

Régression régularisée

Ridge – Lasso – Elasticnet

Ricco Rakotomalala

Université Lumière Lyon 2



Plan

1. Régression linéaire multiple
2. Décomposition biais-variance de l'erreur de prédiction
3. Régression Ridge
4. Régression Lasso
5. Elasticnet
6. Descente de gradient
7. Régression logistique
8. Conclusion



Modélisation prédictive – Variable cible quantitative

RÉGRESSION LINÉAIRE MULTIPLE



Modélisation prédictive d'une variable cible quantitative

$$y = a_0 + a_1x_1 + \dots + a_px_p + \varepsilon$$

← Résume l'information de y que l'on n'arrive pas à capter avec les (x_1, \dots, x_p) variables prédictives.

Disposant d'un échantillon de taille n , nous cherchons les paramètres estimés $\hat{a} = (\hat{a}_0, \hat{a}_1, \dots, \hat{a}_p)$ qui minimise l'erreur quadratique moyenne (MSE : *mean squared error*).

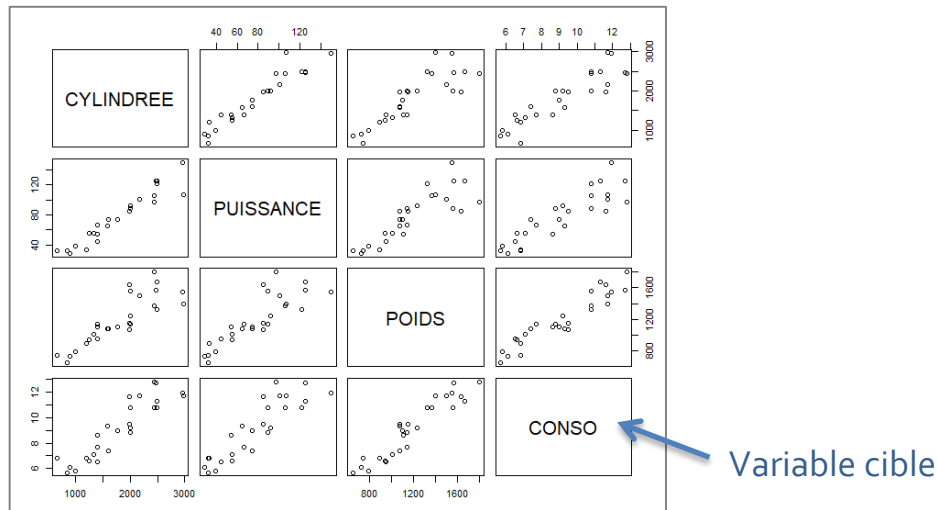
$$\min_{a_0, a_1, \dots, a_p} \frac{1}{n} \sum_{i=1}^n \left(y_i - \left(a_0 + \sum_{j=1}^p a_j x_{ij} \right) \right)^2$$

Estimateur des moindres carrés ordinaires (écriture matricielle)

$$\hat{a}_{MCO} = (X'X)^{-1}X'Y$$



Régression – Exemple : consommation des véhicules



La variable cible (CONSO) est liée positivement avec chaque variable explicative potentielle.

```
Call:
lm(formula = CONSO ~ CYLINDREE + PUISSANCE + POIDS, data = cars)

Residuals:
    Min       1Q   Median       3Q      Max
-1.0008 -0.4283  0.1229  0.3250  1.1995

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.4114272   0.5499731   2.566  0.0173 *
CYLINDREE    0.0012573   0.0007123   1.765  0.0908 .
PUISSANCE    0.0012095   0.0133867   0.090  0.9288
POIDS        0.0044906   0.0008069   5.565 1.16e-05 ***
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.645 on 23 degrees of freedom
Multiple R-squared:  0.9277,    Adjusted R-squared:  0.9183
F-statistic: 98.38 on 3 and 23 DF,  p-value: 2.893e-13
```

La régression ne dit pas ça du tout. Seule POIDS est significative à 5%.



Exemple (suite)

Comparer l'influence des explicatives. Travailler sur les variables centrées et réduites. On a les estimateurs $\hat{\beta}_j$

POIDS pèse 1.76 fois plus que CYLINDREE dans l'explication, et 35.5 fois plus que PUISSANCE ; CYLINDREE pèse 20.11 fois plus que PUISSANCE. On sait que ce n'est pas vrai au regard du graphique.

```
#Régression linéaire multiple
modele <- lm(CONSO ~ CYLINDREE + PUISSANCE + POIDS, data = cars)
#objet summary
sm <- summary(modele)
print(sm)
#données centrées et réduites
S <- as.data.frame(scale(cars))
#régression sur données centrées et réduites
#la constante est mécaniquement nulle
reg <- summary(lm(CONSO ~ .+0, data = S))
print(reg)
```

```
Call:
lm(formula = CONSO ~ . + 0, data = S)

Residuals:
    Min       1Q   Median       3Q      Max
-0.44357 -0.18981  0.05447  0.14404  0.53165

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
CYLINDREE    0.35353    0.19607    1.803  0.0839 .
PUISSANCE    0.01758    0.19044    0.092  0.9272
POIDS        0.62537    0.11001    5.685 7.43e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2799 on 24 degrees of freedom
Multiple R-squared:  0.9277,    Adjusted R-squared:  0.9187
F-statistic: 102.7 on 3 and 24 DF,  p-value: 7.935e-14
```



Variance de l'estimateur des MCO

$$V(\hat{\beta}_j) = \frac{\hat{\sigma}_\varepsilon^2}{n} \times v_j$$

Facteur d'inflation
de la variance (VIF)

$$v_j = \frac{1}{1 - R_j^2}$$

R_j^2 est le coefficient de détermination de la régression de x_j sur les $(p-1)$ autres variables.

Problème : $R_j^2 \approx 1 \Rightarrow v_j \approx +\infty$ (COLINEARITE)

Variance estimée de
l'erreur

$$\hat{\sigma}_\varepsilon^2 = \frac{\sum_{i=1}^n \hat{\varepsilon}_i^2}{n - p}$$

SCR (somme des carrés des résidus) : indicateur de qualité de la régression, divisé par les degrés de liberté.

Problème : $p \approx n \Rightarrow \hat{\sigma}_\varepsilon^2 \approx +\infty$; $(p > n) \Rightarrow (X'X)$ n'est pas inversible (DIMENSIONNALITE)

Ces problèmes entraînent une variance élevée de l'estimation c.-à-d. les coefficients estimés sont très erratiques, exagérément dépendants de l'échantillon d'apprentissage.

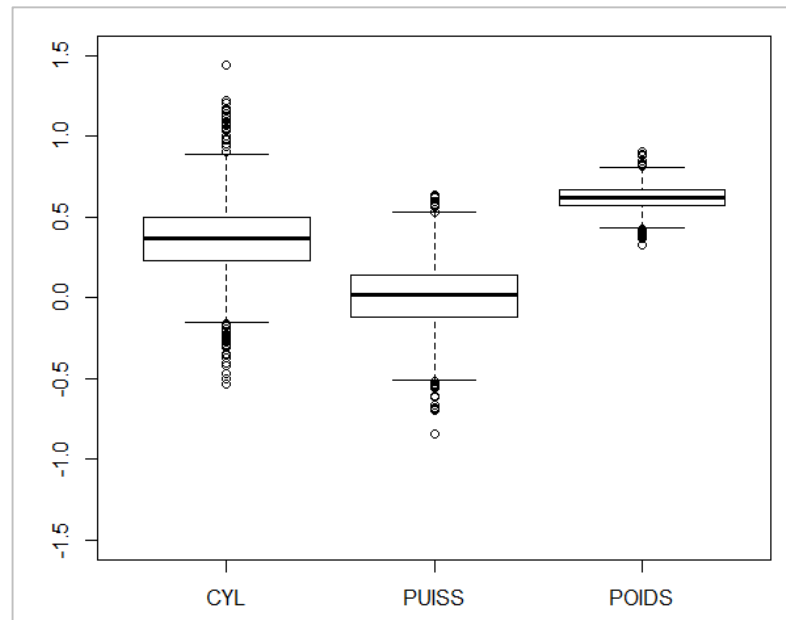


Exemple (suite)

1000 essais bootstrap pour évaluer la dispersion des coefficients estimés.

```
#échantillon bootstrap
bootreg <- fonction(S){
  #index des individus
  idx <- sample(1:nrow(S),nrow(S),replace=TRUE)
  #régression sans constante sur l'éch. bootstrap
  reg <- lm(CONSO ~ . + 0, data = S[idx,])
  #coefficients
  return(reg$coefficients)
}
#répéter 1000 fois
mreg <- replicate(1000,bootreg(S))
#boxplot
boxplot(mreg[1,],mreg[2,],mreg[3,],names=c("CYL", "PUISS", "POIDS"))
```

Les effectifs (n) sont faibles dans l'absolu, les variables explicatives sont colinéaires
→ les coefficients estimés sont particulièrement instables.



Décomposition biais-variance de l'erreur de prédiction

DÉCOMPOSITION BIAIS-VARIANCE



Evaluer la performance prédictive d'un modèle

Pour un individu supplémentaire x^* , la prédiction de Y est \hat{y}^* , avec

$$\hat{y}^* = \hat{a}_0 + \hat{a}_1 x_1^* + \dots + \hat{a}_p x_p^*$$

On évalue la qualité de la prédiction à l'aide de l'espérance de l'écart au carré entre la prédiction \hat{y}^* et la vraie valeur y^* (expected generalization error, expected prediction error)

Erreur incompressible. Variance de la cible Y, on ne pourra jamais faire mieux.

$$E[(y^* - \hat{y}^*)^2] = \sigma^2 + \underbrace{(E[\hat{y}^*] - y^*)^2}_{\text{Biais}^2} + \underbrace{E[(\hat{y}^* - E[\hat{y}^*])^2]}_{\text{Variance}}$$

Biais². Ecart au carré entre l'espérance de la prédiction et la vraie valeur. Indique les insuffisances intrinsèques du modèle (variables explicatives manquantes, ou forme de la relation non captée, etc.).

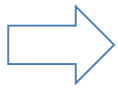
Variance. Dispersion de la prédiction autour de sa propre espérance. Témoigne de **l'instabilité** du modèle, sa dépendance aux fluctuations de l'échantillon d'apprentissage.

Instabilité qui devient patente avec les problèmes de colinéarité et sur-dimensionnalité vus précédemment.

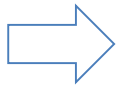


Principe de la régularisation

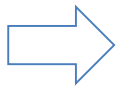
$$E[(y^* - \hat{y}^*)^2] = \sigma^2 + \text{Biais}^2 + \text{Variance}$$



Objectif : éviter le surapprentissage c.-à-d. **apprendre** de l'échantillon de données d'apprentissage, **mais pas trop**... (pas de sur dépendance)



Quelle principe ? Accepter une légère **augmentation du biais** pour obtenir une **réduction plus que proportionnelle de la variance**



Comment ? **Diriger (réguler)** un peu plus fermement la modélisation **en imposant des contraintes sur les paramètres estimés de la régression** (contraintes sur les valeurs que pourront prendre les $\hat{\alpha}_j$ dans leur ensemble pour éviter qu'elles soient totalement erratiques)

Au final, le modèle sera plus performant puisqu'on diminue l'erreur de prédiction espérée



Contrainte sur la norme L2 des coefficients

RÉGRESSION RIDGE



Régression Ridge

Ajouter une contrainte sur les coefficients lors de la modélisation pour maîtriser l'amplitude de leurs valeurs (« pour éviter qu'elles partent dans tous les sens »)

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2$$

Sous la contrainte $\sum_{j=1}^p \beta_j^2 \leq \tau$ $\left\{ \begin{array}{l} \text{Norme L2} \quad \|\beta\|_2^2 = \sum_{j=1}^p \beta_j^2 \\ \tau (\tau \geq 0) \text{ est un paramètre à fixer} \end{array} \right.$

Remarques :

- On parle de « **shrinkage** » (rétrécissement) : on rétrécit les plages de valeurs que peuvent prendre les paramètres estimés.
- Les variables x_j doivent être centrées et réduites (z_j) pour éviter que les variables à forte variance aient trop d'influence
- La variable cible y doit être centrée pour évacuer la constante de la régression (qui ne doit pas être pénalisée), la cible y peut être éventuellement réduite aussi : nous travaillerons alors sur les paramètres β_j
- $(\tau \rightarrow 0) \Rightarrow \beta_j \rightarrow 0$: les variances des coefficients estimés sont nulles
- $(\tau \rightarrow +\infty) \Rightarrow \beta_{\text{Ridge}} = \beta_{\text{MCO}}$



Régression Ridge – Fonction de pénalité

La régression ridge peut être écrite, de manière totalement équivalente :

Somme des carrés des résidus pénalisés :

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

λ ($\lambda \geq 0$) est un paramètre (coefficient de pénalité) qui permet de contrôler l'impact de la pénalité : à fixer

Fonction de pénalité

Remarques :

- Il y a une équivalence directe entre τ et λ
- $\tau \rightarrow 0 \Leftrightarrow \lambda \rightarrow +\infty : \beta_j \rightarrow 0$ (tous), variances des coefficients nulles
- $\tau \rightarrow +\infty \Leftrightarrow \lambda \rightarrow 0 : \beta_{\text{Ridge}} = \beta_{\text{MCO}}$


➔ L'estimateur Ridge s'écrit alors : $\hat{\beta}_{\text{Ridge}} = (X'X + \lambda I_p)^{-1} X'y$ I_p est la matrice identité

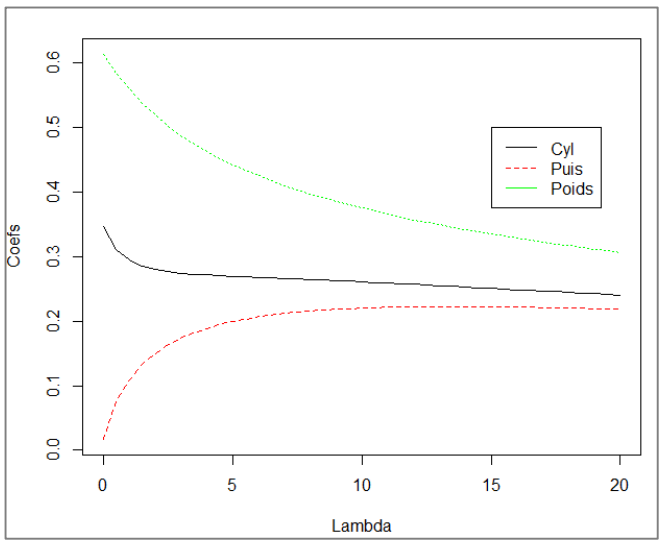
- On peut avoir une estimation même si $(X'X)$ n'est pas inversible
- On voit bien que $\lambda = 0$, alors on a l'estimateur des MCO



Détermination de la valeur de λ

Faire varier lambda et regarder l'évolution (de la stabilité) des coefficients (Ridge coefficients paths)

($5 < \lambda < 10$) ? Cette approche est très critiquée parce que purement empirique sans réellement justification. 



Ou par le calcul (cf. le package MASS de R) via différentes hypothèses simplificatrices.

Hoerl, Kennard, Baldwin (1975)

$$\lambda_{HKB} = \frac{(p - 2)\hat{\sigma}_\epsilon^2}{\sum_{j=1}^p \hat{\beta}_j^2}$$

Lawless, Wang (1976)

$$\lambda_{LW} = \frac{n(p - 2)\hat{\sigma}_\epsilon^2}{\sum_{i=1}^n \hat{y}_i^2}$$

Les paramètres ($\hat{\sigma}_\epsilon^2, \hat{\beta}_j, \hat{y}_i$) sont issus de la MCO. Dans notre exemple CARS, $\lambda_{HKB} = 0.1575$; $\lambda_{LW} = 0.0877$.



Ces approches ont pour principal défaut de ne pas tenir compte concrètement de la qualité de prédiction du modèle sur les données étudiées avec $\hat{\beta}_{Ridge}$



Utiliser la performance prédictive pour déterminer λ

S'appuyer sur un critère de performance pure

Principe :

- Fixer une plage de valeurs de λ
- Construire le modèle sur un échantillon d'apprentissage Ω_{App}
- L'évaluer sur un échantillon test Ω_{Test}
- Choisir λ qui minimise un critère d'erreur en test

Dérivé du PRESS (predicted residual error sum of squares) :

$$ERR = \frac{1}{n_{\text{Test}}} \sum_{i \in \Omega_{\text{Test}}} (y_i - \hat{y}_i)^2$$

Les individus Ω_{Test} ne doivent pas faire partie de ceux qui ont permis de construire le modèle. Sinon, c'est $\lambda = 0$ sera la solution systématique.

Validation croisée :

Si la base est de taille restreinte, la partition apprentissage-test n'est pas judicieuse. Mieux vaut passer par la *K-fold* validation croisée (cf. support de cours « [Validation Croisée, Bootstrap – Diapos](#) », Fév. 2015). L'objectif est toujours de minimiser le PRESS.



Leave-one-out – Cas particulier de la validation croisée

Dans le cas particulier où $K = n$ pour la validation croisée, nous avons le schéma leave-one-out (LOOCV)

$$ERR_{LOOCV} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_{i,-i})^2$$

Prédiction pour l'individu n^o_i à partir du modèle construit sur $(n-1)$ observations (excluant le n^o_i)

L'énorme intérêt est que nous pouvons l'approcher sans avoir à construire explicitement les $(n-1)$ modèles.

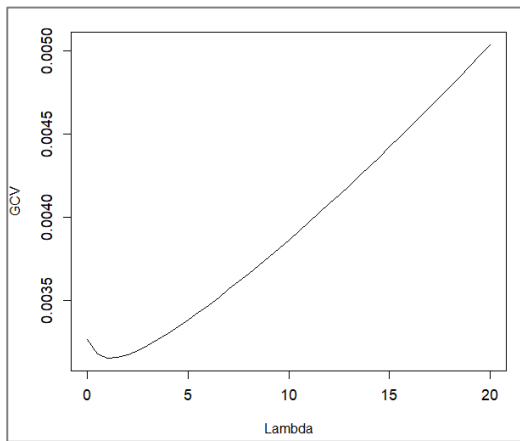
$$ERR_{GCV} = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - \frac{1}{n} tr(Z(Z'Z + \lambda I_p)^{-1} Z')} \right)^2$$

Il s'agit bien d'une prédiction en resubstitution cette fois-ci

Generalized cross validation (GCV).

On reconnaît la « hat-matrix ». Le tout n'est pas sans rappeler l'utilisation du levier dans le calcul des résidus standardisés / studentisés en économétrie.

Courbe GCV en fonction de λ pour les données « CARS »

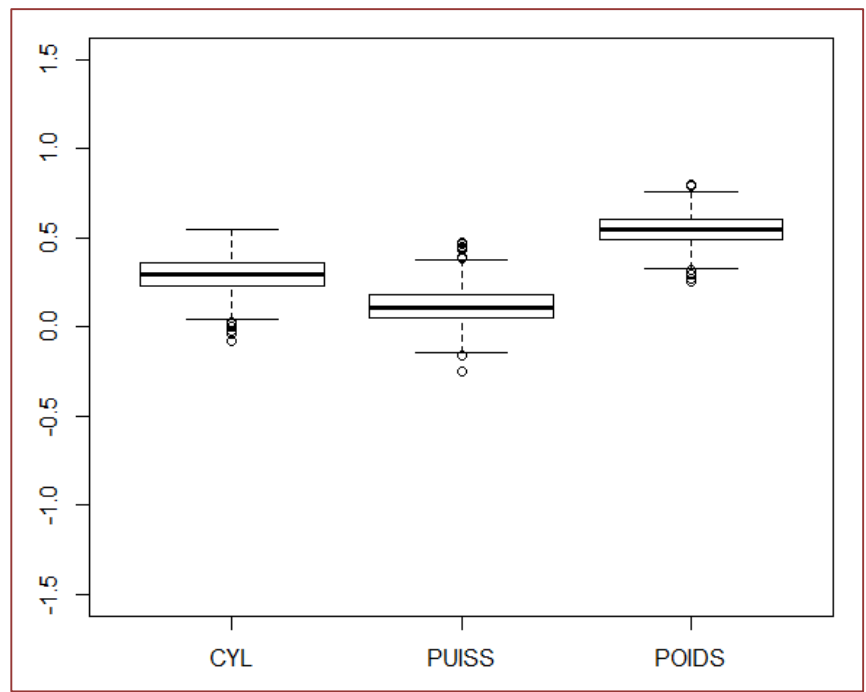
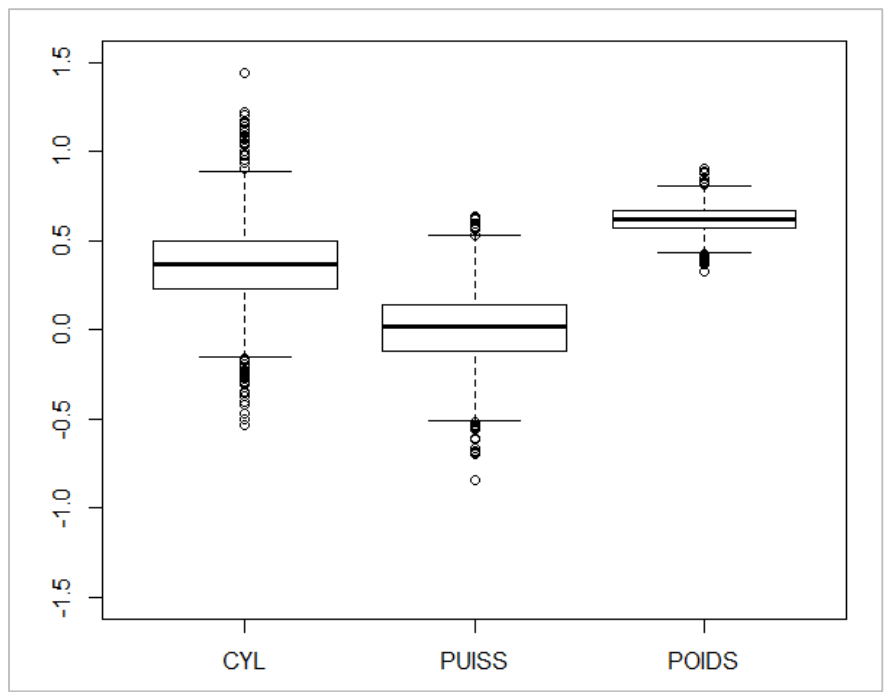


$(\lambda \approx 1)$ semble être la bonne solution pour nos données. Minimise l'ERR estimée par GCV.



Exemple « CARS » - Stabilisation des coefficients estimés

1000 simulations bootstrap



$$\hat{\beta}_{Ridge}(\lambda = 0) = \hat{\beta}_{MCO}$$

$$ERR_{GCV}(\lambda = 0) = 0.003263$$

$$\hat{\beta}_{Ridge}(\lambda = 1)$$

$$ERR_{GCV}(\lambda = 1) = 0.003153$$

C'est un petit exemple (n = 27, p = 3), le gap de performances n'est pas très flamboyant. En revanche, la réduction de la variance des coefficients estimés saute aux yeux.



Astuce de calcul pour la régression ridge

Pour choisir la valeur « optimale » de λ , il faudrait en essayer toute une série en évaluer les performances en validation croisée. *Calculs a priori très lourds.*

$$\hat{\beta}_{Ridge}(\lambda) = (X'X + \lambda I_p)^{-1} X'y$$

Passer par la décomposition en valeurs singulières (factorisation) de Z .

$$Z = UDV'$$

U ($n \times p$) et V ($p \times p$) sont orthogonaux. D ($p \times p$), la matrice des valeurs singulières, est diagonale (d_1, d_2, \dots, d_p).

Estimateur Ridge

$$\hat{\beta}_{Ridge}(\lambda) = V \Delta_\lambda U'y$$

$$\text{Où } \Delta_\lambda = \text{diag}\left(\dots, \frac{d_j^2}{d_j^2 + \lambda}, \dots\right)$$



U , V et D sont calculés une seule fois à partir de Z . L'obtention des différentes estimations $\hat{\beta}_{Ridge}$ pour différentes valeurs de λ revient à effectuer de simples produits matriciels. Facile et rapide.



Contrainte sur la norme L1 des coefficients

Least Absolute Shrinkage and Selection Operation

RÉGRESSION LASSO



Lasso – Contraintes sur la norme L1

Norme L1 : $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$

Très similaire à Ridge

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2 \quad \text{s.c.} \quad \sum_{j=1}^p |\beta_j| \leq \tau$$

Avec la fonction de pénalité

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

λ ($\lambda \geq 0$) est un paramètre (coefficient de pénalité) qui permet de contrôler l'impact de la pénalité : à fixer

- Idem Ridge :
- x_j doivent être centrées et réduites, y au moins centré (mais peut être réduit aussi)
 - Relation inverse entre τ et λ ; ($\lambda = 0 \Leftrightarrow \tau = +\infty$) \Rightarrow MCO ; plus λ élevé, plus la régularisation est forte (τ faible, les coefficients sont rétrécis)

Quel intérêt par rapport à Ridge ? LASSO peut faire office de dispositif de sélection de variables en annulant certains coefficients β_j : les variables associées à ($\beta_j = 0$) sont de facto exclues du modèle prédictif.



Algorithme d'apprentissage – LARS (least-angle regression)

Il n'y a pas de calcul direct pour LASSO. Passer par la régression LARS. C'est une démarche itérative où l'on commence avec tous les ($\beta_j = 0$). On fait évoluer les coefficients sélectivement. On a ainsi des scénarios de solutions (LASSO path) à mesure que $\|\beta\|_1$ augmente.

Algorithme LARS

« Elements of Statistical Learning »,
Hastie, Tibshirani, Friedman,
Corrected 12th, Jan. 2017 ; page 74.

Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \dots, \beta_p = 0$.
2. Find the predictor \mathbf{x}_j most correlated with \mathbf{r} .
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor \mathbf{x}_k has as much correlation with the current residual as does \mathbf{x}_j .
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor \mathbf{x}_l has as much correlation with the current residual.
5. Continue in this way until all p predictors have been entered. After $\min(N - 1, p)$ steps, we arrive at the full least-squares solution.

Légère modification
permettant d'obtenir LASSO

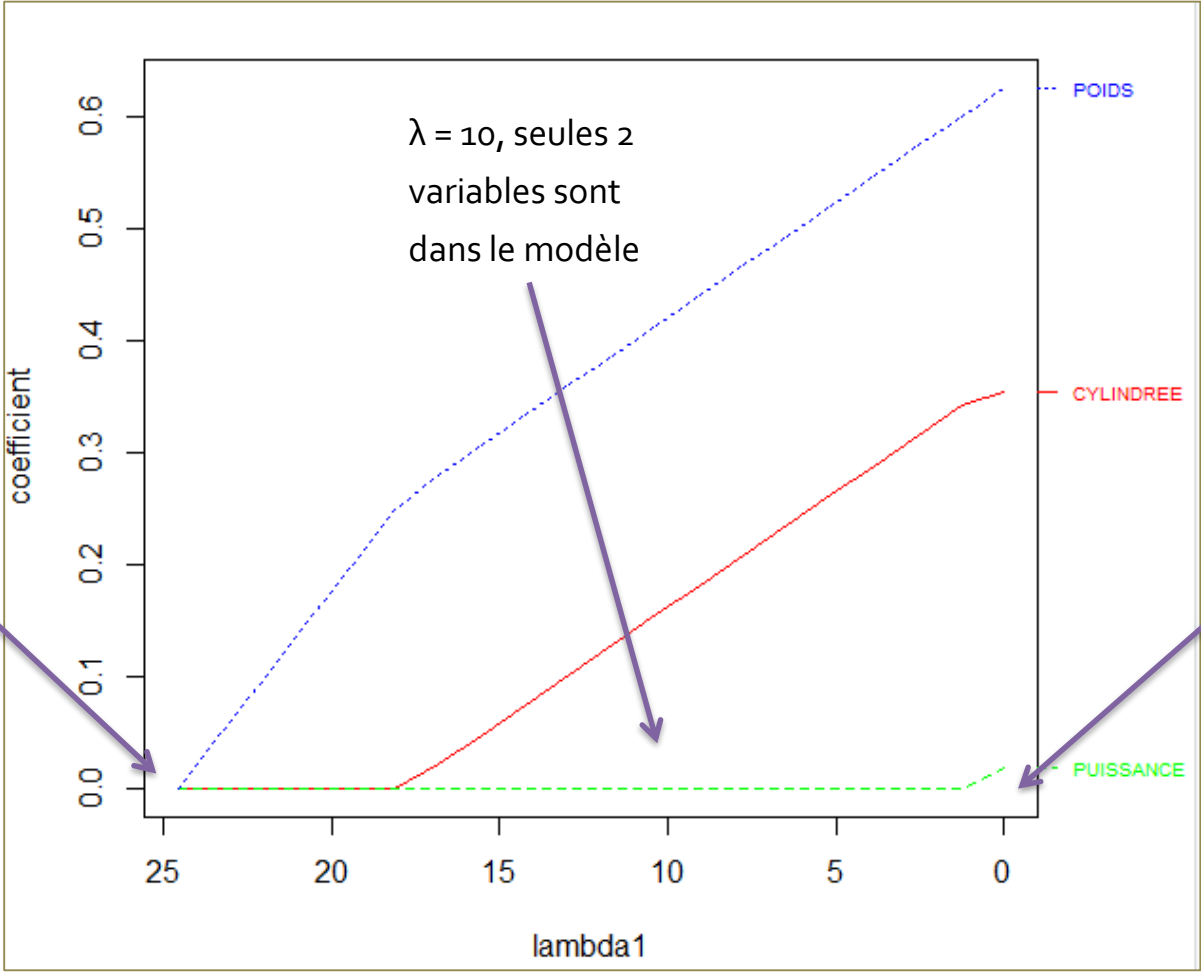
Algorithm 3.2a *Least Angle Regression: Lasso Modification.*

- 4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.



LASSO Path – Exemple “CARS”

$\lambda = 25$, tous les coefficients sont nuls

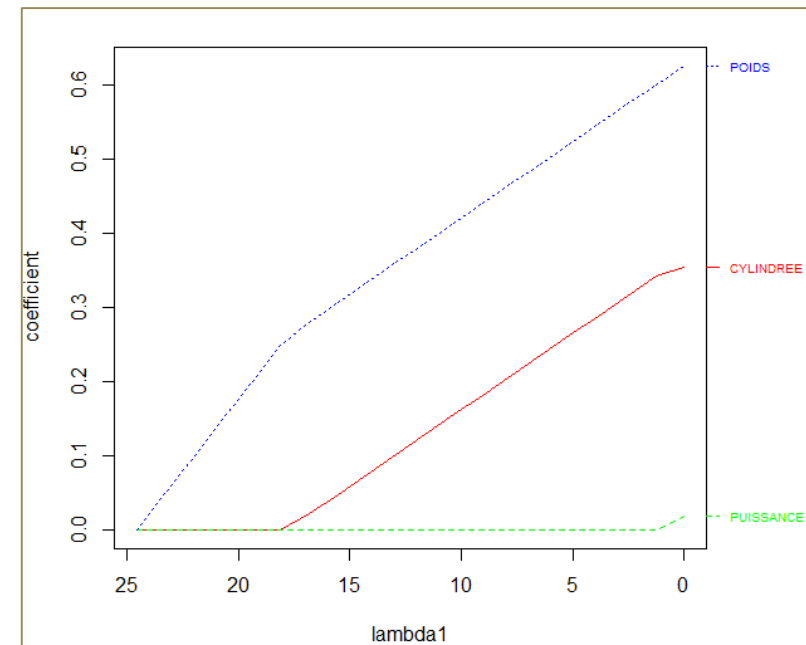


$\lambda = 0$, on a les coefficients de la MCO

Remarque : A la place de λ , on peut avoir en abscisse $||\beta||_1$ qui commence à 0 et sera croissante au fur et à mesure des itérations de LARS.



Lasso Path – Correspondance λ et $\|\beta\|_1$



	bnorm	lambda	cylinders	puissance	poids
[1,]	0.00000000	24.563247	0.00000000	0.000000000	0.00000000
[2,]	0.04972317	23.270445	0.00000000	0.000000000	0.04972317
[3,]	0.09944635	21.977642	0.00000000	0.000000000	0.09944635
[4,]	0.14916952	20.684840	0.00000000	0.000000000	0.14916952
[5,]	0.19889269	19.392037	0.00000000	0.000000000	0.19889269
[6,]	0.24861586	18.099235	0.00000000	0.000000000	0.24861586
[7,]	0.30129935	16.806432	0.02130223	0.000000000	0.27999712
[8,]	0.35473546	15.513630	0.04802029	0.000000000	0.30671517
[9,]	0.40817156	14.220827	0.07473834	0.000000000	0.33343322
[10,]	0.46160767	12.928025	0.10145639	0.000000000	0.36015128
[11,]	0.51504378	11.635222	0.12817445	0.000000000	0.38686933
[12,]	0.56847989	10.342420	0.15489250	0.000000000	0.41358738
[13,]	0.62191599	9.049617	0.18161056	0.000000000	0.44030544
[14,]	0.67535210	7.756815	0.20832861	0.000000000	0.46702349
[15,]	0.72878821	6.464012	0.23504666	0.000000000	0.49374154
[16,]	0.78222431	5.171210	0.26176472	0.000000000	0.52045960
[17,]	0.83566042	3.878407	0.28848268	0.000000000	0.54717775
[18,]	0.88909653	2.585605	0.31520082	0.000000000	0.57389571
[19,]	0.94254606	1.292802	0.34151505	0.0004698009	0.60056120
[20,]	0.99647096	0.000000	0.35352920	0.0175761099	0.62536565

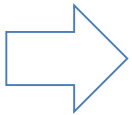
Algorithme (plus simple) – Forward Stagewise Algorithm

L'algorithme est plus simple et plus facile à implémenter que LARS, avec des résultats très similaires.

« [Regularization : Ridge Regression and the LASSO](#) », Tibshirani, 2006/2007.

Choisir une valeur ε (positive, faible)

1. Commencer avec les résidus initiaux $\mathbf{r} = \mathbf{y}$, et $\beta_1 = \beta_2 = \dots = \beta_p = 0$
2. Trouver le prédicteur \mathbf{Z}_j ($j=1, \dots, p$) le plus corrélé avec \mathbf{r}
3. Màj. $\beta_j \leftarrow \beta_j + \delta_j$, où $\delta_j = \varepsilon \cdot \text{signe} \langle \mathbf{r}, \mathbf{Z}_j \rangle$
4. Màj. $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{Z}_j$, et répéter jusqu'à arrêt



Ici aussi on a un continuum de résultats pour différentes valeurs de $\|\beta\|_1$ (et donc de λ) \rightarrow LASSO PATH.



Choix du paramètre λ

Dans l'optique prédictive, la minimisation de l'erreur de prédiction en balayant les valeurs de λ reste la méthode la plus efficace. On procède souvent par validation croisée.

```
#package R
library(glmnet)

#3-fold cross-validation
cvfit <- cv.glmnet(as.matrix(S[1:3]),S$CONSO,nfolds=3)

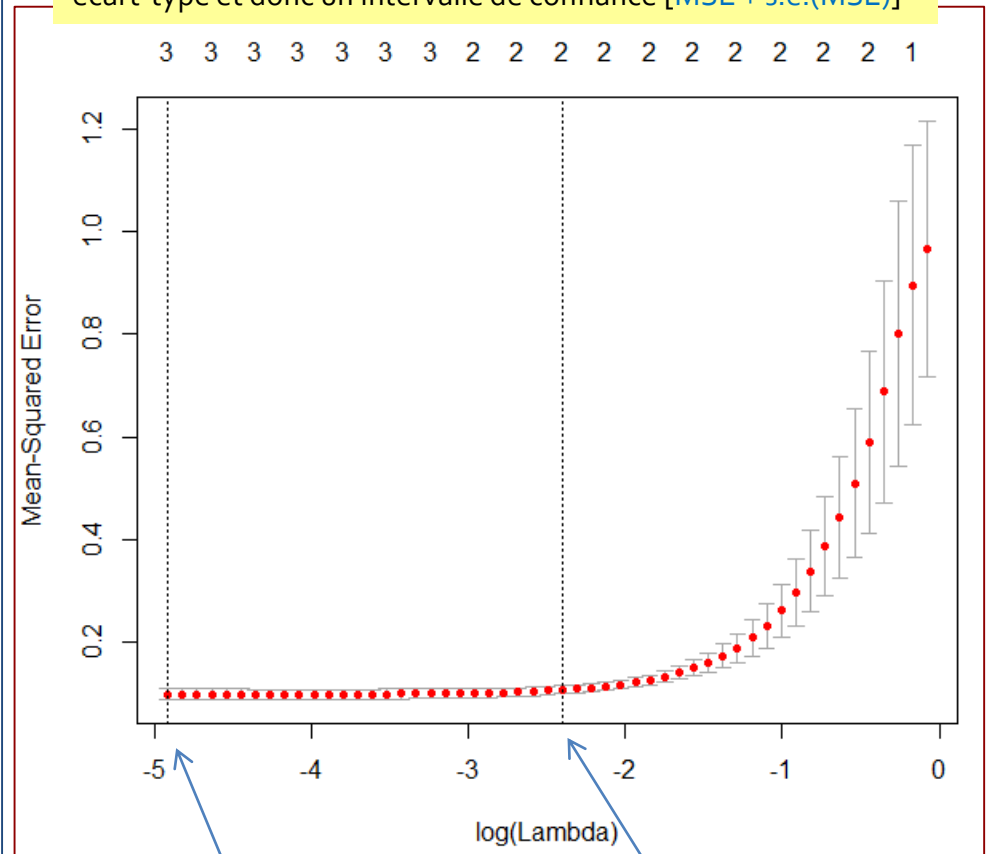
#courbe log(Lambda) vs. MSE
plot(cvfit)

#valeur min de MSE (en validation croisée)
print(min(cvfit$cvm)) #0.0987

#lambda corresp.
print(cvfit$lambda.min) #0.007347

#lambda le plus élevé dont le MSE est inf.
#à la borne haute de l'intervalle de min(MSE)
cvfit$lambda.1se #0.09057664 (log[0.09057] = -2.4)
```

Puisque c'est en validation croisée, on peut même avoir un écart-type et donc un intervalle de confiance $[MSE + s.e.(MSE)]$



λ pour min(MSE)

Le plus grand λ pour
 $MSE < \min(MSE) + s.e.[\min(SE)]$



Avantage

Capacité à sélectionner les variables en acceptant les coefficients nuls

Inconvénients

Dans les problèmes à très grandes dimensions ($p \gg n$), LASSO ne sélectionne que n variables prédictives au maximum, mécaniquement. C'est une vraie limitation de l'algorithme.

Parmi un groupe de variables corrélées, LASSO en choisit une, celle qui est la plus liée à la cible souvent, masquant l'influence des autres. Cet inconvénient est inhérent aux techniques intégrant un mécanisme de sélection de variables (*ex. arbres de décision, quoique...*).



Mix de Ridge et Lasso

ELASTICNET



Elasticnet – Combiner les avantages de Ridge et Lasso

Formulation sous la forme d'un problème d'optimisation

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2$$

λ_1 et λ_2 (≥ 0) sont des paramètres positifs à fixer. ($\lambda_1 = 0$ et $\lambda_2 > 0$) : Ridge ; ($\lambda_1 > 0$ et $\lambda_2 = 0$) : Lasso ; ($\lambda_1 = 0$ et $\lambda_2 = 0$) : MCO.

Formulation alternative de l'optimisation

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2 + \lambda \left[\alpha \sum_{j=1}^p |\beta_j| + (1 - \alpha) \sum_{j=1}^p \beta_j^2 \right]$$

$R(\beta)$ = Fonction de pénalité de la régression elasticnet. ($\alpha = 0$) : Ridge ; ($\alpha = 1$) Lasso.

Autre formulation : optimisation sous contrainte

$$\min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j z_{ij} \right)^2$$

$$\text{s.c. } \alpha \sum_{j=1}^p |\beta_j| + (1 - \alpha) \sum_{j=1}^p \beta_j^2 \leq \tau$$

Avec toujours la relation inverse entre λ et τ

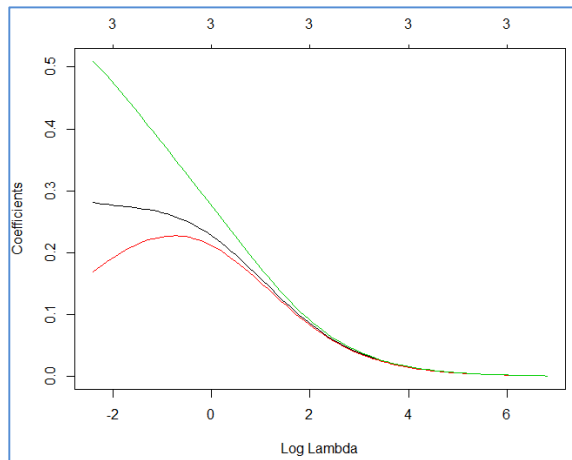


Quel intérêt ?

Double avantage :

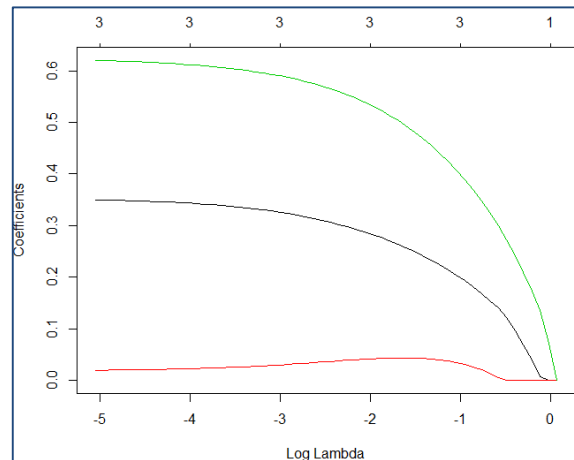
- Capacité de sélection de variables du LASSO conservée (coefficients nuls) : exclusion des variables non pertinentes
- Groupe de variables prédictives corrélées, partage des poids (comme Ridge) et non plus sélection arbitraire

Le « coefficient-path » selon λ dépend de la valeur de α cette fois-ci.



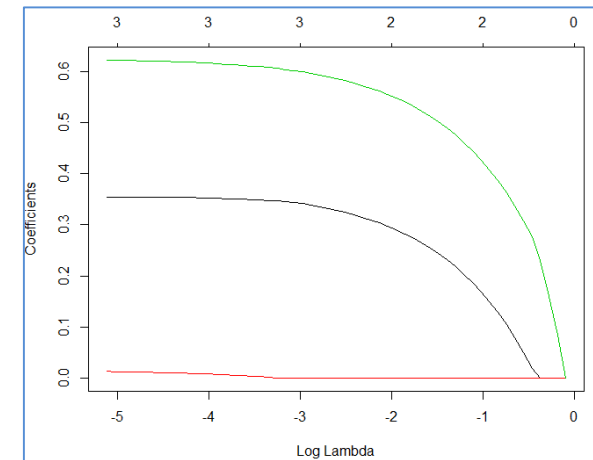
$\alpha = 0$, Ridge

Pas de sélection
de variables



$\alpha = 0.85$

Plus de nuances
dans la sélection



$\alpha = 1$, Lasso

Sélection drastique



Traitement des grandes bases de données à forte dimensionnalité

DESCENTE DE GRADIENT



Algorithme du gradient pour la régression régularisée

Descente de gradient et descente de gradient stochastique permettent de résoudre des problèmes d'optimisation dans les situations de très forte dimensionnalité ($p \gg n$) (cf. « [Descente de gradient – Diapos](#) », Avril 2018).

Scikit-learn documentation,
Section 1.5 "Stochastic
Gradient Descent".

$$E(\beta) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + \lambda \times R(\beta)$$

Fonction de perte confrontant la cible observée et la cible prédite (différente par ex. si la cible est catégorielle)

Terme de pénalité (de régularisation) qui pénalise la complexité du modèle.

$$\left\{ \begin{array}{l} \text{Norme L2 :} \quad R(\beta) = \frac{1}{2} \sum_{j=1}^p \beta_j^2 \\ \text{Norme L1 :} \quad R(\beta) = \sum_{j=1}^p |\beta_j| \\ \text{Elasticnet :} \quad R(\beta) = \frac{\rho}{2} \sum_{j=1}^p \beta_j^2 + (1 - \rho) \sum_{j=1}^p |\beta_j| \end{array} \right.$$

Notations scikit-learn...



Descente de gradient – Mis à jour des coefficients

Règle de mise à jour des coefficients à l'aide du gradient

$$\beta \leftarrow \beta - \eta \left(\lambda \frac{\partial R(\beta)}{\partial \beta} + \frac{\partial L(y_i, f(x_i))}{\partial \beta} \right)$$

Taux d'apprentissage pour moduler la convergence, peut être fixe ou décroissant au fil des itérations (pour plus de précisions)

Vecteur gradient ($p \times 1$), s'écrit différemment selon la fonction de perte $L()$ utilisée (confronte y_i et \hat{y}_i en régression linéaire, y_i et p_i en régression logistique)



$L()$ est choisie dérivable. Pas de souci. Mais $R()$ peut ne pas être dérivable. Heureusement, des solutions existent : « [Proximal gradient](#) », « [Subgradient](#) », « [Generalized gradient](#) », etc.



Attention, si présence de la constante β_0 (ex. en régression logistique), il ne rentre pas dans le terme de régularisation $R(\beta)$ et n'en est pas impactée



Algorithme générique (régression, classement). Capacité à traiter des très grandes bases, y compris à forte dimensionnalité ($p \gg n$).



Elasticnet via la descente de gradient

```
#importation des données
import pandas
D = pandas.read_excel("consommation_vehicules.xlsx",header=0,index_col=0)
print(D.columns)

#centrage-réduction de toutes les variables, y compris la cible
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
Z = scaler.fit_transform(D)

#appel de la classe SGDRegressor
from sklearn.linear_model import SGDRegressor
reg = SGDRegressor(loss="squared_loss",penalty="elasticnet",alpha=0.7,
                   l1_ratio=0.85,fit_intercept=False,max_iter=100)

#application sur les données
reg.fit(Z[:,0:3],Z[:,3])

#affichage
print(reg.coef_) #[0.09312288 0.0285881 0.21409579]
```

Correspond à notre λ

Correspond à notre α ($l_1_ratio = 1$: Lasso ; $l_1_ratio = 0$: Ridge)



Régression régularisée pour le classement

RÉGRESSION LOGISTIQUE



Fonction de perte pour la régression logistique

Fonction de perte spécifique à la variable cible binaire ($y \in \{0,1\}$) additionnée à la fonction de pénalité $R()$

Les conclusions rattachées à la régression linéaire sont conservées : sélection de variables, appréhension de la colinéarité, de la dimensionnalité.

$$E(\beta) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + \lambda \times R(\beta)$$

$$L = -[y_i \ln p_i + (1 - y_i) \ln(1 - p_i)]$$

Où p_i est la probabilité estimée d'appartenir à la classe d'intérêt 1.

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip})}}$$



Les variables prédictives doivent être centrées et réduites. C'est inutile pour la variable cible.



Du coup, il est nécessaire d'introduire la constante dans les coefficients estimés



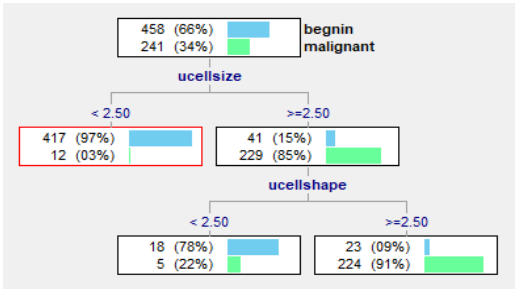
Mais la pénalité ne s'applique pas à la constante

$$R(\beta) = \alpha \sum_{j=1}^p |\beta_j| + (1 - \alpha) \sum_{j=1}^p \beta_j^2$$



Exemple : Breast Cancer Wisconsin

Détecter l'état d'une tumeur (maligne ou bénigne) à partir de la description des cellules. On sait que c'est un problème facile, avec $n = 699$, $p = 9$.



Taux de reconnaissance $\approx 95\%$

```
import pandas
D = pandas.read_excel("breastbin.xlsx",header=0)
#matrices et vecteurs
X = D.iloc[:, :9]
y = D.iloc[:, 9]
#centrage-réduction
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
Z = scaler.fit_transform(X)
#scikit-learn - SGDClassifier - instantiation
from sklearn.linear_model import SGDClassifier
cl = SGDClassifier(loss='log',penalty='elasticnet',max_iter=100)
#plages de paramètres à tester
settings = {'alpha':(0.01,0.1,0.5,1.0,2.0),'l1_ratio':(0.15,0.35,0.5,0.65,0.85,0.95)}
#outil de recherche
from sklearn.model_selection import GridSearchCV
#croisement des paramètres, 5-fold validation croisée, critère 'accuracy'
gs = GridSearchCV(cl,settings,scoring='accuracy',cv=5)
#lancer les calculs
gs.fit(Z,y)
#affichage des résultats
print(pandas.DataFrame.from_dict(gs.cv_results_)[['param_alpha','param_l1_ratio','mean_test_score']])
#coefficients estimés (sur var. centrées et réduites) du meilleur modèle
print(gs.best_estimator_.coef_)
#[[0.979247 0.42184354 0.6591121 0.43513842 0.245023 1.06804219 0.68834487 0.40343115 0.42881106]]
#constante du meilleur modèle
print(gs.best_estimator_.intercept_) #[-1.11509774]
```

	param_alpha	param_l1_ratio	mean_test_score
0	0.01	0.15	0.967096
1	0.01	0.35	0.967096
2	0.01	0.5	0.967096
3	0.01	0.65	0.967096
4	0.01	0.85	0.964235
5	0.01	0.95	0.964235
6	0.1	0.15	0.959943
7	0.1	0.35	0.954220
8	0.1	0.5	0.955651
9	0.1	0.65	0.955651
10	0.1	0.85	0.955651
11	0.1	0.95	0.949928
12	0.5	0.15	0.939914
13	0.5	0.35	0.895565
14	0.5	0.5	0.795422
15	0.5	0.65	0.655222
16	0.5	0.85	0.655222
17	0.5	0.95	0.655222
18	1	0.15	0.888412
19	1	0.35	0.655222
20	1	0.5	0.655222
21	1	0.65	0.655222
22	1	0.85	0.655222
23	1	0.95	0.655222
24	2	0.15	0.656652
25	2	0.35	0.655222
26	2	0.5	0.655222
27	2	0.65	0.655222
28	2	0.85	0.655222
29	2	0.95	0.655222

➔ Pas besoin de forte régularisation (0.01), et qu'importe finalement l'arbitrage Lasso / Ridge.



CONCLUSION



Conclusion



La dimensionalité et la colinéarité sont des problèmes récurrents en régression. La régularisation via des fonctions de pénalités sur les coefficients est une solution viable.



Notamment grâce à sa capacité à sélectionner les variables et à gérer les groupes de variables corrélées.



Applicable à la régression linéaire et logistique, mais aussi à d'autres méthodes dès lors que l'on a des combinaisons linéaires de variables avec des coefficients à estimer (ex. réseaux de neurones, perceptron, SVM linéaire, etc.).



Ne jamais oublier de ramener les explicatives sur la même échelle (centrer et réduire est le plus couramment utilisé) pour éviter les inégalités de traitements entre les coefficients.



Ridge, Lasso et Elasticnet sont proposés dans de nombreux packages pour R et Python, dont certains ont servi dans ce support (MASS, glmnet, elasticnet, lars, penalizedSVM, scikit-learn... mais aussi ceux spécialisés dans les réseaux de neurones telles que tensorflow / [keras](#)).



RÉFÉRENCES



- Hastie T., Tibshirani R., Friedman J., « The Elements of Statistical Learning », Springer, Corrected 12h print, January 2017.
- Tibshirani R., « [Regularization : Ridge Regression and the LASSO](#) », November 2006.
- PennState Eberly College of Science, « [Lesson 5: Regression Shrinkage Methods](#) », in « Applied Data Mining and Statistical Learning », STAT897D.
- Cornillon P.A., Matzner-Lober E., « Régression – Théorie et applications », Springer, 2007.
- Scikit-learn, « [Generalized Linear Models](#) », version 0.19.1.

