

ORF307 – Optimization

7. Linear optimization

Ed Forum

- Is the lowest PSET grade dropped? —> Yes
- What does the z vector represent?
- My question is are constraints equalities always =, or will we see how to handle cases with \leq and \geq ?

Recap

Least squares with equality constraints

The (linearly) constrained least squares problem is

minimize $\|Ax - b\|^2$
subject to $Cx = d$

equality
constraints

objective
function

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^* is a *solution* if

- $Cx^* = d$
- $\|Ax^* - b\|^2 \leq \|Ax - b\|^2$
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, $\|Cx - d\|^2$.

Optimality conditions via calculus

$$z = (z_1, \dots, z_p)$$

$$\begin{array}{l} \text{minimize} \quad f(x) = \|Ax - b\|^2 \\ \text{subject to} \quad Cx = d \end{array}$$



$$\begin{array}{l} \text{minimize} \quad f(x) = \|Ax - b\|^2 \\ \text{subject to} \quad c_i^T x = d_i, \quad i = 1, \dots, p \end{array}$$

Lagrangian function

$$L(x, z) = f(x) + z_1(c_1^T x - d_1) + \dots + z_p(c_p^T x - d_p) \stackrel{\text{red}}{=} f(x) + z^T(Cx - d)$$

Optimality conditions

$$\frac{\partial L}{\partial x_i}(x^*, z) = 0, \quad i = 1, \dots, n,$$

$$\frac{\partial L}{\partial z_i}(x^*, z) = 0, \quad i = 1, \dots, p$$

Optimality conditions via calculus

$$L(x, z) = x^T A^T A x - 2(A^T b)^T x + b^T b + z_1(c_1^T x - d_1) + \cdots + z_p(c_p^T x - d_p)$$

Optimality conditions

Vector form

$$\begin{aligned} \frac{\partial L}{\partial z_i}(x^*, z) &= c_i^T x - d_i = 0 && \text{(we already knew)} && Cx = d \\ \frac{\partial L}{\partial x_i}(x^*, z) &= 2 \sum_{j=1}^n (A^T A)_{ij} x_j^* - 2(A^T b)_i + \sum_{j=1}^p z_j (c_j)_i = 0 && \longrightarrow && 2A^T A x^* - 2A^T b + C^T z = 0 \end{aligned}$$

Karush-Kuhn-Tucker (KKT) conditions

$$\begin{bmatrix} 2A^T A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x^* \\ z \end{bmatrix} = \begin{bmatrix} 2A^T b \\ d \end{bmatrix} \quad \text{(square set of } n + p \text{ linear equations)}$$

Note KKT equations are extension of normal equations to constrained least squares

Today's lecture

Linear optimization

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

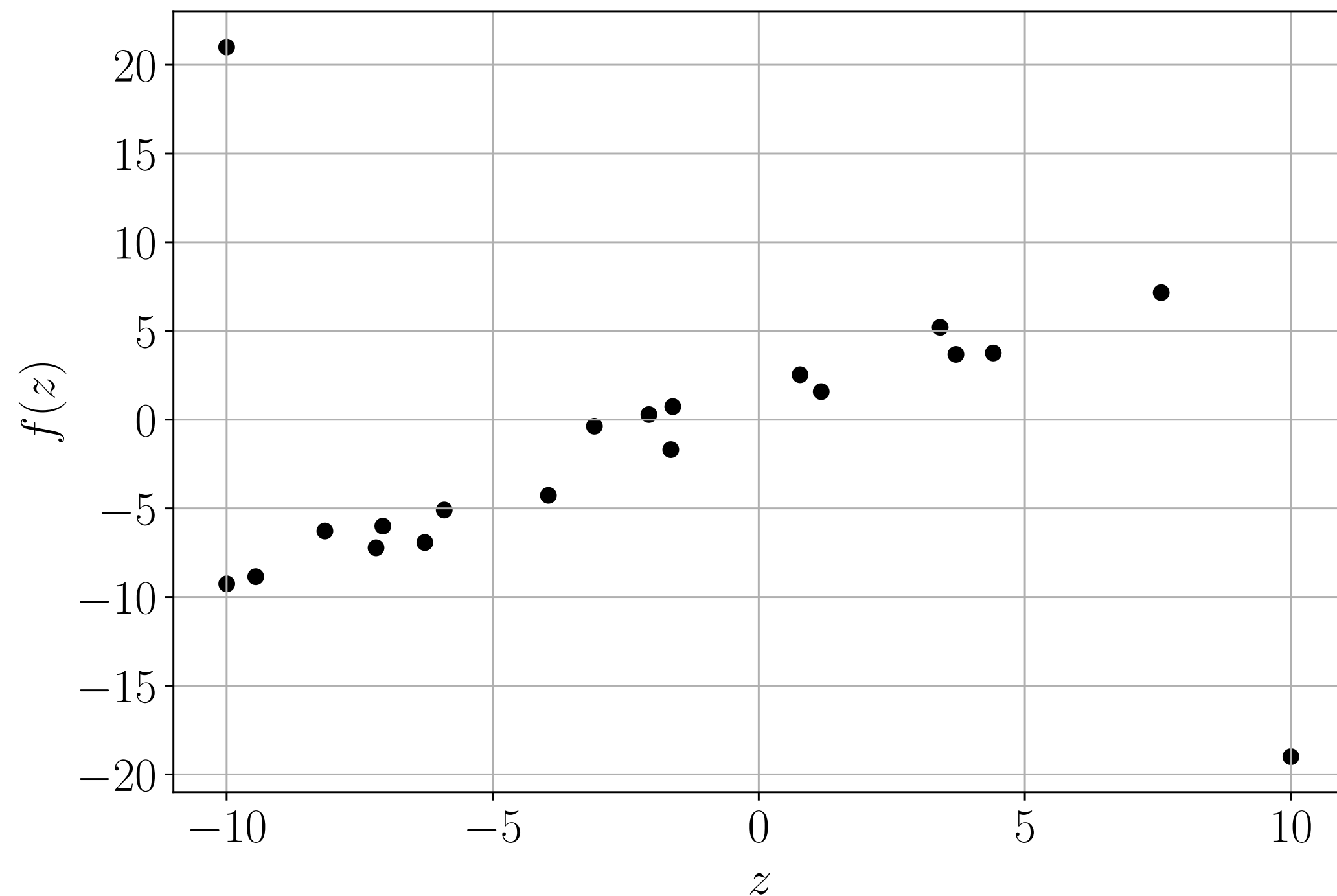
Some simple examples

Data-fitting example

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

Approximation problem $Ax \approx b$ where

$$\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_A \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_b$$

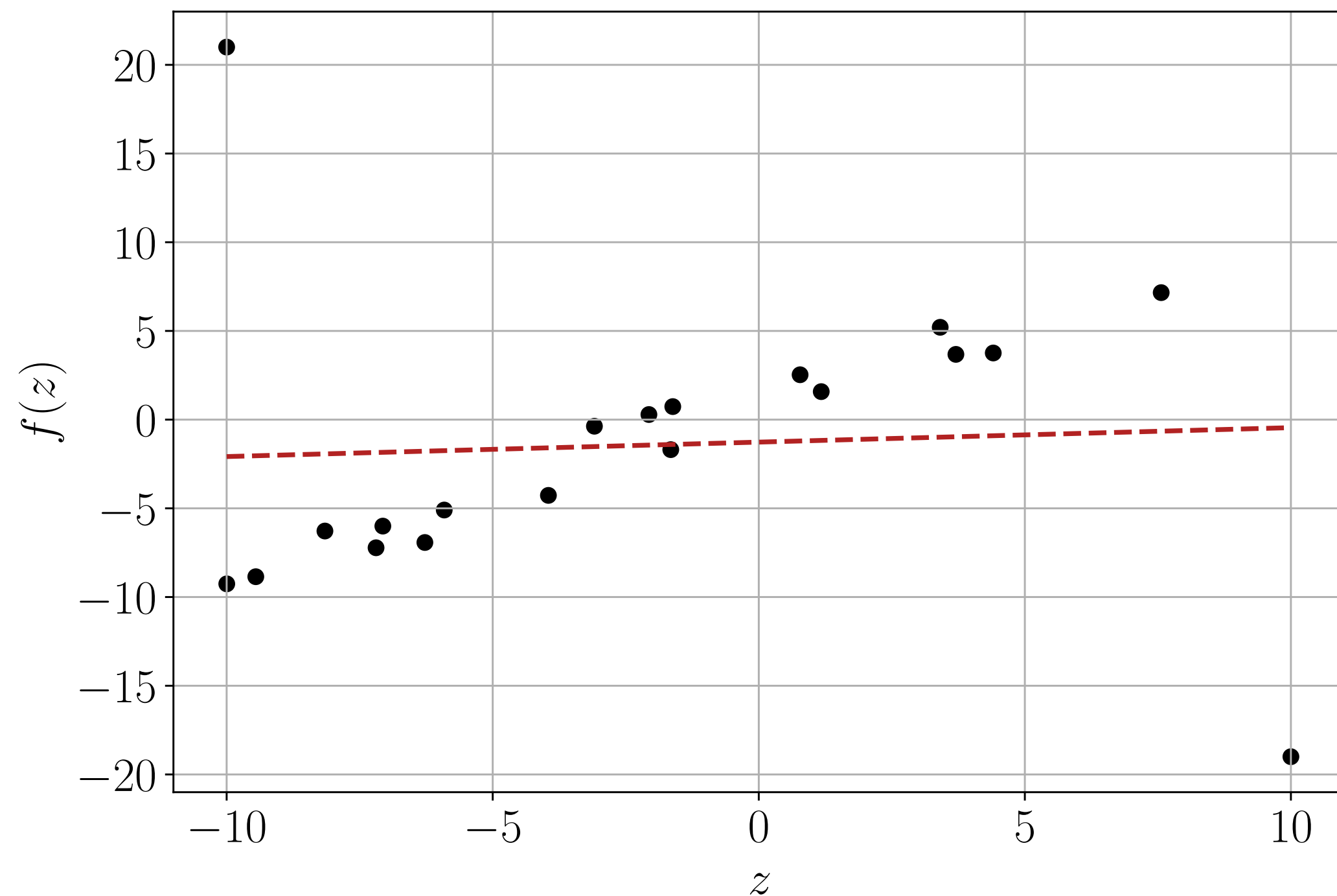


Data-fitting example

Fit a linear function $f(z) = \theta_1 + \theta_2 z$ to m data points (z_i, f_i) :

Approximation problem $Ax \approx b$ where

$$\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_A \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_b$$



Least squares way:

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

Good news: solution is in closed form $x^* = (A^T A)^{-1} A^T b$

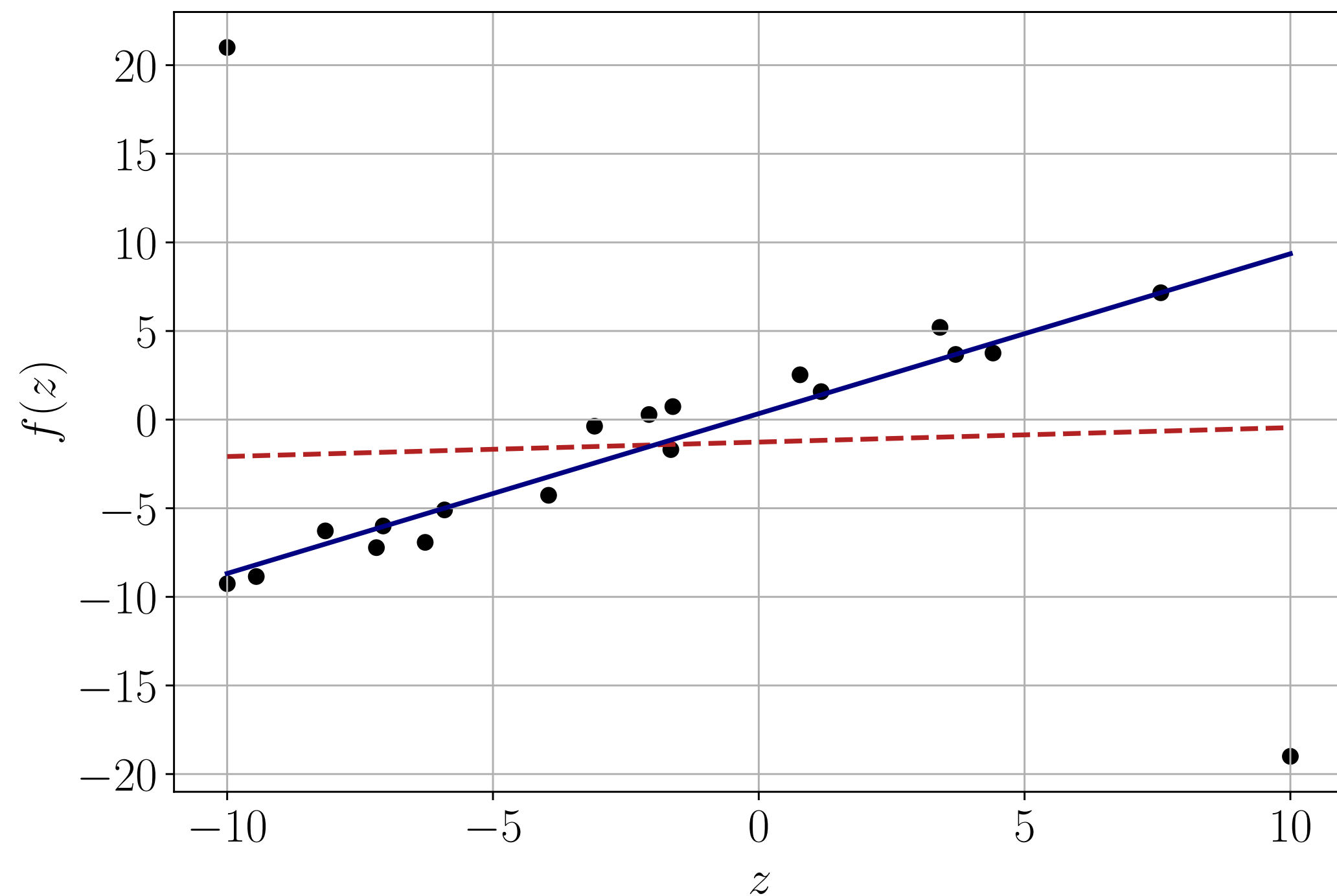
Bad news: solution is very sensitive to outliers!

Data-fitting example

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

Approximation problem $Ax \approx b$ where

$$\underbrace{\begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}}_A \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}}_b$$



A different way:

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

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, \dots, 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 .

Cheapest cat food problem

- Choose quantities x_1, \dots, 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 .

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

Cheapest cat food problem

- Choose quantities x_1, \dots, 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 .

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



[Photo of Phoebe, my cat]

**Would you give her
the optimal food ?**

Linear optimization

Linear optimization

Linear Programming (LP)

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

Ingredients

- n **decision variables** (or optimization variables): x_1, \dots, 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

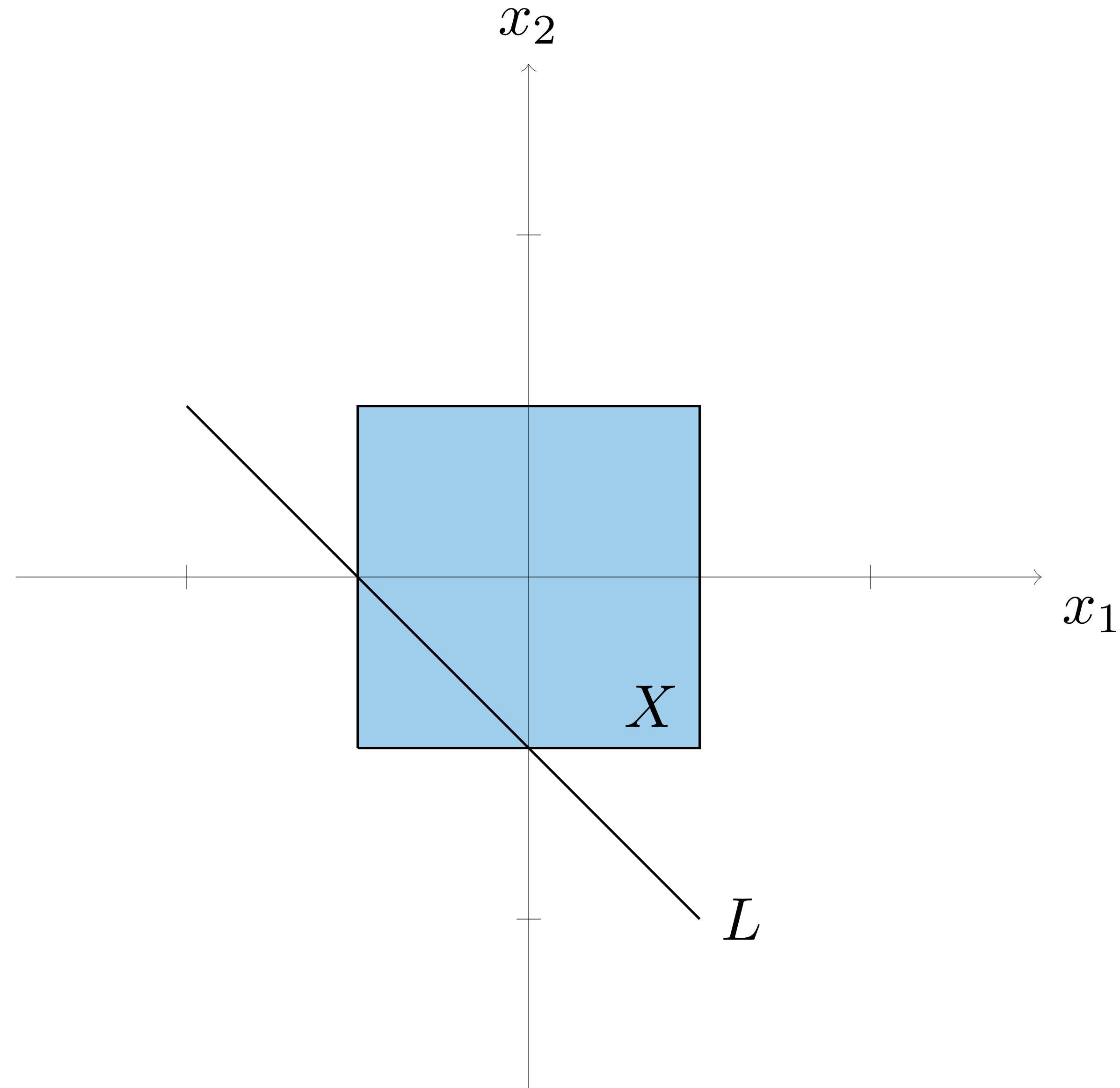
Can model many real-world problems, either exactly or approximately.

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

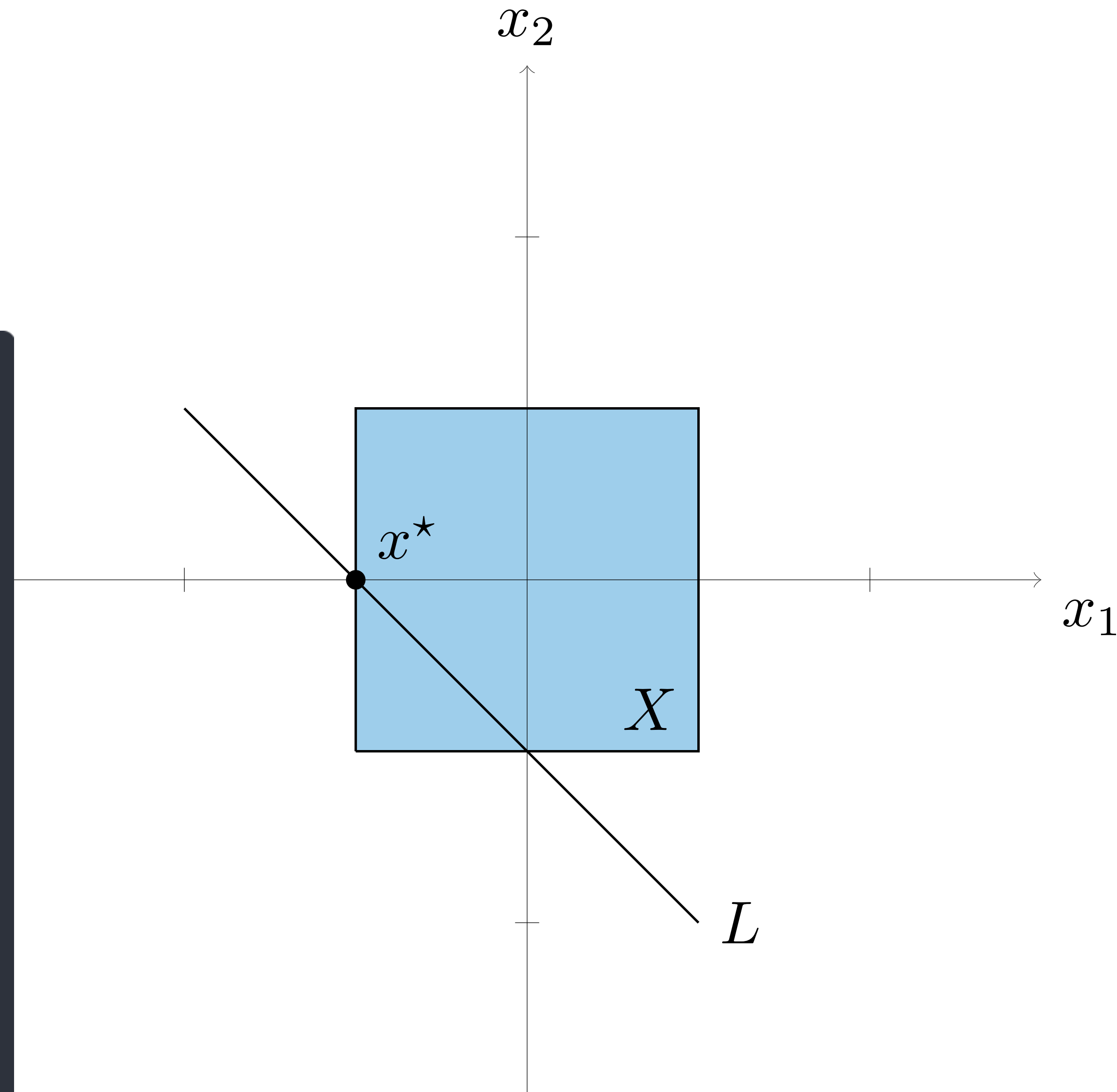
```
import cvxpy as cp

#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

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

c, a_i, d_i are n -vectors

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

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

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

Linear optimization

Using matrices

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

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

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

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

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

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

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

Special cases

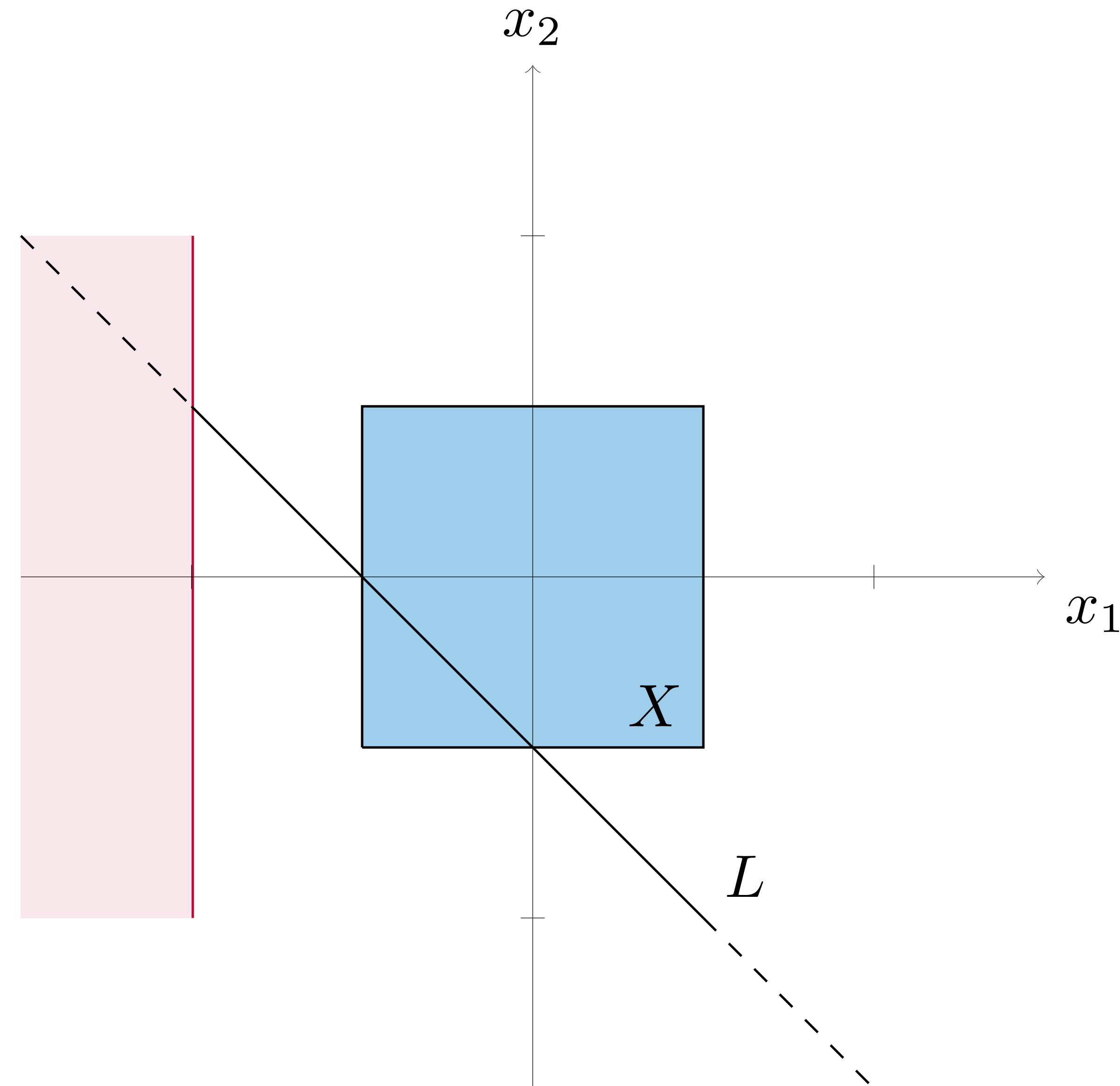
What can go wrong?

Problem might be “too hard”

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

Remarks

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



What can go wrong?

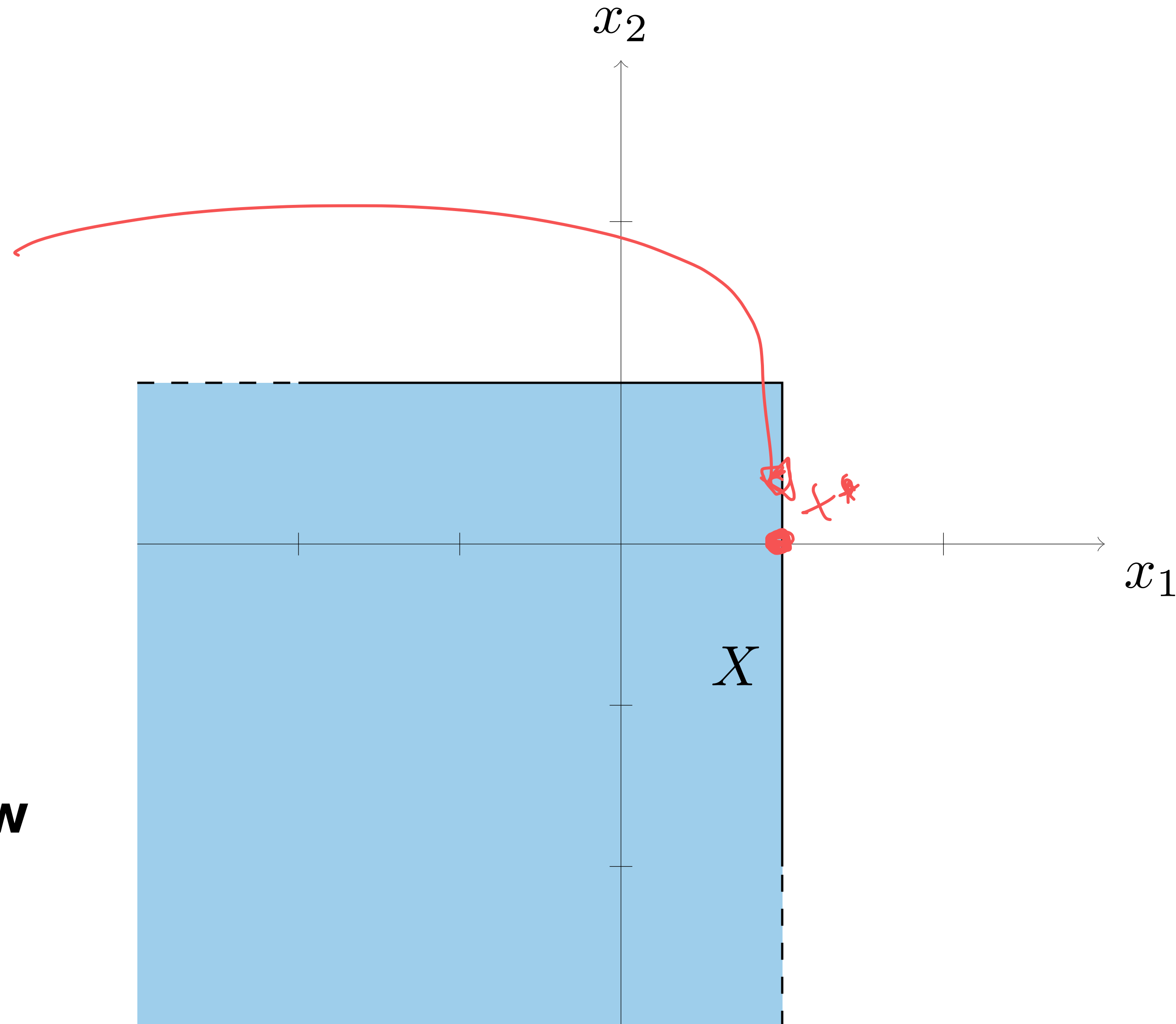
Problem might be “too easy”

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

INSTEAD?
 $-x_1$

Remarks

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



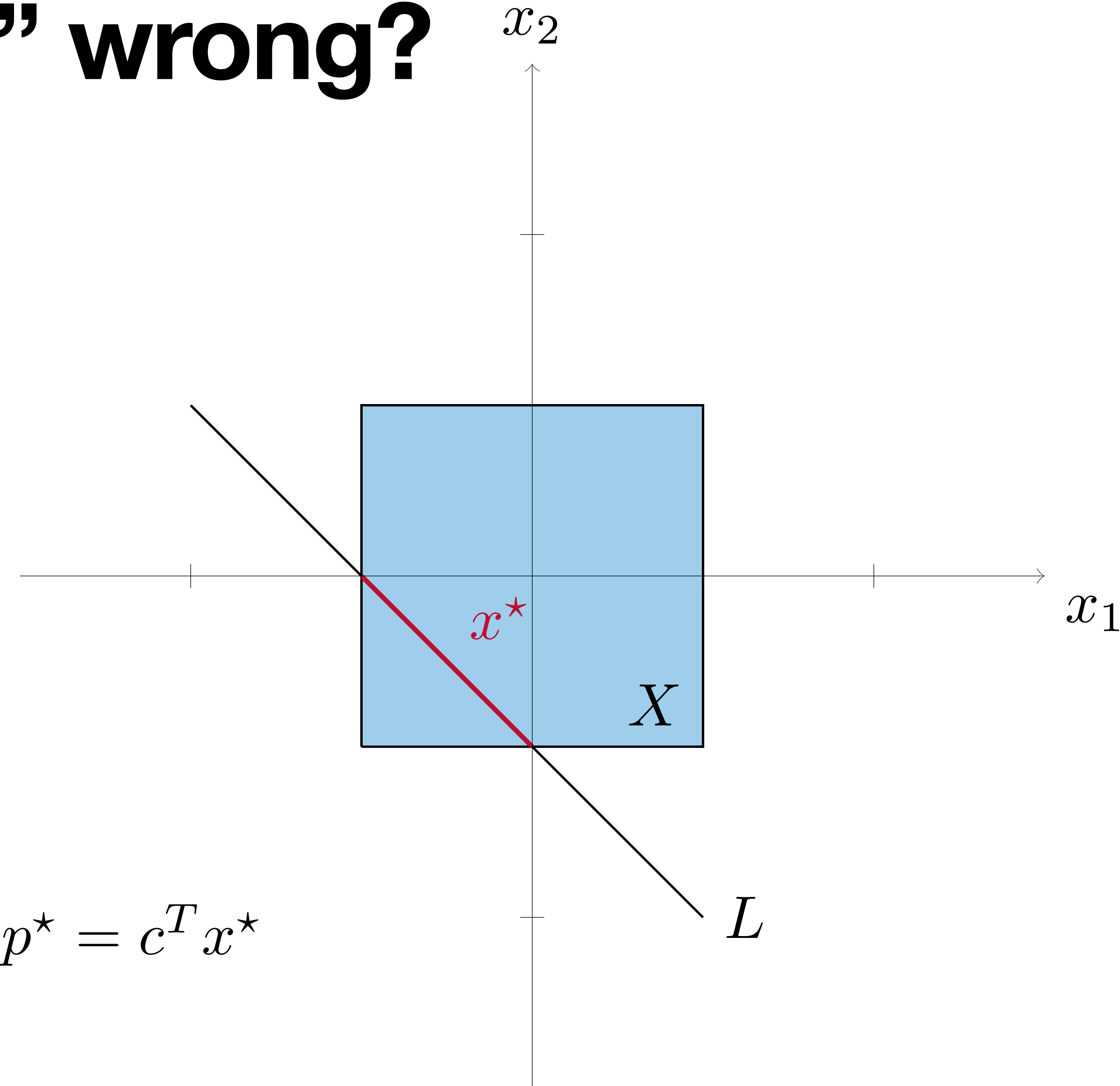
What can go “a little bit” wrong?

More than one optimizer

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

Remarks

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



Feasibility problems

The constraints satisfiability problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

Feasibility problems

The constraints satisfiability problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

is a special case of



$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

Feasibility problems

The constraints satisfiability problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

is a special case of



$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & Ax \leq b \\ & Dx = f \end{array}$$

Remarks

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

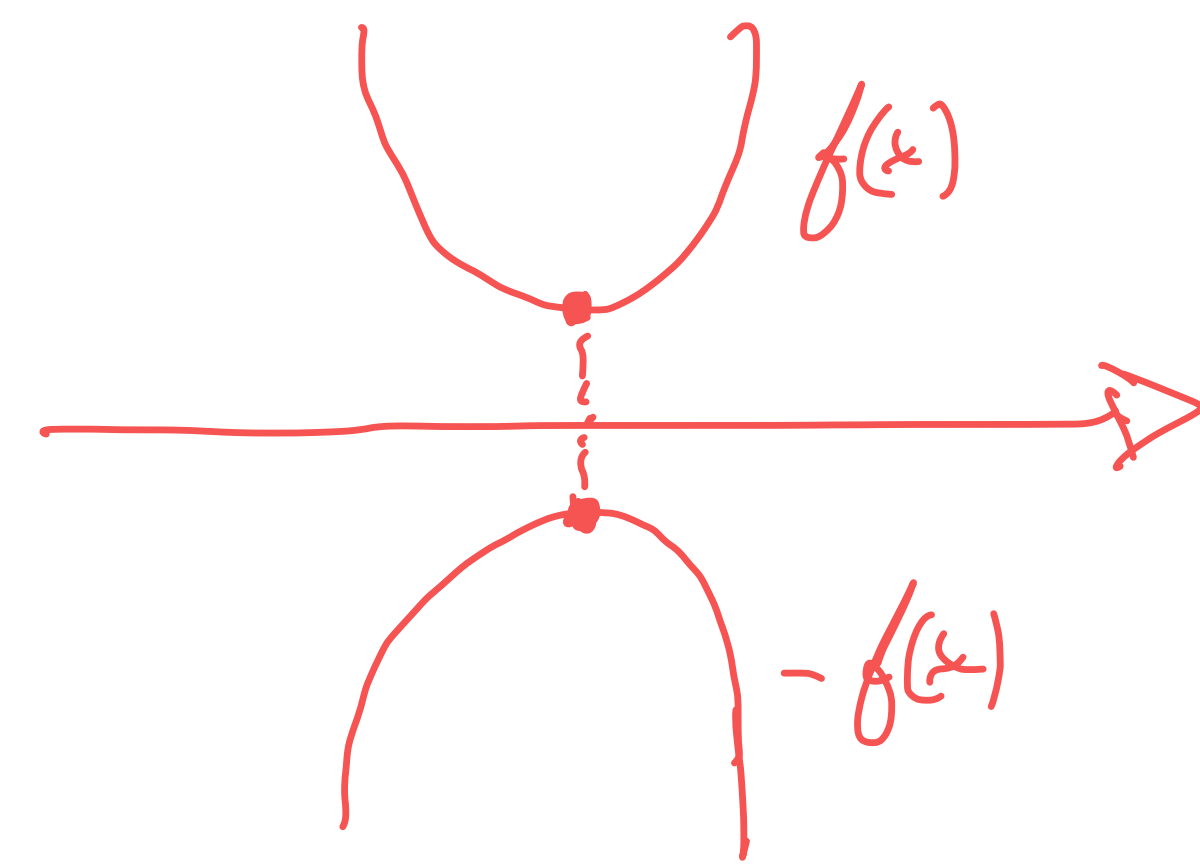
- Matrix notation for **theory**
- Standard form for **algorithms**

Standard form

Transformation tricks

Change objective

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



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**

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 $y_i = -x_i$.

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

$$x_2 = x_2^+ - x_2^-$$

$$x_2^+ \geq 0, x_2^- \geq 0$$

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

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

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

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 , $\| \cdot \|_1, \dots$)
- Recognize problems that can be converted to LPs
- Automatically convert to input format required by a specific LP solver

Modelling tools for linear programs

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

- Accept optimization problem in common notation (\max , $\|\cdot\|_1, \dots$)
- 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

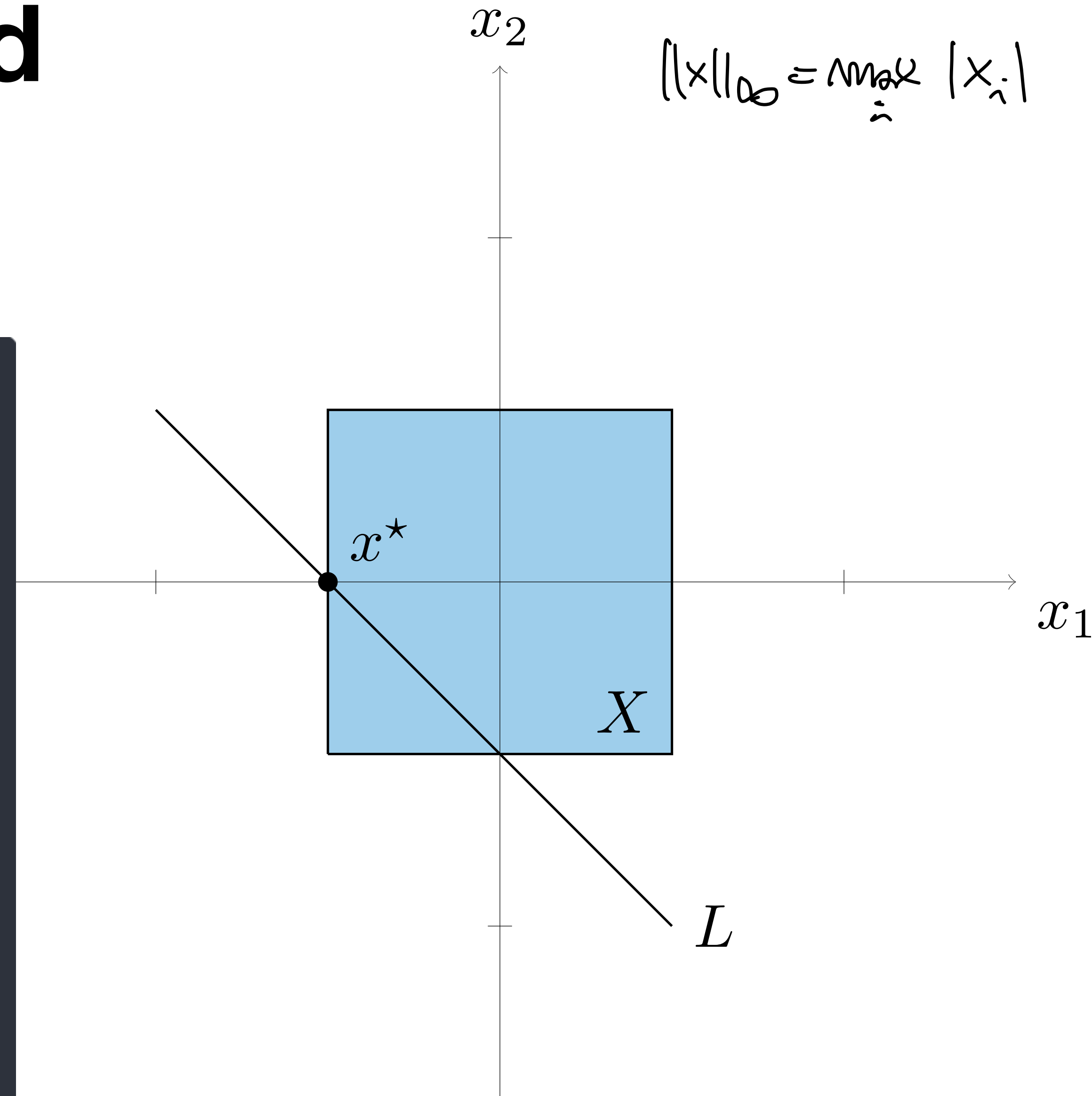
#make decision variable
x = cp.Variable(2)

#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()
```

solver = cp.GLPK



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