

# **ORF307 – Optimization**

## **10. Applications of linear optimization**

# Ed Forum

- **Midterm March 09**

Time: 11:00am — 12:20pm

Location: Same as lecture

Topics: Up to last lecture (excluding equivalence theorem)

Material allowed: Single sheet of paper. Double sided. Hand-written or typed.

- **Questions**

- What is a basic feasible solution? how we solve for those?

$$\begin{array}{ll} \min & c^T x \\ \text{st.} & Ax = b \\ & x \geq 0 \end{array}$$

# Recap

# Constructing a basic solution

**Two equalities** ( $m = 2, n = 3$ )

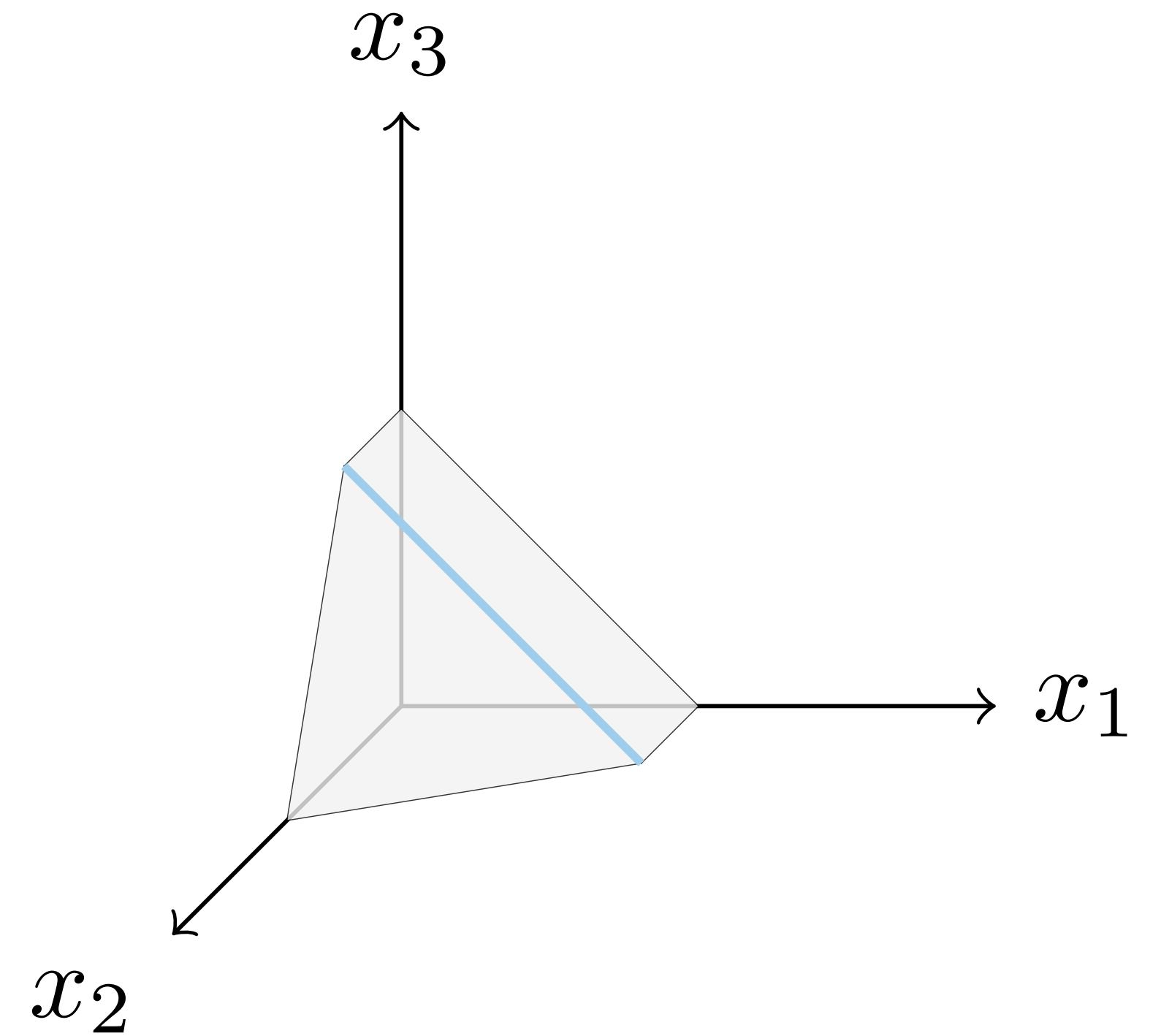
minimize  $c^T x$

subject to  $x_1 + x_3 = 1$

$(1/2)x_1 + x_2 + (1/2)x_3 = 1$

$x_1, x_2, x_3 \geq 0$

$n - m = 1$  inequalities have to be tight:  $x_i = 0$



# Constructing a basic solution

Two equalities ( $m = 2, n = 3$ )

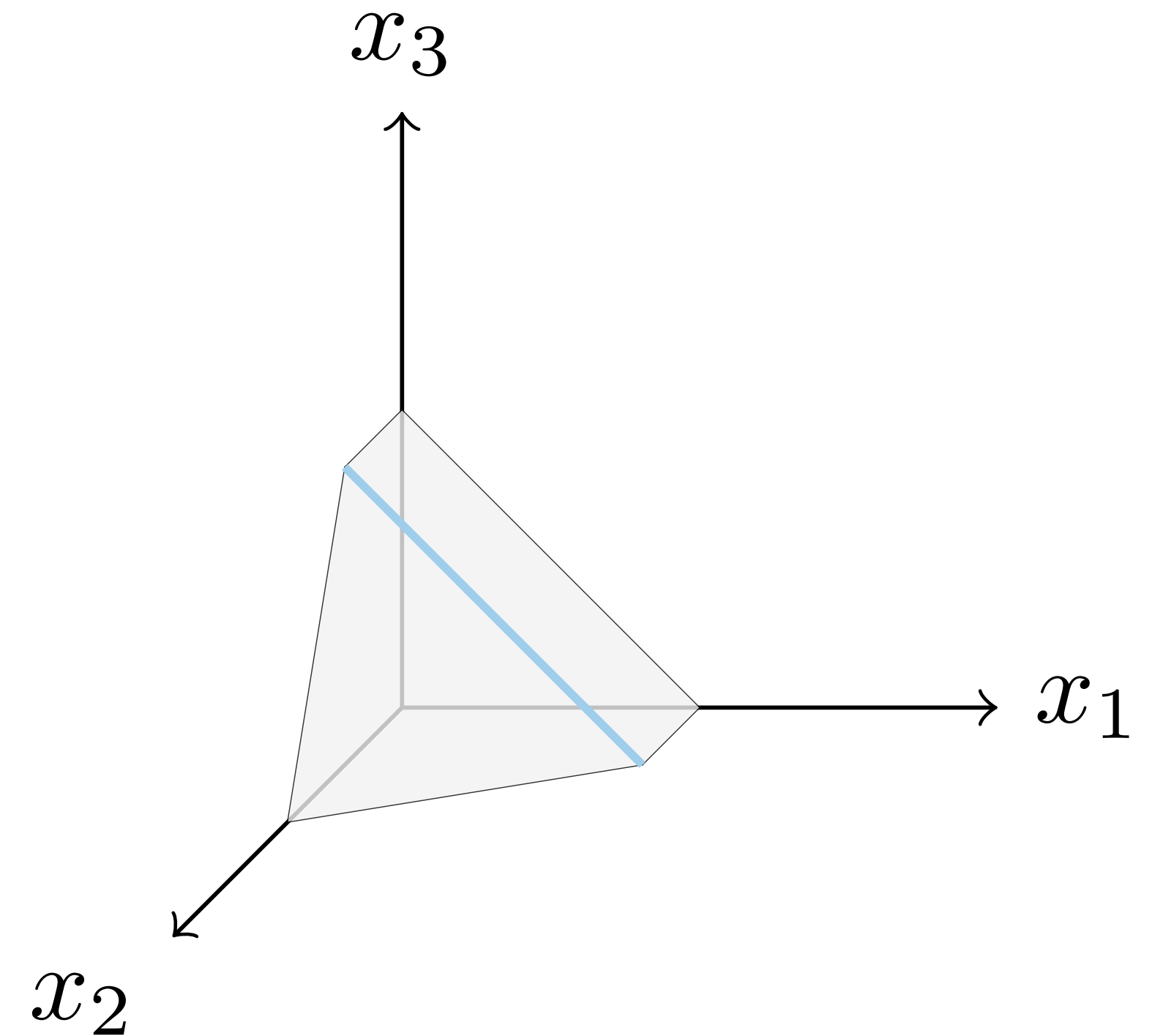
minimize  $c^T x$

subject to  $x_1 + x_3 = 1$

$(1/2)x_1 + x_2 + (1/2)x_3 = 1$

$x_1, x_2, x_3 \geq 0$

$n - m = 1$  inequalities have to be tight:  $x_i = 0$



Set  $x_1 = 0$  and solve

$$\begin{bmatrix} 1 & 0 & 1 \\ 1/2 & 1 & 1/2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 1/2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

# Constructing a basic solution

Two equalities ( $m = 2, n = 3$ )

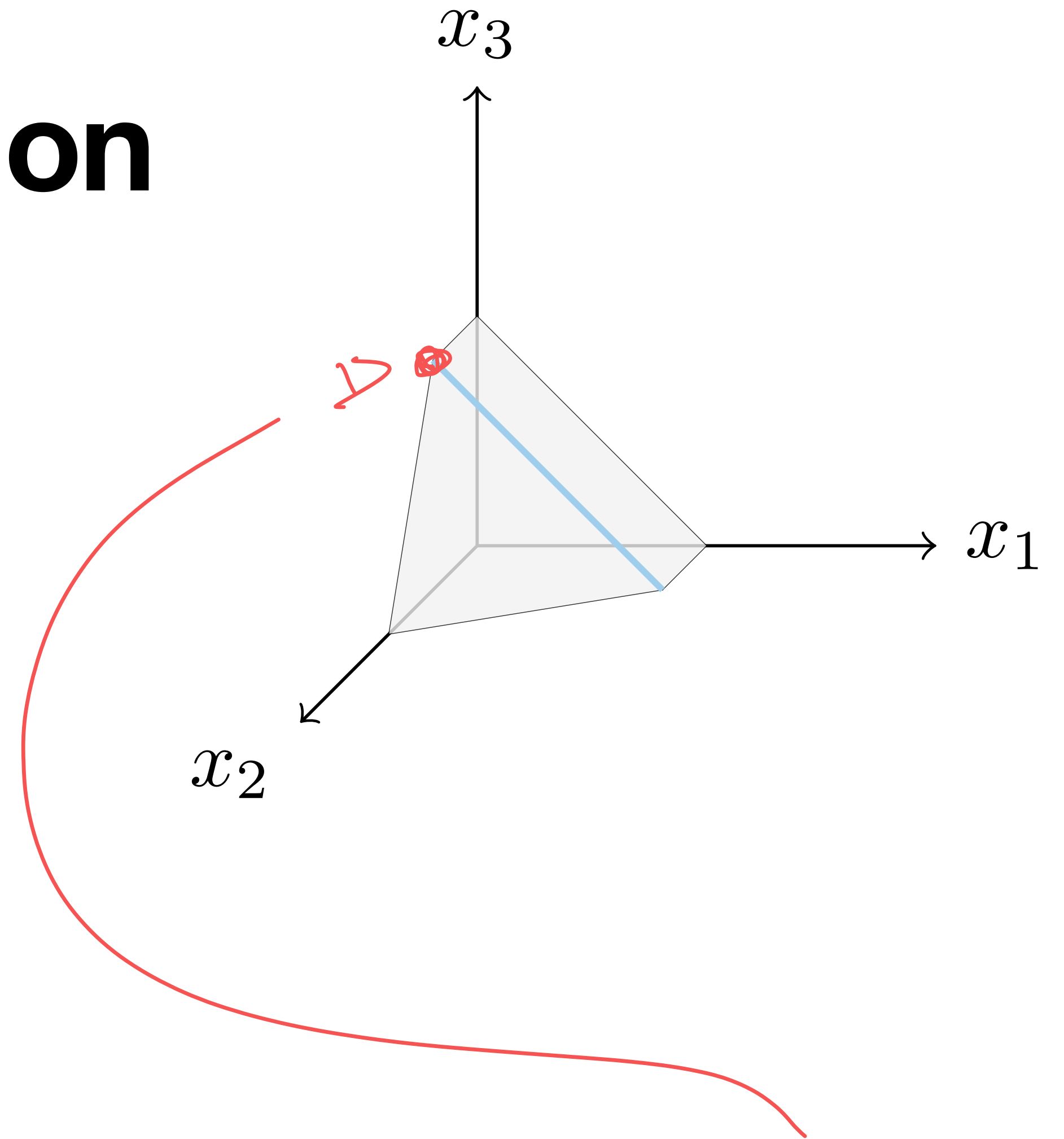
minimize  $c^T x$

subject to  $x_1 + x_3 = 1$

$(1/2)x_1 + x_2 + (1/2)x_3 = 1$

$x_1, x_2, x_3 \geq 0$

$n - m = 1$  inequalities have to be tight:  $x_i = 0$



Set  $x_1 = 0$  and solve

$$\begin{bmatrix} 1 & 0 & 1 \\ 1/2 & 1 & 1/2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 1/2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow (x_2, x_3) = \underline{\underline{(0.5, 1)}}$$

# Basic solutions

## Standard form polyhedra

$$P = \{x \mid Ax = b, x \geq 0\} \quad \text{with} \quad A \in \mathbf{R}^{m \times n} \text{ has full row rank } m \leq n$$

# Basic solutions

## Standard form polyhedra

$$P = \{x \mid Ax = b, x \geq 0\} \quad \text{with} \quad A \in \mathbf{R}^{m \times n} \text{ has full row rank } m \leq n$$

$x$  is a **basic solution** if and only if

- $Ax = b$
- There exist indices  $B(1), \dots, B(m)$  such that
  - columns  $A_{B(1)}, \dots, A_{B(m)}$  are linearly independent
  - $x_i = 0$  for  $i \neq B(1), \dots, B(m)$



# Basic solutions

## Standard form polyhedra

$$P = \{x \mid Ax = b, x \geq 0\} \quad \text{with} \quad A \in \mathbf{R}^{m \times n} \text{ has full row rank } m \leq n$$

$x$  is a **basic solution** if and only if

- $Ax = b$
- There exist indices  $B(1), \dots, B(m)$  such that
  - columns  $A_{B(1)}, \dots, A_{B(m)}$  are linearly independent
  - $x_i = 0$  for  $i \neq B(1), \dots, B(m)$

$x$  is a **basic feasible solution** if  $x$  is a **basic solution** and  $x \geq 0$

# Constructing basic solution

1. Choose any  $m$  independent columns of  $A$ :  $A_{B(1)}, \dots, A_{B(m)}$
2. Let  $x_i = 0$  for all  $i \neq B(1), \dots, B(m)$
3. Solve  $Ax = b$  for the remaining  $x_{B(1)}, \dots, x_{B(m)}$

# Constructing basic solution

1. Choose any  $m$  independent columns of  $A$ :  $A_{B(1)}, \dots, A_{B(m)}$
2. Let  $x_i = 0$  for all  $i \neq B(1), \dots, B(m)$
3. Solve  $Ax = b$  for the remaining  $x_{B(1)}, \dots, x_{B(m)}$

Basis  
matrix

Basis columns

Basic variables

$$A_B = \left[ \begin{array}{c|c|c|c} & & & \\ \hline & & & \\ A_{B(1)} & A_{B(2)} & \dots & A_{B(m)} \\ \hline & & & \\ & & & \end{array} \right], \quad x_B = \begin{bmatrix} x_{B(1)} \\ \vdots \\ x_{B(m)} \end{bmatrix} \longrightarrow \text{Solve } A_B x_B = b$$

# Constructing basic solution

$$\begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

1. Choose any  $m$  independent columns of  $A$ :  $A_{B(1)}, \dots, A_{B(m)}$
2. Let  $x_i = 0$  for all  $i \neq B(1), \dots, B(m)$
3. Solve  $Ax = b$  for the remaining  $x_{B(1)}, \dots, x_{B(m)}$

Basis matrix

Basis columns

Basic variables

$$A_B = \begin{bmatrix} | & | & & | \\ A_{B(1)} & A_{B(2)} & \dots & A_{B(m)} \\ | & | & & | \end{bmatrix}, \quad x_B = \begin{bmatrix} x_{B(1)} \\ \vdots \\ x_{B(m)} \end{bmatrix} \longrightarrow \text{Solve } A_B x_B = b$$

If  $x_B \geq 0$ , then  $x$  is a **basic feasible solution**

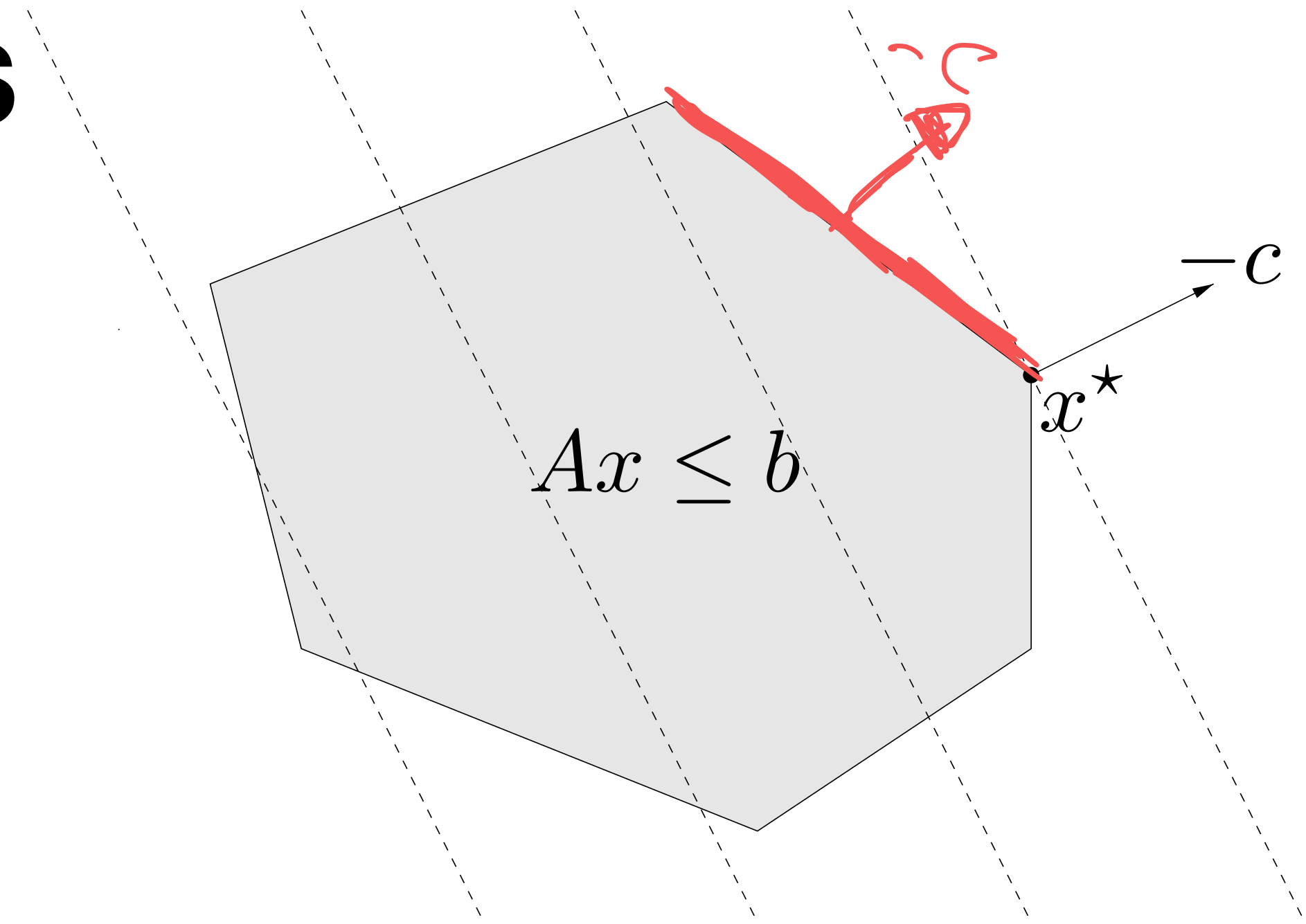
# Optimality of extreme points

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

If

- $P$  has at least one extreme point
- There exists an optimal solution  $x^*$

Then, there exists an optimal solution that is an **extreme point** of  $P$ .

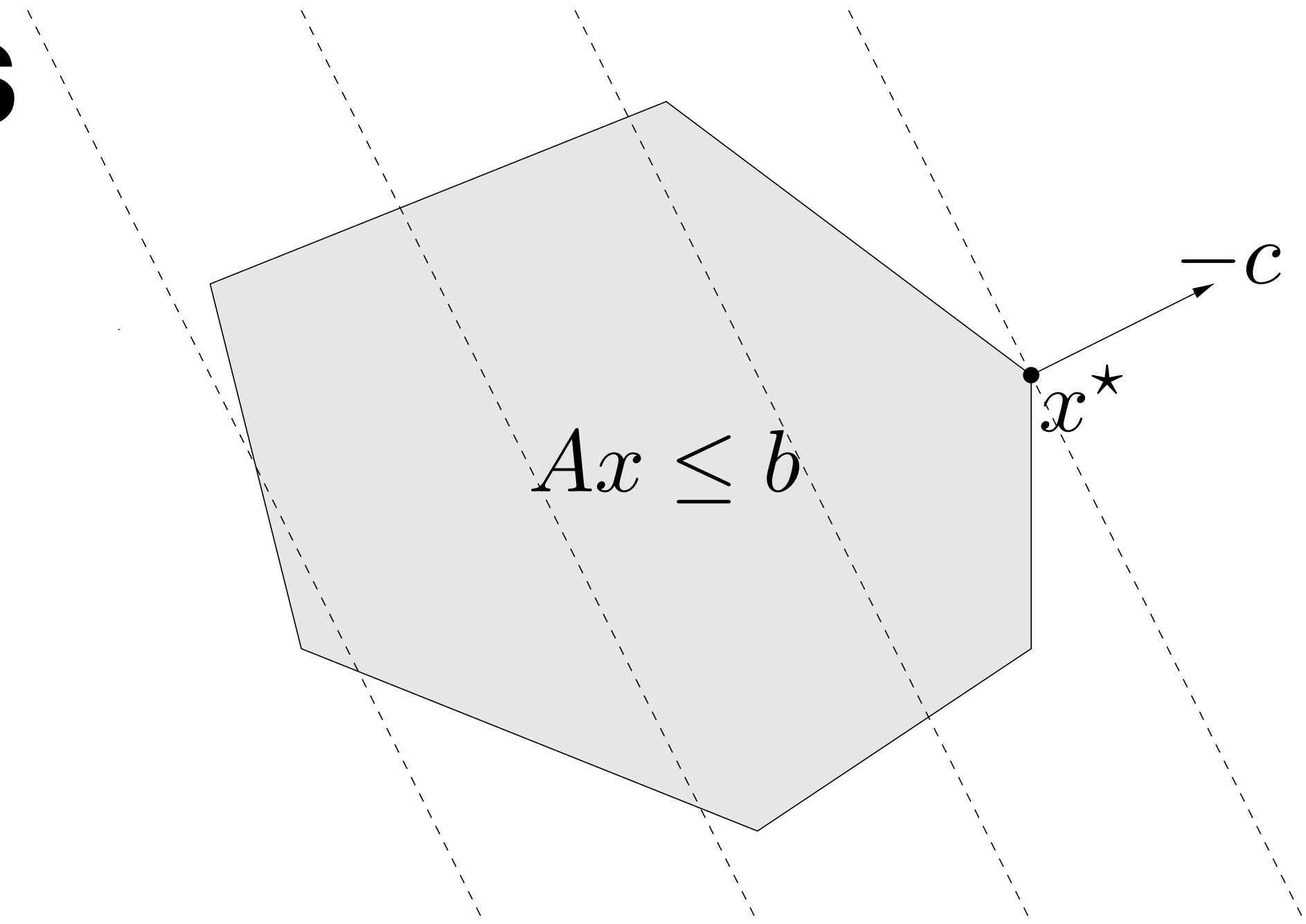


# Optimality of extreme points

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

If

- $P$  has at least one extreme point
- There exists an optimal solution  $x^*$



Then, there exists an optimal solution that is an **extreme point** of  $P$ .

Solution method: restrict search to **extreme points**.

# How to search among basic feasible solutions?

# How to search among basic feasible solutions?

## Idea

List all the basic feasible solutions, compare objective values and pick the best one.



# How to search among basic feasible solutions?

## Idea

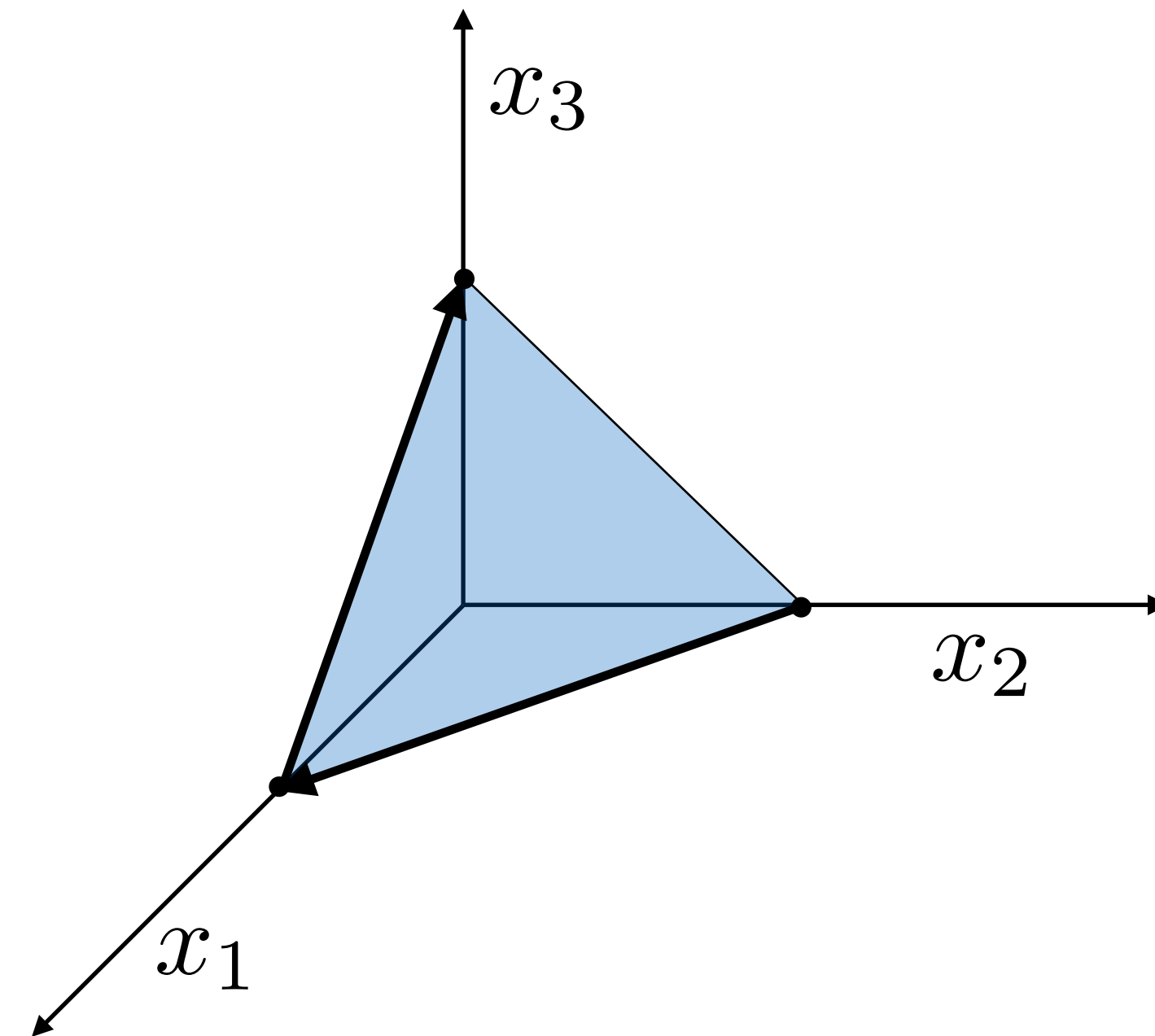
List all the basic feasible solutions, compare objective values and pick the best one.

## Intractable!

If  $n = 1000$  and  $m = 100$ , we have  $10^{143}$  combinations!

# Conceptual algorithm

- Start at corner
- Visit neighboring corner that improves the objective



# Today's agenda

## Applications of linear optimization

- Optimal control
- Character recognition
- Portfolio optimization

# Optimal control

# Optimal control problems

## Linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad t = 1, 2, \dots$$

$$y_t = Cx_t, \quad t = 1, 2, \dots$$

- The  $n$ -vector  $x_t$  is the *state* at time  $t$
- The  $m$ -vector  $u_t$  is the *input* at time  $t$
- The  $p$ -vector  $y_t$  is the *output* at time  $t$
- The  $n \times n$  matrix  $A$  is the *dynamics matrix*
- The  $n \times m$  matrix  $B$  is the *input matrix*
- The  $p \times n$  matrix  $C$  is the *output matrix*

# Optimal control problems

## Linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad t = 1, 2, \dots$$

$$y_t = Cx_t, \quad t = 1, 2, \dots$$

- The  $n$ -vector  $x_t$  is the *state* at time  $t$
- The  $m$ -vector  $u_t$  is the *input* at time  $t$
- The  $p$ -vector  $y_t$  is the *output* at time  $t$
- The  $n \times n$  matrix  $A$  is the *dynamics matrix*
- The  $n \times m$  matrix  $B$  is the *input matrix*
- The  $p \times n$  matrix  $C$  is the *output matrix*

## Simulation

- The sequence  $x_1, x_2, \dots$  is called *state trajectory*
- The sequence  $y_1, y_2, \dots$  is called *output trajectory*
- **Goal:** Given  $x_1, u_1, u_2, \dots$ , find  $x_2, x_3, \dots$  and  $y_2, y_3, \dots$
- Obtained by recursion. For  $t = 1, 2, \dots$ , compute  
 $x_{t+1} = Ax_t + Bu_t$  and  $y_t = Cx_t$

# Optimal control problem

## Linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad t = 1, 2, \dots$$

$$y_t = Cx_t, \quad t = 1, 2, \dots$$

## The problem

- The *initial state*  $x_1 = x^{\text{init}}$  is given
- **Goal.** Choose  $u_1, u_2, \dots, u_{T-1}$  to achieve some goals, e.g.,
  - Get to desired final state  $x_T = x^{\text{des}}$
  - Minimize the input effort (make  $\|u_t\|$  small for all  $t$ )
  - Track desired output  $y_t^{\text{des}}$  (make  $\|y_t - y_t^{\text{des}}\|$  small for all  $t$ )

# Least squares optimal control problem

$$\begin{aligned} &\text{minimize} && \sum_{t=1}^T \|y_t - y_t^{\text{des}}\|^2 + \rho \sum_{t=1}^{T-1} \|u_t\|^2 \\ &\text{subject to} && x_{t+1} = Ax_t + Bu_t, \quad t = 1, \dots, T-1 \\ &&& y_t = Cx_t, \quad t = 1, \dots, T \\ &&& x_1 = x^{\text{init}} \end{aligned}$$

## Remarks

- The variables are  $x_2, \dots, x_T, y_2, \dots, y_T$ , and  $u_1, \dots, u_{T-1}$
- Parameter  $\rho > 0$  controls trade off between control "energy" and tracking error
- It is a multi-objective and constrained least squares problem



# 1-norm optimal control problem

$$\begin{aligned} &\text{minimize} && \sum_{t=1}^T \|y_t - y_t^{\text{des}}\|_1 + \rho \sum_{t=1}^{T-1} \|u_t\|_1 \\ &\text{subject to} && \begin{cases} x_{t+1} = Ax_t + Bu_t, & t = 1, \dots, T-1 \\ y_t = Cx_t, & t = 1, \dots, T \\ Dx_t \leq d, & t = 1, \dots, T \\ Eu_t \leq e, & t = 1, \dots, T-1 \\ x_1 = x^{\text{init}} \end{cases} \end{aligned}$$

## Remarks

- $\|\cdot\|_1$  instead of  $\|\cdot\|_2^2$
- Linear inequality constraints:  $Dx_t \leq d$  for states and  $Eu_t \leq e$  for inputs
- Is a linear optimization problem (with additional variables)

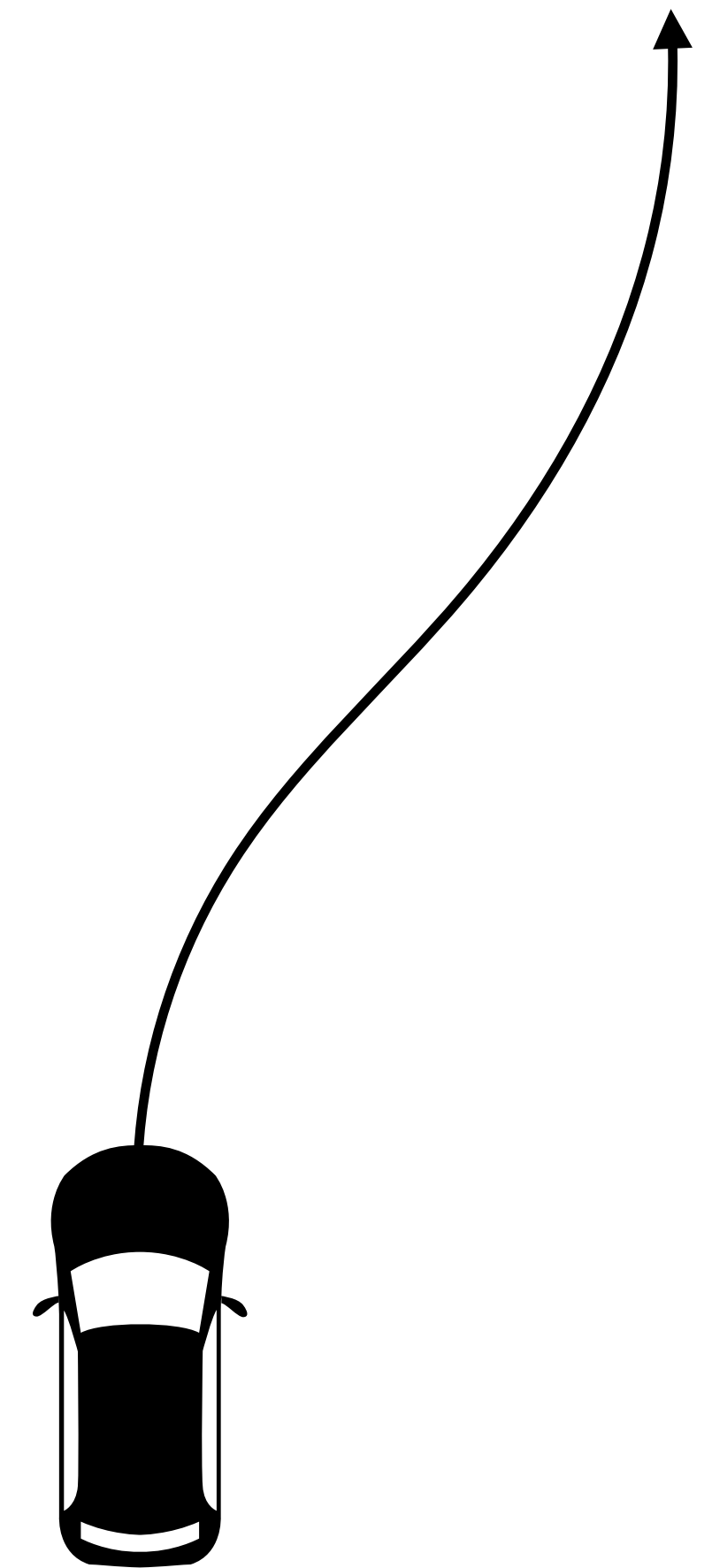
# Vehicle example in a plane

DISCRETIZATION  
INTERVAL

Sample position and velocity at times  $\tau = 0, h, 2h, \dots$

## Vehicle with mass $m$

- 2-vector  $p_t$  is the position at time  $ht$
- 2-vector  $v_t$  is the velocity at time  $ht$
- 2-vector  $u_t$  is the force applied at time  $ht$
- $-\eta v_t$  is the friction force applied at  $ht$

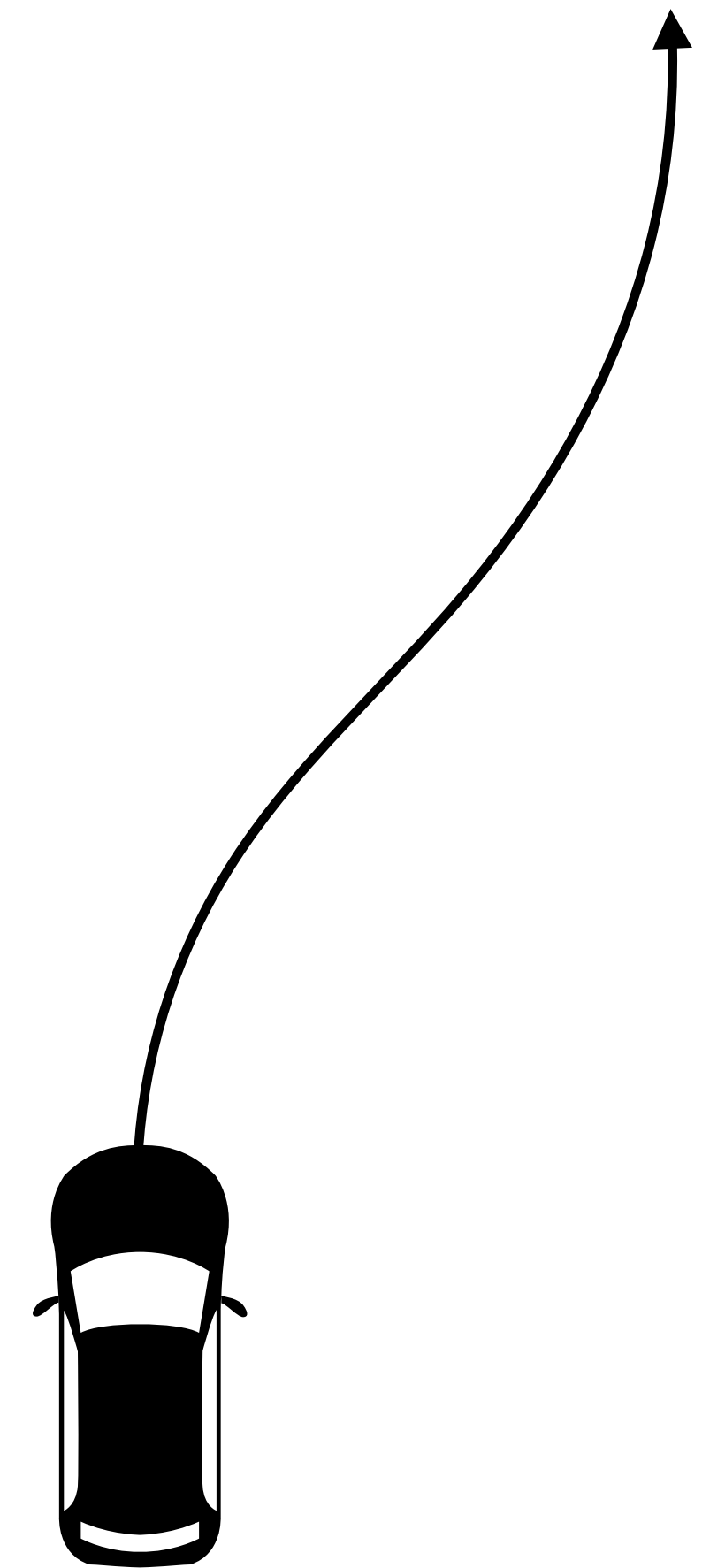


# Vehicle example in a plane

Sample position and velocity at times  $\tau = 0, h, 2h, \dots$

**Vehicle with mass  $m$**

- 2-vector  $p_t$  is the position at time  $ht$
- 2-vector  $v_t$  is the velocity at time  $ht$
- 2-vector  $u_t$  is the force applied at time  $ht$
- $-\eta v_t$  is the friction force applied at  $ht$



**Small time interval  $h$**

$$\frac{p_{t+1} - p_t}{h} \approx v_t$$

$$m \frac{v_{t+1} - v_t}{h} \approx -h v_t + u_t$$

$$p_{t+1} = p_t + h v_t$$

$$v_{t+1} = (1 - h\eta/m)v_t + (h/m)u_t$$

# Vehicle example in a plane

## State

4-vector  $x_t = (p_t, v_t)$

## Laws of physics

$$p_{t+1} = p_t + hv_t$$

$$v_{t+1} = (1 - h\eta/m)v_t + (h/m)u_t$$

output = position

$$y_t = p_t$$

# Vehicle example in a plane

## State

4-vector  $x_t = (p_t, v_t)$

## Dynamics

$$x_{t+1} = Ax_t + Bu_t$$

$$y_t = Cx_t$$

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 - h\eta/m & 0 \\ 0 & 0 & 0 & 1 - h\eta/m \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m & 0 \\ 0 & h/m \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

## Laws of physics

$$p_{t+1} = p_t + hv_t$$

$$v_{t+1} = (1 - h\eta/m)v_t + (h/m)u_t$$

output = position

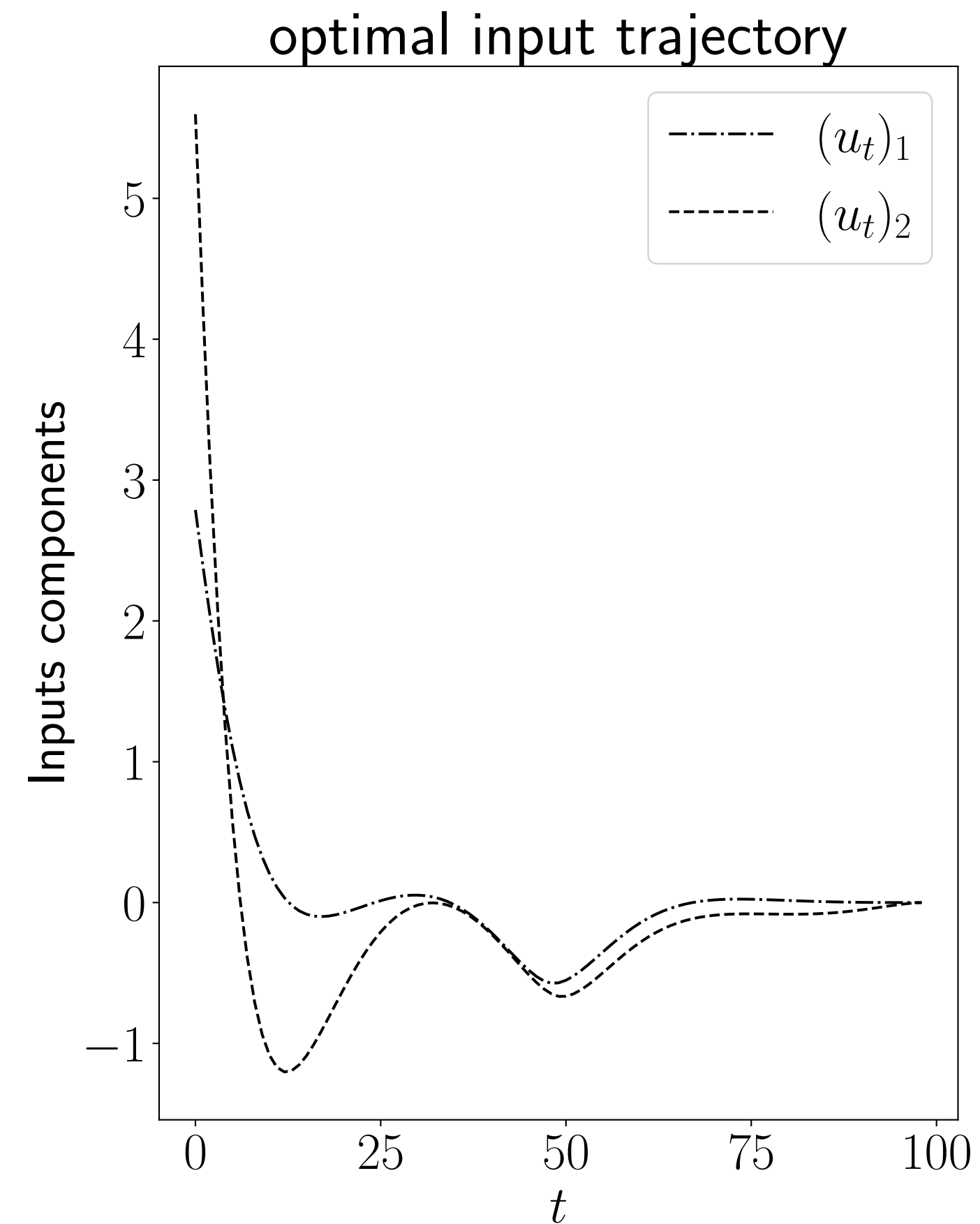
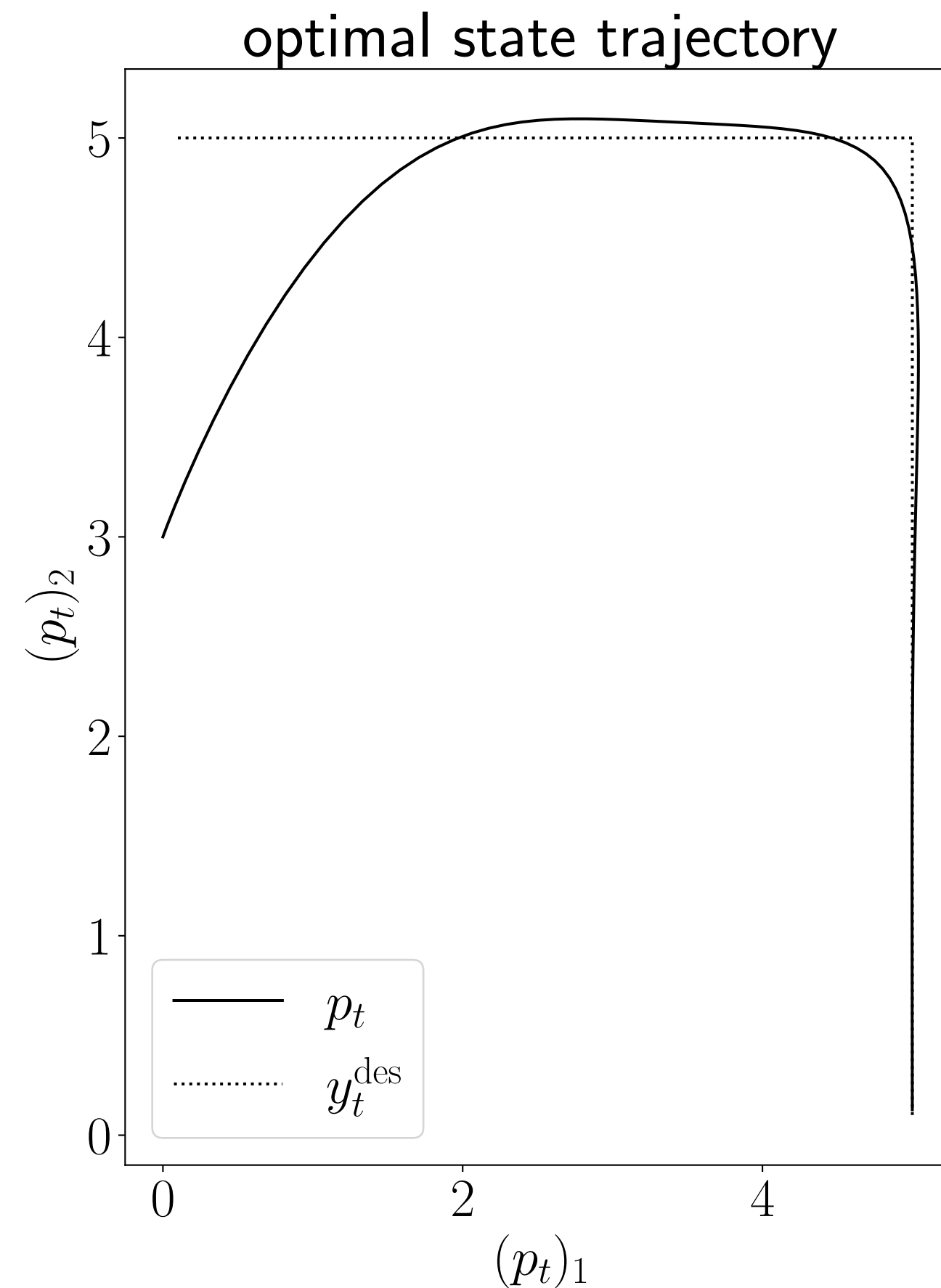
$$y_t = p_t$$

# Vehicle example with output tracking

## Least squares results

### Parameters

$$T = 100, \quad h = 0.1, \quad \eta = 0.1, \quad m = 1$$

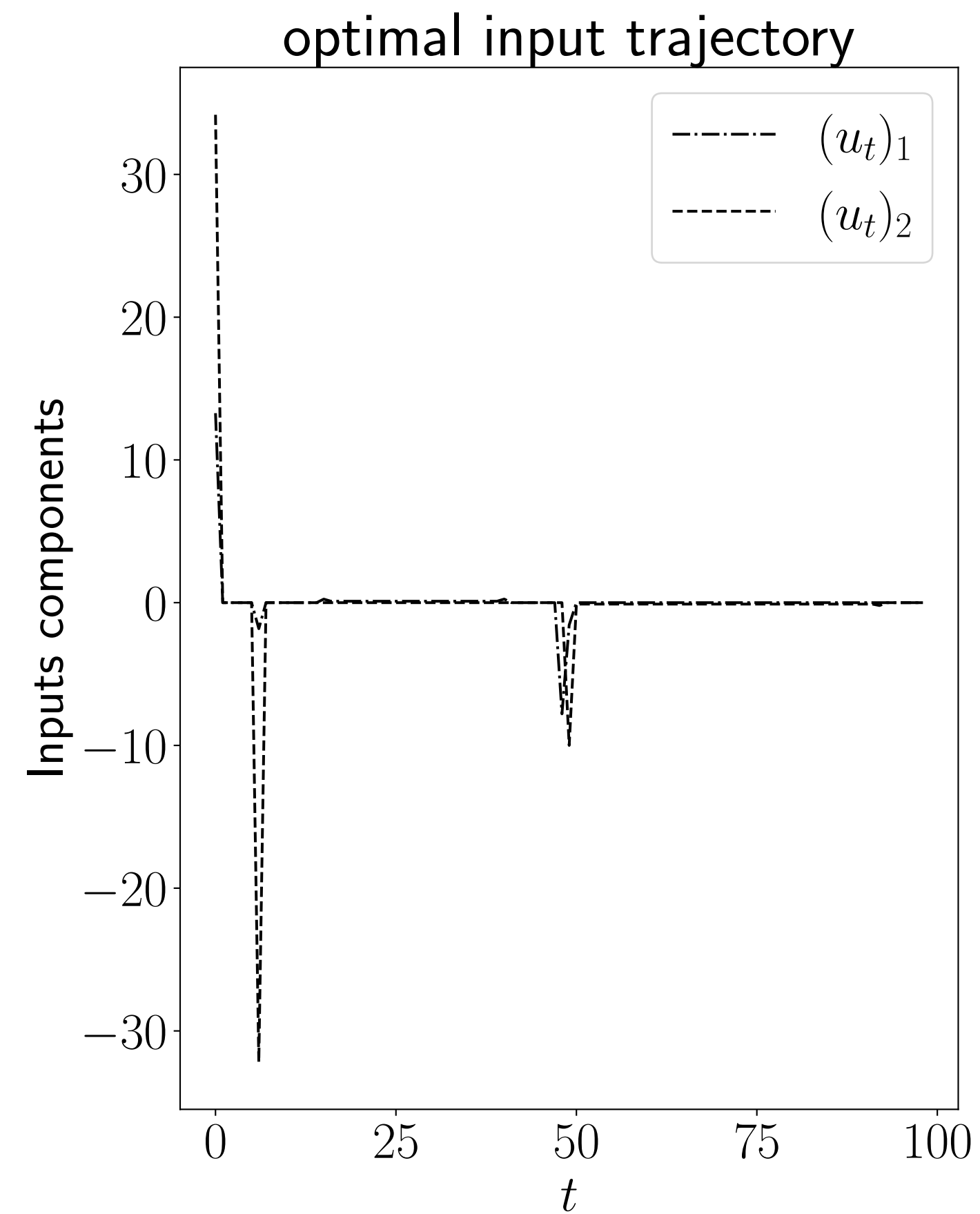
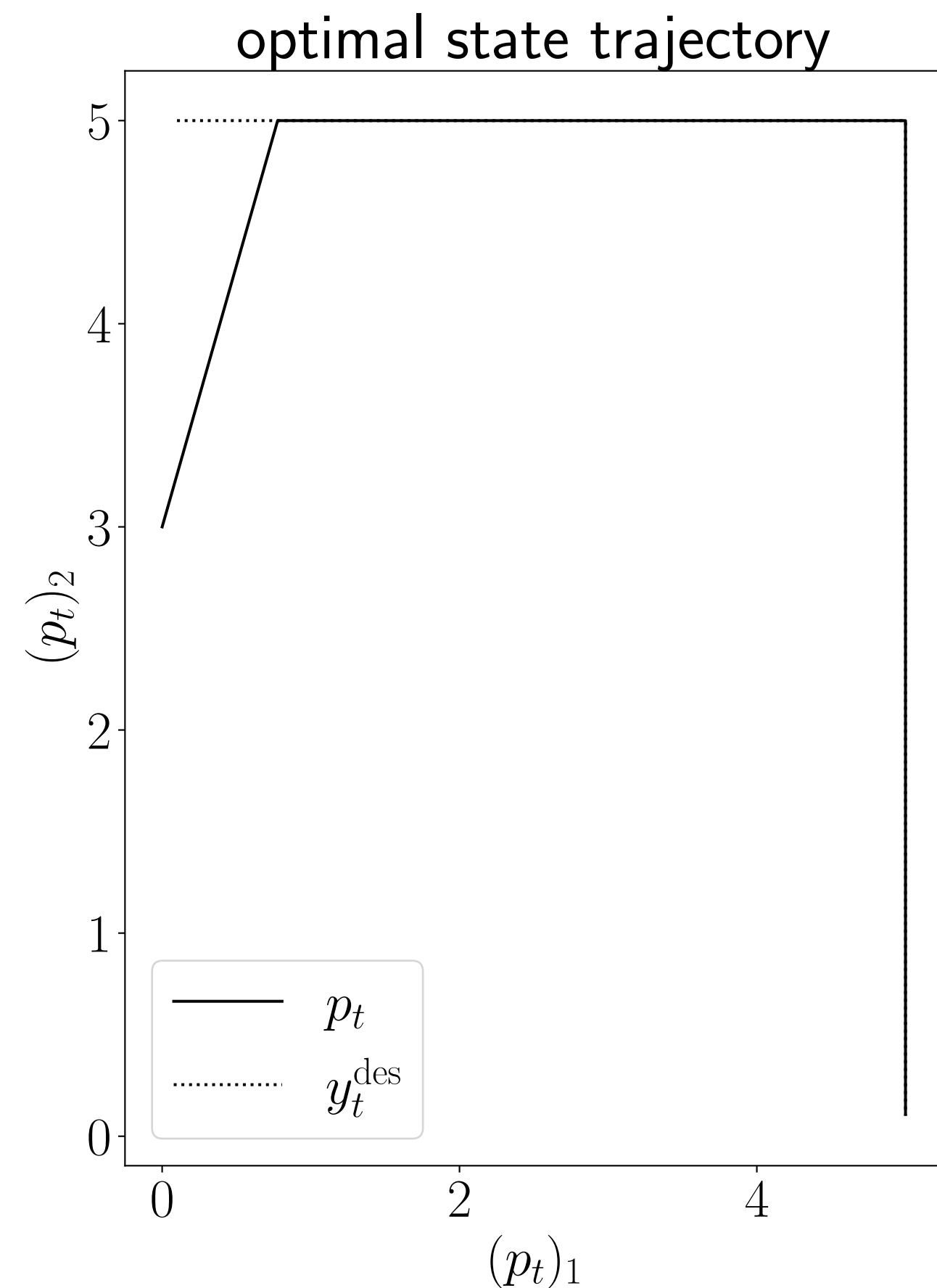


# Vehicle example with output tracking

## 1-norm results

### Parameters

$$T = 100, \quad h = 0.1, \quad \eta = 0.1, \quad m = 1$$



# Vehicle example with output tracking

## 1-norm with constraints

Linear optimization can have more interesting constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{t=1}^T \|y_t - y_t^{\text{des}}\|_1 + \rho \sum_{t=1}^{T-1} \|u_t\|_1 \\ \text{subject to} \quad & x_{t+1} = Ax_t + Bu_t, \quad t = 1, \dots, T-1 \\ & y_t = Cx_t, \quad t = 1, \dots, T \\ & \|u_t\|_\infty \leq u^{\text{max}}, \quad t = 1, \dots, T-1 \\ & \|u_t - u_{t-1}\|_1 \leq s^{\text{max}}, \quad t = 1, \dots, T-1 \\ & x_1 = x^{\text{init}} \end{aligned}$$



# Vehicle example with output tracking

## 1-norm with constraints

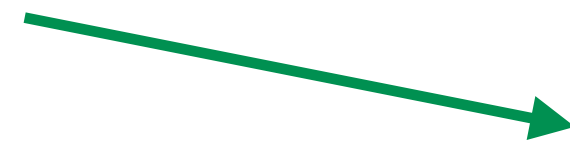
Linear optimization can have more interesting constraints

minimize  $\sum_{t=1}^T \|y_t - y_t^{\text{des}}\|_1 + \rho \sum_{t=1}^{T-1} \|u_t\|_1$

subject to  $x_{t+1} = Ax_t + Bu_t, \quad t = 1, \dots, T-1$

$y_t = Cx_t, \quad t = 1, \dots, T$

max-input



$\|u_t\|_\infty \leq u^{\text{max}}, \quad t = 1, \dots, T-1$

$\|u_t - u_{t-1}\|_1 \leq s^{\text{max}}, \quad t = 1, \dots, T-1$

$x_1 = x^{\text{init}}$

# Vehicle example with output tracking

## 1-norm with constraints

Linear optimization can have more interesting constraints

minimize  $\sum_{t=1}^T \|y_t - y_t^{\text{des}}\|_1 + \rho \sum_{t=1}^{T-1} \|u_t\|_1$   
subject to  $x_{t+1} = Ax_t + Bu_t, \quad t = 1, \dots, T-1$   
 $y_t = Cx_t, \quad t = 1, \dots, T$

$\|u_t\|_\infty \leq u^{\text{max}}, \quad t = 1, \dots, T-1$   
 $\|u_t - u_{t-1}\|_1 \leq s^{\text{max}}, \quad t = 1, \dots, T-1$   
 $x_1 = x^{\text{init}}$

max-input



max-input variation

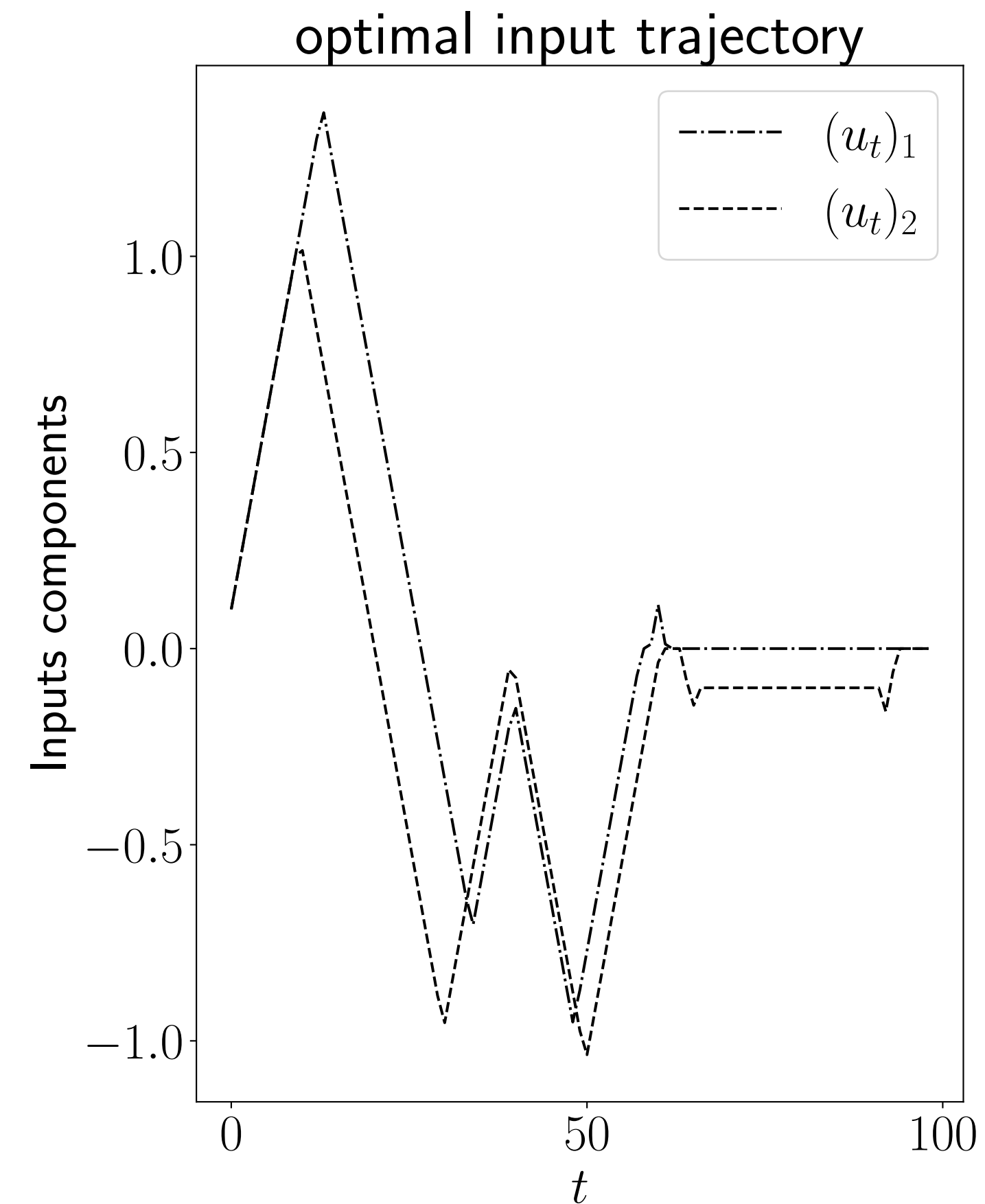
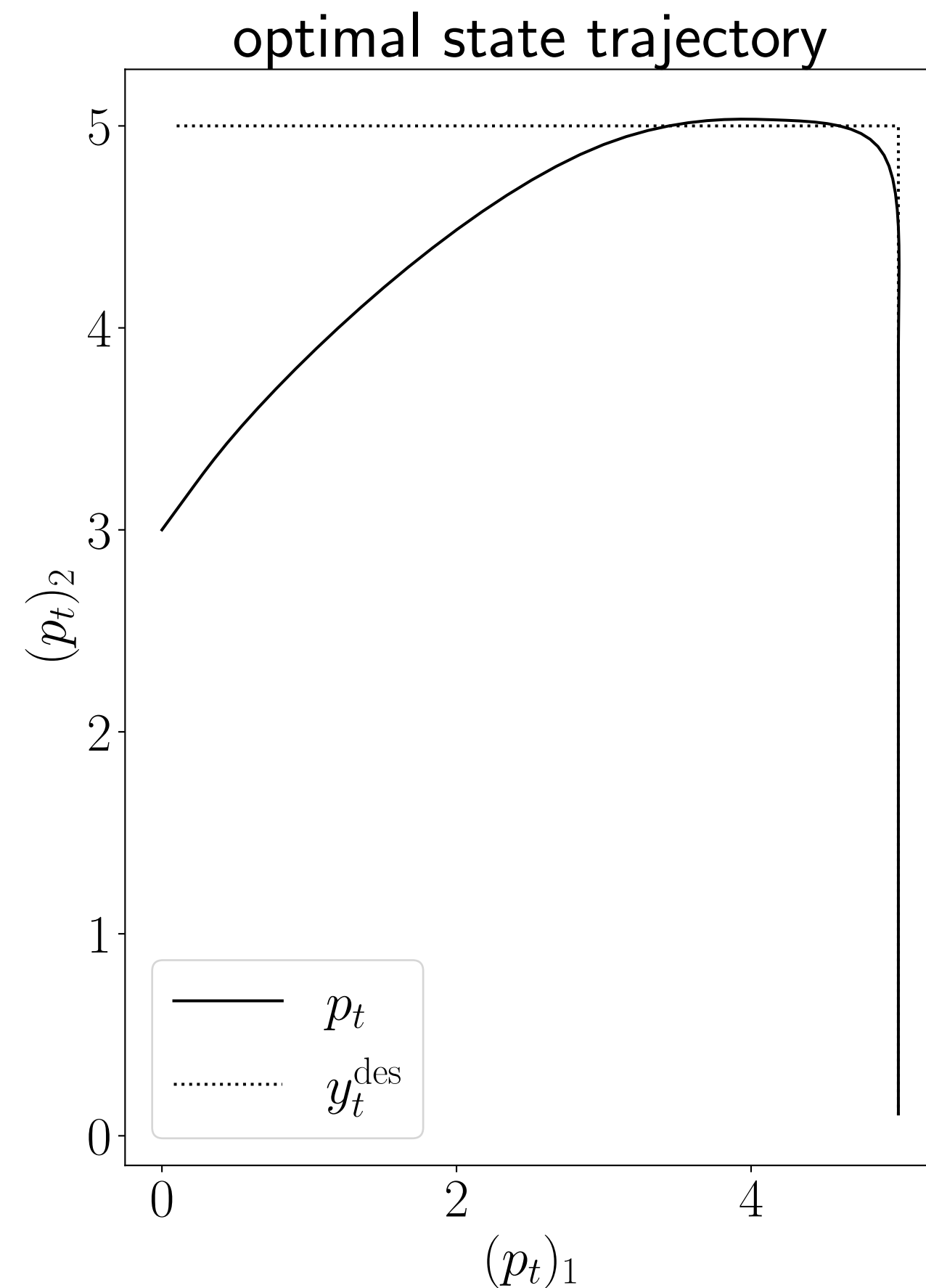


# Vehicle example with output tracking

## 1-norm with constraints results

### Parameters

$$u^{\max} = 10, \quad s^{\max} = 0.1$$

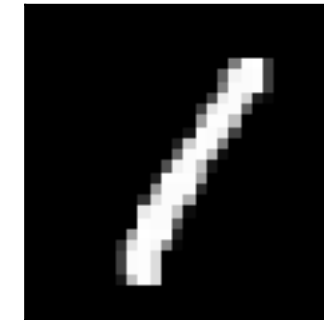


# Character recognition

# Character recognition

## MNIST data set of handwritten numerals

- Each character is 28 x 28 pixels
- 60k example images
- 10k further testing images
- Each sample comes with a label 0 – 9



## Goal

Use linear classification to identify handwritten numbers

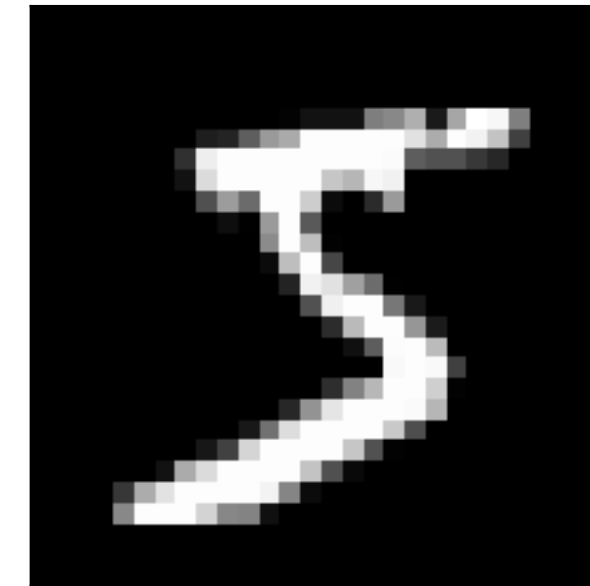
# Images representation

## Monochrome images

Images represented as an  $m \times n$  matrix  $X$

Each value  $X_{ij}$  represents a pixel's intensity (0 = black, and 255 = white)

$X =$



(in MNIST,  $m = n = 28$ )

# Images representation

## Monochrome images

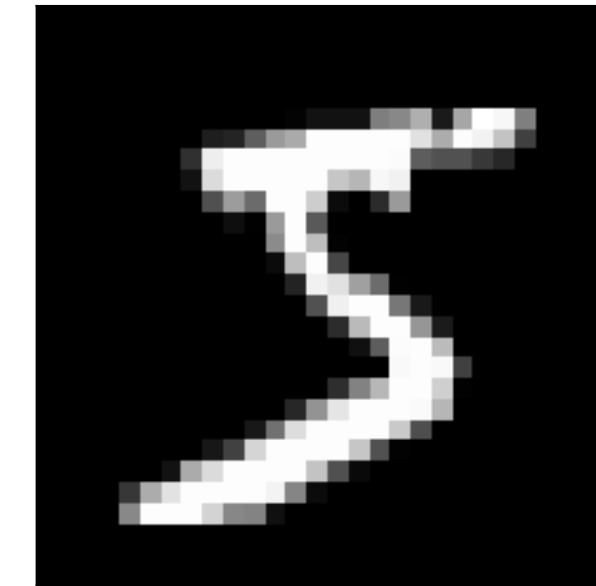
Images represented as an  $m \times n$  matrix  $X$

Each value  $X_{ij}$  represents a pixel's intensity (0 = black, and 255 = white)

We can represent an  $m \times n$  matrix  $X$  by a single vector  $x \in \mathbf{R}^{mn}$

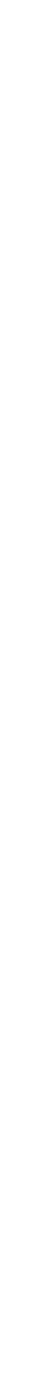
$$X_{ij} = x_k, \quad k = m(j - 1) + i$$

$X =$



(in MNIST,  $m = n = 28$ )

$x =$

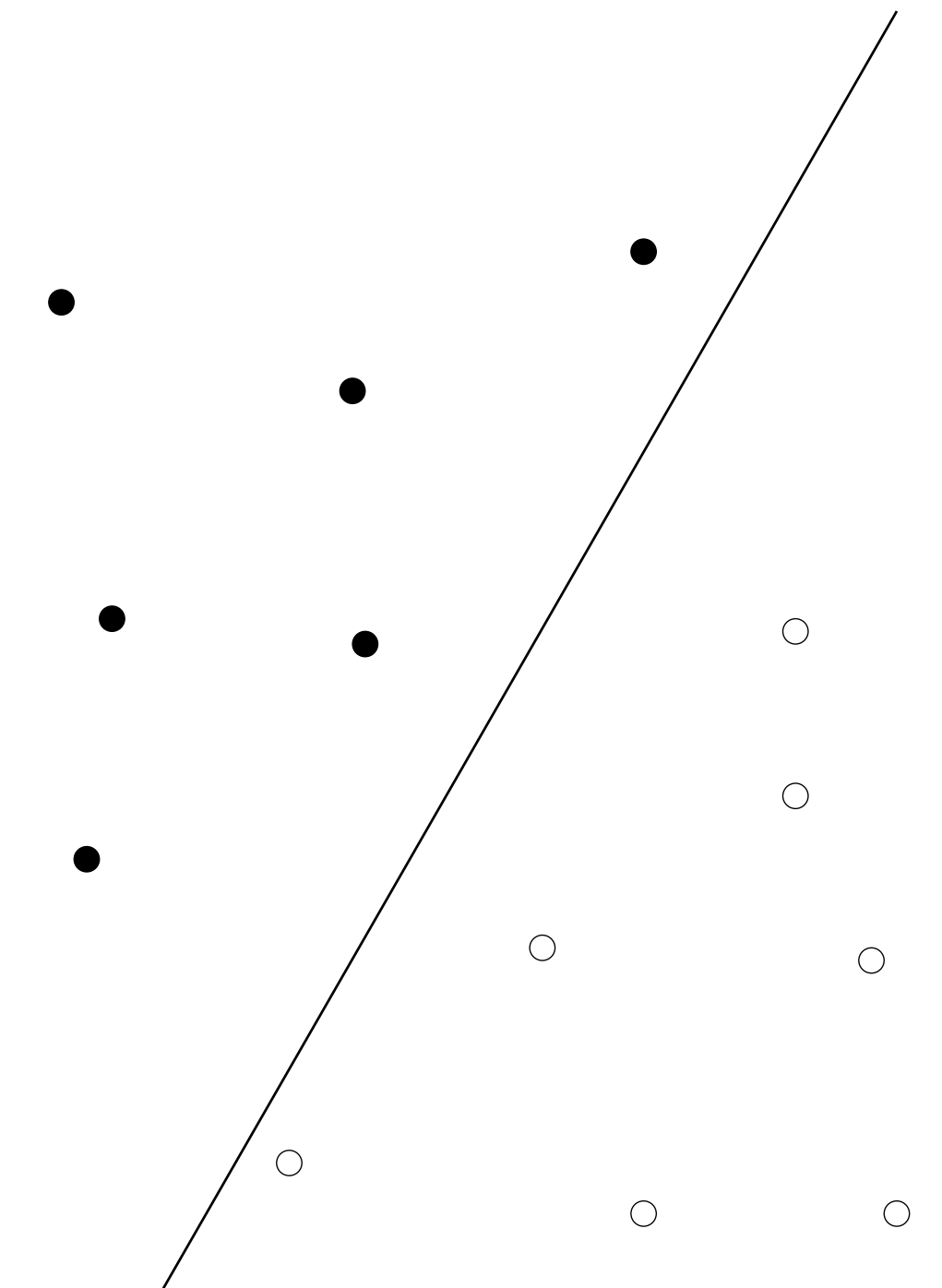


# Linear classification

## Support vector machine (linear separation)

Given a set of points  $\{v_1, \dots, v_N\}$  with binary labels  $s_i \in \{-1, 1\}$   
Find hyperplane that strictly separates the two classes

$$\begin{array}{l} a^T v_i + b > 0 \quad \text{if} \quad s_i = 1 \\ a^T v_i + b < 0 \quad \text{if} \quad s_i = -1 \end{array} \longrightarrow s_i(a^T v_i + b) \geq 1$$





# Linear classification

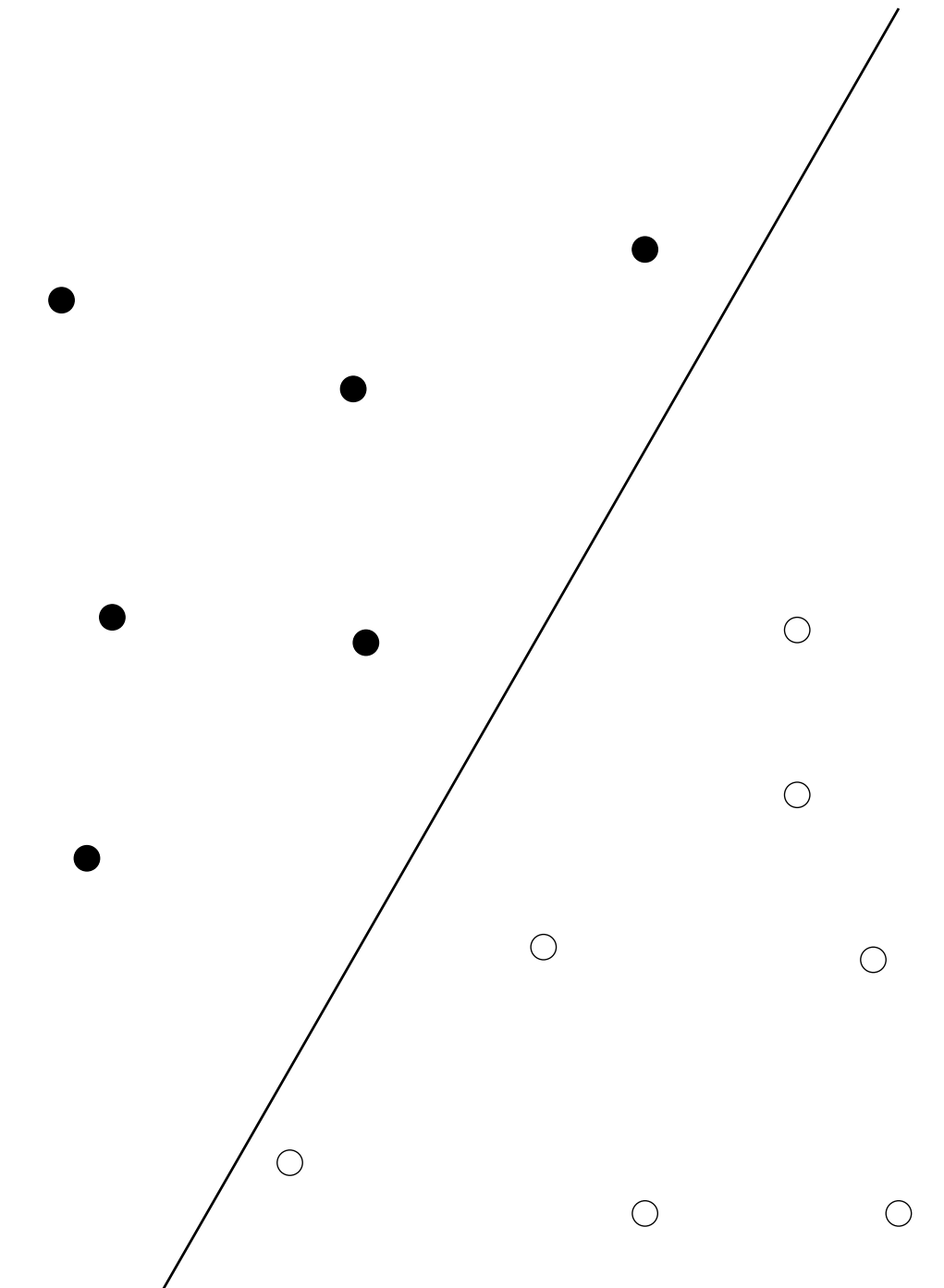
## Support vector machine (linear separation)

Given a set of points  $\{v_1, \dots, v_N\}$  with binary labels  $s_i \in \{-1, 1\}$   
Find hyperplane that strictly separates the two classes

$$\begin{array}{l} a^T v_i + b > 0 \quad \text{if } s_i = 1 \\ a^T v_i + b < 0 \quad \text{if } s_i = -1 \end{array} \longrightarrow s_i(a^T v_i + b) \geq 1$$

## Minimize sum of the violations + regularization

$$\text{minimize} \quad \sum_{i=1}^N \max\{0, 1 - s_i(a^T v_i + b)\} + \lambda \|a\|_1$$



# Linear classification

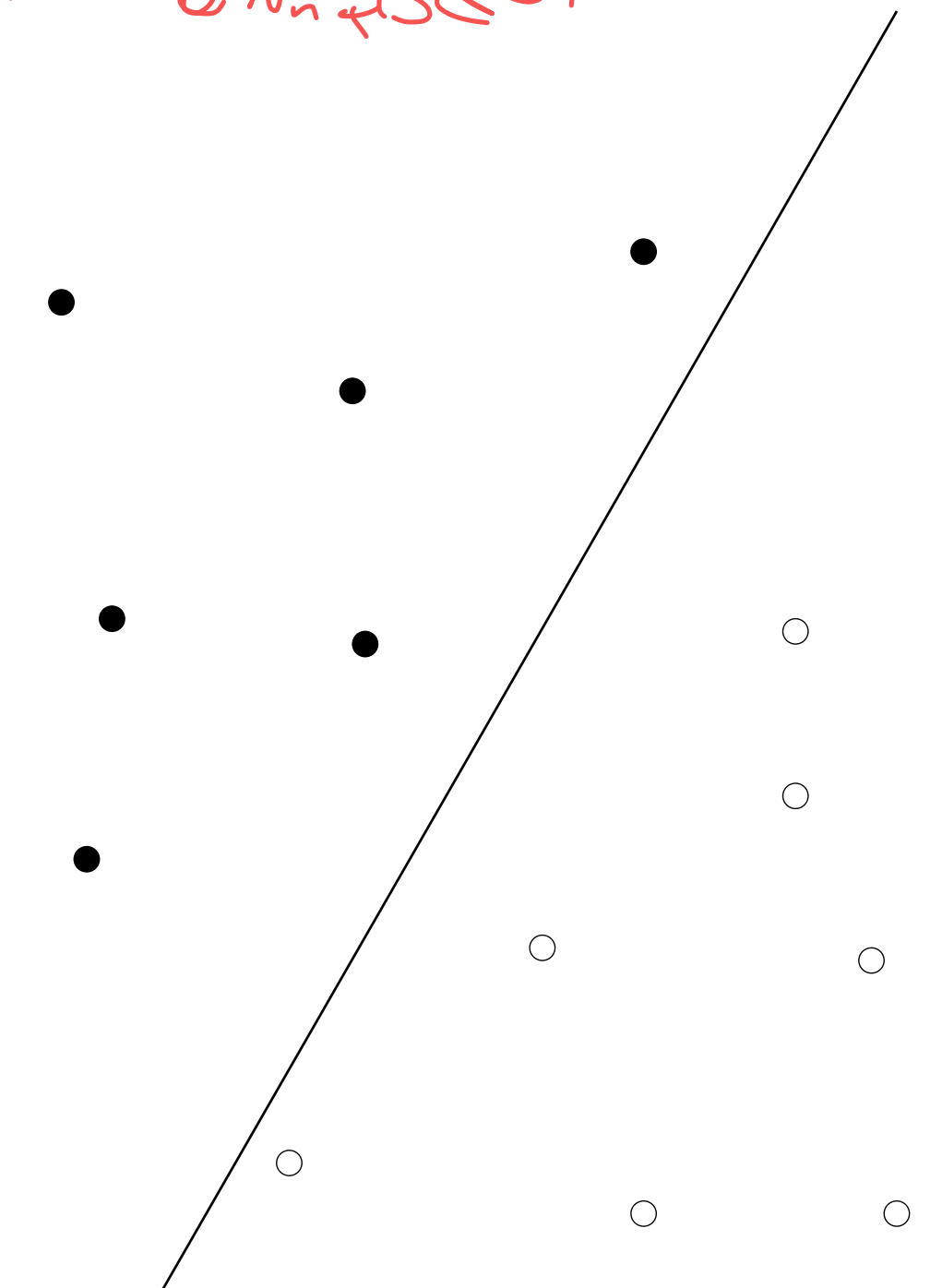
## Support vector machine (linear separation)

Given a set of points  $\{v_1, \dots, v_N\}$  with binary labels  $s_i \in \{-1, 1\}$   
Find hyperplane that strictly separates the two classes

$$\begin{aligned} a^T v_i + b > 0 & \text{ if } s_i = 1 \\ a^T v_i + b < 0 & \text{ if } s_i = -1 \end{aligned} \longrightarrow s_i(a^T v_i + b) \geq 1$$

$$s_i = 1 \Rightarrow a^T v_i + b \geq 1$$

$$s_i = -1 \Rightarrow -(a^T v_i + b) \geq 1 \quad a^T v_i + b \leq -1$$

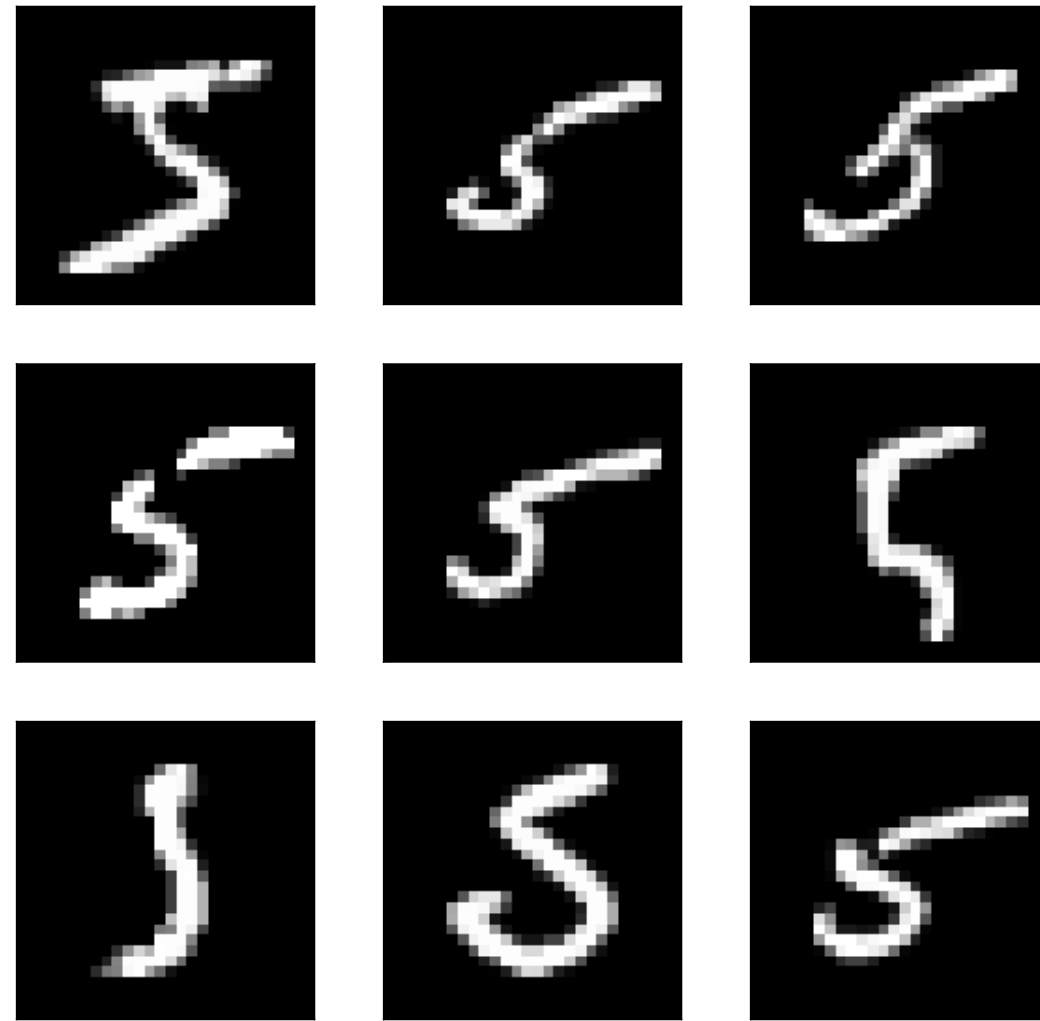


## Minimize sum of the violations + regularization

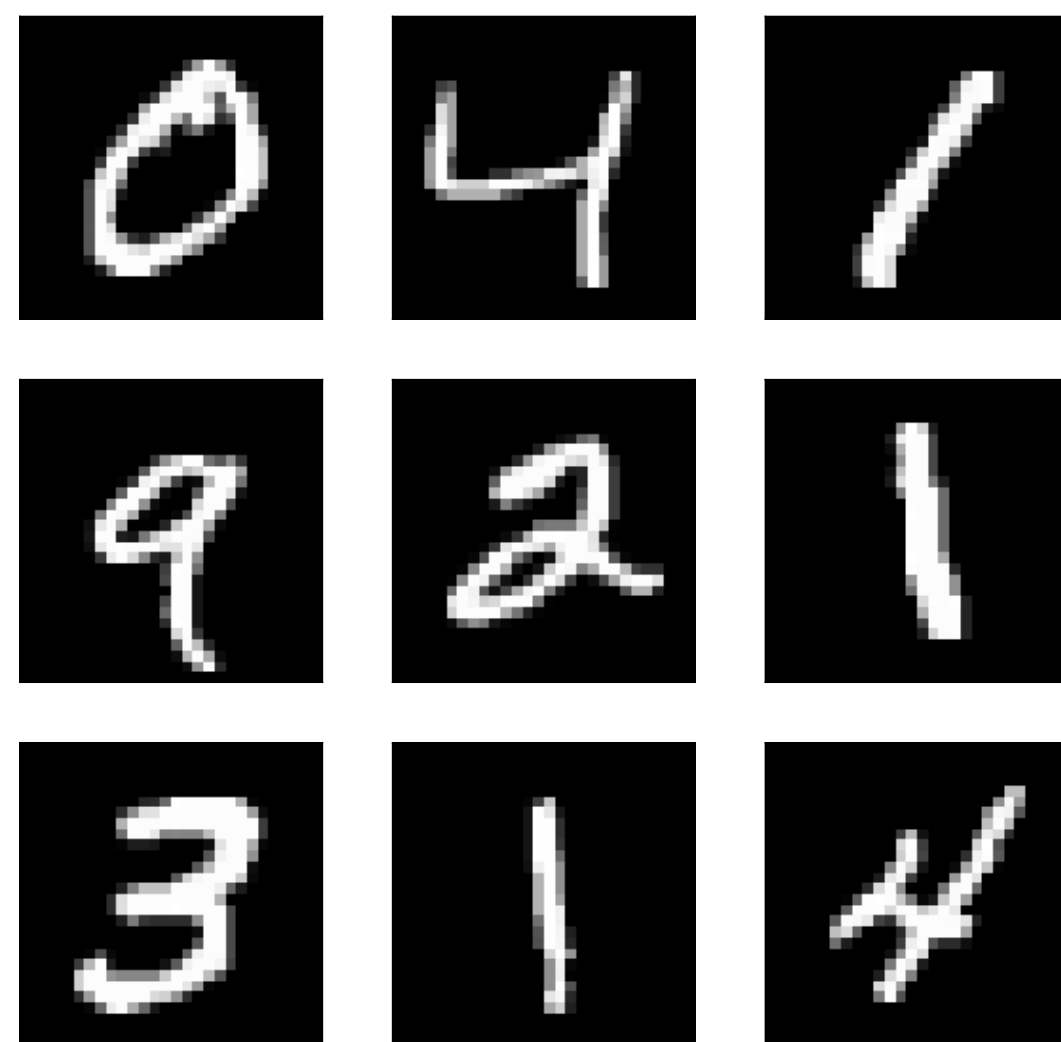
minimize  $\sum_{i=1}^N \max\{0, 1 - s_i(a^T v_i + b)\} + \lambda \|a\|_1$  ← regularization

# Learn to classify 5

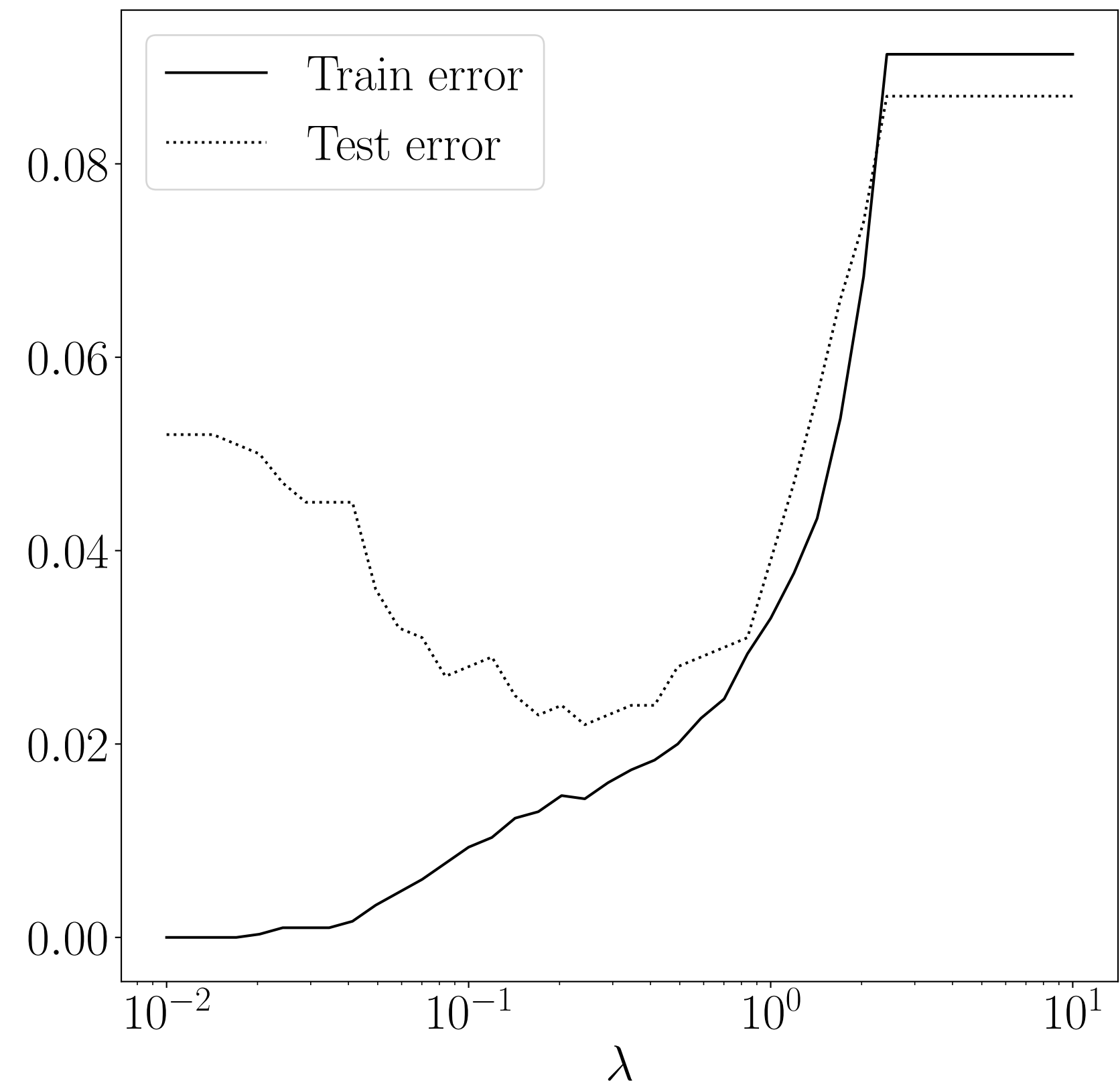
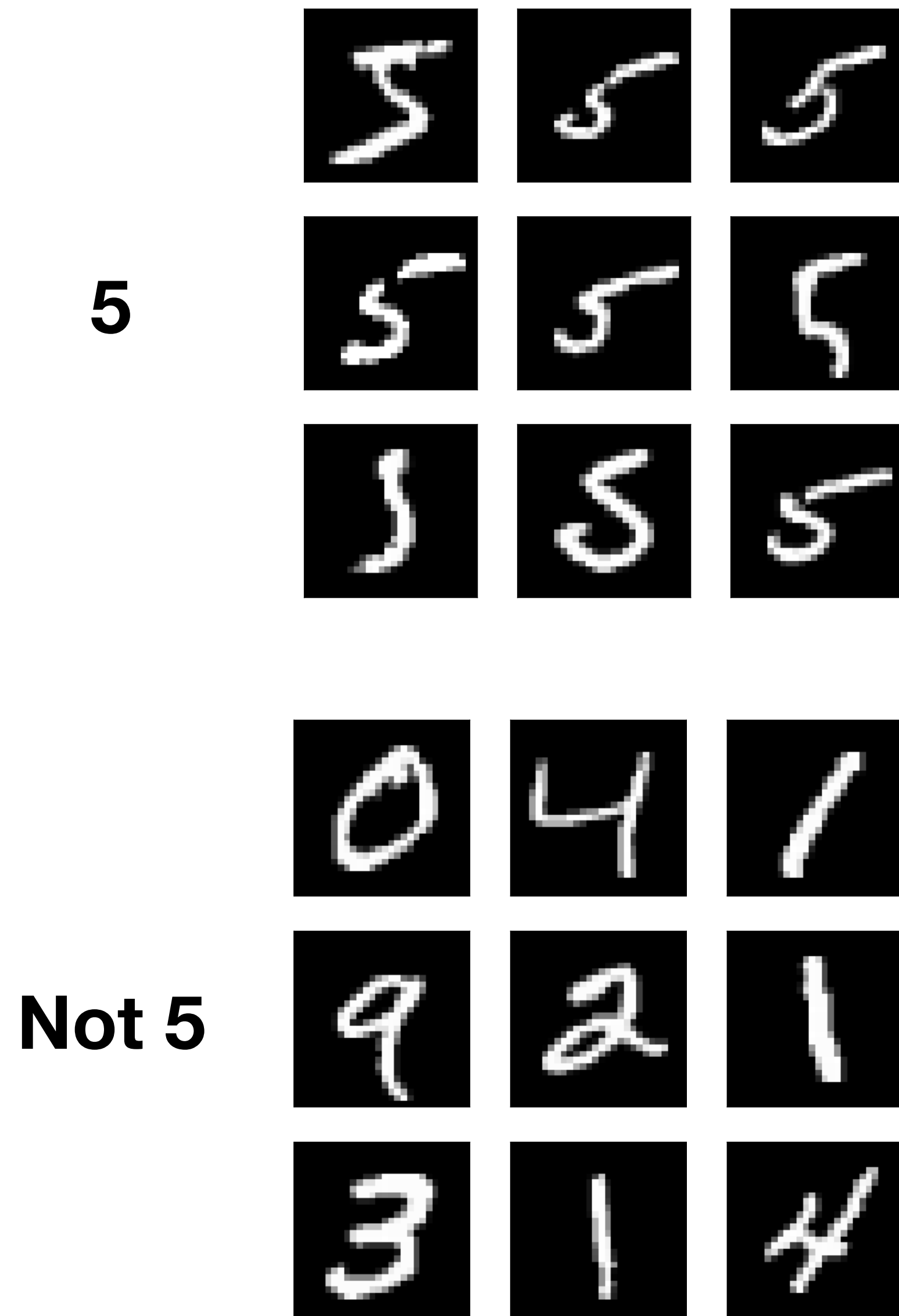
5



Not 5



# Learn to classify 5



# Multiclass classification

1. Train one classifier per label  $k$   
(e.g.,  $k$  vs anything else), obtaining  $(a_k, b_k)$
2. Predict all results and take the maximum

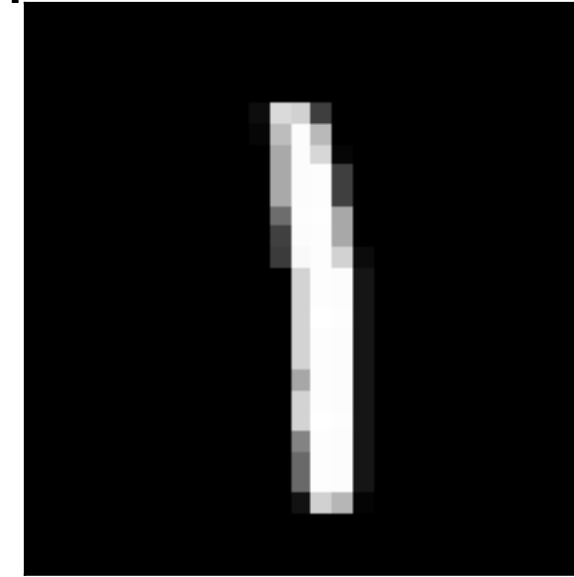
$$\hat{y}^{(i)} = \operatorname{argmax}_k a_k^T v^{(i)} + b_k$$

# Multiclass classification

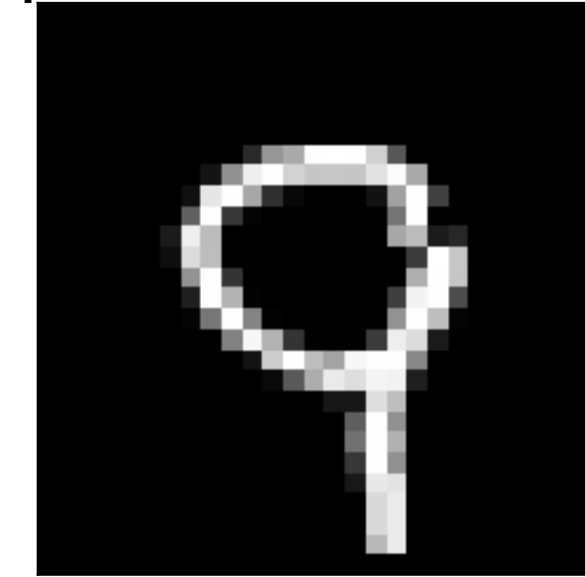
1. Train one classifier per label  $k$  (e.g.,  $k$  vs anything else), obtaining  $(a_k, b_k)$
2. Predict all results and take the maximum

$$\hat{y}^{(i)} = \operatorname{argmax}_k a_k^T v^{(i)} + b_k$$

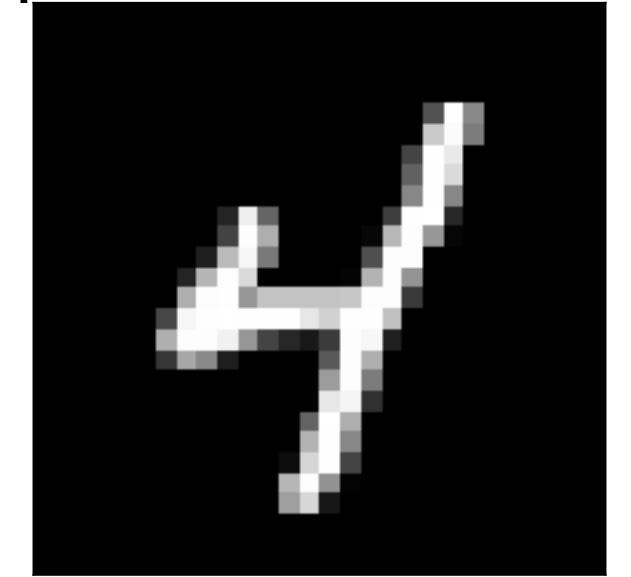
predicted label: 1



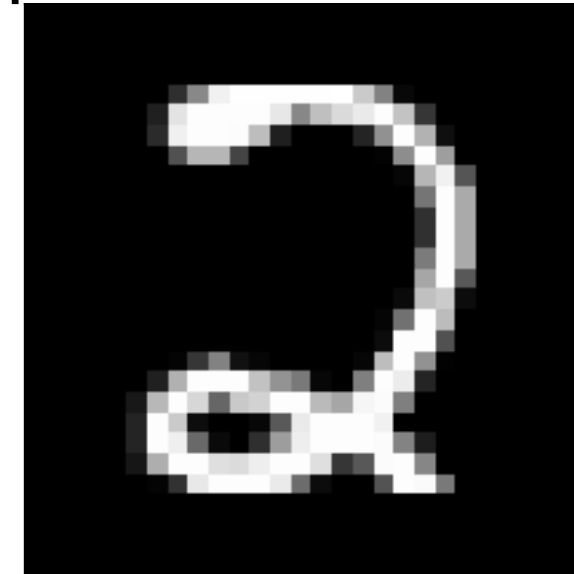
predicted label: 9



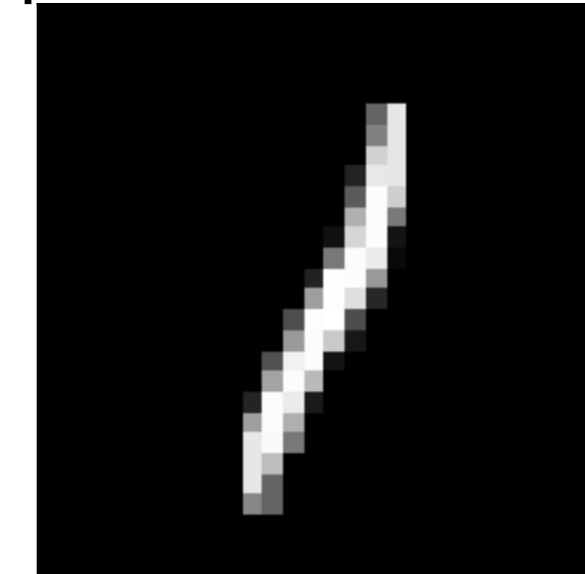
predicted label: 4



predicted label: 2



predicted label: 1



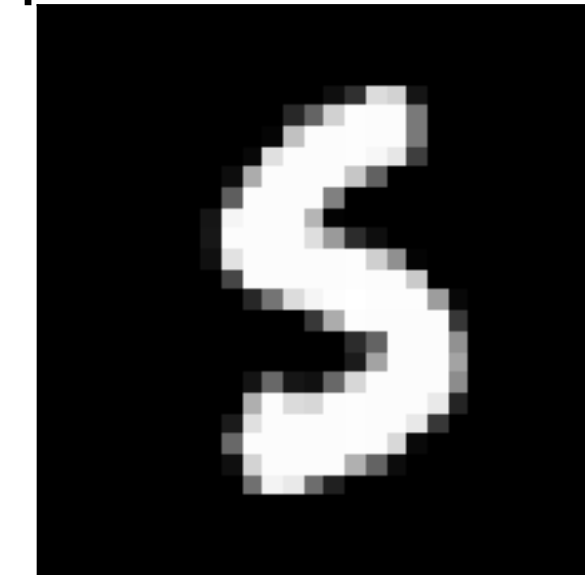
predicted label: 2



predicted label: 2



predicted label: 5



predicted label: 3



# Portfolio optimization

# Portfolio allocation weights

We want to invest  $V$  dollars in  $n$  different *assets* (stocks, bonds, ...)  
over periods  $t = 1, \dots, T$

## Portfolio allocation weights

$n$ -vector  $w$  gives the fraction of our total portfolio held in each asset



# Portfolio allocation weights

We want to invest  $V$  dollars in  $n$  different *assets* (stocks, bonds, ...) over periods  $t = 1, \dots, T$

## Portfolio allocation weights

$n$ -vector  $w$  gives the fraction of our total portfolio held in each asset

### Properties

- $V w_j$  dollar value hold in asset  $j$
- $\mathbf{1}^T w = 1$  (normalized)
- $w_j < 0$  means short positions (you borrow)  
(must be returned at time  $T$ )
- Example:  $w = (-0.2, 0.0, 1.2)$

# Portfolio allocation weights

We want to invest  $V$  dollars in  $n$  different *assets* (stocks, bonds, ...)  
over periods  $t = 1, \dots, T$

## Portfolio allocation weights

$n$ -vector  $w$  gives the fraction of our total portfolio held in each asset

### Properties

- $Vw_j$  dollar value hold in asset  $j$
- $\mathbf{1}^T w = 1$  (normalized)
- $w_j < 0$  means short positions (you borrow)  
(must be returned at time  $T$ )
- Example:  $w = (-0.2, 0.0, 1.2)$

Short position  
of  $0.2V$  on asset 1



# Portfolio allocation weights

We want to invest  $V$  dollars in  $n$  different *assets* (stocks, bonds, ...)  
over periods  $t = 1, \dots, T$

## Portfolio allocation weights

$n$ -vector  $w$  gives the fraction of our total portfolio held in each asset

### Properties

- $Vw_j$  dollar value hold in asset  $j$
- $\mathbf{1}^T w = 1$  (normalized)
- $w_j < 0$  means short positions (you borrow)  
(must be returned at time  $T$ )
- Example:  $w = (-0.2, 0.0, 1.2)$

Short position  
of  $0.2V$  on asset 1

Don't hold any  
of asset 2

# Portfolio allocation weights

We want to invest  $V$  dollars in  $n$  different *assets* (stocks, bonds, ...)  
over periods  $t = 1, \dots, T$

## Portfolio allocation weights

$n$ -vector  $w$  gives the fraction of our total portfolio held in each asset

### Properties

- $Vw_j$  dollar value hold in asset  $j$
- $\mathbf{1}^T w = 1$  (normalized)
- $w_j < 0$  means short positions (you borrow)  
(must be returned at time  $T$ )
- Example:  $w = (-0.2, 0.0, 1.2)$

Short position  
of  $0.2V$  on asset 1

Don't hold any  
of asset 2

Hold  $1.2V$   
in asset 3

# Return over a period

## Asset returns

$\tilde{r}_t$  is the (fractional) return of each asset over period  $t$

example:  $\tilde{r}_t = (0.01, -0.023, 0.02)$   
(often expressed as percentage)

# Return over a period

## Asset returns

$\tilde{r}_t$  is the (fractional) return of each asset over period  $t$

example:  $\tilde{r}_t = (0.01, -0.023, 0.02)$   
(often expressed as percentage)

## Portfolio return

$$r_t = \tilde{r}_t^T w$$

It is the (fractional) return for the entire portfolio over period  $t$

# Return over a period

$$R = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \end{bmatrix}$$

## Asset returns

$\tilde{r}_t$  is the (fractional) return of each asset over period  $t$

example:  $\tilde{r}_t = (0.01, -0.023, 0.02)$   
(often expressed as percentage)

## Portfolio return

$$r_t = \tilde{r}_t^T w$$

It is the (fractional) return for the entire portfolio over period  $t$

## Total portfolio value after a period

$$V_{t+1} = V_t + V_t \tilde{r}_t^T w = V_t (1 + r_t)$$

$$r = R w$$

# Portfolio optimization

How shall we choose the portfolio weight vector  $w$ ?



# Portfolio optimization

How shall we choose the portfolio weight vector  $w$ ?

## Goals

**High (average) return**

**Low risk**

# Portfolio optimization

How shall we choose the portfolio weight vector  $w$ ?

## Goals

**High (average) return**

**Low risk**

## 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)

# Linear optimization for portfolio objective

## Average return

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

$\mu$  is the  $n$ -vector of average returns per asset

$$\text{std}(r) = \frac{1}{\sqrt{T}} \sum_{i=1}^T (r_i - \text{avg}(r))^2 = \frac{1}{\sqrt{T}} \|r - \text{avg}(r)\mathbf{1}\|$$

# Linear optimization for portfolio objective

## Average return

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

$\mu$  is the  $n$ -vector of average returns per asset

## 1-norm risk approximation

$$\|r - \text{avg}(r)\mathbf{1}\|_1 / T$$

- No longer  $\text{std}(r)$  (divide by  $T$  instead of  $\sqrt{T}$ )
- Linear optimization representable
- Induces sparser fluctuations  $|r_i - \text{avg}(r)|$

# Linear optimization for portfolio objective

## Average return

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

$\mu$  is the  $n$ -vector of average returns per asset

## 1-norm risk approximation

$$\|r - \text{avg}(r)\mathbf{1}\|_1/T$$

- No longer  $\text{std}(r)$  (divide by  $T$  instead of  $\sqrt{T}$ )
- Linear optimization representable
- Induces sparser fluctuations  $|r_i - \text{avg}(r)|$

## Risk-return objective

$$-\mu^T w + \lambda\|Rw - (\mu^T w)\mathbf{1}\|_1/T$$

$$Rw = r$$

# Linear optimization for portfolio objective

## Average return

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

$\mu$  is the  $n$ -vector of average returns per asset

## 1-norm risk approximation

$$\|r - \text{avg}(r)\mathbf{1}\|_1/T$$

- No longer  $\text{std}(r)$  (divide by  $T$  instead of  $\sqrt{T}$ )
- Linear optimization representable
- Induces sparser fluctuations  $|r_i - \text{avg}(r)|$

## Risk-return objective

$$-\mu^T w + \lambda \|Rw - (\mu^T w)\mathbf{1}\|_1/T$$



(tradeoff parameter)

# Portfolio optimization

## Minimize risk-return tradeoff

Choose  $n$ -vector  $w$  to solve

$$\text{minimize} \quad -\mu^T w + \lambda \|Rw - (\mu^T w)\mathbf{1}\|_1 / T$$

$$\text{subject to} \quad \mathbf{1}^T w = 1$$

$$w \geq 0$$

$$\|w\|_1 \leq L$$

$$w_1 + w_2 \geq 0.3$$

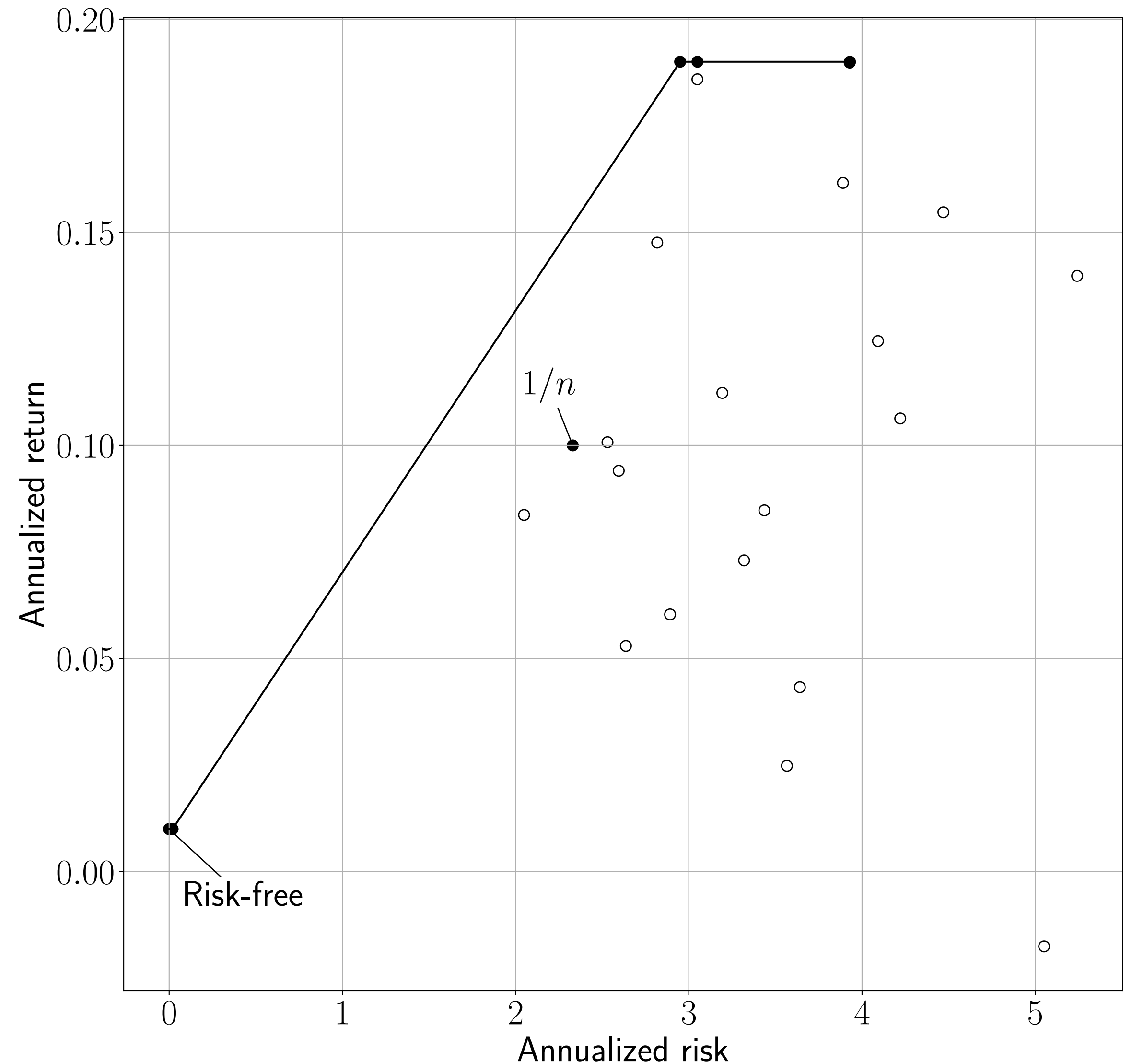
## Remarks

- Can have inequality constraints (e.g., long-only)
- Tune  $\lambda$  to get desired Pareto-optimal point
- Gives the best allocation  $w^*$  given the past returns

# Example

20 assets over 2000 days (past)

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





# The big assumption

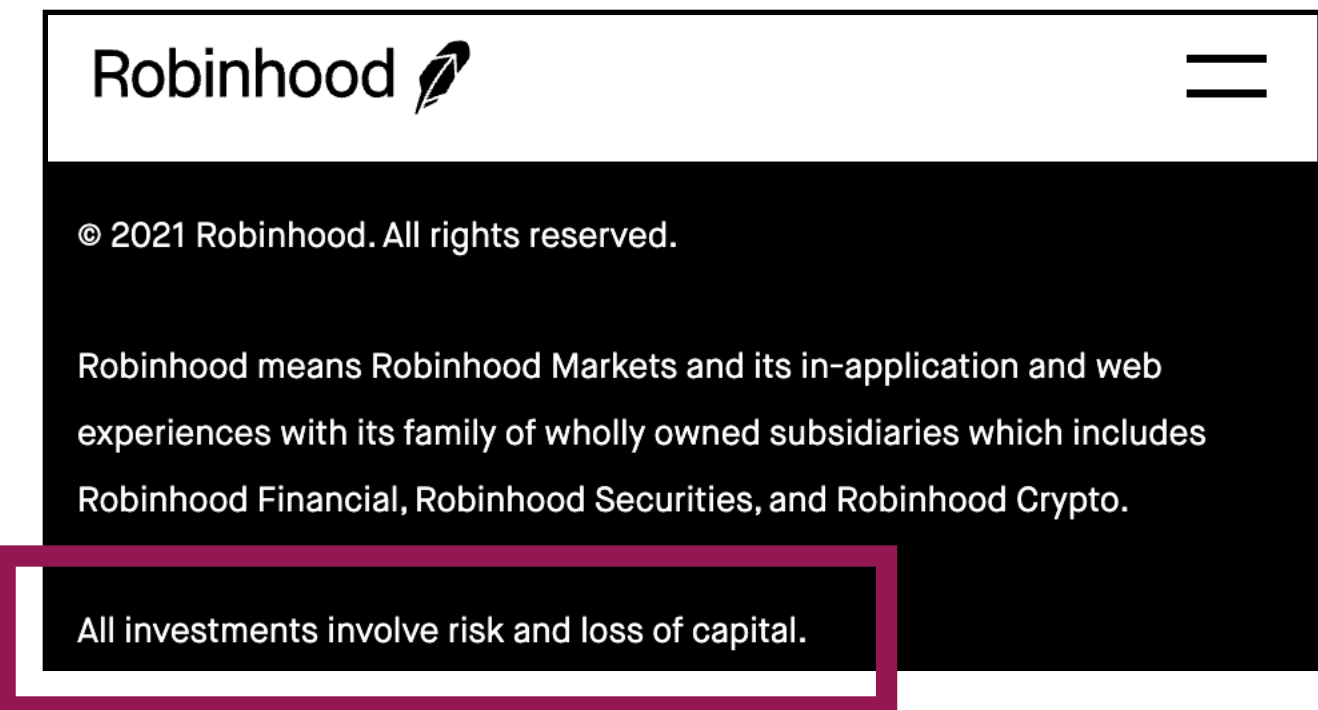
## **Future returns will look like past ones**

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

# The big assumption

## Future returns will look like past ones

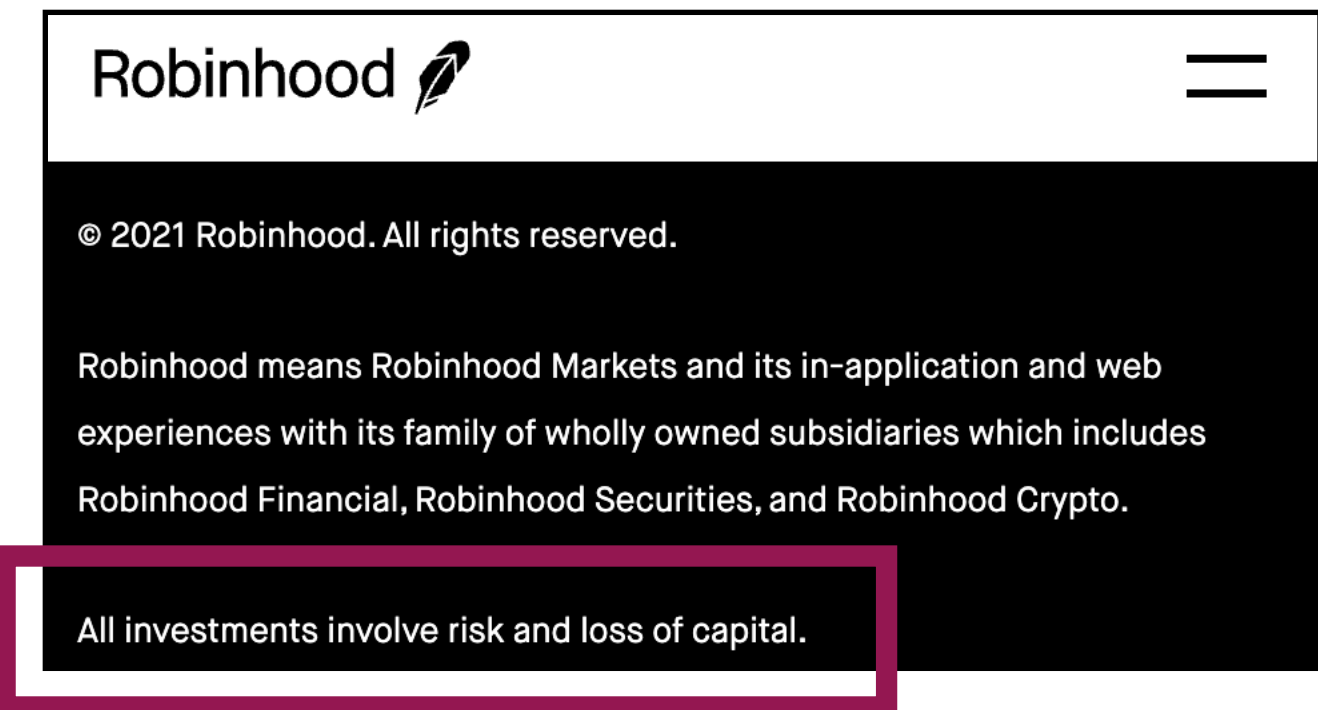
- You are warned this is false, every time you invest
- It is often reasonable
- During crisis, market shifts, other big events not true



# 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

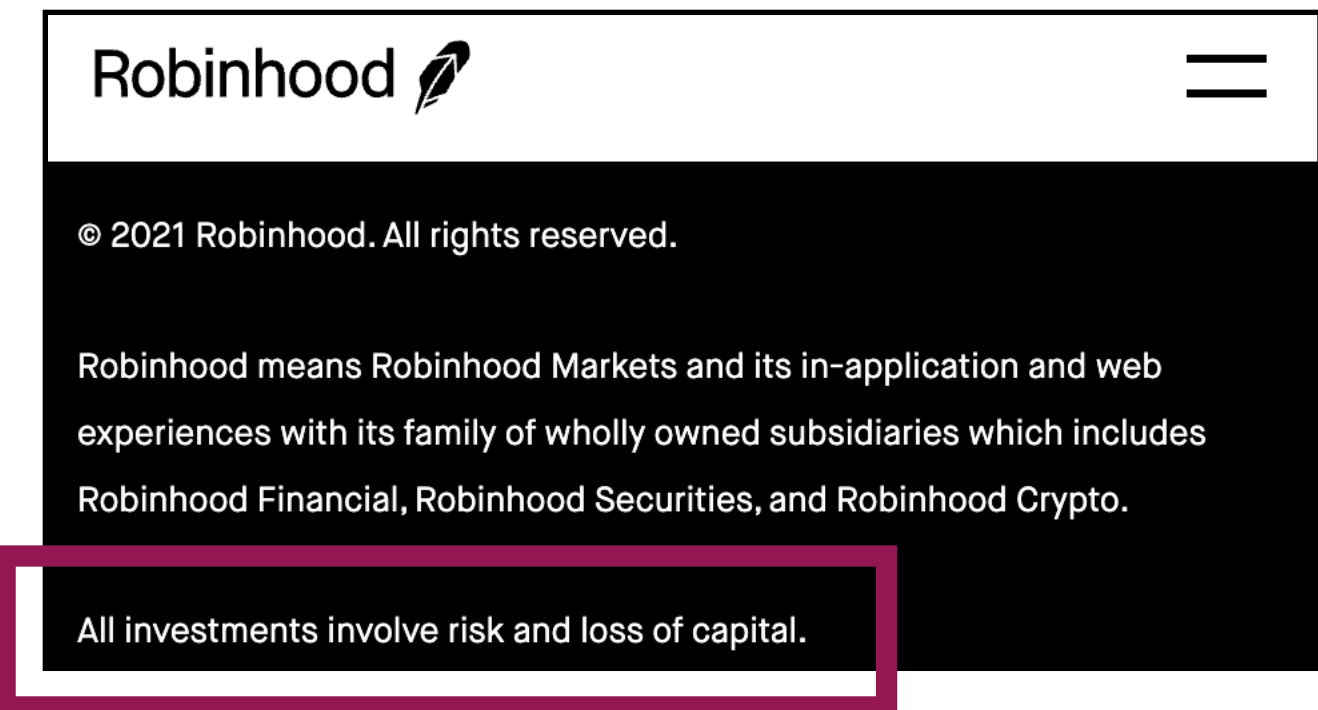


If assumption holds (even approximately), a good  $w$  on past returns leads to good future (unknown) returns

# 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



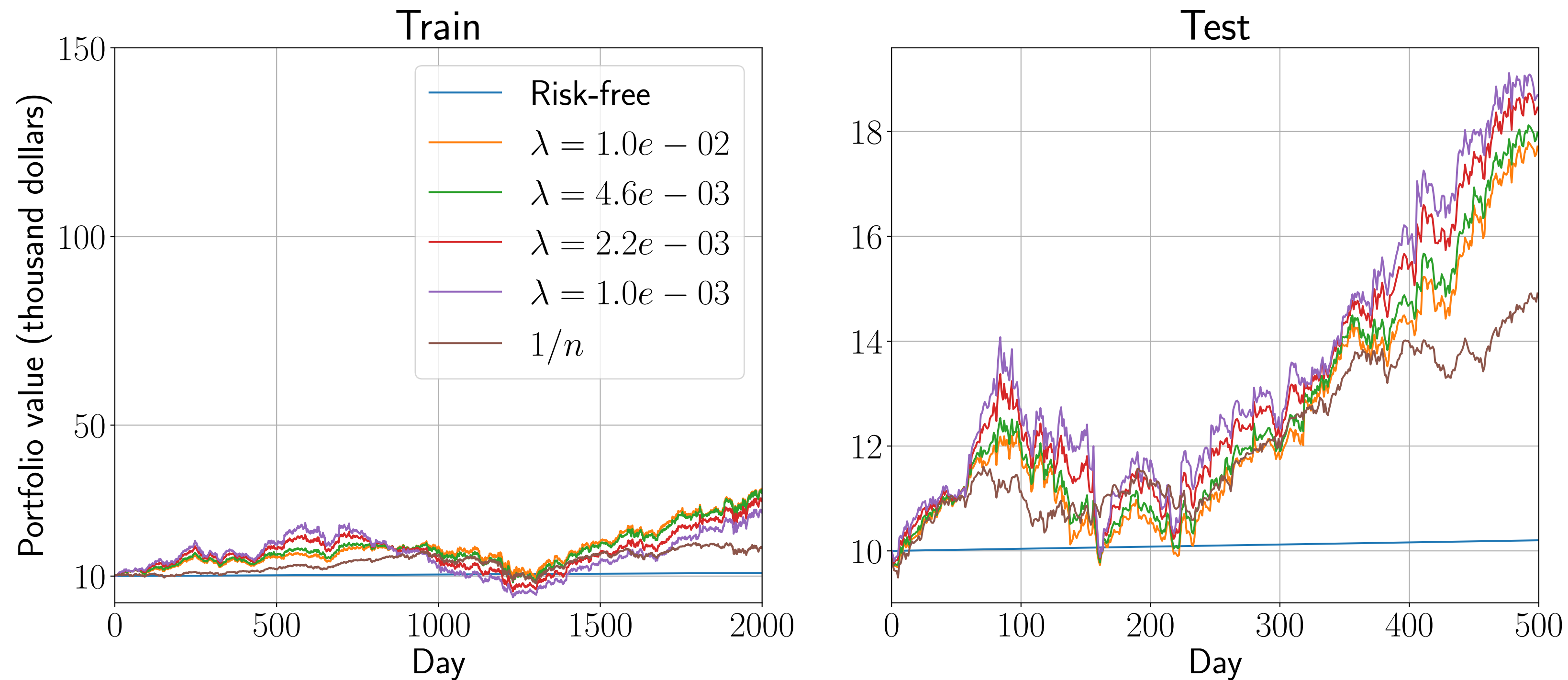
If assumption holds (even approximately), a good  $w$  on past returns leads to good future (unknown) returns

## Example

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

# Total portfolio value

	Train return	Test return	Train risk	Test risk
Risk-free	0.01	0.01	0.00	0.00
$\lambda = 1.0e - 02$	0.19	0.30	2.97	2.18
$\lambda = 4.6e - 03$	0.19	0.31	3.05	2.21
$\lambda = 2.2e - 03$	0.19	0.33	3.45	2.42
$\lambda = 1.0e - 03$	0.19	0.34	3.93	2.73
$1/n$	0.10	0.21	2.33	1.51



# Build your quantitative hedge fund

## Rolling portfolio optimization

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

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

# Build your quantitative hedge fund

## Rolling portfolio optimization

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

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

## Variations

- Update  $w$  every  $K$  periods (monthly, quarterly, ...)
- Add secondary objective  $\lambda \|w_t - w_{t-1}\|_1$  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)

# Applications of linear optimization

Today, we learned to apply linear optimization in

- **Optimal control** problems with vehicle dynamics
- **Machine learning** problems for character recognition
- **Portfolio optimization** for investment strategies



# References

- Github companion notebooks

# Next steps

- Simplex method