SD-TSIA 204: Ridge regression

Ekhine Irurozki
Télécom Paris

Ridge : penalized definition

$$\boxed{\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} = \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\mathbb{R}^p} \left(\underbrace{\|\mathbf{y} - X\boldsymbol{\theta}\|_2^2}_{\textbf{data fitting}} + \underbrace{n\lambda\|\boldsymbol{\theta}\|_2^2}_{\textbf{regularization}} \right)}$$

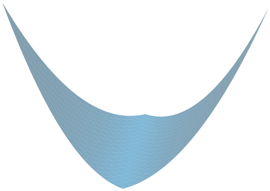
- Note that the *Ridge* estimator is **unique** for any fixed $\lambda > 0$
- We recover the limiting cases:

$$\lim_{\lambda\to 0} \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} = \hat{\boldsymbol{\theta}}^{\mathrm{OLS}} \text{(solution with smallest } \|\cdot\|_2 \text{ norm)}$$

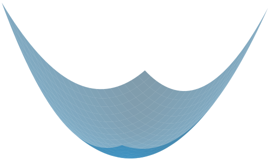
$$\lim_{\lambda\to +\infty} \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} = 0 \in \mathbb{R}^p$$

- First order conditions:
$\nabla f(\boldsymbol{\theta}) = X^\top(X\boldsymbol{\theta} - \mathbf{y}) + n\lambda\boldsymbol{\theta} = 0 \Leftrightarrow (X^\top X + n\lambda \operatorname{Id}_p)\boldsymbol{\theta} = X^\top \mathbf{y}$

Motivation



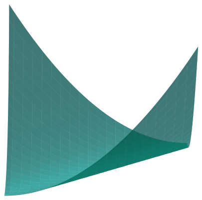
OLS



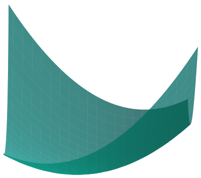
Ridge

x and y-axis are the OLS coefficients β_0 and β_1, z axis is the RSS
Regularize: simplify the problem when ill-conditioned

Motivation



OLS



Ridge

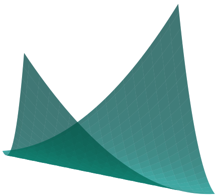
x and y-axis are the OLS coefficients β_0 and β_1, z axis is the RSS
Regularize: simplify the problem when ill-conditioned

Motivation

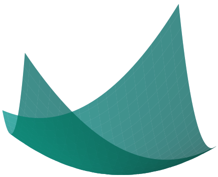
OLS Ridge

x and y-axis are the OLS coefficients β_0 and β_1, z axis is the RSS
Regularize: simplify the problem when ill-conditioned

Motivation



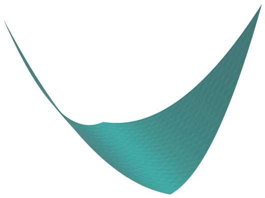
OLS



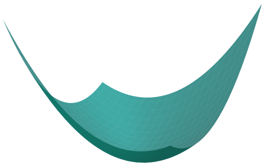
Ridge

x and y-axis are the OLS coefficients β_0 and β_1, z axis is the RSS
Regularize: simplify the problem when ill-conditioned

Motivation



OLS



Ridge

x and y-axis are the OLS coefficients β_0 and β_1, z axis is the RSS
Regularize: simplify the problem when ill-conditioned

Constraint interpretation

A "Lagrangian" formulation is as follows:

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^p}{\arg\min} \left(\underbrace{\|\mathbf{y} - X\boldsymbol{\theta}\|_2^2}_{\textbf{data fitting}} + \underbrace{n\lambda\|\boldsymbol{\theta}\|_2^2}_{\textbf{regularization}} \right)$$

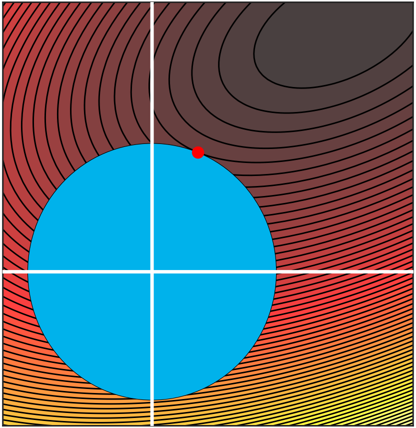
has for a certain $T > 0$ the same solution as:

$$\begin{cases} \underset{\boldsymbol{\theta}\in\mathbb{R}^p}{\arg\min} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 \\ \text{s.t. } \|\boldsymbol{\theta}\|_2^2 \leqslant T \end{cases}$$

Rem: the link $T \leftrightarrow \lambda$ is not explicit!

- If $T \to 0$ we recover the null vector: $0 \in \mathbb{R}^p$
- If $T \to \infty$ we recover $\hat{\boldsymbol{\theta}}^{\mathrm{OLS}}$ (un-constrained)

Level lines and and constraints set



Associated prediction

From the *Ridge* coefficient:

$$\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} = (n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \mathbf{y}$$

the associated prediction is given by:

$$\hat{\mathbf{y}} = X\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} = X(n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \mathbf{y} = H_\lambda \mathbf{y}$$

Rem: the estimator $\hat{\mathbf{y}}$ is linear w.r.t. $\mathbf{y}$

Rem: reminding $X = \sum_{i=1}^{\mathrm{rg}(X)} s_i \mathbf{u}_i \mathbf{v}_i^\top$, (SVD) the matrix $H_\lambda := X(n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top = \sum_{j=1}^{\mathrm{rg}(X)} \frac{s_j^2}{s_j^2 + n\lambda} \mathbf{u}_j \mathbf{u}_j^\top$ is the equivalent of the **hat matrix**

If $\lambda \neq 0$, we do not have $H_\lambda^2 = H_\lambda = \sum_{j=1}^{\mathrm{rg}(X)} \mathbf{u}_j \mathbf{u}_j^\top$ anymore, so H_λ is not a projection (in general).

N remarks

Reminder: normalizing the p features the same way is necessary if you want the penalty to be similar for all features:

- center the observation and the features $\Rightarrow$ no coefficient for the constants (hence no constraint on it)
- not centering features $\Rightarrow$ do not put constraint on the constant feature (bias/intercept)

$$\hat{\boldsymbol{\theta}}_{\lambda}^{\mathrm{rdg}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^p}{\arg\min} \|\mathbf{y} - X\boldsymbol{\theta} - \theta_0 \mathbf{1}_n\|_2^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

Rem:for cross validation one can use $\frac{\|\mathbf{y}-X\boldsymbol{\theta}\|_2^2}{2n}$ rather than $\frac{\|\mathbf{y}-X\boldsymbol{\theta}\|_2^2}{2}$ as the data fitting part

General form of the bias

Under the fixed-design model, $\mathbf{y} = X\boldsymbol{\theta}^\star + \boldsymbol{\varepsilon}$ with $\mathbb{E}(\boldsymbol{\varepsilon}) = 0$:

$$
\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) =& \mathbb{E}[(n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \mathbf{y}] \\
=& \mathbb{E}[(n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top X \boldsymbol{\theta}^\star + (n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \boldsymbol{\varepsilon}] \\
=& (n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top X \boldsymbol{\theta}^\star \\
=& \sum_{i=1}^{\mathrm{rg}(X)} \frac{s_i^2}{s_i^2 + n\lambda} \mathbf{v}_i \mathbf{v}_i^\top \boldsymbol{\theta}^\star
\end{aligned}
$$

<u>Rem</u>: one recovers $\mathbb{E}(\hat{\boldsymbol{\theta}}^{\mathrm{OLS}}) \to \sum_{i=1}^{\mathrm{rg}(X)} \mathbf{v}_i \mathbf{v}_i^\top \boldsymbol{\theta}^\star$ when $\lambda \to 0$

<u>Rem</u>: the bias is $\mathbb{E}(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) - \boldsymbol{\theta}^\star = -n\lambda (X^\top X + n\lambda \operatorname{Id}_p)^{-1} \boldsymbol{\theta}^\star$

Variance in the general case

Under the assumption $\mathbb{E}(\boldsymbol{\varepsilon}) = 0$, and with a homoscedastic model: $\mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top) = \sigma^2 \operatorname{Id}_n$

Variance / Covariance

$$V_\lambda^{\mathrm{rdg}} = \mathbb{E}\left((\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \mathbb{E}(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}))(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \mathbb{E}(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}})^\top\right)$$

Explicit computation:

$$\begin{aligned}
V_\lambda^{\mathrm{rdg}} =& \mathbb{E}((n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top X (n\lambda \operatorname{Id}_p + X^\top X)^{-1}) \\
&= (n\lambda \operatorname{Id}_p + X^\top X)^{-1} X^\top \mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top) X (n\lambda \operatorname{Id}_p + X^\top X)^{-1} \\
&= \sigma^2 (n\lambda \operatorname{Id}_p + X^\top X)^{-2} X^\top X \qquad \text{(matrix commute here)} \\
&= \sum_{i=1}^{\mathrm{rg}(X)} \frac{s_i^2 \sigma^2}{(s_i^2 + n\lambda)^2} \mathbf{v}_i \mathbf{v}_i^\top
\end{aligned}$$

<u>Rem</u>: one recovers $V^{\mathrm{OLS}} = \sum_{i=1}^{\mathrm{rg}(X)} \frac{\sigma^2}{s_i^2} \mathbf{v}_i \mathbf{v}_i^\top$ when $\lambda \to 0$

<u>Rem</u>: one find a null variance when $\lambda \to \infty$

Prediction risk

Homoscedastic assumption: $\mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top) = \sigma^2 \, \mathrm{Id}_n$

Quadratic prediction risk $\mathbb{E}\|X\boldsymbol{\theta}^\star - X\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}\|^2$

Under the Homoscedastic assumption:

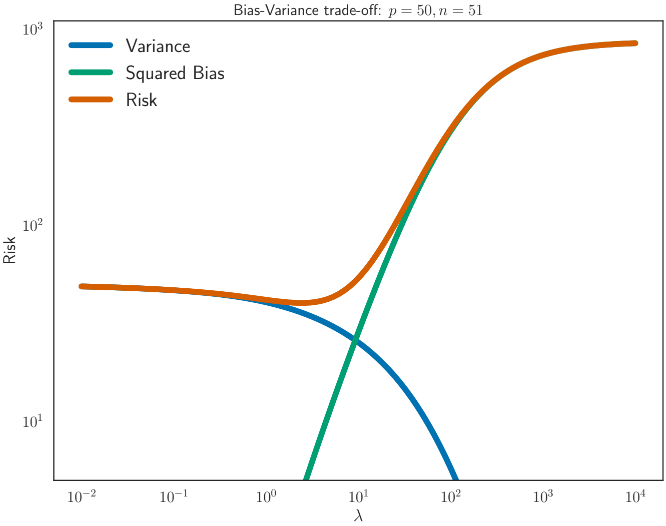
$$R_{\mathrm{pred}}(\boldsymbol{\theta}^\star, \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) = \mathbb{E}\left[(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \boldsymbol{\theta}^\star)^\top (X^\top X)(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \boldsymbol{\theta}^\star)\right]$$

Explicit computation (begins as for OLS):

$$\begin{aligned}
R_{\mathrm{pred}}(\boldsymbol{\theta}^\star, \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) &= \mathbb{E}\left[(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \boldsymbol{\theta}^\star)^\top (X^\top X)(\hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}} - \boldsymbol{\theta}^\star)\right] \\
&= \mathbb{E}\left[(X(X^\top X + n\lambda \, \mathrm{Id}_p)^{-1} X^\top \boldsymbol{\varepsilon})^\top (X(X^\top X + \lambda \mathrm{Id}_p)^{-1} X^\top \boldsymbol{\varepsilon})\right] \\
&\quad + \lambda^2 \boldsymbol{\theta}^{\star\top} (X^\top X + n\lambda \, \mathrm{Id}_p)^{-2} \boldsymbol{\theta}^\star \\
&= \sum_{i=1}^{\mathrm{rg}(X)} \frac{s_i^4 \sigma^2}{(s_i^2 + n\lambda)^2} + n^2 \lambda^2 \boldsymbol{\theta}^{\star\top} (X^\top X + \lambda \, \mathrm{Id}_p)^{-2} \boldsymbol{\theta}^\star
\end{aligned}$$

Rem: $\lim\limits_{\lambda \to 0} R_{\mathrm{pred}}(\boldsymbol{\theta}^\star, \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) = \mathrm{rg}(X)\sigma^2$, $\lim\limits_{\lambda \to \infty} R_{\mathrm{pred}}(\boldsymbol{\theta}^\star, \hat{\boldsymbol{\theta}}_\lambda^{\mathrm{rdg}}) = \|X\boldsymbol{\theta}^\star\|_2^2$

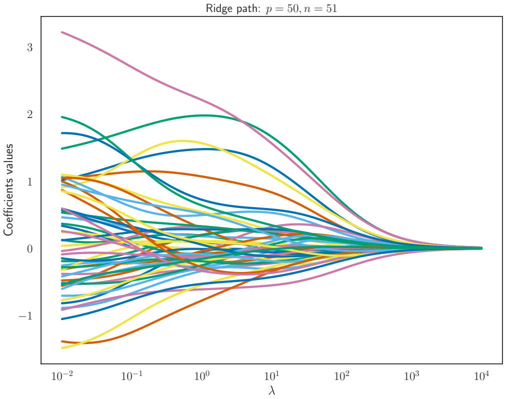
Bias / Variance: simulated example



$$X \in \mathbb{R}^{51 \times 50}, \boldsymbol{\theta}^\star = (2, 2, 2, 2, 2, 0, \ldots, 0)^\top$$

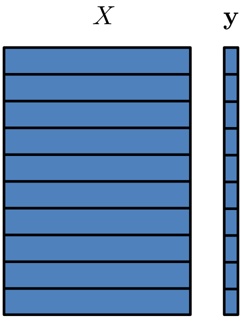
Choosing λ

```
n_features = 50; n_samples = 50
X = np.random.randn(n_samples, n_features)
theta_true = np.zeros([n_features, ])
theta_true[0:5] = 2.
y_true = np.dot(X, theta_true)
y = y_true + 1. * np.random.rand(n_samples,)
```



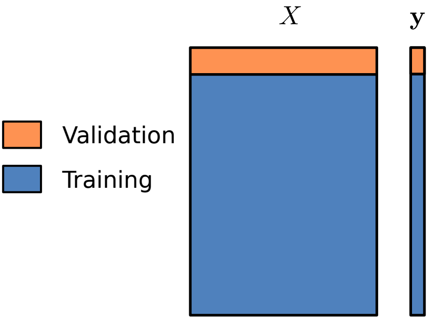
K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

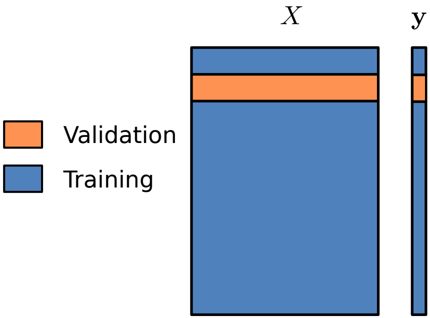


$k = 1$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

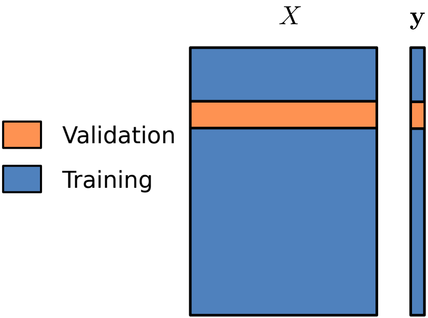


$k = 2$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

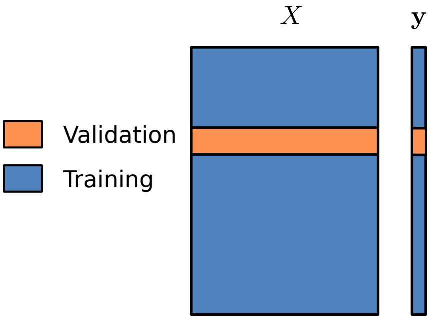


$k = 3$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \dots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

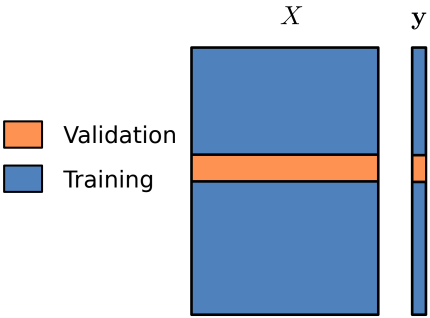


$k = 4$

1. Compute with the training part the estimators for $\lambda_1, \dots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \dots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \dots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

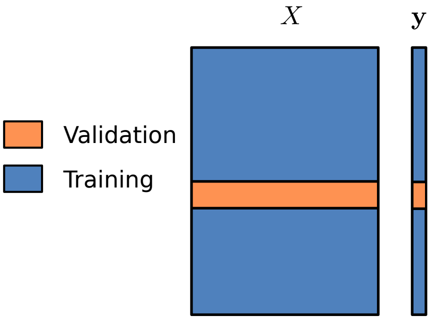


$k = 5$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

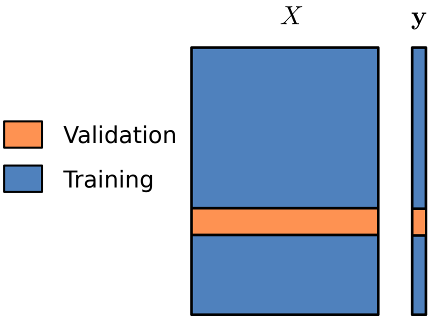


$k = 6$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

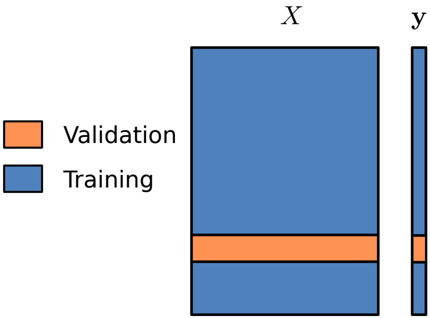


$k = 7$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \dots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

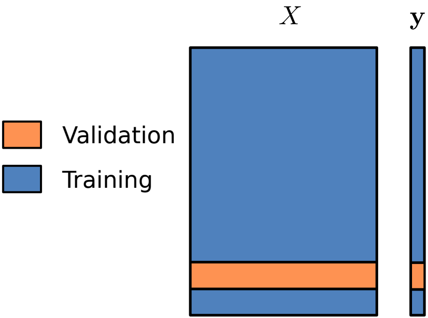


$k = 8$

1. Compute with the training part the estimators for $\lambda_1, \dots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \dots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \dots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

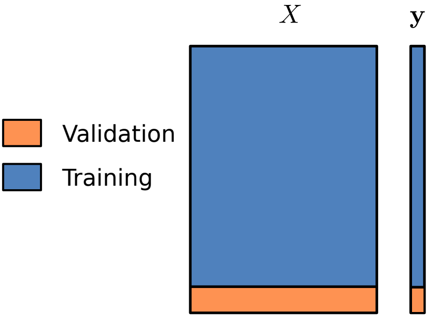


$k = 9$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):

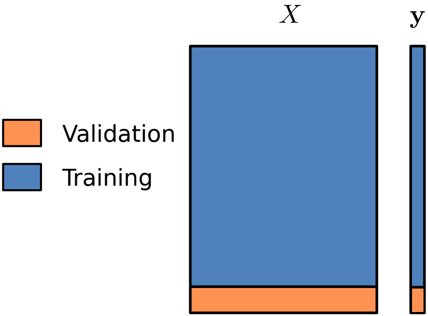


$k = 10$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \ldots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



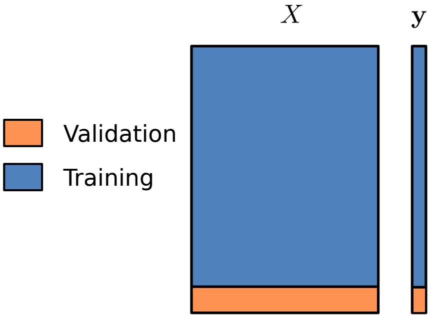
$k = 10$

1. Compute with the training part the estimators for $\lambda_1, \ldots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \ldots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \ldots, \mathrm{Error}_r^k$ over the validation part,

<u>**Parameter choice**</u>: averaging the previous errors over k gives $\widehat{\mathrm{Error}}_1, \ldots, \widehat{\mathrm{Error}}_r$. Then choose $i^* \in [\![1, r]\!]$ achieving the smallest one

K-fold Cross-Validation ($K = 10$)

- Choose a grid of r λ's to test: $\lambda_1, \dots, \lambda_r$
- Divide $(X, \mathbf{y})$ into K blocks (sample-wise):



$k = 10$

1. Compute with the training part the estimators for $\lambda_1, \dots, \lambda_r$: $\hat{\boldsymbol{\theta}}^{\lambda_1}, \dots, \hat{\boldsymbol{\theta}}^{\lambda_r}$
2. Evaluate the (prediction) error $\mathrm{Error}_1^k, \dots, \mathrm{Error}_r^k$ over the validation part,

Parameter choice: averaging the previous errors over k gives $\widehat{\mathrm{Error}}_1, \dots, \widehat{\mathrm{Error}}_r$. Then choose $i^* \in [\![1, r]\!]$ achieving the smallest one

Re-calibration: compute $\hat{\boldsymbol{\theta}}^{\lambda_{i^*}}$ over the whole sample

CV in practice

Extreme cases of CV

- $K = 1$ impossible, needs $K = 2$
- $K = n$, "*leave-one-out*" strategy (*cf.* **Jackknife**): as many blocks as observations
  Rem: $K = n$ (often) computationally efficient but unstable

Practical advice:

- "randomise the sample" : having samples in random order avoid artifacts block (each fold needs to be representative of the whole sample!)
- standard choices: $K = 5, 10$

Alternatives: random partition validation/test, time series variants, etc. http://scikit-learn.org/stable/modules/cross_validation.html

CV variants `sklearn`

Crucial points: the structures train/test artificially created should represent faithfully the underlying learning problem

Classical alternatives:

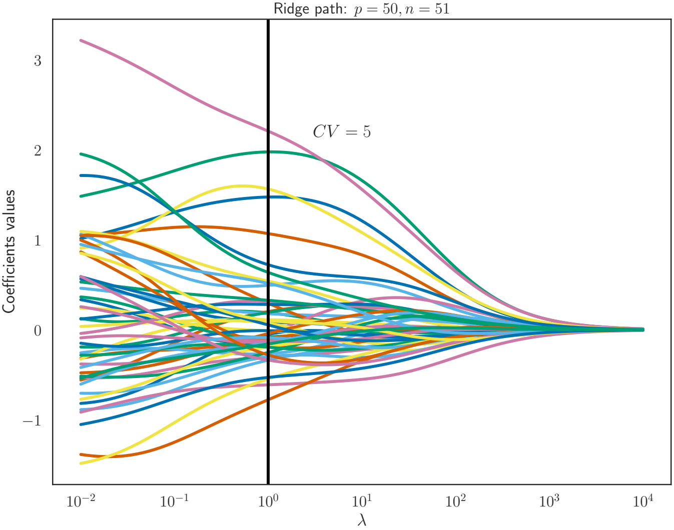
- random partitioning in train/test sets (*cf.*`train_test_split`)
- Time series variant: `TimeSeriesSplit` (never predict the past with future information)
- For classification tasks with unbalanced classes `StratifiedKFold`

Rem: averaging estimators (with weights reflecting their performance) is also relevant for prediction

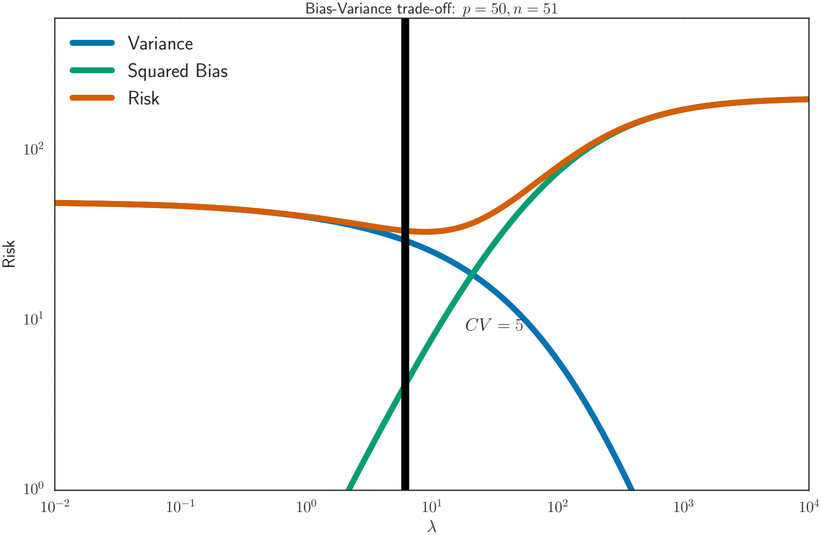
More details:

http://scikit-learn.org/stable/modules/cross_validation.html

Choosing λ : example with $CV = 5$ (I)



Choosing λ : example with $CV = 5$ (II)



Algorithms to compute the *Ridge* estimator

- `'svd'`: most stable method, useful for computing many λ's cause the SVD price is paid only once
- `'cholesky'` : matrix decomposition leading to a close form solution `scipy.linalg.solve`
- `'sparse_cg'`: conjugate gradient descent, useful also for sparse cases and high dimension (set `tol`/`max_iter` to a small value)
- stochastic gradient descent approaches : if n is huge

*cf.*the code of `Ridge`, `ridge_path`, `RidgeCV` in the module `linear_model` of `sklearn`

Rem: it is rare to compute the *Ridge* estimator only for one single λ

Rem: crucial issue of computing SVD for huge matrices...