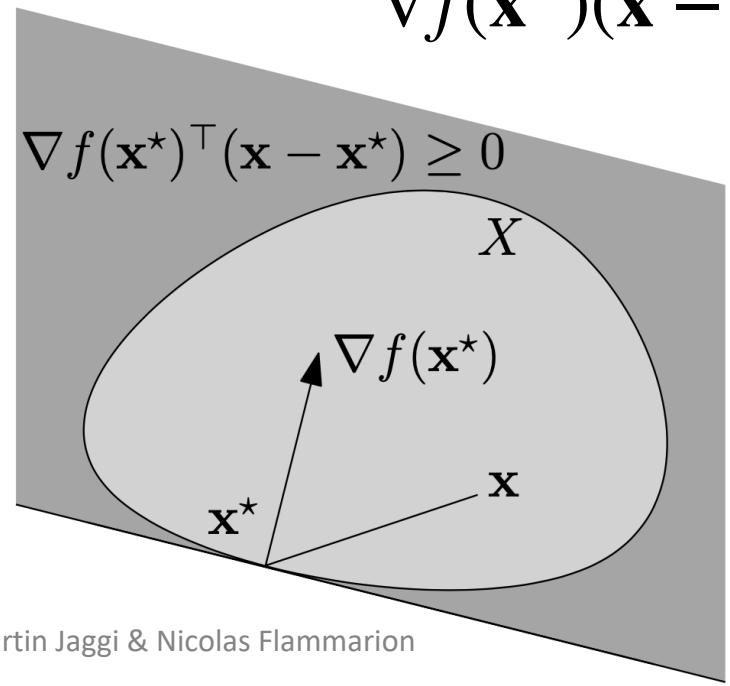
Optimality criterion for differentiable f

Let $f : \text{dom}(f) \mapsto R$ be convex and let $X \in \text{dom}(f)$ be a covnex set. a point $\mathbf{x}$ is a minimizer of f over X if

$$f(\mathbf{x}) \leq f(\mathbf{y}) \; \forall \mathbf{y} \in X$$

- $\mathbf{x}^\star$ is optimal if and only if it is feasible and

$$\nabla f(\mathbf{x}^\star)(\mathbf{x} - \mathbf{x}^\star) \geq 0 \text{ for all feasible } \mathbf{x}$$



In words: all feasible directions from $\mathbf{x}$ are aligned with gradient $\nabla f(\mathbf{x})$

Image: Martin Jaggi & Nicolas Flammarion

Critical Points are Global Minima

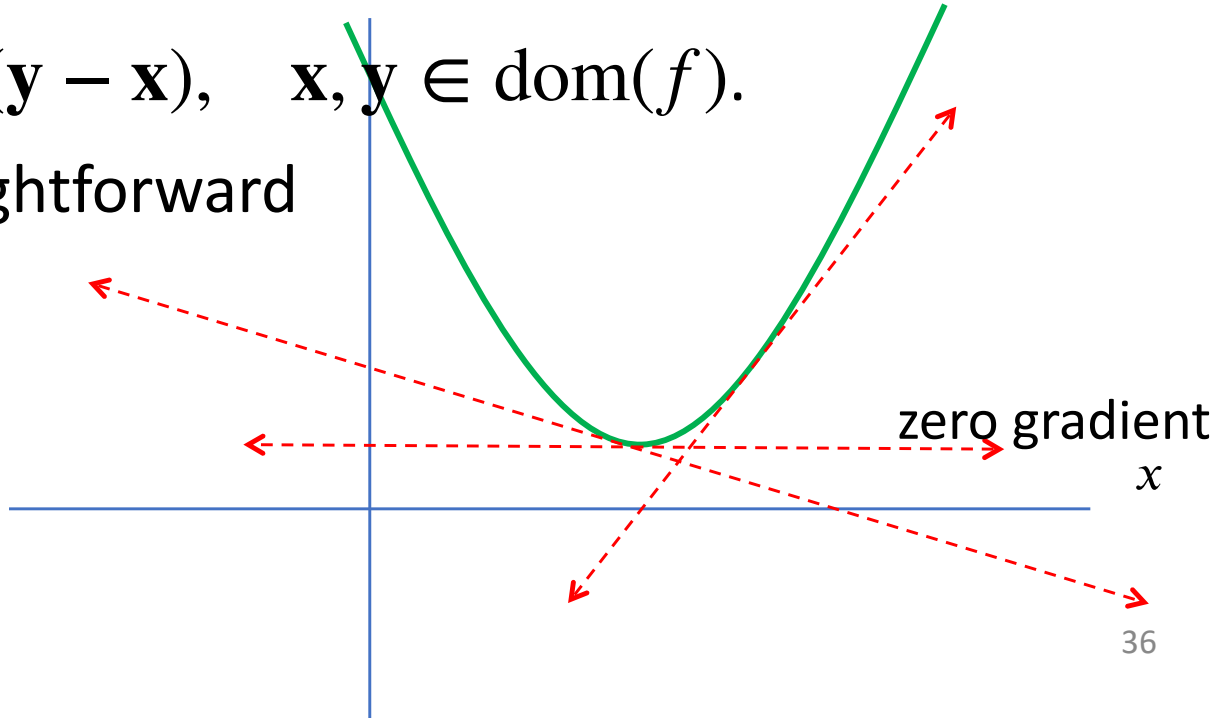
Suppose that f is convex and differentiable over an open domain $\mathrm{dom}(f)$. Let $\mathbf{x} \in \mathrm{dom}(f)$. If $\nabla f(\mathbf{x}) = 0$ (critical point), then $\mathbf{x}$ is a global minimum.

Proof

Recall the first-order condition of convexity,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^{\top}(\mathbf{y} - \mathbf{x}), \quad \mathbf{x}, \mathbf{y} \in \mathrm{dom}(f).$$

The conclusion is straightforward



Questions?

Content

Part 0: Introduction to introduction …

Part 1: Theory of Convexity

Part 2: Gradient Descent and convergence

Part 3: Stochastic Gradient Descent, Non-Convex Optimization, Adam, and Machine Learning Applications

Practical work

https://colab.research.google.com/drive/1r-FTu17Utca7JjblyubSnFEC0n5ph3V6?usp=sharing

Gradient method

Gradient Descent

Consider unconstrained convex optimization

$$\min_{\mathbf{x}} f(\mathbf{x})$$

where f is convex, differentiable with $\text{dom}(f) = \mathbb{R}^d$ and has a global minimum $\mathbf{x}^\star$.

Goal: Find $\mathbf{x} \in \mathbb{R}^d$ such that

$$f(\mathbf{x}) - f(\mathbf{x}^\star) \leq \epsilon\,.$$

Note that there can be several global minima $\mathbf{x}_1^\star \neq \mathbf{x}_2^\star$ with $f(\mathbf{x}_1^\star) = f(\mathbf{x}_2^\star)$.

Gradient Descent Algorithm:

Choose initial point $\mathbf{x}_0 \in \mathbb{R}^d$, repeat

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f\left(\mathbf{x}_t\right), t = 0,1,2,...$$

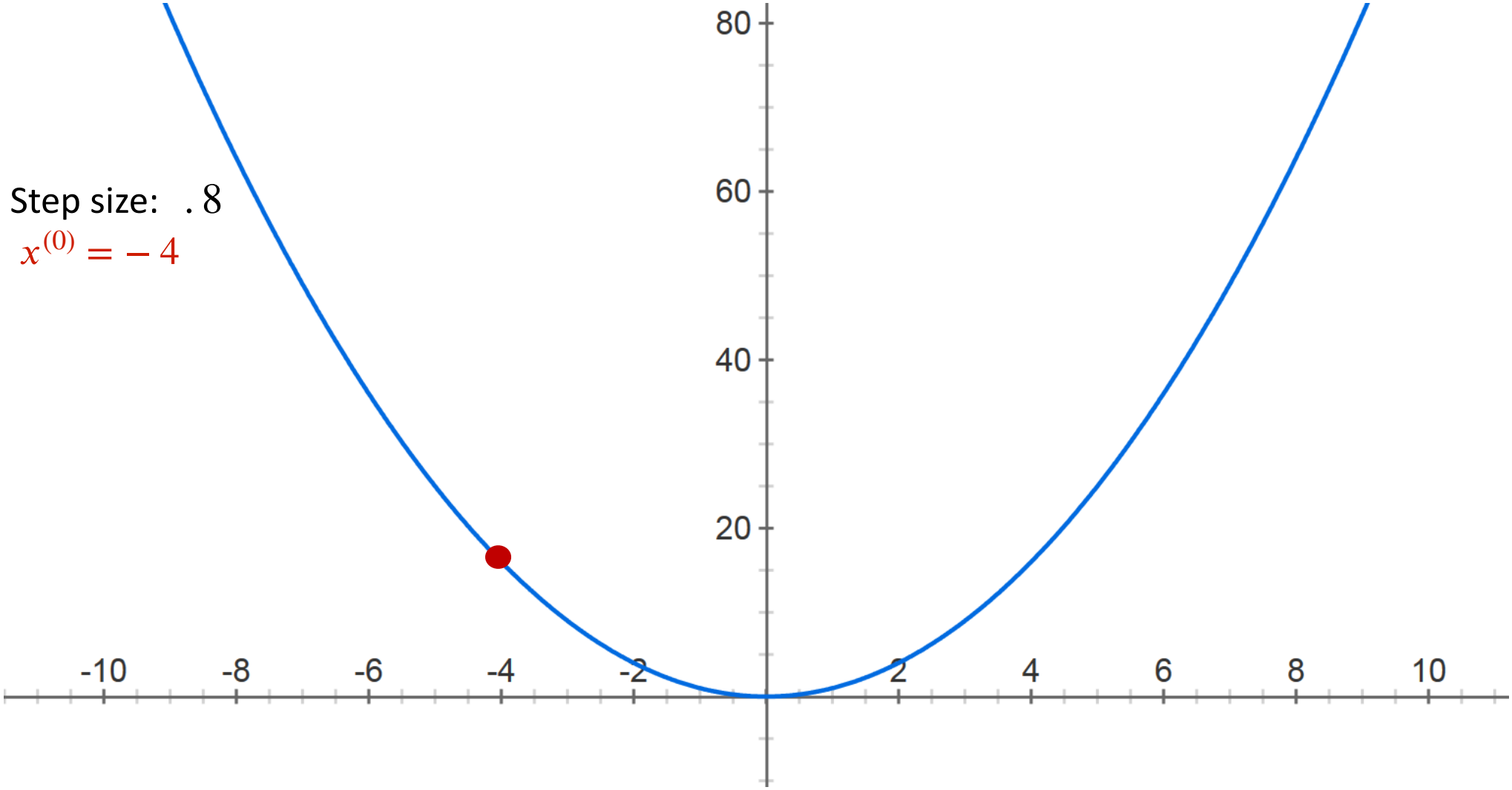
for **step size** γ (or learning rate)

Gradient Descent

$f(x) = x^2$

Step size: .8

$x^{(0)} = -4$



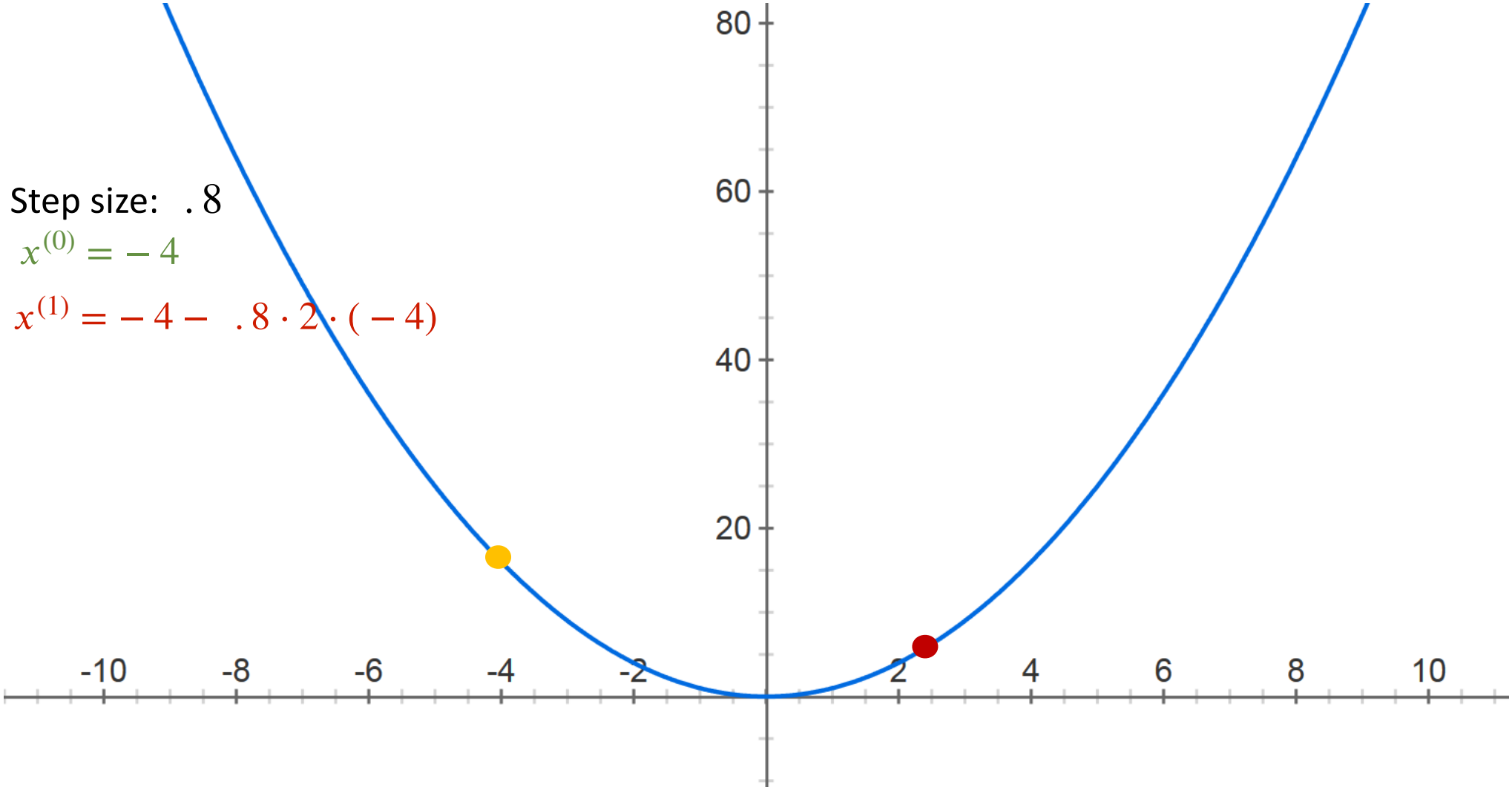
Gradient Descent

$f(x) = x^2$

Step size: .8

$x^{(0)} = -4$

$x^{(1)} = -4 - .8 \cdot 2 \cdot (-4)$



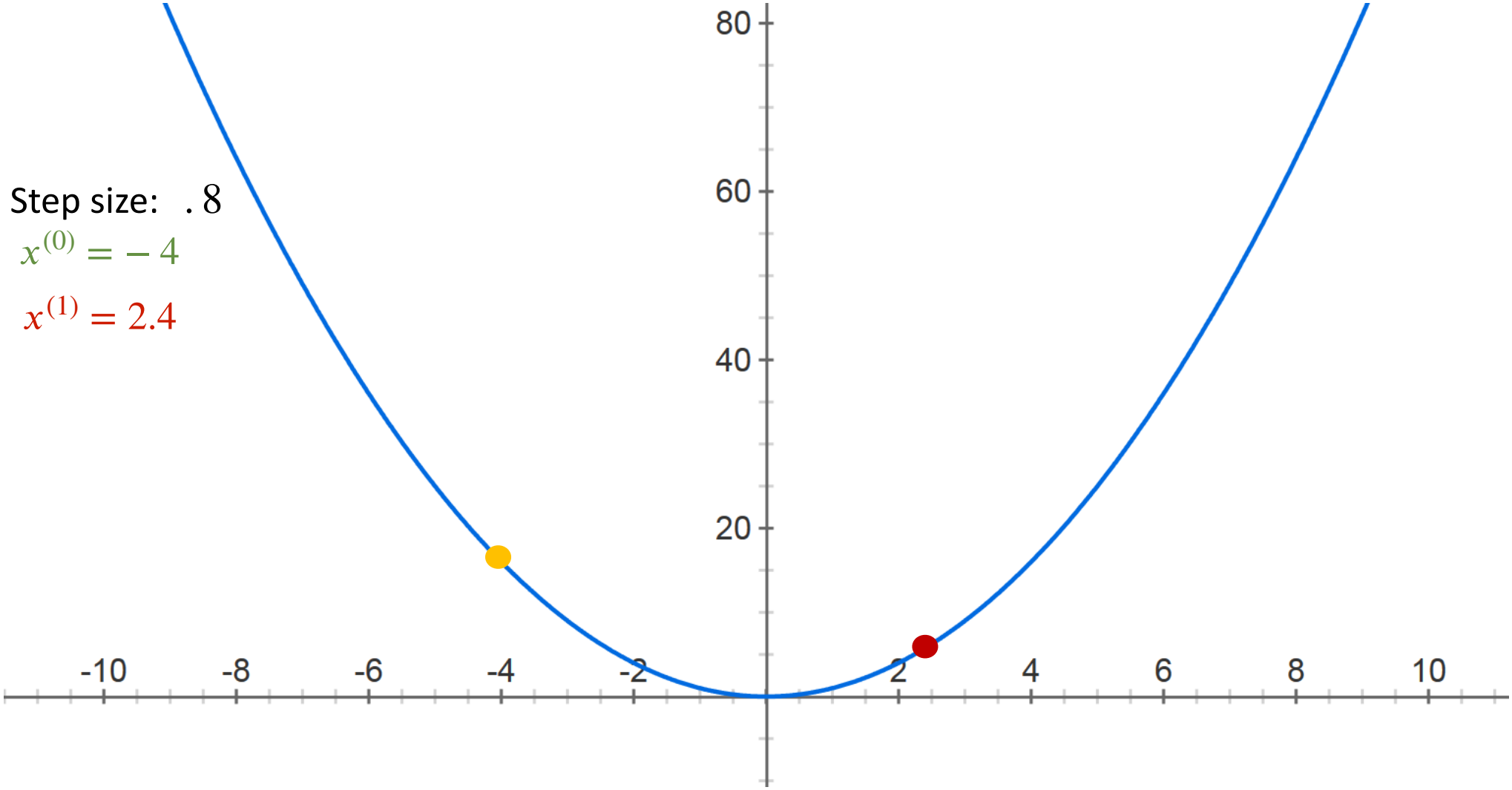
Gradient Descent

$f(x) = x^2$

Step size: .8

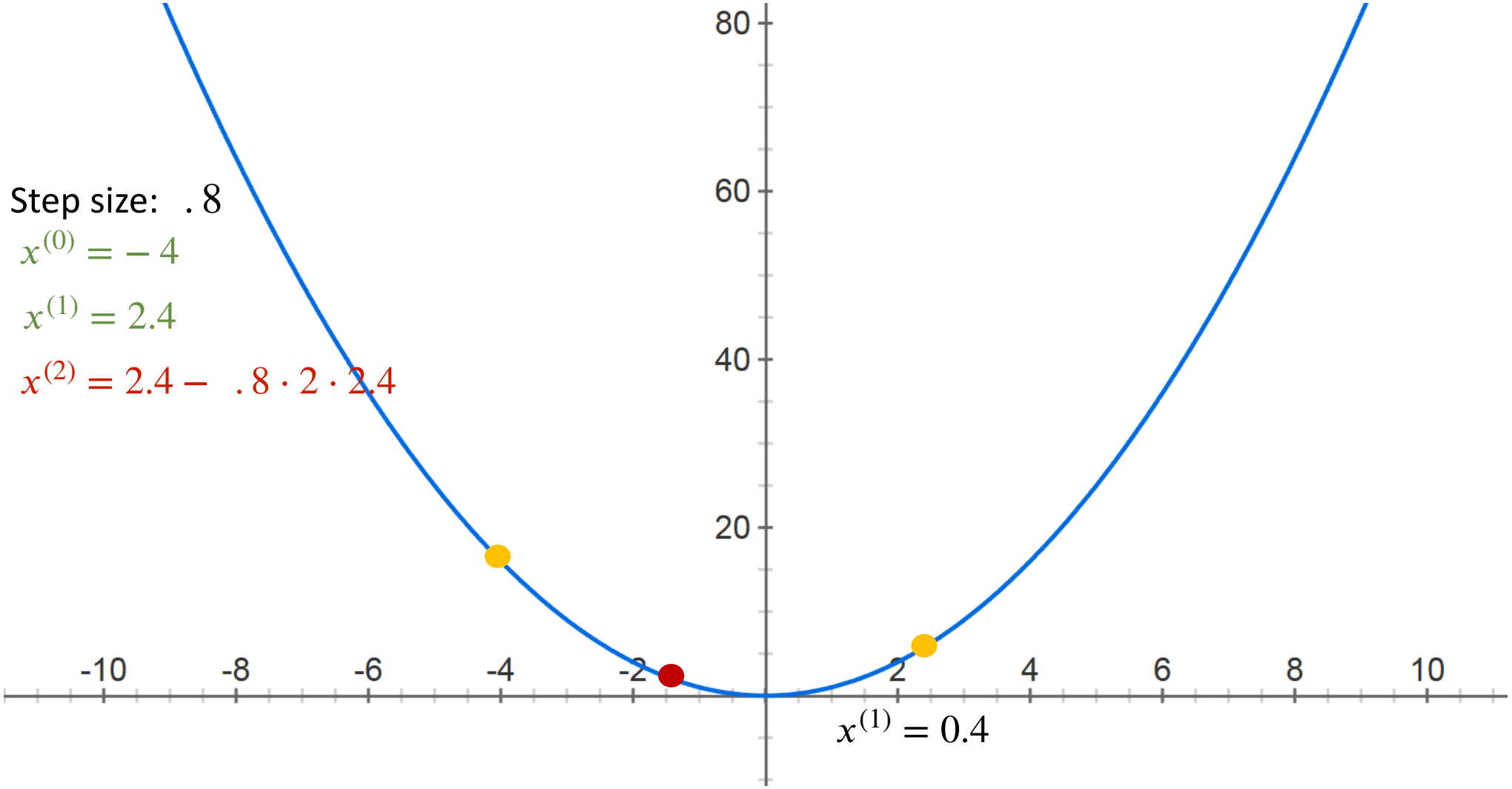
$x^{(0)} = -4$

$x^{(1)} = 2.4$



Gradient Descent

$f(x) = x^2$



Step size: .8

$x^{(0)} = -4$

$x^{(1)} = 2.4$

$x^{(2)} = 2.4 - .8 \cdot 2 \cdot 2.4$

Gradient Descent

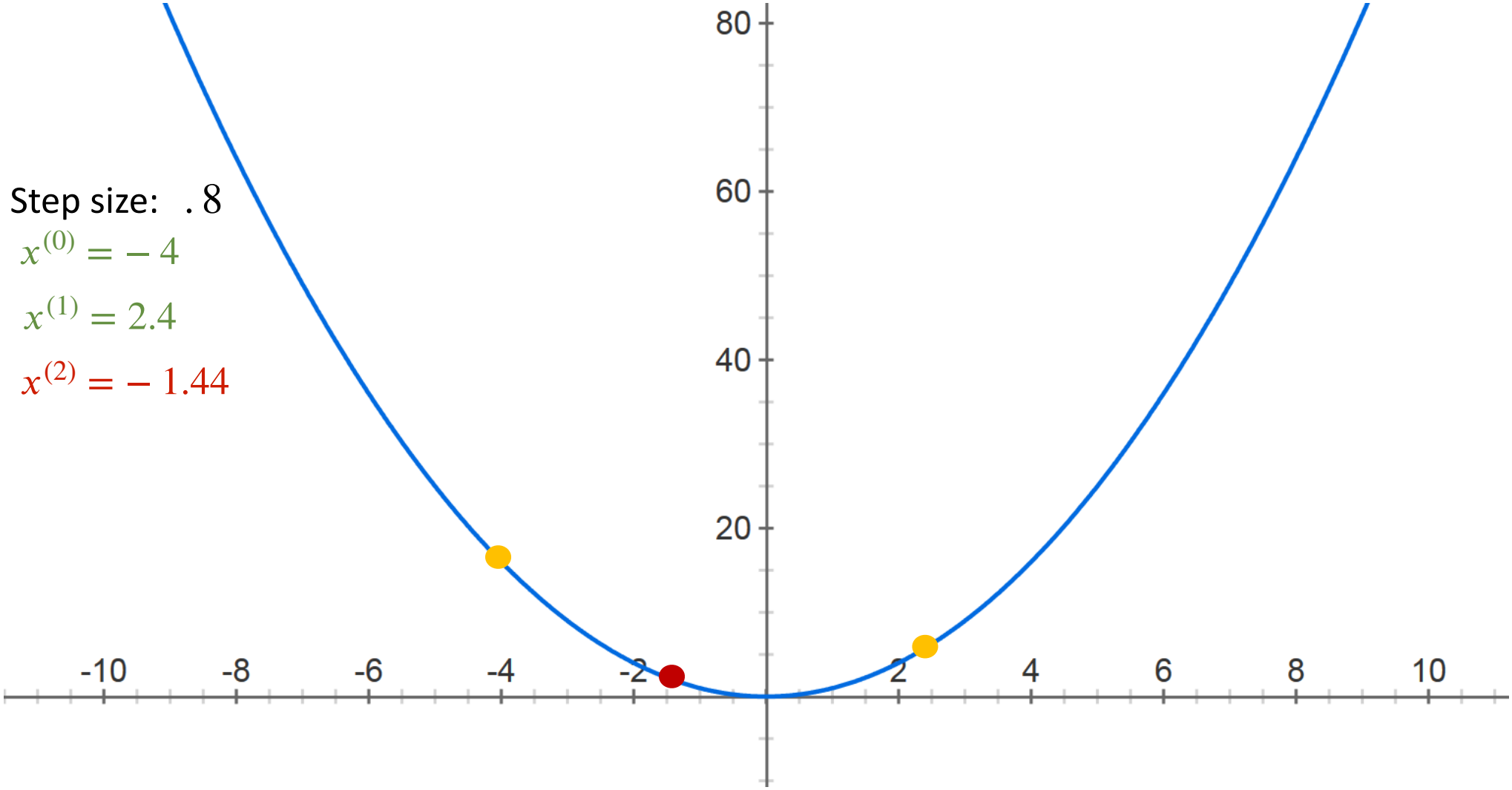
$f(x) = x^2$

Step size: .8

$x^{(0)} = -4$

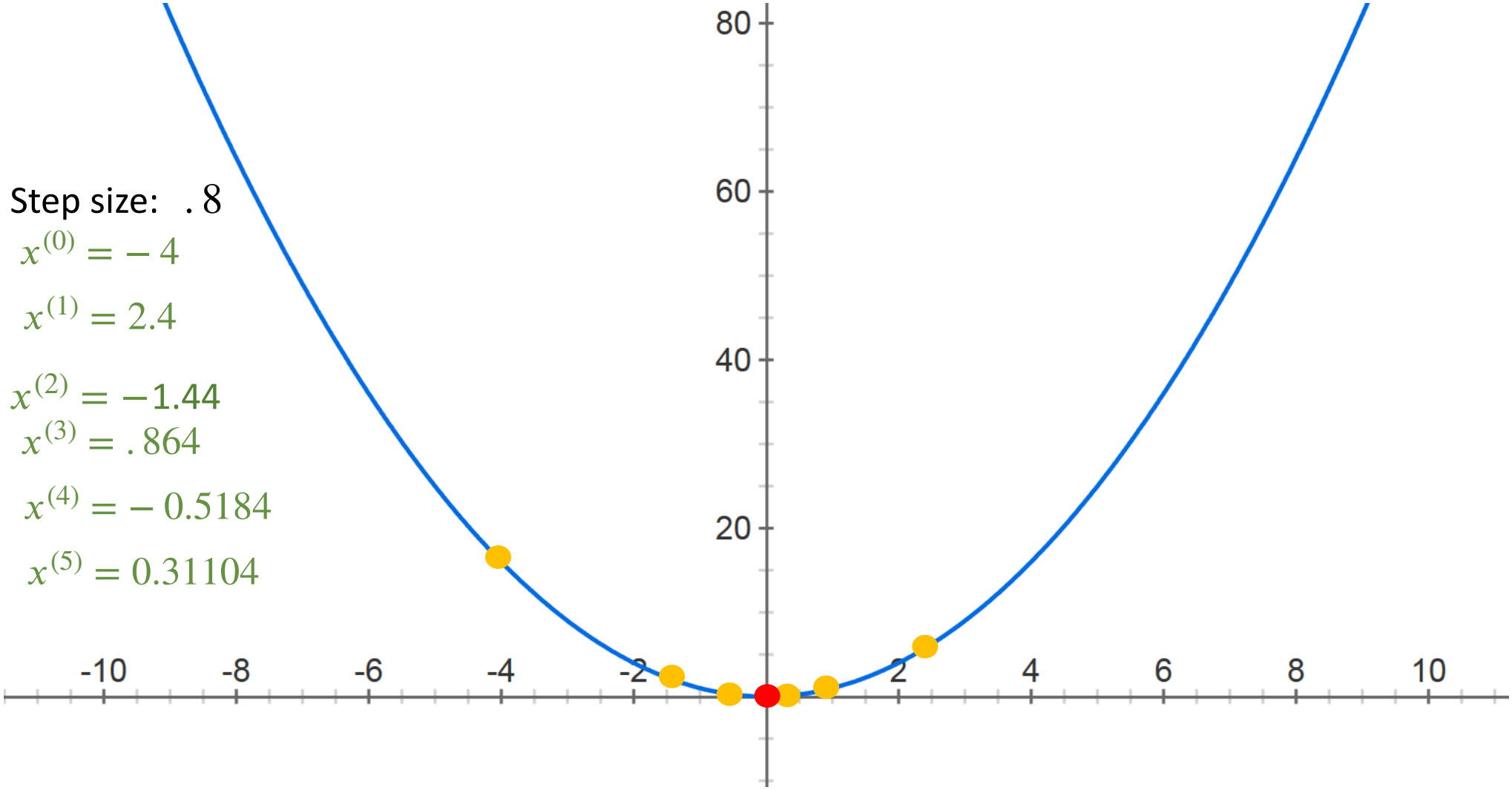
$x^{(1)} = 2.4$

$x^{(2)} = -1.44$



Gradient Descent

$f(x) = x^2$



Step size: .8

$x^{(0)} = -4$

$x^{(1)} = 2.4$

$x^{(2)} = -1.44$

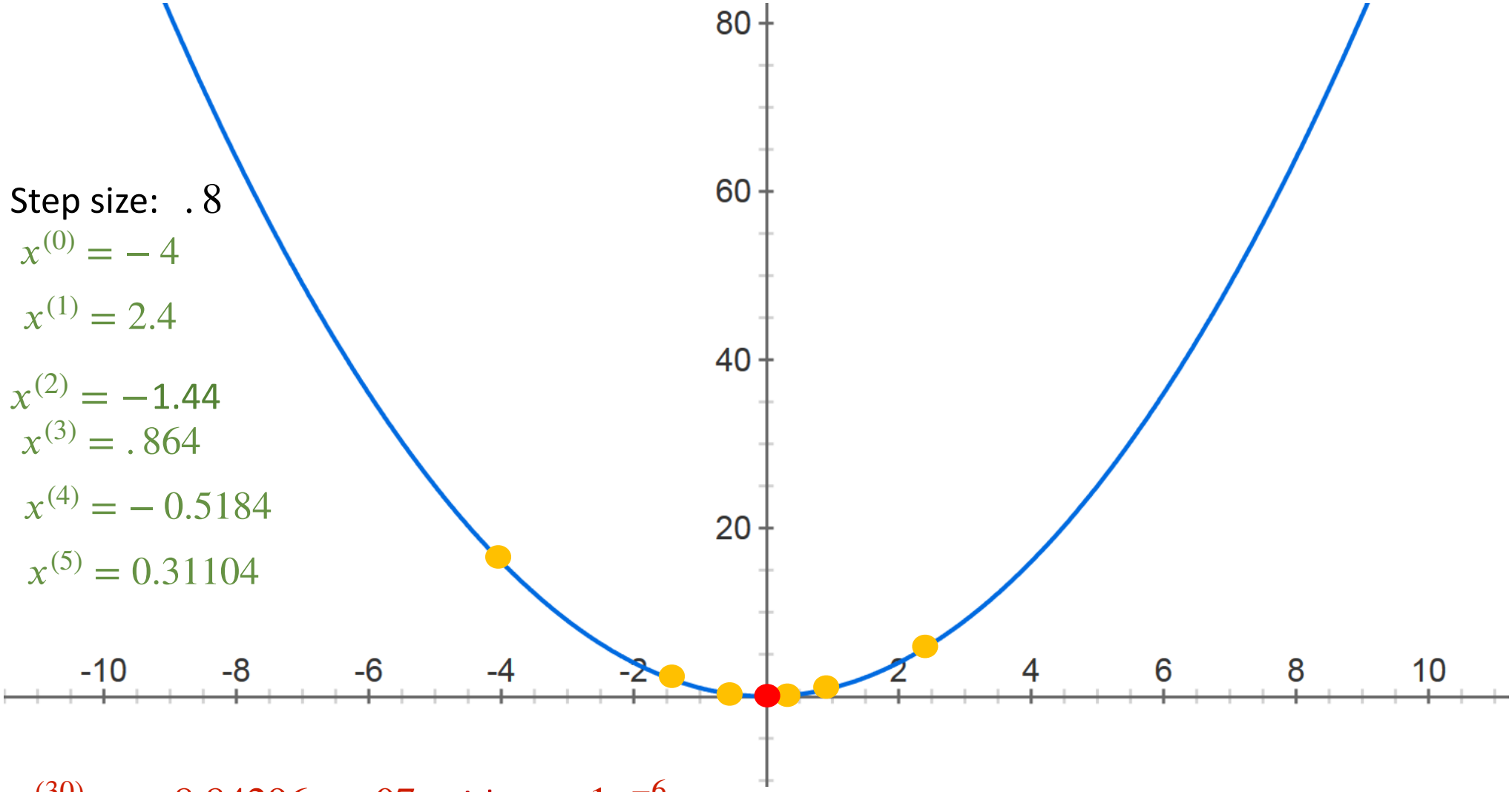
$x^{(3)} = .864$

$x^{(4)} = -0.5184$

$x^{(5)} = 0.31104$

Gradient Descent

$f(x) = x^2$



Step size: .8

$x^{(0)} = -4$

$x^{(1)} = 2.4$

$x^{(2)} = -1.44$

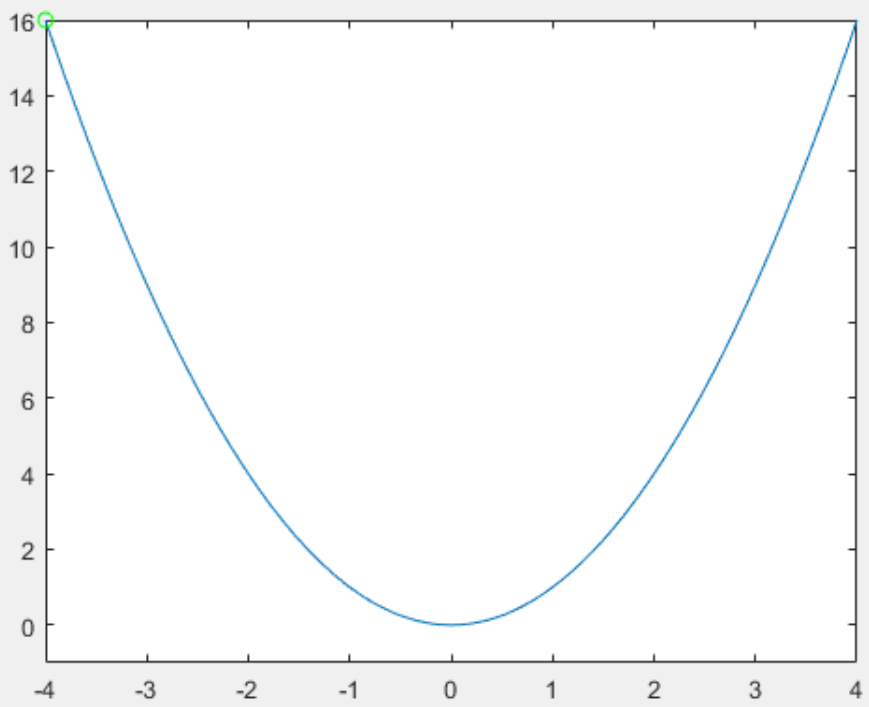
$x^{(3)} = .864$

$x^{(4)} = -0.5184$

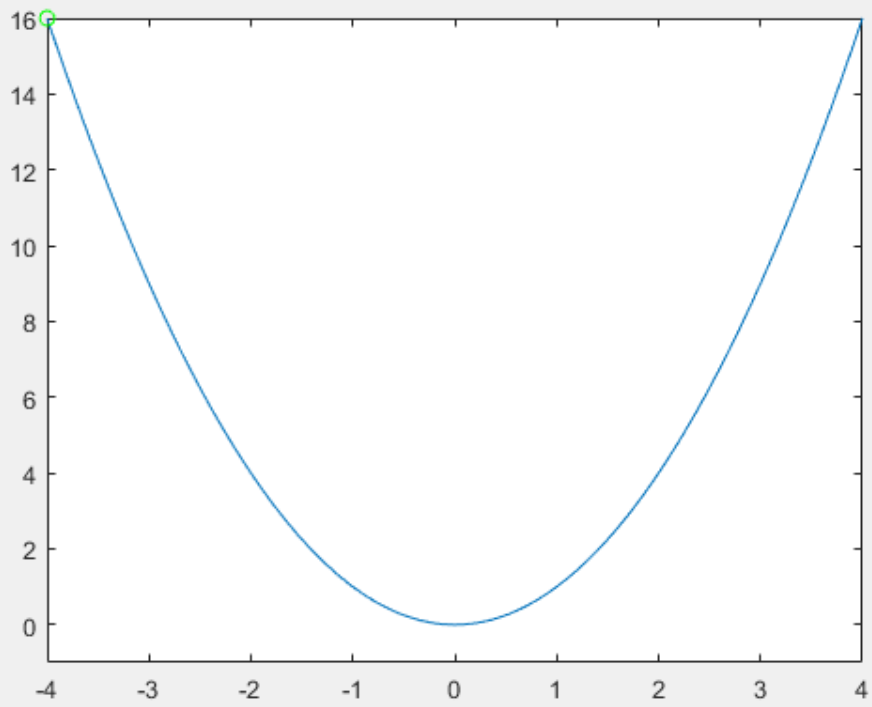
$x^{(5)} = 0.31104$

$x^{(30)} = -8.84296e-07$, with $\epsilon = 1e^{-6}$

Gradient Descent



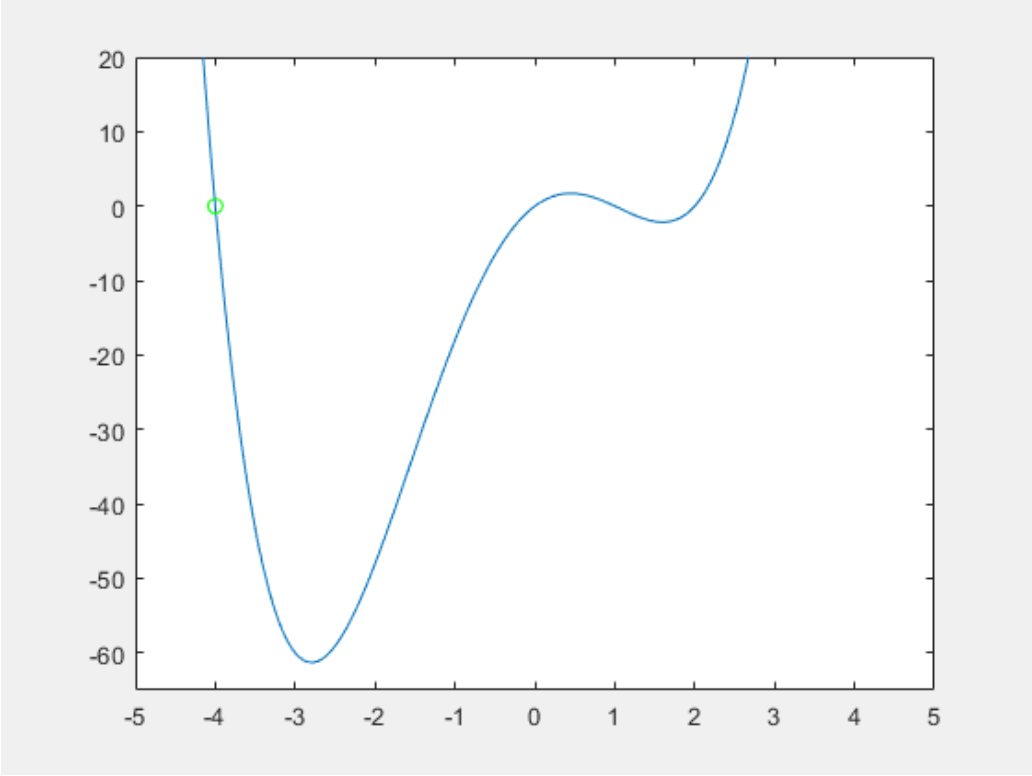
Step size: .2

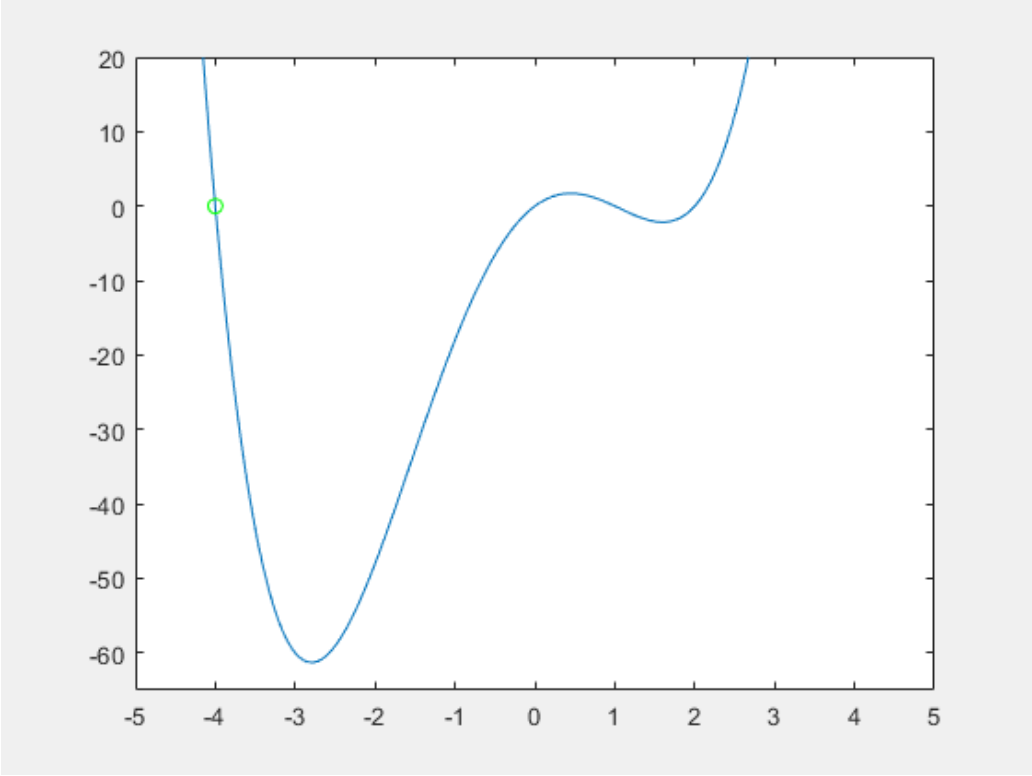


Step size: .9

Gradient Descent

Step size matters!





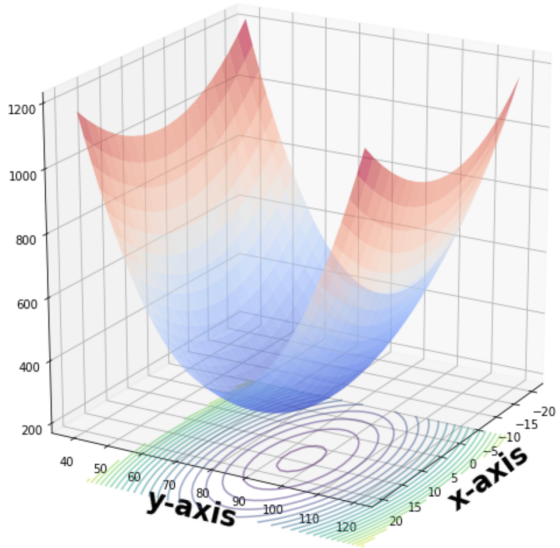
Example: linear regression(MSE loss)

Problem: Given data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$ and $y^1, \ldots, y^n \in \mathbb{R}$, find the best fit line. Consider a simple linear regression model with $\hat{y} = \mathbf{w}^T\mathbf{x} + b$. The formulation can be simplified to $\hat{y} = \boldsymbol{\theta}^T\tilde{\mathbf{x}}$, in which $\boldsymbol{\theta} = [\mathbf{w}^T, b]^T, \tilde{\mathbf{x}} = [\mathbf{x}^T, 1]^T$.

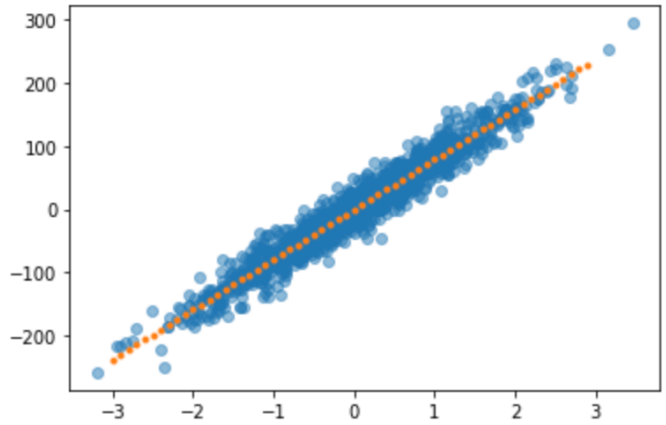
Objective:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{\theta}^T\tilde{\mathbf{x}}_i - y_i)^2$$

Loss Surface:



Data Distribution:



GD result with different step size!

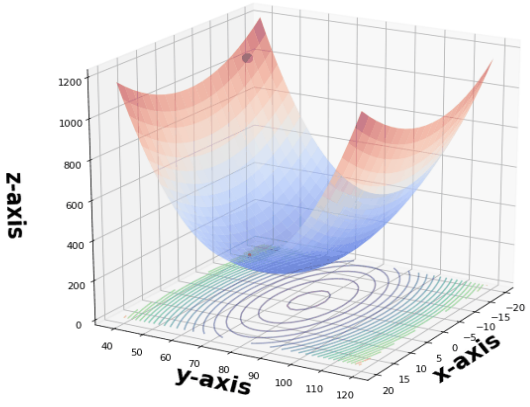
$$\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \gamma \sum_{i=1}^{n} \tilde{\mathbf{x}}_i^T (\boldsymbol{\theta}^T \tilde{\mathbf{x}}_i - y_i), t = 0,1,2,\ldots$$

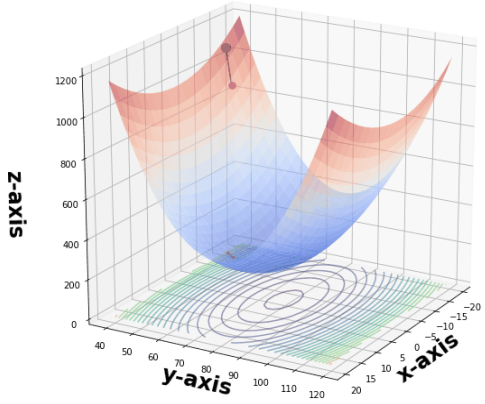
```
pred_y = np.dot(x, theta)
error = pred_y - y
gradient = x.T.dot(error)/sample_number
theta = theta - step_size  * gradient
```

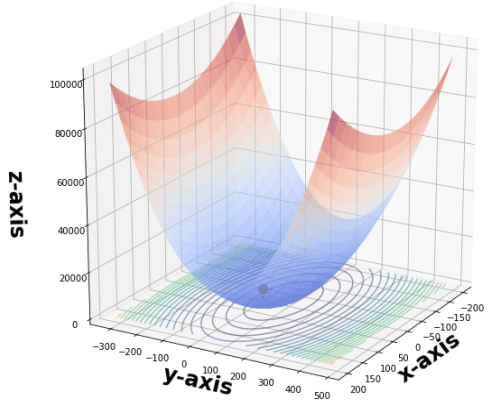
`step_size = 0.5`

`step_size = 0.1`

`step_size = 2.3`







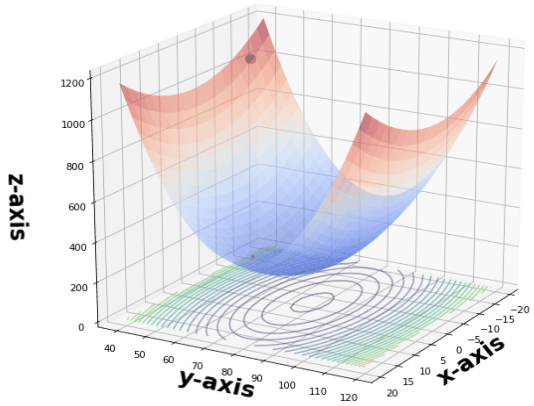
Converge

Slow Converge

Diverge

GD result with different step size!

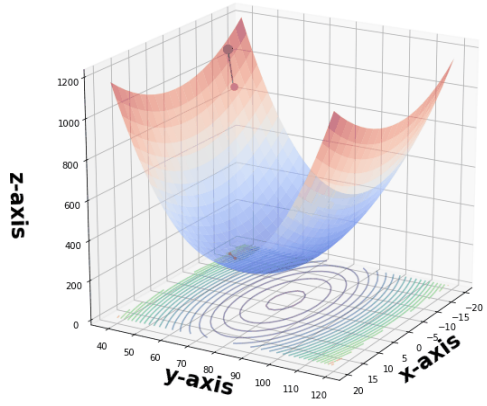
`step_size = 0.5`




```
## Gradient Descent ##
converge to the:
[-0.34,79.17]
iterations :
14 steps
```

Converge

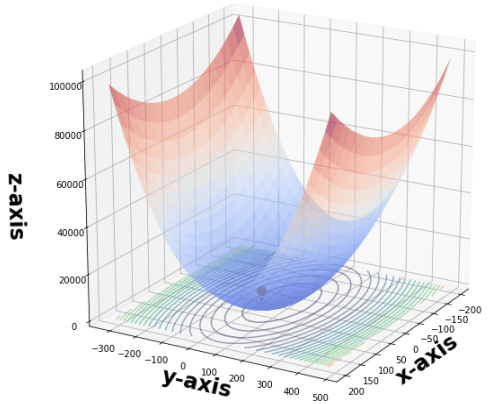
`step_size = 0.1`




```
## Gradient Descent ##
converge to the:
[-0.35,79.15]
iterations :
69 steps
```

Slow Converge

`step_size = 2.3`




```
## Gradient Descent ##
Diverge to the:
[7.55e64,7.31e64]
iterations :
200+ steps
```

Diverge

Convergence of gradient descent

Gradient Descent

Assumption: $f : \mathbb{R}^d \mapsto \mathbb{R}$ is convex, differentiable and has a global minimum $\mathbf{x}^\star$

Goal: Find $\mathbf{x} \in \mathbb{R}^d$ such that

$$f(\mathbf{x}) - f(\mathbf{x}^\star) \leq \epsilon \, .$$

Note that there can be several global minima $\mathbf{x}_1^\star \neq \mathbf{x}_2^\star$ with $f(\mathbf{x}_1^\star) = f(\mathbf{x}_2^\star)$.

Average error:

$$\text{Averaged error} = \frac{1}{T} \sum_{t=0}^{T-1} \left(f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \right)$$

How can we bound the average error?

Warmup

- First let's look at $f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right)$, abbreviating $\mathbf{g}_t := \nabla f\left(\mathbf{x}_t\right)$.
- With $\mathbf{x} = \mathbf{x}_t, \mathbf{y} = \mathbf{x}^\star$

$$f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \mathbf{g}_t^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right)$$

- giving

$$\sum_{t=0}^{T-1} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \sum_{t=0}^{T-1} \mathbf{g}_t^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right)$$

Then?

first-order characterization of convexity

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \forall \mathbf{x}, \mathbf{y}$$

Warmup

As $\mathbf{g}_t := \nabla f\left(\mathbf{x}_t\right)$, from gradient descent, $\mathbf{g}_t = \left(\mathbf{x}_t - \mathbf{x}_{t+1}\right)/\gamma$.

$$\mathbf{g}_t^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right) = \frac{1}{\gamma}\left(\mathbf{x}_t - \mathbf{x}_{t+1}\right)^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right).$$

- Rewrite according to $2\mathbf{v}^\top\mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v}-\mathbf{w}\|^2$

$$\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) = \frac{1}{2\gamma}(\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2)$$

$$= \frac{\gamma}{2}\|\mathbf{g}_t\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2)$$

- Sum this up over the first T iterations:

$$\sum_{t=0}^{T-1} \mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) = \frac{\gamma}{2}\sum_{t=0}^{T-1}\|\mathbf{g}_t\|^2 + \underbrace{\sum_{t=0}^{T-1}\frac{1}{2\gamma}(\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2)}_{\text{Telescoping sum}}$$

$$= \frac{\gamma}{2}\sum_{t=0}^{T-1}\|\mathbf{g}_t\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2 - \|\mathbf{x}_T - \mathbf{x}^\star\|^2)$$

Observation 1

This gives

$$\sum_{t=0}^{T-1} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2 - \|\mathbf{x}_T - \mathbf{x}^\star\|^2)$$

$$\leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2)$$

an upper bound for the total error $f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right)$ over the steps

- last iterate is not necessarily the best one.
- step size is crucial

Lipschitz convex functions: $O(1/\epsilon^2)$ steps

Theorem *Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$ and $\| \nabla f(\mathbf{x}) \| \leq B$ for all $\mathbf{x}$. Choosing the stepsize*

$$\gamma := \frac{R}{B\sqrt{T}}$$

gradient descent yields

$$\frac{1}{T}\sum_{t=0}^{T-1} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{RB}{\sqrt{T}}$$

Assume that all gradients of f are bounded in norm.

f is *B-Lipschitz* with

$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq B\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in X$

$\Leftrightarrow$

$\| \nabla f(\mathbf{x}) \| \leq B \ \forall \mathbf{x} \in X$

Lipschitz convex functions: $O(1/\epsilon^2)$ steps

Proof

- plug $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$ and $\| \nabla f(\mathbf{x}) \| \leq B$ into observation 1:

$$\sum_{t=0}^{T-1} f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2)$$

$$\leq \frac{\gamma}{2} B^2 T + \frac{1}{2\gamma} R^2$$

- choose γ such that

$$q(\gamma) = \frac{\gamma}{2} B^2 T + \frac{1}{2\gamma} R^2$$

is minimized.

- giving $\gamma = \dfrac{R}{B\sqrt{T}}$, and $q(R/B\sqrt{T}) = RB\sqrt{T}$.
- the result follows by dividing $q(\gamma)$ by T.

Lipschitz convex functions: $O(1/\epsilon^2)$ steps

$$T \geq \frac{R^2 B^2}{\varepsilon^2} \quad \Rightarrow \quad \text{average error } \leq \frac{RB}{\sqrt{T}} \leq \varepsilon$$

Advantages:

- dimension-independent (no d in the bound)!
- holds for both average, or best iterate

Smooth functions

Smooth functions

Definition

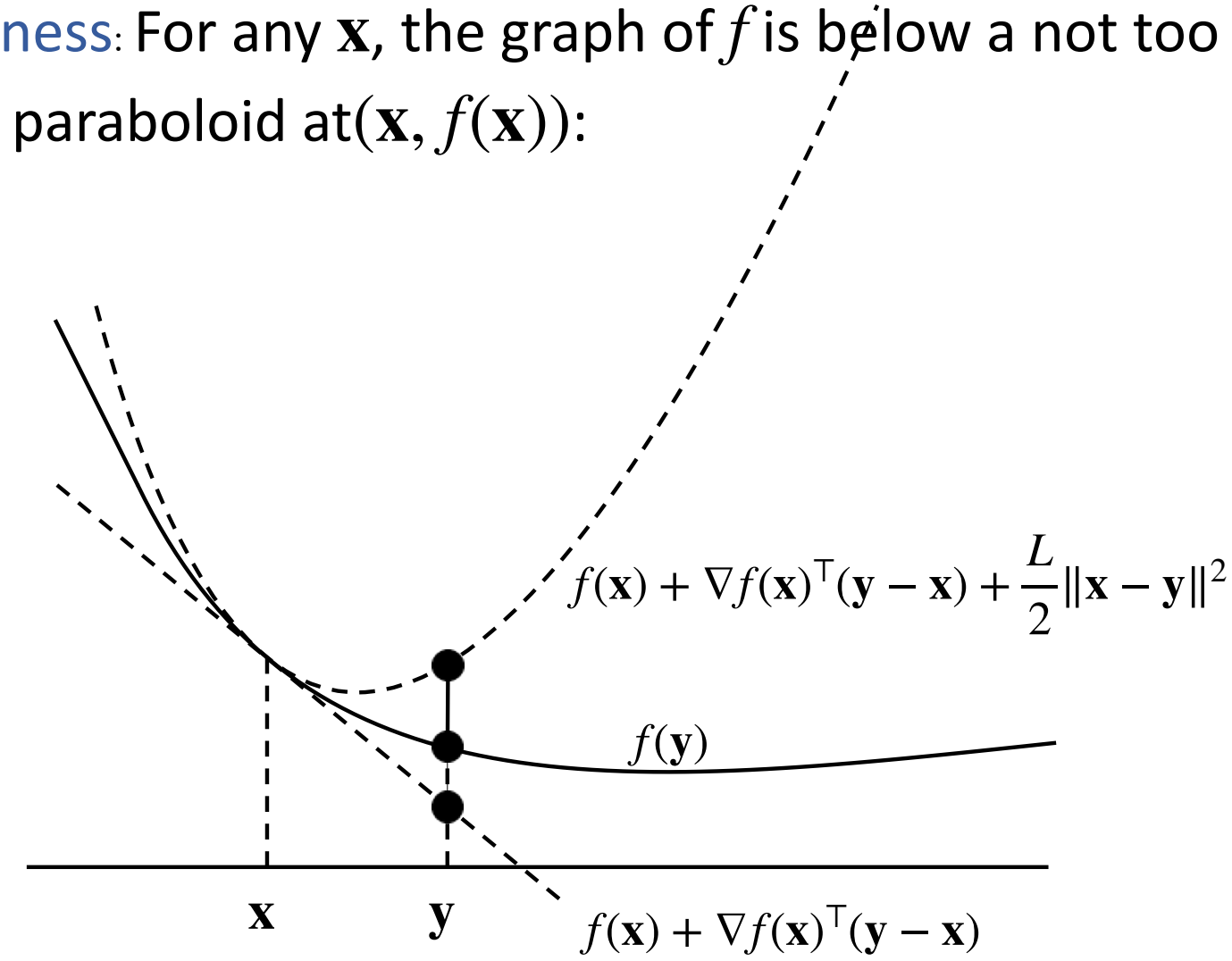
Let $f : \mathrm{dom}(f) \mapsto \mathbb{R}$ be differentiable, $X \subseteq \mathrm{dom}(f), L > 0$. f is called smooth (with parameter L) over X if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^{\top}(\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X$$

- Definition does not require convexity (useful in non-convex world)

Smooth functions

Smoothness: For any $\mathbf{x}$, the graph of f is below a not too steep tangent paraboloid at $(\mathbf{x}, f(\mathbf{x}))$:



Observation 2: Sufficient decrease

Lemma: Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be differentiable and smooth with parameter L. With stepsize

$$\gamma := \frac{1}{L}$$

gradient descent satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0$$

Proof:

Use smoothness and definition of gradient descent ($\mathbf{x}_{t+1} - \mathbf{x}_t = -\nabla f(\mathbf{x})/L$):

$$\begin{aligned} f\left(\mathbf{x}_{t+1}\right) &\leq f\left(\mathbf{x}_t\right) + \nabla f\left(\mathbf{x}_t\right)^{\top}\left(\mathbf{x}_{t+1} - \mathbf{x}_t\right) + \frac{L}{2}\left\|\mathbf{x}_t - \mathbf{x}_{t+1}\right\|^2 \\ &= f\left(\mathbf{x}_t\right) - \frac{1}{L}\left\|\nabla f\left(\mathbf{x}_t\right)\right\|^2 + \frac{1}{2L}\left\|\nabla f\left(\mathbf{x}_t\right)\right\|^2 \\ &= f\left(\mathbf{x}_t\right) - \frac{1}{2L}\left\|\nabla f\left(\mathbf{x}_t\right)\right\|^2 \end{aligned}$$

Smooth convex functions: $O(1/\epsilon)$ steps

Theorem Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that f is smooth with parameter L. Choosing the stepsize

$$\gamma := \frac{1}{L}$$

gradient descent yields

$$f\left(\mathbf{x}_T\right) - f\left(\mathbf{x}^\star\right) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2.$$

Smooth convex functions: $O(1/\epsilon)$ steps

Proof.

Recall: observation 1

$$(1) \sum_{t=0}^{T-1} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2\gamma}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2)$$

The, based on observation 2 (sufficient decrease),

$$(2) \sum_{t=0}^{T-1} \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 \leq \underbrace{\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}))}_{\text{Telescoping sum}} = f(\mathbf{x}_0) - f(\mathbf{x}_T)$$

Observation 2:

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2$$

Putting (1) and (2) together with $\gamma = 1/L$,

$$\sum_{t=0}^{T-1} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{1}{2L} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2}(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2)$$

$$\leq f(\mathbf{x}_0) - f(\mathbf{x}_T) + \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2$$

$$\Updownarrow$$

$$\sum_{t=1}^{T} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2$$

As last iterate is the best (sufficient decrease!)

$$f\left(\mathbf{x}_T\right) - f\left(\mathbf{x}^\star\right) \leq \frac{1}{T} \sum_{t=0}^{T} f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2$$

The result follows!

Smooth convex functions: $O(1/\epsilon)$ steps

Let $R^2 := \left\| \mathbf{x}_0 - \mathbf{x}^\star \right\|^2$, we have

$$T \geq \frac{R^2 L}{2\varepsilon} \quad \Rightarrow \quad \text{error} \leq \frac{L}{2T} R^2 \leq \varepsilon$$

Recall that for Lipschitz-convex

$$T \geq \frac{R^2 B^2}{\varepsilon^2} \quad \Rightarrow \quad \text{average error} \leq \frac{RB}{\sqrt{T}} \leq \varepsilon$$

To achieve an approximation error $\epsilon \leq$ 0.01, we need

- $50 \cdot R^2 L$ iterations for smooth convex function
- $10{,}000 \cdot R^2 B^2$ iterations for Lipschitz convex function

Summary

- Lipschitz convex functions: $O(1/\epsilon^2)$ steps
- Smooth convex functions: $O(1/\epsilon)$ steps

Next

- Stochastic gradient descent
- Non-convex optimization

Questions?

Content

Part 0: Introduction to introduction …

Part 1: Theory of Convexity

Part 2: Gradient Descent and convergence

Part 3: Stochastic Gradient Descent, Non-Convex Optimization, Adam, and Machine Learning Applications

Practical work

https://colab.research.google.com/drive/1r-FTu17Utca7JjblyubSnFEC0n5ph3V6?usp=sharing

Stochastic gradient descent

Stochastic gradient descent

Consider minimizing an average of functions

$$\min f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$$

Stochastic GD:

Choose initial point $\mathbf{x}_0 \in \mathbb{R}^d$, repeat

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f_i\left(\mathbf{x}_t\right), t = 0,1,2,...$$

for **step size** $\gamma \geq 0$, uniformly sampled index $i \in [n]$.

Gradient descent algorithm:

choose initial point $\mathbf{x}_0 \in \mathbb{R}^d$, repeat

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \cdot \frac{1}{n}\sum_{i=1}^{n} \nabla f_i\left(\mathbf{x}_t\right), t = 0,1,2,...$$

for **step size** γ (or learning rate)

Unbiasedness

The vector $\mathbf{g}_t := \nabla f_i(\mathbf{x}_t)$ is call a stochastic gradient

Can't use convexity

$$f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right) \leq \mathbf{g}_t^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right)$$

on top of the **Observation 1**, as this may hold or not hold, depending on how the stochastic gradient $\mathbf{g}_t$ turns out.

- We will show (and exploit): the inequality holds in expectation.

For this, we use that by definition, $\mathbf{g}_t$ is an unbiased estimate of $\nabla f(\mathbf{x}_t)$:

$$\mathbb{E}_{i\in[n]}\left[\mathbf{g}_t \mid \mathbf{x}_t = \mathbf{x}\right] = \frac{1}{n}\sum^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$$

- Main appeal of SGD:
  - Iteration cost is independent of n (number of functions)
  - Can also be a big savings in terms of memory usage
- Further, the below equation holds

$$\mathbb{E}_{i\in[n],\mathbf{x}_t}\left[\mathbf{g}_t^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right)\right] = \mathbb{E}_{\mathbf{x}_t}\left[\nabla f\left(\mathbf{x}_t\right)^\top \left(\mathbf{x}_t - \mathbf{x}^\star\right)\right] \geq \mathbb{E}_{\mathbf{x}_t}[f\left(\mathbf{x}_t\right) - f\left(\mathbf{x}^\star\right)]$$

Bounded stochastic gradients: $O(1/\epsilon^2)$ steps

Theorem Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$ and $\mathbb{E}[\|\mathbf{g}_t\|^2] \leq B^2$ for all $\mathbf{x}$. Choosing the stepsize

$$\gamma := \frac{R}{B\sqrt{T}}$$

gradient descent yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[f(\mathbf{x}_t)] - f\left(\mathbf{x}^\star\right) \leq \frac{RB}{\sqrt{T}}.$$

Same procedure as the proof of gradient descent, except

- we assume bounded stochastic gradients in expectation;
- error bound holds in expectation.

Mini-batch SGD

Instead of using a single element f_i, use an average of several of them:

$$\tilde{\mathbf{g}}_t := \frac{1}{m} \sum_{j=1}^{m} \mathbf{g}_t^j$$

Extreme cases:

m = 1 ⇔ SGD as originally defined

m = n ⇔ full gradient descent

Benefit: Gradient computation can be naively parallelized

Mini-batch SGD

Variance Intuition: Taking an average of many independent random variables reduces the variance. So for larger size of the mini-batch m, $\tilde{\mathbf{g}}_t$ will be closer to the true gradient, in expectation:

$$\mathbb{E}[\|\tilde{\mathbf{g}}_t - \nabla f(\mathbf{x}_t)\|^2] = \mathbb{E}[\|\frac{1}{m}\sum_{j=1}^{m} \mathbf{g}_t^j - \nabla f(\mathbf{x}_t)\|^2]$$

$$= \frac{1}{m}\mathbb{E}[\|\mathbf{g}_t^1 - \nabla f(\mathbf{x}_t)\|^2]$$

$$\text{var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \quad \Rightarrow \quad = \frac{1}{m}\mathbb{E}[\|\mathbf{g}_t^1\|^2] - \frac{1}{m}\|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{B^2}{m}$$

Using a modification of the SGD analysis, can use this quantity to relate convergence rate to the rate of full gradient descent.

Convergence rate

Recall:

- Lipschitz convex functions: $O(1/\epsilon^2)$ steps
- Smooth convex functions: $O(1/\epsilon)$ steps

What about SGD?

- Lipschitz convex functions: $O(1/\epsilon^2)$ steps
- Unfortunately this does not improve when we further assume f has Lipschitz gradient

Non-convex optimization

Nonconvex problem

The problems solved in practice, especially in machine learning/statistics, are mostly convex.

- Linear regression, logistic regression;
- Kernel methods;
- Linear programming, semi-definite programming, SOS (Sum Of Squares programming);

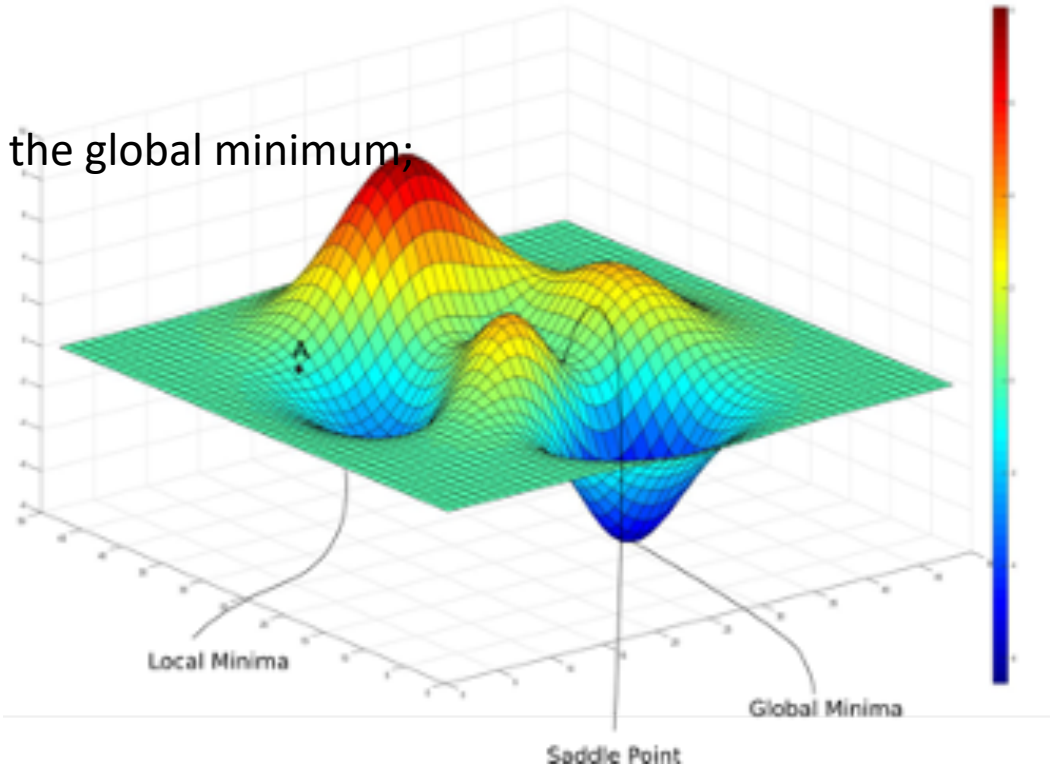
But now, they are mostly non-convex, mainly for one reason:

- Deep learning / Neural networks.

GD may get stuck in a local minimum and miss the global minimum;

- but GD still works well in practice. Why?

Let's look into theoretical explanations!



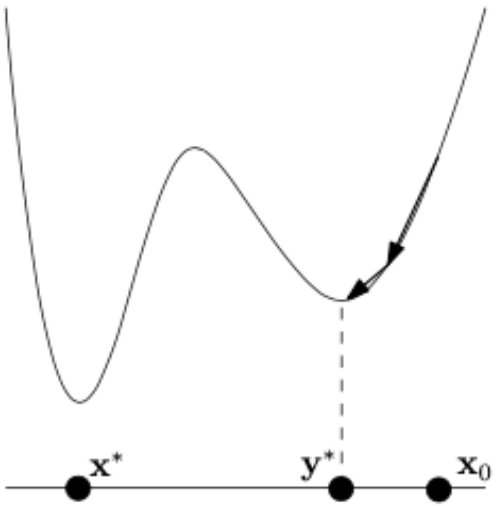
Gradient descent on non-convex problems

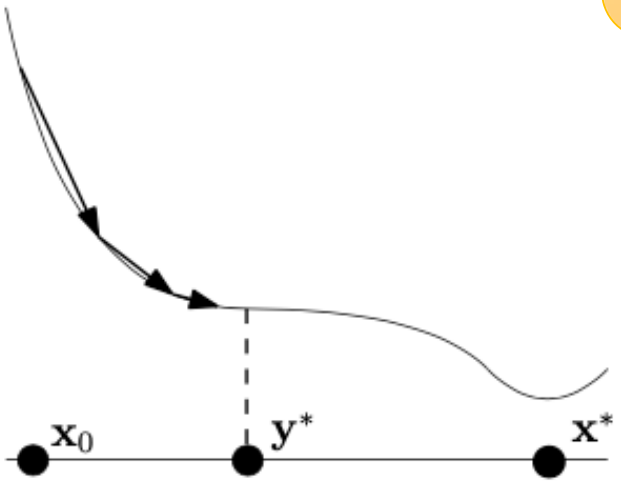
Will prove $\| \nabla f(\mathbf{x}_t) \|_2 \to 0$ for $t \to \infty$ at the same rate as $f(\mathbf{x}_t) - f(\mathbf{x}^\star) \to 0$ in the convex case.

$f(\mathbf{x}_t) - f(\mathbf{x}^\star)$ itself may not converge to 0 in the nonconvex case.

It may or may not mean that we converge to a critical point ($\nabla f(\mathbf{y}^\star) = 0$).

No convexity needed!!





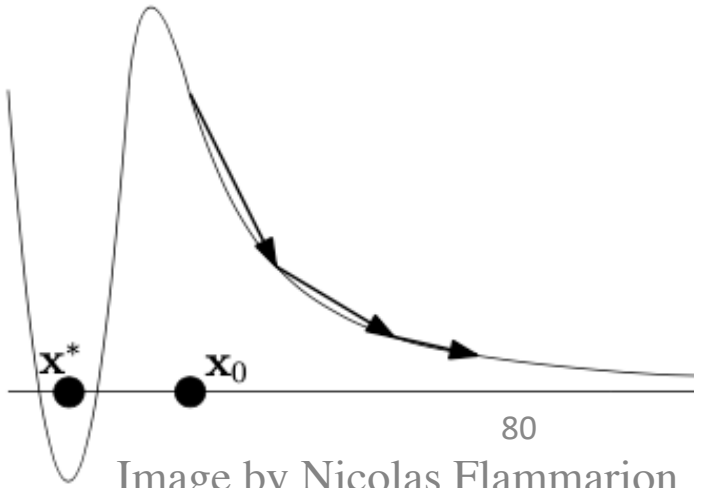


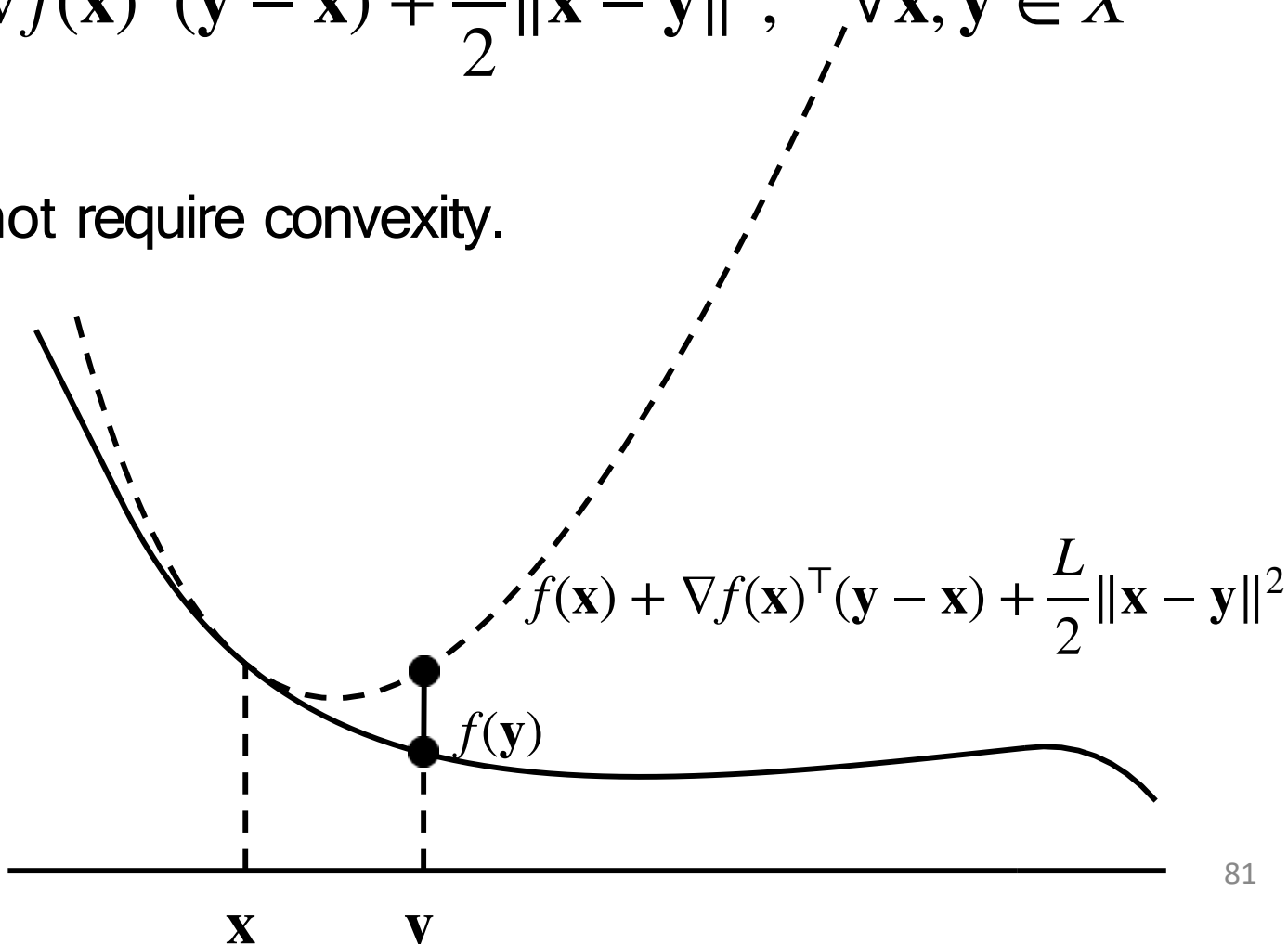
Image by Nicolas Flammarion

Recall: Smooth (but not necessarily convex) functions

Let $f : \text{dom}(f) \mapsto \mathbb{R}$ be differentiable, $X \subseteq \text{dom}(f), L > 0$. f is called smooth (with parameter L) over X if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X$$

Definition does not require convexity.



Smooth gradient descent

Theorem: Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be differentiable with a global minimum $x^\star$; furthermore, suppose that f is smooth with parameter L. Choosing stepsize

$$\gamma := \frac{1}{L}$$

gradient descent yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \left\| \nabla f\left(\mathbf{x}_t\right) \right\|^2 \leq \frac{2L}{T}\left(f\left(\mathbf{x}_0\right) - f\left(\mathbf{x}^\star\right)\right), \quad T > 0.$$

Smooth gradient descent

Proof

Based on observation 2 (sufficient decrease)

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0$$

Rewriting:

$$\|\nabla f(\mathbf{x}_t)\|^2 \leq 2L(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}))$$

Again, we have telescoping sum

$$\begin{aligned}\sum_{t=0}^{T} \|\nabla f(\mathbf{x}_t)\|^2 &\leq \sum_{t=0}^{T} 2L(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})) \\ &= 2L(f(\mathbf{x}_0) - f(\mathbf{x}_T)) \\ &\leq 2L(f(\mathbf{x}_0) - f(\mathbf{x}^\star))\end{aligned}$$

Dividing by T, the proof is completed.

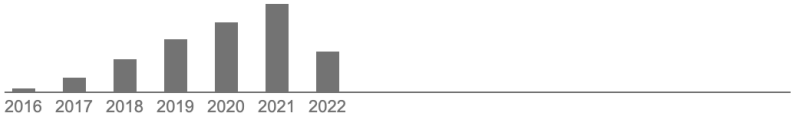
Adaptive methods for SGD

Several methods exist

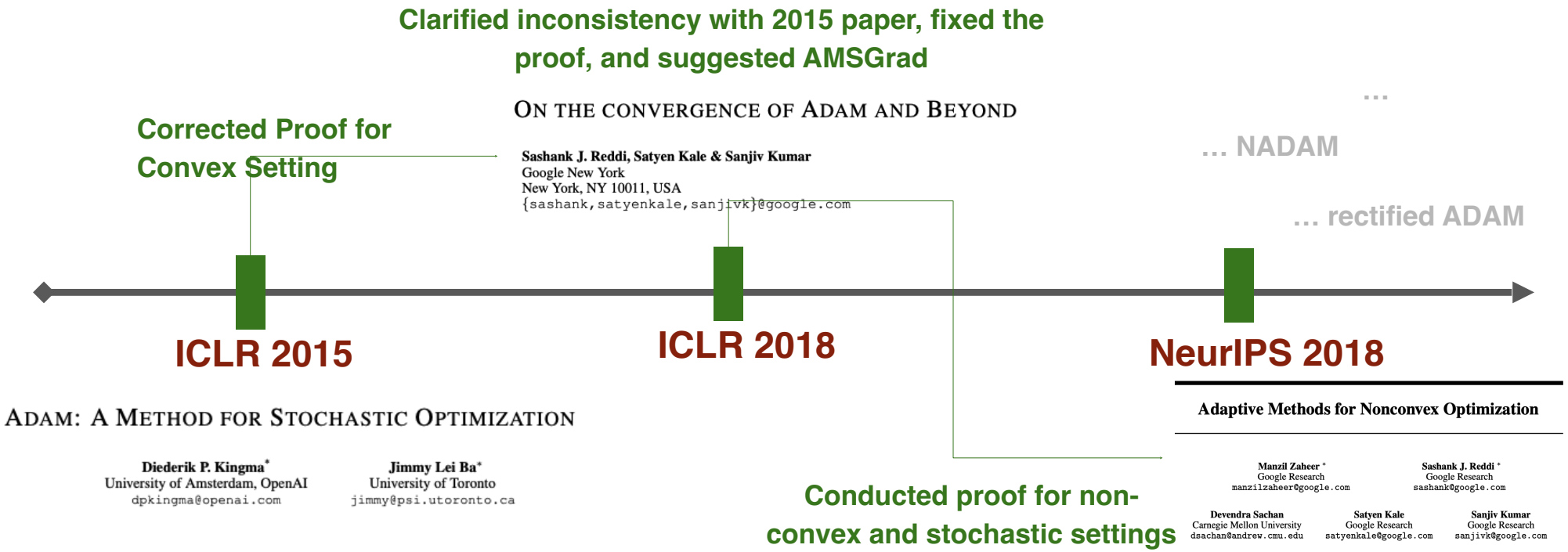
- AdaGrad
- AdaDelta
- RMSProp
- Adam
- …

Adam: A Method for Stochastic Optimization

Authors	Diederik P. Kingma, Jimmy Ba
Publication date	2014/12/22
Journal	Proceedings of the 3rd International Conference on Learning Representations (ICLR)
Description	We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which Adam was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss AdaMax, a variant of Adam based on the infinity norm.
Total citations	Cited by 113113



History of Adam Optimizer



Proposed ADAM and demonstrated a proof which was found to have problems that were corrected in 2018

Proved convergence with increasing batch-sizes, and demonstrated a new algorithm (YOGI) with similar convergence guarantees

Adam

Theorem 1. *Let $\gamma_t = \gamma$ for all $t \in [T]$. Furthermore, assume that ε, β_2 and γ are chosen such that the following conditions satisfied: $\gamma \leq \frac{\varepsilon}{2L}$ and $1 - \beta_2 \leq \frac{\varepsilon^2}{16B^2}$. Then for $\mathbf{x}_t$ generated using* ADAM, *we have the following bound*

$$\mathbb{E} \left\| \nabla f\left(\mathbf{x}_i\right) \right\|^2 \leq O\left(\frac{f(\mathbf{x}_1) - f(\mathbf{x}^\star)}{\gamma T} + \sigma^2\right)$$

where $\mathbf{x}^\star$ is an optimal solution to the problem in (1) *and $\mathbf{x}_i$ is an iterate uniformly randomly chosen from $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. σ^2 is the bound of the on the variance in stochastic gradients. B is upper bound of stochastic gradient norm.*

Adam

Set $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$

for $t = 1$ to T do

Draw a sample $\mathbf{s}_t$.

Compute $\mathbf{g}_t = \nabla \ell\left(\mathbf{x}_t, \mathbf{s}_t\right)$.

$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + \left(1 - \beta_1\right) \mathbf{g}_t$

$\mathbf{v}_t = \mathbf{v}_{t-1} - \left(1 - \beta_2\right)\left(\mathbf{v}_{t-1} - \mathbf{g}_t^2\right)$

$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{m}_t / \left(\sqrt{\mathbf{v}_t} + \epsilon\right)$

end for

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, \mathbf{a}_i, b_i), \text{ with } \ell(\mathbf{x}_i, \mathbf{a}_i, b_i) = \left(b_i - f_{\mathbf{x}}\left(\mathbf{a}_i\right)\right)^2$$

Sample $\xi_t = i_t \in \{1, \ldots, n\}$

$$\Rightarrow \ell\left(\mathbf{x}, i_t\right) = \left(b_{i_t} - f_{\mathbf{x}}\left(\mathbf{a}_{i_t}\right)\right)^2$$

$$\nabla_{\mathbf{x}} \ell\left(\mathbf{x}, i_t\right) = \nabla_{\boldsymbol{\theta}} \left(b_{i_t} - f_{\mathbf{x}}\left(\mathbf{a}_{i_t}\right)\right)^2$$

$$= -2\left(b_{i_t} - f_{\mathbf{x}}\left(\mathbf{a}_{i_t}\right)\right) \nabla f_{\mathbf{x}}\left(\mathbf{a}_{i_t}\right)$$

Adam proof

Key steps in proof (b is minibatch size):

1- assume that f is L-smooth, we have:

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2$$

$$= f(\mathbf{x}_t) - \gamma_t \sum_{i=1}^{d} ([\nabla f(\mathbf{x}_t)]_i \times \frac{g_{t,i}}{\sqrt{v_{t,i}} + \epsilon}) + \frac{L\gamma_t^2}{2} \sum_{i=1}^{d} \frac{g_{t,i}^2}{(\sqrt{v_{t,i}} + \epsilon)^2}$$

2- taking expectation

$$\mathbb{E}[f(\mathbf{x}_{t+1}) \,|\, \mathbf{x}_t] \leq f(\mathbf{x}_t) - \gamma_t \sum_{i=1}^{d} ([\nabla f(\mathbf{x}_t)]_i \times \mathbb{E}_t \left[\frac{g_{t,i}}{\sqrt{v_{t,i}} + \epsilon}\right]) + \frac{L\gamma_t^2}{2} \sum_{i=1}^{d} \mathbb{E}_t \left[\frac{g_{t,i}^2}{(\sqrt{v_{t,i}} + \epsilon)^2}\right]$$

$$\leq f(\mathbf{x}_t) - \frac{\gamma_t}{2(\sqrt{\beta_2}B + \epsilon)} \|\nabla f(\mathbf{x}_t)\|^2 + (\frac{\gamma_t B\sqrt{1-\beta_2}}{\epsilon^2} + \frac{L\gamma_t^2}{2\epsilon^2})\frac{\sigma^2}{b}$$

3-Using telescoping sum and rearranging the inequality, we obtain

$$\frac{\gamma}{2(\sqrt{\beta_2}B + \epsilon)} \sum_{t=1}^{T} \mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 \leq f(\mathbf{x}_1) - \mathbb{E}[f(x_{T+1})] + (\frac{\gamma B\sqrt{1-\beta_2}}{\epsilon^2} + \frac{L\gamma^2}{2\epsilon^2})\frac{T\sigma^2}{b}$$

4- using the fact that $f(\mathbf{x}^\star) \leq f(\mathbf{x}_{t+1})$

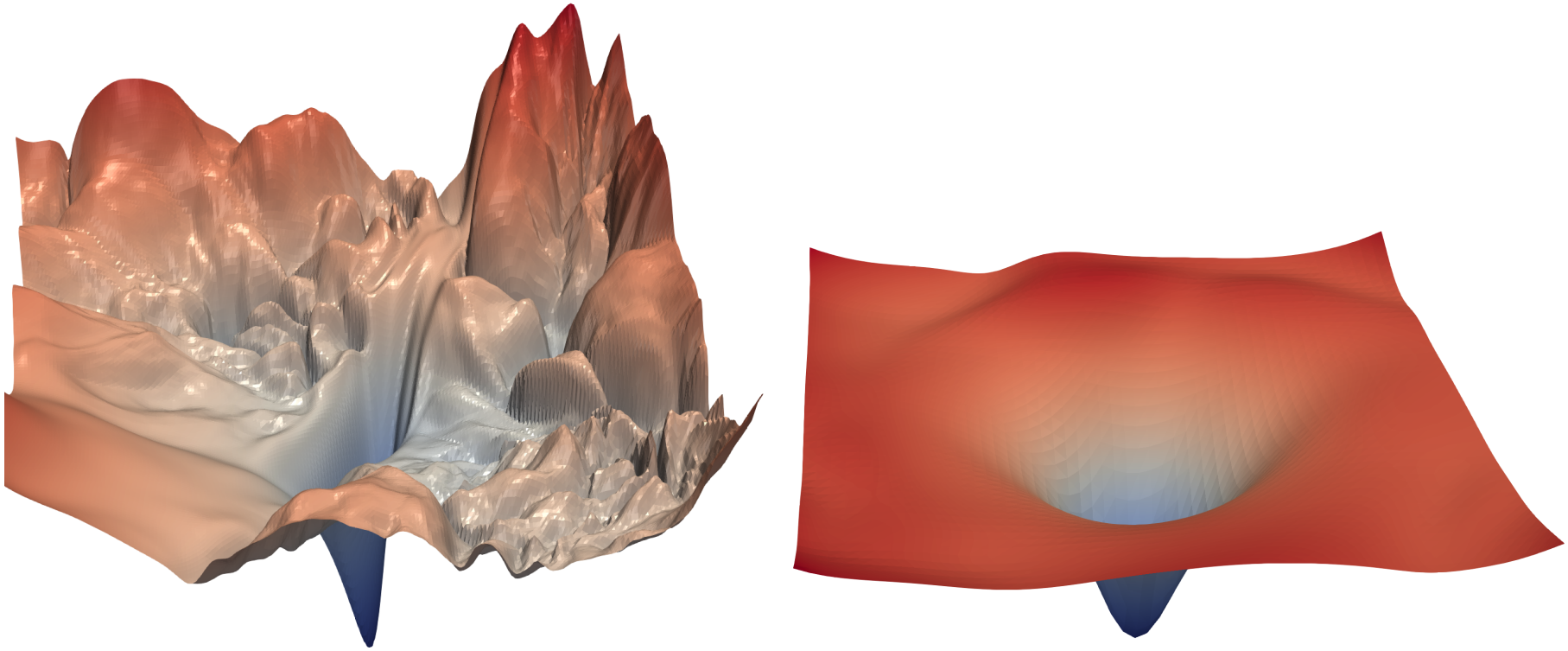
$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla f(\mathbf{x}_t)\|^2 \leq 2(\sqrt{\beta_2}B + \epsilon) \times \left[\frac{f(\mathbf{x}_1) - f(\mathbf{x}^\star)}{\gamma T} + (\frac{B\sqrt{1-\beta_2}}{\epsilon^2} + \frac{L\gamma}{2\epsilon^2})\frac{\sigma^2}{b}\right]$$

Notes:

- $\mathbb{E}[\| \nabla f(\mathbf{x}_i) \|^2] \leq O(1/b)$ as $T \to \infty$. Not necessarily imply that the $\mathbf{x}_i$ is close to a stationary point, but sufficient for many ML applications
- ADAM with $b = \Theta(T) \Rightarrow \mathbb{E}[\| \nabla f(\mathbf{x}_i) \|^2] \leq O(1/T)$, implying converging to the stationary point

[1]Zaheer, Manzil, et al. "Adaptive methods for nonconvex optimization." *Advances in neural information processing systems* 31 (2018).
[2]https://www.youtube.com/watch?v=n65pElMp6x8&ab_channel=MachineLearningandAIAcademy

What one can hope for DNNs?



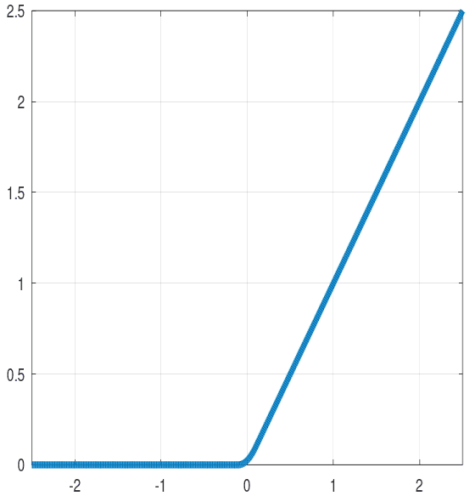
(a) without skip connections

(b) with skip connections

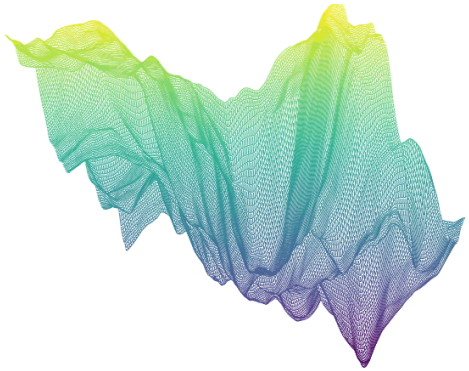
The loss surfaces of ResNet-56 with/without skip connections.

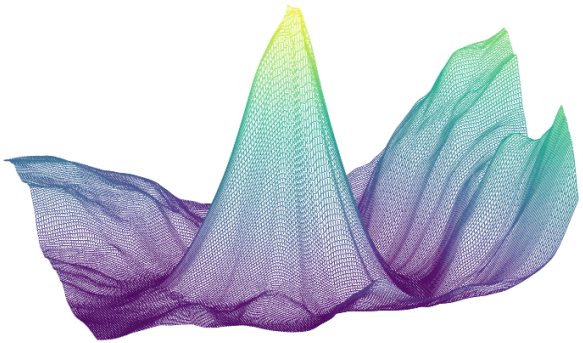
Li, H., et.al., Visualizing the Loss Landscape of Neural Nets. (NeurIPS 2018).

What one can hope for DNNs?



Smoothed ReLU





Loss surfaces (as functions of a 2D input) for two sample loss functions (middle and right) as the activation function's transition region widens, going from from ReLU to an increasingly smoother SmeLU (left).

Shamir, Gil I., and Dong Lin. "Real World Large Scale Recommendation Systems Reproducibility and Smooth Activations." *arXiv preprint arXiv:2202.06499* (2022).

Machine learning applications

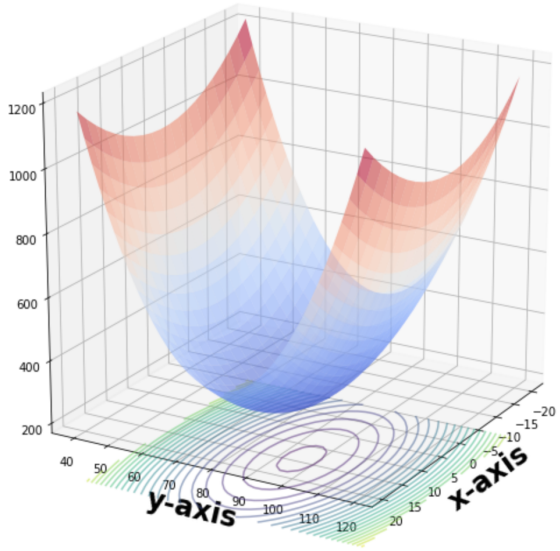
Example: linear regression(MSE loss)

Problem: Given data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$ and $y^1, \ldots, y^n \in \mathbb{R}$, find the best fit line. Consider a simple linear regression model with $\hat{y} = \mathbf{w}^T\mathbf{x} + b$. The formulation can be simplified to $\hat{y} = \boldsymbol{\theta}^T\tilde{\mathbf{x}}$, in which $\boldsymbol{\theta} = [\mathbf{w}^T, b]^T, \tilde{\mathbf{x}} = [\mathbf{x}^T, 1]^T$.

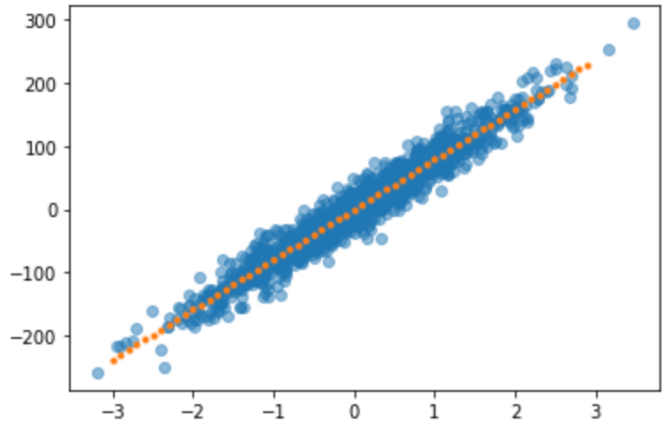
Objective:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{\theta}^T\mathbf{x}_i' - y_i)^2$$

Loss Surface:



Data Distribution:



Example: linear regression(MSE loss)

GD: $\theta_{t+1} := \theta_t - \gamma \sum_{i=1}^{n} \tilde{\mathbf{x}}_i^T(\theta_t^T \tilde{\mathbf{x}}_i - y_i),$

$t = 0,1,2,...$

```
pred_y = np.dot(x, theta)
error = pred_y - y
gradient = x.T.dot(error)/sample_number
theta = theta - step * gradient
```

SGD: $\theta_{t+1} := \theta_t - \gamma \tilde{\mathbf{x}}_i^T(\theta_t^T \tilde{\mathbf{x}}_i - y_i),$

$t \sim \text{uniform}(0,n)$

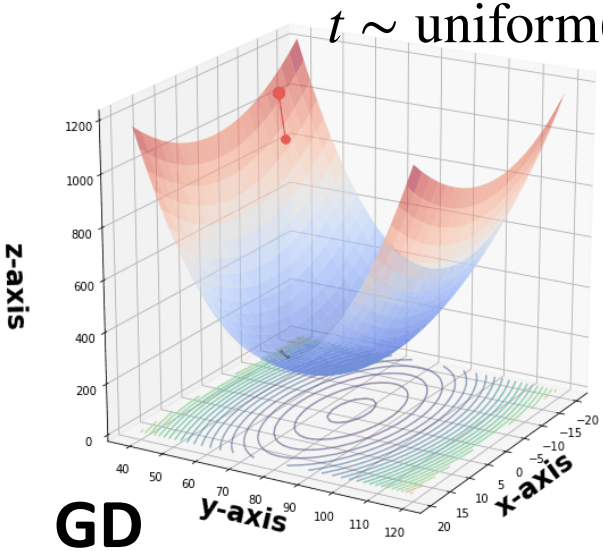
```
pred_y = np.dot(x, theta)
error = pred_y - y
idx = random.sample(range(0, m), 1)
gradient = x[idx].T.dot(error[idx])
theta = theta - step * gradient
```

Mini-SGD:

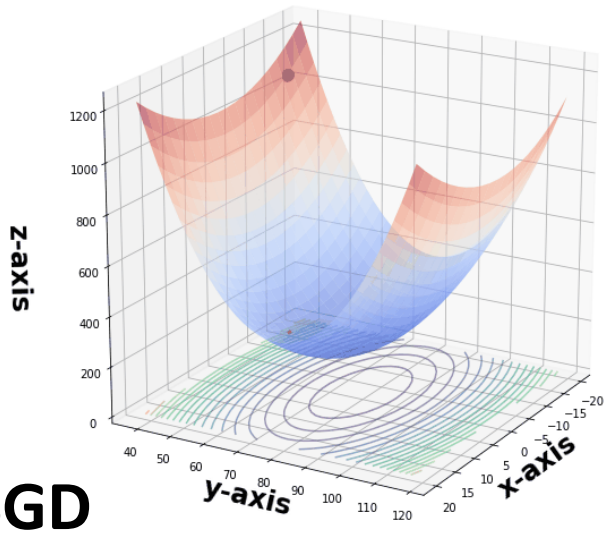
$$\theta_{t+1} := \theta_t - \gamma \sum_{i=1}^{b} \tilde{\mathbf{x}}_i^T(\theta_t^T \tilde{\mathbf{x}}_i - y_i),$$

$t \sim \text{uniform}(0,n)$

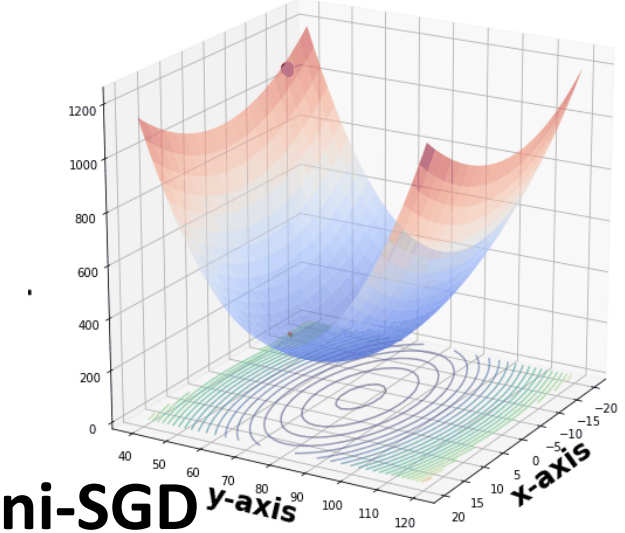
```
pred_y = np.dot(x, theta)
error = pred_y - y
idx = random.sample(range(0, m), batch_size)
gradient = x[idx].T.dot(error[idx])/ batch_size
theta = theta - step * gradient
```



GD



SGD



Mini-SGD

Example: linear regression(MSE loss)

Adam:

Draw a batch of sample .

Compute $\mathbf{g}_t = \sum_{i=1}^{b} \tilde{\mathbf{x}}_i^T(\boldsymbol{\theta}_t^T \tilde{\mathbf{x}}_i - y_i)$

$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\mathbf{g}_t$

$\mathbf{v}_t = \mathbf{v}_{t-1} - (1-\beta_2)\,(\mathbf{v}_{t-1} - \mathbf{g}_t^2)$

(Bias correction)

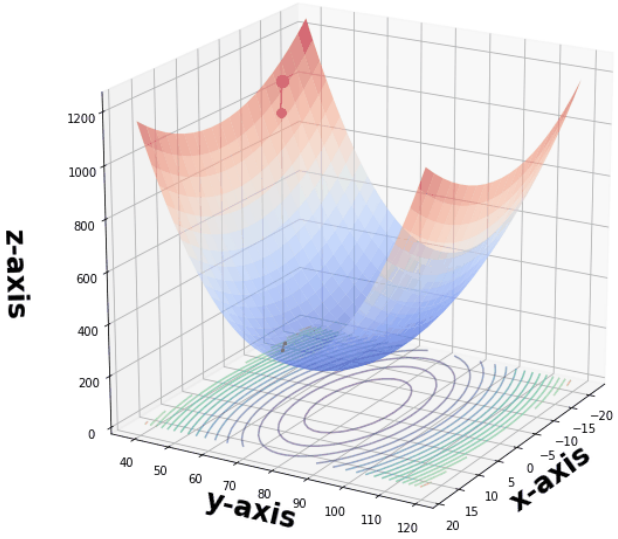
$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{m}_t / \left(\sqrt{\mathbf{v}_t} + \epsilon\right)$

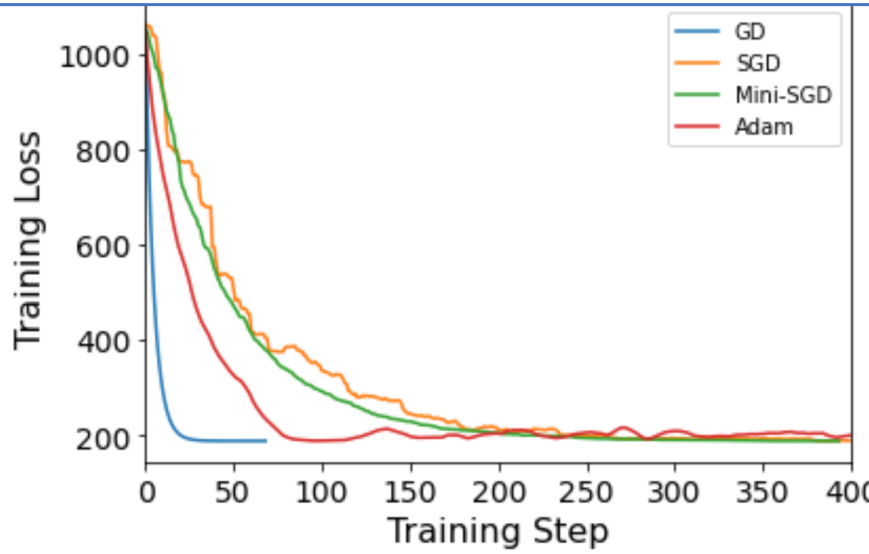
```
pred_y = np.dot(x, theta)
error = pred_y - y
idx = random.sample(range(0, m), batch_size)
batch_gradient = x[idx].T.dot(error[idx])/ batch_size

gred_m = beta1 * gred_m + (1-beta1) * batch_gradient
gred_n = beta2 * gred_n + (1-beta2) * batch_gradient**2

gred_m_adaptive = gred_m/(1-beta1**counter)
gred_n_adaptive = gred_n/(1-beta2**counter)
gradient = gred_m_adaptive/ (np.sqrt(gred_n_adaptive) +
epsilon)

theta = theta - step * gradient
```





Example: Logistic regression(MSE loss)

Problem： Given data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$ and labels $y^1, \ldots, y^n \in \{0,1\}$, find the best classifier.

Denote $\mathbf{x} = [x_1, x_2]^T$. Consider a sigmoid function that predict the soft label:

$$\sigma(\boldsymbol{\theta}^T \tilde{\mathbf{x}}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T \tilde{\mathbf{x}})}, \text{ where } \boldsymbol{\theta} = [1, w, b]^T, \tilde{\mathbf{x}} = [x_1, x_2, 1]^T.$$

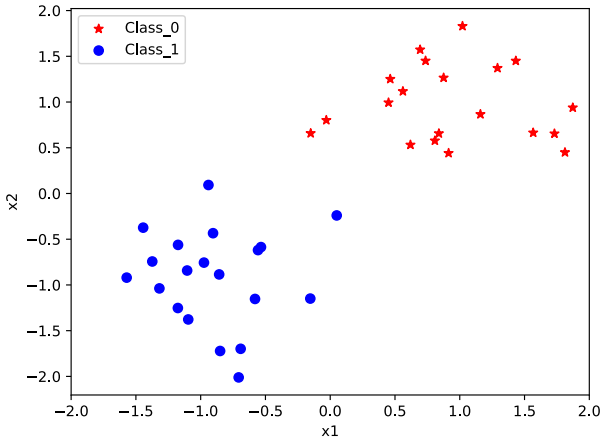
Objective:

$$\min_{w,b} f(w,b) = \frac{1}{2n} \sum_{i=1}^{n} (\sigma(\boldsymbol{\theta}^T \tilde{\mathbf{x}}_i) - y_i)^2$$

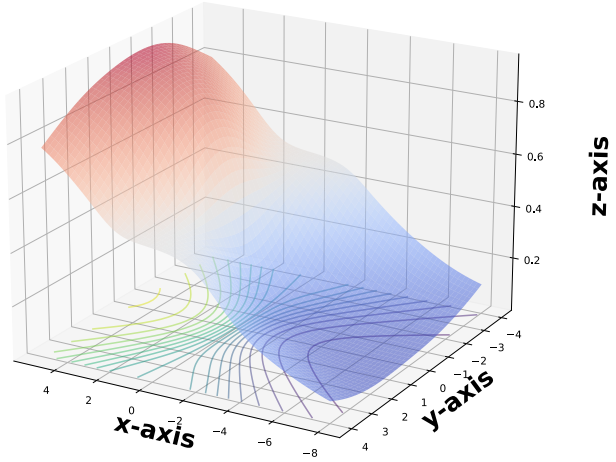
Gradient $\nabla \ell_i(\theta) = \tilde{\mathbf{x}}_i^T (\sigma(\boldsymbol{\theta}_t^T \tilde{\mathbf{x}}) - y_i)\sigma(\boldsymbol{\theta}_t^T \tilde{\mathbf{x}}_i)(1 - \sigma(\boldsymbol{\theta}_t^T \tilde{\mathbf{x}}_i)))$

Derivative of the Sigmoid Function
$\sigma'(x) = \sigma(x)(1-\sigma(x))$

Data Distribution:

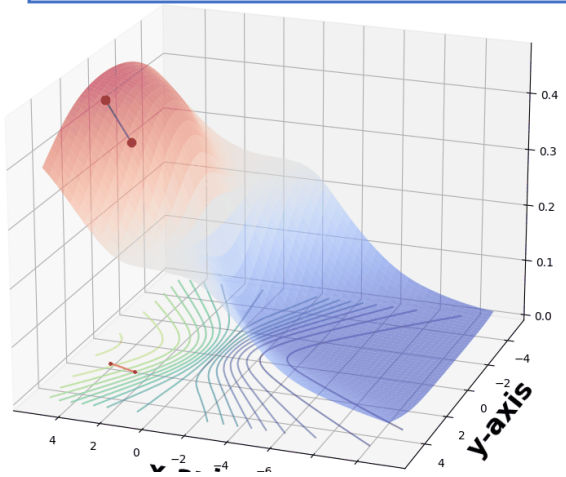


Loss Surface:

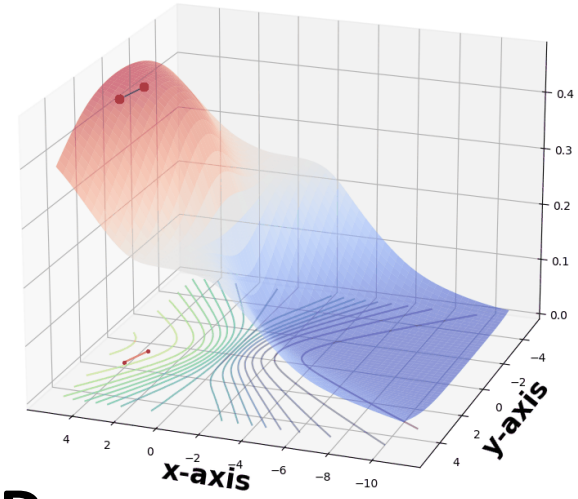


Example: Logistic regression(MSE loss)

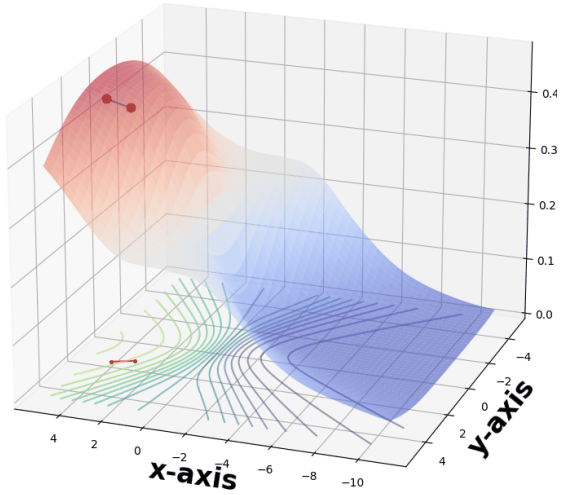
```
step_size = 1 for all algorithms
```

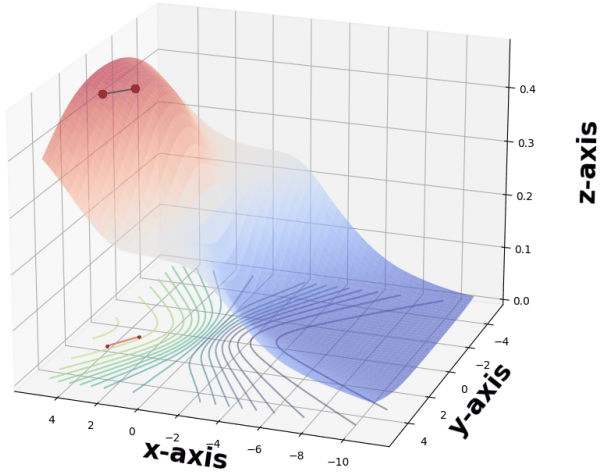


GD

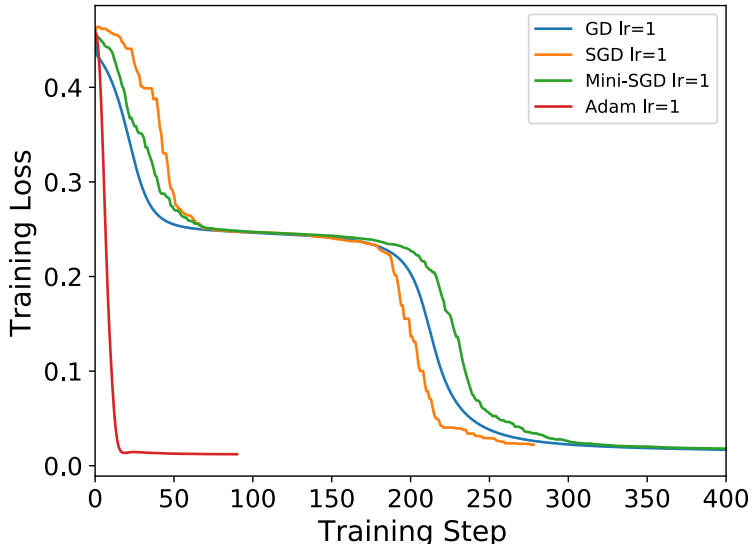


SGD



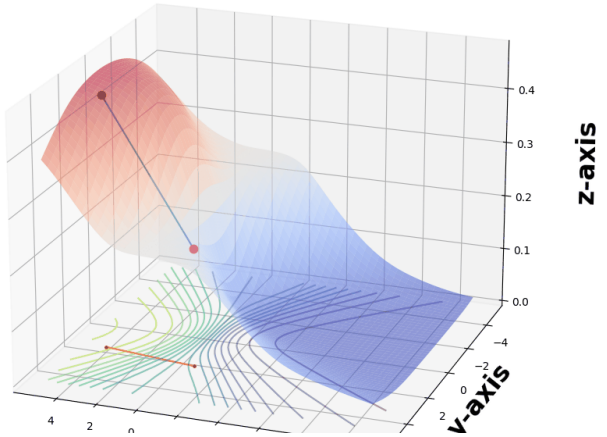


Adam

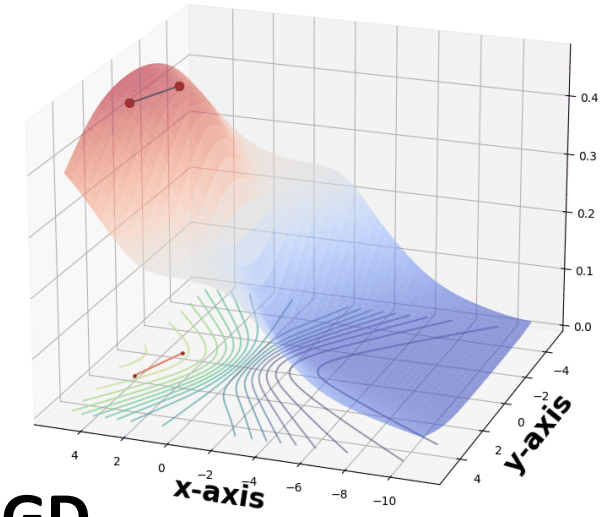


Example: Logistic regression(MSE loss)

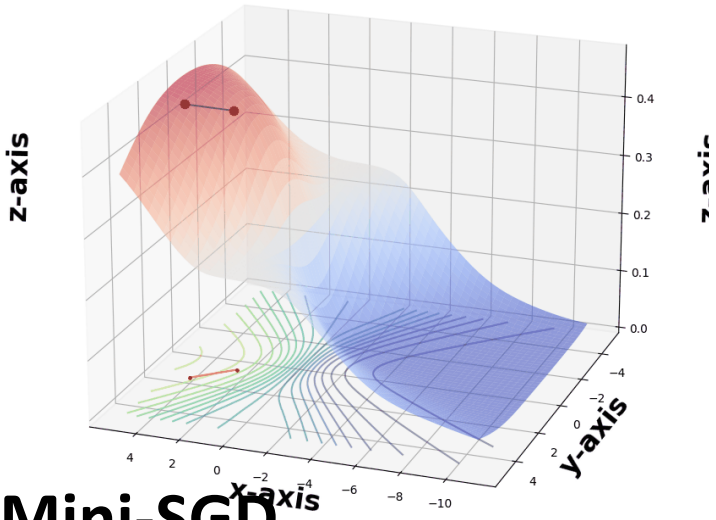
```
step_size = 2 for all algorithms
```



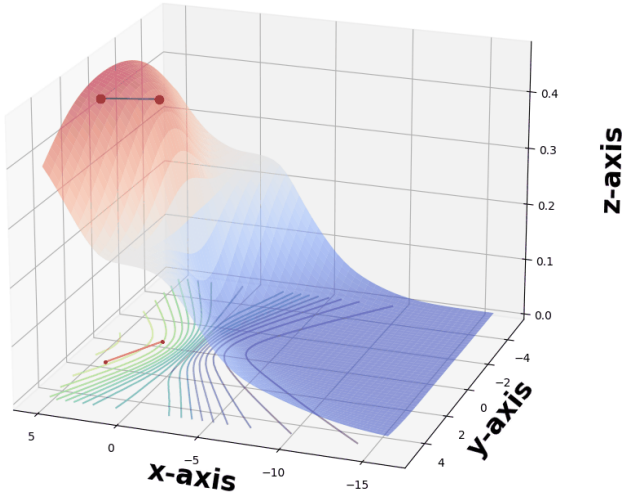
GD



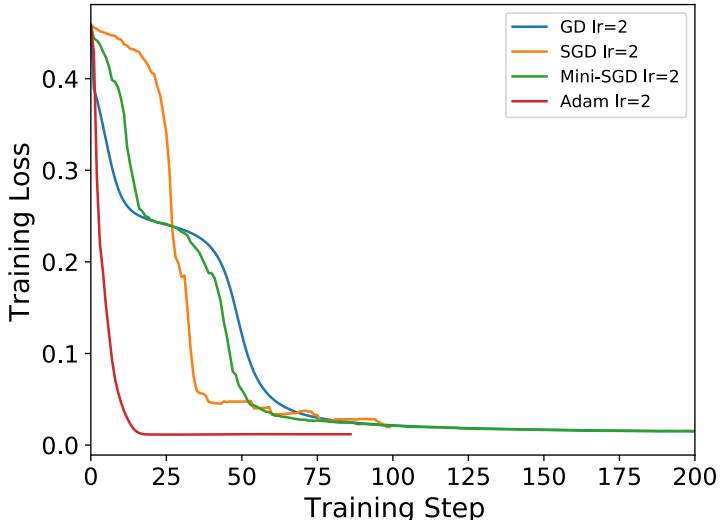
SGD



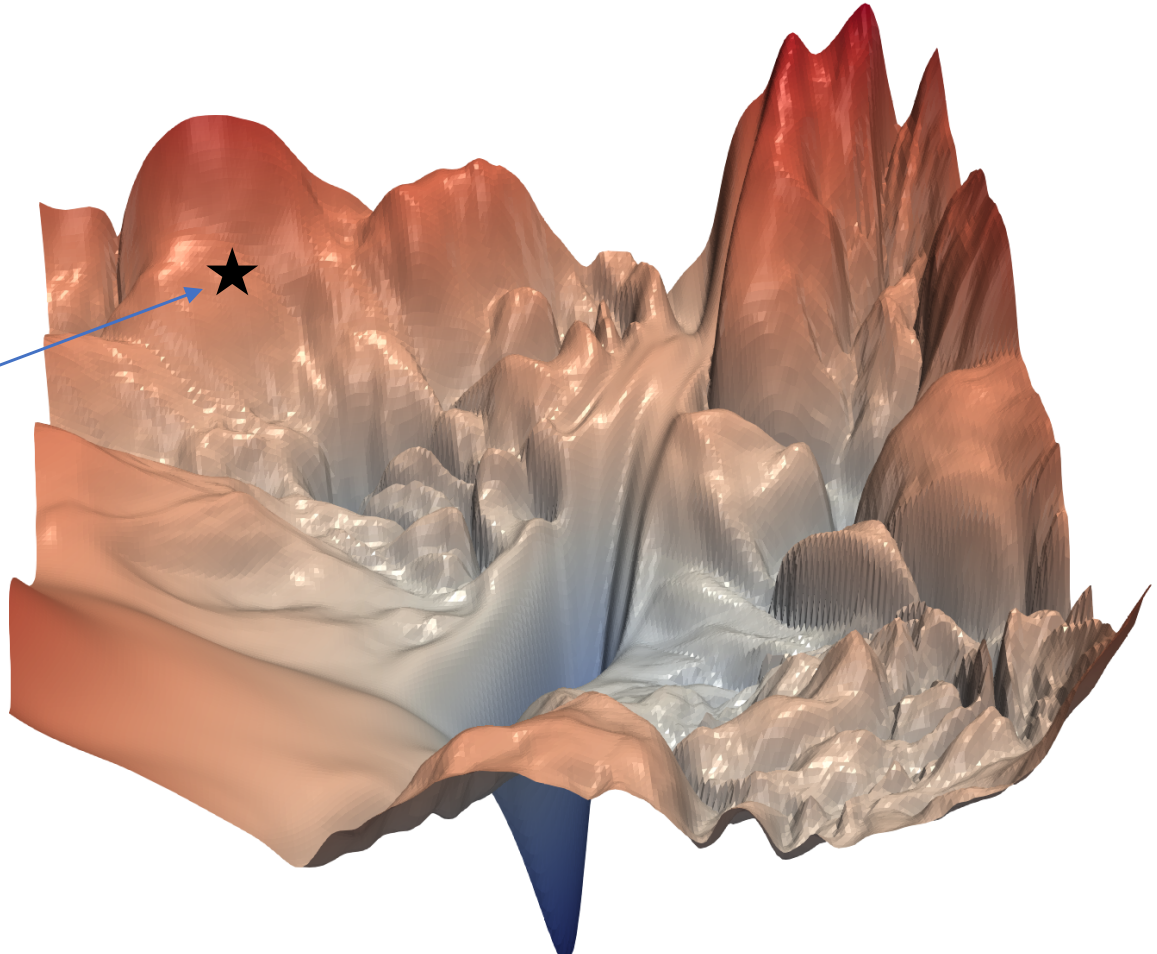
Mini-SGD



Adam



We are here!



Thank you!

https://validu.github.io/