

# **Lec 15: Specialized Optimization Methods II**

**15-369/669/769: Numerical Computing**

**Instructor: Minchen Li**

# Table of Content

- Proximal Algorithms
- Global Optimization
- Online Optimization

# Table of Content

- Proximal Algorithms
- Global Optimization
- Online Optimization

# Proximal Algorithms

## ADMM Recap

Alternating Direction Method of Multipliers (ADMM) solves

$$\min_{\mathbf{x}, \mathbf{z}} f(\mathbf{x}) + h(\mathbf{z}) \quad \text{s.t.} \quad A\mathbf{x} + B\mathbf{z} = \mathbf{c}.$$

Augmented Lagrangian:

$$\Lambda_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{x}) + h(\mathbf{z}) + \frac{\rho}{2} \|A\mathbf{x} + B\mathbf{z} - \mathbf{c}\|_2^2 + \boldsymbol{\lambda}^\top (A\mathbf{x} + B\mathbf{z} - \mathbf{c})$$

ADMM iterations:

$$\mathbf{x}_{i+1} \leftarrow \arg \min_{\mathbf{x}} \Lambda_\rho(\mathbf{x}, \mathbf{z}_i, \boldsymbol{\lambda}_i)$$

$$\mathbf{z}_{i+1} \leftarrow \arg \min_{\mathbf{z}} \Lambda_\rho(\mathbf{x}_{i+1}, \mathbf{z}, \boldsymbol{\lambda}_i)$$

$$\boldsymbol{\lambda}_{i+1} \leftarrow \boldsymbol{\lambda}_i + \rho(A\mathbf{x}_{i+1} + B\mathbf{z}_{i+1} - \mathbf{c})$$

Each step is often simpler and can exploit problem structure or parallelism.

# Proximal Algorithms

## Overview

The ADMM algorithm is an example of a **proximal algorithm**, where each step involves a *proximal operator*.

**Definition (Proximal operator):**

$$\text{prox}_f(\mathbf{v}) := \arg \min_{\mathbf{x} \in \mathbb{R}^n} \left( f(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \mathbf{v}\|_2^2 \right)$$

Minimizing  $f(\mathbf{x})$  while staying close to  $\mathbf{v}$ .

**Scaled form:**

$$\text{prox}_{c \cdot f}(\mathbf{v}) = \arg \min_{\mathbf{x}} \left( f(\mathbf{x}) + \frac{1}{2c} \|\mathbf{x} - \mathbf{v}\|_2^2 \right), \quad c > 0.$$

Balances minimizing  $f(\mathbf{x})$  with staying close to  $\mathbf{v}$ .

# Proximal Algorithms

## Connection to Gradient Descent

Linearize  $f(\mathbf{x})$  near  $\mathbf{v}$ :

$$f(\mathbf{x}) \approx f(\mathbf{v}) + \nabla f(\mathbf{v})^\top (\mathbf{x} - \mathbf{v})$$

Then:

$$\text{prox}_{c.f}(\mathbf{v}) \approx \mathbf{v} - c \nabla f(\mathbf{v})$$

$\Rightarrow$  analogous to a gradient descent step with learning rate  $c$ .

**Proximal point algorithm:**

$$\mathbf{x}_k = \text{prox}_{c.f}(\mathbf{x}_{k-1})$$

Provides stronger convergence guarantees than gradient descent.

Even approximate proximal steps are often sufficient.

# Proximal Algorithms

## Example: Iterative Refinement

Solve  $A\mathbf{x} = \mathbf{b}$  for SPD  $A \in \mathbb{R}^{n \times n}$ .

Define

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top \mathbf{x}.$$

Apply the proximal point method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \left(A + \frac{1}{c}I\right)^{-1} (\mathbf{b} - A\mathbf{x}_k).$$

### Properties:

- $\mathbf{x}_k \rightarrow A^{-1}\mathbf{b}$  as  $k \rightarrow \infty$ .
- $(A + \frac{1}{c}I)$  has better conditioning than  $A$ .
- Each iteration corrects  $\mathbf{x}_k$  for residual  $\mathbf{b} - A\mathbf{x}_k$ .

# Proximal Algorithms

## Proximal Gradient Method

For composite objectives:

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

If  $\text{prox}_g$  is known but not  $\text{prox}_f$ , combine:

$$\mathbf{x}_{k+1} = \text{prox}_{c_k \cdot g}(\mathbf{x}_k - c_k \nabla f(\mathbf{x}_k))$$

### Remarks:

- Gradient step on smooth  $f$ , proximal step on (possibly non-smooth)  $g$ .
- Key in sparse optimization and regularization problems.

# Proximal Algorithms

## Example: L1 Regularization

Optimize

$$f(\mathbf{x}) + \alpha \|\mathbf{x}\|_1, \quad f(\mathbf{x}) = \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2, \quad g(\mathbf{x}) = \alpha \|\mathbf{x}\|_1.$$

Proximal operator of  $\|\cdot\|_1$ :

$$\text{prox}_{c \cdot g}(\mathbf{v}) = \text{sign}(\mathbf{v}) \odot \max(|\mathbf{v}| - c\alpha, 0)$$

Iteration (soft thresholding):

$$\begin{aligned} \mathbf{w}_k &\leftarrow \mathbf{x}_k - c_k (A^\top A \mathbf{x}_k - A^\top \mathbf{b}) \\ \mathbf{x}_{k+1} &\leftarrow \text{sign}(\mathbf{w}_k) \odot \max(|\mathbf{w}_k| - c_k \alpha, 0) \end{aligned}$$

Allows handling non-differentiable regularization terms via proximal steps.

# Table of Content

- Proximal Algorithms
- Global Optimization
- Online Optimization

# Global Optimization

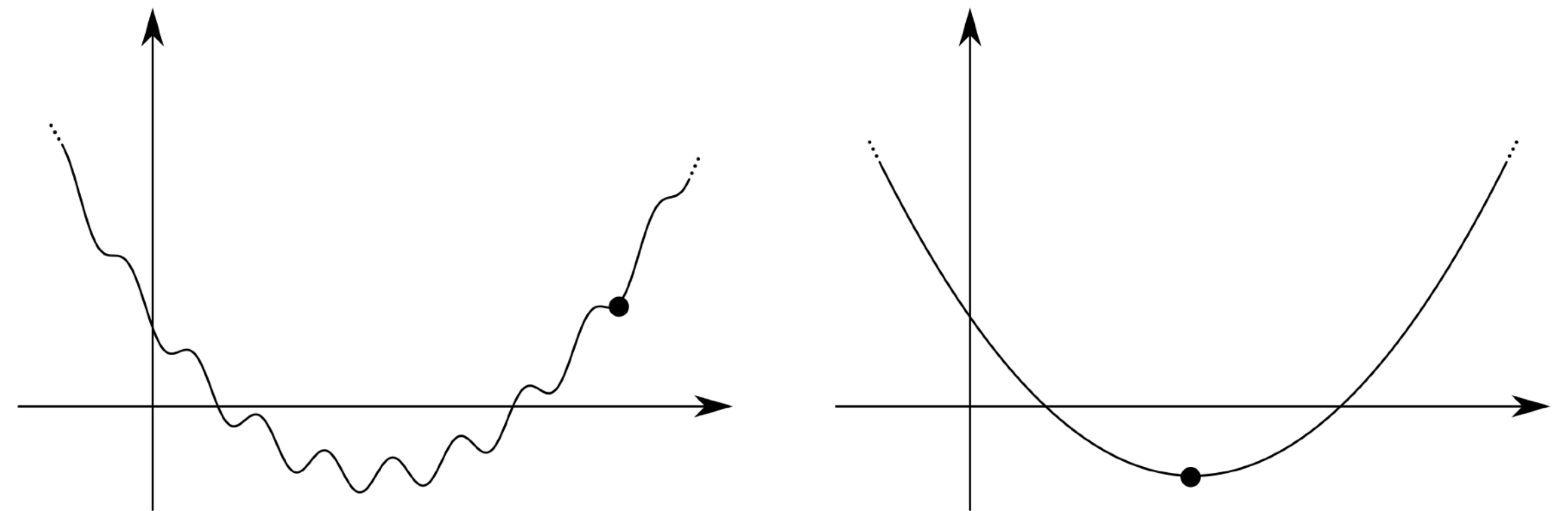
## Motivation

- Lightweight approaches (least-squares, IRLS, proximal methods) exploit special structure for efficient local optimization.
- Some problems lack structure or a good initial guess, making Newton or Taylor-based methods fail.
- When no simplifying assumption like convexity holds, we must perform **global optimization**: searching the entire feasible set.
- **Challenge:** Global optimization is nearly ill-posed — the space  $\mathbb{R}^n$  of possible inputs is huge, and verifying a global optimum is infeasible.
- **Heuristic strategies:**
  - Approximate  $f(\mathbf{x})$  with an easier function to obtain a better initialization.
  - Broadly sample  $\mathbf{x}$  to better understand  $f$ 's global landscape.

# Global Optimization

## Graduated Optimization: Core Idea

- Local minima can trap standard methods.



- **Graduated optimization** minimizes a sequence of progressively harder objectives:

$$f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}) \quad \text{with } f_k = f.$$

- Each solution of  $f_i$  serves as initialization for  $f_{i+1}$ .
- Coarse objectives (e.g., smoothed or blurred) yield better starting points.

# Global Optimization

## Example: Image Alignment



Original

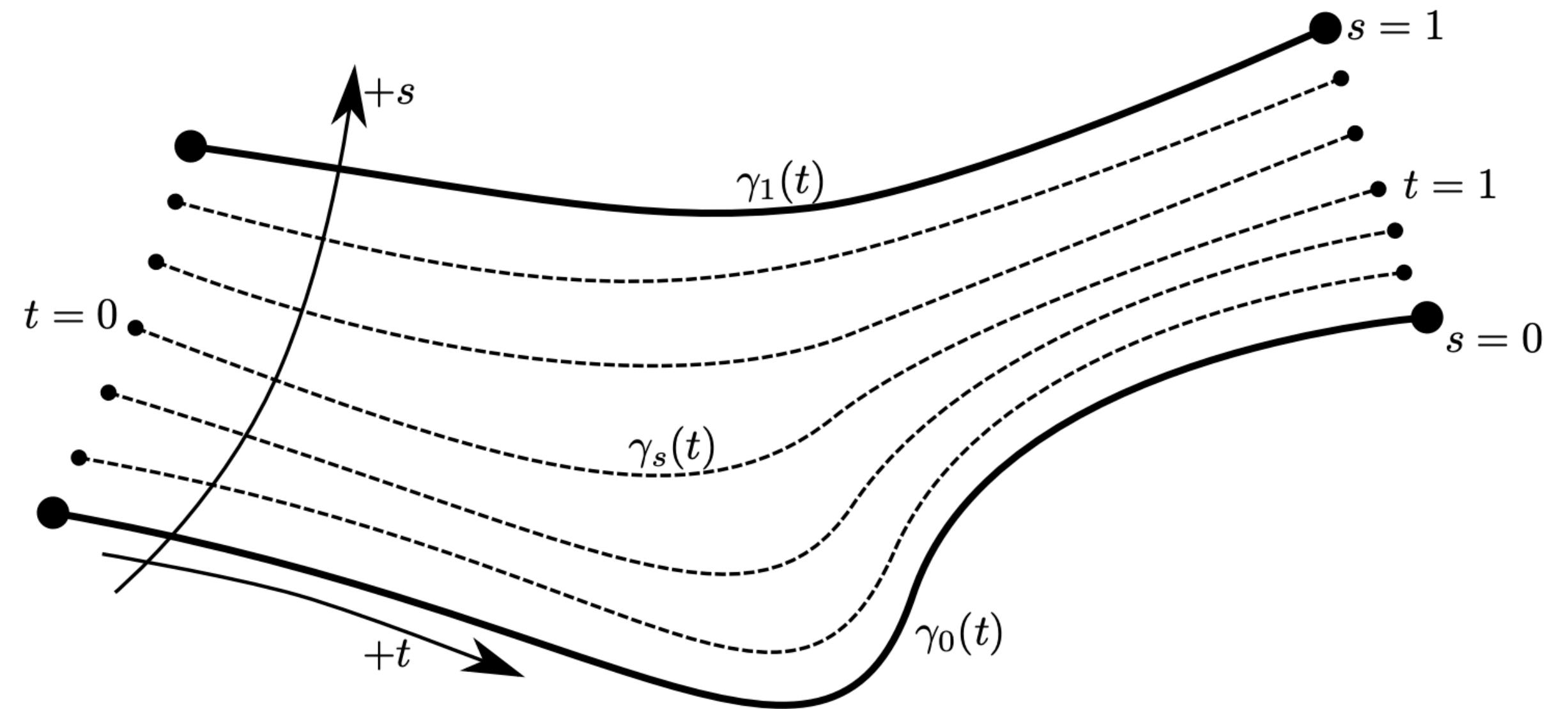


Blurred

- Aligning images directly can fail due to high-frequency details.
- **Strategy:** Blur the images — high frequencies suppressed, coarse structure preserved.
- Coarse alignment gives a strong initialization; finer iterations refine it.
- **Mathematical form:**
$$f_i(\mathbf{x}) = f(\mathbf{x}) * g_{\sigma_i}, \quad \sigma_i \rightarrow 0.$$
- **Interpretation:** Optimization proceeds in a *scale-space*, starting smooth and adding detail gradually.

# Global Optimization

## Homotopy Continuation Methods



- **Definition (Homotopy Functions):** Two functions  $f(\mathbf{x})$  and  $g(\mathbf{x})$  are *homotopic* if

$$\exists H(\mathbf{x}, s) \text{ continuous, } H(\mathbf{x}, 0) = f(\mathbf{x}), H(\mathbf{x}, 1) = g(\mathbf{x}).$$

- Optimization gradually transforms  $H(\mathbf{x}, 0)$  (easy) into  $H(\mathbf{x}, 1) = f(\mathbf{x})$  (hard).
- Each iteration solves  $\min_{\mathbf{x}} H(\mathbf{x}, s_i)$  using previous  $\mathbf{x}_{s_{i-1}}^*$  as initialization.
- This traces a continuous *solution path*  $(\mathbf{x}(t), s(t))$ .

# Global Optimization

## Example: Root-Finding via Homotopy

- **Goal:** Solve  $\arctan(x) = 0$  with poor initial guess  $x_0 = 4$  (Newton diverges).
- Define
$$H(x, s) := \arctan(x) + (s - 1) \arctan(4).$$
- At  $s = 0$ :  $H(x, 0) = \arctan(x) - \arctan(4)$  has root  $x_0 = 4$ .
- Increment  $s$  by 0.1; each time, minimize  $H(x, s_i)$  initialized from the previous root.
- The sequence converges to  $x^* = 0$  at  $s = 1$ .

# Global Optimization

## General Solution Path Perspective

- Define a curve  $(\mathbf{x}(t), s(t))$  with

$$s(0) = 0, \quad s(1) = 1, \quad \text{and } \mathbf{x}(t) \text{ minimizes } H(\mathbf{x}, s(t)).$$

- $s(t)$  can be non-monotone as long as it reaches 1 eventually.
- Advanced continuation methods treat  $(\mathbf{x}(t), s(t))$  as satisfying differential equations derived from  $H$ .
- These techniques link optimization to ODE theory.

# Global Optimization

## Randomized Global Optimization

- When smoothing or gradients fail, sample-based heuristics explore the landscape.
- Gradients may be unreliable or unavailable for noisy  $f$ .
- Inspired by natural systems: ants, flocking, thermodynamics, genetic evolution.
- Focus: **Particle Swarm Optimization (PSO)**.

# Global Optimization

## Particle Swarm Optimization (PSO): Overview

**Setup:** Maintain particles with positions  $\mathbf{x}_i$ , velocities  $\mathbf{v}_i$ , and memories.

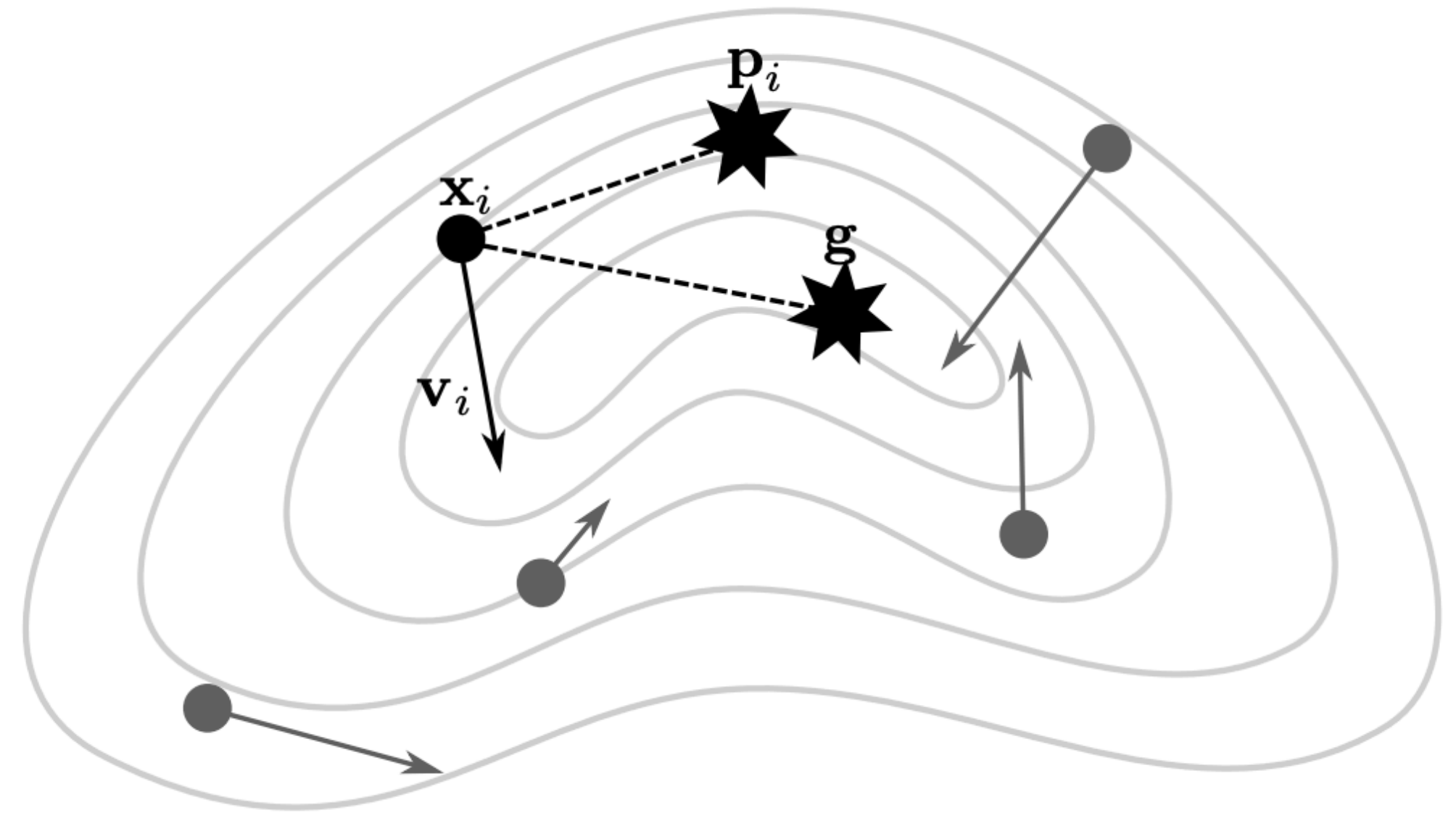
- $\mathbf{p}_i$ : best position of particle  $i$  so far.
- $\mathbf{g}$ : global best position found by any particle.

**Velocity update:**

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \alpha(\mathbf{p}_i - \mathbf{x}_i) + \beta(\mathbf{g} - \mathbf{x}_i),$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i.$$

- $\alpha, \beta \geq 0$  control attraction to local/global bests.
- Larger  $\alpha, \beta \Rightarrow$  faster exploitation, less exploration.



# Global Optimization

## Particle Swarm Optimization (PSO): Algorithm

---

### Algorithm 1 Basic PSO

---

```
1: Initialize  $\{\mathbf{x}_i, \mathbf{v}_i\}_{i=1}^k$ 
2: Set  $\mathbf{p}_i \leftarrow \mathbf{x}_i, \mathbf{g} \leftarrow \arg \min_i f(\mathbf{p}_i)$ 
3: for each iteration do
4:   for each particle  $i$  do
5:      $\mathbf{v}_i \leftarrow \mathbf{v}_i + \alpha(\mathbf{p}_i - \mathbf{x}_i) + \beta(\mathbf{g} - \mathbf{x}_i)$ 
6:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$ 
7:     if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then
8:        $\mathbf{p}_i \leftarrow \mathbf{x}_i$ 
9:     end if
10:  end for
11:   $\mathbf{g} \leftarrow \arg \min_i f(\mathbf{p}_i)$ 
12: end for
13: return  $\mathbf{g}$ 
```

---

- **Note:** PSO is heuristic — not guaranteed to converge, but often effective in practice.

# Global Optimization

## Summary

- **Graduated optimization:** progressively refine smoother objectives to escape local minima.
- **Homotopy methods:** continuously deform an easy problem into the target one.
- **Randomized search:** explore the space stochastically (e.g., PSO).
- All are *heuristics*: improve robustness but do not guarantee global optimality.

# Table of Content

- Proximal Algorithms
- Global Optimization
- Online Optimization

# Online Optimization

## What is Online Optimization?

- In many modern settings (machine learning, game theory, finance), the optimization objective changes over time.
- **Online optimization** models problems where input parameters, priorities, or objectives evolve each iteration.
- Algorithms must react adaptively to this changing environment, often under uncertainty or noise.
- **Example:** dynamic decision-making such as portfolio optimization or adaptive control.

# Online Optimization

## Example: Stock Market Problem

On each day  $t$ :

- Choose investment vector  $\mathbf{x}_t \in (\mathbb{R}^+)^n$ .
- At day's end, observe profit/loss  $f_t(\mathbf{x}_t)$ .
- Each day has a different profit function  $f_t$ , unknown at decision time.

**Goal:** design a policy predicting the optimal  $\mathbf{x}_t$  sequence despite the changing and uncertain  $f_t$ 's.

# Online Optimization

## Online Convex Optimization Framework

- **Setup** At each time  $t = 1, 2, \dots$ :
  1. Predict  $\mathbf{x}_t \in U$
  2. Receive loss function  $f_t : U \rightarrow \mathbb{R}$
  3. Suffer loss  $f_t(\mathbf{x}_t)$
- Assume each  $f_t$  is convex and  $U \subseteq \mathbb{R}^n$  convex.
- We phrase problems as **minimizing loss**, consistent with prior optimization topics.
- The algorithm cannot see  $f_t$  before choosing  $\mathbf{x}_t$ .
- The algorithm may remember past functions  $f_1, \dots, f_{t-1}$ .

# Online Optimization

## Measuring Performance: Regret

Since  $f_t$  is unknown before prediction, perfect performance is impossible.

**Naive measure:** cumulative loss  $\sum_{t=1}^T f_t(\mathbf{x}_t)$  — unfair since adversarial  $f_t$  can make  $\mathbf{x}_t$  look bad.

**Definition 12.3 (Regret):** The regret after  $T$  iterations is

$$R_T := \max_{\mathbf{u} \in U} \sum_{t=1}^T [f_t(\mathbf{x}_t) - f_t(\mathbf{u})].$$

- Measures how much worse we perform than the best fixed  $\mathbf{u}$  chosen in hindsight.
- Goal: minimize average regret  $R_T / T \rightarrow 0$  as  $T \rightarrow \infty$ .

# Online Optimization

## Follow the Leader (FTL) Strategy

**Idea:** assume past losses predict future ones.  $\mathbf{x}_t := \arg \min_{\mathbf{x} \in U} \sum_{s=1}^{t-1} f_s(\mathbf{x})$ .

- Works well if objectives  $f_t$  evolve smoothly.
- However, can fail badly if  $f_t$  fluctuates rapidly.

**Example (Failure of FTL):** For  $U = [0, 1]$  and

$$f_t(x) = \begin{cases} -\frac{1}{2}x, & t = 1, \\ x, & t \text{ even}, \\ -x, & t \text{ odd}, t > 1, \end{cases}$$

FTL alternates between  $x = 0$  and  $x = 1$ , incurring large loss. Fixing  $x = 0$  yields zero loss.

# Online Optimization

## Follow the Regularized Leader (FTRL)

To mitigate oscillation, add a **regularization term**:  $\mathbf{x}_t := \arg \min_{\mathbf{x} \in U} \left[ r(\mathbf{x}) + \sum_{s=1}^{t-1} f_s(\mathbf{x}) \right]$ .

- $r(\mathbf{x})$ : convex regularizer penalizing large or unstable updates.
- Common choices:
  - $r(\mathbf{x}) = \|\mathbf{x}\|_2^2$  (Tikhonov)
  - $r(\mathbf{x}) = \|\mathbf{x}\|_1$  (L1)
  - $r(\mathbf{x}) = \sum_i x_i \log x_i$  (Entropic)
- Analogous to conditioning improvements in numerical optimization.

# Online Optimization

## Strong Convexity and Regret Bounds

**Definition 12.4 (Strong Convexity):** A differentiable  $r(\mathbf{x})$  is  $\sigma$ -strongly convex w.r.t.  $\|\cdot\|$  if

$$(\nabla r(\mathbf{x}) - \nabla r(\mathbf{y}))^\top (\mathbf{x} - \mathbf{y}) \geq \sigma \|\mathbf{x} - \mathbf{y}\|_2^2.$$

A strongly convex regularizer is bowl-shaped with a lower bound for the curvature of that bowl.

**Proposition 12.1:** If each  $f_t$  is convex and  $L$ -Lipschitz, and  $r$  is  $\sigma$ -strongly convex, then

$$R_T \leq \left[ \max_{\mathbf{u} \in U} r(\mathbf{u}) - \min_{\mathbf{v} \in U} r(\mathbf{v}) \right] + \frac{TL^2}{\sigma}.$$

**Interpretation:**

- Larger  $\sigma$  (stronger convexity) reduces  $TL^2/\sigma$  but may widen  $\max r - \min r$ .
- Choose  $\sigma$  based on problem scale  $T$ .

# Online Optimization

## Example: Quadratic Regularizer

**Example (Choice of Regularizer):** Let  $r_\sigma(\mathbf{x}) = \frac{1}{2}\sigma\|\mathbf{x}\|_2^2$ .

- $\nabla r_\sigma(\mathbf{x}) = \sigma\mathbf{x}$ , so  $r_\sigma$  is  $\sigma$ -strongly convex.
- For  $U = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq 1\}$  and runtime  $T$ ,

$$R_T \leq (1 + L^2)\sqrt{T} \quad \text{if we set } \sigma = \sqrt{T}.$$

- $\Rightarrow$  Sublinear regret: FTRL outperforms FTL, which grows linearly in  $T$ .

# Online Optimization

## Discussion and Extensions

- Online optimization remains an active area of research.
- Many variants improve regret bounds or practical usability.
- FTRL trades off stability and adaptivity but can be computationally heavy.
- Simpler variants (e.g., linearized FTRL) can still work well for convex settings.
- Online methods underpin algorithms in online learning, adaptive control, and dynamic decision-making.

# Table of Content

- Proximal Algorithms
- Global Optimization
- Online Optimization