

# **ORF307 – Optimization**

## **5. Multi-objective least squares**

# Ed Forum

- What are *feature mappings*?
- Notation for AR time series. I understand that  $M$  is the number of data points in the buffer set, but I wasn't sure on what capital  $T$  represents?

**Recap**

# Least squares data fitting

## Vector form

Express problems with  $N$ -vectors

- $y^d = (y^{(1)}, \dots, y^{(N)})$ , vector of outcomes
  - $\hat{y}^d = (\hat{y}^{(1)}, \dots, \hat{y}^{(N)})$ , vector of predictions
  - $r^d = (r^{(1)}, \dots, r^{(N)})$ , vector of residuals
- Goal**  
minimize  $\|r^d\|^2$

We can write  $\hat{y}^{(i)} = \hat{f}(x^{(i)})$  in terms of parameters  $\theta_i$

$$\hat{y}^{(i)} = A_{i1}\theta_1 + \dots + A_{ip}\theta_p, \quad A_{ij} = f_j(x^{(i)}) \quad \longrightarrow \quad \hat{y}^d = A\theta$$

## Least squares problem

$$\text{minimize } \|r^d\|^2 = \|y^d - \hat{y}^d\|^2 = \|y^d - A\theta\|^2 = \|A\theta - y^d\|^2$$

## Solution

$$(A^T A)\theta^* = A^T y^d$$

# Auto-regressive time series model

$z_1, z_2, \dots$  is a time series

**auto-regressive (AR) prediction model**

$$\hat{z}_{t+1} = \theta_1 z_t + \dots + \theta_M z_{t-M+1}, \quad t = M, M+1, \dots$$

(predict  $\hat{z}_{t+1}$  based on previous  $M$  values, where  $M$  is the memory)

**Goal:** Chose  $\theta$  to minimize sum of squares of prediction errors

$$(\hat{z}_{M+1} - z_{M+1})^2 + \dots + (\hat{z}_T - z_T)^2$$

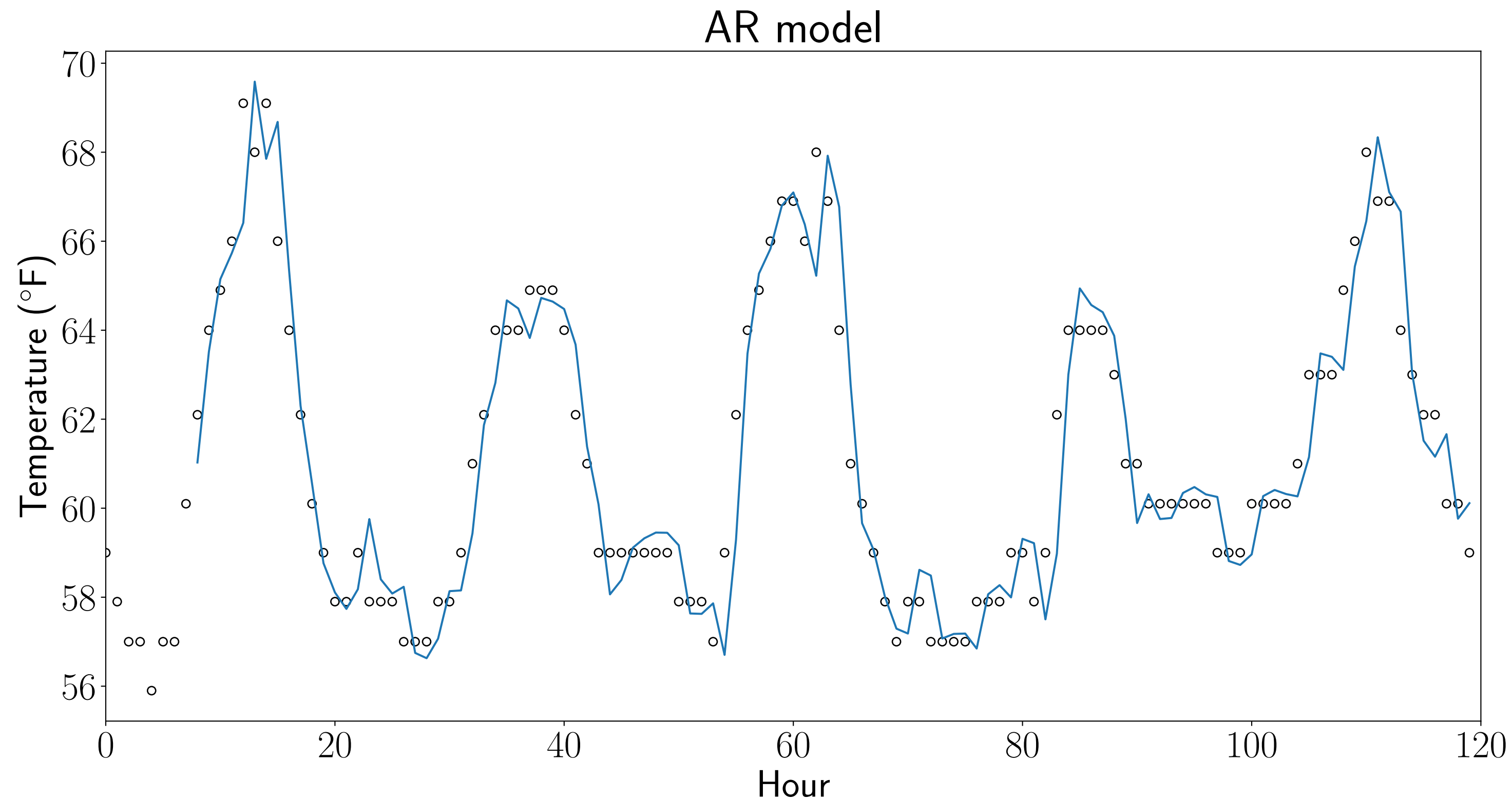
**General data fitting form**

$$y^{(i)} = z_{M+i}, \quad x^{(i)} = (z_{M+i-1}, \dots, z_i), \quad i = 1, \dots, T - M$$

# Auto-regressive time series model

**5 days hourly temperature at  
Los Angeles International  
Airport (LAX)**

- Previous hour:  $\hat{z}_{t+1} = z_t$ , MSE 1.35
- 24 hours before:  $\hat{z}_{t+1} = z_{t-23}$ , MSE = 3.00
- AR model with  $M = 8$ , MSE = 1.02



# Today's lecture

## Multi-objective least squares

- Multi-objective least squares problem
- Control
- Estimation
- Regularized data fitting

# Multi-objective least squares problem



# Multi-objective least squares

**Goal** choose  $n$ -vector  $x$  such that  
 $k$  norm squared objectives are small

$$J_1 = \|A_1 x - b_1\|^2$$

$$\vdots$$

$$J_k = \|A_k x - b_k\|^2$$

$A_i$  are  $m_i \times n$  matrices and  $b_i$  are  $m_i$ -vectors for  $i = 1, \dots, k$

$J_i$  are the objectives in a *multi-objective (-criterion) optimization problem*

Could choose  $x$  to minimize  
any one  $J_i$ , but we want  
to make them all small

# Weighted sum objective

Choose positive weights  $\lambda_1, \dots, \lambda_k$  and form *weighted sum objective*

$$\begin{aligned} J &= \lambda_1 J_1 + \dots + \lambda_k J_k \\ &= \lambda_1 \|A_1 x - b_1\|^2 + \dots + \lambda_k \|A_k x - b_k\|^2 \end{aligned}$$

Choose  $x$  to minimize  $J$

## Primary objective

- Often  $\lambda_1 = 1$  and  $J_1$  is the **primary objective**
- **Interpretation**  $\lambda_i$  is how much we care about  $J_i$  being small, relative to  $J_1$

## Bi-criterion optimization

$$J_1 + \lambda J_2 = \|A_1 x - b_1\|^2 + \lambda \|A_2 x - b_2\|^2$$

# Weighted sum minimization as regular least squares

$$J = \lambda_1 \|A_1 x - b_1\|^2 + \dots + \lambda_k \|A_k x - b_k\|^2$$

$$= \left\| \begin{bmatrix} \sqrt{\lambda_1}(A_1 x - b_1) \\ \vdots \\ \sqrt{\lambda_k}(A_k x - b_k) \end{bmatrix} \right\|^2$$

**stack objectives**

## Regular (single-criterion) least squares

$$J = \|\tilde{A}x - \tilde{b}\|^2$$

$$\tilde{A} = \begin{bmatrix} \sqrt{\lambda_1}A_1 \\ \vdots \\ \sqrt{\lambda_k}A_k \end{bmatrix}, \quad \tilde{b} = \begin{bmatrix} \sqrt{\lambda_1}b_1 \\ \vdots \\ \sqrt{\lambda_k}b_k \end{bmatrix}$$

# Weighted sum solution

Assuming the columns of  $\tilde{A}$  are linearly independent

$$(\tilde{A}^T \tilde{A})x^* = \tilde{A}^T \tilde{b}$$

$$(\lambda_1 A_1^T A_1 + \cdots + \lambda_k A_k^T A_k)x^* = (\lambda_1 A_1^T b_1 + \cdots + \lambda_k A_k^T b_k)$$

## Remarks

- Can compute  $x^*$  via the Cholesky factorization of  $\tilde{A}^T \tilde{A}$
- $A_i$  can be wide or have dependent columns ( $\tilde{A}$  can't)

# Optimal trade-off curve

## Bi-criterion problem

$$\text{minimize } J_1(x) + \lambda J_2(x) \longrightarrow x^*(\lambda)$$

**Pareto optimal**  $x^*(\lambda)$

There is no point  $z$  that satisfies

$$J_1(z) \leq J_1(x^*(\lambda)) \quad \text{and} \quad J_2(z) \leq J_2(x^*(\lambda))$$

with one of the inequalities holding strictly  
(no other point beats  $x^*$  on both objectives)

**Optimal trade-off curve**

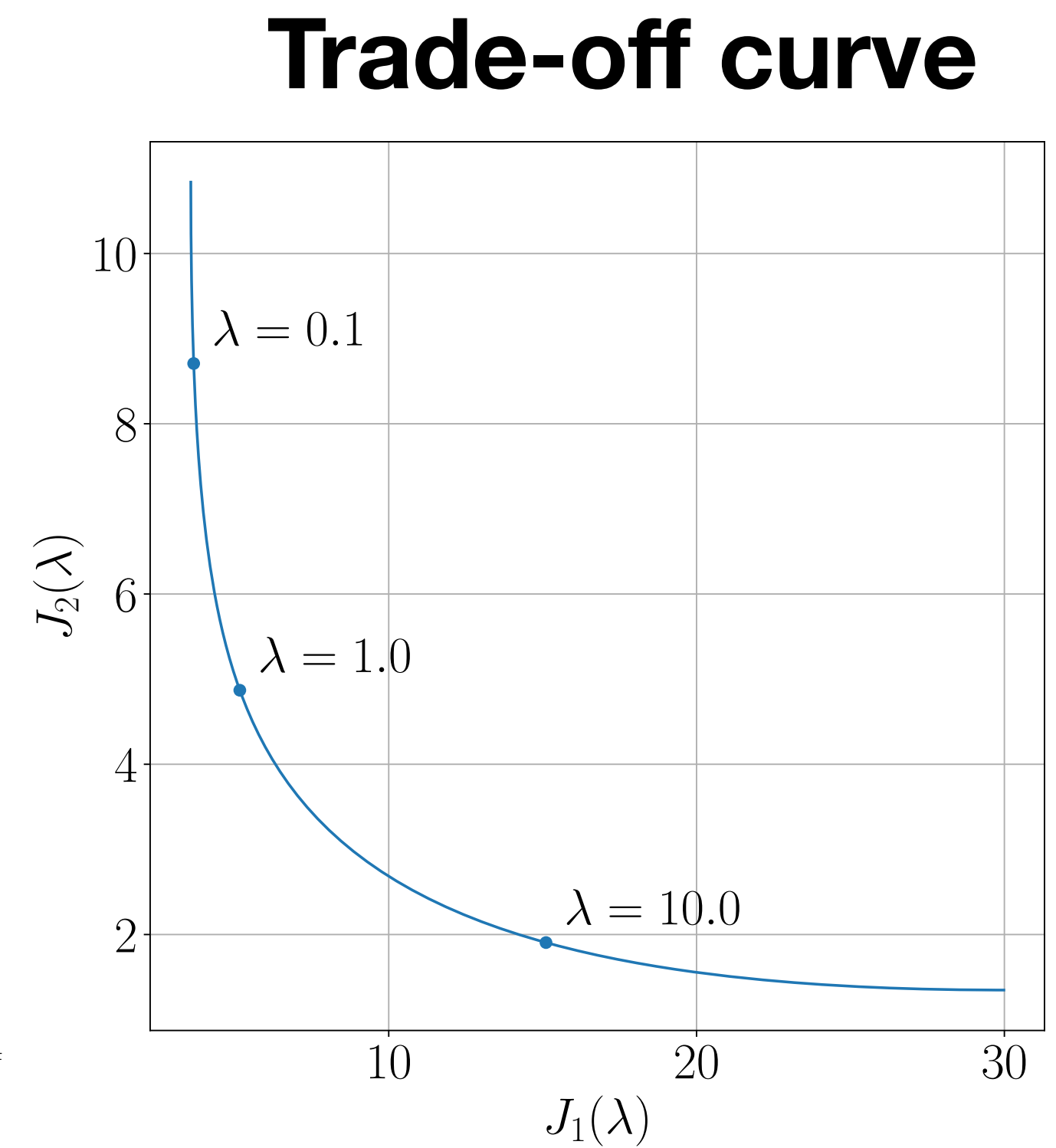
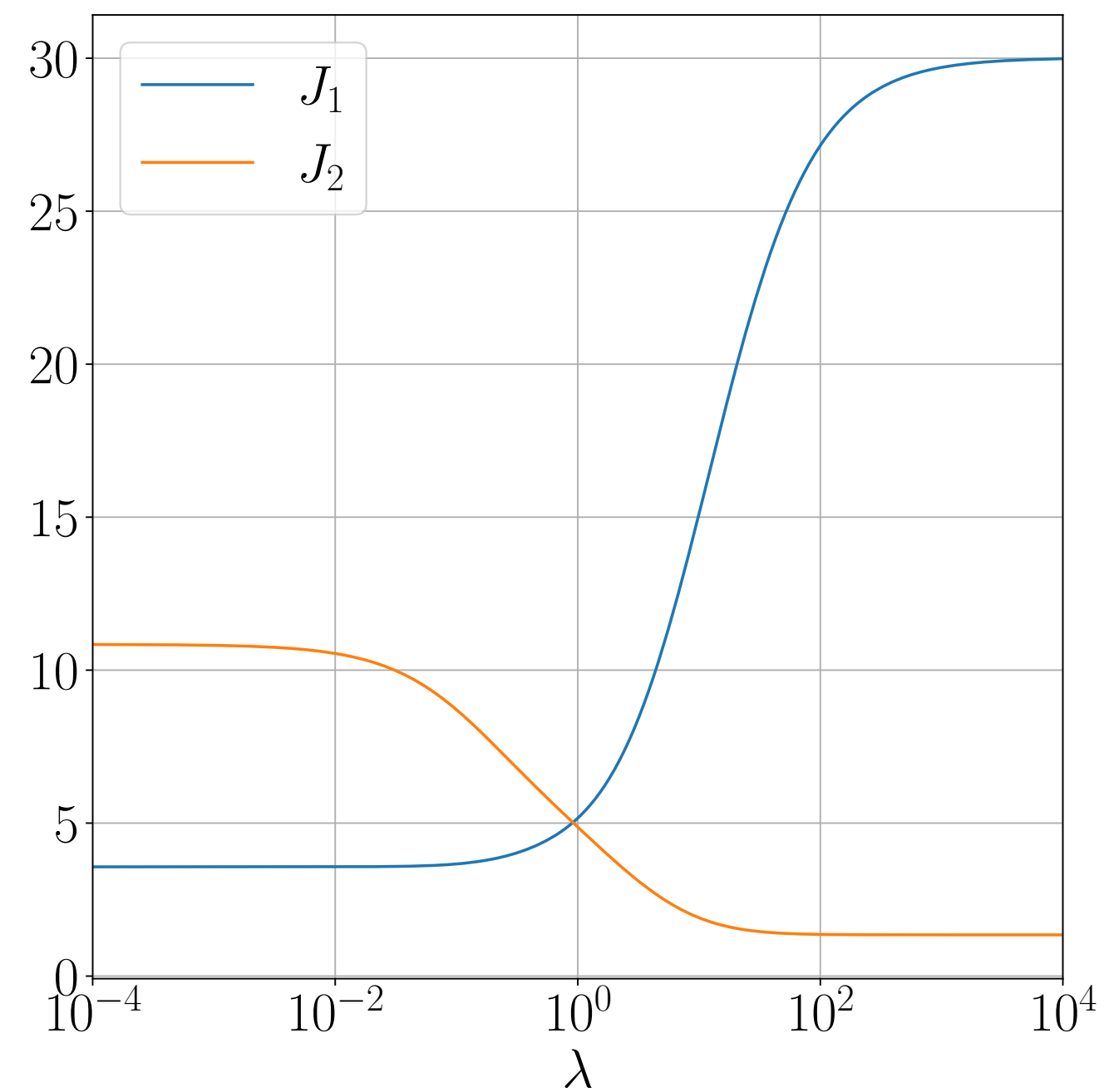
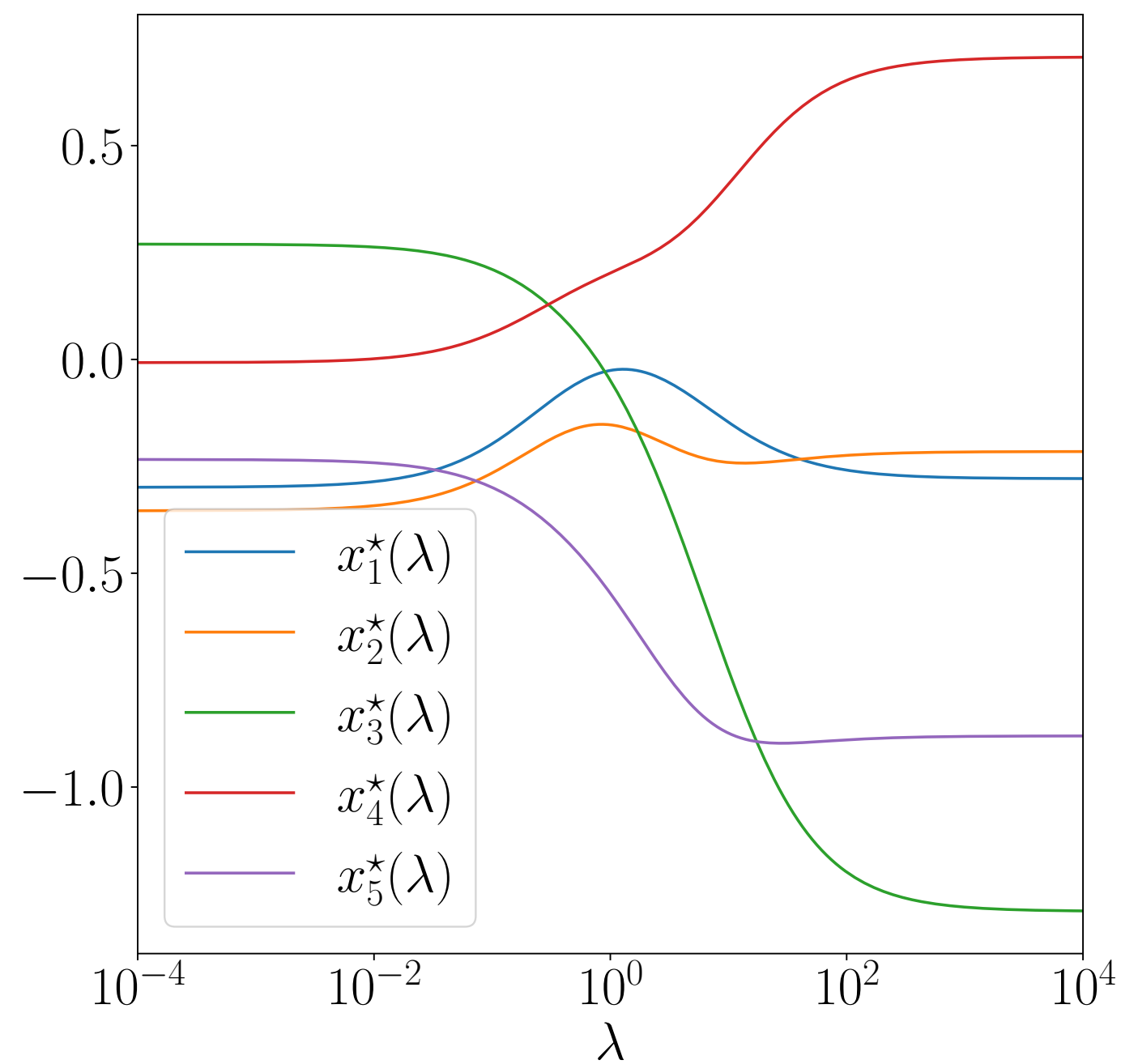
$$(J_1(x^*(\lambda)), J_2(x^*(\lambda))), \quad \lambda > 0$$

# Optimal trade-off curve

## Example

$$\text{minimize } J_1(x) + \lambda J_2(x)$$

( $A_1, A_2$  are both  $10 \times 5$ )



# Using multi-objective least squares

1. Identify **primary objective**  
basic quantity to minimize
2. Choose one or more **secondary objectives**  
quantities that we would like to be small, if possible  
(e.g., size of  $x$ , roughness of  $x$ , distance from give point)
3. Tweak/tune weights until we like  $x^*(\lambda)$

## **Bi-criterion problem**

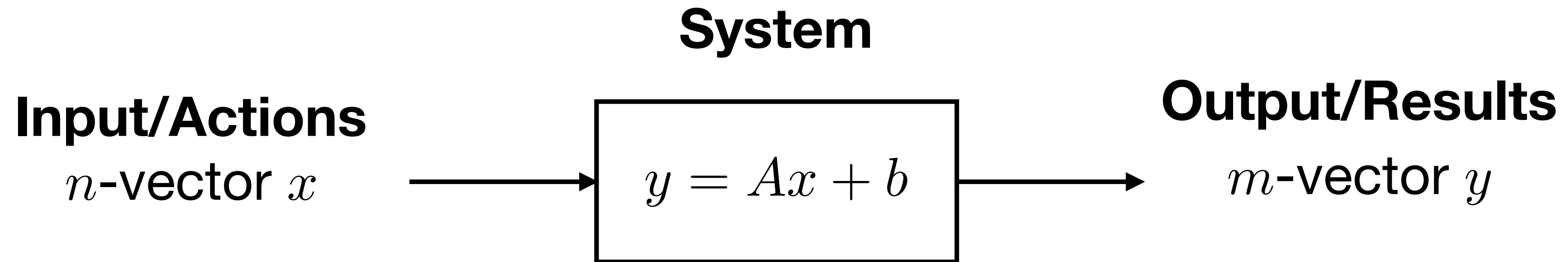
minimize  $J_1(x) + \lambda J_2(x)$

- If  $J_2$  too big, increase  $\lambda$
- If  $J_1$  too big, decrease  $\lambda$

**Control**



# Control



$A$  and  $b$  are the known *input-output mapping* of the system.  
(analytical models, data fitting, etc.)

## Goal

Choose  $x$  (which determines  $y$ ) to optimize  
multiple objectives of  $x$  and  $y$

# Multi-objective control

**Optimization problem**  
minimize  $J_1(x) + \lambda J_2(x)$

## Primary objective

$$J_1 = \|y - y^{\text{des}}\|^2$$

↑  
desired  
output

## Secondary objective

- $J_2 = \|x\|^2$   
(make  $x$  small)
- $J_2 = \|x - x^{\text{nom}}\|^2$   
( $x$  close to nominal input)

# Product demand shaping

Given  $n$ -products,  
induce change in demands,  $n$ -vector  $\delta^{\text{dem}}$ ,  
by adjusting prices,  $n$ -vector  $\delta^{\text{price}}$ ,

$$\delta^{\text{dem}} = E^{\text{d}} \delta^{\text{price}}$$

price elasticity of  
demand matrix

**example**  $E^{\text{d}}$

$$E^{\text{d}} = \begin{bmatrix} -0.4 & * & * \\ 0.2 & * & * \\ * & * & * \end{bmatrix}$$

$$\delta_1^{\text{price}} = 0.01$$

(first price +1%)



- $\delta_1^{\text{dem}} = -0.004$   
(first demand: -0.4%)
- $\delta_2^{\text{dem}} = 0.002$   
(second demand: +0.2%)

# Product demand shaping

## System

$$\delta^{\text{dem}} = E^{\text{d}} \delta^{\text{price}}$$

## Optimization problem

$$\text{minimize } J_1(x) + \lambda J_2(x)$$

### Primary objective

$$\begin{aligned} J_1 &= \|\delta^{\text{dem}} - \delta^{\text{tar}}\|^2 \\ &= \|E^{\text{d}} \delta^{\text{price}} - \delta^{\text{tar}}\|^2 \end{aligned}$$

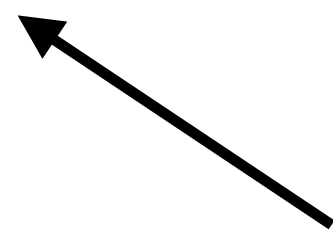
target  
demand



### Secondary objective

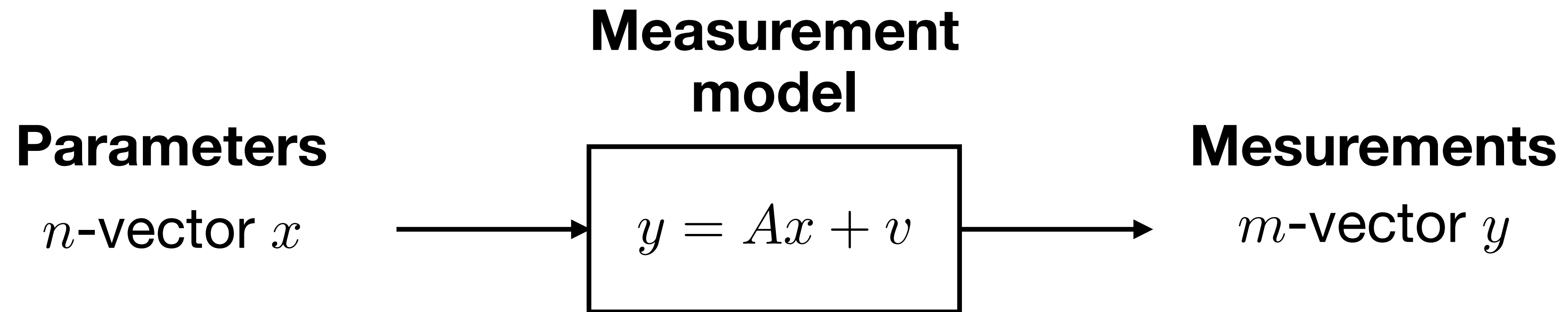
$$J_2 = \|\delta^{\text{price}}\|^2$$

don't change  
prices  
too much



# Estimation and inversion

# Estimation



$m$ -vector  $v$  are (unknown) *noises* or *measurement errors*

## Basic least squares estimation

(assuming  $v$  is small and  $A$  as independent columns)

$$\text{minimize } J_1 = \|Ax - y\|^2$$

# Regularized inversion

## Basic least squares estimation

(assuming  $v$  is small and  $A$  as independent columns)

$$\text{minimize } J_1 = \|Ax - y\|^2$$

## Regularization

We can get much better results by incorporating prior information about  $x$

•  $x$  small:  $J_2 = \|x\|^2$  (“Tikhonov regularization”)

•  $x$  is smooth:  $J_2 = \|Dx\|^2 = \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2$

•  $x$  close to prior:  $J_2 = \|x - x^{\text{prior}}\|^2$

$$D = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \end{bmatrix}$$

# Regularized inversion

## Optimization problem

$$\text{minimize } J_1(x) + \lambda J_2(x)$$

- Adjust  $\lambda$  until you are happy with the results
- curve  $x^*(\lambda)$  is the *regularization path*

## Example Tikhonov regularization

$$\text{minimize } \|Ax - y\|^2 + \lambda\|x\|^2 = \|\tilde{A}x - \tilde{b}\|^2$$

$$\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix}, \quad \tilde{b} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$\tilde{A}$  has always linearly independent columns

$$\tilde{A}x = (Ax, \sqrt{\lambda}x) = 0 \text{ if and only if } \sqrt{\lambda}x = 0 \Rightarrow x = 0$$



# Images representation

## Monochrome images

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

Each value  $X_{ij}$  represents a pixel's intensity (0 = black, 1 = white)  
(sometimes 0 = black, and 255 = white)

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

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

## Monochrome image



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

# Image de-blurring

Given a noisy **blurred image**  $y$  (vector form of  $Y$ )

**Model**  
 $y = Ax + v$

(blurring matrix  $A$ ,  
i.e., convolution)

## Least-squares de-blurring

Find  $x$  (vector form of  $X$ ) by solving

minimize  $\|Ax - y\|^2 + \lambda (\|D_v x\|^2 + \|D_h x\|^2)$

**Smoothing regularization**  
with weight  $\lambda$

**Vertical differences**

$$\sum_i \sum_j (X_{i,j+1} - X_{ij})^2$$

**Horizontal differences**

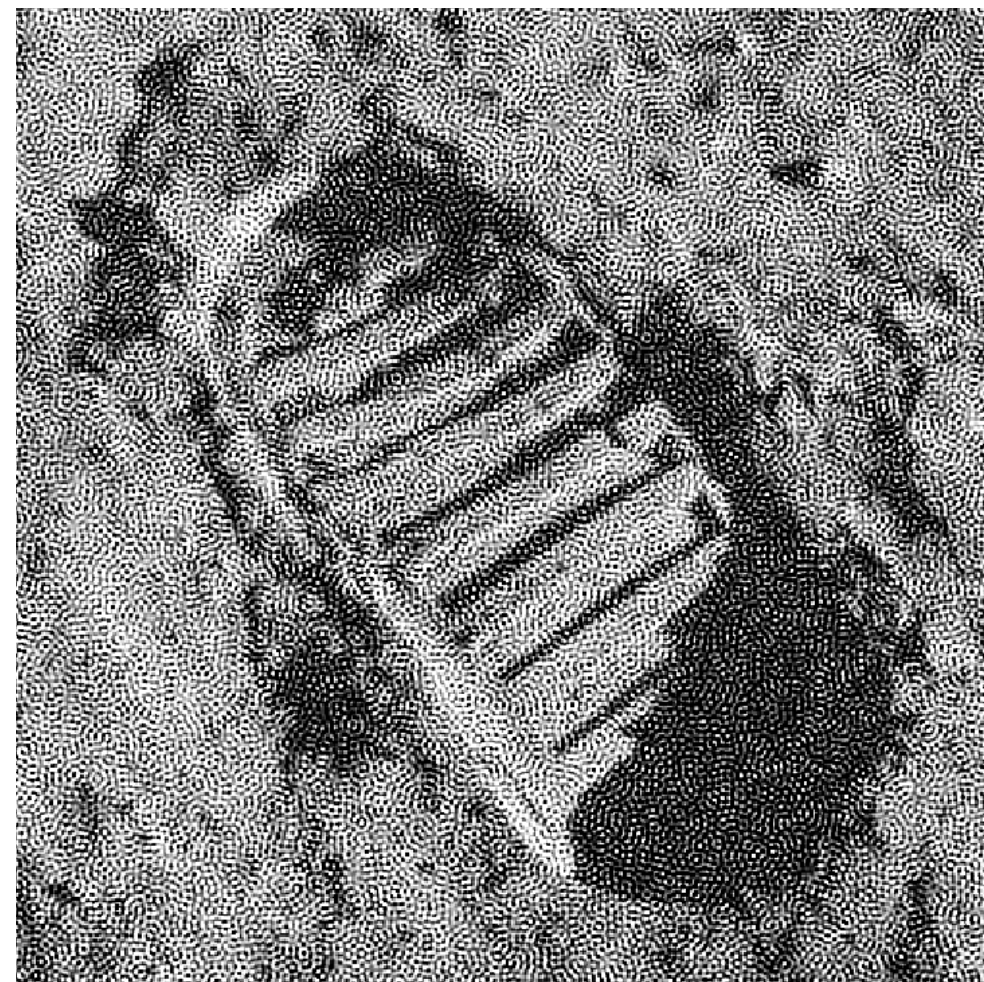
$$\sum_i \sum_j (X_{i+1,j} - X_{ij})^2$$

# Example

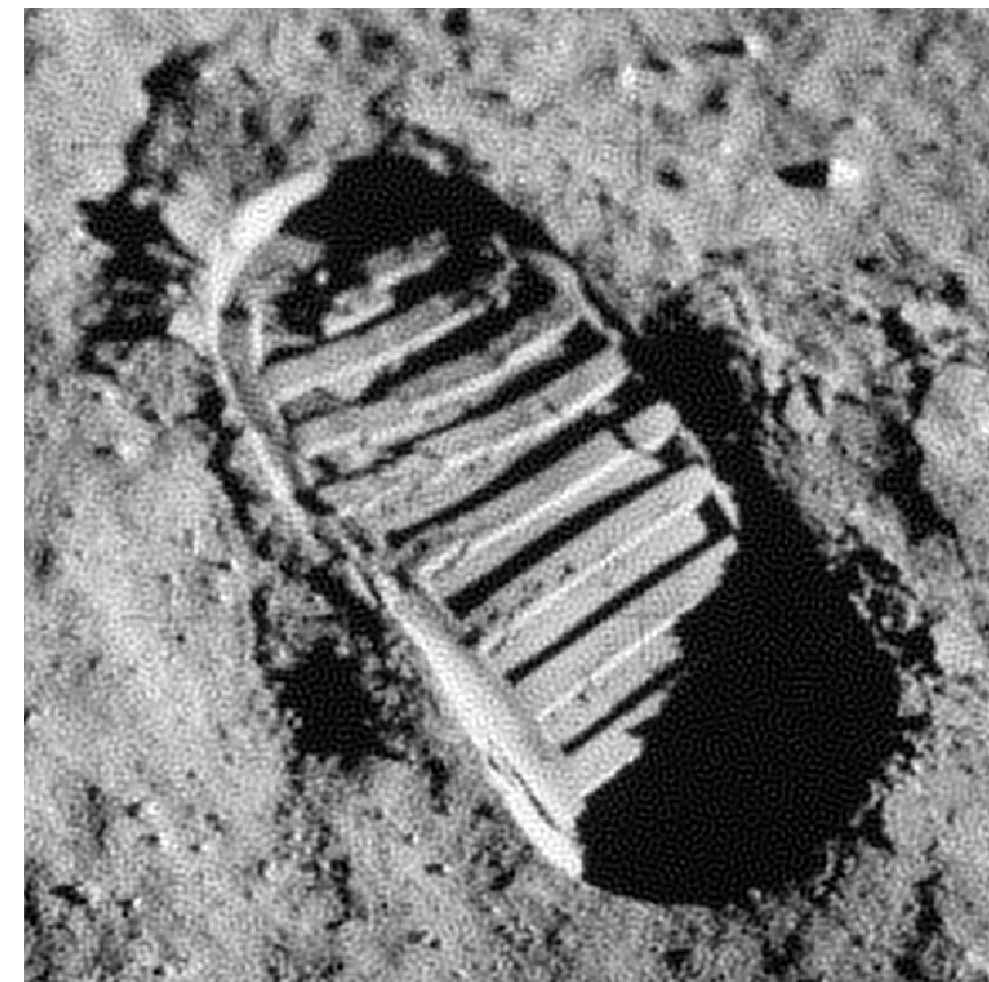
Blurred image



Regularization path



$$\lambda = 10^{-6}$$



$$\lambda = 10^{-4}$$



$$\lambda = 10^{-2}$$



$$\lambda = 10^{-1}$$

# Regularized data fitting

# Motivation for regularization

Consider the data fitting model (of  $y \approx f(x)$ )

$$\hat{f}(x) = \theta_1 f_1(x) + \dots + \theta_p f_p(x)$$

with  $f_1(x) = 1$

$\theta_i$  is the **sensitivity** of  $\hat{f}(x)$  to  $f_i(x)$   $\longrightarrow$  It cannot be too large!

Therefore, we want to **make**  $\theta_2, \dots, \theta_p$  **small**  
( $\theta_1$  is an exception, since  $f_1(x) = 1$  never changes)

# Regularized data fitting

Suppose we have training data

$n$ -vectors  $x^{(1)}, \dots, x^{(N)}$ , and scalars  $y^{(1)}, \dots, y^{(N)}$

We can express the training error as

$$A\theta - y$$

## Ridge regression

$$\hat{y}^{(i)} = (x^{(i)})^T \beta + v \longrightarrow \hat{y} = X^T \beta + v\mathbf{1}$$

## Regularized data fitting

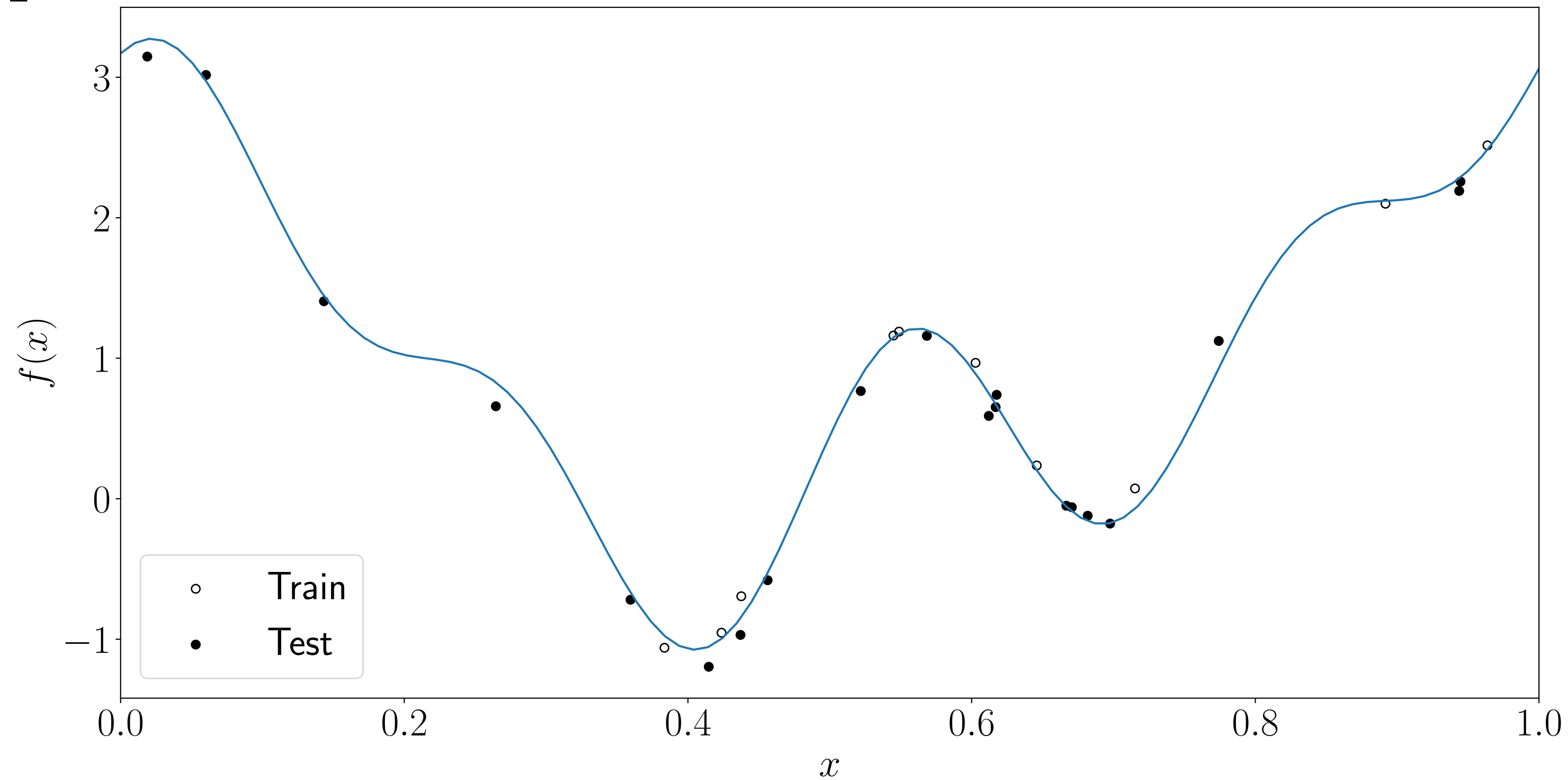
$$\text{minimize } \|A\theta - y\|^2 + \lambda \|\theta_{2:p}\|^2$$



$$\text{minimize } \|X^T \beta + v\mathbf{1} - y\|^2 + \lambda \|\beta\|^2$$

Choose  $\lambda$  with **validation**

# Example

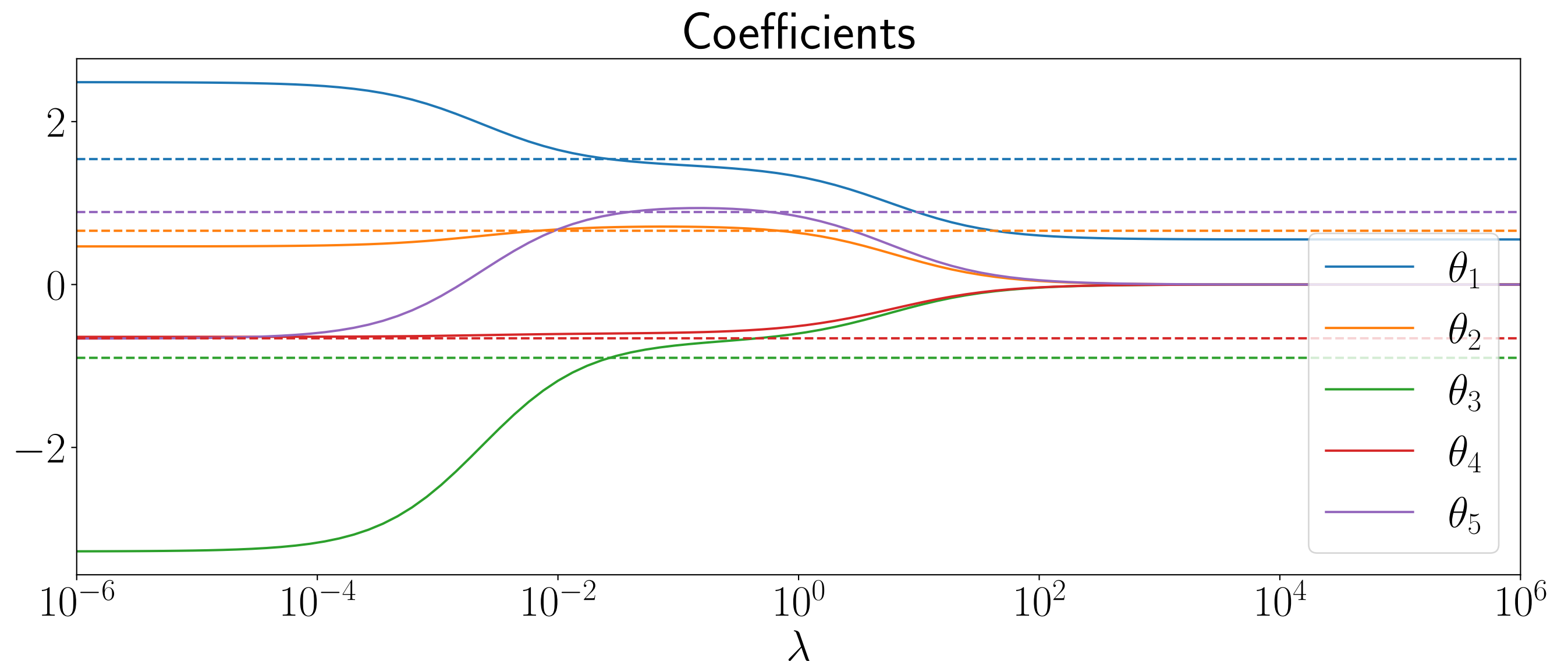
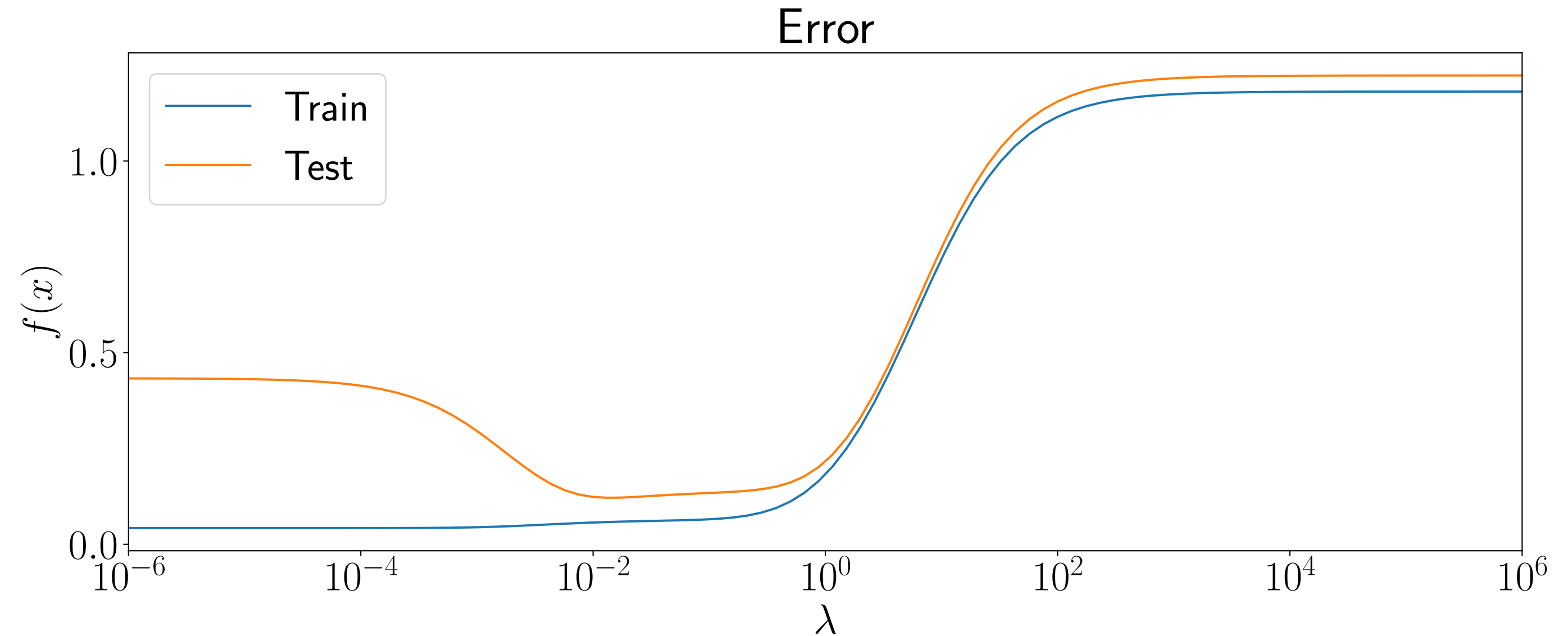


- Solid line to generate synthetic (simulated data)
- Fit a model with 5 parameters  $\theta_1, \dots, \theta_5$

$$\hat{f}(x) = \theta_1 + \sum_{k=1}^4 \theta_{k+1} \sin(w_k x + \phi_k), \quad \text{with given } w_k, \phi_k$$

# Train and test errors across regularization

- Minimum test error  $\lambda \approx 0.013$
- Dashed lines: coefficients to generate data
- For  $\lambda \approx 0.013$ , estimated coefficients close to true values
- $\theta_{2:5} \rightarrow 0$  as  $\lambda \rightarrow \infty$





# Multi-objective least-squares

Today, we learned to:

- **Recognize** and **write** multi-objective least squares problems
- **Solve** multi-objective least squares problems
- **Add regularization** to improve solutions performance

# References

- S. Boyd, L. Vandenberghe: Introduction to Applied Linear Algebra – Vectors, Matrices, and Least Squares
  - Chapter 15: multi-objective least squares

# Next lecture

- Constrained least squares