

Ridge

Nicolas Verzelen, Joseph Salmon

INRA / Université de Montpellier



Plan

Définitions de l'estimateur Ridge

- Point de vue par SVD

- Point de vue par pénalisation

Choix du paramètre de régularisation

Algorithmes et aspects computationnels

Rappel

$$\mathbf{y} = X\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}$$

- ▶ $\mathbf{y} \in \mathbb{R}^n$ est le vecteur des observations
- ▶ $X \in \mathbb{R}^{n \times p}$ est la matrice des variables explicatives
- ▶ $\boldsymbol{\beta}^* \in \mathbb{R}^p$ est le **vrai** paramètre du modèle que l'on veut retrouver.
- ▶ $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ est le bruit

Rem: possiblement une variable supplémentaire pour la constante

La décomposition en valeur singulières

Théorème : Golub et Van Loan (2013)

Pour toute matrice $X \in \mathbb{R}^{n \times p}$, il existe deux matrices orthogonales $U = [\mathbf{u}_1, \dots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$ et $V = [\mathbf{v}_1, \dots, \mathbf{v}_p] \in \mathbb{R}^{p \times p}$, telles que

$$U^\top X V = \text{diag}(s_1, \dots, s_{\text{rg}(X)}) = \Sigma \in \mathbb{R}^{n \times p}$$

avec $s_1 \geq s_2 \geq \dots \geq s_{\text{rg}(X)} > 0$, avec $\text{rg}(X) = \text{rang}(X)$.

$$X = U \Sigma V^\top \Leftrightarrow X = \sum_{i=1}^{\text{rg}(X)} s_i \mathbf{u}_i \mathbf{v}_i^\top$$

Une solution des moindres carrés est alors :

$$\hat{\beta}^{\text{MCO}} = X^+ \mathbf{y} = \sum_{i=1}^{\text{rg}(X)} \frac{1}{s_i} \mathbf{v}_i \mathbf{u}_i^\top \mathbf{y}$$

Retour sur les problèmes numériques

$$\hat{\beta}^{\text{MCO}} = X^+ \mathbf{y} = \sum_{i=1}^{\text{rg}(X)} \frac{1}{s_i} \mathbf{v}_i \mathbf{u}_i^\top \mathbf{y}$$

Si les plus petites valeurs singulières s_i s'approchent de zéro alors la solution numérique de la SVD n'est pas stable !

Rem: le défaut n'est pas propre aux moindres carrés, mais inhérent aux problèmes difficiles (on dit aussi “mal posé” en analyse numérique et en traitement du signal)

Les équations normales

Une solution β des moindres carrés doit vérifier :

$$X^{\top} X \beta = X^{\top} \mathbf{y} \Leftrightarrow V \Sigma^{\top} \Sigma V^{\top} \beta = V \Sigma^{\top} U^{\top} \mathbf{y}$$

et si l'on cherche β sous la forme $\beta = V \beta$, c'est équivalent à

$$\Sigma^{\top} \Sigma \beta = \Sigma^{\top} U^{\top} \mathbf{y}$$

$\Sigma^{\top} \Sigma$ diagonale avec $r = \text{rang}(X)$ éléments non nuls qui sont les s_i^2

$$\Sigma^{\top} \Sigma = \left[\begin{array}{cc|c} s_1^2 & & 0 \\ & \ddots & \\ 0 & & s_r^2 \\ \hline & 0 & 0 \end{array} \right] \in \mathbb{R}^{p \times p}$$

Les équations normales (suite)

Alternative régularisée : résoudre les équations normales

$$\left[\begin{array}{ccc|c} s_1^2 & & 0 & 0 \\ & \ddots & & \\ 0 & & s_r^2 & 0 \\ \hline & 0 & & 0 \end{array} \right] \text{ remplacé par } \left[\begin{array}{ccc|c} s_1^2 & & 0 & 0 \\ & \ddots & & \\ 0 & & s_r^2 & 0 \\ \hline & 0 & & 0 \end{array} \right] + \lambda \text{Id}_p$$

De manière synthétique cela s'écrit : $(\lambda \text{Id}_p + \Sigma^\top \Sigma) \beta = \Sigma^\top U^\top \mathbf{y}$

i.e., on ajoute à toutes les valeurs propres de $X^\top X$ un terme $\lambda > 0$ "petit", λ est nommé **paramètre de régularisation**

$$\beta = (\lambda \text{Id}_p + \Sigma^\top \Sigma)^{-1} \Sigma^\top U^\top \mathbf{y}$$

et donc

$$\beta = V(\lambda \text{Id}_p + \Sigma^\top \Sigma)^{-1} \Sigma^\top U^\top \mathbf{y}$$

Ridge forme explicite

Avec la SVD, l'équation suivante se simplifie :

$$\beta = V(\lambda \text{Id}_p + \Sigma^\top \Sigma)^{-1} \Sigma^\top U^\top \mathbf{y}$$

Cela donne une première forme de l'estimateur *Ridge*

$$\hat{\beta}_\lambda^{\text{rdg}} = (\lambda \text{Id}_p + X^\top X)^{-1} X^\top \mathbf{y}$$


Rappel : sous l'hypothèse de plein rang $\hat{\beta}^{\text{MCO}} = (X^\top X)^{-1} X^\top \mathbf{y}$

Rem:

$$\lim_{\lambda \rightarrow 0^+} \hat{\beta}_\lambda^{\text{rdg}} = \hat{\beta}^{\text{MCO}}$$

$$\lim_{\lambda \rightarrow +\infty} \hat{\beta}_\lambda^{\text{rdg}} = 0 \in \mathbb{R}^p$$

Astuce du noyau

Astuce du noyau ( : *Kernel trick*) : Selon si $n > p$ ou $n \leq p$, une méthode qui cherche à trouver une solution de Ridge par inversion peut préférer l'une des deux formulations suivantes :

$$X^{\top}(XX^{\top} + \lambda \text{Id}_n)^{-1}\mathbf{y} = (X^{\top}X + \lambda \text{Id}_p)^{-1}X^{\top}\mathbf{y}$$

- ▶ membre de gauche : on résout un système $n \times n$
- ▶ membre de droite : on résout un système $p \times p$

Rem: cette propriété est aussi très utile pour les méthodes à noyaux de type SVM (cf. cours de *Machine Learning*)

Exercise: Démontrer la propriété précédente avec la SVD

Ridge / Tikhonov : la définition pénalisée

$$\hat{\beta}_{\lambda}^{\text{rdg}} = \arg \min_{\beta \in \mathbb{R}^p} \left(\underbrace{\|\mathbf{y} - X\beta\|_2^2}_{\text{attache aux données}} + \underbrace{\lambda \|\beta\|_2^2}_{\text{régularisation}} \right)$$

- Noter que l'estimateur *Ridge* est **unique** pour un λ fixé
- On retrouve de nouveau les cas limites :

$$\lim_{\lambda \rightarrow 0} \hat{\beta}_{\lambda}^{\text{rdg}} = \hat{\beta}^{\text{MCO}} \text{ (solution de norme } \|\cdot\|_2 \text{ minimale)}$$

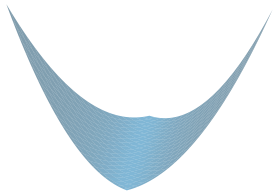
$$\lim_{\lambda \rightarrow +\infty} \hat{\beta}_{\lambda}^{\text{rdg}} = 0 \in \mathbb{R}^p$$

- Lien des deux formulations par les CNO : pour

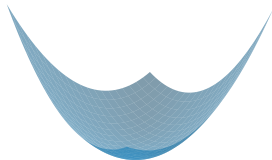
$$f(\beta) = \frac{\|\mathbf{y} - X\beta\|_2^2}{2} + \frac{\lambda \|\beta\|_2^2}{2}$$

$$\nabla f(\beta) = X^{\top}(X\beta - \mathbf{y}) + \lambda\beta = 0 \Leftrightarrow (X^{\top}X + \lambda \text{Id}_p)\beta = X^{\top}\mathbf{y}$$

Motivation



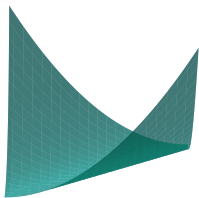
OLS



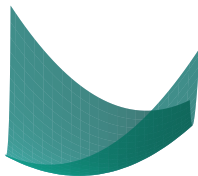
Ridge

Régulariser : simplifie le problème quand il est mal conditionné

Motivation



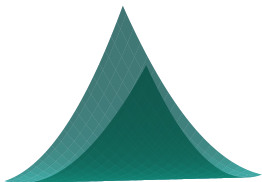
OLS



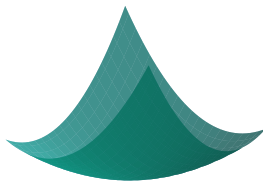
Ridge

Régulariser : simplifie le problème quand il est mal conditionné

Motivation



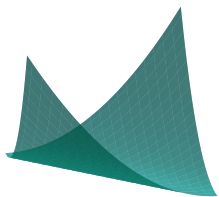
OLS



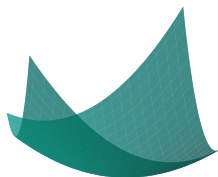
Ridge

Régulariser : simplifie le problème quand il est mal conditionné

Motivation



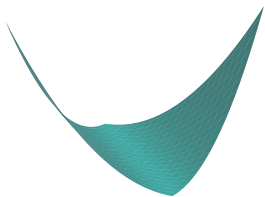
OLS



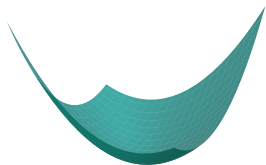
Ridge

Régulariser : simplifie le problème quand il est mal conditionné

Motivation



OLS



Ridge

Régulariser : simplifie le problème quand il est mal conditionné

Interprétation contrainte

Un problème de la forme “Lagrangienne” suivante :

$$\arg \min_{\beta \in \mathbb{R}^p} \left(\underbrace{\frac{1}{2} \|\mathbf{y} - X\beta\|_2^2}_{\text{attache aux données}} + \underbrace{\frac{\lambda}{2} \|\beta\|_2^2}_{\text{régularisation}} \right)$$

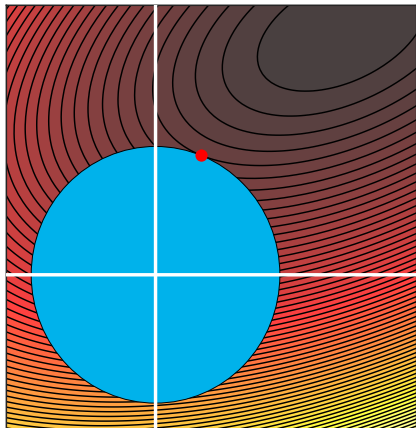
admet pour un certain $T > 0$ la même solution que :

$$\begin{cases} \arg \min_{\beta \in \mathbb{R}^p} \|\mathbf{y} - X\beta\|_2^2 \\ \text{t.q. } \|\beta\|_2^2 \leq T \end{cases}$$

Rem: le lien $T \leftrightarrow \lambda$ n'est pas explicite !

- ▶ Si $T \rightarrow 0$ on retrouve le vecteur nul : $0 \in \mathbb{R}^p$
- ▶ Si $T \rightarrow \infty$ on retrouve $\hat{\beta}^{\text{MCO}}$ (non contraint)

Lignes de niveau et ensemble de contraintes



Optimisation sous contraintes ℓ_2

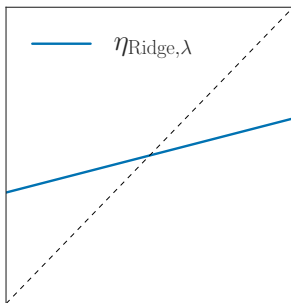
Le cas orthogonal

Retour sur un cas simple $X^\top X = \text{Id}_p$

$$\hat{\beta}_\lambda^{\text{rdg}} = (\lambda \text{Id}_p + X^\top X)^{-1} X^\top \mathbf{y}$$

$$\hat{\beta}_\lambda^{\text{rdg}} = (\lambda \text{Id}_p + \text{Id}_p)^{-1} X^\top \mathbf{y} = \frac{1}{\lambda + 1} X^\top \mathbf{y}$$

$$\hat{\mathbf{y}} = \frac{1}{\lambda + 1} \mathbf{y} = (\eta_{\text{rdg},\lambda}(\mathbf{y}_i))_{i=1,\dots,n}$$



Rem: cas classique en traitement du signal (peu fréquent en statistique)

Rem: la fonction réelle $\eta_{\text{rdg},\lambda}$ est une contraction linéaire (shrinkage)

Prédiction associée


Partant du coefficient *Ridge* :

$$\hat{\beta}_{\lambda}^{\text{rdg}} = (\lambda \text{Id}_p + X^{\top} X)^{-1} X^{\top} \mathbf{y}$$

la prédiction associée s'obtient ainsi :

$$\hat{\mathbf{y}} = X \hat{\beta}_{\lambda}^{\text{rdg}} = X(\lambda \text{Id}_p + X^{\top} X)^{-1} X^{\top} \mathbf{y}$$

Rem: l'estimateur $\hat{\mathbf{y}}$ est toujours linéaire en \mathbf{y}

Rem: l'équivalent de la matrice chapeau ( : *hat matrix*) est


$$H_{\lambda} = X(\lambda \text{Id}_p + X^{\top} X)^{-1} X^{\top} = \sum_{j=1}^{\text{rg}(X)} \frac{s_j^2}{s_j^2 + \lambda} \mathbf{u}_j \mathbf{u}_j^{\top}$$

Attention : si $\lambda \neq 0$, on n'a plus $H_{\lambda}^2 = H_{\lambda} = \sum_{j=1}^{\text{rg}(X)} \mathbf{u}_j \mathbf{u}_j^{\top}$, i.e., H_{λ}

n'est donc pas un projecteur

Point normalisation et centrage

Rappel : normaliser les p variables de la même manière pour que la pénalisation contraigne de manière similaire toutes les variables

- ▶ centrer l'observation et les variables explicatives \Rightarrow pas de coefficient pour la variable constante (donc pas de contrainte)
- ▶ ne pas centrer les variables explicatives \Rightarrow ne pas mettre de contrainte sur la variable constante ( : *bias/intercept*),

$$\hat{\beta}_{\lambda}^{\text{rdg}} = \arg \min_{\beta \in \mathbb{R}^p} \|\mathbf{y} - X\beta - \beta_0 \mathbf{1}_n\|^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Alternative (si l'on n'a pas normalisé) : changer la pénalité en

$$\arg \min_{\beta \in \mathbb{R}^p} \|\mathbf{y} - X\beta\|^2 + \lambda \sum_{j=1}^p \alpha_j \beta_j^2 \quad (\text{e.g., } \alpha_j = \|\mathbf{x}_j\|^2)$$

Rem: pour la validation croisée on utilisera plus naturellement $\frac{\|\mathbf{y} - X\beta\|^2}{2n}$ que $\frac{\|\mathbf{y} - X\beta\|^2}{2}$ pour conserver l'amplitude de λ

Point normalisation et centrage (bis)

Ici $X = [\frac{1_{a_1}}{\sqrt{n_1}}, \dots, \frac{1_{a_K}}{\sqrt{n_K}}]$ ($\mathbf{x}_k = \frac{1_{a_j}}{\sqrt{n_k}}, i.e., \text{on } \cdot$), avec

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{x_i=a_k} y_i \text{ et } n_k = \#\{x_i = a_k\}$$

L'estimateur Ridge (sans pénalité sur la constante) est solution de

$$(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_K)^\top = (\hat{\beta}_0, \tilde{\boldsymbol{\beta}})^\top \in \arg \min_{\beta_0, \dots, \beta_p} f(\beta_0, \dots, \beta_p)$$

$$\text{avec } f(\beta_0, \dots, \beta_K) = \left\| \mathbf{y} - \beta_0 \mathbf{1}_n - \sum_{j=1}^K \beta_j \mathbf{x}_j \right\|^2 + \lambda \sum_{j=1}^K \beta_j^2$$

Partant de $X^\top X = \text{Id}_K$ et $X^\top \mathbf{y} = (\sqrt{n_1} \hat{\mu}_1, \dots, \sqrt{n_K} \hat{\mu}_K)$, on a :

$$\tilde{\boldsymbol{\beta}} = (X^\top X + \lambda \text{Id}_K)^{-1} X^\top (\mathbf{y} - \hat{\beta}_0 \mathbf{1}_n) = \frac{1}{1 + \lambda} \begin{pmatrix} \sqrt{n_1}(\hat{\mu}_1 - \hat{\beta}_0) \\ \vdots \\ \sqrt{n_K}(\hat{\mu}_K - \hat{\beta}_0) \end{pmatrix}$$

Suite

$$\text{CNO de Ridge : } \begin{cases} \hat{\beta}_0 = \frac{1}{n} \langle \mathbf{1}_n, \mathbf{y} - X\tilde{\beta} \rangle & (1) \\ \tilde{\beta} = \frac{1}{\lambda} X^\top (\mathbf{y} - X\tilde{\beta} - \mathbf{1}_n \hat{\beta}_0) & (2) \end{cases}$$

Avec $e = (\sqrt{n_1}, \dots, \sqrt{n_K})$, alors $Xe = \mathbf{1}_n$ et $\mathbf{1}_n^\top X = e^\top$. Avec (2)

$$\langle e, \tilde{\beta} \rangle = \frac{1}{1+\lambda} e^\top X^\top (\mathbf{y} - \mathbf{1}_n \hat{\beta}_0 - X\tilde{\beta}) = \frac{1}{1+\lambda} \mathbf{1}_n^\top (\mathbf{y} - X\tilde{\beta} - \mathbf{1}_n \hat{\beta}_0)$$
$$\langle e, \tilde{\beta} \rangle = \frac{1}{1+\lambda} (n\hat{\beta}_0 - n\hat{\beta}_0) = 0$$

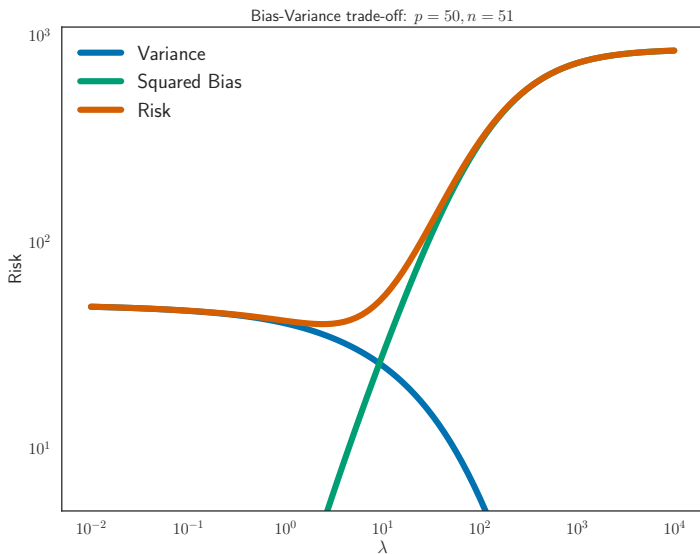
Avec (1) on déduit que : $\hat{\beta}_0 = \bar{y}_n$ puis que pour une nouvelle observation $x_{n+1} = a_k$ on a :

$$y_{n+1} = \frac{1}{1+\lambda} (\lambda \bar{y}_n + \hat{\mu}_k)$$

Rem: λ permet d'osciller entre le prédicateur globale (\bar{y}_n , si $\lambda = +\infty$) et le prédicateur par modalité ($\hat{\mu}_k$, si $\lambda = 0$)

Rem: si l'on ne prend pas en compte la constante, $y_{n+1} = \frac{1}{1+\lambda} \hat{\mu}_k$, et donc pour λ grand on prédit 0!!!

Biais / Variance : exemple de simulation



$$X \in \mathbb{R}^{50 \times 51}, \beta^* = (2, 2, 2, 2, 2, 0, \dots, 0)^\top$$

Plan

Définitions de l'estimateur Ridge

Choix du paramètre de régularisation

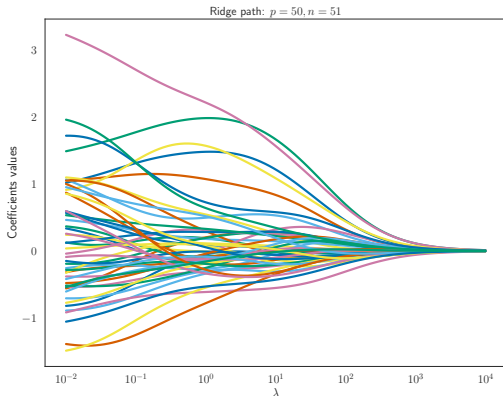
Notion de chemin de régularisation

Validation Croisée (CV)

Algorithmes et aspects computationnels

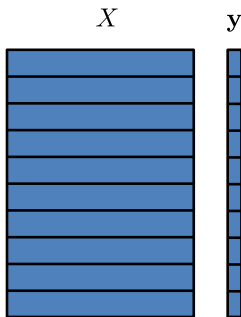
Choix de λ

```
n_features = 50; n_samples = 51
X = np.random.randn(n_samples, n_features)
beta_true = np.zeros([n_features, ])
beta_true[0:5] = 2.
y_true = np.dot(X, beta_true)
y = y_true + 1. * np.random.rand(n_samples,)
```



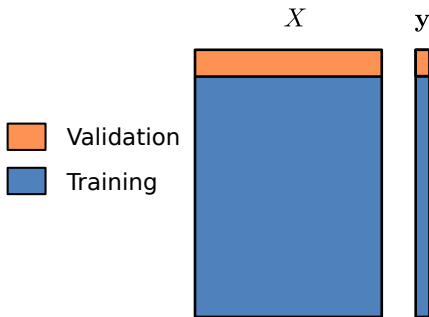
Validation croisée K -fold ($K = 10$)

- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



Validation croisée K -fold ($K = 10$)

- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :

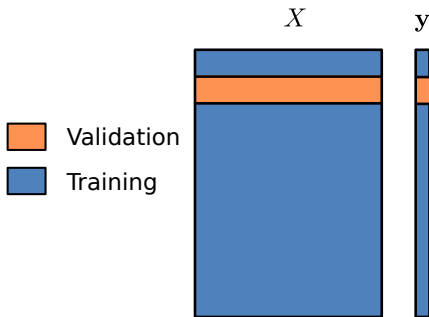


$k = 1$

1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

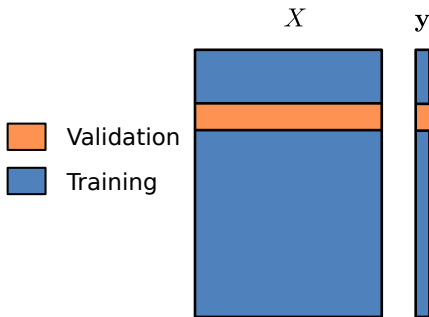
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



- $k = 2$
1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
 2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

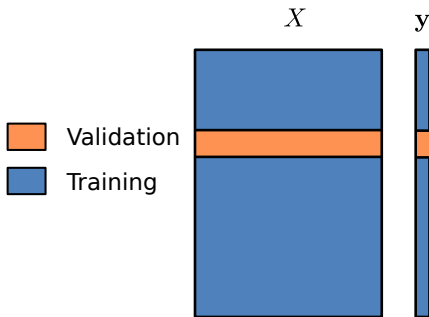
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :

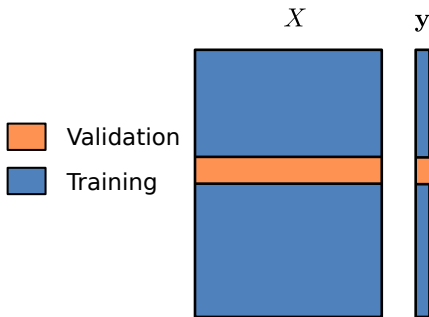


$k = 4$

1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :

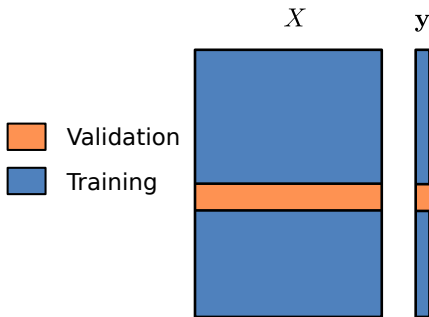


$k = 5$

1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :

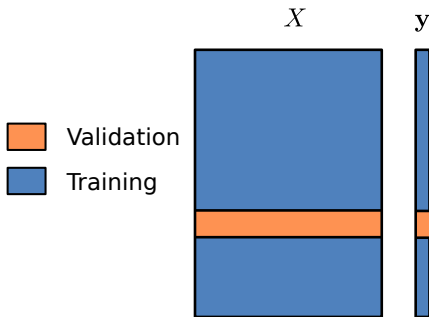


$k = 6$

1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

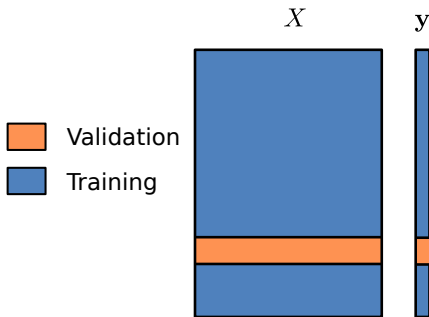
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

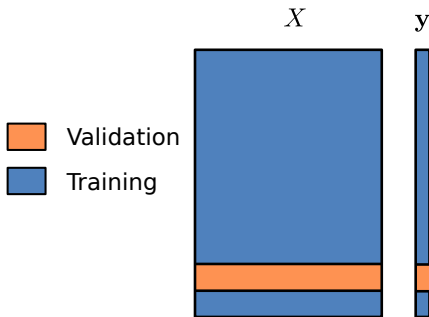
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

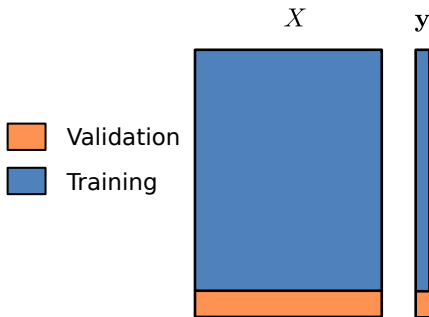
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



- $k = 9$
1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
 2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

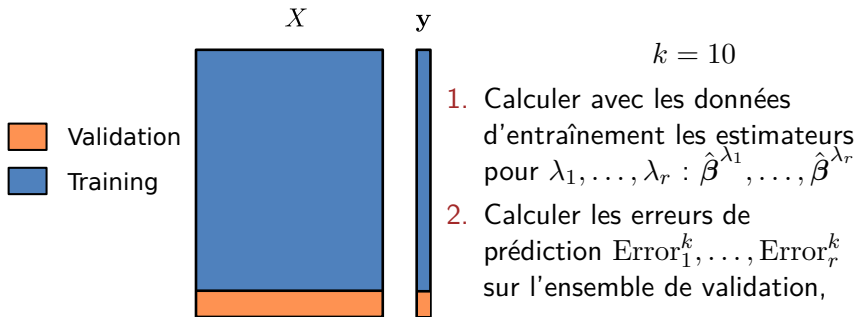
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



1. Calculer avec les données d'entraînement les estimateurs pour $\lambda_1, \dots, \lambda_r$: $\hat{\beta}^{\lambda_1}, \dots, \hat{\beta}^{\lambda_r}$
2. Calculer les erreurs de prédiction $\text{Error}_1^k, \dots, \text{Error}_r^k$ sur l'ensemble de validation,

Validation croisée K -fold ($K = 10$)

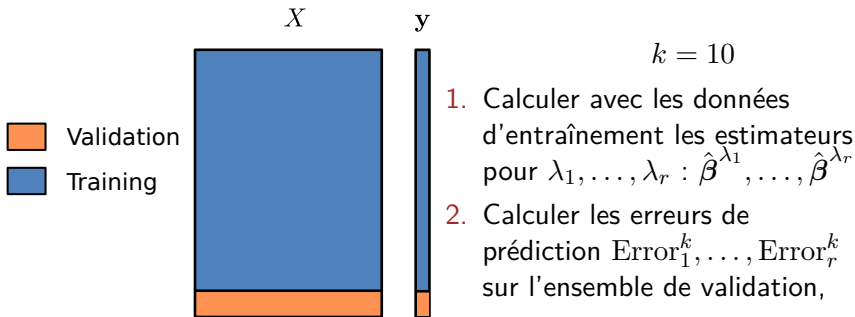
- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



Choix du paramètre : calculer $\widehat{\text{Error}}_1, \dots, \widehat{\text{Error}}_r$, moyennes des erreurs et choisir $\hat{i}^{\text{CV}} \in \llbracket 1, r \rrbracket$ atteignant la plus petite

Validation croisée K -fold ($K = 10$)


- ▶ Choisir une grille de taille r de λ à tester : $\lambda_1, \dots, \lambda_r$
- ▶ Diviser (X, y) selon les observations en K blocs (🇬🇧 : *fold*) :



Choix du paramètre : calculer $\widehat{\text{Error}}_1, \dots, \widehat{\text{Error}}_r$, moyennes des erreurs et choisir $\hat{i}^{\text{CV}} \in \llbracket 1, r \rrbracket$ atteignant la plus petite

Re-calibration : calculer $\hat{\beta}^{\lambda_{\hat{i}^{\text{CV}}}}$ sur toutes les observations (X, y)

CV en pratique

Cas extrême de validation croisée ( : *cross-validation*)

- ▶ $K = 1$: impossible, au moins $K = 2$
- ▶ $K = n$: stratégie “*leave-one-out*” (cf. **Jackknife**) : autant de blocs que de variables

Rem: $K = n$: calcul efficace pour Ridge mais assez instable

Conseils pratiques :

- ▶ “randomiser les observations” : observations dans un ordre aléatoire, évite des blocs de données trop similaires (chaque sous-bloc doit être représentatif de l'ensemble)
- ▶ choix habituels : $K = 5, 10$

Rem: en prédiction on peut aussi moyenner les meilleurs estimateurs obtenus plutôt que de re-calibrer sur toutes les données

Variantes de CV et sklearn

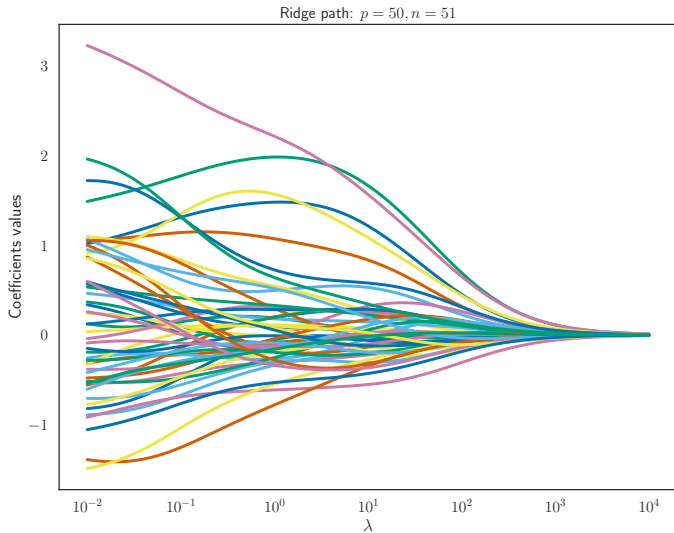
Alternatives classiques :

- ▶ partition aléatoire entre ensemble d'apprentissage et validation ($K = 2$ en gros, cf. `train_test_split`)
- ▶ variante pour séries temporelles : `TimeSeriesSplit`
- ▶ variante pour la classification et pour les cas avec classes déséquilibrées `StratifiedKFold`

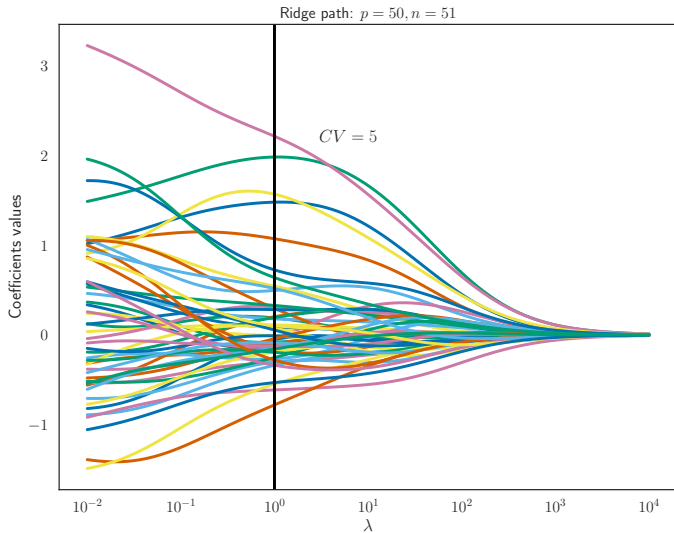
Plus de détails :

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

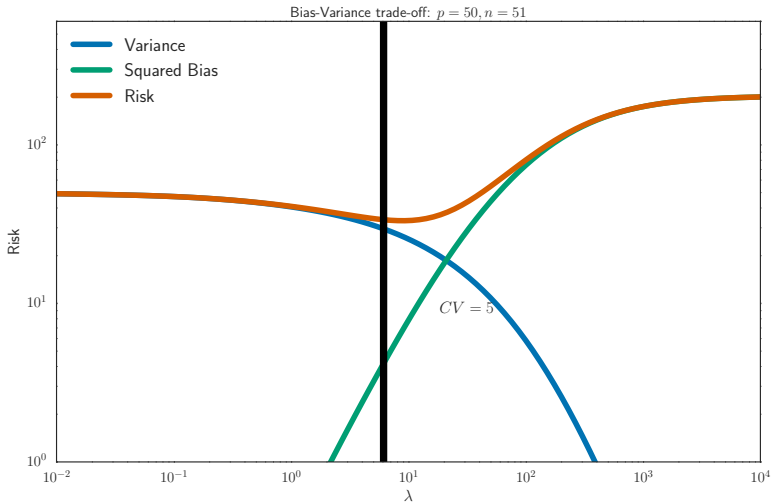
Choix de λ : exemple avec $CV = 5$ (I)



Choix de λ : exemple avec $CV = 5$ (I)



Choix de λ : exemple avec $CV = 5$ (II)



Plan

Définitions de l'estimateur Ridge

Choix du paramètre de régularisation

Algorithmes et aspects computationnels


Régularisation et colonne de zéro

ATTENTION : utilisation de CV avec des données catégorielles
En effet : si on enlève toute les occurrences d'une modalité de la partie apprentissage, on crée une colonne de zéro (les MCO se comportent alors bizarrement...)

Remèdes :

- ▶ “régularisation” : on peut obtenir une unique solution
- ▶ faire une séparation apprentissage/test plus poussée pour équilibrer les *folds*

Algorithmes pour la méthode *Ridge*

- ▶ 'svd' : méthode la plus stable, avantageuse pour calculer plusieurs λ car on ne "paye" la SVD qu'une fois
- ▶ 'cholesky' : décomposition matricielle proposant une formule fermée `scipy.linalg.solve`
- ▶ 'sparse_cg' : gradient conjugué utile dans les cas creux ( : *sparse*) et de grande dimension (baisser `tol/max_iter`)
- ▶ approche de type gradient stochastique si n est très grand

cf. le code des fonctions `Ridge`, `ridge_path`, `RidgeCV` dans le module `linear_model` de `sklearn`

Rem: on calcule rarement l'estimateur *Ridge* pour un λ , en général on en calcule plusieurs (10, 100, ...) et on cherche le meilleur

Rem: enjeu crucial de calculer des SVD de grandes tailles

References

- ▶ GOLUB, G. H. et C. F. VAN LOAN. *Matrix computations*. Fourth. Johns Hopkins University Press, Baltimore, MD, 2013, p. xiv+756.