

From models to predictors

Hicham Janati · INSEA · hjanati@insea.ac.ma

From statistical modeling to statistical learning

Real estate data

apartment	area (m ²)	rooms	floor	age (yr)	dist. to tramway (km)	price (kDh)
1	62	3	2	14	3.1	710
2	118	5	6	32	7.4	1260
3	45	2	1	3	9.0	480
4	103	4	3	8	1.9	1410
⋮	⋮	⋮	⋮	⋮	⋮	⋮
240	87	4	5	21	2.2	935



A **government agency** asks: is the proximity to a tramway station a major factor in housing prices ?

You are the statistician. What do you **do** with this file to answer her ?

Step 1: postulate a model

Step 2: fit it, likelihood to least squares

$$\mathbb{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \text{ the observations,} \quad \mathbf{y} = (y_1, \dots, y_n)^\top \text{ the prices}$$

$$L(\boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2\sigma^2}\right)$$

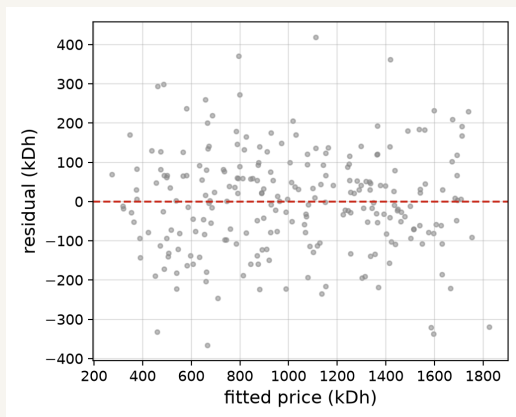
$$\max_{\boldsymbol{\beta}} L \iff \min_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \boldsymbol{\beta}^\top \mathbf{x}_i)^2 = \|\mathbf{y} - \mathbb{X}\boldsymbol{\beta}\|^2$$

$$\nabla_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbb{X}\boldsymbol{\beta}\|^2 = 2\mathbb{X}^\top (\mathbb{X}\boldsymbol{\beta} - \mathbf{y}) = 0 \implies \mathbb{X}^\top \mathbb{X} \boldsymbol{\beta} = \mathbb{X}^\top \mathbf{y}$$

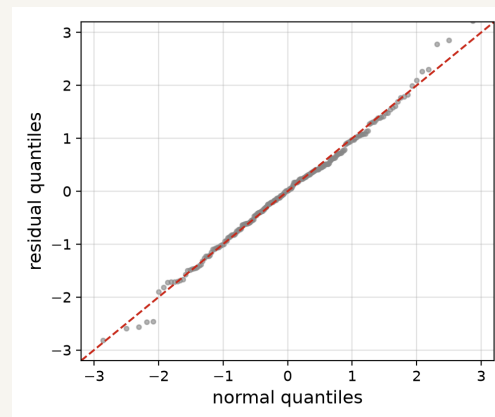
$$\hat{\boldsymbol{\beta}} = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top \mathbf{y}$$

Step 3: check it, diagnostics

(A1)-(A4) can never be **proved**: look for **red flags** in the residuals.



residuals vs fitted: no trend, no funnel



QQ plot: residuals close to Gaussian

No red flag: the inference is worth trusting.

Step 4: results and interpretation

factor	effect (kDh)	95% CI	p
area (per m ²)	+11.3	[10.4, 12.2]	< 10 ⁻⁶⁸
rooms	-9.2	[-33.1, 14.6]	0.45
floor	-4.5	[-11.9, 2.8]	0.23
age (per yr)	-2.0	[-3.1, -1.0]	2.1 · 10 ⁻⁴
distance (per km)	-17.8	[-23.5, -12.0]	4.9 · 10 ⁻⁹

Yes: every km away from the tramway costs about **18 kDh**.

The agency has its answer. This is **inference**: effects, intervals, tests, all resting on (A1)-(A4).

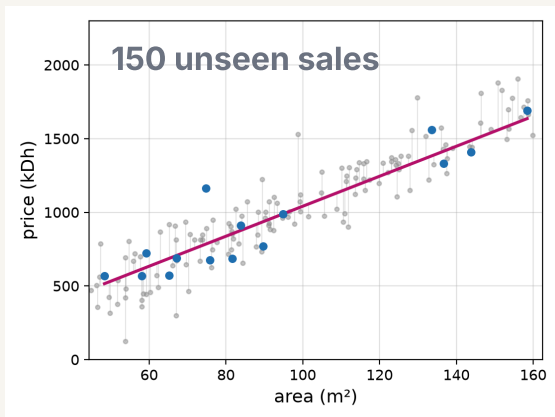
The investor: what will it sell for ?

The investor: what will it sell for ?

Testing on new data

To trust $\hat{\beta}$, first check that $\mathbf{x} \mapsto \hat{\beta}^\top \mathbf{x}$ predicts well on new data.

the original data: the 15 sales the model was fitted on



$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^{15} (y_i - \beta^\top \mathbf{x}_i)^2$$

$$\text{Root mean squared error (RMSE)} = \sqrt{\frac{1}{15} \sum_{i=1}^{15} (y_i - \hat{\beta}^\top \mathbf{x}_i)^2} = 15 \text{ kDh}$$

area (m ²)	price (kDh)
76	678
158	1692
75	1165
⋮	⋮

15 sales

And on sales it **never** saw ?

area (m ²)	price (kDh)	$\hat{y}$ (kDh)
88	767	919
67	814	705
⋮	⋮	⋮

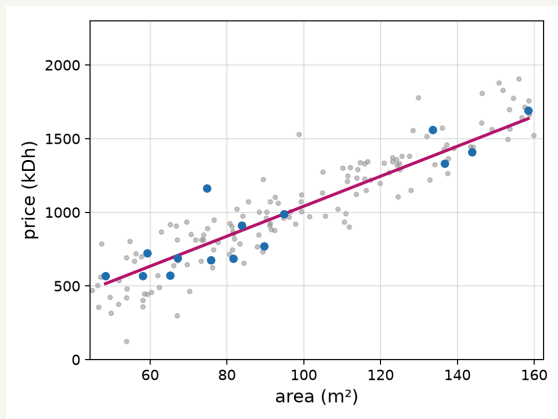
150 unseen

Unseen RMSE: **16 kDh.**

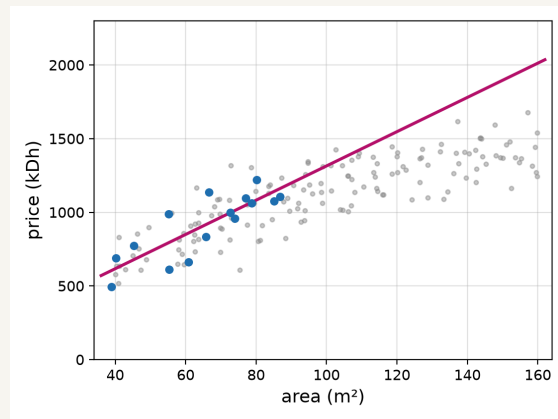
Generalization

The **same linear model**, fitted on two different datasets:

● train sales ● test sales (unseen)



good generalization



poor generalization

Both fits look fine on their 15 sales. Only **unseen data** tells them apart.

The train / test split

area	price
76	678
158	1692
75	1165
149	1564
54	692
⋮	⋮

all 100 sales



76	678
158	1692
⋮	⋮

train (80)

→ fit on **train**:

$$\hat{\beta} = \arg \min_{\beta} \sum_{i \in \text{train}} (y_i - \beta^{\top} \mathbf{x}_i)^2$$

train error: 137 kDh



149	1564
54	692
⋮	⋮

test (20)

→ evaluate prediction error on **test**:

$$\text{RMSE}_{\text{test}} = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i \in \text{test}} (y_i - \hat{\beta}^{\top} \mathbf{x}_i)^2}$$

test error: 128 kDh

The **test error** is the real measure of performance.

What if we only cared about prediction ?

Were **(A1)-(A4)** actually needed for these predictions ?

To predict a new apartment, we only needed $\hat{\beta}^\top \mathbf{x}_{\text{new}}$.

To compute $\hat{\beta}$, we minimized $\frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top \mathbf{x}_i)^2$.

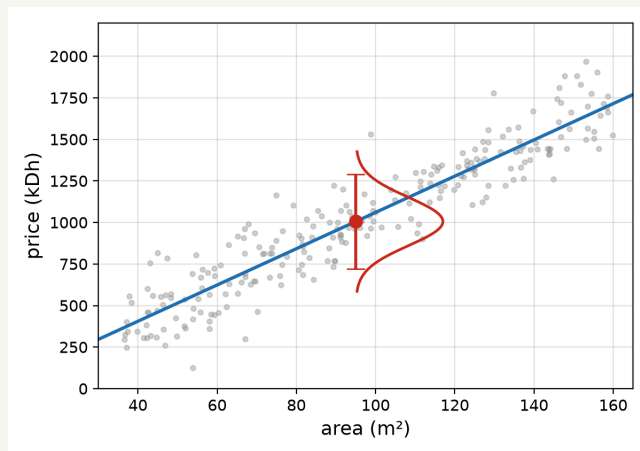
That is, we looked for the best linear function $f_\beta(\mathbf{x}) = \beta^\top \mathbf{x}$, the one minimizing the average prediction error:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^n (y_i - f_\beta(\mathbf{x}_i))^2$$

No assumption was used, except i.i.d. rows: **(A2)**.

Could (A1)-(A4) have improved the forecast ?

Suppose **(A1)-(A4)** do hold. What **changes** for the investor ?



least squares: **1006 kDh**, one number + (A4): a **distribution**, 95% interval **[722, 1290]**

$\hat{\beta}$ is the **same**, the forecasts are **identical**. (A4) adds the **uncertainty** around them:

$$\hat{\beta} \mid \mathbb{X} \sim \mathcal{N}(\beta, \sigma^2(\mathbb{X}^\top \mathbb{X})^{-1}) \implies Y_{\text{new}} - \hat{\beta}^\top \mathbf{x}_{\text{new}} \sim \mathcal{N}(0, \sigma^2(1 + \mathbf{x}_{\text{new}}^\top (\mathbb{X}^\top \mathbb{X})^{-1} \mathbf{x}_{\text{new}}))$$

Linear regression: Statistician vs Machine learner

	the statistician	the machine learner
object	a distribution price x with parameter β	a prediction function f_β
$\hat{\beta}$	$(\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T y$	$(\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T y$
assumptions	(A1)-(A4)	(A2) i.i.d.
purpose	explain, interpret	predict

The statistical approach predicts too, **with an interval for free**: under (A4), $\hat{\beta}$ is Gaussian.

If **(A2)** holds (i.i.d observations), then by the law of large numbers when $n \rightarrow +\infty$:

$$\frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \rightarrow \mathbb{E}(Y - f(\mathbf{X}))^2$$

Supervised learning

The learning problem

The data: rows $(\mathbf{x}_i, y_i)$, i.i.d. from an unknown distribution P .

The goal: learn a **prediction function** f so that $f(\mathbf{X}) \approx Y$ on a new draw.

The price of an error: a **loss** $L(y, \hat{y})$, for example

$$L(y, \hat{y}) = (y - \hat{y})^2 \quad (\text{squared})$$

$$L(y, \hat{y}) = |y - \hat{y}| \quad (\text{absolute})$$

Risk and empirical risk

The risk: the expected loss on a fresh draw

$$R(f) = \mathbb{E}[L(Y, f(\mathbf{X}))], \quad (\mathbf{X}, Y) \sim P \quad \text{unknown}$$

The empirical risk: its sample average, computable from the data.

$$\widehat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i))$$

Empirical risk minimization: minimize it over a chosen **family** $\mathcal{F}$.

$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \widehat{R}_n(f)$$

Previous section: $\mathcal{F} = \{f_{\boldsymbol{\beta}}(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x}\}$, the **linear** functions, with the squared loss.

How should we choose $\mathcal{F}$?

The fraud detective

$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) \quad \text{how big should } \mathcal{F} \text{ be ?}$$

Detect fraudulent card transactions. The training file (fictional):

id	amount (Dh)	hour	card name	phone	fraud
1041	84	14:12	Salma Alaoui	06 61 20 44 27	0
1042	445	02:17	Ilyas Benali	06 67 51 23 81	1
1043	310	01:44	Nadia Berrada	06 42 77 90 58	0
1044	470	02:23	Ilyas Benali	06 67 51 23 81	1
...	...	...	...	...	...

Example of a function with $\mathcal{F}$ **too small** ?

*fraud if **amount > 1000 Dh** and **hour in 0h-3h**.*

Decision rule is too simple

What if $\mathcal{F}$ is **too large** ?

*fraud if **hour in 2:10-2:25** and **amount > 330 Dh** and **phone ends with 81** and **first name is Ilyas**.*

Decision rule too complex

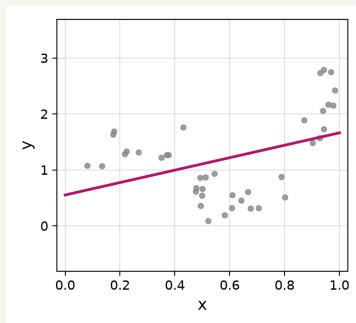
Bias and variance

These estimators have:

high or low bias ?

high or low variance ?

small $\mathcal{F}$

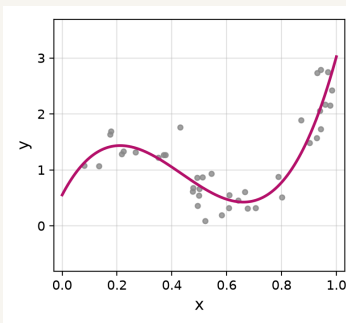


underfits

high bias

low variance

medium $\mathcal{F}$: matches the data

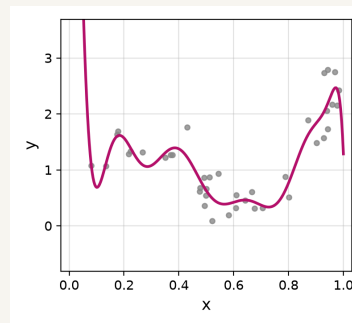


just right

average bias

average variance

large $\mathcal{F}$



overfits

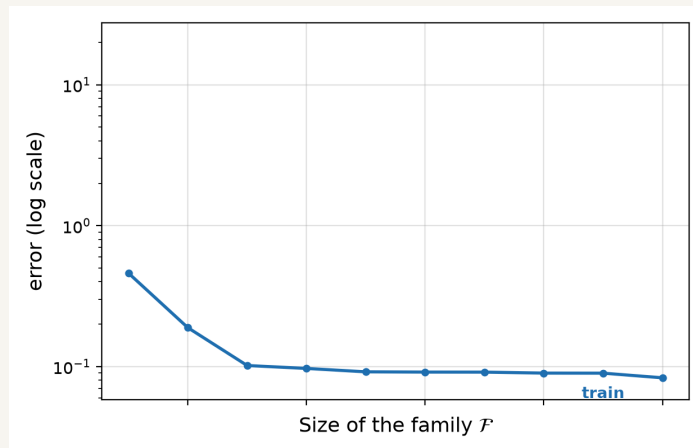
low bias

high variance

A **tradeoff**: growing $\mathcal{F}$ trades bias away for variance.

The U-curve

How should the **train** error evolve as the size of $\mathcal{F}$ grows ?

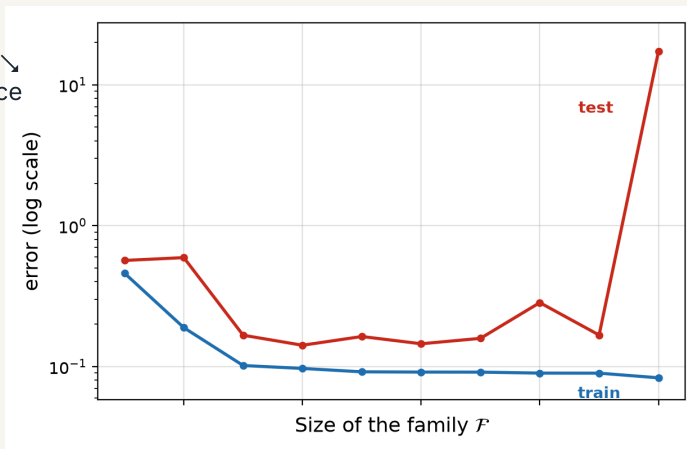


And the **test** error ?

The U-curve

How should the **train** error evolve as the size of $\mathcal{F}$ grows ?

underfitting regime ↘
high bias, low variance



↗ **overfitting regime**
low bias, high variance

And the **test** error ?

Watching a model overfit

Can linear regression overfit ?

A linear model looks too simple to build a complex rule. Can it overfit ?

Yes, with additional (*engineered* non-linear) features:

$$f_{\beta}(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Can linear regression overfit ?

A linear model looks too simple to build a complex rule. Can it overfit ?

Yes, with additional (*engineered* non-linear) features:

$$f_{\beta}(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2$$

Can linear regression overfit ?

A linear model looks too simple to build a complex rule. Can it overfit ?

Yes, with additional (*engineered* non-linear) features:

$$f_{\beta}(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2$$

Can linear regression overfit ?

A linear model looks too simple to build a complex rule. Can it overfit ?

Yes, with additional (*engineered* non-linear) features:

$$f_{\beta}(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$



The model is still linear in β , the same math applies with a new design matrix:

$\mathbb{X} =$	x_1	x_2
	2	3
	1	5
	$\vdots$	$\vdots$

Can linear regression overfit ?

A linear model looks too simple to build a complex rule. Can it overfit ?

Yes, with additional (*engineered* non-linear) features:

$$f_{\beta}(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$



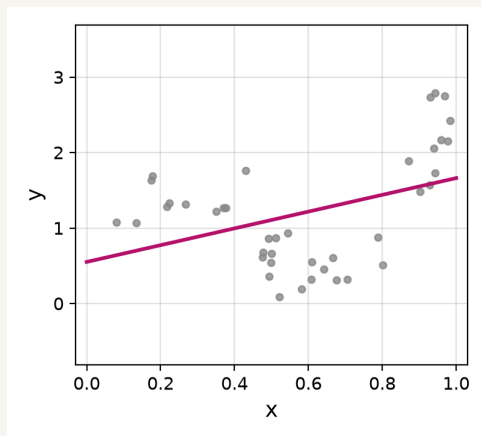
The model is still linear in β , the same math applies with a new design matrix:

$\mathbb{X} =$	x_1	x_2	x_1^2	x_2^2	$x_1 x_2$
	2	3	4	9	6
	1	5	1	25	5
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

The **same** OLS formula applies:

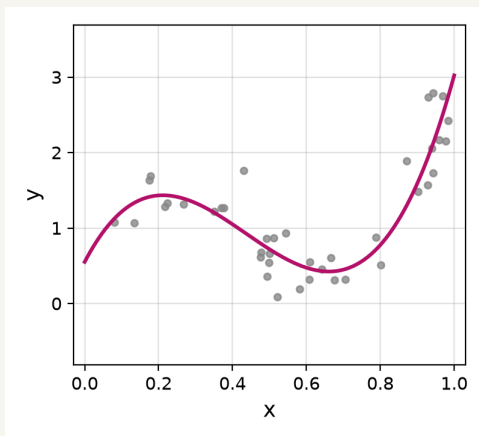
$$\hat{\beta} = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T \mathbf{y}$$

Watching it overfit



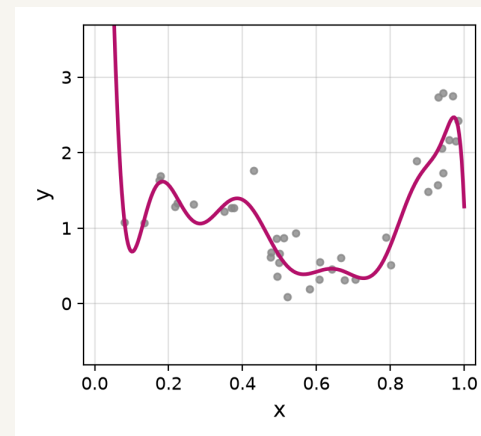
degree 1

train MSE 0.46 · test MSE 0.57



degree 3

train MSE 0.10 · test MSE 0.17



degree 12

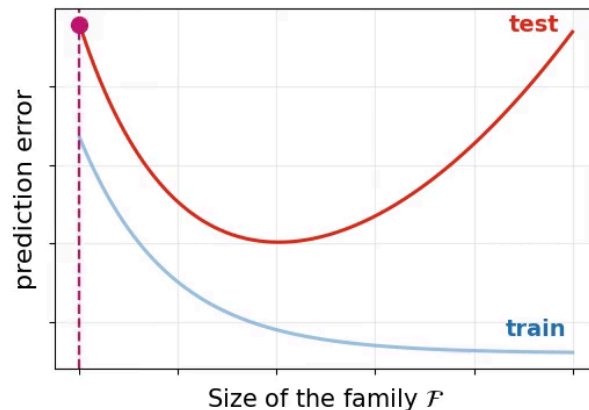
train MSE 0.08 · test MSE 6.5

Adding features always improves the **training** fit. Only the **test error** climbs again.

A dial for complexity



Complexity knob



How does the **training** error evolve as $\mathcal{F}$ grows ?

And the **test** error ?

Recap: test error traces a **U** in the complexity of $\mathcal{F}$. It would be nice to control it somehow:



How can we **make** such a knob ?

Regularization: Ridge, Lasso, Elastic-net

Shrink the family, continuously

To optimize over $\mathcal{F}$, we need a finite-dimensional set: a function is identified with a parameter $\beta \in \mathbb{R}^d$.

e.g. linear regression: $f_\beta(\mathbf{x}) = \beta^\top \mathbf{x}$

To control the **size** of $\mathcal{F}$, constrain $\beta \in \mathcal{C} \subset \mathbb{R}^d$.

$\mathcal{C}$ small $\iff \mathcal{F}$ small.

Constrained ERM

$$\min_{\beta \in \mathcal{C}} \widehat{R}_n(f_\beta)$$

Simplify: a **size** function $\Omega(\beta) \geq 0$ defines $\mathcal{C}$,

$$\mathcal{C} = \{\beta \in \mathbb{R}^d : \Omega(\beta) \leq \tau\}$$

$$\min_{\beta \in \mathbb{R}^d} \widehat{R}_n(f_\beta) \quad \text{subject to} \quad \Omega(\beta) \leq \tau$$

Tikhonov (1963): this is equivalent to

$$\min_{\beta \in \mathbb{R}^d} \widehat{R}_n(f_\beta) + \lambda \Omega(\beta)$$

Shrink the family, continuously

To optimize over $\mathcal{F}$, we need a finite-dimensional set: a function is identified with a parameter $\beta \in \mathbb{R}^d$.

e.g. linear regression: $f_\beta(\mathbf{x}) = \beta^\top \mathbf{x}$

To control the **size** of $\mathcal{F}$, constrain $\beta \in \mathcal{C} \subset \mathbb{R}^d$.

$\mathcal{C}$ small $\iff \mathcal{F}$ small.

Constrained ERM

$$\min_{\beta \in \mathcal{C}} \widehat{R}_n(f_\beta)$$

Simplify: a **size** function $\Omega(\beta) \geq 0$ defines $\mathcal{C}$,

$$\mathcal{C} = \{\beta \in \mathbb{R}^d : \Omega(\beta) \leq \tau\}$$

$$\min_{\beta \in \mathbb{R}^d} \widehat{R}_n(f_\beta) \quad \text{subject to} \quad \Omega(\beta) \leq \tau$$

Tikhonov (1963): this is equivalent to

$$\min_{\beta \in \mathbb{R}^d} \underbrace{\widehat{R}_n(f_\beta)}_{\text{data fitting}} + \underbrace{\lambda \Omega(\beta)}_{\text{regularization}}$$

The effect of λ

$$\min_{\beta \in \mathbb{R}^d} \underbrace{\widehat{R}_n(f_\beta)}_{\text{data fitting}} + \underbrace{\lambda \Omega(\beta)}_{\text{regularization}}$$

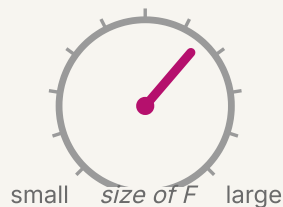
What is the effect of λ ?

$\lambda \rightarrow 0$  $\lambda \rightarrow \infty$

$\lambda = 0.25$

$\lambda = 0$?

Large $\mathcal{F}$



$\lambda = \infty$?

$\mathcal{F} = \{f_0\}$

We can control complexity through a **continuous dial**: λ is called a **hyperparameter**

Which penalty Ω ?

From now on, **regression** with the **squared loss**: the data-fitting term is $\frac{1}{n} \|\mathbf{y} - \mathbb{X}\boldsymbol{\beta}\|^2$,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^d} \underbrace{\frac{1}{n} \|\mathbf{y} - \mathbb{X}\boldsymbol{\beta}\|^2}_{\text{data fitting}} + \underbrace{\lambda \Omega(\boldsymbol{\beta})}_{\text{regularization / penalty}}$$

Which Ω ? What should it measure ?

The **size** of $\boldsymbol{\beta}$: $\Omega(\boldsymbol{\beta}) \geq 0$, $\Omega(\mathbf{0}) = 0$, growing with the coefficients.

ℓ_2 penalty: **ridge**

$$\Omega(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{j=1}^d \beta_j^2$$

ℓ_1 penalty: **lasso**

$$\Omega(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^d |\beta_j|$$

Start with ℓ_2 : what does $\hat{\boldsymbol{\beta}}_\lambda$ look like ?

Ridge

Exercise 1

Ridge in closed form

Assume $\mathbb{E}(X) = 0$. Let $\widehat{\Sigma} = \frac{1}{n} \mathbb{X}^\top \mathbb{X}$ and $\lambda > 0$.

1. Compute the gradient of $\beta \mapsto \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \|\beta\|^2$ and write the first-order condition.

2. Can $\widehat{\Sigma} + \lambda \mathbf{I}$ be singular? Deduce a closed form for $\hat{\beta}_\lambda$.

3. Columns of $\mathbb{X}$ are strongly correlated. Compare $\hat{\beta}_\lambda$ with the OLS estimator ($\lambda = 0$).

4. Assume $\widehat{\Sigma} = \mathbf{I}$. What is $\hat{\beta}_\lambda$ as a function of $\hat{\beta}_{OLS}$?

Ridge regression, solved

$$\nabla_{\beta} = \frac{2}{n} \mathbb{X}^{\top} (\mathbb{X}\beta - \mathbf{y}) + 2\lambda\beta = 0 \implies \left(\frac{1}{n}\mathbb{X}^{\top}\mathbb{X} + \lambda\mathbf{I}\right)\beta = \frac{1}{n}\mathbb{X}^{\top}\mathbf{y}$$

with $\widehat{\Sigma} = \frac{1}{n}\mathbb{X}^{\top}\mathbb{X}$ (standardized covariates):

$$\hat{\beta}_{\lambda} = (\widehat{\Sigma} + \lambda\mathbf{I})^{-1} \frac{1}{n}\mathbb{X}^{\top}\mathbf{y}$$

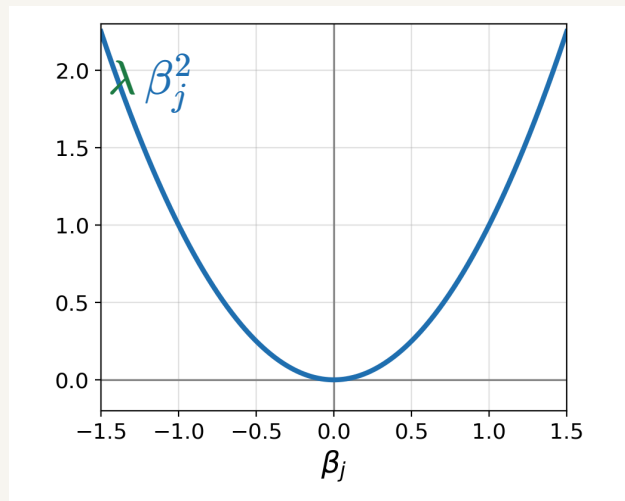
$\lambda > 0$: every eigenvalue of $\widehat{\Sigma} + \lambda\mathbf{I}$ is at least λ . Ridge **stabilizes multicollinearity**.

More columns than rows, $d > n$: is the solution still unique ?

Ridge: $\widehat{\Sigma} + \lambda\mathbf{I}$ stays invertible: a **unique** solution, even when $d > n$.

Ridge effect on β

$$\hat{\beta}_\lambda = \operatorname{argmin}_{\beta} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \|\beta\|_2^2$$



Where is $\lambda \|\beta\|_2^2$
minimal ?

At $\beta = \mathbf{0}$: every coordinate is
pulled toward 0.

For a coordinate β_j :
what is $\partial_j \lambda \|\beta\|_2^2$?

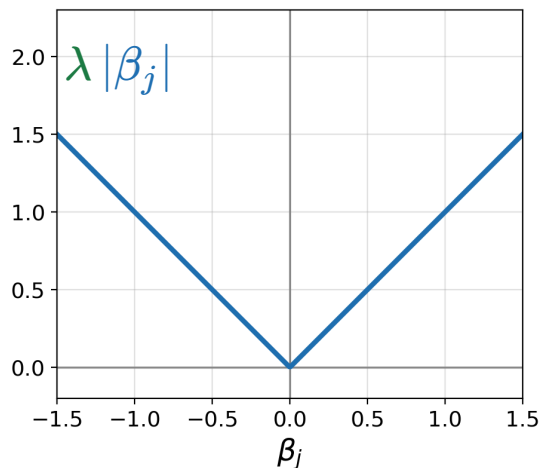
$2\lambda\beta_j$. What does it mean when
 $\beta_j \rightarrow 0$?

Every β_j is pushed toward 0 but **never actually reaches** 0: ridge gives no exact zeros.

Lasso

Lasso: effect of ℓ_1

$$\hat{\beta}_{\lambda}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \sum_{j=1}^d |\beta_j|$$



Where is $\lambda\|\beta\|_1$
minimal ?

At $\beta = \mathbf{0}$: every coordinate is
pulled toward 0.

For $\beta_j \neq 0$ the norm is
differentiable: what is
 $\partial_j \lambda\|\beta\|_1$?

for $\beta_j > 0$: $+\lambda$; for $\beta_j < 0$: $-\lambda$.
Whatever the size of β_j !

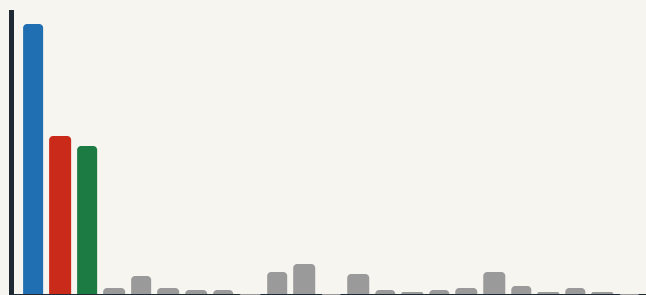
With a large enough λ , $\hat{\beta}_{\lambda}$ has **lots of zeros**: it is **sparse** (parcimonieux).

Ridge vs Lasso

Ridge vs lasso, turning the dial

Ridge (ℓ_2 norm)

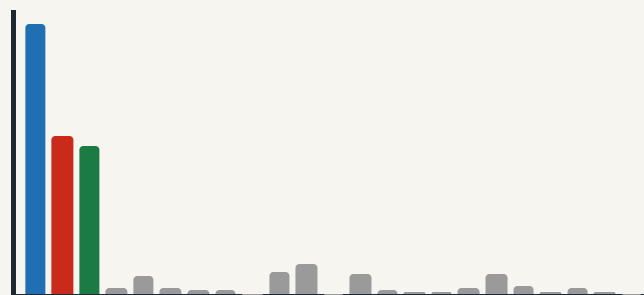
$$\min_{\beta} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \sum_j \beta_j^2$$



ridge: 23 / 23 nonzero

Lasso (ℓ_1 norm)

$$\min_{\beta} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \sum_j |\beta_j|$$



lasso: 23 / 23 nonzero

$\lambda = 1.0\text{e-}3$



area

age

distance

20 columns with no effect

ridge: all shrink, none hits zero

lasso: coefficients die **one by one**

When is **sparsity** useful in practice ?

The lasso in genomics

Data: **tumor growth (%)** + differential expression of **10 000 genes** on **200 patients**.

patient	gene 1	gene 2	...	gene 10 000	tumor growth
1	0.09	-0.01	...	-1.79	0.31
2	-0.21	1.92	...	1.08	-1.64
3	0.45	-1.21	...	-1.98	-6.88
⋮	⋮	⋮		⋮	⋮

$$y_i = \beta_0 + \beta^\top \mathbf{x}_i + \varepsilon_i, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad d = 10\,000 > n = 200$$

Biology says: it is very unlikely to have more than 20 genes being significant in tumor growth.

Formally, what does this say about β ?

Sparsity: at most 20 nonzero coordinates,
 $\#\{j : \beta_j \neq 0\} \leq 20$. Most $\beta_j = 0$.

Which penalty, when ?

Lasso (ℓ_1)

- a **sparsity** bias: few relevant features
- **feature selection**: a short list to study
- $d \gg n$ with few true signals

Ridge (ℓ_2)

- many **small** effects, nothing to drop
- **correlated** columns: the effect is shared (the lasso keeps one of them)
- a unique, stable solution in closed form:
$$\hat{\beta}_\lambda = (\widehat{\Sigma} + \lambda \mathbf{I})^{-1} \frac{1}{n} \mathbb{X}^\top \mathbf{y}$$

Both ? **Elastic net**: $\Omega(\beta) = \alpha \|\beta\|_1 + (1 - \alpha) \|\beta\|_2^2$, a **second** hyperparameter $\alpha \in [0, 1]$

Preprocessing

Everything in this section holds for any penalty Ω : ridge is written for clarity.

Ridge regression: the intercept

$$\mathcal{F} = \{f : \mathbf{x} \mapsto \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}, \beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^d\}$$

(A) ~~$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \boldsymbol{\beta}^\top \mathbf{x}_i)^2 + \lambda (\beta_0^2 + \|\boldsymbol{\beta}\|^2)$$~~

λ very large $\Rightarrow \hat{\beta}_0 \approx 0, \hat{\boldsymbol{\beta}} \approx \mathbf{0} \Rightarrow \hat{f}(\mathbf{x}) \approx 0$: every prediction is **0**

(B)
$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \boldsymbol{\beta}^\top \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\beta}\|^2$$

λ very large $\Rightarrow \hat{\boldsymbol{\beta}} \approx \mathbf{0} \Rightarrow \hat{\beta}_0 \approx \bar{y} \Rightarrow \hat{f}(\mathbf{x}) \approx \bar{y}$: the **average price**

Which formulation is better ? Compare their predictions $\hat{f}(\mathbf{x})$ for a **very large** λ .

We use (B): Strong regularization should remove the influence of the features, not force the predicted price to zero.

Exercise 2

Centering removes the intercept

Penalized regression with an intercept, $\Omega(\beta) = \|\beta\|_2^2$ (ridge) or $\|\beta\|_1$ (lasso):

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta^\top \mathbf{x}_i)^2 + \lambda \Omega(\beta).$$

1. For a fixed β , find the β_0 that minimizes the objective. Does it depend on Ω ?
2. Plug it back: show that β solves the **same penalized problem without intercept** on the centered data $y_i - \bar{y}$, $\mathbf{x}_i - \bar{\mathbf{x}}$.
3. Given the solution $\hat{\beta}_\lambda$ of the centered problem, how do you recover $\hat{\beta}_0$?

In practice this is how Ridge and Lasso are carried out using centered $\mathbb{X}$ and $\mathbf{y}$.

Ridge and units

OLS on two versions of the housing data:

distance in km

$$\widehat{\text{price}} = \hat{\beta}_0 + \hat{\beta}_{\text{area}} \text{area} + \hat{\beta}_{\text{dist}} \text{dist}_{\text{km}}$$

distance in meters

$$\widehat{\text{price}} = \hat{\beta}'_0 + \hat{\beta}'_{\text{area}} \text{area} + \hat{\beta}'_{\text{dist}} \text{dist}_{\text{m}}$$

Are these two formulations **equivalent** ?

Yes: $\text{dist}_{\text{m}} = 1\,000 \text{dist}_{\text{km}}$, so $\hat{\beta}'_{\text{dist}} \text{dist}_{\text{m}} = (1\,000 \hat{\beta}'_{\text{dist}}) \text{dist}_{\text{km}}$.

Same fit, **identical predictions**: OLS does not care about units.

Ridge and units

Ridge, the same λ , the same two versions of the data:

distance in km

$$\min \frac{1}{n} \sum_i (\dots)^2 + \lambda (\beta_{\text{area}}^2 + \beta_{\text{dist}}^2)$$

distance in meters

$$\min \frac{1}{n} \sum_i (\dots)^2 + \lambda (\beta_{\text{area}}'^2 + \beta_{\text{dist}}'^2)$$

Are these two formulations **equivalent** ?

No: with the same change of variable $\beta_{\text{dist}} = 1\,000 \beta_{\text{dist}}'$, the meters penalty becomes

$$\lambda \left(\beta_{\text{area}}^2 + \frac{\beta_{\text{dist}}^2}{10^6} \right)$$

With Ridge: a different scale is a **different** problem.

Standardize the columns

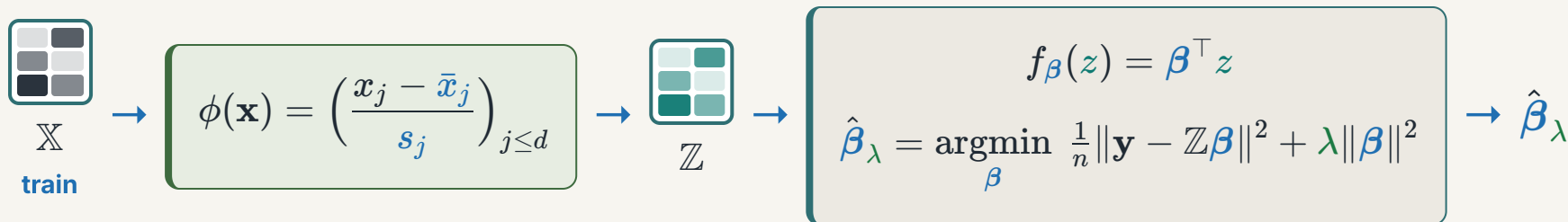
How do we make the penalty independent of the **units** and of the **scales** ?

Standardize every column j , with its mean $\bar{x}_j$ and standard deviation s_j :

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

Ridge on the standardized columns: $\widehat{\text{price}} = \tilde{\beta}_0 + \tilde{\beta}_{\text{area}} z_{\text{area}} + \tilde{\beta}_{\text{dist}} z_{\text{dist}}$

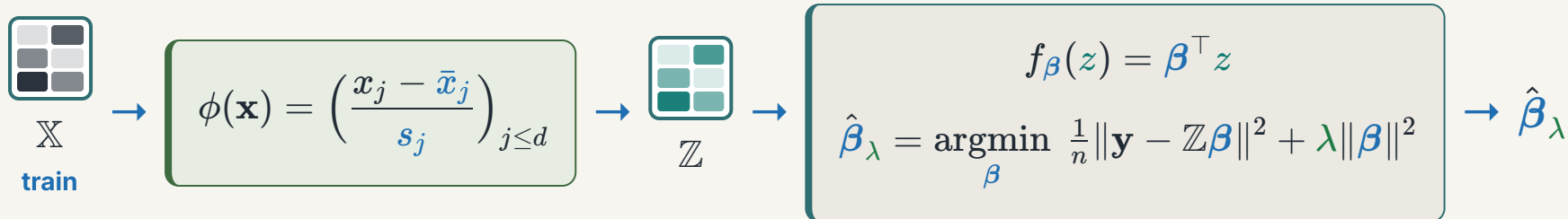
Standardize on train, apply on test



$$\hat{f}(\mathbf{x}) = f_{\hat{\beta}_{\lambda}} \circ \phi(\mathbf{x}) = \sum_{j=1}^d \hat{\beta}_{\lambda,j} \frac{x_j - \bar{x}_j}{s_j}$$

What are the **parameters** of $\hat{f}$?

Standardize on train, apply on test



$$\hat{f}(\mathbf{x}) = f_{\hat{\beta}_{\lambda}} \circ \phi(\mathbf{x}) = \sum_{j=1}^d \boxed{\hat{\beta}_{\lambda,j}} \frac{x_j - \boxed{\bar{x}_j}}{\boxed{s_j}}$$

What are the **parameters** of $\hat{f}$?

$\hat{\beta}_{\lambda}$, and the scaler's $\bar{x}_j$, s_j : all learned on $\mathbb{X}$.

A new row $\mathbf{x}'$ (**test**): how do we predict it ?

$\hat{y}' = \hat{f}(\mathbf{x}') = \hat{\beta}_{\lambda}^{\top} \phi(\mathbf{x}')$: nothing re-learned.



Categorical variables

Columns that are not numbers

apartment	area (m ²)	dist. to tramway (km)	listing ID	ad posted on	door color	neighborhood	agency	price (kDh)
1	62	3.1	48213	Tuesday	green	Agdal	A	710
2	118	7.4	50977	Sunday	white	Hay Riad	B	1260
3	45	9.0	51302	Friday	brown	Océan	A	480
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

How do we feed the **neighborhood** to a linear model $\beta_0 + \beta^\top \mathbf{x}$?

Agdal = 1, Hay Riad = 2, Océan = 3 ?

No: This implicitly creates an *order* between the different neighborhoods

One-hot encoding

neighborhood		Agdal	Hay Riad	Océan	...
Agdal	→	1	0	0	...
Hay Riad		0	1	0	...
Océan		0	0	1	...

One-hot encoding: Every category becomes a binary variable.

Each neighborhood gets its **own** price effect:

$$\beta_0 + \beta_{\text{Agdal}} x_{\text{Agdal}} + \beta_{\text{H.Riad}} x_{\text{H.Riad}} + \beta_{\text{Océan}} x_{\text{Océan}} + \dots$$

These binary columns are called **dummy variables**.

One column too many

intercept	area (m ²)	Agdal	Hay Riad	Océan	...
1	62	1	0	0	...
1	118	0	1	0	...
1	45	0	0	1	...

Fit this $\mathbb{X}$ by OLS: what goes wrong ?

Multicollinearity: $\mathbb{X}^\top \mathbb{X}$ is singular, so OLS is **not uniquely defined**.

How do we repair OLS ?

Drop one of the categories: no information is lost !

The reference category

Assume the **Agdal** column dropped, then:

$$\mathbb{E}(\text{price} \mid \mathbf{x}) = \beta_0 + \beta_{\text{area}} \text{area} + \beta_{\text{H.Riad}} x_{\text{H.Riad}} + \beta_{\text{Océan}} x_{\text{Océan}}$$

Average price of an **Agdal** apartment ?

$$\beta_0 + \beta_{\text{area}} \text{area}$$

Average price of a **H.Riad** apartment ?

$$\beta_0 + \beta_{\text{area}} \text{area} + \beta_{\text{H.Riad}}$$

What does $\beta_{\text{H.Riad}}$ measure ?

The price gap **H.Riad** — **Agdal** at equal area: **Agdal** is the **reference** category.

Drop **Océan** instead of **Agdal**: what changes ?

The coefficients change (new reference),
the **predictions do not**.

Ridge keeps every category

$$\hat{\beta}_{\lambda} = \underset{\beta}{\operatorname{argmin}} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda \|\beta\|^2, \quad \mathbb{X} \text{ with every neighborhood column}$$

Agdal	Hay Riad	Océan	...
1	0	0	...
0	1	0	...
0	0	1	...

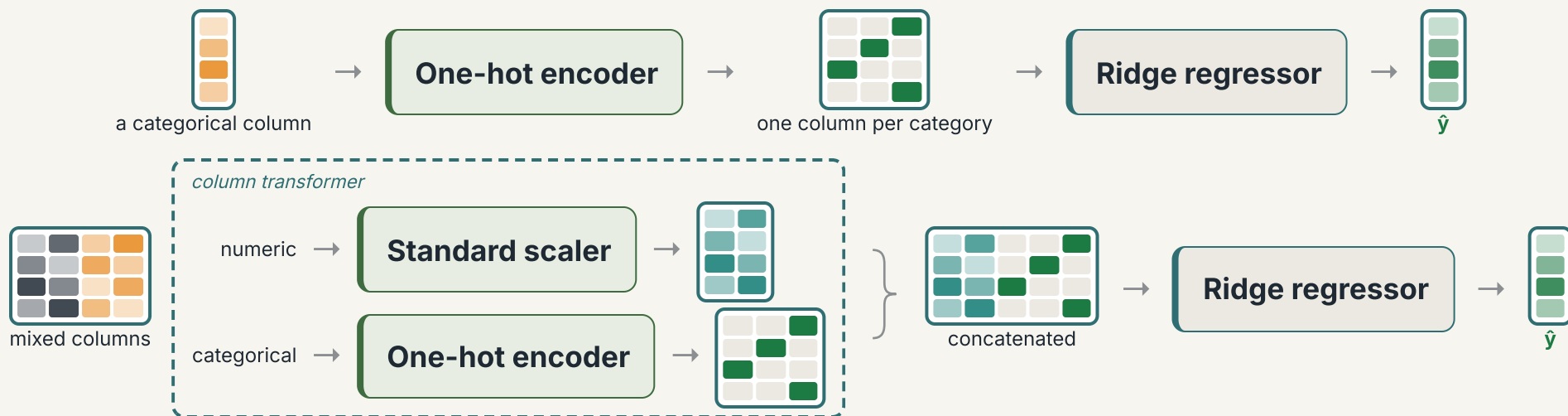
With ridge, must we still drop a category ?

No: $\widehat{\Sigma} + \lambda \mathbf{I}$ is invertible for $\lambda > 0$, even with all the columns.

Should we drop one anyway to further reduce multicollinearity ?

No: for e.g, if we drop **Agdal** the penalty shrinks $\beta_{\text{H.Riad}}$ and $\beta_{\text{Océan}}$ but not **Agdal**, whose effect sits in the **unpenalized** intercept β_0 !

Pipelines: transformers, then an estimator



train `Z_train = column_tr.fit_transform(X_train)`

`ridge.fit(Z_train, y_train)`

test `Z_test = columnun_tr.transform(X_test)`

`y_hat = ridge.predict(Z_test)`

fit **learns** (means, categories, $\hat{\beta}_\lambda$) on the train rows only; **transform** and **predict** **apply** what was learned, on train and test alike.

Hyperparameter selection

Choosing the penalty λ

We want the λ with smallest **validation** error:

the data

Choosing the penalty λ

We want the λ with smallest **validation** error:

train on 80%, **validate** on 20%



$$\hat{\beta}_{\lambda} = \underset{\beta}{\operatorname{argmin}} \frac{1}{n_{\text{tr}}} \sum_{i \in \text{train}} (y_i - \beta^{\top} \mathbf{x}_i)^2 + \lambda \|\beta\|^2$$

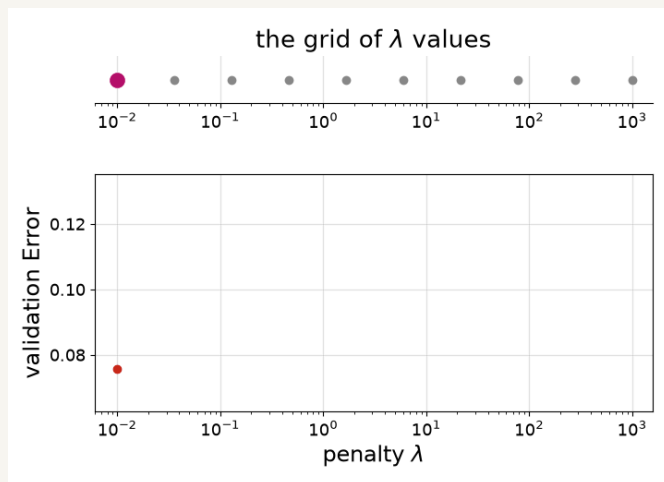
$$e(\lambda) = \frac{1}{n_{\text{val}}} \sum_{i \in \text{val}} (y_i - \hat{\beta}_{\lambda}^{\top} \mathbf{x}_i)^2$$

How do we **minimize** $e(\lambda)$? Gradient descent ?

Minimizing $e(\lambda)$ is not easy. Approximation: try several values for λ and pick the best.

Choosing the penalty λ

$$e(\lambda) = \frac{1}{n_{\text{val}}} \sum_{i \in \text{val}} (y_i - \hat{\beta}_{\lambda}^{\top} \mathbf{x}_i)^2$$



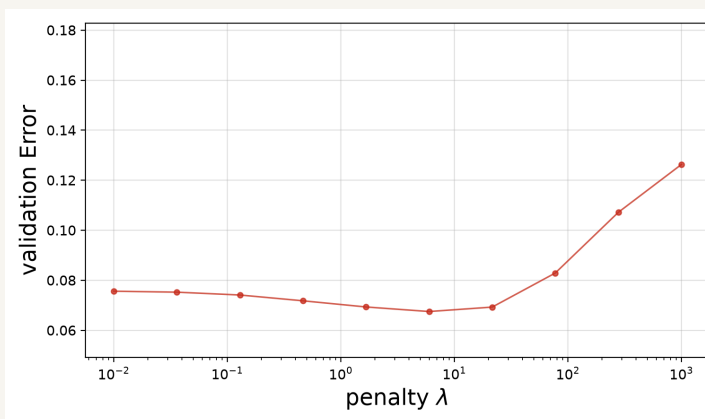
Can we **trust** this choice of λ ?

No ! The chosen λ depends on the **split**.

Cross-validation

Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

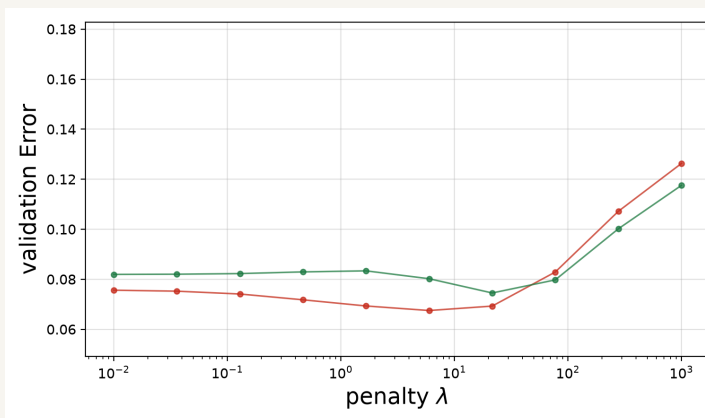
validation train



Cross-validation

Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

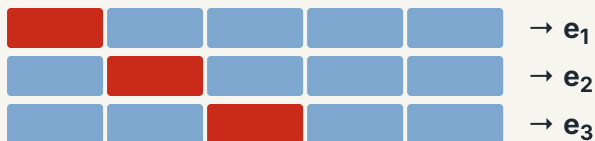
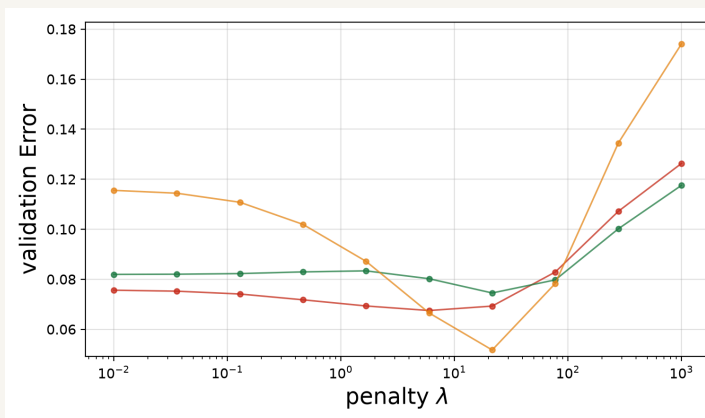
validation train



Cross-validation

Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

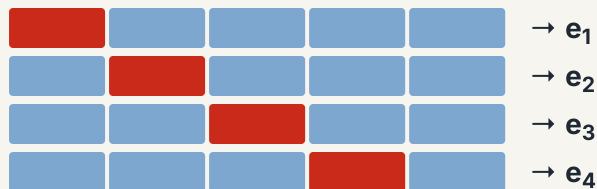
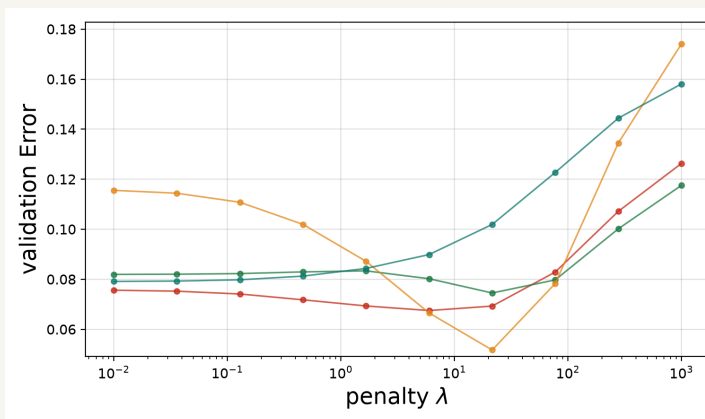
validation train



Cross-validation

Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

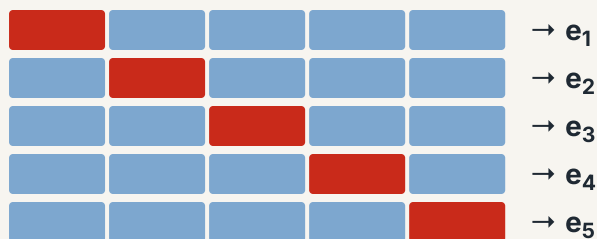
validation train



Cross-validation

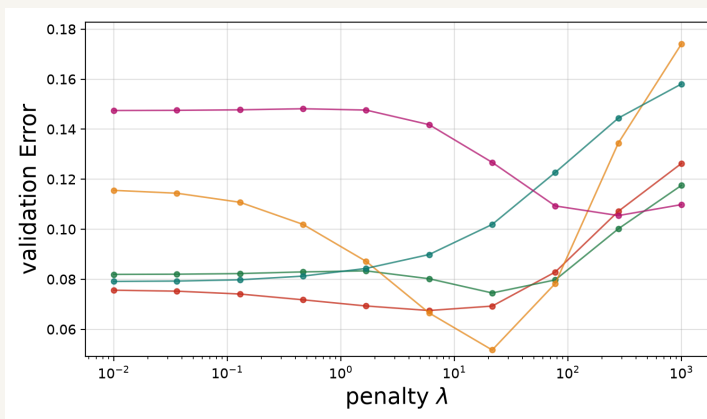
Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

validation train



$$CV(\lambda) = \frac{1}{5}(e_1 + \dots + e_5)$$

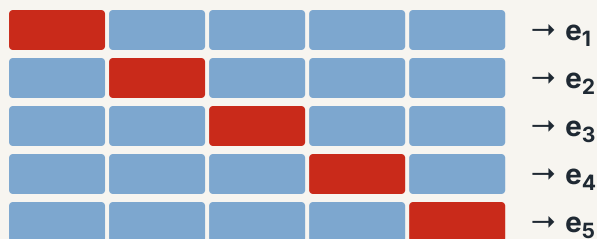
each e_k : $\bar{x}_j, s_j, \hat{\beta}_\lambda$ learned on 4 folds,
applied to the 5th



Cross-validation

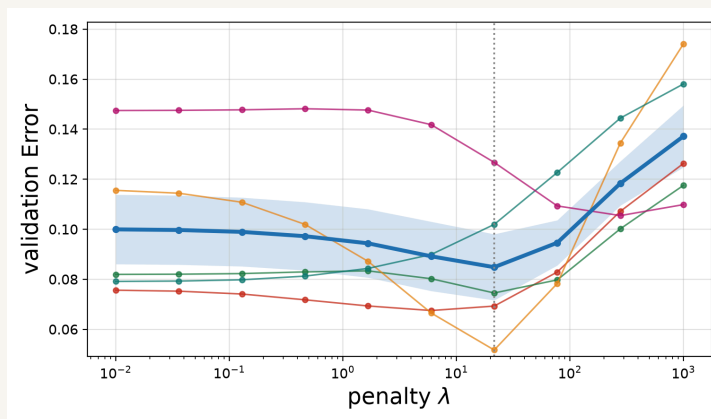
Randomize the splits: $K = 5$ rounds, each fold takes one turn as the validation part; the **whole pipeline** (ϕ , then ridge) is refit on the other four.

validation train



$$CV(\lambda) = \frac{1}{5}(e_1 + \dots + e_5)$$

each e_k : $\bar{x}_j, s_j, \hat{\beta}_\lambda$ learned on 4 folds, applied to the 5th



$$\lambda^* = \operatorname{argmin}_{\lambda \in \text{grid}} CV(\lambda) = 21.5$$



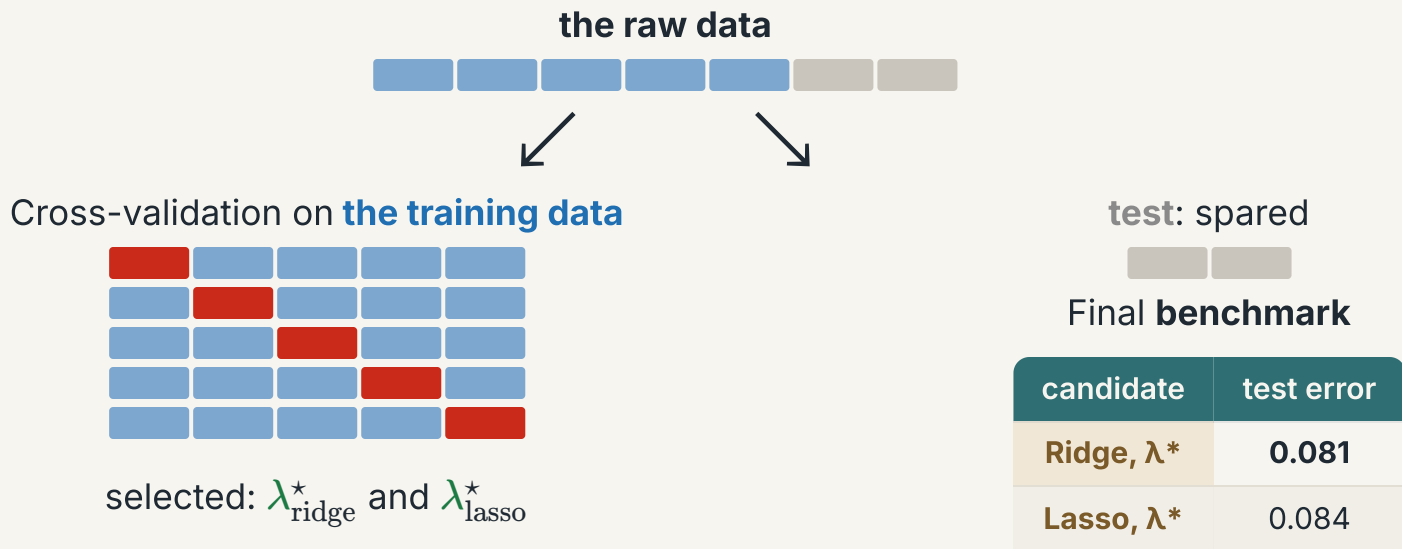
refit on **all** training rows:

$$\hat{\beta}_{\text{final}} = \operatorname{argmin}_{\beta} \frac{1}{n} \|\mathbf{y} - \mathbb{X}\beta\|^2 + \lambda^* \|\beta\|^2$$

The test set stays outside



Important: we never touched the **test set** !



Why keep the test set out of the choice of λ ?

Choosing λ is part of **learning**: it must not touch the final test evaluation.

What to remember

Supervised learning as **penalized empirical risk minimization**:

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n L(y_i, f_{\beta}(\mathbf{x}_i)) + \lambda \Omega(\beta)$$

- **Ridge** ($\|\beta\|_2^2$): every coefficient shrinks, none reaches 0; a unique solution even when $d > n$; shares the effect of correlated columns.
- **Lasso** ($\|\beta\|_1$): a constant pull, coefficients hit **exactly zero**: feature selection; unstable between correlated twins. **Elastic net** mixes both.
- **Preprocessing** is part of the model: standardize (the penalty compares coefficients across units), one-hot with every category under ridge, every statistic learned on **train** and applied unchanged.
- **Hyperparameters**: the training error cannot choose λ ; **cross-validation** on the training data picks it, the **test set** stays untouched.

Contents

1. From statistical modeling to statistical learning
2. Supervised learning
3. Watching a model overfit
4. Regularization: Ridge, Lasso, Elastic-net
5. Preprocessing
6. Hyperparameter selection