ORF522 – Linear and Nonlinear Optimization

2. Linear optimization

Bartolomeo Stellato — Fall 2020

Today's agenda

Readings: [Chapter 1, Bertsimas, Tsitsiklis]

- Linear optimization in inner-product and matrix notation
- Optimization terminology
- Standard form
- Piecewise-linear minimization
- Examples

Vector notations

By default, all vectors are column vectors and denoted by

$$x = (x_1, \ldots, x_n)$$

The transpose of a vector is x^T

$a^T x$ is the inner product between a and x

$$a^T x = a_1 x_1 + \cdots + a_n x_n = \sum_{i=1}^{n} a_i x_i$$

Linear optimization

Linear Programming (LP)

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

Objective function and constraints are **linear in the decision variables**

Belongs to **continuous optimization**

Linear optimization

Inner product notation

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

$c,\ a_i,\ d_i$ 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

Matrix notation

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

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

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

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

Unbounded problem: $c^T x$ is unbounded below on the feasible set ($p^\star = -\infty$)

Infeasible problem: feasible set is empty ($p^\star = +\infty$)

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

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

$$\Big\downarrow$$

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

Linear, affine and convex functions

Linear function: $f(x) = a^T x$

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y), \quad \forall x, y \in \mathbf{R}^n,\ \alpha, \beta \in \mathbf{R}$$

Affine function: $f(x) = a^T x + b$

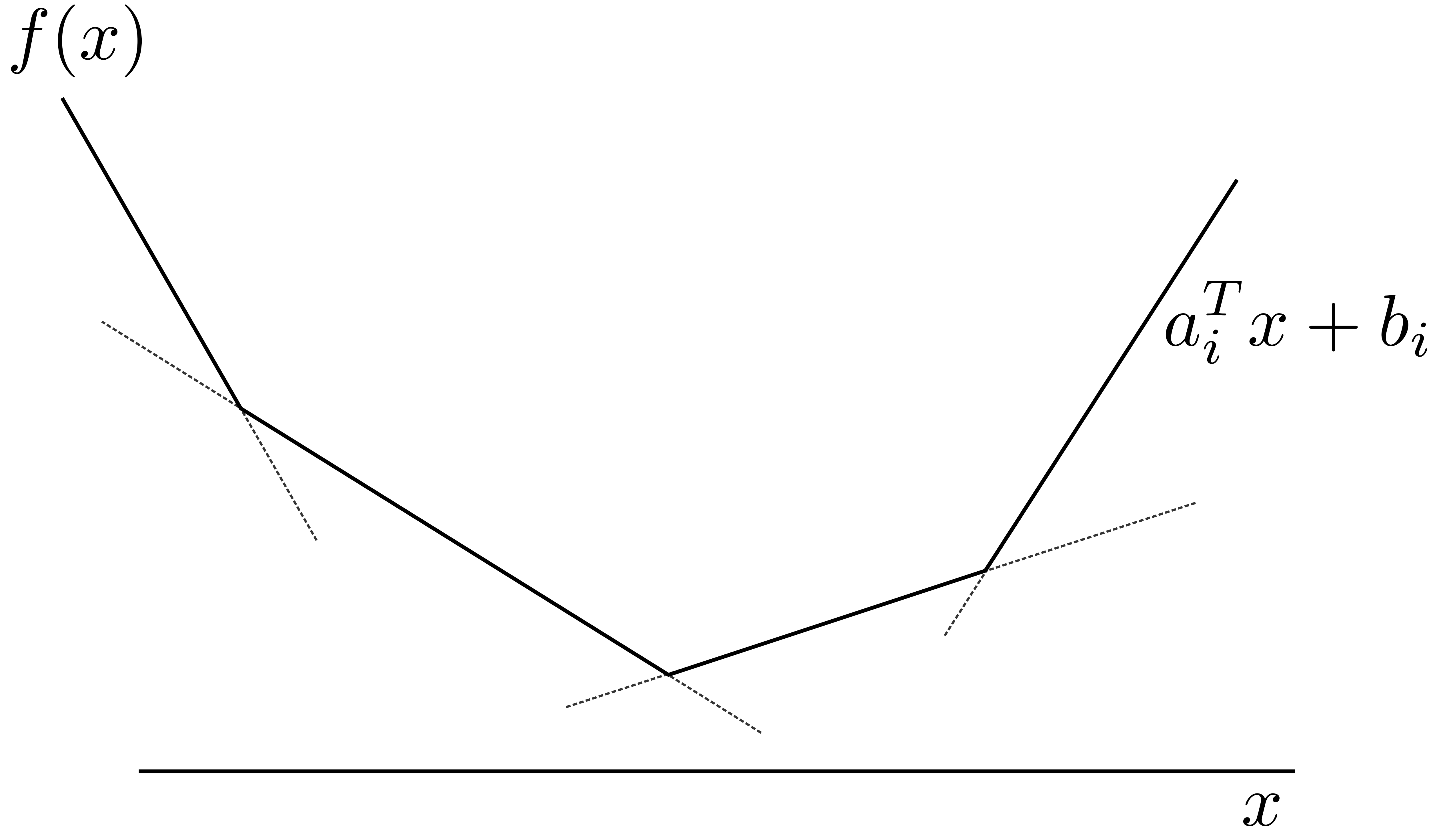
$$f(\alpha x + (1-\alpha) y) = \alpha f(x) + (1-\alpha) f(y), \quad \forall x, y \in \mathbf{R}^n,\ \alpha \in \mathbf{R}$$

Convex function:

$$f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y), \quad \forall x, y \in \mathbf{R}^n,\ \alpha \in [0, 1]$$

Convex piecewise-linear functions

$$f(x) = \max_{i=1,\ldots,m} (a_i^T x + b_i)$$



Convex piecewise-linear minimization

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m} (a_i^T x + b_i)$$

Equivalent linear optimization

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & a_i^T x + b_i \leq t, \quad i = 1, \ldots, m \end{array}$$

Matrix notation

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

$$\tilde{x} = \begin{bmatrix} x \\ t \end{bmatrix}, \quad \tilde{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} a_1^T & -1 \\ \vdots & \vdots \\ a_m^T & -1 \end{bmatrix}, \quad \tilde{b} = \begin{bmatrix} -b_1 \\ \vdots \\ -b_m \end{bmatrix}$$

Sum of piecewise-linear functions

minimize $\quad f(x) + g(x) = \max_{i=1,\dots,m} (a_i^T x + b_i) + \max_{i=1,\dots,p} (c_i^T x + d_i)$

Cost function is piecewise-linear

$$f(x) + g(x) = \max_{\substack{i=1,\dots,m \\ j=1,\dots,p}} ((a_i + c_j)^T x + (b_i + d_j))$$

Equivalent linear optimization

$$\begin{array}{ll} \text{minimize} & t_1 + t_2 \\ \text{subject to} & a_i^T x + b_i \leq t_1, \quad i = 1, \dots, m \\ & c_i^T x + d_i \leq t_2, \quad i = 1, \dots, p \end{array}$$

Matrix notation?

Examples

1-norm approximation

$$\text{minimize} \quad \|Ax - b\|_1$$

The 1-**norm** of m-vector y is

$$\|y\|_1 = \sum_{i=1}^{m} |y_i| = \sum_{i=1}^{m} \max\{y_i, -y_i\}$$

Equivalent problem

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{m} u_i \\ \text{subject to} & -u \leq Ax - b \leq u \end{array}$$

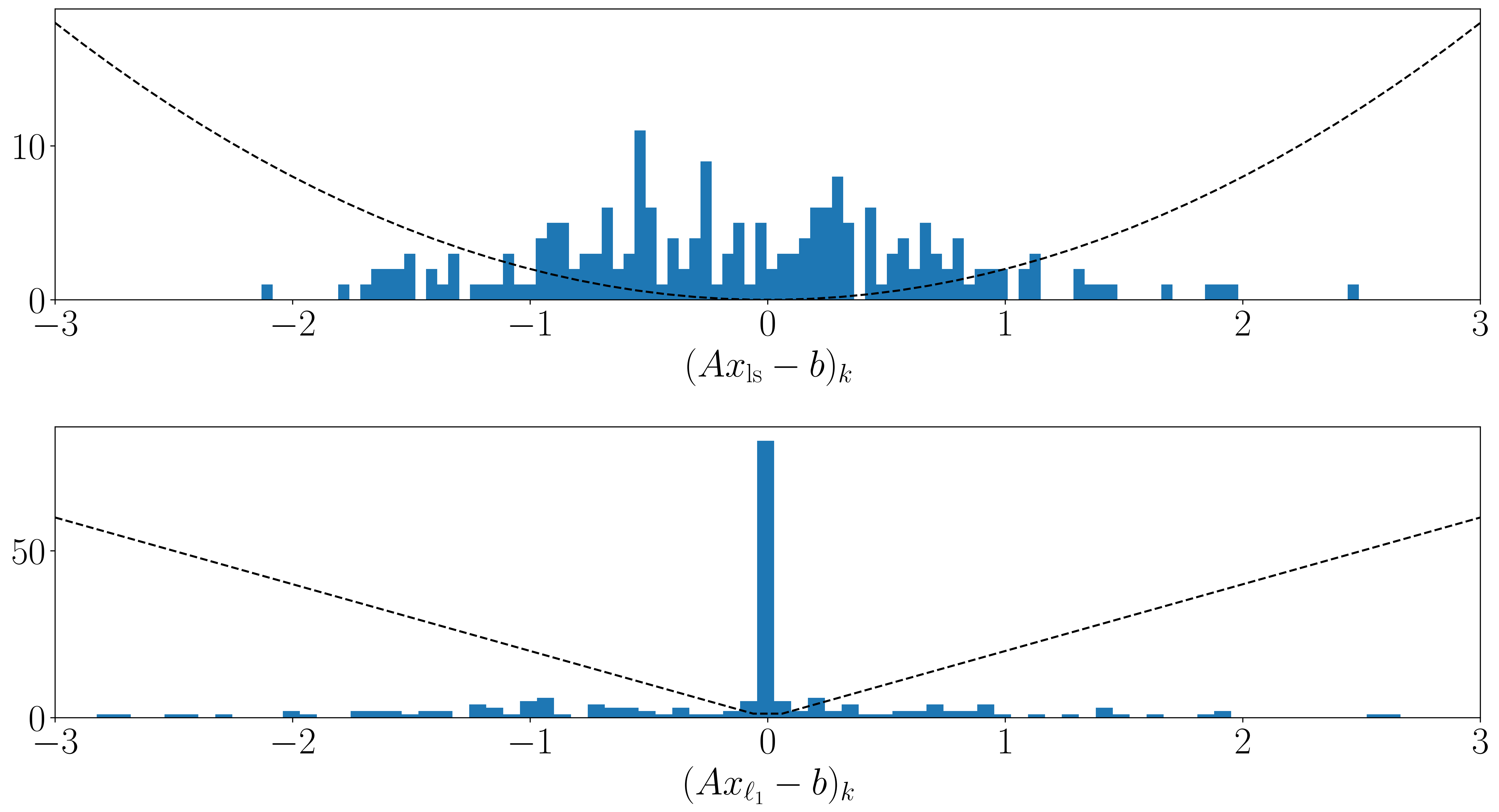
Matrix notation

$$\begin{array}{ll} \text{minimize} & \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}^T \begin{bmatrix} x \\ u \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} A & -I \\ -A & -I \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} b \\ -b \end{bmatrix} \end{array}$$

Comparison with least-squares

Histogram of residuals $Ax - b$ with randomly generated $A \in \mathbf{R}^{200 \times 80}$

$$x_{\text{ls}} = \text{argmin } \|Ax - b\|, \qquad x_{\ell_1} = \text{argmin } \|Ax - b\|_1$$



ℓ_1-norm distribution is **wider** with a **high peak at zero**

ℓ_∞-norm (Chebyshev) approximation

$$\text{minimize} \quad \|Ax - b\|_\infty$$

The ∞**-norm** of m-vector y is

$$\|y\|_\infty = \max_{i=1,\ldots,m} |y_i| = \max_{i=1,\ldots,m} \max\{y_i, -y_i\}$$

Equivalent problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t\mathbf{1} \leq Ax - b \leq t\mathbf{1} \end{array}$$

Matrix notation

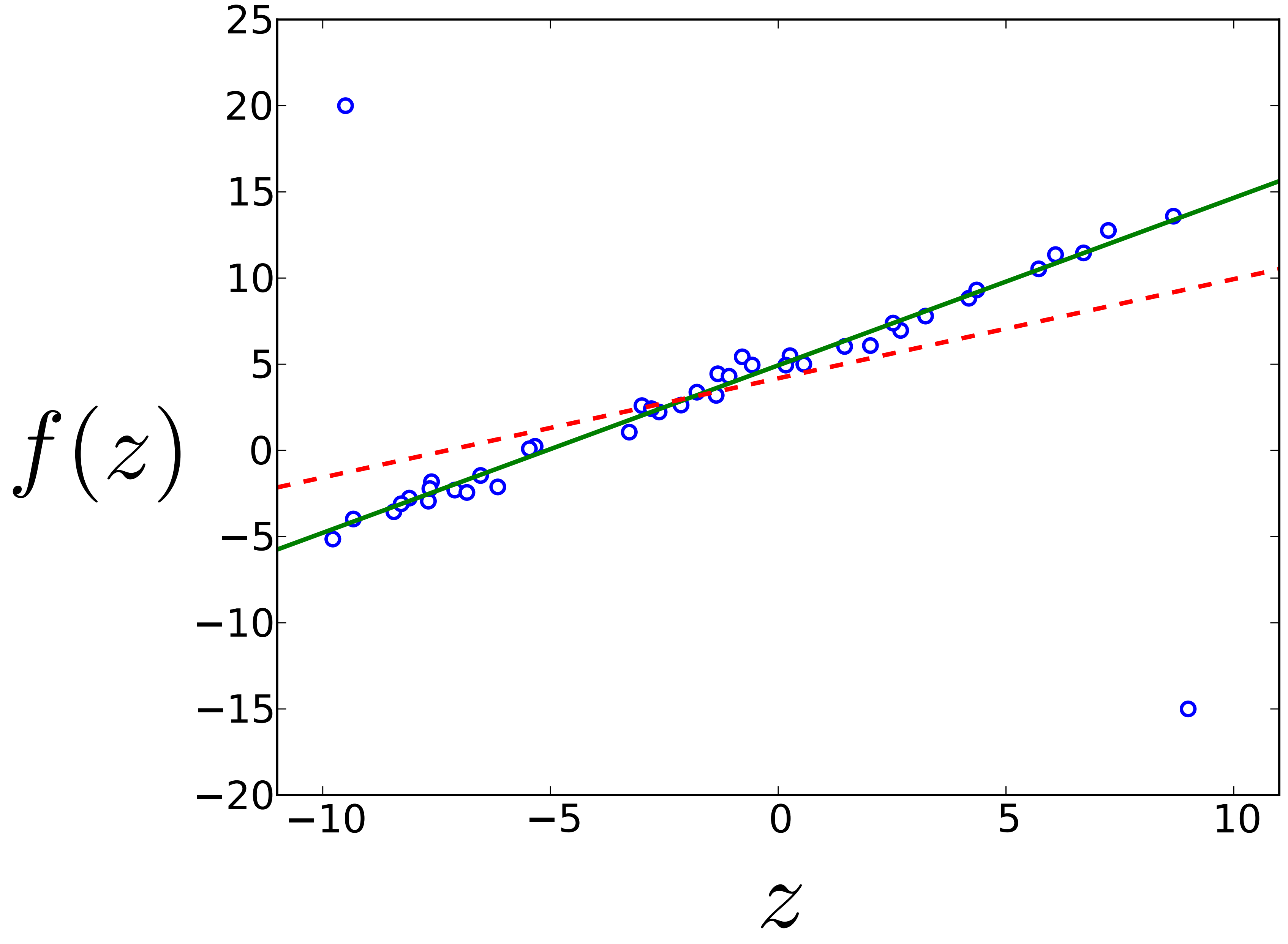
$$\begin{array}{ll} \text{minimize} & \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} x \\ t \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} A & -\mathbf{1} \\ -A & -\mathbf{1} \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \leq \begin{bmatrix} b \\ -b \end{bmatrix} \end{array}$$

Robust curve fitting

Fit affine function $f(z) = \alpha + \beta z$ to m points (z_i, y_i)

Approximation problem $Ax \approx b$ where

$$A = \begin{bmatrix} 1 & z_1 \\ \vdots & \vdots \\ 1 & z_m \end{bmatrix}, \quad x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad b = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$



- Dashed: minimize $\|Ax - b\|$
- Solid: minimize $\|Ax - b\|_1$

ℓ_1-norm approximation **more robust** against outliers

Sparse signal recovery via ℓ_1-norm minimization

$\hat{x} \in \mathbf{R}^n$ is unknown signal, known to be sparse

We make linear measurements $y = A\hat{x}$ with $A \in \mathbf{R}^{m \times n}, m < n$

Estimate signal with smallest ℓ_1-norm, consistent with measurements

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & Ax = y \end{array}$$

Equivalent linear optimization

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T u \\ \text{subject to} & -u \leq x \leq u \\ & Ax = y \end{array}$$

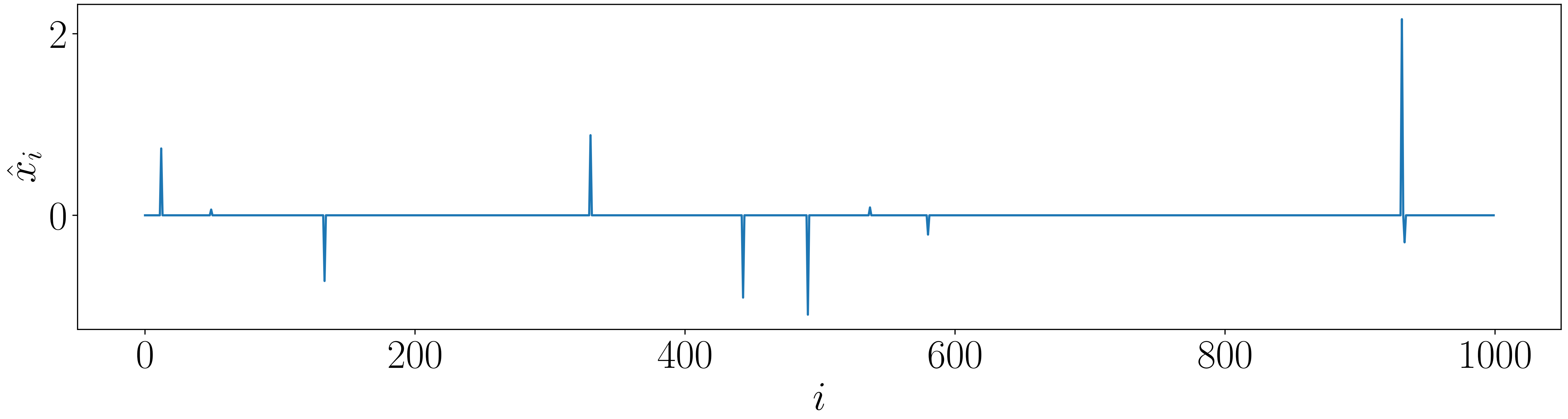
Sparse signal recovery via ℓ_1-norm minimization

Example

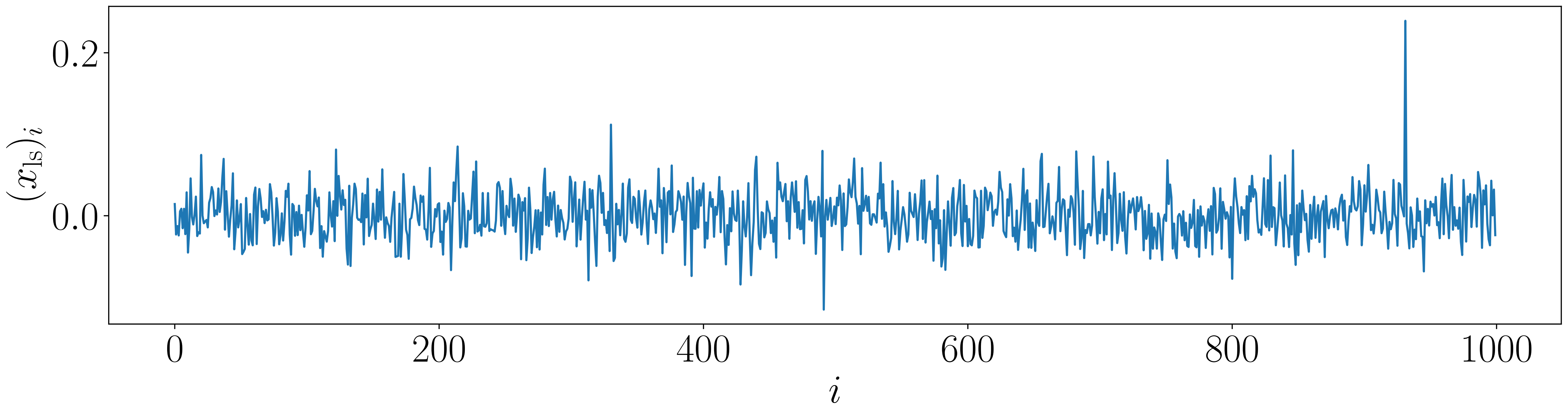
Exact signal $\hat{x} \in \mathbf{R}^{1000}$

10 nonzero components

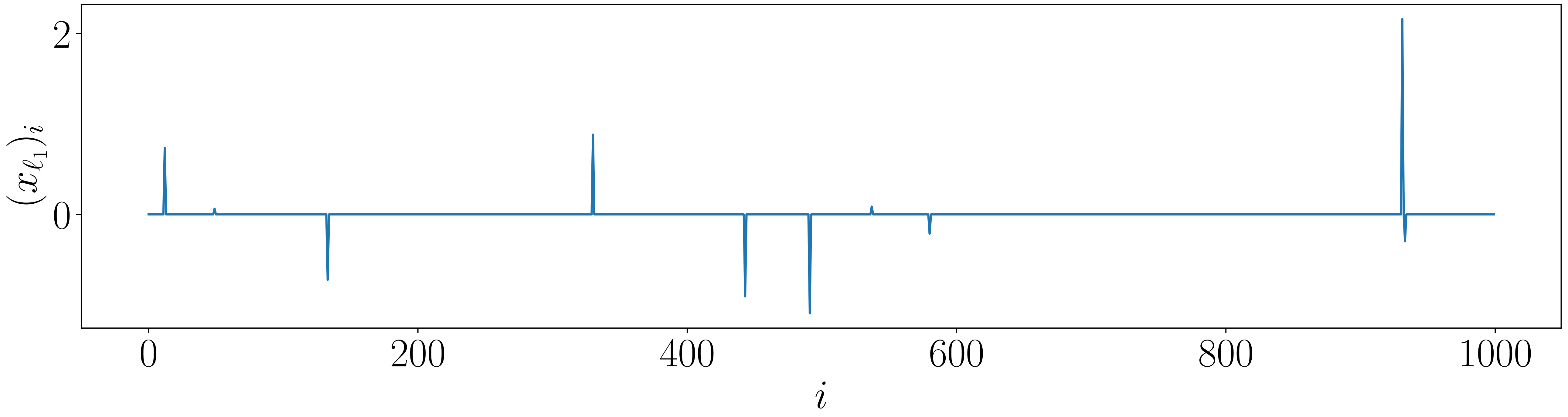
Random $A \in \mathbf{R}^{100 \times 1000}$



The least squares estimate cannot recover the sparse signal



The ℓ_1-norm estimate is **exact**



Sparse signal recovery via ℓ_1-norm minimization

Exact recovery

When are these two problems equivalent?

minimize	$\mathrm{card}(x)$	minimize	$\|x\|_1$
subject to	$Ax = y$	subject to	$Ax = y$

$\mathrm{card}(x)$ is cardinality (number of nonzero components) of x

We say A allows **exact recovery** of k-sparse vectors if

$$\hat{x} = \underset{Ax=y}{\mathrm{argmin}}\, \|x\|_1 \quad \text{when } y = A\hat{x} \text{ and } \mathrm{card}(\hat{x}) \le k$$

It depends on the nullspace[1] of the "measurement matrix" A

1. Feuer & Nemirovski (IEEE Trans. On Information Technology, 2003) and several other papers on compressed sensing.

Scheduling problem

A hospital wants to make a weekly night shift schedule for its nurses

- The demand for the night shift on day j is d_j (7 days)
- Every nurse works 5 days in a row
- Minimize the total number of nurses used while meeting all demand

Scheduling problem

Decision variables

How about x_i to be the number of nurses on day i?

Can't capture the constraints that nurses have to work 5 days in a row!

Let's define x_i to be the number of nurses starting at day i

Objective

minimize $\quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$

Scheduling problem

Formulation

$$
\begin{array}{lllllllllll}
\text{minimize} & \sum_{j=1}^{7} x_j \\
\text{subject to} & x_1+ & & & x_4+ & x_5+ & x_6+ & x_7 & \geq & d_1 \\
& x_1+ & x_2 & & & x_5+ & x_6+ & x_7 & \geq & d_2 \\
& x_1+ & x_2+ & x_3 & & & x_6+ & x_7 & \geq & d_3 \\
& x_1+ & x_2+ & x_3+ & x_4+ & & & x_7 & \geq & d_4 \\
& x_1+ & x_2+ & x_3+ & x_4+ & x_5 & & & \geq & d_5 \\
& & x_2+ & x_3+ & x_4+ & x_5+ & x_6 & & \geq & d_6 \\
& & & x_3+ & x_4+ & x_5+ & x_6+ & x_7 & \geq & d_7 \\
& x_j \geq 0 & \cancel{x_j \text{ integer}} & \text{continuous}
\end{array}
$$

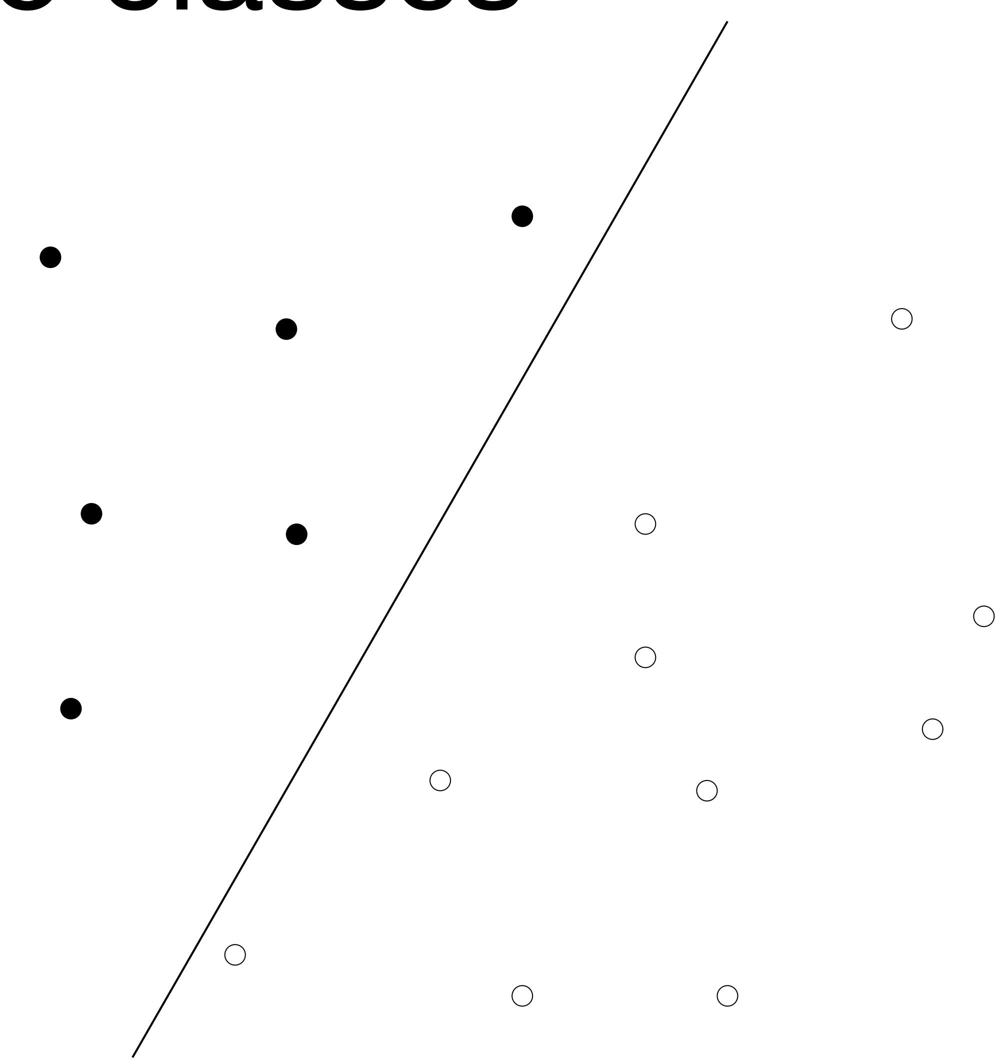
Linear classification

Support vector machine (linear separation)

Given a set of points $\{v_1, \ldots, v_N\}$ with binary labels $s_i \in \{-1, 1\}$

Find hyperplane that strictly separates the tho classes

$$a^T v_i + b > 0 \quad \text{if} \quad s_i = 1$$
$$a^T v_i + b < 0 \quad \text{if} \quad s_i = -1$$



Homogeneous in (a, b), hence equivalent to the linear inequalities (in a, b)

$$s_i(a^T v_i + b) \geq 1$$

Linear classification

Separable case

Feasibility problem

$$\begin{array}{ll} \text{find} & a, b \\ \text{subject to} & s_i(a^T v_i + b) \geq 1, \quad i = 1, \dots, N \end{array}$$

Which can be seen as a special case of LP with

$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & s_i(a^T v_i + b) \geq 1, \quad i = 1, \dots, N \end{array}$$

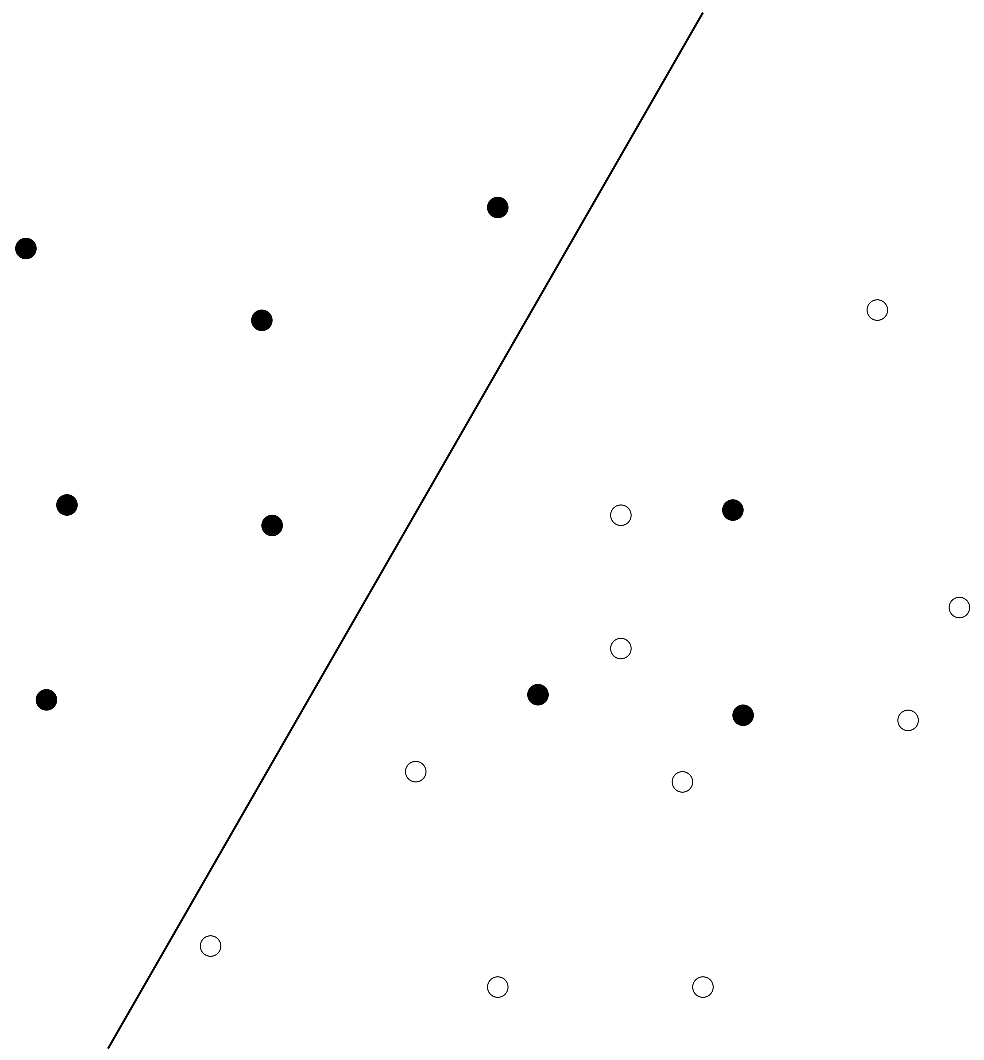
$p^\star = 0$ if problem feasible (points separable)

$p^\star = \infty$ if problem infeasible (points not separable) $\longrightarrow$ **What then?**

Linear classification

Approximate linear separation of non-separable points

minimize $\quad \sum_{i=1}^{N}(1 - s_i(a^T v_i + b))_+ = \sum_{i=1}^{N} \max\{0, 1 - s_i(a^T v_i + b)\}$



If v_i misclassified, $1 - s_i(a_i^T v_i + b)$ is the penalty

Piecewise-linear minimization problem with variables a, b

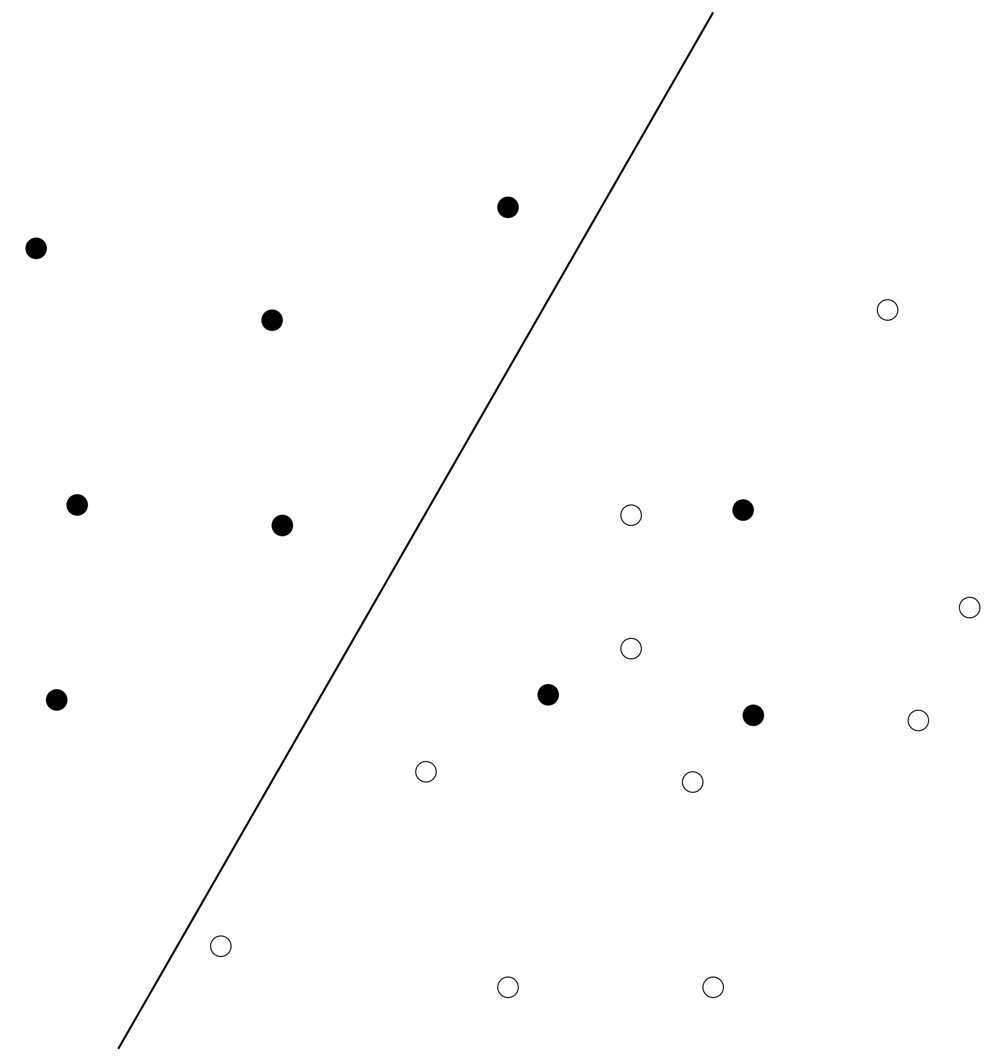
Linear classification

Approximate linear separation of non-separable points

minimize $\quad \sum_{i=1}^{N} \max\{0, 1 - s_i(a^T v_i + b)\}$

Equivalent problem

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{N} u_i \\
\text{subject to} & 1 - s_i(v_i^T a + b) \leq u_i, \quad i = 1, \ldots, N \\
& 0 \leq u_i, \quad i = 1, \ldots, N
\end{array}
$$



Matrix notation?

Modelling software 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
- Express the problem in input format required by a specific LP solver

Examples

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

CVXPY example

$$
\begin{array}{ll}
\text{minimize} & \|Ax - b\|_1 \\
\text{subject to} & 0 \leq x \leq 1
\end{array}
$$

```python
x = cp.Variable(n)
objective = cp.Minimize(cp.norm(A*x - b, 1))
constraints = [0 <= x, x <= 1]
problem = cp.Problem(objective, constraints)

# The optimal objective value is returned by `problem.solve()`.
result = problem.solve()

# The optimal value for x is stored in `x.value`.
print(x.value)
```

Why linear optimization?

Easy to solve

- It is solvable in polynomial time, and it is 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

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

Fundamental

The theory of linear optimization lays the foundation for most optimization theories

Next lecture

Geometry of linear optimization

- Polyhedra
- Extreme points
- Basic feasible solutions