

Online Convex Optimization

Tutorial

Chu Duc Thang

Agenda

Online Convex Optimization

1. Convex Optimization
2. Online Convex Optimization
3. First order solution
4. Second order solution
5. Regularization

1. Convex Optimization

Motivation

- Online Convex Optimization (OCO) can be decomposed into 2 parts: **Convex** and **Online** Optimization.
- Why focus on **Convex Optimization (CO)**?
 - **Simple**, but **powerful** properties (No spoilers!)
 - **Reliable algorithms** to solve these problems.
 - In practice: CO problems usually do not exist.
- **Disclaimer:** non-exhausted list of all topics in CO!

1. Convex Optimization

1.1 Optimization problem

- **General form:** $\min_x f(x)$ subject to $x \in C$
 - **Decision variables:** $x = (x_1, \dots, x_n)^T$:
 - If $x_i : \forall i = 1, \dots, n$ are continuous, then the problem is a **continuous** optimization. Otherwise, it is a **discrete** optimization.
 - **Objective function:** $f : \mathbb{R}^n \rightarrow \mathbb{R}$
 - **Decision space/Constraint set/Feasible region:** C

1. Convex Optimization

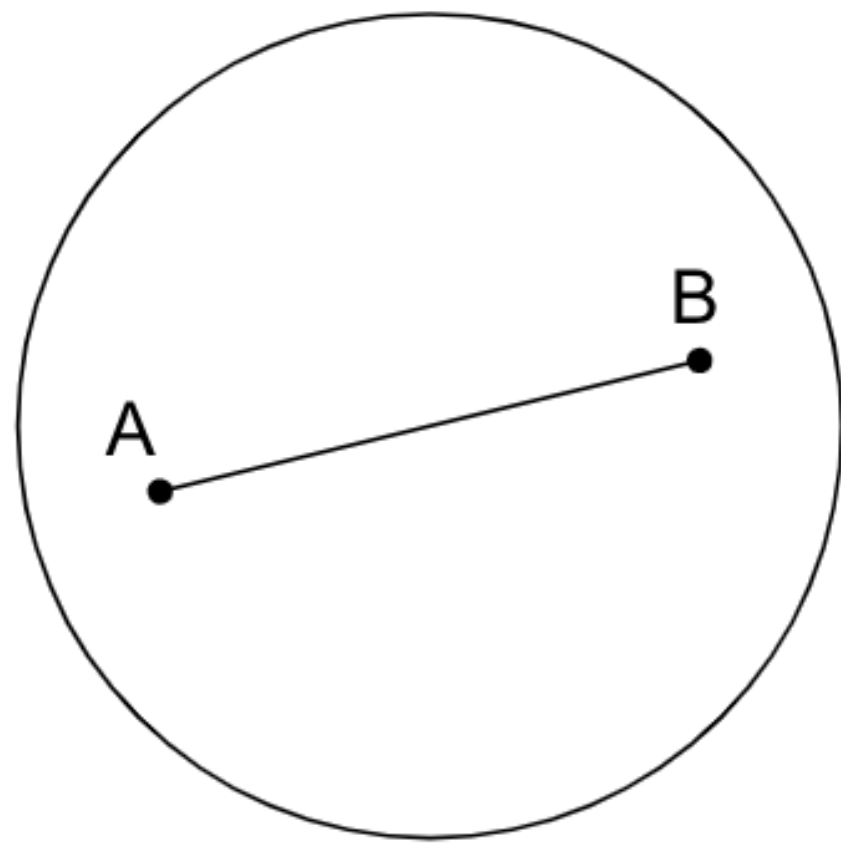
1.1 Optimization problem

- **General form:** $\min_x f(x)$ subject to $x \in C \quad \Rightarrow \quad$ **Specific form:** $\min_x f_0(x)$ subject to $\begin{cases} f_i(x) \leq 0 & i = 1, \dots, m \\ h_j(x) = 0 & j = 1, \dots, l \end{cases}$
- $C = \{x \mid f_i(x) \leq 0, \forall i = 1, \dots, m; h_j(x) = 0, \forall j = 1, \dots, l\}$
- **Inequality constraint** f_i , **Equality constraint:** h_j
- **Some examples:**
 - **Unconstrained optimization:** Simply does not have the “subject to” part ($C \subset \mathbb{R}^n$)
 - **Linear programming:** $f_i, \forall i = 0, \dots, m; h_j, \forall j = 1, \dots, l$ are linear functions
 - **Integer programming:** x are all integer
 - **Convex optimization:** $f_i, \forall i = 0, \dots, m$ are **convex functions** and $h_j : \forall j = 1, \dots, n$ are **equality constraints**. (C is a convex set)

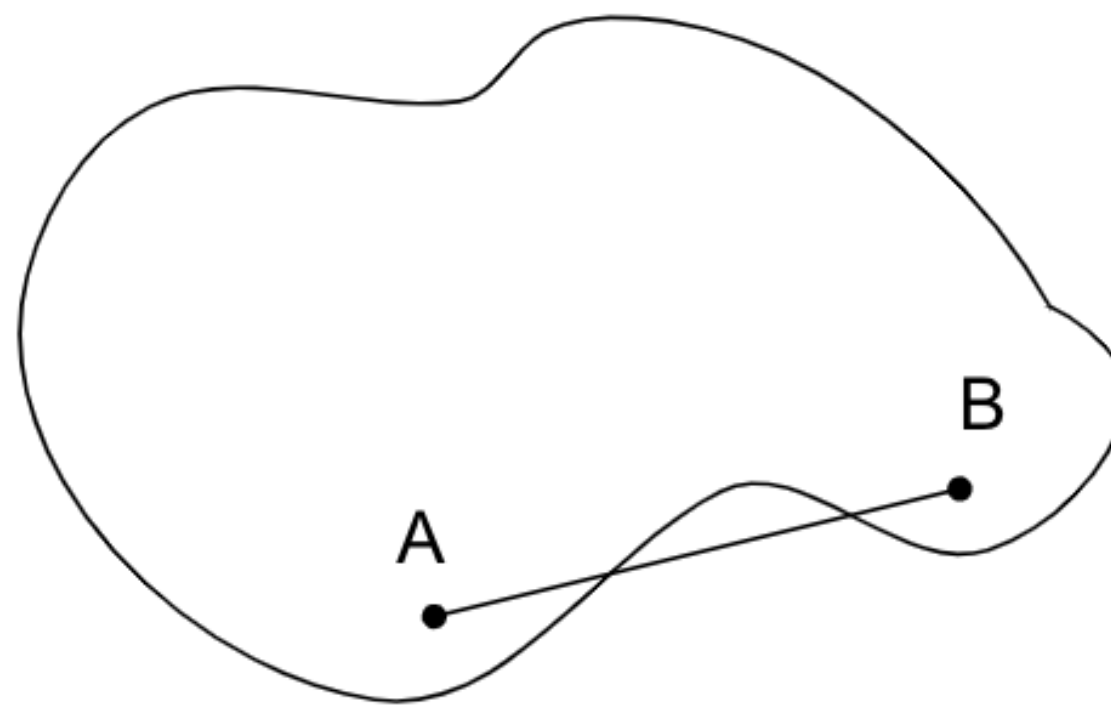
1. Convex Optimization

1.2 Convex set

- A set \mathbb{K} is **convex** if $\forall x, y \in \mathbb{K}$, then $\alpha x + (1 - \alpha)y \in \mathbb{K}, \forall \alpha \in [0,1]$
 - **Example:** Singleton set, \mathbb{R}^n , lines, etc.



Convex Set

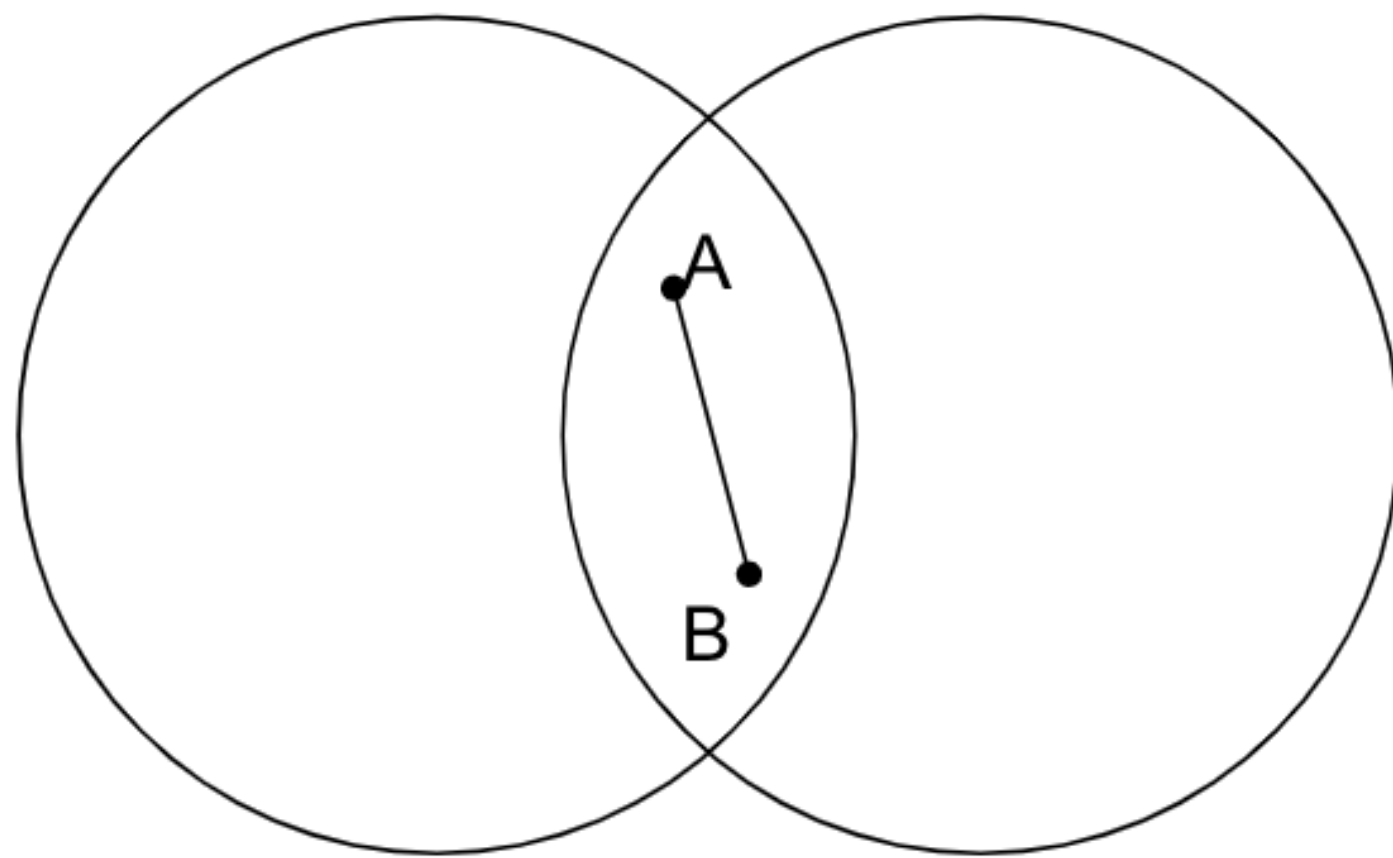


Non-Convex Set

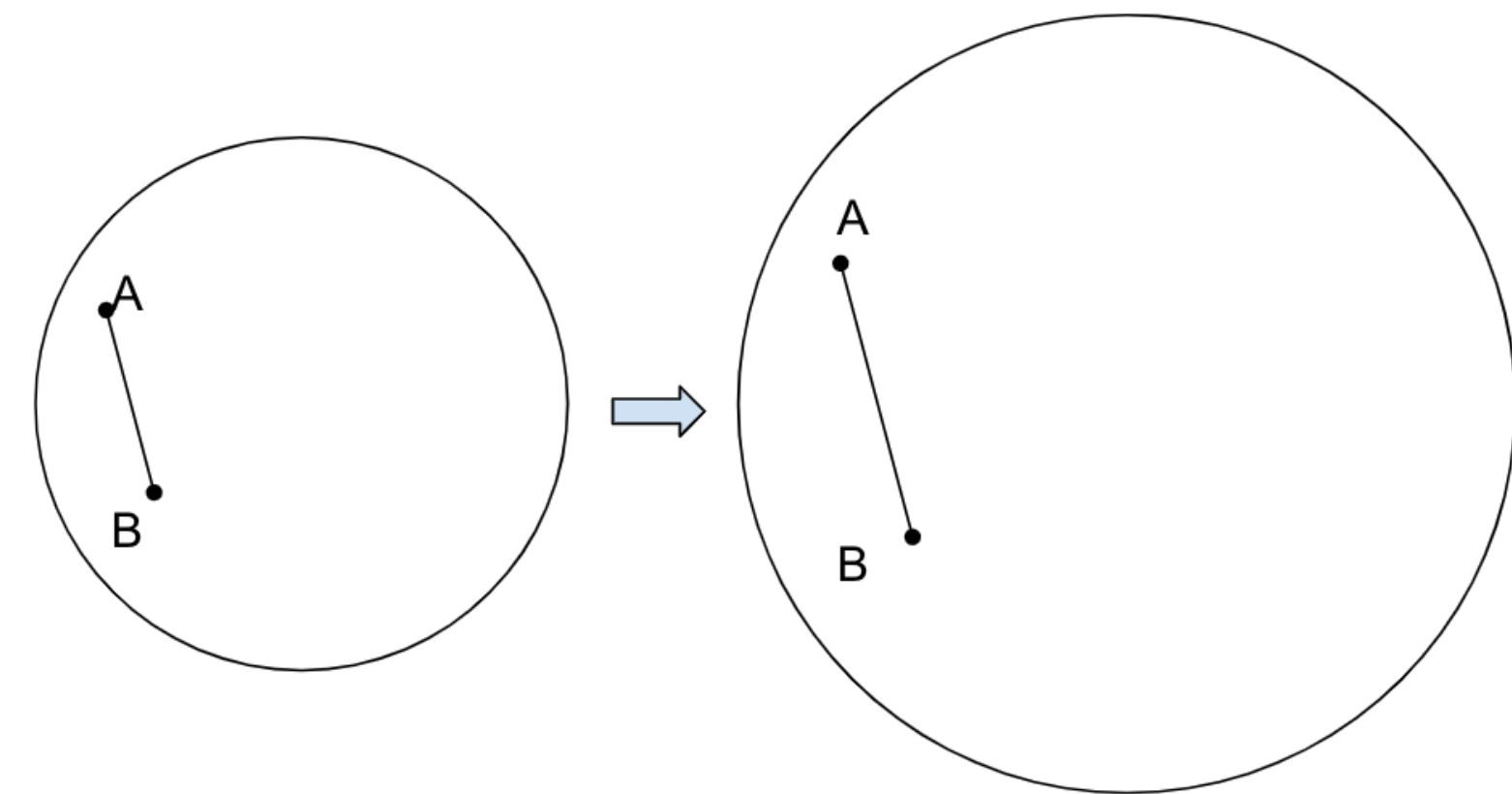
1. Convex Optimization

1.2 Convex set

- A set \mathbb{K} is **convex** if $\forall x, y \in \mathbb{K}$, then $\alpha x + (1 - \alpha)y \in \mathbb{K}, \forall \alpha \in [0,1]$
 - **Convexity preserving operations:** Intersection, Affine transformation, etc.



Intersection

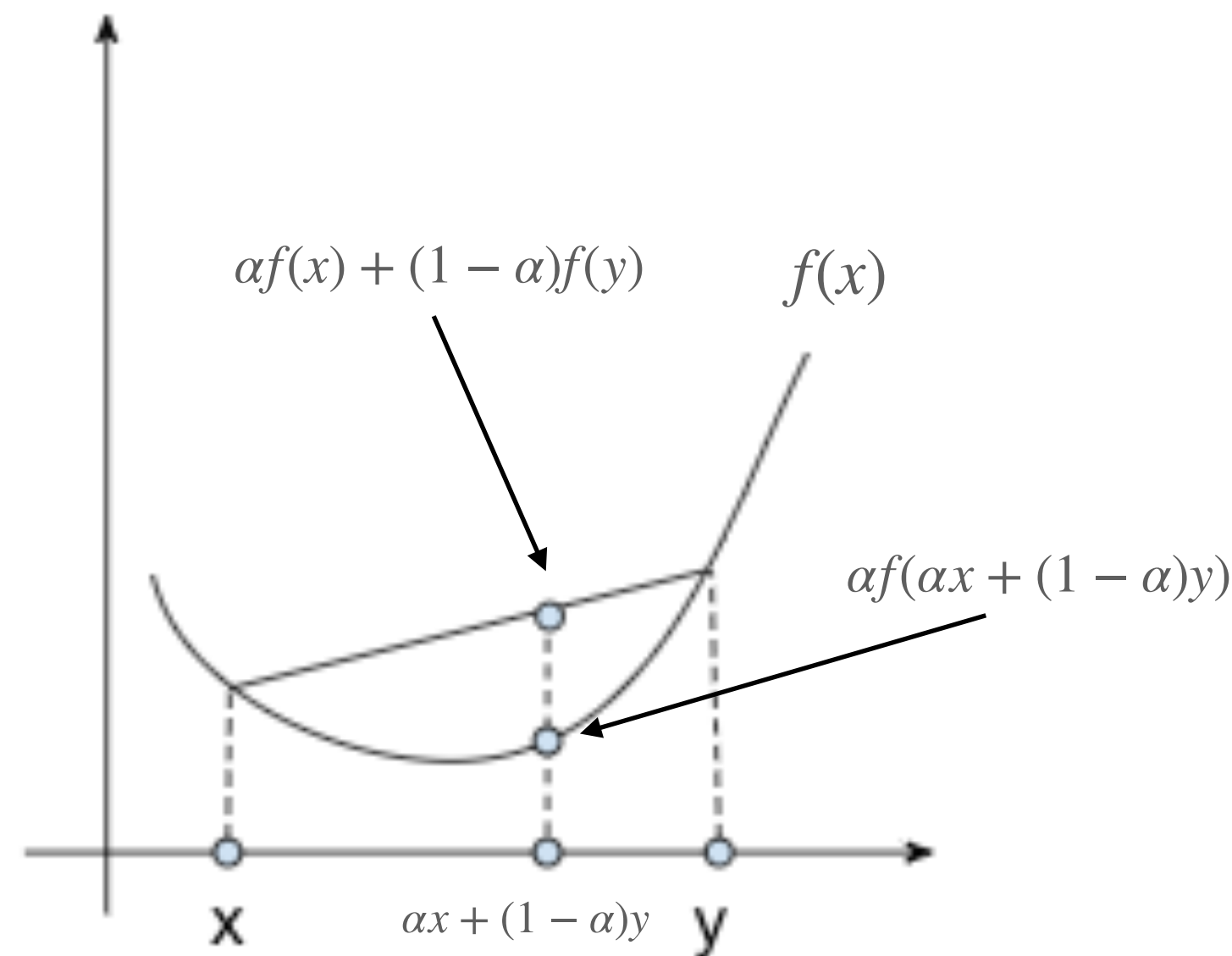


Affine transformation - Scaling

1. Convex Optimization

1.3 Convex function

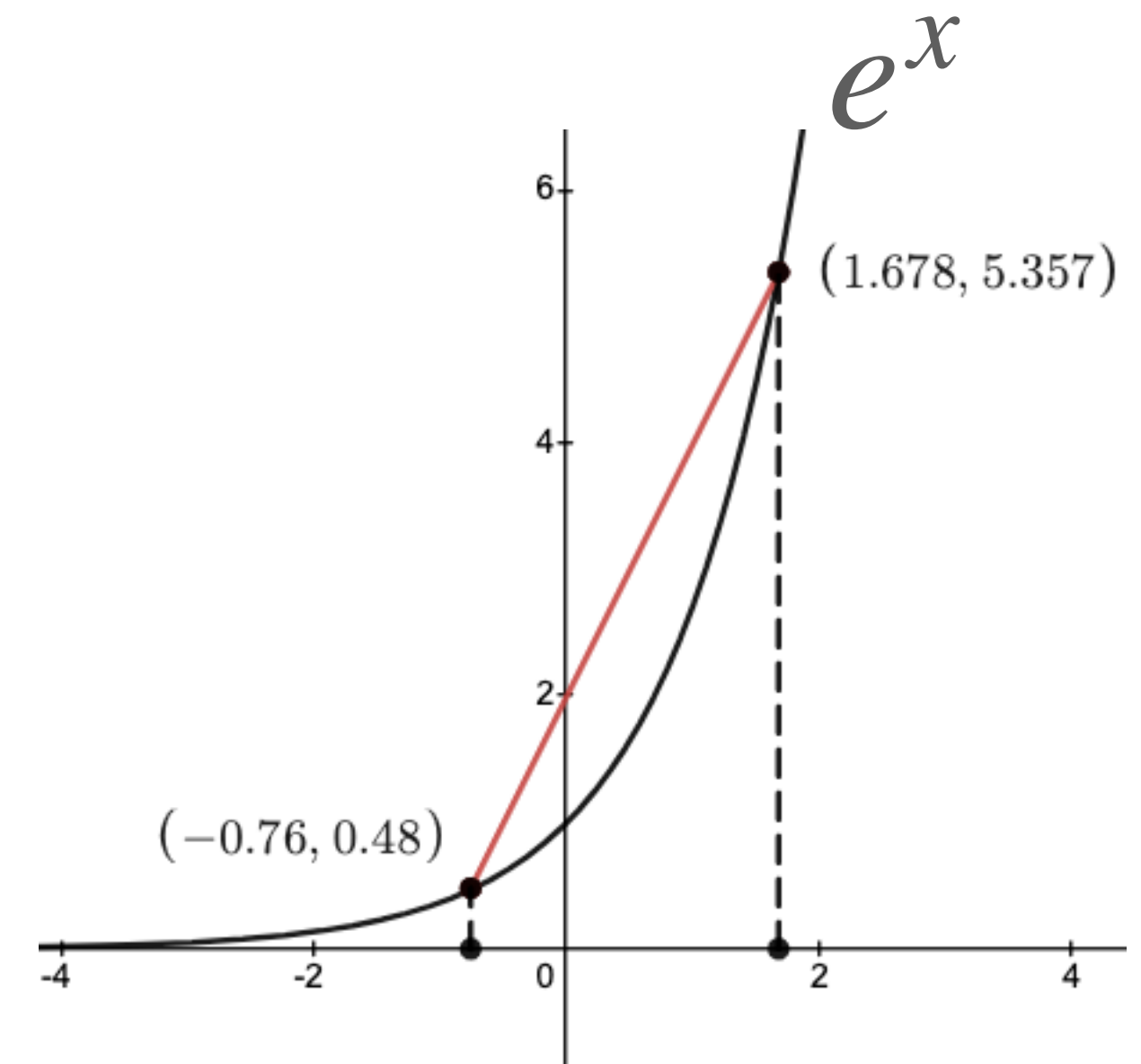
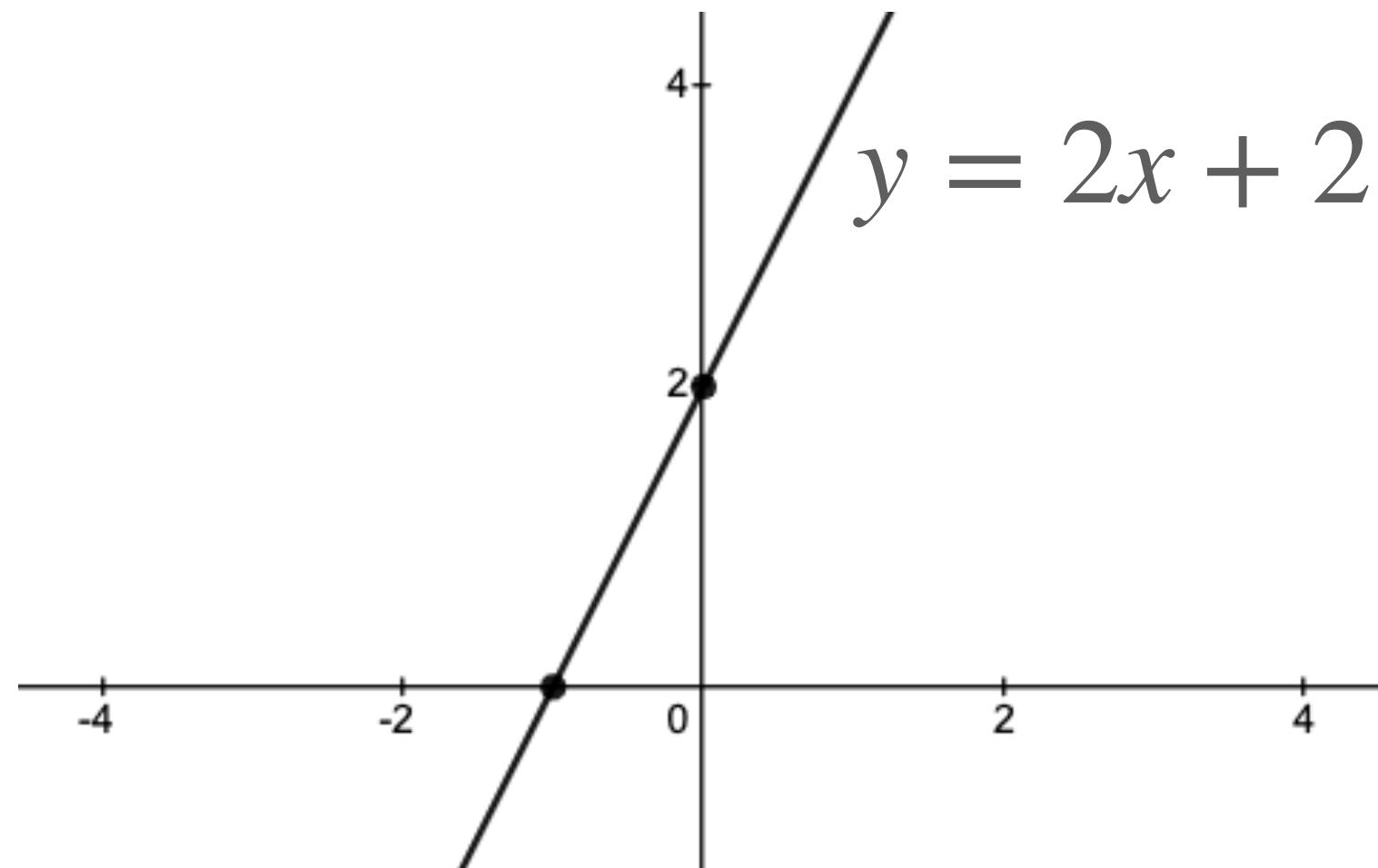
- $f: \mathbb{K} \rightarrow \mathbb{R}$ is convex if \mathbb{K} is a **convex set** and $\forall x, y \in \mathbb{K}$,
 $f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$, $\forall 0 \leq \alpha \leq 1$
 - **Remark:** If the inequality is $<$, then the function f is **strictly convex**.



1. Convex Optimization

1.3 Convex function

- $f: \mathbb{K} \rightarrow \mathbb{R}$ is convex if \mathbb{K} is a **convex set** and $\forall x, y \in \mathbb{K}$,
 $f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \forall 0 \leq \alpha \leq 1$
 - **Example (in \mathbb{R}):** $ax + b (\forall a, b \in \mathbb{R}), e^{ax} (\forall a \in \mathbb{R})$, etc.



1. Convex Optimization

1.4 Convex optimization problem

$$\bullet \min_x f_0(x) \text{ subject to } \begin{cases} f_i(x) \leq 0 & i = 1, \dots, m \\ h_j(x) = 0 & j = 1, \dots, l \end{cases} \quad \min_x f_0(x) \text{ subject to } \begin{cases} f_i(x) \leq 0 & i = 1, \dots, m \\ a_i^T x = b_i & j = 1, \dots, l \end{cases}$$

► $f_i, \forall i = 0, \dots, m$ are **convex functions**. $h_j, \forall j = 1, \dots, l$ are **linear constraints**.

► **Feasible set** $C = \{x \mid f_i(x) \leq 0, \forall i = 1, \dots, n; Ax = b\}$ is a **convex set**.

Proof: Suppose $x_1, x_2 \in C \rightarrow f_i(x_1) \leq 0, f_i(x_2) \leq 0, \forall i = 1, \dots, m; Ax_1 = Ax_2 = b$

$$f_i(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f_i(x_1) + (1 - \alpha)f_i(x_2) \leq 0 \text{ (Since } f_i \text{ is a convex function)}$$

$$\alpha Ax_1 + (1 - \alpha)Ax_2 = \alpha b + b - \alpha b = b$$

Therefore, $\alpha x_1 + (1 - \alpha)x_2 \in C, \alpha \in [0, 1]$. Therefore, C is a convex set by definition.

1. Convex Optimization

1.4 Convex optimization problem

- $\min_x f_0(x)$ subject to $\begin{cases} f_i(x) \leq 0 & i = 1, \dots, m \\ Ax = b \end{cases}$
- **Theorem 1.4.1:** Let \mathbb{K} be a convex set. If f is a **convex function**, then any **local minimums** of f are also **global minimums**.

Proof: Let \mathbb{K} be a convex set, f is a convex function, and x^* is a local minimum of f . By definition of local minimum, for some neighbourhood $N \subseteq \mathbb{K}$ about x^* , we have $f(x) > f(x^*)$, $\forall x \in N$. Suppose there exists $\tilde{x} \in \mathbb{K}$ s.t $f(\tilde{x}) < f(x^*)$, we will prove by contradiction.

Through \tilde{x} and x^* , we can have a line segment $\alpha\tilde{x} + (1 - \alpha)x^*$, $\alpha \in [0, 1]$.

$\bar{x} = \alpha\tilde{x} + (1 - \alpha)x^* \in \mathbb{K}$ (Definition of convex set)

$f(\bar{x}) = f(\alpha\tilde{x} + (1 - \alpha)x^*) \leq \alpha f(\tilde{x}) + (1 - \alpha)f(x^*) \leq \alpha f(x^*) + (1 - \alpha)f(x^*) = f(x^*)$ (Definition of convex function and assumption)

We can pick α such that $\bar{x} \in N$. Then, $f(\bar{x}) > f(x^*)$, but this contradicts with above inequality $f(\bar{x}) \leq f(x^*)$. Therefore, there are no point \tilde{x} s.t $f(\tilde{x}) < f(x^*)$. In other words, x^* is a **global minimum**.

1. Convex Optimization

1.4 Convex optimization problem

- $\min_x f_0(x)$ subject to
$$\begin{cases} f_i(x) \leq 0 & i = 1, \dots, m \\ Ax = b \end{cases}$$
- **Theorem 1.4.2:** Let $\text{dom}(f)$ be a convex set. If f is a strictly convex function, then there is **at most 1** local minimum. Consequently, it is the **unique global minimum** of f .

Proof:

1. Convex Optimization

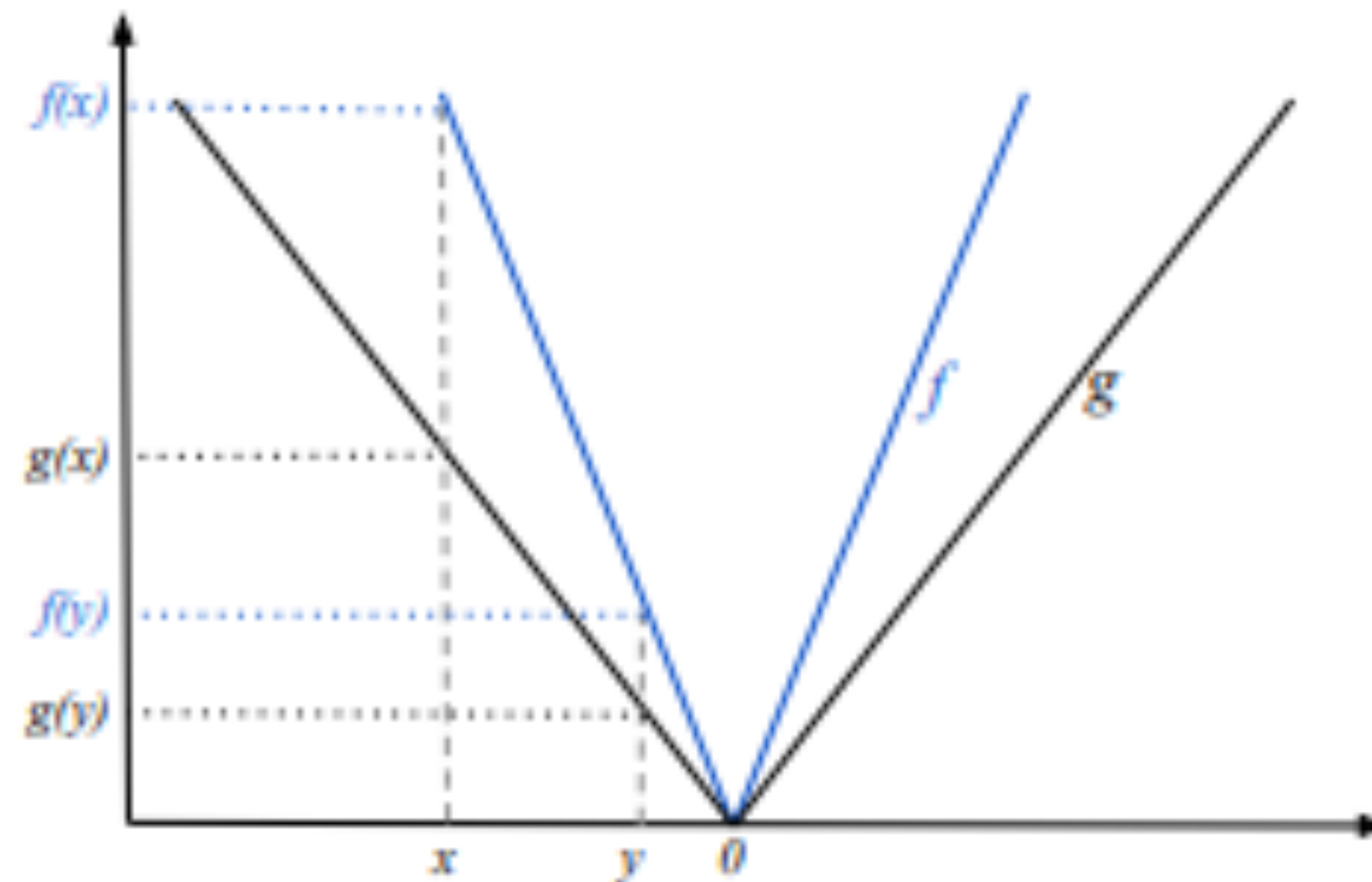
1.5 Upper bound of set's diameter

- **Upper bound of the diameter of set \mathbb{K} :** $\forall x, y \in \mathbb{K}, ||x - y|| \leq D$
 - **Remark:** \mathbb{K} is closed set on \mathbb{R}^d ($\mathbb{K} \subseteq \mathbb{R}^d$)

1. Convex Optimization

1.6 Bound norm of the sub-gradients

- **G-Lipschitz function** $f : \mathbb{K} \rightarrow \mathbb{R}$: $\forall x, y \in \mathbb{K}$, we have $|f(x) - f(y)| \leq G ||x - y||$
 - **Intuition:** G -Lipschitz function is bounded in how fast it change.
 - **Remark:** f can be G -Lipschitz, but not continuous



1. Convex Optimization

1.6 Bound norm of the subgradients

- **G-Lipschitz function** $f : \mathbb{K} \rightarrow \mathbb{R}$: $\forall x, y \in \mathbb{K}$, we have $|f(x) - f(y)| \leq G ||x - y||$
- **First order condition for Lipschitz continuity**: f is differentiable G -Lipschitz function if and only if $G > 0 : \forall x \in \mathbb{K}, ||\nabla f(x)|| \leq G$.

Proof:

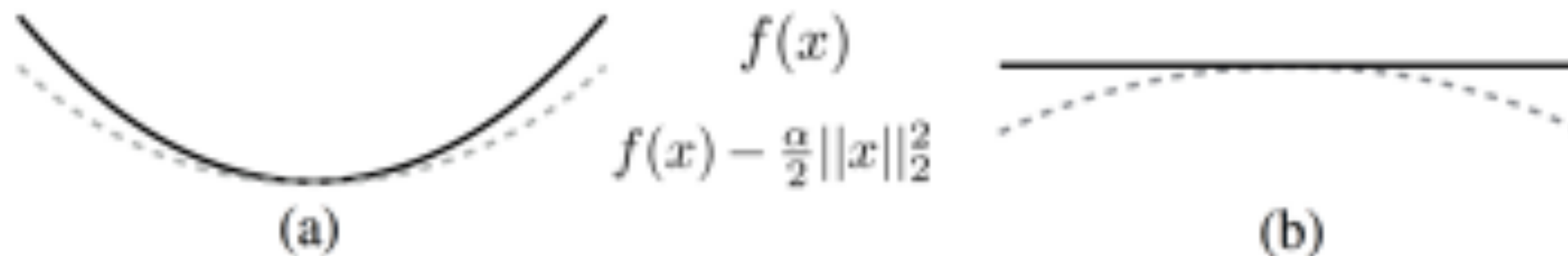
1. Convex Optimization

1.7 α -strongly convex

- $f : \mathbb{K} \rightarrow \mathbb{R}$ is α -strongly convex if

$$\forall x, y \in \mathbb{K}, f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2} \|y - x\|^2$$

- **Intuition:** α is a **lower-bound measure of curvature**. For instance, if α is large, then the convergence is faster (point far from optimum will have a large gradient).

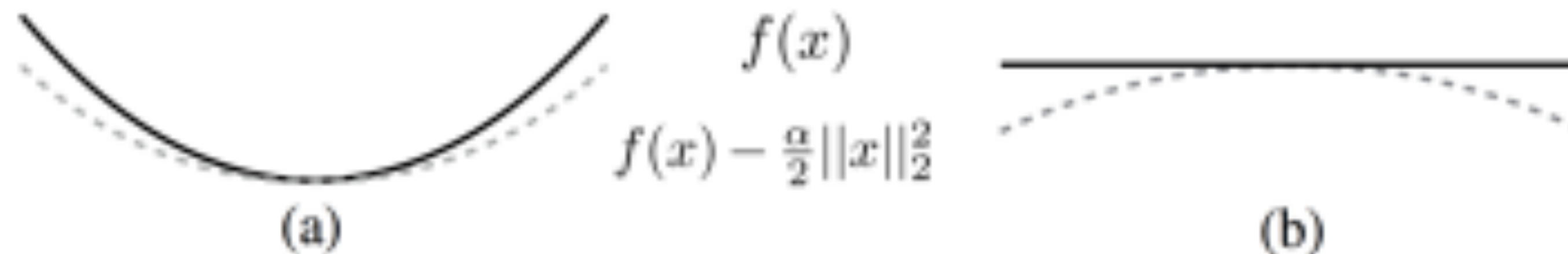


1. Convex Optimization

1.7 α -strongly convex

- $f: \mathbb{K} \rightarrow \mathbb{R}$ is α -strongly convex if
$$\forall x, y \in \mathbb{K}, f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\alpha}{2} ||y - x||^2$$
- **Lemma 1.7.1:** If $f(x) - \frac{\alpha}{2} ||x||^2$ is a convex function, then $f(x)$ is a α -strongly convex.

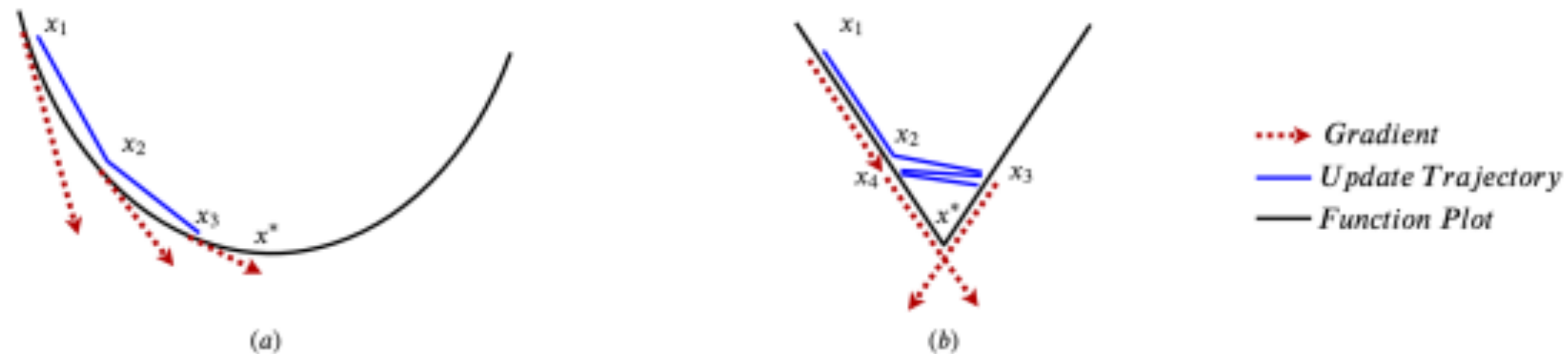
Proof:



1. Convex Optimization

1.8 β -smooth

- $f: \mathbb{K} \rightarrow \mathbb{R}$ is β -smooth function if $||\nabla f(x) - \nabla f(y)|| \leq \beta ||x - y||$. In other words, the gradient $\nabla f(x)$ is β -Lipschitz.
- **Intuition:** β is an **upper-bound measure of the curvature**. In other words, the gradient tends to decay as x gets closer to x^*



1. Convex Optimization

1.8 β -smooth

- $f: \mathbb{K} \rightarrow \mathbb{R}$ is β -smooth function if $||\nabla f(x) - \nabla f(y)|| \leq \beta ||x - y||$. In other words, the gradient $\nabla f(x)$ is β -Lipschitz.
- **Lemma 1.8.1:** $|f(y) - f(x) - \nabla f(x)^T(y - x)| \leq \frac{\beta}{2} ||y - x||^2$

Proof: (Fundamental of Calculus, Cauchy-Schwarz inequality, β -smooth function)

$$\begin{aligned} |f(y) - f(x) - \nabla f(x)^T(y - x)| &= \left| \int_0^1 \nabla f(x + t(y - x))^T(y - x) dt - \nabla f(x)^T(y - x) \right| \\ &\leq \int_0^1 ||\nabla f(x + t(y - x)) - \nabla f(x)|| \cdot ||y - x|| dt \\ &\leq \int_0^1 \beta t ||y - x||^2 dt \\ &= \frac{\beta}{2} ||y - x||^2 \end{aligned}$$

1. Convex Optimization

1.8 β -smooth

- $f: \mathbb{K} \rightarrow \mathbb{R}$ is β -smooth function if $||\nabla f(x) - \nabla f(y)|| \leq \beta ||x - y||$. In other words, the gradient $\nabla f(x)$ is β -Lipschitz.
- **Lemma 1.8.2:** $0 \leq f(y) \leq f(x) + \nabla f(x)^T(y - x) - \frac{1}{2\beta} ||\nabla f(y) - \nabla f(x)||^2$

Proof:

1. Convex Optimization

1.8 α -strongly convex and β -smooth properties

- Some useful properties:

- $\frac{\alpha}{2} ||x_t - x^*||^2 \leq f(x_t) - f(x^*)$

- $f(x_t) - f(x^*) \leq \frac{\beta}{2} ||x_t - x^*||^2$

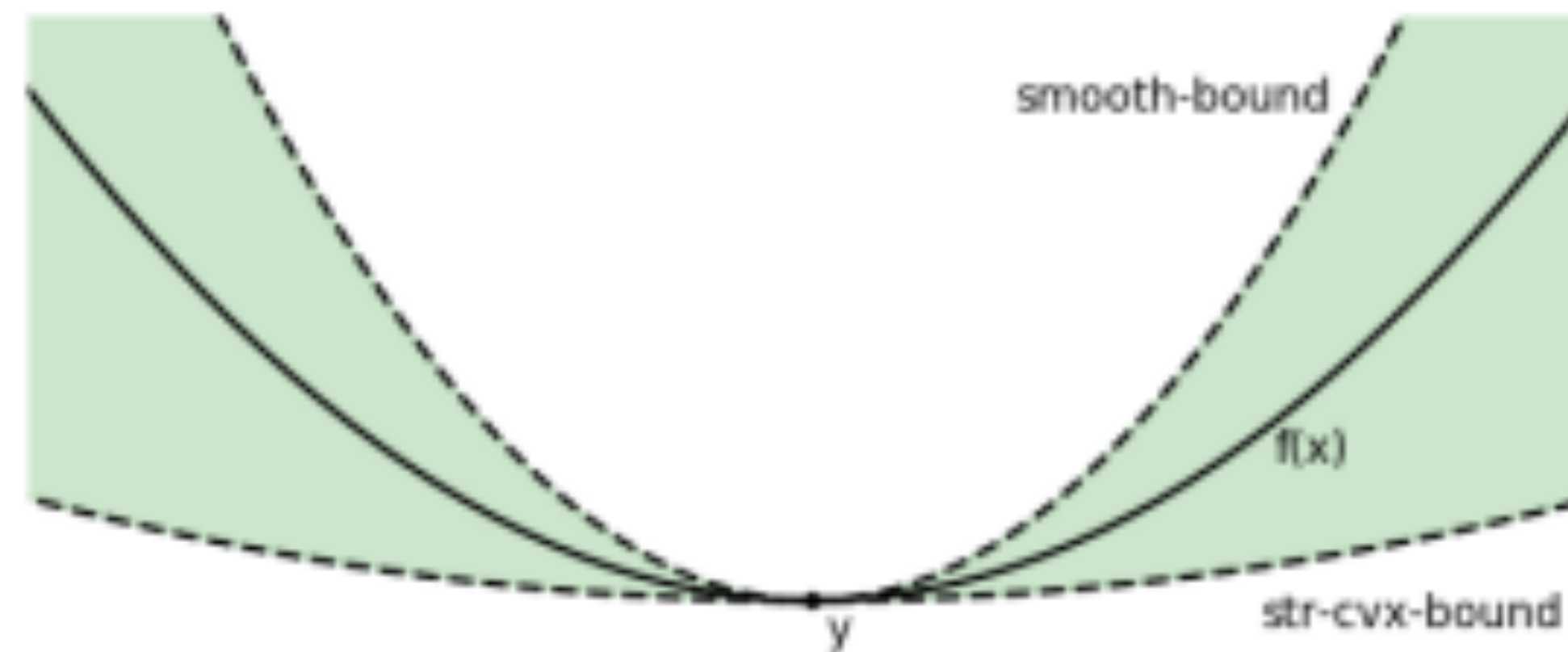
- $\frac{1}{2\beta} ||\nabla f(x_t)||^2 \leq f(x_t) - f(x^*)$

- $f(x_t) - f(x^*) \leq \frac{1}{2\alpha} ||\nabla f(x_t)||^2$

1. Convex Optimization

1.9 γ -well conditioned

- If f is both α -strongly convex and β -smooth, then f is γ -well-conditioned where $\gamma = \frac{\alpha}{\beta} \leq 1$
- **Corollary 1.9.1:** $\alpha I \preceq \nabla^2 f(x) \preceq \beta I$. In other words, eigenvalues of the Hessian is between α and β



1. Convex Optimization

1.9 γ -well conditioned

- If f is both α -strongly convex and β -smooth, then f is γ -well-conditioned where $\gamma = \frac{\alpha}{\beta} \leq 1$
- **Corollary 1.9.1:**
$$(\nabla f(x) - \nabla f(y))^T(x - y) \geq \frac{\alpha\beta}{\alpha + \beta} ||x - y||^2 + \frac{1}{\alpha + \beta} ||\nabla f(x) - \nabla f(y)||^2$$

1. Convex Optimization

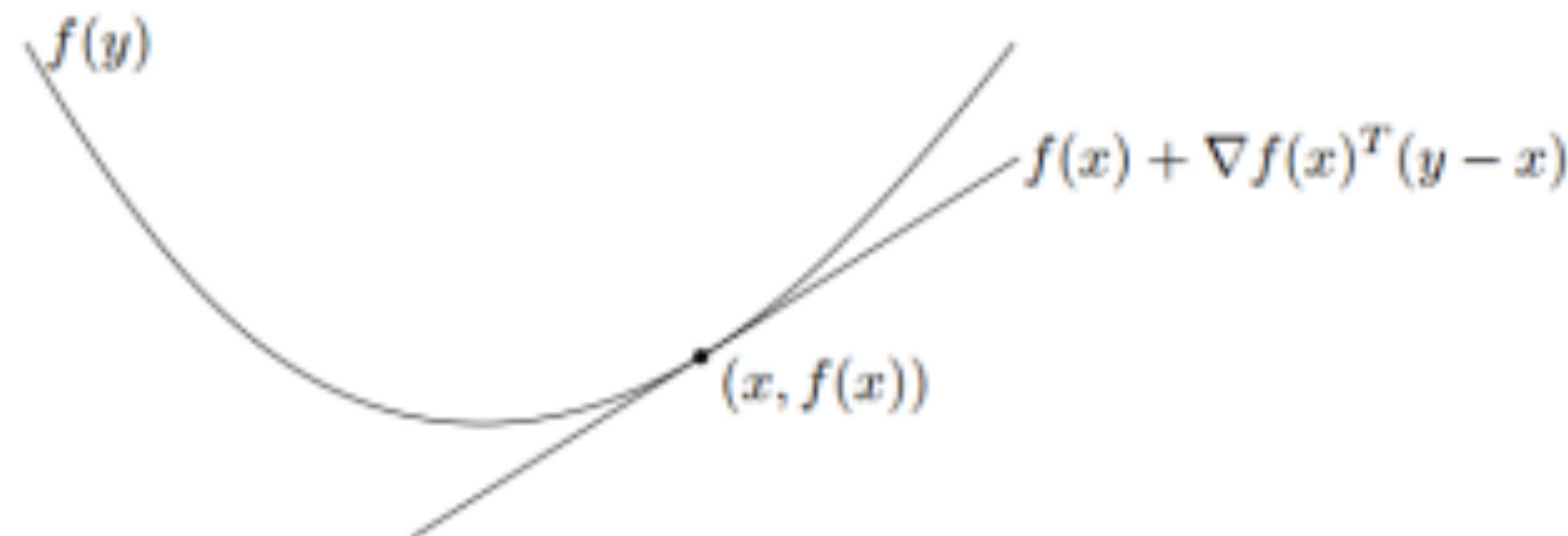
$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$

1.10 First-order condition

- **Theorem:** Suppose f is **differentiable** and \mathbb{K} is a **convex set**, the function f is **convex** if and only if $f(y) \geq f(x) + \nabla f(x)^T(y - x), \forall x, y \in \mathbb{K}$

First-order Taylor series approximation

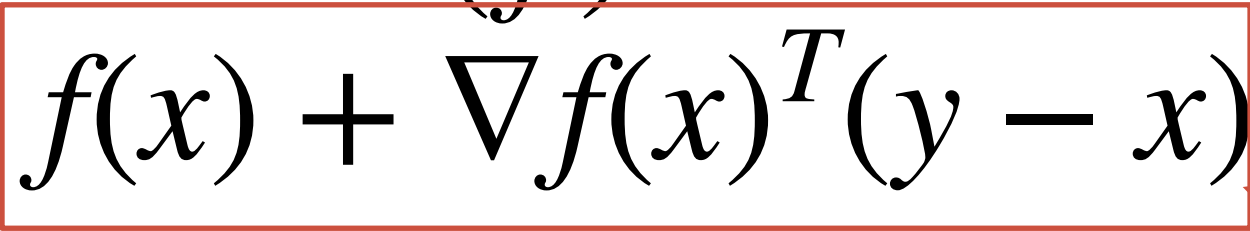
- **Intuition:** The first-order approximation of f **underestimates** f at every point in its domain. Then, the first-order approximation is a **global underestimator**.



1. Convex Optimization

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$

1.10 First-order condition

- **Theorem:** Suppose f is **differentiable** and $\text{dom}(f)$ is a **convex set**, the function f is **convex** if and only if $f(y) \geq f(x) + \nabla f(x)^T(y - x), \forall x, y \in \mathbb{K}$

First-order Taylor series approximation
- **Corollary 1.10.1:** If x^* is a **local minimum** s.t $\nabla f(x^*) = 0$ for a **convex function** f , then $f(y) \geq f(x^*), \forall y \in \mathbb{K}$. Therefore, x^* is a **global minimum**.

1. Convex Optimization

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$

1.10 First-order condition

- **Theorem:** Suppose f is **differentiable** and \mathbb{K} is a **convex set**, the function f is **convex** if and only if $f(y) \geq f(x) + \nabla f(x)^T(y - x), \forall x, y \in \mathbb{K}$

Proof: (\Rightarrow) Suppose f is a **convex function**, then

$$f(\alpha y + (1 - \alpha)x) \leq \alpha f(y) + (1 - \alpha)f(x)$$
$$\frac{f(x + \alpha(y - x)) - f(x)}{\alpha} \leq (f(y) - f(x))$$

$$\boxed{\nabla f(x)^T(y - x)} \leq f(y) - f(x)$$

Definition of directional derivative

As $\alpha \rightarrow 0$

$$\therefore f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

1. Convex Optimization

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$

1.10 First-order condition

- **Theorem:** Suppose f is **differentiable** and \mathbb{K} is a **convex set**, the function f is **convex** if and only if $f(y) \geq f(x) + \nabla f(x)^T(y - x), \forall x, y \in \mathbb{K}$

Proof: (\Leftarrow) Suppose $f(y) \geq f(x) + \nabla f(x)^T(y - x)$, then

1. Convex Optimization

1.10 First-order condition

- **First-order optimality condition:** Let f be a differentiable convex function, \mathbb{K} be a convex set. Then,
$$x^* \in \operatorname{argmin}_{x \in \mathbb{K}} f(x) \Leftrightarrow \nabla f(x^*)^T (y - x^*) \geq 0, \forall y \in \mathbb{K}$$

► **Proof:**

1. Convex Optimization

1.10 First-order condition

- **Subgradients:** Let $f : \mathbb{K} \rightarrow \mathbb{R}$, g is a **subgradient** of f at \mathbf{x} if $\forall y \in \mathbb{K}$, we have $f(x) - f(y) \leq g^T(x - y)$. The set of all subgradient of f at x is denoted as $\partial f(x)$
 - **Example:** $f(x)$ is non-differentiable at $x = 0$, $g \in \partial f(x) = [-1, 1]$

1. Convex Optimization

1.10 First-order condition

- **Subgradients:** Let $f: \mathbb{K} \rightarrow \mathbb{R}$, g is a **subgradient** of f at x if $\forall y \in \mathbb{K}$, we have $f(x) - f(y) \leq g^T(x - y)$. The set of all subgradient of f at x is denoted as $\partial f(x)$
- **Proposition 1:** Let \mathbb{K} be a convex set and $f: \mathbb{K} \rightarrow \mathbb{R}$.
 - 1. $\forall x \in \mathbb{K}$, if $\partial f(x) \neq \emptyset$, then f is convex.
 - 2. If f is convex and x is in the interior of \mathbb{K} , then $\partial f(x) \neq \emptyset$
- **Proposition 2:** If f is convex and differentiable at x , then $\nabla f(x) \in \partial f(x)$
 - **Remark:** Subgradient **generalizes** the notion of gradient for **non-differentiable** and (or) **non-smooth** function by replacing $\nabla f(x)^T$ with g^T
- **Proof:** Skip for now

1. Convex Optimization

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \forall i, j = 1, \dots, n$$

1.11 Second-order condition

- **Theorem:** Suppose f is **twice differentiable** and \mathbb{K} is a **convex set**, then the function f is **convex** if and only if $\nabla^2 f(x) \succeq 0, \forall x \in \mathbb{K}$ (Hessian is **positive semidefinite**)

▸ **Remark:** $\nabla^2 f(x) \succ 0$, then the function f is **strictly convex**.

▸ **Proof:** (\Rightarrow) Suppose f is a **convex function**, then

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} (y - x)^T \nabla^2 f(x) (y - x)$$

Second-order Taylor series

$$\Leftrightarrow \frac{1}{2} (y - x)^T \nabla^2 f(x) (y - x) \geq 0$$

First-order condition

$$\Leftrightarrow \nabla^2 f(x) \succeq 0$$

Def. of positive semidefinite matrix

1. Convex Optimization

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \forall i, j = 1, \dots, n$$

1.11 Second-order condition

- Suppose f is **twice differentiable** and \mathbb{K} is a **convex set**, then the function f is **convex** if and only if $\nabla^2 f(x) \succeq 0, \forall x \in \mathbb{K}$ (Hessian is **positive semidefinite**)
 - **Remark:** $\nabla^2 f(x) \succ 0$, then the function f is **strictly convex**.
 - **Proof:** (\Leftarrow) Suppose $\nabla^2 f(x) \succeq 0$, then
 -

1. Convex Optimization

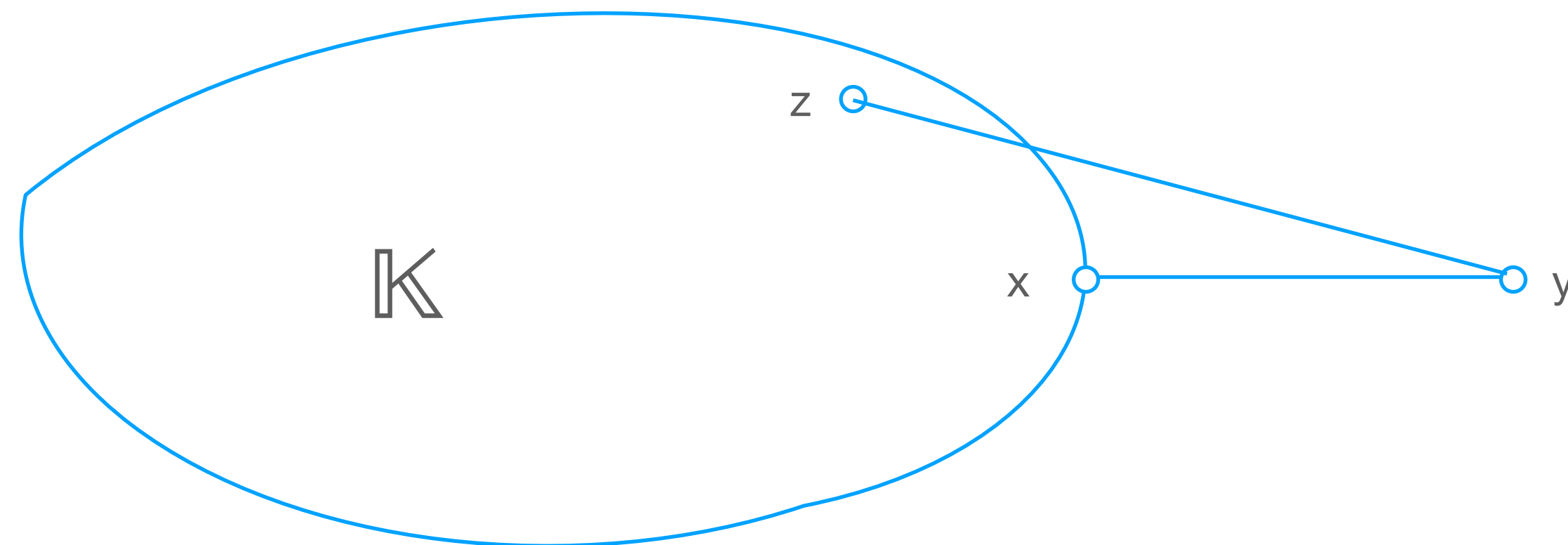
1.12 Projection over convex set (POCV)

- Basic updates of Gradient Descent (GD): $x_{t+1} = x_t - \eta_t \nabla f(x_t)$
 - What if $x_{t+1} \notin \mathbb{K}$? Projection to convex set
- **Projection over convex set (POCV):** $\Pi_{\mathbb{K}}(x_{k+1}) \doteq \arg \min_x ||x - x_{k+1}||$
 - **Remark:** if \mathbb{K} is a **closed, bounded and convex** set, then $\Pi_{\mathbb{K}}(x_{k+1})$ is **unique**.

1. Convex Optimization

1.13 Pythagoras theorem

- **“Pythagorean theorem”**, named in OCO book: Let $\mathbb{K} \subset \mathbb{R}^d$ be a convex set, $y \in \mathbb{R}^d$ and $x = \Pi_{\mathbb{K}}(y)$. Then for any $z \in \mathbb{K}$, $\|y - z\| \geq \|x - z\|$
 - **Intuition:** Think of $y \notin \mathbb{K}$, $x, z \in \mathbb{K}$, and x is the closest point of set \mathbb{K} to y .



1. Convex Optimization

1.13 Pythagoras theorem

- “**Pythagorean theorem**”, named in OCO book: Let $\mathbb{K} \subset \mathbb{R}^d$ be a convex set, $y \in \mathbb{R}^d$ and $x = \Pi_{\mathbb{K}}(y)$. Then for any $z \in \mathbb{K}$, $\|y - z\| \geq \|x - z\|$
- **Proof:**

1. Convex Optimization

1.14 Gradient Descent (GD) - Convergence rate

Algorithm 2 Gradient Descent

1: Input: time horizon T , initial point x_0 , step sizes $\{\eta_t\}$

2: **for** $t = 0, \dots, T - 1$ **do**

3: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$

4: **end for**

5: **return** $\bar{\mathbf{x}} = \arg \min_{\mathbf{x}_t} \{f(\mathbf{x}_t)\}$

Gradient Descent Algorithm

	general	α -strongly convex	β -smooth	γ -well conditioned
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$	$e^{-\gamma T}$
Accelerated GD	—	—	$\frac{\beta}{T^2}$	$e^{-\sqrt{\gamma} T}$

Convergence rate under different conditions (T - #iterations)

Remark: We omitted some quantities in the final convergence.

1. Convex Optimization

1.14 Gradient Descent (GD) - Convergence rate

	general	α -strongly convex	β -smooth	γ -well conditioned
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$	$e^{-\gamma T}$

Convergence rate under different conditions (T - #iterations)

- **First case:** L -Lipschitz continuous differentiable convex function. \sqrt{T} is quite **slow** compared to other conditions.
- **Second case:** Recall that α is a **lower-bound measure of curvature**. High α means fast convergence. In other words, higher α means smaller $\frac{1}{\alpha T}$ or faster convergence. Note that this is T , not \sqrt{T} .
- **Third case:** Recall that β is an **upper-bound measure of curvature**. In other words, β prevents the function from **changing too fast**, which is consistent with the convergence rate.
- **Fourth case:** Recall that $\gamma = \frac{\alpha}{\beta}$, which is the ratio between the lower-bound and upper-bound curvature. We want α to be large and β to be small so that γ is large, which is consistent with second and third case.
- **Remark:** Function with γ -well-conditioned has the **strongest assumption**, but also has the **fastest convergence**.

1. Convex Optimization

1.14 Gradient Descent (GD) with Polyak step-size

Algorithm 2 Gradient Descent

```
1: Input: time horizon  $T$ , initial point  $x_0$ , step sizes  $\{\eta_t\}$ 
2: for  $t = 0, \dots, T - 1$  do
3:    $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$ 
4: end for
5: return  $\bar{\mathbf{x}} = \arg \min_{\mathbf{x}_t} \{f(\mathbf{x}_t)\}$ 
```

What is the optimal choice for step-size?

- **Polyak step-size:** Optimal choice of step-size in gradient descent method.

1. Convex Optimization

1.14 Gradient Descent (GD) with Polyak step-size - Algorithm

Algorithm 3 Gradient Descent with Polyak stepsize

```
1: Input: time horizon  $T$ ,  $x_0$ 
2: for  $t = 0, \dots, T - 1$  do
3:   Set  $\eta_t = \frac{h_t}{\|\nabla_t\|^2}$ 
4:    $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$ 
5: end for
6: Return  $\bar{\mathbf{x}} = \arg \min_{\mathbf{x}_t} \{f(\mathbf{x}_t)\}$ 
```

- $h_t = h(x_t) = f(x_t) - f(x^*)$: Distance between current value and optimal value
- $\|\nabla_t\|^2 = \|\nabla f(x_t)\|^2$: Squared norm of Gradient of f at x_t

1. Convex Optimization

1.14 Gradient Descent (GD) with Polyak step-size - Convergence analysis

- $h_t = h(x_t) = f(x_t) - f(x^*)$: Distance between current value and optimal value
- $||\nabla_t||^2 = ||\nabla f(x_t)||^2$: Squared norm of Gradient of f at x_t

Algorithm 3 Gradient Descent with Polyak stepsize

```
1: Input: time horizon  $T, x_0$ 
2: for  $t = 0, \dots, T - 1$  do
3:   Set  $\eta_t = \frac{h_t}{||\nabla_t||^2}$ 
4:    $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla_t$ 
5: end for
6: Return  $\bar{\mathbf{x}} = \arg \min_{\mathbf{x}_t} \{f(\mathbf{x}_t)\}$ 
```

- **Theorem:** Suppose $||\nabla f(x_t)|| \leq G$ (function f is G -Lipschitz), $B_T = \min\{\frac{Gd_0}{\sqrt{T}}, \frac{2\beta d_0^2}{T}, \frac{3G^2}{\alpha T}, \beta d_0^2(1 - \frac{\gamma}{4})^T\}$ and a sequence x_0, \dots, x_t satisfies $||x_{t+1} - x^*||^2 \leq ||x_t - x^*||^2 - \frac{f(x_t) - f(x^*)}{||\nabla f(x_t)||^2}$. Then, the algorithm 3 is guaranteed to converge after T steps:
 - $f(\bar{x}) - f(x^*) \leq \min_{0 \leq t \leq T} \{f(x_t) - f(x^*)\} \leq B_T$

1. Convex Optimization

1.14 Gradient Descent (GD) with Polyak step-size - Compare the convergence with GD

	general	α -strongly convex	β -smooth	γ -well conditioned
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$	$e^{-\gamma T}$

If T is **large** (many iterations), then $(1 - \frac{\gamma}{4})^T$ and whole expression will be **extremely small**. Also, notice that B_T is the minimum

- Theorem:** Suppose $\|\nabla f(x_t)\| \leq G$ (function f is G -Lipschitz), $B_T = \min\{\frac{Gd_0}{\sqrt{T}}, \frac{2\beta d_0^2}{T}, \frac{3G^2}{\alpha T}, \beta d_0^2(1 - \frac{\gamma}{4})^T\}$ and a sequence x_0, \dots, x_t satisfies $\|x_{t+1} - x^*\|^2 \leq \|x_t - x^*\|^2 - \frac{f(x_t) - f(x^*)}{\|\nabla f(x_t)\|^2}$. Then, the algorithm 3 is guaranteed to converge after T steps: $f(\bar{x}) - f(x^*) \leq \min_{0 \leq t \leq T} \{f(x_t) - f(x^*)\} \leq B_T$

► **Remark:** The convergence rate **no longer depends** on the property α -strongly convex, β -smooth or γ -well-conditioned function. However, we need to **know** x^* and d_0 , which is usually **not feasible**.

1. Convex Optimization

1.14 Constrained GD - Algorithm

Algorithm 4 Basic gradient descent

1: Input: f , T , initial point $\mathbf{x}_1 \in \mathcal{K}$, sequence of step sizes $\{\eta_t\}$

2: **for** $t = 1$ to T **do**

3: Let $\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)$, $\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$ ← **POCS**

4: **end for**

5: **return** \mathbf{x}_{T+1}

- **Motivation:** What if \mathbf{y}_{t+1} is **outside** the constrained set? **POCS!**

1. Convex Optimization

1.14 Constrained GD - Convergence analysis

- $h_t = h(x_t) = f(x_t) - f(x^*)$: Distance between current value and optimal value
- $||\nabla_t||^2 = ||\nabla f(x_t)||^2$: Squared norm of Gradient of f at x_t

Algorithm 4 Basic gradient descent

```
1: Input:  $f$ ,  $T$ , initial point  $\mathbf{x}_1 \in \mathcal{K}$ , sequence of step sizes  $\{\eta_t\}$ 
2: for  $t = 1$  to  $T$  do
3:   Let  $\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)$ ,  $\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$ 
4: end for
5: return  $\mathbf{x}_{T+1}$ 
```

← POCS

- **Theorem:** For γ -well-conditioned functions and $\eta_t = \frac{1}{\beta}$, the algorithm 4 converges as $h_{t+1} \leq h_t \cdot e^{-\frac{\gamma t}{4}}$

Similar to GD in the case of γ -well-conditioned function, but slower ($\frac{1}{4}$)

1. Convex Optimization

1.14 Reductions to smooth and non-strongly convex

- **Motivation:** Extends previous projected GD method to non- γ -well-conditioned function.
- **Theorem:** For β -smooth convex function, if $\tilde{\alpha} = \frac{\beta \log t}{D^2 t}$, then $h_{t+1} = O(\frac{\beta \log t}{t})$
- ▶ **Remark:** The above convergence is **suboptimal** by a factor of $\log T$ due to controlled amount of strong convexity

Algorithm 5 Gradient descent, reduction to β -smooth functions

1: Input: $f, T, \mathbf{x}_1 \in \mathcal{K}$, parameter $\tilde{\alpha}$.

2: Let $g(\mathbf{x}) = f(\mathbf{x}) + \frac{\tilde{\alpha}}{2} \|\mathbf{x} - \mathbf{x}_1\|^2$

← Added strong convexity

3: Apply Algorithm 4 with parameters $g, T, \{\eta_t = \frac{1}{\beta}\}, \mathbf{x}_1$, return \mathbf{x}_T .

1. Convex Optimization

1.14 Reductions to non-smooth and strongly convex

Algorithm 6 Gradient descent, reduction to non-smooth functions

1: Input: $f, \mathbf{x}_1, T, \delta$

2: Let $\hat{f}_\delta(\mathbf{x}) = \mathbf{E}_{\mathbf{v} \sim \mathbb{B}} [f(\mathbf{x} + \delta \mathbf{v})]$ ← Smooth approximation of f

3: Apply Algorithm 4 on $\hat{f}_\delta, \mathbf{x}_1, T, \{\eta_t = \delta\}$, return \mathbf{x}_T

- f is G -Lipchitz continuous and α -strongly convex function, $\delta > 0$,
 $\mathbb{B} = \{x \in \mathbb{R}^n : ||x|| \leq 1\}$ (Euclidean ball), $v \sim \mathbb{B}$ (sample from uniform distribution over \mathbb{B})
- **Corollary:**
 - \hat{f}_δ is α -strongly convex
 - \hat{f}_δ is $\frac{nG}{\delta}$ -smooth
 - $|\hat{f}_\delta(x) - f(x)| \leq \delta G, \forall x \in \mathbb{K}$

1. Convex Optimization

1.14 Reductions to non-smooth and strongly convex

Algorithm 6 Gradient descent, reduction to non-smooth functions

1: Input: $f, \mathbf{x}_1, T, \delta$

2: Let $\hat{f}_\delta(\mathbf{x}) = \mathbf{E}_{\mathbf{v} \sim \mathbb{B}} [f(\mathbf{x} + \delta \mathbf{v})]$ ← Smooth approximation of f

3: Apply Algorithm 4 on $\hat{f}_\delta, \mathbf{x}_1, T, \{\eta_t = \delta\}$, return \mathbf{x}_T

• **Theorem:** For $\delta = \frac{nG \log t}{\alpha t}$, the algorithm converges as $h_t = O(\frac{G^2 n \log t}{\alpha t})$

▸ **Remark:** The convergence rate depends on n , the dimension of the ball we are sampling from.

1. Convex Optimization

Summary

- 1.1
- 1.2
- 1.3
- 1.4
- 1.5
- 1.6
- 1.7
- 1.8
- 1.9
- 1.10
- 1.11
- 1.12
- 1.13
- 1.14


2. Online Convex Optimization

2.1 Motivation

- Now, we focus on the “Online” part!
- **Statistical learning (Supervised learning):**
 - Training-Testing phase
 - Independent and Identically distributed assumption (i.i.d)
- **Online learning:** Does not exist such separation
 - **Data efficiency:** Receive an instance x_t at time t and make prediction \hat{y}_t as it arrives. In other words, you will see instances **one-by-one**, instead of seeing all instances \mathbf{x} .
 - **No i.i.d assumption:** More practical since preferences are changing (recommended system, spam filters, etc.)
 - **Remark:** Online learning is a **generalization** version of statistical learning.

2. Online Convex Optimization

2.2 Formulation

- **Game iteration:** $t = 1, \dots, T$
- **Decision set:** $\mathbb{K} \subseteq \mathbb{R}^n$ with assumption to be a bounded convex set.
- **Loss function:** $f \in \mathbb{F} : \mathbb{K} \rightarrow \mathbb{R}$, f belongs to a bounded family of cost function. We do not want the loss to be infinite
- **Algorithm:** $\mathbb{A} : \{f_1, \dots, f_{t-1}\} \rightarrow \mathbb{K}$
- **Decision action at time t:** $x_t \in \mathbb{K}$
- **Decision action at time t given by Algorithm \mathbb{A} :** $x_t^{\mathbb{A}} = \mathbb{A}(f_1, \dots, f_{t-1}) \in \mathbb{K}$
- **Regret:** $Regret_T(\mathbb{A}) = \sum_{t=1}^T f_t(x_t^{\mathbb{A}}) - \min_{x \in \mathbb{K}} \sum_{t=1}^T f_t(x)$
 $x^* - \text{Consistent}$
- ▶ **Remark:** Adversarial environment - f_1, \dots, f_T are arbitrary (not follow any distributions) but decided before starting. (Formal definition is in the reference)
- ▶ **Remark:** Full-information feedback model - Observes f_t for all decision in decision set at time t.

2. Online Convex Optimization

2.3 Goal

- **Objective:** $\min_{x_1, \dots, x_T} \text{Regret}_T(\mathbb{A})$
 - **Intuitive:** how much we “regret” for not choosing best (fixed) action in hindsight.
- **No-regret:** If $\text{Regret}_T(\mathbb{A})$ is **sublinear** in T ($\lim_{T \rightarrow \infty} \frac{\text{Regret}_T(\mathbb{A})}{T} = 0$), then on average the learner is doing almost as well as the best (fixed) action in hindsight.

2. Online Convex Optimization

2.4 Example - Prediction from expert advice

- **Simplified version:** $\mathbb{K} = \{\text{up}, \text{down}\}$, $N = 3$ Experts, $T = 5$, cost of making mistakes is 1 and cost of making right decision is 0.
 - **Remark:** Your choice is the majority of the experts' choice. At each time, you observe the cost (The last column is cumulative). Note that you make 3 mistakes, while the best expert (expert 2) only makes 1.

Expert 1	Expert 2	Expert 3	My choice	Truth	Cost
Down	Down	Up	Down	Down	0
Up	Down	Up	Up	Down	1
Down	Up	Down	Down	Up	2
Down	Down	Up	Down	Down	2
Down	Down	Up	Down	Up	3

2. Online Convex Optimization

2.5 Example - Prediction from expert advice - Halving strategy

- **Simplified version:** $\mathbb{K} = \{\text{up}, \text{down}\}$, $N = 3$ Experts, $T = 5$, cost of making mistakes is 1 and cost of making right decision is 0. Instead, we assume that the **best expert makes no mistake**. Also, we will keep a **set of experts whose decisions are correct** so far and only consider the action from these experts.
- **Theorem:** This strategy has regret no more than $\log_2 N$. In other words, the learner makes at most $\log_2 N$ mistakes.
 - **Proof:** in the reference
 - **Intuition:** When the learner makes the mistakes, the set of perfect expert decreases by at least $1/2$.

2. Online Convex Optimization

2.6 Example - Prediction from expert advice - Weighted majority strategy

- **Motivation:** What if the best expert also makes mistake?

Weighted Majority Algorithm

Initialization: Fix $\delta \leq 1/2$. For every $i \in [n]$, let $w_i^{(1)} := 1$

Update: For $t = 1, 2, \dots, T$:

- Make prediction which is the weighted majority of the experts' predictions
- For every expert i who predicts wrongly, decrease his weight by a factor of $(1 - \delta)$:

$$w_i^{(t+1)} = (1 - \delta)w_i^{(t)}$$

2. Online Convex Optimization

2.6 Example - Prediction from expert advice - Weighted majority strategy

Let $\delta = 1/2$, $n = 3$



t	Expert Weights	Expert Predictions	Our Pred.	Result	Our Errors
1	1, 1, 1	1, 1, 0	1 ✓	1	0
2	1, 1, 1/2	0, 1, 0	0 ✗	1	1
3	1/2, 1, 1/4	1, 0, 1	0 ✓	0	1
4	1/4, 1, 1/8	0, 1, 1	1 ✗	0	2
5	1/4, 1/2, 1/16	1, 1, 0	1 ✓	1	2

2. Online Convex Optimization

2.6 Example - Prediction from expert advice - Weighted majority strategy analysis

- **Theorem:** Denote M_t the number of mistakes the algorithm makes until the time t , $M_t(i)$ the number of mistakes made by expert i until time t . Then, for any expert $i \in [N]$, we have $M_T \leq 2(1 + \delta)M_T(i) + \frac{2 \log N}{\delta}$
- **Proof:**
- **Intuition:** Note that the best expert is making mistakes! Besides that we also have $\log N$ in the convergence rate.

2. Online Convex Optimization

2.6 Example - Prediction from expert advice - Weighted majority strategy analysis

- **Theorem:** Let M denote the number of mistakes made by best expert. Then, there are no deterministic algorithm that can guarantee less than $2M$ mistakes
 - **Proof:** Two experts: one always “Up” and other always “Down”. For deterministic prediction algorithm, if you know the history, you know what is the prediction. Therefore, the adversary can set the real outcome to be opposite with the prediction. So the algorithm makes 0 corrects, while one of two experts makes at least 50% corrects.

2. Online Convex Optimization

2.7 Example - Prediction from expert advice - Randomized weighted majority strategy

- **Motivation:** The factor 2 in the convergence cannot be improved by deterministic algorithm.

Randomised Weighted Majority Algorithm

Initialization: Fix $\delta \leq 1/2$. For every $i \in [n]$, let $w_i^{(1)} := 1$

Update: For $t = 1, 2, \dots, T$:

- Pick expert i with probability proportional to w_i and follow that prediction
- For every expert i who predicts wrongly, decrease his weight by a factor of $(1 - \delta)$:

$$w_i^{(t+1)} = (1 - \delta)w_i^{(t)}$$

2. Online Convex Optimization

2.7 Example - Prediction from expert advice - Randomized weighted majority strategy example

• **Example:**

t	Weights	Predictions	Actual Result	Our Prediction	Our Errors
1	1,1	1,0	0	1	1
2	1/2,1	1,0	1	0	2
3	1/2,1/2	0,1	1	0	3
4	1/4,1/2	1,0	1	0	4
5	1/4,1/4	—	—	—	—

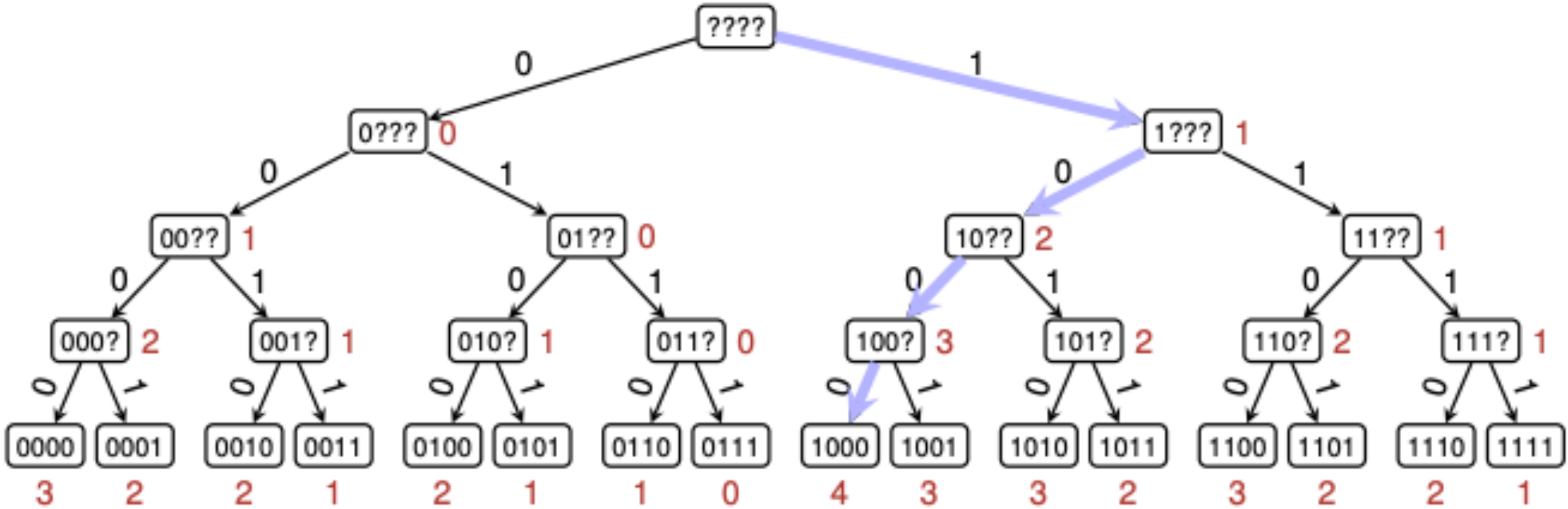
Total #mistakes

Probability of selecting advice by expert i = $\frac{\text{Weight of expert i}}{\text{Total weight of all experts}}$

$x^{(t)}$: 0/1 random variable indicating if our t prediction is wrong

$$E[x^{(1)}] = \frac{1}{2}, E[x^{(2)}] = \frac{2}{3}, E[x^{(3)}] = \frac{1}{2}, E[x^{(3)}] = \frac{2}{3}$$

Expected number of mistakes at time T = 4 is $\frac{7}{3}$. Much better!



2. Online Convex Optimization

2.7 Example - Prediction from expert advice - Randomized weighted majority strategy analysis

- **Theorem:** Denote M_t the number of mistakes the algorithm makes until the time t , $M_t(i)$ the number of mistakes made by expert i until time t . Then, for any expert $i \in [N]$, we have $E[M_T] \leq (1 + \delta)M_T(i) + \frac{\log N}{\delta}$
- **Remark:** The (expected) number of mistakes is less!
- **Remark:** We can still improve it! If you are interested, you can read about Multiplicative weights algorithms, which generalizes the setting and deals with poor experts.

3. First-order solution

3.1 Online Gradient Descent - Algorithm

Algorithm 8 online gradient descent

- 1: Input: convex set \mathcal{K} , T , $\mathbf{x}_1 \in \mathcal{K}$, step sizes $\{\eta_t\}$
- 2: **for** $t = 1$ to T **do**
- 3: Play \mathbf{x}_t and observe cost $f_t(\mathbf{x}_t)$.
- 4: Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$$

5: **end for**

Algorithm 4 Basic gradient descent

- 1: Input: f , T , initial point $\mathbf{x}_1 \in \mathcal{K}$, sequence of step sizes $\{\eta_t\}$
 - 2: **for** $t = 1$ to T **do**
 - 3: Let $\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)$, $\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$
 - 4: **end for**
 - 5: **return** \mathbf{x}_{T+1}
-

Negative gradient of **Previous cost**

3. First-order solution

3.2 Online Gradient Descent - Analysis

$$\forall x, y \in \mathbb{K}, ||x - y|| \leq D$$
$$G > 0 : \forall x \in \mathbb{K}, ||\nabla f(x)|| \leq G$$

Algorithm 8 online gradient descent

- 1: Input: convex set \mathcal{K} , T , $\mathbf{x}_1 \in \mathcal{K}$, step sizes $\{\eta_t\}$
- 2: **for** $t = 1$ to T **do**
- 3: Play \mathbf{x}_t and observe cost $f_t(\mathbf{x}_t)$.
- 4: Update and project:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}(\mathbf{y}_{t+1})$$

5: **end for**

• **Theorem:** OGD with stepsize $\eta_t = \frac{G}{D\sqrt{t}}$ guarantees to have the following regret for all $T \geq 1$, $\text{regret}_T(\mathbb{A}) \leq \frac{3}{2}GD\sqrt{T}$.

▸ **Proof:** Skip for now

▸ **Remark:** The regret is bounded in $O(\sqrt{T})$, which is sublinear in T . Can we do better than this?

3. First-order solution


$$\begin{aligned} &\forall x, y \in \mathbb{K}, ||x - y|| \leq D \\ &G > 0 : \forall x \in \mathbb{K}, ||\nabla f(x)|| \leq G \end{aligned}$$

3.3 Lower bound - Logarithmic regret

- **Motivation - Theorem:** Any algorithms for OCO incurs $\Omega(DG\sqrt{T})$ regret in the worst case. This is true even if the cost functions are generated from a fixed stationary distribution.
 - ▶ **Proof:** Skip the proof
- **Theorem:** For α -strongly convex function, OGD with step sizes $\eta_t = \frac{1}{\alpha t}$ achieves the following guarantee for all $T \geq 1$ $regret_T(\mathbb{A}) \leq \frac{G^2}{2\alpha}(1 + \log T)$
 - ▶ **Proof:** Skip the proof

2. Online Convex Optimization

2.2 Formulation - Recap

- **Game iteration:** $t = 1, \dots, T$
- **Decision set:** $\mathbb{K} \subseteq \mathbb{R}^n$ with assumption to be a bounded convex set.
- **Loss function:** $f \in \mathbb{F} : \mathbb{K} \rightarrow \mathbb{R}$, f belongs to a bounded family of cost function. We do not want the loss to be infinite
- **Algorithm:** $\mathbb{A} : \{f_1, \dots, f_{t-1}\} \rightarrow \mathbb{K}$
- **Decision action at time t:** $x_t \in \mathbb{K}$
- **Decision action at time t given by Algorithm \mathbb{A} :** $x_t^{\mathbb{A}} = \mathbb{A}(f_1, \dots, f_{t-1}) \in \mathbb{K}$
- **Regret:** $Regret_T(\mathbb{A}) = \sum_{t=1}^T f_t(x_t^{\mathbb{A}}) - \min_{x \in \mathbb{K}} \sum_{t=1}^T f_t(x)$
 $x^* - \text{Consistent}$
- ▶ **Remark:** Adversarial environment - f_1, \dots, f_T are arbitrary (not follow any distributions) but decided before starting. (Formal definition is in the reference)
- ▶ **Remark:** Full-information feedback model - Observes f_t for all decision in decision set at time t.

3. First-order solution

$$\forall x, y \in \mathbb{K}, ||x - y|| \leq D$$

$$G > 0 : \forall x \in \mathbb{K}, ||\nabla f(x)|| \leq G$$

3.3 Lower bound - Logarithmic regret - Recap

- **Remark:** β -smooth function does not improve the regret bound

	α -strongly convex	β -smooth
Upper bound	$\frac{1}{\alpha} \log T$	\sqrt{T}
Lower bound	$\frac{1}{\alpha} \log T$	\sqrt{T}

Recall that regret bound for general case in the online setting is $O(\sqrt{T})$

Online setting

	general	α -strongly convex	β -smooth
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$

Offline setting

4. Second-order solution

4.1 Motivation

- What if the function is not α -strongly convex, can we get $O(\log T)$ regret?

4. Second-order solution

4.2 Example - Universal Portfolio Selection

- **Distributions over n assets:** $x_t \in \Delta_n : \{x \in \mathbb{R}_+^n, \sum_i x_i = 1\}$
- **A price ratio of assets between time t and t + 1 chosen by adversary:** $r_t \in \mathbb{R}_+^n$:
- **Total wealth at time t:** $W_t \in \mathbb{R}$
 - **Update rule:** $W_{t+1} = W_t \cdot r_t^T x_t$
 - **Assumption:** No transaction costs
- **Total wealth of investor:** $W_T = W_1 \cdot \prod_{t=1}^T r_t^T x_t$

4. Second-order solution

4.2 Example - Universal Portfolio Selection

- **Distributions over n assets:** $x_t \in \Delta_n : \{x \in \mathbb{R}_+^n, \sum_i x_i = 1\}$
- **A price ratio of assets between time t and t + 1 chosen by adversary:** $r_t \in \mathbb{R}_+^n$:
- **Total wealth at time t:** $W_t \in \mathbb{R}$
 - **Update rule:** $W_{t+1} = W_t \cdot r_t^T x_t$
 - **Assumption:** No transaction costs
- **Total wealth of investor:** $W_T = W_1 \cdot \prod_{t=1}^T r_t^T x_t$

4. Second-order solution

4.2 Example - Universal Portfolio Selection

- **Example:** 2 assets (stocks), $W_0 = \$100$
- $t = 1$, $x_1 = [0.5, 0.5]$, $r_1 = [1.5, 1]$, $W_1 = 100 \times 1.25 = \125
- $t = 2$, $x_2 = [0.7, 0.3]$, $r_2 = [0.4, 1]$, $W_2 = 125 \times 0.58 = \72.5

4. Second-order solution

4.2 Example - Universal Portfolio Selection

- **Goal:** $\max_{x_1, \dots, x_T} \frac{W_T}{W_0} = \max_{x_1, \dots, x_T} \log \frac{W_T}{W_0} = \max_{x_1, \dots, x_T} \sum_{t=1}^T \boxed{\log r_t^T x_t} = \max_{x_1, \dots, x_T} \sum_{t=1}^T \boxed{f_t(x_t)}$
Concave function
- **Regret:** $Regret_T = \max_{x^* \in \mathbb{K} = \Delta_n} \sum_{t=1}^T f_t(x^*) - \sum_{t=1}^T \boxed{f_t(x_t)}$
Convex, but not strongly convex
- **Remark:** We still want to minimize the regret!

4. Second-order solution

4.3 Exponentially concave function

First-order condition: $\forall x, y \in \mathbb{K}, f(y) \geq f(x) + \nabla f(x)^T(y - x)$

α -strongly convex: $\forall x, y \in \mathbb{K}, f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\alpha}{2} \|y - x\|^2$

α -strongly convex: $\forall x, y \in \mathbb{K}, f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\alpha}{2} \|y - x\|^2$

- **Definition:** A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined to be γ -exp-concave over $\mathbb{K} \subseteq \mathbb{R}^n$ if there exists concave function $g : \mathbb{K} \rightarrow \mathbb{R}$ and g has the form $g(x) = e^{-\gamma f(x)}$
 - **Example:** $f(x) = -y \log x - (1 - y) \log(1 - x)$ with y is the ground-truth. If $y = 1$, then $g(x) = e^{-\gamma f(x)} = e^{\log x} = x$, which is concave.
- **Theorem:** For twice differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is γ -exp-concave at x if and only if $\nabla^2 f(x) \succeq \gamma \nabla f(x) \nabla f(x)^T$
 - **Proof:** $g'(x) = -\gamma f'(x) e^{-\gamma f(x)}, g''(x) = -\gamma f''(x) e^{-\gamma f(x)} + \gamma^2 f'(x)^2 e^{-\gamma f(x)} \leq 0 \Rightarrow f''(x) \geq \gamma f'(x)^2$ (1d case)
- **Theorem:** For $f : \mathbb{K} \rightarrow \mathbb{R}$ be an γ -exp-concave function and D, G denote the diameter of \mathbb{K} and a bound on the subgradients of f . $\forall x, y \in \mathbb{K}, \forall \eta \leq \frac{1}{2} \min\{\frac{1}{GD}, \gamma\}$, we have

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\eta}{2} (y - x)^T \nabla f(x) \nabla f(x)^T (y - x)$$
- **Remark:** General case of α -strongly convex function ($\gamma \leq \frac{f''(x)}{f'(x)^2} = \frac{\alpha}{G^2}$) \rightarrow Weaker assumption

4. Second-order solution

4.4 Example of exp-concave function -Exponentially weighted OCO

Algorithm 11 Exponentially Weighted Online Optimizer

1: Input: convex set \mathcal{K} , T , parameter $\alpha > 0$.

2: **for** $t = 1$ to T **do**

3: Let $w_t(\mathbf{x}) = e^{-\alpha \sum_{\tau=1}^{t-1} f_{\tau}(\mathbf{x})}$.

4: Play \mathbf{x}_t given by

$$\mathbf{x}_t = \frac{\int_{\mathcal{K}} \mathbf{x} w_t(\mathbf{x}) d\mathbf{x}}{\int_{\mathcal{K}} w_t(\mathbf{x}) d\mathbf{x}}.$$

Not trivial to implement

5: **end for**

▸ **Remark:** $f_t(x)$ is exp-concave function, but very similar to previous algorithm Weighted majority algorithm.

• **Theorem:** $\text{Regret}_T \leq \frac{d}{\alpha} \log T + \frac{2}{\alpha}$ (Independence of G and D)

• **Proof:**

4. Second-order solution

4.5 Online Newton step algorithm

Algorithm 12 online Newton step

- 1: Input: convex set \mathcal{K} , T , $\mathbf{x}_1 \in \mathcal{K} \subseteq \mathbb{R}^n$, parameters $\gamma, \varepsilon > 0$, $A_0 = \varepsilon \mathbf{I}_n$
- 2: **for** $t = 1$ to T **do**
- 3: Play \mathbf{x}_t and observe cost $f_t(\mathbf{x}_t)$.
- 4: Rank-1 update: $A_t = A_{t-1} + \nabla_t \nabla_t^\top$ Approximation of Hessian matrix
- 5: Newton step and generalized projection:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}^{A_t}(\mathbf{y}_{t+1}) = \arg \min_{\mathbf{x} \in \mathcal{K}} \{ \|\mathbf{y}_{t+1} - \mathbf{x}\|_{A_t}^2 \}$$

6: **end for**

- **Remark:** The projection is based on the norm defined by matrix A_t rather than Euclidean norm.

4. Second-order solution

4.5 Online Newton step algorithm

Algorithm 12 online Newton step

- 1: Input: convex set \mathcal{K} , T , $\mathbf{x}_1 \in \mathcal{K} \subseteq \mathbb{R}^n$, parameters $\gamma, \varepsilon > 0$, $A_0 = \varepsilon \mathbf{I}_n$
- 2: **for** $t = 1$ to T **do**
- 3: Play \mathbf{x}_t and observe cost $f_t(\mathbf{x}_t)$.
- 4: Rank-1 update: $A_t = A_{t-1} + \nabla_t \nabla_t^\top$ Approximation of Hessian matrix
- 5: Newton step and generalized projection:

$$\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$$

$$\mathbf{x}_{t+1} = \Pi_{\mathcal{K}}^{A_t}(\mathbf{y}_{t+1}) = \arg \min_{\mathbf{x} \in \mathcal{K}} \{ \|\mathbf{y}_{t+1} - \mathbf{x}\|_{A_t}^2 \}$$

6: **end for**


- **Theorem:** With f_t is α -exp-concave, $\gamma = \frac{1}{2} \min\{\frac{1}{GD}, \alpha\}$, $\varepsilon = \frac{1}{\gamma^2 D^2}$, and $T \geq 4$, we have $\text{Regret}_T \leq 2(\frac{1}{\alpha} + GD)n \log T$
- **Proof:**

4. Second-order solution

4.6 First-order method versus Second-order method

- Example: OGD
- General case: $O(\sqrt{T})$
- α -strongly convex case: $O(\log T)$
- β -smooth convex case: $O(\sqrt{T})$
- Example: ONS
- Exponentially concave case: $O(\log T)$
- Weaker assumption, faster convergence rate, but very LARGE runtime

Algorithm	Regret bound	Running time
OGD	$O(\frac{G^2}{H} \log T)$	$\tilde{O}(n) + T_{\text{proj}}$
ONS	$O((\frac{1}{\alpha} + GD)n \log T)$	$\tilde{O}(n^2) + T_{\text{proj}}^g$

 Due to matrix A

4. Second-order solution

4.7 Recent work

- Reduces computational expense, but maintains $\log(T)$ regret.
- Some works have been done, but it seems that it is impossible to have computational complexity $O(N)$

First-order method

Algorithm	Regret	Time (per round)
Universal Portfolio [8, 15]	$N \ln T$	$T^{14} N^4$
ONS [12]	$GN \ln T$	$N^{3.5}$
FTRL [3]	$G^2 N \ln(NT)$	$TN^{2.5}$
EG [14]	$G\sqrt{T \ln N}$	N
Soft-Bayes [19]	$\sqrt{NT \ln N}$	N
ADA-BARRONS (this work)	$N^2(\ln T)^4$	$TN^{2.5}$

Ada-Barrons

Algorithm	Regret	Run-Time
Universal Portfolio	$d \ln T$	$d^4 T^{15}$
ONS	$Gd \ln T$	$d^{3.5} T$
FTRL	$G^2 d \ln(dT)$	$d^{2.5} T^2$
EG	$G\sqrt{T \ln d}$	dT
Soft-Bayes	$\sqrt{dT \ln d}$	dT
Ada-BARRONS	$d^2 \ln^4 T$	$d^{2.5} T^2 + d^{3.5} T$
AdaMix + DONS	$d^2 \ln^5 T$	$d^3 T \ln^2 T$

AdaMix + Dons

4. Second-order solution

4.7 Recent work

Efficient Online Portfolio with Logarithmic Regret

Haipeng Luo
Department of Computer Science
University of Southern California
haipengl@usc.edu

Chen-Yu Wei
Department of Computer Science
University of Southern California
chenyu.wei@usc.edu

Kai Zheng
Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
Center for Data Science, Peking University, Beijing Institute of Big Data Research
zhengk92@pku.edu.cn

Abstract

We study the decades-old problem of online portfolio management and propose the first algorithm with logarithmic regret that is not based on Cover’s Universal Portfolio algorithm and admits much faster implementation. Specifically Universal Portfolio enjoys optimal regret $\mathcal{O}(N \ln T)$ for N financial instruments over T rounds, but requires log-concave sampling and has a large polynomial running time. Our algorithm, on the other hand, ensures a slightly larger but still logarithmic regret of $\mathcal{O}(N^2 (\ln T)^4)$, and is based on the well-studied Online Mirror Descent framework with a novel regularizer that can be implemented via standard optimization methods in time $\mathcal{O}(TN^{2.5})$ per round. The regret of all other existing works is either polynomial in T or has a potentially unbounded factor such as the inverse of the smallest price relative.

Ada-Barrons

Damped Online Newton Step for Portfolio Selection

Zakaria Mhammedi
Alexander Rakhlin
Massachusetts Institute of Technology
{mhammedi, rakhlin}@mit.edu

February 16, 2022

Abstract

We revisit the classic online portfolio selection problem, where at each round a learner selects a distribution over a set of portfolios to allocate its wealth. It is known that for this problem a logarithmic regret with respect to Cover’s loss is achievable using the Universal Portfolio Selection algorithm, for example. However, all existing algorithms that achieve a logarithmic regret for this problem have per-round time and space complexities that scale polynomially with the total number of rounds, making them impractical. In this paper, we build on the recent work by Haipeng et al. 2018 and present the first practical online portfolio selection algorithm with a logarithmic regret and whose per-round time and space complexities depend only logarithmically on the horizon. Behind our approach are two key technical novelties of independent interest. We first show that the Damped Online Newton steps can approximate mirror descent iterates well, even when dealing with time-varying regularizers. Second, we present a new meta-algorithm that achieves an adaptive logarithmic regret (i.e. a logarithmic regret on any sub-interval) for mixable losses.

AdaMix-DONS

Efficient and Near-Optimal Online Portfolio Selection

Rémi Jézéquel^{*†} Dmitrii M. Ostrovskii^{*‡} Pierre Gaillard[§]

Abstract

In the problem of online portfolio selection as formulated by Cover (1991) [12], the trader repeatedly distributes her capital over d assets in each of $T > 1$ rounds, with the goal of maximizing the total return. Cover proposed an algorithm, termed Universal Portfolios, that performs nearly as well as the best (in hindsight) static assignment of a portfolio, with an $\mathcal{O}(d \log(T))$ logarithmic regret. Without imposing any restrictions on the market this guarantee is known to be worst-case optimal, and no other algorithm attaining it has been discovered so far. Unfortunately, Cover’s algorithm crucially relies on computing certain d -dimensional integral, which must be approximated in any implementation; this results in a prohibitive $\tilde{O}(d^4(T+d)^{14})$ per-round runtime for the fastest known implementation due to Kalai and Vempala (2002) [22]. We propose an algorithm for online portfolio selection that admits essentially the same regret guarantee as Universal Portfolios—up to a constant factor and replacement of $\log(T)$ with $\log(T+d)$ —yet has a drastically reduced runtime of $\tilde{O}(d^2(T+d))$ per round. The selected portfolio minimizes the observed logarithmic loss regularized with the log-determinant of its Hessian—equivalently, the hybrid logarithmic-volumetric barrier of the polytope specified by the asset return vectors. As such, our work reveals surprising connections of online portfolio selection with two classical topics in optimization theory: cutting-plane and interior-point algorithms.

VB-FTRL

5. Regularization

5.1 - Motivation

- Follow-The-Leader (FTL) learning framework: $x_{t+1} = \operatorname{argmin}_{x \in \mathbb{K}} \sum_{t=1}^T f_t(x)$
 - **Remark:** Linear regret in the **worst case!** $O(T)$
 - **Example:** Suppose $\mathbb{K} = [-1, 1]$ the loss is linear, $f_1(x) = \frac{x}{2}$ and $\sum_{t=1}^T f_t(x) = \begin{cases} \frac{1}{2}x & T \text{ is odd} \\ \frac{-1}{2}x & \text{otherwise} \end{cases} \Rightarrow$
 $x_t = \begin{cases} 1 & t \text{ is odd} \\ -1 & t \text{ is even} \end{cases} \Rightarrow T \text{ mistakes}$

Fluctuations/Unstable

5. Regularization

5.2 - Regularization function

- **Regularization function:** $R : \mathbb{K} \rightarrow \mathbb{R}$ (strongly convex and smooth)
 - **Assumption:** Twice differentiable and $\nabla^2 R(x) \geq 0$
- **Diameter of set \mathbb{K} relative to R :** $D_R = \sqrt{\max_{x,y \in \mathbb{K}} \{R(x) - R(y)\}}$
- **Dual norm:** $\|y\|^* \doteq \sup_{\|x\| \leq 1} x^T y$
 - **Example:** $\|x\|_A = \sqrt{x^T A x}$, $\|x\|_A^* = \|x\|_{A^{-1}}$, dual norm of $\|\cdot\|_2$ is $\|\cdot\|_2$, dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$
- **Legendre—Fenchel transformation:** $f^*(u) = \sup_x u^T x - f(x)$
 - **Corollary:** $f^{**} = f$
- **Bregman divergence:** $B_R(x \| y) = R(x) - R(y) - \nabla R(y)^T (x - y)$
 - **Example:** $R(x) = \frac{1}{2} \|x\|^2 \Rightarrow B_R(x \| y) = \frac{1}{2} \|x - y\|^2$
 - **Intuition:** The **distance** between value of **regularization function** at x and value of **first-order Taylor series approximation** at x .
 - **Corollary:** $B_R(x \| y) = B_{R^*}(\nabla R(y) \| \nabla R(x))$
- **Remark:** $B_R(x \| y) = \frac{1}{2} \|x - y\|_z^2$ where $z \in [x, y]$
 - **Intuition:** Bregman divergence is **equal** to second derivative at intermediate point.

5. Regularization

5.3 - Regularized-Follow-The-Leader (RFTL)

- Change the objective: $x_{t+1} = \arg \min_{x \in \mathbb{K}} \left\{ \sum_{t=1}^T f_t(x) + R(x) \right\}$
- ▶ **Intuition:** Balance between optimize $\sum_{t=1}^T f_t(x)$ and $R(x)$

Algorithm 13 Regularized Follow The Leader

- 1: Input: $\eta > 0$, regularization function R , and a bounded, convex and closed set \mathcal{K} .
- 2: Let $\mathbf{x}_1 = \arg \min_{\mathbf{x} \in \mathcal{K}} \{R(\mathbf{x})\}$.
- 3: **for** $t = 1$ to T **do**
- 4: Play \mathbf{x}_t and observe cost $f_t(\mathbf{x}_t)$.
- 5: Update

$$\mathbf{x}_{t+1} = \arg \min_{\mathbf{x} \in \mathcal{K}} \left\{ \eta \sum_{s=1}^t \nabla_s^\top \mathbf{x} + R(\mathbf{x}) \right\}$$

6: **end for**

5. Regularization

$$B_R(x || y) = \frac{1}{2} ||x - y||_z^2 \text{ where } z \in [x, y]$$

5.3 RFTL - Analysis

- **Theorem:** $\forall x \in \mathbb{K}$, we can have the following bound

$$\text{Regret}_t \leq 2\eta \sum_{t=1}^T ||\nabla f_t(x_t)||_{x_t, x_{t+1}}^{*2} + \frac{R(x) - R(x_1)}{\eta}$$

- **Corollary:** With $||\nabla f_t(x_t)||_{x_t, x_{t+1}}^{*} \leq G_R$ and optimize the choice of η , then

$$\text{Regret}_t \leq 2D_R G_R \sqrt{2T}$$

Similar to OGD in general setting

5. Regularization

5.4 - Online Mirror Descent - Motivation

- Recall the update of GD:

$$\begin{aligned}x_{k+1} &= \pi_{\mathbb{K}}(x_k - \alpha_k \nabla f(x_k)) \\&= \arg \min_{x \in \mathbb{K}} ||x_k - \alpha_k \nabla f(x_k) - x||_2^2 \\&= \arg \min_{x \in \mathbb{K}} ||x - x_k||_2^2 + 2\alpha_k \langle x, \nabla f(x_k) \rangle + C \\&= \arg \min_{x \in \mathbb{K}} \langle x, \alpha_k \nabla f(x_k) \rangle + \frac{1}{2} ||x - x_k||_2^2\end{aligned}$$

- C: constant does not depend on x
- **Intuition:** First term is related to update x_k in the **negative direction of the gradient**, while second term constrain the **distance between x_k and x** . In other words, the second term is **regularization**.
- **Remark:** We can **generalize** the regularization with Bregman divergence.

5. Regularization

5.4 - Online Mirror Descent - Motivation

- **Generalize** the notation of distance:

$$\begin{aligned} x_{k+1} &= \arg \min_{x \in \mathbb{K}} \langle x, \alpha_k \nabla f(x_k) \rangle + B(x \mid x_k) \\ &= \arg \max_{x \in \mathbb{K}} \langle x, \nabla R(x_k) - \alpha_k \nabla f(x_k) \rangle - R(x) \end{aligned}$$

- **Fenchel transformation** of $R(x)$ with $u = \nabla R(x_k) - \alpha_k \nabla f(x_k)$, $R^*(u) = \sup_x \langle x, u \rangle - R(x)$

- **Update equation:** $x_{k+1} = \nabla R^*(\nabla R(x_k) - \alpha_k \nabla f(x_k))$

- **Intuition:** Map x_k to dual space using ∇R , then update gradient descent in that space with $-\alpha_k \nabla f(x_k)$. Finally, we map back to primal space using ∇R^*

5. Regularization

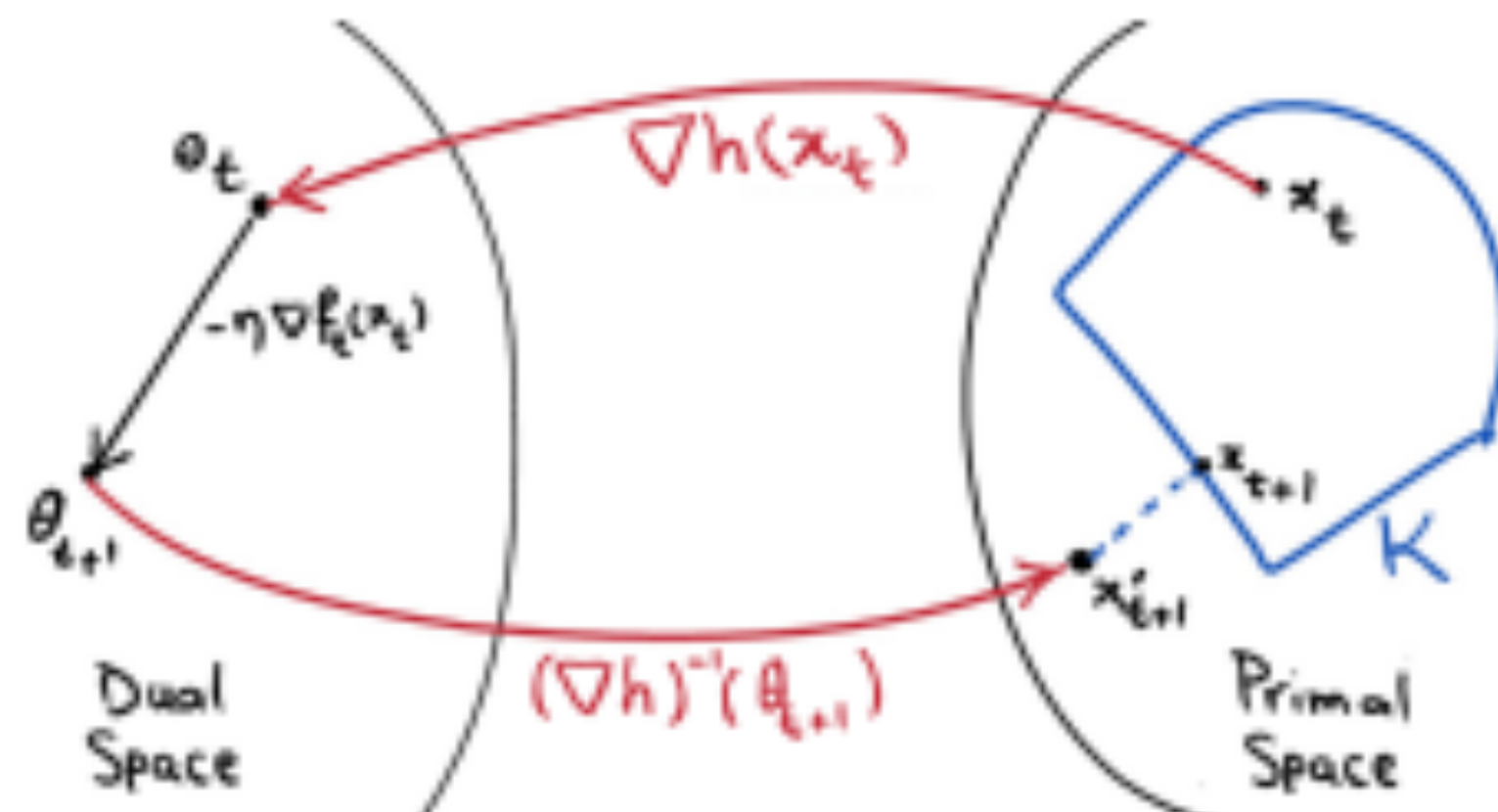
5.4 - Online Mirror Descent - Motivation

- **Generalize** the notation of distance:

$$x_{k+1} = \arg \min_{x \in \mathbb{K}} \langle x, \alpha_k \nabla f(x_k) \rangle + B(x \mid x_k)$$

$$= \arg \max_{x \in \mathbb{K}} \langle x, \nabla R(x_k) - \alpha_k \nabla f(x_k) \rangle - R(x) \quad \text{Fenchel conjugate}$$

- **Remark:** Intuitively, instead of updating in the primal space, we project the x_k to a **dual space** and then update in that space.



$||\cdot||$ on \mathbb{R}^n
 Distance generating function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ (R in our case)
 Mirror map $\nabla h : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (∇R in our case)

5. Regularization

5.4 - Online Mirror Descent - Algorithm

- **Lazy:** Instead of going back and forth, update only in the dual space, and (only) project to \mathbb{K} at decision.
- **Agile:** Maintains a feasible point at all times

Algorithm 14 Online Mirror Descent

- 1: Input: parameter $\eta > 0$, regularization function $R(\mathbf{x})$.
- 2: Let \mathbf{y}_1 be such that $\nabla R(\mathbf{y}_1) = \mathbf{0}$ and $\mathbf{x}_1 = \arg \min_{\mathbf{x} \in \mathcal{K}} B_R(\mathbf{x} || \mathbf{y}_1)$.
- 3: **for** $t = 1$ to T **do**
- 4: Play \mathbf{x}_t .
- 5: Observe the loss function f_t and let $\nabla_t = \nabla f_t(\mathbf{x}_t)$.
- 6: Update \mathbf{y}_t according to the rule:

[Lazy version]	$\nabla R(\mathbf{y}_{t+1}) = \nabla R(\mathbf{y}_t) - \eta \nabla_t$
[Agile version]	$\nabla R(\mathbf{y}_{t+1}) = \nabla R(\mathbf{x}_t) - \eta \nabla_t$

Project according to B_R :

$$\mathbf{x}_{t+1} = \arg \min_{\mathbf{x} \in \mathcal{K}} B_R(\mathbf{x} || \mathbf{y}_{t+1})$$

7: **end for**

5. Regularization

5.4 - Online Mirror Descent - Convergence

- For every $x \in \mathbb{K}$, we have the following regret bound

- $$Regret_T \leq \frac{\eta}{4} \sum_{t=1}^T ||\nabla f_t(x_t)||_{x_t, x_{t+1}}^{*2} + \frac{R(x) - R(x_1)}{2\eta}$$

- If $||\nabla f_t(x_t)||_{x_t, x_{t+1}}^{*} \leq G_R$, then $Regret_T \leq D_R G_R \sqrt{T}$

- **Remark:** OMD is a general version of OGD

Similar to OGD in regular case

Outline

- 1. **Convex Optimization:** convex set, convex function, G -Lipschitz function, α -strongly convex, β -smooth, first-order condition, second-order condition
- 2. **Online Convex Optimization:** Formulation, Prediction from expert advices
- 3. **First-order method:** Online Gradient Descent
- 4. **Second-order method:** γ -exp-concave function, Online Newton Step
- 5. **Regularization:** Regularized-Follow-The-Leader, Online Mirror Descent

Thank you!

References

1. Convex Optimization

- [Why do we care about convex optimization](#)
- [Convex set wikipedia](#)
- [Convex function - UCLA's Convex Optimization, Lecture 3](#)
- [Directional derivative](#)
- [Why first-order condition is important?](#)
- [Taylor series expansion: First and Second-order](#)
- [Proof second-order condition](#)
- [Subgradient](#)
- [Proof of first order optimality condition](#)
- [Fundamental of calculus in higher dimension](#)
- [Proof and example of Lipschitz continuity](#)

References

2. Online Convex Optimization

- [Sublinear function](#)
- [Intuition behind \$\log_2\$](#)
- [Proof of theorem 1.1 and definition of adversary](#)
- [Weighted majority algorithm example](#)
- [Proof of weighted majority algorithm](#)
- [Proof of randomized weighted majority algorithm](#)

References

3. First-order solution

References

4. Second-order method

References

5. Regularization