ORF522 – Linear and Nonlinear Optimization

22. Data-driven algorithms

Bartolomeo Stellato — Fall 2020

Ed forum

- Updated proof of spacial branch and bound convergence to clarify last step.
- Although on slide 15 we assume that lower bound L is non-decreasing, what if after a new refinement and a new relaxation process at step k+1, our new lower bound L^k+1 <= L^k? Does this happen in applications? If it happens, do we keep the new one (L^k+1) or do we keep the "better" one(L^k).

Today's lecture

[Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon, Bengio, Lodi, Prouvost]
[The Voice of Optimization, Bertsimas and Stellato]
[Online Mixed-Integer Optimization in Milliseconds, Bertsimas and Stellato]
[On learning and branching: a survey, Lodi and Zarpellon]

Data-driven algorithms (research topics)

- Machine learning
- Learning heuristics in branch and bound algorithms
- Learning strategies for parametric optimization
    - Strategies definition
    - Learning and sampling the strategies
    - Examples

Methods for nonconvex optimization

Convex optimization algorithms: global and typically **fast**

Nonconvex optimization algorithms: must give up one, global or fast

- **Local methods: fast** but **not global**
  Need not find a global (or even feasible) solution.
  They cannot certify global optimality because
  KKT conditions are not sufficient.

- **Global methods: global** but often **slow**
  They find a global solution and certify it.

Data to the rescue!

Nonconvex optimization is hard

Many algorithmic choices inside solvers

Lots of data available from experience

Can we use machine learning to build better algorithms?

Similar problems

- In practice, we solve many **similar problems with varying data**
- Most solvers do not exploit it
- We will consider families of similar problems

Machine learning

Imitation learning

Machine Learning →

- Discover patterns
- Understand structure

Minimize expected loss

$$\underset{w}{\text{minimize}} \quad \underset{X,Y \in \mathcal{P}}{\mathbf{E}} \ell(Y, f_w(X))$$

f_w: model
w: parameters

(we do not know $\mathcal{P}$)

Empirical probability

Training data
$\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^N$ →

$$\underset{w}{\text{minimize}} \quad \sum_{i=1}^{N} \ell(y_i, f_w(x_i))$$

Learning algorithmic decisions

Learning from demonstrations

$$\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$$

state,
situation,
conditions…

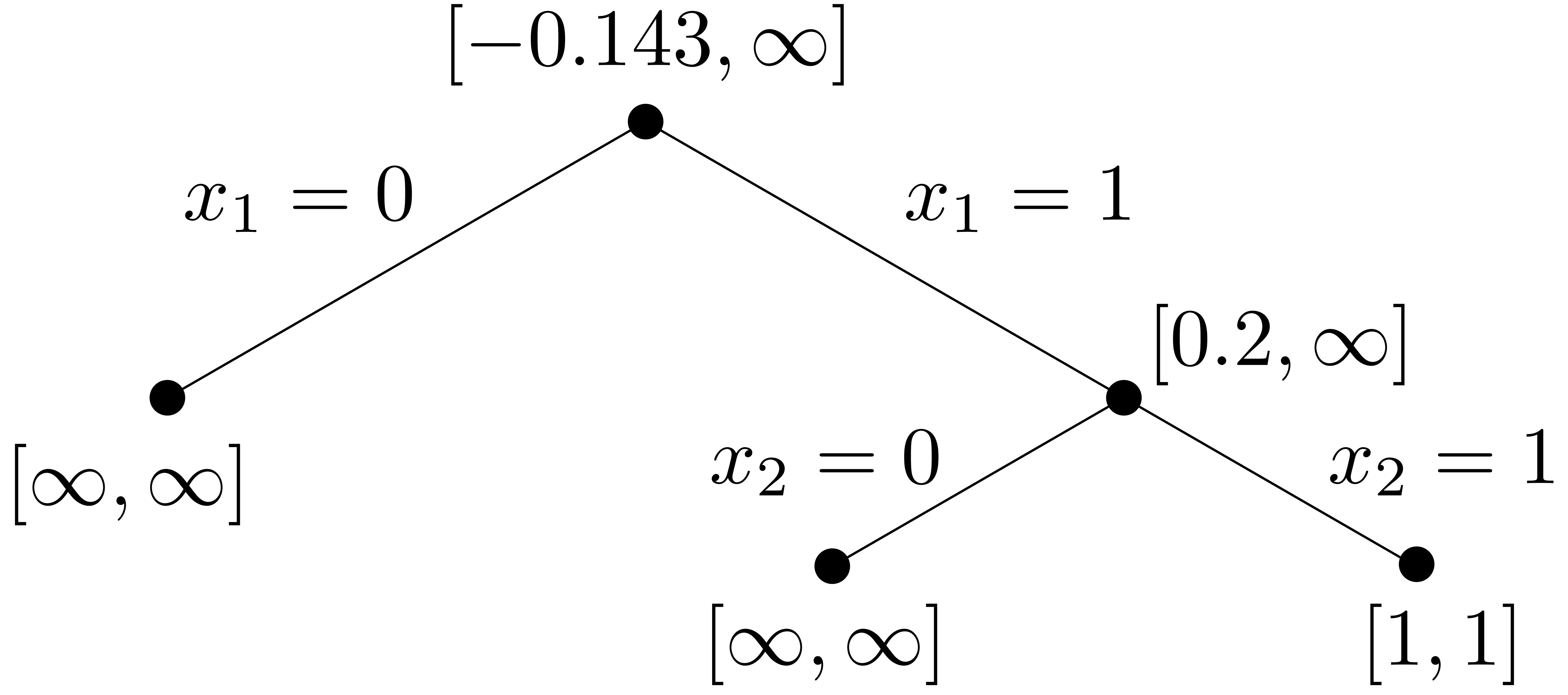
expert decisions

Goal: mimic expert decisions as closely as possible

Learning heuristics in branch and bound algorithms

Branch and bound for integer optimization

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \in \{0,1\}^n \end{array}$$



1. **Branch**: pick node i and index k
   form subproblems for $x_k = 0$ and $x_k = 1$
2. **Bound:**
   - Compute **lower** and **upper bounds**
   - Update global lower bounds on $f(x^\star)$

   $$L = \min_i \{L_i\}, \quad U = \min_i \{U_i\}$$
3. If $U - L \leq \epsilon$, **break**

Branch and bound decisions

Node selection: which node i?

- best-first: node with smallest lower bound
- depth-first: node with greatest depth

Variable selection: which fractional variable k?

- "least ambivalent": $x_k^\star \approx 0$ or 1
- "most ambivalent": $|x_k^\star - 1/2|$ is minimum

Can we learn better heuristics from data?

Heuristic selection: which upper bound algorithm? when?

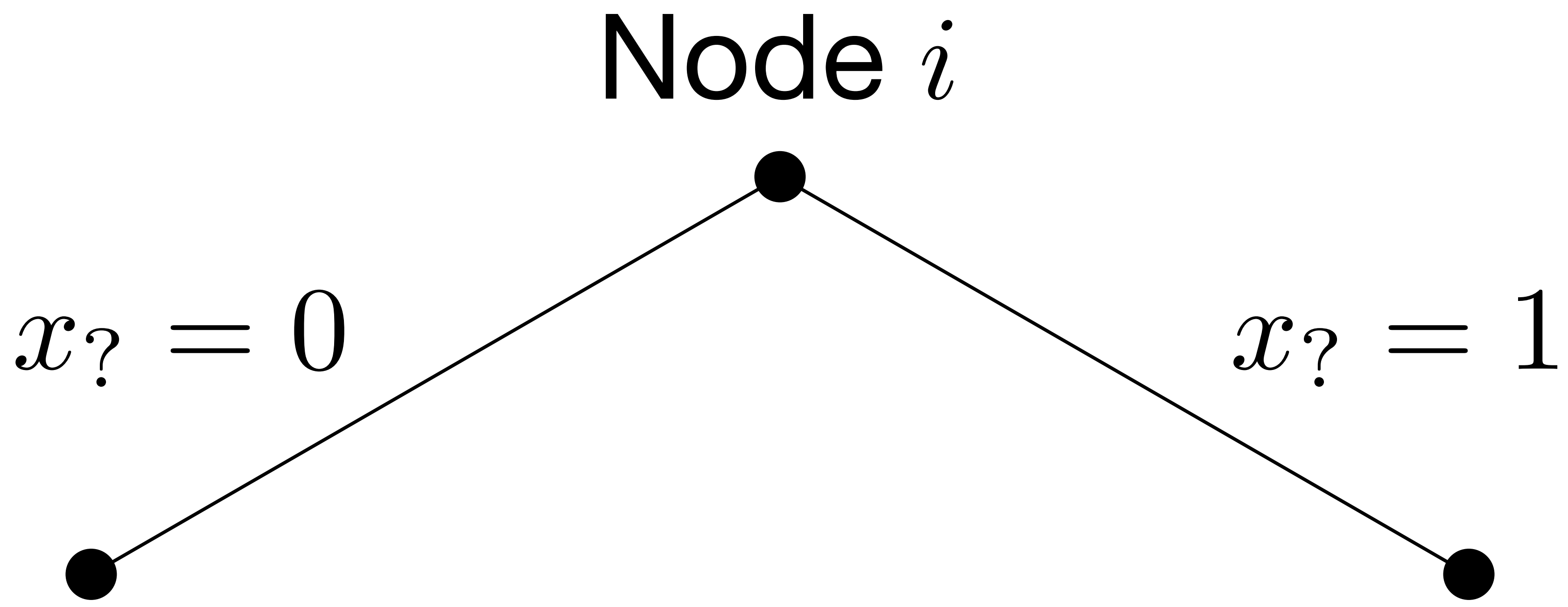
- Rounding
- Randomization
- Neighborhood search

Variable selection and strong branching

Relaxed problem at node i

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x_{\mathcal{I}} = \bar{x} \\ & 0 \leq x \leq 1 \end{array}$$

integer fixed components → $x_{\mathcal{I}} = \bar{x}$



Potential branching variables

Fractional $x_k, \quad k \in \mathcal{F} = \{1, \ldots, n\} \setminus \mathcal{I}$

Strong branching

- Split all potential candidates k
- For each one, solve relaxed problems for $x_k = 0$ and $x_k = 1$
- Pick k with highest “score”:
  the left and right lower bound increase the most

Too expensive!

Learning strong branching

Node features θ_i ⟶ **Strong branching scores** $(f_w(\theta_i))_k = s_k, \quad k \in \mathcal{F}$ ⟶ **Best variable** $k = \underset{k}{\text{argmax}}\ s_k$

Feature types

- **Static (problem instance)**:
  objective function coefficients,
  constraint coefficients stats.,
  constraint degrees (# of variables), etc.
- **Dynamic (incumbent, current LP relaxation, etc.)**:
  relaxed $x^\star$ distance to rounding,
  constraint degrees (# of variables), etc.

Multiclass classifier

- Linear function ($\mathrm{SVM}^{\mathrm{rank}}$)
- Decision tree
- Neural network

[Learning to Branch in Mixed Integer Programming, Khalil, Le Bodic, Song, Menhauser, Dilkina]
[A Machine Learning-Based Approximation of Strong Branching, Marcos Alvarez, Wehenkel, Louveaux]

Learning strong branching results

MIPLIB Examples with node limit $10,000$

	Solved by all methods			Not solved by at least one method			
	S/T	Nodes	Time (s)	S/T	Cl. Gap	Nodes	Time (s)
Most ambivalent	9/44	2,532	6.03	6/44	0.50	9,274	233.19
Strong	9/44	692	14.48	12/44	0.73	7,184	629.87
Learned	9/44	1,194	2.73	10/44	0.62	8,073	162.87

nodes reduction

faster than strong branching

Extensions

- What if we learn the 2-step strong branching (doubly-strong branching)?
- Can we learn while we solve the problem?

[A Machine Learning-Based Approximation of Strong Branching, Marcos Alvarez, Wehenkel, Louveaux]

Many more directions in branch and bound

Optimal node selection

[Learning to Search in Branch-and-Bound Algorithms, He et al]

Upper bound heuristic selection

[Learning to Run Heuristics in Tree Search, Khalil et al]

What if we do not have expert demonstrations?

[Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon, Bengio, Lodi, Prouvost]

Reinforcement learning

`ecole.ai`: OpenAI gym-like environment for Reinforcement Learning and Combinatorial Optimization

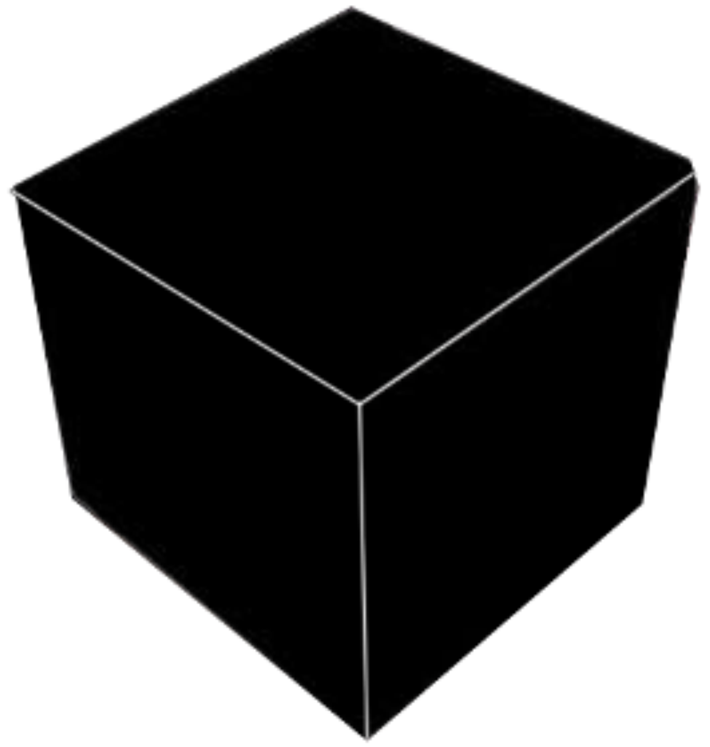
Learning for parametric optimization

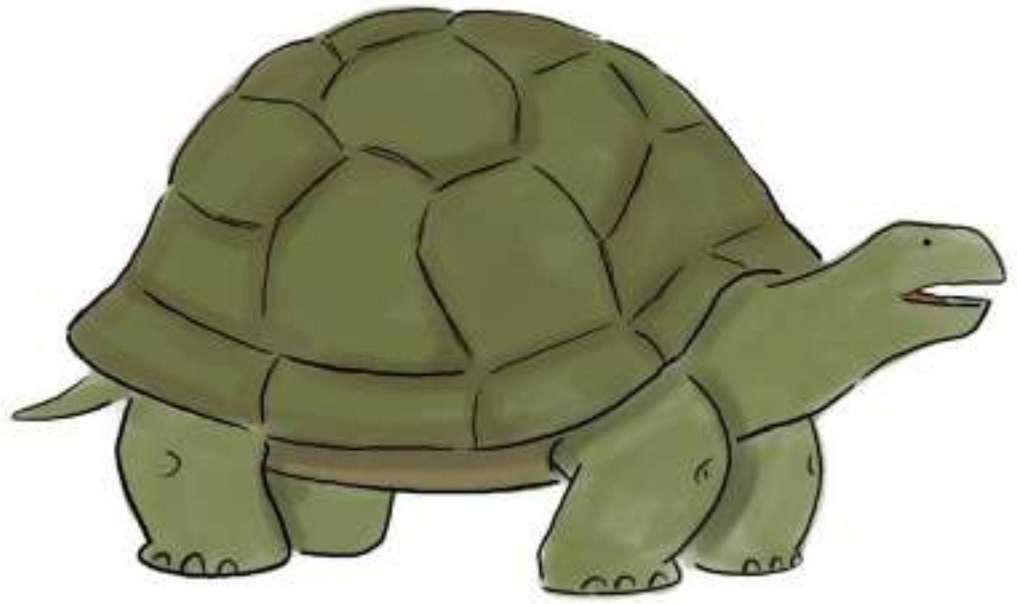
Parametric optimization

Limitations

$$\begin{array}{ll} \text{minimize} & f(x, \theta) \\ \text{subject to} & g(x, \theta) \leq 0 \end{array}$$

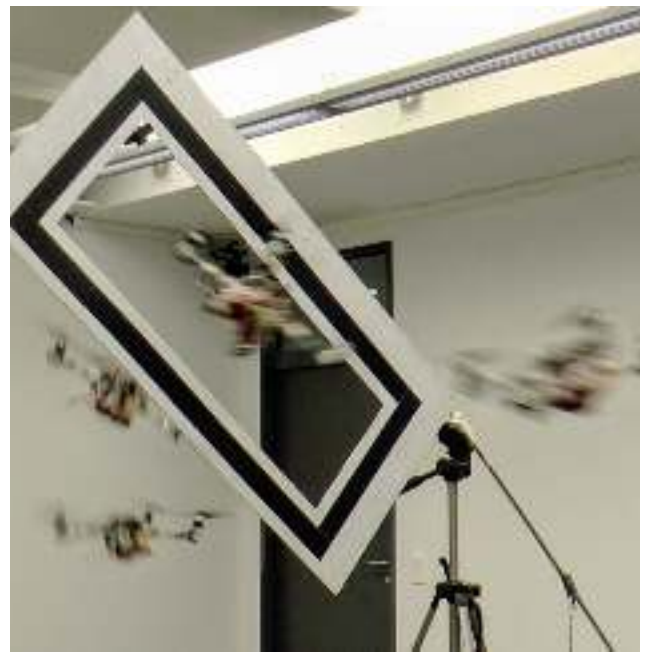
θ → **Optimization** → $x^\star$





Real-time optimization

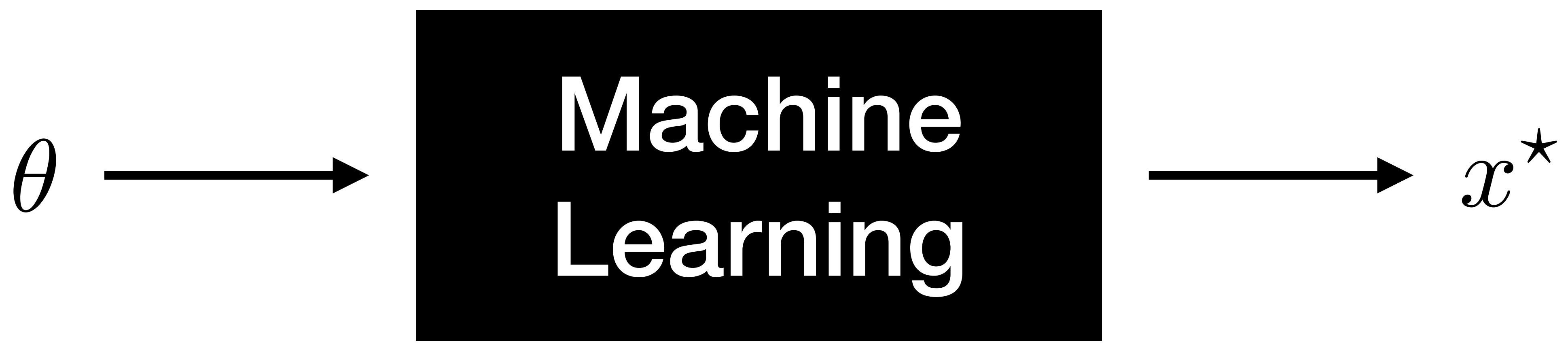
Fast real-time requirements



+

Low-cost computing platforms

End to end learning



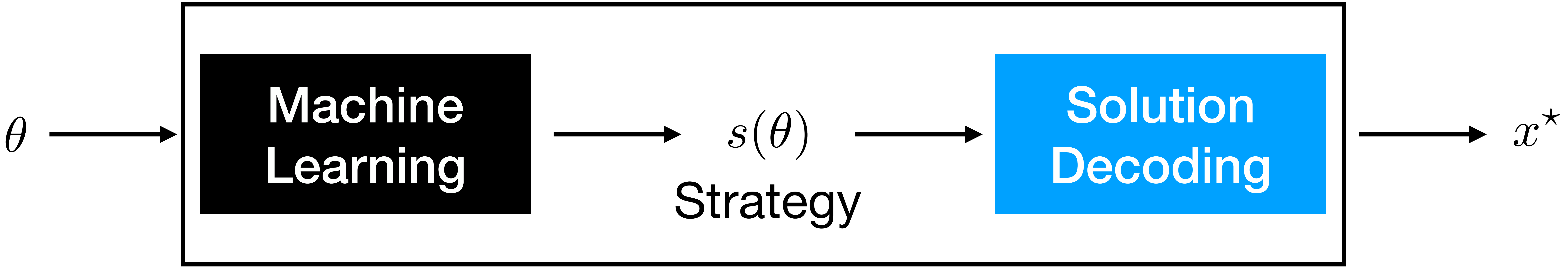
[Smith (1999)]
[Bello et al (2017]
[Vinyals et al (2017)]

Very small problems

Imprecise

Needs lots of “babysitting”

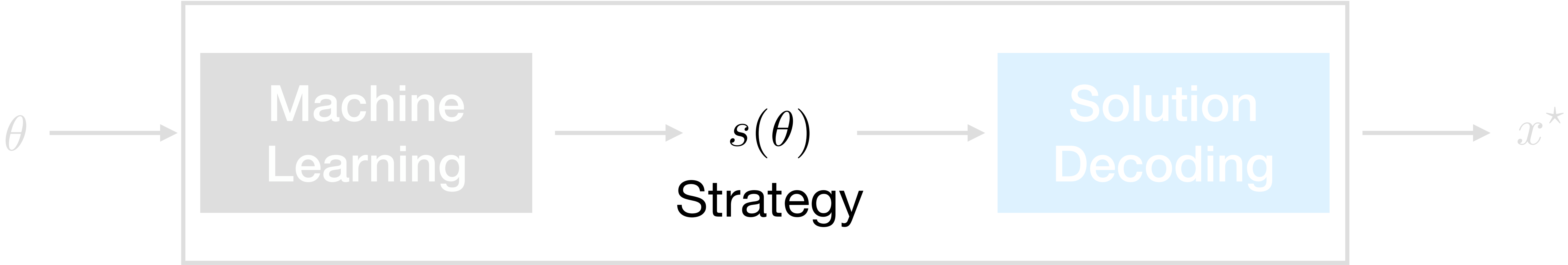
Machine learning optimizer



[The Voice of Optimization, Bertsimas and Stellato]
[Online Mixed-Integer Optimization in Milliseconds, Bertsimas and Stellato]

Strategies in optimization

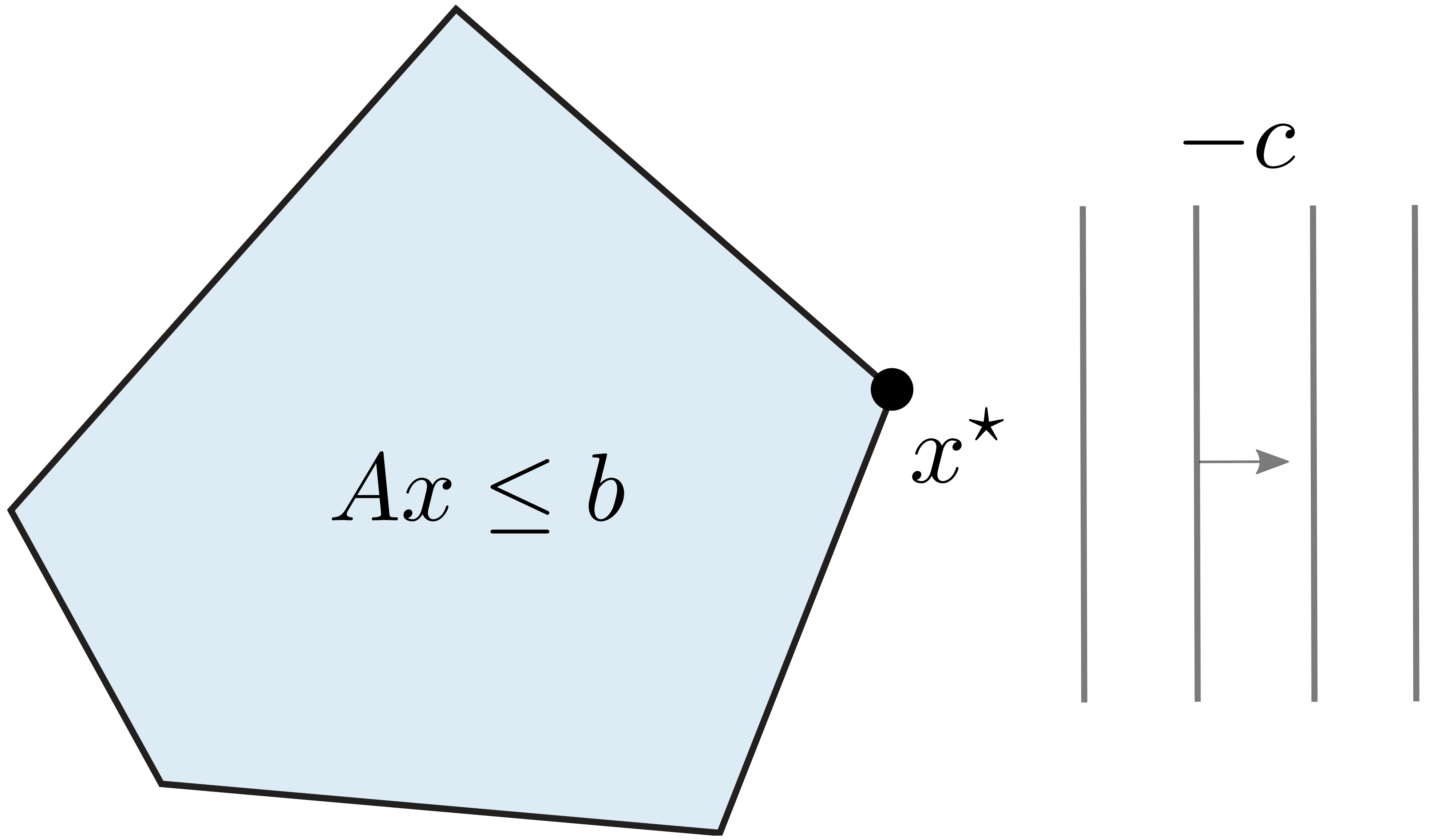
What is a strategy?



The complete information we need to efficiently compute the optimal solution

Parametric linear optimization

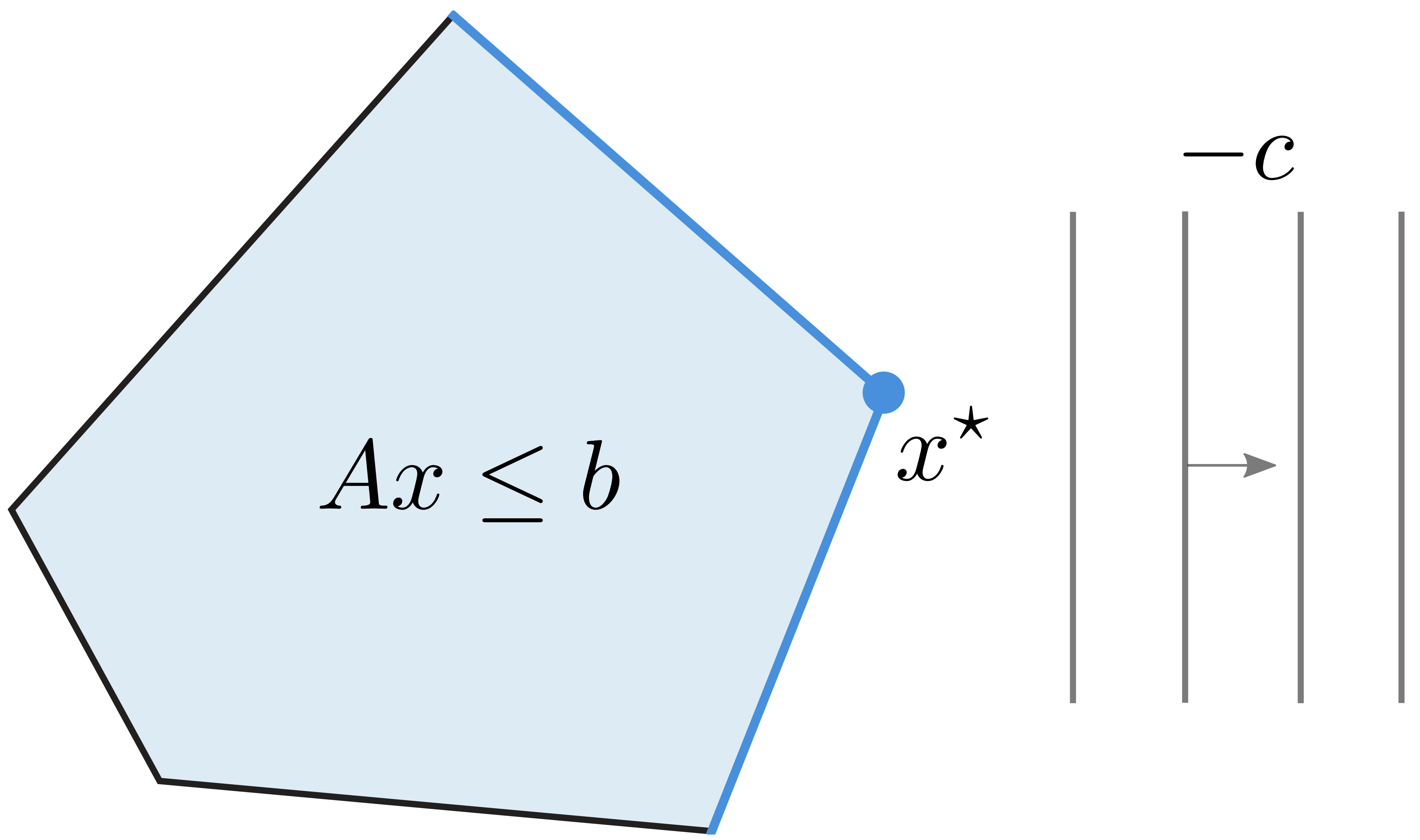
$$\begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A(\theta)x \leq b(\theta) \end{array}$$



How can we define a strategy?

Tight constraints in linear optimization

$$\mathcal{T}(\theta) = \{i \mid A_i(\theta)x^\star = b_i(\theta)\}$$



Strategies for linear optimization

$$s(\theta) = \mathcal{T}(\theta)$$

$|\mathcal{T}(\theta)| =$ # variables — if non-degenerate

$|\mathcal{T}(\theta)| \ll$ # constraints — in general

Computing the solution from the strategy

$s(\theta) \longrightarrow$ **Solution Decoding** $\longrightarrow x^\star$

Convex optimization

minimize	$c(\theta)^T x$
subject to	$A(\theta)x \leq b(\theta)$

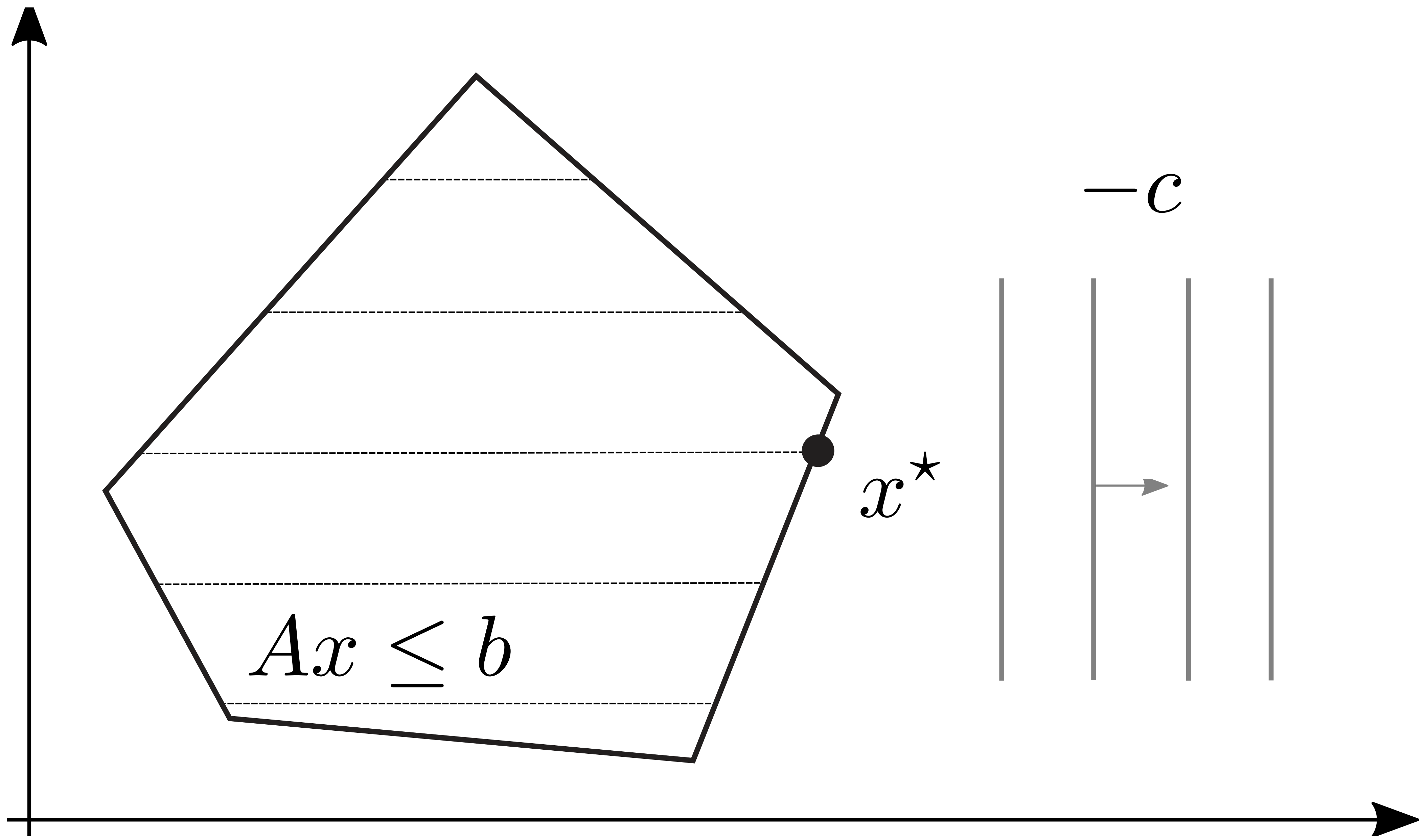
$\xrightarrow{s(\theta)}$

minimize	$c(\theta)^T x$
subject to	$A_i(\theta)x = b_i(\theta), \quad \forall i \in \mathcal{T}(\theta)$

KKT Linear system

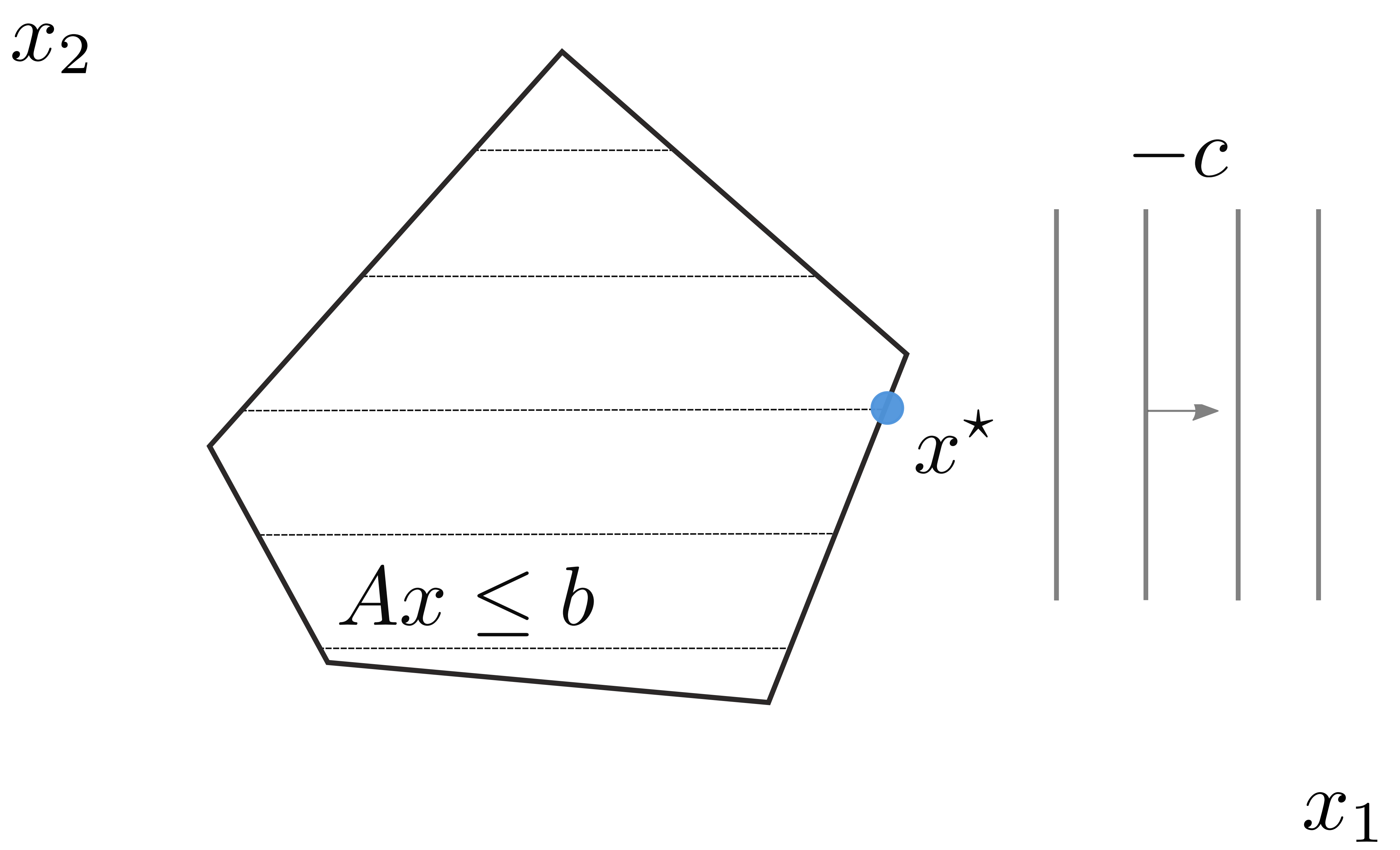
Parametric mixed-integer linear optimization

minimize	$c(\theta)^T x$	
subject to	$A(\theta)x \leq b(\theta)$	
	$x_{\mathcal{I}} \in \mathbf{Z}^d$	integers



How can we define a strategy?

Tight constraints are not enough

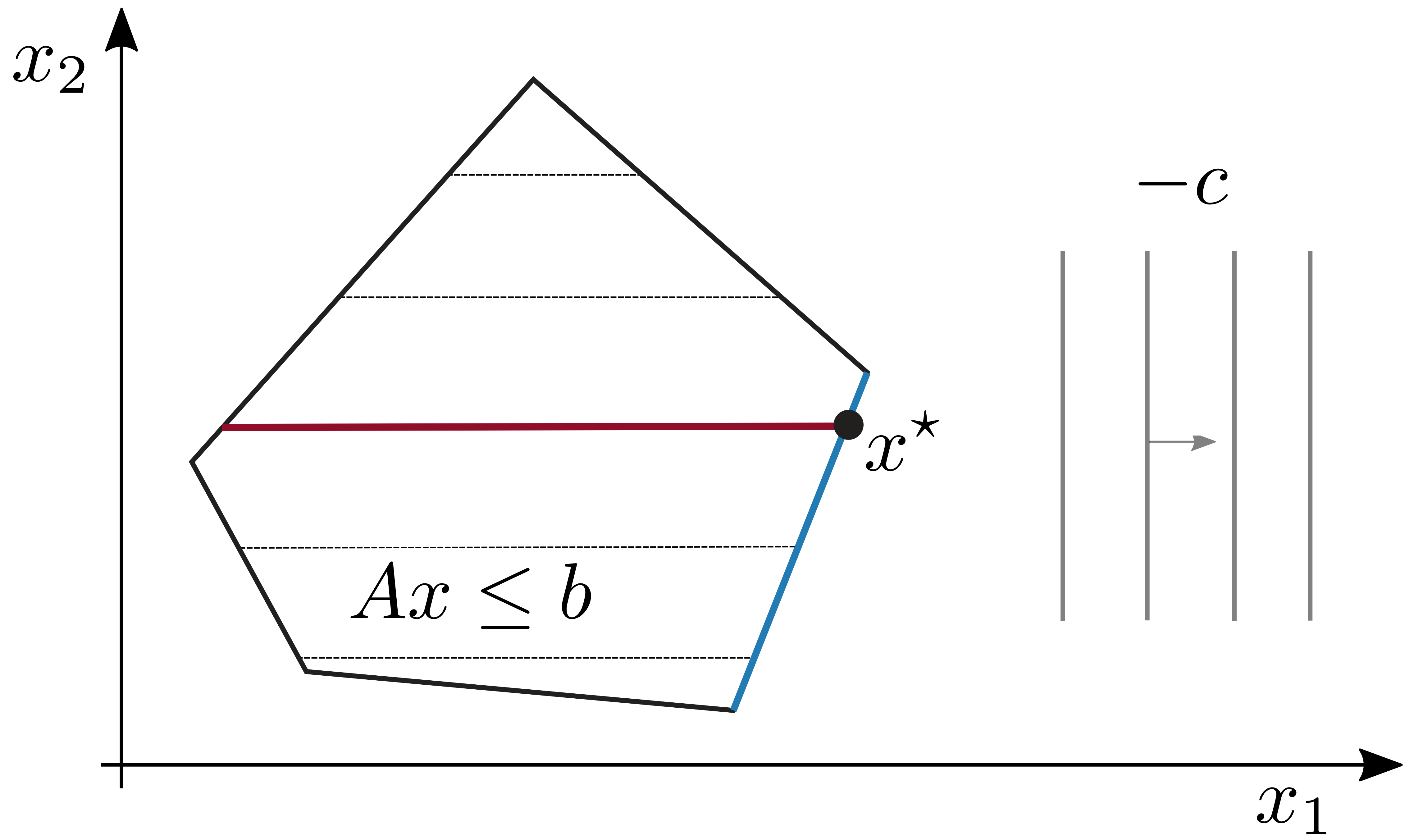


Strategies for mixed-integer optimization

$$s(\theta) = (\mathcal{T}(\theta), x^\star_{\mathcal{I}}(\theta))$$

Tight constraints

Integer variables



Computing the solution from the strategy

$s(\theta) \longrightarrow$ **Solution Decoding** $\longrightarrow x^\star$

Convex optimization

minimize	$c(\theta)^T x$
subject to	$A(\theta)x \leq b(\theta)$
	$x_{\mathcal{I}} \in \mathbf{Z}^d$

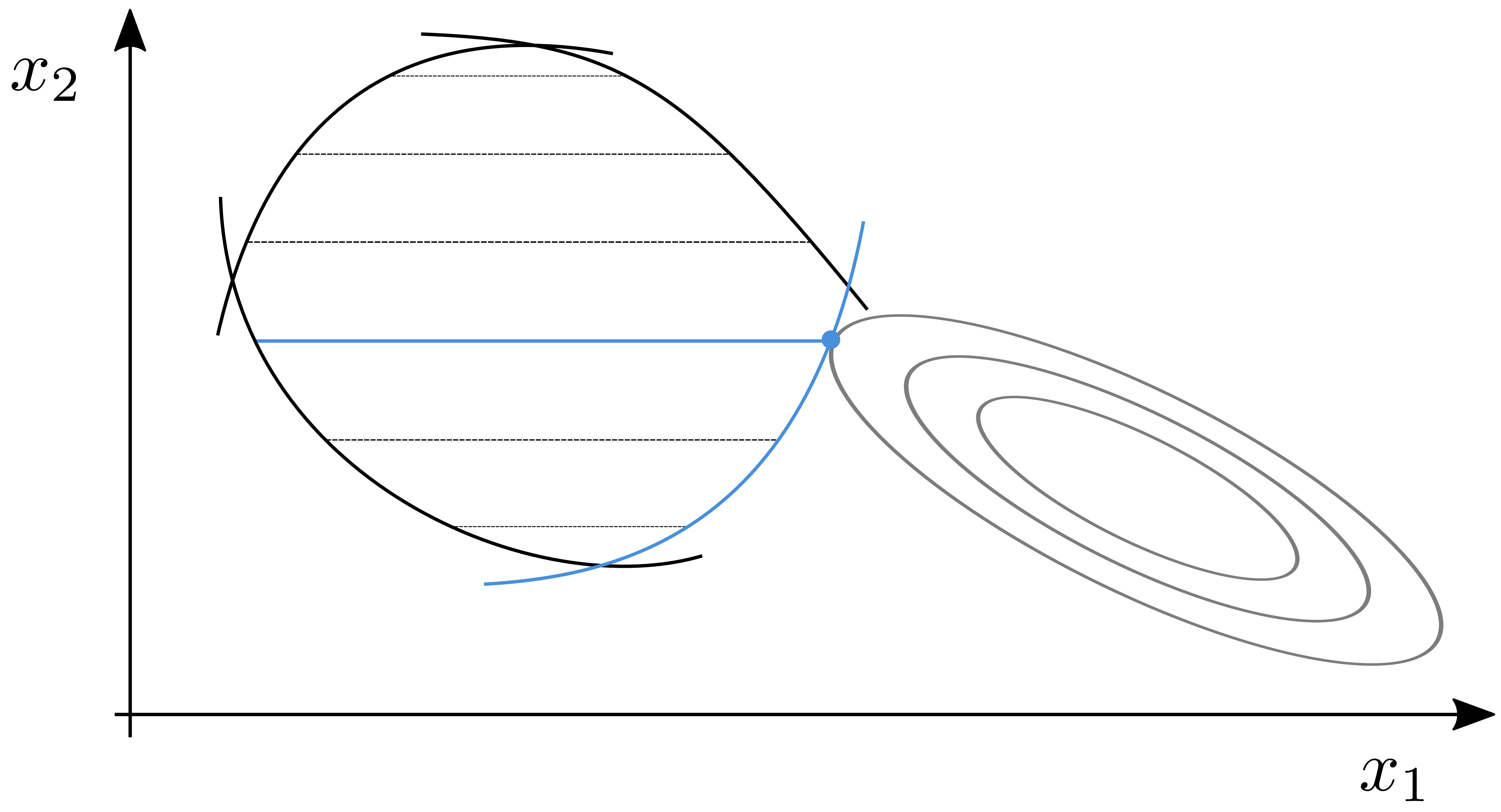
$$\xrightarrow{s(\theta)}$$

minimize	$c(\theta)^T x$
subject to	$A_i(\theta)x = b_i(\theta), \quad \forall i \in \mathcal{T}(\theta)$
	$x_{\mathcal{I}} = x^\star_{\mathcal{I}}(\theta)$

KKT Linear system

Mixed-integer convex optimization

$$\begin{array}{ll} \text{minimize} & f(x,\theta) \\ \text{subject to} & g(x,\theta) \le 0 \\ & z_{\mathcal{I}} \in \mathbf{Z}^d \end{array}$$



Same strategy definition

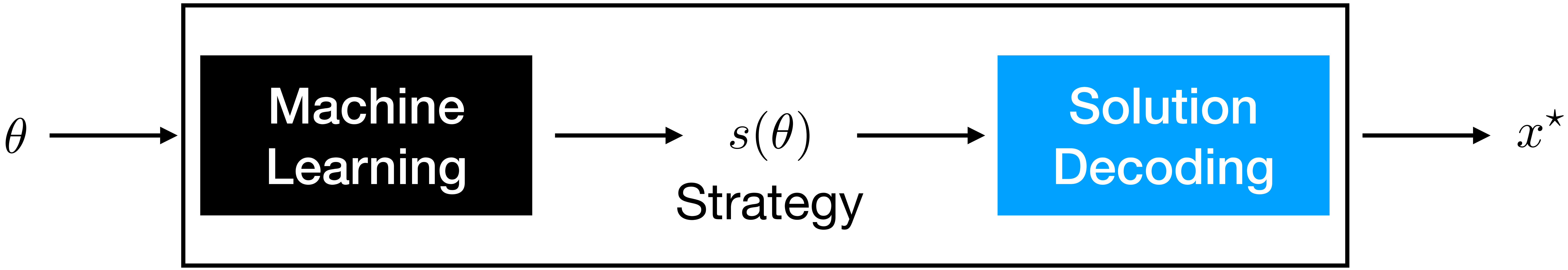
$$s(\theta) = (\mathcal{T}(\theta), x^{\star}_{\mathcal{I}}(\theta))$$

↓

How can we recover the solution?

Learning the strategies

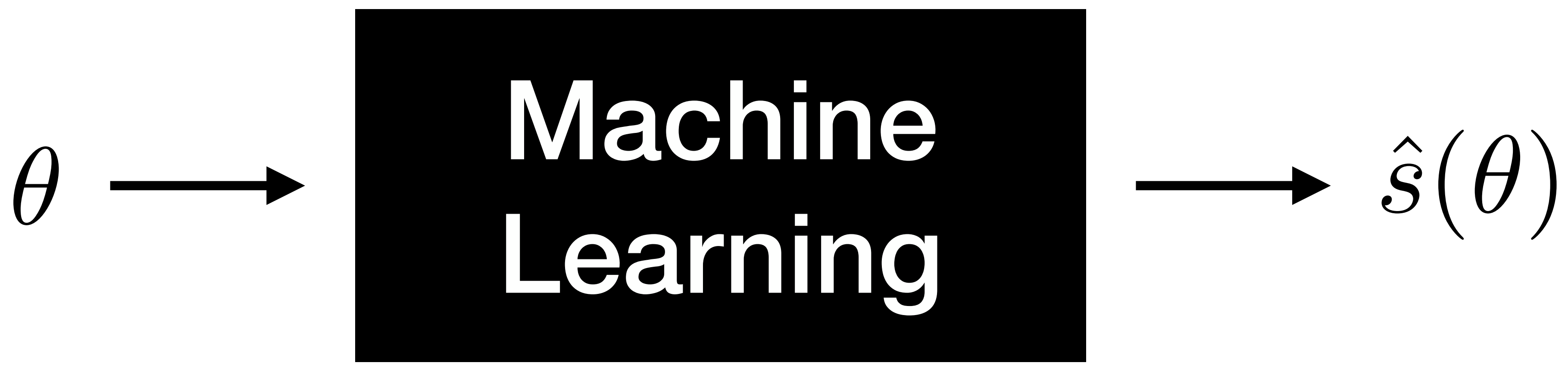
Predicting the strategies



N data $(\theta_i, s(\theta_i))$

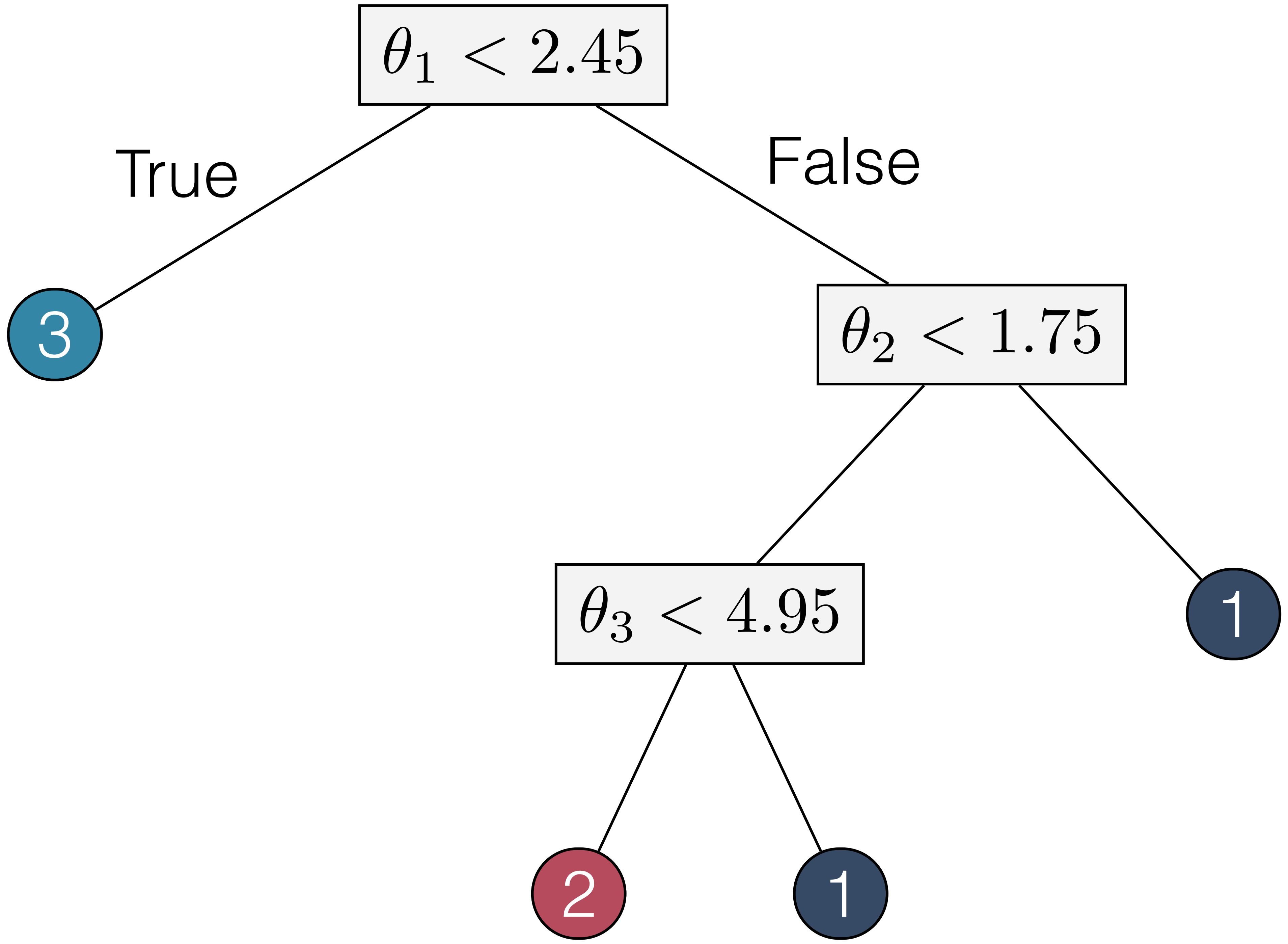
M labels (strategies) $\mathcal{S}$

Multiclass classification



Interpretable classifier

Decision Trees

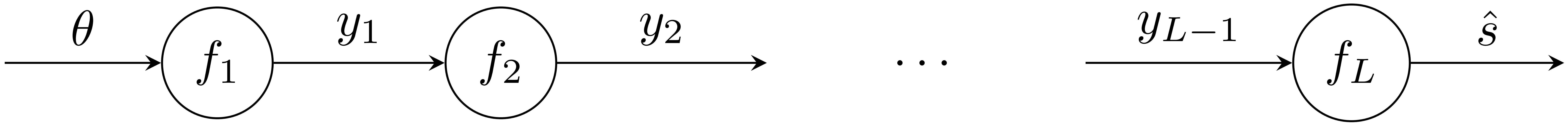


Features

- Easy to understand
- It works for small problems

[Optimal Classification Trees, Bertsimas and Dunn]

Neural network classifiers



Single layer

$$y_l = f(y_{l-1}) = (W_l y_{l-1} + b_l)_+$$

Output layer
(softmax)

$$\hat{s} = f(y_L) = \sigma(y_L), \quad \text{with} \quad (\sigma(x))_i = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}$$

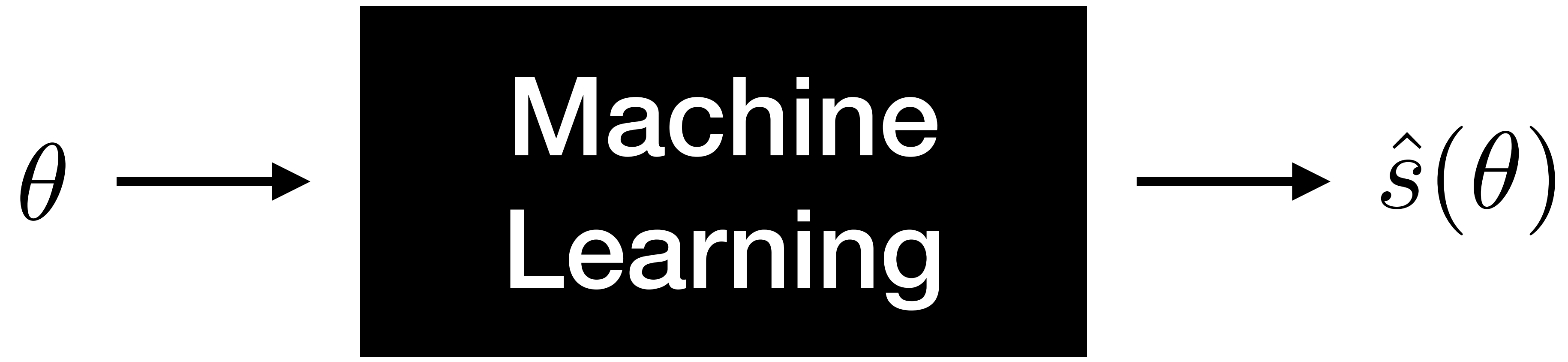
Features

- Hard to understand
- It works for large problems

Sampling the strategies

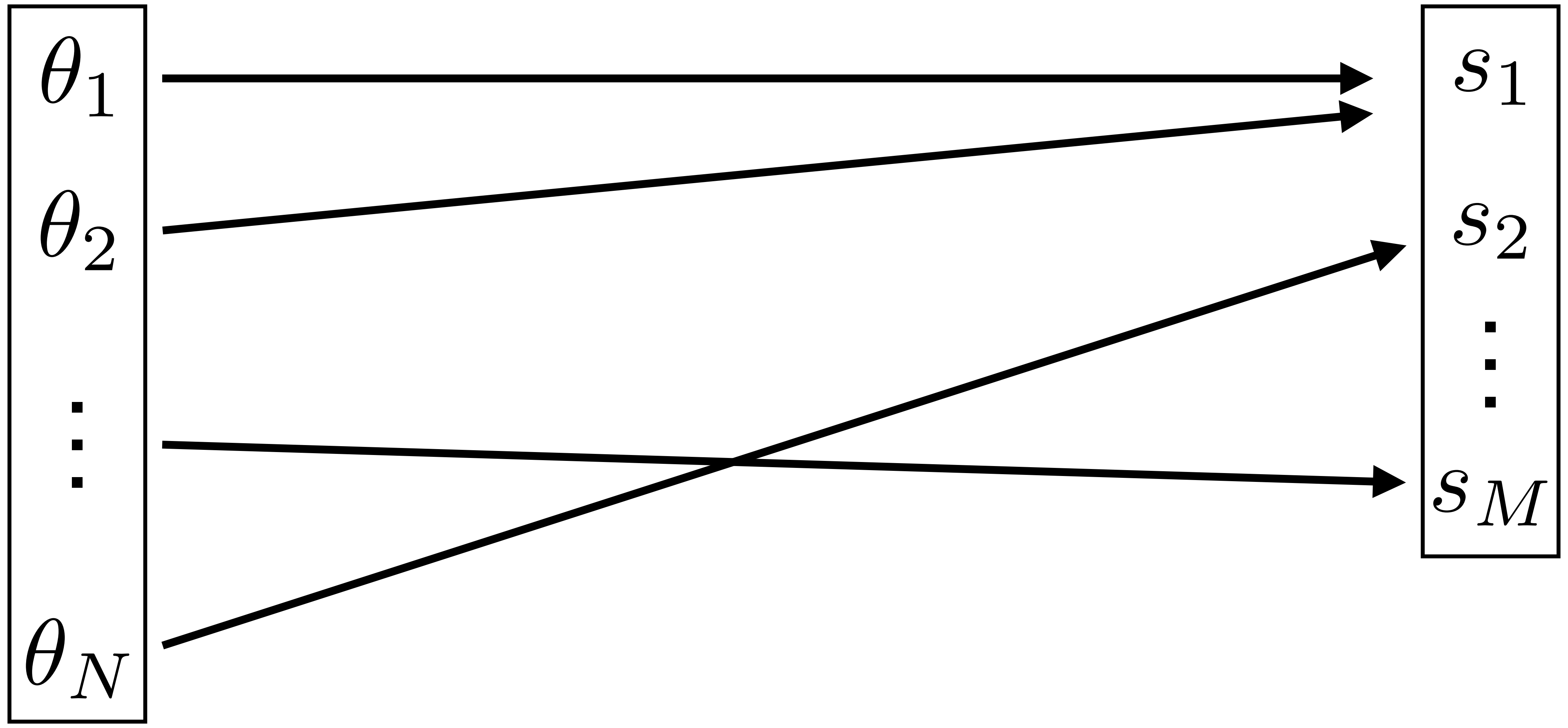
Have we seen enough data?

Multiclass classification



N data $(\theta_i, s(\theta_i))$

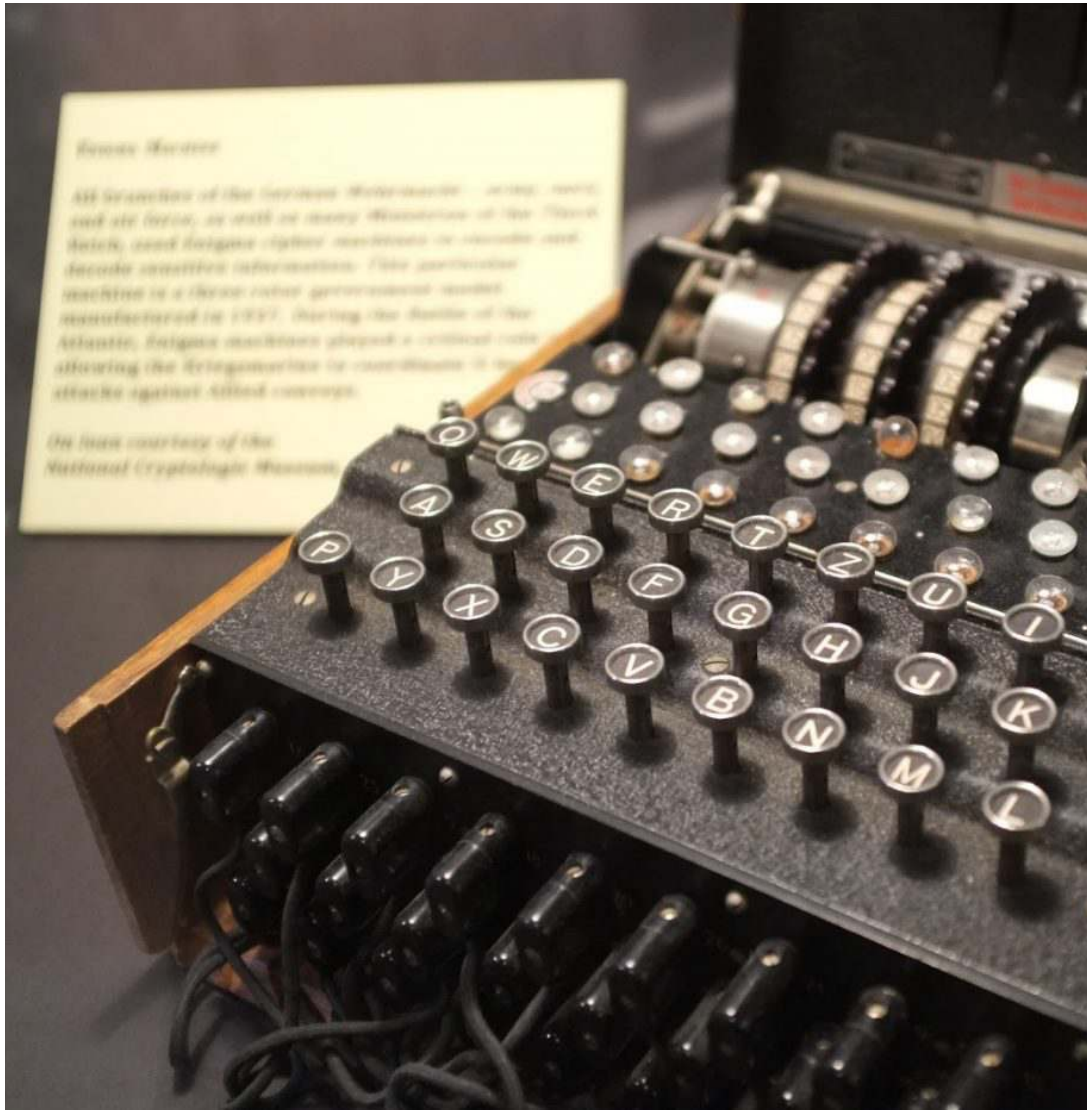
M labels (strategies) $\mathcal{S}$



What happens with θ_{N+1}?

Alan Turing

Already worked on this…



Good-Turing estimator

strategies appeared once

$$GT = \frac{N_1}{N} \approx \mathbf{P}(s(\theta_{N+1}) \notin \mathcal{S}(\Theta_N))$$

samples

Probability of unseen strategies

Concentration bound (confidence β)

$$\mathbf{P}(s(\theta_{N+1}) \notin \mathcal{S}(\Theta_N)) \leq GT + C\sqrt{(1/N)\ln(3/\beta)}$$

Sample until $\leq \epsilon$

Example

$N = 15$
$M = 5$

→

s_1 6 times
s_2 3 times
s_3 1 time
s_4 3 times
s_5 2 times

→

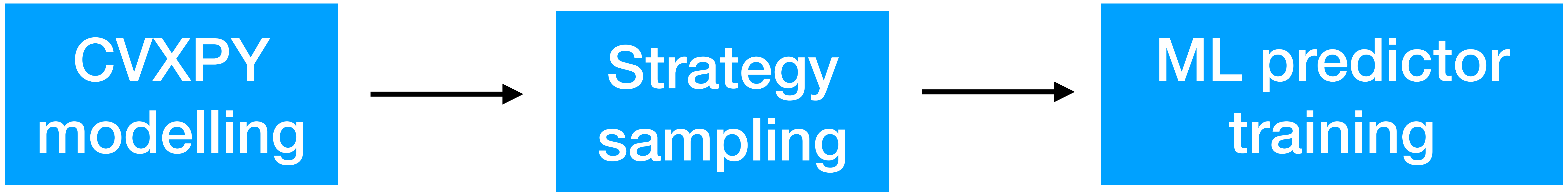
$GT = 1/15$

[On the Convergence Rate of Good-Turing Estimators, McAllester and Schapire]

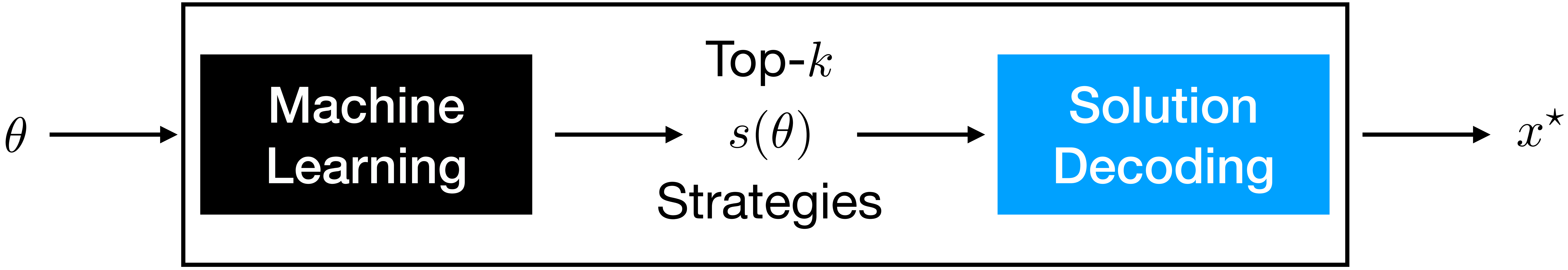
MLOPT: Machine Learning Optimizer

`github.com/bstellato/mlopt`

Offline learning



Fast online predictions



Examples

Inventory management

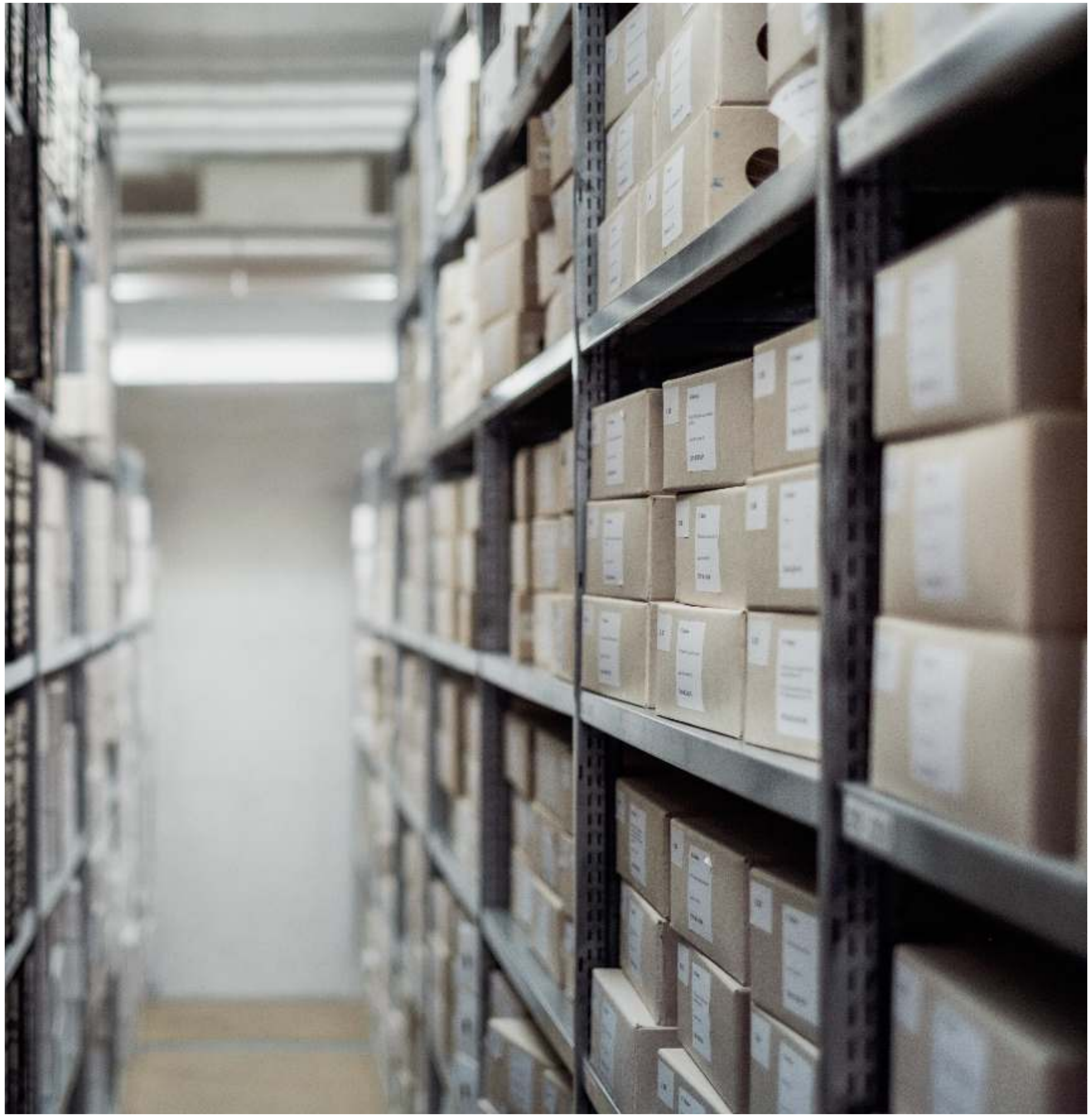
minimize	$\sum_{t=0}^{T-1} h(x_t) + o(u_t)$
subject to	$x_{t+1} = x_t + u_t - d_t$
	$x_0 = x_{\text{init}}$
	$0 \leq u_t \leq M$

inventory

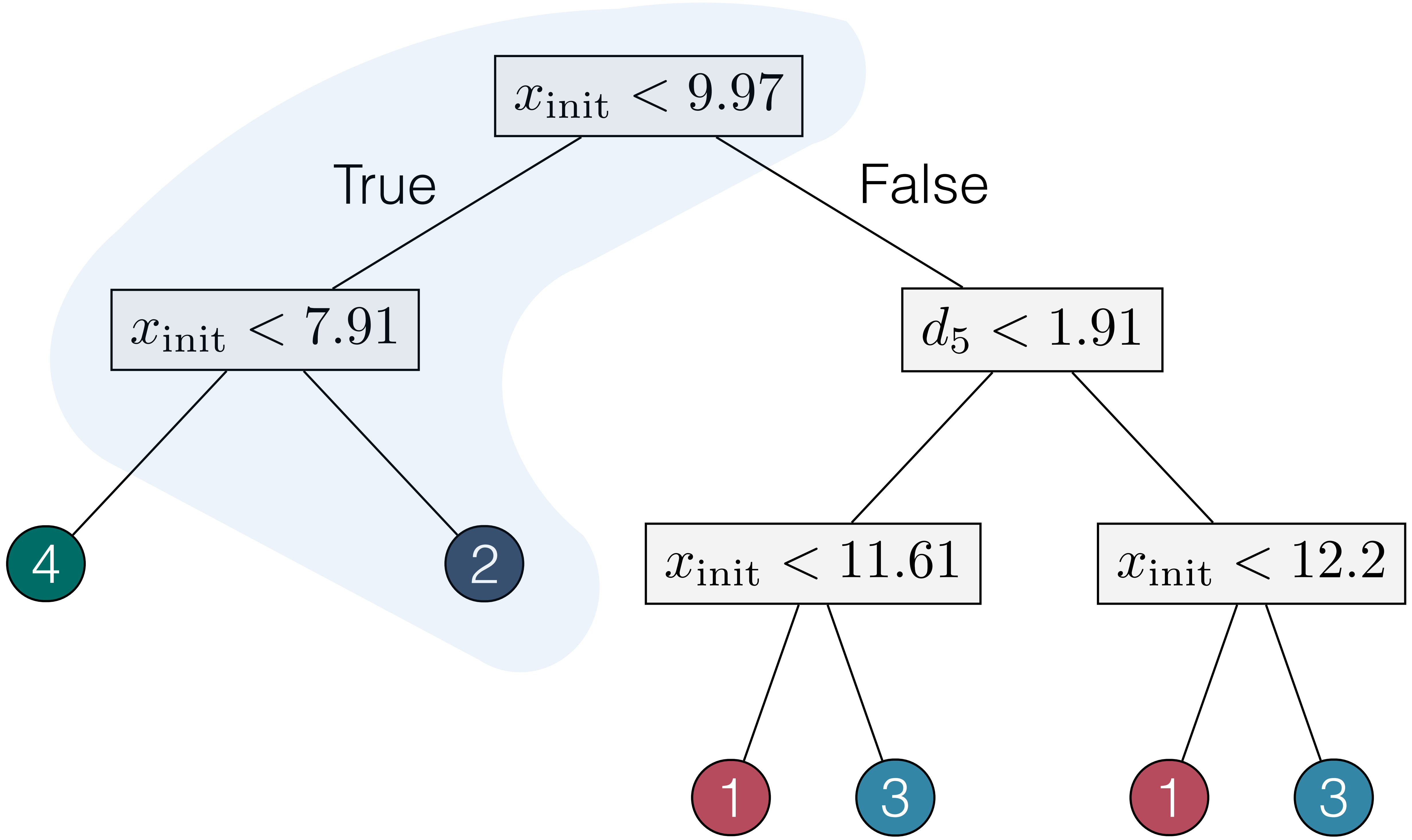
demand

order

parameters



Inventory management strategies



minimize $\sum_{t=0}^{T-1} h(x_t) + o(u_t)$

subject to $x_{t+1} = x_t + u_t - d_t$

$x_0 = x_{\text{init}}$

$0 \leq u_t \leq M$

Strategy 4

$u_t = 0 \qquad t \leq 3$

$0 \leq u_t \leq M \qquad t > 3$

Strategy 2

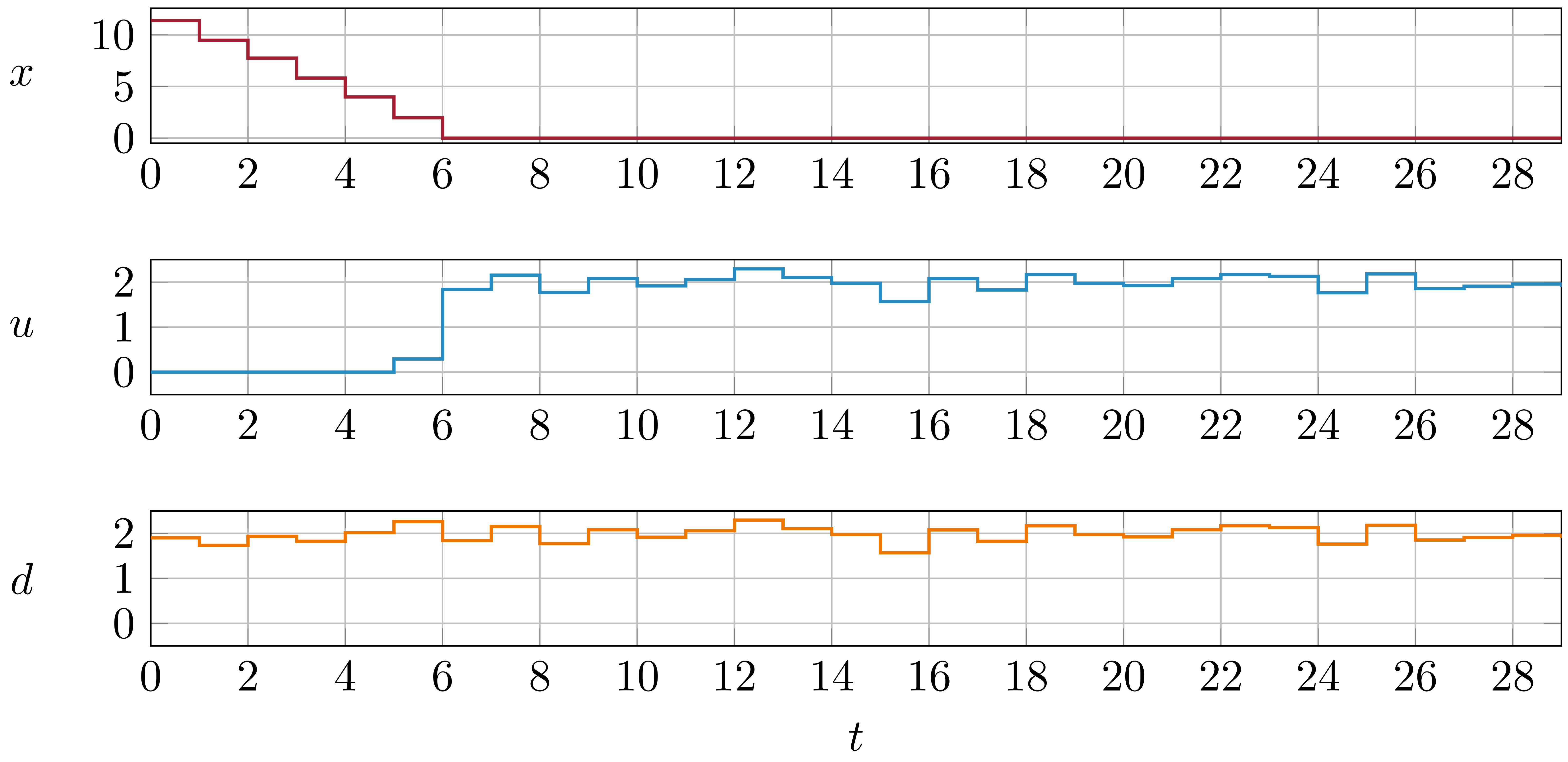
$u_t = 0 \qquad t \leq 4$

$0 \leq u_t \leq M \qquad t > 4$

Inventory management trajectory

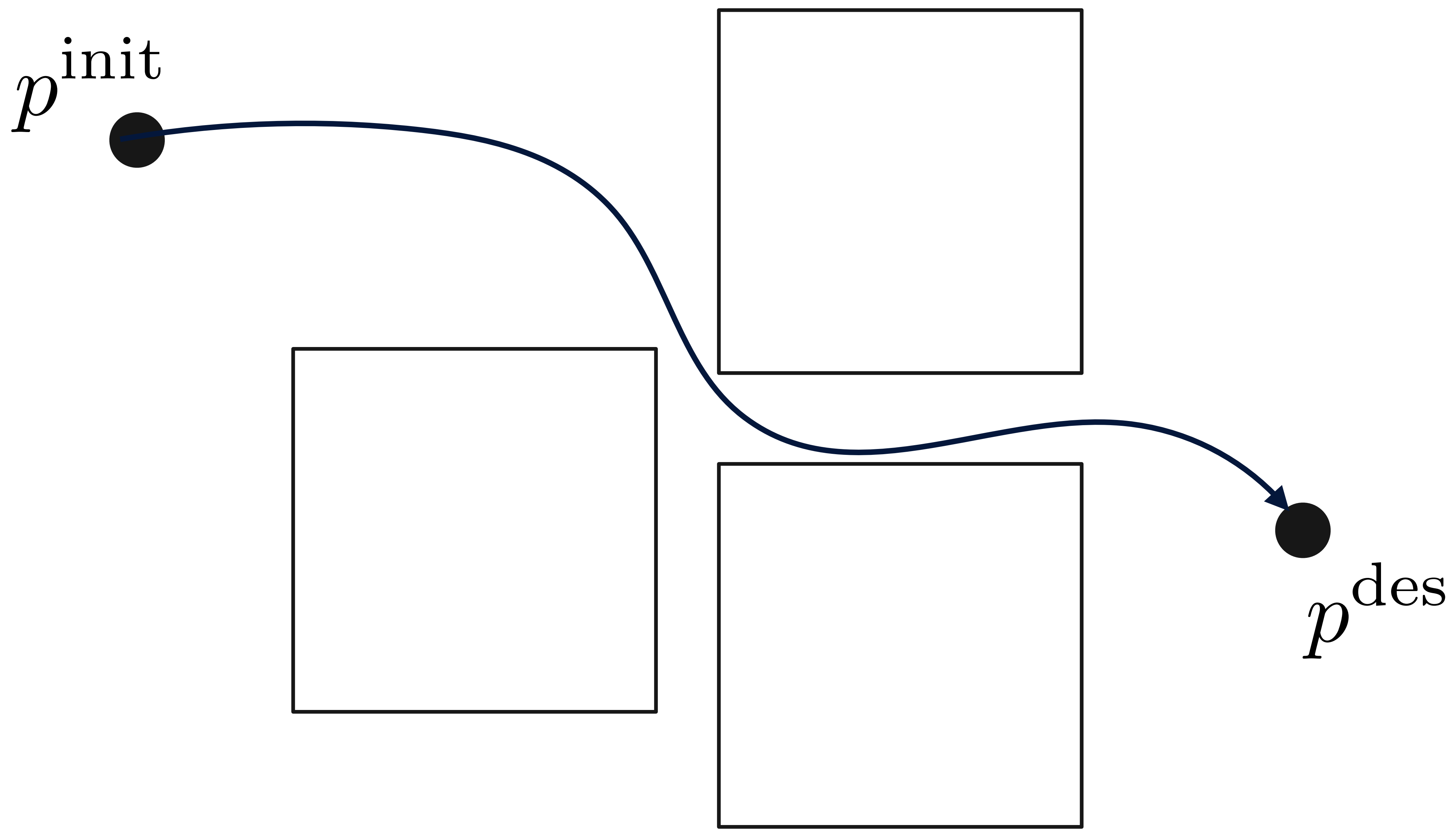
Strategy 2

$$\begin{aligned} u_t &= 0 & t &\leq 4 \\ 0 \leq u_t &\leq M & t &> 4 \end{aligned}$$



Example

Motion planning with obstacles



p_t position $\in \mathbf{R}^d$
v_t velocity $\in \mathbf{R}^d$

p^{init} initial position
v^{init} initial velocity

p^{des} desired position

Obstacles

Obstacle i is a box $[\underline{o}^i, \overline{o}^i]$

Motion planning formulation

minimize $\|p_T - p^{\text{des}}\|_2^2 + \sum_{t=0}^{T-1} \|p_t - p^{\text{des}}\|_2^2 + \gamma \|u_t\|_2^2$

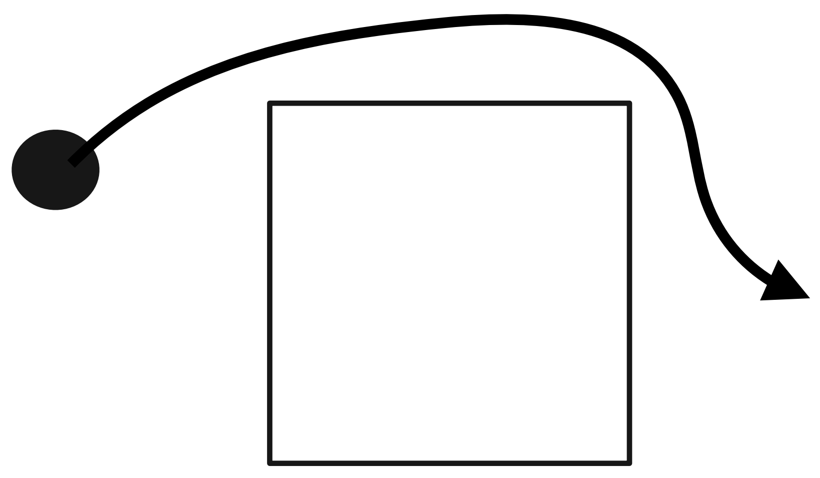
subject to

$$(p_{t+1}, v_{t+1}) = A(p_t, v_t) + Bu_t$$
$$p_0 = p^{\text{init}}, \quad v_0 = v^{\text{init}}$$

Dynamics

$$\overline{o}^i - M\overline{\delta}_t^i \leq p_t \leq \underline{o}^i + M\underline{\delta}_t^i, \quad i = 1, \ldots, n_{\text{obs}}$$
$$\mathbf{1}^T \underline{\delta}_t^i + \mathbf{1}^T \overline{\delta}_t^i \leq 2d - 1$$

Obstacle avoidance



$$\overline{\delta}_t^i, \underline{\delta}_t^i \in \{0, 1\}^d, \qquad i = 1, \ldots, n_{\text{obs}}$$

Motion planning with obstacles

Worst-case timings

$n_{\text{obstacles}}$	n_{var}	n_{constr}	$t_{\max}$ MLOPT [s]	$t_{\max}$ Gurobi [s]	$t_{\max}$ Gurobi heuristic [s]
2	1135	3773	0.4145	2.3776	2.2962
4	1615	10133	0.1878	11.8172	8.1443
6	2095	20333	0.3173	33.7869	11.5292
8	2575	34373	0.2235	392.3073	128.4948
10	3055	52253	0.2896	773.1476	206.4520

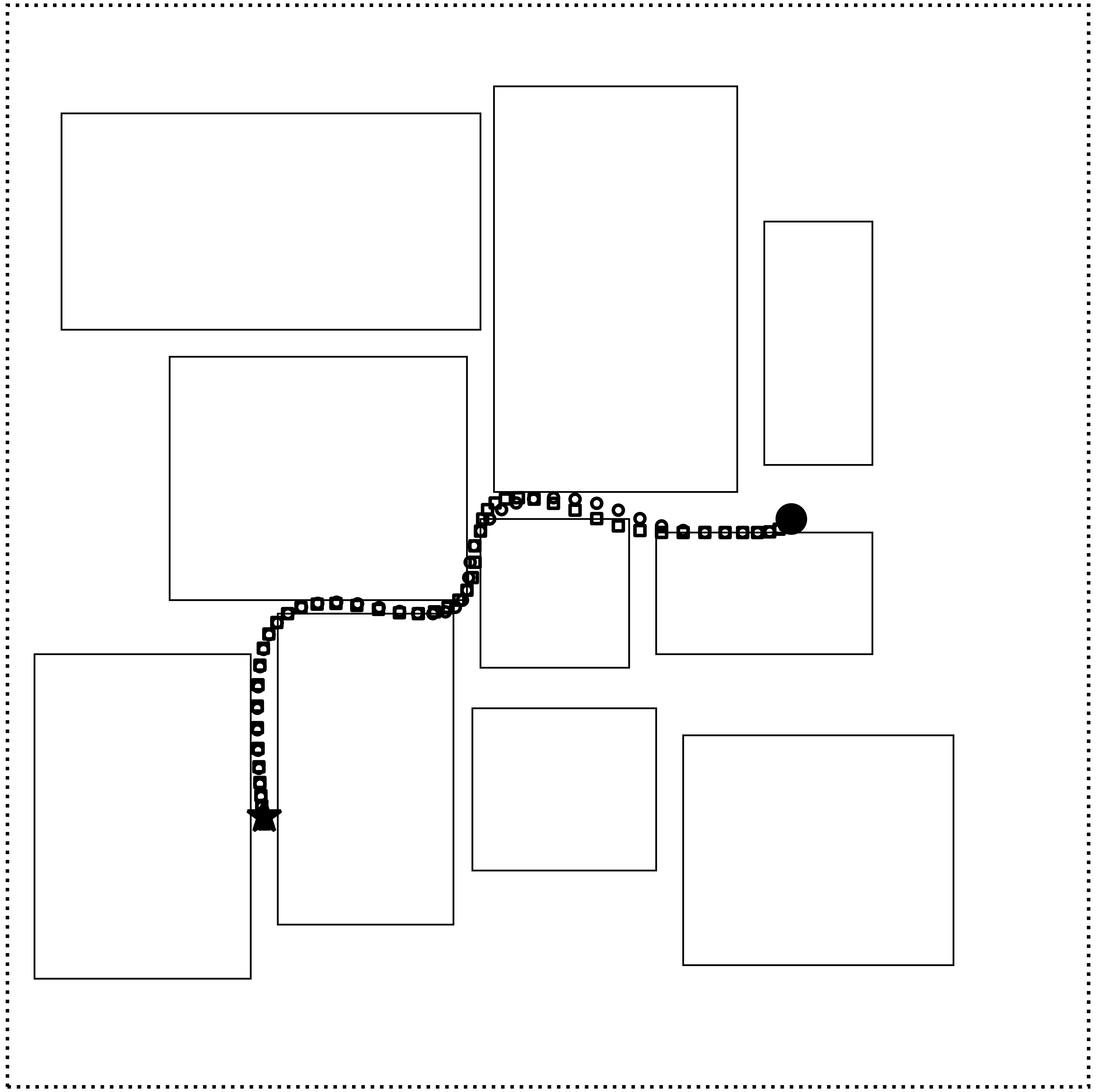
2600x speedups

[Online Mixed-Integer Optimization in Milliseconds, Bertsimas and Stellato]

Motion planning with obstacles

Circles
optimal

Squares
MLOPT



Learning strategies in parametric optimization

Benefits

- Extremely fast
- Simple online method for nonconvex optimization
- It learns from your pool of problems

Downsides

- No optimality guarantees
- Relies on many offline solutions (expert demonstrations)

Future directions

- Better NN architectures
- Optimality guarantees
- Reinforcement learning when we do not have offline solutions

Data-driven algorithms

Today, we learned recent research on data-driven algorithms:

- **Learning heuristics** in branch and bound search (global algorithm)
- **Learning strategies** in parametric optimization (heuristic algorithm)

Many more exciting directions

Differentiable optimization layers, reinforcement learning in optimization, learning-augmented first order methods, …

[CS159 Caltech, https://sites.google.com/view/cs-159-spring-2020/]

Next lecture

- Course recap and conclusions