ORF307 – Optimization

7. Linear optimization

Bartolomeo Stellato — Spring 2024

Ed Forum

- I was wondering if there would be any chance to delve further into portfolio theory for this class.
- How does the inclusion of a risk-free asset impact the overall optimization strategy and the resulting asset allocation?

Recap

Least squares with equality constraints

The (linearly) constrained least squares problem is

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & Cx = d \end{array}$$

objective function

equality constraints

Problem data

- $m \times n$ matrix A, m-vector b
- $p \times n$ matrix C, p-vector d

Definitions

x is *feasible* if $Cx = d$

$x^\star$ is a *solution* if

- $Cx^\star = d$
- $\|Ax^\star - b\|^2 \leq \|Ax - b\|^2$ for any x satisfying $Cx = d$

Interpretations

- Combine solving linear equations with least squares.
- Like a bi-objective least squares with ∞ weight on second objective, $\|Cx - d\|^2$.

Portfolio optimization

How shall we choose the portfolio weight vector w?

Goals

High (mean) return $\mathbf{avg}(r)$

Low risk $\mathbf{std}(r)$

Data

- We know **realized asset returns** but not future ones
- **Optimization.** We choose w that would have worked well in the past
- **True goal.** Hope it will work well in the future (just like data fitting)

Portfolio optimization

As constrained least squares

$$\begin{array}{ll} \text{minimize} & \|Rw - \rho\mathbf{1}\|^2 \\ \text{subject to} & \begin{bmatrix} \mathbf{1}^T \\ \mu^T \end{bmatrix} w = \begin{bmatrix} 1 \\ \rho \end{bmatrix} \end{array}$$

μ is the n-vector of average returns per asset

$$\begin{aligned} \mathbf{avg}(r) &= (1/T)\mathbf{1}^T(Rw) \\ &= (1/T)(R^T\mathbf{1})^T w = \mu^T w \end{aligned}$$

Solution via KKT linear system

$$\begin{bmatrix} 2R^TR & \mathbf{1} & \mu \\ \mathbf{1}^T & 0 & 0 \\ \mu^T & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 2\rho T\mu \\ 1 \\ \rho \end{bmatrix}$$

Optimal portfolios

Rewrite right-hand side

$$\begin{bmatrix} 2\rho T\mu \\ 1 \\ \rho \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \rho \begin{bmatrix} 2T\mu \\ 0 \\ 0 \end{bmatrix}$$

Two fund theorem

Optimal portfolio w is an affine function of ρ

$$\begin{bmatrix} w \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 2R^T R & \mathbf{1} & \mu \\ \mathbf{1}^T & 0 & 0 \\ \mu^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \rho \begin{bmatrix} 2R^T R & \mathbf{1} & \mu \\ \mathbf{1}^T & 0 & 0 \\ \mu^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2T\mu \\ 0 \\ 1 \end{bmatrix}$$

We can rewrite the first n-components as the combination of two portfolios (funds)

$$w = w_0 + \rho v$$

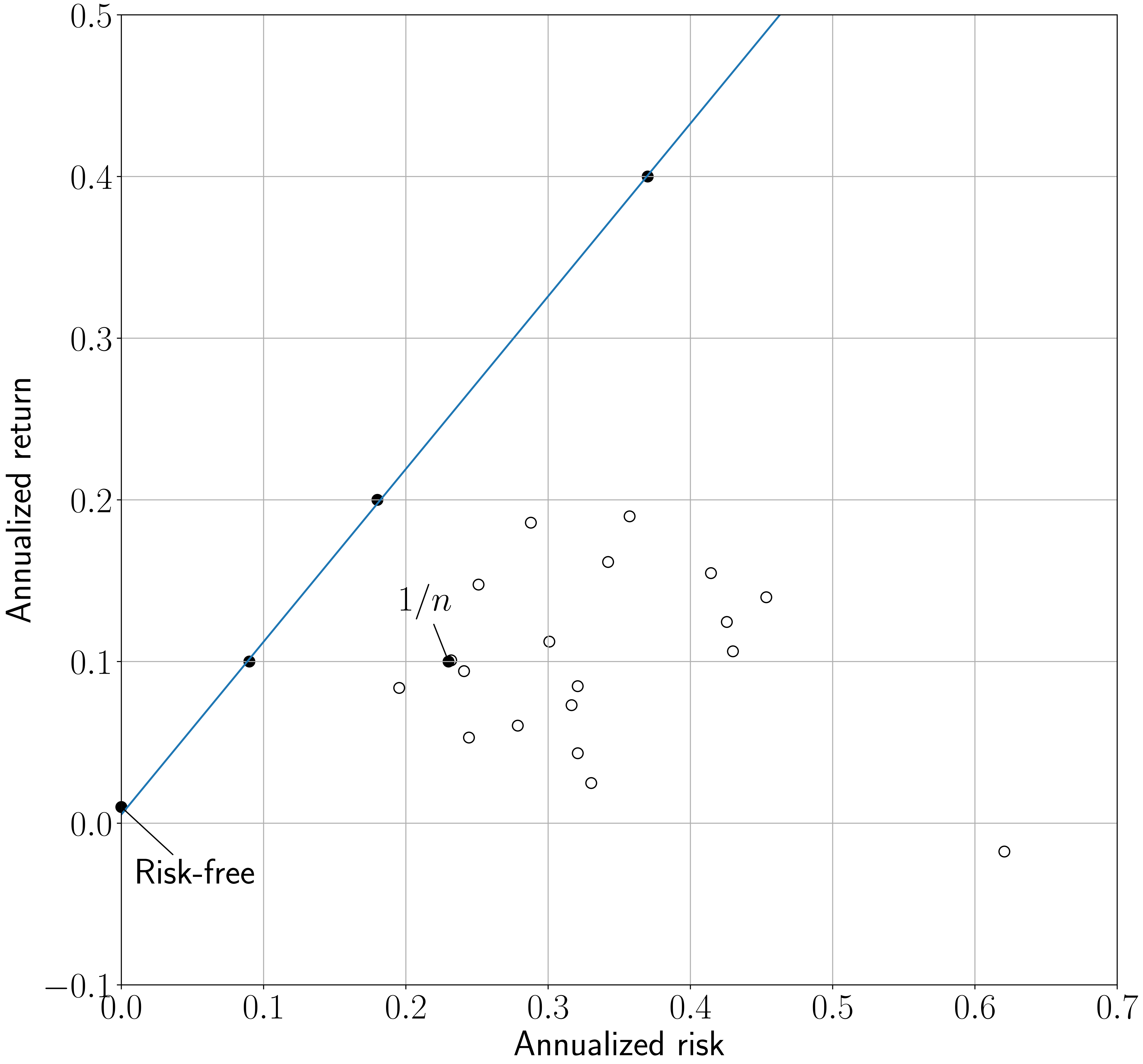
Risk-free ($\rho = 0$)

Other optimal portfolio

Example

20 assets over 2000 days (past)

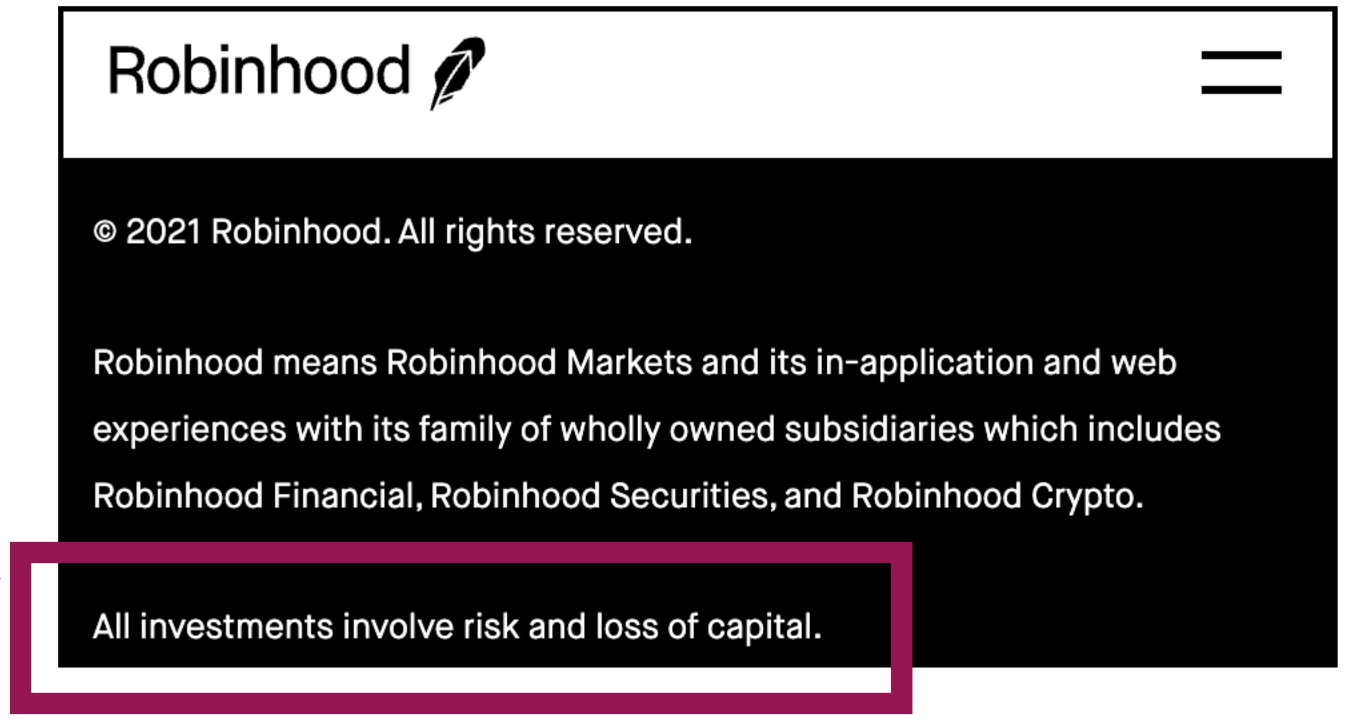
- Optimal portfolios on a **straight line**
- Line starts at risk-free portfolio ($\rho = 0$)
- $1/n$ much better than single portfolios



The big assumption

Future returns will look like past ones

- You are warned this is false, every time you invest
- It is often reasonable
- During crisis, market shifts, other big events not true



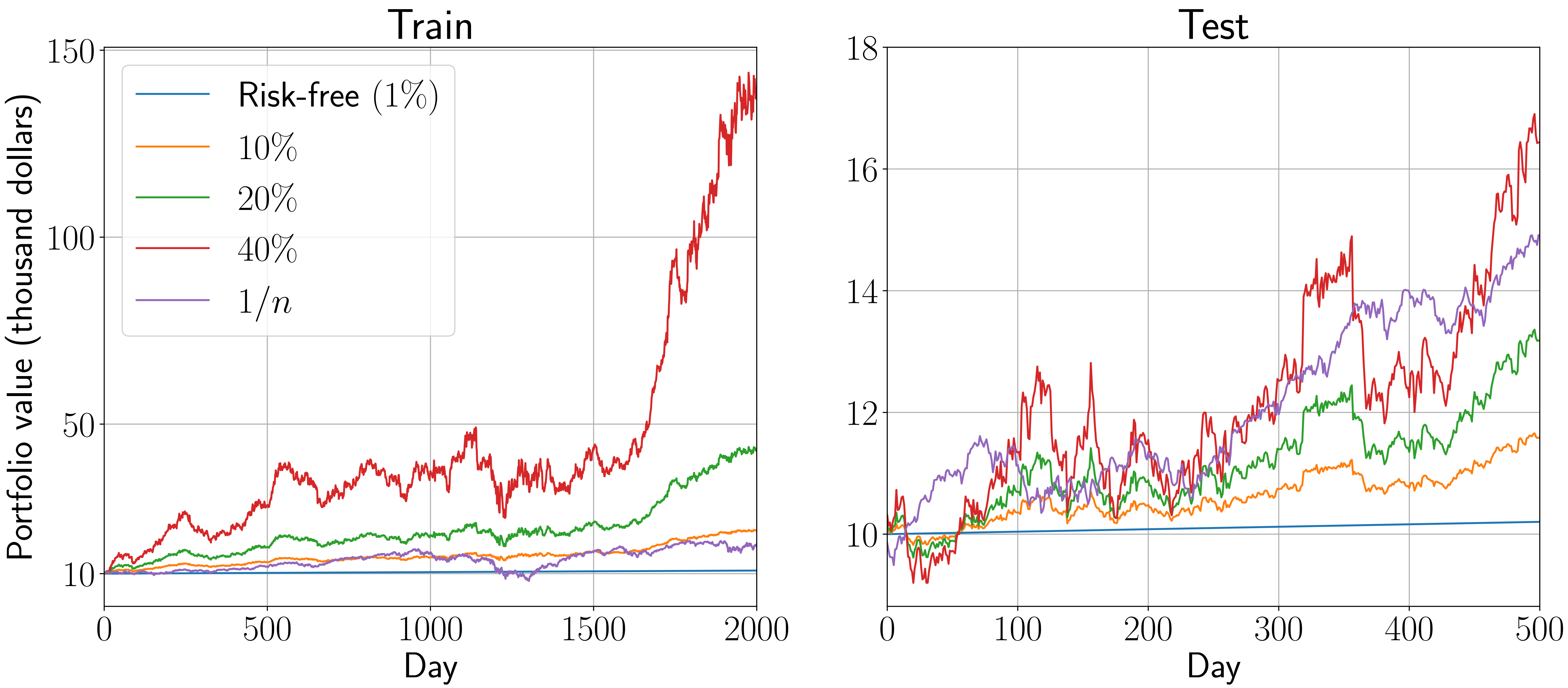
If assumption holds (even approximately), a good w on past returns leads to good future (unknown) returns

Example

- Pick w based on last 2 years of returns
- Use w during next 6 months

Total portfolio value

	Return		Risk		
	Train	Test	Train	Test	Leverage
Risk-free (1%)	0.01	0.01	0.00	0.00	1.00
10%	0.10	0.08	0.09	0.07	1.96
20%	0.20	0.15	0.18	0.15	3.03
40%	0.40	0.30	0.37	0.31	5.48
$1/n$	0.10	0.21	0.23	0.13	1.00



Build your quantitative hedge fund

Rolling portfolio optimization

For each period t, find weight w_t using L past returns

$$r_{t-1}, \ldots, r_{t-L}$$

Variations

- Update w every K periods (monthly, quarterly, ...)
- Add secondary objective $\lambda \|w_t - w_{t-1}\|^2$ to discourage turnover, reduce transaction cost
- Add logic to detect when the future is likely to not look like the past
- Add “signals” that predict future return of assets (Twitter sentiment analysis)

Today’s lecture

Linear optimization

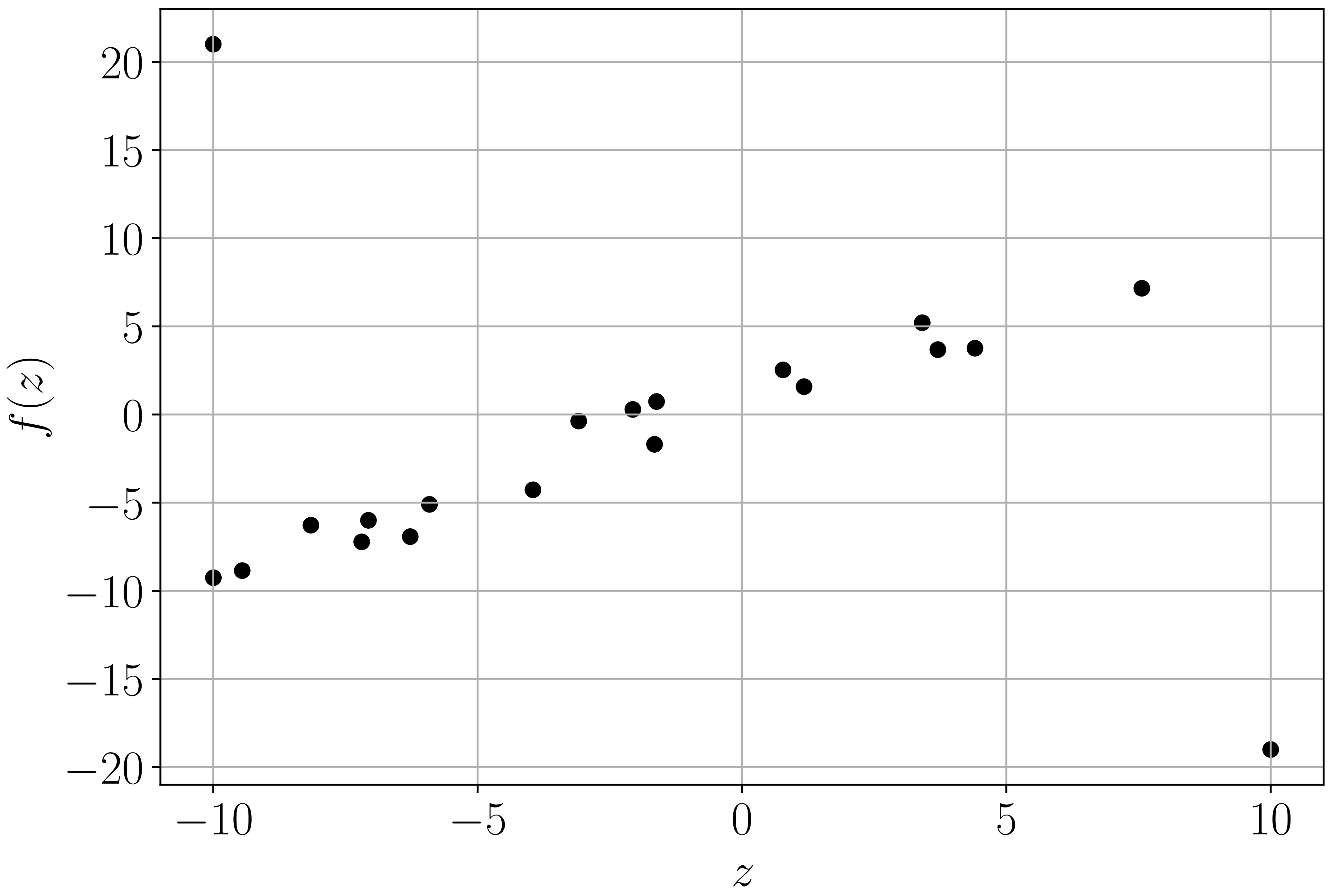
- Some simple examples
- Linear optimization
- Special cases
- Standard form
- Software and solution methods

Some simple examples

Data-fitting example

Fit a linear function $f(z) = x_1 + x_2 z$ to m data points (z_i, f_i):

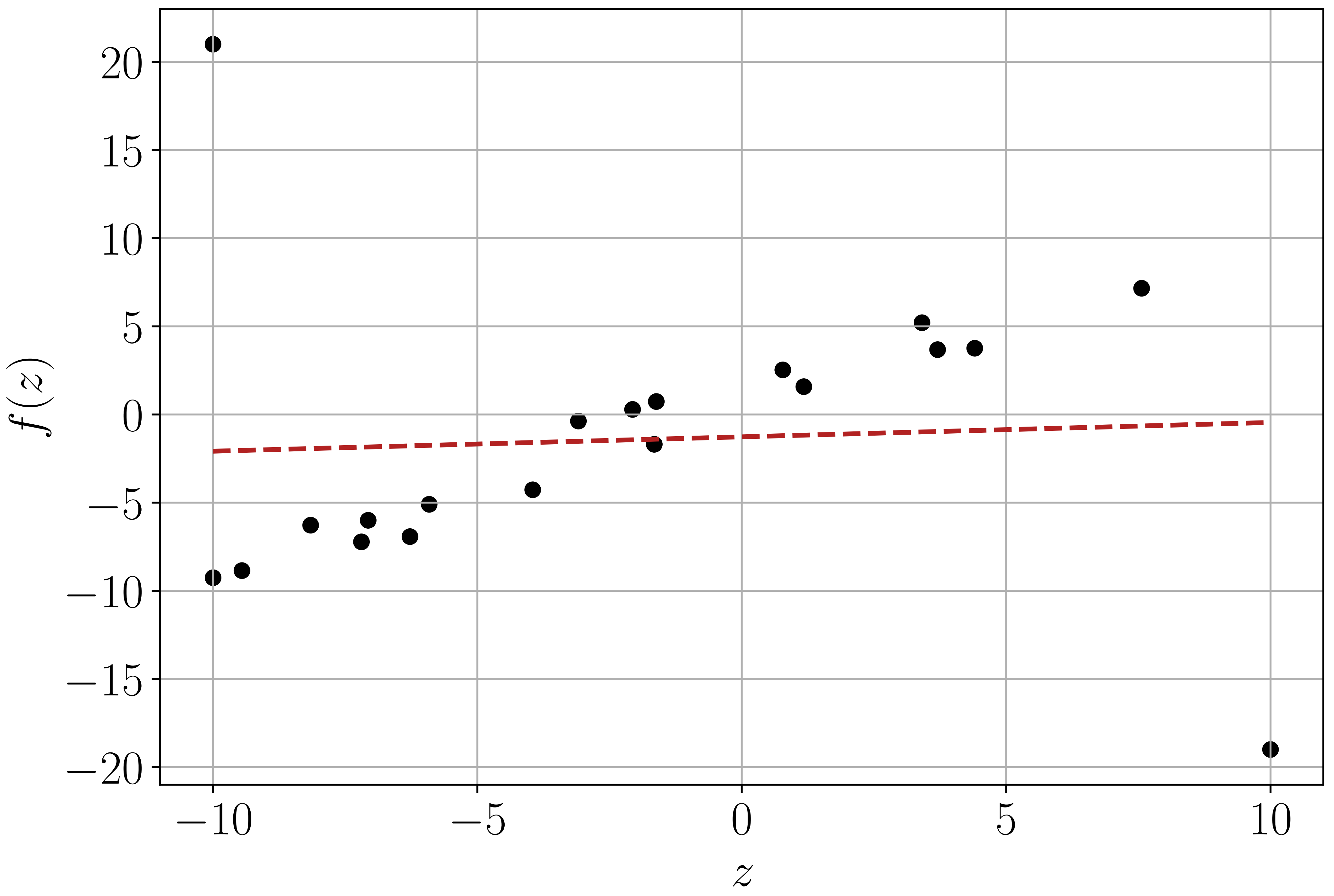
Approximation problem $Ax \approx b$ where $\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_{b}$



Data-fitting example

Fit a linear function $f(z) = x_1 + x_2 z$ to m data points (z_i, f_i):

Approximation problem $Ax \approx b$ where $\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_{b}$



Least squares way:

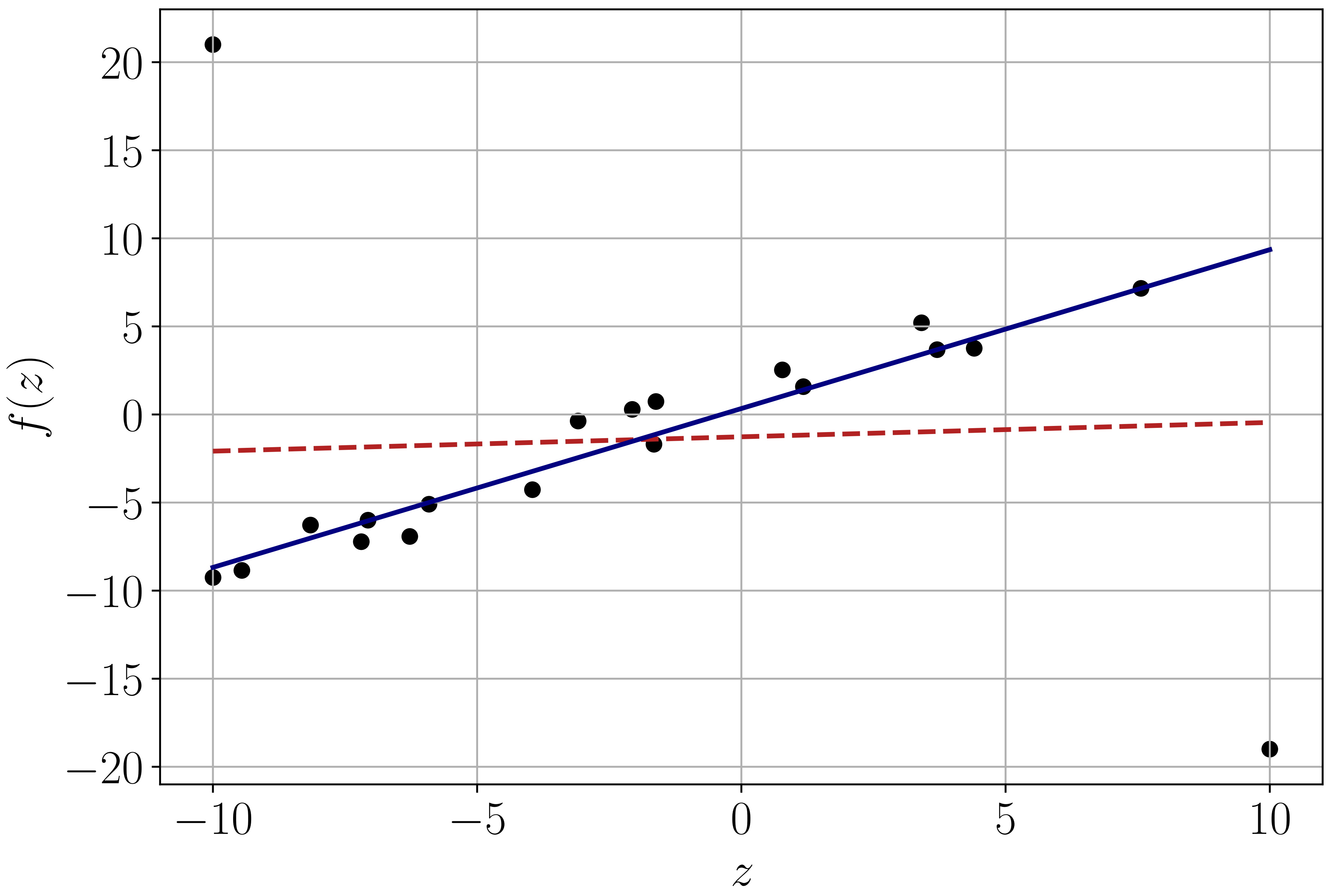
minimize $\quad \sum_{i=1}^{m} (Ax - b)_i^2 = \|Ax - b\|_2^2$

Good news: solution is in closed form $x^\star = (A^T A)^{-1} A^T b$
Bad news: solution is very sensitive to outliers!

Data-fitting example

Fit a linear function $f(z) = x_1 + x_2 z$ to m data points (z_i, f_i):

Approximation problem $Ax \approx b$ where $\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_{b}$



A different way:

minimize $\displaystyle\sum_{i=1}^{m} |Ax - b|_i = \|Ax - b\|_1$

Good news: solution is much more robust to outliers.
Bad news: there is no closed form solution.

Cheapest cat food problem

- Choose quantities $x_1, \ldots, x_n$ of n ingredients each with unit cost c_j.
- Each ingredient j has nutritional content a_{ij} for nutrient i.
- Require a minimum level b_i for each nutrient i.

$$
\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{n} c_j x_j \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1 \ldots m \\
& x_j \geq 0, \quad j = 1 \ldots n
\end{array}
$$

[Photo of Phoebe, my cat]

Would you give her the optimal food ?

Linear optimization

Linear optimization

Linear Programming (LP)

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{n} c_i x_i \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \\
& \sum_{j=1}^{n} d_{ij} x_j = f_i, \quad i = 1, \ldots, p
\end{array}
$$

Ingredients

- n **decision variables** (or optimization variables): $x_1, \ldots, x_n$
- Constant **parameters** (or problem data) : c_j, a_{ij}, b_i, d_{ij}, f_i
- A linear **objective function**
- A collection of m **inequality constraints** and p **equality constraints**

Where does linear optimization appear?

Supply chain management

Assignment problems

Scheduling and routing problems

Finance

Optimal control problems

Network design and network operations

Many other domains…

A brief history of linear optimization

1940s :

- Foundations and applications in economics and logistics (Kantorovich, Koopmans)
- **1947** : Development of the **simplex method** by Dantzig

1950s—70s:

- Applications expand to engineering, OR, computer science…
- **1975** : Nobel prize in economics for Kantorovich and Koopmans

1980s:

- Development of polynomial time algorithms for LPs
- **1984** : Development of the **interior point method** by Karmarkar

—Today:

- Continued algorithm development. Expansion to very large problems.

Why linear optimization?

“Easy” to solve

- It is solvable in polynomial time, tractable in practice
- State-of-the-art software can solve LPs with tens of thousands of variables. We can solve LPs with millions of variables with specific structure.

Extremely versatile

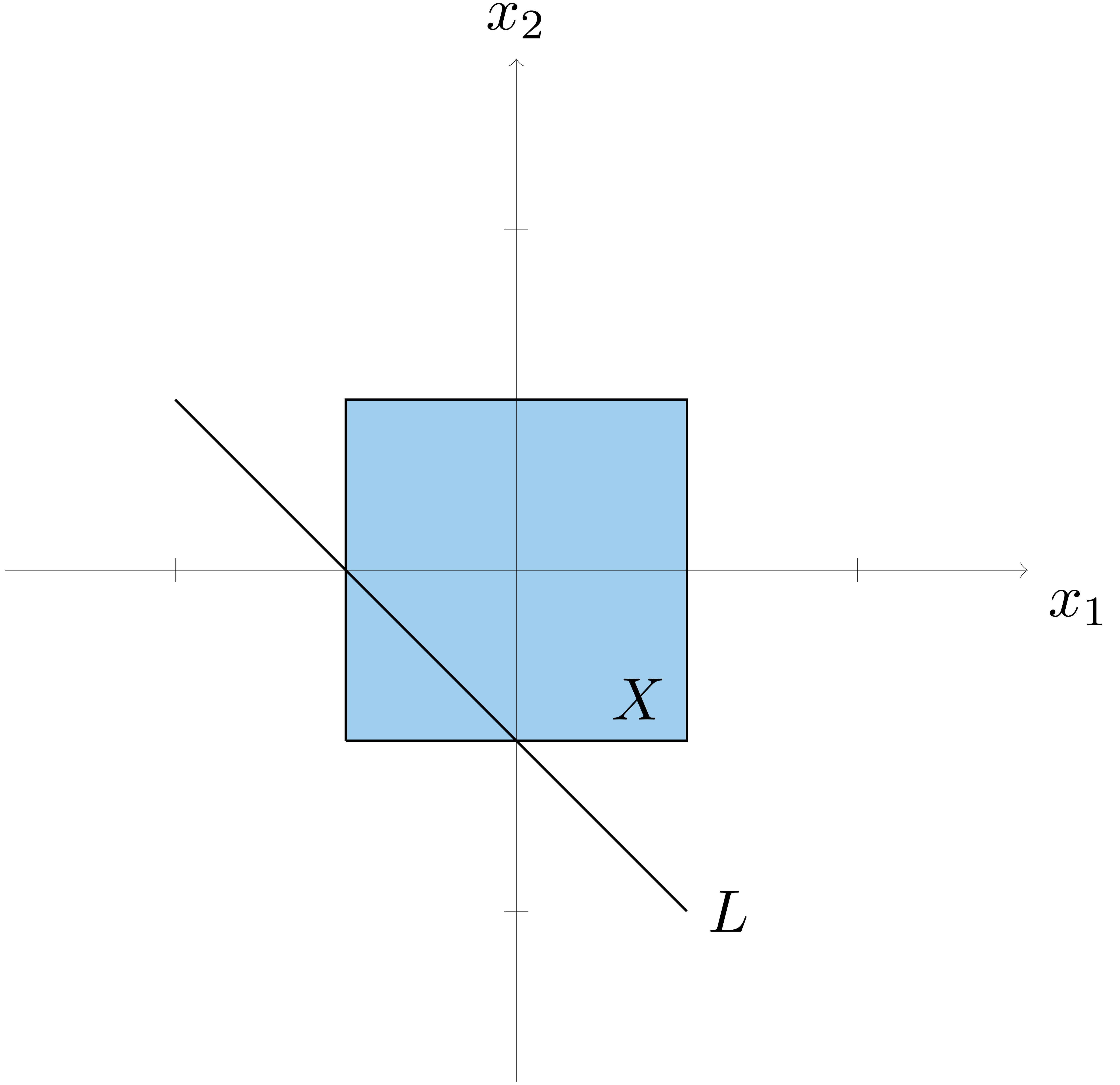
Can model many real-world problems, either exactly or approximately.

Fundamental

- The theory of linear optimization lays the foundation for most optimization theories
- Underpins solutions for more complicated problems, e.g. integer problems.

A simple example

Goal find point as far left as possible,
in the unit box X,
and restricted to the line L



A simple example

Goal find point as far left as possible,
in the unit box X,
and restricted to the line L

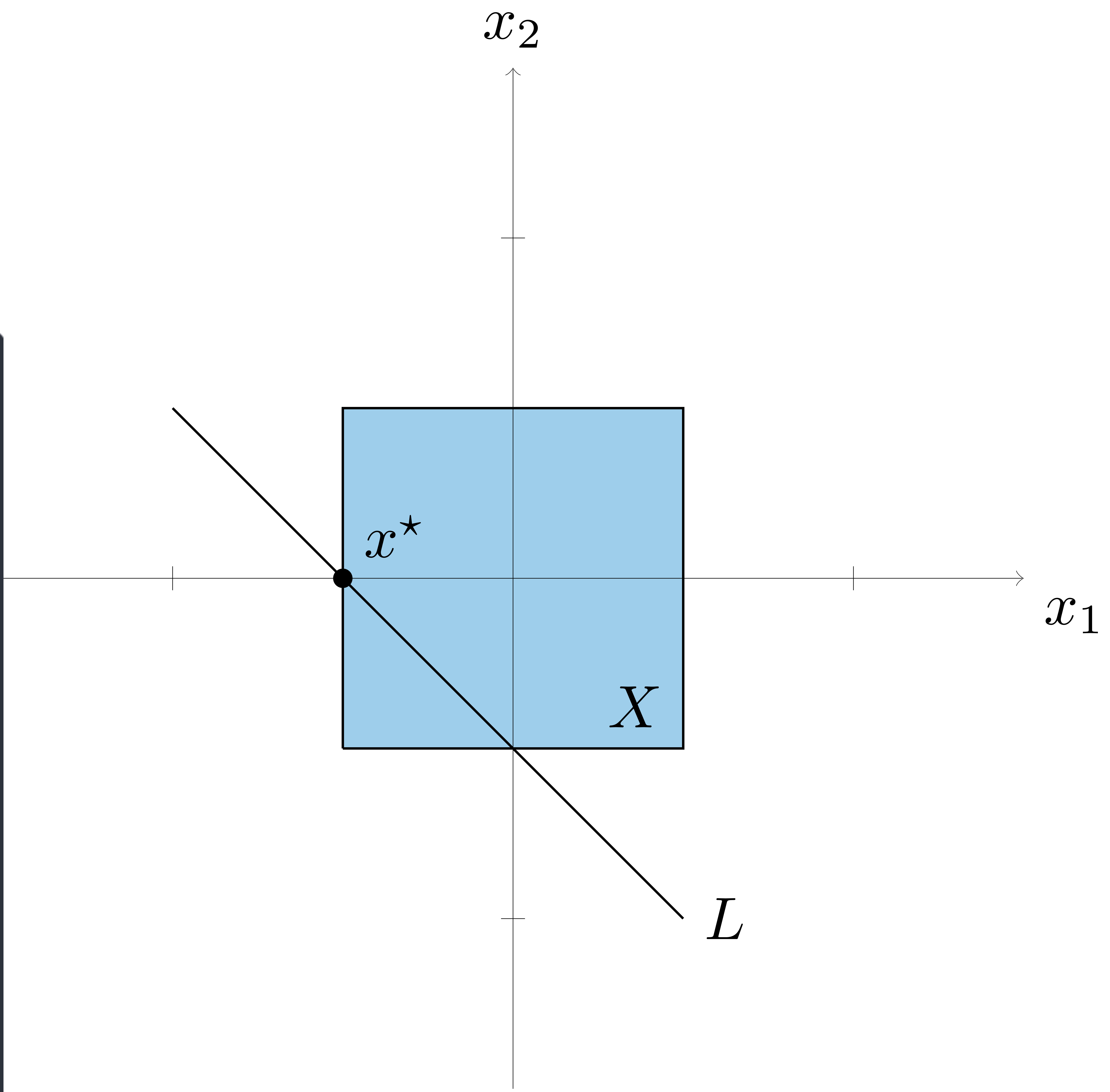
```
import cvxpy as cp

#make decision variable
x = cp.Variable(2)

#define objective
objective = x[0]

#define constraints
constraints = [-1 <= x[0], x[0] <= 1, #inequalities
               -1 <= x[1], x[1] <= 1, #inequalities
               x[0] + x[1] == -1]     #equalities

#make problem and solve
prob = cp.Problem(cp.Minimize(objective), constraints)
prob.solve()
```



Linear optimization

Using vectors

minimize	$\sum_{i=1}^n c_i x_i$		minimize	$c^T x$	
subject to	$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m$	$\longrightarrow$	subject to	$a_i^T x \leq b_i, \quad i = 1, \ldots, m$	
	$\sum_{j=1}^n d_{ij} x_j = f_i, \quad i = 1, \ldots, p$			$d_i^T x = f_i, \quad i = 1, \ldots, p$	

$c,\ a_i,\ d_i$ are n-vectors

$$c = (c_1, \ldots, c_n)$$

$$a_i = (a_{i1}, \ldots, a_{in})$$

$$d_i = (d_{i1}, \ldots, d_{in})$$

Linear optimization

Using matrices

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n c_i x_i \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \\ & \sum_{j=1}^n d_{ij} x_j = f_i, \quad i = 1, \ldots, p \end{array} \longrightarrow \begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

A is $m \times n$-matrix with elements a_{ij} and rows a_i^T

D is $p \times n$-matrix with elements d_{ij} and rows d_i^T

All (in)equalities are elementwise

Optimization terminology

$$
\begin{array}{ll}
\text{minimize} & c^T x \\
\text{subject to} & Ax \le b \\
& Dx = f
\end{array}
$$

x is **feasible** if it satisfies the constraints $Ax \le b$ and $Dx = f$

The **feasible set** is the set of all feasible points

$x^\star$ is **optimal** if it is feasible and $c^T x^\star \le c^T x$ for all feasible x

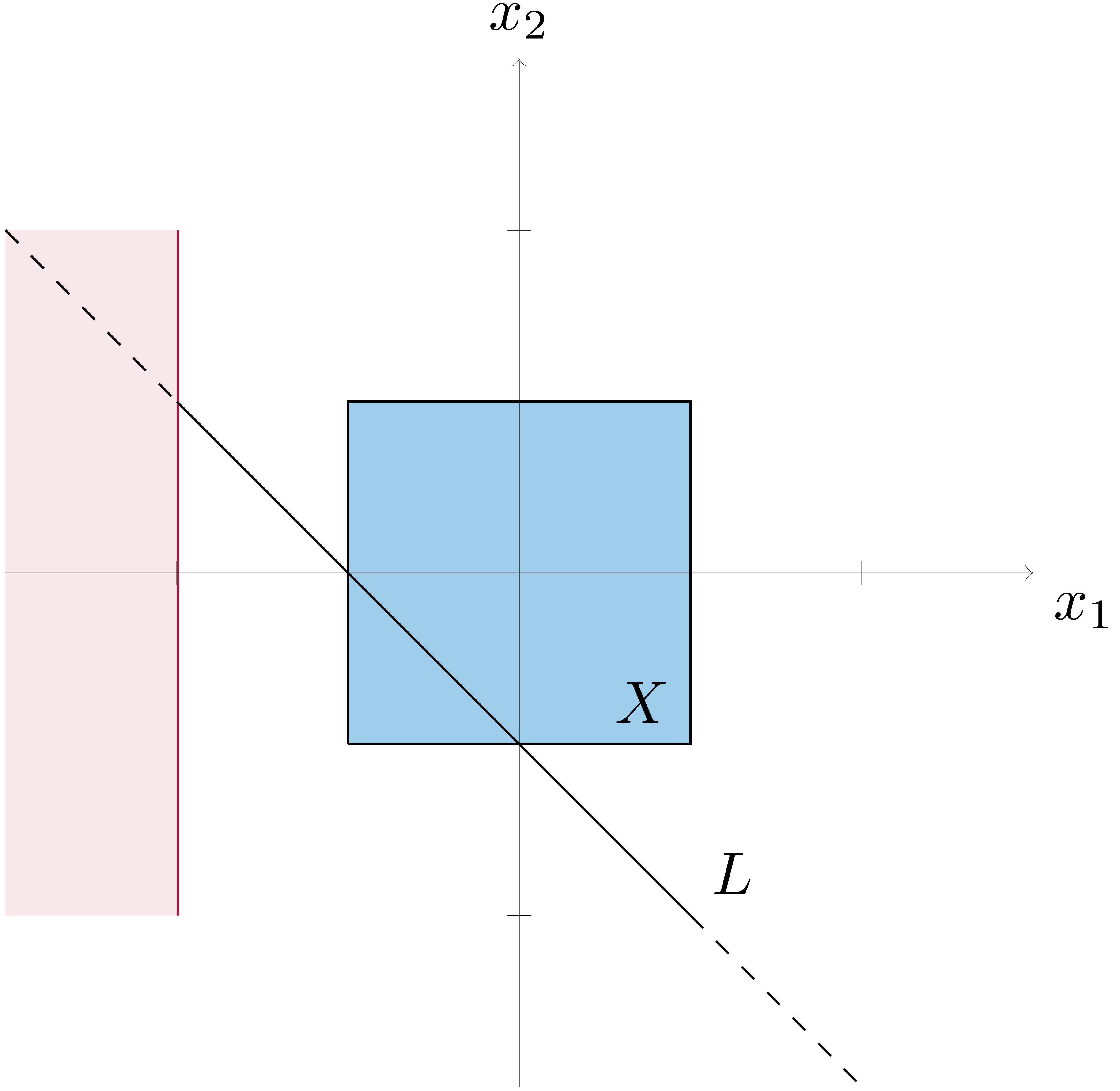
The **optimal value** is $p^\star = c^T x^\star$

Special cases

What can go wrong?

Problem might be “too hard”

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & -1 \le x_1 \le 1 \\ & -1 \le x_2 \le 1 \\ & x_1 + x_2 = -1 \\ & \color{red}{x_1 \le -2} \end{array}$$



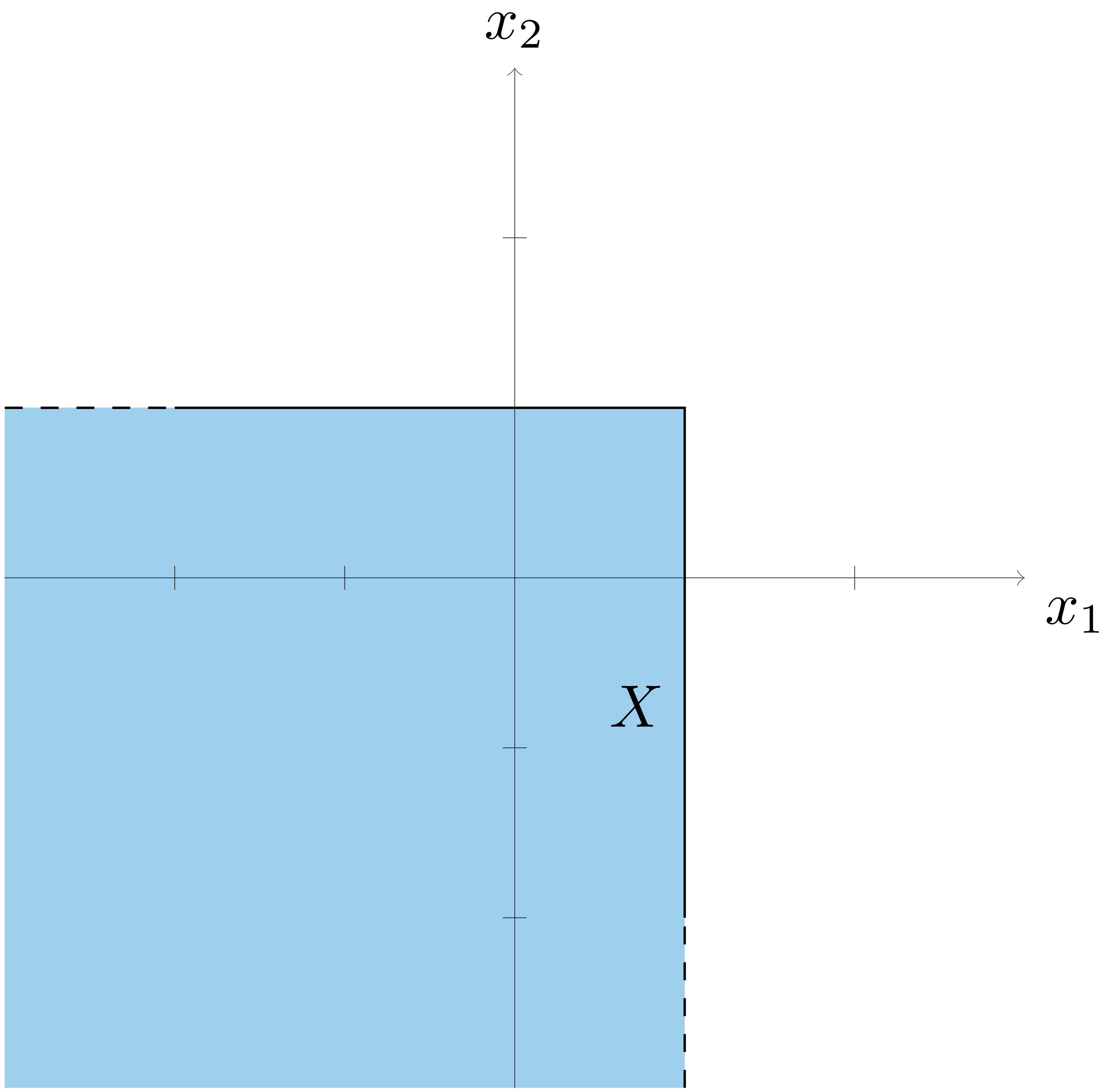
Remarks

- The feasible set is empty.
- The problem is therefore **infeasible**.
- Define the optimal value as $p^\star = +\infty$.

What can go wrong?

Problem might be “too easy”

$$
\begin{array}{ll}
\text{minimize} & x_1 \\
\text{subject to} & \cancel{-1 \leq} x_1 \leq 1 \\
 & \cancel{-1 \leq} x_2 \leq 1 \\
 & \cancel{x_1 + x_2 = -1}
\end{array}
$$



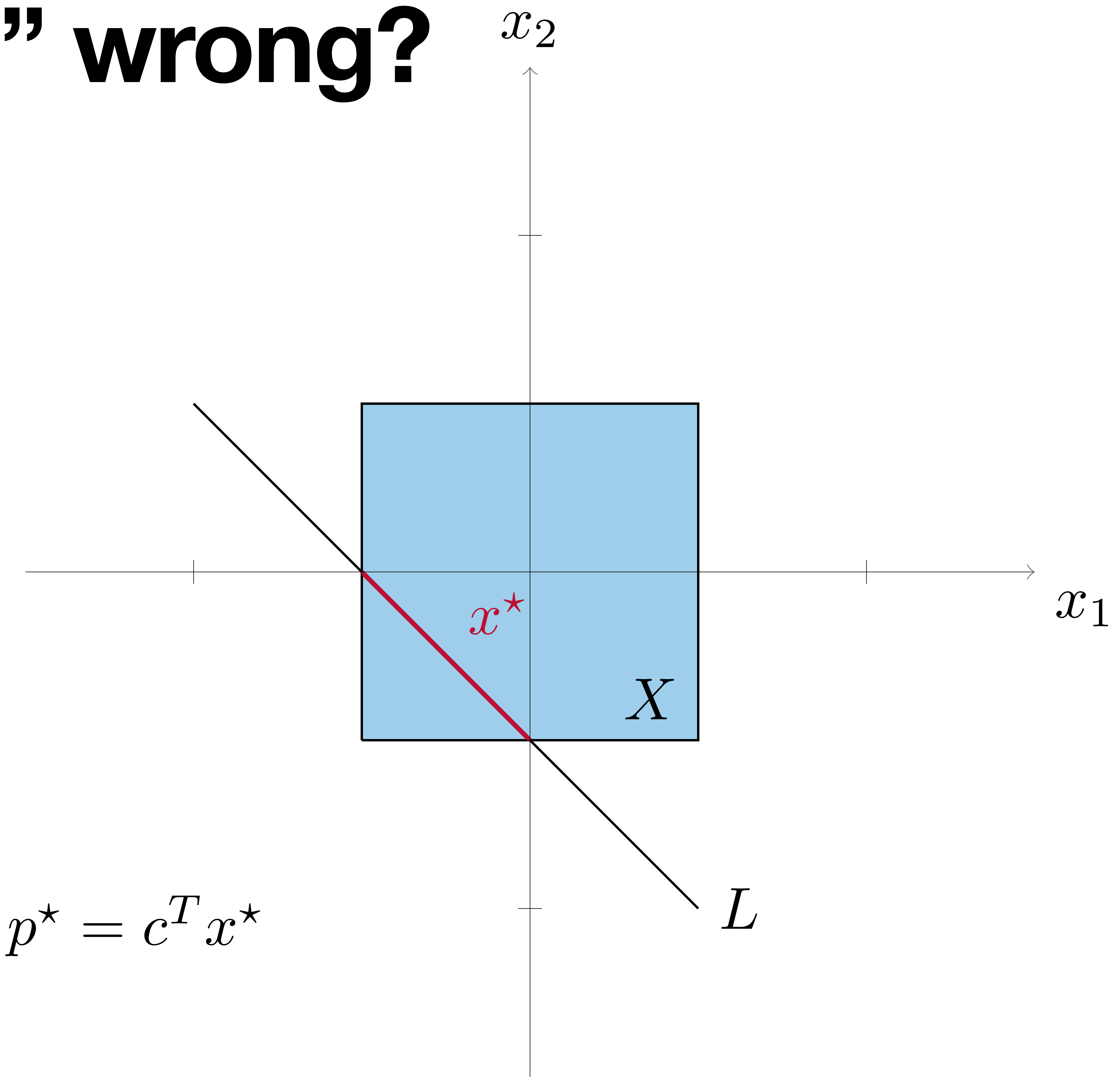
Remarks

- The value of $c^T x$ is **unbounded below** on the feasible set.
- Define the optimal value as $p^\star = -\infty$.

What can go “a little bit” wrong?

More than one optimizer

$$\begin{array}{ll} \text{minimize} & x_1 + x_2 \\ \text{subject to} & -1 \leq x_1 \leq 1 \\ & -1 \leq x_2 \leq 1 \\ & x_1 + x_2 = -1 \end{array}$$



Remarks

- The optimal value is $p^\star = -1$
- There is more than one $x^\star$ that achieves $p^\star = c^T x^\star$
- The optimizer is **non-unique**

Feasibility problems

The constraints satisfiability problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array} \quad \xrightarrow{\text{is a special case of}} \quad \begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

Remarks

- $p^\star = 0$ if constraints are feasible (consistent). Every feasible x is optimal
- $p^\star = \infty$ otherwise

Standard form

Standard form

Definition

$$
\begin{array}{ll}
\text{minimize} & c^T x \\
\text{subject to} & Ax = b \\
 & x \geq 0
\end{array}
$$

- Minimization
- Equality constraints
- Nonnegative variables

- Matrix notation for **theory**
- Standard form for **algorithms**

Standard form

Transformation tricks

Change objective

If “maximize”, use $-c$ instead of c and change to “minimize”.

Eliminate inequality constraints

If $Ax \leq b$, define s and write $Ax + s = b, \; s \geq 0$.

If $Ax \geq b$, define s and write $Ax - s = b, \; s \geq 0$.

s are the **slack variables**

Change variable signs

If $x_i \leq 0$, define $y_i = -x_i$.

Eliminate “free” variables

If x_i unconstrained, define $x_i = x_i^+ - x_i^-$, with $x_i^+ \geq 0$ and $x_i^- \geq 0$.

Standard form

Transformation example

$$
\begin{array}{lrcrl}
\text{minimize} & 2x_1 & + & 4x_2 & \\
\text{subject to} & x_1 & + & x_2 & \geq 3 \\
 & 3x_1 & + & 2x_2 & = 14 \\
 & x_1 & & & \geq 0
\end{array}
$$

$$\Big\downarrow$$

$$
\begin{array}{lrcrcrcrl}
\text{minimize} & 2x_1 & + & 4x_2^+ & - & 4x_2^- & & & \\
\text{subject to} & x_1 & + & x_2^+ & - & x_2^- & - & x_3 & = 3 \\
 & 3x_1 & + & 2x_2^+ & - & 2x_2^- & & & = 14 \\
 & x_1, & & x_2^+, & & x_2^-, & & x_3 & \geq 0.
\end{array}
$$

Software

Solvers for linear programs

Algorithms and **theory** are very mature:

- Simplex methods, interior-point methods, first order methods etc

Software is widely available:

- Can solve problems up to several million variables
- Widely used in industry and academic research

Examples

- Commercial solvers : Mosek, CPLEX, Gurobi, Matlab (linprog)
- Free solvers : GLPK, CLP, SCS, OSQP

Modelling tools for linear programs

Modelling tools simplify the formulation of LPs (and other problems)

- Accept optimization problem in common notation ($\max, \|\cdot\|_1, \ldots$)
- Recognize problems that can be converted to LPs
- Automatically convert to input format required by a specific LP solver

Examples

- AMPL, GAMS
- CVX, YALMIP (Matlab)
- **CVXPY, Pyomo (Python)**
- JuMP.jl, Convex.jl (Julia)

Simple example revisited

Goal find point as far left as possible,
in the unit box X,
and restricted to the line L

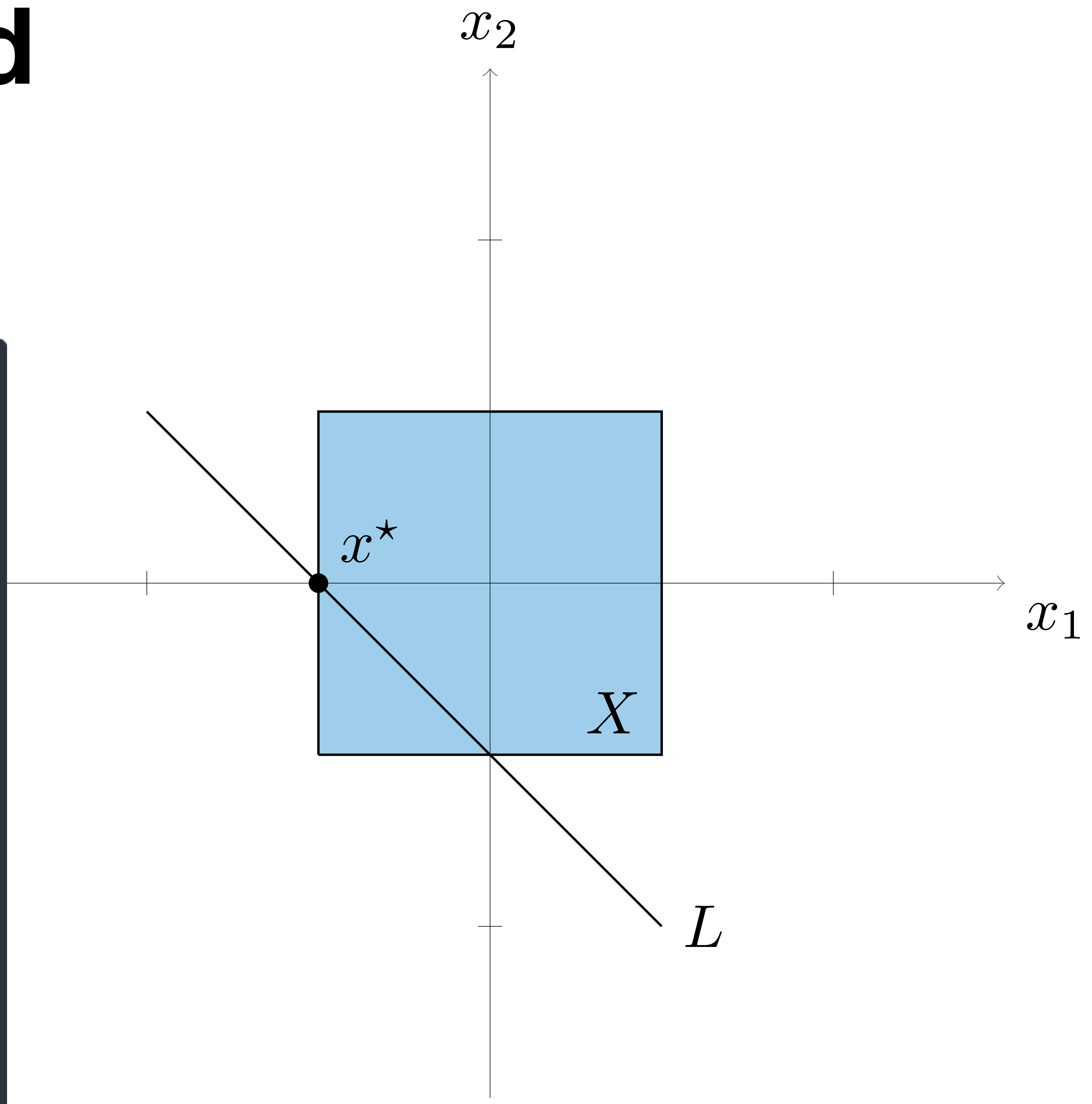
```
import cvxpy as cp

#make decision variable
x = cp.Variable(2)

#define objective
objective = x[0]

#define constraints
constraints = [ cp.norm(x, 'inf') <= 1, #inequalities
                cp.sum(x) == -1]   #equalities

#make problem and solve
prob = cp.Problem(cp.Minimize(objective), constraints)
prob.solve()
```



References

- Bertsimas, Tsitsiklis: Introduction to Linear Optimization
  - Chapter 1: Introduction
- R. Vanderbei: Linear Programming — Foundations and Extensions
  - Chapter 1: intro to linear programming

Next time

Piecewise linear optimization

- Optimization problems with norms and max functions
- Some applications