

Optimization for Machine Learning

Aymeric Dieuleveut

January 22, 2026

Teaching team

Lectures

- ① Aymeric Dieuleveut: main teacher optimization.
- ② Adrien Taylor: main teacher deep learning, practicals

Practicals

- ① Adrien Taylor
- ② Pascal Bianchi
- ③ Baptiste Goujaud
- ④ Daniel Berg Thomsen

Rooms: PC 16-19 this afternoon, BEM afterwards

Ressources

Information on Moodle: → moodle

APM_52445_EP - Optimization / Deep Learning (2024-2025).

Communication on Slack (corrections, etc.) and interactive polls via Wooclap

1 Slack: join link. ←

2 Wooclap:

Also shared on Moodle.

Général Tout replier

Welcome to this class about Deep Learning and Optimization composed of seven sessions/days. Each session is divided into a lecture (morning) and a lab session (afternoon).

Lectures
Lectures be three-hours lecture from 9:30 to 12:30, labs will take place between 2pm and 5pm.

Labs
Lab sessions will occur from 2pm to 5pm. We will use Python, PyTorch and Keras depending on the aim of the lab.

Evaluation
Evaluation will be composed of two grades:
- **mandatory presence to the lectures/labs** and participation to Wooclap (between 0% and 20%)
- **a lab session** (50% of the final grade): you will be warned in advance and you will have a few more days after the lab session to submit your lab (to be done by groups of two students).
- **a final written exam** that will occur on the week of the 17th of March (between 50% and 70% of the grade). You have examples of typical questions below.

Slack.
We will use slack to communicate between sessions (no email). You can use slack to ask your questions between the sessions. You can also use slack when you watch the videos of the course to ask your questions. Slack allows us to organize questions and answers (you can reply to a question, which automatically creates a thread, you can put smiley on questions to agree, disagree... and you can also send private messages).
Installing slack: cf web
Tutorial on slack:



Planning

Lec 1 , Lec

- ① Thu 22 Jan – Lecture 1 – Optimization
- ② Wed 28 Jan – Lecture 2 – Deep Learning
- ③ Wed 4 Feb – Lecture 3 – Optimization
- ④ Wed 11 Feb – Lecture 4 – Deep Learning
- ⑤ Wed 18 Feb – Lecture 5 – Deep Learning
- ⑥ Wed 25 Feb – Lecture 6 – Deep Learning
- ⑦ Wed 4 Mar no lecture
- ⑧ Wed 11 Mar – Lecture 7 – Optimization

Recap +
/ last year's exam

Planning

- ① Thu 22 Jan – Lecture 1 – Optimization
- ② Wed 28 Jan – Lecture 2 – Deep Learning
- ③ Wed 4 Feb – Lecture 3 – Optimization
- ④ Wed 11 Feb – Lecture 4 – Deep Learning
- ⑤ Wed 18 Feb – Lecture 5 – Deep Learning
- ⑥ Wed 25 Feb – Lecture 6 – Deep Learning
- ⑦ Wed 4 Mar no lecture
- ⑧ Wed 11 Mar – Lecture 7 – Optimization

*Todo: check
exam date.*

Grading:

- ① Attendance, lab participation
- ② Graded Lab at home (groups of 3 students)
- ③ Exam

Outline

- 1 Optimization for machine learning: motivation and course objectives
 - General introduction: what is optimization and why study it?
 - Approach: what makes optimization difficult?
 - From supervised machine learning to optimization.
 - Summary: approach and lecture's goals
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method

Bul?

GD vs SGD

Outline

- 1 Optimization for machine learning: motivation and course objectives
 - General introduction: what is optimization and why study it?
 - Approach: what makes optimization difficult?
 - From supervised machine learning to optimization.
 - Summary: approach and lecture's goals
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method

Optimization: finding the minimal (maximal) value of a function over a set

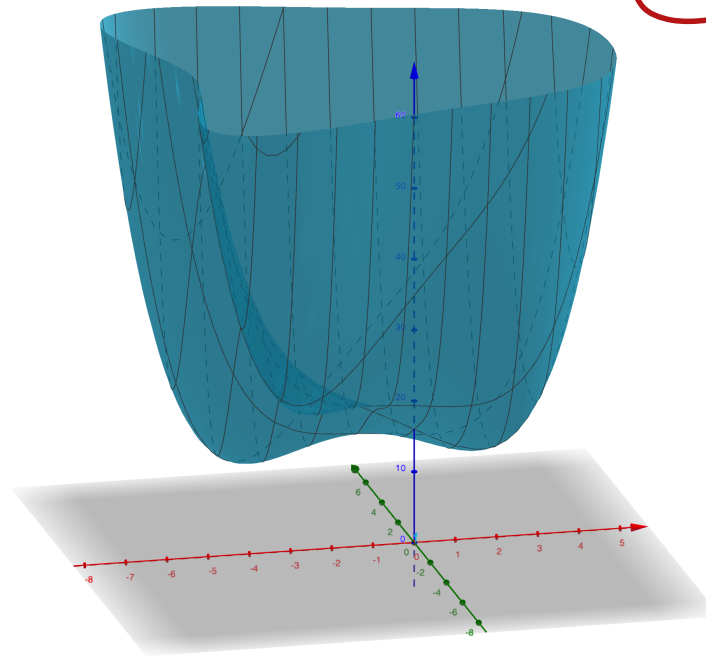
① is parameter space

function

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$$

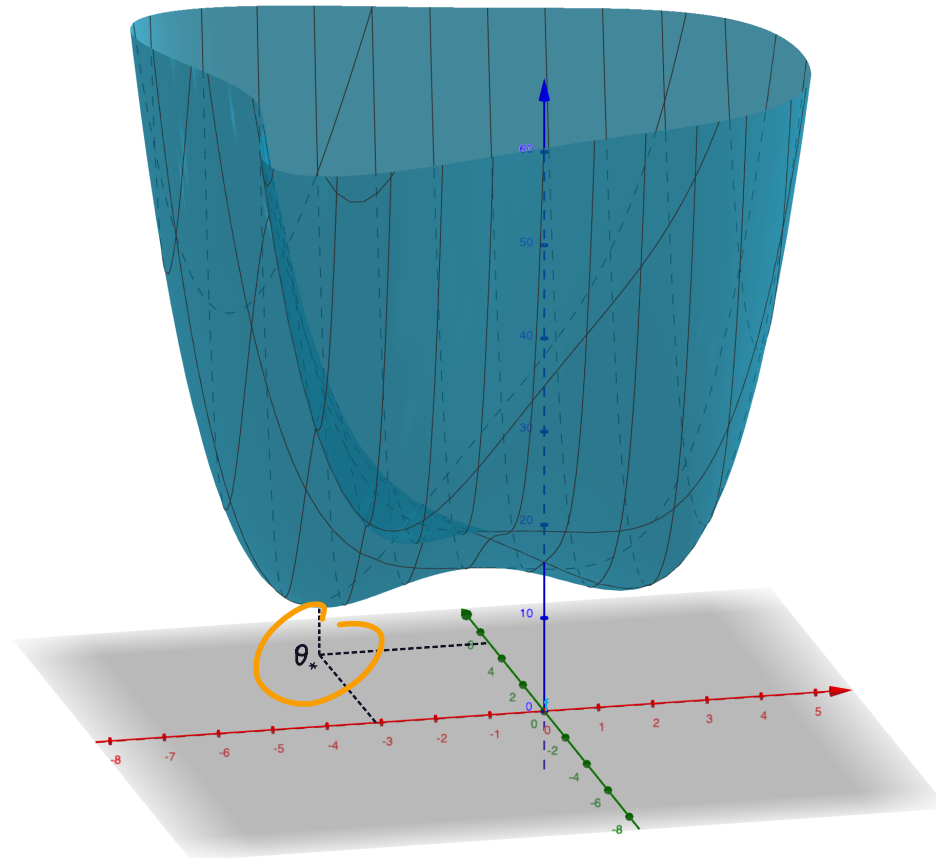
parameter — weights in a NN

minimize



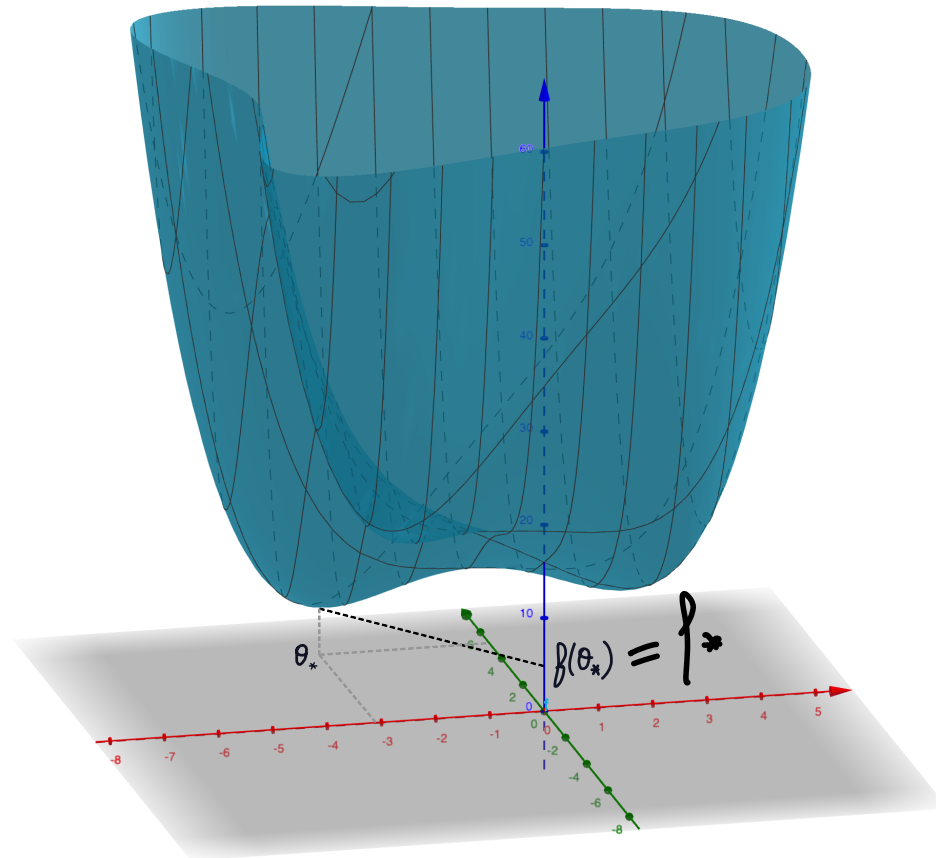
Optimization : finding the minimal (maximal) value of a function over a set

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$$



Optimization : finding the minimal (maximal) value of a function over a set

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$$



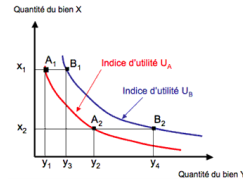
$(f_*, \theta_*$ may
or may not
exist)

Optimization is everywhere

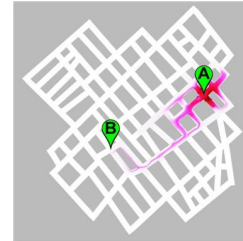
Many problems are formalized as finding the **optimum** of a function: $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$

1. In various domains:

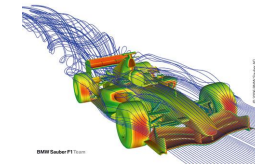
Economics



GPS

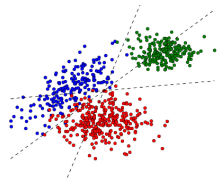


Aeronautics

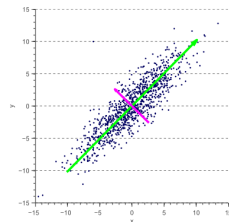


2. In Machine learning related applications

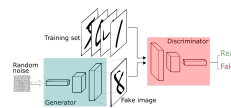
Supervised Learning



Unsupervised

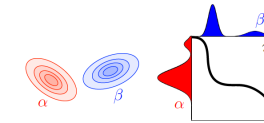


Gans



L_2 min max
optim pb

Optimal transport



Study of moving pts distributo
optimally.

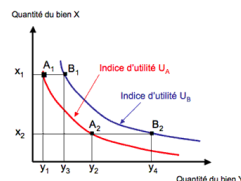
our main focus
yet what we say may apply everywhere

Optimization is everywhere

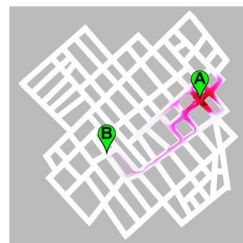
Many problems are formalized as finding the **optimum** of a function: $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$

1. In various domains:

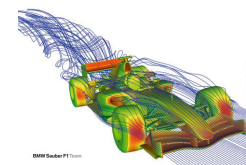
Economics



GPS

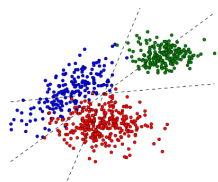


Aeronautics

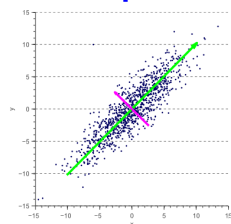


2. In Machine learning related applications

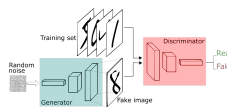
Supervised Learning



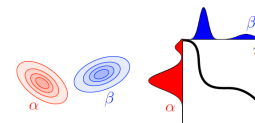
Unsupervised



Gans



Optimal transport



In these sessions:

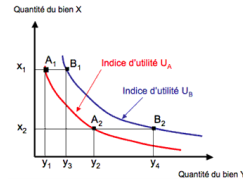
- ① We will consider the minimization of a generic function f .
- ② With the supervised learning case as our main motivation.

Optimization is everywhere

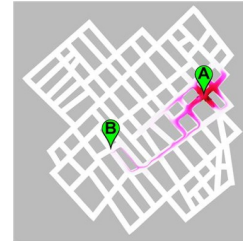
Many problems are formalized as finding the **optimum** of a function: $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$

1. In various domains:

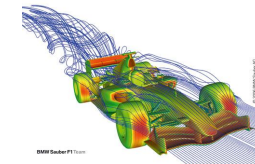
Economics



GPS

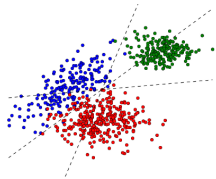


Aeronautics

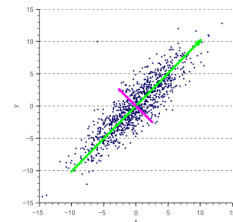


2. In Machine learning related applications

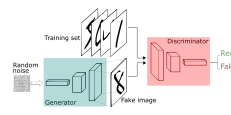
Supervised Learning



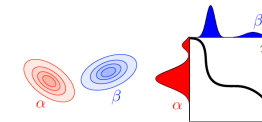
Unsupervised



Gans



Optimal transport



In these sessions:

- 1 We will consider the minimization of a generic function f .
- 2 With the supervised learning case as our main motivation.

→ Is it difficult?
→ Why study it?

Is Optimization a (hard) problem? Why study it?

↪ It depends!



Is Optimization a (hard) problem? Why study it?

↪ It depends!

Yet, important to understand the
methods

Is Optimization a (hard) problem? Why study it?

↪ It depends!

Yet, important to understand the methods

The problem is hard to solve

The choice of the algorithm impacts the performance

⇒ Crucial to understand the algorithms!

e.g. you never actually get
the exact min

!.

Is Optimization a (hard) problem? Why study it?

↪ It depends!

Yet, important to understand the methods

The problem is hard to solve

The choice of the algorithm impacts the performance

⇒ Crucial to understand the algorithms!

Last 20 years in ML?

- 1 More computational power
- 2 More data
- 3 New algorithms & models

↔ Large scale framework → nb of parameters

↔ Deep Learning 8 - 1000 B
→ nb of layers

huge

~ 15 T

Goal of the sessions on optimization

Depending on the characteristics of the problem:

- Understand if optimization is likely to be challenging
- If so, understand **which algorithm to choose**.

To do so:

- Understand what makes optimization hard!
- Relate supervised learning problems to optimization.
- Discover algorithms:
 - high-level intuition
 - their features and strengths, complexity
 - some mathematical quantification of their performance
 - Insights on how to tune the hyperparameters.

Outline

- 1 Optimization for machine learning: motivation and course objectives
 - General introduction: what is optimization and why study it?
 - Approach: what makes optimization difficult?
 - From supervised machine learning to optimization.
 - Summary: approach and lecture's goals
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method

What makes optimization easy or difficult?

what is $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta)$

$\rightarrow n$ can be huge
1 funcⁿ per observ.

f is often the empirical Risk

multiple minimizers
x convexity vs non convexity

regularity
is the funcⁿ
differentiable, smooth,
strongly conv, ...

$\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$

nb of parameters
 d can be huge

is $\Theta = \mathbb{R}^d$

or are there

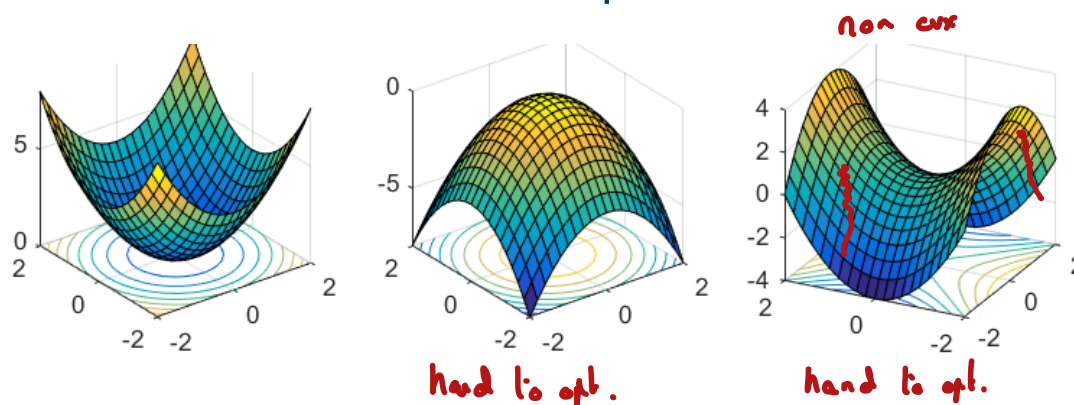
constraints - $\Theta = B(0,1)$?

$$\Theta = \{ \theta / \|\theta\| = 1 \}$$

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

1. Shape of f - convexity.

Informally: f is convex if it looks like the first plot on the left.



Why does convexity help?

- A non-convex function can have many local minima
- For a convex function, a local minimum is always global.
- Local information provides global information!

Challenges: Non-convexity, ...

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

2. Regularity or irregularity of the function

a. Smoothness

∇f is L-Lipschitz.

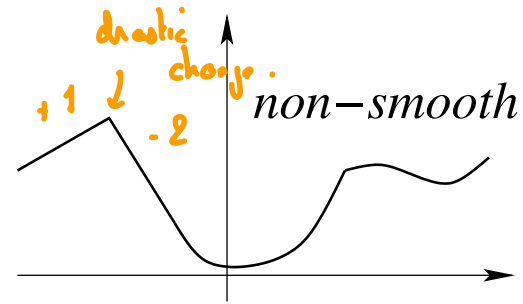
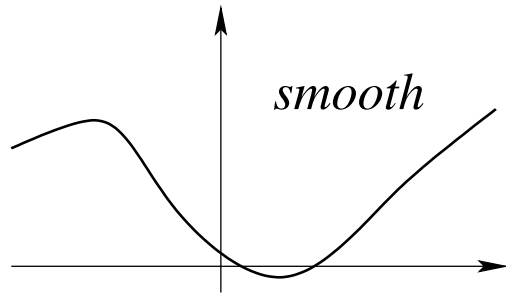
- *Informally*: A function f is **smooth** if its derivative is regular enough.

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

2. Regularity or irregularity of the function

a. Smoothness

- *Informally*: A function f is **smooth** if its derivative is regular enough.

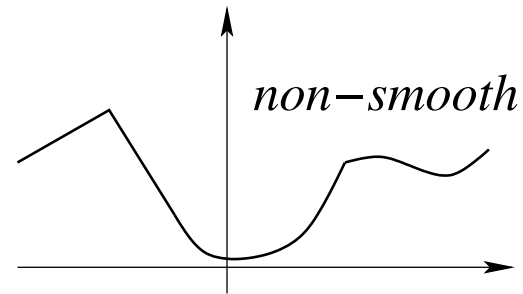
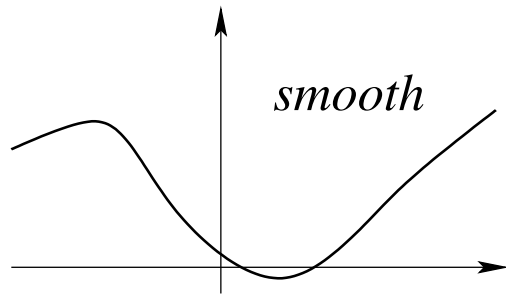


What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

2. Regularity or irregularity of the function

a. Smoothness

- *Informally:* A function f is **smooth** if its derivative is regular enough.



b. Strong Convexity

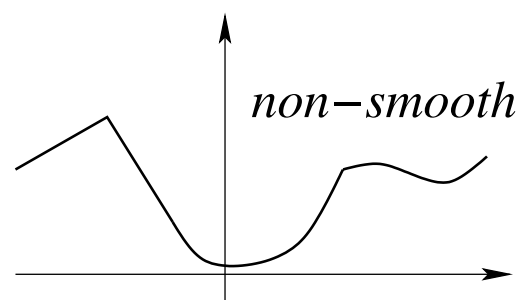
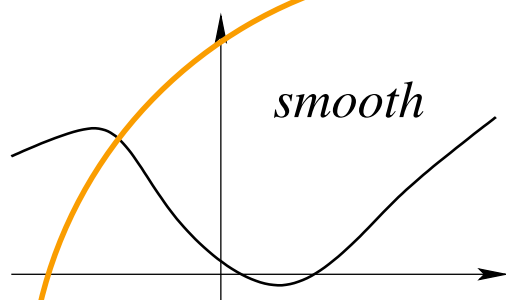
- *Informally:* f is **strongly convex** if it has lower bounded a curvature.

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

2. Regularity or irregularity of the function

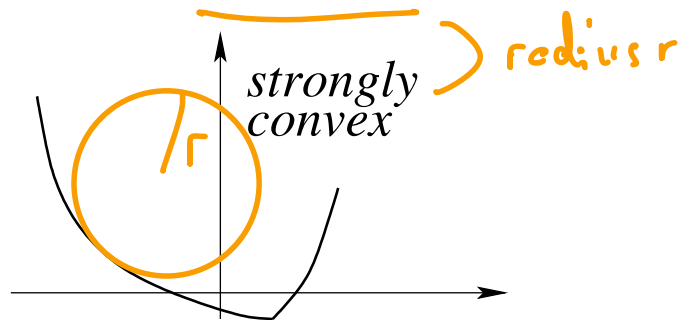
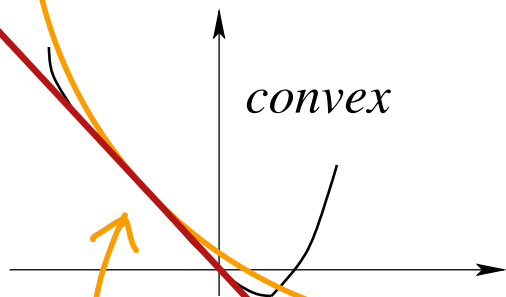
a. Smoothness

- Informally: A function f is **smooth** if its derivative is regular enough.



b. Strong Convexity

- Informally: f is **strongly convex** if it has lower bounded a curvature.



not strongly conv - no curvature

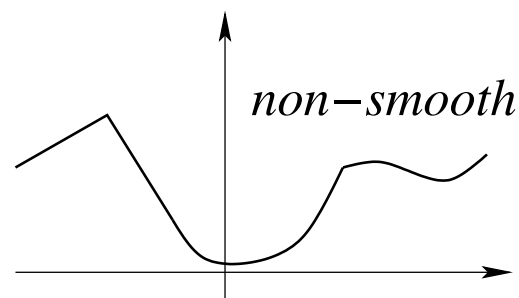
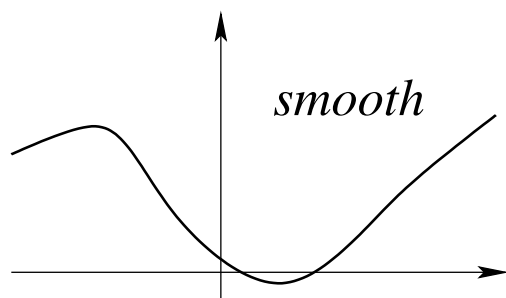
flat

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

2. Regularity or irregularity of the function

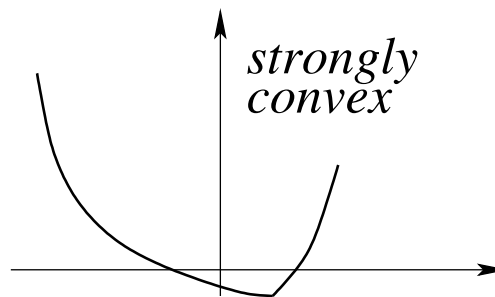
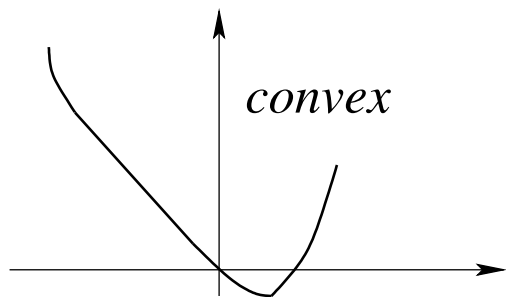
a. Smoothness

- *Informally:* A function f is **smooth** if its derivative is regular enough.



b. Strong Convexity

- *Informally:* f is **strongly convex** if it has lower bounded a curvature.



c. Other conditions: Lipschitz, Holder, Kurdyka-Lojasiewicz, etc.

Challenges: Non-convexity of f , non-smoothness, non-strongly-convex.

) each condⁿ we
have simpl^y f^u opt^m

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

3. Dimension of θ , set Θ .

a. Dimension d : $\Theta \subset \mathbb{R}^d$, d might be very large (typically millions)

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

3. Dimension of θ , set Θ .

a. Dimension d : $\Theta \subset \mathbb{R}^d$, d might be very large (typically millions)

b. Set Θ : (if Θ is a convex set.)

- May be described implicitly (via equations):

$$\Theta = \{\theta \in \mathbb{R}^d \text{ s.t. } \|\theta\|_2 \leq R \text{ and } \langle \theta, 1 \rangle = r\}.$$

↗ Use **dual formulation** of the problem.

} not covered too much in
the lecture.

What makes optimizing $\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$ easy or difficult?

3. Dimension of θ , set Θ .

a. Dimension d : $\Theta \subset \mathbb{R}^d$, d might be very large (typically millions)

b. Set Θ : (if Θ is a convex set.)

- May be described implicitly (via equations):

$$\Theta = \{\theta \in \mathbb{R}^d \text{ s.t. } \|\theta\|_2 \leq R \text{ and } \langle \theta, 1 \rangle = r\}.$$

↗ Use **dual formulation** of the problem.

Step 1: have
challenges in mind!

4. Structure of f .

In ML: If $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta)$, is the average of n functions

→ computing a gradient has a cost proportional to n .

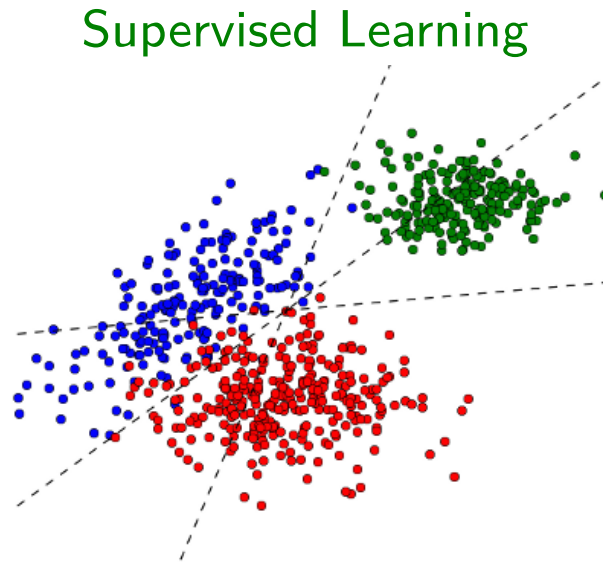
Challenges: Non-convexity of f , non-regularity (smoothness, strong-convexity), large d , implicit set Θ , large n (in ML).

Conclusion: Those are the most frequent challenges. What happens for the examples?

Outline

- 1 Optimization for machine learning: motivation and course objectives
 - General introduction: what is optimization and why study it?
 - Approach: what makes optimization difficult?
 - From supervised machine learning to optimization. → next step.
 - Summary: approach and lecture's goals
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method

Focus on Supervised Machine learning



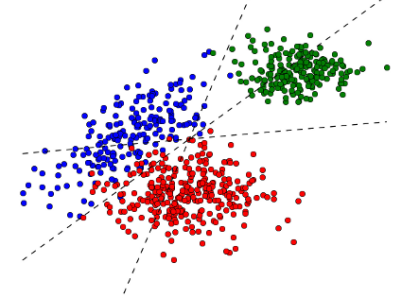
Supervised Machine Learning - Empirical risk minimization

- Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: predict the output Y from X .

- Prediction function $g_\theta : \mathcal{X} \rightarrow \mathbb{R}$, s.t. $g_\theta(X)$ prediction for Y .

- Model $\theta \in \mathbb{R}^d$ parametrizing g_θ .



Netw'd exmple is NN:

$NN_\theta(\text{input}) = \text{predict}$

$$\theta = (w, b)$$

↑
weights

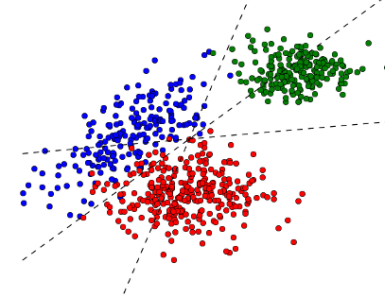
↓
biases

Supervised Machine Learning - Empirical risk minimization

- Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: predict the output Y from X .

- Prediction function $g_\theta : \mathcal{X} \rightarrow \mathbb{R}$, s.t. $g_\theta(X)$ prediction for Y .
- Model $\theta \in \mathbb{R}^d$ parametrizing g_θ .



↗ Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \rightarrow \mathbb{R}_+$

↗ Define the **generalization** risk:

$$\mathcal{R}(\theta) := \underbrace{\mathbb{E}_\rho [\ell(Y, g_\theta(X))]}_{\text{Measure of the average quality of the prediction}}.$$

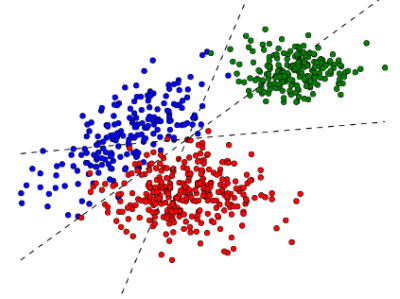
Measure of the average quality of the prediction.

Supervised Machine Learning - Empirical risk minimization

- Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: predict the output Y from X .

- Prediction function $g_\theta : \mathcal{X} \rightarrow \mathbb{R}$, s.t. $g_\theta(X)$ prediction for Y .
- Model $\theta \in \mathbb{R}^d$ parametrizing g_θ .



↗ Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \rightarrow \mathbb{R}_+$

↗ Define the generalization risk:

$$\mathcal{R}(\theta) := \underbrace{\mathbb{E}_\rho [\ell(Y, g_\theta(X))]}_{\text{Measure of the average quality of the prediction}}.$$

Measure of the average quality of the prediction.

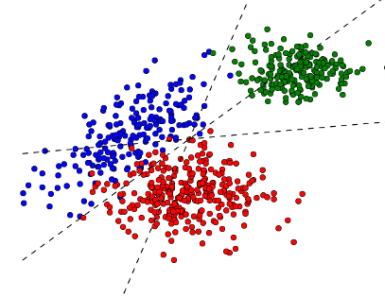
Problem: ρ is un-known, thus $\mathcal{R}(\theta)$ is unknown.

Supervised Machine Learning - Empirical risk minimization

- Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: predict the output Y from X .

- Prediction function $g_\theta : \mathcal{X} \rightarrow \mathbb{R}$, s.t. $g_\theta(X)$ prediction for Y .
- Model $\theta \in \mathbb{R}^d$ parametrizing g_θ .



↪ Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \rightarrow \mathbb{R}_+$

↪ Define the generalization risk:

$$\mathcal{R}(\theta) := \mathbb{E}_\rho [\ell(Y, g_\theta(X))].$$

Measure of the average quality of the prediction.

Problem: ρ is un-known, thus $\mathcal{R}(\theta)$ is unknown.

- Training data:** n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$, **i.i.d.**

- Define the **Empirical risk**:

$$\hat{\mathcal{R}}_n(\theta) := \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i))$$

Average over the training data quality of the prediction.

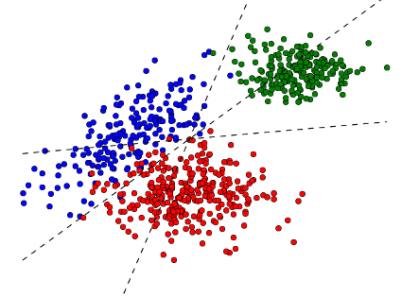
output
prediction on example x_i

Supervised Machine Learning - Empirical risk minimization

- Consider an input/output pair $(X, Y) \in \mathcal{X} \times \mathcal{Y}$, $(X, Y) \sim \rho$.

Goal: predict the output Y from X .

- Prediction function $g_\theta : \mathcal{X} \rightarrow \mathbb{R}$, s.t. $g_\theta(X)$ prediction for Y .
- Model $\theta \in \mathbb{R}^d$ parametrizing g_θ .



↪ Consider a loss function $\ell : \mathcal{Y} \times \mathbb{R} \rightarrow \mathbb{R}_+$

↪ Define the generalization risk:

$$\mathcal{R}(\theta) := \mathbb{E}_\rho [\ell(Y, g_\theta(X))].$$

Measure of the average quality of the prediction.

choice 2

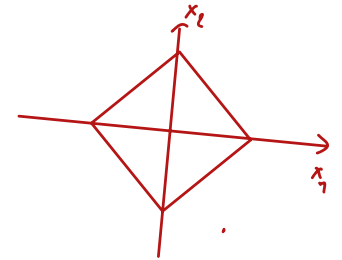
choice 1

Problem: ρ is un-known, thus $\mathcal{R}(\theta)$ is unknown.

- Training data:** n observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$, **i.i.d.**
- Define the **Empirical risk:**

$$\hat{\mathcal{R}}_n(\theta) := \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i))$$

Average over the training data quality of the prediction.



↪ **Penalized Empirical Risk minimization (ERM):** find $\hat{\theta}$ solution of

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} \underbrace{\hat{\mathcal{R}}_n(\theta)}_{\text{convex data fitting term}} + \underbrace{\mu \Omega(\theta)}_{\text{penalty}}.$$

Loss: $\|\theta\|_1$

$\mathcal{L}_2 = \|\theta\|_2^2$

Supervised machine learning - classical examples

Penalized ERM:

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} \underbrace{\hat{\mathcal{R}}_n(\theta)}_{\text{convex data fitting term}} + \underbrace{\mu\Omega(\theta)}_{\text{penalty}} = \min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i)) + \mu\Omega(\theta).$$

Encompasses many methods:

Model $g_\theta(X)$	Linear Models $\langle \theta, \Phi(X) \rangle^*$				Non-linear	
Loss ℓ	Square loss		Logistic loss	Hinge loss	01-loss	(Sq. loss, cross-entropy)
Penalty $\Omega(\theta)$	$\ \cdot \ _2$ (Ridge)	$\ \cdot \ _1$	None or $\ \cdot \ _2$ or $\ \cdot \ _1$			
Method name	Least Squares	Lasso	Logistic Reg.	SVM	Binary	Neural Nets

cf introductory lecture on supervised machine learning.

have been covered
in ML lecture

never
used.

will
be covered
by Adrian!

*for features $\Phi(X) \in \mathbb{R}^d$.



Approach in optimization for (supervised) machine learning

① Understand how the **method** (least-squares regression, logistic regression, neural networks,) choice influences the **regularity** (thus difficulty) of the problem:

- is the resulting objective convex / not
- is the resulting objective differentiable / not
- is the resulting objective smooth
- is the resulting objective strongly convex
- dimensionality
- etc.

→ see previous slide.



Approach in optimization for (supervised) machine learning

- 1 Understand how the **method** (least-squares regression, logistic regression, neural networks,) choice influences the **regularity** (thus difficulty) of the problem:

- is the resulting objective convex / not
- is the resulting objective differentiable / not
- is the resulting objective smooth
- is the resulting objective strongly convex
- dimensionality
- etc.

→ see previous slide.

- 2 Understand the **specificities** of the empirical-risk minimization problem

- The objective is a sum of n components
- A moderate precision may be enough

→ see second slide hereafter.

↳ deep - and not always understood.



Approach in optimization for (supervised) machine learning

- 1 Understand how the **method** (least-squares regression, logistic regression, neural networks,) choice influences the **regularity** (thus difficulty) of the problem:

- is the resulting objective convex / not
- is the resulting objective differentiable / not
- is the resulting objective smooth
- is the resulting objective strongly convex
- dimensionality
- etc.

→ see previous slide.

- 2 Understand the **specificities** of the empirical-risk minimization problem

- The objective is a sum of n components
- A moderate precision may be enough

→ see second slide hereafter.

- 3 Understand which **algorithm** (Gradient descent, stochastic GD, etc.) is well-suited depending on the **regularity** context and the **specificities**.

→ main objective of the lecture (all slides after that!).

Supervised machine learning - regularity of classical examples

Penalized ERM:

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} \underbrace{\hat{\mathcal{R}}_n(\theta)}_{\text{convex data fitting term}} + \underbrace{\mu\Omega(\theta)}_{\text{penalty}} = \min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i)) + \mu\Omega(\theta).$$

Regularity of the encompassed methods:

Model $g_\theta(X)$	Linear Models $\langle \theta, \Phi(X) \rangle^\dagger$				Non-linear	
Loss ℓ	Square loss		Logistic loss		Hinge loss	01-loss
Penalty $\Omega(\theta)$	$\ \cdot\ _2$ (Ridge)	$\ \cdot\ _1$	None or $\ \cdot\ _2$ or $\ \cdot\ _1$			
Method name	Least Squares	Lasso	Logistic Reg.		SVM	Binary
						Neural Nets

Supervised machine learning - regularity of classical examples

very important slide

Lemma: any loss which is written as $\ell(y_i, g_\theta(x_i))$, which is cvx wrt. its 2nd argument, if used

Penalized ERM:

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} \underbrace{\hat{\mathcal{R}}_n(\theta)}_{\text{convex data fitting term}} + \underbrace{\mu \Omega(\theta)}_{\text{penalty}} = \min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i)) + \mu \Omega(\theta).$$

Regularity of the encompassed methods:

on a linear model, gives a cvx $\hat{\mathcal{R}}_n, \mathcal{R}$.

Model $g_\theta(X)$	Linear Models $\langle \theta, \Phi(X) \rangle^\dagger$				Non-linear	
Loss ℓ	$\ell(y_i, \hat{y}_i)^2$ Square loss <u>cvx</u>	$\ell(y_i, \hat{y}_i) = \log(1 + e^{-y\hat{y}})$ Logistic loss <u>cvx</u>	Hinge loss <u>cvx</u>	01-loss	(Sq. loss, cross-entropy)	
Penalty $\Omega(\theta)$	$\ \cdot\ _2$ (Ridge)	$\ \cdot\ _1$	None or $\ \cdot\ _2$ or $\ \cdot\ _1$			
Method name	Least Squares	Lasso	Logistic Reg.	SVM	Binary	Neural Nets
Q1 ntps Convex	✓	✓	✓	✓	✗	✗
Smooth	✓	✗	✓	✗	not diff	depends!! on activation func
Strongly convex	if regularized yes ✓		✓	✓ if regularized		↑ more in Lec 2...
Large d	sometimes $10^2 \cdot 10^6$	sometimes	sometimes	sometimes $10^1 \cdot 10^6$		always $10^6 \cdot 10^{11}$
Large n	always possibly large					
typically always more than 1000s . $10^3 - 10^{13}$.						

† for features $\Phi(X) \in \mathbb{R}^d$.

† for features $\Phi(X) \in \mathbb{R}^d$.
 typically always more than 1000s. $10^3 - 10^{13}$.

Supervised machine learning - specificities of empirical risk minimization

- 1 The empirical risk minimization writes as an average of n functions (one per observation)

$$\underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_{\theta}(x_i)) + \mu \Omega(\theta)}_{\text{regularized ERM problem}} \quad \leftrightarrow \quad \underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(\theta)}_{\text{generic form to study optimization}} .$$

→ n can be very large (millions to billions of observations)

Supervised machine learning - specificities of empirical risk minimization

- ① The empirical risk minimization writes as an average of n functions (one per observation)

$$\underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_{\theta}(x_i)) + \mu \Omega(\theta)}_{\text{regularized ERM problem}} \leftrightarrow \underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(\theta)}_{\text{generic form to study optimization}} .$$

→ n can be very large (millions to billions of observations)

- ② Our *true* objective is to minimize the *generalization risk*.

$$\mathcal{R}(\theta) := \mathbb{E}_{\rho} [\ell(Y, g_{\theta}(X))] .$$

as the distribution ρ is unknown, we minimize a *proxy* instead, the empirical risk.

→ risk of overfitting, impact of regularization.

$\mathcal{L}, \hat{\mathcal{R}}_n$



→ a moderate precision may be enough: no need to optimize ~~very~~ precisely a function which is only an approximation of our true objective.

exactly



Supervised machine learning - specificities of empirical risk minimization

- ① The empirical risk minimization writes as an average of n functions (one per observation)

$$\underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i)) + \mu \Omega(\theta)}_{\text{regularized ERM problem}} \leftrightarrow \underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(\theta)}_{\text{generic form to study optimization}} .$$

→ n can be very large (millions to billions of observations)

- ② Our *true* objective is to minimize the *generalization risk*.

$$\mathcal{R}(\theta) := \mathbb{E}_\rho [\ell(Y, g_\theta(X))] .$$

as the distribution ρ is unknown, we minimize a *proxy* instead, the empirical risk.

→ risk of overfitting, impact of regularization.

→  a moderate precision may be enough: no need to optimize very precisely a function which is only an approximation of our true objective.

→ Difference $\mathcal{R} - \hat{\mathcal{R}}_n$? **Statistical question!**

- ▶ Short answer: rates driven by Central limit theorem and law of large numbers → difference of order $\frac{1}{n}$ to $\frac{1}{\sqrt{n}}$.

Supervised machine learning - specificities of empirical risk minimization

- ① The empirical risk minimization writes as an average of n functions (one per observation)

$$\underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_{\theta}(x_i)) + \mu \Omega(\theta)}_{\text{regularized ERM problem}} \quad \leftrightarrow \quad \underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(\theta)}_{\text{generic form to study optimization}} .$$

→ n can be very large (millions to billions of observations)

- ② Our *true* objective is to minimize the *generalization risk*.

$$\mathcal{R}(\theta) := \mathbb{E}_{\rho} [\ell(Y, g_{\theta}(X))] .$$

as the distribution ρ is unknown, we minimize a *proxy* instead, the empirical risk.

→ risk of overfitting, impact of regularization.

→  a moderate precision may be enough: no need to optimize very precisely a function which is only an approximation of our true objective.

→ Difference $\mathcal{R} - \hat{\mathcal{R}}_n$? **Statistical question!**

- ▶ Short answer: rates driven by Central limit theorem and law of large numbers → difference of order $\frac{1}{n}$ to $\frac{1}{\sqrt{n}}$. → Order of magnitude to remember!

Supervised machine learning - specificities of empirical risk minimization

- ① The empirical risk minimization writes as an average of n functions (one per observation)

$$\underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(y_i, g_\theta(x_i)) + \mu \Omega(\theta)}_{\text{regularized ERM problem}} \leftrightarrow \underbrace{\min_{\theta \in \Theta \subset \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(\theta)}_{\text{generic form to study optimization}} .$$

→ n can be very large (millions to billions of observations)

- ② Our *true* objective is to minimize the *generalization risk*.

$$\mathcal{R}(\theta) := \mathbb{E}_\rho [\ell(Y, g_\theta(X))] .$$

as the distribution ρ is unknown, we minimize a *proxy* instead, the empirical risk.

→ risk of overfitting, impact of regularization.

→  a moderate precision may be enough: no need to optimize very precisely a function which is only an approximation of our true objective.

→ Difference $\mathcal{R} - \hat{\mathcal{R}}_n$? **Statistical question!**

- ▶ Short answer: rates driven by Central limit theorem and law of large numbers → difference of order $\frac{1}{n}$ to $\frac{1}{\sqrt{n}}$. → Order of magnitude to remember!
- ▶ Long answer: advanced - see e.g. Rademacher complexity derivation, Chapter 4 in bach2021learning.

Remark: those specificities extend to other un-supervised learning techniques, beyond ERM.

Let's recap! Wooclap quizz!

Methods and regularities

Logistic regression, no regularization

Least squares regression, Lasso regularization (L1)

cvx

non smooth

Support vector machine (SVM) (hinge loss) with L2 regularization

strongly convex

Fully connected Neural network with 2 layers and sigmoid activation, cross entropy loss

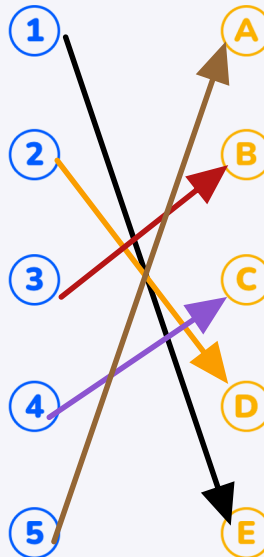
non cvx

Fully connected Neural network with 2 layers and relu activation, square loss

Least squares regression with L2 regularization

cvx quadratic

quadratic



Non convex, non smooth

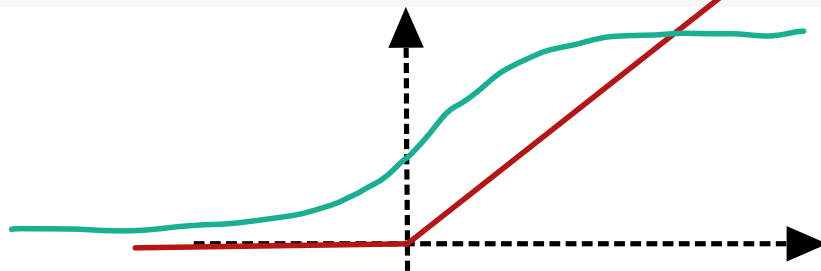
Strongly Convex, Non smooth

Non convex, smooth

Convex, Non smooth

Smooth, convex, not necessarily strongly convex

Quadratic



Relu activation

Sigmoid activation

Restart at 20!!

Outline

- 1 Optimization for machine learning: motivation and course objectives
 - General introduction: what is optimization and why study it?
 - Approach: what makes optimization difficult?
 - From supervised machine learning to optimization.
 - Summary: approach and lecture's goals
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method

Optimization : finding the minimal (maximal) value of a function over a set

We will study algorithms to solve:

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} f(\theta)$$

- depending on the *regularity context* (convexity, smoothness, strong-convexity)
- and the problem constraints (dimensionality)
- with specificities of ERM in mind.

In Machine Learning, key example: **ERM**

$$\min_{\theta \in \Theta \subset \mathbb{R}^d} \left\{ f(\theta) := \frac{1}{n} \sum_{i=1}^n f_i(\theta) \right\}$$

- Regularity context depends on the method.
- Specificities: large number n of observations, moderate precision sufficient.

Goals of the optimization sessions.

- ① **Goal 0:** Relate machine learning methods (least-squares, neural networks, penalty choices), and the learning context (model size, number of observations) to conditions of regularity on the ERM problem.
→ see above.

Goals of the optimization sessions.

- ① **Goal 0:** Relate machine learning methods (least-squares, neural networks, penalty choices), and the learning context (model size, number of observations) to conditions of regularity on the ERM problem.
→ see above.
- ② **Goal 1:** understanding
 - what optimization algorithms are.
 - the motivation for the introduction of each algorithm, their links.
 - ultimately, how algorithms used in deep learning today emerged.

Goals of the optimization sessions.

- ① **Goal 0:** Relate machine learning methods (least-squares, neural networks, penalty choices), and the learning context (model size, number of observations) to conditions of regularity on the ERM problem.
→ see above.
- ② **Goal 1:** understanding
 - what optimization algorithms are.
 - the motivation for the introduction of each algorithm, their links.
 - ultimately, how algorithms used in deep learning today emerged.
- ③ **Goal 2:** Given an algorithm, e.g., “**Nesterov accelerated gradient**”, understand:
 - if it is **deterministic** / **stochastic**, **first** / **second order**, **adaptive** / **not**,
 - thus *which optimization context it is well-suited for*.

Goals of the optimization sessions.

- ① **Goal 0:** Relate machine learning methods (least-squares, neural networks, penalty choices), and the learning context (model size, number of observations) to conditions of regularity on the ERM problem.
→ see above.
- ② **Goal 1:** understanding
 - what optimization algorithms are.
 - the motivation for the introduction of each algorithm, their links.
 - ultimately, how algorithms used in deep learning today emerged.
- ③ **Goal 2:** Given an algorithm, e.g., “**Nesterov accelerated gradient**”, understand:
 - if it is **deterministic** / **stochastic**, **first** / **second order**, **adaptive** / **not**,
 - thus *which optimization context it is well-suited for*.
- ④ **Goal 3:** Conversely, understand which algorithm is suitable to solve a problem depending on the regularity context.
 - E.g., solving a problem **in large dimension**, **small number of observations**, **non smooth**, **convex**, and that requires moderate precision.

Optimization sessions take-aways:

- better understanding of optimization, especially in the context of machine learning algorithms.
- $\rightsquigarrow$ **Goal 0+1+2+3** $\Rightarrow$ for a given machine learning problem, understand which algorithms you expect to work best.

Step 1: for each algorithm

① Basics

- ① Intuitive construction and behavior, motivation
- ② Pseudo-code
- ③ Characteristics
- ④ Complexity per iteration, memory requirements.
- ⑤ Hyper-parameters
- ⑥ Intuition on the impact and choice of hyper-parameters.

② **Theory(★).** Theoretical framework to support the motivation or the choice of hyper-parameters.

③ **Comparison.** Comparison to previously studied algorithms.

④ **Limits.** Supported by intuition and theory.

~~~~~> **Goal 1 & 2.**

## Step 2: relate algorithms, their characteristics, corresponding challenges

### ① List of challenges / difficulties.

- Understand how they impact algorithms

### ② Algorithms' features

- Understand to which “difficulty” each feature corresponds
- Why it is helpful.

### ③ Ultimately(★)

- **Choosing algorithms.** For any situation or set of difficulties, understand which algorithms to propose.
- **Best situation for an algorithm.** For any algorithm, describe the “perfect situation” to use it.

~~~~~> **Goal 2 & 3.**


Step 2: relate algorithms, their characteristics, corresponding challenges

1 List of challenges / difficulties.

- Understand how they impact algorithms

2 Algorithms' features

- Understand to which “difficulty” each feature corresponds
- Why it is helpful.

3 Ultimately(★)

- **Choosing algorithms.** For any situation or set of difficulties, understand which algorithms to propose.
- **Best situation for an algorithm.** For any algorithm, describe the “perfect situation” to use it.

~~~~~> **Goal 2 & 3.**

**Example.** Understand the doc of `sklearn.linear_model.LogisticRegression`:

**solver** : {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' is limited to one-versus-rest schemes.

**Warning:** The choice of the algorithm depends on the penalty chosen: Supported penalties by solver:

- 'newton-cg' - ['l2', 'none']
- 'lbfgs' - ['l2', 'none']
- 'liblinear' - ['l1', 'l2']
- 'sag' - ['l2', 'none']
- 'saga' - ['elasticnet', 'l1', 'l2', 'none']

# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
  - Basic properties
  - Visualization
  - Gradients, Hessians, and optimality conditions
  - Regularity conditions
- 3 Gradient descent method, stochastic gradient method

# Goal of this section

Recall a few elementary definitions:

- Gradients, Hessians, and their computation on simple examples.
- A reminder on Taylor approximation formula and its meaning.
- Level sets, and how to visualize a 2-dimensional function.
- Mathematical definition of convexity, strong convexity, smoothness, intuition on regularity.

# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
  - Basic properties
  - Visualization
  - Gradients, Hessians, and optimality conditions
  - Regularity conditions
- 3 Gradient descent method, stochastic gradient method

# Gradient, Hessian

grid search is out of the picture  
→

## Motivation:

- Observing the function everywhere is not possible in a high dimensional spaces
- Instead, we can observe the function *at a given point*, and access information on its variation *around that point*.  
→ such variations are given by the **derivatives** of the functions.

# Gradient, Hessian

## Motivation:

- Observing the function everywhere is not possible in a high dimensional spaces
- Instead, we can observe the function *at a given point*, and access information on its variation *around that point*.  
→ such variations are given by the **derivatives** of the functions.

**Gradient.** The gradient  $\nabla f(\theta)$  of a differentiable function  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is the vector of partial derivatives:

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1}(\theta) \\ \vdots \\ \frac{\partial f}{\partial \theta_d}(\theta) \end{pmatrix} \in \mathbb{R}^d$$

# Gradient, Hessian

## Motivation:

- Observing the function everywhere is not possible in a high dimensional spaces
- Instead, we can observe the function *at a given point*, and access information on its variation *around that point*.  
→ such variations are given by the **derivatives** of the functions.

**Gradient.** The gradient  $\nabla f(\theta)$  of a differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is the vector of partial derivatives:

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1}(\theta) \\ \vdots \\ \frac{\partial f}{\partial \theta_d}(\theta) \end{pmatrix} \in \mathbb{R}^d \quad \text{with} \quad \frac{\partial f}{\partial \theta_i}(\theta) = \lim_{h \rightarrow 0} \frac{f(\theta + h e_i) - f(\theta)}{h}, \text{ and } e_i \text{ the } i^{\text{th}} \text{ canon. vec.}$$

gradient points in the direct<sup>n</sup> of maximal increase!!

# Gradient, Hessian

## Motivation:

- Observing the function everywhere is not possible in a high dimensional spaces
- Instead, we can observe the function *at a given point*, and access information on its variation *around that point*.  
→ such variations are given by the **derivatives** of the functions.

**Gradient.** The gradient  $\nabla f(\theta)$  of a differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is the vector of partial derivatives:

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1}(\theta) \\ \vdots \\ \frac{\partial f}{\partial \theta_d}(\theta) \end{pmatrix} \in \mathbb{R}^d \quad \text{with} \quad \frac{\partial f}{\partial \theta_i}(\theta) = \lim_{h \rightarrow 0} \frac{f(\theta + h e_i) - f(\theta)}{h}, \text{ and } e_i \text{ the } i^{\text{th}} \text{ canon. vec.}$$

**Hessian matrix.** The Hessian matrix  $\nabla^2 f(\theta)$  of a twice **continuously differentiable** function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is:

$$\nabla^2 f(\theta) = \begin{pmatrix} \frac{\partial^2 f}{\partial \theta_1^2}(\theta) & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_d}(\theta) \\ \frac{\partial^2 f}{\partial \theta_2 \partial \theta_1}(\theta) & \frac{\partial^2 f}{\partial \theta_2^2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_2 \partial \theta_d}(\theta) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial \theta_d \partial \theta_1}(\theta) & \frac{\partial^2 f}{\partial \theta_d \partial \theta_2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_d^2}(\theta) \end{pmatrix} \in \mathbb{R}^{d \times d}$$

↪ second derivative w.r.t.  $\theta_1$  and  $\theta_2$

if  $d$  is large



# Gradient, Hessian

## Motivation:

- Observing the function everywhere is not possible in a high dimensional spaces
- Instead, we can observe the function *at a given point*, and access information on its variation *around that point*.  
→ such variations are given by the **derivatives** of the functions.

**Gradient.** The gradient  $\nabla f(\theta)$  of a differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is the vector of partial derivatives:

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1}(\theta) \\ \vdots \\ \frac{\partial f}{\partial \theta_d}(\theta) \end{pmatrix} \in \mathbb{R}^d \quad \text{with} \quad \frac{\partial f}{\partial \theta_i}(\theta) = \lim_{h \rightarrow 0} \frac{f(\theta + h e_i) - f(\theta)}{h}, \text{ and } e_i \text{ the } i^{\text{th}} \text{ canon. vec.}$$

**Hessian matrix.** The Hessian matrix  $\nabla^2 f(\theta)$  of a twice continuously differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  at point  $\theta$  is:

$$\nabla^2 f(\theta) = \begin{pmatrix} \frac{\partial^2 f}{\partial \theta_1^2}(\theta) & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_d}(\theta) \\ \frac{\partial^2 f}{\partial \theta_2 \partial \theta_1}(\theta) & \frac{\partial^2 f}{\partial \theta_2^2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_2 \partial \theta_d}(\theta) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial \theta_d \partial \theta_1}(\theta) & \frac{\partial^2 f}{\partial \theta_d \partial \theta_2}(\theta) & \cdots & \frac{\partial^2 f}{\partial \theta_d^2}(\theta) \end{pmatrix} \in \mathbb{R}^{d \times d}$$

**Remark:** if  $d = 1$ , thus  $f : \mathbb{R} \rightarrow \mathbb{R}$ , then

$$\nabla f(\theta) = f'(\theta) \quad \text{and} \quad \nabla^2 f(\theta) = f''(\theta).$$

# Gradient and Hessian computations: examples on simple functions

- ① **Linear function:** if there exists a vector  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \langle \underline{b}, \theta \rangle + c = \underbrace{b^\top \theta}_{\sum_{i=1}^d b_i \theta_i} + c,$$

then, for all  $\theta \in \mathbb{R}^d$ ,

$$\nabla f(\theta) = b$$

$$\nabla^2 f(\theta) = 0$$

if not obvious,  
work on it.

- ② **Quadratic function:** if there exists a matrix  $A \in \mathbb{R}^{d \times d}$ ,  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \frac{1}{2} \theta^\top A \theta + b^\top \theta + c$$

then, for all  $\theta \in \mathbb{R}^d$ ,

$$\nabla f(\theta) = \frac{1}{2} (A + A^\top) \theta + b$$

$$\nabla^2 f(\theta) = \frac{1}{2} (A + A^\top)$$

$(ax^2)'$  in  $d=1$   
 $2ax$

gradient | quadratic part  
→ linear

$$\frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d a_{ij} \theta_j \theta_i$$

# Gradient and Hessian computations: examples on simple functions

- ① **Linear function:** if there exists a vector  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \langle b, \theta \rangle + c = b^\top \theta + c,$$

then, for all  $\theta \in \mathbb{R}^d$ ,

$$\nabla f(\theta) = b \qquad \nabla^2 f(\theta) = 0$$

- ② **Quadratic function:** if there exists a matrix  $A \in \mathbb{R}^{d \times d}$ ,  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \frac{1}{2} \theta^\top A \theta + b^\top \theta + c$$

then, for all  $\theta \in \mathbb{R}^d$ ,

$$\nabla f(\theta) = \frac{1}{2}(A + A^\top)\theta + b \qquad \nabla^2 f(\theta) = \frac{1}{2}(A + A^\top)$$

- if  $A$  is symmetric,  $b = 0$ ,

$$\nabla f(\theta) = A\theta \qquad \nabla^2 f(\theta) = A$$

- if  $f(\theta) = \frac{1}{2} \|\theta\|^2$ , i.e.,  $A = \text{Id}$ , then

$$\nabla f(\theta) = \theta \qquad \nabla^2 f(\theta) = \text{Id}$$

How to obtain the gradient beyond those simple cases?

- **Analytically:** often a simple exercise, excellent cheatsheet given in [Matrix cookbook](#)
- **Numerically:** in particular in deep learning → [auto-differentiation](#).  
↪ cf the part of the lecture on [backpropagation](#) in deep learning.

link



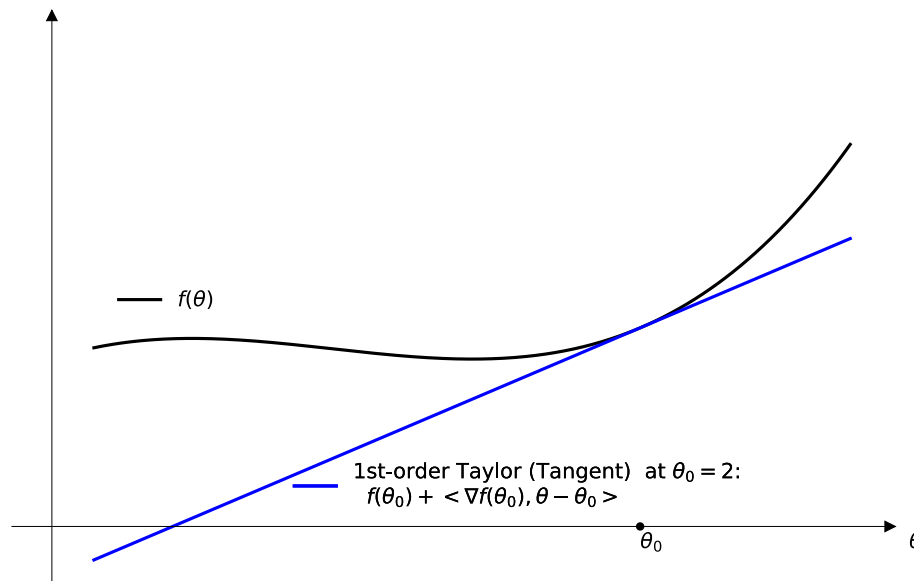
key block of NN learning

# Taylor Expansion: local approximation of the function

The gradient  $\nabla f(\theta_0)$  and Hessian  $\nabla^2 f(\theta_0)$  enable to obtain an approximation of the function, locally around a point  $\theta_0$ . This is given by *Taylor expansion formula*.

## ① First-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st order})$$



# Taylor Expansion: local approximation of the function

The gradient  $\nabla f(\theta_0)$  and Hessian  $\nabla^2 f(\theta_0)$  enable to obtain an approximation of the function, locally around a point  $\theta_0$ . This is given by *Taylor expansion formula*.

## ① First-order Taylor approximation

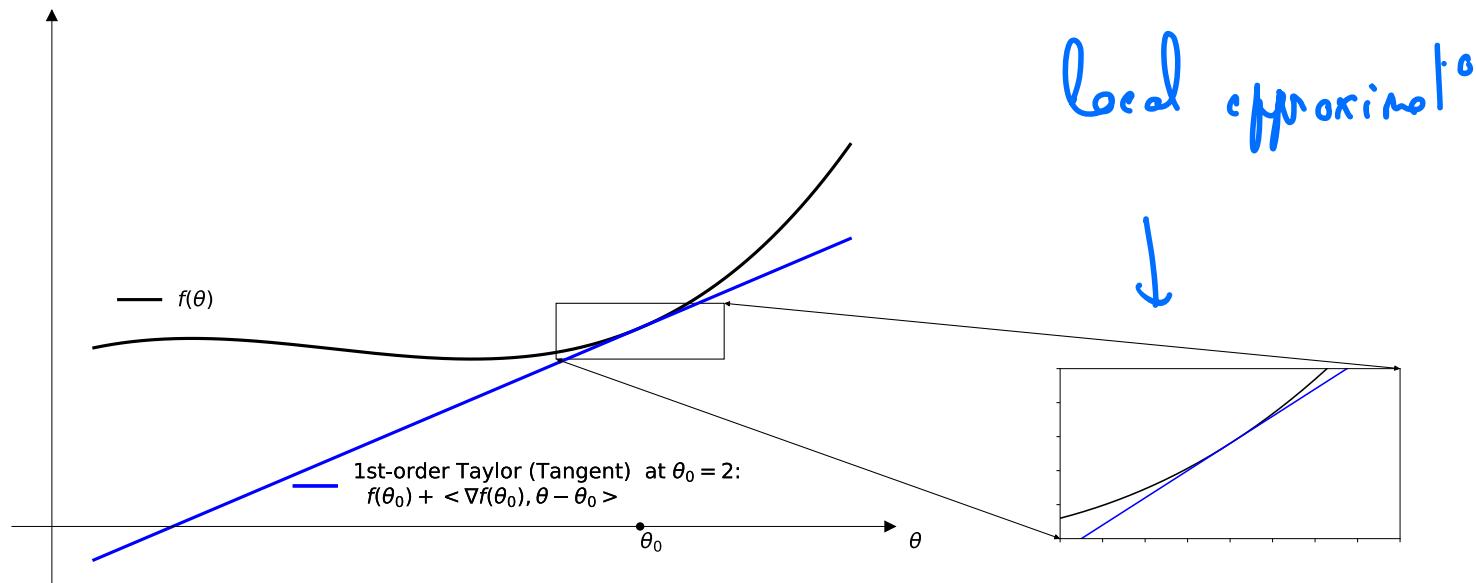
$$f(\theta) = f(\theta_0) + \underbrace{\langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} +$$

$$\underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2}$$

(Taylor, 1st order)

can be seen as <sup>another</sup> definition of the gradient

mathematical notation for a negligible residual scaling as  $\|\theta - \theta_0\|^2$



# Taylor Expansion: local approximation of the function

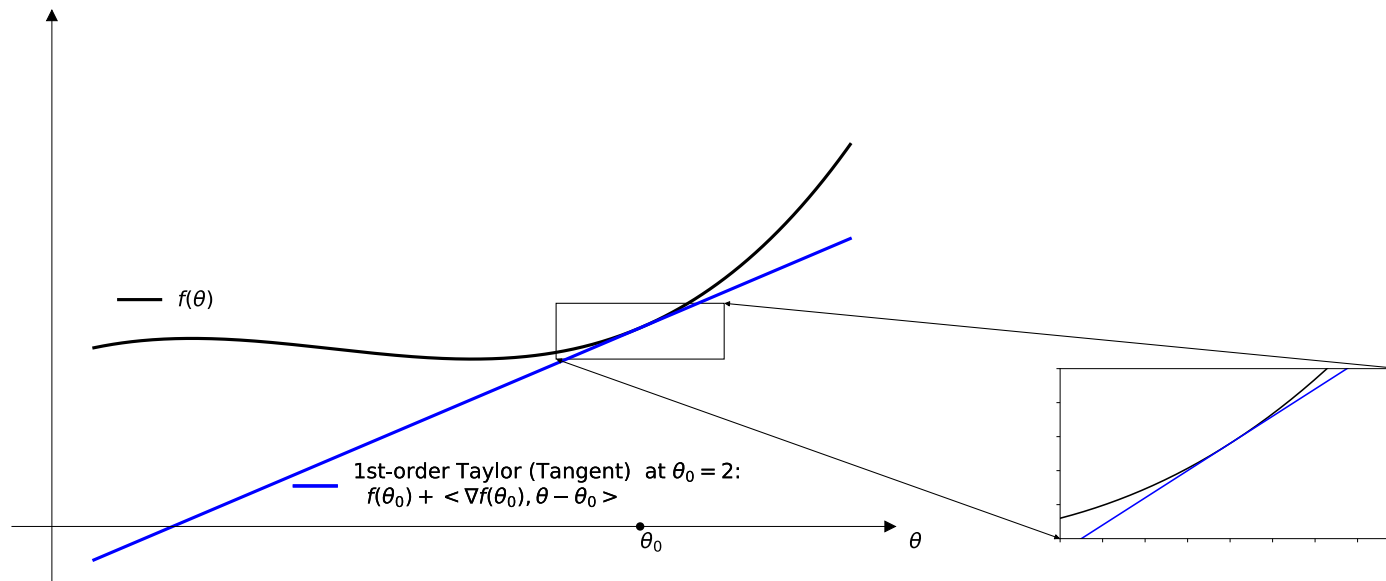
The gradient  $\nabla f(\theta_0)$  and Hessian  $\nabla^2 f(\theta_0)$  enable to obtain an approximation of the function, locally around a point  $\theta_0$ . This is given by *Taylor expansion formula*.

## ① First-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st order})$$

## ② Second-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{1}{2}(\theta - \theta_0)^\top \nabla^2 f(\theta_0) (\theta - \theta_0)}_{\text{Second-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^3)}_{\text{negligible residual scaling as } \|\theta - \theta_0\|^3} \quad (\text{Taylor, 2nd order})$$



# Taylor Expansion: local approximation of the function

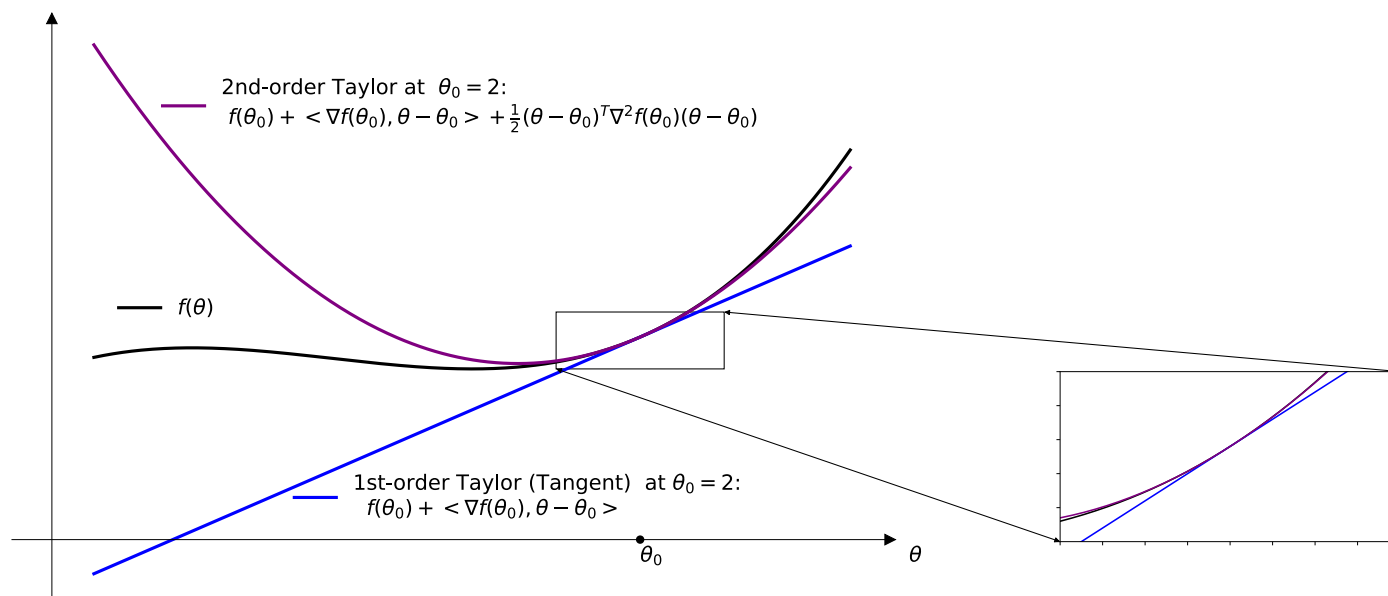
The gradient  $\nabla f(\theta_0)$  and Hessian  $\nabla^2 f(\theta_0)$  enable to obtain an approximation of the function, locally around a point  $\theta_0$ . This is given by *Taylor expansion formula*.

## ① First-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st order})$$

## ② Second-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{1}{2}(\theta - \theta_0)^\top \nabla^2 f(\theta_0) (\theta - \theta_0)}_{\text{Second-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^3)}_{\text{negligible residual scaling as } \|\theta - \theta_0\|^3} \quad (\text{Taylor, 2nd order})$$



# Taylor Expansion: visualization in dimension 2

## ① First-order Taylor approximation

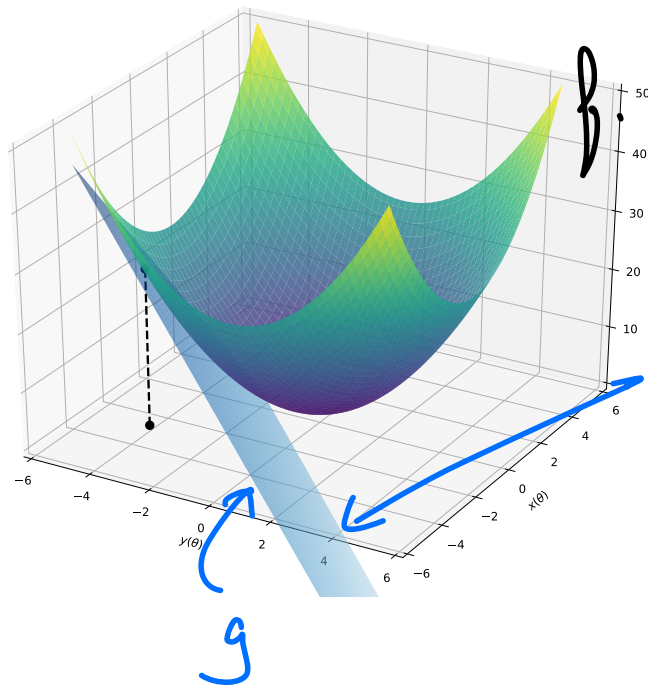
$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} +$$

$$\underbrace{O(\|\theta - \theta_0\|^2)}$$

mathematical notation for a negligible residual scaling as  $\|\theta - \theta_0\|^2$

(Taylor, 1st-order)

(Serre!)



first order approximation

tangent plane



# Taylor Expansion: visualization in dimension 2

## ① First-order Taylor approximation

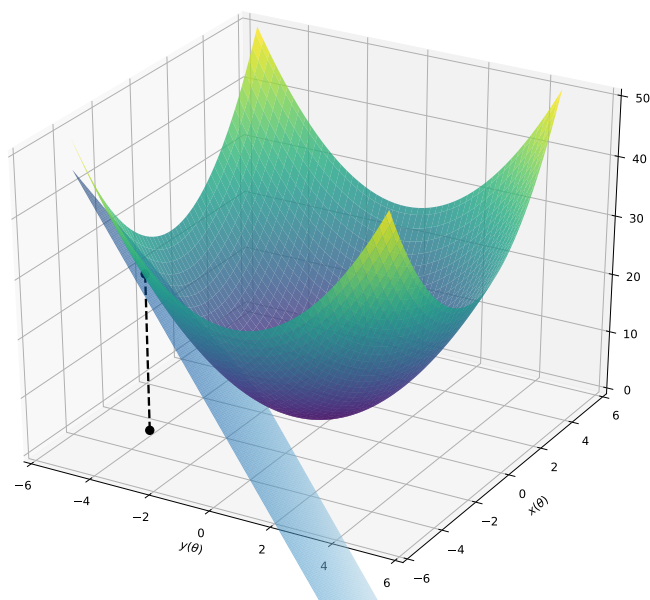
$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} +$$

First-order approximation

$$\underbrace{O(\|\theta - \theta_0\|^2)}$$

mathematical notation for a negligible residual scaling as  $\|\theta - \theta_0\|^2$

(Taylor, 1st-order)



Property: The gradient  $\nabla f(\theta_0)$  corresponds to the direction in which the function increases the fastest around  $\theta_0$ .

# Taylor Expansion: visualization in dimension 2

## ① First-order Taylor approximation

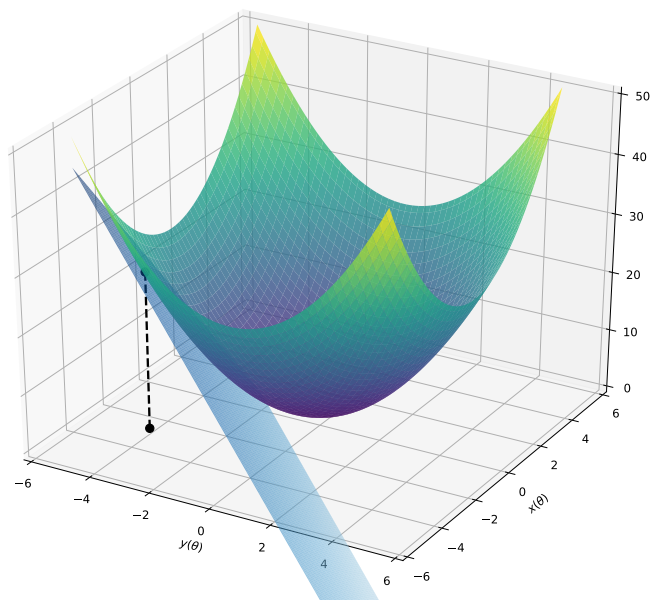
$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} +$$

First-order approximation

$$\underbrace{O(\|\theta - \theta_0\|^2)}$$

mathematical notation for a negligible residual scaling as  $\|\theta - \theta_0\|^2$

(Taylor, 1st-order)



Property: The gradient  $\nabla f(\theta_0)$  corresponds to the direction in which the function increases the fastest around  $\theta_0$ .

Indeed, as  $h \simeq 0$ , Taylor, 1st-order writes:

$$f(\theta_0 + h) \simeq f(\theta_0) +$$

$$\underbrace{\langle \nabla f(\theta_0), h \rangle}$$

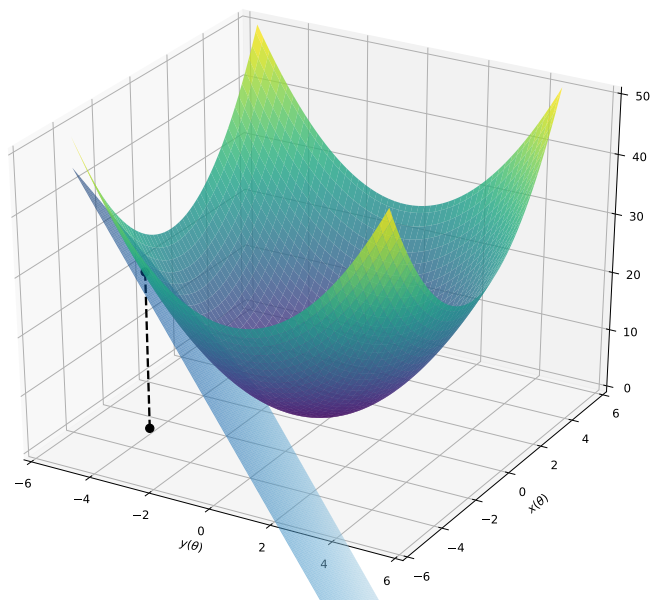
(1)

→ Maximal if  $h$  is aligned with  $\nabla f(\theta_0)$

# Taylor Expansion: visualization in dimension 2

## ① First-order Taylor approximation

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st-order})$$



Property: The gradient  $\nabla f(\theta_0)$  corresponds to the direction in which the function increases the fastest around  $\theta_0$ .

Indeed, as  $h \simeq 0$ , Taylor, 1st-order writes:

$$f(\theta_0 + h) \simeq f(\theta_0) + \underbrace{\langle \nabla f(\theta_0), h \rangle}_{\rightarrow \text{Maximal if } h \text{ is aligned with } \nabla f(\theta_0)} \quad (1)$$

Proof: by Cauchy-Schwarz,  $\langle \nabla f(\theta_0), h \rangle \leq \|\nabla f(\theta_0)\| \|h\|$ , with an equality iff  $h = \tau \nabla f(\theta_0)$ ,  $\tau > 0$ .

# Taylor Expansion: visualization in dimension 2

## ① First-order Taylor approximation

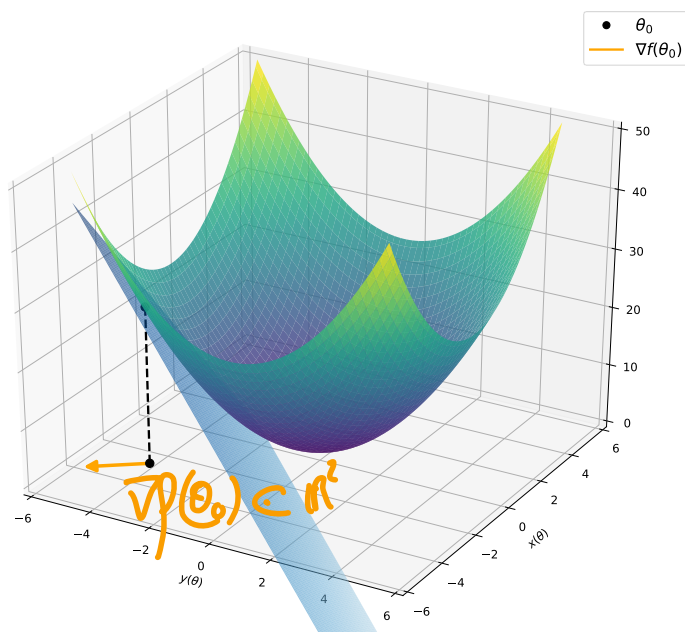
$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} +$$

First-order approximation

$$\underbrace{O(\|\theta - \theta_0\|^2)}$$

mathematical notation for a negligible residual scaling as  $\|\theta - \theta_0\|^2$

(Taylor, 1st-order)



Property: The gradient  $\nabla f(\theta_0)$  corresponds to the direction in which the function increases the fastest around  $\theta_0$ .

Indeed, as  $h \simeq 0$ , Taylor, 1st-order writes:

$$f(\theta_0 + h) \simeq f(\theta_0) +$$

$$\underbrace{\langle \nabla f(\theta_0), h \rangle}_{\text{Maximal if } h \text{ is aligned with } \nabla f(\theta_0)}$$

(1)

→ Maximal if  $h$  is aligned with  $\nabla f(\theta_0)$

Proof: by Cauchy-Schwarz,  $\langle \nabla f(\theta_0), h \rangle \leq \|\nabla f(\theta_0)\| \|h\|$ , with an equality iff  $h = \tau \nabla f(\theta_0)$ ,  $\tau > 0$ .

## Taylor Expansion: case of quadratic functions

**Remark:** in the case of a quadratic function, i.e. if there exists a matrix  $A \in \mathbb{R}^{d \times d}$ ,  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \frac{1}{2} \theta^T A \theta + b^T \theta + c$$

then, for all  $\theta \in \mathbb{R}^d$ , then the second-order Taylor approximation is **exact** at all points: for all  $\theta, \theta_0$

$$f(\theta) = f(\theta_0) + \underbrace{\langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{1}{2} (\theta - \theta_0)^T \nabla^2 f(\theta_0) (\theta - \theta_0)}_{\text{Second-order approximation}}.$$

$\uparrow$   
 $=$  exactly everywhere.

# Taylor Expansion: case of quadratic functions

**Remark:** in the case of a quadratic function, i.e. if there exists a matrix  $A \in \mathbb{R}^{d \times d}$ ,  $b \in \mathbb{R}^d$  and  $c \in \mathbb{R}$  such that

$$f : \theta \mapsto \frac{1}{2} \theta^T A \theta + b^T \theta + c$$

then, for all  $\theta \in \mathbb{R}^d$ , then the second-order Taylor approximation is exact at all points: for all  $\theta, \theta_0$

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{1}{2} (\theta - \theta_0)^T \nabla^2 f(\theta_0) (\theta - \theta_0)}_{\text{Second-order approximation}}.$$

Up to now, we have seen:

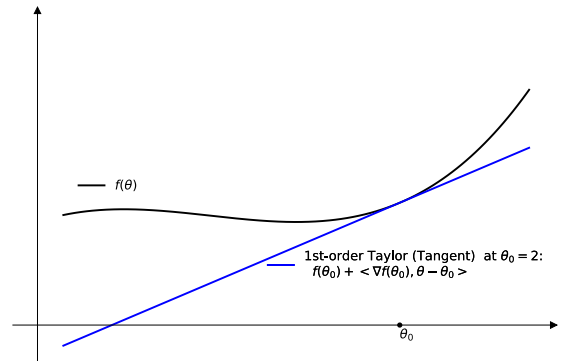
- Definition of  $\nabla f$ ,  $\nabla^2 f$
- Taylor approximation formulas
- Interpretation of the gradient direction.

# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
  - Basic properties
  - Visualization
  - Gradients, Hessians, and optimality conditions
  - Regularity conditions
- 3 Gradient descent method, stochastic gradient method

# Level sets of a function

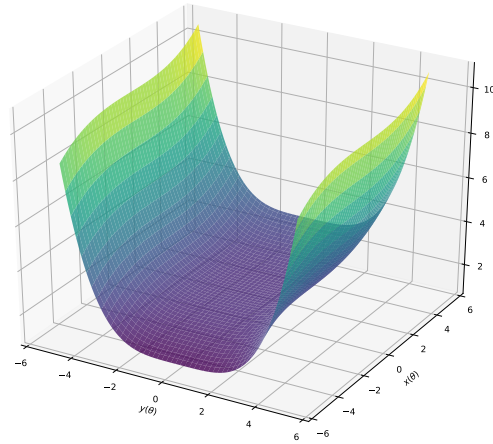
- 1 One-dimensional ( $d = 1$ ) representations are often misleading, as multiple phenomena only occur in dimension at least 2.





# Level sets of a function

- 1 One-dimensional ( $d = 1$ ) representations are often misleading, as multiple phenomena only occur in dimension at least 2.

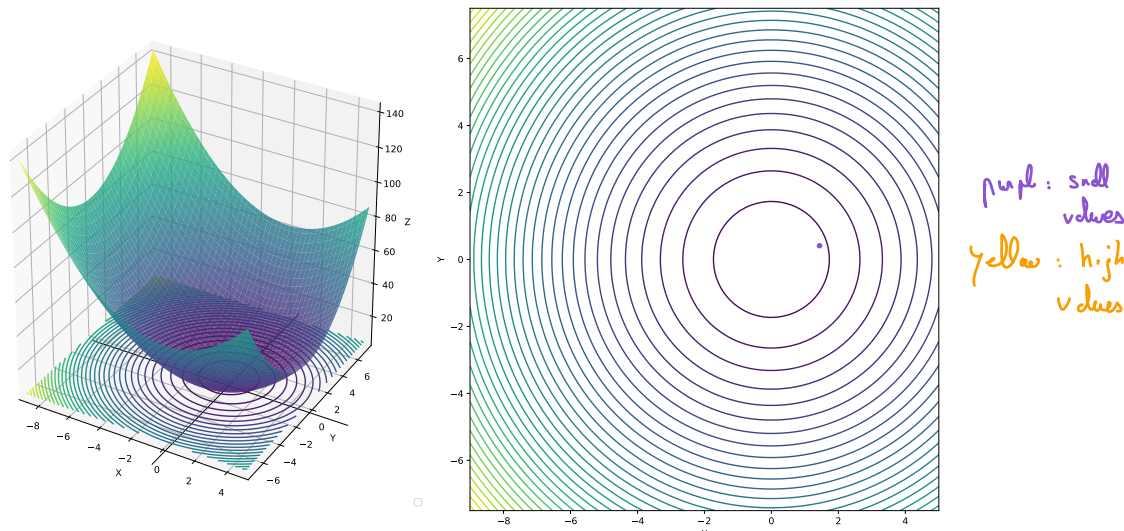


# Level sets of a function

- 1 One-dimensional ( $d = 1$ ) representations are often misleading, as multiple phenomena only occur in dimension at least 2.
- 2 However, representing a dimension over  $\mathbb{R}^2$  as a 3d plot is not always convenient.  
→ we therefore often represent *level-sets* of the function:

$$\mathcal{C}_c = \{\theta \in \mathbb{R}^d, f(\theta) = c\}.$$

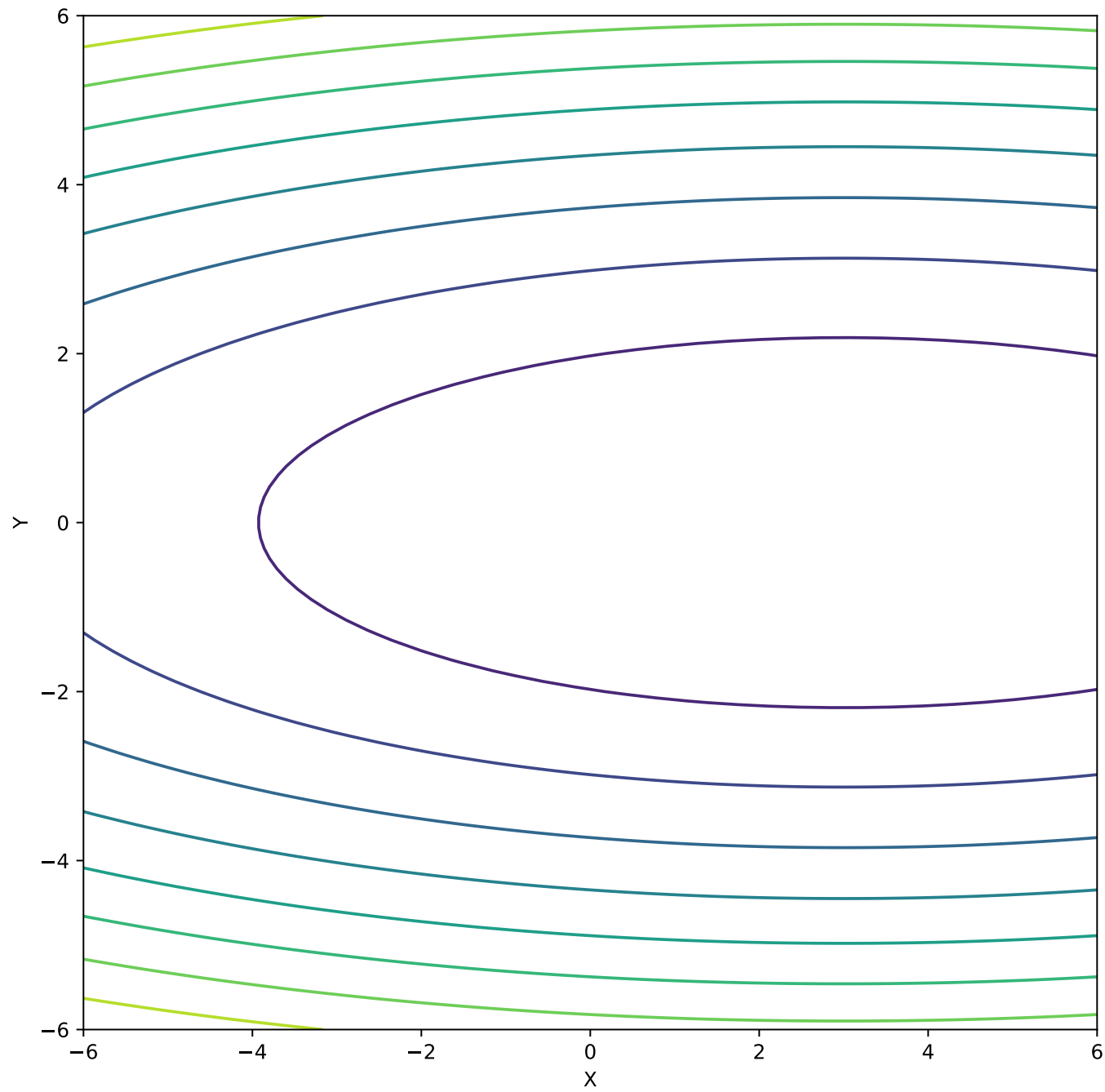
**Example** a function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  (on the left) and its level-sets (on both plots).



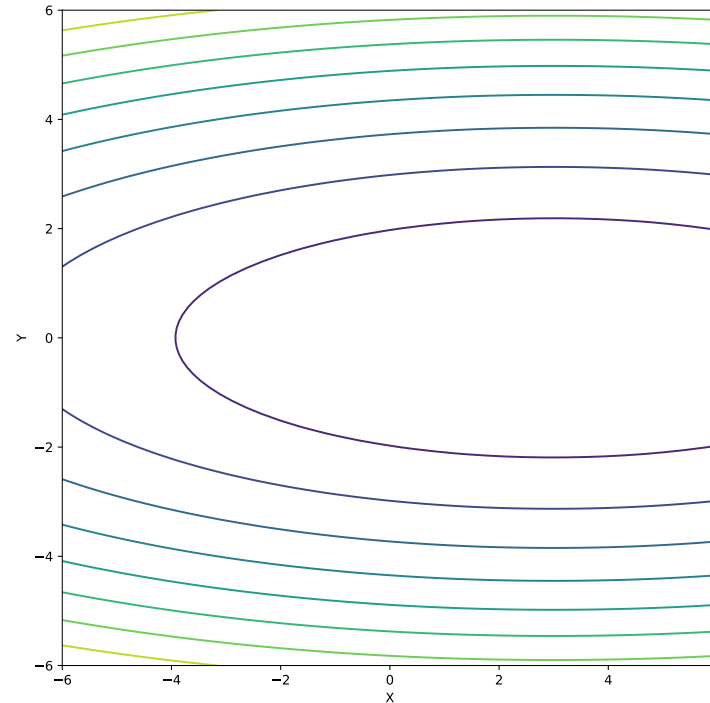
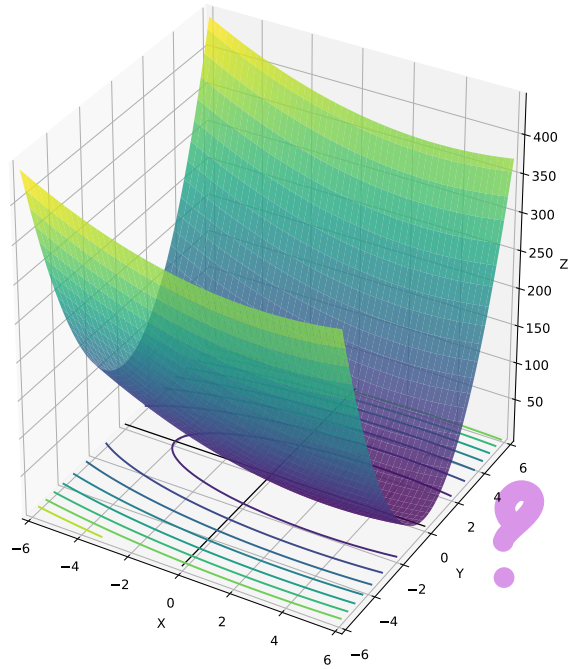
On the right-hand plot:

- continuous lines indicate the level-sets (points on which  $f$  has the same value).
- colors indicate the corresponding value.

## Reading in the level sets 1/3



# Reading in the level sets 1/3



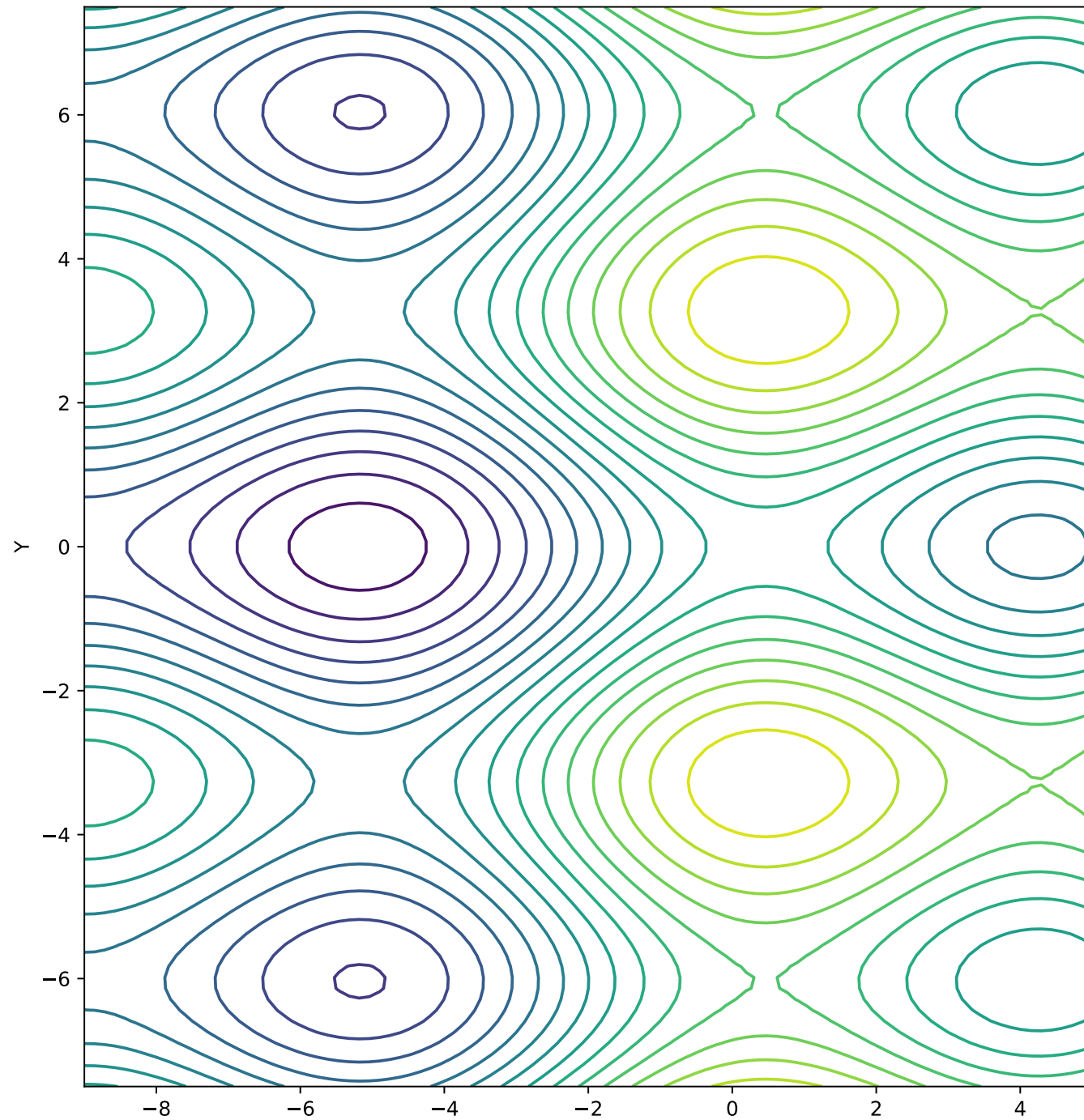
On this graph:

- ① the level sets  $\rightarrow$  ellipsoids with the same center
- ②  $\rightsquigarrow$  quadratic function with Hessian matrix  $\neq I_d$ .

Property: The level sets of a quadratic function are ellipsoids with the same center and axis.

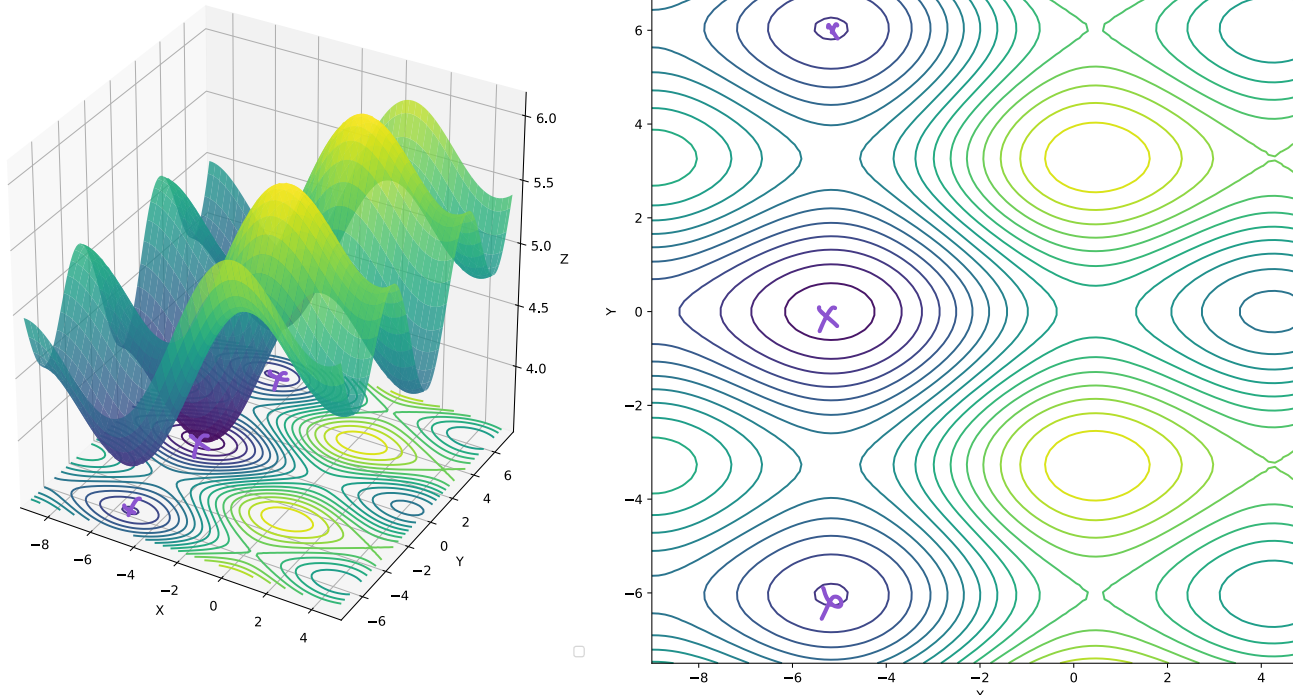
(reciprocal not true!)

## Reading in the level sets 2/3



## Reading in the level sets 2/3

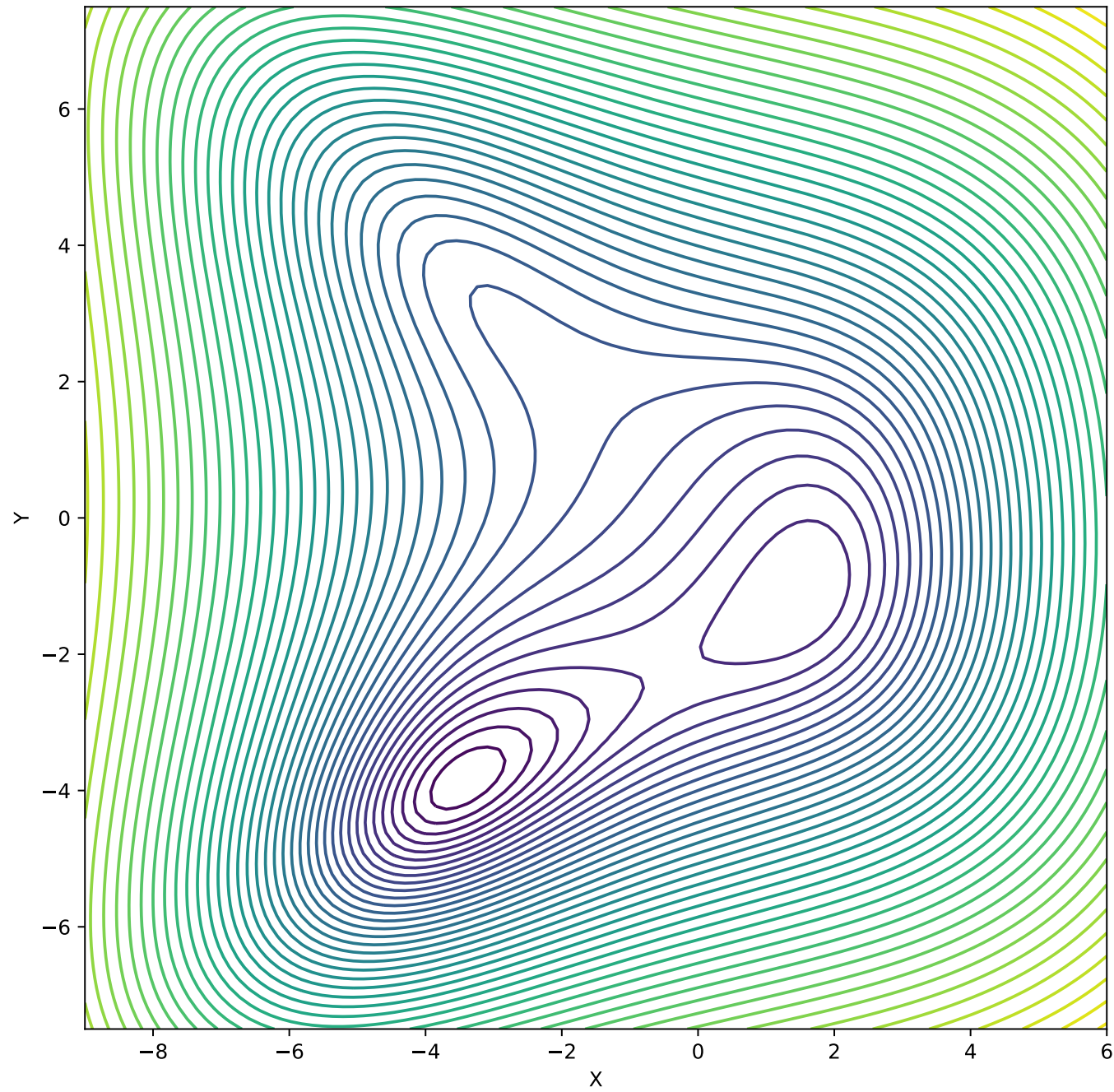
3 local minima



On this graph:

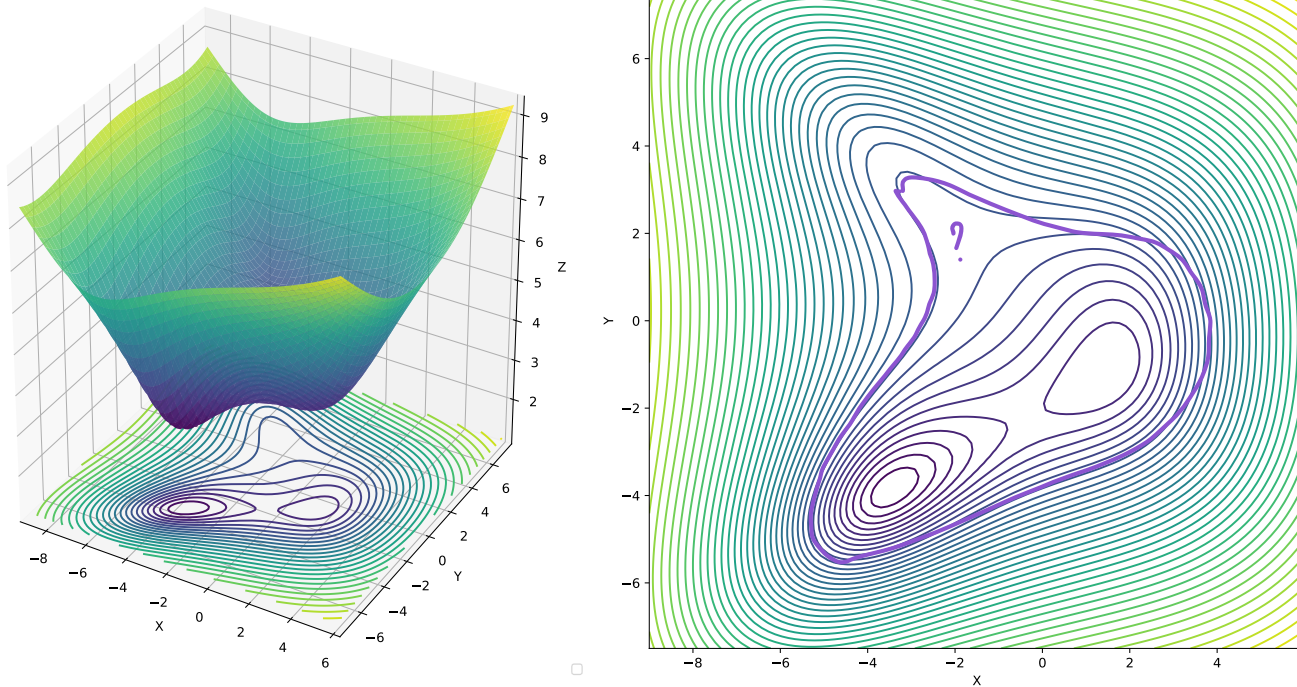
- 1  $f$  has multiple local minima and local maxima.
- 2  $f$  has periodic oscillations (cos/sin dependence?).
- 3  $f$  increases as the first coordinate increases.

## Reading in the level sets 3/3





# Reading in the level sets 3/3



On this graph:

- 1  $f$  has 2 local minima. (maybe 3)
- 2  $f$  the function is non convex. Indeed

Property: all level sets of a convex function are convex.

- 3  $f$  increases as point goes away from the center.

non  
conv level  
set  
 $\Rightarrow$  non conv  
fct

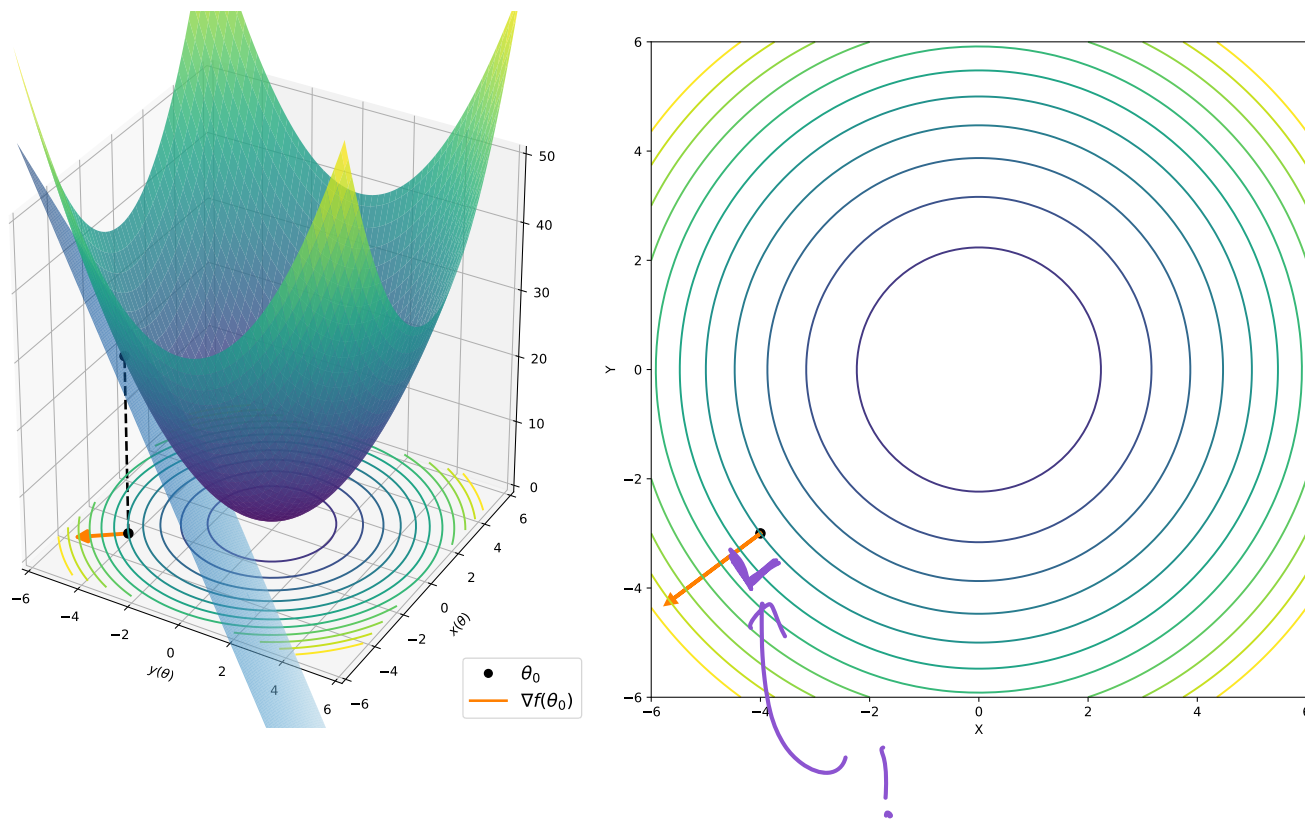


# Gradients and level sets

As seen before:

- 1 the gradient points to the highest increase direction.
- 2 the level sets correspond to points where the function is constant.

**Property:** The gradient is orthogonal to the level set.

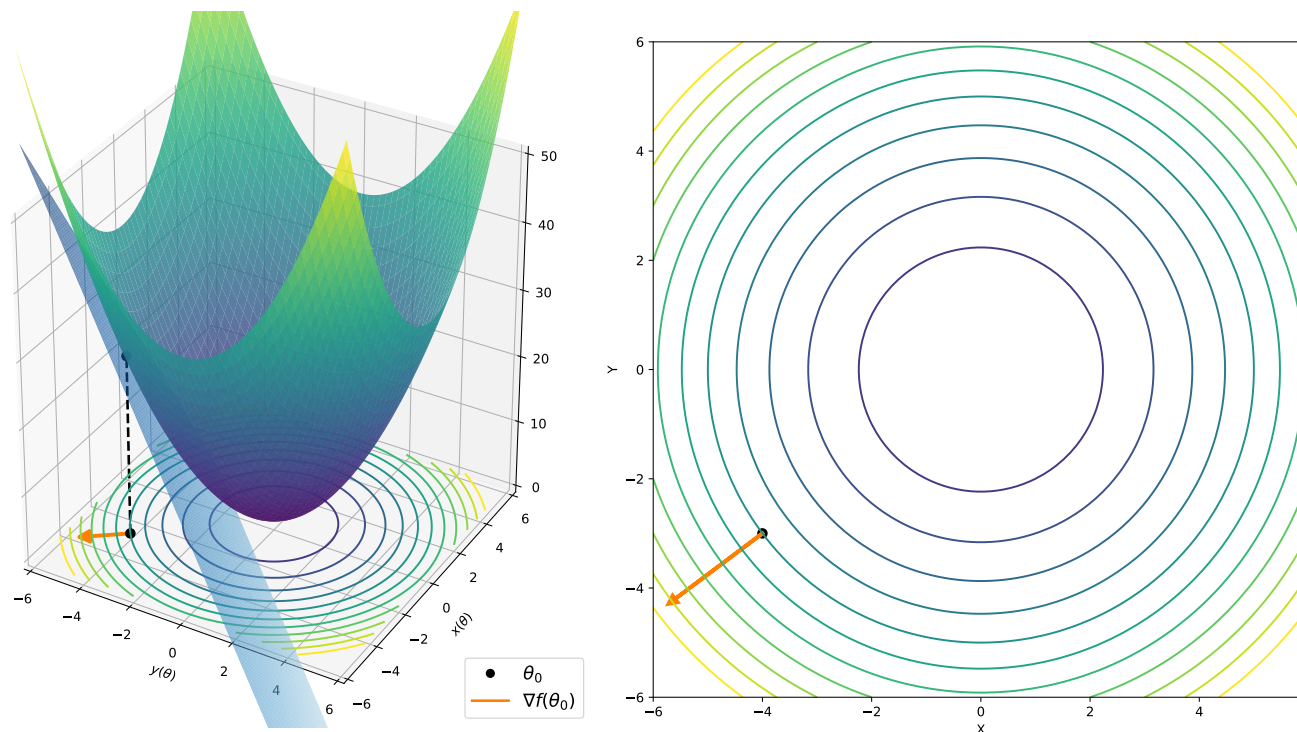


# Gradients and level sets

As seen before:

- 1 the gradient points to the highest increase direction.
- 2 the level sets correspond to points where the function is constant.

Property: The gradient is orthogonal to the level set.



Proof: Locally near  $\theta_0$ ,

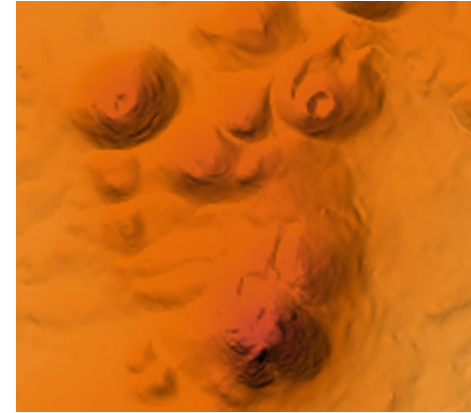
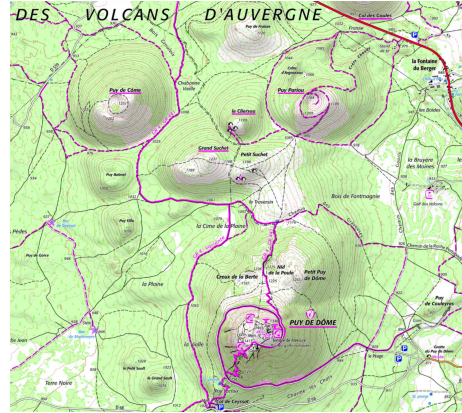
$$f(\theta) = f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + O(\|\theta - \theta_0\|^2).$$

Thus, for all  $\theta \in \mathcal{C}_{f(\theta_0)}$ ,

$$\lim_{\theta \rightarrow \theta_0, \theta \in \mathcal{C}_{f(\theta_0)}} \left\langle \nabla f(\theta_0), \frac{\theta - \theta_0}{\|\theta - \theta_0\|} \right\rangle = 0.$$

# Level sets - intuition from the altitude

Level sets are use everywhere on topographic maps!<sup>†</sup>

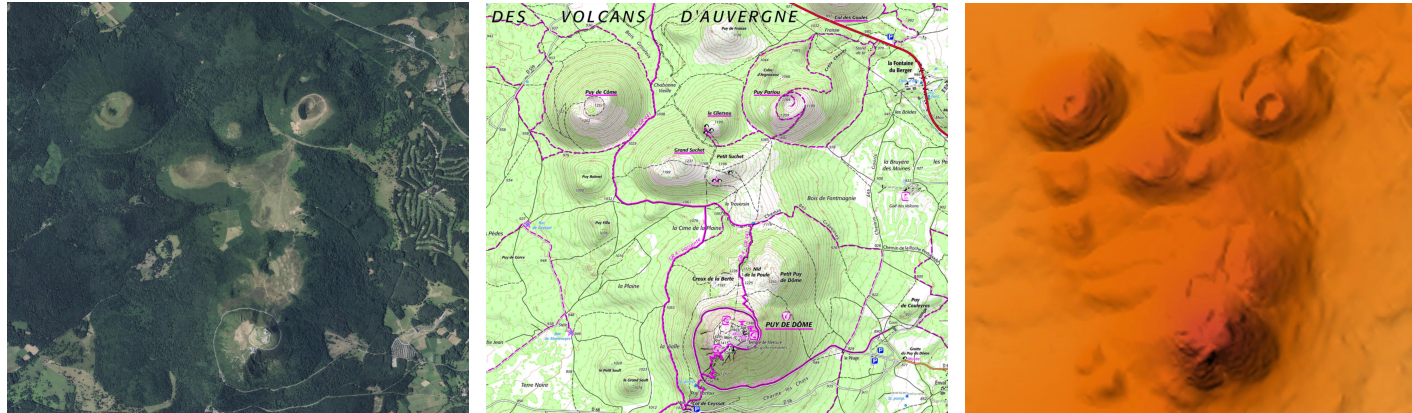


---

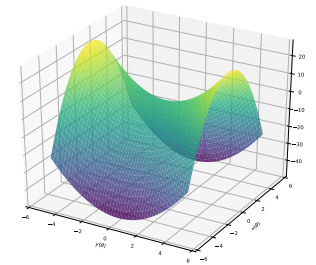
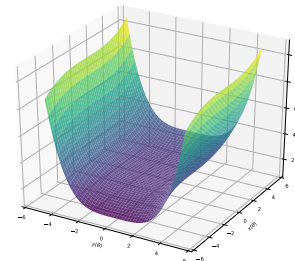
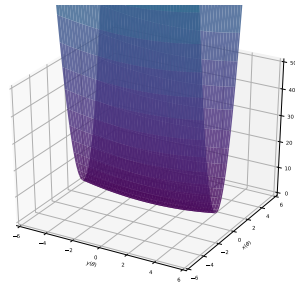
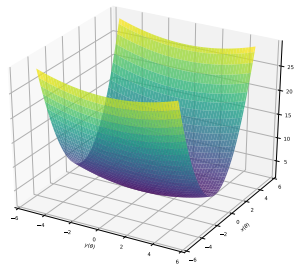
<sup>†</sup>source: geoportail

# Level sets - intuition from the altitude

Level sets are use everywhere on topographic maps!<sup>‡</sup>



→ informal description of some typical shapes of the *loss landscape*.

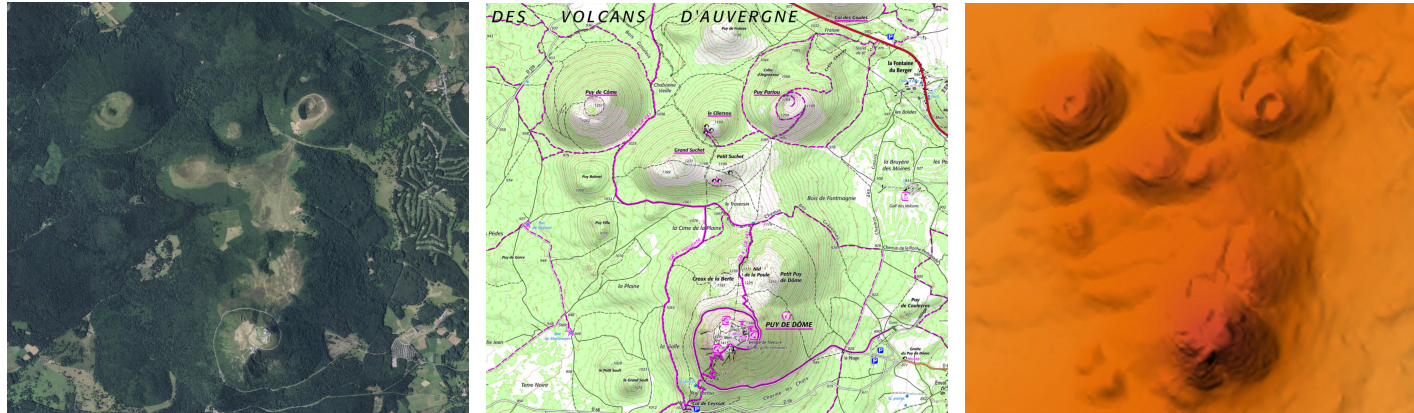


<sup>‡</sup>source: geoportail

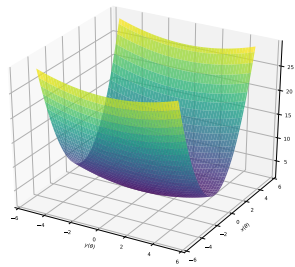


# Level sets - intuition from the altitude

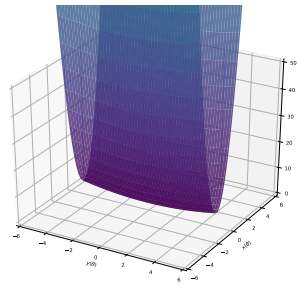
Level sets are use everywhere on topographic maps!<sup>†</sup>



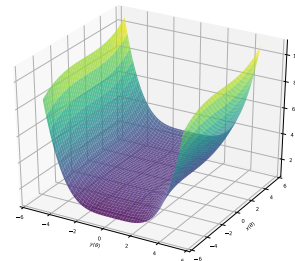
→ informal description of some typical shapes of the *loss landscape*.



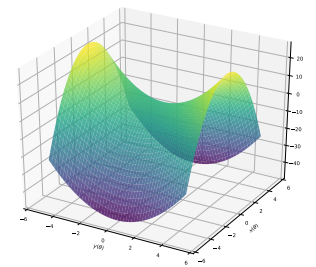
A valley



A narrow valley



A flat area



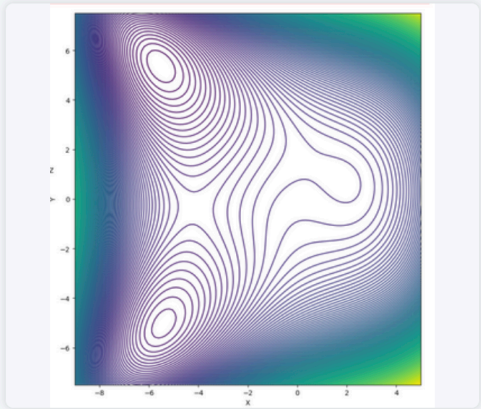
A saddle point (“Point col” in French)

<sup>†</sup>source: geoportail

# Let's recap! Wooclap quizz!

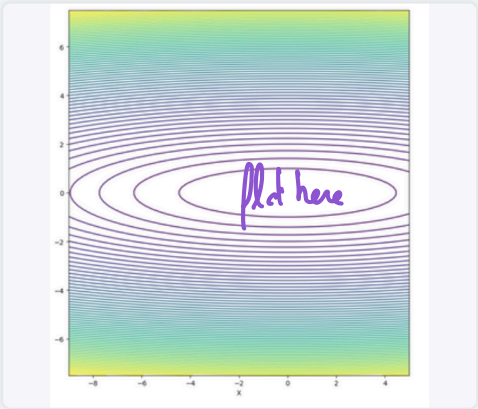
Convexity

all level sets  
are convex



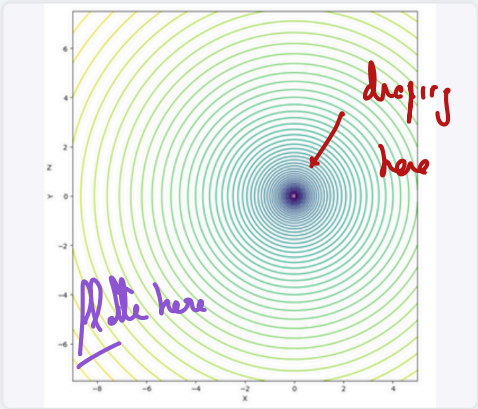
1

A



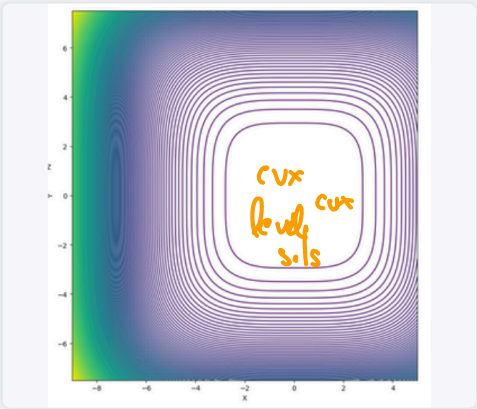
2

B



3

C



4

D

Convex, flat around optimum

Quadratic convex

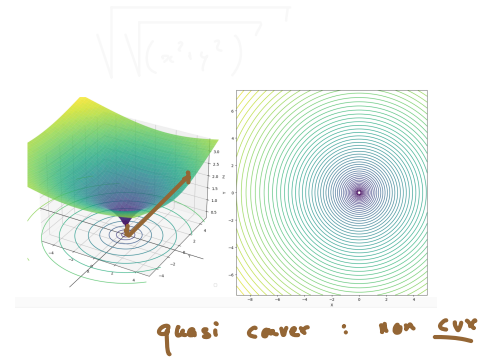
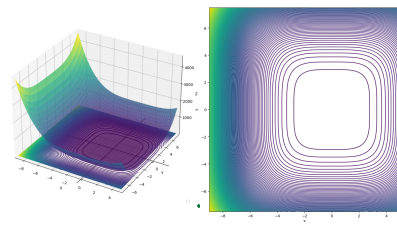
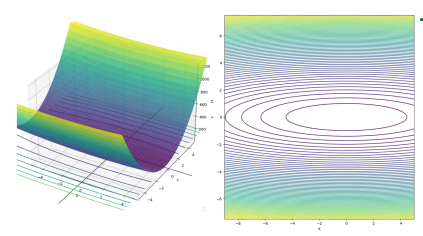
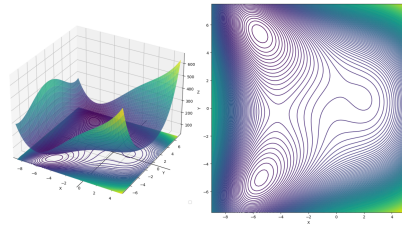
Convex level sets, non convex

Non convex level sets, non convex









# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
  - Basic properties
  - Visualization
  - Gradients, Hessians, and optimality conditions
  - Regularity conditions
- 3 Gradient descent method, stochastic gradient method

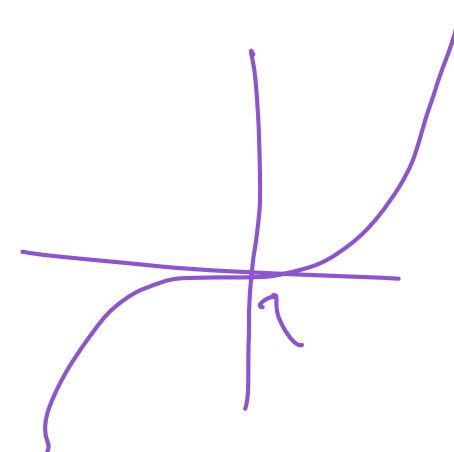
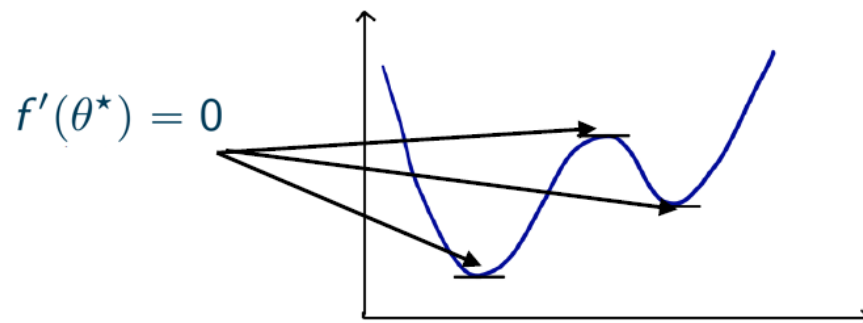
# Necessary condition in dimension 1

Local extrema  $\Rightarrow$  critical pt

## First order necessary condition

- In dimension one.

Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a differentiable function. If  $\theta^*$  is a local extremum (minimum/maximum) then  $f'(\theta^*) = 0$ .



## Remark.

- Points such that  $\nabla f(\theta^*) = 0$  are called critical points.
- Critical points are not always extrema (consider  $\theta \mapsto \theta^3$ )

## Optimality conditions in higher dimension

Generalization for  $d > 1$ .

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a differentiable function. If  $\theta^*$  is a local extremum then  $\nabla f(\theta^*) = 0$ .



Assume that  $f$  is twice continuously differentiable ( $C^2$ ).

### Necessary condition

If  $\theta^*$  is a local minimum, then  $\nabla f(\theta^*) = 0$  and  $\nabla^2 f(\theta^*)$  is positive semi-definite.

## Optimality conditions in higher dimension

Generalization for  $d > 1$ .

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a differentiable function. If  $\theta^\star$  is a local extremum then  $\nabla f(\theta^\star) = 0$ .

Assume that  $f$  is twice continuously differentiable ( $C^2$ ).

### Necessary condition

If  $\theta^\star$  is a local minimum, then  $\nabla f(\theta^\star) = 0$  and  $\nabla^2 f(\theta^\star)$  is positive semi-definite.

### Sufficient condition

If  $\nabla f(\theta^\star) = 0$  and  $\nabla^2 f(\theta^\star)$  is positive definite then  $\theta^\star$  is a strict local optimum.

## Optimality conditions in higher dimension

Generalization for  $d > 1$ .

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a differentiable function. If  $\theta^\star$  is a local extremum then  $\nabla f(\theta^\star) = 0$ .

Assume that  $f$  is twice continuously differentiable ( $C^2$ ).

### Necessary condition

If  $\theta^\star$  is a local minimum, then  $\nabla f(\theta^\star) = 0$  and  $\nabla^2 f(\theta^\star)$  is positive semi-definite.

### Sufficient condition

If  $\nabla f(\theta^\star) = 0$  and  $\nabla^2 f(\theta^\star)$  is positive definite then  $\theta^\star$  is a strict local optimum.

**Remark.** For  $d = 1$ , this condition boils down to  $f'(\theta^\star) = 0$  and  $f''(\theta^\star) > 0$ .

# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
  - Basic properties
  - Visualization
  - Gradients, Hessians, and optimality conditions
  - Regularity conditions
- 3 Gradient descent method, stochastic gradient method

# Regularity

Regularity can be thought as how much information I gain from a local observation of the function on its behavior beyond that point (or even, everywhere)

- ① Convexity
  - ②  $L$ -smoothness
  - ③  $\mu$ -strong convexity
  - ④ Lipschitz, etc.
- } condition number  $\kappa$

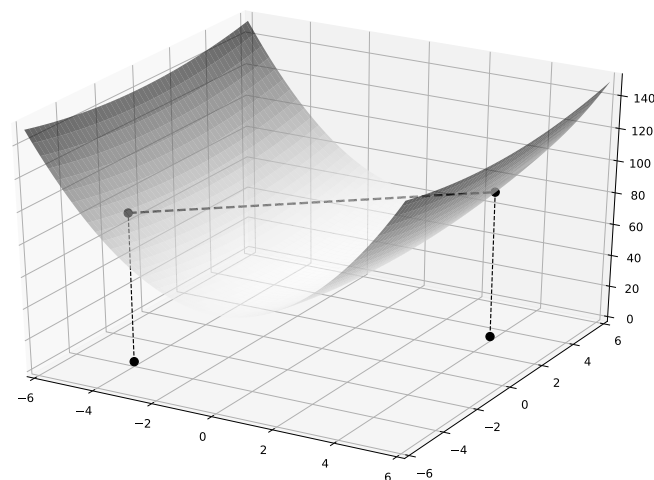
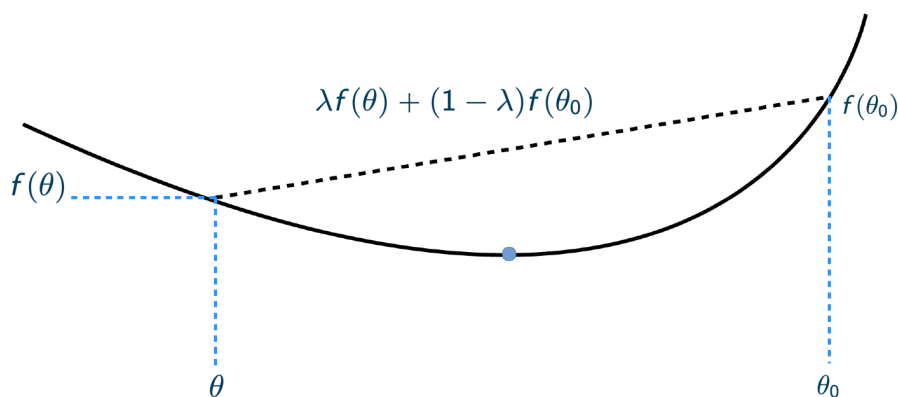


# 1. Convexity

## Definition: Convexity - Three characterizations

- ① We say that  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if ( $\mathbb{R}^d$  is convex and if)
- $$f(\lambda\theta + (1 - \lambda)\theta_0) \leq \lambda f(\theta) + (1 - \lambda)f(\theta_0), \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d, \lambda \in [0, 1].$$

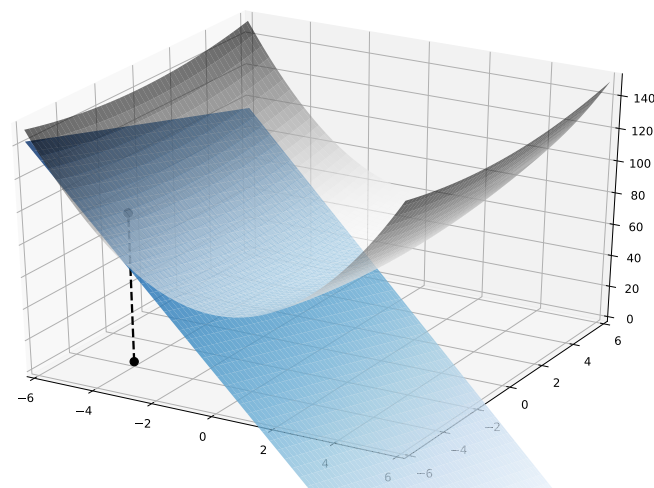
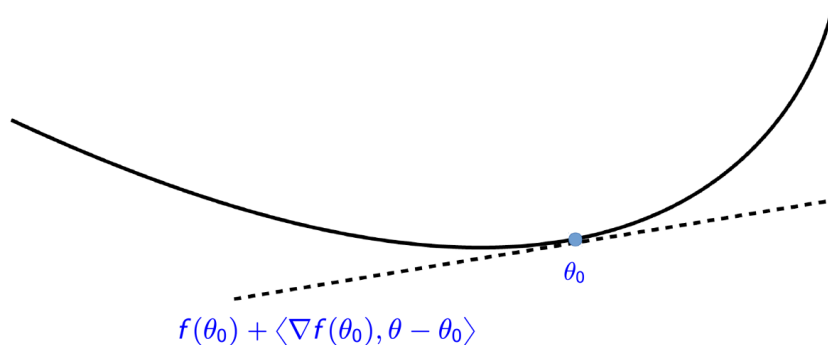
$$f_{\text{under}}(\text{average}) \leq \text{average}(f_{\text{under}} \text{ values})$$



# 1. Convexity

## Definition: Convexity - Three characterizations

- 1 We say that  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if ( $\mathbb{R}^d$  is convex and if)  
$$f(\lambda\theta + (1 - \lambda)\theta_0) \leq \lambda f(\theta) + (1 - \lambda)f(\theta_0), \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d, \lambda \in [0, 1].$$
- 2 A differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if and only if  
$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle \leq f(\theta) \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$

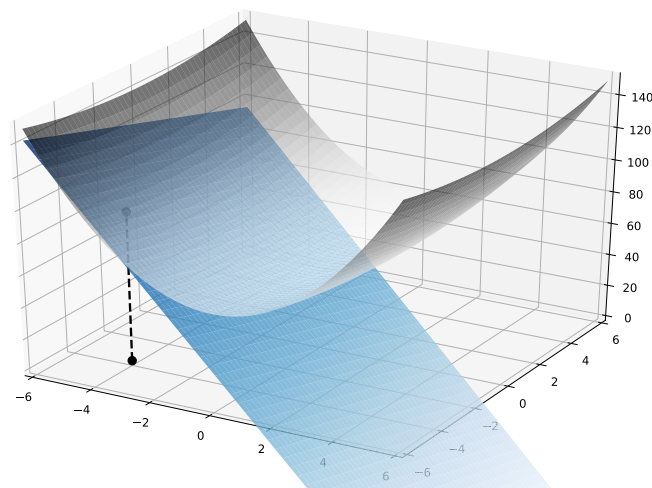
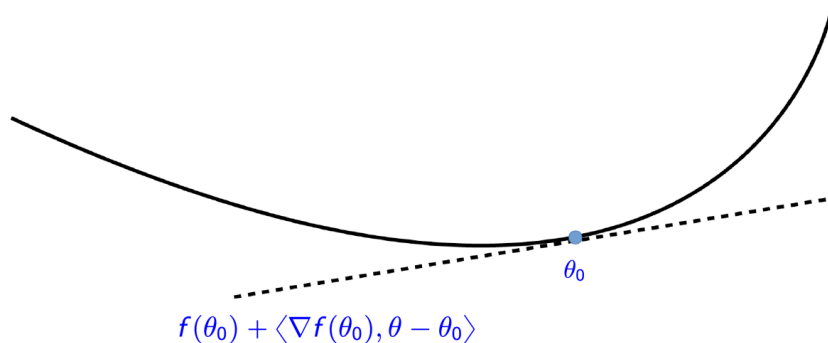


Property: For a convex function, any local minimum is a global minimum.

# 1. Convexity

## Definition: Convexity - Three characterizations

- ① We say that  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if ( $\mathbb{R}^d$  is convex and if)  
$$f(\lambda\theta + (1 - \lambda)\theta_0) \leq \lambda f(\theta) + (1 - \lambda)f(\theta_0), \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d, \lambda \in [0, 1].$$
- ② A differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if and only if  
$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle \leq f(\theta) \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$
- ③ A twice differentiable function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex if and only if  
$$\nabla^2 f(\theta) \succeq 0, \quad \text{for all } \theta,$$
  
that is  $h^T \nabla^2 f(\theta) h \geq 0$ , for all  $h \in \mathbb{R}^d$ .



Property: For a convex function, any local minimum is a global minimum.

## 2. Smoothness

### Definition: Smoothness

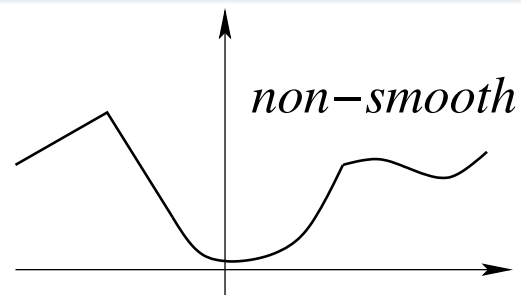
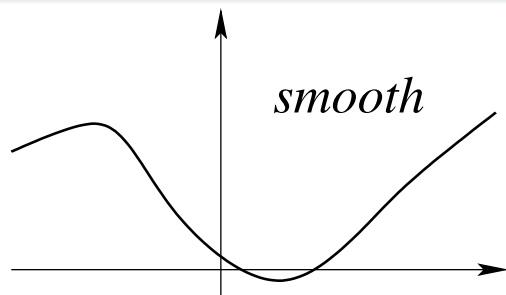
- A differentiable function  $f$  is  **$L$ -smooth** iff  $\nabla f$  is  $L$ -Lipshitz, i.e.  
$$\|\nabla f(\theta) - \nabla f(\theta_0)\| \leq L\|\theta - \theta_0\| \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$

## 2. Smoothness

### Definition: Smoothness

- A differentiable function  $f$  is  **$L$ -smooth** iif  $\nabla f$  is  $L$ -Lipshitz, i.e.

$$\|\nabla f(\theta) - \nabla f(\theta_0)\| \leq L\|\theta - \theta_0\| \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$



## 2. Smoothness

### Definition: Smoothness

*def missing*

- A differentiable function  $f$  is  **$L$ -smooth** iif  $\nabla f$  is  $L$ -Lipshitz, i.e.

$$\|\nabla f(\theta) - \nabla f(\theta_0)\| \leq L\|\theta - \theta_0\| \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$

**Smooth-convex:** the function **above** the tangent and **below** the tangent line + quadratic:

### Proposition: Smoothness + convexity

- A differentiable function  $f$  is  **$L$ -smooth** and convex iif for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle \leq f(\theta) \leq f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2$$

QUB

$$\frac{L}{2} (\theta - \theta_0)^T \underset{\text{Identity}}{I} (\theta - \theta_0)$$

## 2. Smoothness

### Definition: Smoothness

- A differentiable function  $f$  is  **$L$ -smooth** iff  $\nabla f$  is  $L$ -Lipshitz, i.e.  

$$\|\nabla f(\theta) - \nabla f(\theta_0)\| \leq L\|\theta - \theta_0\| \quad \text{for all } \theta, \theta_0 \in \mathbb{R}^d.$$

**Smooth-convex:** the function **above** the tangent and **below** the tangent line + quadratic:

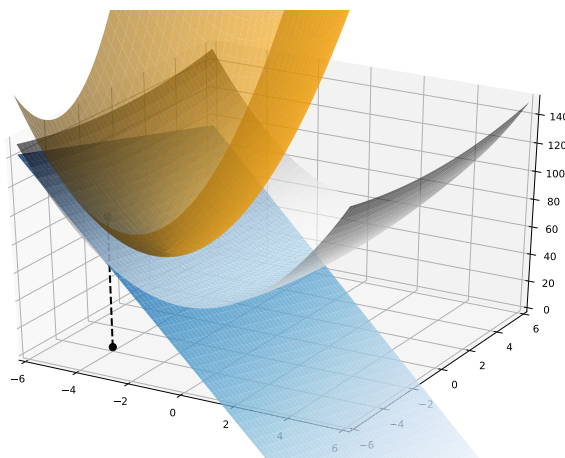
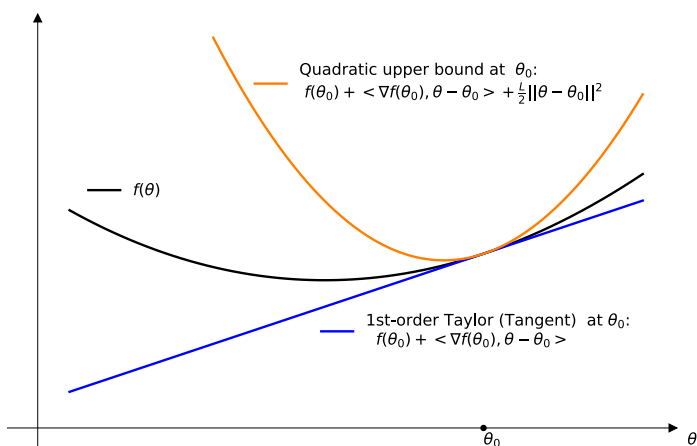
### Proposition: Smoothness + convexity

- A differentiable function  $f$  is  **$L$ -smooth** and convex iff for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle \leq f(\theta) \leq \boxed{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle} + \underbrace{\frac{L}{2} \|\theta - \theta_0\|^2}_{\text{quadratic isotropic bound}}$$

FO Taylor approx.

quadratic isotropic bound



Similarly, observing  $f$  and  $\nabla f$  at  $\theta_0$  provides information on the function *everywhere*.

### 3. Strong convexity

#### Definition: Strong convexity - Two characterizations

$f$  is  $\mu$ -strongly convex iif

- $\theta \mapsto f(\theta) - \frac{\mu}{2}\|\theta\|^2$  is convex.

- equivalently, for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{\mu}{2}\|\theta - \theta_0\|^2 \leq f(\theta)$$

Quadratic Lower Bound!



### 3. Strong convexity

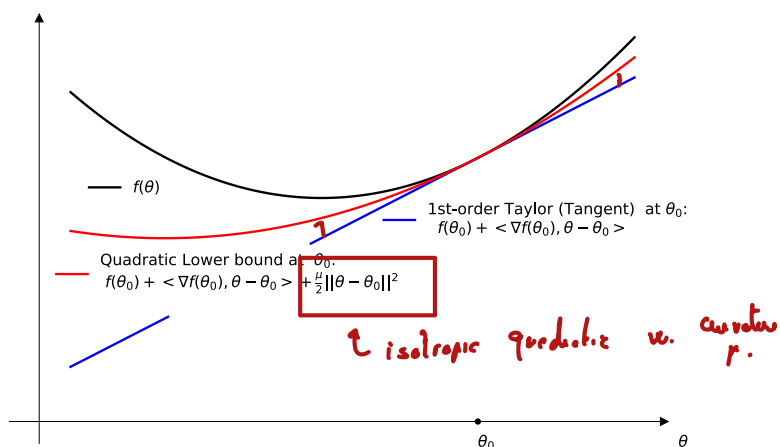
#### Definition: Strong convexity - Two characterizations

$f$  is  $\mu$ -strongly convex iif

- $\theta \mapsto f(\theta) - \frac{\mu}{2}\|\theta\|^2$  is convex.

- equivalently, for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{\mu}{2}\|\theta - \theta_0\|^2 \leq f(\theta)$$



Again, observing  $f$  and  $\nabla f$  at  $\theta_0$  provides information on the function *everywhere*.

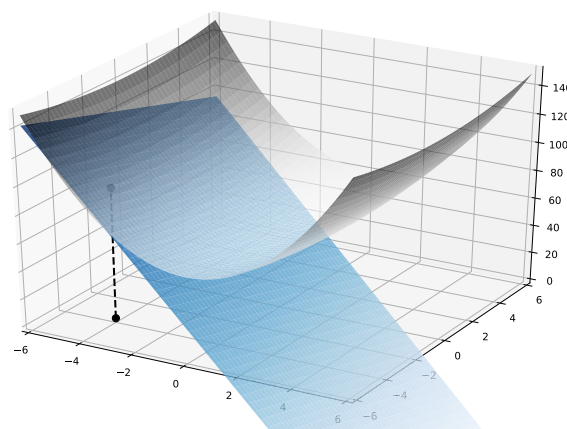
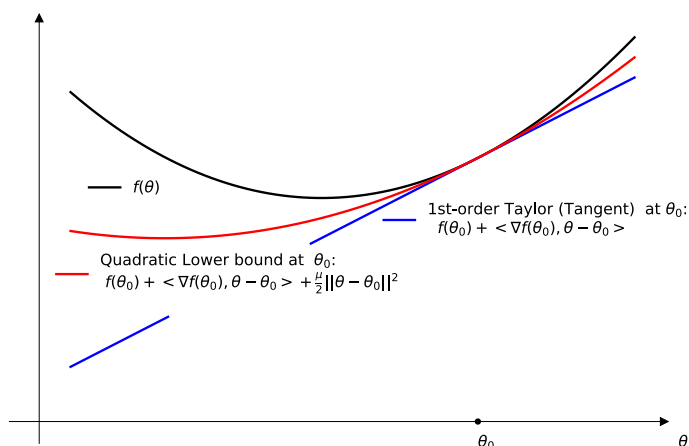
### 3. Strong convexity

#### Definition: Strong convexity - Two characterizations

$f$  is  $\mu$ -strongly convex iif

- $\theta \mapsto f(\theta) - \frac{\mu}{2}\|\theta\|^2$  is convex.
- equivalently, for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{\mu}{2}\|\theta - \theta_0\|^2 \leq f(\theta)$$



Again, observing  $f$  and  $\nabla f$  at  $\theta_0$  provides information on the function *everywhere*.

### 3. Strong convexity

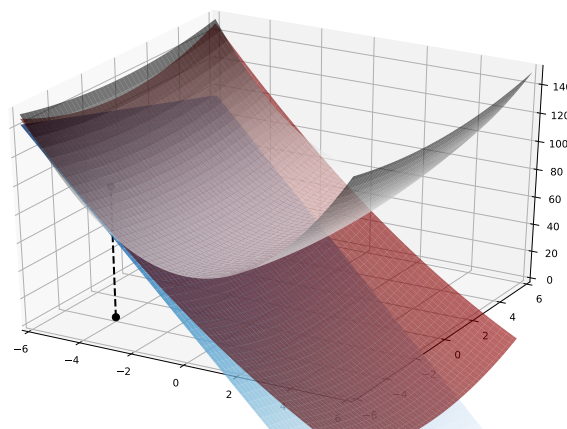
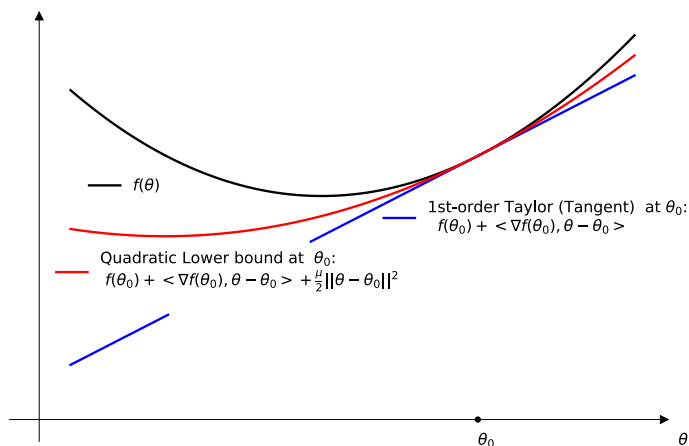
#### Definition: Strong convexity - Two characterizations

$f$  is  $\mu$ -strongly convex iif

- $\theta \mapsto f(\theta) - \frac{\mu}{2}\|\theta\|^2$  is convex.

- equivalently, for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{\mu}{2}\|\theta - \theta_0\|^2 \leq f(\theta)$$



Again, observing  $f$  and  $\nabla f$  at  $\theta_0$  provides information on the function *everywhere*.

## Some useful properties (advanced)

(optional)

(for proofs!)

### Characterization of twice differentiable $L$ -smooth and $\mu$ -strongly convex functions

A twice differentiable  $f$  is

- ①  $L$ -smooth iif for all  $\theta$ ,

$$\nabla^2 f(\theta) \preceq L\mathbf{I}_d \quad \Leftrightarrow \quad \lambda_{\max}(\nabla^2 f(\theta)) \leq L.$$

- ②  $L$ -smooth iif for all  $\theta$ ,

$$\nabla^2 f(\theta) \succeq \mu\mathbf{I}_d \quad \Leftrightarrow \quad \lambda_{\min}(\nabla^2 f(\theta)) \geq \mu.$$

### Co-coercivity

If  $f$  is  $L$ -smooth and convex then for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$\|\nabla f(\theta) - \nabla f(\theta_0)\|^2 \leq L \langle \nabla f(\theta_0) - \nabla f(\theta), \theta_0 - \theta \rangle$$

### Strong convexity

If  $f$  is  $\mu$ -strongly convex then for all  $\theta, \theta_0 \in \mathbb{R}^d$

$$\langle \nabla f(\theta_0) - \nabla f(\theta), \theta_0 - \theta \rangle \geq \mu \|\theta - \theta_0\|^2$$

# Conditioning (1/2)

$\kappa$  small (close to 1)  
→ easier to optimize!

## Condition number $\kappa$

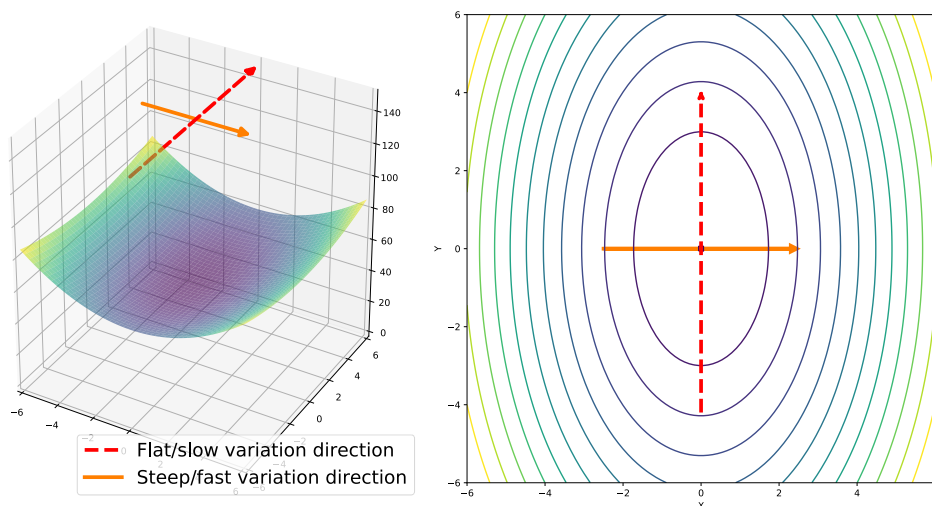
Let  $f$  be an  $L$ -smooth and  $\mu$  strongly convex function. The condition number is:

$$\kappa := \frac{L}{\mu}.$$

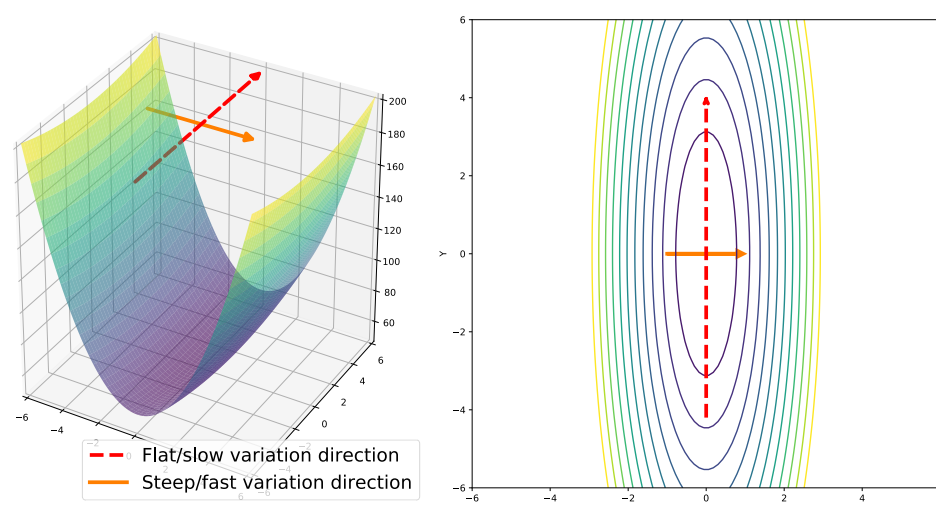
The condition number “quantifies” how difficult to optimize a function is:

- the smoothness  $L$  is given by the curvature on fastest variation direction
- the strong convexity  $\mu$  is given by the curvature on the slowest variation direction.

Well conditioned function  $\kappa = 1.5$



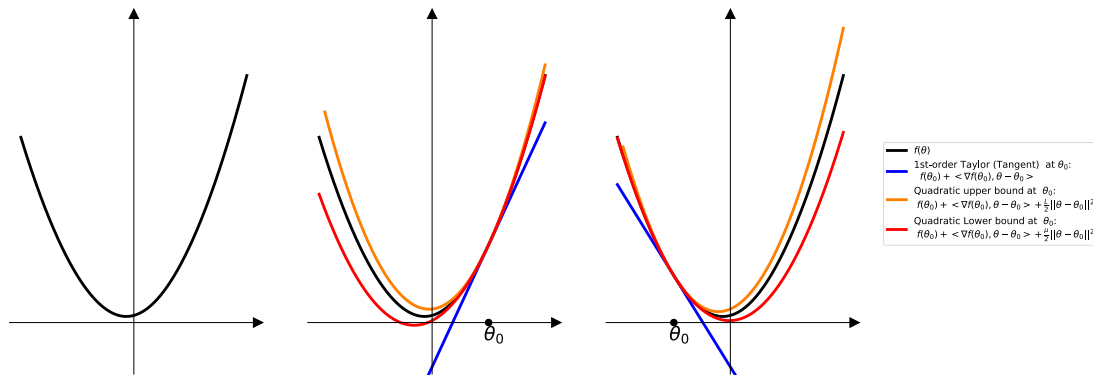
Poorly conditioned function  $\kappa = 10$



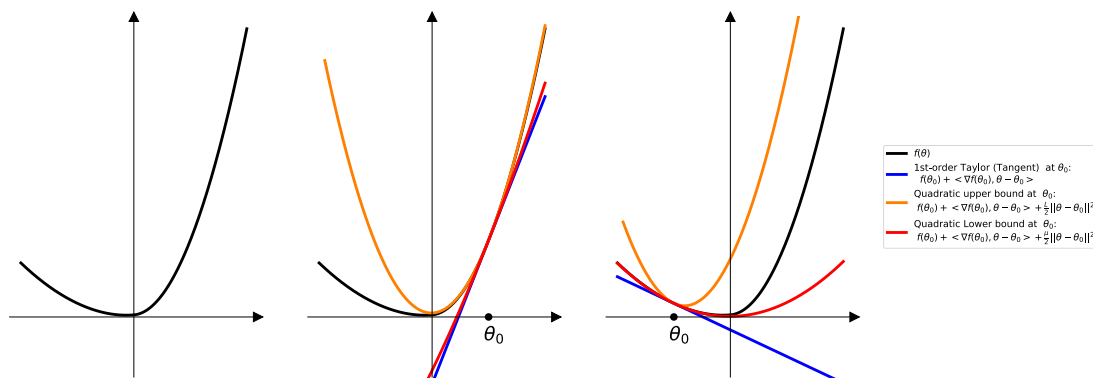
→  $\kappa$  large  $\leftrightarrow$  the scale at which  $f$  varies changes a lot (here, depending on the direction)

# Conditioning - dimension 1 visualization (2/2)

Well conditioned function  $\kappa = 1.25$

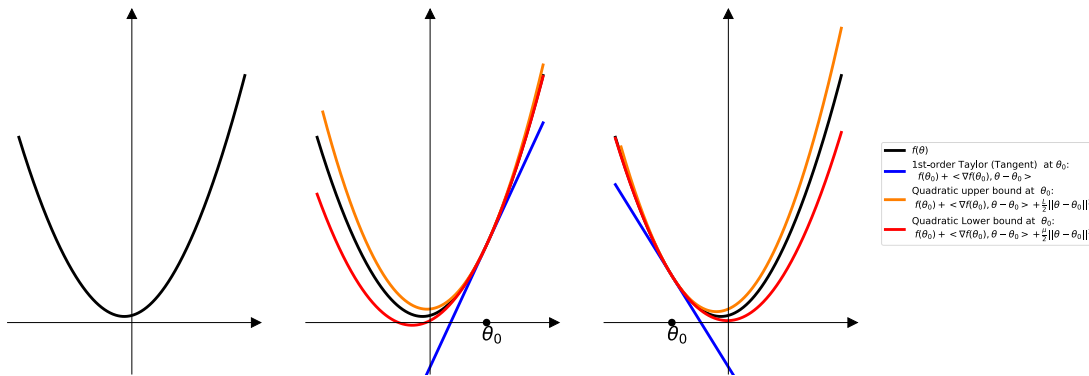


Poorly conditioned function  $\kappa = 5$

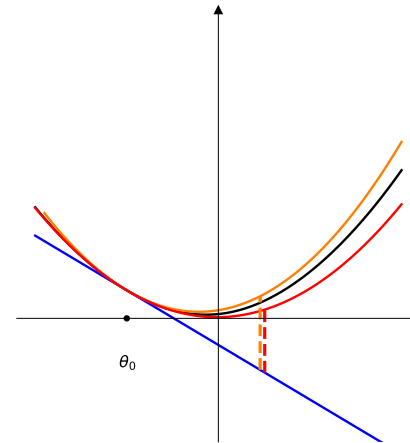


# Conditioning - dimension 1 visualization (2/2)

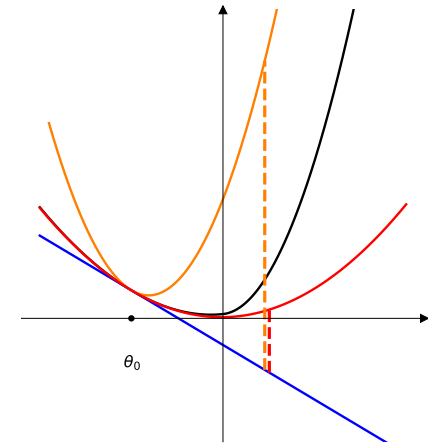
Well conditioned function  $\kappa = 1.25$



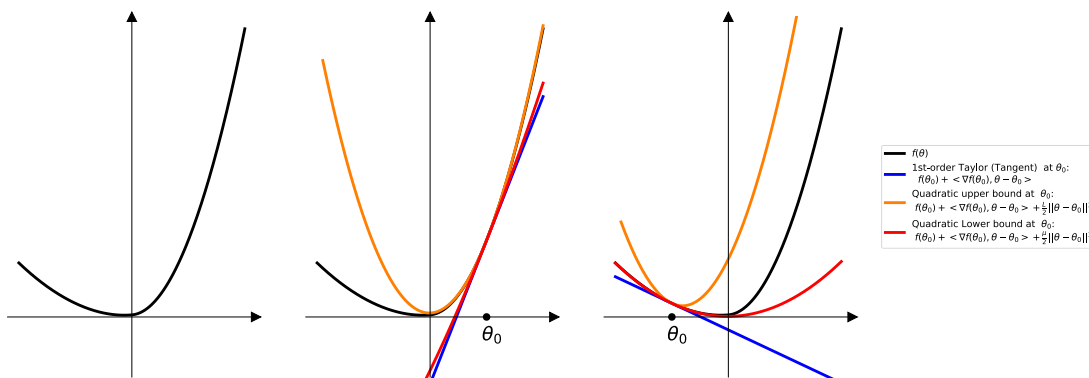
$\kappa = 1.25$



$\kappa = 5$



Poorly conditioned function  $\kappa = 5$



$\rightarrow \kappa = \frac{\text{Quadratic upper bound} - \text{Quadratic Lower bound}}{\text{1st-order Taylor (Tangent)}} = \text{ratio of differences to the tangent between quadratic upper and lower bounds}$

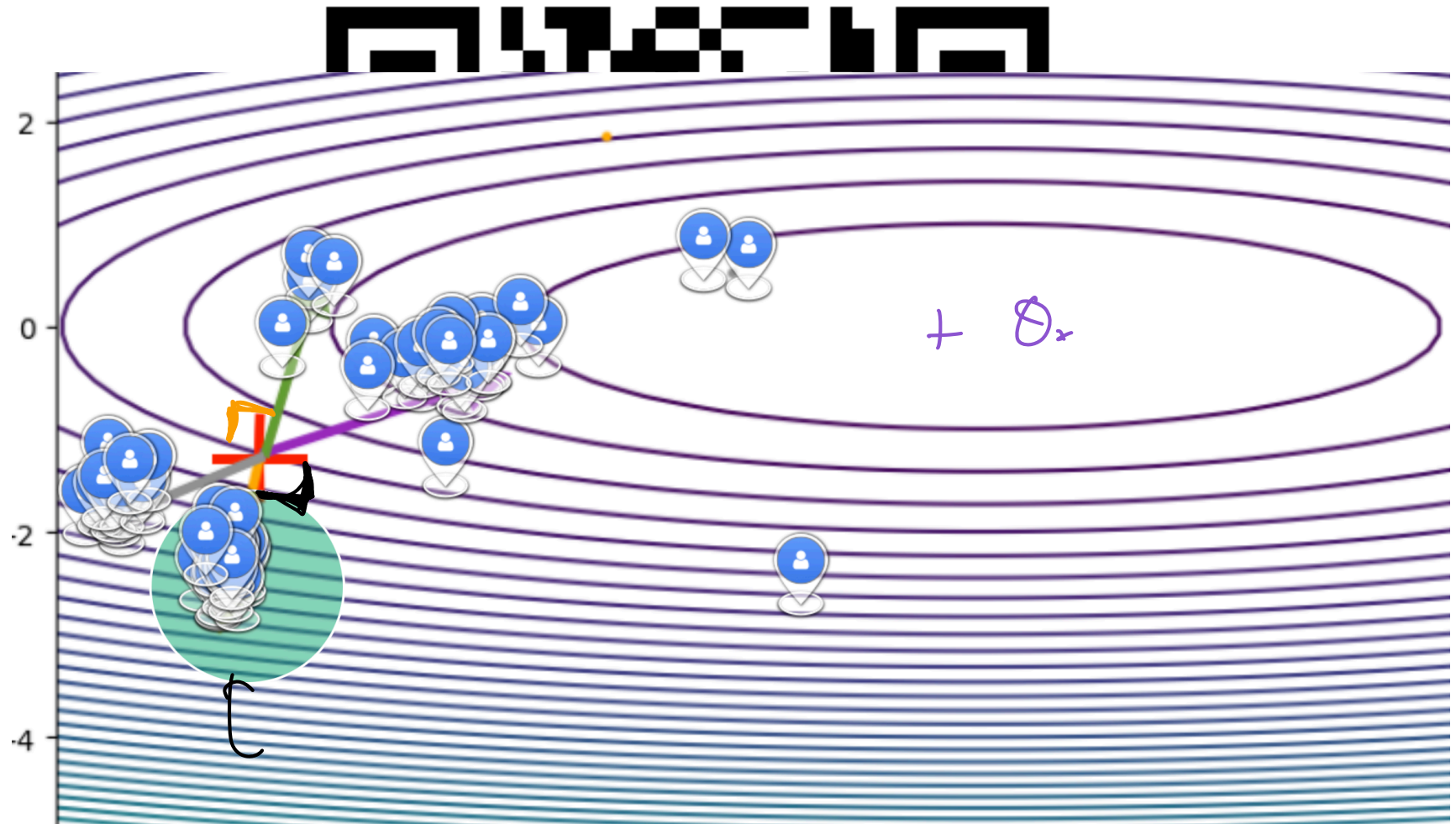
$\rightarrow \kappa \text{ small} \rightarrow \text{local information provides "better" global further.}$

$\rightarrow \kappa \text{ large} \leftrightarrow \text{the scale at which } f \text{ varies changes a lot (here, depending on the point)}$

# Let's recap! Wooclap quizz!

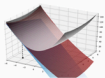
gradient  $\perp$  level set

gradient pls towards increase

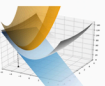




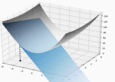
[Again] Which conditions do these graphs illustrate?



Red corresponds to



Orange corresponds to



Blue corresponds to

1

2

3

A

B

C

Fonction is convex

Fonction is smooth

Fonction is strongly convex



# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
  - End of lecture 1 - at your own pace quizz!

# First attempt: Exhaustive search

Consider the problem

$$\min_{\theta \in [0,1]^d} f(\theta)$$

One can optimize this problem on a grid of  $[0, 1]^d$ .

E.g., if the function  $f$  is regular enough, to achieve a precision of  $\varepsilon$

- in dimension 1, we need  $\lfloor 1/\varepsilon \rfloor$  evaluation of  $f$ ,
- in dimension  $d$ , we need  $\lfloor 1/\varepsilon \rfloor^d$  evaluations.

For example, minimizing

$$f(\theta) = \|\theta\|_2^2,$$

with a precision of  $\varepsilon = 10^{-2}$  requires:

- $1,75 \cdot 10^{-3}$  seconds in dimension 1
- $1,75 \cdot 10^{15}$  seconds in dimension 10, i.e., nearly 32 millions years.

→ Prohibitive in high dimensions

→ **Solution** Use **local information**.

Use local information: two classes of algorithms

(skipped slide)

**Key idea:** At any point  $\theta_0$  we can compute the value of the function  $f(\theta_0)$ , but also the direction in which the function increases the most  $\nabla f(\theta_0)$  and the curvature  $\nabla^2 f(\theta_0)$ .

## First-order methods/algorithms

↪ use  $f$  and  $\nabla f$ .

Standard algorithms when  $f$  is differentiable and convex.

## Second-order methods/algorithms

↪ use  $f$ ,  $\nabla f$  and  $\nabla^2 f$ .

→ useful when computing the Hessian matrix is not too costly.

First-order vs second-order → first fundamental characteristic of algorithms.

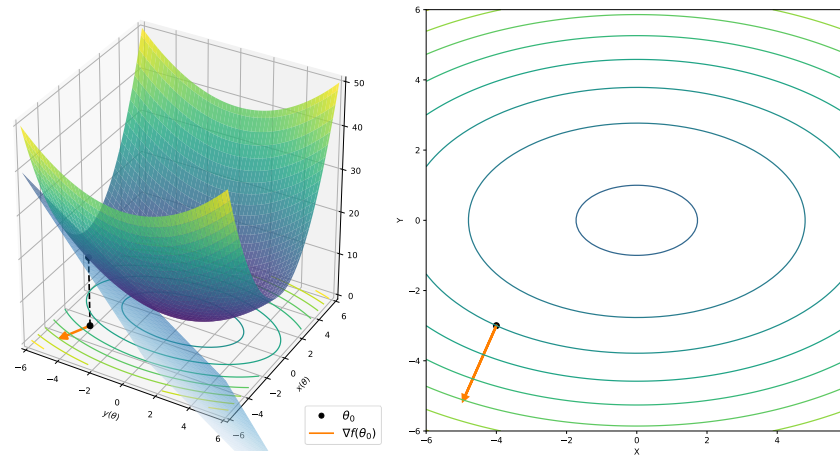
→ first focus on first order: what is a natural algorithm?

# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method
  - Gradient Descent (GD)
    - Gradient descent: algorithm and intuition
    - GD: theoretical grounds
    - Convergence rates for GD and interpretation
  - Stochastic Gradient Descent (SGD)
  - End of lecture 1 - at your own pace quizz!

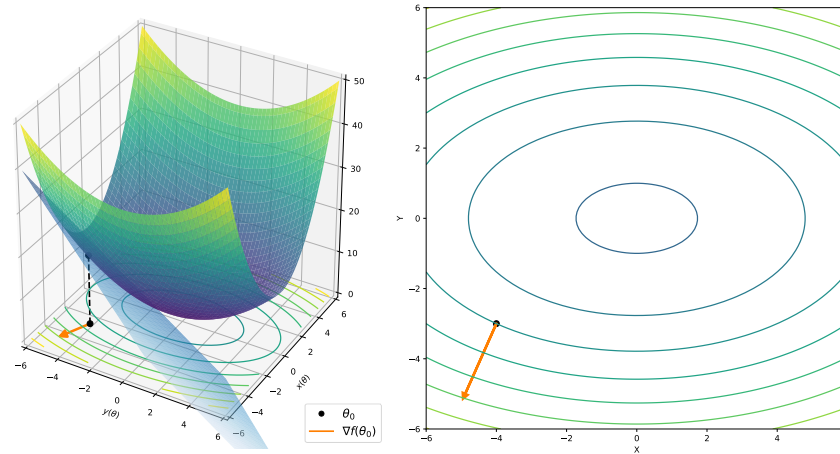
# Gradient Descent: intuition

Given an initial iterate  $\theta_0$  and information  $f(\theta_0), \nabla f(\theta_0)$  how to improve the model?



# Gradient Descent: intuition

Given an initial iterate  $\theta_0$  and information  $f(\theta_0), \nabla f(\theta_0)$  how to improve the model?

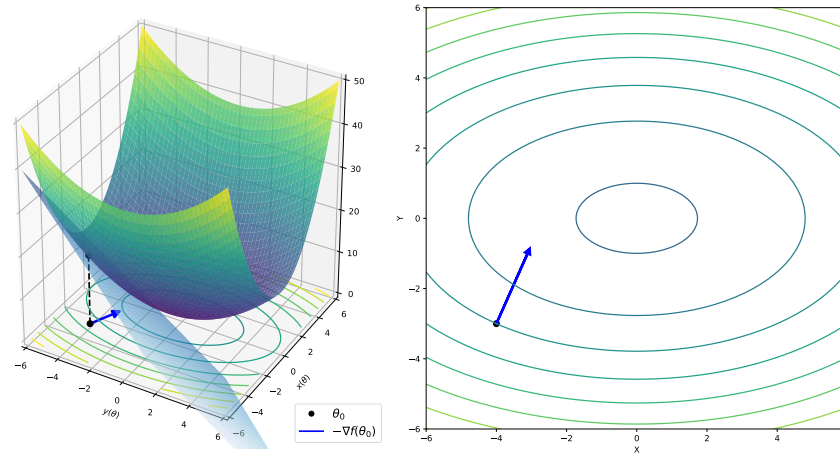


- $\nabla f(\theta_0)$  points in the steepest increase direction from  $\theta_0$



# Gradient Descent: intuition

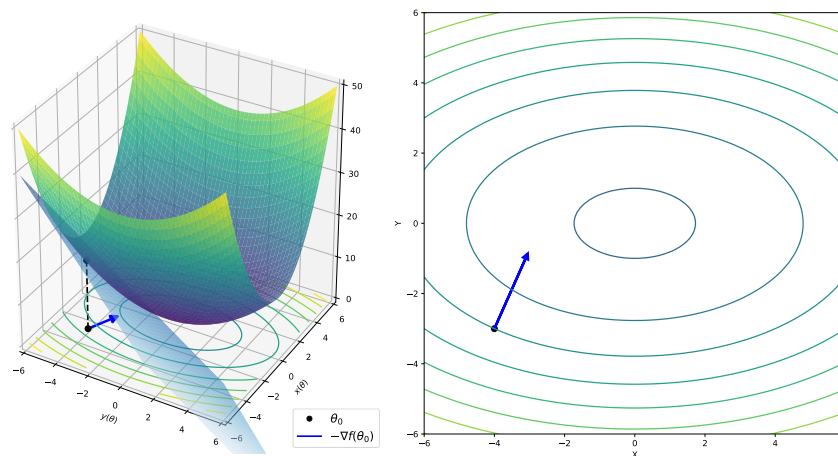
Given an initial iterate  $\theta_0$  and information  $f(\theta_0), \nabla f(\theta_0)$  how to improve the model?



- $\nabla f(\theta_0)$  points in the steepest increase direction from  $\theta_0$   
→  $-\nabla f(\theta_0)$  points in the steepest **decrease** direction from  $\theta_0$

# Gradient Descent: intuition

Given an initial iterate  $\theta_0$  and information  $f(\theta_0), \nabla f(\theta_0)$  how to improve the model?



- $\nabla f(\theta_0)$  points in the steepest increase direction from  $\theta_0$   
→  $-\nabla f(\theta_0)$  points in the steepest **decrease** direction from  $\theta_0$

## Algorithm: Gradient descent (GD)

**Input:** Function  $f$  to minimize.

**Initialization:** initial weight vector  $\theta_{(0)}$

**Hyper-parameters:** step size  $\eta > 0$ , number of iterations  $T$ .

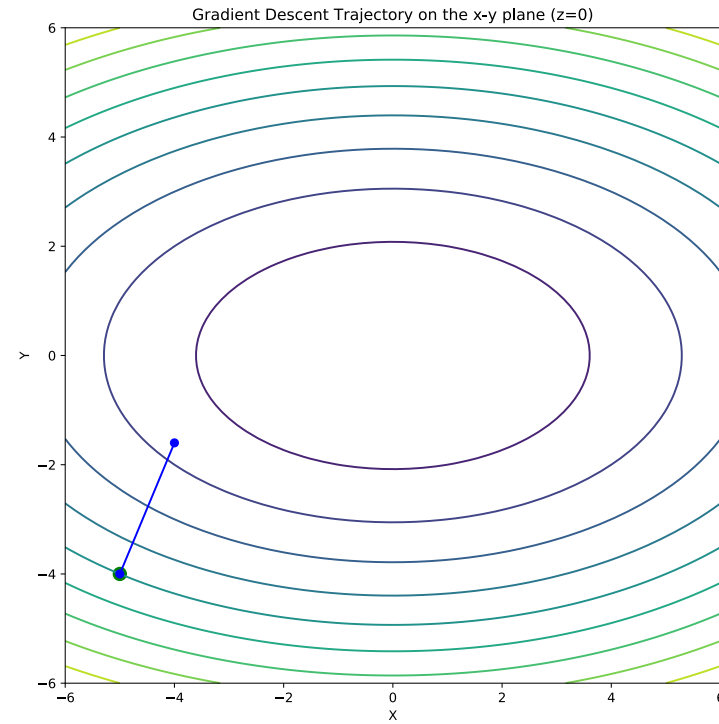
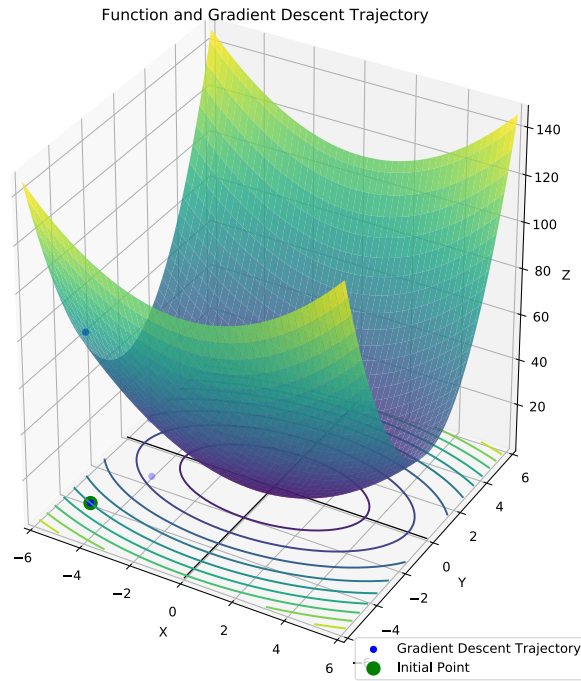
For  $t \in \{1, \dots, T\}$ :

- $\theta_{t+1} \leftarrow \theta_t - \eta \nabla f(\theta_t)$
- $t \leftarrow t + 1$ .

**Output:**  $\theta_T$ .

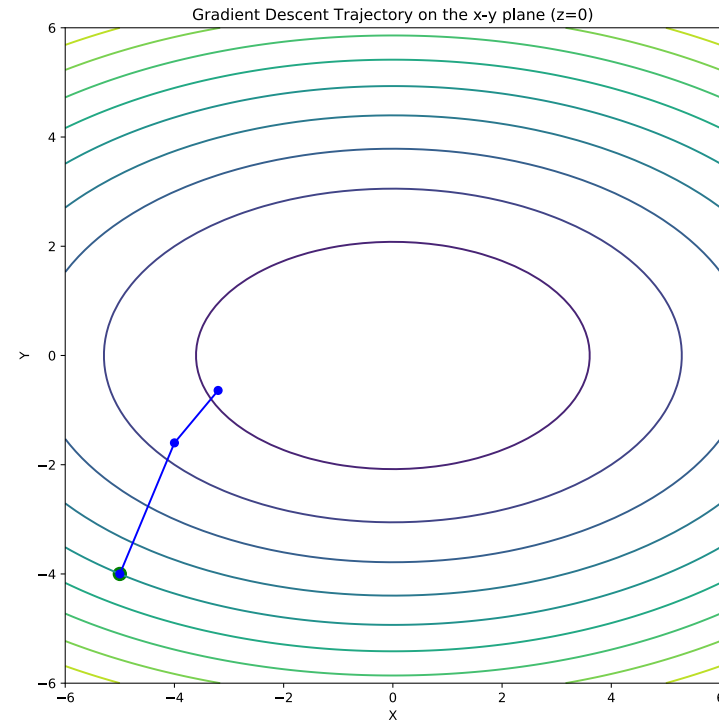
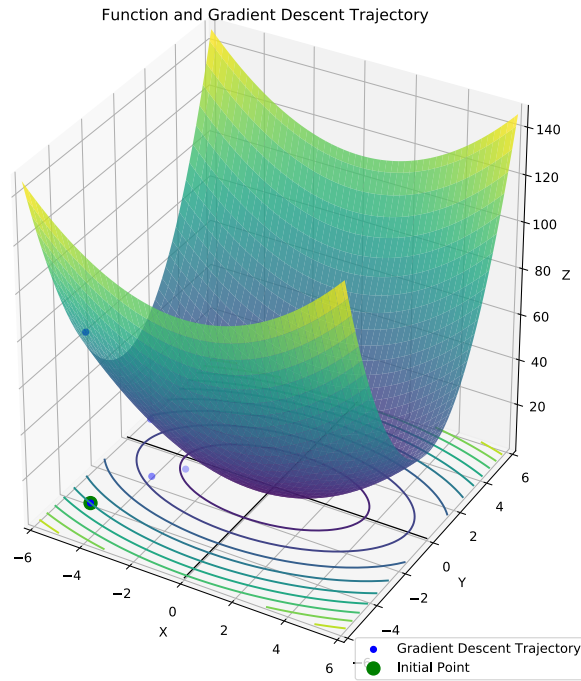
# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



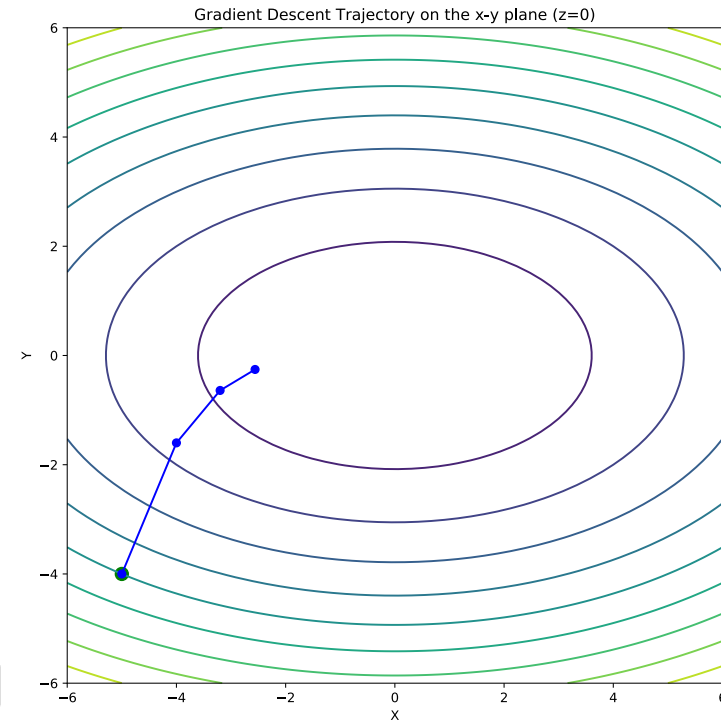
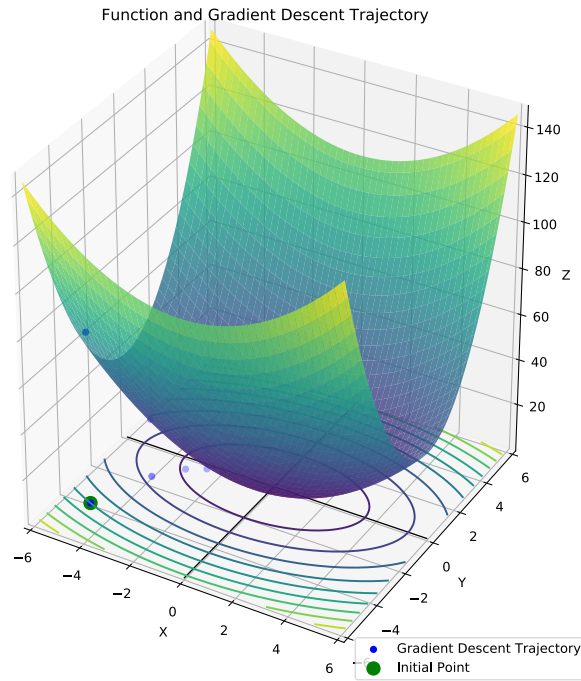
# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



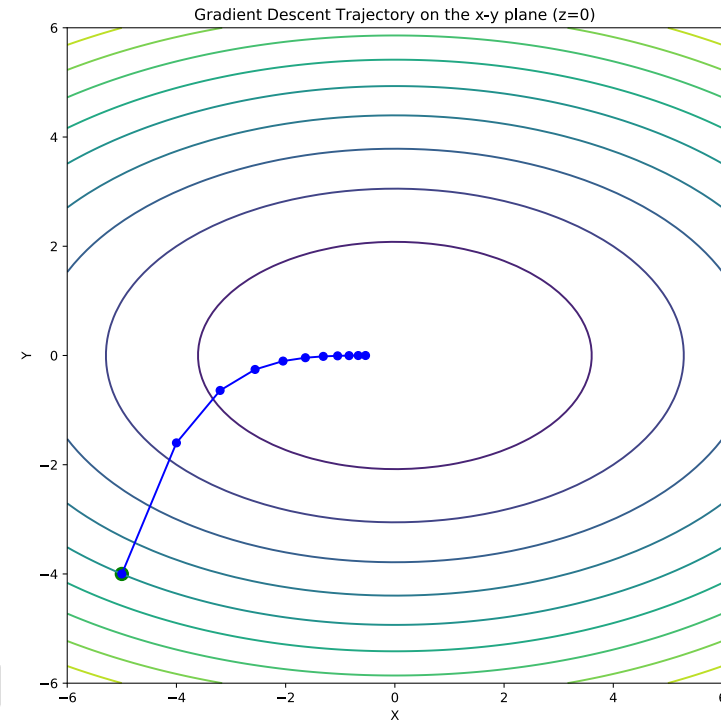
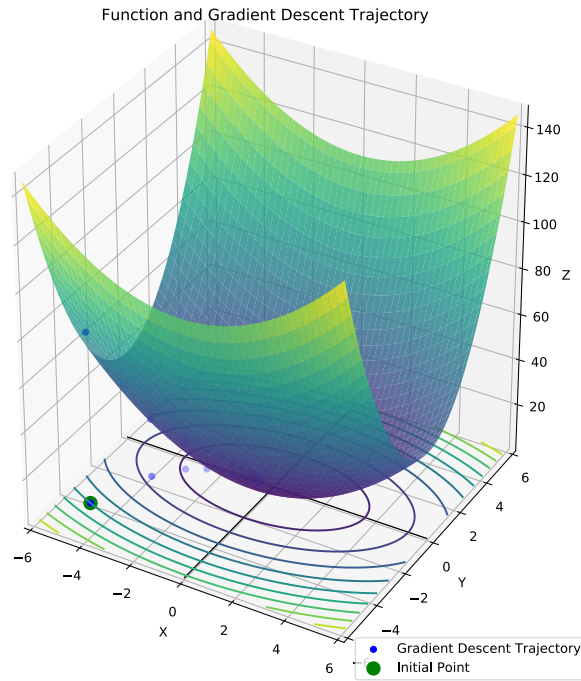
# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



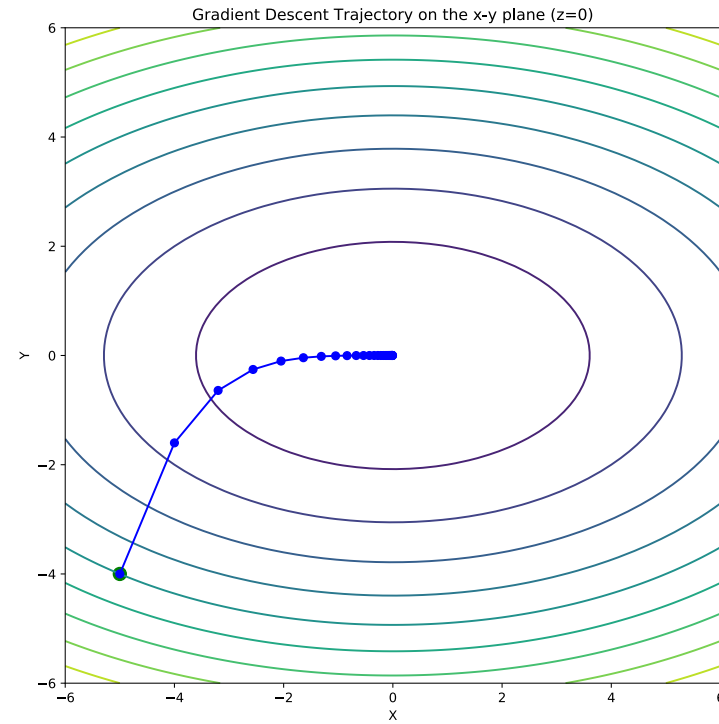
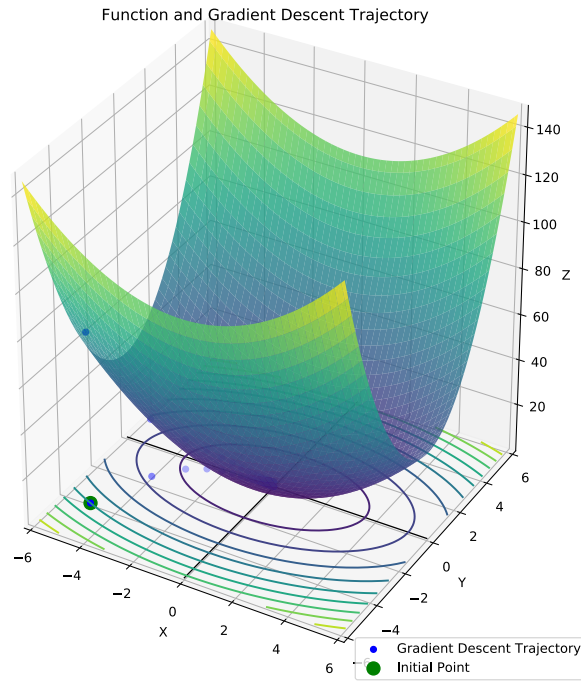
# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



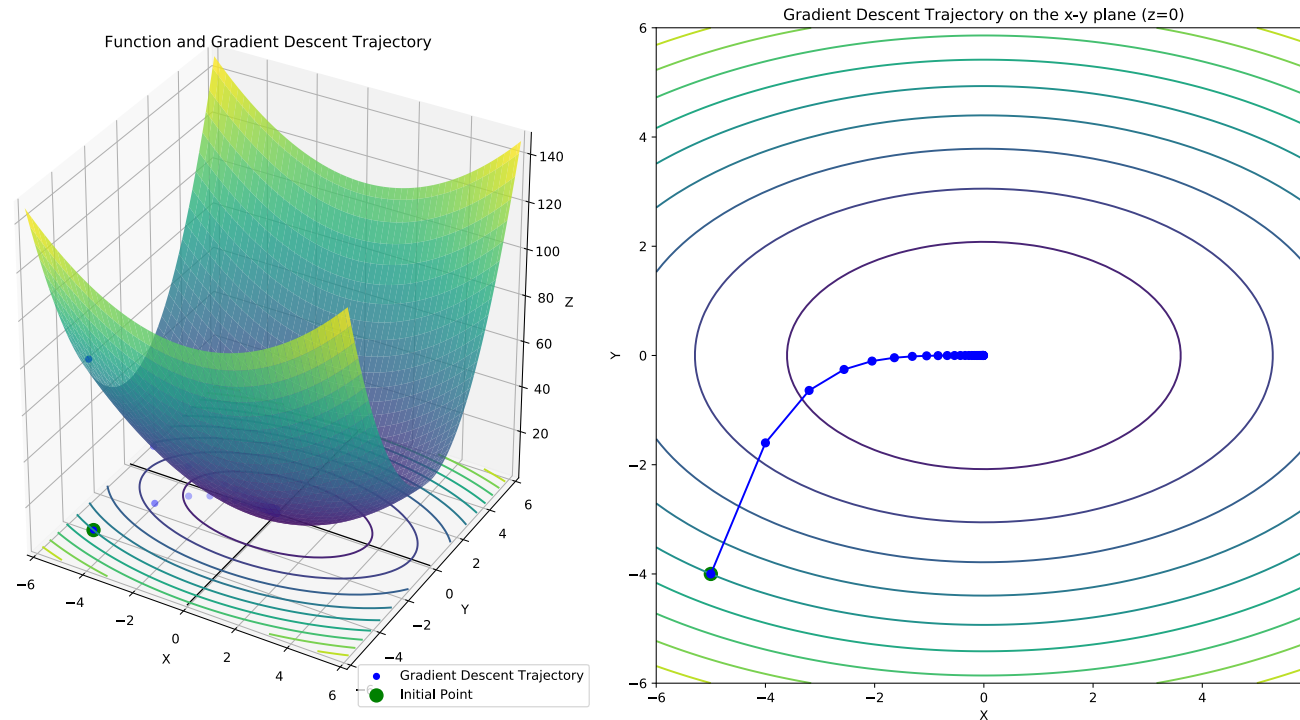
# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



# Gradient Descent: illustration and impact of hyperparameters

## Impact of number of iterations (hyperparameter $T$ )



Conclusion: choose  $T$  as large as possible (depending on the available time).



# Gradient Descent: illustration and impact of hyperparameters

Impact of the step-size/learning-rate (hyperparameter  $\eta$ ):

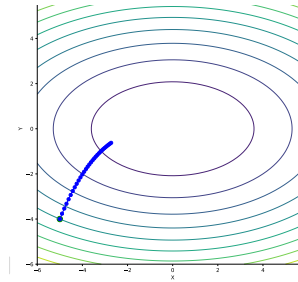
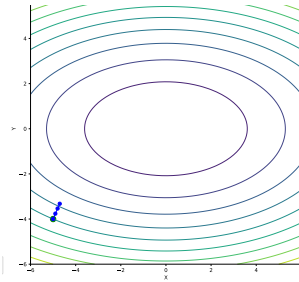
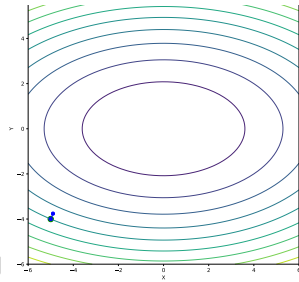
$\eta$

$T = 1$

$T = 3$

$T = 30$

too small

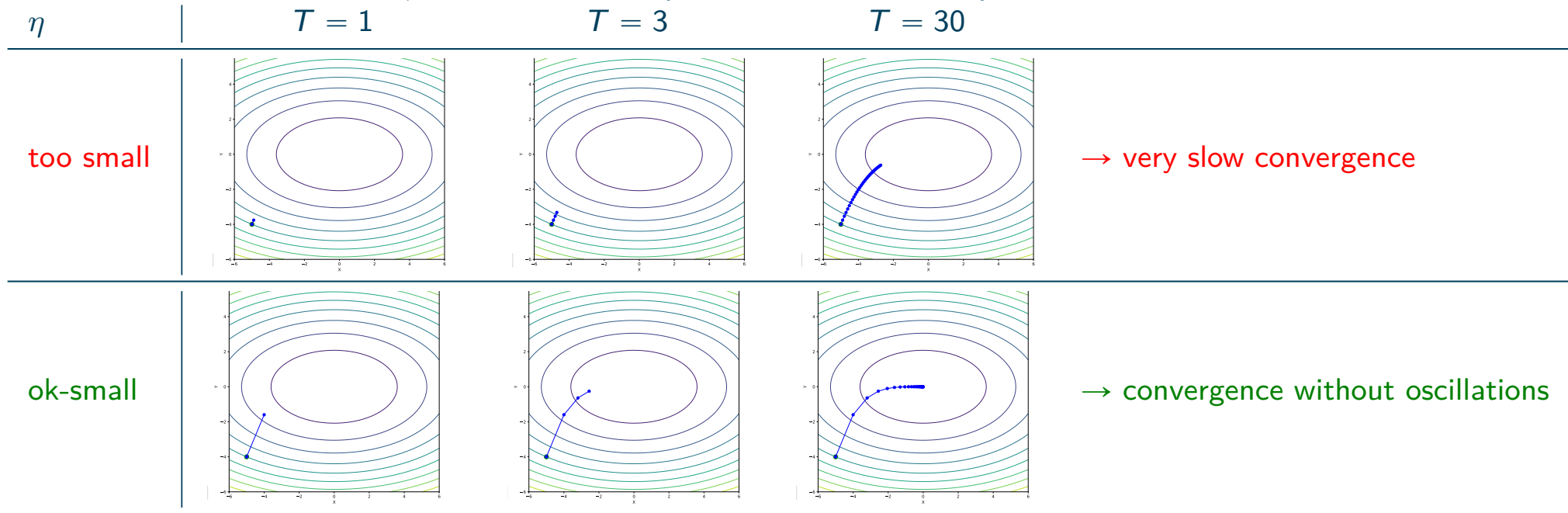


→ very slow convergence

Conclusion: choose  $\eta$  as large as possible while maintaining convergence (avoiding amplification in oscillations)

# Gradient Descent: illustration and impact of hyperparameters

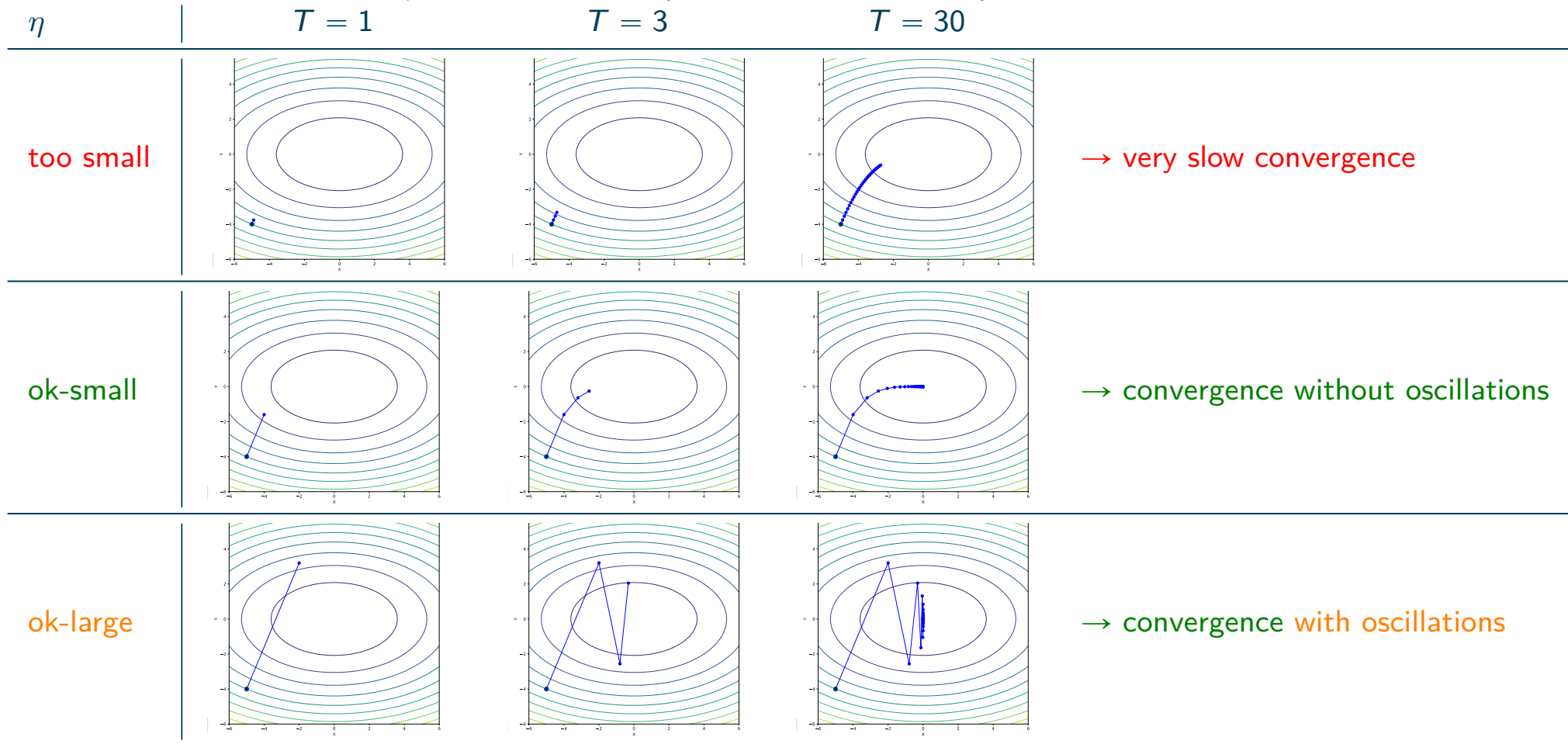
Impact of the step-size/learning-rate (hyperparameter  $\eta$ ):



Conclusion: choose  $\eta$  as large as possible while maintaining convergence (avoiding amplification in oscillations)

# Gradient Descent: illustration and impact of hyperparameters

Impact of the step-size/learning-rate (hyperparameter  $\eta$ ):



Conclusion: choose  $\eta$  as large as possible while maintaining convergence (avoiding amplification in oscillations)

# Gradient Descent: illustration and impact of hyperparameters

Impact of the step-size/learning-rate (hyperparameter  $\eta$ ):

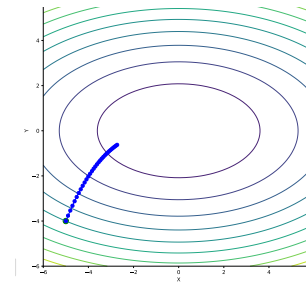
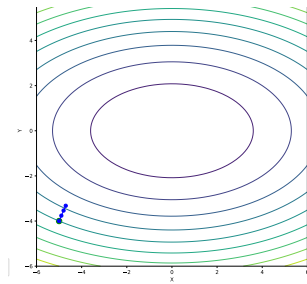
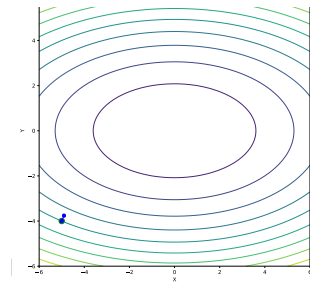
$\eta$

$T = 1$

$T = 3$

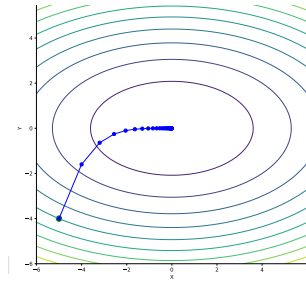
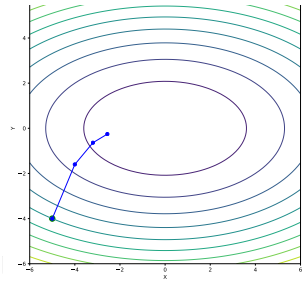
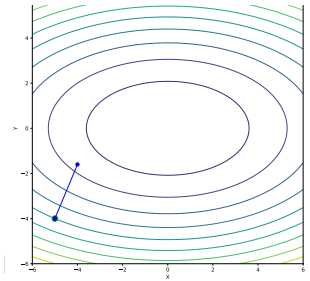
$T = 30$

too small



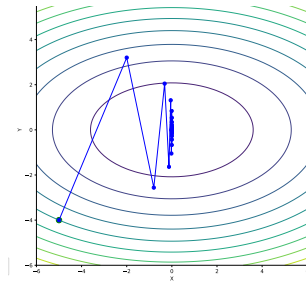
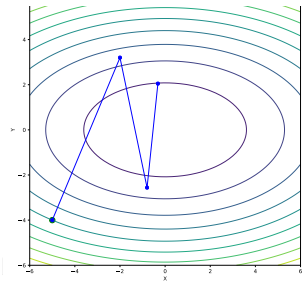
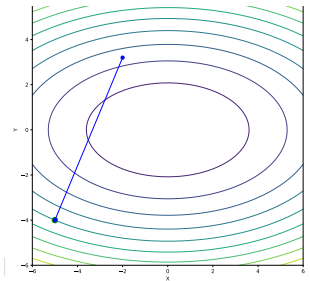
→ very slow convergence

ok-small



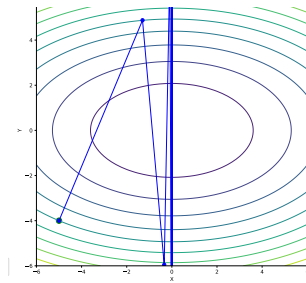
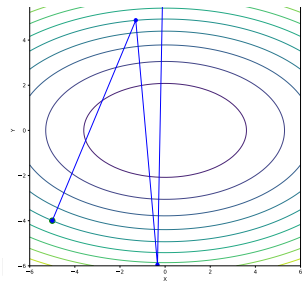
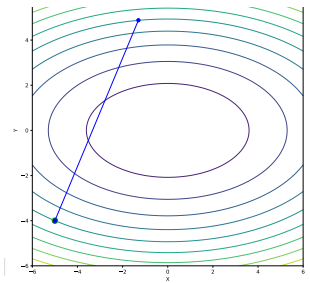
→ convergence without oscillations

ok-large



→ convergence with oscillations

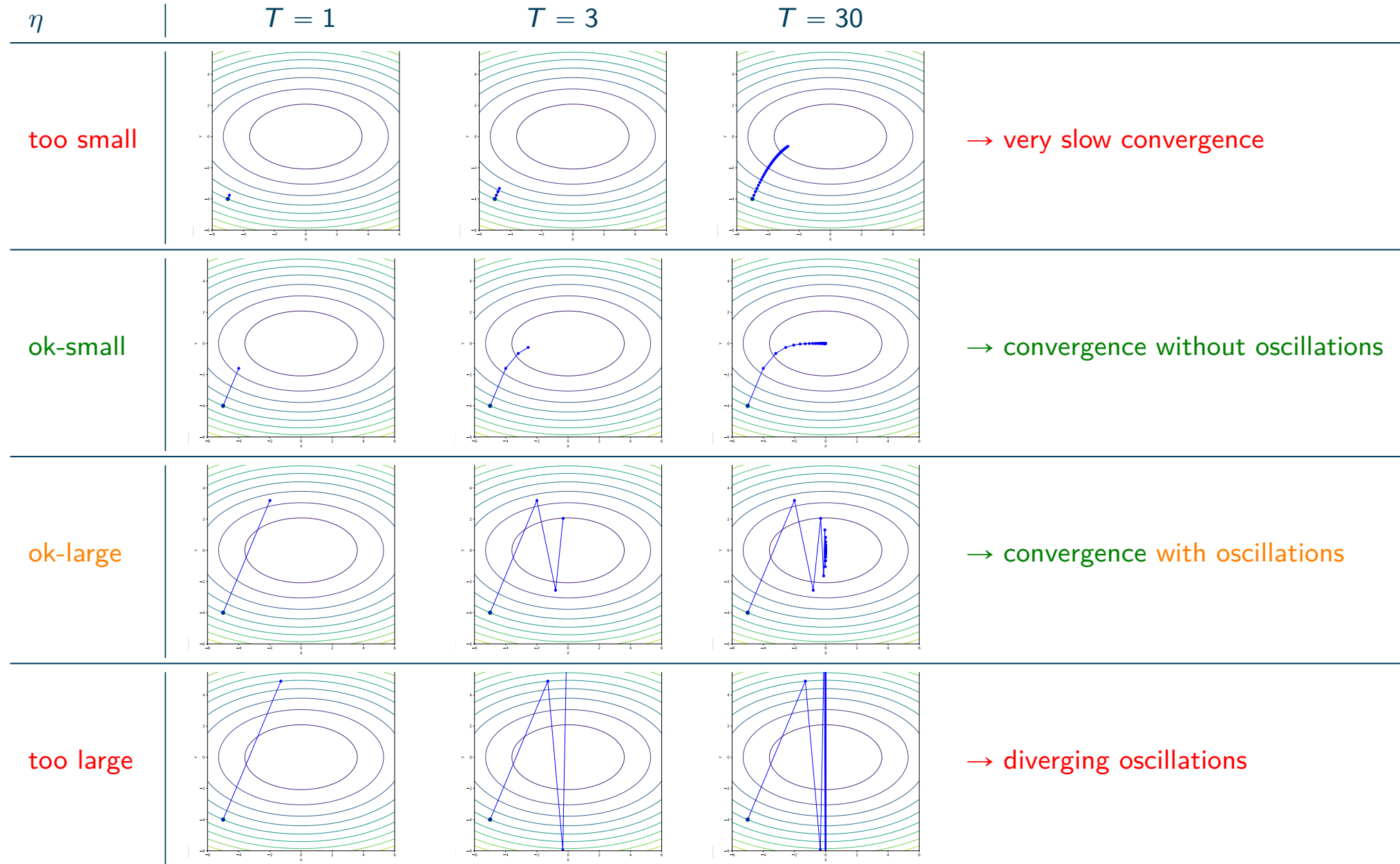
too large



→ diverging oscillations

# Gradient Descent: illustration and impact of hyperparameters

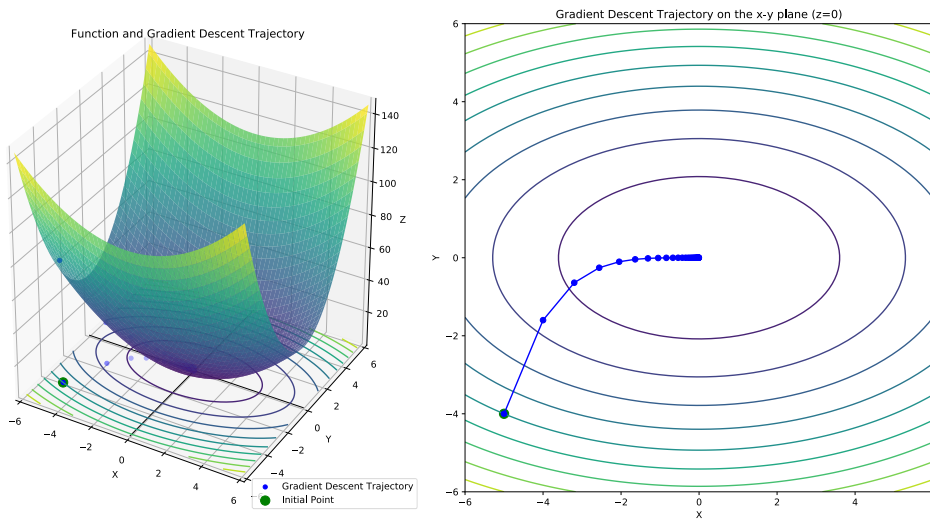
Impact of the step-size/learning-rate (hyperparameter  $\eta$ ):



Conclusion: choose  $\eta$  as large as possible while maintaining convergence (avoiding amplification in oscillations)

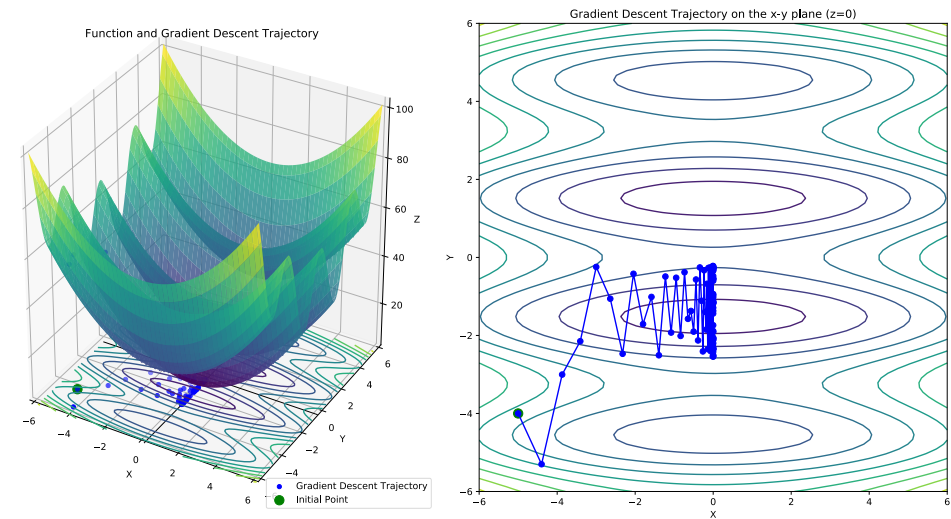
# Gradient Descent: illustration and impact of the shape of the function

## Convex function:



→ typically converges to the **global minimum**.

## Non-convex function:



→ typically converges to a **local minimum**.

# Theoretical grounds for GD method

**Informal statement:** GD converges, for a correct choice of steps, for most convex functions.

→ **Formal results:**

- ① Grounded formulation of gradient descent
  - a. Local minimization of the first order approximation
  - b. Global minimization of the quadratic upper bound for smooth functions
- ② Convergence results and optimization complexity.
  - a. For convex and smooth functions.
  - b. For strongly convex and smooth functions.

skip



*Remark:* Why do we want convergence rates and proofs:

- Rates support the choice of “optimal” hyperparameters.
- Rates allow us to **compare algorithms**.

## Grounded formulation of gradient descent:

### a. local minimization of the first order approximation

! skipped slide

Recall that, around  $\theta_0$ , the first-order Taylor approximation gives

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st-order})$$

## Proposition

Consider a neighborhood  $B(\theta_0, R)$  of  $\theta_0$ , then:

$$\operatorname{argmin}_{\theta \in B(\theta_0, R)} \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} = \underbrace{\theta_0 - \eta \nabla f(\theta_0)}_{\text{GD!}} \quad \text{with } \eta = \frac{R}{\|\nabla f(\theta_0)\|}$$



## Grounded formulation of gradient descent:

### a. local minimization of the first order approximation

Recall that, around  $\theta_0$ , the first-order Taylor approximation gives

$$f(\theta) = \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} + \underbrace{O(\|\theta - \theta_0\|^2)}_{\text{mathematical notation for a negligible residual scaling as } \|\theta - \theta_0\|^2} \quad (\text{Taylor, 1st-order})$$

## Proposition

Consider a neighborhood  $B(\theta_0, R)$  of  $\theta_0$ , then:

$$\operatorname{argmin}_{\theta \in B(\theta_0, R)} \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle}_{\text{First-order approximation}} = \underbrace{\theta_0 - \eta \nabla f(\theta_0)}_{\text{GD!}} \quad \text{with } \eta = \frac{R}{\|\nabla f(\theta_0)\|}$$

Proof:

$$\begin{aligned} \operatorname{argmin}_{\theta \in B(\theta_0, R)} f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle &= \operatorname{argmin}_{\theta \in B(\theta_0, R)} \langle \nabla f(\theta_0), \theta - \theta_0 \rangle \\ &= \operatorname{argmax}_{\theta \in B(\theta_0, R)} \langle -\nabla f(\theta_0), \theta - \theta_0 \rangle \end{aligned}$$

And by Cauchy-Schwarz inequality  $\langle -\nabla f(\theta_0), \theta - \theta_0 \rangle \leq \|-\nabla f(\theta_0)\| \|\theta - \theta_0\|$  with an equality iif  $\theta - \theta_0$  is positively aligned with  $-\nabla f(\theta_0)$ , i.e.  $\theta = \theta_0 - \eta \nabla f(\theta_0)$ .

Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Consider an  $L$ -smooth function  $f$ .

Recall that, at point  $\theta_0$ , uniformly over  $\theta$ , we have:

$$f(\theta) \leq \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2}_{\text{Quadratic upper bound}} \quad (\text{Smoothness})$$

### Proposition

We have that:

$$\operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2}_{\text{Quadratic upper bound}} = \underbrace{\theta_0 - \eta \nabla f(\theta_0)}_{\text{GD!}} \quad \text{with } \eta = \frac{1}{L}$$

Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Consider an  $L$ -smooth function  $f$ .

Recall that, at point  $\theta_0$ , uniformly over  $\theta$ , we have:

$$f(\theta) \leq \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2}_{\text{Quadratic upper bound}} \quad (\text{Smoothness})$$

### Proposition

We have that:

$$\operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2}_{\text{Quadratic upper bound}} = \underbrace{\theta_0 - \eta \nabla f(\theta_0)}_{\text{GD!}} \quad \text{with } \eta = \frac{1}{L}$$

Proof: let

$$g(\theta) := f(\theta_0) + \langle \nabla f(\theta_0), \theta - \theta_0 \rangle + \frac{L}{2} \|\theta - \theta_0\|^2.$$

Then  $g$  is convex and

$$\nabla g(\theta) = \nabla f(\theta_0) + L(\theta - \theta_0)$$

Thus  $g$  is minimized iif.  $\nabla g(\theta) = 0$ , i.e., iif.  $\theta = \theta_0 - \frac{1}{L} \nabla f(\theta_0)$ .

Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$

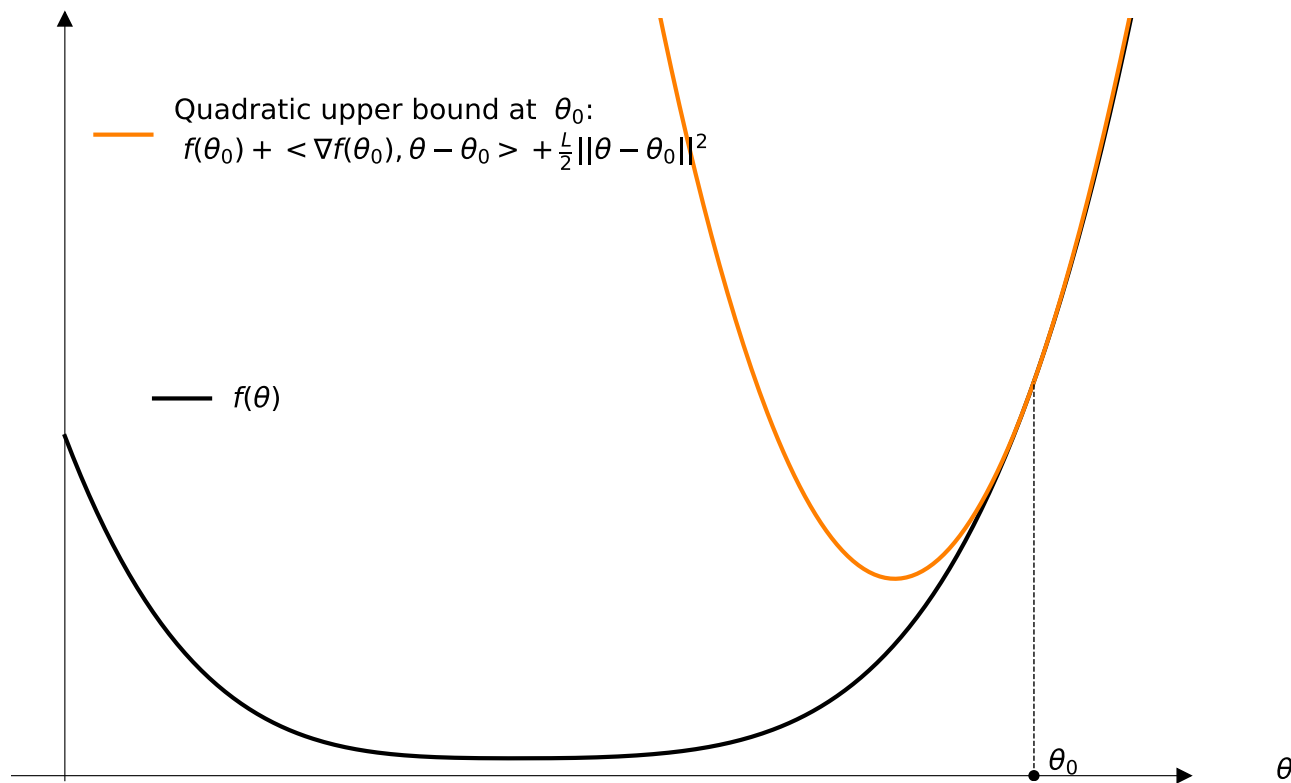
$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$

Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$

$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$

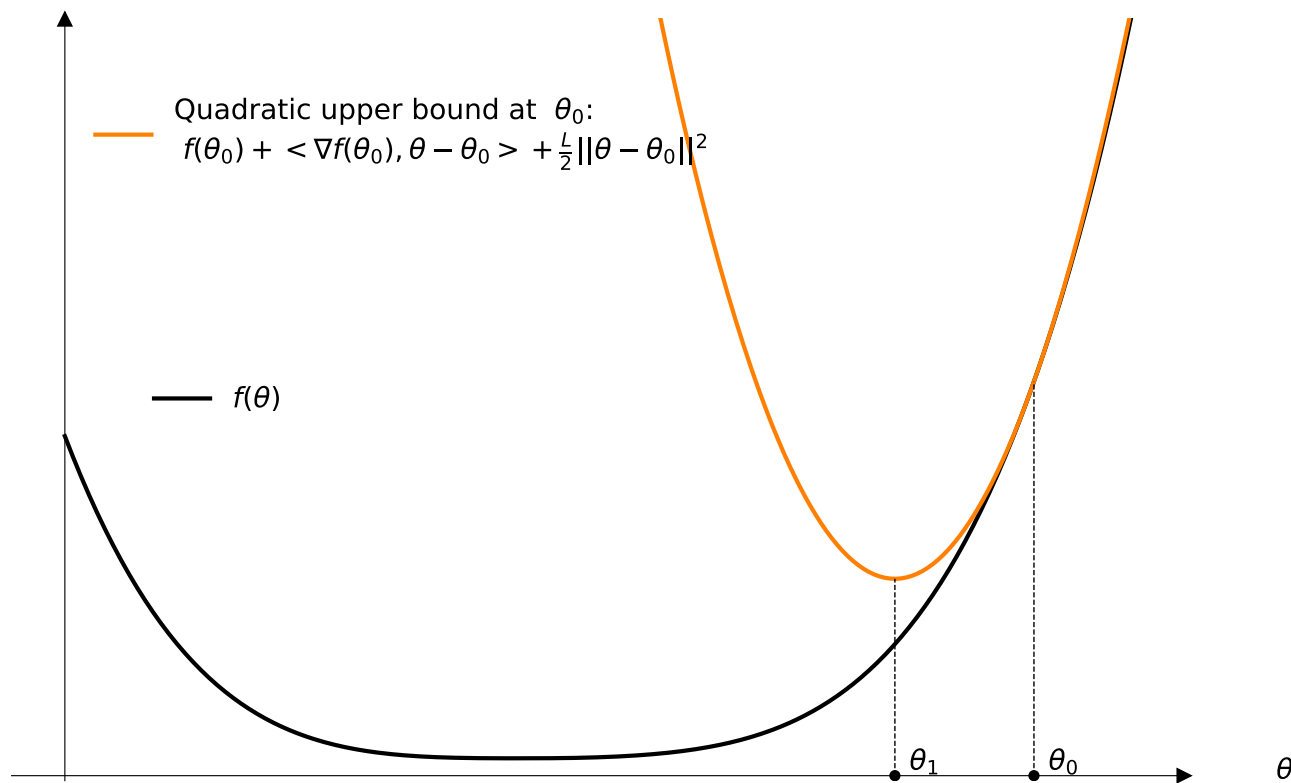


## Grounded formulation of gradient descent:

### b. Global minimization of the quadratic upper bound for smooth functions

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$

$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$



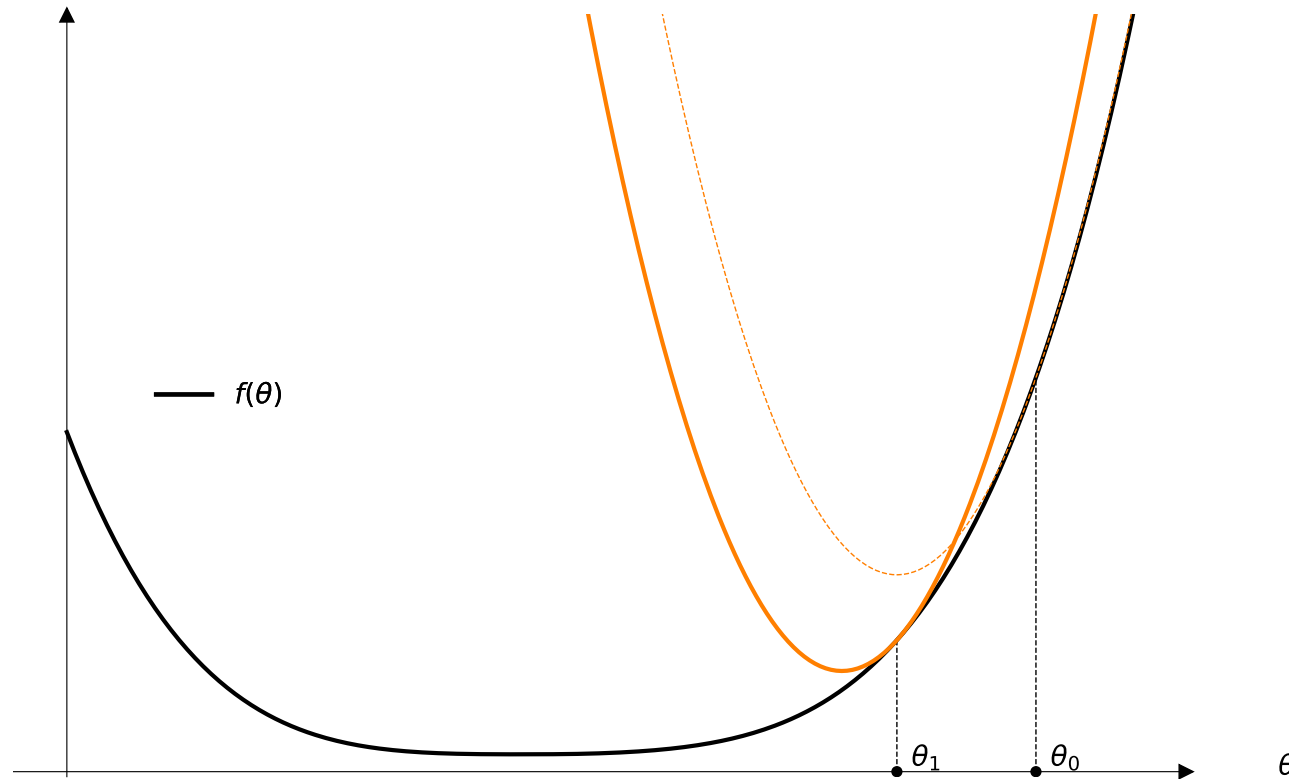
Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

See n!

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$

$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$

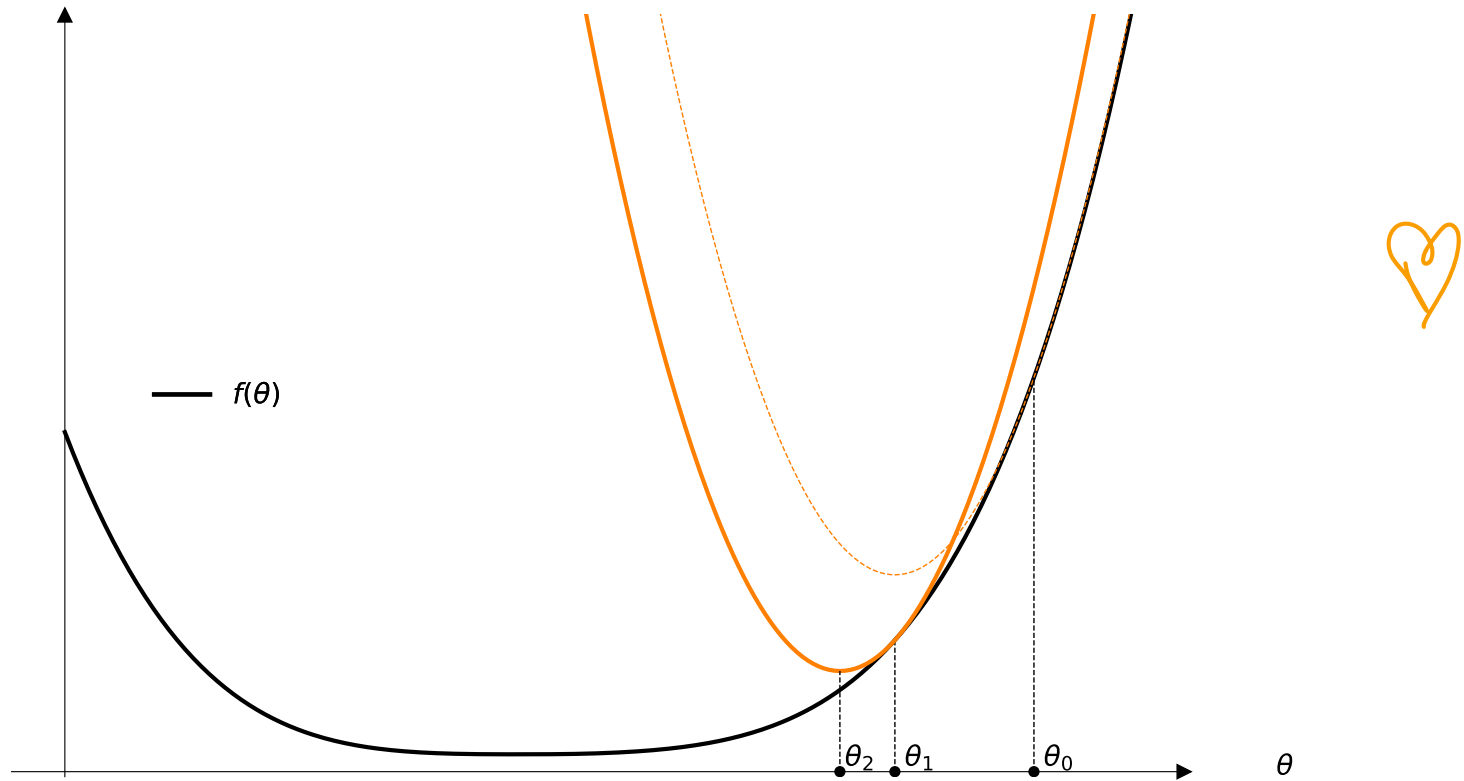


Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$

$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$



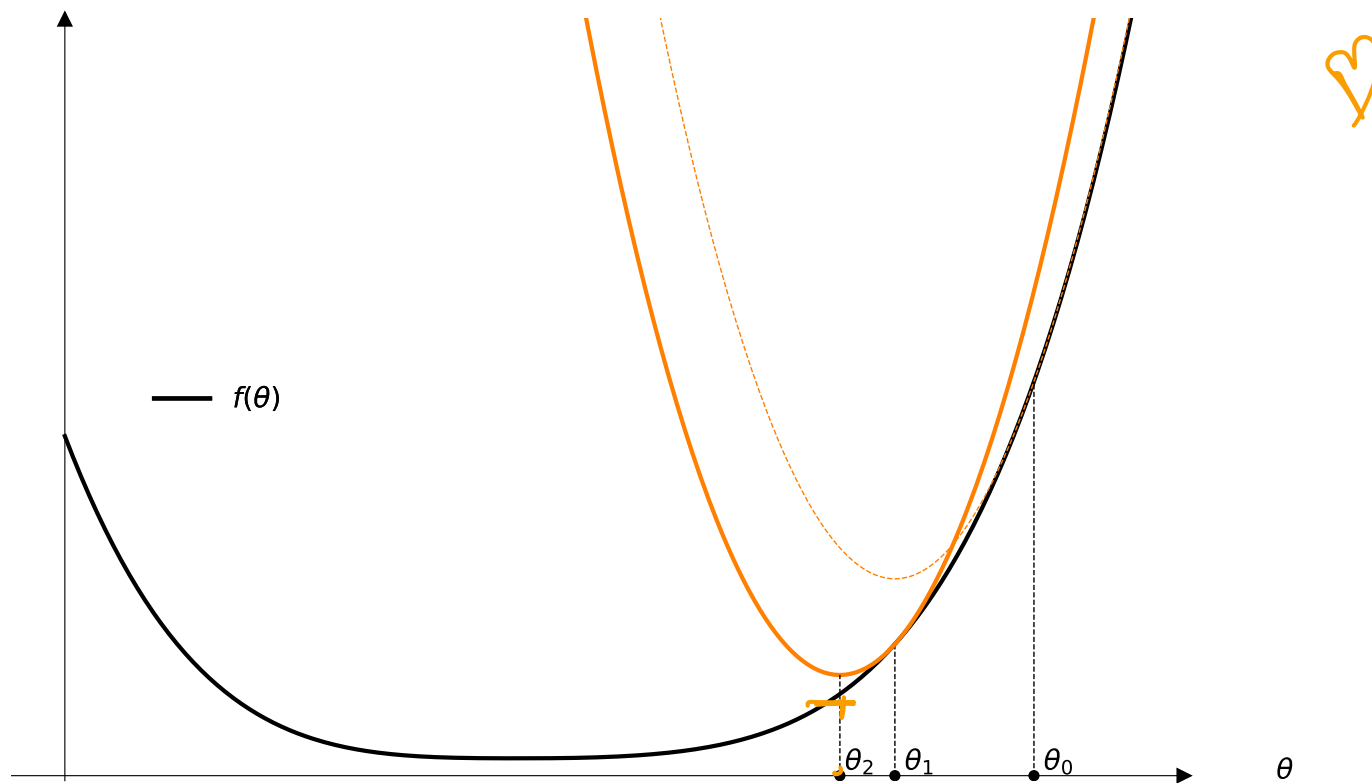


Grounded formulation of gradient descent:

b. Global minimization of the quadratic upper bound for smooth functions

Geometric interpretation. GD with step-size  $\frac{1}{L}$  corresponds to iteratively minimizing the quadratic upper bound around  $\theta_{t-1}$ : for all  $t$  ↪ quadratic descent

$$\underbrace{\theta_{t+1} = \theta_t - \frac{1}{L} \nabla f(\theta_t)}_{\text{GD from } \theta_t} = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \underbrace{f(\theta_t) + \langle \nabla f(\theta_t), \theta - \theta_{t-1} \rangle + \frac{L}{2} \|\theta - \theta_t\|^2}_{\text{QUB}_{\theta_t} \text{ around } \theta_t}$$



Consequence: *descent lemma*  $f(\theta_{t+1}) \leq \text{QUB}_{\theta_t}(\theta_{t+1}) \leq \text{QUB}_{\theta_t}(\theta_t) = f(\theta_t)$ .

## Convergence & optimization complexity: a. smooth convex $f$ .

Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Convergence & optimization complexity: a. smooth convex  $f$ .

skipped

Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Interpretation:

- 1 Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .

## Convergence & optimization complexity: a. smooth convex $f$ .

### Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
- ② Impact of  $\eta$  (confirms illustration/intuition):
  - $\eta > 2/L$  no guarantee (possible divergence)
  - best bound  $\rightarrow$  choose  $\eta$  to maximize  $\eta(1 - \eta L/2)$   
 $\rightarrow$  choose  $\eta$  as large as possible while avoiding oscillations  $\rightarrow \eta = \frac{1}{L}$ , giving:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L\|\theta_0 - \theta_\star\|_2^2}{t}.$$

## Convergence & optimization complexity: a. smooth convex $f$ .

### Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
- ② Impact of  $\eta$  (confirms illustration/intuition):
  - $\eta > 2/L$  no guarantee (possible divergence)
  - best bound  $\rightarrow$  choose  $\eta$  to maximize  $\eta(1 - \eta L/2)$   
 $\rightarrow$  choose  $\eta$  as large as possible while avoiding oscillations  $\rightarrow \eta = \frac{1}{L}$ , giving:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L\|\theta_0 - \theta_\star\|_2^2}{t}.$$

- ③ Initial condition  $\|\theta_0 - \theta_\star\|_2^2$ : starting further  $\rightarrow$  more iterations needed.

## Convergence & optimization complexity: a. smooth convex $f$ .

### Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
- ② Impact of  $\eta$  (confirms illustration/intuition):
  - $\eta > 2/L$  no guarantee (possible divergence)
  - best bound  $\rightarrow$  choose  $\eta$  to maximize  $\eta(1 - \eta L/2)$   
 $\rightarrow$  choose  $\eta$  as large as possible while avoiding oscillations  $\rightarrow \eta = \frac{1}{L}$ , giving:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L\|\theta_0 - \theta_\star\|_2^2}{t}.$$

- ③ Initial condition  $\|\theta_0 - \theta_\star\|_2^2$ : starting further  $\rightarrow$  more iterations needed.
- ④ Rate  $1/t$ : to achieve an error  $10\times$  smaller, need to perform  $10\times$  more iteration.  
Precisely, to ensure  $f(\theta_t) - f(\theta_\star) \leq \epsilon$ , we need at least  $T_\epsilon$  iterations with

$$\underbrace{T_\epsilon}_{\text{iter. complexity}} \geq \frac{L\|\theta_0 - \theta_\star\|_2^2}{\epsilon}.$$

Thus  $T_{\frac{\epsilon}{10}} = 10 \times T_\epsilon$ .

## Convergence & optimization complexity: a. smooth convex $f$ .

### Theorem: convergence of GD on smooth **and convex** functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 2/L$  satisfies, after  $t$  steps:

$$f(\theta_t) - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|_2^2}{2\eta(1 - \eta L/2)t}.$$

Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
- ② Impact of  $\eta$  (**confirms illustration/intuition**):
  - $\eta > 2/L$  no guarantee (possible divergence)
  - best bound  $\rightarrow$  choose  $\eta$  to maximize  $\eta(1 - \eta L/2)$   
 $\rightarrow$  choose  $\eta$  as large as possible while avoiding oscillations  $\rightarrow \eta = \frac{1}{L}$ , giving:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L\|\theta_0 - \theta_\star\|_2^2}{t}.$$

- ③ Initial condition  $\|\theta_0 - \theta_\star\|_2^2$ : starting further  $\rightarrow$  more iterations needed.
- ④ Rate  $1/t$ : to achieve an error  $10\times$  smaller, need to perform  $10\times$  more iterations.  
Precisely, to ensure  $f(\theta_t) - f(\theta_\star) \leq \epsilon$ , we need at least  $T_\epsilon$  iterations with

$$\underbrace{T_\epsilon}_{\text{iter. complexity}} \geq \frac{L\|\theta_0 - \theta_\star\|_2^2}{\epsilon}.$$

Thus  $T_{\frac{\epsilon}{10}} = 10 \times T_\epsilon$ .

Proof: see exercise sheet 1.

## Convergence & optimization complexity: b. smooth and strongly-convex $f$ .

### Theorem: convergence of GD on smooth & strongly convex functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth,  $\mu$ -strongly-convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 1/L$  satisfies, after  $t$  iterations:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L}{2} \left(1 - \eta\mu\right)^t \|\theta_0 - \theta_\star\|_2^2.$$



## Convergence & optimization complexity: b. smooth and strongly-convex $f$ .

### Theorem: convergence of GD on smooth & strongly convex functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth,  $\mu$ -strongly-convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 1/L$  satisfies, after  $t$  iterations:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L}{2} \left(1 - \eta\mu\right)^t \|\theta_0 - \theta_\star\|_2^2.$$

### Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
- ② Impact of  $\eta \rightarrow$  choose “maximal”  $\eta = 1/L$ .
- ③ Initial condition  $\|\theta_0 - \theta_\star\|_2^2$ : starting further  $\rightarrow$  more iterations needed.

}  $\rightsquigarrow$  Same as before!

shipped  
Convergence & optimization complexity: b. smooth and strongly-convex  $f$ .

Theorem: convergence of GD on smooth & strongly convex functions

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a  $L$ -smooth,  $\mu$ -strongly-convex function. Let  $\theta_\star$  be the minimum of  $f$  on  $\mathbb{R}^d$ . Then, GD with step size  $\eta \leq 1/L$  satisfies, after  $t$  iterations:

$$f(\theta_t) - f(\theta_\star) \leq \frac{L}{2} \left(1 - \eta\mu\right)^t \|\theta_0 - \theta_\star\|_2^2.$$

Interpretation:

- ① Convergence:  $f(\theta_t) \rightarrow f(\theta_\star)$  as  $t \rightarrow \infty$ .
  - ② Impact of  $\eta \rightarrow$  choose “maximal”  $\eta = 1/L$ .
  - ③ Initial condition  $\|\theta_0 - \theta_\star\|_2^2$ : starting further  $\rightarrow$  more iterations needed.
- Rate  $\left(1 - \frac{\mu}{L}\right)^t = \left(1 - \frac{1}{\kappa}\right)^t$ : to achieve an error 10× smaller, need to perform a *constant number of extra iterations*. Precisely, to ensure  $f(\theta_t) - f(\theta_\star) \leq \epsilon$ , we need at least  $T_\epsilon$  iterations with

}  $\rightsquigarrow$  Same as before!

$$\frac{L}{2} \left(1 - \frac{1}{\kappa}\right)^{T_\epsilon} \|\theta_0 - \theta_\star\|_2^2 \leq \epsilon \quad \Leftrightarrow \quad T_\epsilon \geq \frac{\ln\left(\frac{L\|\theta_0 - \theta_\star\|_2^2}{2\epsilon}\right)}{\ln\left(1 + \frac{1}{\kappa-1}\right)} \simeq \kappa \ln\left(\frac{L\|\theta_0 - \theta_\star\|_2^2}{2\epsilon}\right)$$

Thus  $T_{\frac{\epsilon}{10}} = T_\epsilon + \kappa \ln(10)$ .

## Impact of the condition number $\kappa$

For smooth and strongly convex functions, iteration complexity thus scales as

$$T_\epsilon \simeq \kappa \ln \left( \frac{L \|\theta_0 - \theta_\star\|_2^2}{2\epsilon} \right)$$

Bad conditioning ( $\kappa$  large)  $\rightarrow$  more iterations needed, proportionally to  $\kappa$ .

# Impact of the condition number $\kappa$

skipped

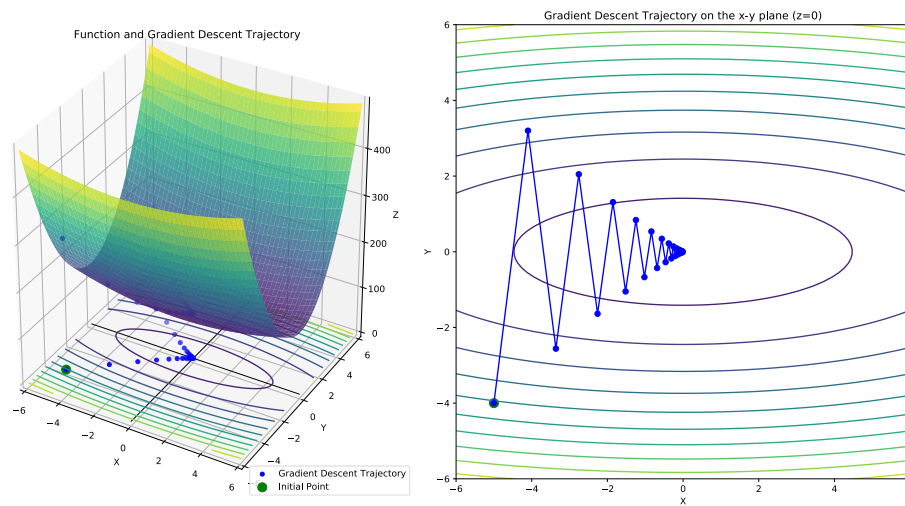
For smooth and strongly convex functions, iteration complexity thus scales as

$$T_\epsilon \simeq \kappa \ln \left( \frac{L \|\theta_0 - \theta_\star\|_2^2}{2\epsilon} \right)$$

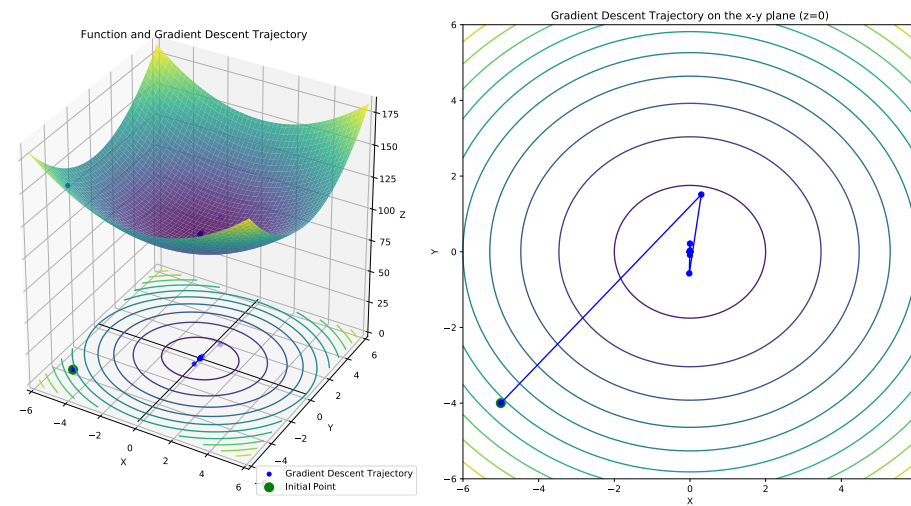
Bad conditioning ( $\kappa$  large)  $\rightarrow$  more iterations needed, proportionally to  $\kappa$ .

Why is there such a dependency on  $\kappa$ ?  $\rightsquigarrow$  see discussion on conditioning before.

Moreover, for a poorly conditioned function, the gradient does not point towards the global optimum.



Large  $\kappa$  (10 here)  
harder to optimize



Small  $\kappa$  (1.25 here)  
easier to optimize

Remark: in machine learning,  $\kappa$  can be extremely large (typically  $\geq 10^6$ )

## Cost per iteration and total cost for complexity $\epsilon$

skipped

### Cost per iteration $C$ :

- computing  $\nabla f(\theta_t) \in \mathbb{R}^d \rightsquigarrow$  cost  $O(d)$ .
- when  $f$  is the empirical risk, i.e.,  $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta) \rightsquigarrow$  cost  $O(n \times d)$ .

**Summary.** To optimize with precision  $\epsilon$  the empirical risk  $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta)$ , the computational cost is

| Class                                                 | $L$ -smooth convex functions        | $L$ -smooth & $\mu$ -strongly-convex functions              |
|-------------------------------------------------------|-------------------------------------|-------------------------------------------------------------|
| Number of iter. $T_\epsilon$ for $\epsilon$ precision | $O\left(\frac{L}{\epsilon}\right)$  | $O\left(\kappa \ln\left(\frac{1}{\epsilon}\right)\right)$   |
| Cost per iteration $C$                                | $O(nd)$                             | $O(nd)$                                                     |
| Total cost = $T_\epsilon \times C$                    | $O\left(\frac{nd}{\epsilon}\right)$ | $O\left(nd\kappa \ln\left(\frac{1}{\epsilon}\right)\right)$ |

# GD: Summary

## First-order method     Deterministic

### Update

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla f(\theta_t)$$

**Hyperparameters:**  $\eta, T$ .

$\rightsquigarrow$  choose  $\eta$  as large as possible without oscillations.

**Complexity/iter in ML:**

$$O(nd)$$

### Intuition:

- steepest descent direction
- local minimization of first-order approx
- global minimization of QUB for  $f$  smooth.

### Theory:

- Rate for smooth and (strongly) convex functions



Impact of  $\kappa$ .



Impact of  $T$ .



Impact of  $\eta$ .

Let's recap! Wooclap quizz!









# Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
    - Motivation and algorithm
    - Theoretical convergence results for SGD
  - End of lecture 1 - at your own pace quizz!

# Motivation for Stochastic Gradient Descent.

We consider the ERM problem:

$$f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta).$$

GD uses a *full gradients*, i.e. computes at each iteration:

$$\nabla f(\theta) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\theta),$$

which depends on the whole dataset (all observations are used).

For GD, the complexity per iteration on ML examples,  $O(d \times n)$ , is prohibitive. Think of  $d \geq 10^6$ ,  $n \geq 10^8$ ...

# Motivation for Stochastic Gradient Descent.

We consider the ERM problem:

$$f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta).$$

GD uses a *full gradients*, i.e. computes at each iteration:

$$\nabla f(\theta) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\theta),$$

which depends on the whole dataset (all observations are used).

For GD, the complexity per iteration on ML examples,  $O(d \times n)$ , is prohibitive. Think of  $d \geq 10^6$ ,  $n \geq 10^8$ ...

**Idea:** stochastic gradients: use only some (random) elements of the sum!

- 1 At iteration  $t$ , sample of batch  $\mathcal{I}_t$  of  $b$  observations out of the  $n$  available.

Define

$$g_t(\theta) = \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta).$$

- 2  $g_t$  will be used instead of  $\nabla f$ .
- 3  $g_t$  is a random quantity.
- 4 Complexity  $O(d \times b)$  at each iteration.

Stochastic gradient  $g_t$   
average of the gradients  
of the loss on only  $b$   
points

Remark: Goes back to “old” algorithms!

# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

*replaces*

**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

$$\begin{array}{ccc} \nabla f(\theta_{t-1}) & \text{by} & g_t(\theta_{t-1}) \\ \uparrow & & \\ \frac{1}{b} \sum_{i=1}^b \nabla f_i(\theta_{t-1}) & & \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1}) \end{array}$$

# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

---

Choice of  $b$  ?

# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

| Choice of $b$ ? | Name     | Gradient oracle $g_t$                                      | Complexity |
|-----------------|----------|------------------------------------------------------------|------------|
| $b = 1$         | Pure SGD | $g_t = \nabla f_{i_t}$ with $i_t \sim \mathcal{U}[[1; n]]$ | $O(d)$     |

# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

| Choice of $b$ ? | Name               | Gradient oracle $g_t$                                                                | Complexity      |
|-----------------|--------------------|--------------------------------------------------------------------------------------|-----------------|
| $b = 1$         | Pure SGD           | $g_t = \nabla f_{i_t}$ with $i_t \sim \mathcal{U}[[1; n]]$                           | $O(d)$          |
| $1 < b < n$     | $b$ -minibatch SGD | $g_t = \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i$ with $\#\mathcal{I}_t = b$ | $O(d \times b)$ |



# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

| Choice of $b$ ? | Name                        | Gradient oracle $g_t$                                                                | Complexity      |
|-----------------|-----------------------------|--------------------------------------------------------------------------------------|-----------------|
| $b = 1$         | Pure SGD                    | $g_t = \nabla f_{i_t}$ with $i_t \sim \mathcal{U}[[1; n]]$                           | $O(d)$          |
| $1 < b < n$     | $b$ -minibatch SGD          | $g_t = \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i$ with $\#\mathcal{I}_t = b$ | $O(d \times b)$ |
| $b = n$         | deterministic/full batch GD | $g_t = \frac{1}{n} \sum_{i=1}^n \nabla f_i$                                          | $O(d \times n)$ |

# Stochastic Gradient Descent (SGD)

## Algorithm: Stochastic gradient descent method (SGD)

**Initialization:** initial weight vector  $\theta_0$ ,

**Hyperparameters:** learning rate  $(\eta_t)_{t \geq 0}$ , number of steps  $T$ , batch-size  $b$ .

For  $t \in \{1, \dots, T\}$  do

- Pick at random a subset  $\mathcal{I}_t$  of  $\{1, \dots, n\}$  with cardinal  $b$ .
- Compute

$$\theta_t = \theta_{t-1} - \eta_t \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})$$

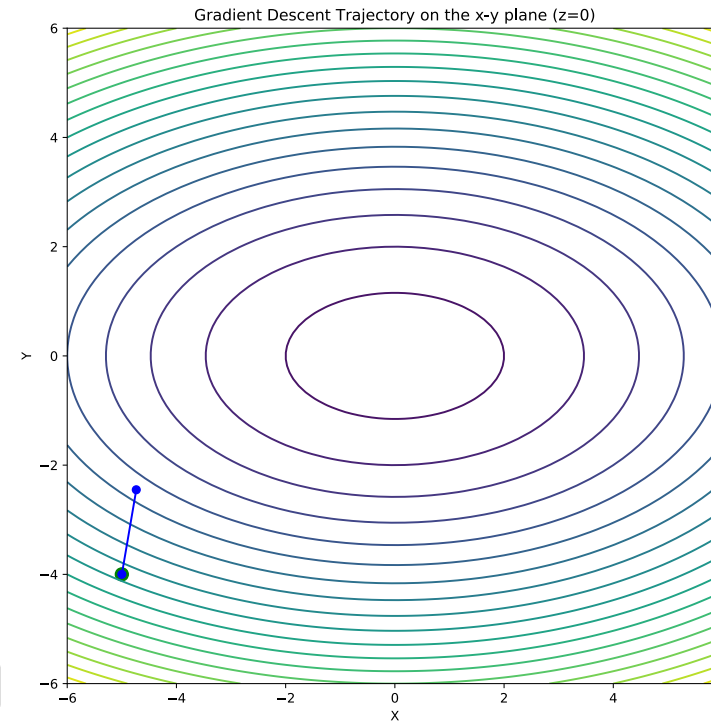
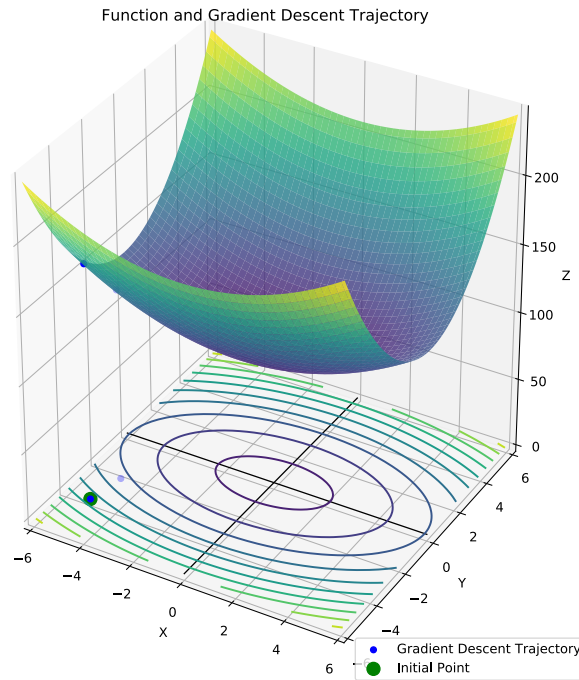
**Output:** Return last  $\theta_T$ , or the average  $\bar{\theta}_T = \frac{1}{T} \sum_{t=1}^T \theta_t$ .

| Choice of $b$ ? | Name                        | Gradient oracle $g_t$                                                                | Complexity      |
|-----------------|-----------------------------|--------------------------------------------------------------------------------------|-----------------|
| $b = 1$         | Pure SGD                    | $g_t = \nabla f_{i_t}$ with $i_t \sim \mathcal{U}[[1; n]]$                           | $O(d)$          |
| $1 < b < n$     | $b$ -minibatch SGD          | $g_t = \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i$ with $\#\mathcal{I}_t = b$ | $O(d \times b)$ |
| $b = n$         | deterministic/full batch GD | $g_t = \frac{1}{n} \sum_{i=1}^n \nabla f_i$                                          | $O(d \times n)$ |

- In practice, good choice of  $b$  is typically 32, 64, 128, 256   
~~~~~ benefits from parallelization of computation on GPU/CPU.
- For theoretical purposes, focus on $b = 1$, which is the simplest case and sufficient.

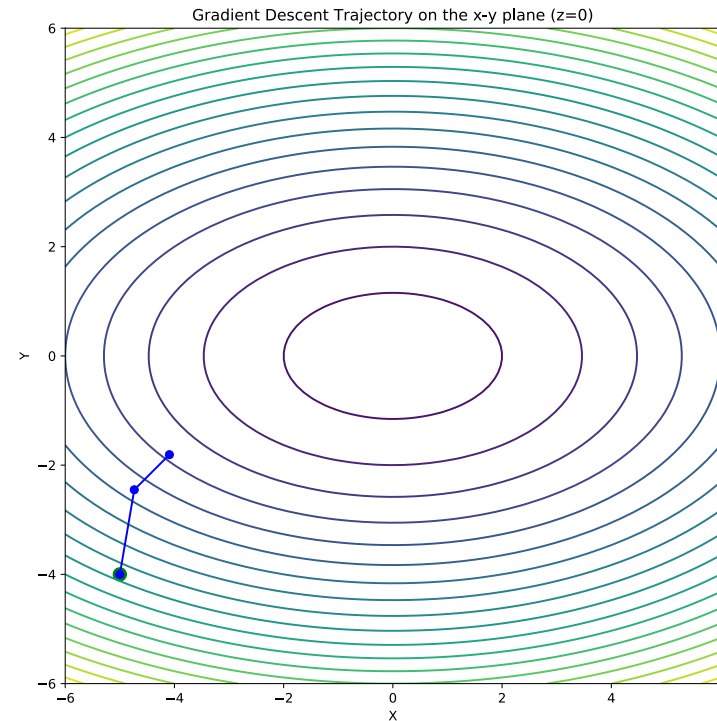
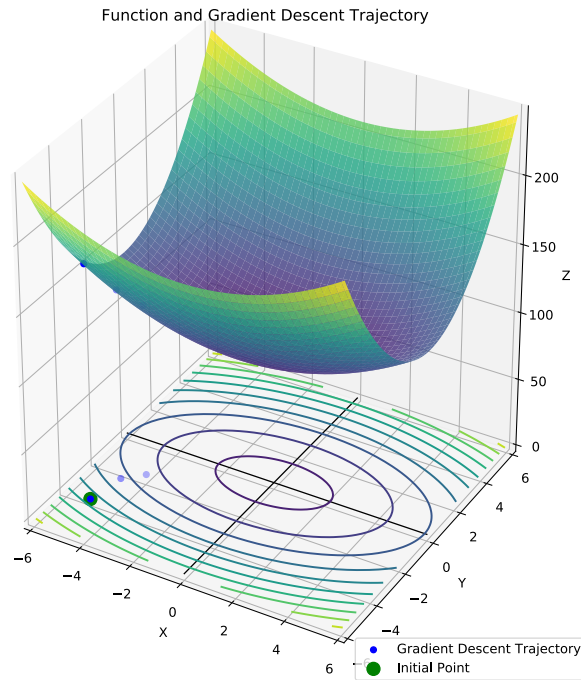
SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



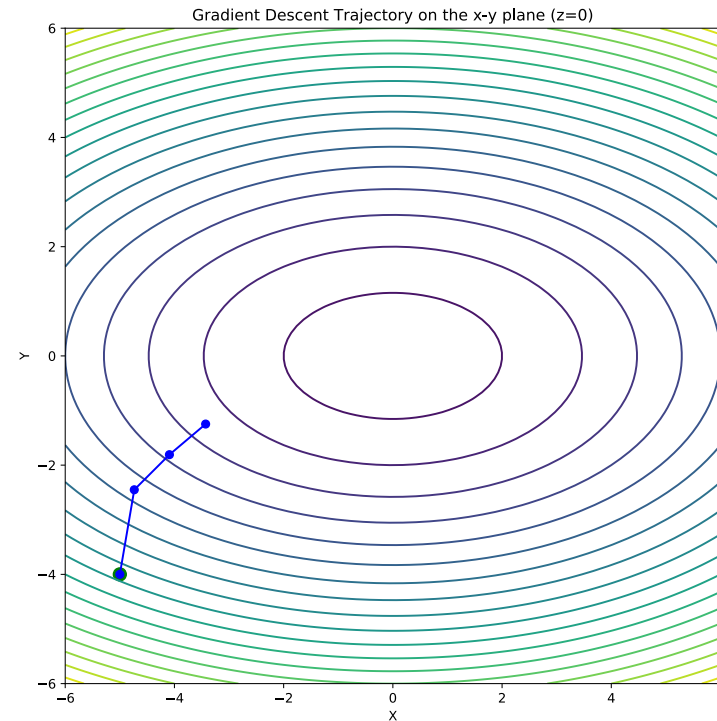
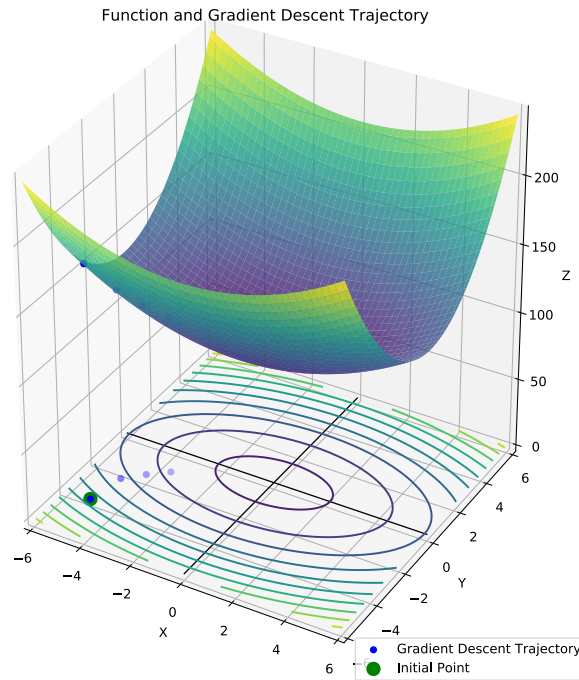
SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



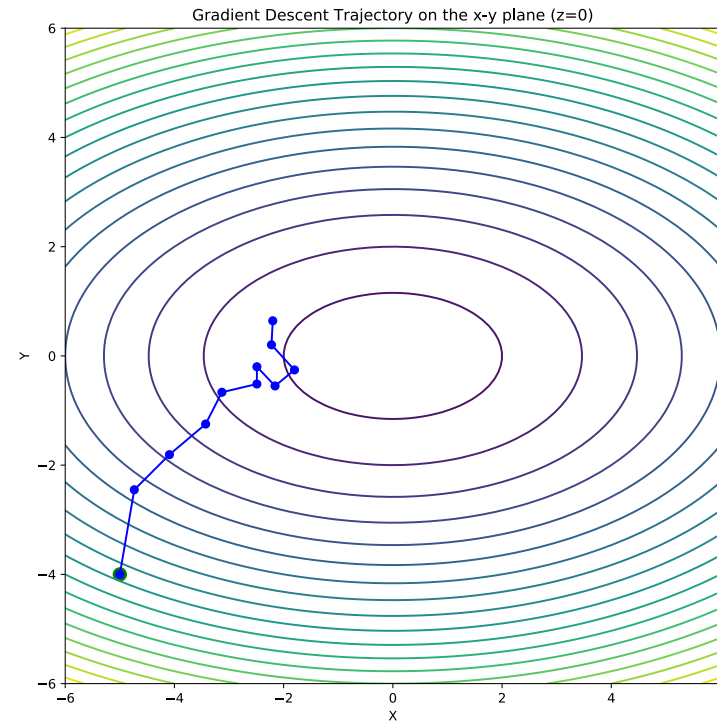
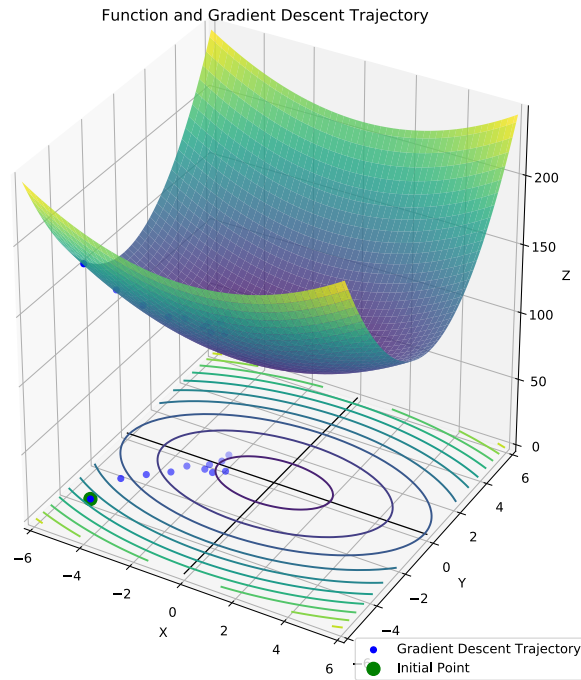
SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



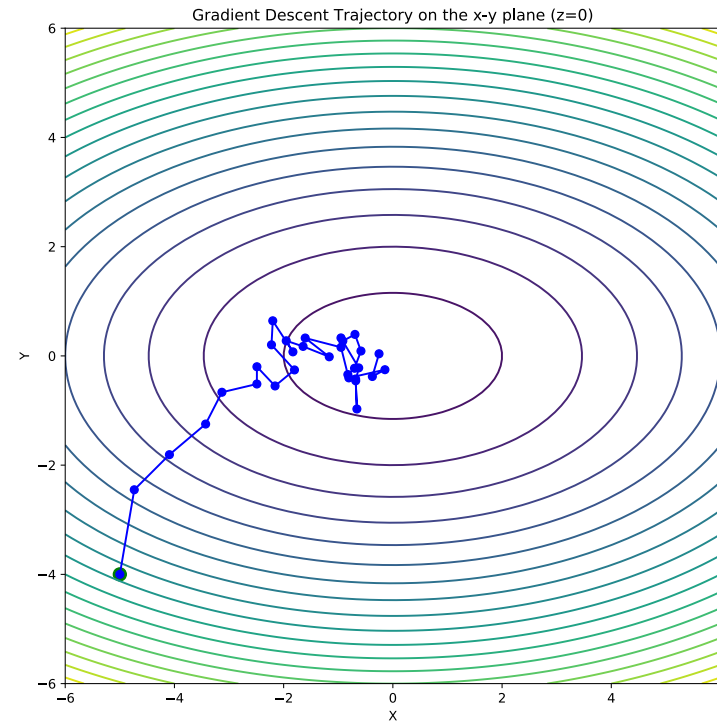
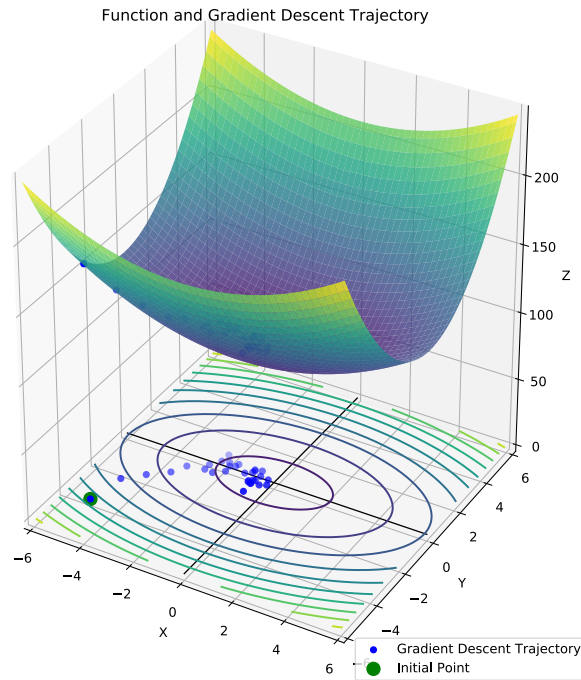
SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



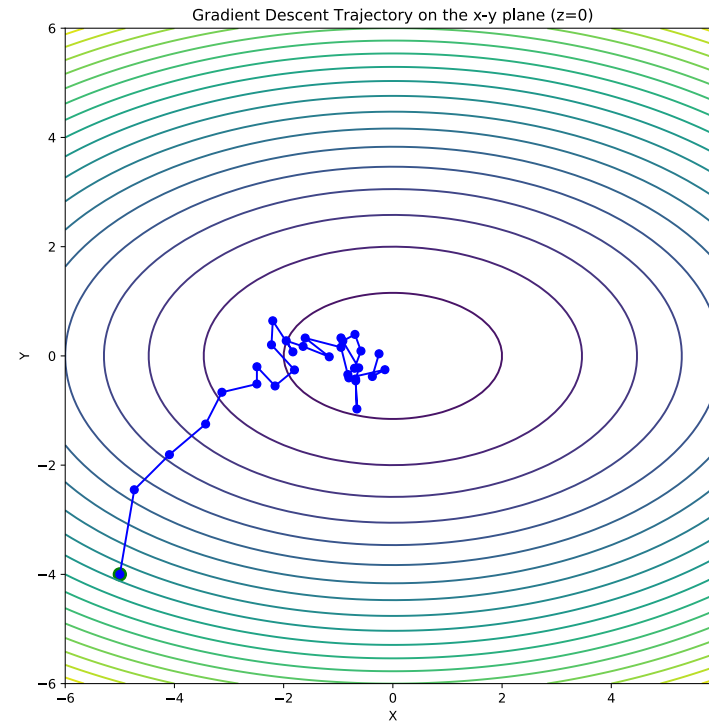
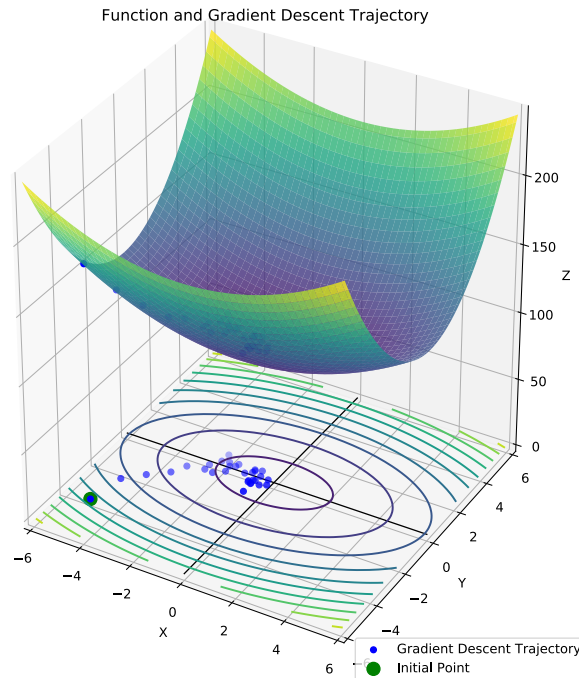
SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.

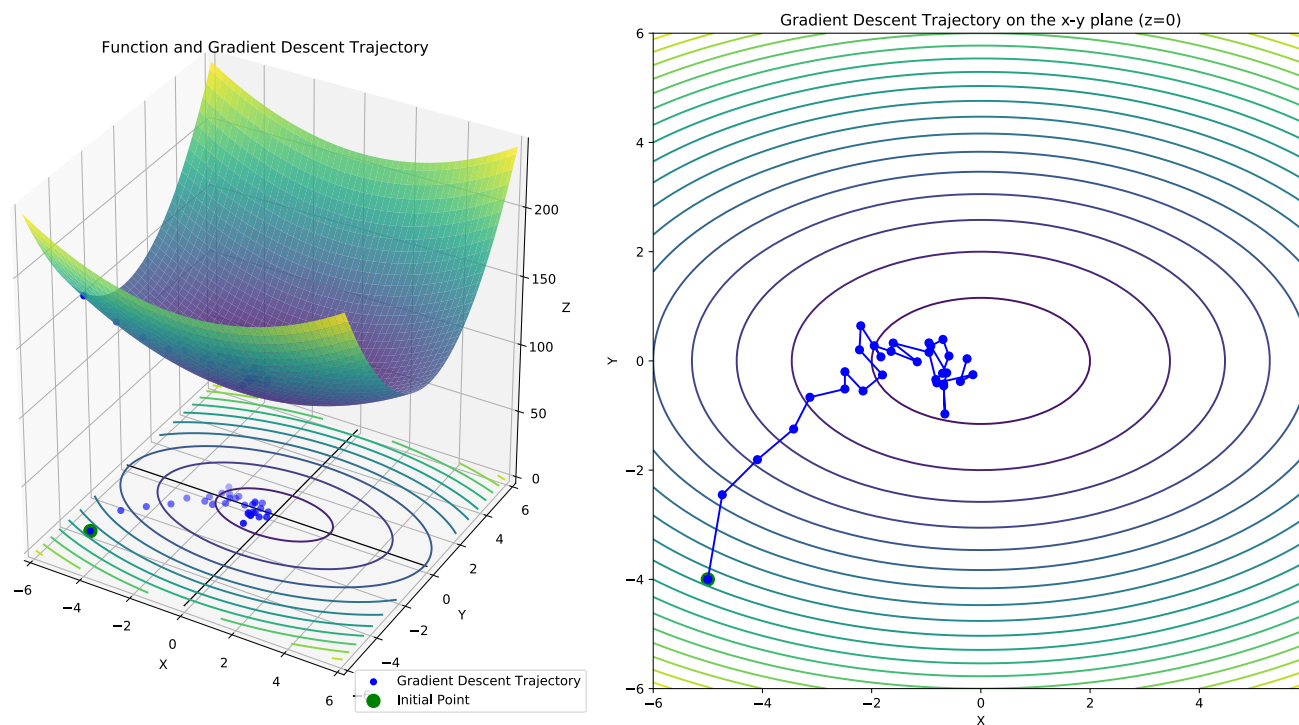


Fundamental tradeoff:

- 1 **Impact of the initial condition** (as for GD): using a **larger step** helps to forget faster the initial point.
- 2 **Impact of the noise**: using a **smaller step** makes the algorithm less sensitive to the noise.

SGD: illustration, typical behavior, impact of the learning rate

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



Fundamental tradeoff:

- 1 **Impact of the initial condition** (as for GD): using a **larger step** helps to forget faster the initial point.
- 2 **Impact of the noise**: using a **smaller step** makes the algorithm less sensitive to the noise.



The ideal step size corresponds to a balance between those two terms.

SGD: illustration, typical behavior, impact of the learning rate η

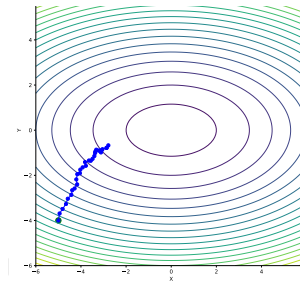
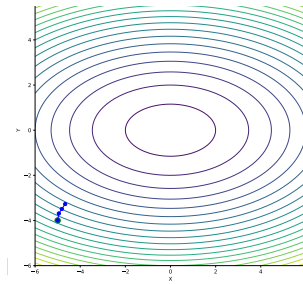
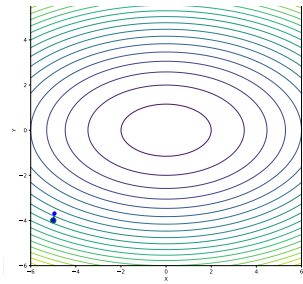
η

$T = 1$

$T = 3$

$T = 30$

too small



→ very slow convergence, as for GD. We do not forget the initial condition fast enough.

SGD: illustration, typical behavior, impact of the learning rate η

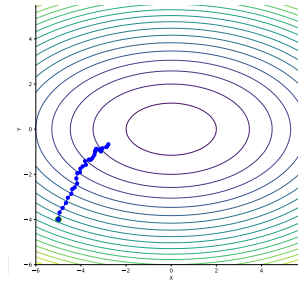
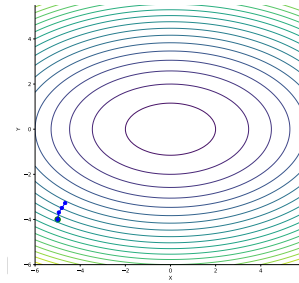
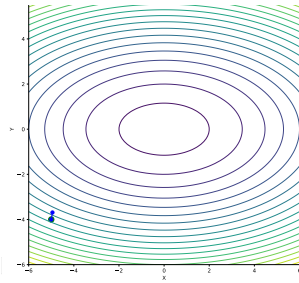
η

$T = 1$

$T = 3$

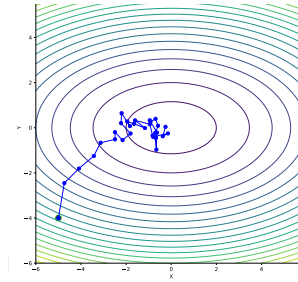
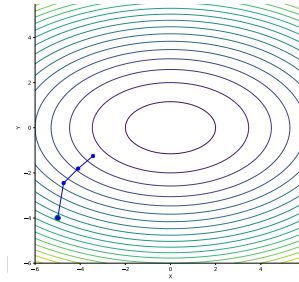
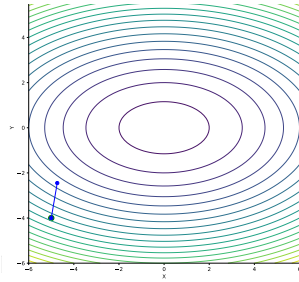
$T = 30$

too small



→ very slow convergence, as for GD. We do not forget the initial condition fast enough.

ok



→ reasonable convergence: we balance the noise and initial condition term

SGD: illustration, typical behavior, impact of the learning rate η

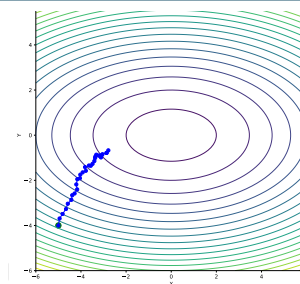
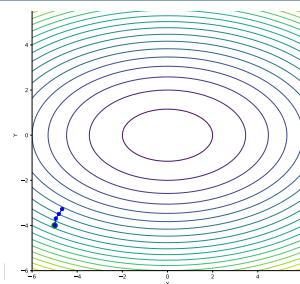
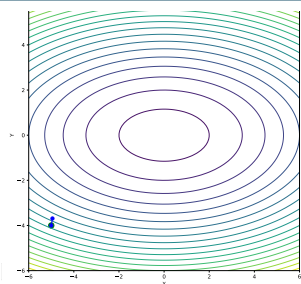
η

$T = 1$

$T = 3$

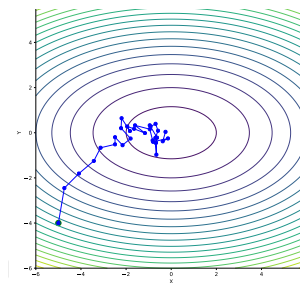
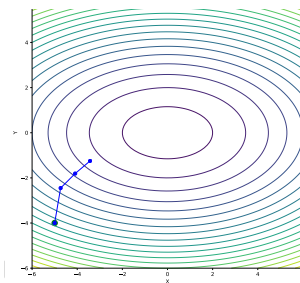
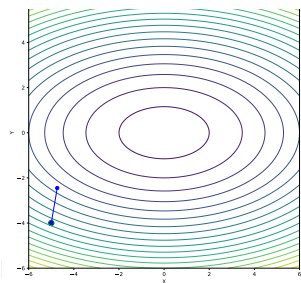
$T = 30$

too small



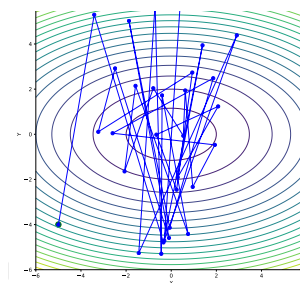
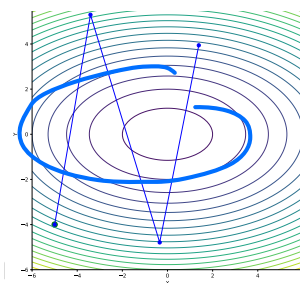
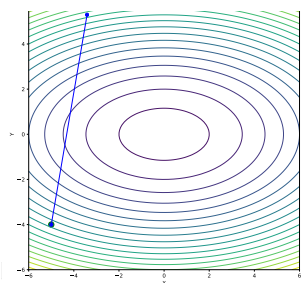
→ very slow convergence, as for GD. We do not forget the initial condition fast enough.

ok



→ reasonable convergence: we balance the noise and initial condition term

too large



→ noise dominates, although the step-size choice would be ok for GD, it is too large for SGD

$\neq$ GD.

SGD: illustration, typical behavior, impact of the learning rate η

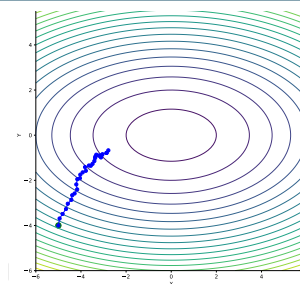
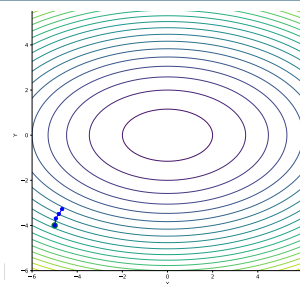
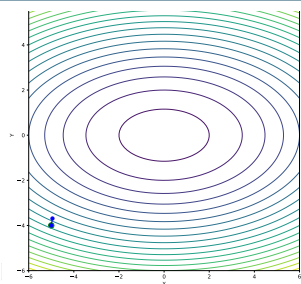
η

$T = 1$

$T = 3$

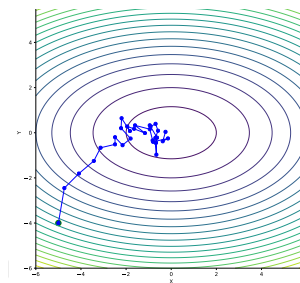
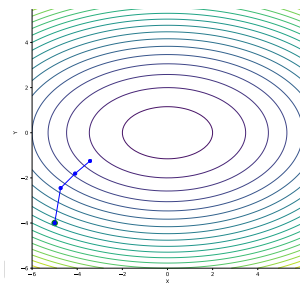
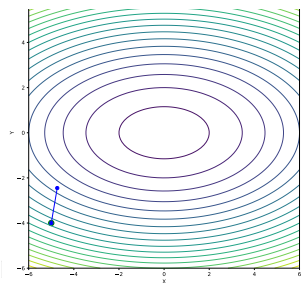
$T = 30$

too small



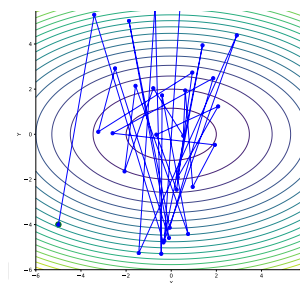
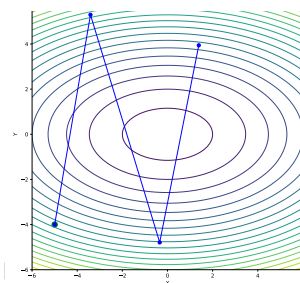
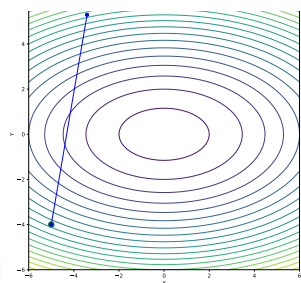
→ very slow convergence, as for GD. We do not forget the initial condition fast enough.

ok



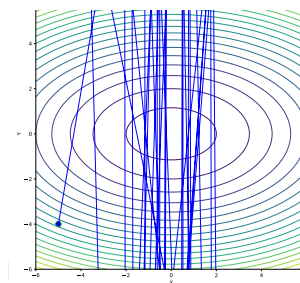
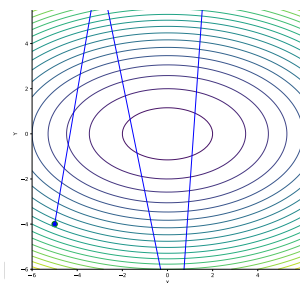
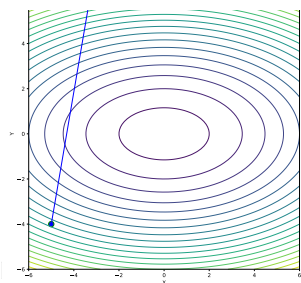
→ reasonable convergence: we balance the noise and initial condition term

too large



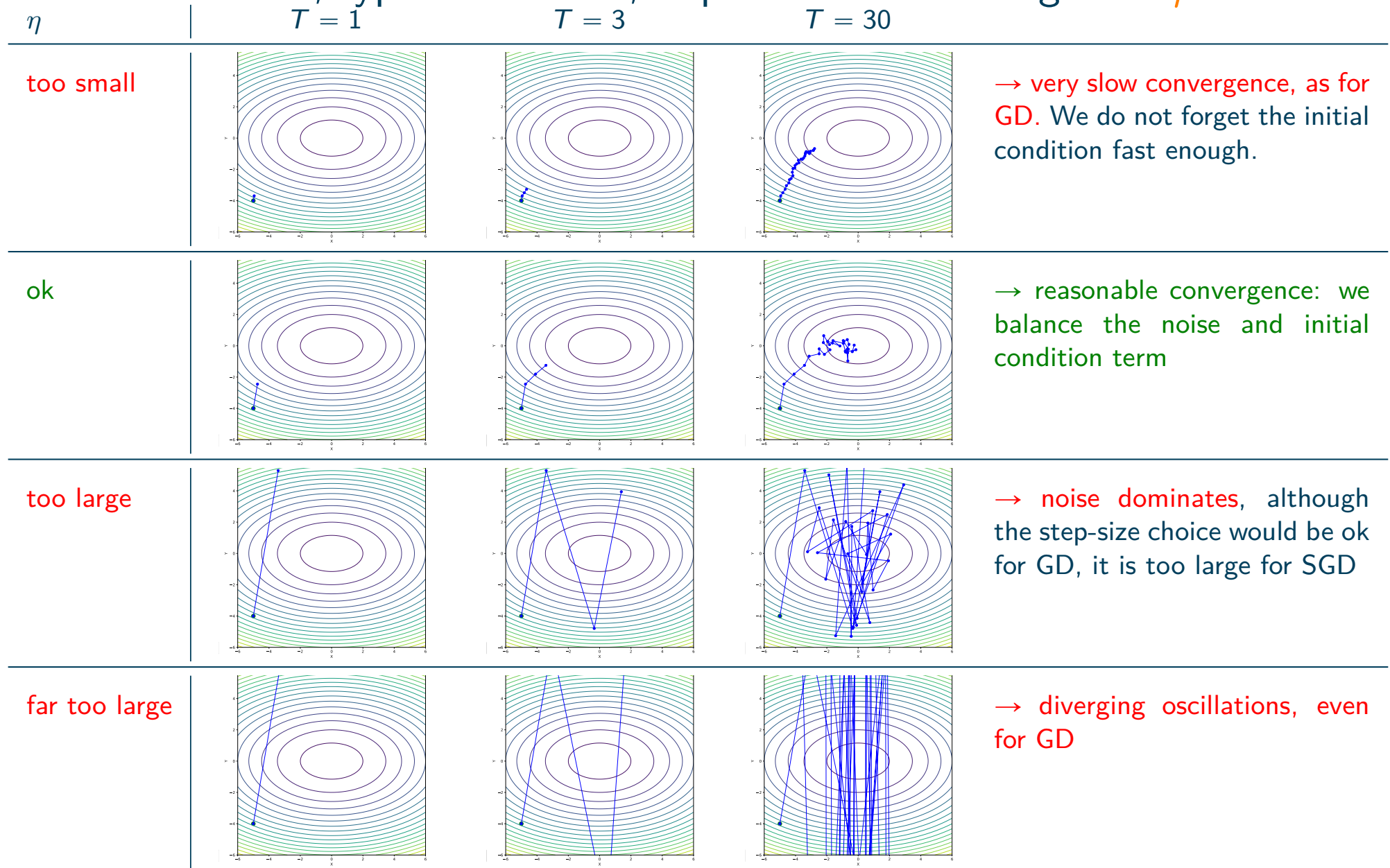
→ noise dominates, although the step-size choice would be ok for GD, it is too large for SGD

far too large



→ diverging oscillations, even for GD

SGD: illustration, typical behavior, impact of the learning rate η



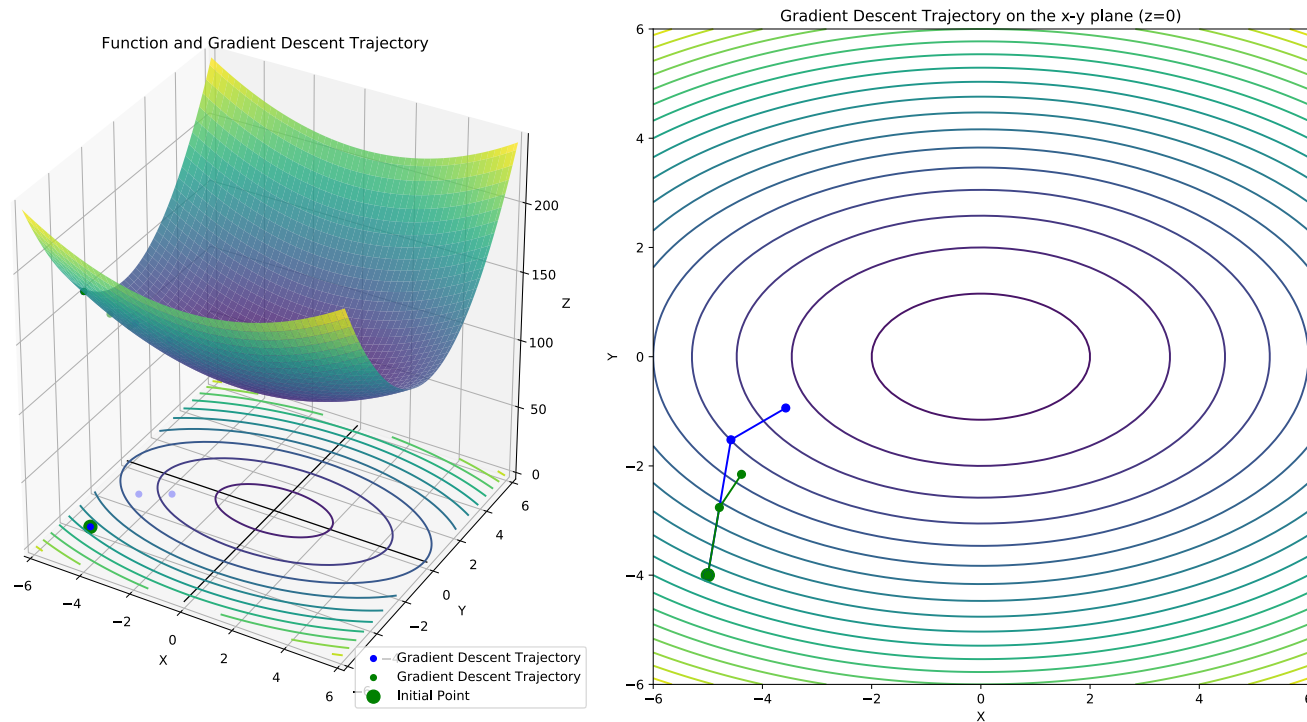
Conclusion: η needs to be much smaller than for GD, to encompass the impact of noise.

→ use a decaying sequence $\eta_t \propto \frac{1}{\sqrt{t}}$ or

→ use a step size that decreases with the total number of steps $\eta \propto \frac{1}{\sqrt{T}}$.

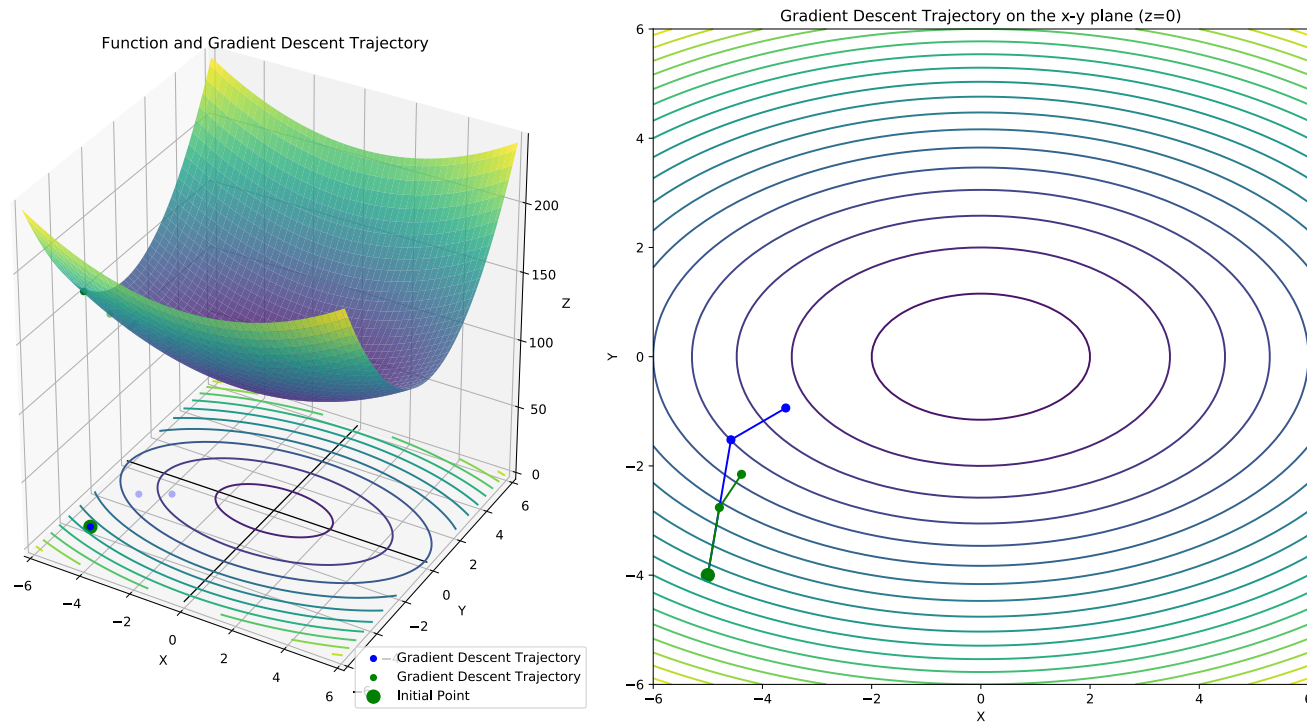
SGD: illustration, typical behavior, impact of averaging

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



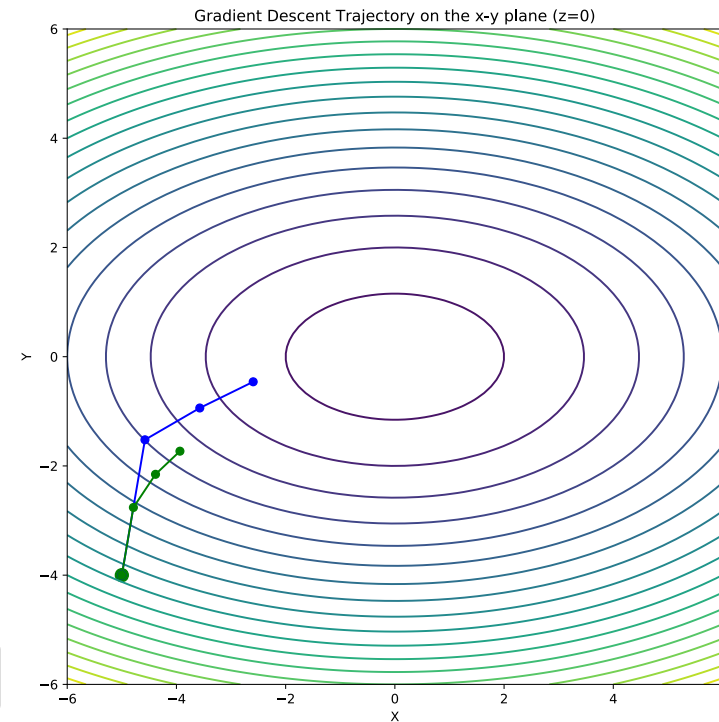
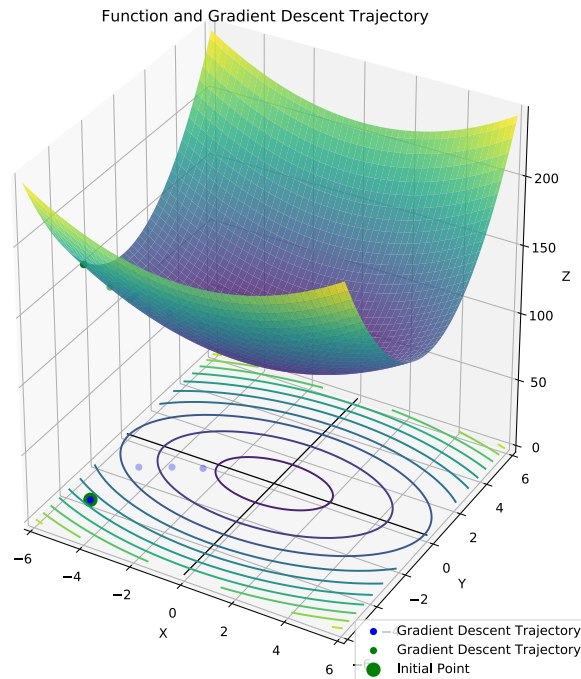
SGD: illustration, typical behavior, impact of averaging

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



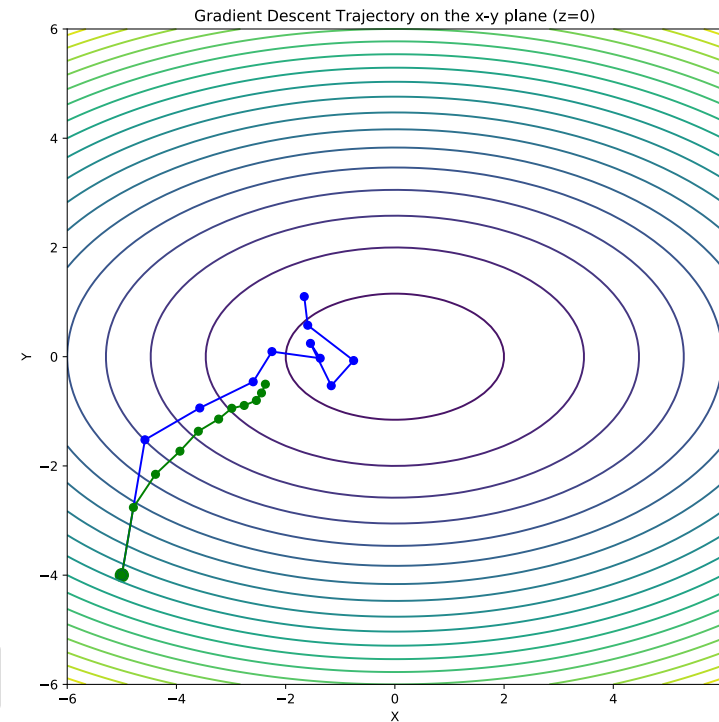
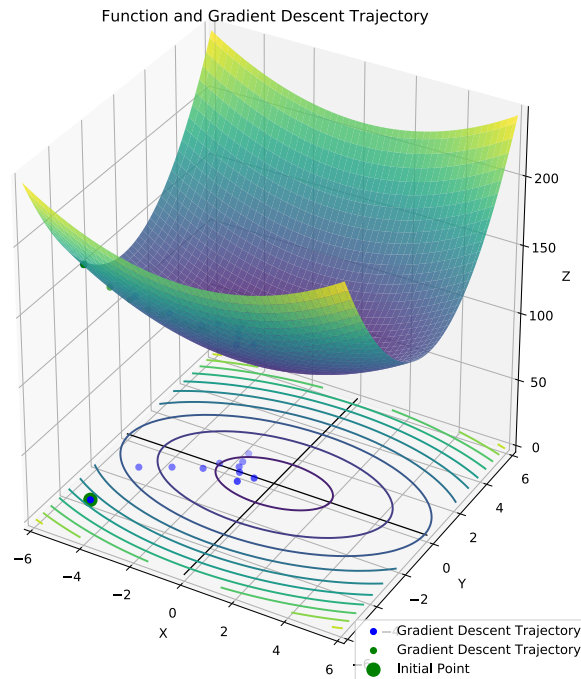
SGD: illustration, typical behavior, impact of averaging

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



SGD: illustration, typical behavior, impact of averaging

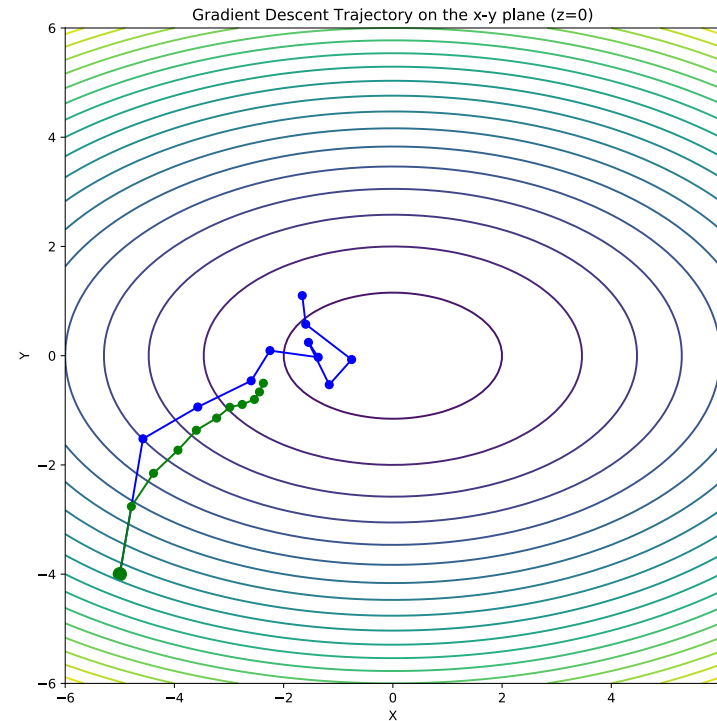
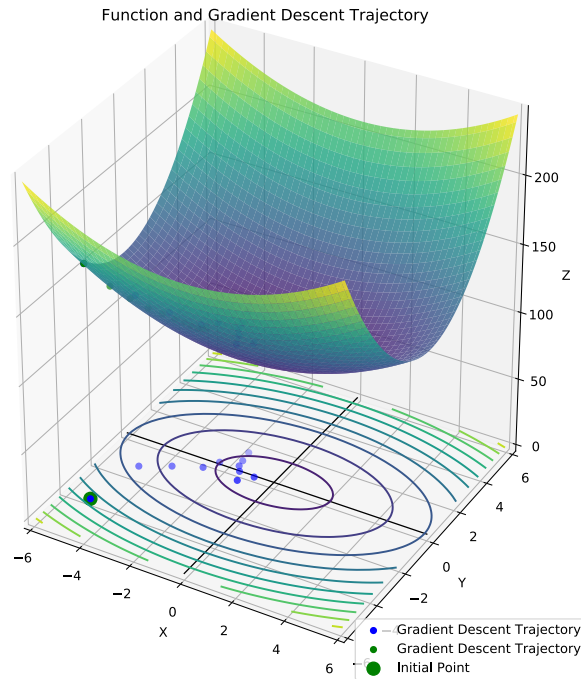
Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



SGD: illustration, typical behavior, impact of averaging

not covered yet

Behavior: follows a trajectory somehow similar to GD, but with a perturbed direction at each step.



Impact of averaging:

- 1 Reduces the impact of noise



Choice of hyperparameters, in practice

Convex case:

- Step-size: theoretically guided, typically decaying sequences.
- Averaging: often helpful from a theoretical standpoint.

In deep learning:

- Step-size: million dollar question!
Tuning the step size is known to be difficult.
→ trial and error, schedulers, etc.
Most common strategy: *reduce by a factor the step size when no progress is made.*
- Averaging: not used too much.

Question: theoretical guarantees to support intuition?

Theoretical grounds for SGD



If the batch is chosen **uniformly at random** and **independently from the past**, then g_t is, **on average**, ∇f .

Case $b = 1$: if i_t is chosen **uniformly** at random in $\{1, \dots, n\}$, independently from the past iterates, then

$$\begin{aligned}\mathbb{E}[g_t(\theta_{t-1})|\theta_{t-1}] &= \mathbb{E}[\nabla f_{i_t}(\theta_{t-1})|\theta_{t-1}] \\ &= \frac{1}{n} \sum_{i=1}^n \nabla f_i(\theta_{t-1}) \\ &= \nabla f(\theta_{t-1})\end{aligned}$$

Case $b > 1$: if $\mathcal{I}_t$ is chosen **uniformly** at random in $\{1, \dots, n\}$, with or without resampling points, independently from the past iterates, then

$$\begin{aligned}\mathbb{E}[g_t(\theta_{t-1})|\theta_{t-1}] &= \mathbb{E}\left[\frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta_{t-1})|\theta_{t-1}\right] \\ &= \frac{1}{n} \sum_{i=1}^n \nabla f_i(\theta_{t-1}) \\ &= \nabla f(\theta_{t-1})\end{aligned}$$

$g_t(\theta_{t-1})$ is an **unbiased** but **noisy** estimate of the full gradient $\nabla f(\theta_{t-1})$.

Convergence of *pure SGD* algorithm ($b = 1$), to minimize $f = \frac{1}{n} \sum_{i=1}^n f_i$

Theorem: convergence of SGD on smooth functions

If for all $i \in \{1, \dots, n\}$, f_i is convex and L -smooth and there exists $\sigma^2 > 0$:

$$\text{Var}(g_t(\theta)) = \mathbb{E}_{i \sim \mathcal{U}\{1, \dots, n\}} [\|\nabla f_i(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2.$$

- 1 We need to control the *variance* of the *unbiased* stochastic gradient.

Remark: one can use a decreasing sequence $\eta_t = \|\theta_0 - \theta_\star\|/(\sigma\sqrt{t})$ for all $t \in \{1, \dots\}$.

Convergence of *pure SGD* algorithm ($b = 1$), to minimize $f = \frac{1}{n} \sum_{i=1}^n f_i$

Theorem: convergence of SGD on smooth functions

If for all $i \in \{1, \dots, n\}$, f_i is convex and L -smooth and there exists $\sigma^2 > 0$:

$$\text{Var}(g_t(\theta)) = \mathbb{E}_{i \sim \mathcal{U}\{1, \dots, n\}} [\|\nabla f_i(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2.$$

Then, with a constant learning rate sequence $\eta_t = \eta \leq \frac{1}{L}$ for all $t \leq T$:

$$\mathbb{E}[f(\bar{\theta}_T)] - f(\theta_\star) \leq \frac{\|\theta_0 - \theta_\star\|^2}{\eta T} + \eta \sigma^2. \quad (2)$$

- 1 We need to control the *variance* of the *unbiased* stochastic gradient.

Remark: one can use a decreasing sequence $\eta_t = \|\theta_0 - \theta_\star\|/(\sigma\sqrt{t})$ for all $t \in \{1, \dots\}$.

Convergence of *pure SGD* algorithm ($b = 1$), to minimize $f = \frac{1}{n} \sum_{i=1}^n f_i$

Theorem: convergence of SGD on smooth functions

If for all $i \in \{1, \dots, n\}$, f_i is convex and L -smooth and there exists $\sigma^2 > 0$:

$$\text{Var}(g_t(\theta)) = \mathbb{E}_{i \sim \mathcal{U}\{1, \dots, n\}} [\|\nabla f_i(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2.$$

Then, with a constant learning rate sequence $\eta_t = \eta \leq \frac{1}{L}$ for all $t \leq T$:

$$\mathbb{E} [f(\bar{\theta}_T)] - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\eta T} + \eta \sigma^2. \quad (2)$$

- ① We need to control the *variance* of the *unbiased* stochastic gradient.
- ② Eq. 2 confirms the existence of an initial condition / noise tradeoff.
 - The initial condition term $\frac{\|\theta_0 - \theta_*\|^2}{\eta T}$ is the *same* as for GD, decreases with η .
 - The extra noise term $\eta \sigma^2$ *increases* with η .
- ③ The control is for the averaged iterate $\bar{\theta}_T$.

Remark: one can use a decreasing sequence $\eta_t = \|\theta_0 - \theta_*\|/(\sigma\sqrt{t})$ for all $t \in \{1, \dots\}$.

Convergence of *pure SGD* algorithm ($b = 1$), to minimize $f = \frac{1}{n} \sum_{i=1}^n f_i$

Theorem: convergence of SGD on smooth functions

If for all $i \in \{1, \dots, n\}$, f_i is convex and L -smooth and there exists $\sigma^2 > 0$:

$$\text{Var}(g_t(\theta)) = \mathbb{E}_{i \sim \mathcal{U}\{1, \dots, n\}} [\|\nabla f_i(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2.$$

Then, with a constant learning rate sequence $\eta_t = \eta \leq \frac{1}{L}$ for all $t \leq T$:

$$\mathbb{E}[f(\bar{\theta}_T)] - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\eta T} + \eta \sigma^2. \quad (2)$$

Moreover setting $\eta = \|\theta_0 - \theta_*\|/(\sigma\sqrt{T})$ to optimize this sum,

$$\mathbb{E}[f(\bar{\theta}_T)] - f(\theta_*) \leq \frac{2\|\theta_0 - \theta_*\|\sigma}{\sqrt{T}}. \quad (3)$$

- ① We need to control the *variance* of the *unbiased* stochastic gradient.
- ② Eq. 2 confirms the existence of an initial condition / noise tradeoff.
 - The initial condition term $\frac{\|\theta_0 - \theta_*\|^2}{\eta T}$ is the *same* as for GD, decreases with η .
 - The extra noise term $\eta \sigma^2$ *increases* with η .
- ③ The control is for the averaged iterate $\bar{\theta}_T$.

Remark: one can use a decreasing sequence $\eta_t = \|\theta_0 - \theta_*\|/(\sigma\sqrt{t})$ for all $t \in \{1, \dots\}$.

Convergence of *pure SGD* algorithm ($b = 1$), to minimize $f = \frac{1}{n} \sum_{i=1}^n f_i$

Theorem: convergence of SGD on smooth functions

If for all $i \in \{1, \dots, n\}$, f_i is convex and L -smooth and there exists $\sigma^2 > 0$:

$$\text{Var}(g_t(\theta)) = \mathbb{E}_{i \sim \mathcal{U}\{1, \dots, n\}} [\|\nabla f_i(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2.$$

Then, with a constant learning rate sequence $\eta_t = \eta \leq \frac{1}{L}$ for all $t \leq T$:

$$\mathbb{E} [f(\bar{\theta}_T)] - f(\theta_*) \leq \frac{\|\theta_0 - \theta_*\|^2}{\eta T} + \eta \sigma^2. \quad (2)$$

Moreover setting $\eta = \|\theta_0 - \theta_*\|/(\sigma\sqrt{T})$ to optimize this sum,

$$\mathbb{E} [f(\bar{\theta}_T)] - f(\theta_*) \leq \frac{2\|\theta_0 - \theta_*\|\sigma}{\sqrt{T}}. \quad (3)$$

- ① We need to control the *variance* of the *unbiased* stochastic gradient.
- ② Eq. 2 confirms the existence of an initial condition / noise tradeoff.
 - The initial condition term $\frac{\|\theta_0 - \theta_*\|^2}{\eta T}$ is the *same* as for GD, decreases with η .
 - The extra noise term $\eta \sigma^2$ *increases* with η .
- ③ The control is for the averaged iterate $\bar{\theta}_T$.
- ④ Eq. 3 give a good choice of step size and the corresponding rate.
 - The “**optimal**” **step-size** to minimize the bound decreases with T , scaling as $1/\sqrt{T}$.
 - The corresponding rate scales as $1/\sqrt{T}$.

Remark: one can use a decreasing sequence $\eta_t = \|\theta_0 - \theta_*\|/(\sigma\sqrt{t})$ for all $t \in \{1, \dots\}$.

Convergence rate of SGD, smooth and strongly convex functions.

Consider the stochastic gradient descent algorithm introduced previously but where each iteration is projected into the ball $B(0, R)$ with $R > 0$ fixed.

Let's recap! Wooclap quizz!



Outline

- 1 Optimization for machine learning: motivation and course objectives
- 2 Preliminaries and mathematical pre-requisites
- 3 Gradient descent method, stochastic gradient method
 - Gradient Descent (GD)
 - Stochastic Gradient Descent (SGD)
 - End of lecture 1 - at your own pace quizz!

End of lecture 1 - at your own pace quizz!

Go to *[this link](#)* !